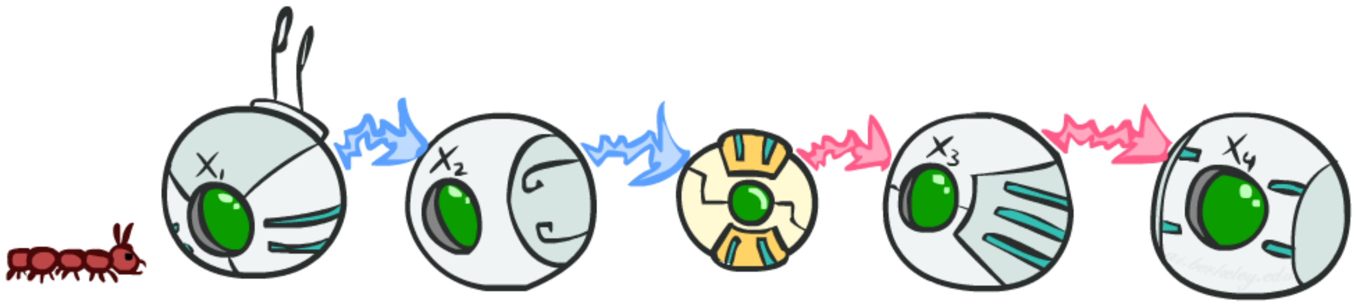
CSE 573: Artificial Intelligence

Hidden Markov Models



slides adapted from
Stuart Russel, Dan Klein, Pieter Abbeel from ai.berkeley.edu
And Hanna Hajishirzi, Jared Moore, Dan Weld

Uncertainty and Time

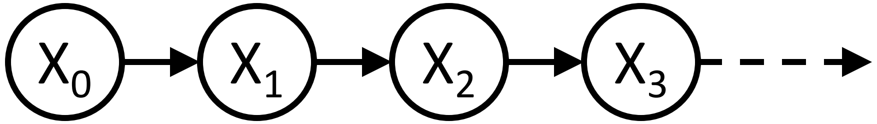
- Often, we want to reason about a ***sequence*** of observations
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring

- Generalize MDPs by adding sensing noise (and removing actions)

Video of Demo Pacman – Sonar

Markov Models (aka Markov chain/process)

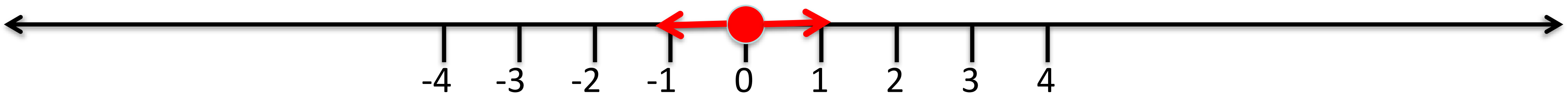
- Value of X at a given time is called the ***state*** (usually discrete, finite)



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

- The ***transition model*** $\boldsymbol{P}(X_t \mid X_{t-1})$ specifies how the state evolves over time
- ***Stationarity*** assumption: transition probabilities are the same at all times
- ***Markov*** assumption: "future is independent of the past given the present"
  - X_{t+1} is independent of $X_0,\ldots, X_{t-1}$ given X_t
  - This is a ***first-order*** Markov model (a *k*th-order model allows dependencies on *k* earlier steps)
- Joint distribution $\boldsymbol{P}(X_0,\ldots, X_T) = \boldsymbol{P}(X_0) \prod_t \boldsymbol{P}(X_t \mid X_{t-1})$

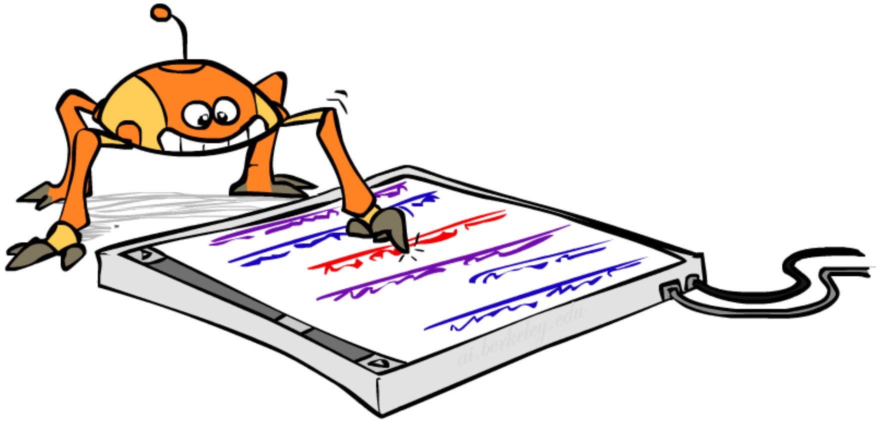
Example: Random walk in one dimension



- State: location on the unbounded integer line
- Initial probability: starts at 0
- Transition model: $P(X_t = k \mid X_{t-1} = k \pm 1) = 0.5$
- Applications: particle motion in crystals, stock prices, gambling, genetics, etc.
- Questions:
  - How far does it get as a function of t?
    - Expected distance is $O(\sqrt{t})$
  - Does it get back to 0 or can it go off for ever and not come back?
    - In 1D and 2D, returns w.p. 1; in 3D, returns w.p. 0.34053733

Example: Web browsing

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



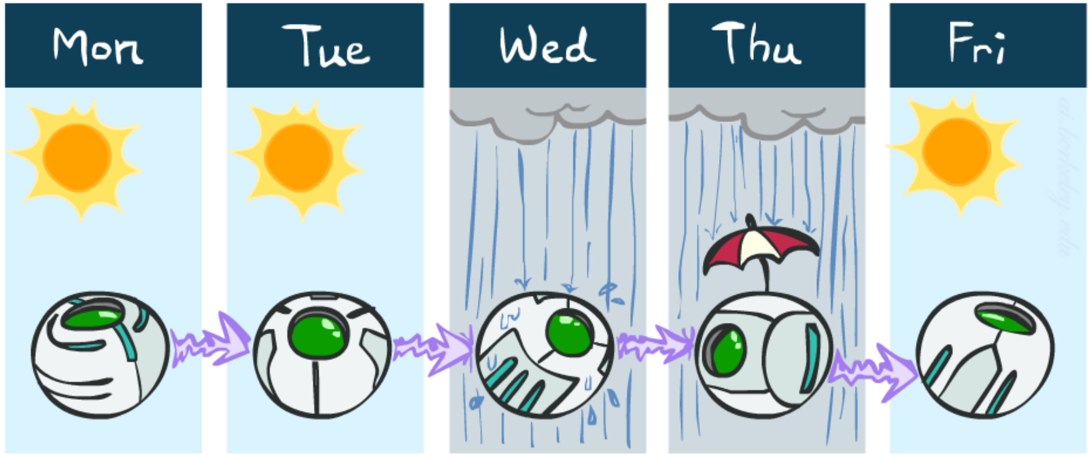
Example: Weather

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

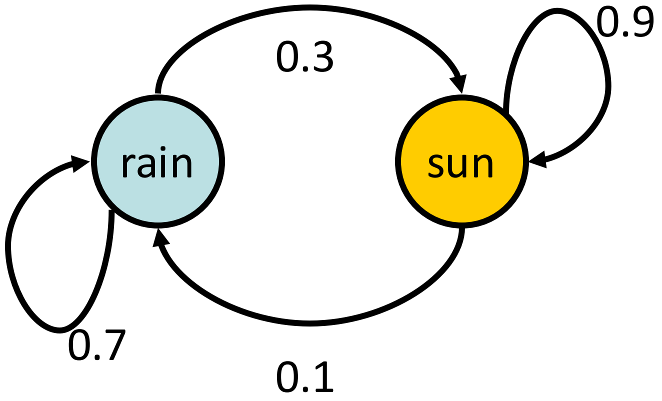
$P(X_0)$	
sun	rain
0.5	0.5

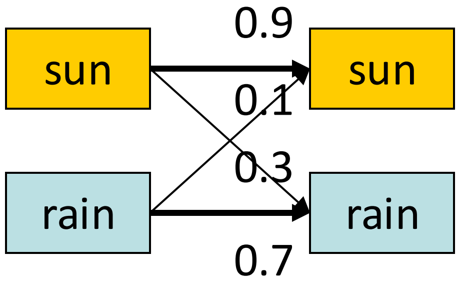
- Transition model $P(X_t \mid X_{t-1})$

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



Two new ways of representing the same CPT

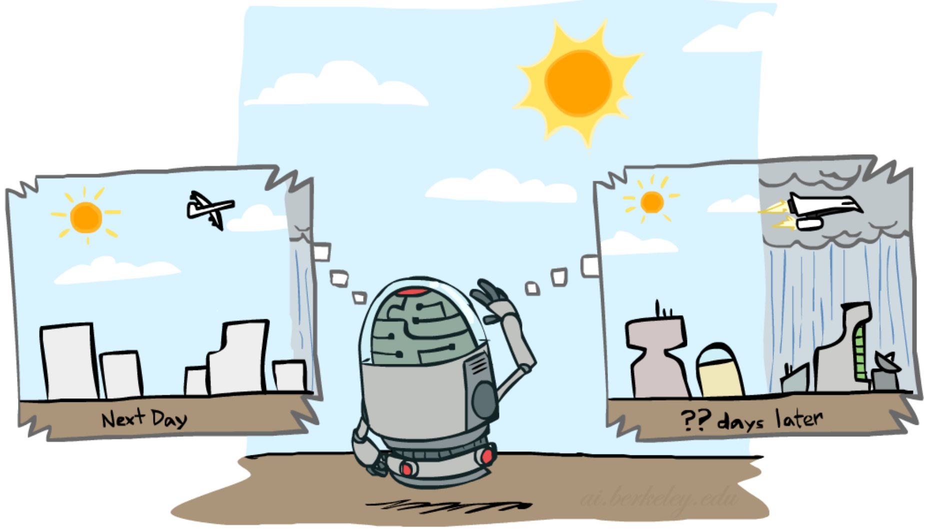




Weather prediction

- 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?
  - $\mathbf{P}(X_1) = \sum_{x_0} \mathbf{P}(X_1, X_0 = x_0)$

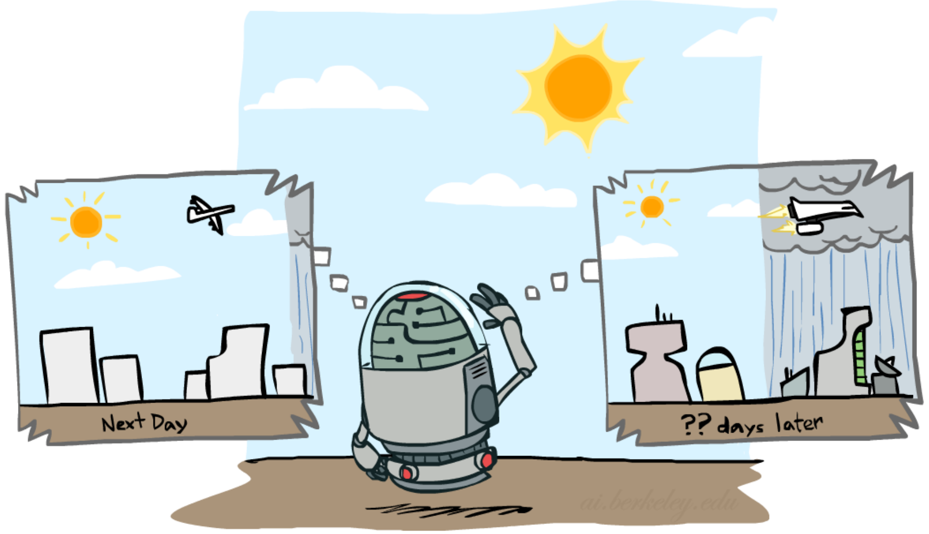
    $= \sum_{x_0} P(X_0 = x_0)\, \mathbf{P}(X_1 \mid X_0 = x_0)$

    $= 0.5\langle 0.9, 0.1\rangle + 0.5\langle 0.3, 0.7\rangle = \langle 0.6, 0.4\rangle$

Weather prediction, contd.

- Time 1: <0.6,0.4>

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



- What is the weather like at time 2?
  - $\mathbf{P}(X_2) = \sum_{x_1} \mathbf{P}(X_2, X_1 = x_1)$

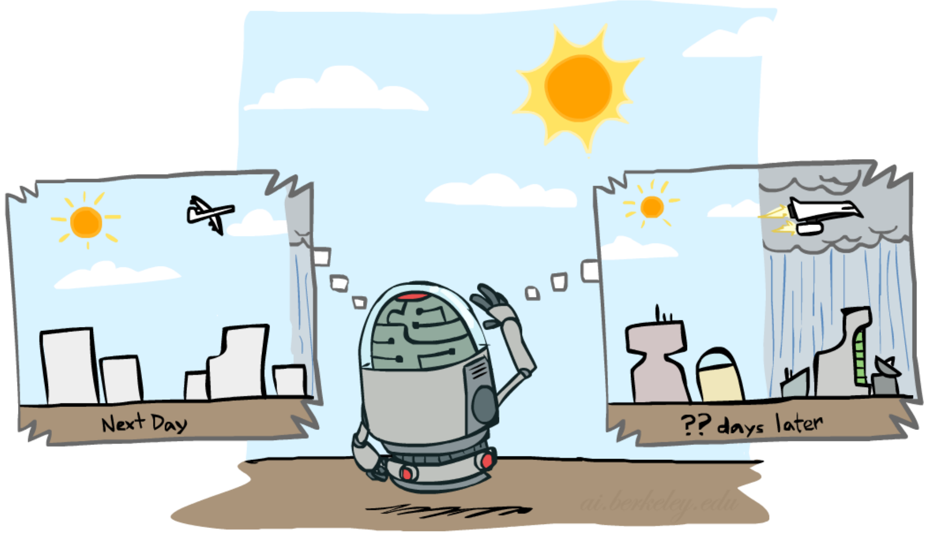
    $= \sum_{x_1} P(X_1 = x_1)\, \mathbf{P}(X_2 \mid X_1 = x_1)$

    $= 0.6\langle 0.9, 0.1\rangle + 0.4\langle 0.3, 0.7\rangle = \langle 0.66, 0.34\rangle$

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?
  - $\mathbf{P}(X_3) = \sum_{x_2} \mathbf{P}(X_3, X_2{=}x_2)$

    $= \sum_{x_2} P(X_2{=}x_2)\, \mathbf{P}(X_3 \mid X_2{=}x_2)$

    $= 0.66\langle 0.9, 0.1\rangle + 0.34\langle 0.3, 0.7\rangle = \langle 0.696, 0.304\rangle$

Forward algorithm (simple form)

- What is the state at time t?
  - $\boldsymbol{P}(X_t) = \sum_{x_{t-1}} \boldsymbol{P}(X_t, X_{t-1}=x_{t-1})$

    $= \sum_{x_{t-1}} P(X_{t-1}=x_{t-1})\, \boldsymbol{P}(X_t \mid X_{t-1}=x_{t-1})$
- Iterate this update starting at t=0

Probability from previous iteration

Transition model

And the same thing in linear algebra

- What is the weather like at time 2?

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

- In matrix-vector form:

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

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

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

Stationary Distributions

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

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

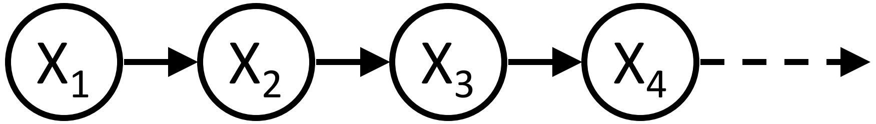
$$0.9p + 0.3(1-p) = p$$

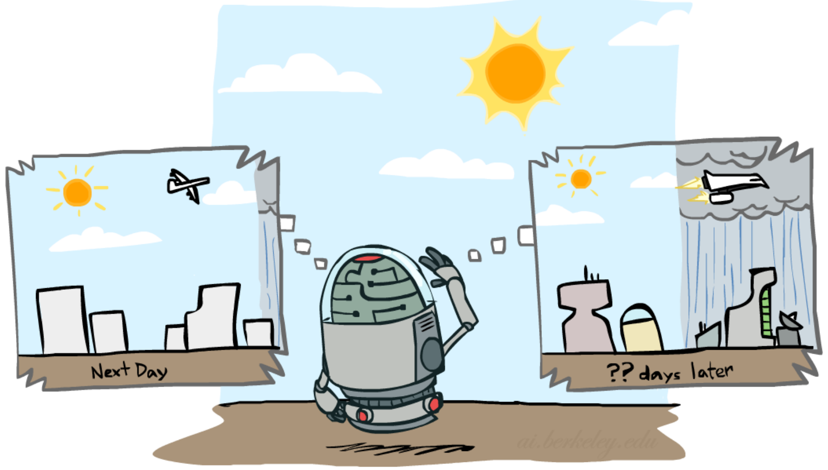
$$p = 0.75$$

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

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

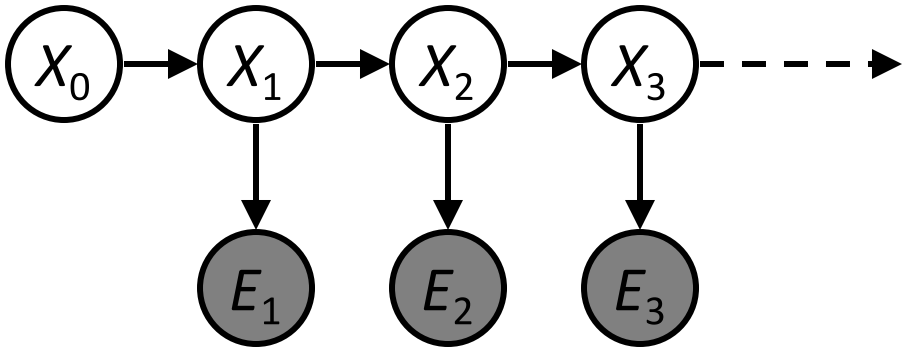
$$P_\infty(sun) = 3/4$$
$$P_\infty(rain) = 1/4$$

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

Hidden Markov Models

Hidden Markov Models

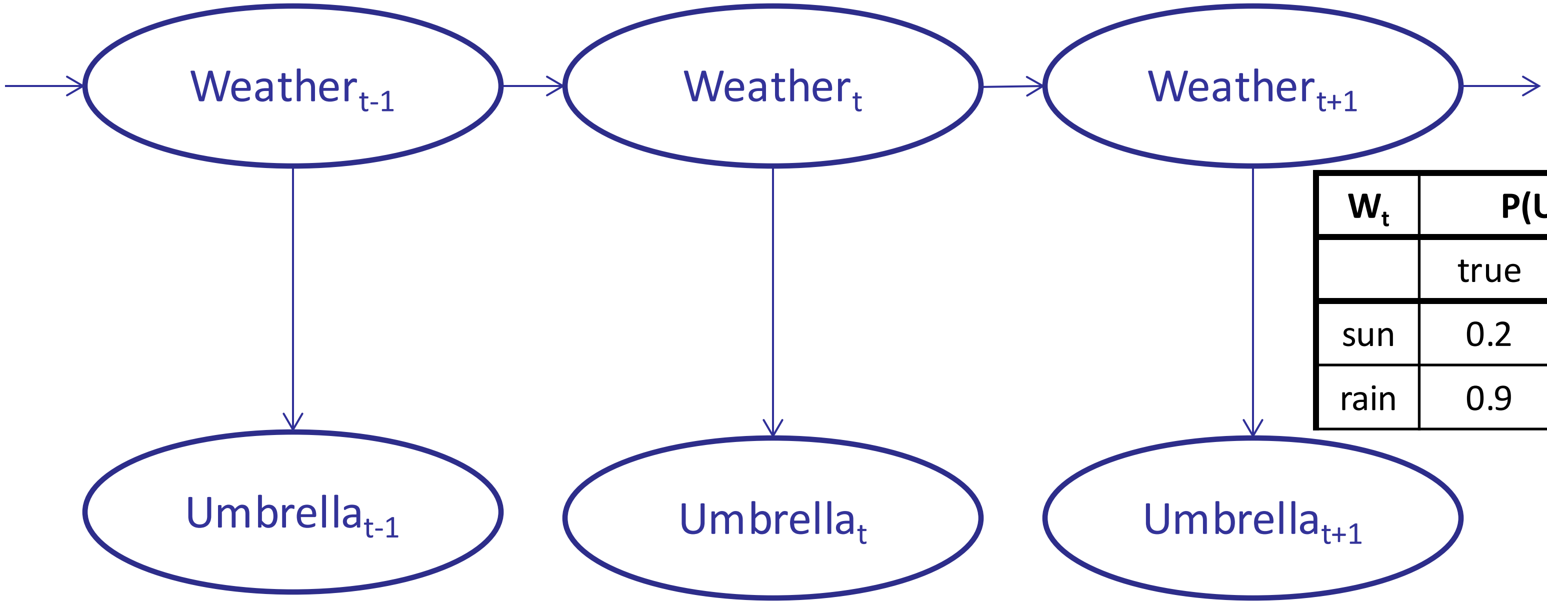
- Usually the true state is not observed directly
- Hidden Markov models (HMMs)
  - Underlying Markov chain over states X
  - You observe evidence E at each time step
  - X_t is a single discrete variable; E_t may be continuous and may consist of several variables



Example: Weather HMM

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

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



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


HMM as probability model

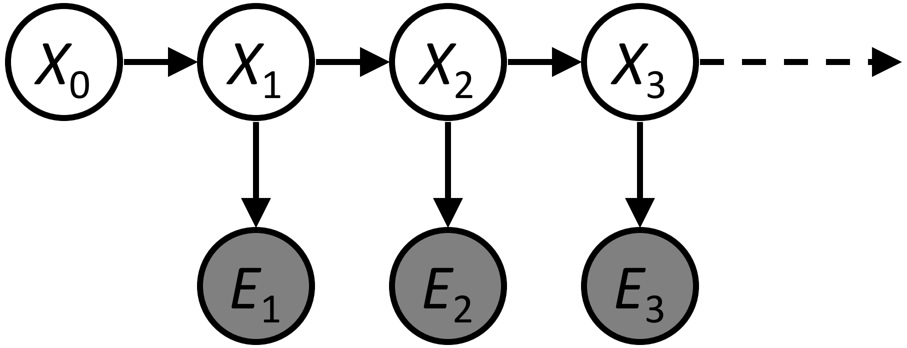
- Joint distribution for Markov model:

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

- Joint distribution for hidden Markov model:

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

- Future states are independent of the past given the present
- Current evidence is independent of everything else given the current state
- **Question: Are evidence variables independent of each other?**



Useful notation:

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

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

Filtering / Monitoring

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

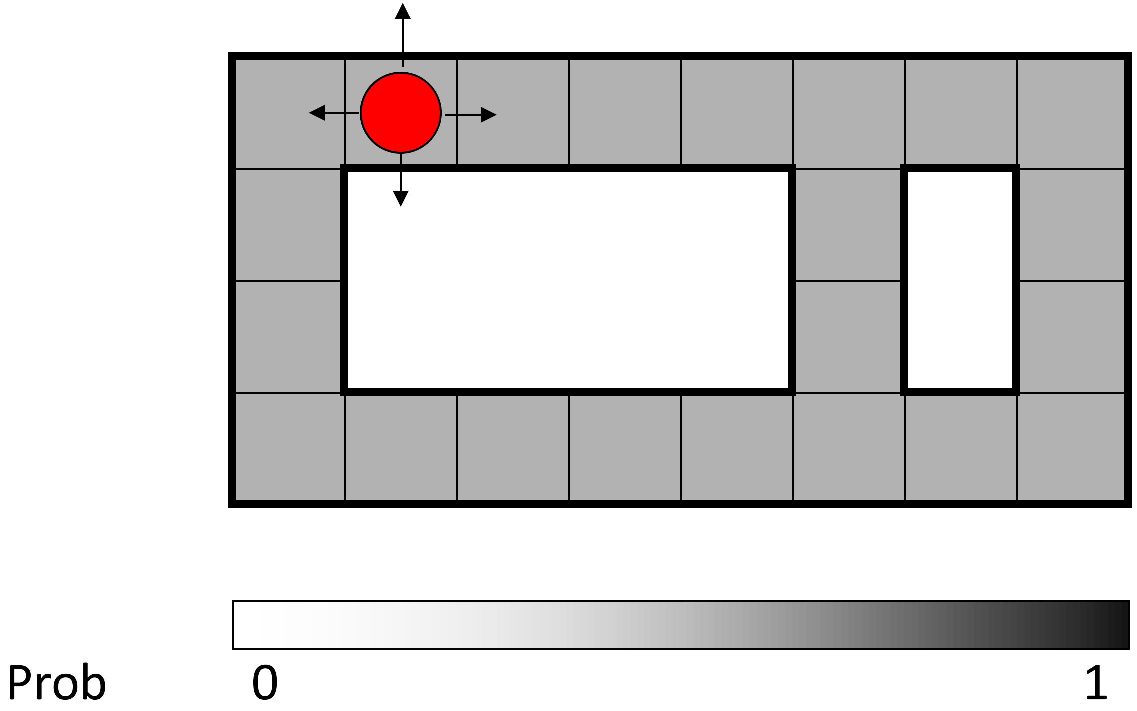
- We start with $\boldsymbol{f}_0$ in an initial setting, usually uniform

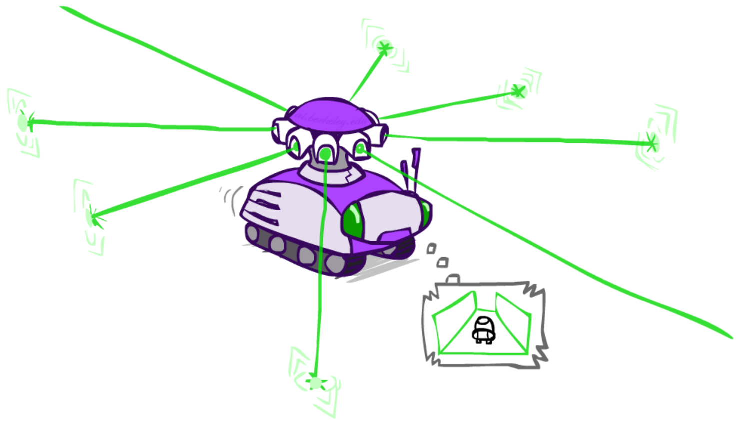
- Filtering is a fundamental task in engineering and science

- 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

Example: Robot Localization

Example from Michael Pfeiffer



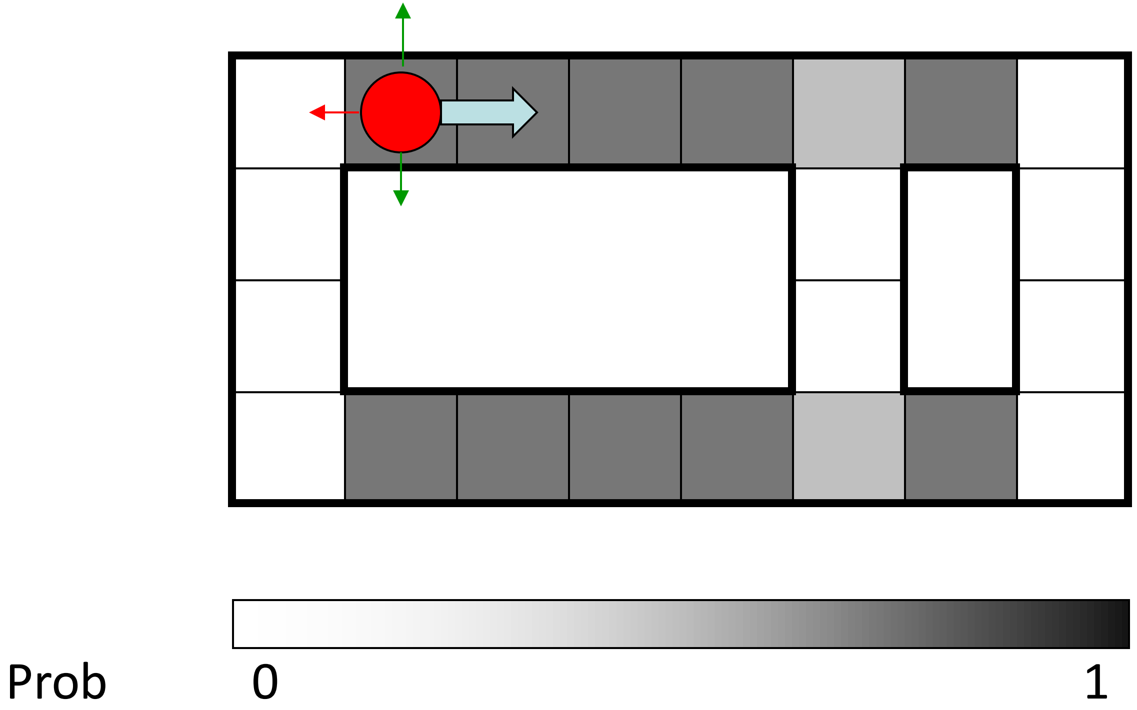


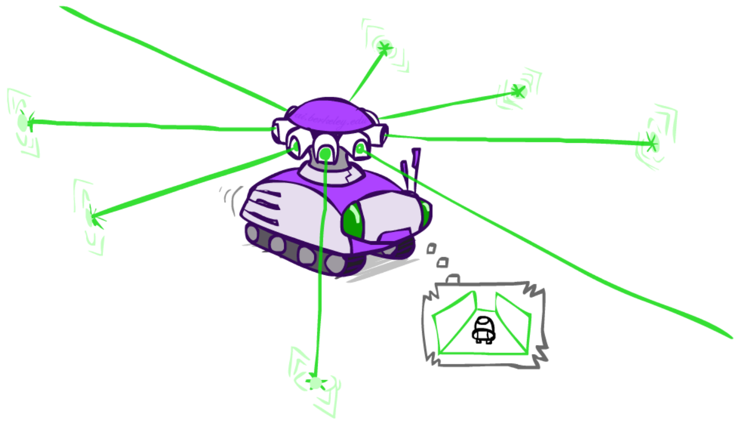
t=0

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

Transition model: action may fail with small prob.

Example: Robot Localization



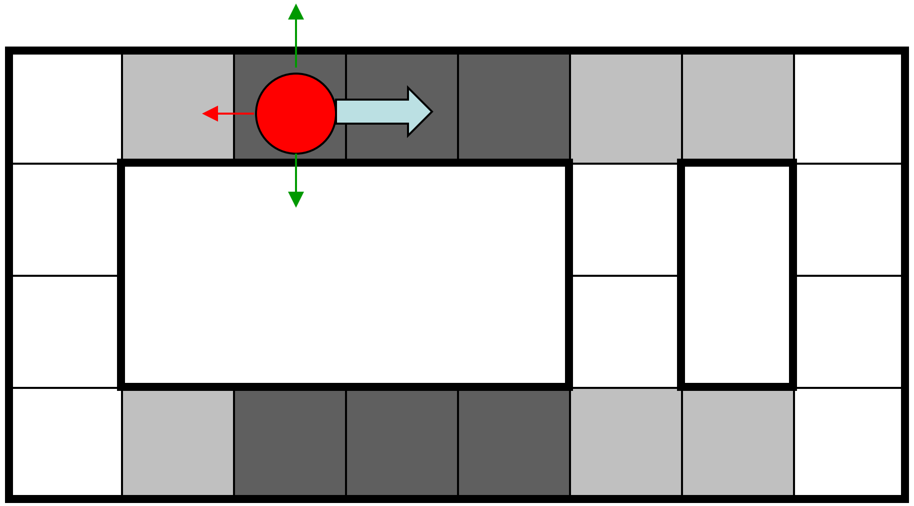


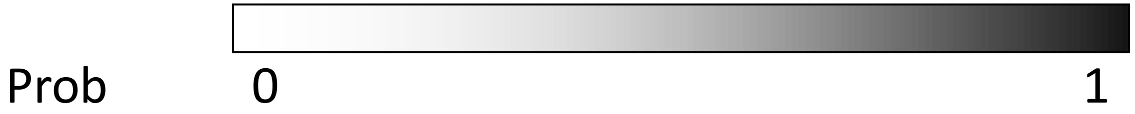
t=1

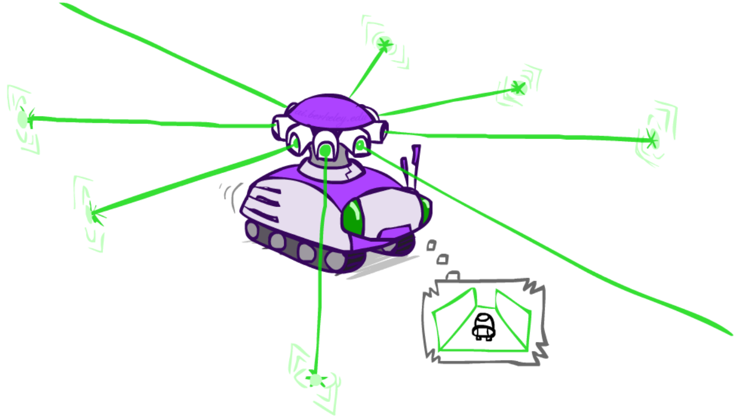
Lighter grey: was ***possible*** to get the reading,

but ***less likely*** (required 1 mistake)

Example: Robot Localization

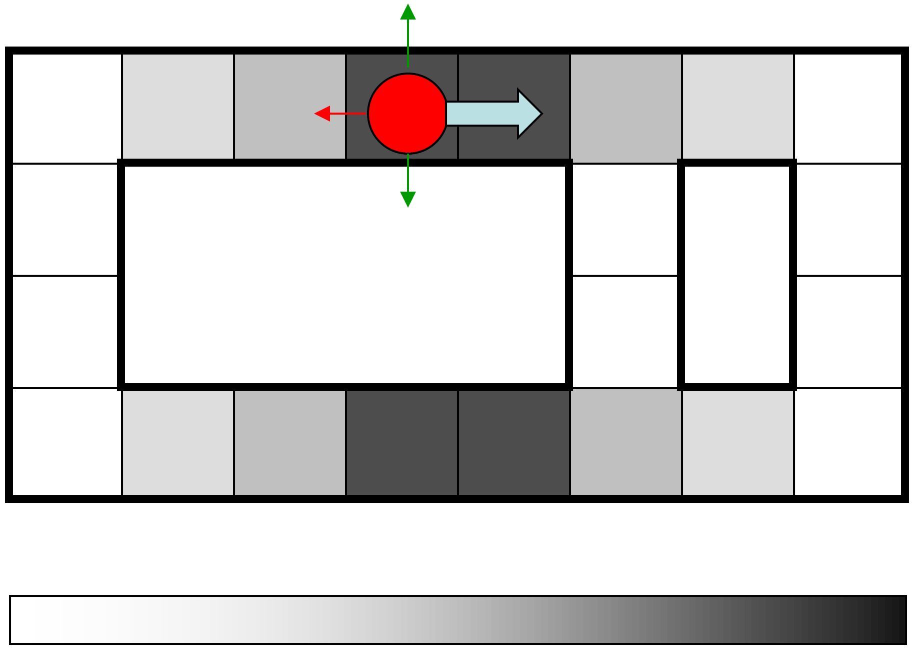


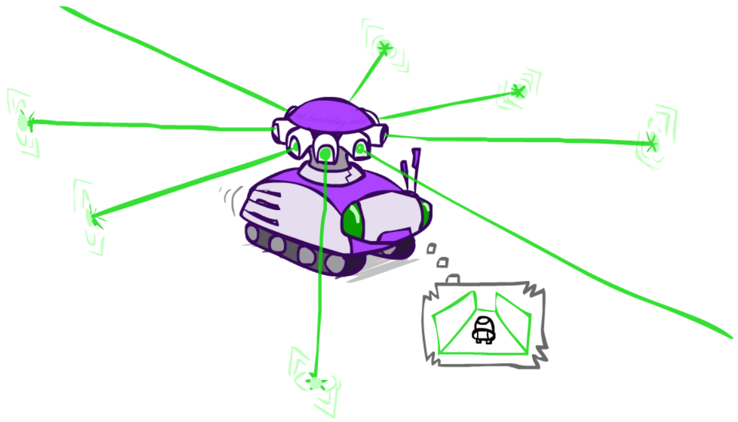




t=2

Example: Robot Localization

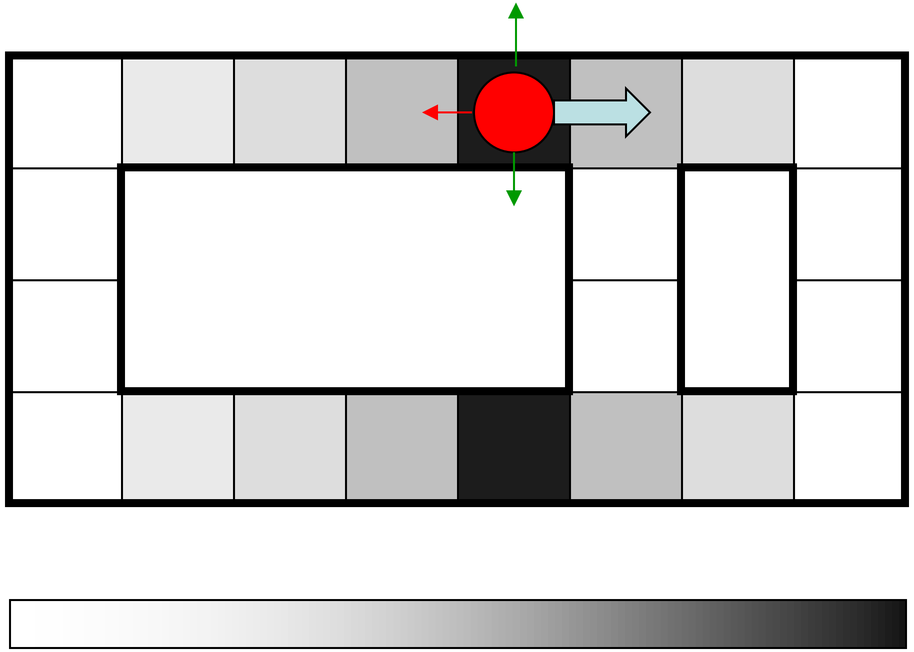




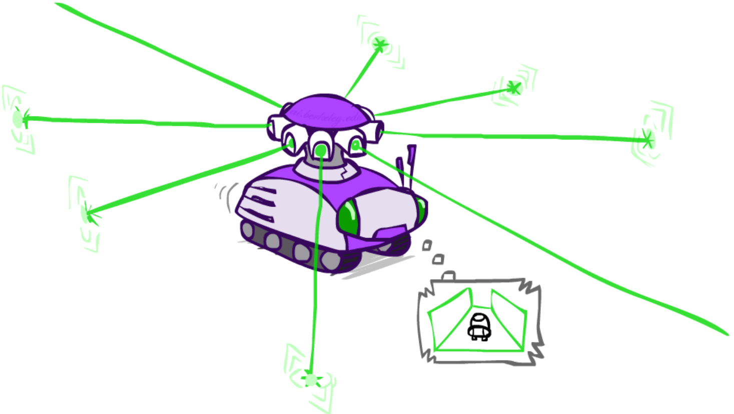
Prob 0 1

t=3

Example: Robot Localization

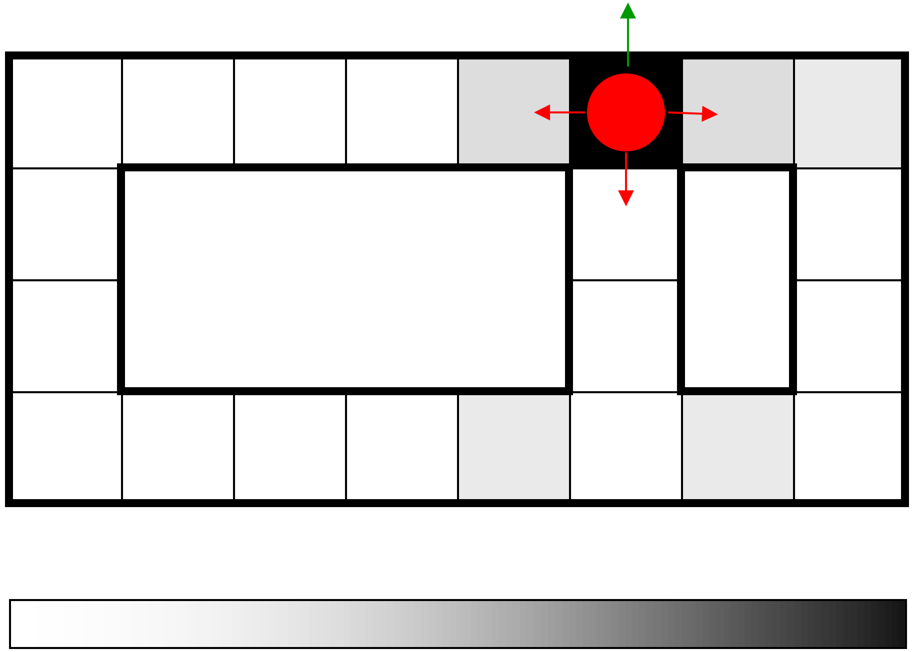


Prob 0 1

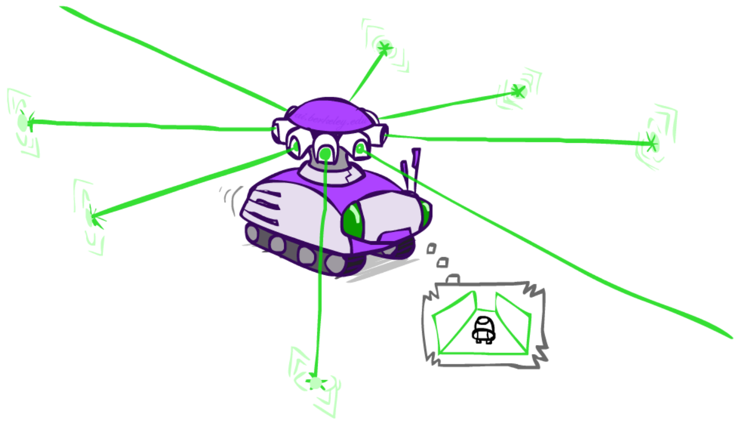


t=4

Example: Robot Localization



Prob 0 1



t=5

Filtering algorithm

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

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

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

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

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

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

Marginal Probability

Normalization Trick / Bayes Rule

Definition of HMM

Simple factoring of a constant

Filtering algorithm

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

Normalize — Update — Predict

- $\boldsymbol{f}_{1:t+1} = \textsc{Forward}(\boldsymbol{f}_{1:t}, e_{t+1})$
- 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
  - Will introduce approximate filtering algorithms soon

Summary: Filtering

- Filtering is the inference process of finding a distribution over X_T given e_1 through e_T : $P(X_T \mid e_{1:t})$
- We first compute $P(X_1 \mid e_1)$: $P(x_1|e_1) \propto P(x_1) \cdot P(e_1|x_1)$
- For each t from 2 to T, we have $P(X_{t-1} \mid e_{1:t-1})$
- **Elapse time:** compute $P(X_t \mid e_{1:t-1})$

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

- **Observe:** compute $P(X_t \mid e_{1:t-1}, e_t) = P(X_t \mid e_{1:t})$

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

Example: Weather HMM

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

predict

$P(s_1|w_0) = .6 = .5*.9+.5*.3$
$P(r_1|w_0) = .4 = 5*.1+.5*.7$

update

$f(sun) = 0.25 = .2*.6/(.2*.6 + .9*.4)$
$f(rain) = 0.75 = .9*.4/(.2*.6 + .9*.4)$

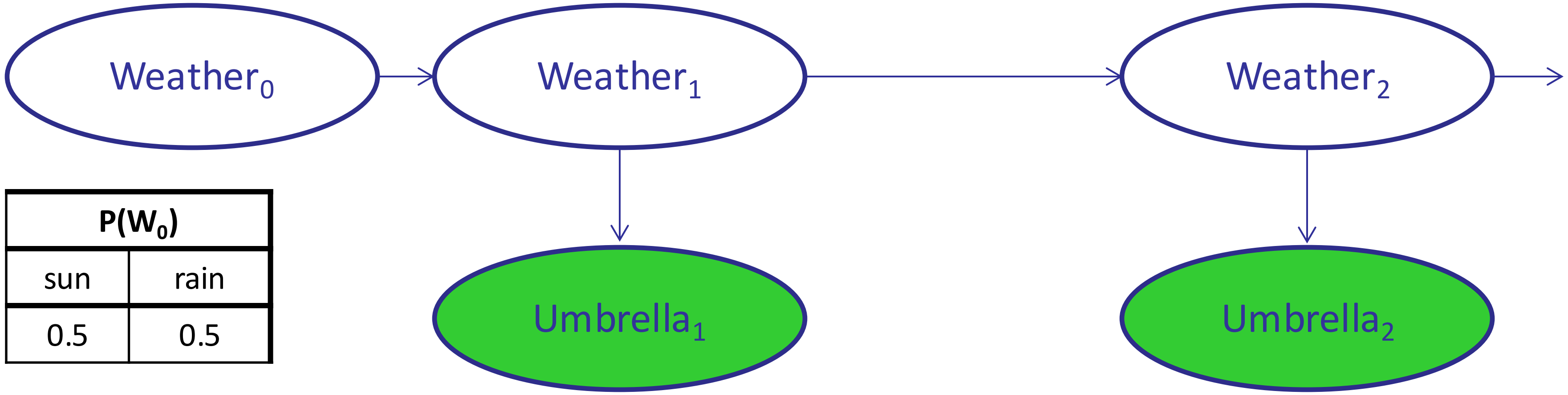
predict

$P(s_2|w_1) = 0.45 = .25*.9 + .75*.3$
$P(r_2|w_1) = 0.55 = .25*.1 + .75*.7$

update

$f(sun) = 0.154 = .2*.45/(.2*.45 + .9*.55)$
$f(rain) = 0.846 = .9*.55/(.2*.45+ .9*.55)$

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



P(W0)	
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

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

Video of Demo Pacman – Sonar

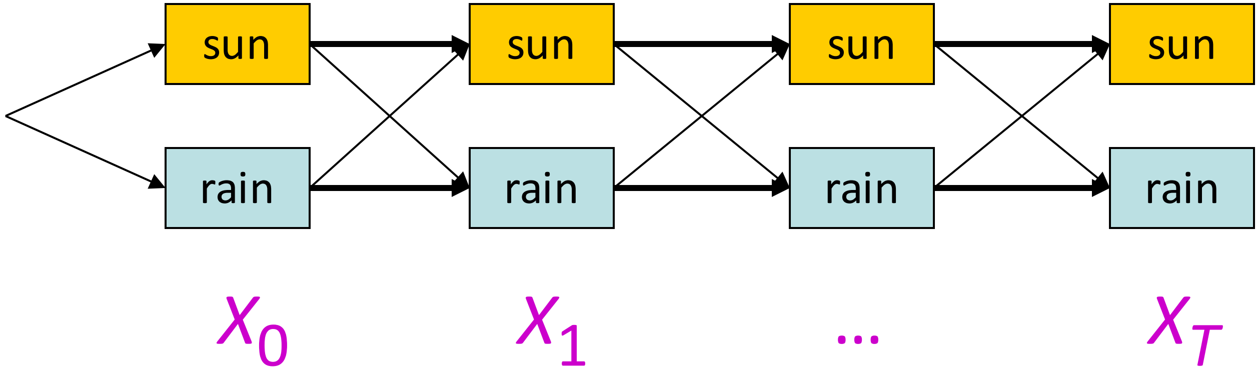
Most Likely Explanation

Inference tasks

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

Most likely explanation = most probable path*

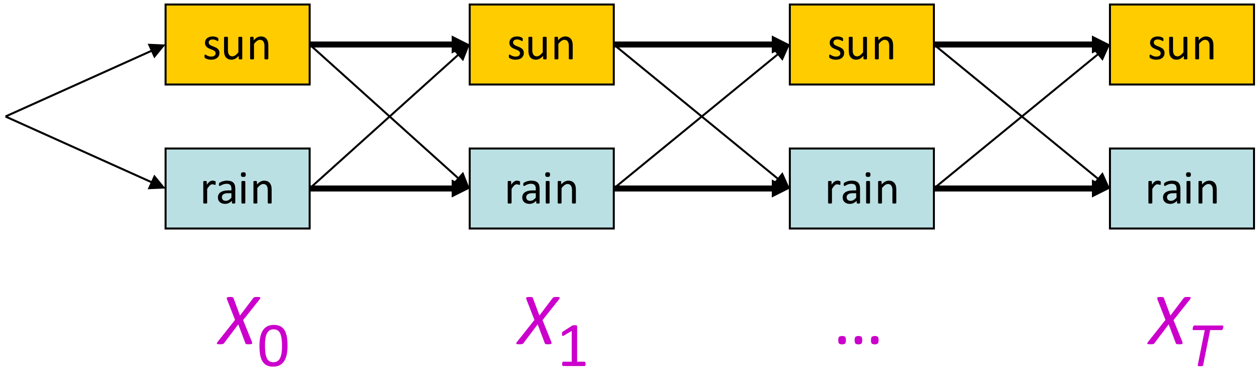
- ***State trellis***: graph of states and transitions over time



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

- 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

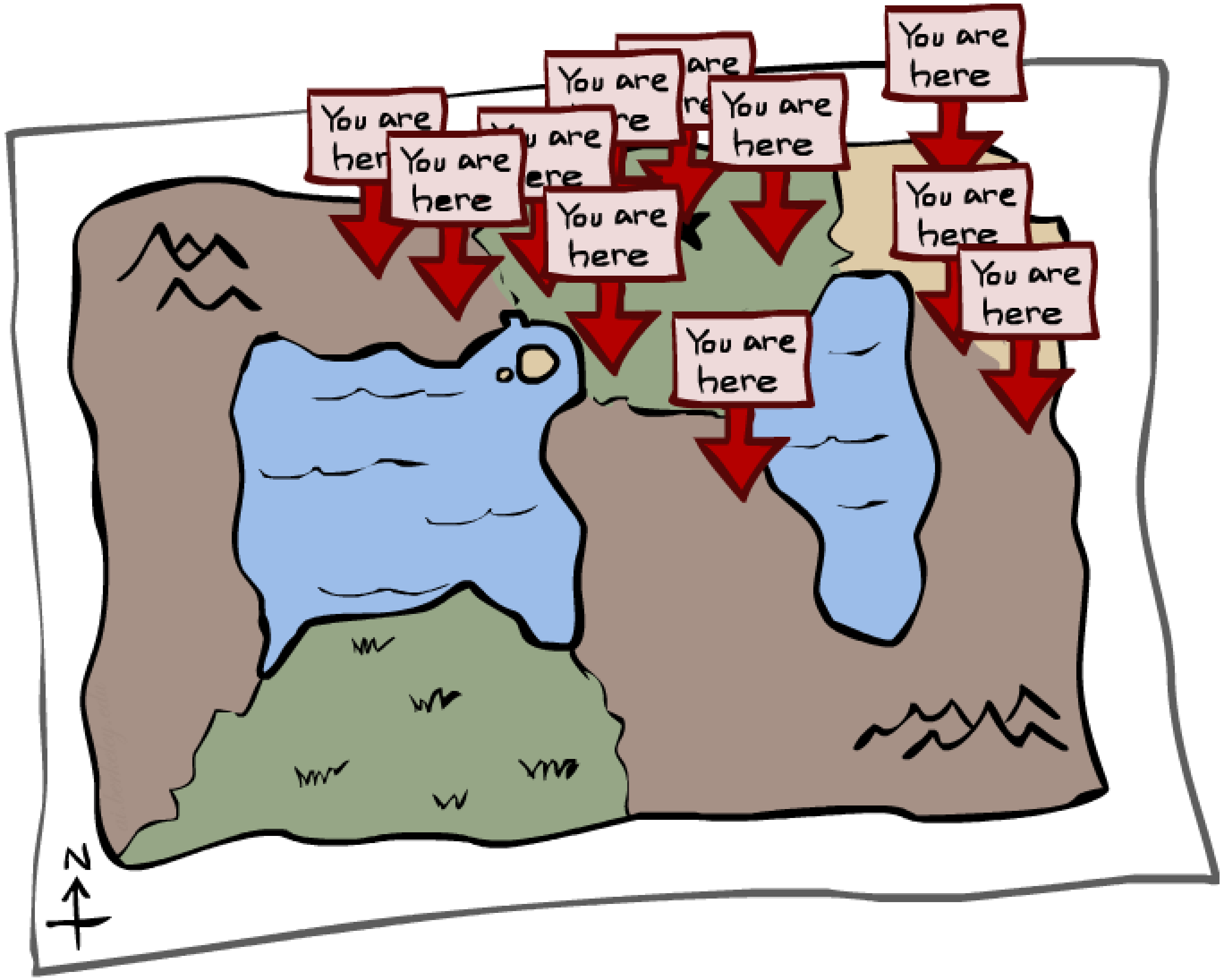
$$f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$$
$$= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} \mathbf{P}(X_{t+1} | x_t)\, f_{1:t}$$

Viterbi Algorithm (max)

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

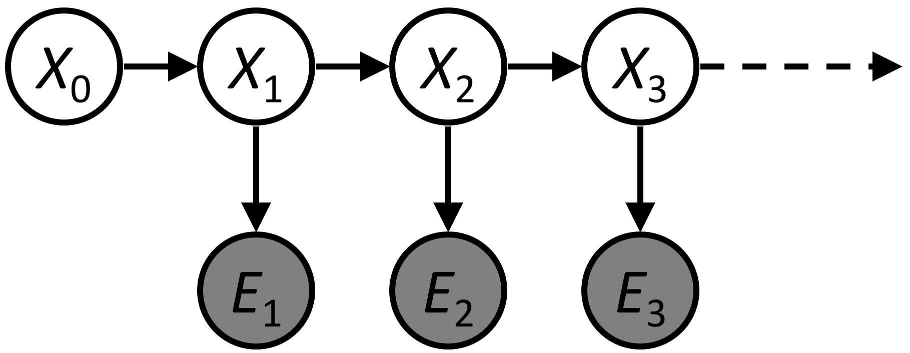
$$m_{1:t+1} = \text{VITERBI}(m_{1:t}, e_{t+1})$$
$$= \mathbf{P}(e_{t+1} | X_{t+1}) \max_{x_t} \mathbf{P}(X_{t+1} | x_t)\, m_{1:t}$$

Particle Filtering



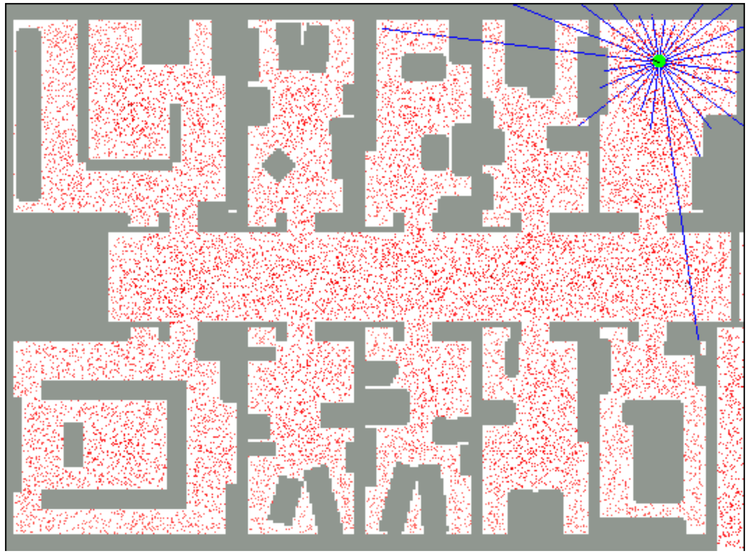
We need a new algorithm!

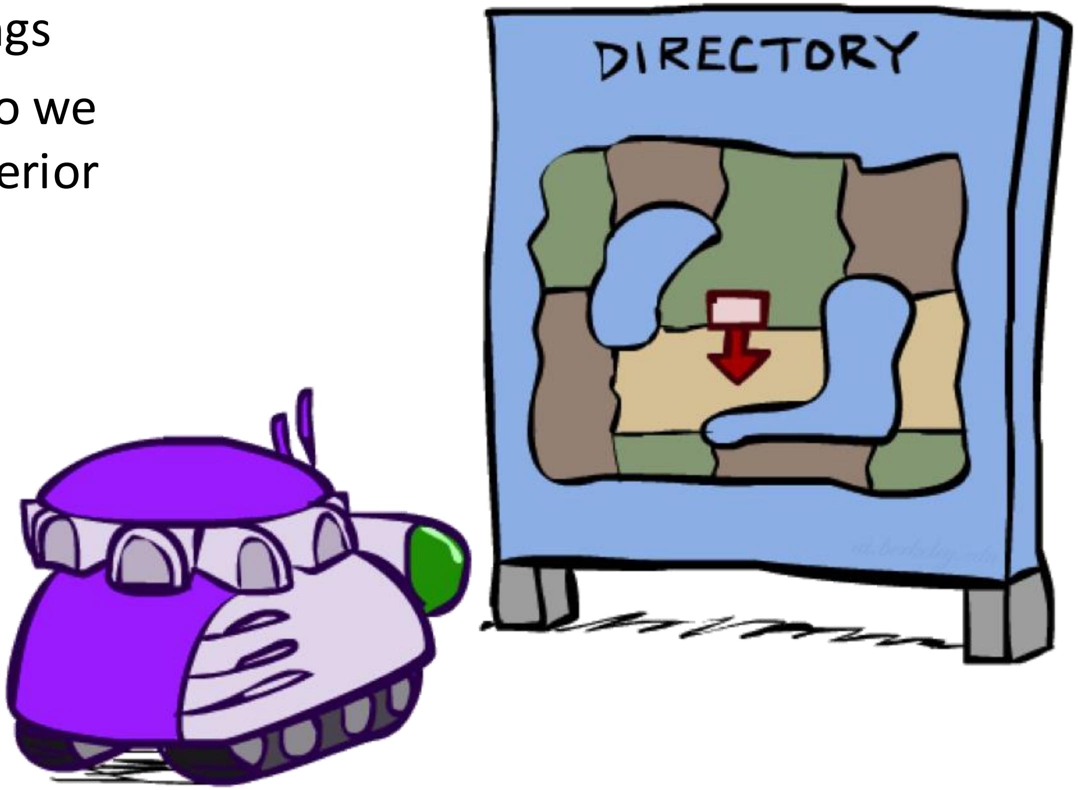
- When $|X|$ is grows, exact inference becomes infeasible
  - $O(|X|^2)$ cost per time step
  - (e.g., 3 ghosts in a 10x20 world, continuous domains)



Robot Localization

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





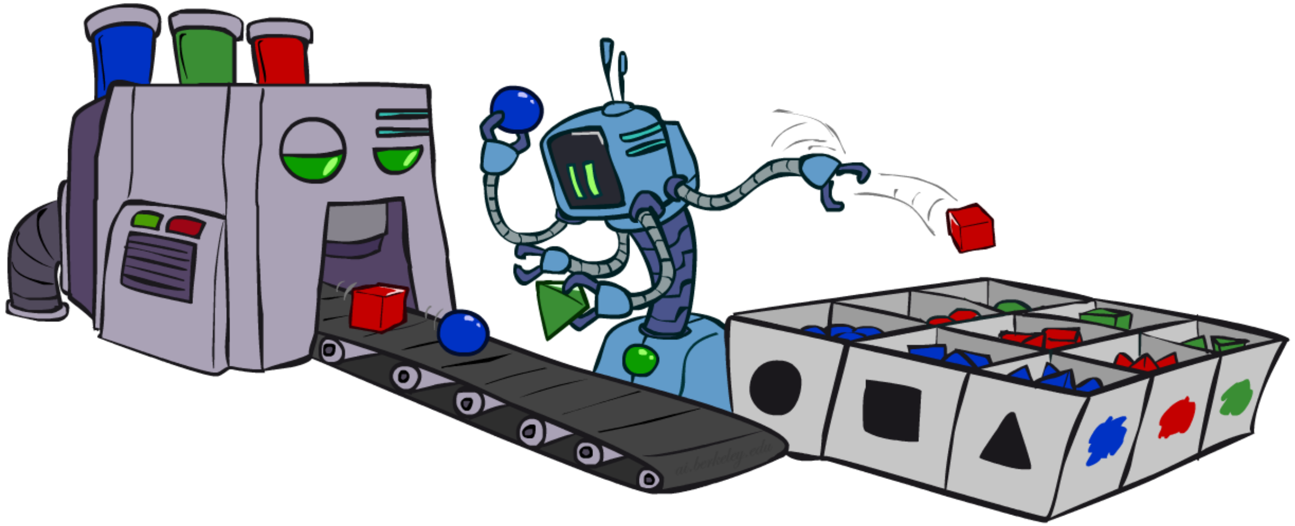
Particle Filter Localization (Sonar)

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

Sampling

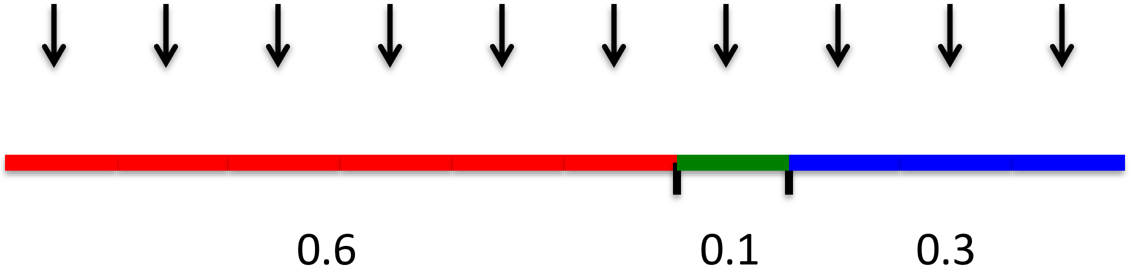
- Basic idea
  - Draw *N* samples from a ***sampling distribution*** *S*
  - Compute an approximate posterior probability
  - Show this converges to the true probability *P*

- Why sample?
  - Often very fast to get a decent approximate answer
  - The algorithms are very simple and general (easy to apply to fancy models)
  - They require very little memory ($O(n)$)
  - They can be applied to large models, whereas exact algorithms blow up



Sampling basics: discrete (*categorical*) distribution

- To simulate a biased d-sided coin:
  - Step 1: Get sample *u* from uniform distribution over [0, 1)
    - E.g. random() in python
  - Step 2: Convert this sample *u* into an outcome for the given distribution by associating each outcome *x* with a *P*(*x*)-sized sub-interval of [0,1)



- Example

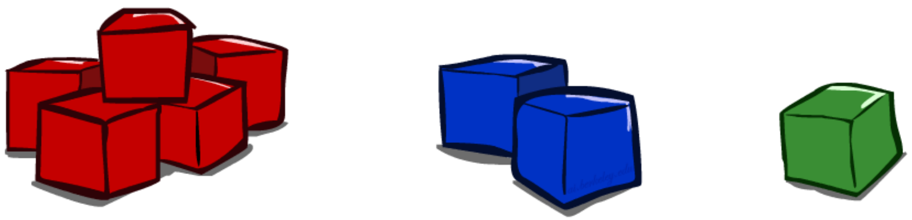
C	*P*(*C*)
red	0.6
green	0.1
blue	0.3

$0.0 \leq u < 0.6, \rightarrow C=\text{red}$

$0.6 \leq u < 0.7, \rightarrow C=\text{green}$

$0.7 \leq u < 1.0, \rightarrow C=\text{blue}$

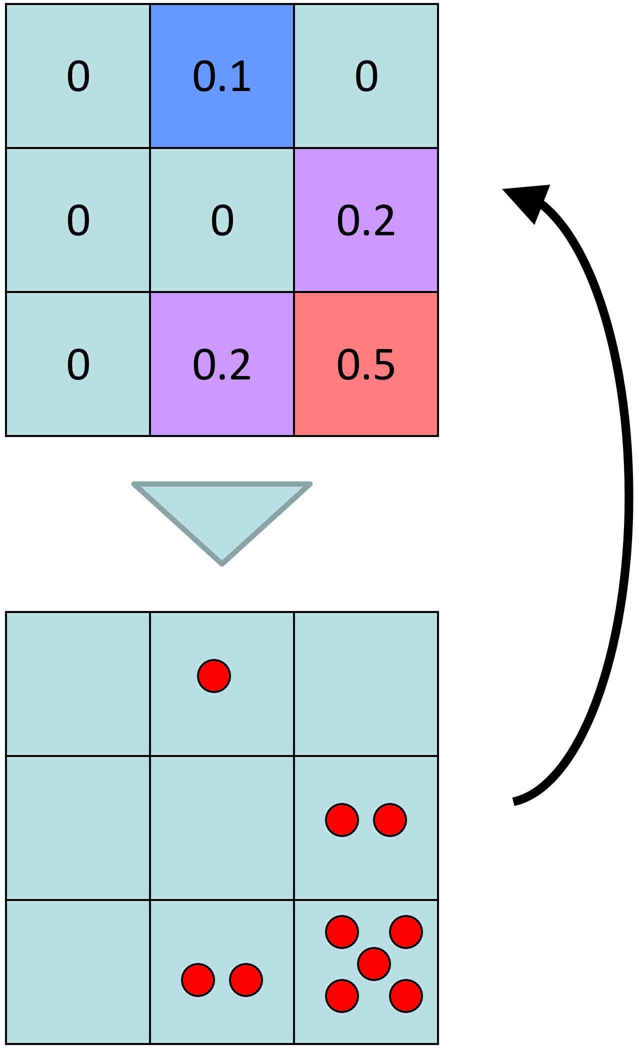
- If random() returns *u* = 0.83, then the sample is *C* = *blue*
- E.g, after sampling 8 times:



Particle Filtering

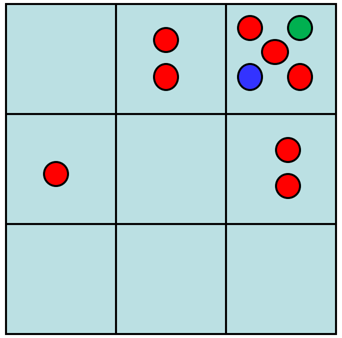
- Represent belief state by a set of samples
  - Samples are called ***particles***
  - Time per step is linear in the number of samples
  - But: number needed may be large

- This is how robot localization works in practice



Representation: Particles

- Our representation of $P(X)$ is now a list of $N << |X|$ particles
- $P(x)$ approximated by number of particles with value x
  - So, many x may have $P(x) = 0$!
  - More particles => more accuracy (cf. frequency histograms)
  - Usually we want a ***low-dimensional*** marginal
    - E.g., “Where is ghost 1?” rather than “Are ghosts 1,2,3 in [2,6], [5,6], and [8,11]?”



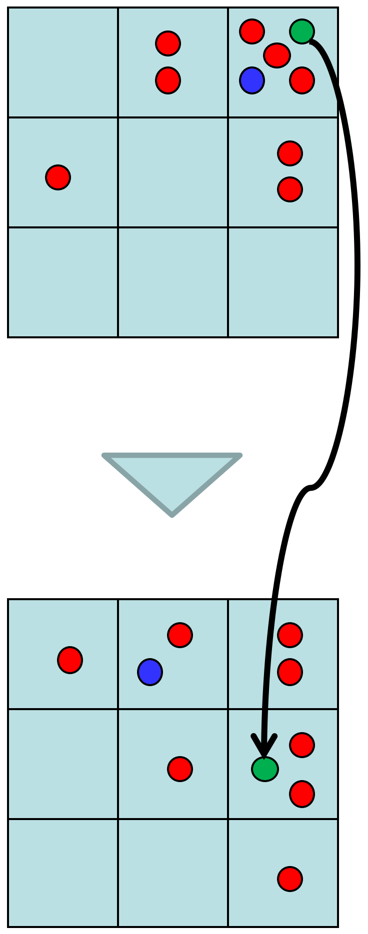
Particles:
(1,2)
(2,3)
(2,3)
(3,2)
(3,2)
(3,3)
(3,3)
(3,3)
(3,3)
(3,3)

Particle Filtering: Prediction step

- Particle j in state $x_t^{(j)}$ samples a new state directly from the transition model:
  - $x_{t+1}^{(j)} \sim P(X_{t+1} \mid x_t^{(j)})$
  - Here, most samples move clockwise, but some move in another direction or stay in place
- For example:

$$x_{t+1}^{(j)} \sim P(X_{t+1} \mid x_t^{(green)}) = \langle P((3,3) \mid (3,3)), P((2,3) \mid (3,3)), P((3,2) \mid (3,3)) \rangle$$
$$= \langle 1/3, 1/3, 1/3 \rangle$$

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

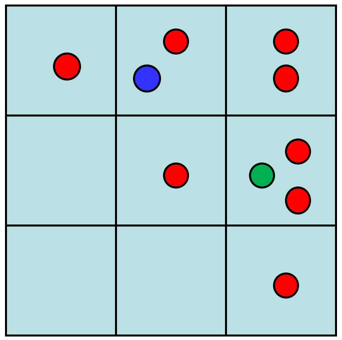


Particles:
(1,3)
(2,2)
(2,3)
(2,3)
(3,1)
(3,2)
(3,2)
(3,2)
(3,3)
(3,3)

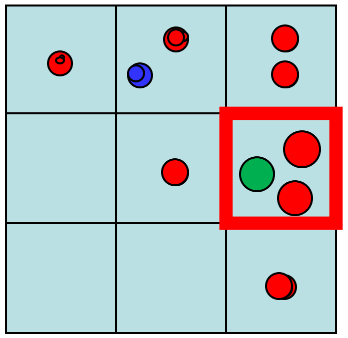
Particle Filtering: Update step

- After observing e_{t+1} :
  - As in likelihood weighting, weight each sample based on the evidence
    - $w^{(j)} = P(e_{t+1} | x_{t+1}^{(j)})$
  - Normalize the weights: particles that fit the data better get higher weights, others get lower weights
- For example, say $e_{t+1} = (3,2)$
  - $w^{(green)} = P((3,2) | (3,2)) = .9$
  - $w^{(blue)} = P((3,2) | (2,3)) = .2$

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

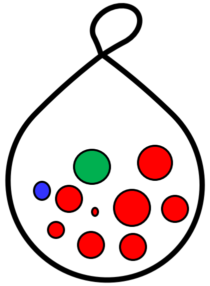


Particles:
(1,3) w=.1
(2,2) w=.4
(2,3) w=.2
(2,3) w=.2
(3,1) w=.4
(3,2) w=.9
(3,2) w=.9
(3,2) w=.9
(3,3) w=.4
(3,3) w=.4



Particle Filtering: Resample

- Rather than tracking weighted samples, we ***resample***
- *N* times, we choose from our weighted sample distribution
  - $x_{t+1}^{(j)} \sim N(X_{t+1} \mid e_{1:t}) / N = \alpha\, W(X_{t+1} \mid e_{1:t})$
  - (i.e., draw with replacement)
- Now the update is complete for this time step, continue with the next one (with weights reset to 1)

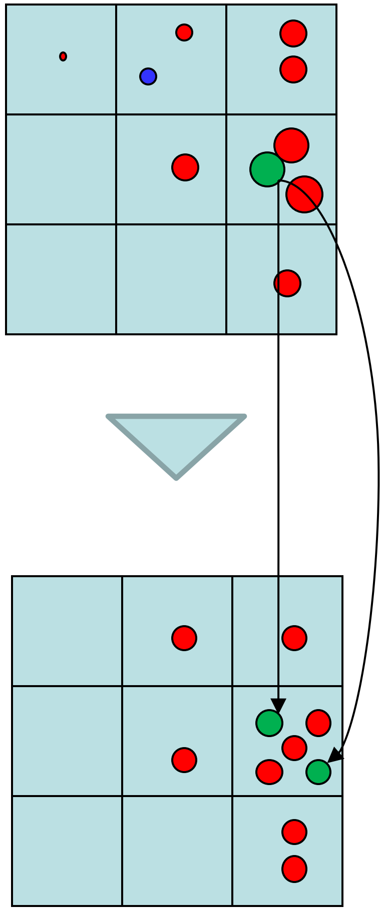


.02	.08	.17
0	.08	.56
0	0	.08

routine weighted-sample:
  return random() in $\alpha\, W(X_{t+1} \mid e_{1:t})$

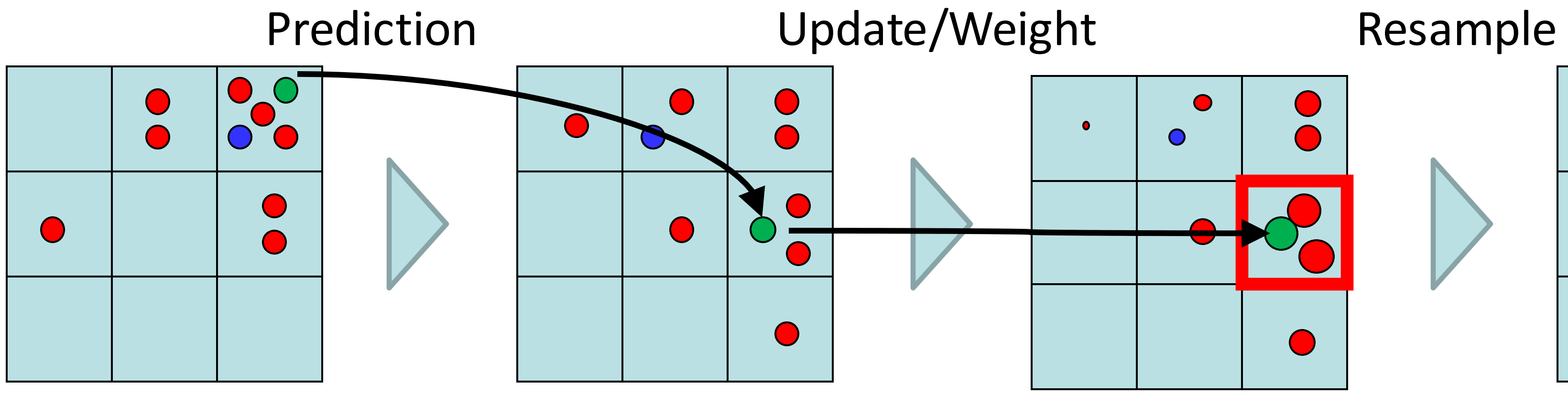
Particles:
(1,3) w=.1
(2,2) w=.4
(2,3) w=.2
(2,3) w=.2
(3,1) w=.4
(3,2) w=.9
(3,2) w=.9
(3,2) w=.9
(3,3) w=.4
(3,3) w=.4

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



Summary: Particle Filtering

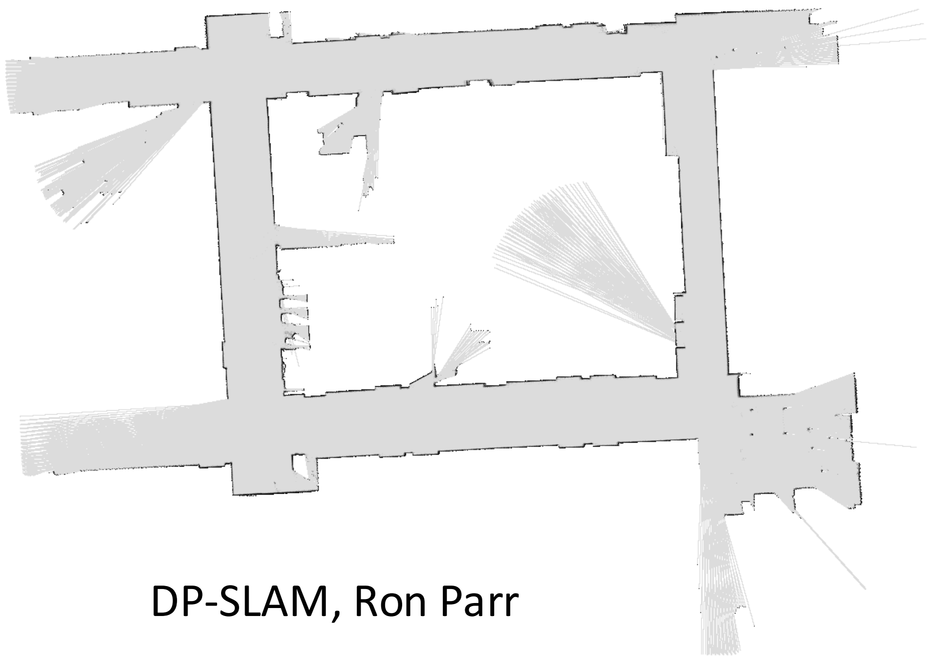
- Particles: track samples of states rather than an explicit distribution



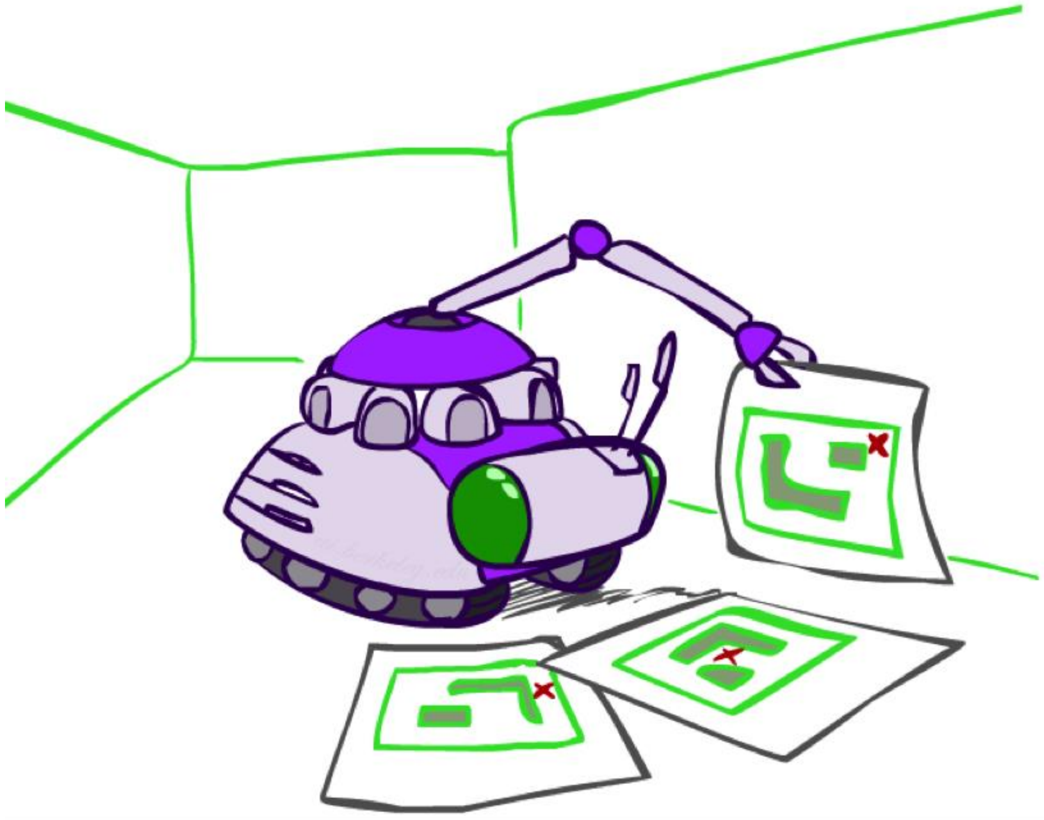
Particles:	Particles:	Particles:	(New) Particles:
(1,2)	(1,3)	(1,3) w=.1	(2,2)
(2,3)	(2,2)	(2,2) w=.4	(2,3)
(2,3)	(2,3)	(2,3) w=.2	(3,1)
(3,2)	(2,3)	(2,3) w=.2	(3,1)
(3,2)	(3,1)	(3,1) w=.4	(3,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,3)	(3,3)	(3,3) w=.4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

Robot Mapping

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

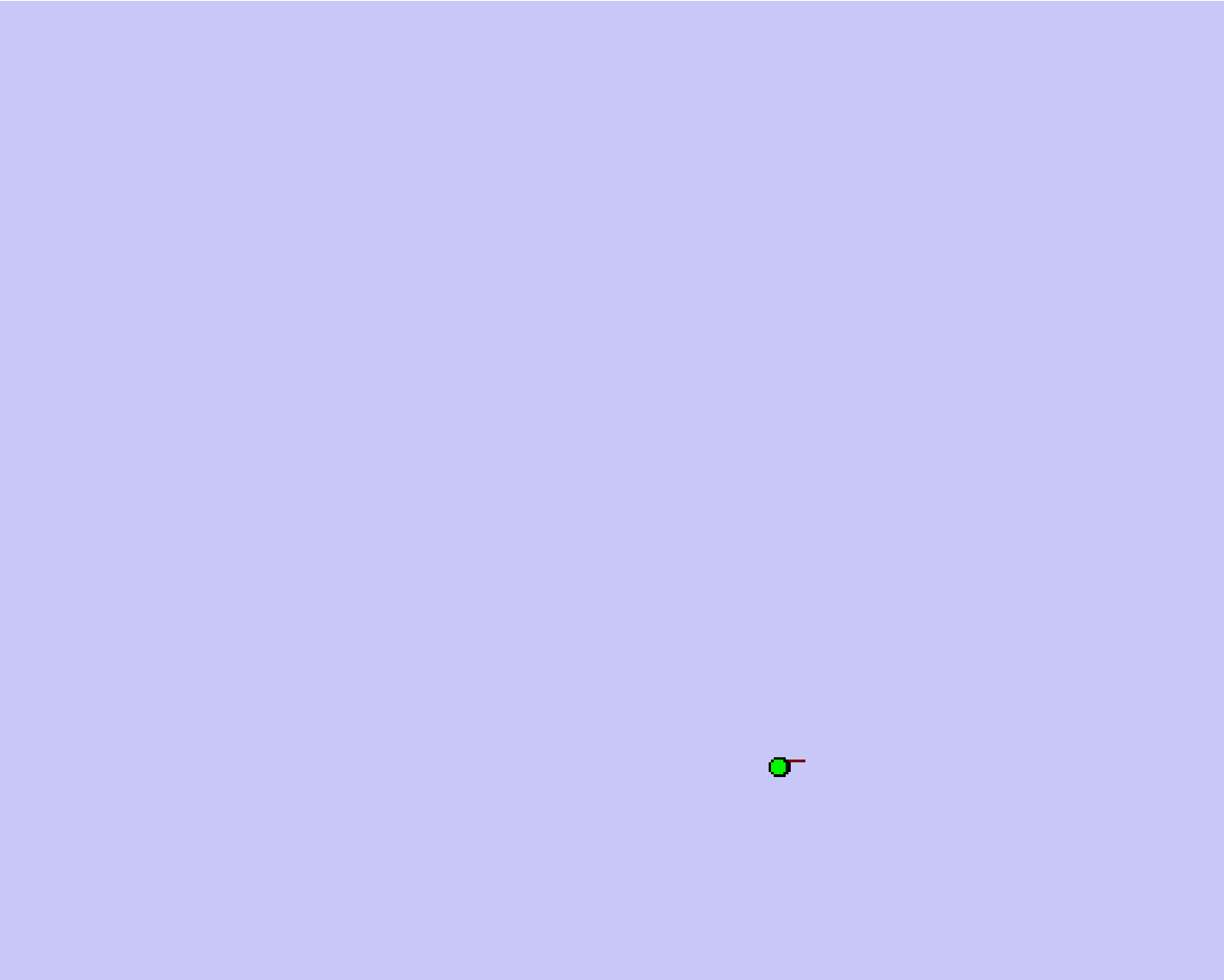


DP-SLAM, Ron Parr



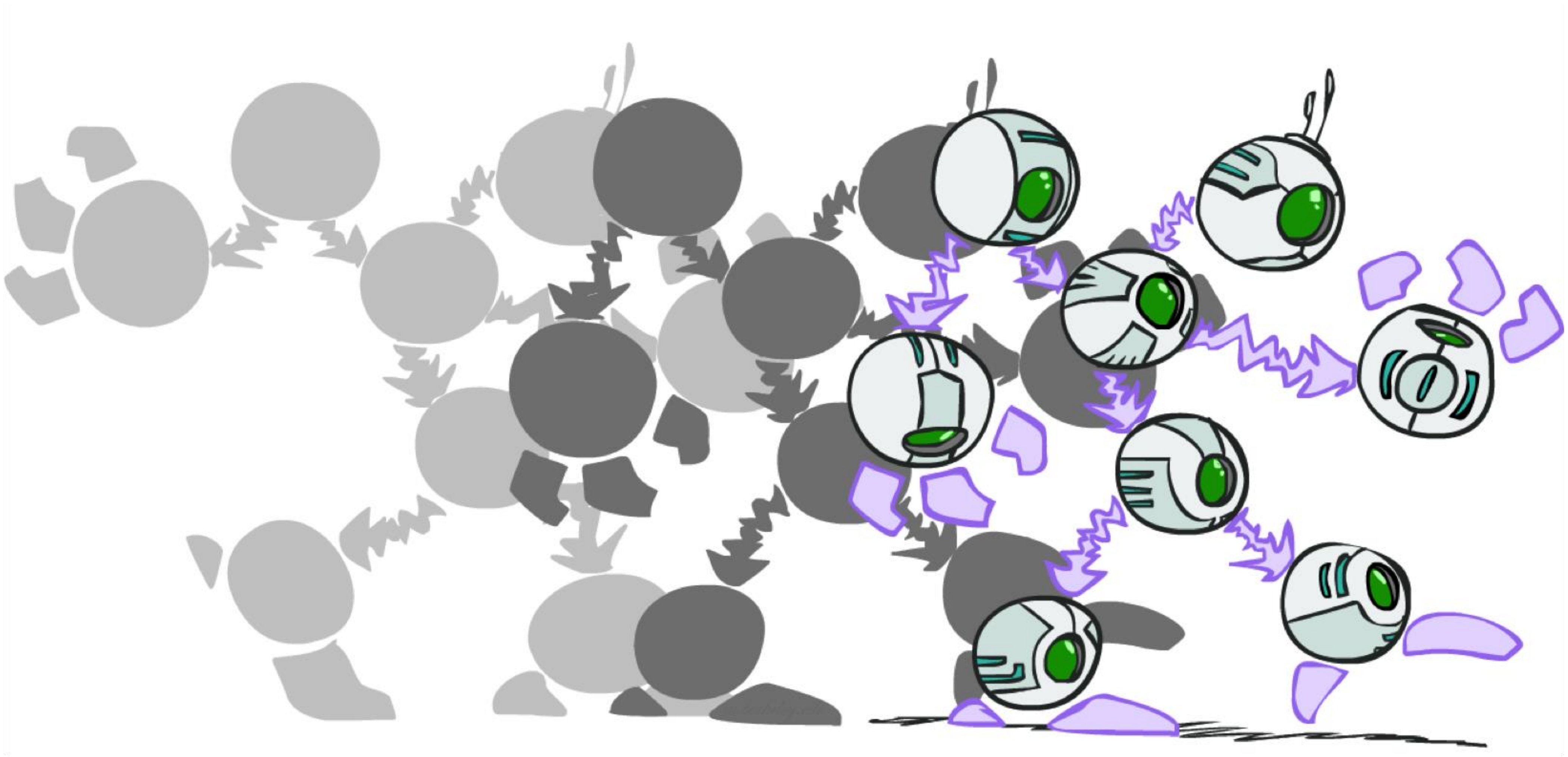
[Demo: PARTICLES-SLAM-mapping1-new.avi]

Particle Filter SLAM – Video



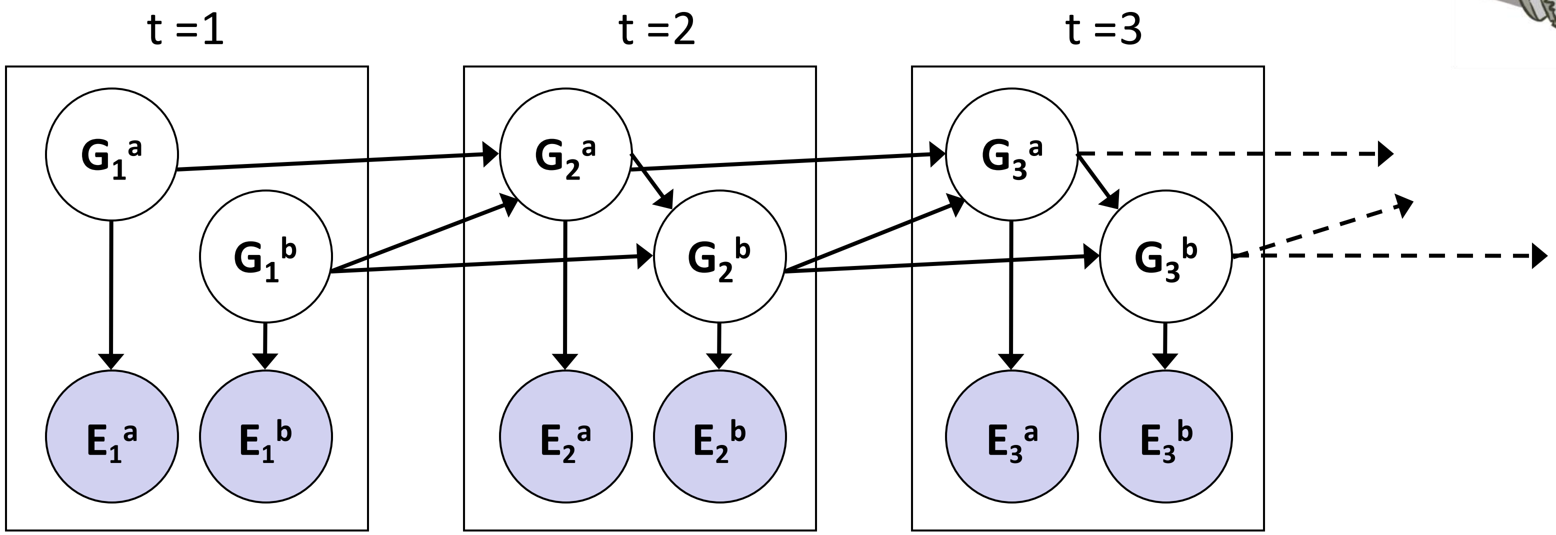
[Demo: PARTICLES-SLAM-fastslam.avi]

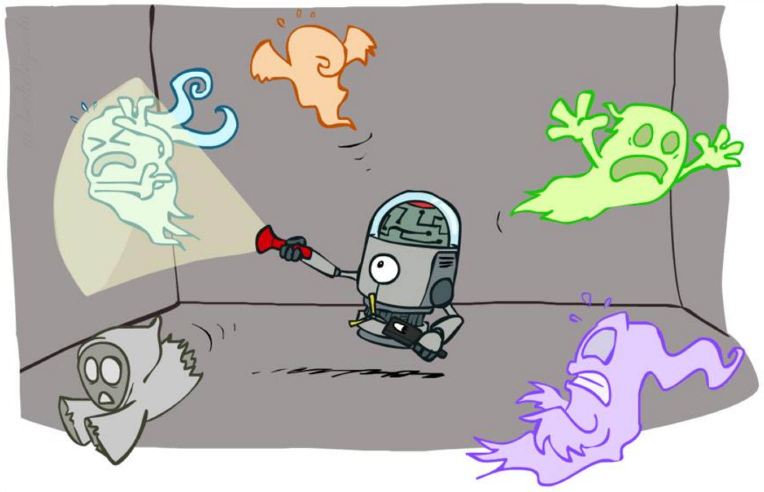
Dynamic Bayes' Nets

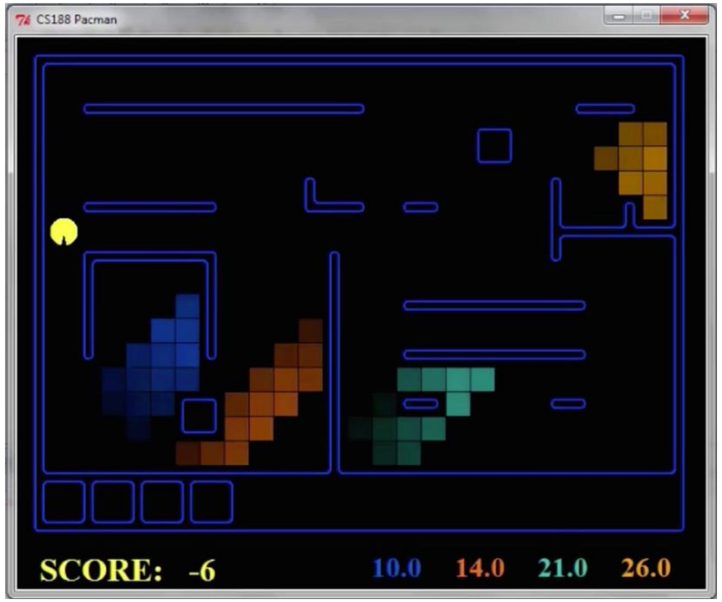


Dynamic Bayes Nets (DBNs)

- We want to track multiple variables over time, using multiple sources of evidence
- Idea: Repeat a fixed Bayes net structure at each time
- Variables at time *t* can have parents at time *t-1*

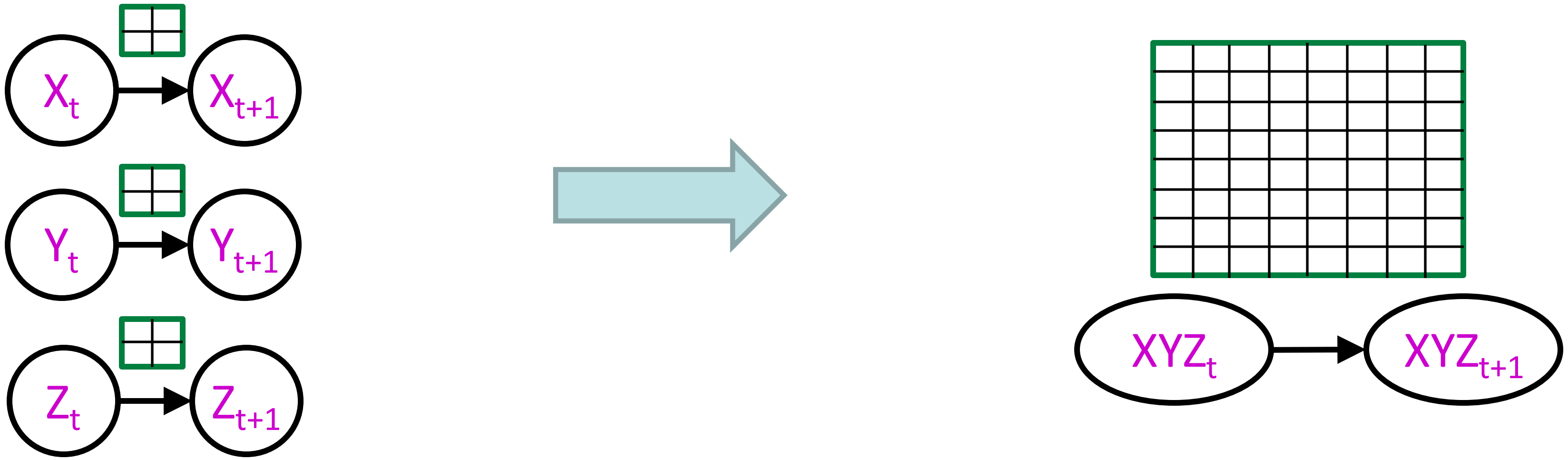






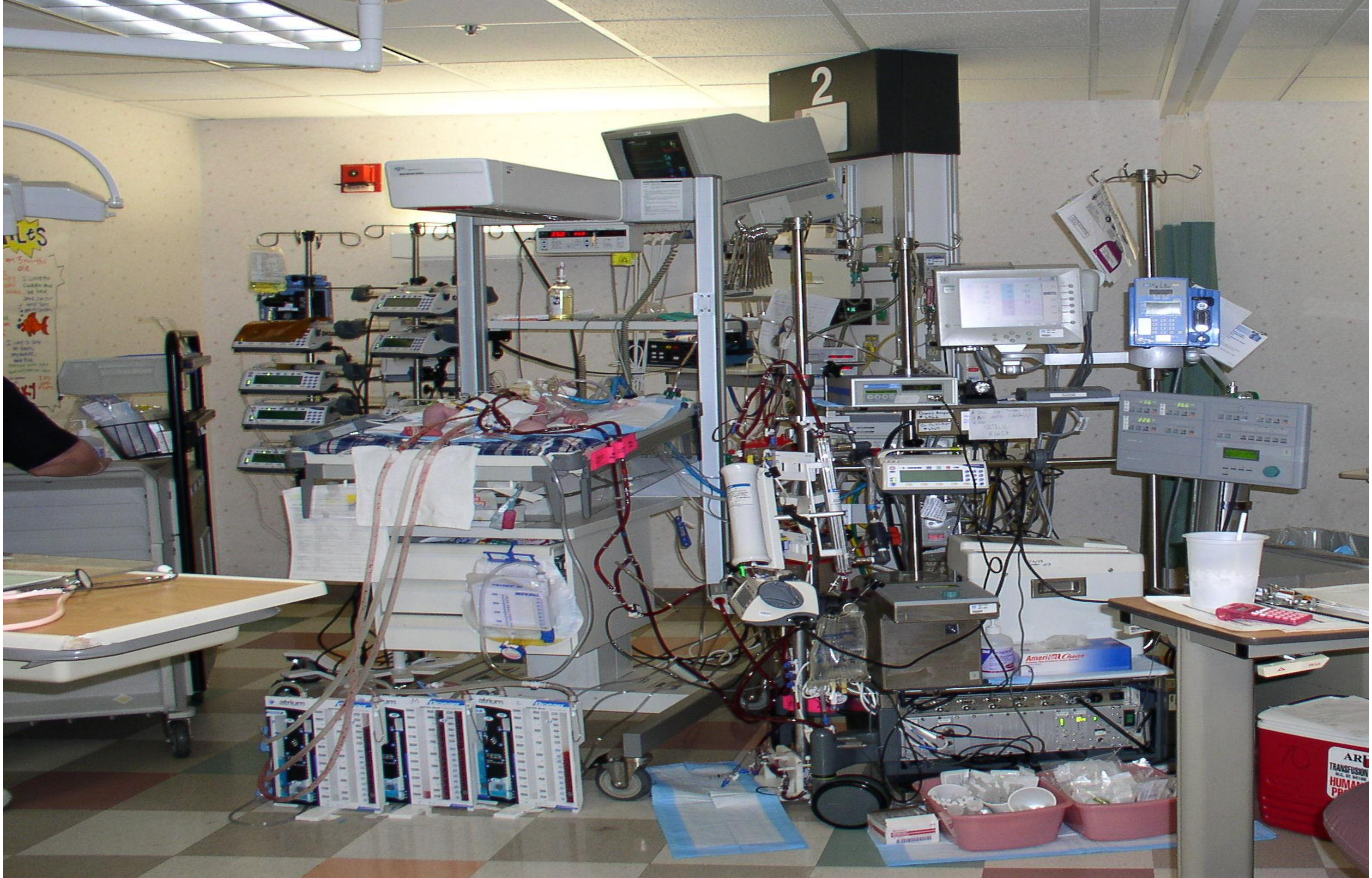
DBNs and HMMs

- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables



- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 state variables, 3 parents each;

    DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} =\sim 10^{12}$ parameters

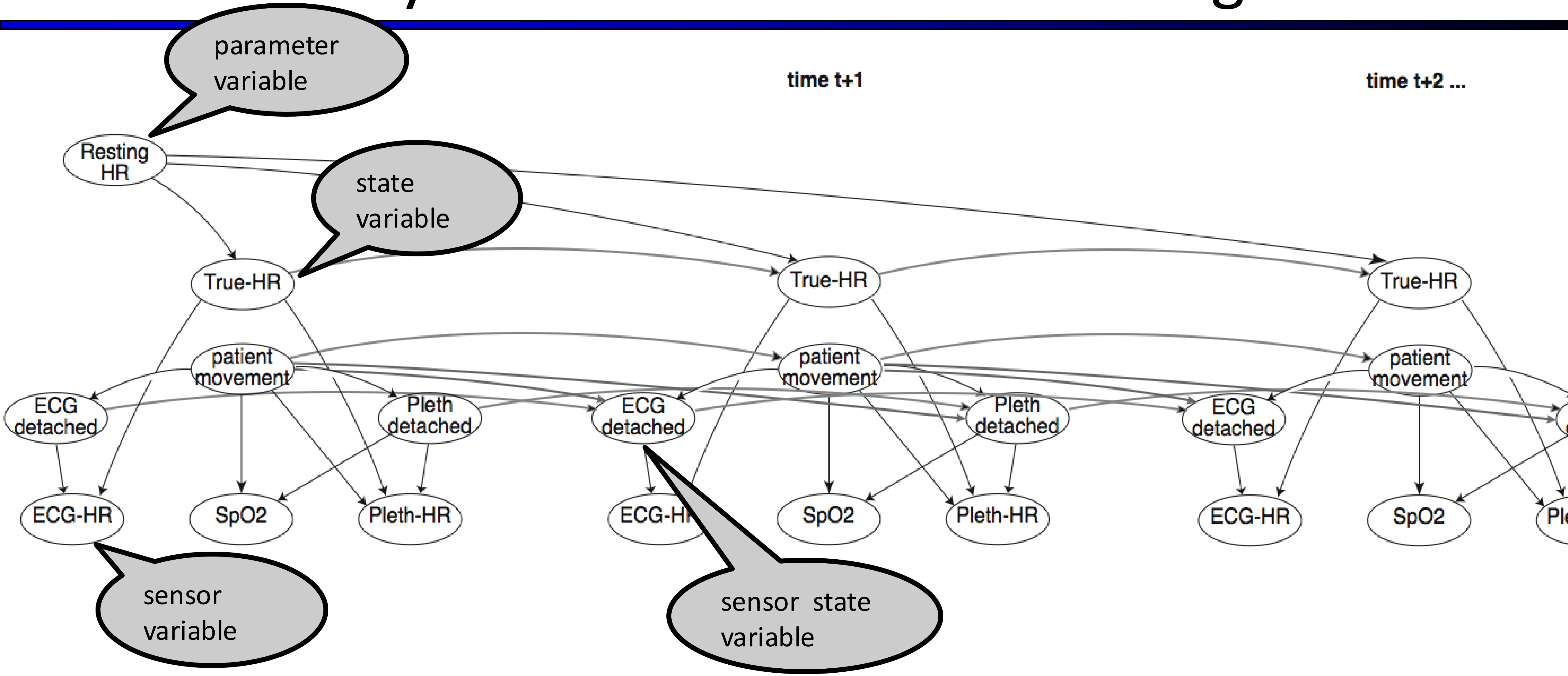


2
atrium
atrium

Application: ICU monitoring

- ***State***: variables describing physiological state of patient
- ***Evidence***: values obtained from monitoring devices
- ***Transition model***: physiological dynamics, sensor dynamics
- ***Query variables***: pathophysiological conditions (a.k.a. bad things)

Toy DBN: heart rate monitoring



The enhanced heart-rate DBN's inferences on data from a healthy 40-year-old man