CS 4100 // artificial intelligence

instructor: byron wallace

Hidden Markov Models

Attribution: many of these slides are modified versions of those distributed with the *UC Berkeley CS188* materials
Thanks to John DeNero and Dan Klein

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 ($X \perp\!\!\!\perp Y \,|\, Z$) if and only if:

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

Last time: Markov Models

Assume the weather has three states

1. Rainy
2. Cloudy
3. Sunny

What is the probability of the sequence

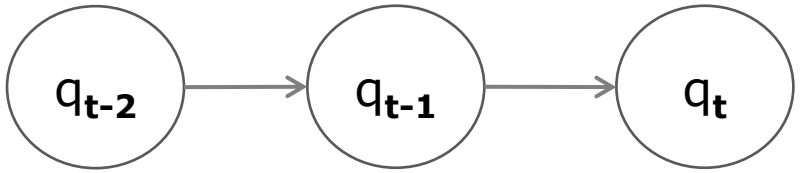
$$\mathbf{O} = \langle S_3, S_3, S_1, S_2 \rangle$$

$$\begin{aligned} p(\mathbf{O}) = \; & p(q_4 = S_2 | q_3 = S_1, q_2 = S_3, q_1 = S_3) \\ & p(q_3 = S_1 | q_2 = S_3, q_1 = S_3) \\ & p(q_2 = S_3 | q_1 = S_3) p(q_1 = S_3) \end{aligned}$$

Chain Rule

Markov Models

Markov assumption: The state at time *t* (q_t) is conditionally independent of all other states given the value of the previous state (q_{t-1})



$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | Parents(X_i))$$

$$\begin{aligned} p(\mathbf{O}) &= p(q_4 = S_2 | q_3 = S_1) p(q_3 = S_1 | q_2 = S_3) \\ & \quad p(q_2 = S_3 | q_1 = S_3) p(q_1 = S_3) \end{aligned}$$

Markov Models

Generally parameterized by

- Transition matrix ***A***

$$a_{ij} \geq 0 \qquad \sum_j a_{ij} = 1$$

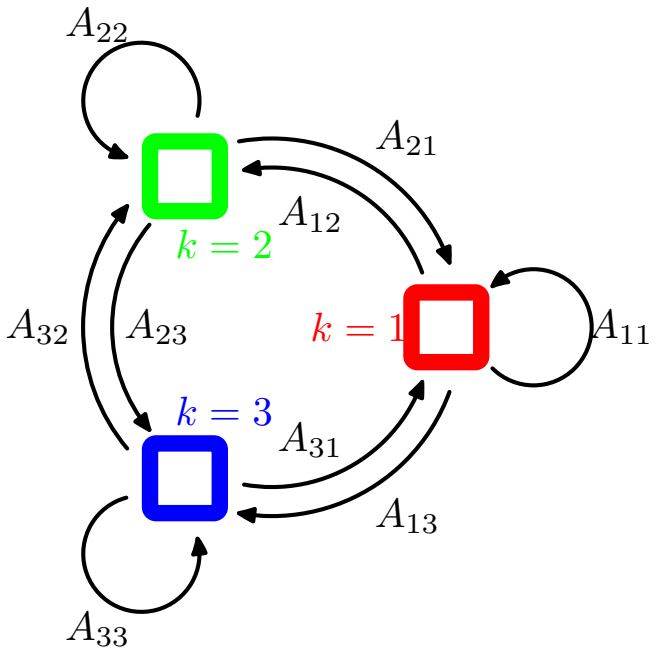
$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

- Initial State Distribution $\boldsymbol{\pi}$

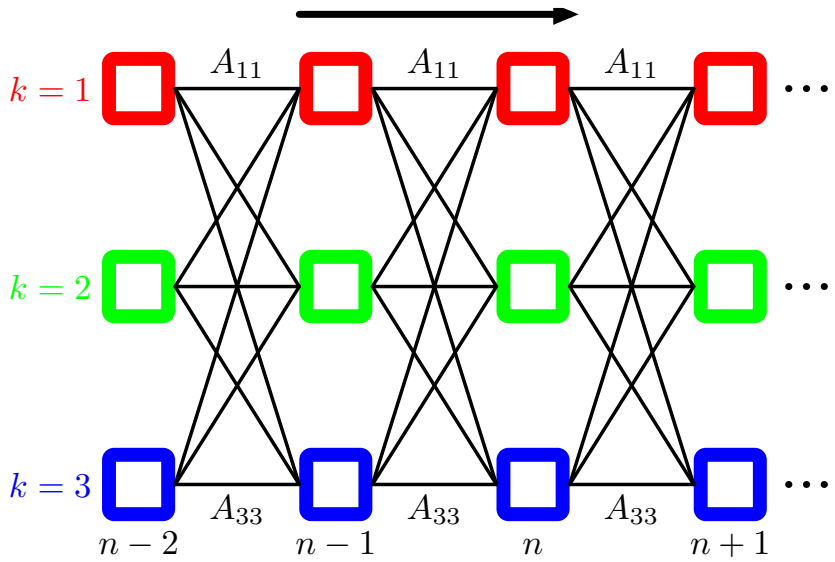
$$\pi_i \geq 0 \qquad \sum_i \pi_i = 1$$

$$\boldsymbol{\pi} = \{\pi_i\} = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix}$$

Markov Models



State Diagram



Unfolded “Lattice”

[Bishop]

Today: *Hidden* Markov Models (HMMs)

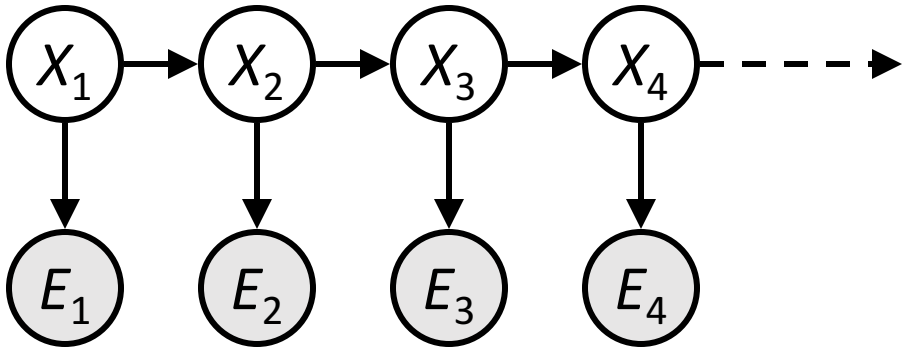
Hidden Markov Models

Markov chains not so useful for most agents

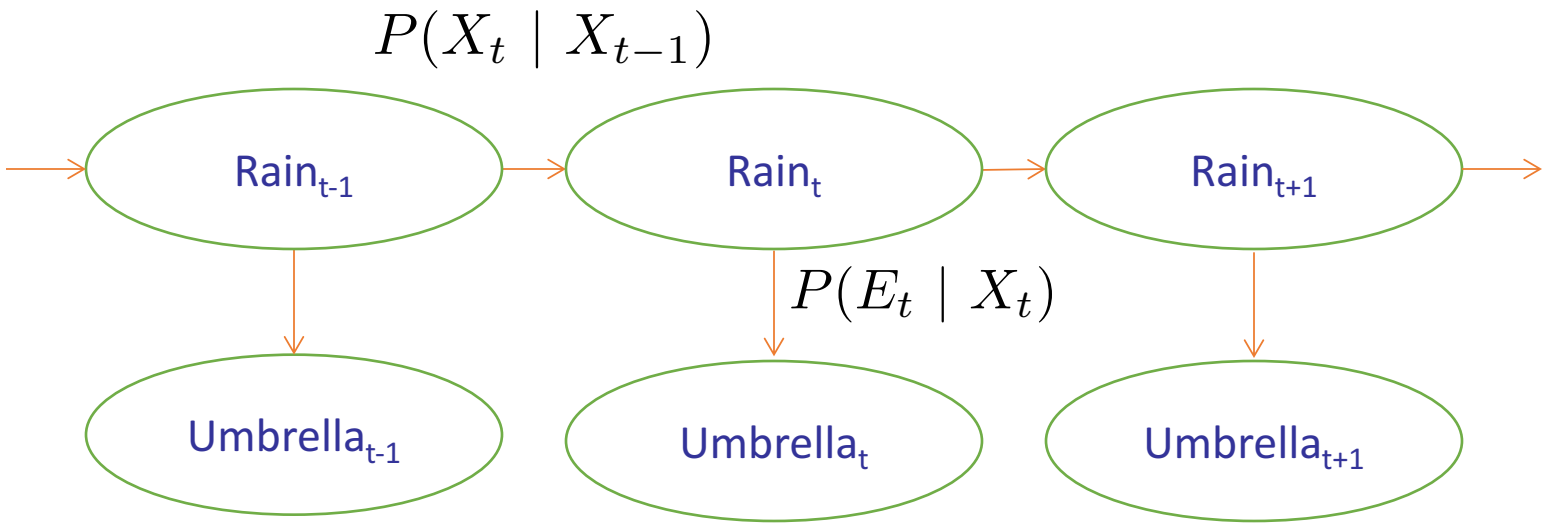
- Need observations to update your beliefs

Hidden Markov models (HMMs)

- Underlying Markov chain over states X
- You observe outputs (effects) at each time step



Example: weather HMM



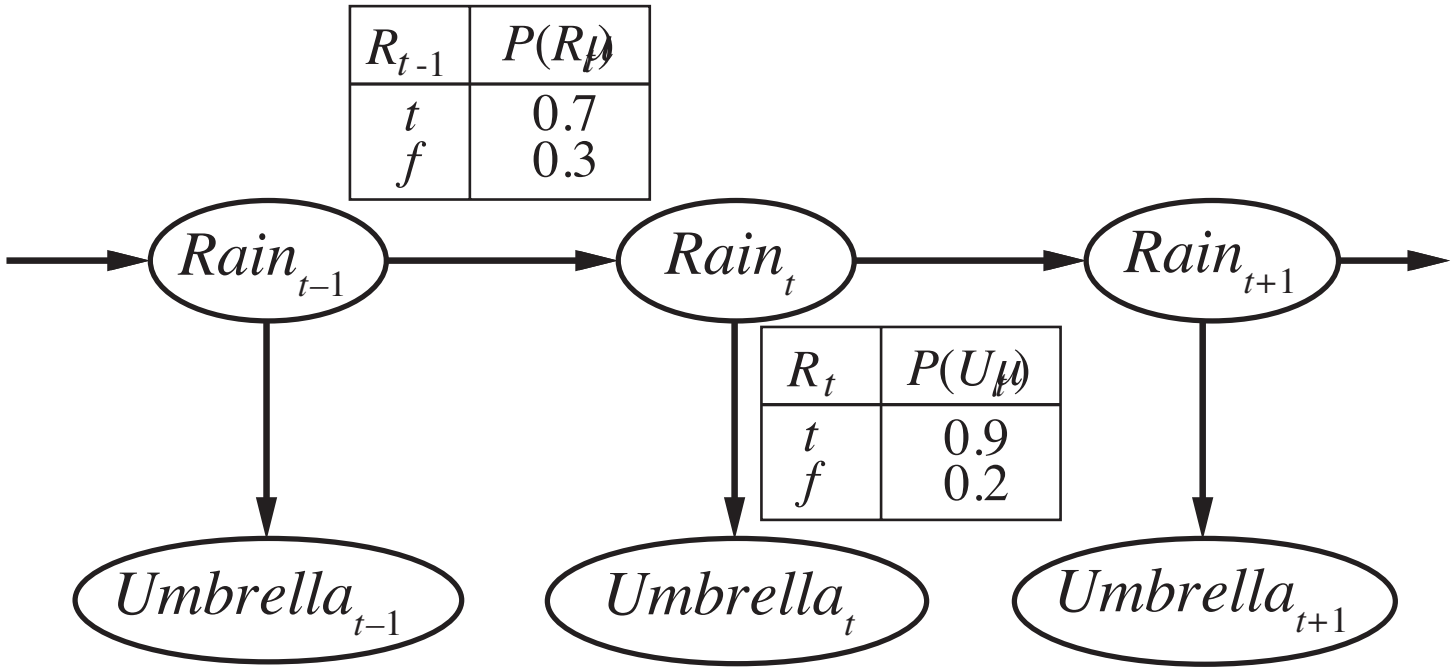
An HMM is defined by

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

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

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

Example: weather HMM

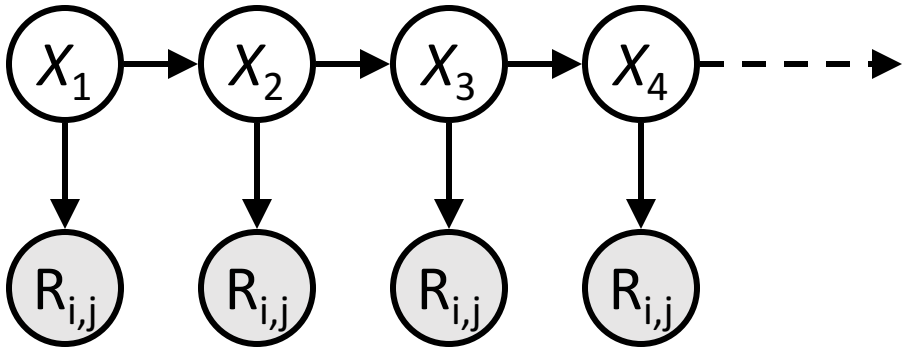


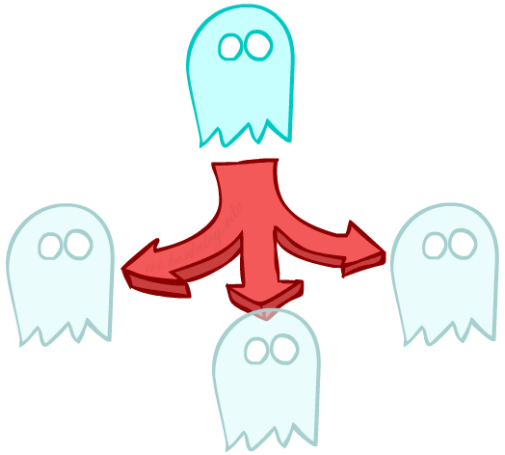
Support we observe an umbrella two days in a row.

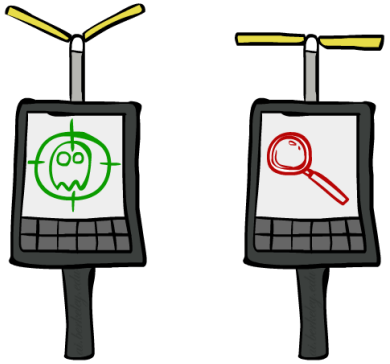
What is the probability that it's raining on the 2nd day?

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.







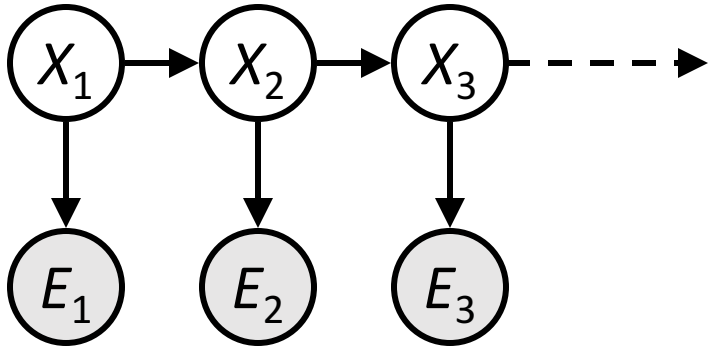
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_1)$

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

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

Joint distribution of an HMM



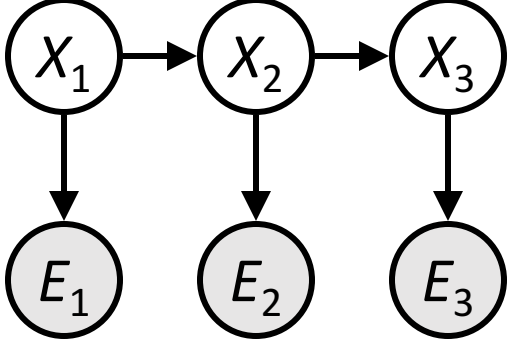
Joint distribution:

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

More generally:

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

The chain rule and HMMs



From the chain rule, every joint distribution over $X_1, E_1, X_2, E_2, X_3, E_3$ can be written as:

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

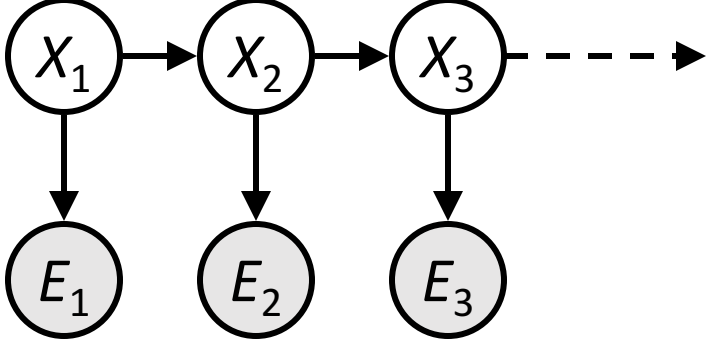
For HMMs we make the following (conditional) independence assumptions:

$$X_2 \perp\!\!\!\perp E_1 \mid X_1, \quad E_2 \perp\!\!\!\perp X_1, E_1 \mid X_2, \quad X_3 \perp\!\!\!\perp X_1, E_1, E_2 \mid X_2, \quad E_3 \perp\!\!\!\perp X_1, E_1, X_2, E_2 \mid X_3$$

And can rewrite as:

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

The chain rule and HMMs, in general



From the chain rule, every joint distribution over $X_1, E_1, \ldots, X_T, E_T$ can be written as:

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

Assuming that for all *t*:

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

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

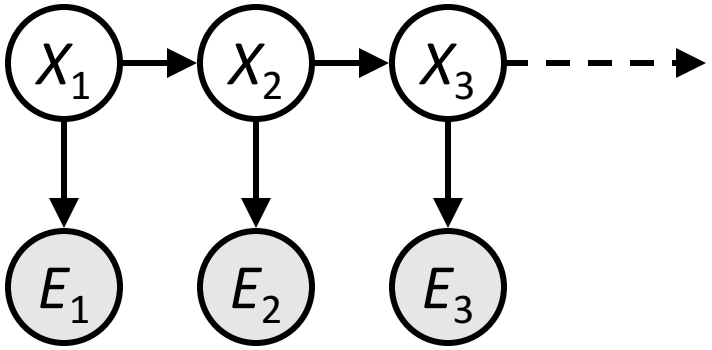
Evidence is independent of all past states and all past evidence given the current state, i.e.:

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

Which gives us:

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

Implied conditional independencies



Many implied conditional independencies, e.g.,

$$E_1 \perp\!\!\!\perp X_2, E_2, X_3, E_3 \mid X_1$$

We can prove these as we did last class for Markov models (but we won't today)

This also comes from the graphical model; we'll cover this more formally in a later lecture

An occasionally dishonest casino

Consider the following casino game

1. You bet $1 and roll a fair die
2. Casino dealer rolls a (sometimes fair, sometimes loaded die)
3. Highest number wins $2

Casino sequence of rolls

12455264621461461361366616646616366163661636616515615115146123562344

Which die is being used for each play?

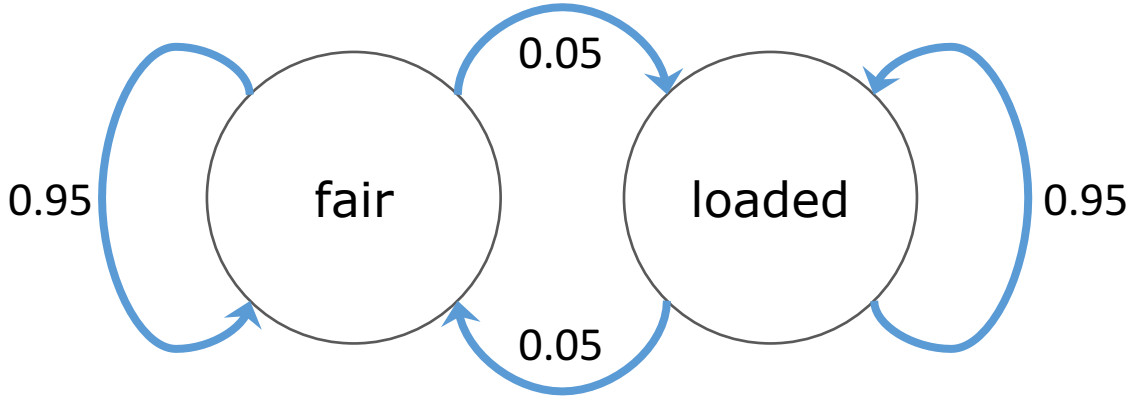
Dishonest casino model

- Assume 1 fair and 1 loaded die

$$p(x_i|f) = \frac{1}{6} \qquad p(x_i \neq 6|\neg f) = \frac{1}{10}$$

$$p(x_i = 6|\neg f) = \frac{1}{2}$$

- Secondly, assume that the casino cannot always use loaded die, but only switches occasionally



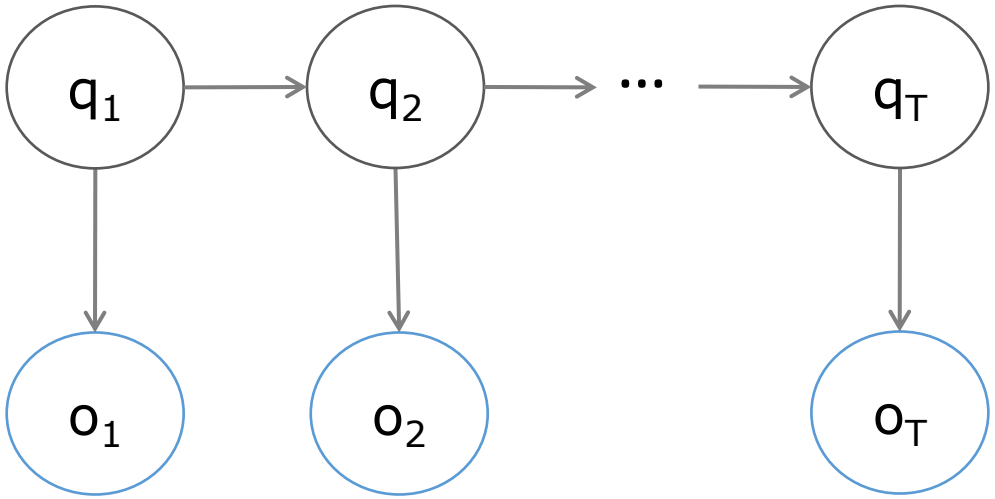
[example credit: Tom Mitchell]

Our betting game

Casino sequence of rolls

12455264621461461361366616646616366163661636165156151151461235623 44

Which die is being used for each play?



“Generative model” for casino game

1. *t=1*
2. Casino chooses initial die q_t (state selected)
3. Casino rolls die o_t (observation made)
4. Casino selects die for next roll q_{t+1} (state transition)
5. *t=t+1* and if *t<T*, goto 3

HMMs can be used to answer the following

1. Given observation sequence $\boldsymbol{o}=\{o_1,o_2,\ldots,o_T\}$, compute $p(\boldsymbol{o};\Theta)$ **[evaluation]**
2. Given observation sequence $\boldsymbol{o}=\{o_1,o_2,\ldots,o_T\}$, compute $p(\boldsymbol{q}|\boldsymbol{o};\Theta)$ **[decoding]**
3. Given a set of observation sequences $\boldsymbol{O}$, estimate Θ **[learning]**

Q1: evaluation

$$P(\mathbf{o} \mid \lambda) =$$

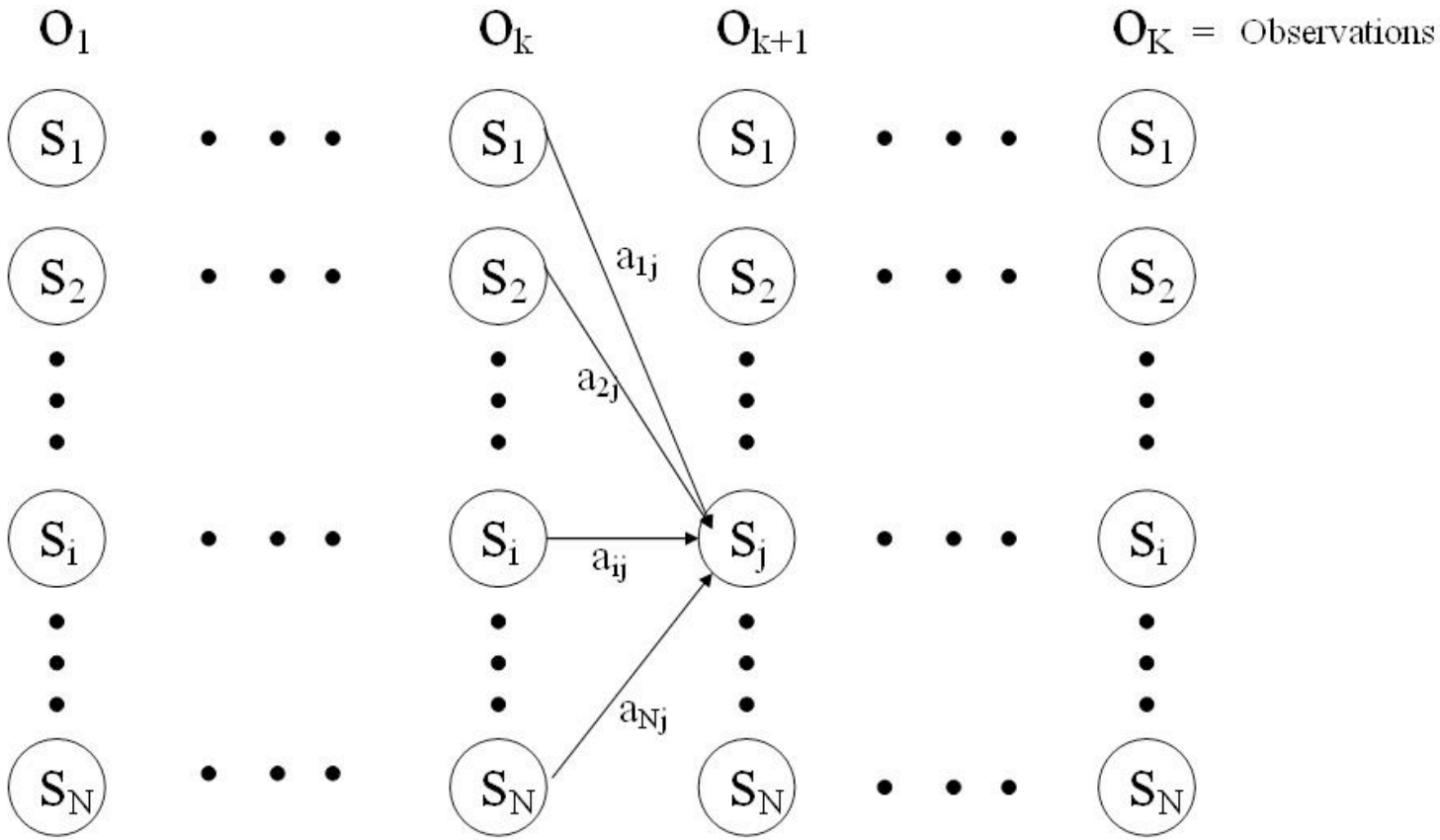
Q1: evaluation

Given observation sequence $\boldsymbol{o}=\{o_1,o_2,\ldots,o_T\}$, compute $p(\boldsymbol{o}; \boldsymbol{\lambda})$

$$\mathrm{P}(\mathbf{o} \mid \boldsymbol{\lambda}) = \sum_{\text{all Q}} \mathrm{P}(\mathbf{o} \mid \mathbf{q}, \boldsymbol{\lambda}) \mathrm{P}(\mathbf{q} \mid \boldsymbol{\lambda})$$

Unfortunately, this requires $O(N^T)$ calculations. Oh no!

Dynamic programming (the “forward algorithm”)



time T

Dyn  rogramming

1) **Initialization:**

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N.$$

2) **Induction:**

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \le t \le T-1$$

$$1 \le j \le N.$$

3) **Termination:**

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

We've been saved! *$O(N^2T)$!*

Q2: Decoding

Given observation sequence $\boldsymbol{o}=\{o_1,o_2,\ldots,o_T\}$, compute $p(\boldsymbol{q}|\boldsymbol{o}; \boldsymbol{\lambda})$

We are not guaranteed a unique solution

- Do we find most likely individual states q_i or the most likely (joint) sequence $\boldsymbol{q}$?
- Optimizing for the most likely state at a specific time can lead to impossible sequences under Θ
- Therefore, we will consider maximizing the joint probability of the sequence

Decoding the state sequence

We want to find $\boldsymbol{q}^*$

$$q^* \leftarrow \underset{q}{\operatorname{argmax}} P(q, o \mid \Theta)$$

In maximizing joint, enumerating all possible sequences is intractable

Oh no!

Viterbi algortihm

Viterbi algortihm

Define

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1\, q_2 \cdots q_t = i, O_1\, O_2 \cdots O_t | \lambda]$$

(the most likely sequence to time *t-1* that ends in state *i*). The corresponding inductive step is

$$\delta_{t+1}(j) = [\max_i \delta_t(i)\, a_{ij}] \cdot b_j(O_{t+1}).$$

Essentially, from the best path thus far, we find the most probable transition/emission probability

Q3: Learning

Given a set of observation sequences $\boldsymbol{O}$, estimate $\boldsymbol{\lambda}$

We can do this using an iterative estimation algorithm, i.e., an instance of **Expectation Maximization**

We will come back to this in a later lecture on learning more generally

For more, see

A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition

LAWRENCE R. RABINER, FELLOW, IEEE

Although initially introduced and studied in the late 1960s and early 1970s, statistical methods of Markov source or hidden Markov modeling have become increasingly popular in the last several years. There are two strong reasons why this has occurred. First the models are very rich in mathematical structure and hence can form the theoretical basis for use in a wide range of applications. Second the models, when applied properly, work very well in practice for several important applications. In this paper we attempt to carefully and methodically review the theoretical aspects of this type of statistical modeling and show how they have been applied to selected problems in machine recognition of speech.

I. Introduction

Real-world processes generally produce observable outputs which can be characterized as signals. The signals can

In this case, with a good signal model, we can simulate the source and learn as much as possible via simulations. Finally, the most important reason why signal models are important is that they often work extremely well in practice, and enable us to realize important practical systems—e.g., prediction systems, recognition systems, identification systems, etc., in a very efficient manner.

These are several possible choices for what type of signal model is used for characterizing the properties of a given signal. Broadly one can dichotomize the types of signal models into the class of deterministic models, and the class of statistical models. Deterministic models generally exploit some known specific properties of the signal, e.g., that the signal is a sine wave, or a sum of exponentials, etc. In these

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:

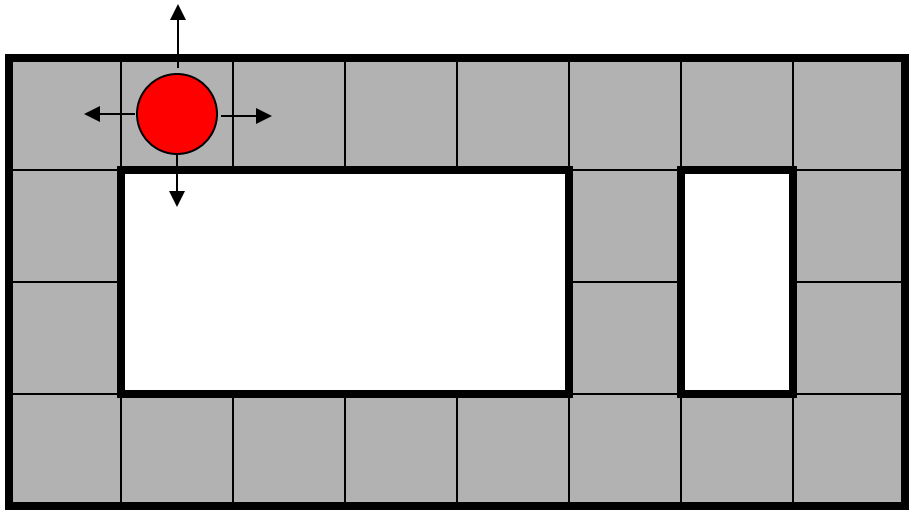
- Observations are range readings (continuous)
- States are positions on a map (continuous)

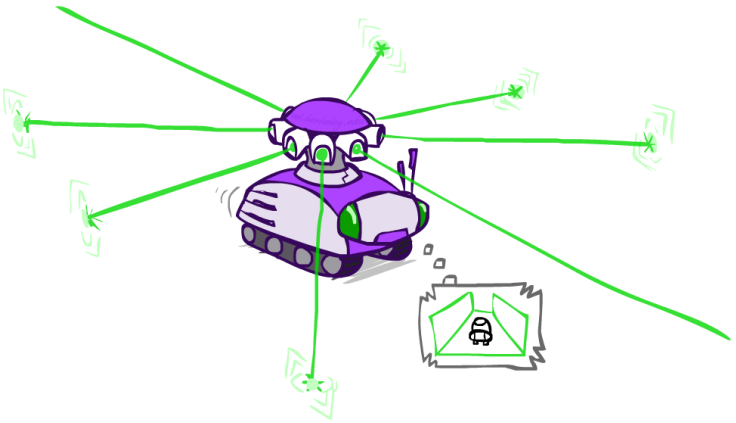
Filtering / monitoring

- 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



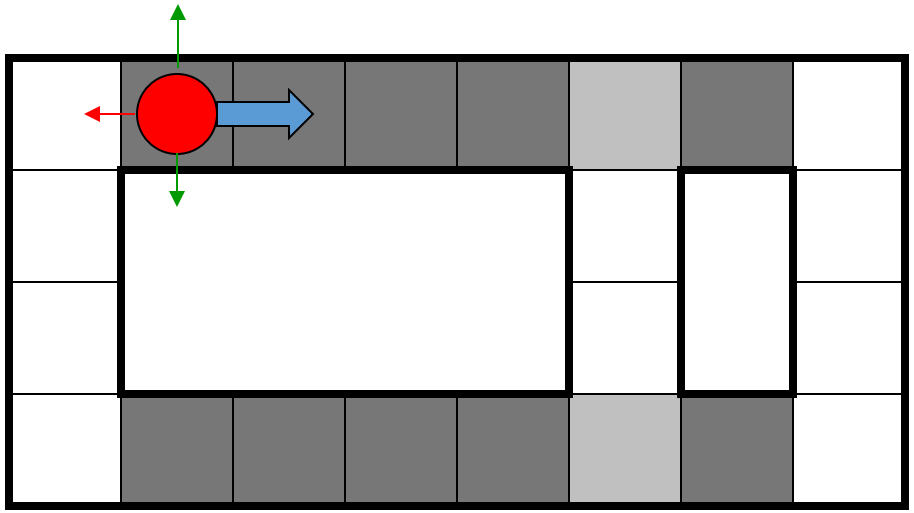


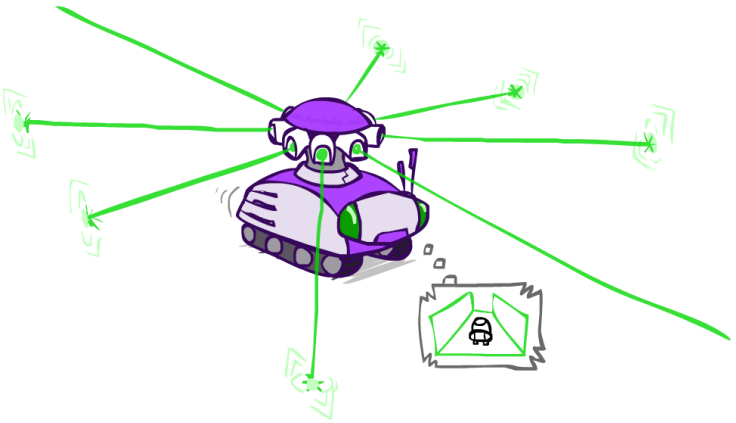
t=0

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



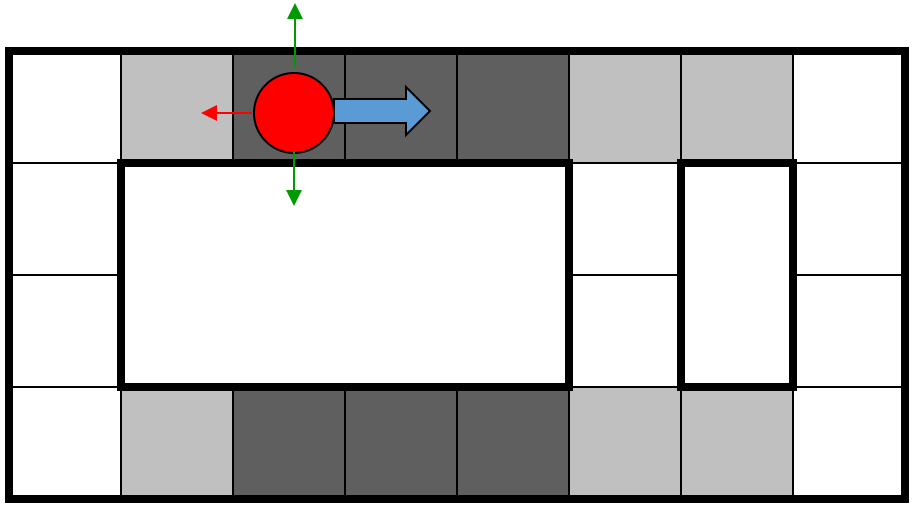


Prob 0 1

t=1

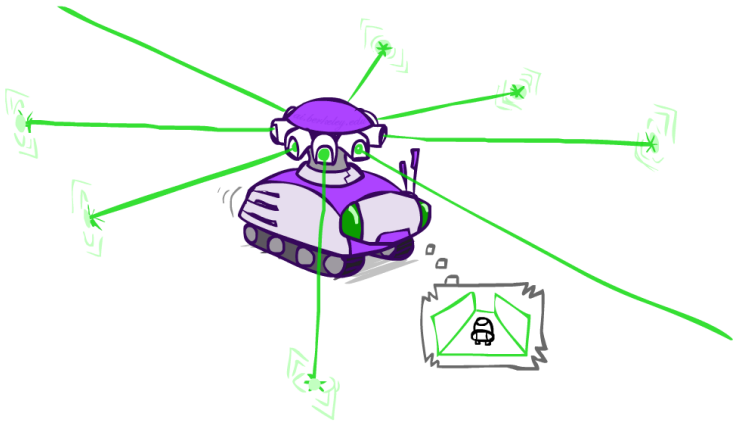
Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

Example: robot localization

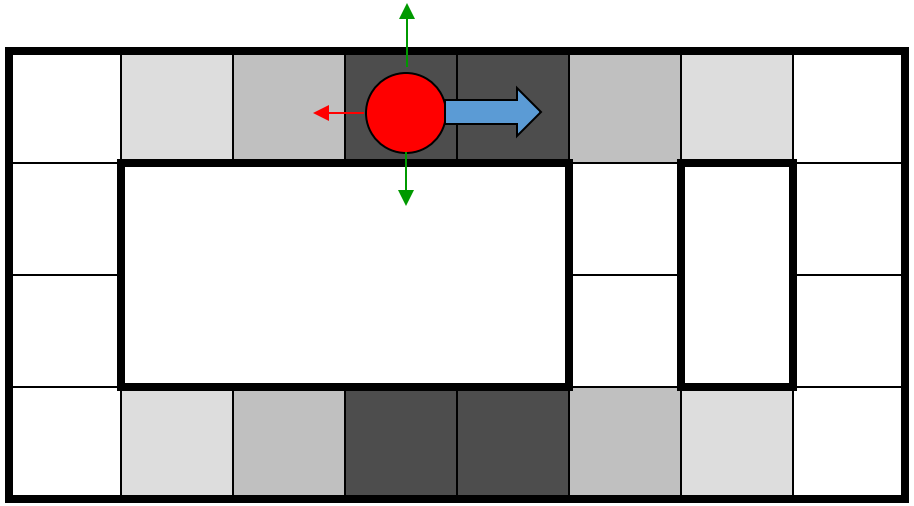


Prob 0 1

t=2

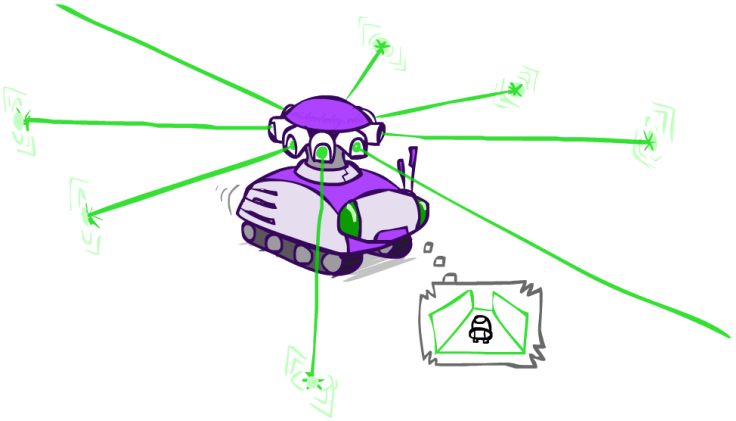


Example: robot localization

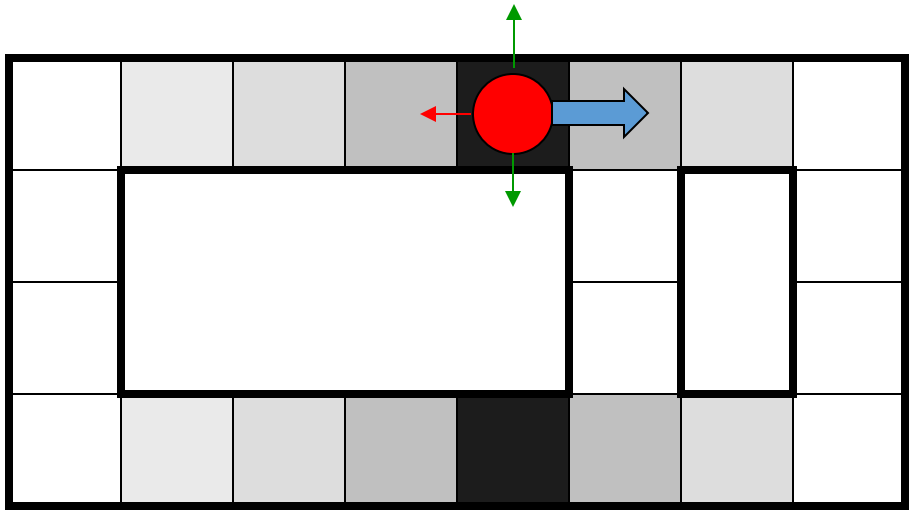


Prob 0 1

t=3

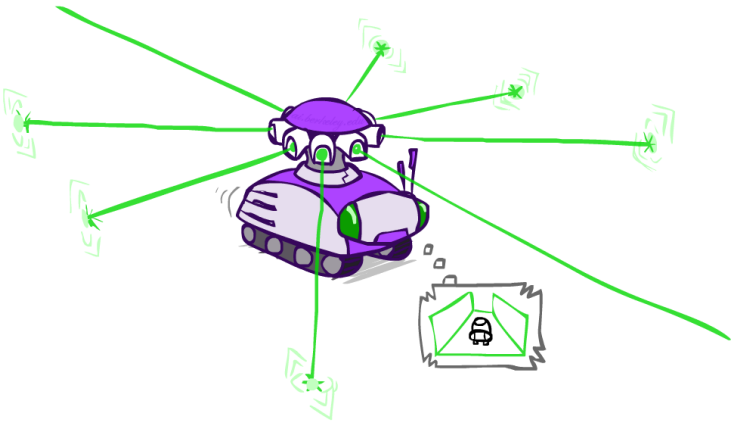


Example: robot localization

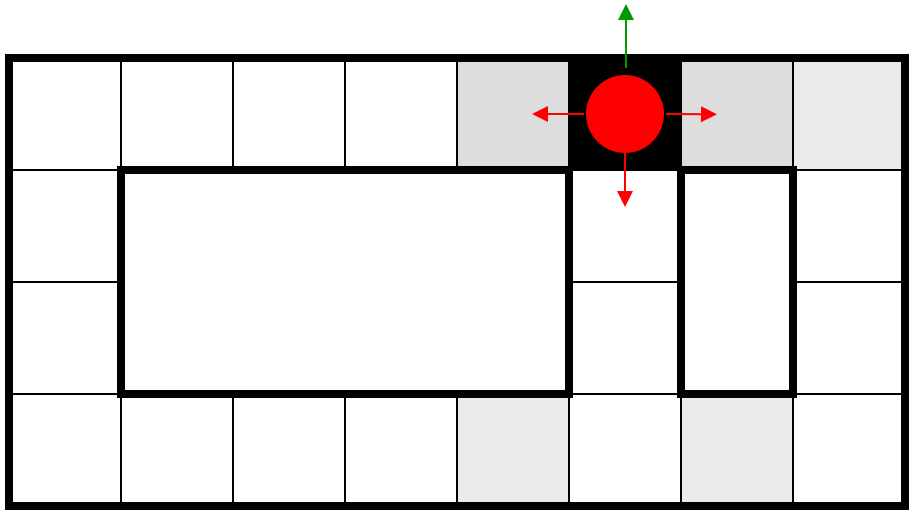


Prob 0 1

t=4

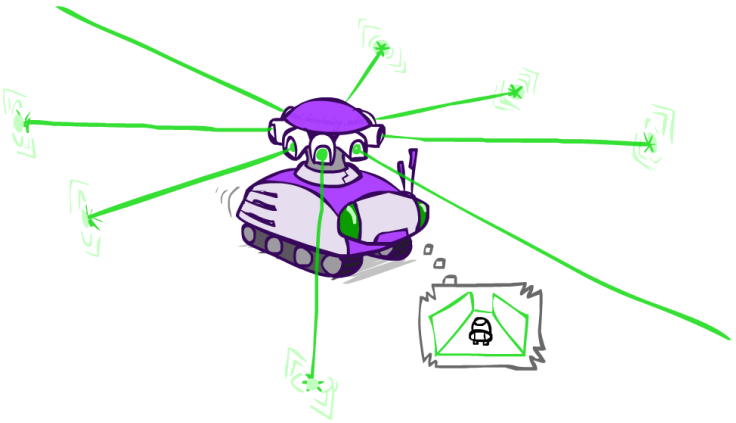


Example: robot localization



Prob 0 1

t=5



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

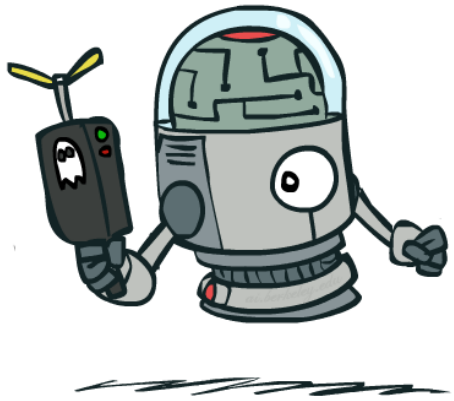
T = 1

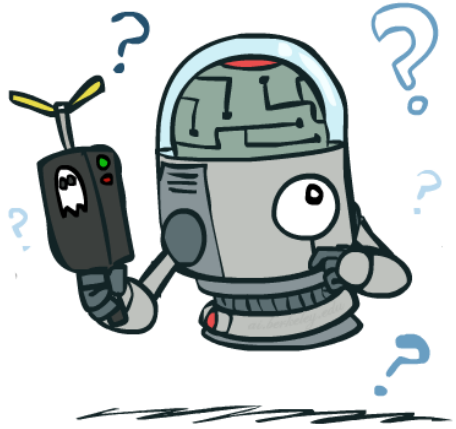
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

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





Example: observation

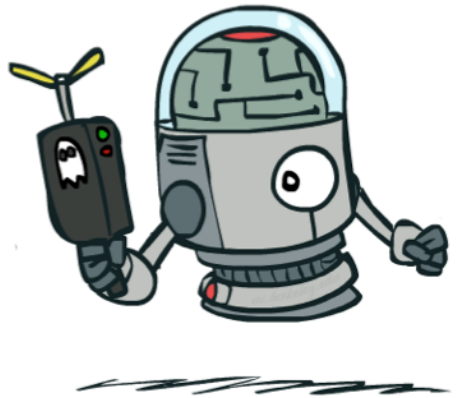
As we get observations, beliefs get reweighted, uncertainty “decreases”

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

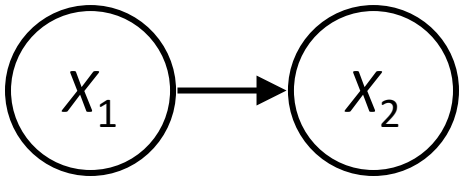
After observation



Online belief updates

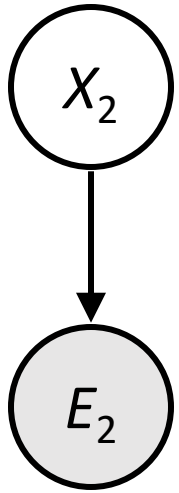
- Every time step, we start with current P(X | evidence)
- We update for time:

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



- We update for evidence:

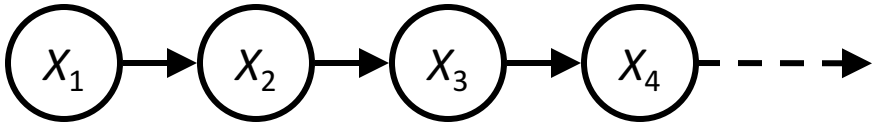
$$P(x_t|e_{1:t}) \propto_X P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$



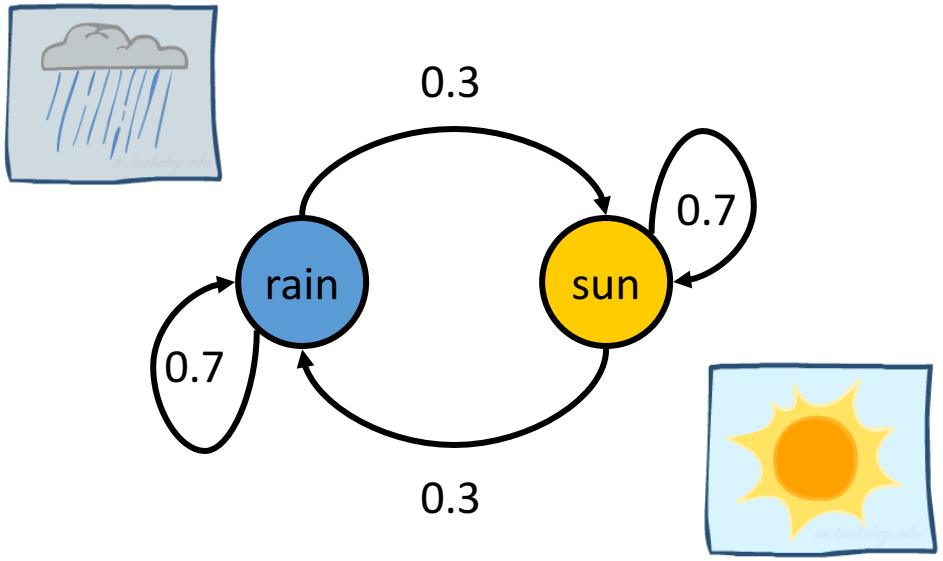
- The forward algorithm does both at once (and doesn't normalize)

Summary: reasoning over time

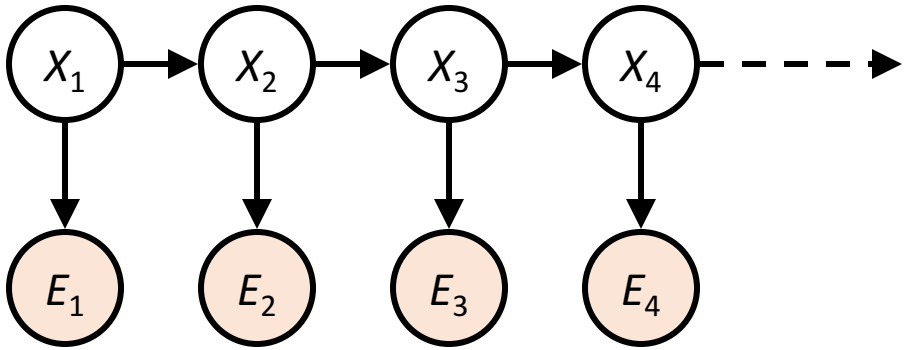
Markov models



$P(X_1)$ $P(X|X_{-1})$



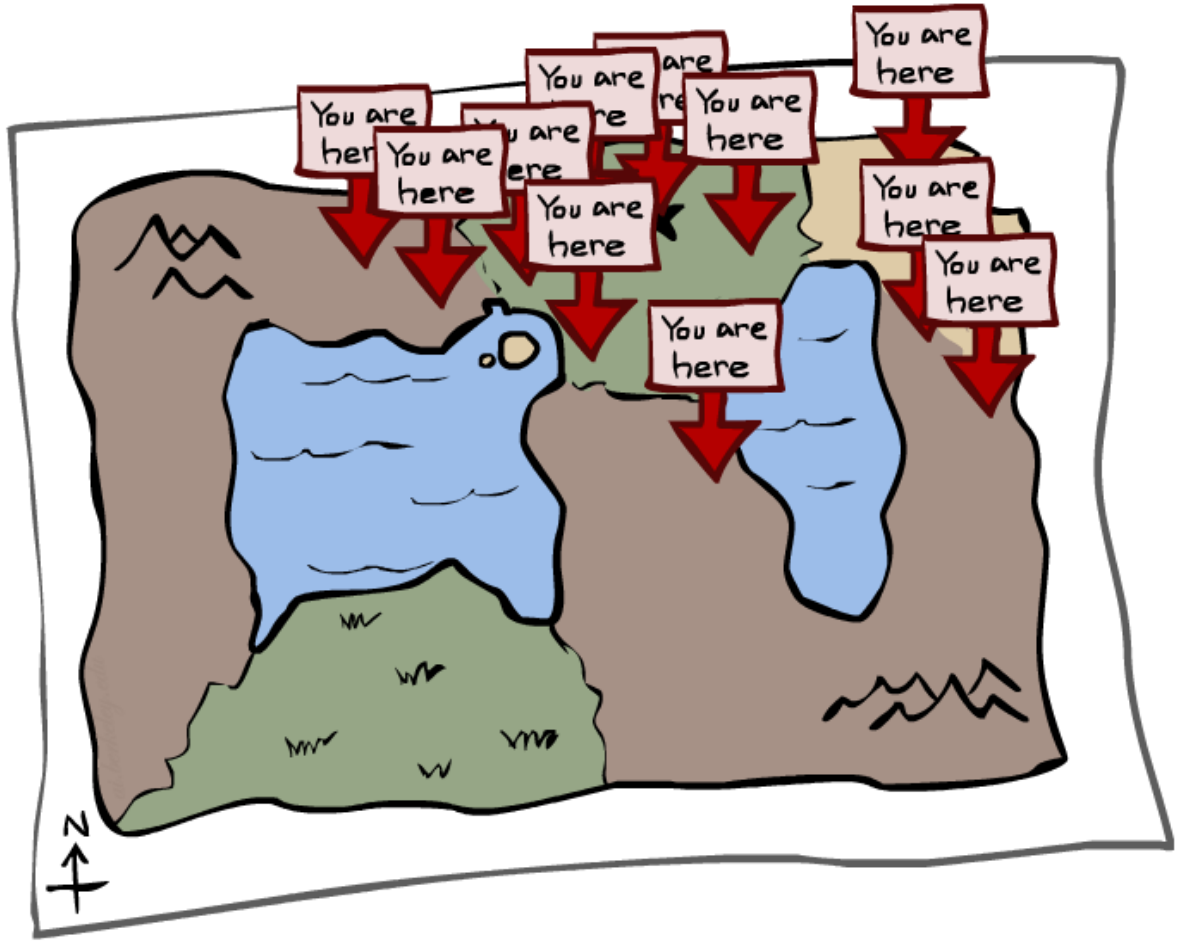
Hidden Markov models



$P(E|X)$

X	E	P
rain	umbrella	0.9
rain	no umbrella	0.1
sun	umbrella	0.2
sun	no umbrella	0.8

Particle filtering: a different way of tracking



Monte Carlo Localization: Efficient Position Estimation for Mobile Robots

Dieter Fox, Wolfram Burgard[†], Frank Dellaert, Sebastian Thrun

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

[†]Computer Science Department III
University of Bonn
Bonn, Germany

Particle filtering

Filtering: approximate solution

Sometimes |X| is too big to use exact inference

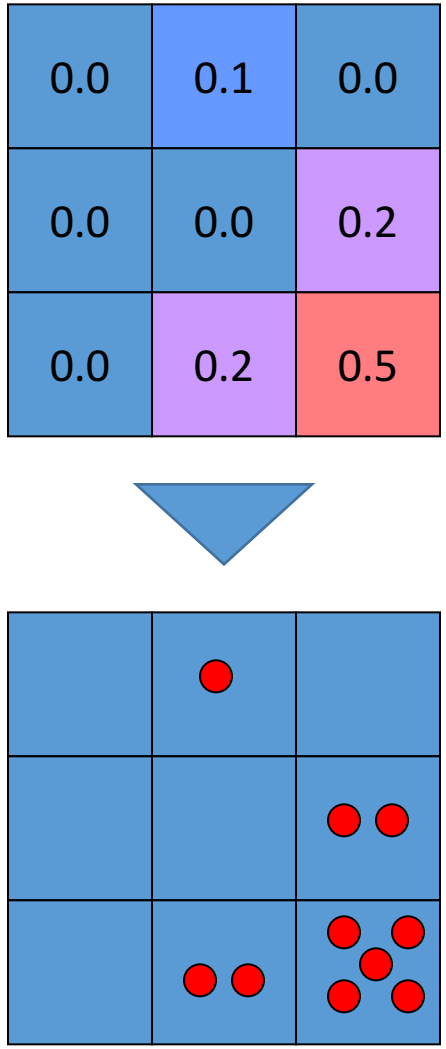
- |X| may be too big to even store B(X)
- E.g. X is continuous

Solution: approximate inference

- Track samples of X, not all values
- Samples are called particles
- Time per step is linear in the number of samples
- But: number needed may be large
- In memory: list of particles, not states

This is how robot localization works in practice

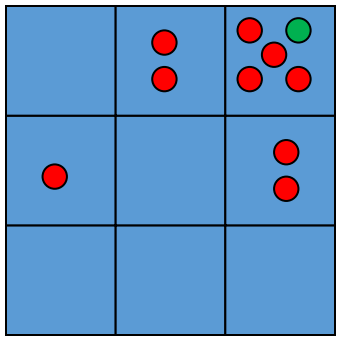
Particle is just new name for sample



Representation: particles

Our representation of P(X) is now a list of N particles (samples)

- Generally, N << |X|
- Storing map from X to counts would defeat the point



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

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: elapse time

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

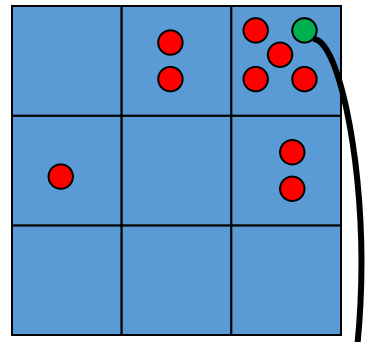
$$x' = \text{sample}(P(X'|x))$$

- 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

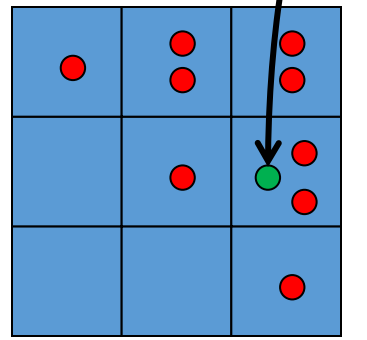
This captures the passage of time

- If enough samples, close to exact values before and after (consistent)

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: when we get evidence

Slightly trickier:

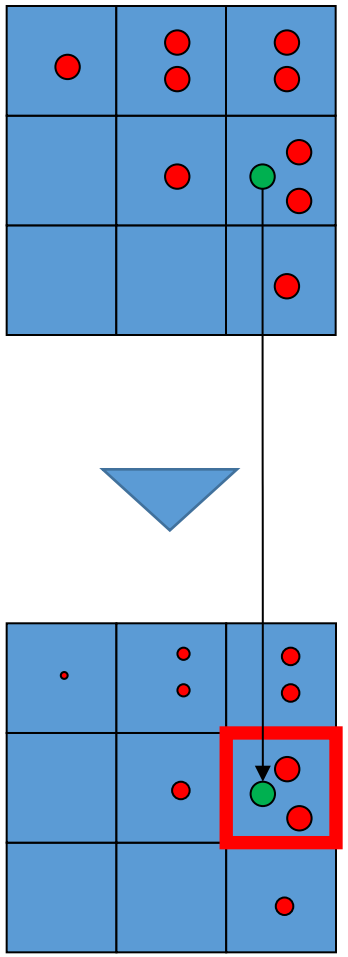
- Don't sample observation, fix it
- Similar to likelihood weighting, downweight samples based on the evidence

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

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

- 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: resampling

Rather than tracking weighted samples, we resample

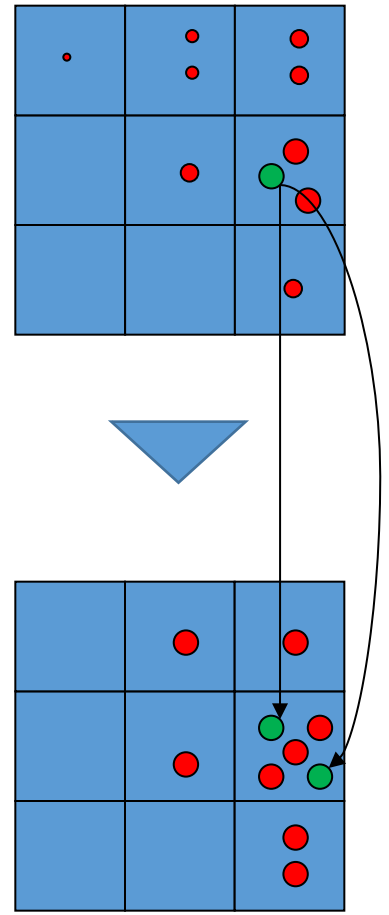
N times, we choose from our weighted sample distribution (i.e. draw with replacement)

This is equivalent to renormalizing the distribution

Now the update is complete for this time step, continue with the next one

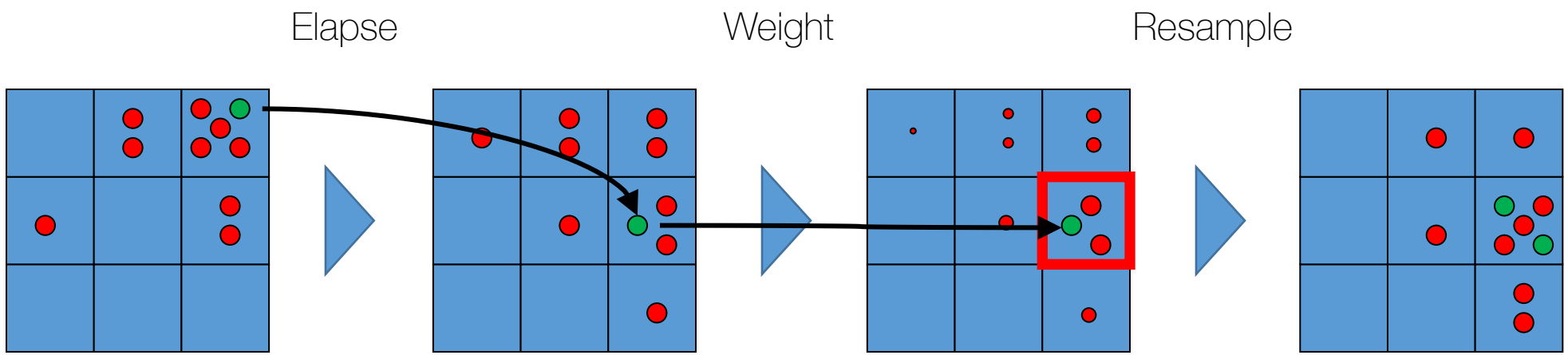
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(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)

Transition particles based on models

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

Observe evidence and re-weight to reflect this

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

Sample N new particles using weights

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

That’s it for today!

- Midterm next week!
- Homeworks due Sunday!
- Review session in class Tuesday: come with questions!