

# Markov Chain Monte Carlo

- MCMC (Markov chain Monte Carlo) is a family of randomized algorithms for approximating some quantity of interest over a very large state space
  - Markov chain = a sequence of randomly chosen states (“random walk”), where each state is chosen conditioned on the previous state
  - Monte Carlo = an algorithm (usually based on sampling) that has some probability of producing an incorrect answer
- MCMC = wander around for a bit, average what you see

# Gibbs sampling

- A particular kind of MCMC

- States are complete assignments to all variables

- (Cf local search: closely related to simulated annealing!)

- Evidence variables remain fixed, other variables change

- To generate the next state, pick a variable and sample a value for it conditioned on all the other variables:  $X_i' \sim P(X_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$

- Will tend to move towards states of higher probability, but can go down too

- In a Bayes net,  $P(X_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(X_i \mid \text{markov\_blanket}(X_i))$

- Theorem: Gibbs sampling is consistent\*

- Provided all Gibbs distributions are bounded away from 0 and 1 and variable selection is fair

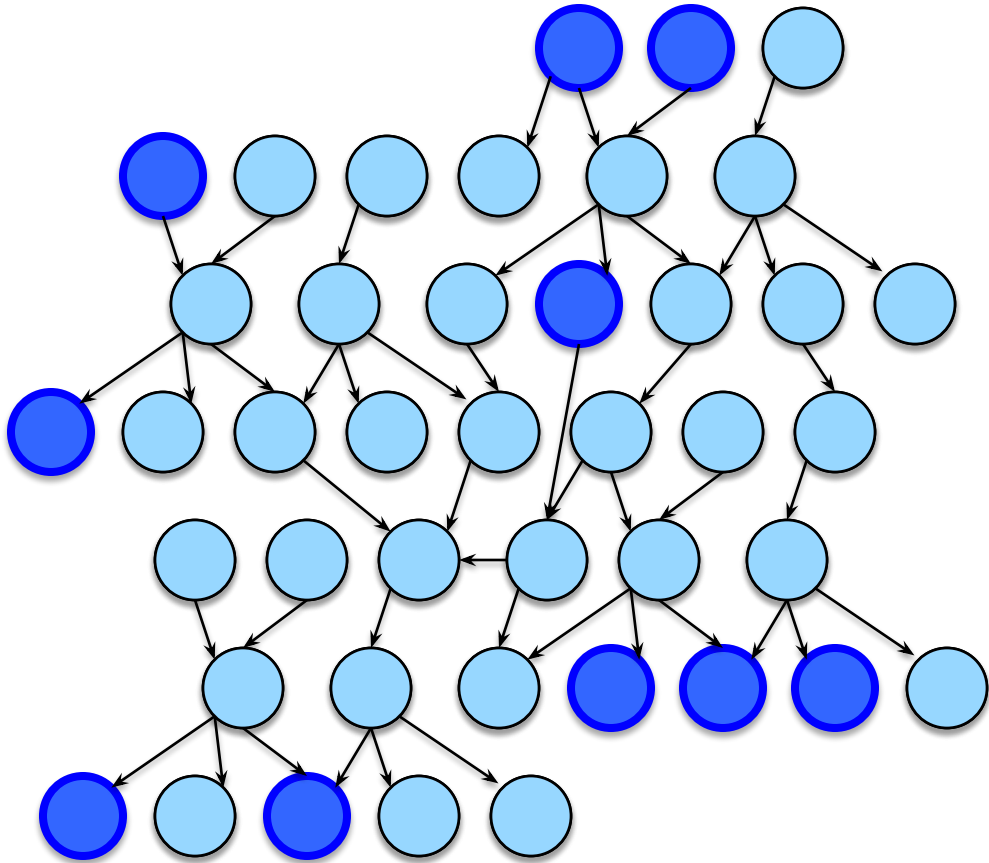
Markov blanket =

Parents +

Children +

Children's other parents

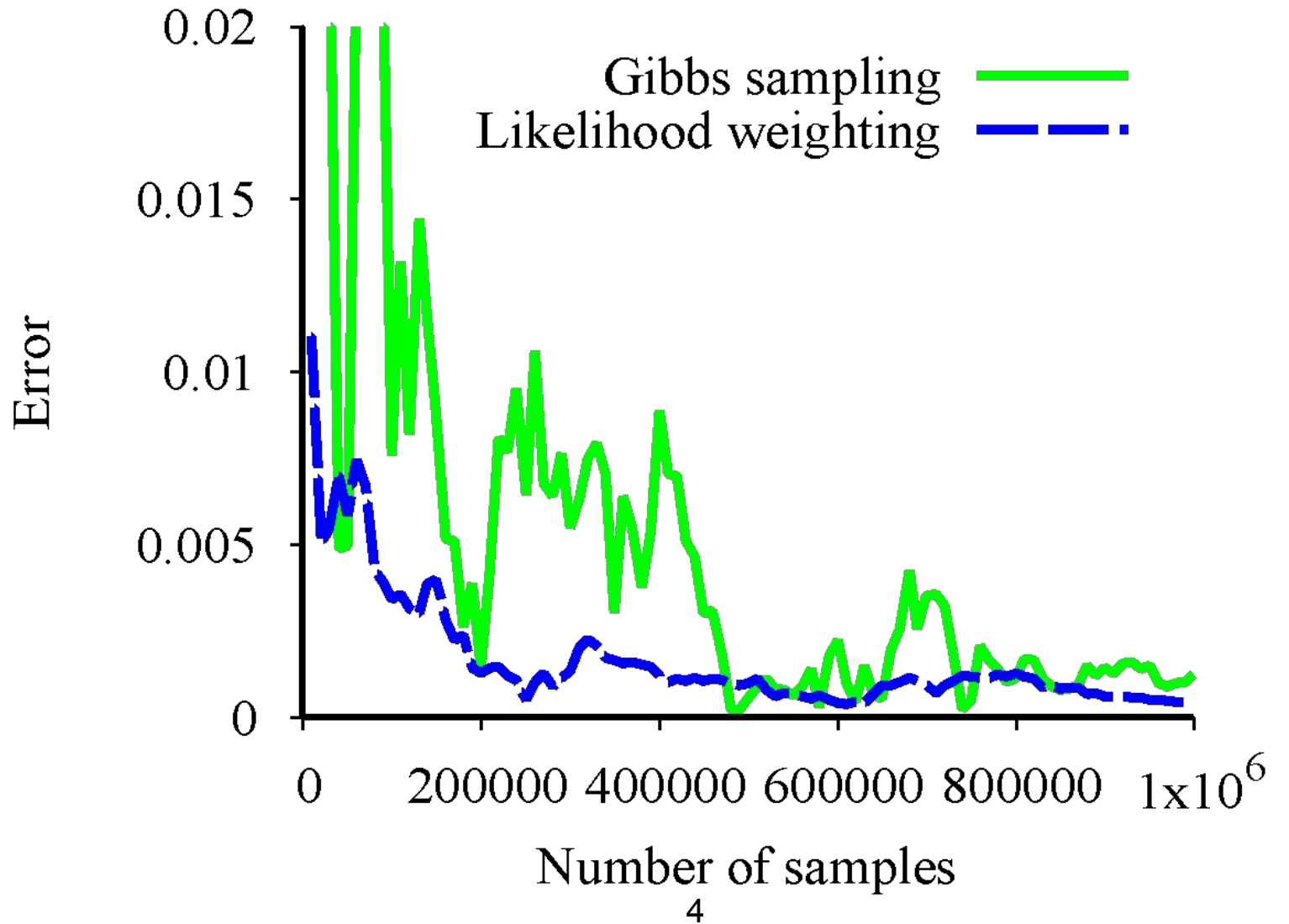
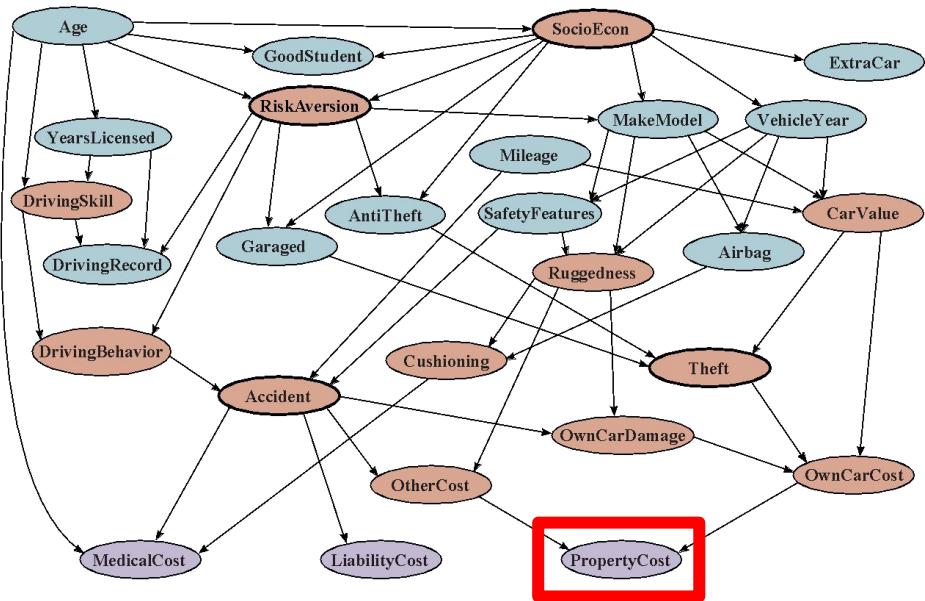
# Advantages of MCMC



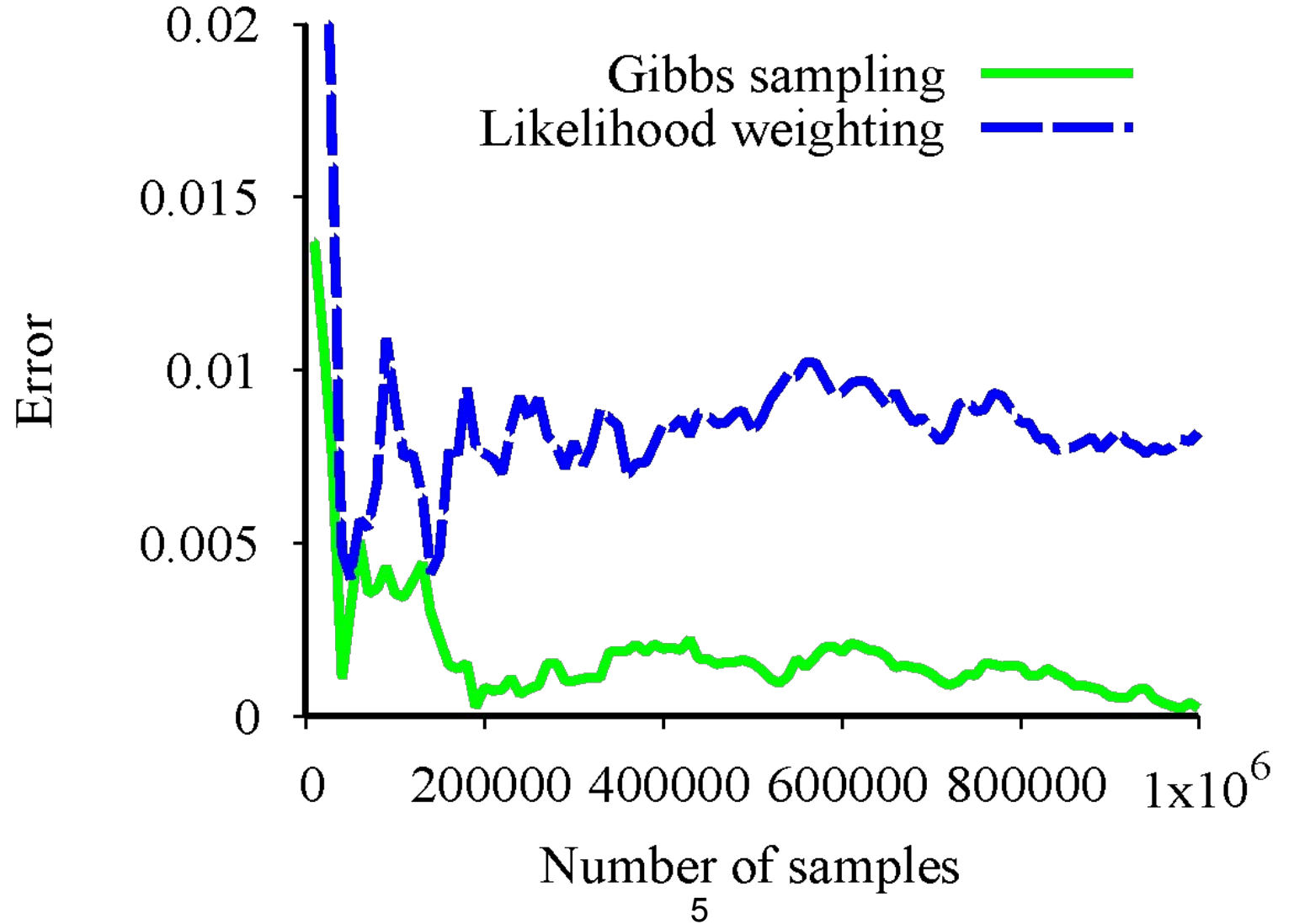
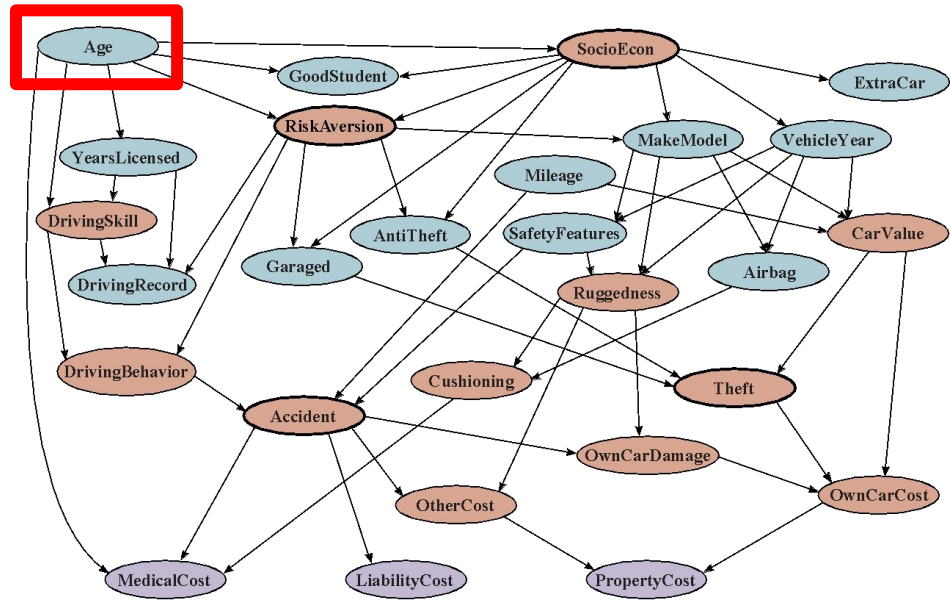
Samples soon begin to reflect all the evidence in the network

Eventually they are being drawn from the true posterior!

# Car Insurance: $P(\text{PropertyCost} \mid e)$



# Car Insurance: $P(\text{Age} \mid e)$

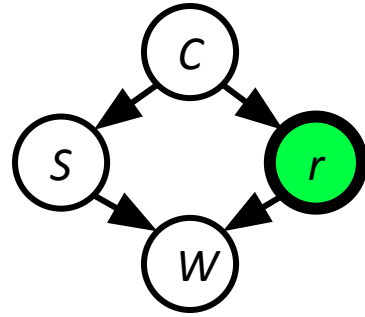




# Gibbs Sampling Example: $P(S | r)$

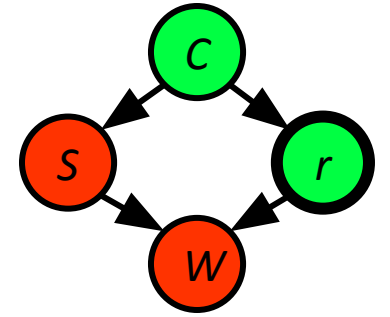
- Step 1: Fix evidence

- $R = \text{true}$



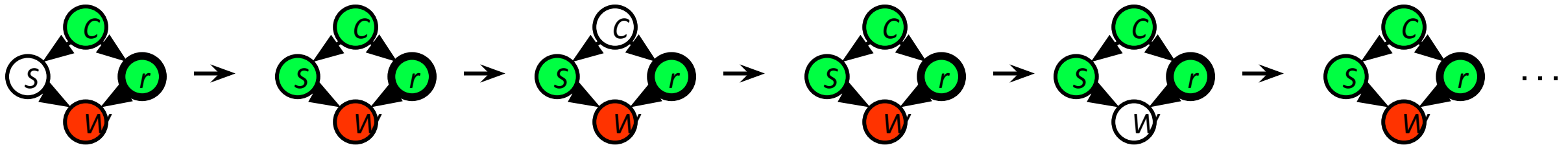
- Step 2: Initialize other variables

- Randomly



- Step 3: Repeat

- Choose a non-evidence variable  $X$
- Resample  $X$  from  $P(X | \text{markov\_blanket}(X))$

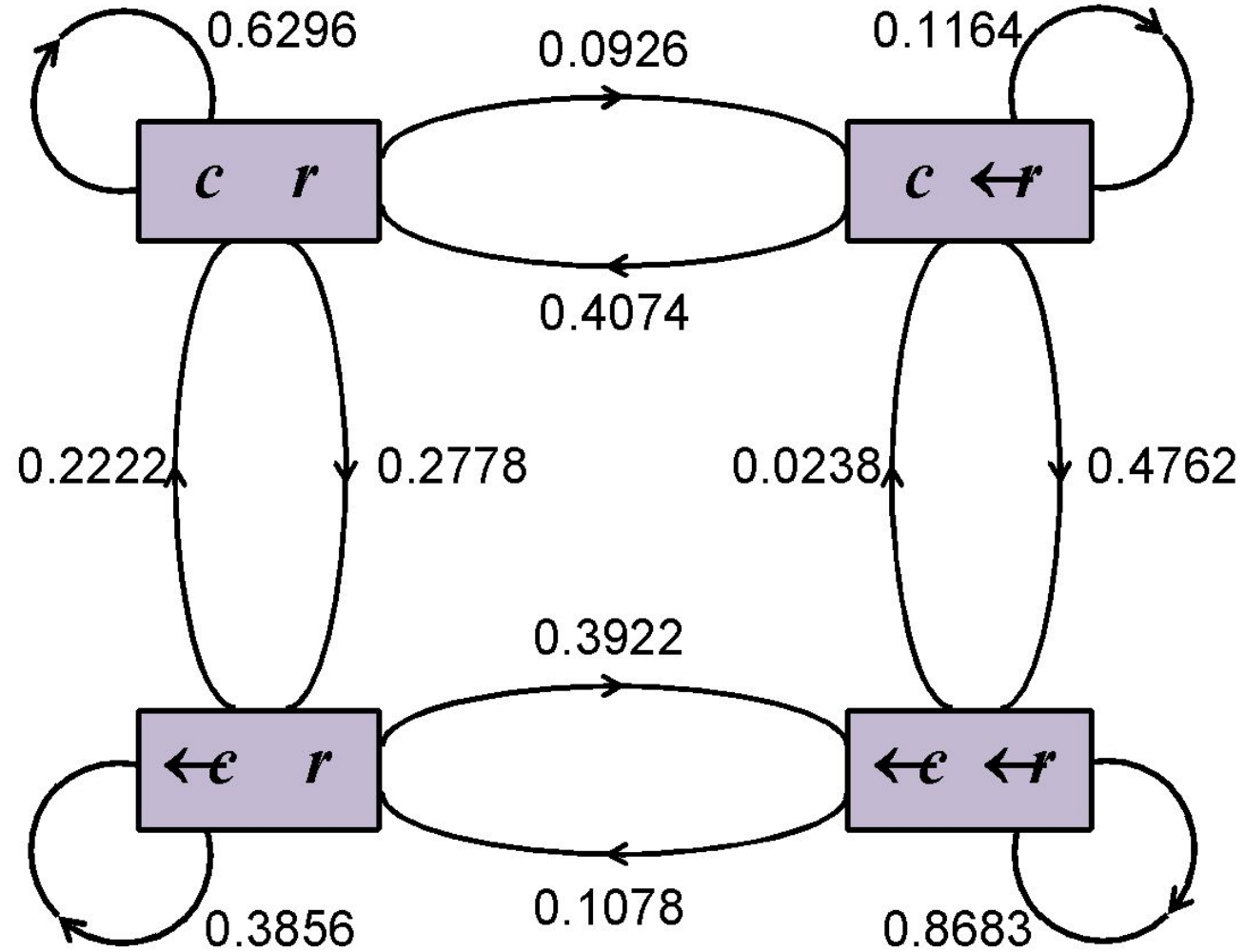


Sample  $S \sim P(S | c, r, \neg w)$

Sample  $C \sim P(C | s, r)$

Sample  $W \sim P(W | s, r)$

# Markov chain given $s, w$



# Gibbs sampling and MCMC in practice

---

- The most commonly used method for large Bayes nets
  - See, e.g., BUGS, JAGS, STAN, infer.net, BLOG, etc.
- Can be compiled to run very fast
  - Eliminate all data structure references, just multiply and sample
  - ~100 million samples per second on a laptop
- Can run asynchronously in parallel (one processor per variable)
- Many cognitive scientists suggest the brain runs on MCMC

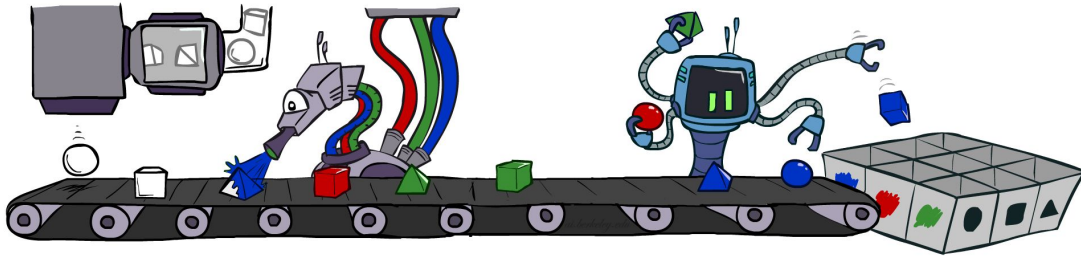
# Consistency of Gibbs (see AIMA 13.4.2 for details)

- Suppose we run it for a long time and predict the probability of reaching any given state at time  $t$ :  $\pi_t(x_1, \dots, x_n)$  or  $\pi_t(\underline{x})$
- Each Gibbs sampling step (pick a variable, resample its value) applied to a state  $\underline{x}$  has a probability  $k(\underline{x}' | \underline{x})$  of reaching a next state  $\underline{x}'$
- So  $\pi_{t+1}(\underline{x}') = \sum_{\underline{x}} k(\underline{x}' | \underline{x}) \pi_t(\underline{x})$  or, in matrix/vector form  $\pi_{t+1} = \mathbf{K}\pi_t$
- When the process is in equilibrium  $\pi_{t+1} = \pi_t = \pi$  so  $\mathbf{K}\pi = \pi$
- This has a unique\* solution  $\pi = P(x_1, \dots, x_n | e_1, \dots, e_k)$ 
  - \* Markov chain must be *ergodic*, i.e., completely connected and aperiodic
  - Satisfied if all probabilities are bounded away from 0 and 1
- So for large enough  $t$  the next sample will be drawn from the true posterior
  - “Large enough” depends on CPTs in the Bayes net; takes *longer* if nearly deterministic

# Bayes Net Sampling Summary

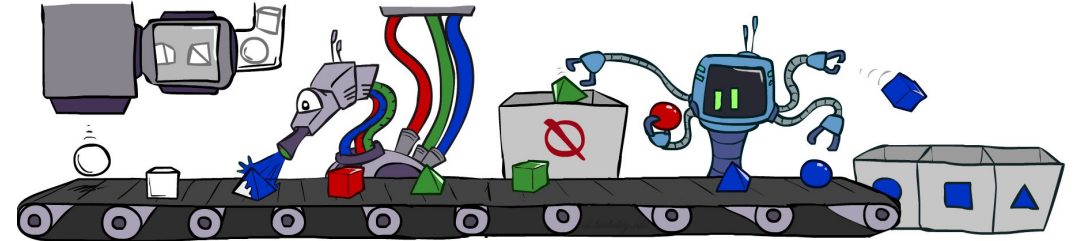
- Prior Sampling  $P$  :

- Generate complete samples from  $P(x_1, \dots, x_n)$



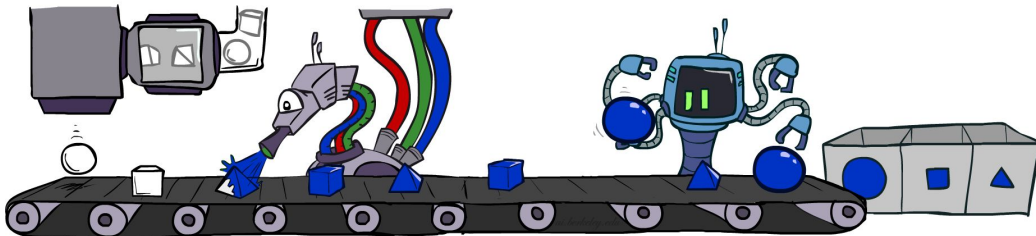
- Rejection Sampling  $P(Q | e)$  :

- Reject samples that don't match  $e$



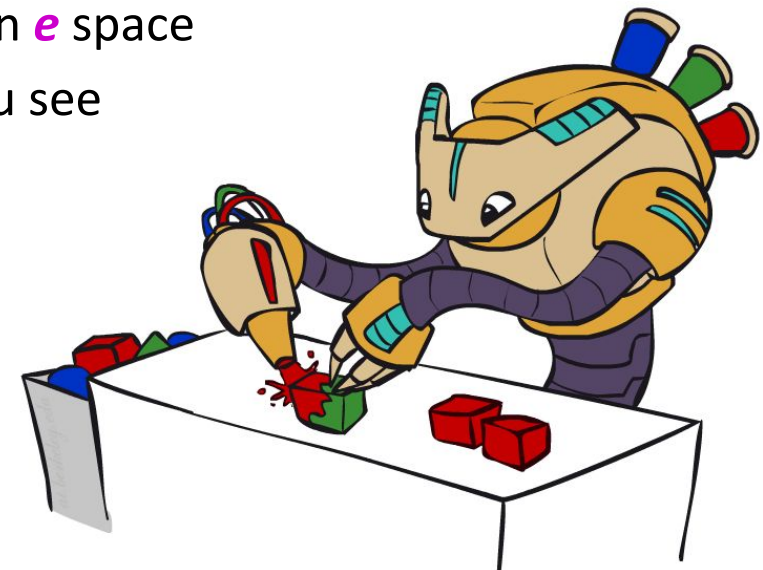
- Likelihood Weighting  $P(Q | e)$  :

- Weight samples by how well they predict  $e$



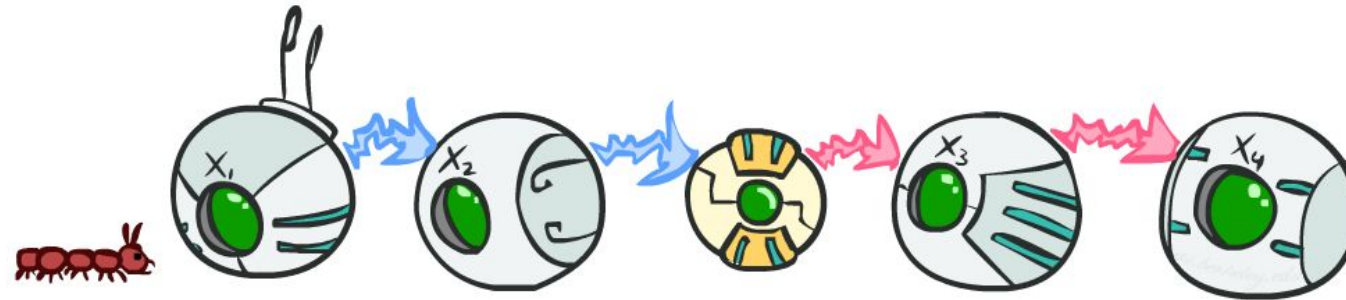
- Gibbs sampling  $P(Q | e)$  :

- Wander around in  $e$  space
- Average what you see



# CS 188: Artificial Intelligence

## Markov Models



Instructors: Stuart Russell and Peyrin Kao

University of California, Berkeley

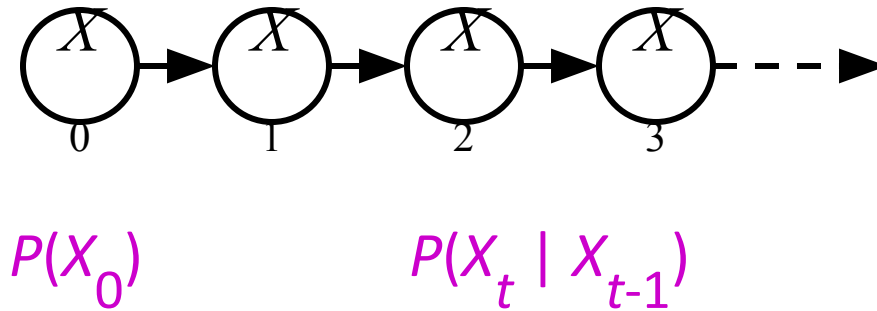
# Uncertainty and Time

---

- Often, we want to reason about a *sequence* of observations where the state of the underlying system is *changing*
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring
  - Global climate
- Need to introduce time into our models

# Markov Models (aka Markov chain/process)

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



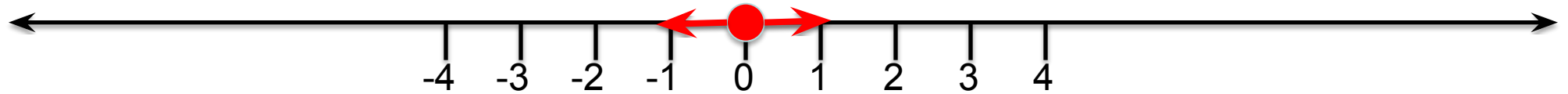
- The **transition model**  $P(X_t | X_{t-1})$  specifies how the state evolves over time
- Stationarity** assumption: transition probabilities are the same at all times
- Markov** assumption: “future is independent of the past given the present”
  - $X_{t+1}$  is independent of  $X_0, \dots, X_{t-1}$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

# 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

# Example: Random walk in one dimension



- State: location on the unbounded integer line
- Initial probability: starts at 0
- Transition model:  $P(X_t = k | X_{t-1} = k \pm 1) = 0.5$
- 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

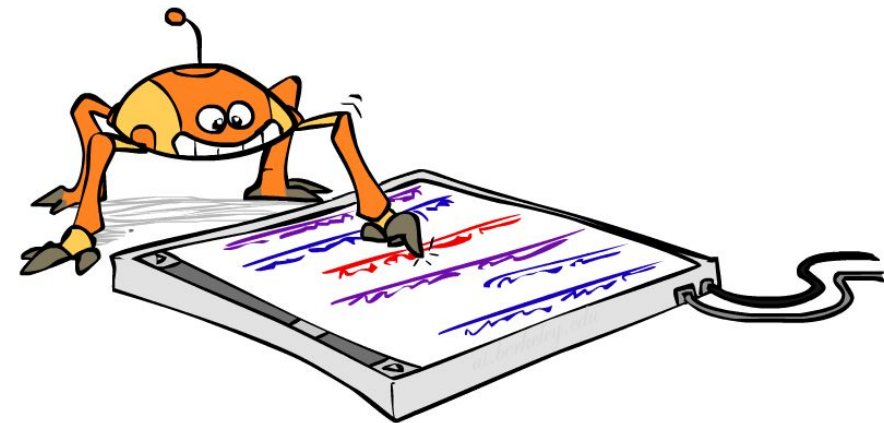
# Example: n-gram models

We call ourselves *Homo sapiens*—man the wise—because our **intelligence** is so important to us. For thousands of years, we have tried to understand *how we think*; that is, how a mere handful of matter can perceive, understand, predict, and manipulate a world far larger and more complicated than itself. ....

- State: word at position  $t$  in text (can also build letter n-grams)
- Transition model (probabilities come from empirical frequencies):
  - Unigram (zero-order):  $P(\text{Word}_t = i)$ 
    - “logical are as are confusion a may right tries agent goal the was . . .”
  - Bigram (first-order):  $P(\text{Word}_t = i \mid \text{Word}_{t-1} = j)$ 
    - “systems are very similar computational approach would be represented . . .”
  - Trigram (second-order):  $P(\text{Word}_t = i \mid \text{Word}_{t-1} = j, \text{Word}_{t-2} = k)$ 
    - “planning and scheduling are integrated the success of naive bayes model is . . .”
- Applications: text classification, spam detection, author identification, typeahead, language classification, speech recognition, LLMs

# 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

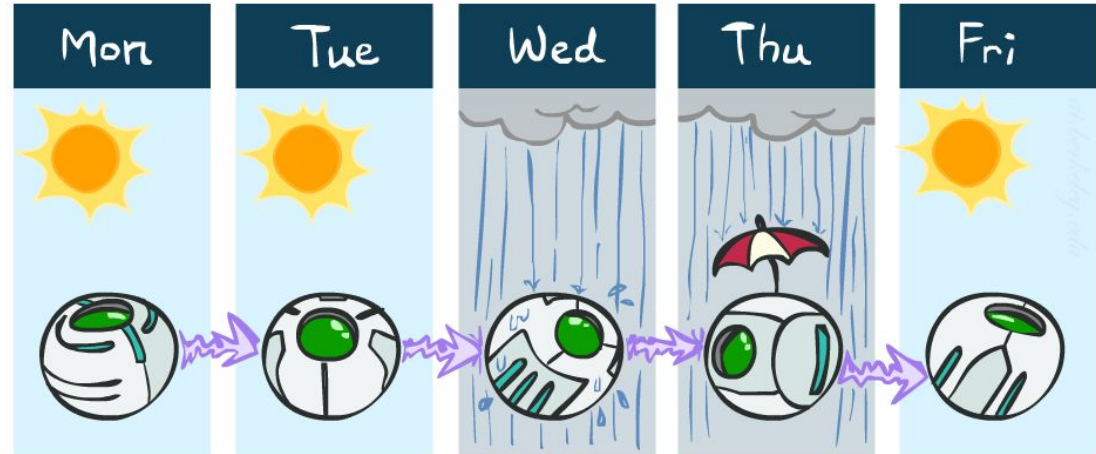


# Example: Weather

- States {rain, sun}

- Initial distribution  $P(X_0)$

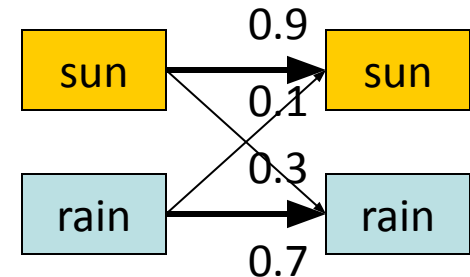
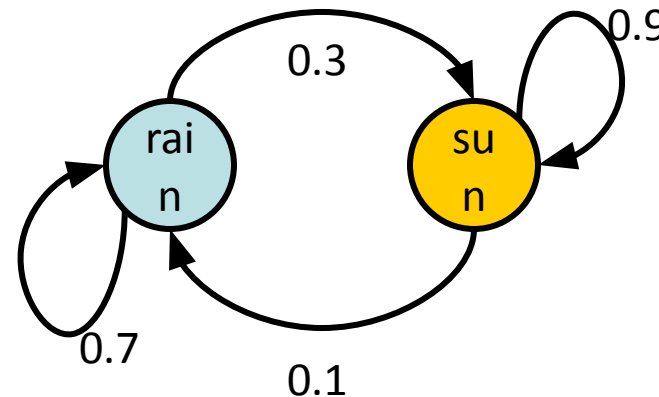
| $P(X_0)$ |      |
|----------|------|
| sun      | rain |
| 0.5      | 0.5  |



Two new ways of representing the same CPT

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

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



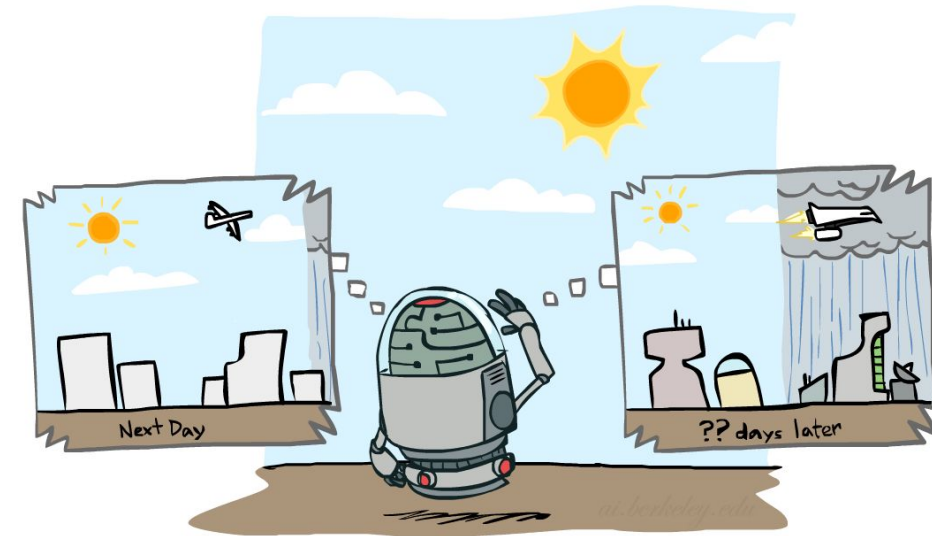
# Weather prediction

- Time 0:  $\langle 0.5, 0.5 \rangle$

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

- What is the weather like at time 1?

- $P(X_1) = \sum_{x_0} P(X_1, X_0=x_0)$
- $= \sum_{x_0} P(X_0=x_0) P(X_1 | X_0=x_0)$
- $= 0.5 \langle 0.9, 0.1 \rangle + 0.5 \langle 0.3, 0.7 \rangle = \langle 0.6, 0.4 \rangle$



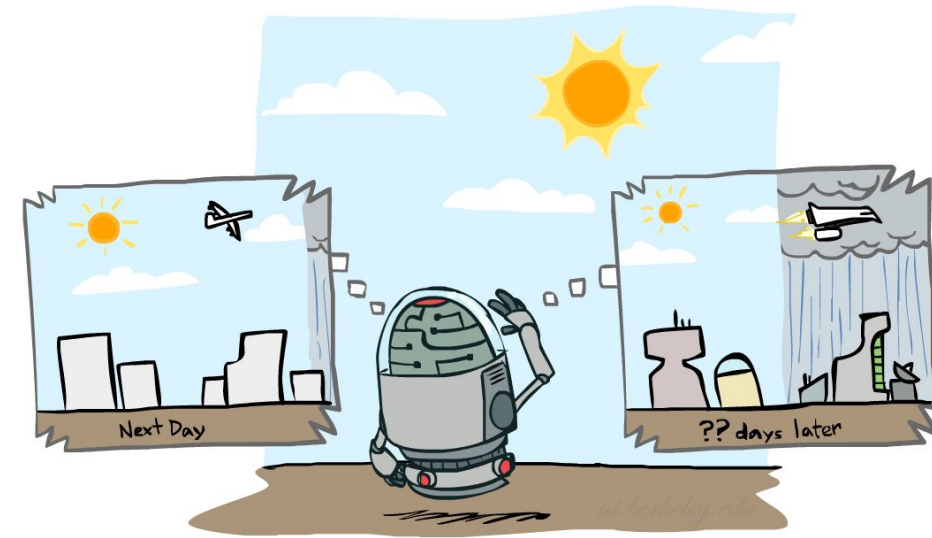
# Weather prediction, contd.

- Time 1:  $\langle 0.6, 0.4 \rangle$

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

- What is the weather like at time 2?

- $P(X_2) = \sum_{x_1} P(X_2, X_1=x_1)$
- $= \sum_{x_1} P(X_1=x_1) P(X_2 | X_1=x_1)$
- $= 0.6 \langle 0.9, 0.1 \rangle + 0.4 \langle 0.3, 0.7 \rangle = \langle 0.66, 0.34 \rangle$



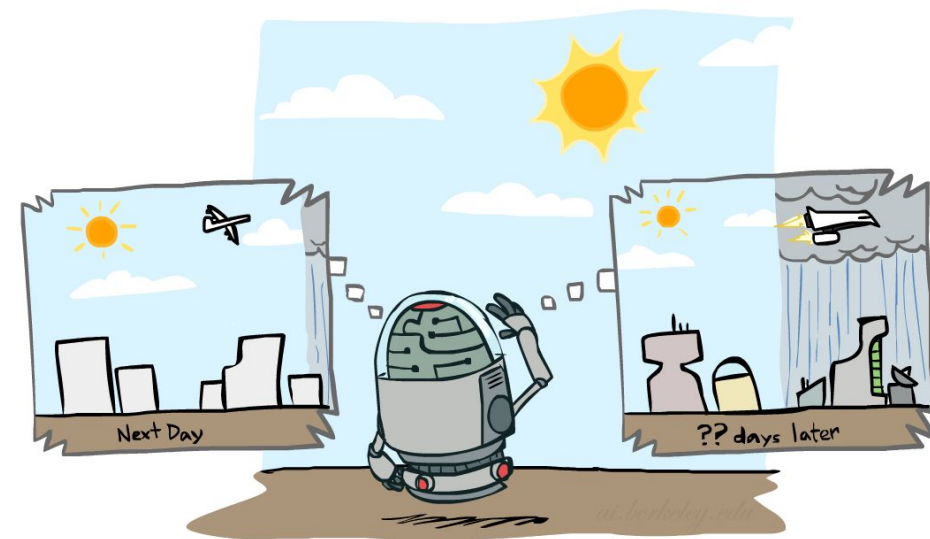
# Weather prediction, contd.

- Time 2:  $\langle 0.66, 0.34 \rangle$

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

- What is the weather like at time 3?

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



# Forward algorithm (simple form)

- What is the state at time  $t$ 
  - $P(X_t) = \sum_{x^{t-1}} P(X_t, X_{t-1}=x_{t-1})$
  - $= \sum_{x^{t-1}} P(X_{t-1}=x_{t-1}) P(X_t | X_{t-1}=x_{t-1})$
- Iterate this update starting at  $t=0$ 
  - This is called a **recursive** update:  $P_t = g(P_{t-1}) = g(g(g(g( \dots P_0))))$

Probability from previous iteration

Transition model

# And the same thing in linear algebra

- What is the weather like at time 2?
  - $P(X_2) = 0.6\langle 0.9, 0.1 \rangle + 0.4\langle 0.3, 0.7 \rangle = \langle 0.66, 0.34 \rangle$
- In matrix-vector form:
  - $P(X_2) = \begin{pmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0.66 \\ 0.34 \end{pmatrix}$
- I.e., multiply by  $T^T$ , transpose of transition matrix

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

# Stationary Distributions

- The limiting distribution is called the **stationary distribution**  $P_\infty$  of the chain
- It satisfies  $P_\infty = P_{\infty+1} = T^T P_\infty$
- Solving for  $P_\infty$  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  $\langle 0.75, 0.25 \rangle$  **regardless of starting distribution**



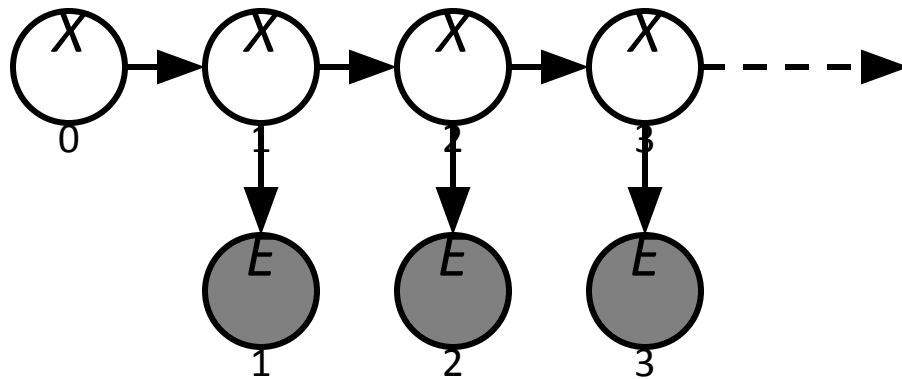
# Hidden Markov Models

---



# Hidden Markov Models

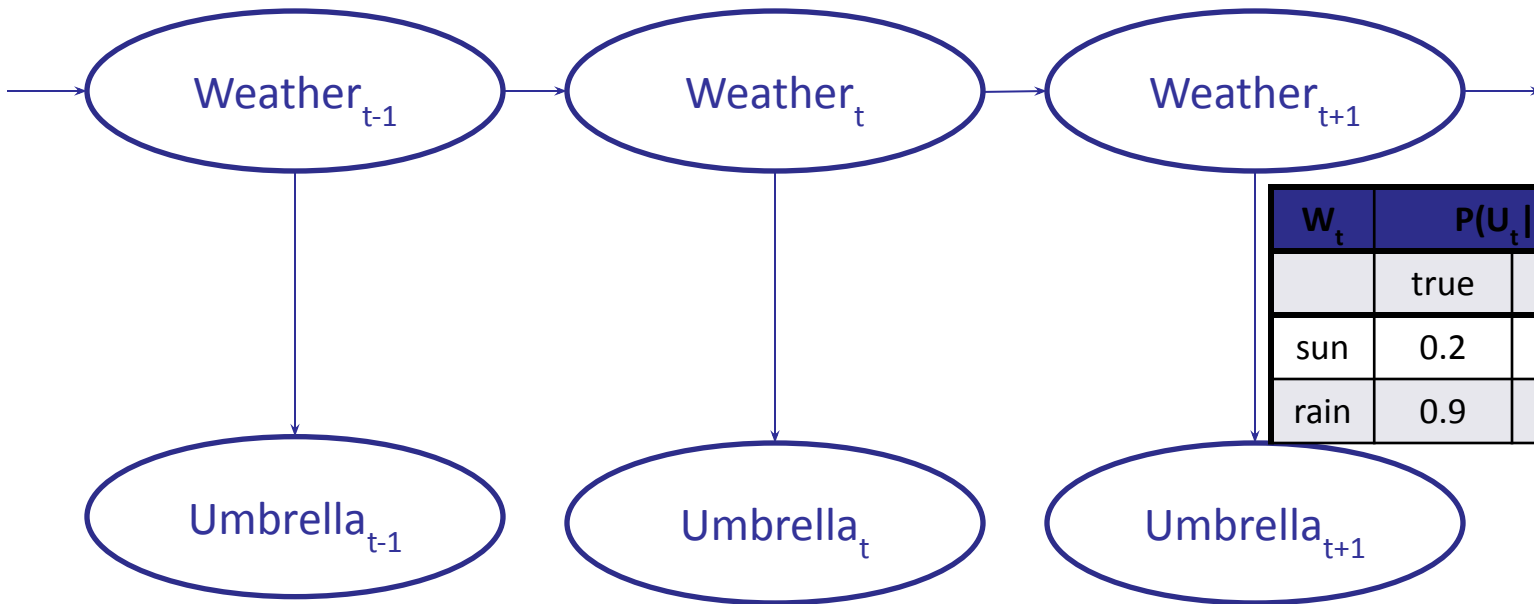
- Usually the true state is not observed directly
- Hidden Markov models (HMMs)
  - Underlying Markov chain over states  $X$
  - You observe evidence  $E$  at each time step
  - $X_t$  is a single discrete variable;  $E_t$  may be continuous and may consist of several variables



# 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  |

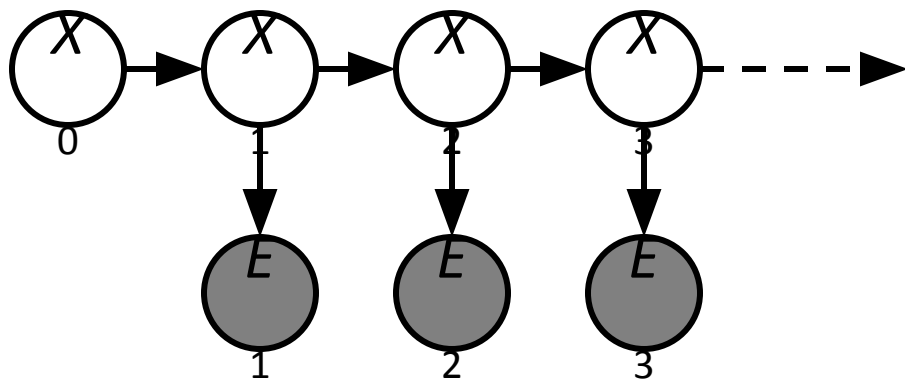


| $W_t$ | $P(U_t   W_t)$ |       |
|-------|----------------|-------|
|       | true           | false |
| sun   | 0.2            | 0.8   |
| rain  | 0.9            | 0.1   |



# HMM as probability model

- Joint distribution for Markov model:  $P(X_0, \dots, X_T) = P(X_0) \prod_{t=1:T} P(X_t | X_{t-1})$
- Joint distribution for hidden Markov model:  
 $P(X_0, X_1, \dots, X_T, E_T) = P(X_0) \prod_{t=1:T} P(X_t | X_{t-1}) P(E_t | X_t)$
- Future states are independent of the past given the present
- Current evidence is independent of everything else given the current state
- Are evidence variables independent of each other?



Useful notation:

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

# Real HMM Examples

---

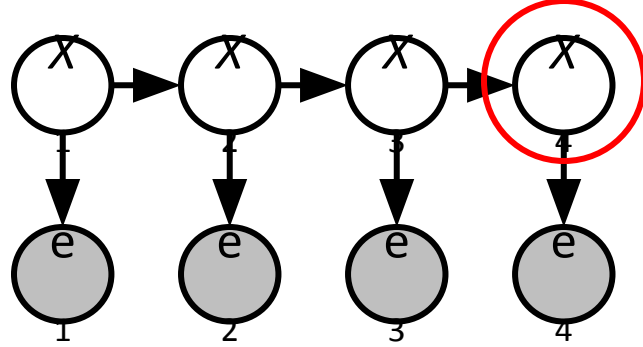
- **Speech recognition HMMs:**
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- **Machine translation HMMs:**
  - Observations are words (tens of thousands)
  - States are translation options
- **Robot tracking:**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

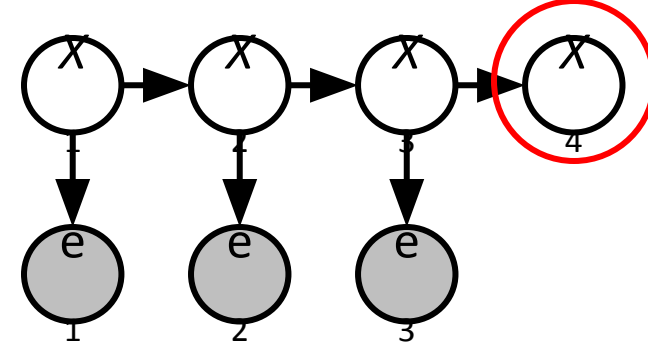
- **Filtering**:  $P(X_t | e_{1:t})$ 
  - **belief state**—input to the decision process of a rational agent
- **Prediction**:  $P(X_{t+k} | e_{1:t})$  for  $k > 0$ 
  - evaluation of possible action sequences; like filtering without the evidence
- **Smoothing**:  $P(X_k | e_{1:t})$  for  $0 \leq k < t$ 
  - better estimate of past states, essential for learning
- **Most likely explanation**:  $\arg \max_{x_{1:t}} P(x_{1:t} | e_{1:t})$ 
  - speech recognition, decoding with a noisy channel

# Inference tasks

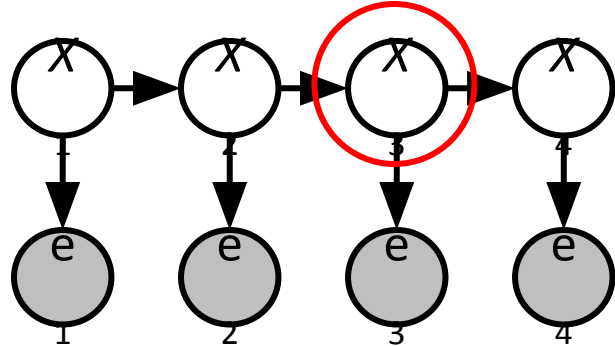
Filtering:  $P(X_t | e_{1:t})$



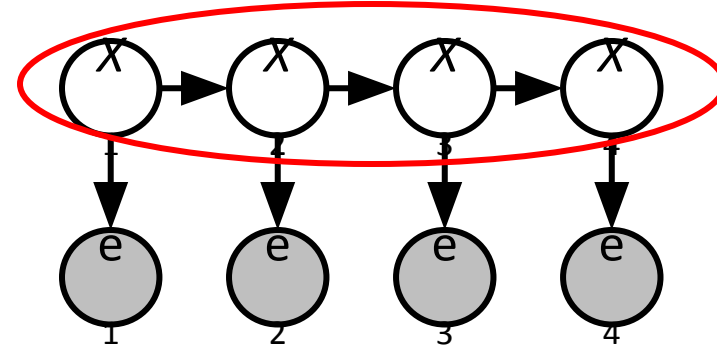
Prediction:  $P(X_{t+k} | e_{1:t})$



Smoothing:  $P(X_k | e_{1:t}), k < t$



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



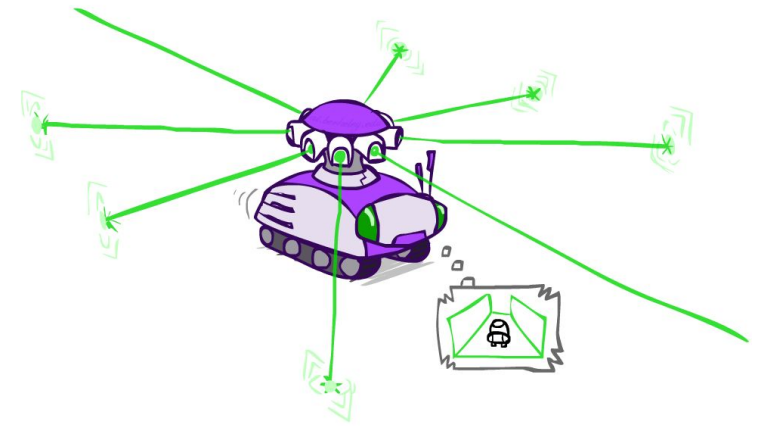
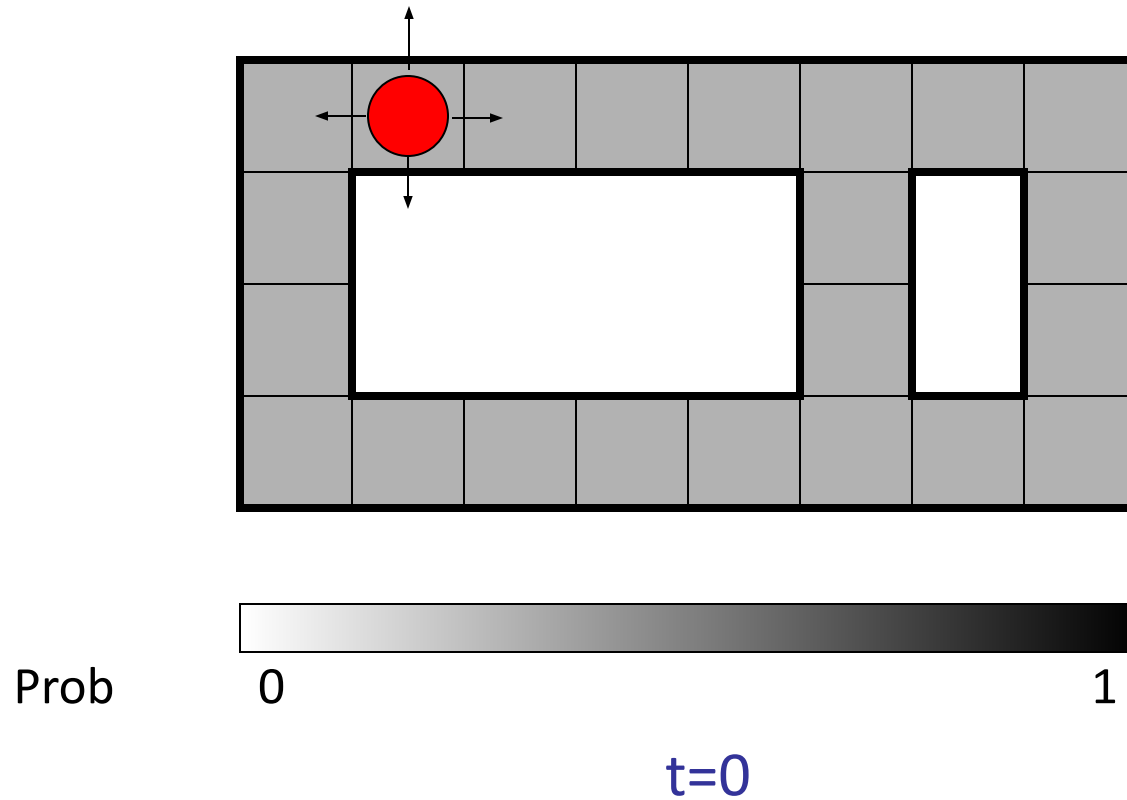
# Filtering / Monitoring

---

- Filtering, or monitoring, or state estimation, is the task of maintaining the distribution  $f_{1:t} = P(X_t | e_{1:t})$  over time
- We start with  $f_0$  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; **1,320,000** papers on Google Scholar

# Example: Robot Localization

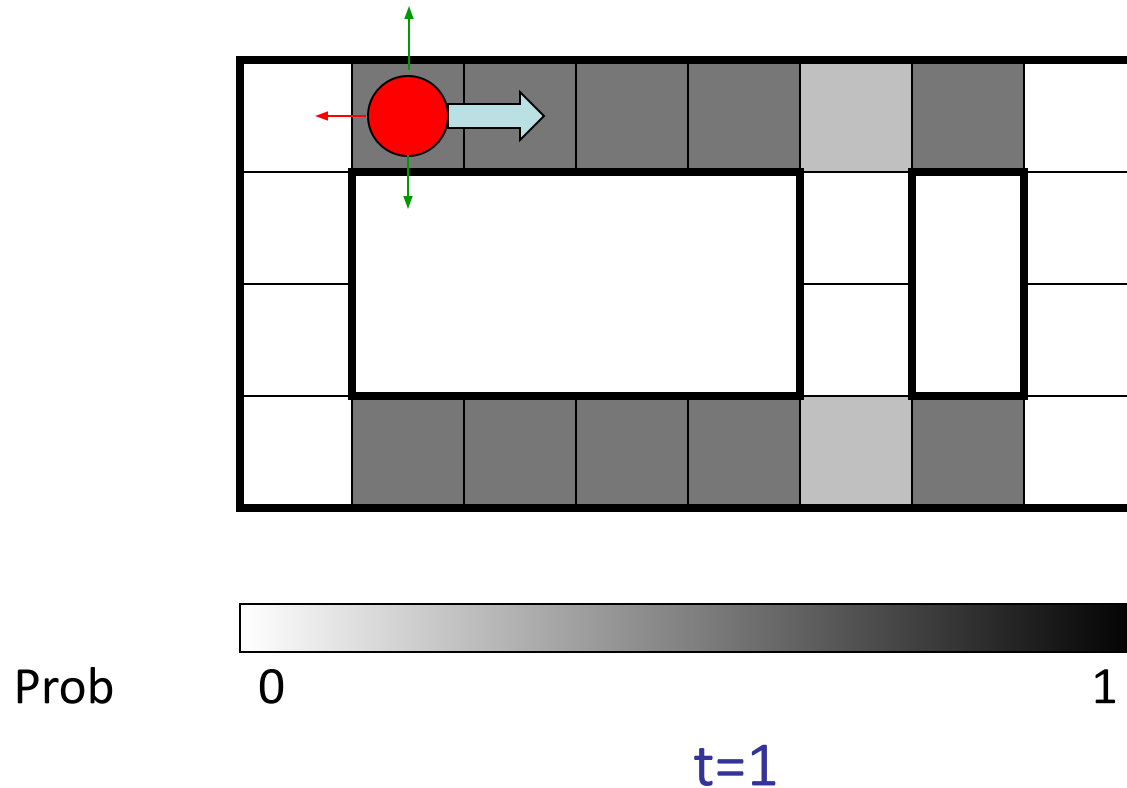
Example from  
Michael Pfeiffer



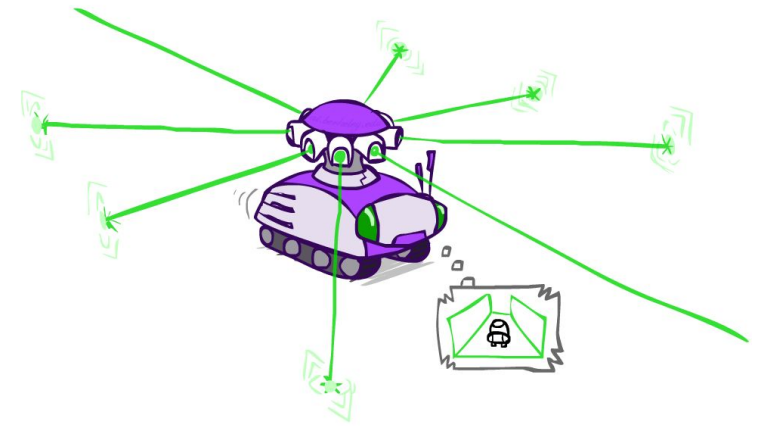
Sensor model: four bits for wall/no-wall in each direction,  
never more than 1 mistake

Transition model: action may fail with small prob.

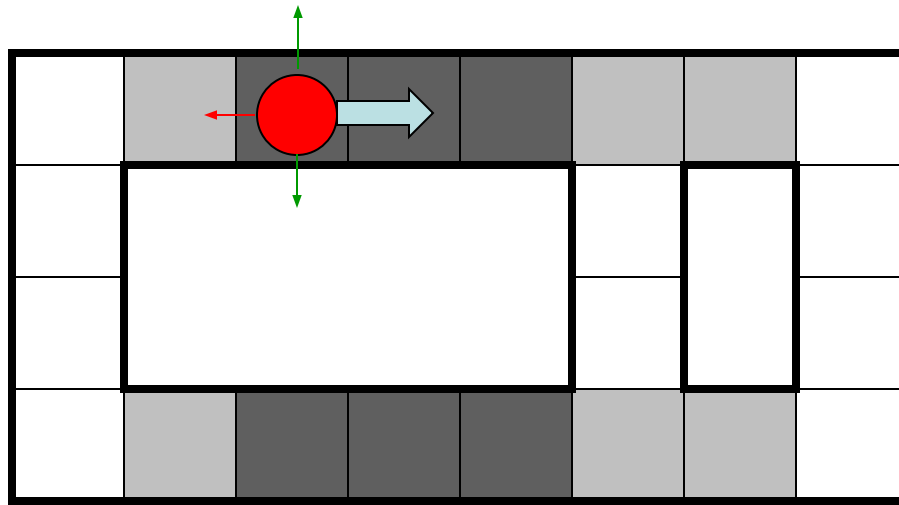
# Example: Robot Localization



Lighter grey: was *possible* to get the reading,  
but *less likely* (required 1 mistake)



# Example: Robot Localization

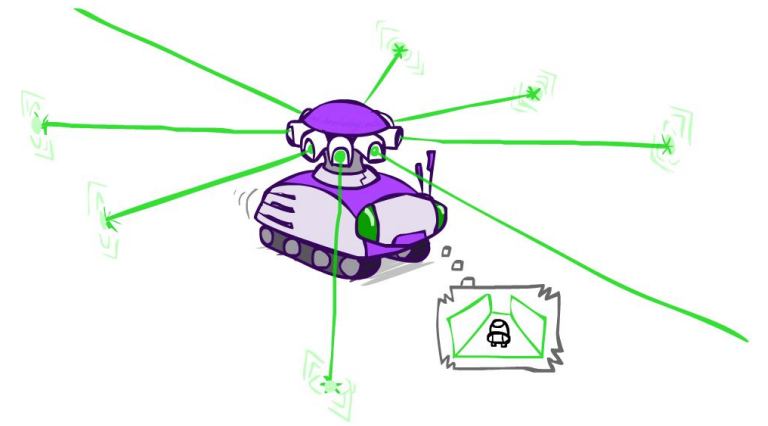


Prob

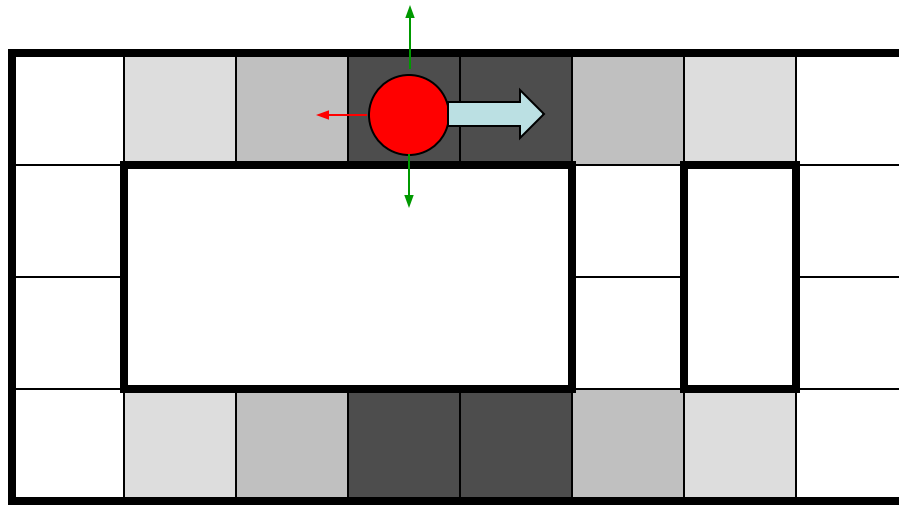
0

1

t=2



# Example: Robot Localization

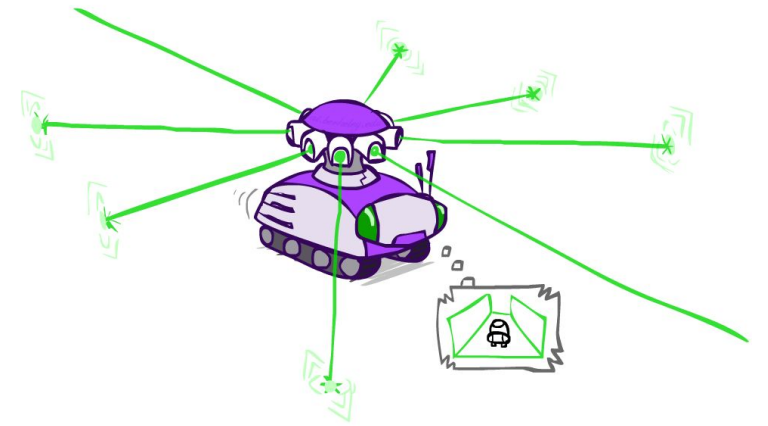


Prob

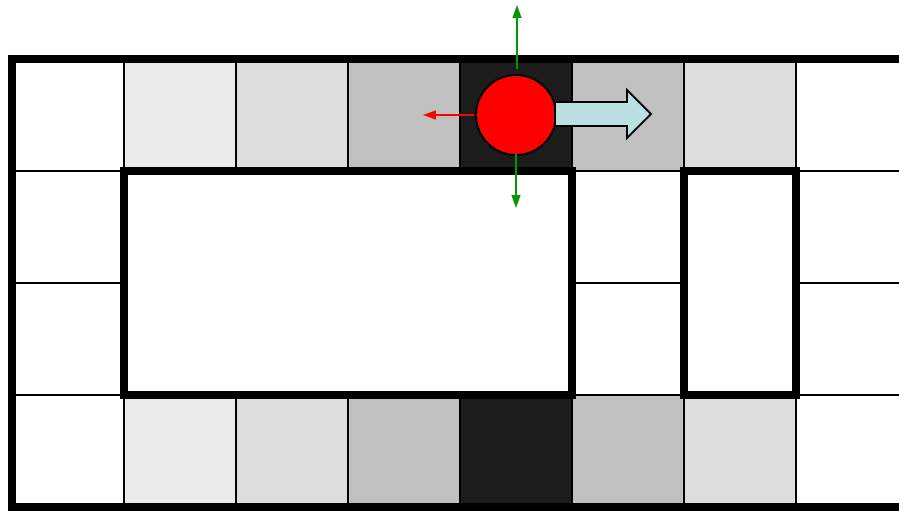
0

1

t=3



# Example: Robot Localization

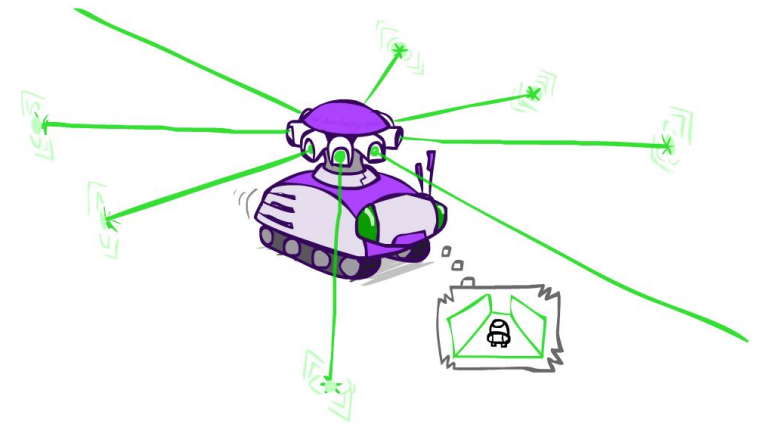


Prob

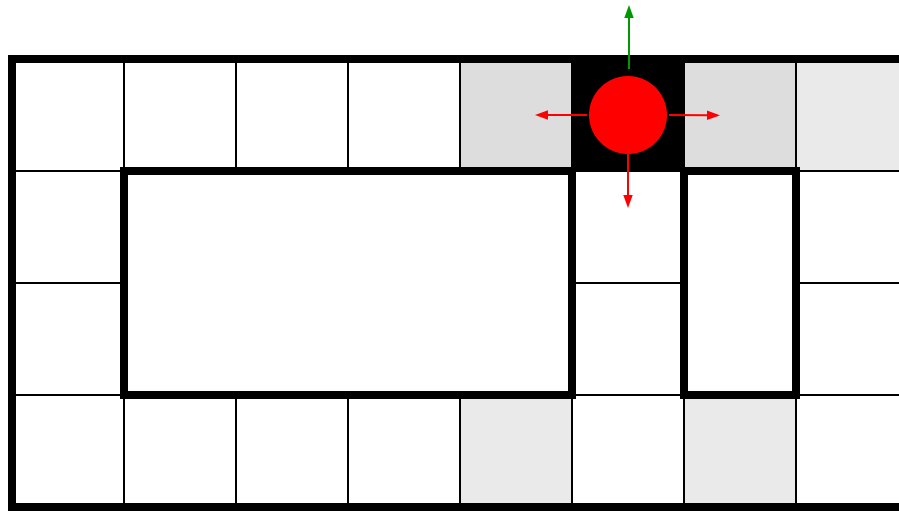
0

1

$t=4$



# Example: Robot Localization

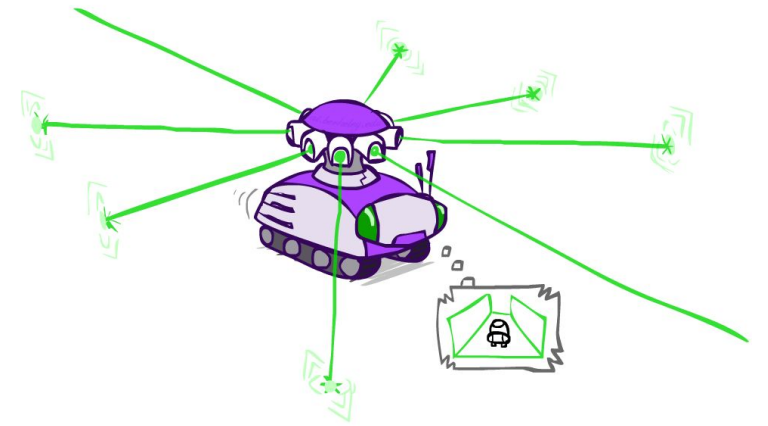


Prob

0

1

$t=5$



# Filtering algorithm

- Aim: devise a **recursive filtering** algorithm of the form

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

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

# Filtering algorithm

- Aim: devise a **recursive filtering** algorithm of the form

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

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

Apply Bayes' rule

Apply conditional independence

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

Condition on  $X_t$

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

Apply conditional independence

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

Normalize

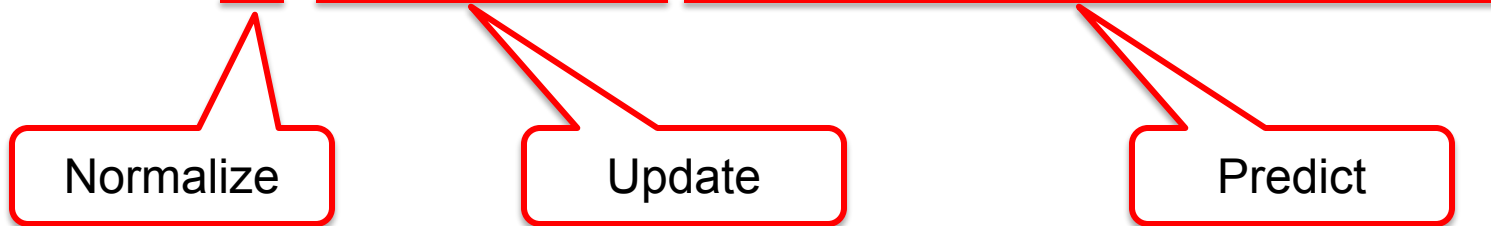
Update

Predict

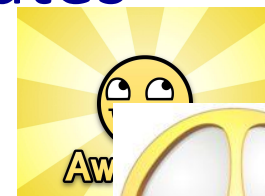
- $P(X_{t+1} | x_t)$

# Filtering algorithm

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



- $f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$
- 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
- We get to invent really cool approximate filtering algorithms



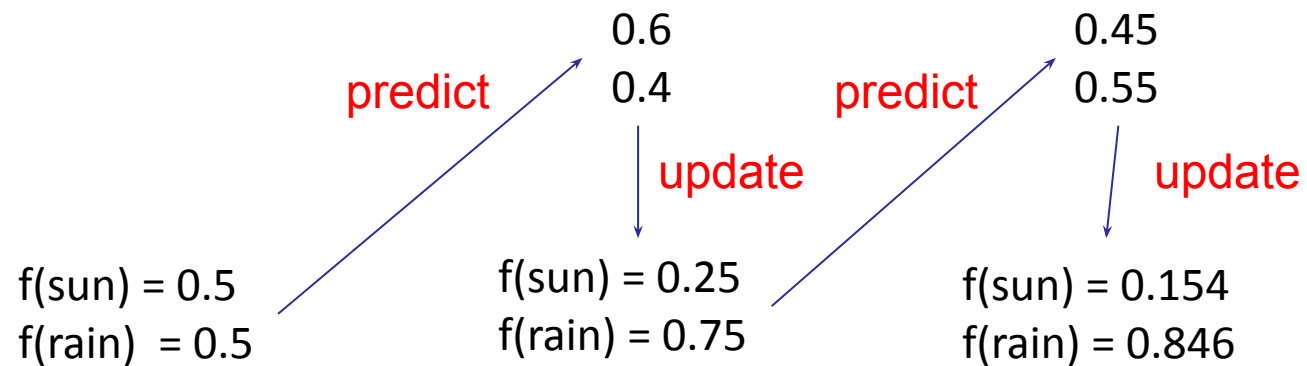
# And the same thing in linear algebra

- Transition matrix  $T$ , observation matrix  $O_t$ 
  - Observation matrix has state likelihoods for  $E_t$  along diagonal
  - E.g., for  $U_1 = \text{true}$ ,  $O_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$
- Filtering algorithm becomes
  - $f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$

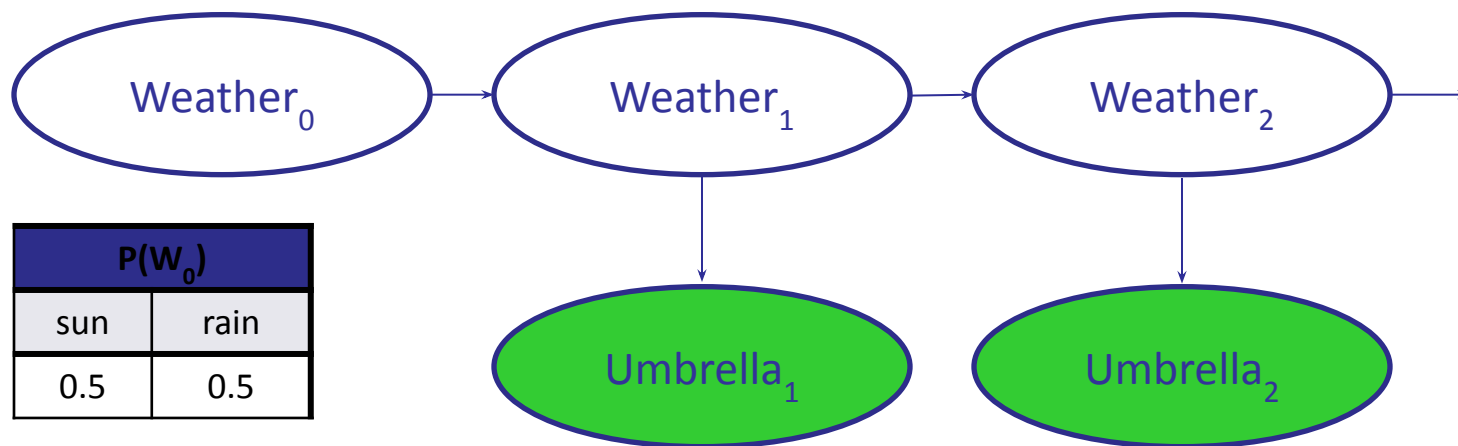
| $X_{t-1}$ | $P(X_t   X_{t-1})$ |      |
|-----------|--------------------|------|
|           | sun                | rain |
| sun       | 0.9                | 0.1  |
| rain      | 0.3                | 0.7  |

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

# Example: Weather HMM



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



| $P(W_0)$ |      |
|----------|------|
| sun      | rain |
| 0.5      | 0.5  |

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

# Pacman – Hunting Invisible Ghosts with Sonar



[Demo: Pacman – Sonar – No Beliefs(L14D1)]

# Video of Demo Pacman – Sonar

---



# Most Likely Explanation

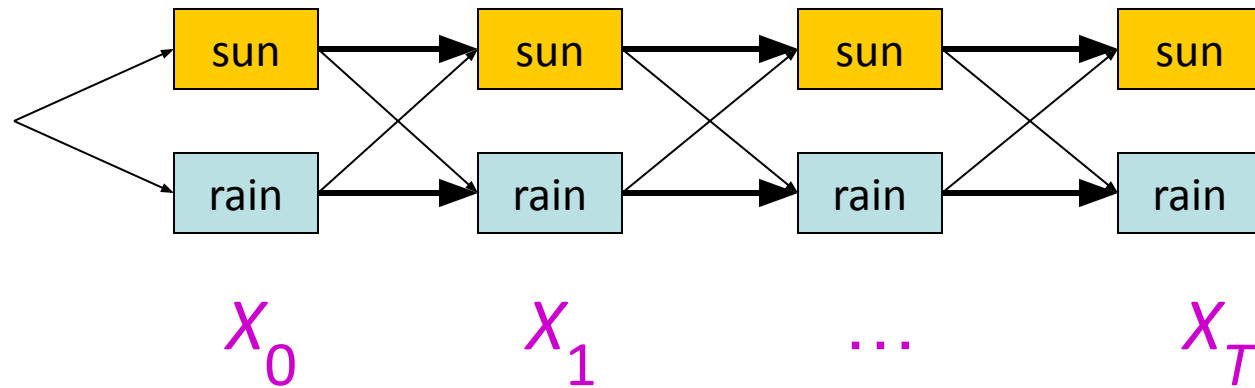


# Inference tasks

- **Filtering:**  $P(X_t | e_{1:t})$ 
  - **belief state**—input to the decision process of a rational agent
- **Prediction:**  $P(X_{t+k} | e_{1:t})$  for  $k > 0$ 
  - evaluation of possible action sequences; like filtering without the evidence
- **Smoothing:**  $P(X_k | e_{1:t})$  for  $0 \leq k < t$ 
  - better estimate of past states, essential for learning
- **Most likely explanation:**  $\arg \max_{x_{1:t}} P(x_{1:t} | e_{1:t})$ 
  - speech recognition, decoding with a noisy channel

# Most likely explanation = most probable path

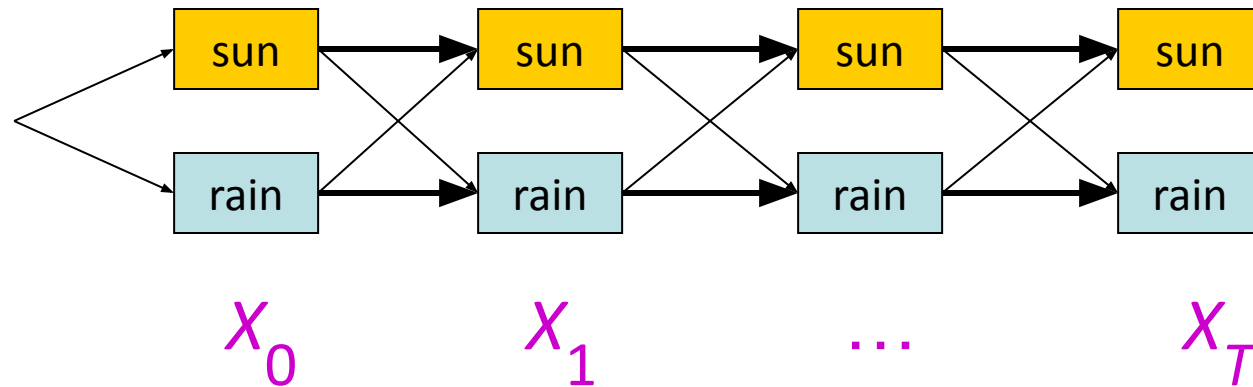
- **State trellis**: graph of states and transitions over time



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

- Each arc represents some transition  $x_{t-1} \rightarrow x_t$
- Each arc has weight  $P(x_t | x_{t-1}) P(e_t | x_t)$  (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

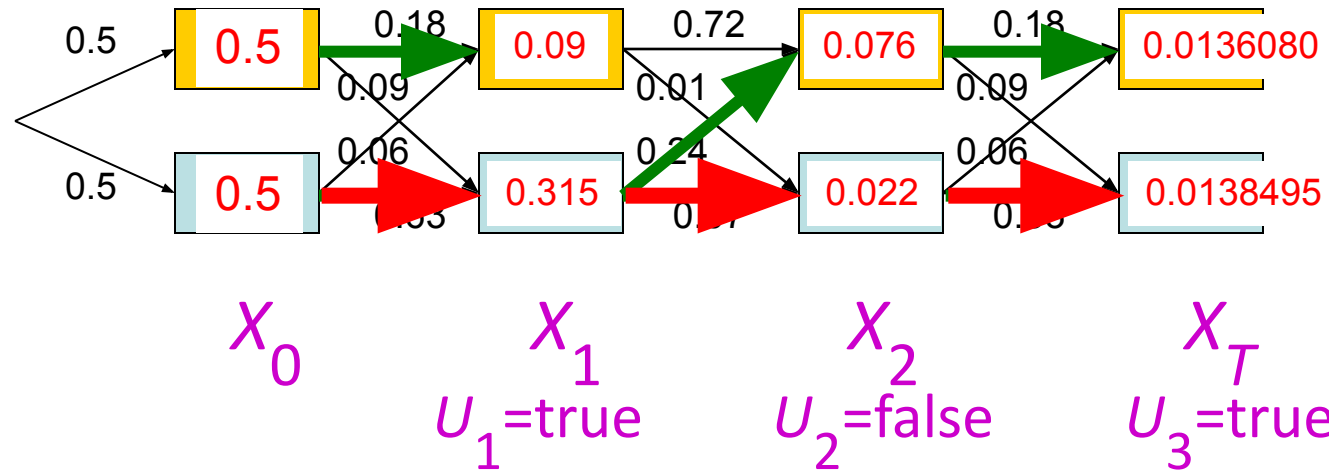
$$\begin{aligned} \mathbf{f}_{1:t+1} &= \text{FORWARD}(\mathbf{f}_{1:t}, e_{t+1}) \\ &= \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) \mathbf{f}_{1:t} \end{aligned}$$

## Viterbi Algorithm (max)

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

$$\begin{aligned} \mathbf{m}_{1:t+1} &= \text{VITERBI}(\mathbf{m}_{1:t}, e_{t+1}) \\ &= P(e_{t+1} | X_{t+1}) \max_{x_t} P(X_{t+1} | x_t) \mathbf{m}_{1:t} \end{aligned}$$

# Viterbi algorithm contd.



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

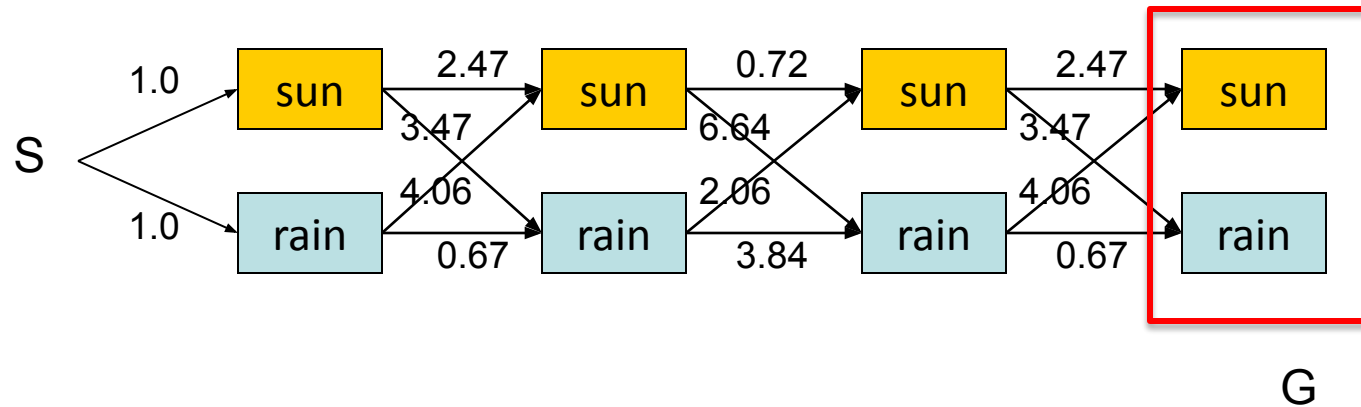
| $W_t$ | $P(U_t   W_t)$ |       |
|-------|----------------|-------|
|       | true           | false |
| sun   | 0.2            | 0.8   |
| rain  | 0.9            | 0.1   |

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

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

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

# Viterbi in negative log space



| $W_{t-1}$ | $P(W_t   W_{t-1})$ |      |
|-----------|--------------------|------|
|           | 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   |

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

Viterbi is essentially breadth-first graph search  
What about A\*?