COMP9418: Advanced Topics in Statistical Machine Learning

Markov Chains and Hidden Markov Models

Instructor: Gustavo Batista

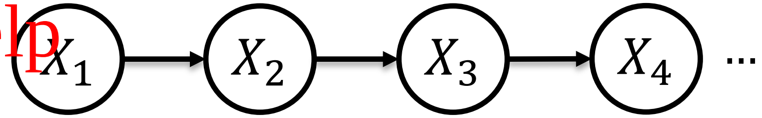
University of New South Wales

Introduction

- This lecture discusses two classes of Graphical Models
  - Markov chains
  - Hidden Markov Models (HMM)
- Both models are instances of Dynamic Bayesian Networks (DBN)
  - They have a repeating structure that grows with time or space
  - Such structure is simple and uses the *Markov property*
- The Markov property states that future states are independent of past ones given the current state
  - In Markov chains, all states are observable
  - HMM extend the chains by allowing hidden states
- We will discuss specialised inference algorithms for both classes
  - Applications of these graphical models in domains such as robot localisation

Time and Space

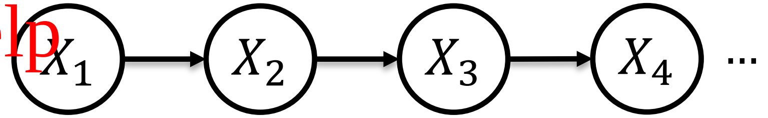
- Several problems require reasoning about sequences
  - Such sequences may represent the problem dynamics in time or space
  - Examples of applications are speech recognition and robot localization



- Dynamic Bayesian Networks (DBN) allow to incorporate time or space in our models
  - The two simplest instances of DBNs are Markov chains and Hidden Markov models

Markov Chains

- Markov chain is a *state machine*
  - X is a discrete variable and each value is called a *state*
  - Transitions between states are nondeterministic



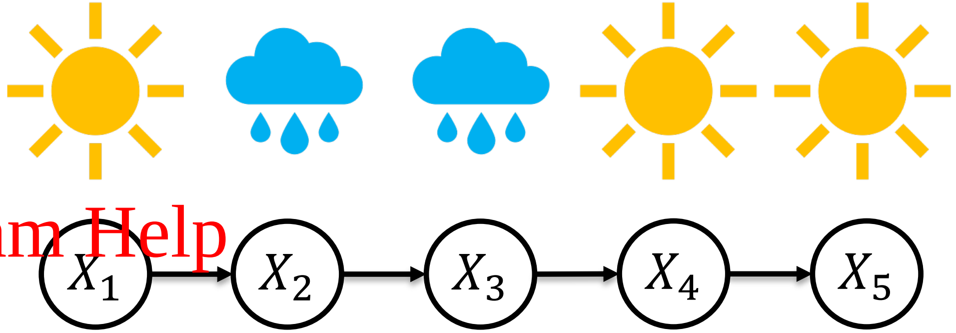
- Parameters
  - Prior probabilities $P(X_1)$
  - Transition probabilities or *dynamics* $P(X_t | X_{t-1})$

- Stationary assumption
  - Transition probabilities are the same for all values of t
  - Also known as a *time-homogeneous* chain

Markov Chains: Weather

- States
  - $X = \{sun, rain\}$
- Initial distribution

$$X_1 = \begin{pmatrix} 1 & sun \\ 0 & rain \end{pmatrix}$$

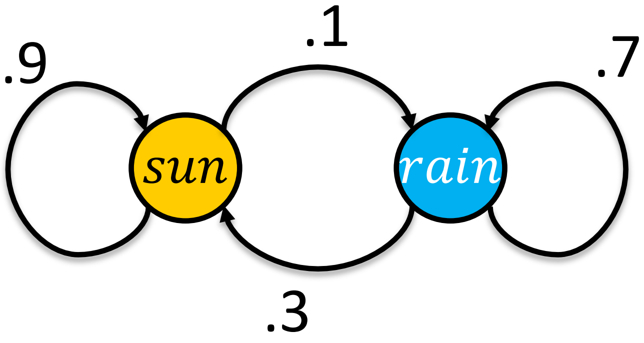


- Transition probabilities

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	.9
sun	$rain$	.1
$rain$	sun	.3
$rain$	$rain$	.7

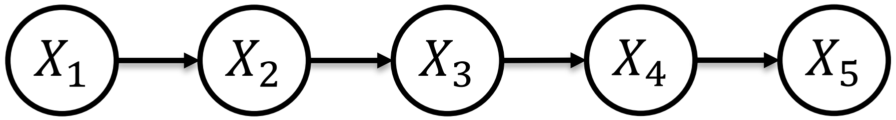
$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{X_t} \\ & & sun & rain \\ \hline X_{t-1} & sun & .9 & .1 \\ & rain & .3 & .7 \end{array}$$

Matrix of transition probabilities



Transition or state diagram

Markov Chains: Independencies



- An relevant question is which independencies are implied in this chain
  - We can use d-separation to visually infer the independencies
  - This independence assumption is known as (first order) Markov property

$X_1 \perp X_3 | X_2$ $\quad$ $X_2 \perp X_4 | X_3$

More generally, $X_{t+1} \perp X_{t-1} | X_t$

- Independences are also apparent when we look at the chain rule
  - Chain rule if Bayesian networks for this example

$$P(X_1, X_2, X_3, X_4, X_5) = P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)P(X_5|X_4)$$

  - Chain rule in general

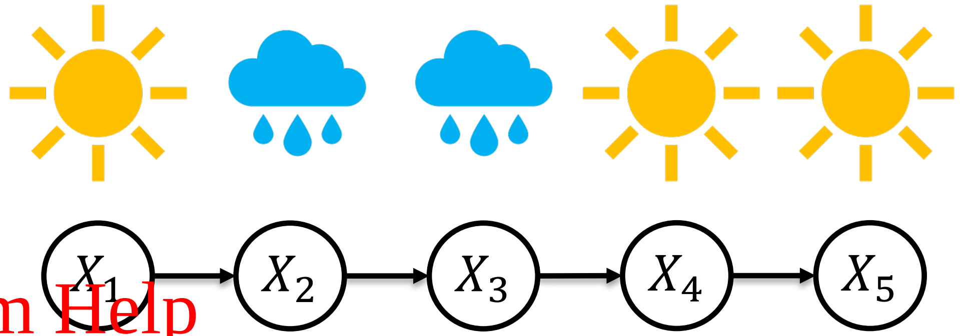
$$P(X_1, X_2, X_3, X_4, X_5) = P(X_1)P(X_2|X_1)P(X_3|X_2, X_1)P(X_4|X_3, X_2, X_1)P(X_5|X_4, X_3, X_2, X_1)$$

$X_3 \perp X_1 | X_2$ $\quad$ $X_4 \perp X_1, X_2 | X_3$ $\quad$ $X_5 \perp X_1, X_2, X_3 | X_4$

Markov Chains: Independencies

- In general

$$P(X_1, \dots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_t | X_{t-1})$$



$|X|^n$ parameters $\quad$ $|X| + (n-1)|X|^2$ parameters

We also assume that $P(X_t|X_{t-1})$ is the same for all t (stationarity)

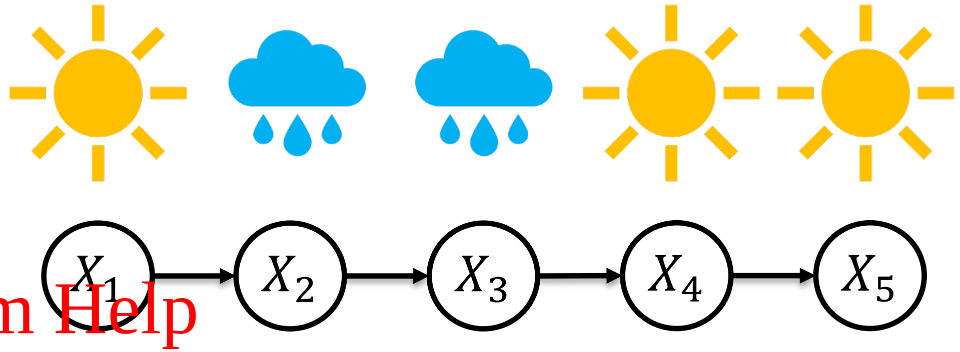
$|X| + |X|^2$ parameters

$$X_1 = \begin{pmatrix} 1 & sun \\ 0 & rain \end{pmatrix}$$

X_{t-1}	X_t	$P(X_t\|X_{t-1})$
sun	sun	.9
sun	$rain$	.1
$rain$	sun	.3
$rain$	$rain$	.7

Probability of a State Sequence

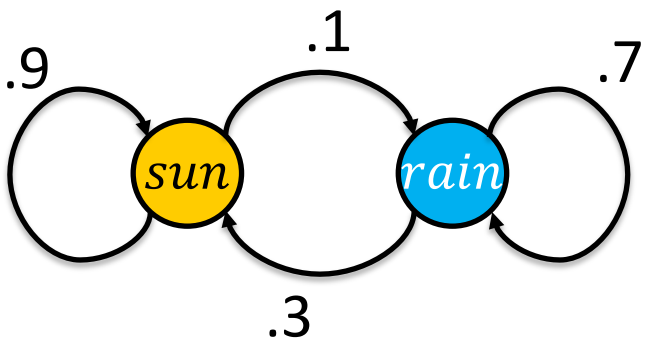
- The probability of a sequence is the product of the transition probabilities
  - This comes directly from the chain rule



$$P(X_1, X_2, X_3, X_4, X_5) =$$
$$P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)P(X_5|X_4)$$

For example, what is the probability of the sequence: sun, rain, rain, sun, sun?

$$P(sun, rain, rain, sun, sun) =$$
$$1(.1)(.7)(.3)(.9) = .189$$



$$X_1 = \begin{pmatrix} 1 & sun \\ 0 & rain \end{pmatrix}$$

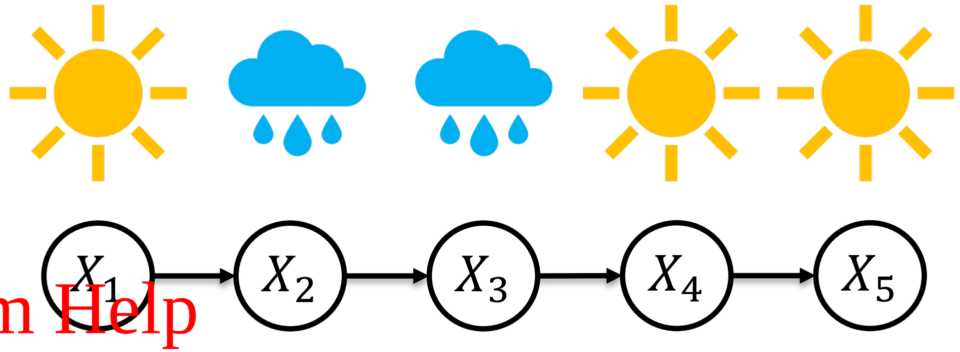
Probability of Staying in a Certain State

- The probability of staying in a certain state for d steps
  - It is the probability of a sequence in this state for $d-1$ steps then going to a different state

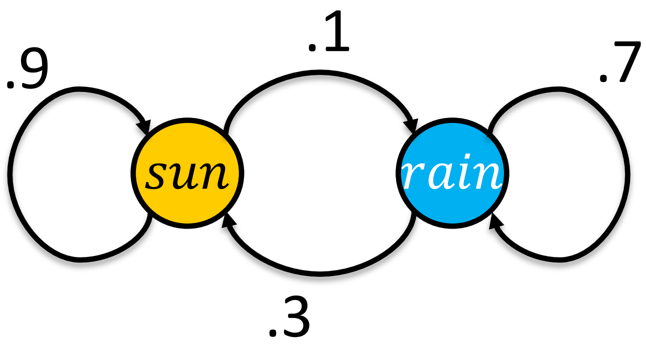
$$\boldsymbol{S}_s^d = \{X_i = s : 1 \leq i \leq d\}$$

$$P(\boldsymbol{S}_s^d) = P(s|s)^{d-1}(1 - P(s|s))$$

$$P(\boldsymbol{S}_{rain}^3) = P(rain|rain)^{3-1}(1 - P(rain|rain)) = (.7^2)(1 - .7) = .147$$



For example, what is the probability of three raining days?



$$X_1 = \begin{pmatrix} 1 & sun \\ 0 & rain \end{pmatrix}$$

Expected Time in a State

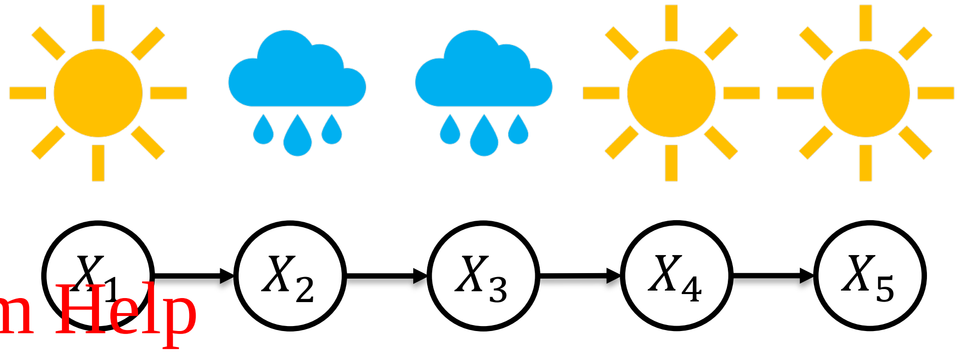
- The average duration of a sequence is a certain state
  - It is the expected number of time steps in that state

$$\mathbb{E}[\boldsymbol{S}_s] = \sum_{i}^{\infty} P(\boldsymbol{S}_s^i) i$$

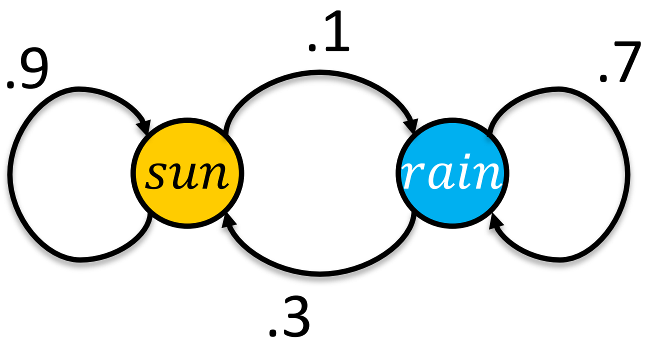
$$= \sum_{i}^{\infty} i \, P(s|s)^{i-1}(1 - P(s|s))$$

$$= \frac{1}{1 - P(s|s)}$$

$$\mathbb{E}\,(S_{rain}) = \frac{1}{1 - .7} = 3.33$$



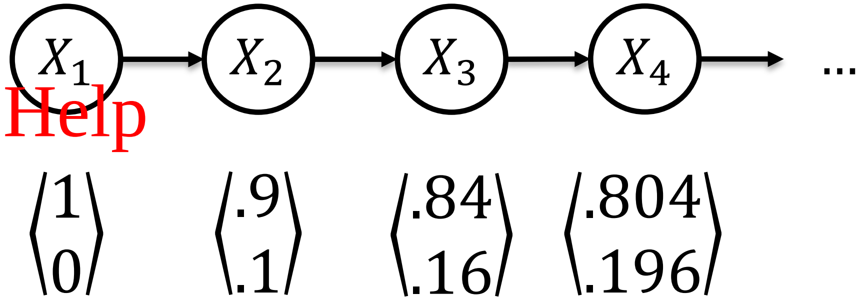
For example, what is the expected number of raining days?



$$X_1 = \begin{cases} 1 & sun \\ 0 & rain \end{cases}$$

Mini-Forward Algorithm

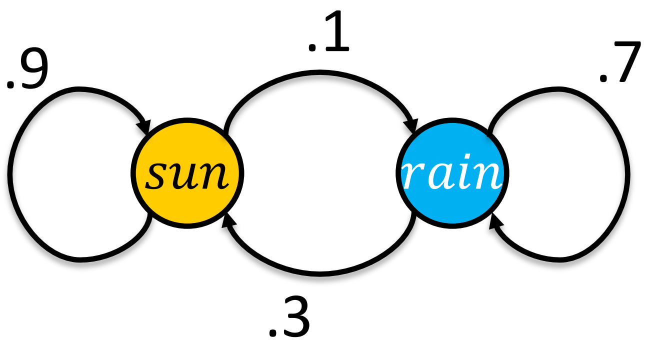
- What is $P(X)$ on some day t?
  - We can obtain an answer by simulating the chain



$P(x_1)$ is known

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

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



Mini-Forward Algorithm

Input: time n, transition probability $P(X_t|X_{t-1})$, prior probability of states $P(X_1)$

Output: $P(X_n)$

for each state x **do**

  $p[x, 1] \leftarrow P(X_1 = x)$

for $t \leftarrow 2$ to n **do**

  for each state x_t **do**

    $p[x_t, t] = 0$

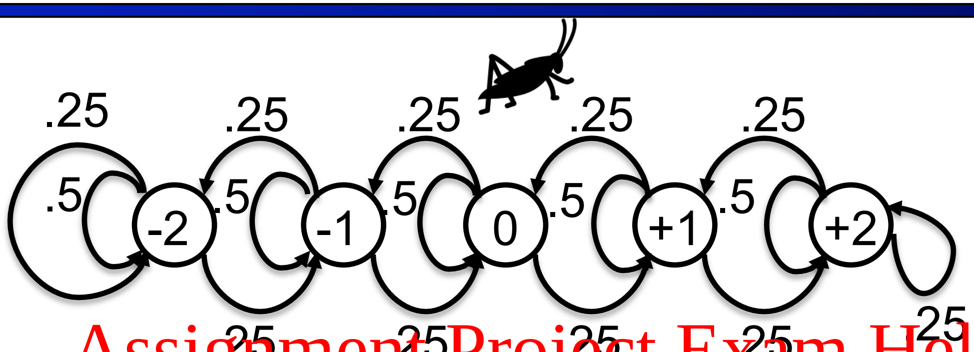
    for each state x_{t-1} **do**

      $p[x_t, t] \leftarrow p[x_t, t] + p[x_{t-1}, t-1]P(x_t|x_{t-1})$

return $p[x, n]$

$$O(n|X|^2)$$

Grasshopper Example



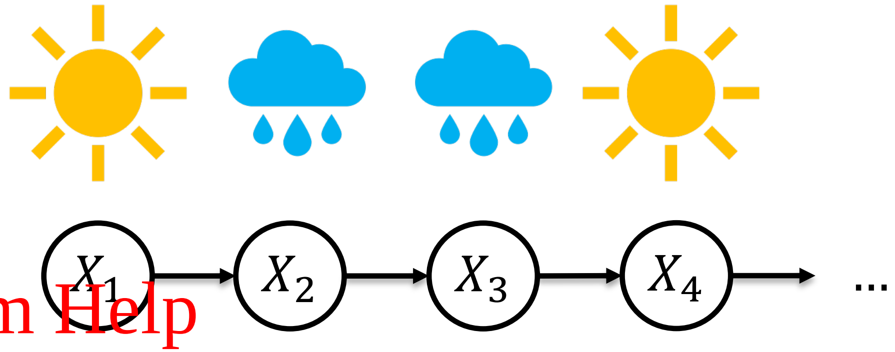
	-2	-1	0	1	2
$P(X_1)$	0	0	1	0	0
$P(X_2)$	0	$.25$	$.5$	$.25$	0
$P(X_3)$	$.25^2 = .0625$	$2(.5)(.25) = .25$	$.5^2 + 2(.25)^2 = .375$	$2(.5)(.25) = .25$	$.25^2 = .0625$

$$P(X_t) = P(X_{t-1})T \qquad [0 \quad 0 \quad 1 \quad 0 \quad 0]\begin{bmatrix} .75 & .25 & & & \\ .25 & .5 & .25 & & \\ & .25 & .5 & .25 & \\ & & .25 & .5 & .25 \\ & & & .25 & .75 \end{bmatrix} = [0 \quad .25 \quad .5 \quad .25 \quad 0]$$

Stationary Distributions

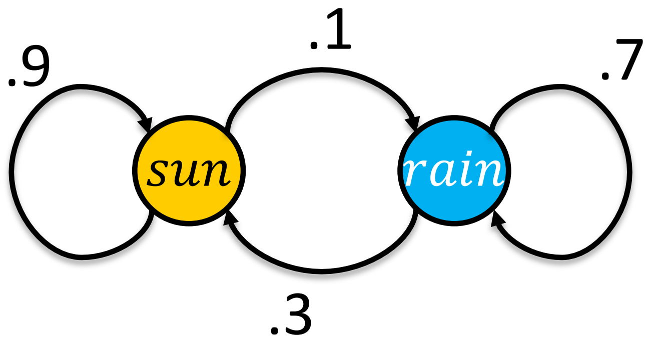
- Starting with a sunny day

$$\begin{pmatrix}1\\0\end{pmatrix} \quad \begin{pmatrix}.9\\.1\end{pmatrix} \quad \begin{pmatrix}.84\\.16\end{pmatrix} \quad \begin{pmatrix}.804\\.196\end{pmatrix} \quad \cdots \quad \begin{pmatrix}.75\\.25\end{pmatrix}$$



- Starting with a rainy day

$$\begin{pmatrix}0\\1\end{pmatrix} \quad \begin{pmatrix}.3\\.7\end{pmatrix} \quad \begin{pmatrix}.48\\.52\end{pmatrix} \quad \begin{pmatrix}.588\\.412\end{pmatrix} \quad \cdots \quad \begin{pmatrix}.75\\.25\end{pmatrix}$$



- Starting with an unknown day

$$\begin{pmatrix}p\\1-p\end{pmatrix} \quad \cdots \quad \begin{pmatrix}.75\\.25\end{pmatrix}$$

Stationary Distributions

- For most chains
  - Influence of the initial distribution gets less and less over time
  - The distribution we end up in is independent of the initial distribution

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

- Stationary distribution
  - The *stationary distribution* π of the chain is the distribution we obtain if the chain converges
  - The stationary distribution satisfies

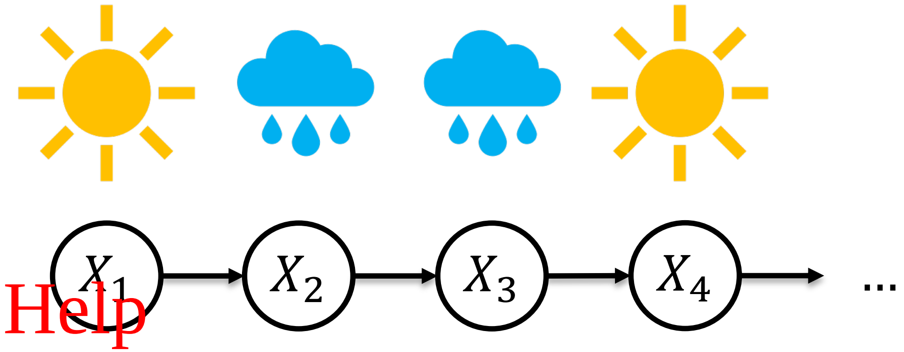
$$\pi(X) = \sum_x P(X|x)\pi(x)$$

$$\pi = \pi T$$

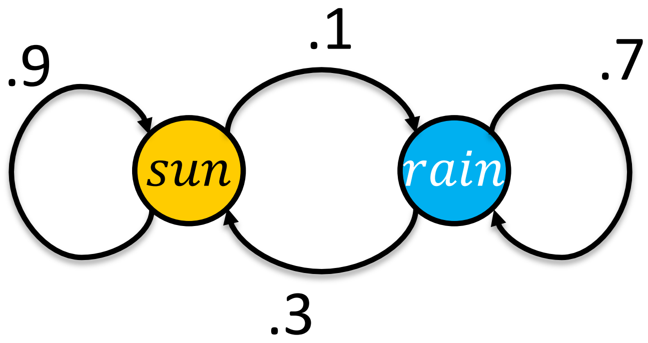
Stationary Distributions

- Question: What is $P(X_\infty)$?

$$\pi(sun) = P(sun|sun)\pi(sun) + P(sun|rain)\pi(rain)$$
$$\pi(rain) = P(rain|sun)\pi(sun) + P(rain|rain)\pi(rain)$$



$$\pi(sun) = 0.9\,\pi(sun) + 0.3\,\pi(rain)$$
$$\pi(rain) = 0.1\,\pi(sun) + 0.7\,\pi(rain)$$

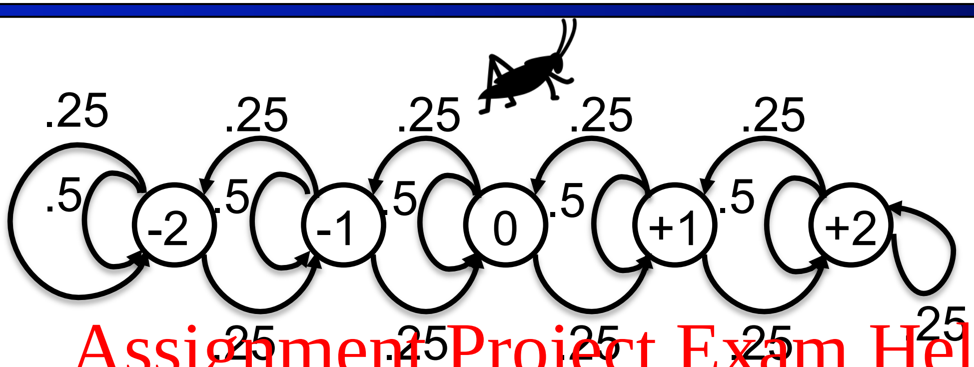


$$\pi(sun) = 3\pi(rain)$$
$$\pi(rain) = 1/3\pi(sun)$$

$$\pi(sun) + \pi(rain) = 1$$

$$\pi(sun) = 3/4$$
$$\pi(rain) = 1/4$$

Stationary Distributions: Grasshopper

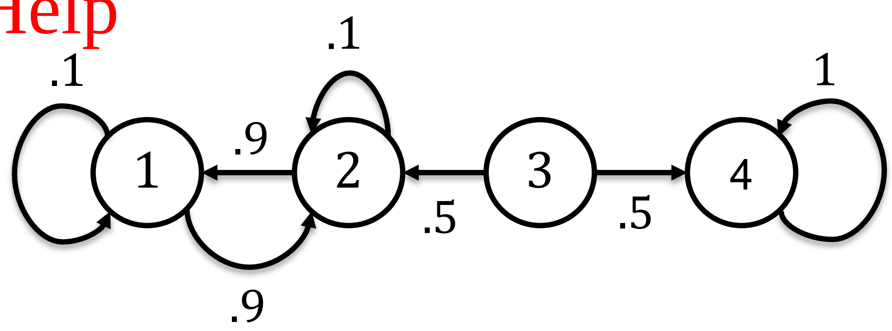


- What is the stationary distribution?

$$T = \begin{bmatrix} .75 & .25 & & & \\ .25 & .5 & .25 & & \\ & .25 & .5 & .25 & \\ & & .25 & .5 & .25 \\ & & & .25 & .75 \end{bmatrix}$$

Irreducible Markov Chains

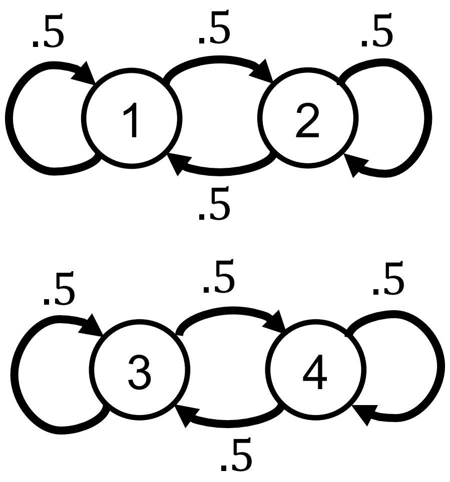
- A Markov chain is *irreducible* if every state x' is reachable from every other state x
  - That is, for every pair of states x and x', there is some time *t* such that the $P(X_t = x' | X_1 = x) > 0$
  - Also known as *regular* or *ergodic* chain
- In this case, the states of the Markov chain are said to be *recurrent*
  - Each state is guaranteed to be visited an infinite number of times when we simulate the chain



A reducible Markov chain

Stationary Distribution

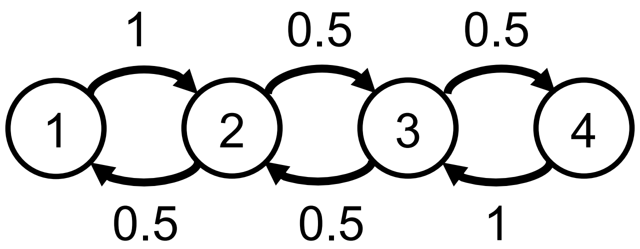
- Every (finite state) Markov chain has at least one stationary distribution
  - Yet an irreducible Markov chain is guaranteed to have a unique stationary distribution



- An irreducible chain may or may not converge to its stationary distribution
  - To guarantee convergence, we need an additional property: *Aperiodicity*

Aperiodic Markov Chains

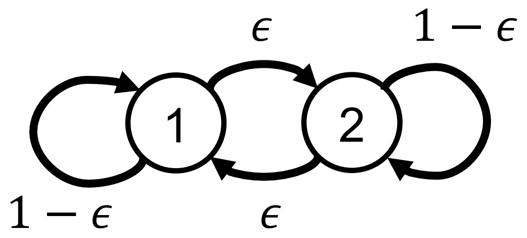
- A Markov chain is aperiodic if it is possible to return to any state at any time
  - There exists an t such that for all state x and all $t' \geq t$, $P(X_{t'} = x \mid X_1 = x) > 0$
- An irreducible and aperiodic Markov chain converges to a unique stationary distribution
  - Irreducible: we can go from any state to any state
  - Aperiodic: avoids chains that alternates forever between states without ever settling in a stationary distribution



An irreducible but periodic Markov chain

Markov Chains Convergence

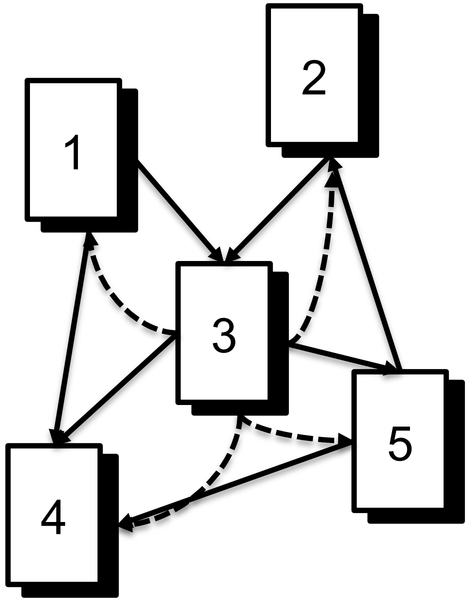
- Although an irreducible and aperiodic Markov chain converges to a single stationary distribution, the convergence can be slow
  - In this example, the stationary distribution is close to $(0.5, 0.5)$
  - For a small ϵ it will take a very long time to reach the stationary distribution
  - We stay in the same state with high probability and rarely transition to another state
  - The average of these states will converge to $(0.5, 0.5)$, but the convergence will be very slow



Applications of Markov Chains

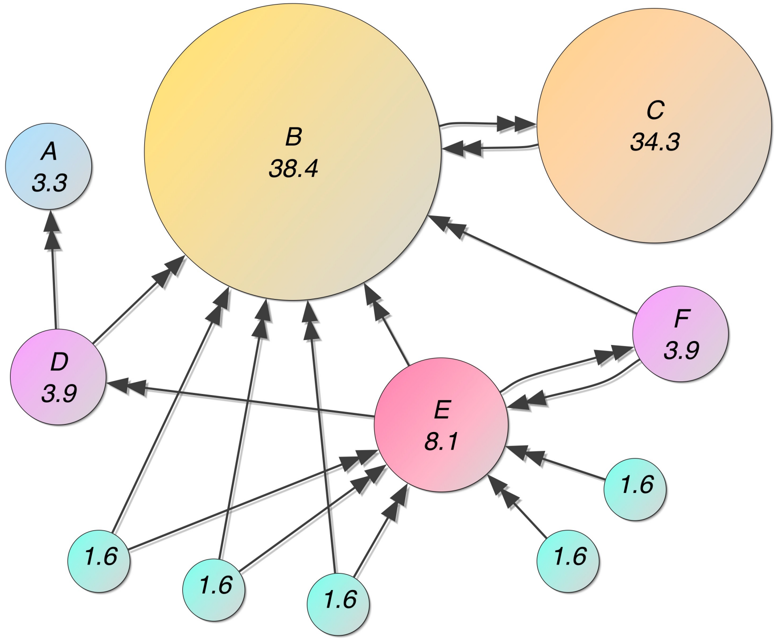
- Markov chains have several well-know applications
  - Markov chain Monte Carlo (MCMC) is a powerful approximate inference algorithm used in statistical software such as Stan
  - MCs are part of the (LZMA) Lempel-Ziv-Markov compression algorithm used in 7zip
  - PageRank algorithm used by Google 1.0 is a direct application of MCs

- PageRank
  - Model the web as a state graph: pages are states and hyperlinks are transitions
  - Each transition from state i has a probability $\frac{\alpha}{k_i}$, where α is a constant parameter and k_i is the outgoing degree of node i
  - Compute a stationary distribution. But it is not unique. Why?
  - Augment the graph with phantom transitions of weight $\frac{1-\alpha}{N}$, where N is the number of nodes in the graph



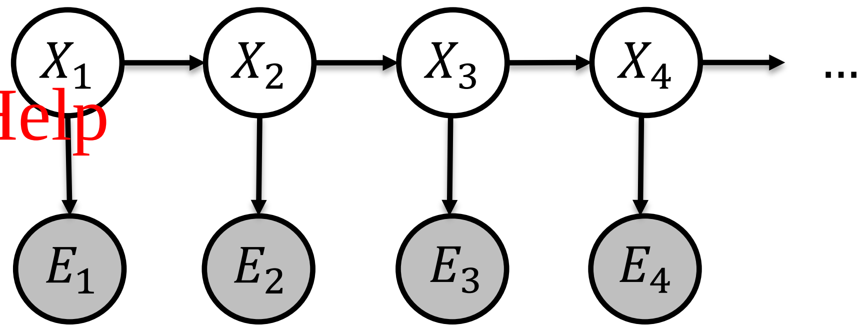
Applications of Markov Chains

- Markov chains have several well-know applications
  - Markov chain Monte Carlo (MCMC) is a powerful approximate inference algorithm used in statistical software such as Stan
  - MCs are part of the (LZMA) Lempel-Ziv-Markov compression algorithm used in 7zip
  - PageRank algorithm used by Google 1.0 is a direct application of MCs
- PageRank
  - Model the web as a state graph: pages are states and hyperlinks are transitions
  - Each transition from state i has a probability $\frac{\alpha}{k_i}$, where α is a constant parameter and k_i is the outgoing degree of node i
  - Compute a stationary distribution. But it is not unique. Why?
  - Augment the graph with phantom transitions of weight $\frac{1-\alpha}{N}$, where N is the number of nodes in the graph



Hidden Markov Models (HMM)

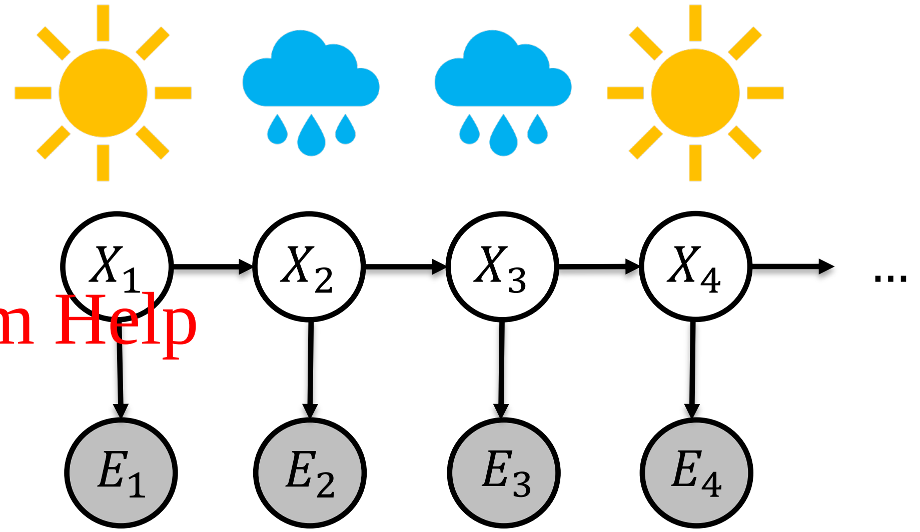
- Hidden Markov Models (HMM) are Markov chains where the states are not directly observable
  - In the weather example, the weather may not be directly observable
  - Instead, we use sensors, such as temperature, air pressure, humidity, wind speed, etc.



- HMM has two components
  - Underlying Markov chain over states X
  - Observable outputs (effects of the states) at each time step
  - These outputs are often called *emissions*

HMM Weather Example

- HMM parameters
  - Initial distribution $P(X_1)$
  - Transition probabilities $P(X_t|X_{t-1})$
  - Emission probabilities $P(E_t|X_t)$

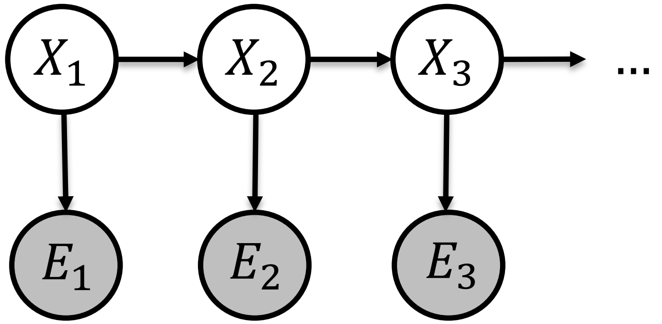


X_1	$P(X_1)$
sun	.5
$rain$	.5

X_{t-1}	X_t	$P(X_t\|X_{t-1})$
sun	sun	.7
sun	$rain$	.3
$rain$	sun	.3
$rain$	$rain$	.7

X_t	E_t	$P(E_t\|X_t)$
sun	umb	.2
sun	$\overline{umb}$	.8
$rain$	umb	.9
$rain$	$\overline{umb}$	.1

HMM: Independencies



- The chain rule of Bayesian networks for HMMs

$$P(X_1, E_1, \ldots, X_n, E_n) = P(X_1)P(E_1|\mathrm{X}_1)\prod_{t=1}^{n} P(X_t|X_{t-1})P(E_t|X_t)$$

- Independences are also apparent when we look at the chain rule

  - Chain rule for Bayesian networks for this example

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

  - Chain rule in general

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

$X_2 \perp E_1|X_1$

$X_3 \perp X_1, E_1, E_2|X_2$

$E_2 \perp X_1, E_1|X_2$

$E_3 \perp X_1, X_2, E_1, E_2|X_3$

HMM: Independencies

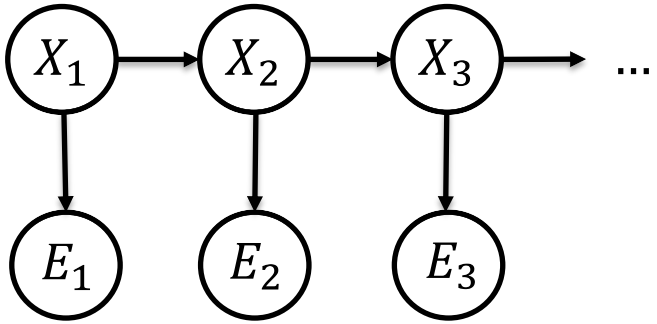
- In general, HMM have the following independency assumptions
  - A state is independent of all past states and all past evidence given the previous state (Markov property)

    $$X_t \perp X_1, \dots, X_{t-2}, E_1, \dots, E_{t-2} | X_{t-1}$$

  - Evidence is independent of all past evidence and all past states given the current state (independence of observations)

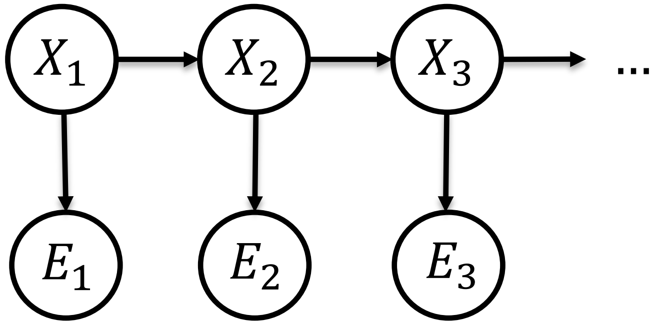
    $$E_t \perp X_1, \dots, X_{t-1}, E_1, \dots, E_{t-1} | X_t$$

  - Transition and emission probabilities are the same for all values of t (stationary process)

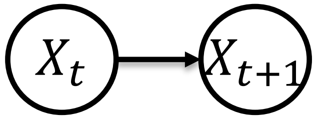


HMM: Inference

- We start with a first task of tracking the distribution $P(X_t)$ over time
  - This task is known as *filtering* or *monitoring*
  - We use $\mathrm{B}(X_t) = P(X_t|e_1, \ldots, e_t)$ to denote the *belief of state*
  - We start with $B(X_1)$, usually using a uniform distribution
  - Update $B(X_t)$ as time passes and we get new observations
- The inference has two main steps
  - Passage of time
  - Observation



Passage of Time



- Suppose we know the current state of belief $B(X_t)$

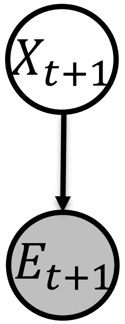
$$B(X_t) = P(X_t | e_{1:t})$$

  - We need to update it as one unit of time passes. Our aim is to compute $P(X_{t+1} | e_{1:t})$

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

Observation

- Given we updated the belief with passage of time
  - We know $P(X_{t+1}|e_{1:t})$ and we need to update it to $B(X_{t+1}) = P(X_{t+1}|e_{1:t+1})$



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

We must renormalise the results by $\sum B(X_{t+1})$

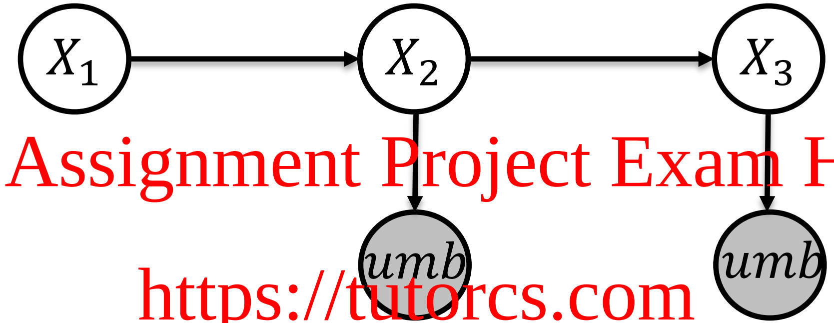
$$B(X_{t+1}) \propto P(e_{t+1}|X_{t+1}) P(X_{t+1}|e_{1:t})$$

HMM Weather Example

$B(sun) = .5$
$B(rain) = .5$

$B(sun) = .5$
$B(rain) = .5$

$B(sun) = .373$
$B(rain) = .627$



$B(sun) = .182$
$B(rain) = .818$

$B(sun) = .117$
$B(rain) = .883$

X_1	$P(X_1)$
sun	.5
$rain$	.5

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	.7
sun	$rain$	.3
$rain$	sun	.3
$rain$	$rain$	.7

X_t	E_t	$P(E_t \mid X_t)$
sun	umb	.2
sun	$\overline{umb}$	.8
$rain$	umb	.9
$rain$	$\overline{umb}$	.1

Forward Algorithm

- Suppose we have a sequence of evidence observations and we want to know the state belief at the end of the sequence

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

$$
\begin{aligned}
P(X_t|e_{1:t}) \propto\ & P(X_t, e_{1:t}) \\
&= \textstyle\sum_{x_{t-1}} P(X_t, x_{t-1}, e_{1:t}) \\
&= \textstyle\sum_{x_{t-1}} P(X_t, x_{t-1}, e_t, e_{1:t-1}) \\
&= \textstyle\sum_{x_{t-1}} P(x_{t-1})P(X_t|x_{t-1})P(e_t|x_{t-1}, X_t)P(e_{1:t-1}|e_t, x_{t-1}, X_t) \\
&= \textstyle\sum_{x_{t-1}} P(x_{t-1})P(X_t|X_{t-1})P(e_t|X_t)P(e_{1:t-1}|x_{t-1}) \\
&= \textstyle\sum_{x_{t-1}} P(X_t|x_{t-1})P(e_t|X_t)P(e_{1:t-1}, x_{t-1}) \\
&= P(e_t|X_t) \textstyle\sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}, e_{1:t-1})
\end{aligned}
$$

You can renormalise every step, but this algorithm often renormalised only the final one

Forward Algorithm

Input: time n, transition probability T, emission probability E, prior probability of states $P(X_1)$, sequence of observations $\{e_2, \dots, e_t\}$

Output: $B(X_t)$

for each state x **do**

$\quad p[x, 1] \leftarrow P(X_1 = x)$

for $t \leftarrow 2$ to n **do**

$\quad$ **for each** state x_t **do**

$\quad\quad p[x_t, t] = 0$

$\quad\quad$ **for each** state x_{t-1} **do**

$\quad\quad\quad p[x_t, t] \leftarrow p[x_t, t] + p[x_{t-1}, t-1] T(x_t | x_{t-1})$

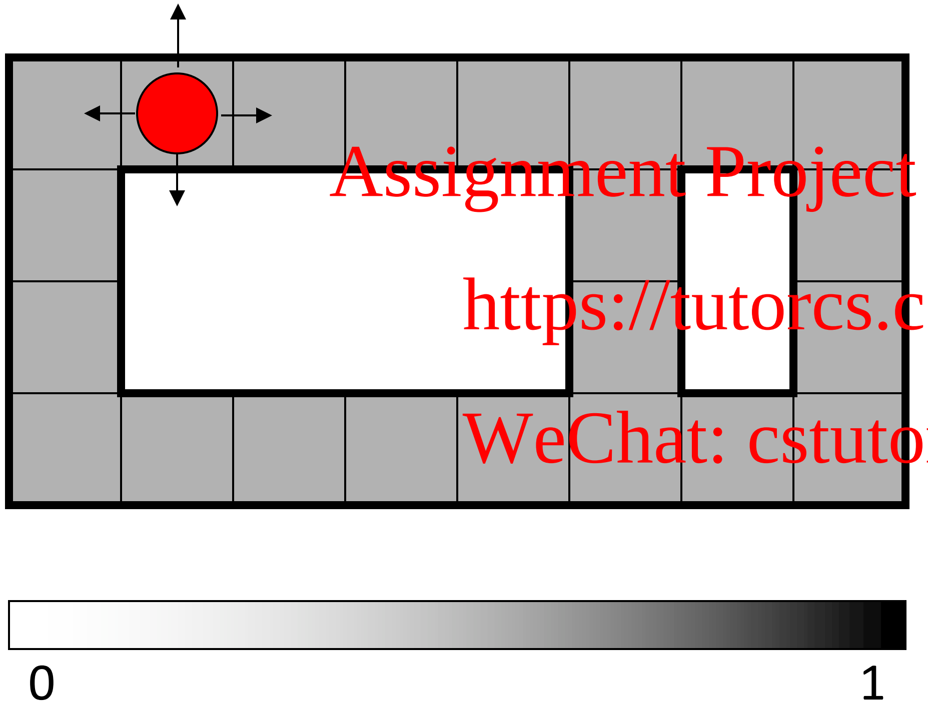
$\quad\quad p[x_t, t] \leftarrow p[x_t, t] E(e_t | x_t)$

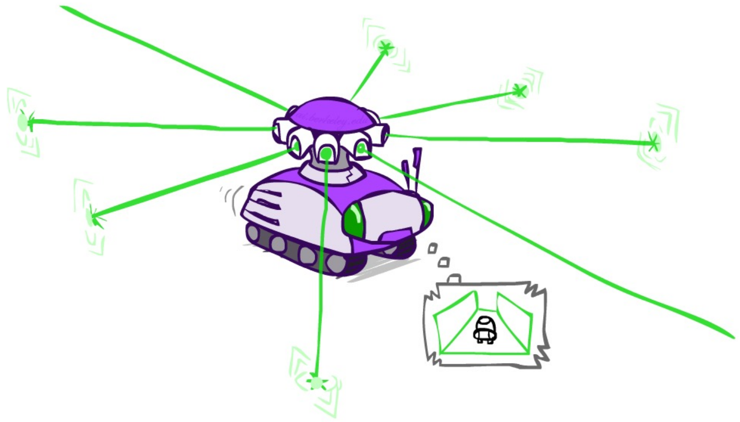
return normalised $p[x, n]$ for all states x

$$O(n|X|^2)$$

Example: Robot Localization

Example from
Michael Pfeiffer





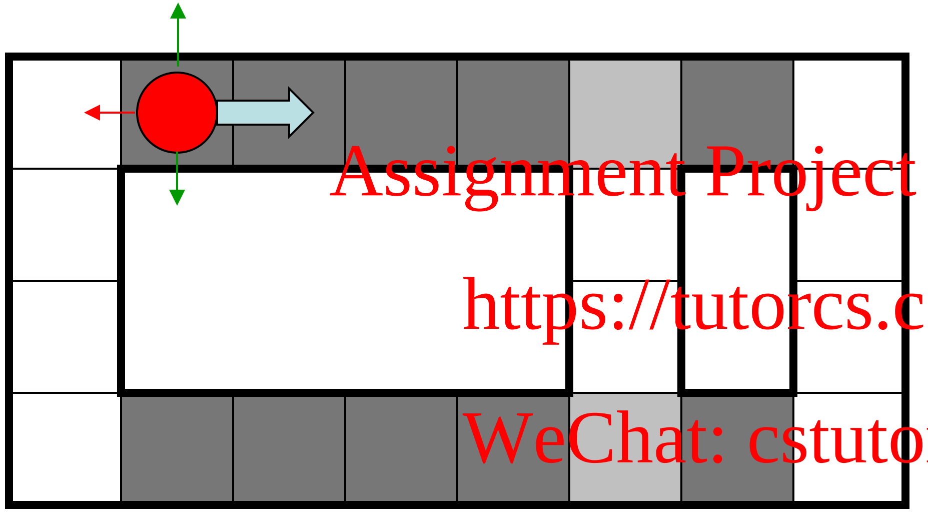
t=0

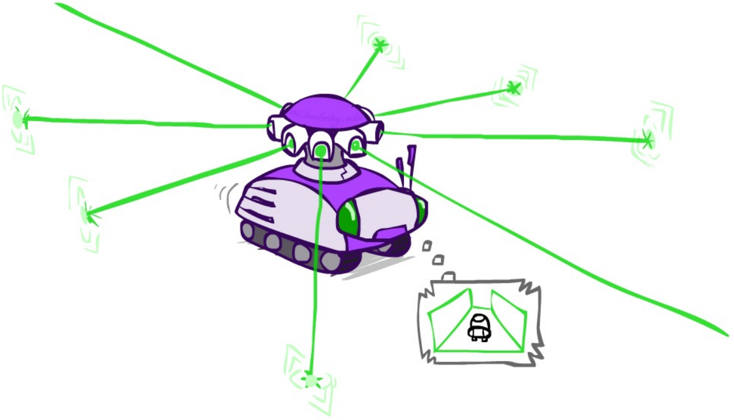
Sensor model: can read in which directions there is a wall, never more than 1 mistake

Motion model: may not execute action with small prob.

Slide from Berkeley AI course

Example: Robot Localization





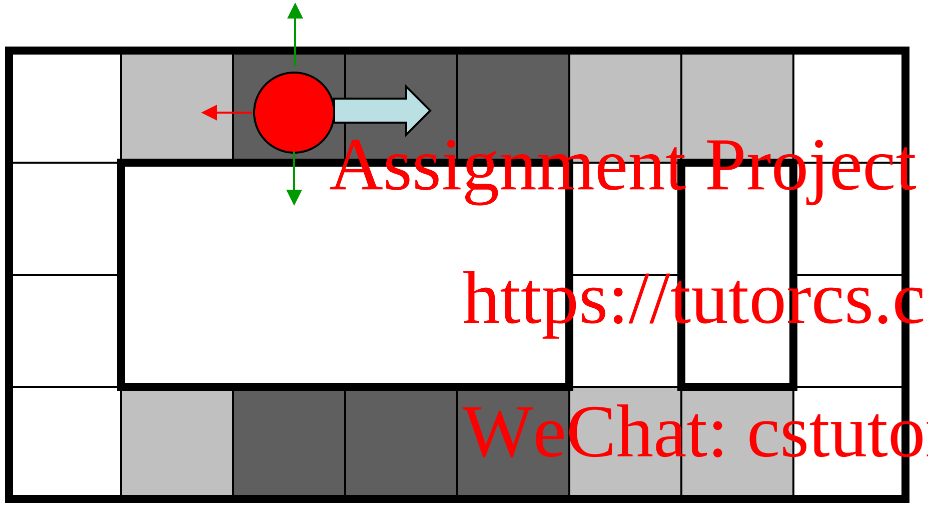
Prob 0 1

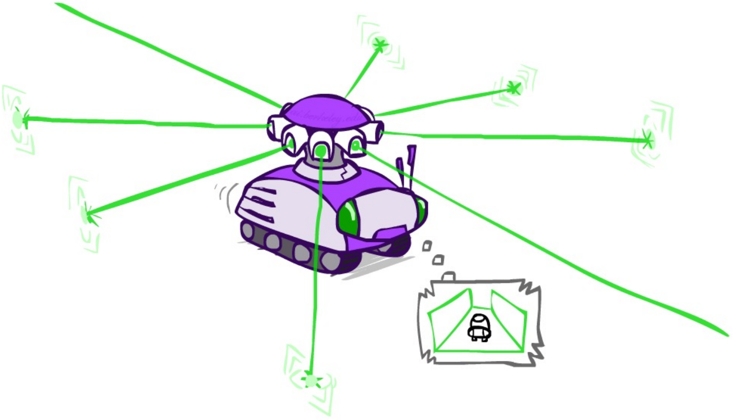
t=1

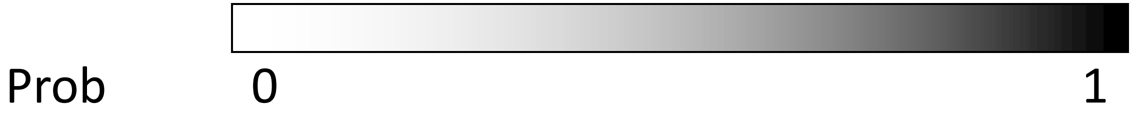
Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

Slide from Berkeley AI course

Example: Robot Localization



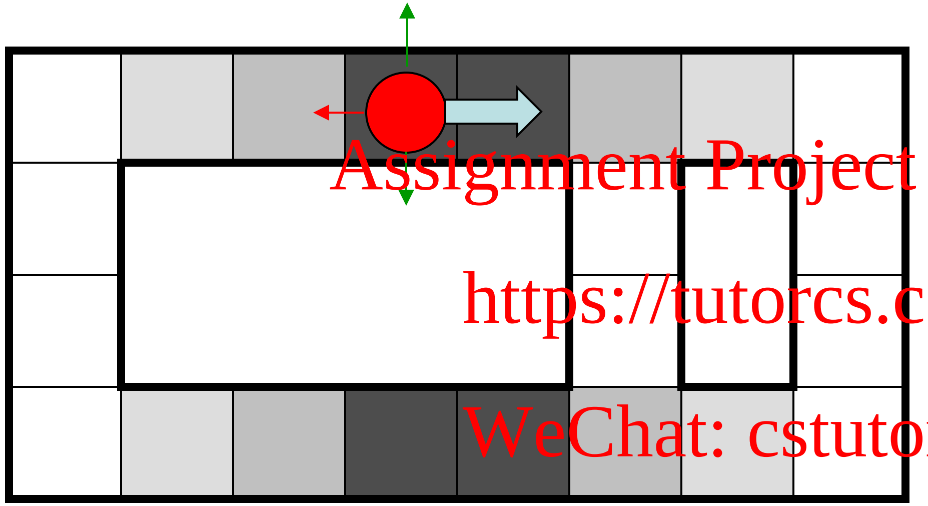


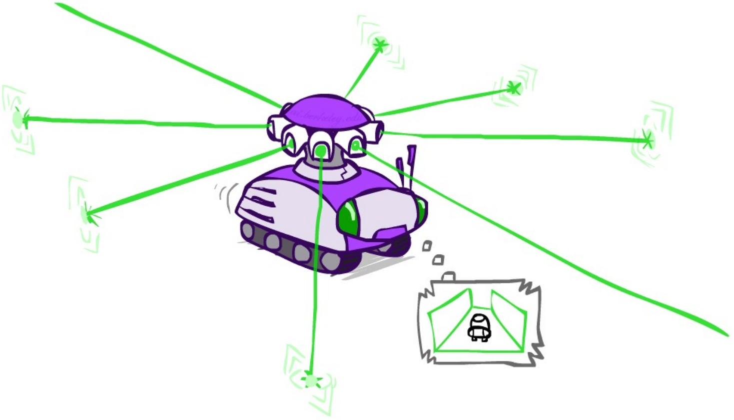


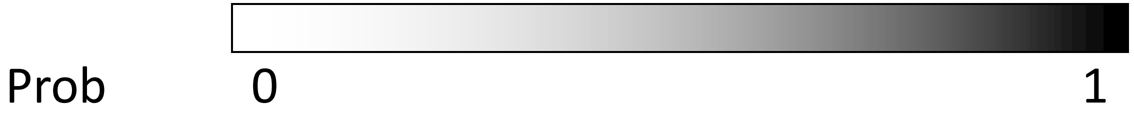
t=2

Slide from Berkeley AI course

Example: Robot Localization



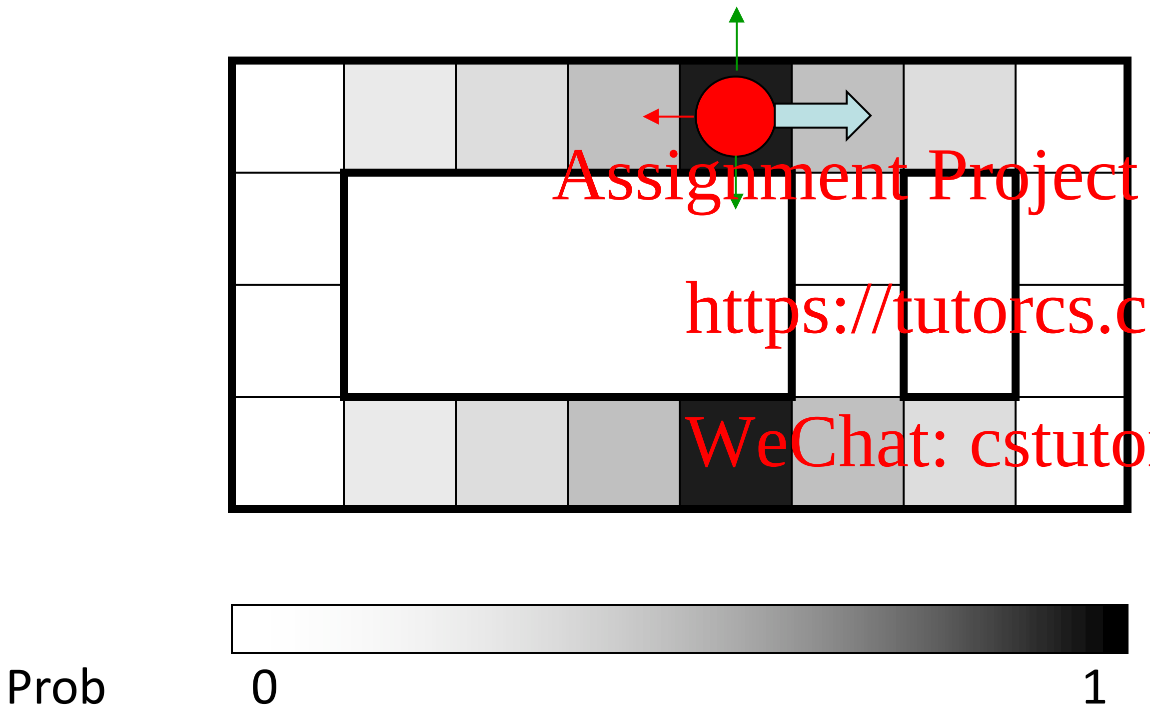


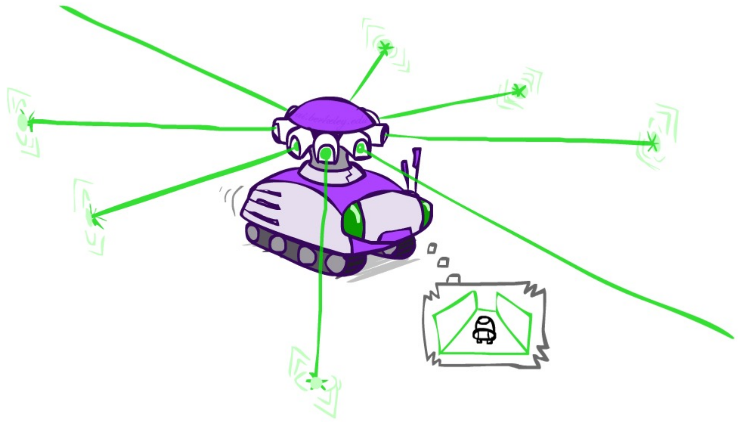


t=3

Slide from Berkeley AI course

Example: Robot Localization

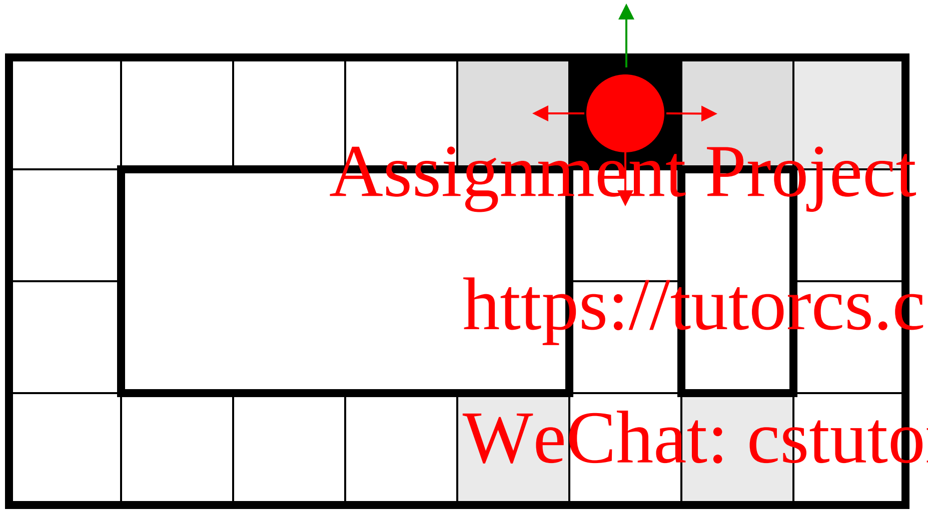


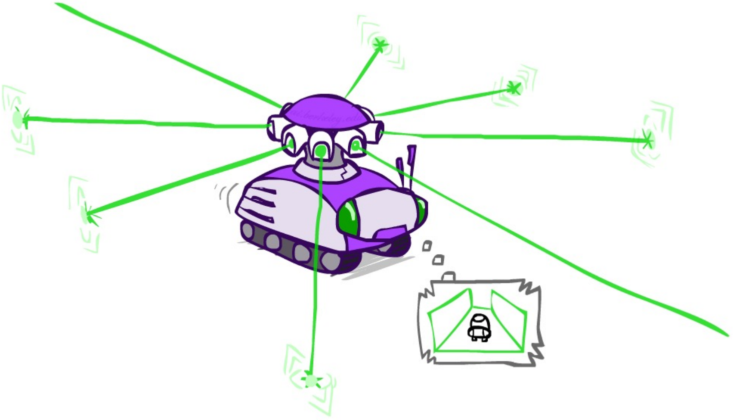


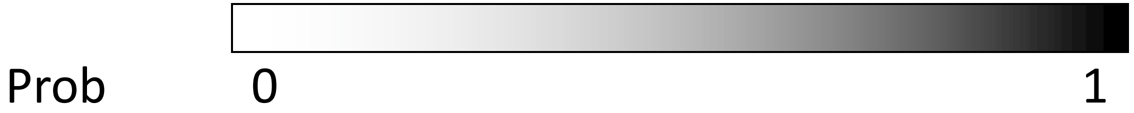
t=4

Slide from Berkeley AI course

Example: Robot Localization



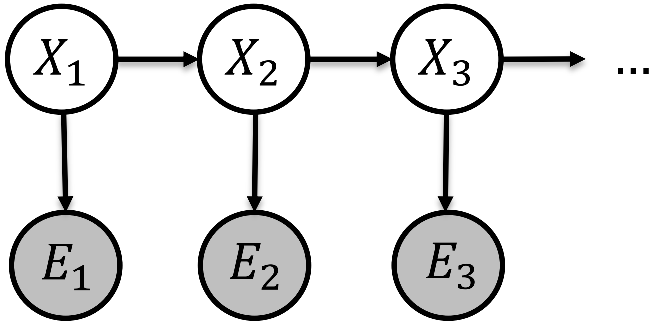




t=5

Slide from Berkeley AI course

Most Probable Explanation (MPE)

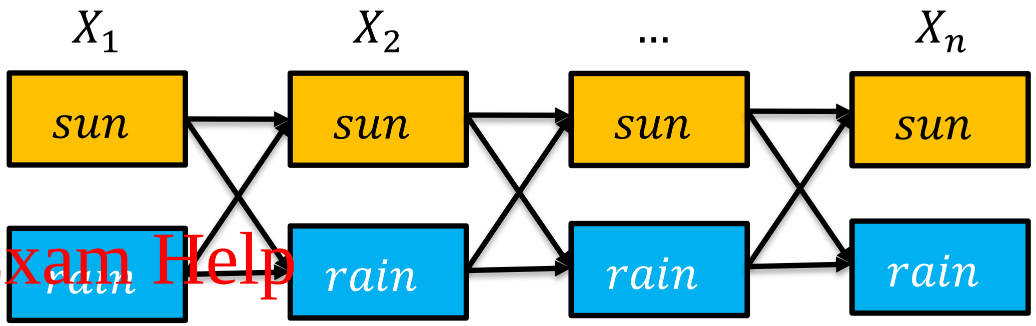


- The forward algorithm tracks the probability of the states
  - These probabilities are updates with as time passes and we observe evidence
- A different task is to provide the most likely explanation
  - Considering all possible state combinations, which one has the highest probability considering the evidence
  - Therefore, we want to compute

$$argmax_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

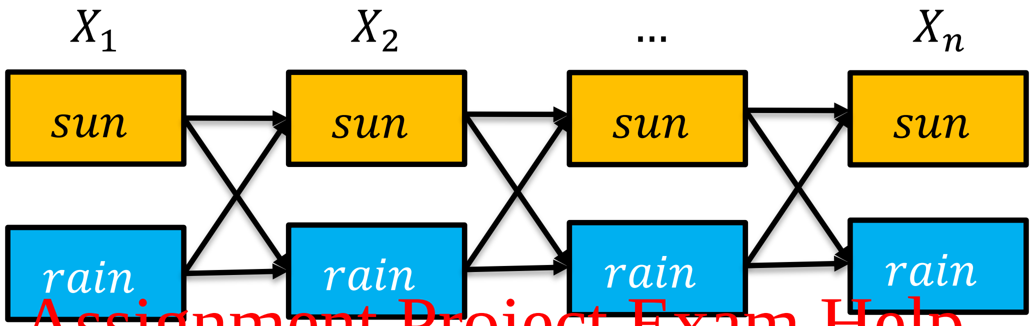
State Trellis

- A state trellis is a graph that illustrates the state transition over time
  - Each arc represents a time passage/evidence observation with weight $P(x_t|x_{t-1})P(e_t|x_t)$
- A *path* is a sequence of states
  - The product of weights on a path is the sequence probability according to the evidence
  - The forward algorithm computes sums of paths probabilities that end in a same state, such as $X_n = sun$
  - We will see now the *Viterbi algorithm* that computes the path with highest probability



State Trellis

Forward and Viterbi Algorithms



State Trellis

- The forward algorithm computes the *sum* of the path probabilities that lead to the same final state

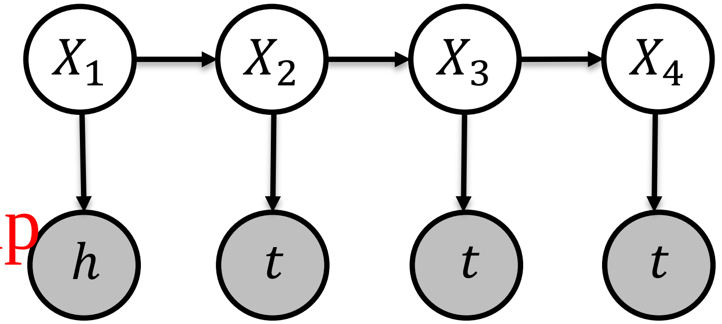
- The Viterbi algorithm computes the *maximum* of the path probabilities that lead to the same final state

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

$$m[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t|e_{1:t}) = P(e_t|X_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m[x_{t-1}]$$

Viterbi Algorithm

- Consider we have two unfair coins, c_1 and c_2
  - Someone flips the coins sequentially, but we do not know which one. We only observe the outcomes *heads* or *tails*
  - But we know that c_1 has a higher probability of *heads* and c_2 of *tails*
  - Also, the person has a preference to keep the same coin and he/she starts with a coin chosen randomly with equal probabilities

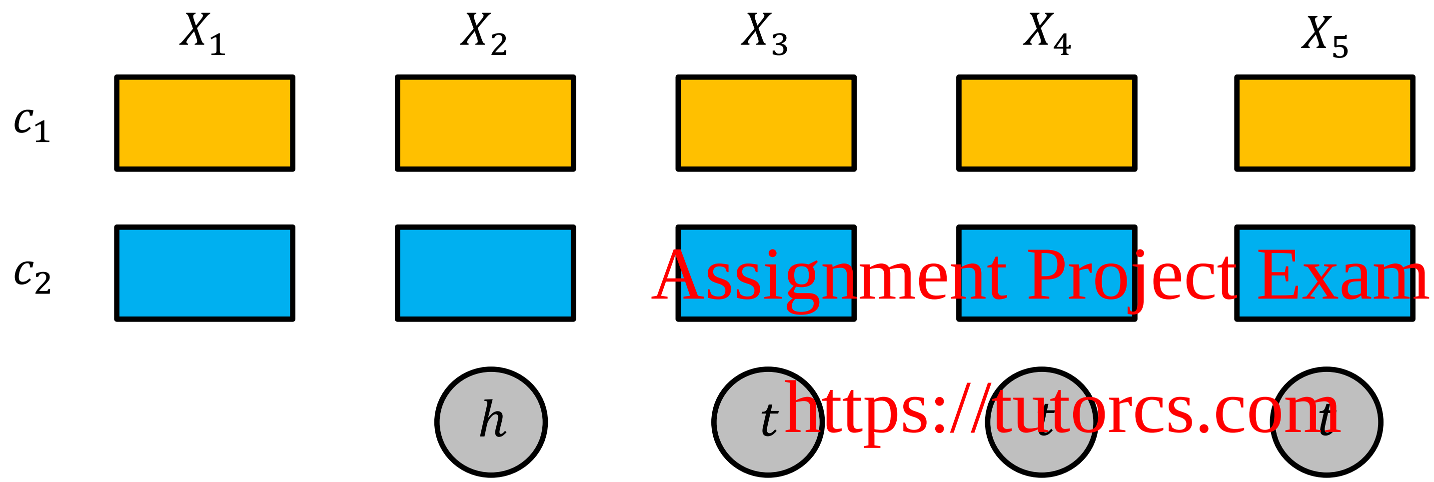


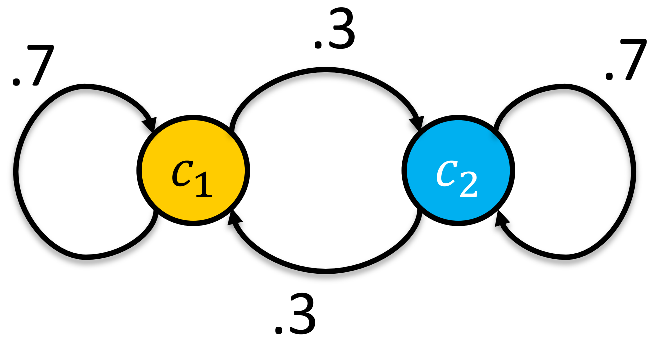
X_1	$P(X_1)$
c_1	.5
c_2	.5

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
c_1	c_1	.7
c_1	c_2	.3
c_2	c_1	.3
c_2	c_2	.7

X_t	E_t	$P(E_t \mid X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8

Viterbi Algorithm



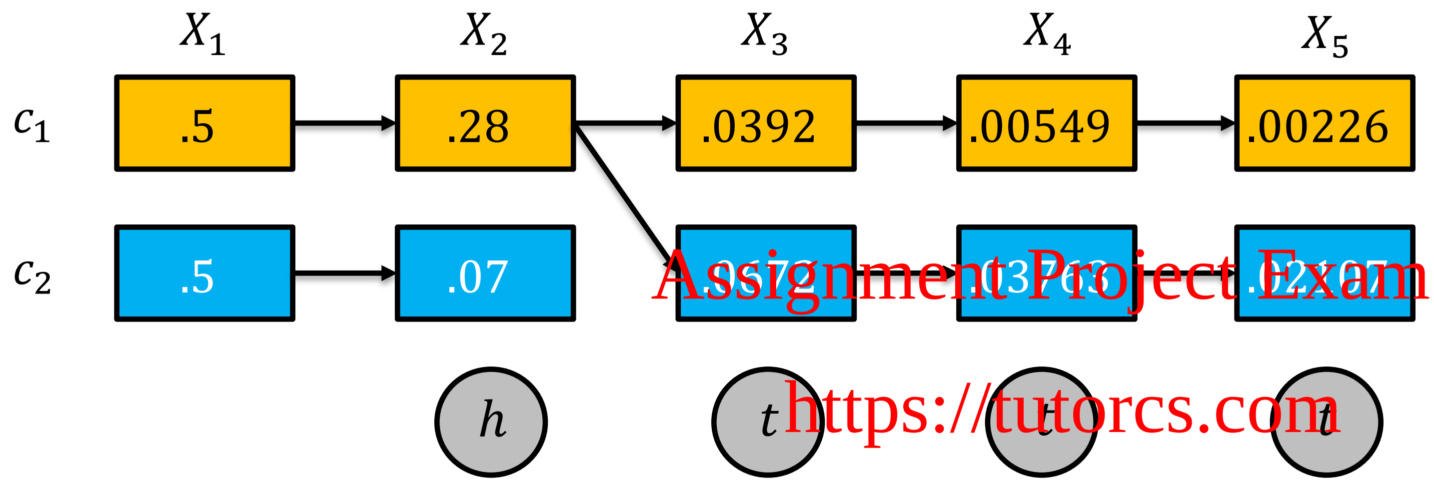


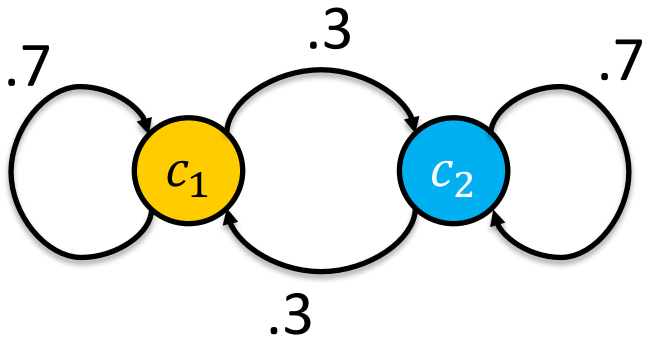
X_1	$P(X_1)$
c_1	.5
c_2	.5

X_t	E_t	$P(E_t\|X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8

$$m[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t | e_{1:t})$$
$$= P(e_t|X_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m[x_{t-1}]$$

Viterbi Algorithm





X_1	$P(X_1)$
c_1	.5
c_2	.5

X_t	E_t	$P(E_t \vert X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8

$$m[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t | e_{1:t})$$
$$= P(e_t|X_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m[x_{t-1}]$$

Viterbi Algorithm

Input: time n, transition probability T, emission probability E, prior probability of states $P(X_1)$, sequence of observations $\{e_2, \ldots, e_t\}$

Output: $\max_{x_{1:t-1}} P(x_{1:t-1}, x_t | \boldsymbol{e}_{2:t})$

for each state x **do**

  $m[x, 1] \leftarrow P(X_1 = x)$

for $t \leftarrow 2$ to n **do**

  for each state x_t **do**

    $m[x_t, t] = 0$

    for each state x_{t-1} **do**

      if $m[x_{t-1}, t-1]T(x_t|x_{t-1}) > m[x_t, t]$

        $m[x_t, t] \leftarrow m[x_{t-1}, t-1]T(x_t|x_{t-1})$

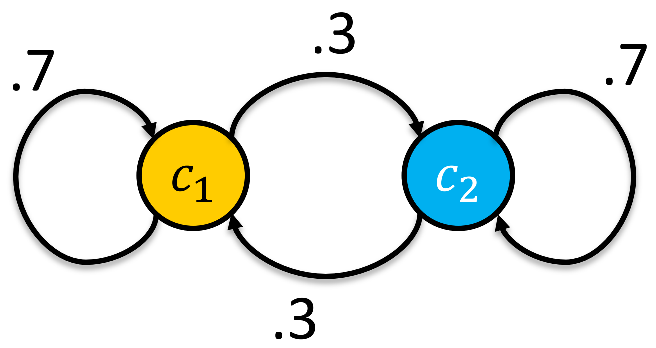
    $m[x_t, t] \leftarrow m[x_t, t]E(e_t|x_t)$

return $p[x, n]$ for all states x

$$O(n|X|^2)$$

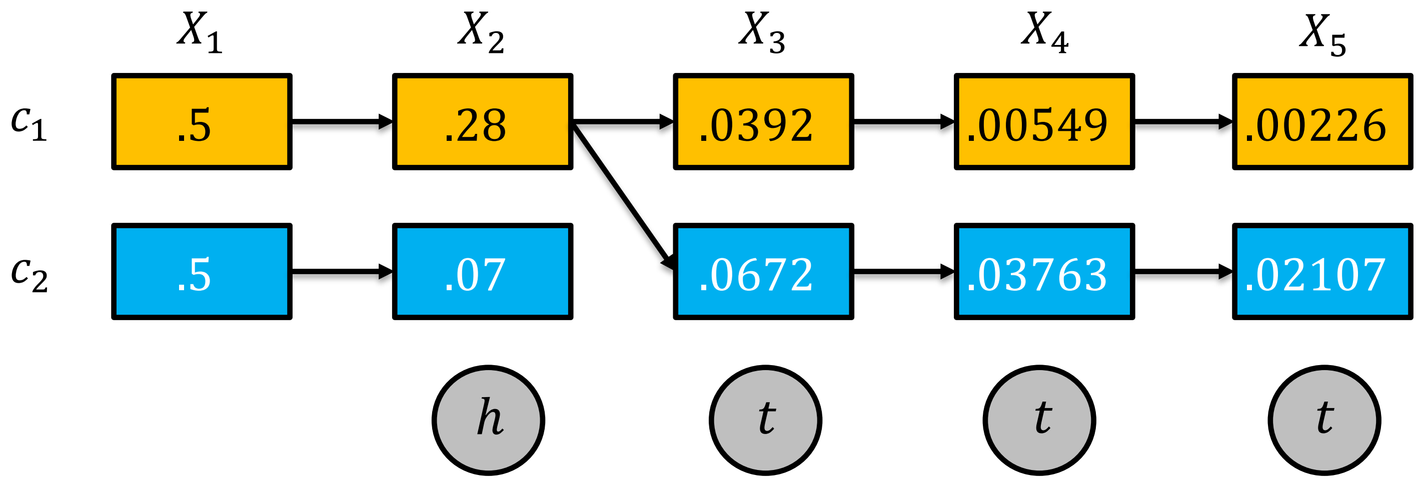
Viterbi Algorithm

- The Viterbi algorithm of the previous slide provides the probability of the most likely sequence
  - However, often we are more interested in the sequence instead of its probability
- There are to common solutions
  - Keep an additional structure pointing to the parent of each node
  - Backtrack the computation from the last node



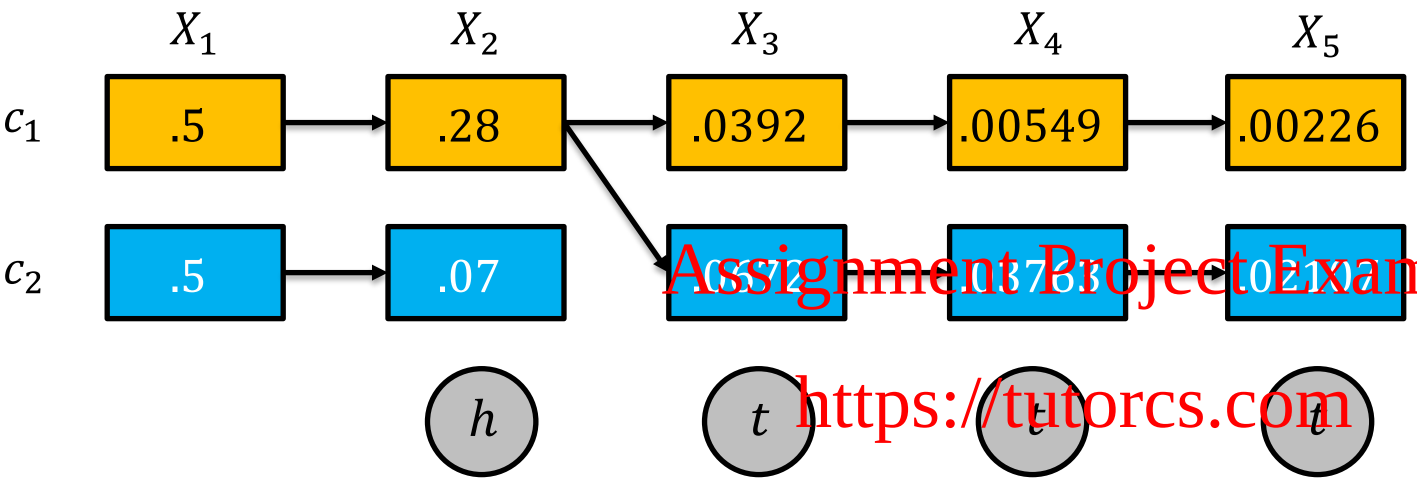
X_1	$P(X_1)$
c_1	.5
c_2	.5

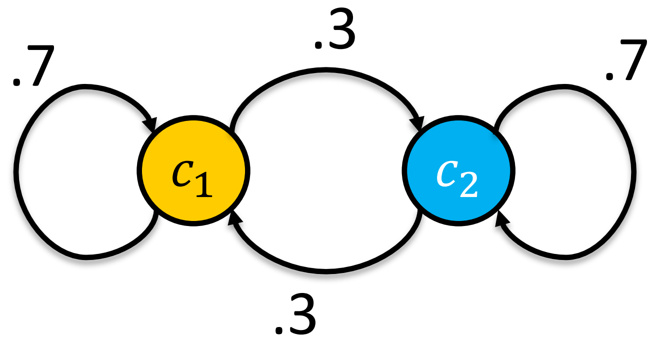
X_t	E_t	$P(E_t \mid X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8



$$
\begin{aligned}
m[x_t] &= \max_{x_{1:t-1}} P(x_{1:t-1}, x_t \mid e_{1:t}) \\
&= P(e_t \mid X_t) \max_{x_{t-1}} P(x_t \mid x_{t-1}) m[x_{t-1}]
\end{aligned}
$$

Viterbi Algorithm: Backtracking Computation





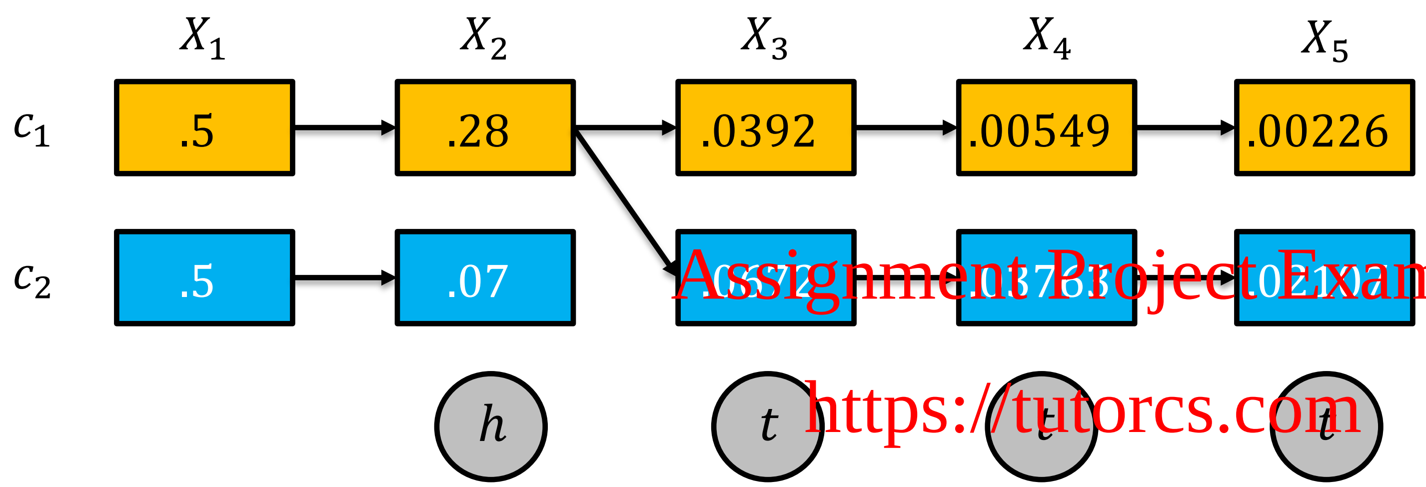
X_1	$P(X_1)$
c_1	.5
c_2	.5

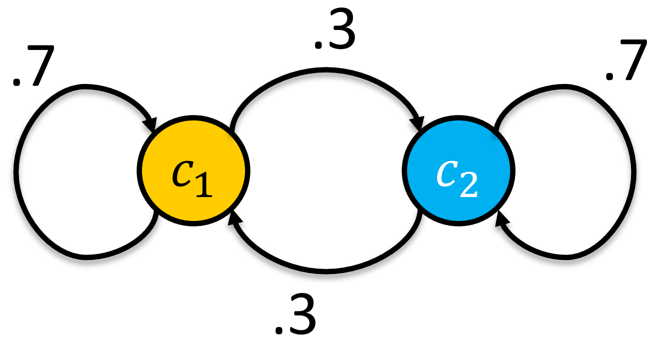
X_t	E_t	$P(E_t \mid X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8

- Repeat
  1. Divide by the probability of evidence
  2. For each state x_{t-1} divide by $P(x_t \mid x_{t-1})$
  3. See which value of x_{t-1} matches the result

$$m[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t \mid e_{1:t})$$
$$= P(e_t \mid X_t) \max_{x_{t-1}} P(x_t \mid x_{t-1}) m[x_{t-1}]$$

Viterbi Algorithm: Backtracking Computation





X_1	$P(X_1)$
c_1	.5
c_2	.5

X_t	E_t	$P(E_t \mid X_t)$
c_1	h	.8
c_1	t	.2
c_2	h	.2
c_2	t	.8

- Repeat
  1. Divide by the probability of evidence
  2. For each state x_{t-1} divide by $P(x_t|x_{t-1})$
  3. See which value of x_{t-1} matches the result

$$m[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t | e_{1:t})$$
$$= P(e_t|X_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m[x_{t-1}]$$

1.

$$\frac{.02107}{.8} = 0.0263375$$

2.

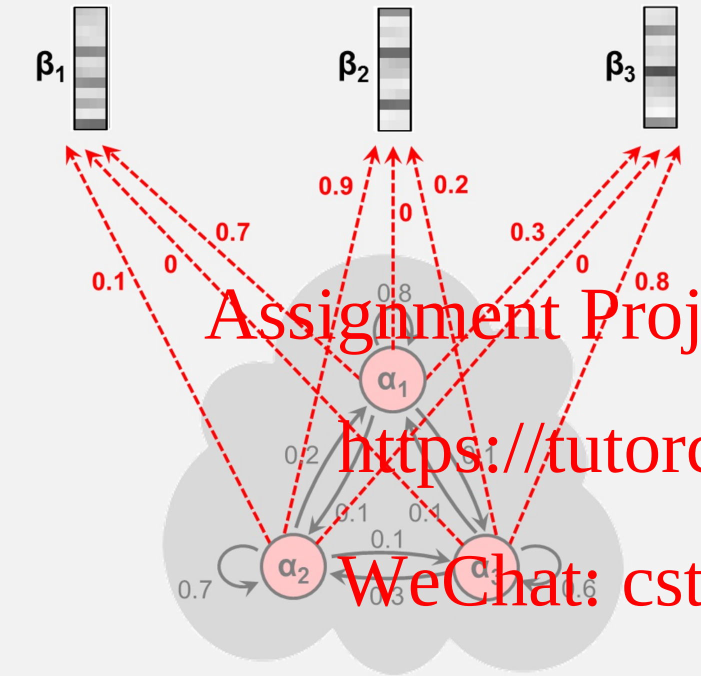
$$x_4 = c_1: \frac{0.0263375}{.3} \approx 0.08779$$

$$x_4 = c_2: \frac{0.0263375}{.7} \approx 0.03763$$

3.

$$x_4 = c_2: 0.03763$$

Viterbi Algorithm: Backtracking Computation



State transition probabilities

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Initial state probabilities

C	α_1	α_2	α_3
	0.6	0.2	0.2

Emission probabilities

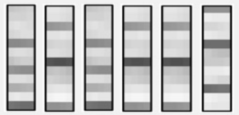
B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Figure 5.28a from [Müller, FMP, Springer 2015]

Input

Observation sequence

$O = (o_1,o_2,o_3,o_4,o_5,o_6)$



Viterbi algorithm

D	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$	$o_6=\beta_2$
α_1	0.4200	0.1008	0.0564	0.0135	0.0033	0
α_2	0.0200	0	0.0010	0	0	0.0006
α_3	0	0.0336	0	0.0045	0.0022	0.0003

E	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$
α_1	$\boxed{1}$	$\boxed{1}$	1	1	1
α_2	1	1	1	1	$\boxed{3}$
α_3	1	3	$\boxed{1}$	$\boxed{3}$	3

$\boxed{i_6 = 2}$

Output

Optimal state sequence

$S^* = (\alpha_1,\alpha_1,\alpha_1,\alpha_3,\alpha_3,\alpha_2)$

Figure 5.28b from [Müller, FMP, Springer 2015]

Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer.

Viterbi Algorithm: Vanishing Probabilities

- Notice the probabilities decrease as we observe more evidence
  - It is intuitive since the number of paths grows exponentially with the sequence size
  - In this example, the probabilities are around 10^{-4} with just 6 steps
  - Long sequences (such as 100 steps) will cause an underflow and the probabilities will become zero
  - We can fix that using log probabilities, similarly to the Naïve Bayes classifier
  - This approach also replaces multiplications by sums

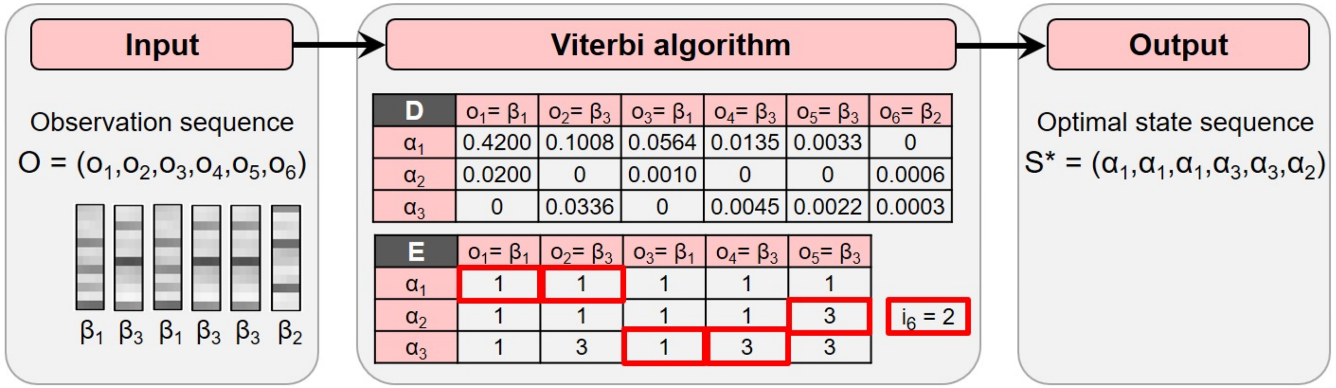


Figure 5.28b from [Müller, FMP, Springer 2015]

Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer.

Viterbi Algorithm: Log Probabilities

Input: time n, transition probability T, emission probability E, prior probability of states $P(X_1)$, sequence of observations $\{e_2, \ldots, e_t\}$

Output: $\max_{x_{1:t-1}} \log P(x_{1:t-1}, x_t | \boldsymbol{e}_{1:t})$

for each state x **do**

$\quad m[x, 1] \leftarrow \log P(X_1 = x)$

for $t \leftarrow 2$ to n **do**

$\quad$ **for each** state x_t **do**

$\quad\quad m[x_t, t] = -\infty$

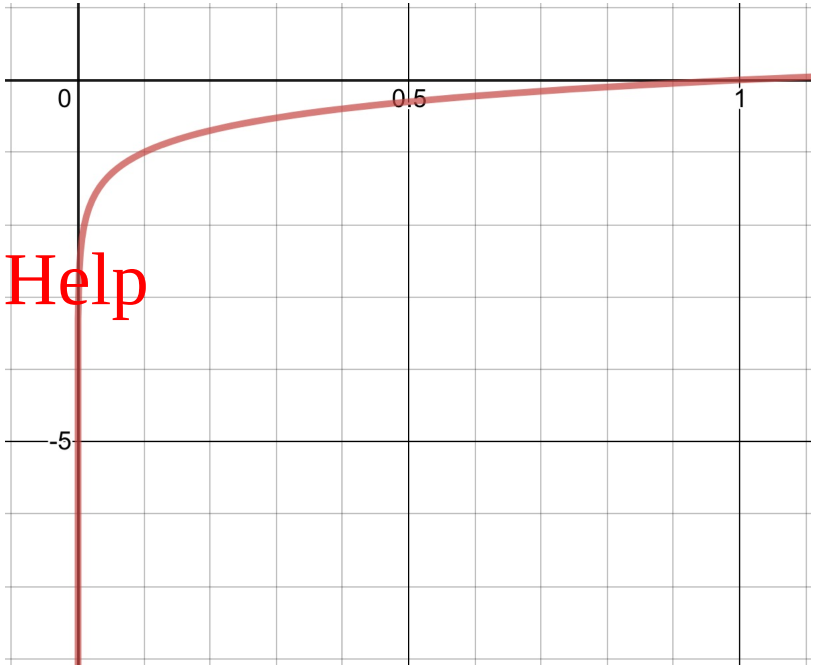
$\quad\quad$ **for each** state x_{t-1} **do**

$\quad\quad\quad$ **if** $m[x_{t-1}, t-1] + \log T(x_t | x_{t-1}) > m[x_t, t]$

$\quad\quad\quad\quad m[x_t, t] \leftarrow m[x_{t-1}, t-1] + \log T(x_t | x_{t-1})$

$\quad\quad m[x_t, t] \leftarrow m[x_t, t] + \log E(e_t | x_t)$

return $p[x, n]$ for all states x

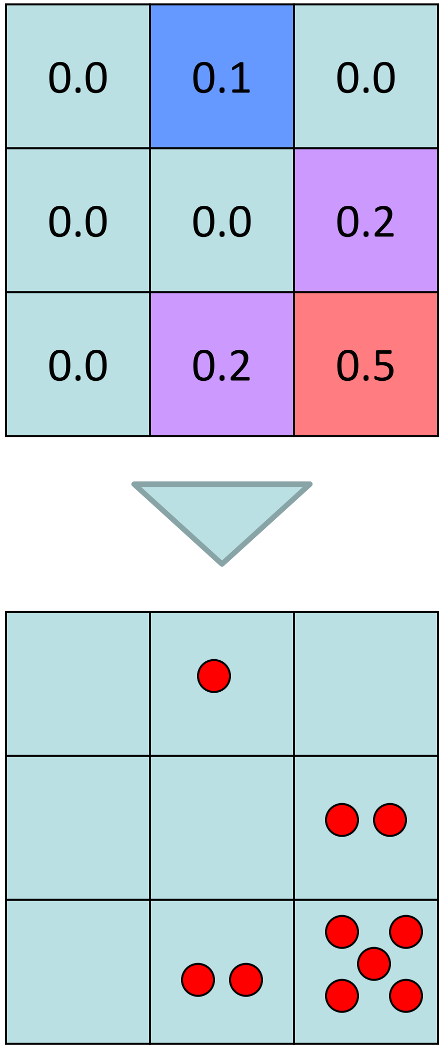


$$m[x_t] = \log \max_{x_{1:t-1}} P(x_{1:t-1}, x_t | e_{1:t-1})$$
$$= \log P(e_t | X_t) + \max_{x_{t-1}} \log P(x_t | x_{t-1}) + m[x_{t-1}]$$

$$O(n|X|^2)$$

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



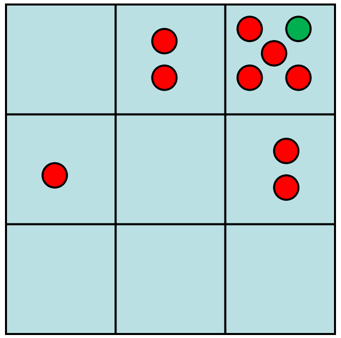
Slide from Berkeley AI course

Representation: Particles

- Our representation of $P(X)$ is now a list of N particles (samples)
  - Generally, $N \ll |X|$
  - Storing map from X to counts would defeat the point

- $P(x)$ approximated by number of particles with value x
  - So, many x may have $P(x) = 0$!
  - More particles, more accuracy

- For now, all particles have a weight of 1



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

Slide from Berkeley AI course

Particle Filtering: Elapse Time

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

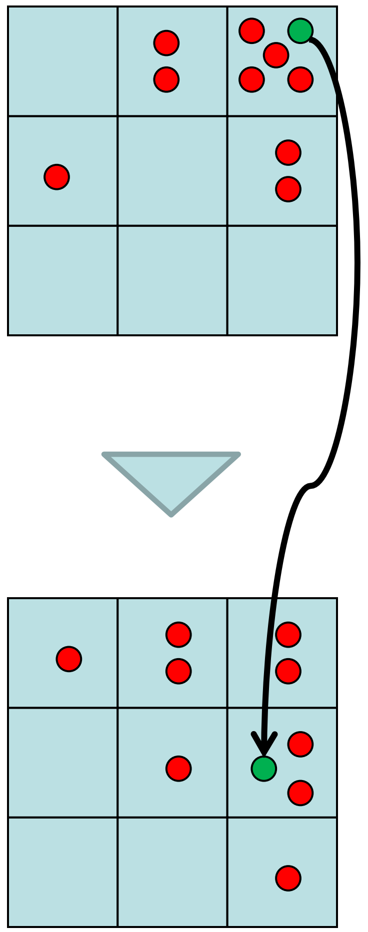
$$x' = \text{sample}(P(X'|x))$$

  - 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:
(1,3)
(1,2)
(1,3)
(2,3)
(1,3)
(2,3)
(2,1)
(1,3)
(1,3)
(1,2)

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



Slide from Berkeley AI course

Particle Filtering: Observe

- Slightly trickier:
  - Don't sample observation, fix it
  - Downweigh samples based on the evidence

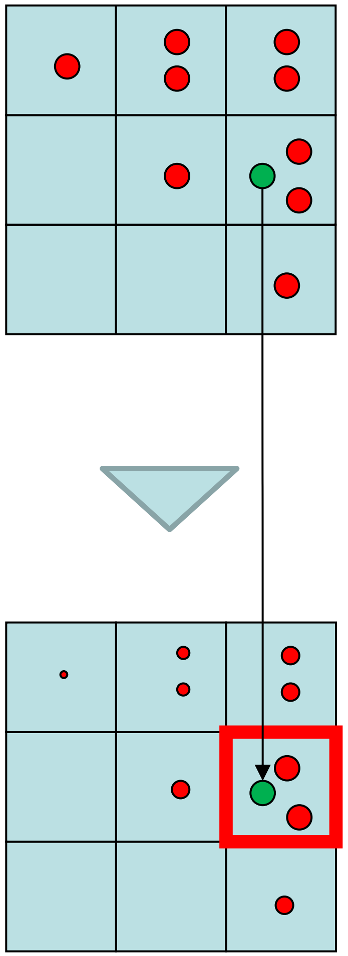
$$w(x) = P(e|x)$$

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

  - As before, the probabilities don't sum to one, since all have been downweighed (in fact they now sum to (N times) an approximation of $P(e)$)

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

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



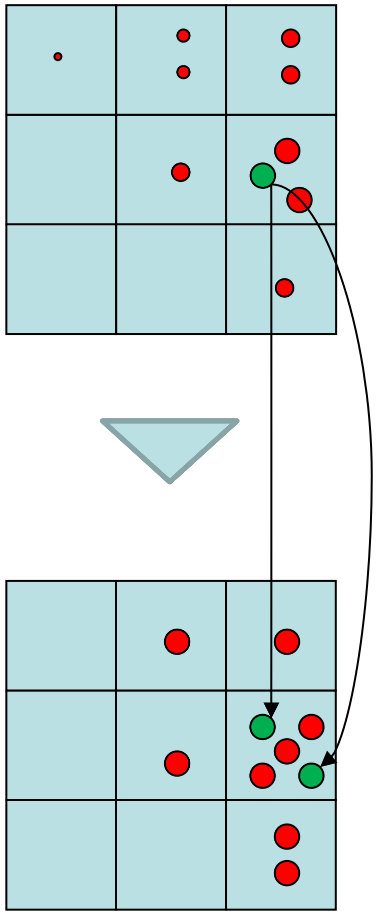
Slide from Berkeley AI course

Particle Filtering: Resample

- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This is equivalent to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

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

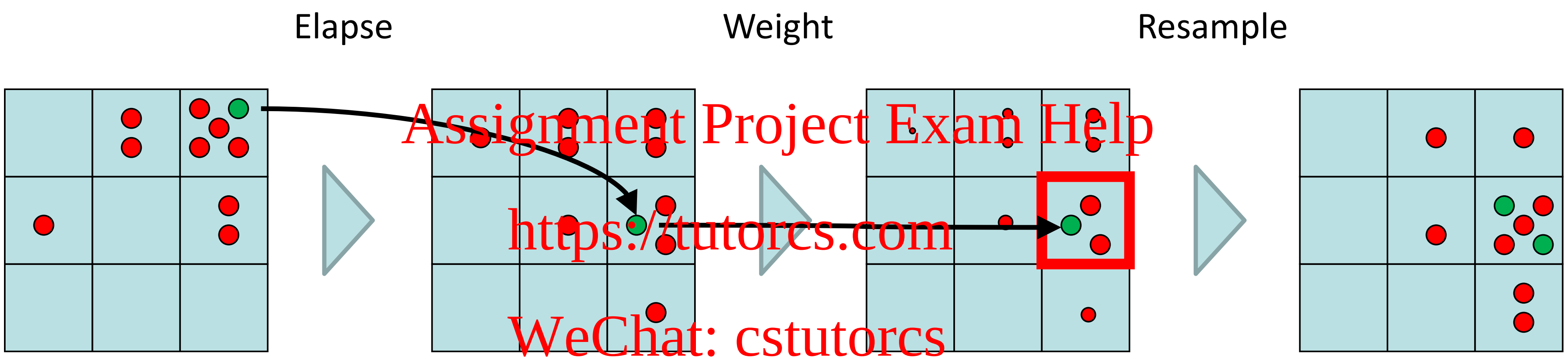
(New) Particles:
(2,3)
(2,2)
(2,3)
(1,2)
(1,3)
(2,3)
(3,3)
(3,3)
(2,3)
(2,3)



Slide from Berkeley AI course

Recap: Particle Filtering

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

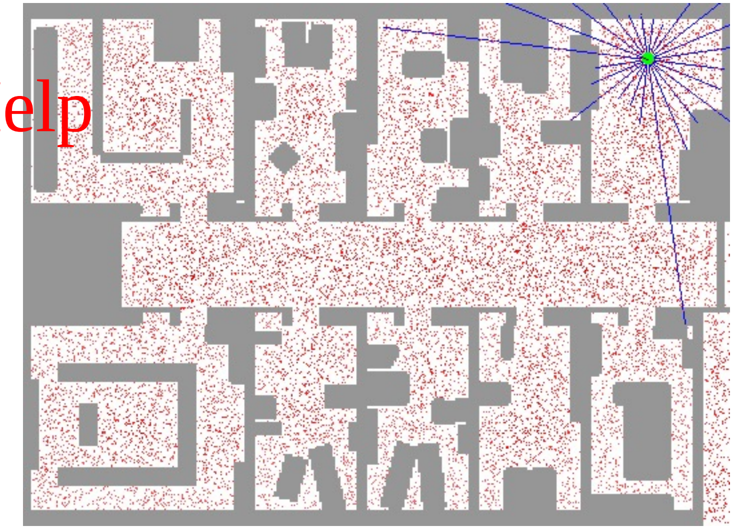


Particles:	Particles:	Particles:	(New) Particles:
(1,3)	(2,3)	(2,3) w=.9	(2,3)
(1,2)	(1,2)	(1,2) w=.2	(2,2)
(1,3)	(2,3)	(2,3) w=.9	(2,3)
(2,3)	(3,3)	(3,3) w=.4	(1,2)
(1,3)	(1,3)	(1,3) w=.4	(1,3)
(2,3)	(2,3)	(2,3) w=.9	(2,3)
(2,1)	(1,1)	(1,1) w=.1	(3,3)
(1,3)	(1,2)	(1,2) w=.2	(3,3)
(1,3)	(1,3)	(1,3) w=.4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

Slide from Berkeley AI course

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



Slide from Berkeley AI course

Conclusion

- Markov chains and Hidden Markov models are simple examples of Dynamic Bayesian networks
  - DBNs are networks that allow us to model changes in time or space
  - Changes in time are specified using transition probabilities
- Markov chains are sequence models
  - It tracks the probability distribution over a series of transitions
  - For many sequences, the probability distribution converges to a stationary distribution
  - The stationary distribution has several applications such as the MCMC algorithms used for approximate inference
- Hidden Markov models are Markov chains with hidden states
  - Those states are never directly observed, but we can make indirect inference through emissions
  - These models are used in several applications such as language and signal processing and robot localisation