Lecture 16

Hidden Markov Models

CS 3630

Recap

Markov Decision Processes (MDPs)

A controlled Markov chain is defined by

- A set of states: $\mathcal{X}$
- A set of actions: $\mathcal{A}$
- A set of transition probabilities: $P(X_{t+1} = x_{t+1} | X_t = x_t, a_t)$
- A reward function: $R: \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$
- A discount factor: γ

The Markov chain is the sequence of random states: $X_0, X_1, \dots, X_n$

The key idea behind Markov chains is that the past and future are completely decoupled if we know the current state. This can be written mathematically as:

$$P(X_{t+1} = x_{t+1} \mid a_0, \dots a_t, X_0 = x_0, \dots, X_t = x_t) = P(X_{t+1} = x_{t+1} | X_t = x_t, a_t)$$

The belief state b_{t+1} is a vector that specifies $P(X_{t+1} = x_{t+1} \mid a_0, \dots a_t)$ for each possible x_{t+1}.

Markov Decision Processes (MDPs)

A controlled Markov chain is defined by

- A set of states: $\mathcal{X}$
- A set of actions: $\mathcal{A}$
- A set of transition probabilities: $P(X_{t+1} = x_{t+1} | X_t = x_t, a_t)$
- A reward function: $R: \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$
- A discount factor: γ

The Markov chain is the sequence of random states: $X_0, X_1, \dots, X_n$

The key idea behind Markov chains is that the past and future are completely decoupled if we know the current state. This can be written mathematically as:

$$P(X_{t+1} = x_{t+1} \mid \cancel{a_0, \dots} a_t, \cancel{X_0 = x_0, \dots,} X_t = x_t) = P(X_{t+1} = x_{t+1} | X_t = x_t, a_t)$$

The belief state b_{t+1} is a vector that specifies $P(X_{t+1} = x_{t+1} \mid a_0, \dots a_t)$ for each possible x_{t+1}.

- *Expected Reward:* $\bar{R}(x, a) = E[R(x, a, X')]$

- *Expected Utility:*

$$\mathrm{E}\left[U\left(a_{1,} \dots, a_n, x_1, X_2, \dots X_{n+1}\right)\right] = \mathrm{E}[R(x_1, a_1, X_2) + \gamma R(X_2, a_2, X_3) + \cdots \gamma^{n-1} R(X_n, a_n, X_{n+1})\ ]$$

- *Policy:* $\pi: \mathcal{X} \rightarrow \mathcal{A}$

- Value function for a policy: $V^\pi(x) = \bar{R}(x, \pi(x))\ + \gamma \sum_{x'} P(x' | x, \pi(x)) V^\pi(x')$

Markov Decision Processes

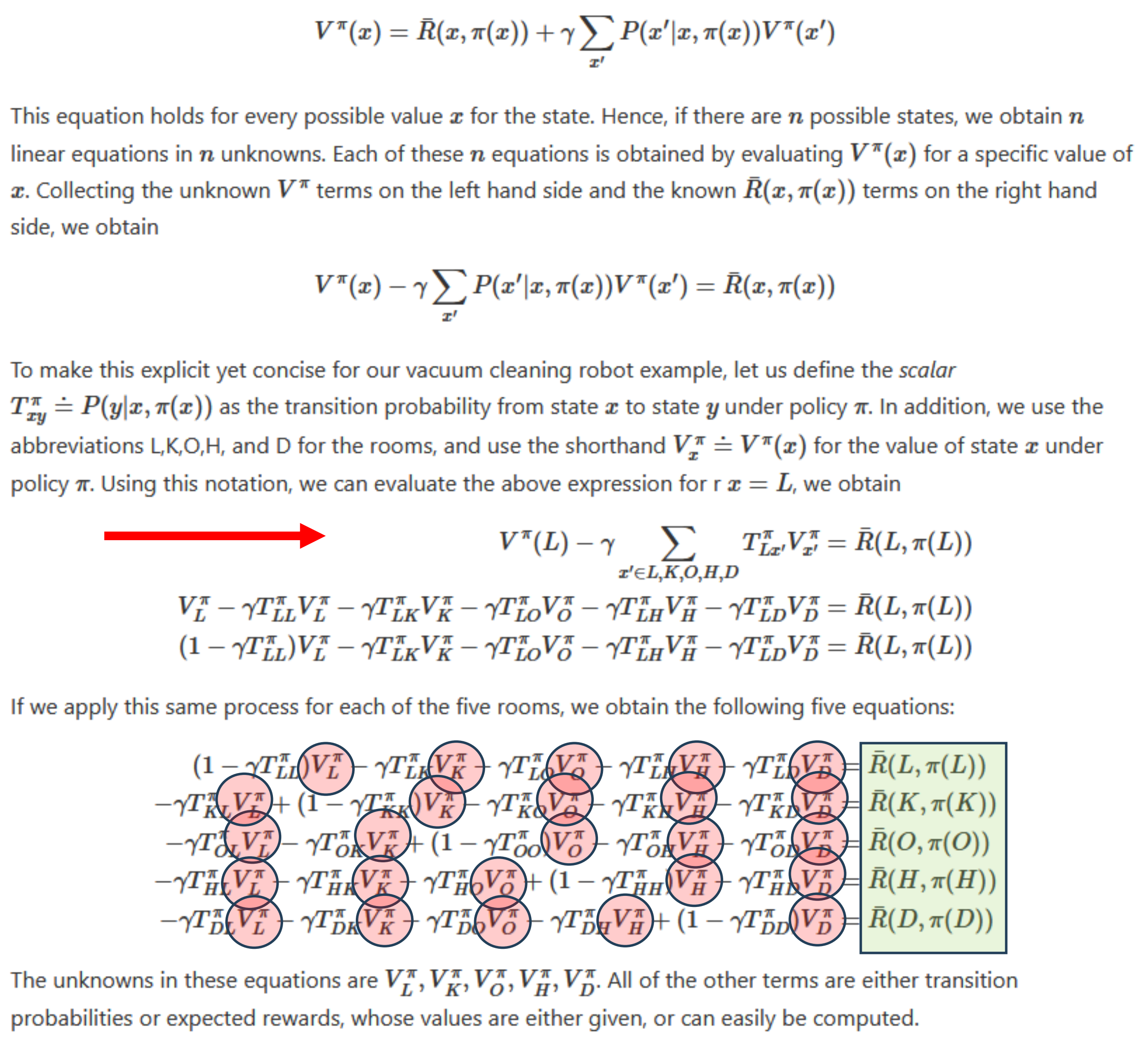
- We have seen controlled Markov chains that are driven by a sequence of actions, $a_0, \ldots a_n$.
- Planning is the process of choosing which actions to perform.
- In order to plan effectively, we need quantitative criteria to evaluate actions and their effects.
- MDPs include a reward function that characterizes the immediate benefit of applying an action.
- Policies describe how to act in a given state.
- The value function characterizes the long-term benefits of a policy.
- We assume that the robot is able to ***know*** its current state with certainty.

➢ ***We will see how to define reward functions and use these to compute optimal policies for MDPs.***

Exact Computation for V^π

$$T^\pi_{xy} \triangleq P(X' = y | X = x, \pi(x))$$

$$V^\pi_x \triangleq V^\pi(x)$$



$$V^\pi(x) = \bar{R}(x, \pi(x)) + \gamma \sum_{x'} P(x'|x, \pi(x)) V^\pi(x')$$

This equation holds for every possible value x for the state. Hence, if there are n possible states, we obtain n linear equations in n unknowns. Each of these n equations is obtained by evaluating $V^\pi(x)$ for a specific value of x. Collecting the unknown V^π terms on the left hand side and the known $\bar{R}(x, \pi(x))$ terms on the right hand side, we obtain

$$V^\pi(x) - \gamma \sum_{x'} P(x'|x, \pi(x)) V^\pi(x') = \bar{R}(x, \pi(x))$$

To make this explicit yet concise for our vacuum cleaning robot example, let us define the *scalar* $T^\pi_{xy} \doteq P(y|x, \pi(x))$ as the transition probability from state x to state y under policy π. In addition, we use the abbreviations L,K,O,H, and D for the rooms, and use the shorthand $V^\pi_x \doteq V^\pi(x)$ for the value of state x under policy π. Using this notation, we can evaluate the above expression for r $x = L$, we obtain

$$\longrightarrow \quad V^\pi(L) - \gamma \sum_{x' \in L,K,O,H,D} T^\pi_{Lx'} V^\pi_{x'} = \bar{R}(L, \pi(L))$$

$$V^\pi_L - \gamma T^\pi_{LL} V^\pi_L - \gamma T^\pi_{LK} V^\pi_K - \gamma T^\pi_{LO} V^\pi_O - \gamma T^\pi_{LH} V^\pi_H - \gamma T^\pi_{LD} V^\pi_D = \bar{R}(L, \pi(L))$$

$$(1 - \gamma T^\pi_{LL}) V^\pi_L - \gamma T^\pi_{LK} V^\pi_K - \gamma T^\pi_{LO} V^\pi_O - \gamma T^\pi_{LH} V^\pi_H - \gamma T^\pi_{LD} V^\pi_D = \bar{R}(L, \pi(L))$$

If we apply this same process for each of the five rooms, we obtain the following five equations:

$$\begin{aligned}
(1 - \gamma T^\pi_{LL}) V^\pi_L - \gamma T^\pi_{LK} V^\pi_K - \gamma T^\pi_{LO} V^\pi_O - \gamma T^\pi_{LH} V^\pi_H - \gamma T^\pi_{LD} V^\pi_D &= \bar{R}(L, \pi(L)) \\
-\gamma T^\pi_{KL} V^\pi_L + (1 - \gamma T^\pi_{KK}) V^\pi_K - \gamma T^\pi_{KO} V^\pi_O - \gamma T^\pi_{KH} V^\pi_H - \gamma T^\pi_{KD} V^\pi_D &= \bar{R}(K, \pi(K)) \\
-\gamma T^\pi_{OL} V^\pi_L - \gamma T^\pi_{OK} V^\pi_K + (1 - \gamma T^\pi_{OO}) V^\pi_O - \gamma T^\pi_{OH} V^\pi_H - \gamma T^\pi_{OD} V^\pi_D &= \bar{R}(O, \pi(O)) \\
-\gamma T^\pi_{HL} V^\pi_L - \gamma T^\pi_{HK} V^\pi_K - \gamma T^\pi_{HO} V^\pi_O + (1 - \gamma T^\pi_{HH}) V^\pi_H - \gamma T^\pi_{HD} V^\pi_D &= \bar{R}(H, \pi(H)) \\
-\gamma T^\pi_{DL} V^\pi_L - \gamma T^\pi_{DK} V^\pi_K - \gamma T^\pi_{DO} V^\pi_O - \gamma T^\pi_{DH} V^\pi_H + (1 - \gamma T^\pi_{DD}) V^\pi_D &= \bar{R}(D, \pi(D))
\end{aligned}$$

The unknowns in these equations are $V^\pi_L, V^\pi_K, V^\pi_O, V^\pi_H, V^\pi_D$. All of the other terms are either transition probabilities or expected rewards, whose values are either given, or can easily be computed.

$$\begin{bmatrix} (1-\gamma)T^\pi_{LL} & \cdots & -\gamma T^\pi_{LD} \\ \vdots & \ddots & \vdots \\ -\gamma T^\pi_{DL} & \cdots & (1-\gamma)T^\pi_{DD} \end{bmatrix} \begin{bmatrix} V^\pi_L \\ V^\pi_K \\ V^\pi_O \\ V^\pi_H \\ V^\pi_D \end{bmatrix} = \begin{bmatrix} \bar{R}(L, \pi(L)) \\ \bar{R}(K, \pi(K)) \\ \bar{R}(O, \pi(O)) \\ \bar{R}(H, \pi(H)) \\ \bar{R}(D, \pi(D)) \end{bmatrix}$$

The Bellman Equation

Principle of Optimality: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

$$V^*(x) = \max_{\pi}\left\{\bar{R}(x,\pi(x)) + \gamma \sum_{x'} P(x'|x,\pi(x))V^{\pi}(x')\right\}$$

$\pi(x) = a$ $\pi(x) = a$ ***Principle of Optimality***

$$= \max_{a}\left\{\bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a)V^*(x')\right\}$$

R. Bellman and R. Kalaba, "Dynamic programming and adaptive processes: Mathematical foundation," in IRE Transactions on Automatic Control, vol. AC-5, no. 1, pp. 5-10, Jan. 1960, doi: 10.1109/TAC.1960.6429288.

The Bellman Equation

Another look:

$$V^*(x) = \max_a \left\{ \bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a) V^*(x') \right\}$$

Sub-solution in the recursion

- $V^*(x)$ is on both sides of the equal sign → recursive definition!
- This is ***not*** a linear equation, because ***max is not a linear operation***!

Optimal Policy

Given the $\boldsymbol{V^*(x)}$, computing the optimal policy is a straightforward optimization:

$$\boldsymbol{\pi^*(x) = \arg\max_a \left\{ \bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a)) V^*(x') \right\}}$$

For convenience, we define the $\boldsymbol{Q^*}$ function as

$$\boldsymbol{Q^*(x,a) = \bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a)) V^*(x')}$$

and we can write the optimal policy as:

$$\boldsymbol{\pi^*(x) = \arg\max_a Q^*(x,a)}$$

Policy Iteration

Start with a random policy π^0, and repeat until convergence:

1. Compute the value function V^{π^k}
2. Improve the policy for each state x using the update rule:

$$\boldsymbol{\pi^{k+1}(x) \leftarrow \underset{a}{arg\,max}\left\{\bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a)V^{\pi^k}(x')\right\}}$$

Improve the
current guess for $\boldsymbol{\pi}^*$

The current guess for $\boldsymbol{\pi}^*$

Value Iteration

Start with a random value function V^0, and repeat until convergence:

- Improve the value function V^k using the update rule:

$$\boldsymbol{V^{k+1}(x) \leftarrow \max_a \left\{ \bar{R}(x,a) + \gamma \sum_{x'} P(x'|x,a)) V^k(x') \right\}}$$

Improve the current guess for $\boldsymbol{V}^*$

The current guess for $\boldsymbol{V}^*$

Note:

- ***For policy iteration we used arg max and selected an action, which implicitly updates the value function.***
- ***For value iteration, we use max, and update directly the value function.***

Sensing

- For the trash sorting robot, we had multiple sensors, and their measurements were conditionally independent (given state).
- We could combine those measurements using Bayes to formulate state estimates.
- For the vacuum cleaning robot, we'll use a single sensor that has only three possible outputs: ***not very powerful***.
- We'll take measurements at each time step, and combine these with the robot's knowledge about its actions and the environment to make inferences about state.
- Bayes networks – and various special cases of Bayes nets – will be the key inference tool.

Trash Sorting Sensors

- Three sensors (weight, conductivity, vision-classifier).
- At any time t, collect measurements from the three sensors: z_t^w, z_t^c, z_t^v and use Bayes to compute $P(X_t = x \mid z_t^w, z_t^c, z_t^v)$.
- Measurements are conditionally independent given state, which gives a nice computational simplification after applying Bayes.

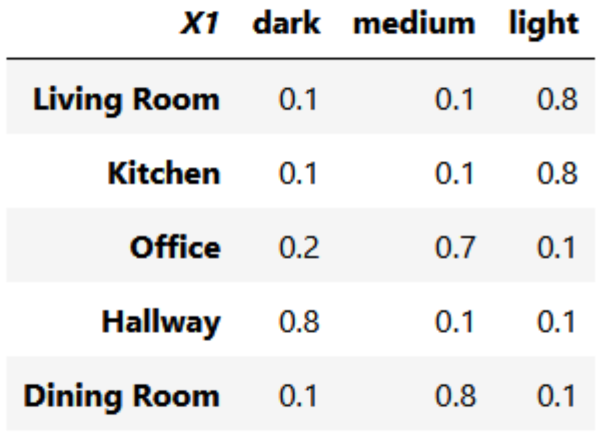
➢ The passing of time was irrelevant – each new sensor measurement was for a new piece of trash:

- Completely independent of previous measurements
- Completely independent of previous actions
- Completely independent of previous states

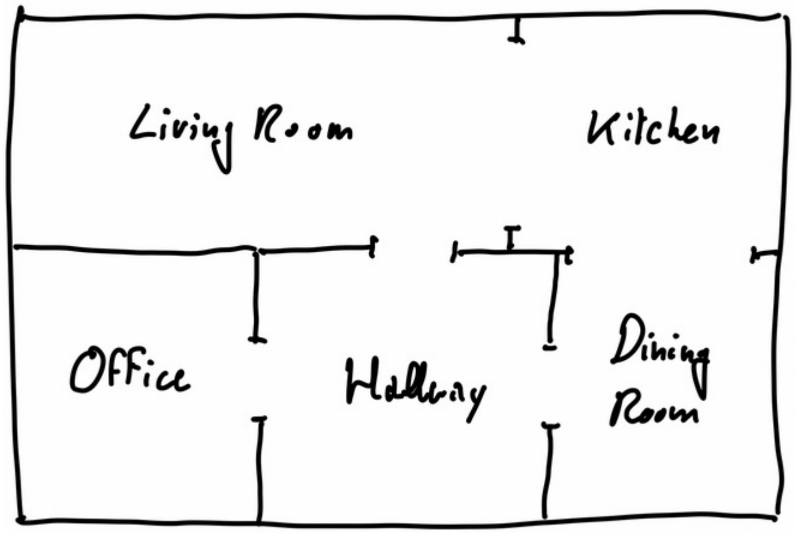
This is not the case for our vacuuming robot!

Vacuuming robot sensor

- A single sensor that detects light levels, and returns a measurement z:
    - Bright, $z = 2$
    - Medium, $z = 1$
    - Dark, $z = 0$



X1	**dark**	**medium**	**light**
Living Room	0.1	0.1	0.8
Kitchen	0.1	0.1	0.8
Office	0.2	0.7	0.1
Hallway	0.8	0.1	0.1
Dining Room	0.1	0.8	0.1



- Sun is to the south, so plenty of light for living room and kitchen.
- Office and Dining room are poorly lit via windows.
- Hallway has no windows, and is always dark.

- ***For Hallway, $(z = 0 \,|H) = 0.8$, MLE will do the job!***
- ***For $z = 1, z = 2$, there's really no way to uniquely identify state from one measurement.***

Exploiting History

- Suppose we observe a sequence of measurements and actions:

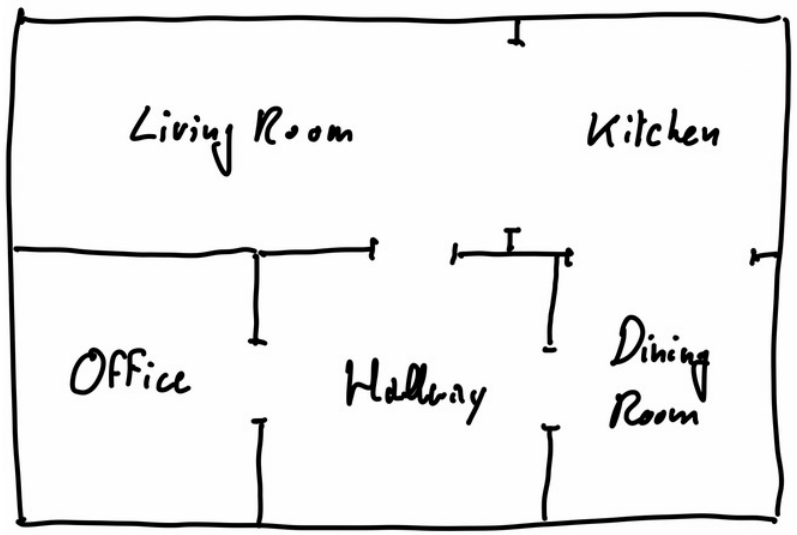
$$z_1 = 0, a_1 = up, z_2 = 2$$

➢ It seems likely that $x_1 = H, x_2 = L$

- Suppose we observe a sequence of measurements and actions:

$$z_1 = 1, a_1 = right$$
$$z_2 = 0, a_2 = right$$
$$z_3 = 1$$

➢ It seems likely that $x_1 = O, x_2 = H, x_3 = D$

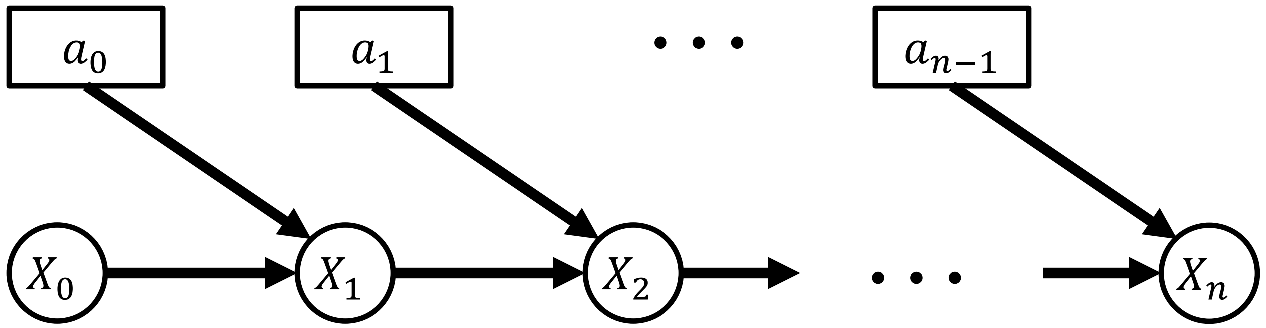


These examples illustrate the basic idea, but these examples are really simple.

How do we formalize/generalize this into a sensor model that accounts for actions and measurements as time sequences?

Bayes Networks

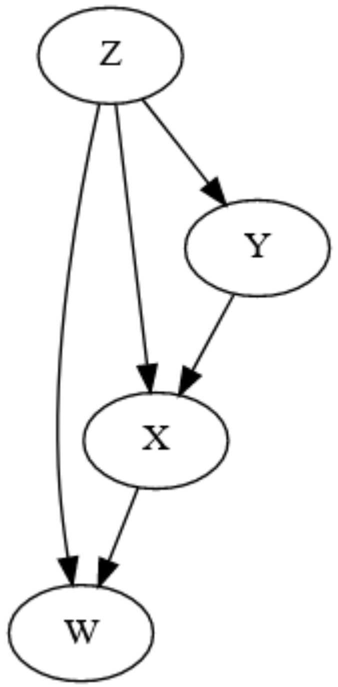
In the past, we have seen graphical models for various sorts of Markov chains:



These models are special cases of the more general Bayesian Networks (Bayes nets):

- Directed Acyclic Graph (DAG)
- For conditional probability $P(X|Y_1, \dots, Y_m)$ there are directed edges from each of Y_i to X.
- There are no other edges in the graph.

Bayes Nets



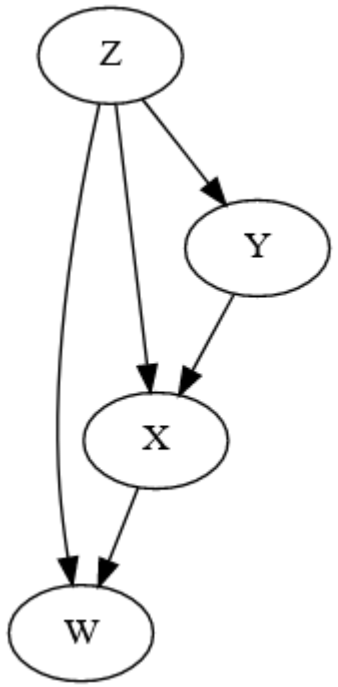
This network represents several conditional probability relationships:

- $P(W|X,Z)$
- $P(X|Y,Z)$
- $P(Y|Z)$
- $P(Z)$

Perhaps more importantly, Bayes nets explicitly encode ***conditional independence*** relationships:

- W is conditionally independent of Y given X

The (first) Magic of Bayes Nets



For a Bayes net with variables $X_1 \dots X_n$, the joint distribution is given by:

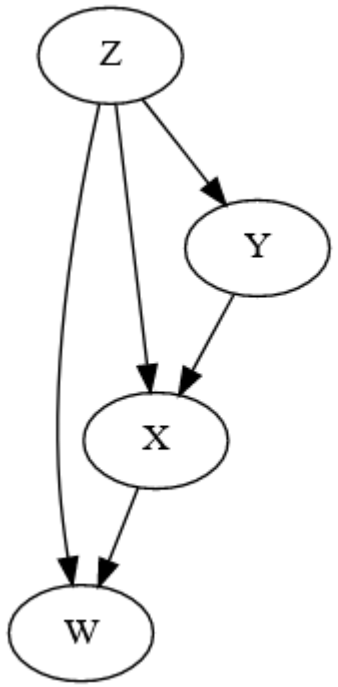
$$\boldsymbol{P(X_1 \dots X_n) = \prod_i P(X_i | \mathcal{P}(X_i))}$$

where $\boldsymbol{\mathcal{P}(X_i)}$ denotes the set of parents of node $\boldsymbol{X_i}$

For this specific network, the joint distribution is given by

$$\boldsymbol{P(W, X, Y, Z) = P(W|X, Z) P(X|Y, Z) P(Y|Z) P(Z)}$$

The (first) Magic of Bayes Nets



We can see why this works (for this example) by expanding the chain rule for joint probability distributions:

$$P(W,X,Y,Z) = P(W|X,Y,Z)P(X|Y,Z)P(Y|Z)P(Z)$$

But from the topology of the Bayes net, we know

$$P(W|X,Y,Z) = P(W|X,Z)$$

Making this substitution, we arrive to the desired result:

$$P(W,X,Y,Z) = P(W|X,Z)P(X|Y,Z)P(Y|Z)P(Z)$$

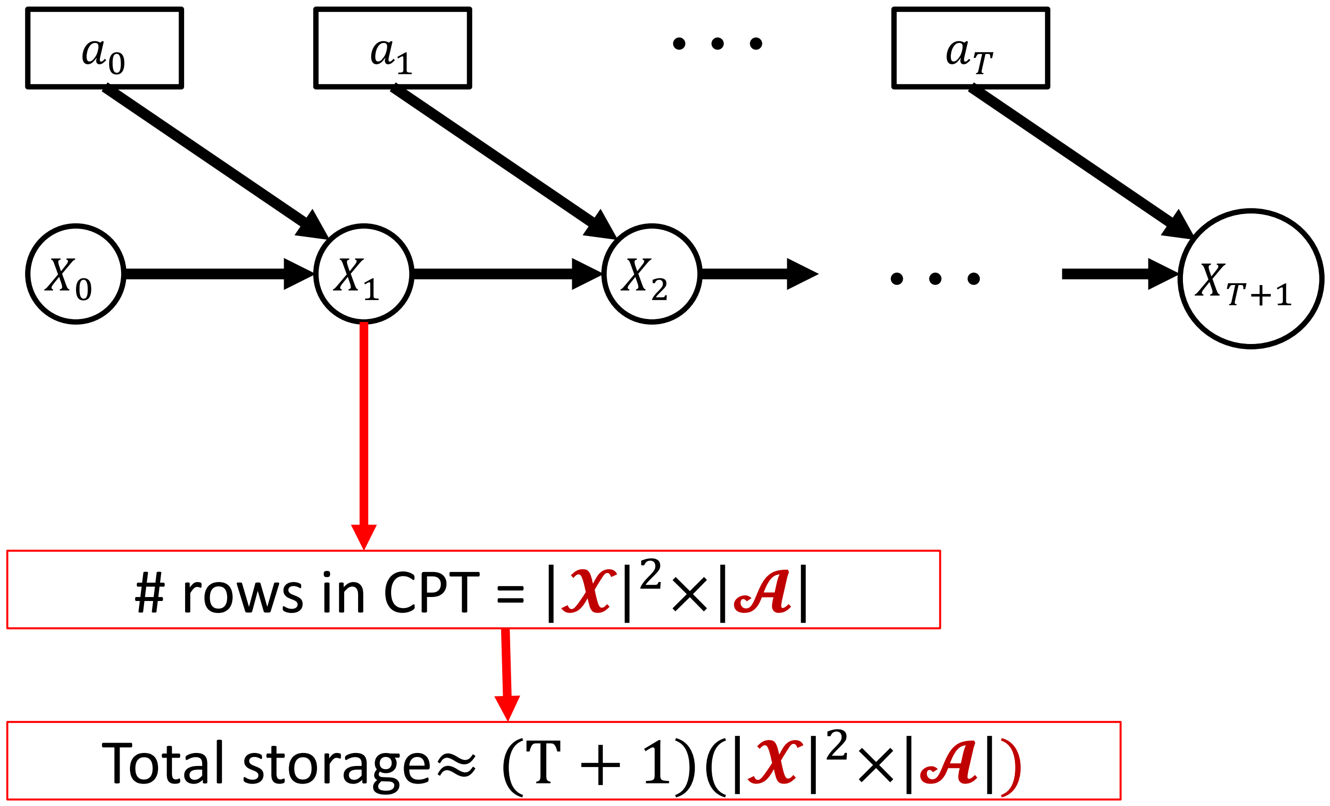
More Magic of Bayes Nets

How difficult would it be to explicitly encode the joint distributions for our vacuum cleaning robot?

Suppose we consider $X_1, \dots X_{T+1}$, and we want to encode $P(X_1, \dots X_{T+1})$

X_1	X_1	$\cdots$	X_T	X_{T+1}	$P(X_1, \dots X_{T+1})$
L	L		L	L	
L	L		L	K	
			L	O	
			L	H	
			L	D	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Number of rows = $|\mathcal{X}|^{T+1}$

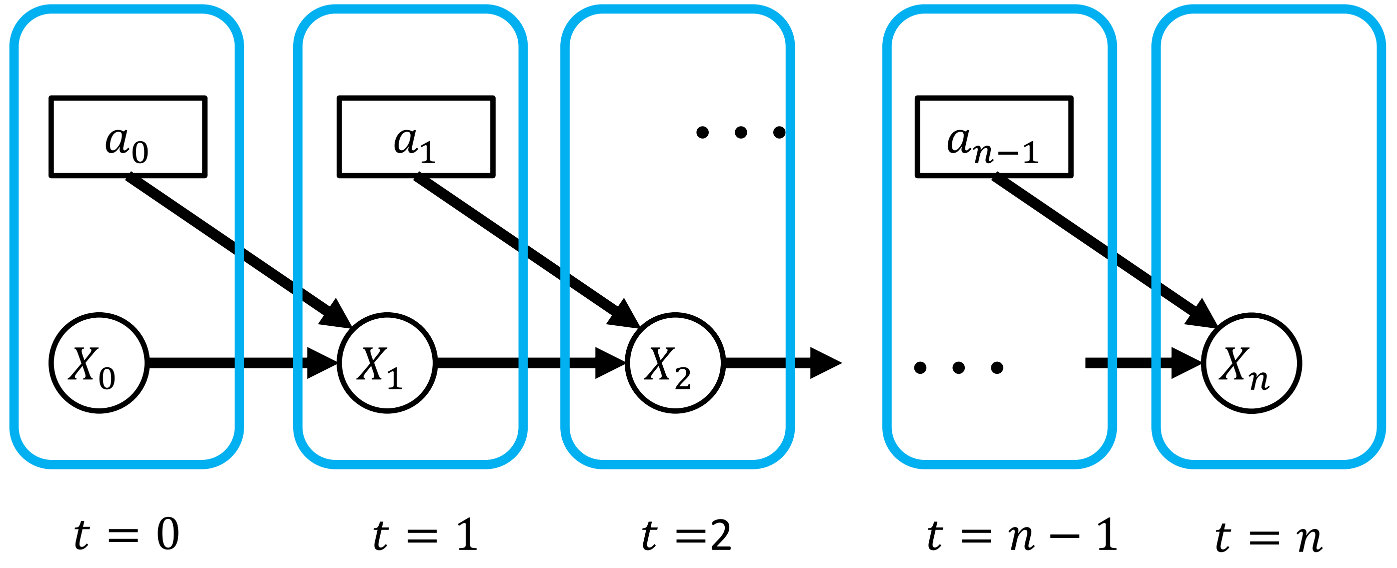


# rows in CPT = $|\mathcal{X}|^2 \times |\mathcal{A}|$

Total storage $\approx (T+1)(|\mathcal{X}|^2 \times |\mathcal{A}|)$

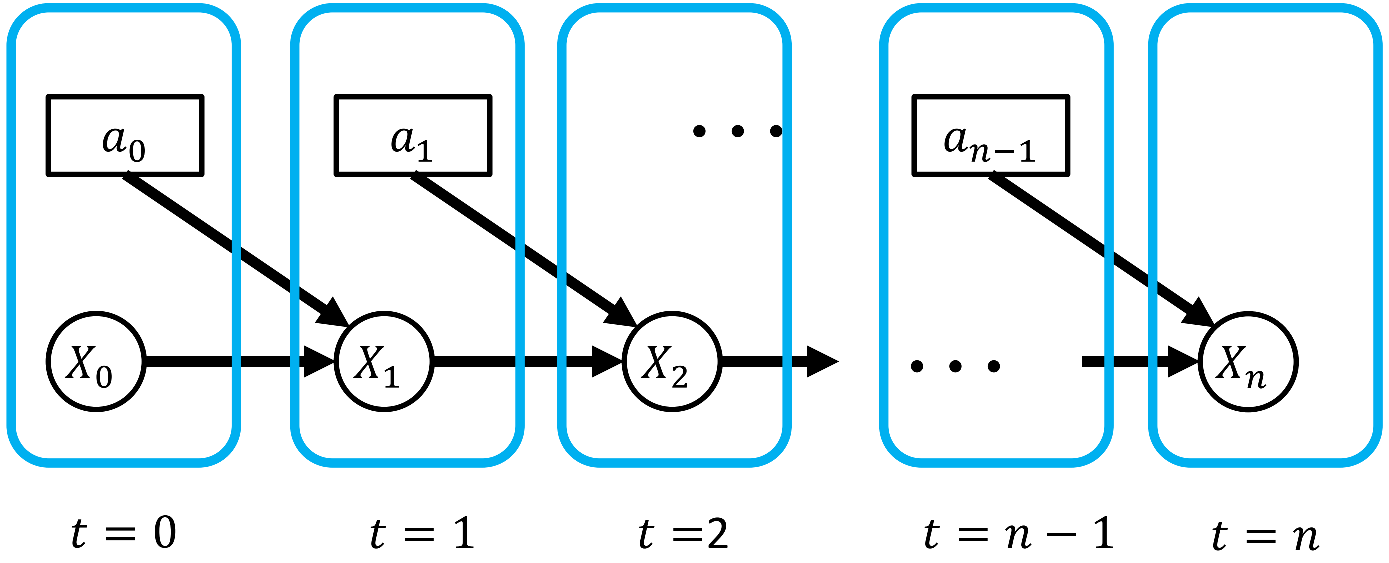
Dynamic Bayes Nets

- Bayes nets can be used to represent systems that evolve over time.
- Our vacuum cleaning robot is an example of such a system, at any time t, we have x_t and a_t and together, these determine (probabilistically) what happens for x_{t+1}.
- A dynamic Bayes net has a simple structure that repeats at each time step:



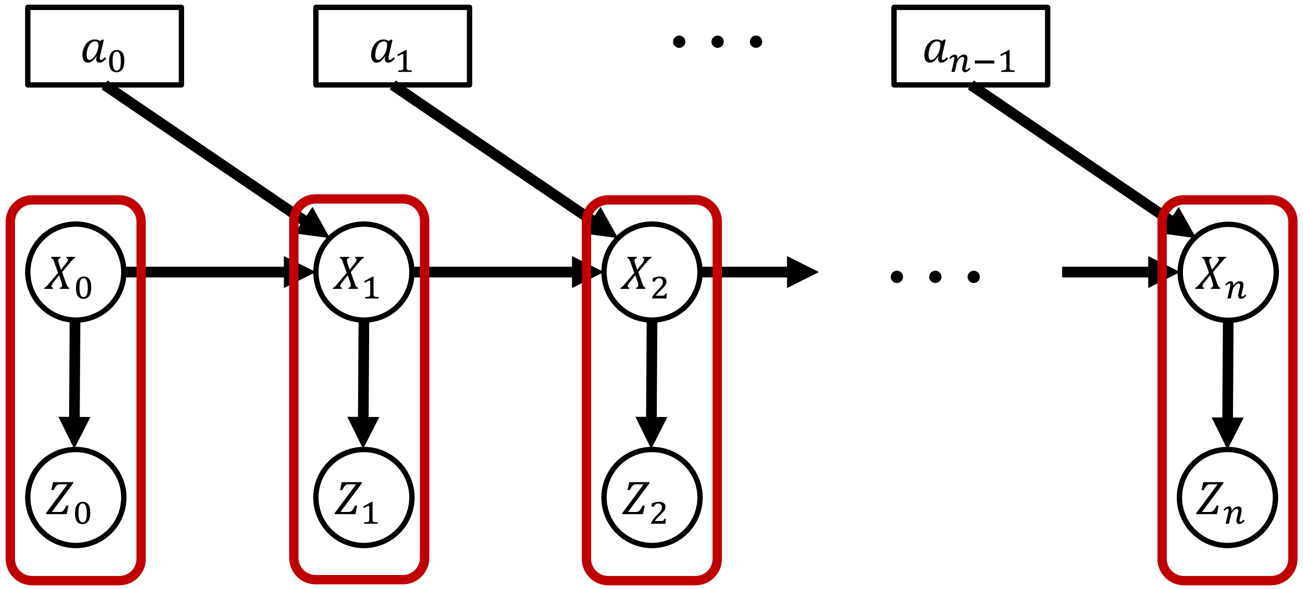
Simulation

- Forward simulation is easy for Dynamic Bayes Nets (DBNs).
- Sample initial state x_0 from the prior $P(X_0)$
- For each k generate a sample x_{k+1} from the distribution $P(X_{k+1}|X_k = x_k, a_k)$
- This is sometimes called ancestral sampling (to generate a sample for some node, look at its immediate ancestors).



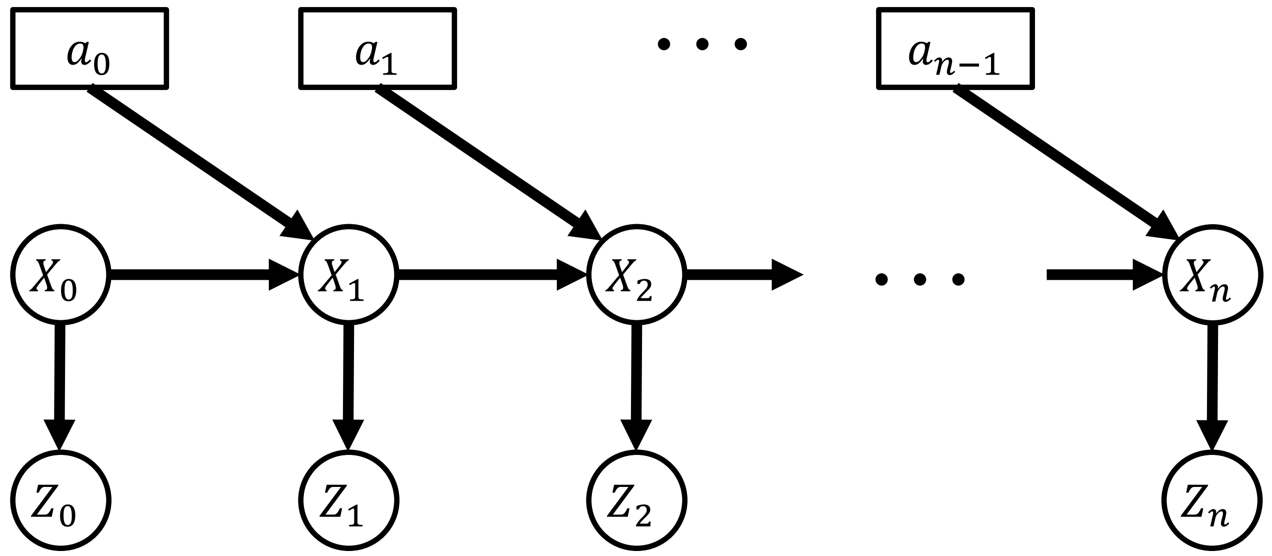
Observations

- The motivation for all of this Bayes net machinery was the idea that the history of sensor measurements was interesting. How do we encode this in a Bayes net?
- Recall our sensor model: $\boldsymbol{P(Z_t | X_t)}$.
- **This is easy to encode in a Bayes net!**



Hidden Markov Models (HMMs)

- Notice that in the system shown below,
  - we know $\boldsymbol{Z_t = z_t}$ for all $\boldsymbol{t}$
  - We know $\boldsymbol{a_t}$ for all $\boldsymbol{t}$
- We do not know any of $\boldsymbol{X_0 \dots X_1}$, but we do know that the states form a Markov chain.
- We say that the states, $\boldsymbol{X_0 \dots X_n}$, are hidden.



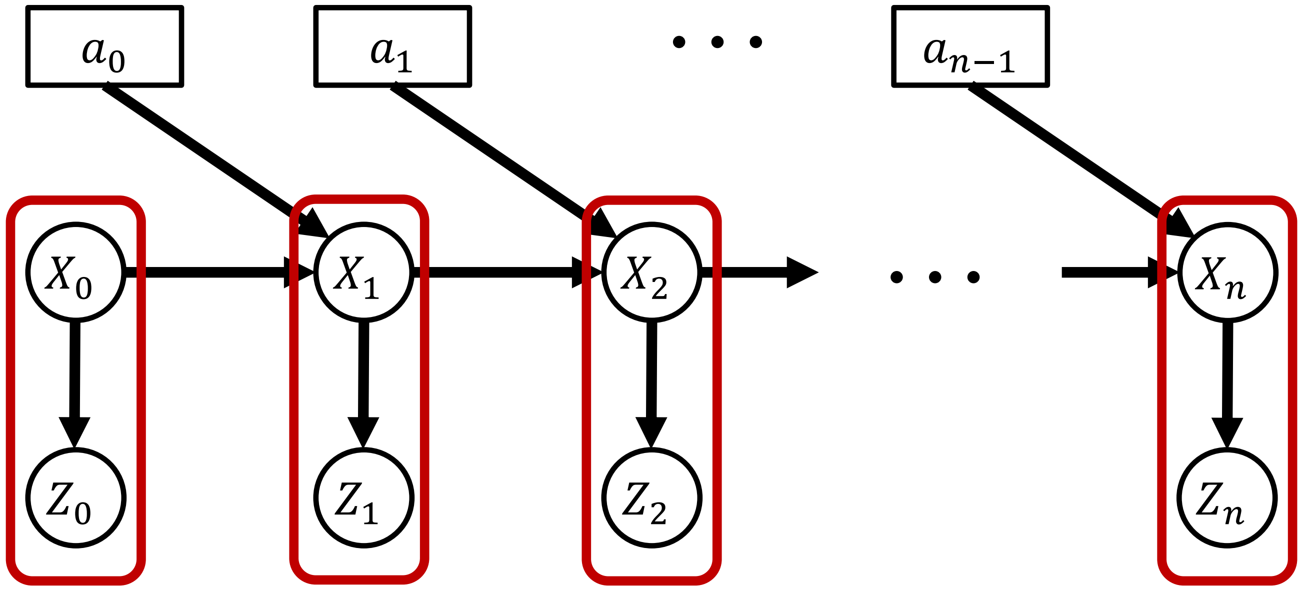
HMMs are a good model for speech recognition systems:

- Spoken words behave like a Markov chain (if you know the current word, you know a lot about what will be the next word).
- Measurements are audio signals.

Note: If we increase the relevant history, e.g., so that state X_t depends on $X_{t-1}, X_{t-2} \dots X_{t-n}$, we have an nth order Markov chain.
Larger n gives better prediction.

Simulation Revisited

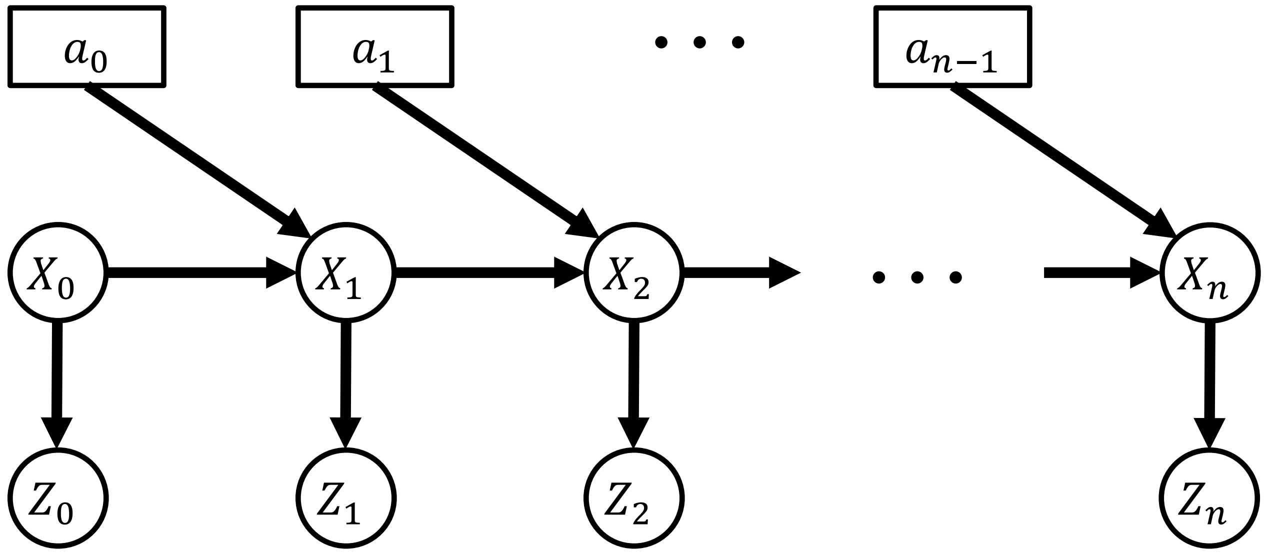
- Forward simulation is easy for Dynamic Bayes Nets (DBNs).
- Sample initial state x_0 from the prior $P(X_0)$
- For each k
    - generate a sample z_k from the distribution $P(Z_k|X_k = x_k)$
    - generate a sample x_{k+1} from the distribution $P(X_{k+1}|X_k = x_k, a_k)$



Still More Magic of Bayes Nets

For a controlled HMM with states $X_0, \dots, X_n$, and observations $Z_0, \dots, Z_n$, *the* joint distribution is given by:

$$P(Z_0, \dots, Z_n, X_0 \dots X_n | a_0 \dots a_n) = P(Z_0|X_0)P(X_0)\prod_i P(Z_i|X_i)P(X_i|X_{i-1}, a_i)$$



Perception

As before, perception is the problem of inferring things about the world given sensor information and context.

For our controlled HMM, we have

- a sequence of measurements $Z_t = z_t$
- the known sequence of applied actions $a_1, \ldots, a_n$

and we want to infer the states, $X_1, \ldots, X_n$

- ***There is a lot of structure in this problem, and we can exploit this structure to obtain computationally efficient inference algorithms.***

Inference in Bayes Nets

Our perception problem is straightforward:

- Given $\mathrm{Z}_1 = z_1 \ldots Z_n = z_n$, and the sequence of applied actions $a_1, \ldots, a_n$,
- Infer the states, $X_1, \ldots, X_n$

The description of the problem almost immediately tells us the mathematical specification:

➢ Use $\mathrm{P}(X_1, \ldots, X_n \mid Z_1 = z_1 \ldots Z_n = z_n, a_1, \ldots, a_n)$ to determine an estimate of the state sequence.

Most Probable Explanation

- Recall the definition of conditional probability:

$$\boldsymbol{P(A,B) = P(A|B)P(B)}$$

- We want to compute $\boldsymbol{P(X|Z,A)}$:

$$\boldsymbol{P(X|Z,A) = \frac{P(X,Z,A)}{P(Z,A)}}$$

- We know how to compute $\boldsymbol{P(X,Z,A)}$! (Bayes net magic)

$$\begin{gathered}\boldsymbol{X = X_1, \dots X_n} \\ \boldsymbol{Z = Z_1, \dots Z_n} \\ \boldsymbol{A = a_1, \dots a_n}\end{gathered}$$

But since we know $\boldsymbol{Z, A}$, the value of $\boldsymbol{P(Z,A)}$ is a constant!

To maximize $\boldsymbol{P(X|Z,A)}$, we need only to maximize $\mathrm{P}(\boldsymbol{X,Z,A})$

Most Probable Explanation

We are given $Z_t = z_t$, and a_t for all t.

For every possible value of $x_0, \dots, x_n$, compute

$$P(X, Z, A) = P(Z_0 = z_0 | X_0 = x_0) P(X_0 = x_0) \prod_i P(Z_i = z_i | X_i = x_i) P(X_i = x_i | X_{i-1} = x_{i-1}, a_i)$$

Our estimate is given by

$$X^* = arg\ max_X\ P(X, Z, A)$$

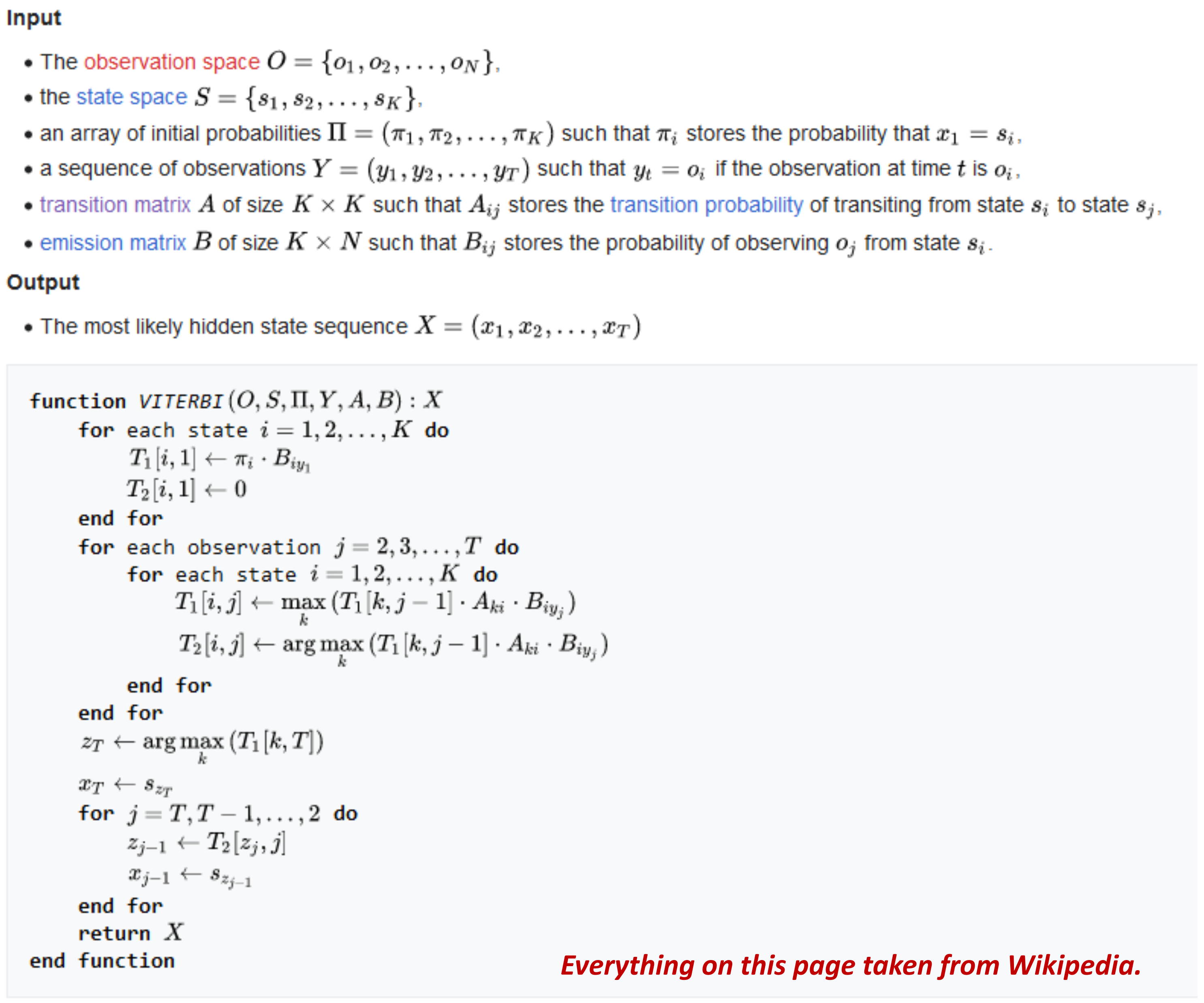
Not the most efficient algorithm, but in principle, this gets the job done.

The Viterbi Algorithm

- The Viterbi algorithm is a super-efficient dynamic programming algorithm that computes the MAP estimate for the hidden states in an HMM.

- We aren't going to learn the Viterbi algorithm in this class, but you should know that it exists, that it's efficient, and that it's a standard tool for HMM's.

- The Viterbi algorithm was invented by Andrew Viterbi, who co-founded Qualcomm. He became pretty rich, and gave $52M to USC, where he is now the Presidential Chair Professor of Electrical Engineering in the Viterbi School of Engineering.



Input

- The observation space $O = \{o_1, o_2, \ldots, o_N\}$,
- the state space $S = \{s_1, s_2, \ldots, s_K\}$,
- an array of initial probabilities $\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$ such that π_i stores the probability that $x_1 = s_i$,
- a sequence of observations $Y = (y_1, y_2, \ldots, y_T)$ such that $y_t = o_i$ if the observation at time t is o_i,
- transition matrix A of size $K \times K$ such that A_{ij} stores the transition probability of transiting from state s_i to state s_j,
- emission matrix B of size $K \times N$ such that B_{ij} stores the probability of observing o_j from state s_i.

Output

- The most likely hidden state sequence $X = (x_1, x_2, \ldots, x_T)$

```
function VITERBI $(O, S, \Pi, Y, A, B) : X$
    for each state $i = 1, 2, \ldots, K$ do
        $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
        $T_2[i, 1] \leftarrow 0$
    end for
    for each observation $j = 2, 3, \ldots, T$ do
        for each state $i = 1, 2, \ldots, K$ do
            $T_1[i, j] \leftarrow \max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
            $T_2[i, j] \leftarrow \arg\max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $x_T \leftarrow s_{z_T}$
    for $j = T, T-1, \ldots, 2$ do
        $z_{j-1} \leftarrow T_2[z_j, j]$
        $x_{j-1} \leftarrow s_{z_{j-1}}$
    end for
    return $X$
end function
```

Everything on this page taken from Wikipedia.