Sundaram PAGE No: DATE:

Markov Decision Process

A) Markov Decision Process formally describes an environment for reinforcement learning, where the environment is fully observable to the agent.
This simply means that the current state fully characterizes the decision process

B) Markov Property
A state is Markov if & only if

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \ldots, S_t]$$

The state S_t captures all relevant information for the state S_{t+1} to depend on it that it can discard all the previous states & still have no effect.
Once the state S_t is known, we can throw away the history.

C] For a Markov state s & its successor state s', we can define a probability that we transfer from state s to s' by

$$P_{ss'} = P[S_{t+1} = s' | S_t = s]$$

Since $S_t = s$ is a markov state it means it characterizes all information about the history & hence given such a state we have all the information to figure out what will happen next

So if we imagine n possible states, for any given state assuming it is markov we can define a matrix with the probability that given a particular state what is the probability we move into the next state.

$$P = \begin{array}{c} \\ \text{from} \end{array} \begin{array}{c} \text{To} \\ \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \end{array}$$

here each row of the matrix sums upto 1

We can follow this through to multiple processes & continue sampling through next possible states.

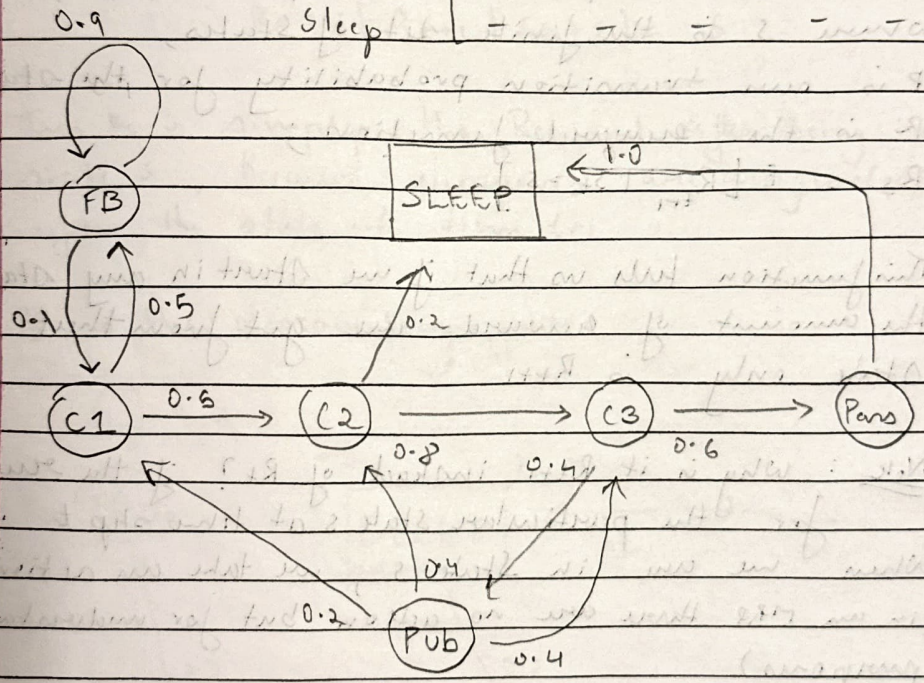
D) Amarkov Process is hence a memory less random process i.e a sequence of Random states $S_1, S_2 \ldots$ with the markov property

So a Markov Process (or a markov chain) is a tuple (S, P) where
S is the finite set of states (the possibilities of where we can be in)
P is a state transition probability matrix, which characterizes how transition from one state to the next

Example :

$P =$

	Class 1	Class 2	Class 3	Pass	FB	Pub	Sleep
Class 1	–	0.5	–	–	0.5	–	–
Class 2	–	–	0.8	–	–	–	0.2
Class 3	–	–	–	0.6	–	0.4	–
Pass	–	–	–	–	–	–	1
Facebook	0.1	–	–	–	0.9	–	–
Pub	0.2	0.4	0.4	–	–	–	–
Sleep	–	–	–	–	–	–	1



Markov Reward Process is a Markov chain with values which assign a reward / value judgement which says how good it is to be in or how much reward is accumulated across a particular sequence sampled from this state transition matrix.

A markov reward process is a tuple $\langle S, P, R, \gamma \rangle$
→ where S is the finite set of states,
→ P is our transition probability for the states
→ R is the reward function;

$$R_S = E[R_{t+1} \mid S_t = S]$$

This function tells us that if we start in any state s, the amount of reward we get from that state only is R_{t+1}

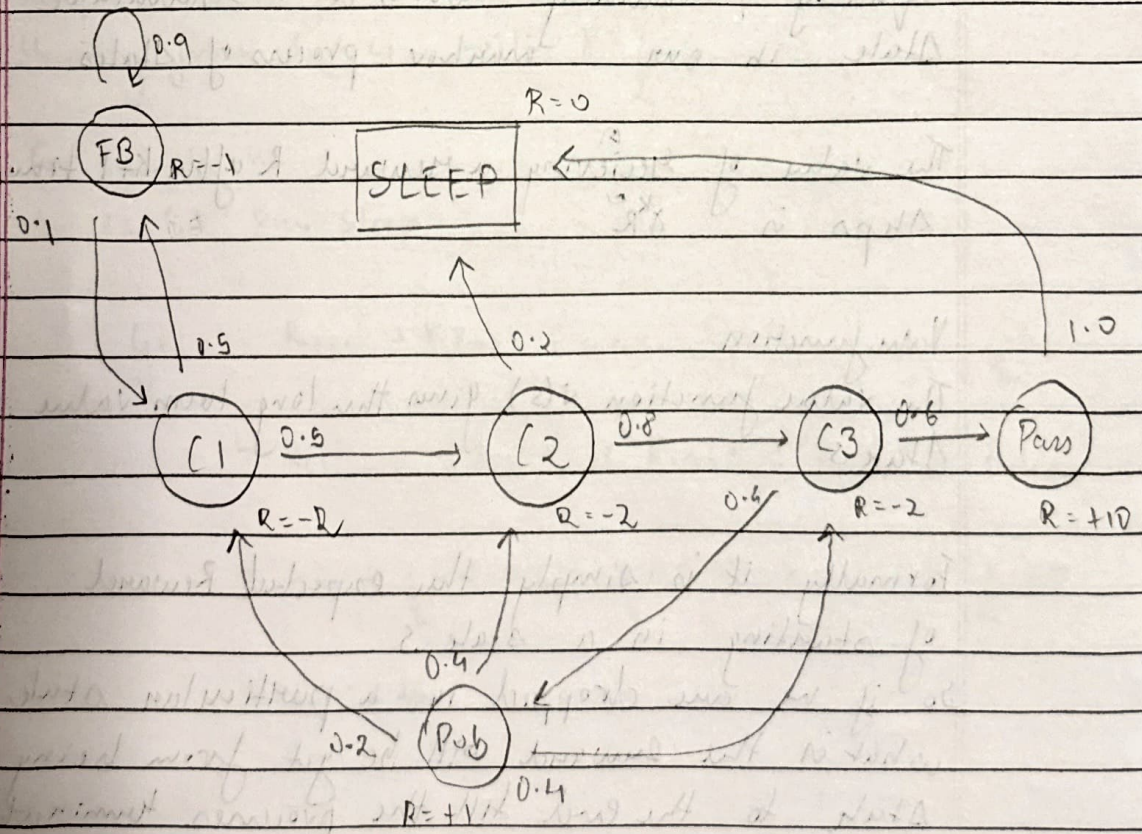
Note : Why is it R_{t+1} instead of R_t? if the reward is for the particular state s at time step t
When we are in state s, we take an action (in an MRP there are no action but for understanding purposes)
The environment moves to state S_{t+1} & gives reward R_{t+1}.
So reward belong to a transition
Reward for being in a state is same as Reward emmited when we transition out of state 1

Eg In a grid world
we are at state $S = (3,4)$
this is a fire tile (bad location)
so each time we are in this state if we
enter of continue being in this state
we receive a -5 reward at the next
time step

$$\therefore R_s = E[R_{t+1} \mid S_t = s] = -5$$

Thus R_s is simply the Reward after being in state s, because rewards are assigned/produced after the state at time $t+1$

→ γ is the discounting factor



f) Return

Since Rewards at a particular state are not simply what we wish to calculate but rather cummulative reward across the whole action chain

G_t is the total discounted reward from time-step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The discounting factor we choose is $\gamma \in [0,1]$

if $\gamma = 0$ this implies we only care about the immediate reward

if $\gamma = 1$, that maximally farsighted & it means we care about all rewards going into the future equally, assuming there is a terminal state in our markov process of states

The value of recieving a reward R after k+1 time steps is $\gamma^k R$

g) Value function

The value function v(s) gives the long term value of state s

Formally it is simply the expected Reward of starting in a state s

So if we are dropped in a particular state what is the reward will be get from being in that state to the end till the process terminate

The state value function $V(s)$ of an MRP is the expected return starting from state s

$$V(s) = E[G_t | S_t = s]$$

The expectation exists because the environment is stochastic, so the evolution of states goes one way one step or another. So for one particular state we want to calculate the value over as an expectation i.e as an average over all the possible random variable of all possible states, when the starting state is s

Example

Sample Returns for Our Student MRP

Starting from $S_1 = C_1$ with $\gamma = \frac{1}{2}$

For Sample =>

C_1 C_2 C_3 Pass Sleep

$$G_t = R_{t+1} + \gamma R_{t+2} \cdots$$

$$= R_{C_1} + \gamma R_{C_2} + \gamma^2 R_{C_3} + \gamma^4 R_{pass} + \gamma^5 R_{sleep}$$

$$= -2 + \frac{1}{2} \times (-2) + \frac{1}{4} \times (-2) + \frac{1}{8} \times 10 + \frac{1}{16} \times 0$$

$$= -2.25$$

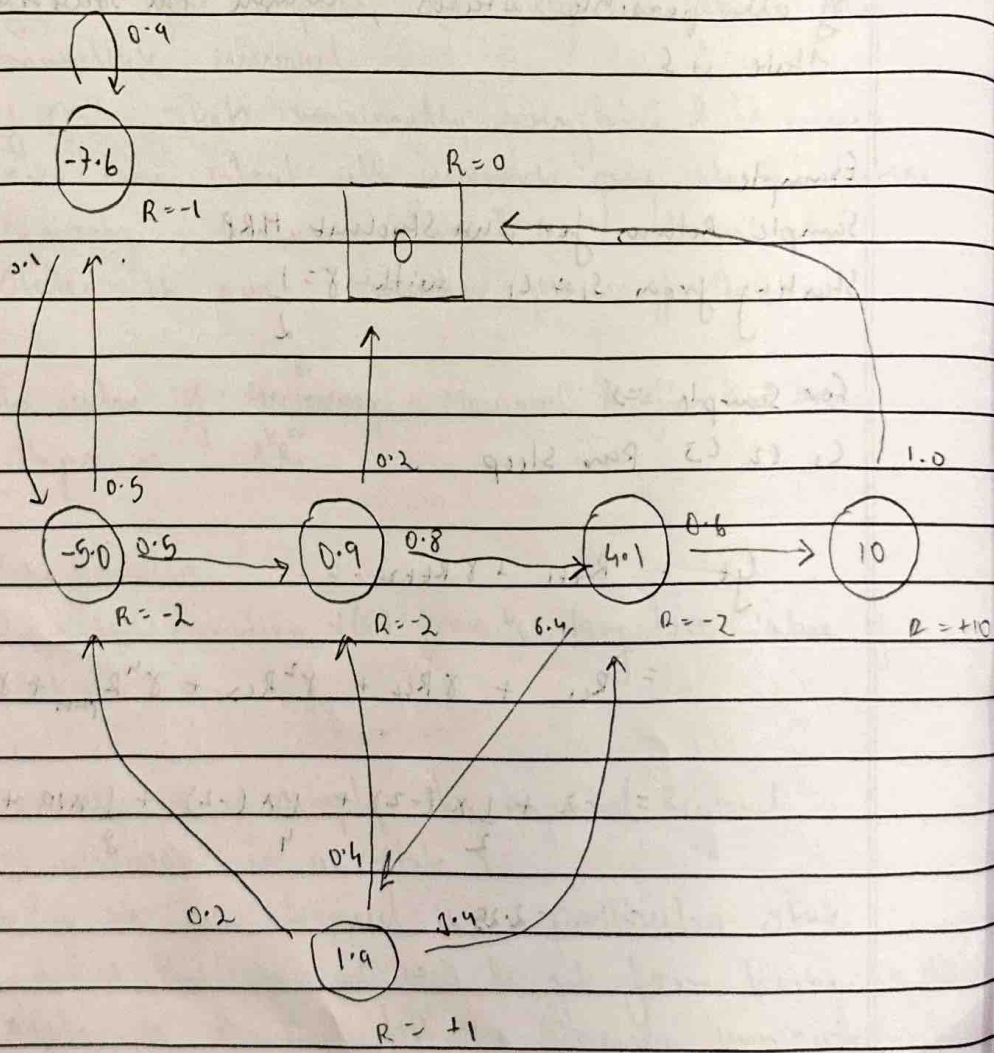
So now one way to get the value of being in this state (C_1) is to take such samples like:

C1, FB, FB, C1, C2, SLEEP or

C1, C2, C3, Pub, C2, C3, Pass, SLEEP or etc...

& then after calculating G_t we can (for each sample process) we average them, to get an estimate value of the state C_1

Example



Now to calculate the value function of a particular state (Assume in each state the number inside the shape is the calculated value function for that state)

So for Assuming class C_2 State

$$V(C_2) = R_{C_2} + \gamma\left(0.8 \times V(C_3) + 0.2\,(V(\text{sleep}))\right)$$

$$= -2 + 0.9\,(0.8 \times 4.1 + 0)$$

$$= 0.952$$

$$\approx 0.9$$

To calculate $V(C_2)$ we would need $V(C_3)$ & hence each value of the states that C_2 depends on in the future.

This formalizing can be better represented & understood by the bellman equation explanation in the following parts.

4) Bellman Equation

The value function can be decomposed into two parts

→ The immediate reward R_{t+1}

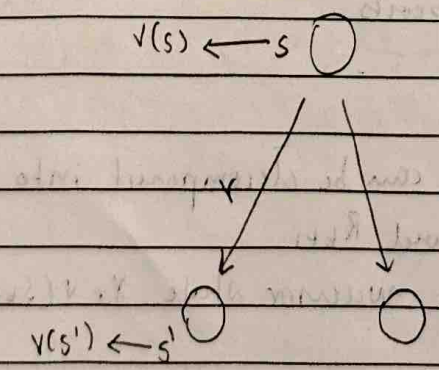
→ discounted value of a successor state $\gamma * V(S_{t+1})$

$$V(s) = E[G_t \mid S_t = s]$$
$$= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$
$$= E[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} \dots) \mid S_t = s]$$
$$= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$
$$= E[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$

So now in the previous example calculating the value of state C_1 i.e $V(C_1)$ is simply the summand of being in that state which is $-2 (R_{t+1})$ & then discounting factor (0.9) times the value of the next state, here since there are possible two next states thus the second term is $\gamma [0.8 \times V(C_2) + 0.2 \times V(Sleep)]$

1) Understanding Bellman Equations.

$$V(s) = E[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$



The circles represent the states S at each state we have value functions for that state

$$V(s) = R_s + \gamma \left[\sum_{s' \in S} P_{ss'} V(s') \right]$$

J) The Bellman equation can be expressed concisely using matrices as well;

$$V = R + \gamma P V$$

where V is a column vector with one entry per state

$$\begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix}$$

K) Solving the Bellman equation

$$V = R + \gamma P V$$
$$(1 - \gamma P) V = R$$
$$V = (1 - \gamma P)^{-1} R$$

The computational complexity for n states is $O(n^3)$

1) Markov Decision Process

An MDP is a Markov Reward Process with decisions. It is an environment in which all states are Markov.

A markov decision process is a tuple $\langle S, A, P, R, \gamma \rangle$

→ S is a set of all possible states

→ A is our finite set of actions
It is a discrete set of actions of finite size

→ P is a state transition probability matrix,

$$P^{a}_{ss'} = P[S_{t+1} = s' \mid S_t = s, A_t = a]$$

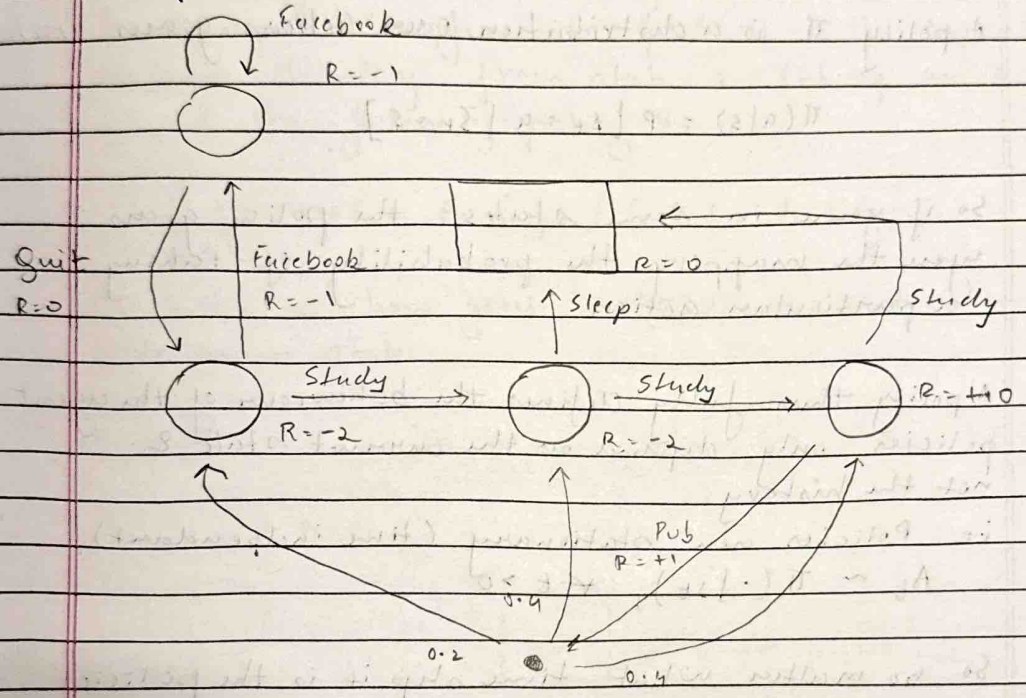
Our transition probability matrix now depends on what action we take
So now there will be a separate matrix of $P_{ss'}$ for each action a

→ R is a reward function,

$$R^{a}_{s} = E[R_{t+1} \mid S_t = s \mid A_t = a]$$

→ γ is the discounting factor

Example



Now we have more control over the MDP with actions through our path, also rewards are now attached to actions.

Our goal now is to find the best path through our decision making process that maximizes the sum of reward we get.

iii) Policies

A policy π is a distribution over actions given states

$$\pi(a|s) = P[A_t = a \mid S_t = s]$$

So if youre in some states the policy gives you the mapping the probability of taking a particular action

A policy thus fully defines the behaviour of the agent policies only depend on the current state & not the history

i.e Policies are stationary (time independant)

$$A_t \sim \pi(\cdot|s_t), \quad \forall t > 0$$

So no matter what time step it is the policies only depend on our given state

iv) Value Function

The state - value function $V_\pi(s)$ of an MDP is the expected return starting from state s & then following policy π

$$V_\pi(s) = E_\pi[G_t \mid S_t = s]$$

So now $V_\pi(s)$ calculates the rewards dependant on a specific policy given we are in a particular state.

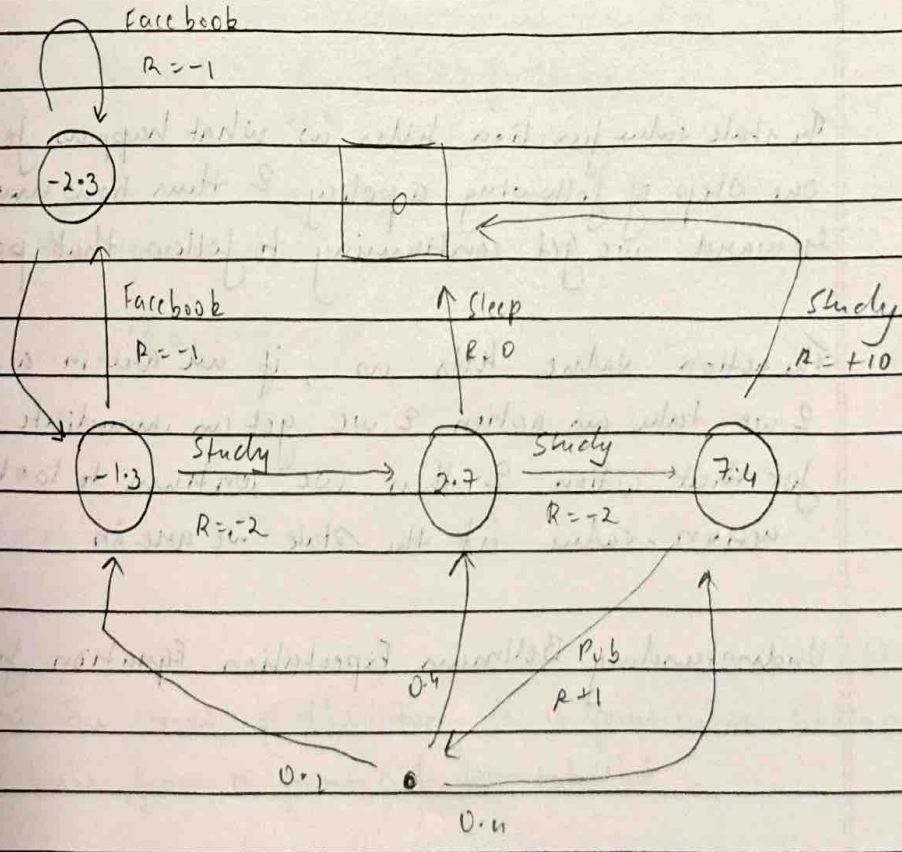
Now we sample with action according to the policy π

We also define another value function:
The Action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking an action a, & then following policy π.

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

It tells us how good is taking an action in a particular state.

Example: $V_{\pi}(s)$ for $\pi(a|s) = 0.5$; $\gamma = 1$



Here we are fixing our possible policy to be a 50-50 choice

c) Bellman Expectation Equation

The state value function can again be decomposed into immediate reward plus discounted value of a successor state,

$$V_\pi(s) = E_\pi [R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s]$$

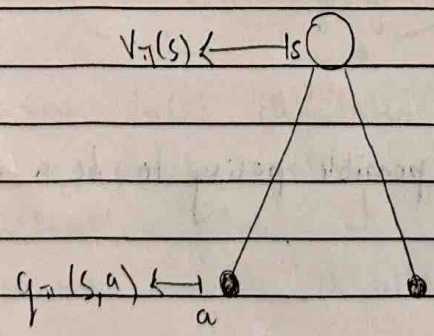
The action-value function can similarly be decomposed,

$$q_\pi(s,a) = E_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

The state value function tells us what happens for one step of following a policy & thus how much more reward we get continuing to follow that policy

The action value tells us, if we are in a state s & we take an action & we get an immediate reward for that action & then we continue to look at the action-value at the state we are in

d) Understanding Bellman Expectation Equation for V^π

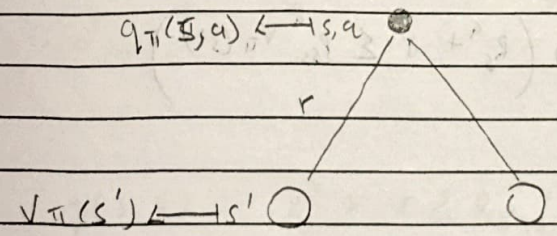


The white circle represents the state & the black dots represent the actions we can take from the given state

The probability of taking an action or a possible action is defined by our policy
And for each of the action we take there is a q value associated to it & telling us how good it is taking action a from the given state we are in, & we simply average this across all actions

$$V_\pi(s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$$

Now the converse,

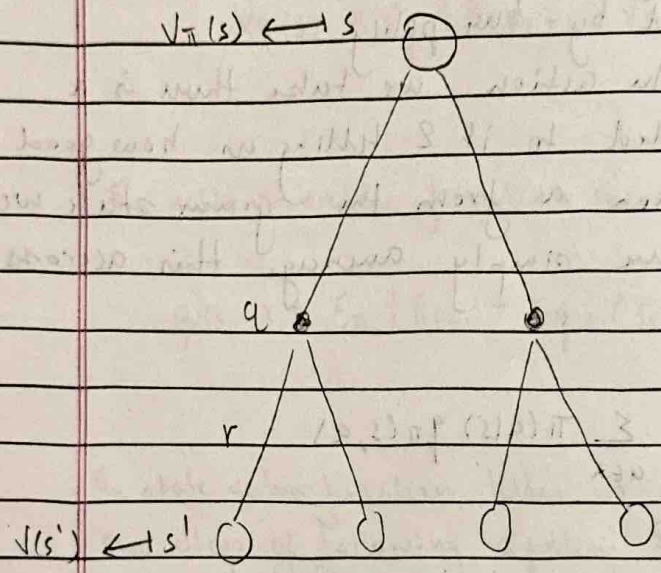


Now the root of the tree is a particular action we take from a particular state

Now for all the possible states we can go in taking a particular action, we average over the value (state value function) of being in the new state

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s')$$

Now to imagine how both work

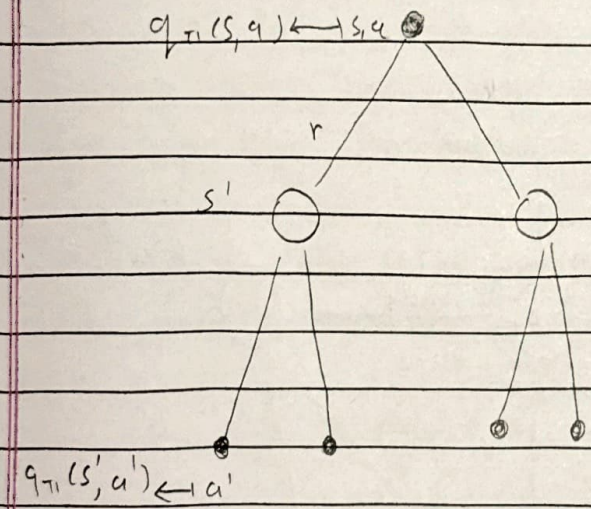


$$V_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum P_{ss'}^a V_\pi(s') \right)$$

So at the root of the tree we have a value function for a particular state, to get an idea of how good is it to be in that state we can average over first the action given our policy & then given that action the possible states we can reach & their value functions (state value functions).
So this can be thought as it takes some

probability to take an action, & from that action to go to a state it takes/gets a reward & then we also add a discounting factor to the state-value of the states we can reach from the particular action

To calculate action value

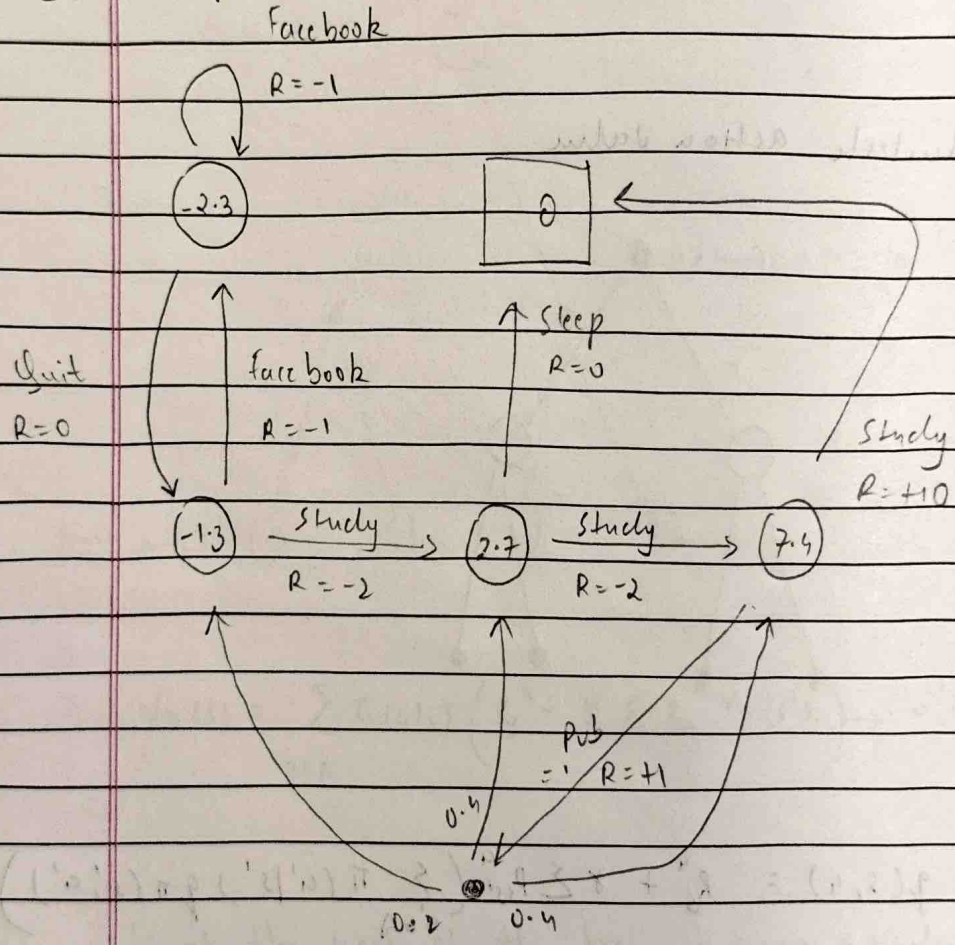


$$q(s,a) = R_s^a + \gamma \sum P_{ss'}^a \left(\sum_{a' \in A} \pi(a'|s')\, q_\pi(s',a') \right)$$

So this is simply for the value of taking a particular action from a state we look at all possible states we can reach from the action in a given state, we get the R_s^a since we take that action, within each state we discount by the discount factor the probability of going into

that state & then simple average over all possible actions out of that state given those state action values.

Eg:- Example



Now lets calculate & verify our value function for the 7.4 state

$$\text{Value(State)} = \frac{1}{2}\text{Study} + \frac{1}{2}\text{Pub}$$

$$= \frac{1}{2} \times (+10 + \gamma \times 1 \times V_\pi(\text{Sleep})) + \frac{1}{2} \times (+1 + \gamma (0.4 \times 7.4 + 0.4 \times 2.7 + 0.2 \times (-1.3)))$$

$$= 5 + 2.39$$

$$= 7.39$$

$$\simeq 7.4$$

A) Optimal Value Function

There could be multiple policies we follow in our markov chain, we wish to focus on the one that maximizes the reward.

The optimal state-value function $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

$v_*(s)$ tells us what the maximum possible reward we can extract from the system.

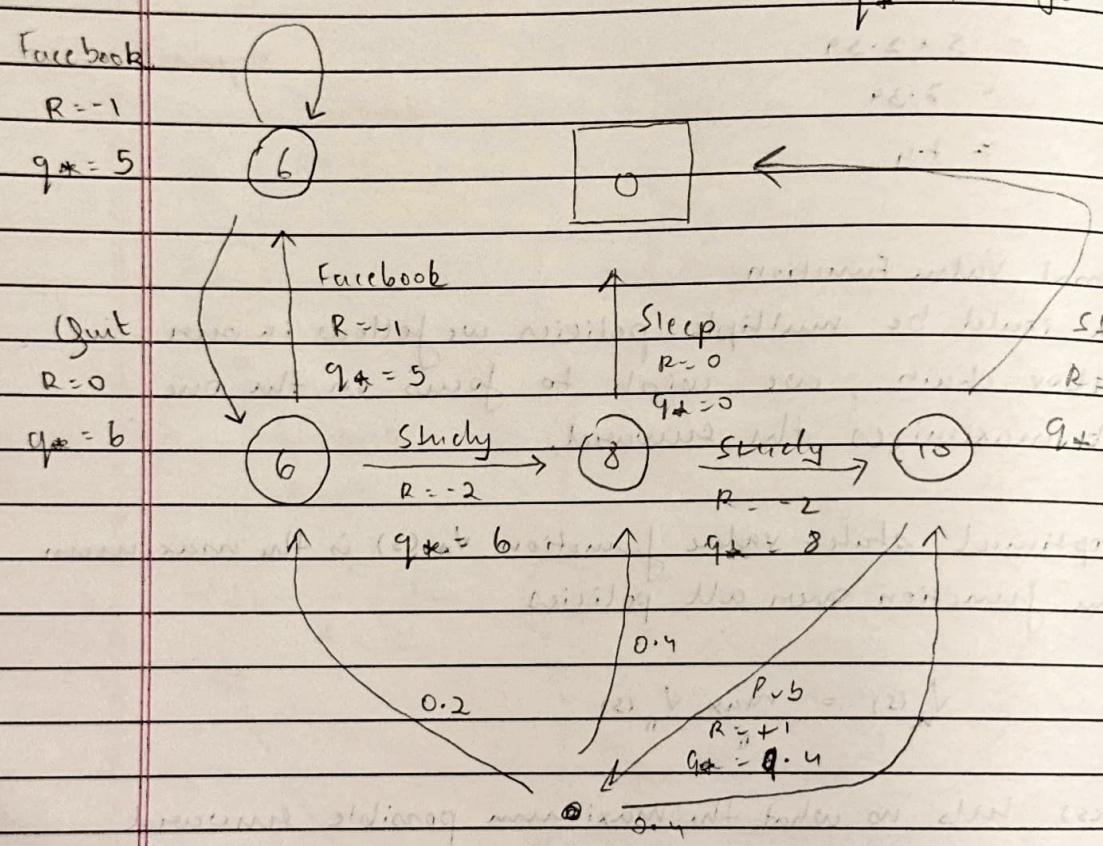
The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

$q_*(s,a)$ tells us the max reward we can extract starting in state s & taking action a, so given that we commit to an action what's the most possible reward we can get from that

point onwards:

Example:



Each state denoted by a circle contains the optimal state-value, which tells us the maximum possible value of being in that state.
The $q_*(s,a)$ values tell us that from each state given an action, which one brings the most value

5) Optimal Policy

We can define a partial ordering over policies

$$\pi \geq \pi' \quad \text{if} \quad v_\pi(s) \geq v_{\pi'}(s), \quad \forall s$$

Theorem

For any Markov Decision Process

→ There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$

→ All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$

→ All optimal policies achieve the optimal action value function, $q_{\pi_*}(s,a) = q_*(s,a)$

7) Finding an Optimal policy

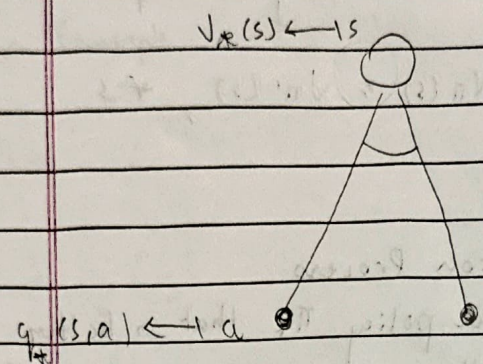
An optimal policy can be found by maximizing over $q_*(s,a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in A}{\operatorname{argmax}}\, q_*(s,a) \\ 0 & \text{Otherwise} \end{cases}$$

So if we are in state s we pick the action with the max q_* value with a full probability

There is always a deterministic optimal policy & once we have $q_*(s,a)$ we know the optimal policy

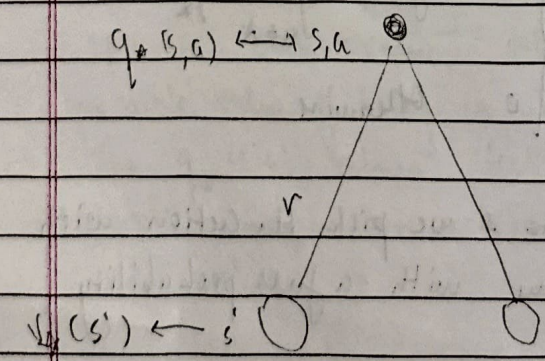
v) Bellman Optimality Equation



Now for a given state s to get the $V_*(s)$ we look at the possible actions to be taken from each the given state, & we look at the $q_*(s,a)$ for each of them & then instead of averaging over all of them we take the max

$$V_*(s) = \max_a q_*(s,a)$$

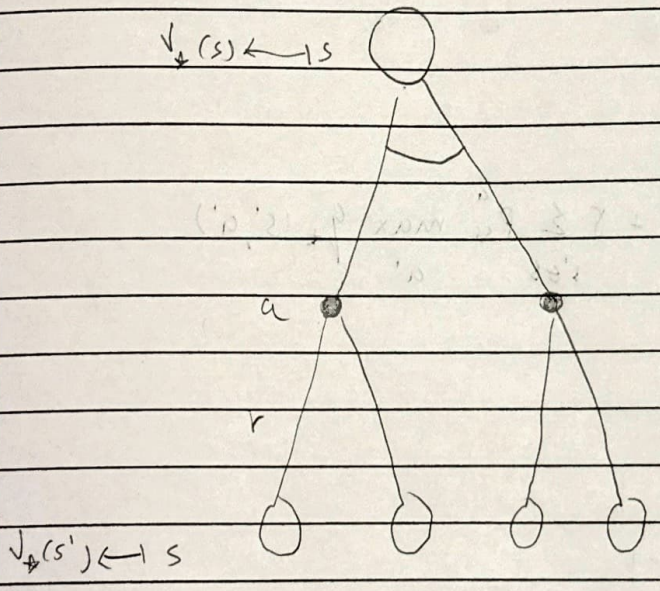
Now for the actions,



After taking an action, we could end up in a possibility of states, & wf we average over the value function of each state that tells us how good our action is.

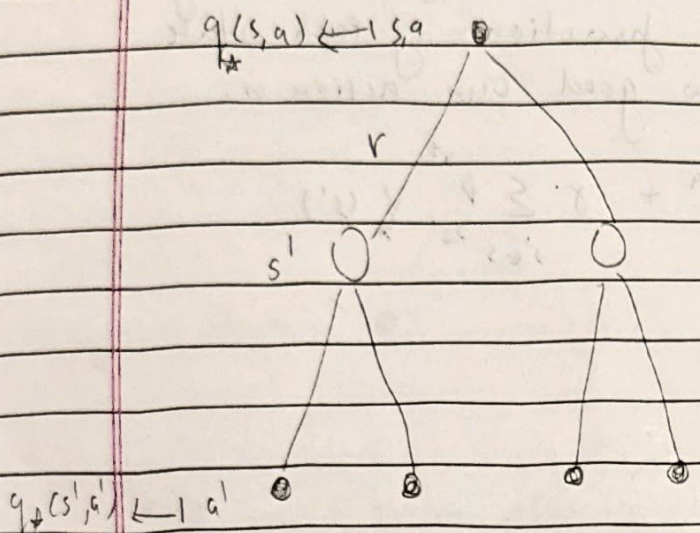
$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s')$$

Now for,



$$V_*(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s') \right)$$

For finding the optimal action-value



$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s',a')$$

Example