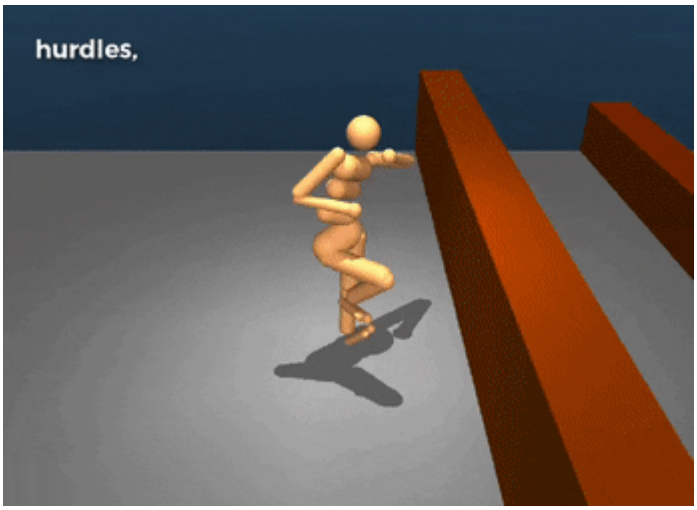
#OPTIMIZINGRL

Reinforcement Learning: Markov-Decision Process (Part 2)

Ayush Singh Follow
Aug 31, 2019 · 10 min read ★

> *This story is in continuation with the previous,* Reinforcement Learning : Markov-Decision Process (Part 1) *story, where we talked about how to define MDPs for a given environment.We also talked about Bellman Equation and also how to find Value function and Policy function for a state.**In this story** we are going to go a step **deeper** and learn about **Bellman Expectation equation** , how we find the **optimal Value** and **Optimal Policy function** for a given state and then we will define **Bellman Optimality Equation**.*



Google's Parkour using Reinforcement Learning

Let's go through a quick overview of this story:

- Bellman Expectation Equation

- Optimal Policy
- Bellman Optimality Equation for State-Value Function
- Bellman Optimality Equation for State-action value Function

So, as always grab your coffee and don't stop until you are proud.🤓

Let's start with, **What is Bellman Expectation Equation?**

Bellman Expectation Equation

A quick review of **Bellman Equation** we talked about in the previous story :

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

Bellman Equation for Value Function (State-Value Function)

From the above **equation**, we can see that the value of a state can be **decomposed** into immediate reward(**R[t+1]**) **plus** the value of successor state(**v[S (t+1)]**) with a discount factor(γ). This still stands for Bellman Expectation Equation. But now what we are doing is we are finding the value of a particular state **subjected** to some policy(π). This is the difference between the Bellman Equation and the Bellman Expectation Equation.

Mathematically we can define Bellman Expectation Equation as :

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

Bellman Expectation Equation for Value Function (State-Value Function)

Let's call this Equation 1. The above equation tells us that the value of a particular state is determined by the immediate reward plus the value of successor states when we are following a certain policy(π).

Similarly, we can express our state-action Value function (Q-Function) as follows :

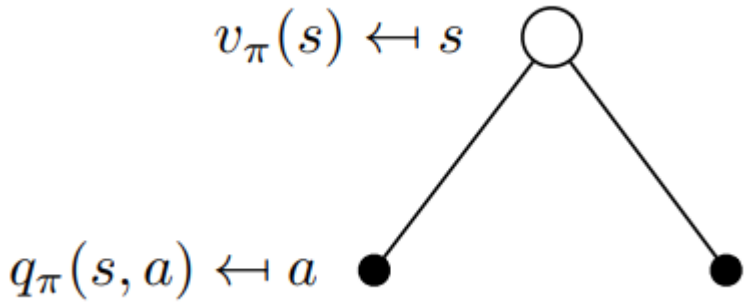
$$q_\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$$

Bellman Expectation Equation for State-Action Value Function (Q-Function)

Let's call this Equation 2.From the above equation, we can see that the State-Action Value of a state can be decomposed into the **immediate reward** we get on performing a certain action in state(**s**) and moving to another state(**s'**) plus the discounted value of the state-action value of the state(**s'**) **with respect to** the some action(**a**) our agent will take from that state on-wards.

Going Deeper into Bellman Expectation Equation :

First, let's understand Bellman Expectation Equation for State-Value Function with the help of a backup diagram:



Backup Diagram for State-Value Function

This backup diagram describes the value of being in a particular state. From the state s there is some probability that we take both the actions. There is a Q-value(State-action value function) for each of the action. We average the Q-values which tells us how good it is to be in a particular state. Basically, it defines Vπ**(s).[Look Equation 1]**

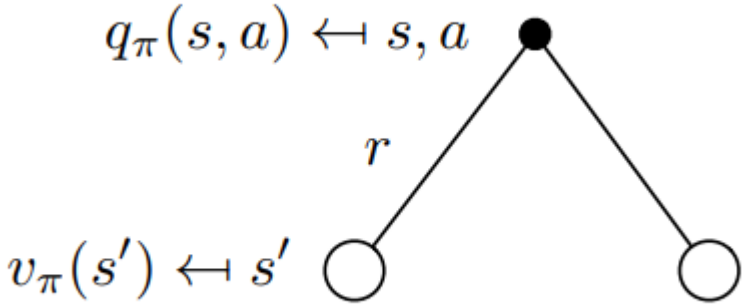
Mathematically, we can define it as follows:

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

Value of Being in a state

This equation also tells us the connection between State-Value function and State-Action Value Function.

Now, let's look at the backup diagram for State-Action Value Function:



Backup Diagram for State-action Value Function

This backup diagram says that suppose we start off by taking some action(a). So, because of the action(a) the agent might be blown to any of these states by the environment. Therefore, we are asking the question, ***how good it is to take action(a)?***

We again average the state-values of both the states, added with an immediate reward which tells us how good it is to take a particular action(a).This defines our q**π(s,a).**

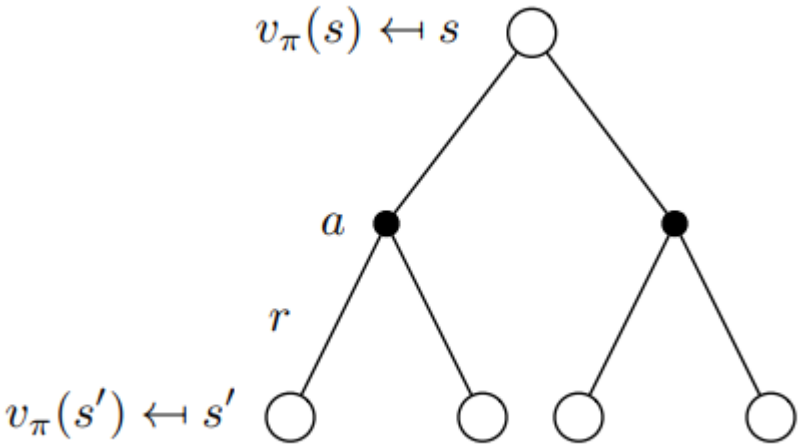
Mathematically, we can define this as follows :

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Equation defining how good it is to take a particular action a in state s

where P is the Transition Probability.

Now let's stitch these backup diagrams together to define State-Value Function, V**π(s)**:



Backup Diagram for State-Value Function

From the above diagram, if our agent is in some **state(s)** and from that state suppose our agent can take two actions due to which environment might take our agent to any of the **states(s')**. Note that the probability of the action our agent might take from state **s** is weighted by our policy and after taking that action the probability that we land in any of the states(**s'**) is weighted by the environment.

Now our question is, how good it is to be in state(s) after taking some action and landing on another state(s') and following our policy(π) after that?

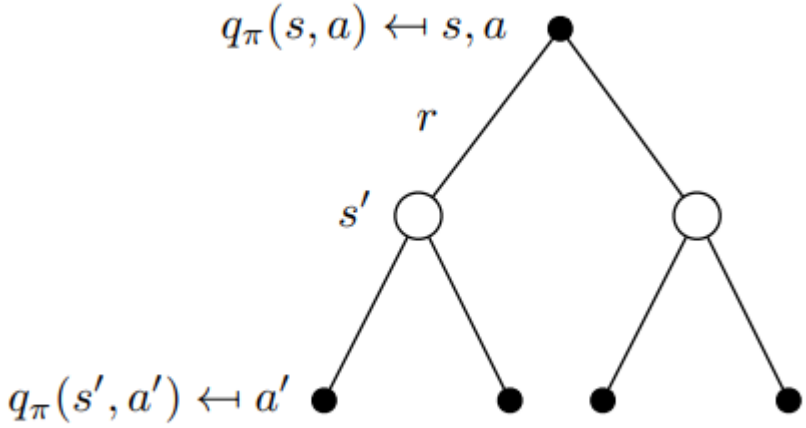
It is similar to what we have done before,we are going to average the value of successor states(**s'**) with some transition probability(P) weighted with our policy.

Mathematically, we can define it as follows:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

State-Value function for being in state S in Backup Diagram

Now, let's do the same for State-Action Value Function, qπ**(s,a)** :



Backup Diagram for State-Action Value Function

It's very similar to what we did in **State-Value Function** and just it's inverse, so this diagram basically says that our agent take some action(**a**) because of which the environment might land us on any of the states(**s**), then from that state we can choose to take any actions(**a'**) weighted with the probability of our policy(π). Again, we average them together and that gives us how good it is to take a particular action following a particular policy(π) all along.

Mathematically, this can be expressed as :

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

State-Action Value Function from the Backup Diagram

So, this is how we can formulate Bellman Expectation Equation for a given MDP to find it's State-Value Function and State-Action Value Function. ***But, it does not tell us the best way to behave in an MDP***. For that let's talk about what is meant by **Optimal Value** and **Optimal Policy Function**.

Optimal Value Function

> *Defining Optimal State-Value Function*

In an MDP environment, there are many different value functions according to different policies. ***The optimal Value function is one which yields maximum value compared to all other value function***. When we say we are solving an MDP it actually means we are finding the Optimal Value Function.

So, mathematically Optimal State-Value Function can be expressed as :

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

Optimal State-Value Function

In the above formula, v∗(s) tells us what is the maximum reward we can get from the system.

> ***Defining Optimal State-Action Value Function (Q-Function)***

Similarly, **Optimal State-Action Value Function** tells us the maximum reward we are going to get if we are in state s and taking action a from there on-wards.

Mathematically, It can be defined as :

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

Optimal State-Action Value Function

> ***Optimal State-Value Function*** *:It is the maximum Value function over all policies.*
>
> ***Optimal State-Action Value Function****: It is the maximum action-value function over all policies.*

Now, let's look at, what is meant by Optimal Policy ?

Optimal Policy

Before we define Optimal Policy, let's know, ***what is meant by one policy better than other policy?***

We know that for any MDP, there is a policy (π) better than any other policy(π'). **But How?**

We say that one policy(π) is better than other policy (π') if the value function with the policy π for all states is greater than the value function with the policy π' for all states. Intuitively, it can be expressed as :

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

Now, let's define ***Optimal Policy :***

> ***Optimal Policy*** *is one which results in optimal value function.*

Note that, there can be more than one optimal policy in a MDP. But, ***all optimal policy achieve the same optimal value function and optimal state-action Value Function(Q-function)***.

Now, the question arises how we find Optimal Policy.

> ***Finding an Optimal policy :***

We find an optimal policy by maximizing over $q_*(s, a)$ i.e. our optimal state-action value function.We solve q*(s,a) and then we pick the action that gives us most optimal state-action value function(q*(s,a)).

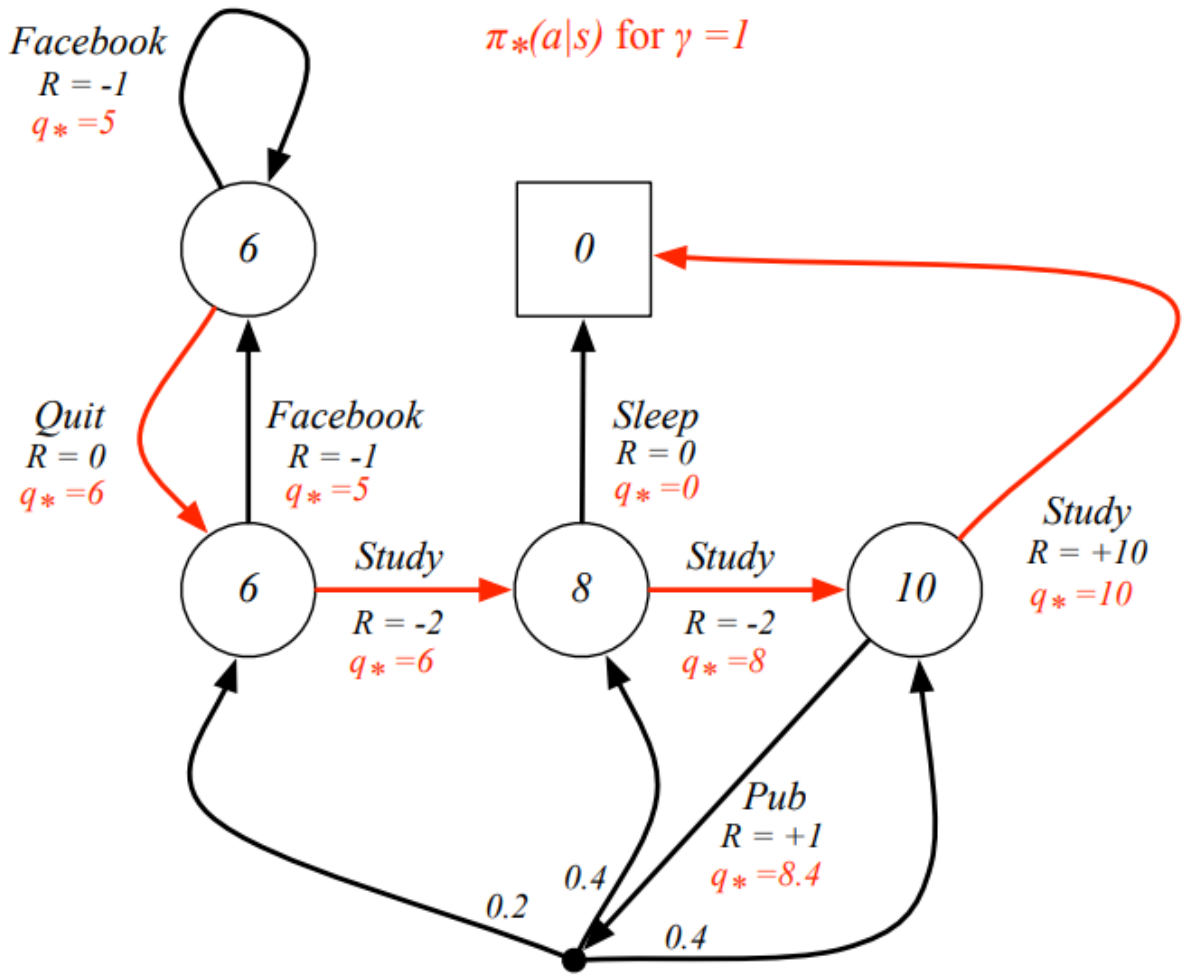
The above statement can be expressed as:

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

Finding Optimal Policy

What this says is that for a state s we pick the action a with probability 1, if it gives us the maximum q*(s,a). So, if we know q*(s,a) we can get an optimal policy from it.

Let's understand it with an example :



Example for Optimal Policy

In this example, the red arcs are the **optimal policy** which means that if our agent follows this path it will **yield maximum reward** from this MDP. Also, by seeing the **q*** values for each state we can say the actions our agent will take that yields maximum reward. So, optimal policy always takes action with higher q* value(State-Action Value Function). For example, in the state with value 8, there is q* with value 0 and 8. Our agent chooses the one with greater **q*** value i.e. 8.

Now, the question arises, ***How do we find these q*(s,a) values ?***

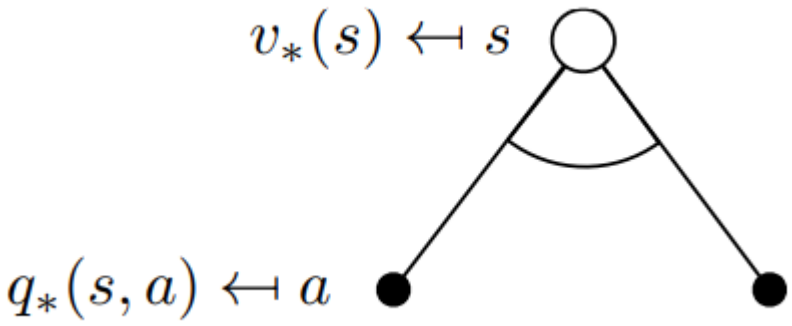
This is where Bellman Optimality Equation comes into play.

Bellman Optimality Equation

> The Optimal Value Function is recursively related to the Bellman Optimality Equation.

Bellman Optimality equation is the same as Bellman Expectation Equation but the only difference is instead of taking the average of the actions our agent can take we take the action with the max value.

Let's understand this with the help of Backup diagram:



Backup diagram for State-Value Function

Suppose our agent is in state S and from that state it can take two actions (a). So, we look at the action-values for each of the actions and **unlike**, Bellman Expectation Equation, **instead** of taking the **average** our agent takes the action with **greater q* value**. This gives us the value of being in the state S.

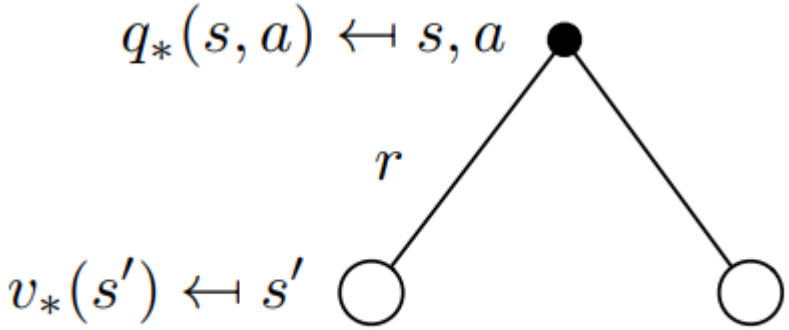
Mathematically, this can be expressed as :

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

Bellman Optimality Equation for State-value Function

Similarly, let's define Bellman Optimality Equation for **State-Action Value Function (Q-Function)**.

Let's look at the Backup Diagram for State-Action Value Function(Q-Function):



Backup Diagram for State-Action Value Function

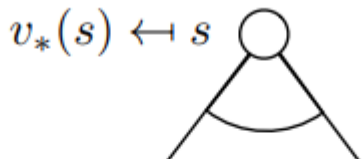
Suppose, our agent has taken an action a in some state s. Now, it's on the environment that it might blow us to any of these states (**s'**). We still take the average of the values of both the states, but the only **difference** is in Bellman Optimality Equation we know the **optimal values** of each of the states.Unlike in Bellman Expectation Equation we just knew the value of the states.

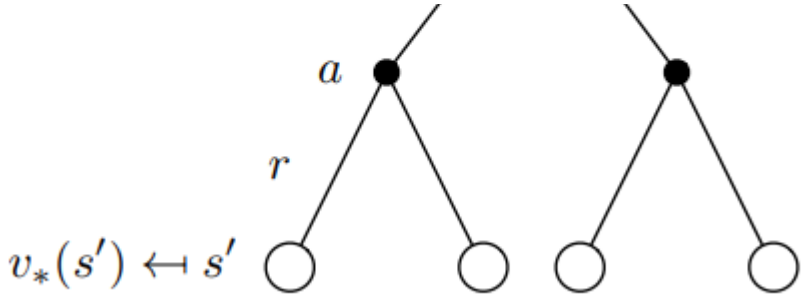
Mathematically, this can be expressed as :

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for State-Action Value Function

Let's again stitch these backup diagrams for State-Value Function :





Backup Diagram for State-Value Function

Suppose our agent is in state s and from that state it took some action (a) where the probability of taking that action is **weighted** by the policy. And because of the action (a), the agent might get blown to any of the states(**s'**) where probability is weighted by the environment. **In order to find the value of state S we simply average the Optimal values of the States(s').** This gives us the value of being in state S.

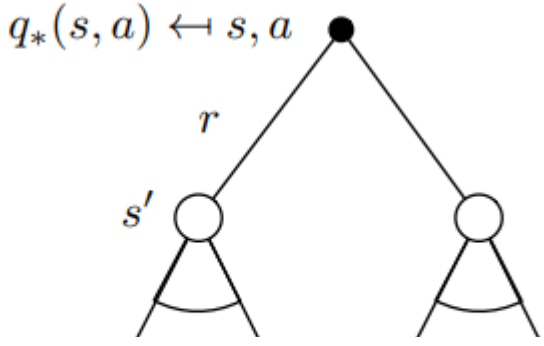
Mathematically, this can be expressed as :

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for State-Value Function from the Backup Diagram

The max in the equation is because we are maximizing the actions the agent can take in the upper arcs. This equation also shows how we can relate V* function to itself.

Now, let's look at the Bellman Optimality Equation for State-Action Value Function,q* (s,a) :



Backup Diagram for State-Action Value Function

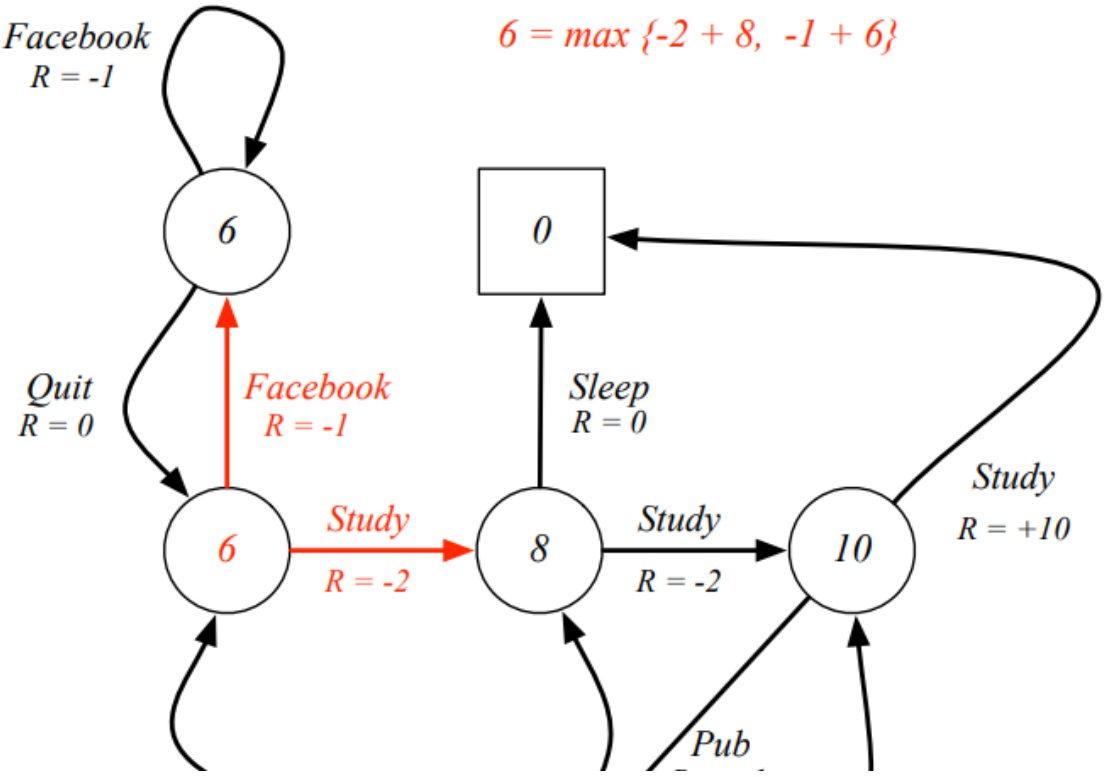
Suppose, our agent was in state **s** and it took some action(**a**). Because of that action, the environment might land our agent to any of the states (**s'**) and from these states we get to **maximize** the action our agent will take i.e. choosing the action with **maximum q* value**. We back that up to the top and that tells us the value of the action a.

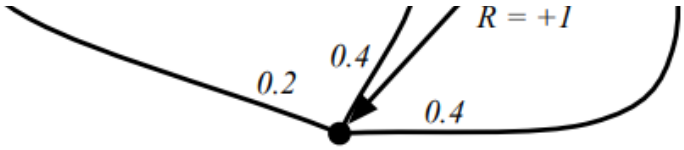
Mathematically, this can be expressed as :

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

Bellman Optimality Equation for State-Action Value Function from the Backup Diagram

Let's look at an example to understand it better :





Example for Bellman Optimality Equation

Look at the red arrows, suppose we wish to find the **value** of state with value 6 (**in red**), as we can see we get a reward of -1 if our agent chooses Facebook and a reward of -2 if our agent choose to study. In order to find the value of state in red, we will use the **Bellman Optimality Equation for State-Value Function** i.e. *considering the other two states have optimal value we are going to take an average and maximize for both the action (choose the one that gives maximum value)*. So, from the diagram we can see that going to Facebook yields a value of 5 for our red state and going to study yields a value of 6 and then we maximize over the two which gives us 6 as the answer.

Now, how do we solve Bellman Optimality Equation for large MDPs. In order to do so we use **Dynamic Programming algorithms** like **Policy iteration** and Value iteration which we will cover in next story and other methods like **Q-Learning** and **SARSA** that are used for Temporal Difference Learning which we will cover in a future story.

. . .

Superb!

Congratulations on coming this far!👍

Hope this story adds value to your understanding of MDP. Would Love to connect with you on #instagram.

Thanks for sharing your time with me!

If you love this one, please do let me know by clicking on the 👏.It helps me write more!

. . .

References :

- https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
- Hand-On Reinforcement Learning with Python
- DeepMind Reinforcement Learning Course by David Silver

STAY DEEP

Machine Learning | Deep Learning | Reinforcement Learning | Artificial Intelligence | Markov Decision Process