PART-1

EXERCISE - 5.4

MC-ES estimates

Initialize:

$\pi(s) \in A(s)$ arbitrarily, for $\forall s \in S$

$Q(s,a) \in \mathbb{R}$ arbitrarily, for $\forall s \in S$, $a \in A(s)$

$N(s,a) \leftarrow$ for $\forall s \in S$, $a \in A(s)$

Loop forever (for each episode)

Choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly such that all pairs have probability > 0.

Generate an episode from S_0, A_0 following π: $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

loop for each step of episode, $t = T-1, T-2, \dots 0$

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \dots S_t, A_{t-1}$:

$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \times \left(G_t - Q(S_t, A_t)\right)]$

Basically Equation:-

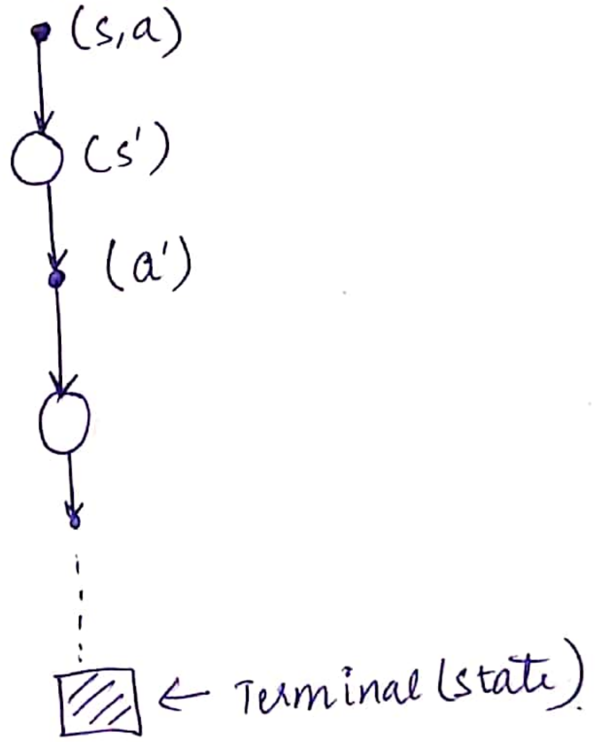
$$Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}\left[G - Q(s,a)\right]$$

PART-2

Ex 5.3.

Monte-Carlo updates the state-action pairs values only after generating the entire episode.

Hence, the Back-up diagram for this would be:-



Where ~~we~~ we want to update root node (s, a) which is a state-action pair & followed by all the transitions of the states & actions appear in an episode till termination which contribute to the update.

PART-3

EX 5.6

According to Monte-Carlo, importance sampling ratio is given by:-

$$\rho_{t:T-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

& this ratio transforms the returns to have right expected value as:-

$$V_\pi(s) = E[\rho_{t:T-1} G_t | S_t = s]$$

Now, when we have state action-value function

then $q_{v_\pi}(s,a) = E[\rho_{t:T-1} G_t | S_t = s, A_t = a]$

$$Q(s,A) = \frac{\sum_{t \in \mathcal{T}(s,a)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s,a)} \rho_{t:T(t)-1}}$$

Part -5

EX 6.2

⇒ The new scenario has shifts to the new building for work. In this case too, TD learning is expected to be much better than MC learning. ~~one does~~ since there is only a change in the initial route. Once we enter highway, many of the states after highway are common to reach home. Therefore, the value function estimates for these common states should be very close to the values that were there in the original problem/building.

This would happen same in our original learning task if our initial guess at the value function is very close to that of the true value function.

Part 6.

EX 6.3.

~~Given~~ The equation for TD(0) update for a state value function is :-

$$V(S_t) \leftarrow V(S_t) + 0.1\,[R_{t+1} + V(S_{t+1}) - V(S_t)]$$

V(A) update after constant first episode V(St) update will be:-

$$V(A) \leftarrow V(A) + 0.1(0 + 0 - V(A))$$

$$= 0.9\,V(A) = 0.45.$$

~~which~~ which is also there in graph & happens only when the left action is chosen from state A.

So, the change is by $0.5 - 0.45 = 0.05$.

Part 6.

EX 6.3.

~~Given~~ The equation for TD(0) update for a state value function is :-

$$V(S_t) \leftarrow V(S_t) + 0.1\,[R_{t+1} + V(S_{t+1}) - V(S_t)]$$

V(A) update after constant first epistode V(st) dupdate will be:-

$$V(A) \leftarrow V(A) + 0.1(0 + 0 - V(A))$$

$$= 0.9\,V(A) = 0.45 .$$

~~which~~ which is also there in graph & happens only when the left action is chosen from state A.

So, the change is by $0.5 - 0.45 = 0.05$.

Exercise 6.4.

No, I don't think that the wide range of alphas would affect ~~the~~ concluding about which algorithm is better. As can be observed, that for different values of alpha for TD, the plots were deflecting a bit for higher alphas & ~~[illegible]~~ same goes for MC. but overall everytime TD performs better than MC.

It cannot be fixed i.e. value of alpha since it will depend on the task to be learnt. And Also, it is seen that TD performs better than MC for this Tasks like there where one step Reward is known ~~& don't need the entire episode to be generated~~.

Exercise 6.5:

Since the equation for TD is :-

$$V(S) = V(S) + \alpha[R + \gamma V(S') - V(S)]$$

Where $R + \gamma V(s') - V(s)$ is the error. So, the large values of alpha help converging faster but this actually impacts the random step taken at each time step during the algorithm ~~where~~. ~~the algorithm~~ and values become highly sensitive to these steps with large alpha values.

Also, with large alpha, the error part during updation will be higher which initially helps converging but later will result in more value state function.

Scanned by CamScanner

Part 8.

Exercise 6.12

Since action selection is greedy, then Q-learning will no more be off-policy method, since initially in Q-learning, we select actions on the basis of ∈-greedy & value updation is done using greedy policy. Hence the same policy is used for selecting action as well as value updation. Therefore it converts to on-policy method.

However, the action selection & updates for this changed Q-learning will not be same as SARSA because although both are now on-policy methods but SARSA select action & updates ~~values~~ weights according to ε-greedy policy than greedy policy used ~~there~~ here in changed Q-learning. unless SARSA also uses the ~~same~~ greedy policy.