RL ASSIGNMENT - 3

Question 1

The code $\begin{bmatrix} \text{Append } G \text{ to List } R. \\ Q(S_t, A_t) \leftarrow avg(R) \end{bmatrix}$ is highly inefficient.

This stores all returns in a list. And the list needs to be stored for every iteration and the Action-Value function $Q(S_t, A_t)$ is updated by taking an average of all returns stored in the list. We update the update rule for $\boxed{Q(S_t, A_t)}$ as follows.

$$Q_n(S_t, A_t) = \frac{1}{n} \sum_{i=1}^{n} G_i(S_t, A_t) \quad \text{—— (i)}$$

= We split it as follows.

$$= \frac{1}{n} \left(G_n(S_t, A_t) + \sum_{i=1}^{n-1} G_i(S_t, A_t) \right)$$

$$= \frac{1}{n} \left[G_n(S_t, A_t) + \frac{(n-1)}{(n-1)} \cdot \sum_{i=1}^{n-1} G_i(S_t, A_t) \right]$$

= Using (i)

$$= \frac{1}{n} \left[G_n(S_t, A_t) + (n-1) Q_{n-1}(S_t, A_t) \right]$$

$$= \frac{1}{n} \left[G_n(S_t,A_t) + n\, Q_{n-1}(S_t,A_t) - Q_{n-1}(S_t,A_t) \right]$$

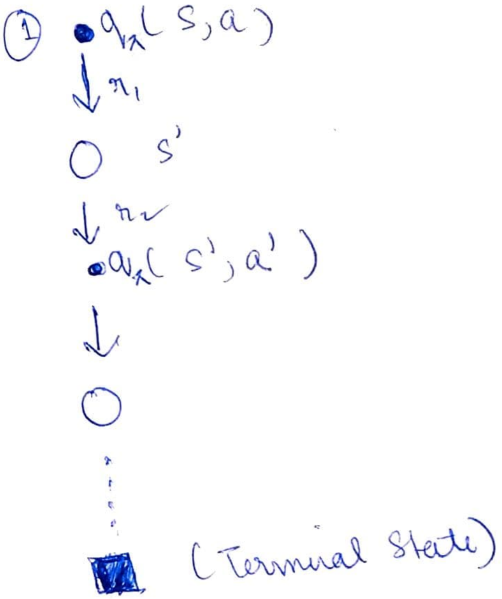
$$= Q_{n-1}(S_t,A_t) + \frac{1}{n} \left[G_n(S_t,A_t) - Q_{n-1}(S_t,A_t) \right]$$

The update rule is finally as follows

$$\boxed{Q_n(S_t,A_t) = Q_{n-1}(S_t,A_t) + \frac{1}{n} \left[G_n(S_t,A_t) - Q_{n-1}(S_t,A_t) \right]}$$

Question 2

The backup diagram for Q_π is ..



The (1) i.e. the topmost node shows the initial (state-action) pair. The rest of the nodes represent all intermediate nodes

Index

● → (state - action) Nodes.

○ → State Nodes

Question 3

Let $J(s,a)$ represent the set of all state action pair (s,a) which have been visited by an agent.

Given the equation (5.6)

$$V(s) = \frac{\sum_{t \in J(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in J(s)} \rho_{t:T(t)-1}}.$$

We rewrite $Q(s)$ as follows

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

where G_t is the return at steps t to $T(t)$

& $\rho_{t:T(t)-1}$ & $\rho_{t+1:T(t)-1}$ are the imp. sampling ratios.

Question 5

The mentioned example scenario in the text would show significantly better performance of a TD Based approach, as many of the initial states in the problem remain unchanged, i.e. For the route to home while the home itself is changed the highway remains the same.

→ TD learning would be able to make quicker updates and as the number of ~~optimal~~ common states is comparatively higher we'll reach an optimal solution in quicker time.

→ MC methods on the other hand would have to wait for an whole episode to finish as they are not bootstrapped and would end up being inefficient as compared to TD methods,

Also,

Question 8

Even if the action selection policy is greedy Q-learning is not exactly as same as SARSA learning. Q learning algorithms first update the values and then the next action is picked on the basis of the updated values.

On the other hand Sarsa learning algorithms first select an action and then update the Q values for every (state - action) pair.

x A v.

Q6 Exercise accompanying the generated graphs in the $\boxed{\text{code}}$.

Graph 1 : Estimated Values Graph

Graph 2 : Empirical RMS Error.

6.3

We start from the middle i.e. state 'C'. We also know that transition to any state has a reward '0', except +1 for terminating right.

We follow the following update rule for updating our state value function

$$V(S) = V(S) + \alpha\left[R_{t+1} + \gamma(V(S_{t+1})) - V(S)\right]$$

According to our initialization

$V(B) = 0.5$, $V(D) = 0.5$

R=0 for B2D.

↳ $V(C) = 0.5 + 0.1(0 + \cancel{0.5}^{\to 0} - 0.5)$

$= 0.5$ (remains unchanged)

Now we can either go to E or A (in the direction of). Since V(E) is unchanged the agent must have gone towards 'A', as a result of which rest all values are unchanged.

<u>Also</u>,

As left terminal state's value function is 0 by definition.

$V(A) = 0.5 + 0.1(0.\cancel{5} + 0 - 0.5)$

$= 0.45$

Change = 0.05 i.e. a 10% deduction.

<u>6.4</u>

The conclusions about the performance of algorithms would somewhat depend on the range of alpha that we choose. TD Learning would always perform better the MC methods given α is reasonably small.

However taking larger values of α might affect the TD approaches as they may result in more sudden and large jumps / oscillations which might stop the algorithm from converging.

6.5

Larger values of 'α' can cause such oscillations, in the error of TD learning algorithms. As the algorithm tries to reach an Optimal State, a larger value of 'α' may stop the algorithm from converging completely.

This is true for all larger values of 'α' independent of ~~the~~ how the value function is initialised.