RL H-3

Date: ....

Q.1) If we get the states and actions and rewards (or returns corresponding to them) when we update the Q values, we will average all occurances of the returns for each state and action. We first pick Q values as random.

First-Visit Monte Carlo:

$$Q_{n+1}(s,a) = \frac{1}{n}\left(\sum_{i=1}^{n} G_i\right)$$

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

$$= \frac{1}{n}\left((n-1)Q_n(s,a) + G_n\right)$$

$$Q_{n+1}(s,a) = Q_n(s,a) + \frac{1}{n}\left(G_n - Q_n(s,a)\right)$$

This eq^n updates the current estimate of state action value function based on returns, ~~and~~ previous estimate and number of time state and action is visited.

Pseudo code

Initialize : (randomly)
$\pi(a|s) \in A(s) \quad \forall s \in S$
$Q(s,a) \in R \quad \forall s \in S$ and $a \in A$
$Returns(s,a) = 0 \quad \forall \; s \in S$ and $a \in A$.

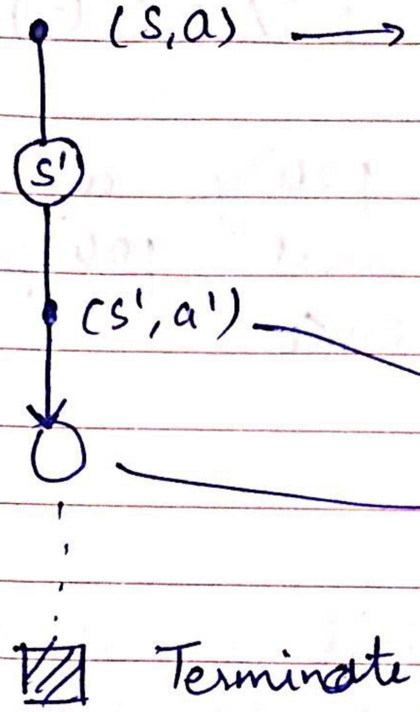
Loop forever (n episodes)
choose initial $S_0 \in S$ and $a_0 \in A$ such that each state & action has prob $\neq 0$.
Generate entire episode begining S_0 & A_0
$\pi : S_0, A_0, R_1, S_1, A_1, R_2 \ldots$
Loop for all episode (in reverse)
$G = \gamma G + R$
$Returns(s,a) \mathrel{+}= 1$
While ((s,a) appears in sequence)

$$Q(s,a) = Q(s,a) + \frac{1}{Returns(s,a)} [G - Q(s,a)]$$

$$\pi(a|s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

This pseudo code works similar to that of monte carlo exploring starts with evaluation & improvement while also have same update straligy and initial state /action choice.

Ques2) Backup Diagram:



Initial state and action of the episode. The states and actions that appear after that are represented by nodes. ~~Black dot is~~

Black dot state and action pairs.

Circles are states.

Ques3)

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

$\tau(s,a) \rightarrow$ no. of time step where we observe (s,a).

T is termination after t

G_t is return from $t : T^{(t)}-1$

$\rho_{t:T(t)-1}$ is the important sampling ratio

$$\rho_{t:T(t)-1} = \prod_{K=t}^{T-1} \frac{\pi(A_K \mid S_K)}{b(A_K \mid S_K)}$$

π is the target policy while b is the behavioural policy used to generate sequence.

Question 5) The initial route changes b/we the old and new initial states, we would still get some states that are similar to previous ones. Since, TD0 method involves bootstrapping and also that we are given we already have lot of experience driving home from work., ideally TD0 performs better than MC due to the reason that we utilize knowledge of state values of common states even when the route has changed. The bootstrapping in TD0 would ensure faster convergence as well.

Yes, same sort of thing should happen in the original ~~state~~ scenario if estimated values are close to actual values.

Question 6)

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

If $S_{t+1} \neq$ terminal (besides E):
reward is ~~not~~ zero.

$$V(S_t) = V(S_t) + 0.1[V(S_{t+1}) - V(S_t)]$$

We initialize our value function with same values :

$$V(S_0) = \begin{cases} 0.5 & \text{non-terminal} \\ 0 & \text{terminal} \end{cases}$$

Hence, second term becomes zero.
when $V(S_{t+1}) = V(S_t) = 0.5$.
When we move to terminal state where reward received = 0.

$$V(A) = V(A) + 0.1(-V(A))$$
$$V(A) = 0.9\,V(A)$$
$$V(A) = 0.45$$

At the end of the episode, V(A) becomes 0.5 times it's value (i.e. becomes 0.45).
Decrease = 0.05.

So, from A we go to terminal state (nearest) and received 0 reward and change V(A) by 0.05.

Question(3) If action selection is greedy, Q learning and SARSA will still be different. While SARSA chooses next action based on the current estimates of Q values and applying an ε-greedy policy on it, and later updates its estimates. But Q-Learning would first update and then ~~ac~~ choose action based on ε-greedy policy on Q-values. Hence, there is a possibility that after updation, we get a different action choosen at that State. Hence, weight updates & action will be different.

Step size parameter (α) tells us the amount of variation we want at each time step. Greater the α, greater is the sensitivity towards reward received hence a larger update.

~~the~~ This would be affected by a wider range of alphas used in terms of RMSE with episodes. As a larger alpha means faster learning, we would observe less RMSE. There is no different fixed alpha at which algo would perform better ~~there~~ as decreasing alphas beyond a point wouldn't improve performance.

In TD0, update rule is similar to gradient descent update rule. Here, alpha is the sensitivity parameter that decides how much we move the estimates in each iteration.

After we achieve a stable performance, larger alphas would cause the value function to oscillate around actual value. No, it isn't the function of the way value function is initialised