Assignment-3

Monte Carlo and TD.

Q1 Rather than calculating the mean of return repeatedly, we could follow an iterative approach. Here we wish to calculate the update state value $\&$ or action-value function in each episode depending upon the state, action and type of visit approach.

The updated sub-part pseudo-code is as follows:

updated step.

Loop for each step of episode, $t = T-1, T-2 \cdots 0$:

$G = \gamma (G + R_{t+1})$

$\underset{\text{updated}}{Q_\pi (S_t, A_t)} = (1-\alpha) \underset{\text{old}}{Q_\pi (S_t, A_t)} + \alpha G$

$\pi (S_t) = \underset{a \in A(S_t)}{\text{argmax}} \, Q_\pi (S_t, a)$

Note: Here $(1-\alpha)$ and α weights are used in case of Exponentially weighted average.

For Sample mean average, we can replace α by $\frac{1}{n}$ where $n =$ the nth instance or for state S_t, action A_t among all episodes.

\# The intuition behind this is that:

$Q_\pi (S_t, A_t) = Q_\pi (S_t, A_t) + \alpha (G - Q_\pi (S_t, A_t))$

for the error.

So 'α' is the weighting factor to improve the current action-state value function by.

Q2 we initially choose a random state action pair which is then followed by reward and proceeding towards the next state.

Root node is the top most node which is
State
an action - value pairs.

$q_{\pi}(s, a)$

r_1

s → state nodes

→ State - action nodes

terminal state.
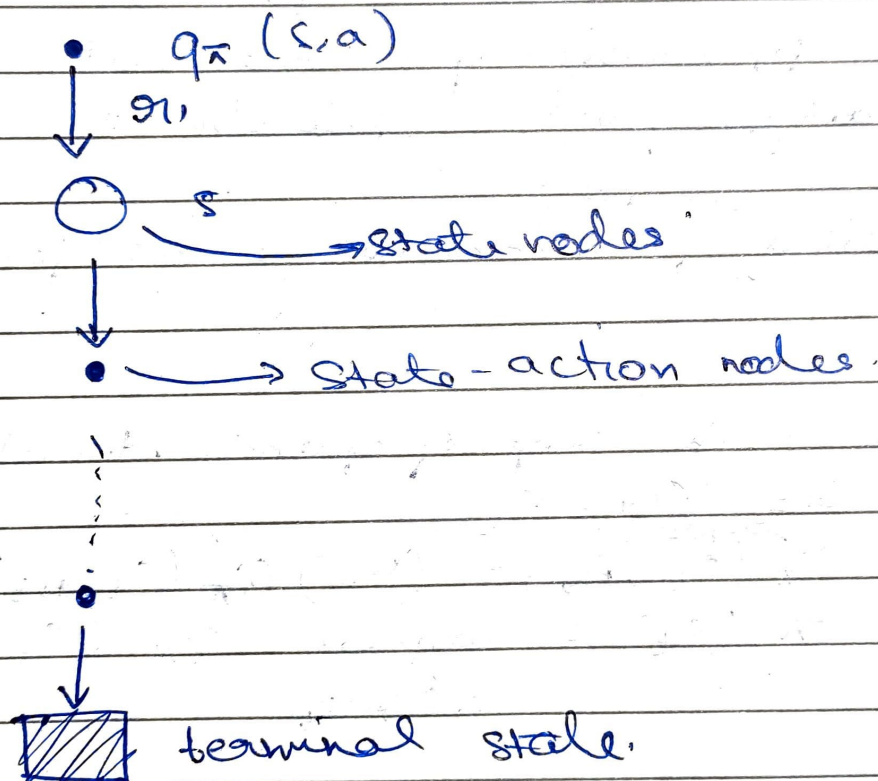
Q3. Equation analogous for action-values $Q(s, a)$ would be:

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

Here $Q(s, a)$ is the state-action pair and $\mathcal{T}(s, a)$ is the set of all time steps when $S_t = s$ and $A_t = a$.

Q5 TD learning would be very efficient in the case mentioned because:

→ Since only a small part of the route has changed, thus keeping most of the states similar in the new as well as old problem. ~~Also~~

→ Since the device has prior experience on the initial route this state-value func. of TD for old problem would have already converged to optimal values. Also TD bootstraps thus original state values are retained

→ We can update the state values for other/prev. states during the episode unlike Monte Carlo, wherein we need to wait for the episode to end.

Even if the initial state values are close to the true values, the same could be seen and they would converge faster.

Q6 **(a)** left graph: Estimated values Graph

Right graph: Empirical Rms error, avg over

all states

6.3

we have the following hyperparameters :

- $\alpha = 0.1$

- episodes = 100

- Reward to each state for transition to each state

except terminating right state = 0.

- Reward for transition to terminating right = 1

→ Updates in state-value func are as follows:

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

→ Initially, $V = [0, 0.5, 0.5, 0.5, 0.5, 0.5, 0]$

→ we start at state : C

Thus

→ both right and left action.

$$V(C) = 0.5 + 0.1 \left(0 + 0.5 - 0.5 \right)$$

$$= 0.5$$

thus $V(C)$ is unchanged

Now we can either go left or go right

~~V(C) so V(C) is also unchanged~~

In the 1st ep, when we go to A, i.e.

left terminal state, reward is 0.

Also ~~Also~~ as value function for
left terminal state is 0 thus,

~~V(A) = 0.5 + 0.1~~

$$V(A) = V(A) + 0.1 (0 + 0 - V(A))$$

$$V(A) = 0.5 + 0.1 (-0.5)$$

$$\Rightarrow 0.45$$

Thus, we can conclude that only ~~only~~
state value of A gets changed.

The decrease is 0.05 i.e there
is a 10% deduction in the state
value of A.

6.4

The conclusions about the performance
of each algorithm are dependent
on the ~~value~~ range of α that we look
to observe in.

TD(0) would perform poorly for larger
values of α as then larger jumps/
oscillations would start to occur and
it would be hard to get a
better result through as convergence
would become difficult in this
case.

whereas TD(0) would always perform better for smaller learning rate (α) relative to Monte-Carlo algorithm.

Also a specific alpha could be found where these algos might perform relatively well.

6.5

As 'α' increases, oscillation might start to occur in the errors of TD learning algo. These oscillations or jumps would prevent the algorithm from converging.

This phenomena is true for all larger values of α and is not a function of how the value function is initialized.

jumps due to larger α.

⌐ point of convergence.
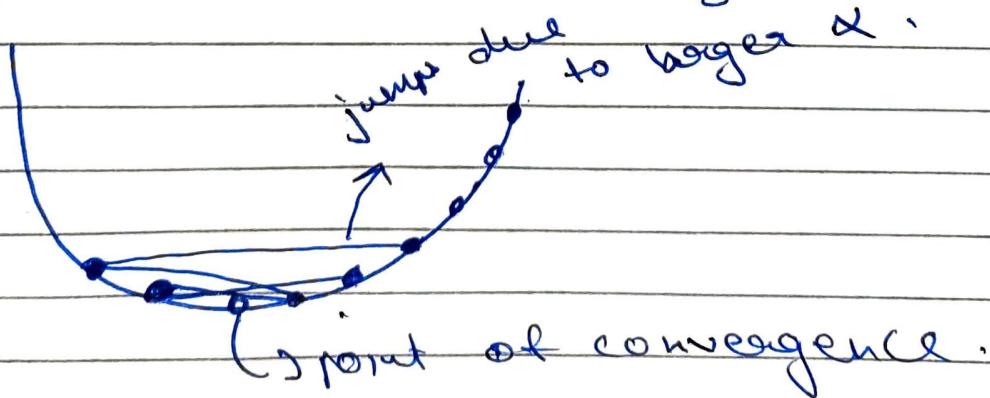
Q2 Even if the action selection is greedy, Q-learning and SARSA would be different as In Q-learning, $Q(s,a)$ is updated using greedy approach i.e by finding the best action for which target is max. ($\arg\max_{a \in A(s)} (R + Q(s', a'))$). here first Q values are updated and then the next action is picked whereas In SARSA, we pick the next action from given the Q-values according to the next state. Further, the Q-values can now be updated according to the next-action.

Thus, due to these reasons Q-learning is different from TD learning