01 Incremental update for mean calculation

> The only change in the pseudocode given is to recalculate the mean using its recursive definition rather than averaging returns over and over again.

This can be done by keeping a count (S_t, A_t) variable that keeps track of how many times the pair (S_t, A_t) has appeared

Change in pseudocode

:
:
:

unless the pair (S_t, A_t) appears in $S_0, A_0, S_1, A_1 \ldots S_{t-1}, A_{t-1}$

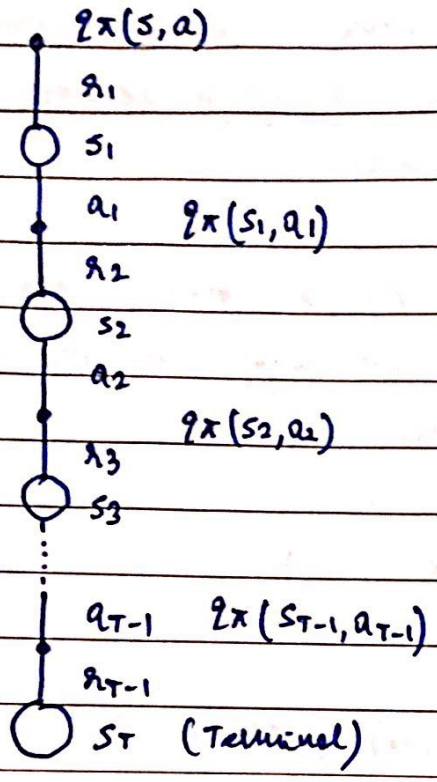
$$Q(S_t, A_t) \leftarrow \frac{Q(S_t, A_t) * \text{count}(S_t, A_t) + G_t}{\text{count}(S_t, A_t) + 1}$$

$$\text{count}(S_t, A_t) \leftarrow \text{count}(S_t, A_t) + 1$$

$$\pi(S_t) \leftarrow \text{argmax}_a Q(S_t, a)$$

This is equivalent to first calculating sum of previous returns and then adding the current return G_t followed by dividing it by the count of the pair + (1) since it is seen now.

Q2 Backup diagram for MC estimation of q_π



Q3 Equation Analogous for action-values $Q(s,a)$ instead of state values $V(s)$ → given returns generated using b

[redacted]

The eqn for $V(s) = \sum_{t \in \mathcal{T}(s)} \frac{\rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$

For $Q(s,a)$, we define $\mathcal{T}(s,a)$ as the set of all timesteps (s,a) as a pair are visited

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

Q5 As mentioned in the hint,

considering the scenario when I move to a new building and a new parking lot but while returning home, I use the same highway.

Since we have already used the highway a lot of times from the old office → we have a good estimate of the time it takes to reach home through the highway.

Now, when I need to estimate the time to reach home from the new building, I can either

a) wait for the whole episode to end aka reach home and use this total time to estimate the collect time

OR

b) I can estimate the collect time by observing time till the highway and then using my previous estimate of time from highway to home.

Clearly, (b) is BETTER → ∴ TD updates are better

Q6 The first episode leads to staying from A and then ending in the left terminal state.

(6-3) This because for all other states B, C, D

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

R_{t+1} → 0; $V(S_{t+1})$ → equal; $V(S_t)$ → equal

∴ no update

However for state A

$$V_1(A) = V_0(A) + \alpha [0 + \gamma(0) - V_0(A)]$$

$$V_1(A) = (1-\alpha) V_0(A) = (1-0.1) V_0(A)$$

$$V_1(A) = 0.45$$

Amount changed = $V_1(A) - V_0(A) = 0.45 - 0.5 = -0.05$

6.4

Yes, conclusions about which algorithm is better will be affected if a wider range of α values were used

From the graph given

TD is better for α greater than 0.1

MC is better for α less than 0.05

Moreover, decreasing α (in general) makes the curve more smooth but also leads to slower convergence.

Also, in general, TD is better than MC regardless of which α is chosen → benefits outweigh

6.5 In the example given,

State C is initialized to its true value

while other states are NOT.

In the beginning, updates start to happen which leads to 2 things

i) values for other states become more accurate

↳ decreases RMSE

ii) value for state C is updated which makes it less accurate → increases RMSE

explains going down and then going up again

This behaviour really depends on how the approximate value function is initialized as seen by initialization of C in the above case.

Q8. YES, Q-learning is exactly the same algorithm as SARSA if action-selection is greedy.

~~They~~ ~~that~~

This is TRUE by virtue of their formulations

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

↓ Q-learning

↕ ONLY DIFFERENCE

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S', A') - Q(S_t, A_t) \right]$$

↓ SARSA

For greedy selection

$$Q(S', A') = \max_a Q(S', a)$$

$\therefore$ SAME