Reinforcement Learning
Homework 3.

1. Monte Carlo ES using incremental approach:

(Ex 5.4) We know,

$$Q_{n+1} = Q_n + \frac{1}{n}\left[R_n - Q_n\right]$$

monte carlo ES can be estimated using similar incremental approach, as follows:

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

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

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

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

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

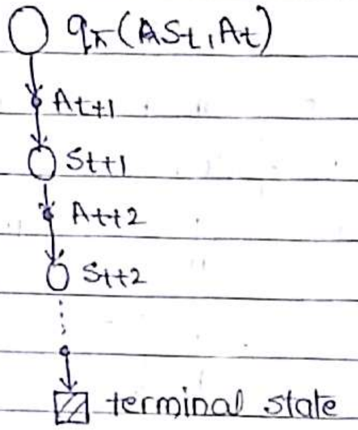
$$Q_n(S_t, A_t) = Q_{n-1}(S_t, A_t) + \frac{1}{n}\left[G_n(S_t, A_t) - Q_{n-1}(S_t, A_t)\right]$$

Pseudocode:

loop forever (for each episode):
- choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly
- generate an episode from S_0, A_0
- $G \leftarrow 0$
- loop for each step of episode t
  - $G \leftarrow \gamma G + R_{t+1}$
  - unless the pair S_t, A_t appears in the episode
    - append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow Q(S_{t-1}, A_{t-1}) + \frac{1}{t}\left[G - Q(S_{t-1}, A_{t-1})\right]$
    - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

(Ex:5.3)

2. Backup diagram for Monte Carlo Estimation of q_π:
Monte Carlo diagram shows only those sampled on the one episode, and it goes all the way to the end of the episode. Also, estimation for each state are independent, i.e., they don't bootstrap.



(Ex:5.6)

3. Eq 5.6:

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

In terms of $Q(s,a)$

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

Where,
$\mathcal{T}(s)$ is the set of all time steps in which state s is visited.
$T(t)$ is the time of termination
ρ is the importance sampling ratios.

(Ex:6.2)

5. Since only the initial state is different & we expect in general that episodes will be same later in the episodes while moving to new building or parking lot, hence temporal difference would be better than MC.

Since many states are same in the episode, value state function estimates for these states would be close to computi[ng] when we start afresh from new building starting with good initial guess and hence results in faster convergence.

6. (Ex: 6.3) Considering the example 6.2

$V(S_t) = V(S_t) + \alpha(r_{t+1} + V(S_{t+1}) - V(S_t))$

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

for state A

$V(A) = V(A) + 0.1(r_{t+1} + V(A+1) - V(A))$

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

$= 0.5 + 0.1(-0.5)$

$= 0.45$

this shows if we take left action in our random walk then we decrease the state value function by 0.05.

(Ex: 6.4) Small values of α helps in convergence both for temporal difference and Monte Carlo. Also small value of α is better than large value because if it causes faster learning and hence lower error. Hence the α values give similar results in both the algorithms. Hence we can't generalize which α is better for which algorithm.

(Ex: 6.5) Since α is large enough, it causes more change in the state value function for each timestep. Thus temporal difference is heavily dependent on specific returns. This results in downfall of RMS error in the graph initially and then up.
At smaller values of α, learning takes longer & hence is less sensitive to specific random step. But if the initialization

Date / /
Page No.

of value state generates linear relations about the updates during transition, they may over-estimate the result around terminal state.

(Ex: 6.12)

8. Q-learning updates Q function first as Q-learning is off policy algorithm but SARSA is on policy and it chooses A and S and then updates the Q function. Q learning selects A and S in the next iteration derived from q function. Hence they do not have same action selections and Q updates.