# CSE564: Reinforcement Learning

**Assignment-3**

Divyajeet Singh (2021529)

November 12, 2023

## Question-1

Rewrite the psuedocode of Monte Carlo ES as mentioned in Exercise 5.4. You must explain your code and why it is equivalent to the code provided in the book.

### Solution

The modified pseudocode of Monte Carlo ES is given in Algorithm 1. It includes a counter variable $N(s, a)$ for each state-action pair $(s, a)$. Due to the incremental nature of the update rule for $Q(s, a)$, the $Returns(s, a)$ function is no longer required. The modification can be seen on line 11.

**Algorithm 1** Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

1: **procedure** MONTE-CARLO-ES:
2: Initialize:
$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$N(s, a) \leftarrow 0$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
3: Loop forever (for each episode):
4: Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability $> 0$
5: Generate an episode starting from $S_0, A_0$, following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
6: $G \leftarrow 0$
7: Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
8: $G \leftarrow \gamma G + R_{t+1}$
9: Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$:
10: $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
11: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G - Q(S_t, A_t))$
12: $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
13: **end procedure**

This is equivalent to the psuedocode provided in the book as the update rule for $Q(s, a)$ is the same.

$$
\begin{aligned}
Q^{(n)}(S_t, A_t) &= \frac{1}{n} \sum_{i=1}^{n} G^{(i)}(S_t, A_t) \\
&= \frac{1}{n} \left( G^{(n)}(S_t, A_t) + \sum_{i=1}^{n-1} G^{(i)}(S_t, A_t) \right) \\
&= \frac{1}{n} \left( G^{(n)}(S_t, A_t) + (n-1) Q^{(n-1)}(S_t, A_t) \right) \\
&= Q^{(n-1)}(S_t, A_t) + \frac{1}{n} \left( G^{(n)}(S_t, A_t) - Q^{(n-1)}(S_t, A_t) \right)
\end{aligned}
$$

where $G^{(i)}(S_t, A_t)$ and $Q^{(i)}(S_t, A_t)$ are the returns and estimates of $Q(S_t, A_t)$ after $i^{th}$ occurrence of $(S_t, A_t)$.

## Question-2

Draw the backup diagram asked for in exercise 5.3.

### Solution

Exercise 5.3 asks for the backup diagram for the Monte Carlo estimation of $q_\pi$. The backup diagram is shown in Figure 1. This backup diagram is the same as that of the MC estimation of $v_\pi$, except that it starts with a fixed action $a$. The backup diagram is the entire trajectory of the transitions along a single episode, ending at a terminal state. Only the transitions sampled on the episode are shown.

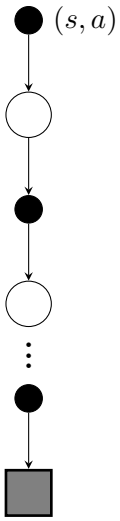


Figure 1: Backup Diagram for the MC estimate of $q_\pi$

## Question-3

Solve Exercise 5.6.

### Solution

For $V(s)$, we are given

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

where $\mathcal{T}(s)$ is the set of all time steps at which state $s$ is visited, $T(t)$ is the time of termination following time step $t$, and $\rho_{t:T(t)-1}$ is the importance sampling factor of the episode following time step $t$ till termination.
$Q(s, a)$ is the value of a state-action pair, i.e. the expected return after taking action $a$ in state $s$. So, we define $\mathcal{T}(s, a)$ as the set of all time steps at which state-action pair $(s, a)$ is visited. Then,

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

Note that the importance sampling factor is calculated from $t+1$ to $T(t)$, as the action at time step $t$ is $a$ (fixed). This can be seen as follows. Let $\rho'_{t:T(t)}$ be the importance sampling factor from $t$ to $T(t)$ for the equation of $Q(s, a)$. Then,

$$\rho'_{t:T(t)-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)} = \prod_{k=t+1}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} = \rho_{t+1:T(t)-1} \quad (\text{as } A_t = a \text{ is fixed})$$

## Question-4

Solve the blackjack game problem and generate the corresponding figures.

### Solution

The solution to the blackjack game problem is given in `main.ipynb`. The figures are also generated in the notebook.

## Question-5

Solve Exercise 6.2.

### Solution

TD methods are often more efficient than MC methods because of a few reasons. First, TD methods are fully online in nature, i.e. they update the value function after every few time steps. On the other hand, MC methods update the value function only after the end of an episode. So, in case of very long episodes, learning is delayed. Second, MC methods must ignore or discount the episodes where experimental actions are taken, which slows down the learning. TD methods, however, learn from individual transitions and hence, are less affected by this.
The scenario mentioned in the book is one where TD update is better than MC update. In it, we assume that the initial and terminal states have changed, but the highway states remain the same. Since the highway states remain the same, TD methods will accelerate the adjustment to the new state values. MC methods, on the other hand, will have to wait till the end of each episode to bring an average improvement to all states. This average improvement can also result in fluctuating value for some highway states which might not be needed.

## Question-6

Write the necessary code and generate the figures in Example 6.2. Answer the related questions asked in Exercises 6.3, 6.4, and 6.5.

### Solution

The code and figures for Example 6.2 are given in `main.ipynb`. The figures suggest that TD(0) methods are consistently better than MC methods. The answers to the questions are as follows.

#### Exercise 6.3

**Why does the first episode result in change in the value of one state only?**

The first episode results in change in the value of only one state (state `A` in the book), suggesting that the episode ends in that state. Since $\gamma$ is 1 and intermediate rewards are zero, we can expect this change to be non-zero, specifically $0.5 * (-\alpha) = 0.05$, since all values were initialized to 0.5.
If the episode ends in the right terminal state, then the value of state `E` increases by $0.5 * \alpha = 0.05$.

#### Exercise 6.4

**Does either algorithm perform better than the other on a specific values of $\alpha$? Do you think the conclusions about which one is better depends on the step-size parameter?**

No. In the plot, both algorithms are tested on very small values of the step-size parameter. A small value of $\alpha$ is necessary for convergence, and so, the algorithms are already tested on their limits. So there is no fixed value for $\alpha$ that could make a big difference for either method.

Reducing the value of $\alpha$ after a sufficiently large number of steps might help in convergence of both methods.

**Exercise 6.5**

**What is the cause of the decrease and then increase in the RMS error of TD(0) methods with different $\alpha$s? Is it a function of how the estimate of the value function was initialized?**

The decrease and then increase in the RMS error of TD(0) methods with different $\alpha$s is due to the initialization of the value function and the step-size parameter. For even smaller $\alpha$s, the algorithm should converge to lower and lower RMS errors. Moreover, the TD error can increase again when some states are overestimated and some are underestimated. This can happen when the step-size parameter is too large.

## Question-7

Write the code and generate the figures that compare the sum of rewards during episodes when using Q-learning and SARSA.

### Solution

The code and figures for the solution are given in `main.ipynb`.

## Question-8

Solve Exercise 6.12

### Solution

Even if action selection is greedy, even then Q-learning may not be equivalent to SARSA. This is because Q-learning works as follows:

- Take action $A_t$ derived from current $Q$ estimates (greedily), and observe $R_{t+1}$ and $S_{t+1}$
- Update $Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a) - Q(S_t, A_t)\right]$
- Proceed to state $S'_{t+1}$ that maximizes the updated $Q$ estimates

However, SARSA works as follows:

- Take action $A_t$ derived from current $Q$ estimates (greedily), and observe $R_{t+1}$ and $S_{t+1}$
- Update $Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a) - Q(S_t, A_t)\right]$
- Proceed to the already picked state, $S_{t+1}$

So, SARSA chooses $S_{t+1}$ with stale estimates of $Q$, while Q-learning chooses $S_{t+1}$ in the next iteration with the updated values of $Q$. This means that they might not necessarily pick the same states.