Reinforcement Learning
Exercise 5: Temporal-Difference Learning

Q1) Temporal difference vs. Monte-Carlo
a) for the original scene, Monte Carlo would perform better than TD. Because, the relevant past experience is not available. As we know, the Monte Carlo process would learn all the updates in a single episode, while for the TD process, it would take multiple runs of episodes to learn the complete journey.

Now, considering the driving home example,
Looks like for the TD process, it would be more efficient compared to Monte Carlo, because→ here, the entrance into the highway remains the same. But we need to see how to get there from the new parking slot. So, we only need to update the initial condition due to change in the new state while the past experiences remain the same. But in the case of Monte Carlo, it has to be learned for a complete episode or the complete journey.

b)
In the case of a deterministic environment, and an agent has access to the complete environment.
Example, in a game of chess, an agent wishes to evaluate a new opening strategy. Here, with the use of Monte Carlo, agents will evaluate the opening by simulating multiple games from the desired opening position while calculating the average reward. Hence, this info can be used to decide if it's a better opening strategy or not.
But, TD would only estimate the outcome of each game based on incomplete simulations.

Q.2) Q-learning vs. SARSA.
a) The concept of off-policy is → the agent is learning the value of actions in a state while following a different strategy for exploration and exploitation. Basically, is learning a policy which is completely different from Behaviour policy – the one which is used to interact with the environment..
Q-learning learns the optimal policy independently of the policy it is using to explore the environment. It evaluates value function for greedy epsilon policy.
It uses the best action leading to better value at next action, without using epsilon -greedy policy to update the action.

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
According to Bell,ams's equation →
Q(s,a) updated based on the difference between the observed reward and estimate of the future reward. Agent uses its current estimate to update its knowledge. Which makes it off-policy.

b)
We know that →

1. Q-learning updates the Q-value of the chosen action based on the maximum Q-value of the next state
2. SARSA updates the Q-value of the chosen action based on the Q-value of the actual next action taken.

But, when action selection is greedy, both algorithms choose the action with the maximum Q-value → $a' = \text{argmax}\ a' Q(s',a')$
If action selection is greedy, and both Q-learning and SARSA use a greedy policy, they converge to the same algorithm. Resulting in making the same weight updates and action selection.

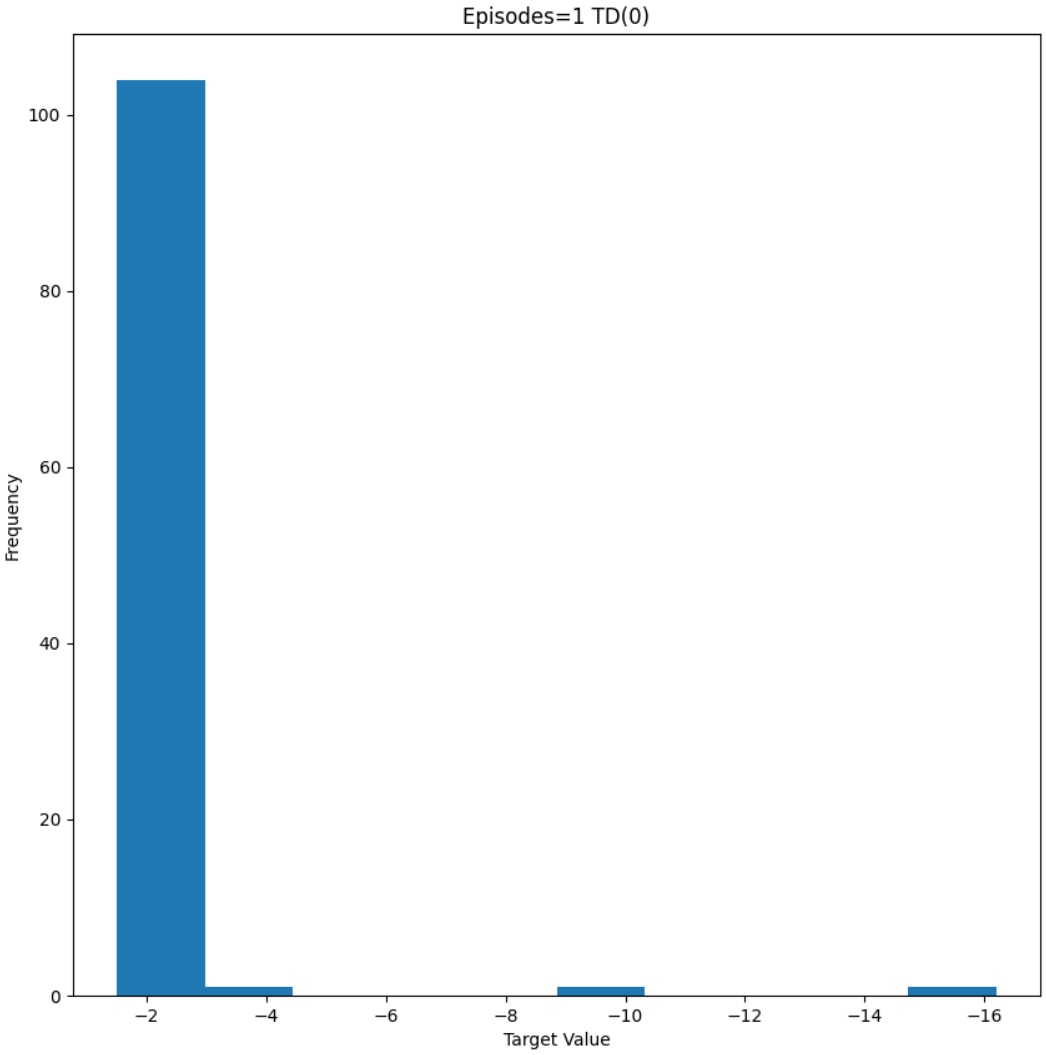
Q.3) Random-walk task
a)
The first episode ends on the left side. Starting from point A. We can see that the change in prediction was 0.5 to 0 while going from A to goal state,
V(s) = V(st) + alpha [R(t+1) + gamma * V(st+1) - V(st)]
= 0 + alpha (gamma - V(A)
= 0.1 * (0 -0.5) = -0.05
Only needed to change with value as other states had no prediction errors.

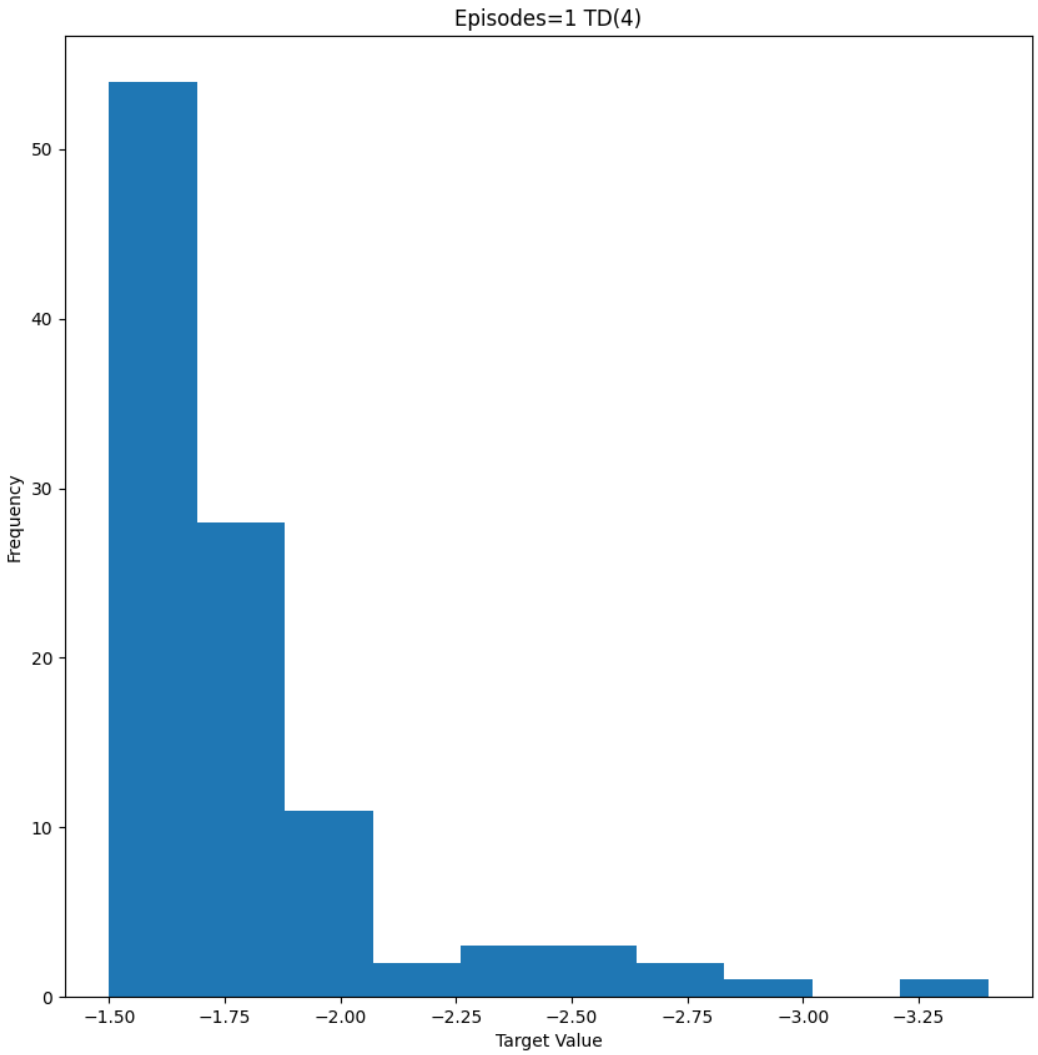
b)
Using a higher value of learning rate - alpha usually leads to a faster learning but small value of alpha is better for long term.
Based on observations from the graph, TD performs best when alpha = 0.05 and MC performs best when alpha is 0.01. But even with different sets of alpha values, TD always has improved performance. Fixed value of alpha would not make any algorithm better. Plus the error performance and alpha relation is U curve, as the alpha decreases the overall performance also reduces over a period of time.

c)
As we can see that the episode finishes on either one side or the other, estimates might change a bit due to random changes. The different value of alpha may be the reason that causes this. If we start with an initial guess like we did for state C which is exactly right, then the initial error becomes 0. (ideally) but estimates will still change due to random changes.

d)
a larger random walk task with 19 states was used in Example 7.1 because it allows for a more detailed understanding of how the algorithms behave and perform.
– When we use traces to look back , steps during learning, having more states provides better observations for analysis.
– A change in the left-side outcome from 0 to -1 in the larger walk should not have a bigger impact, as initial values were also adjusted from 0.5 to 0, and the only effect would be an overall increase in the error level due to -1 to +1 border.

Q.5)


Episodes=1 TD(0)
Frequency
Target Value
0
20
40
60
80
100
−2
−4
−6
−8
−10
−12
−14
−16



b)
We know that MC is High variance and low bias, we see the histograms with wider range of values which tells about the variability in returns observed during different episodes.

TD → it introduces bias, and makes estimates based on current approximations,
It has low variance compared to Monte carlo.

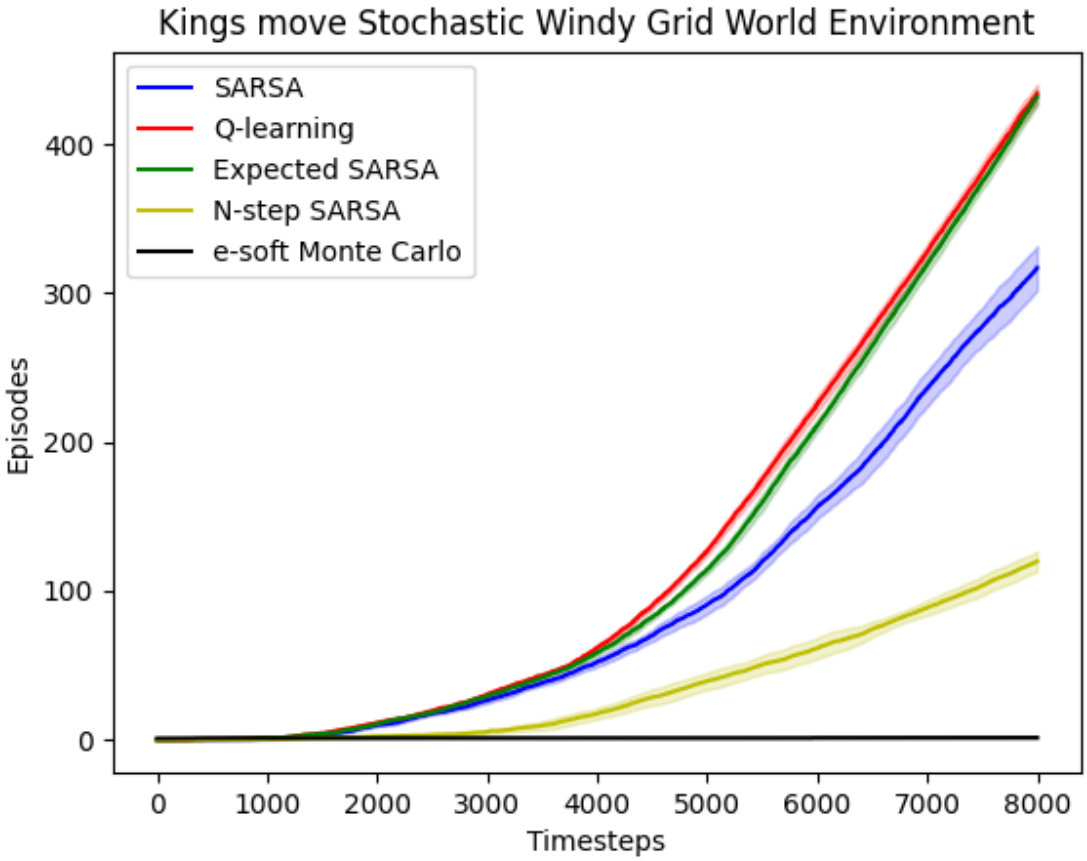
Histograms are less spread out.

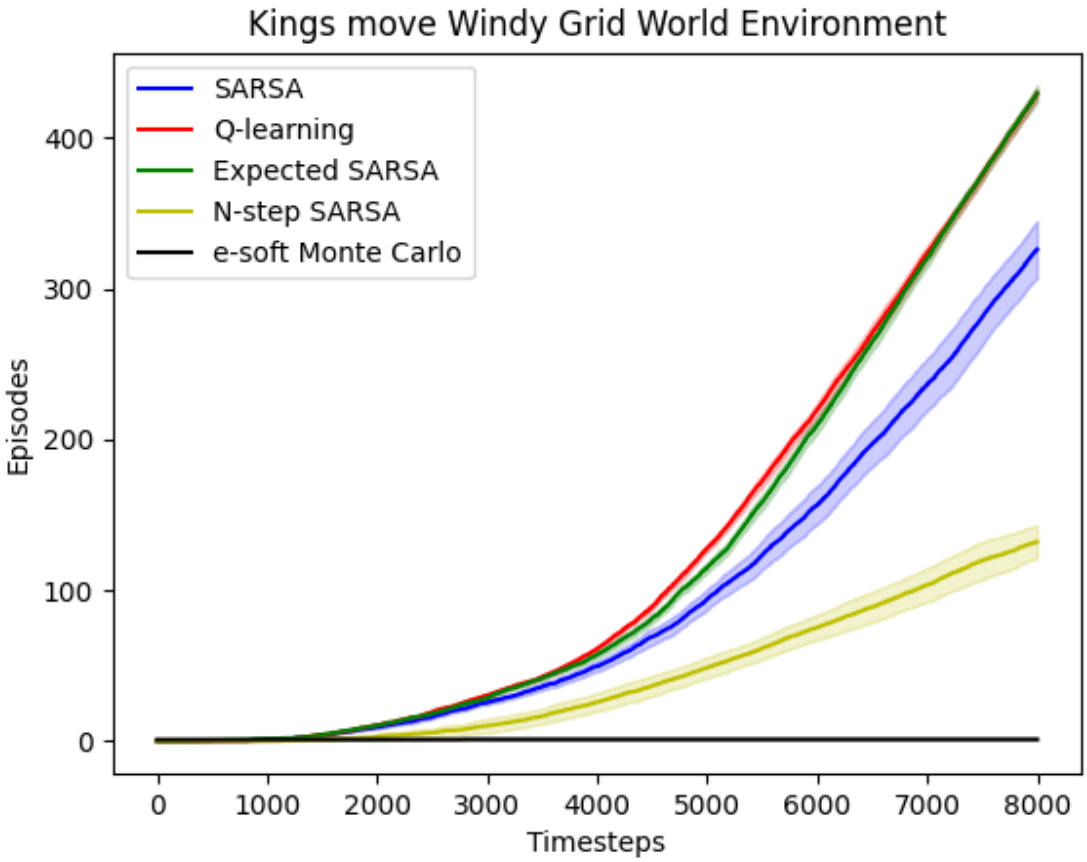
Effect of Training →
Example episode 1 →
We can expect higher variance for MC while TD can be biased due to limited samples.

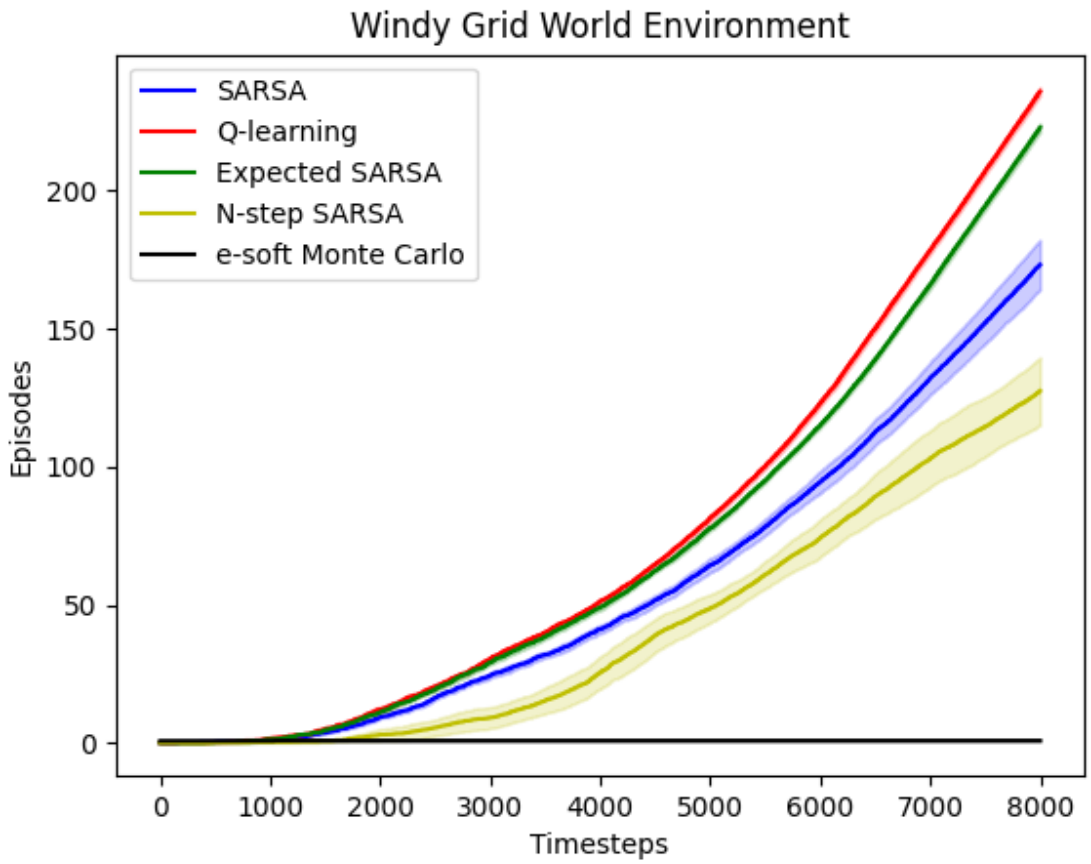
Example episode 10 / 50 →
The variance might reduce in MC with more episodes.
But, in the case of TD, they might have a better trade - off between bias and variance with more training.

c)
If you switch from off-policy methods (like Q-learning) to on-policy methods (like SARSA) in the context of control →
– on-policy methods directly update the policy that is used to generate data, while off-policy methods use a separate policy for exploration and a separate policy for learning.
– on-policy methods tend to converge more slowly than off-policy methods as they are sensitive to the exploration-exploitation tradeoff.
– If On policy explore too much, they will not learn the optimal policy quickly. If they exploit too much, they may get stuck in a local optimum
– On-policy methods may converge more slowly due to bias
– we can use off-policy for deterministic problems and off-policy for stochastic problems.

Q.4)


Kings move Stochastic Windy Grid World Environment
SARSA
Q-learning
Expected SARSA
N-step SARSA
e-soft Monte Carlo
Episodes
0
100
200
300
400
0
1000
2000
3000
4000
5000
6000
7000
8000
Timesteps


Kings move Windy Grid World Environment
SARSA
Q-learning
Expected SARSA
N-step SARSA
e-soft Monte Carlo
Episodes
0
100
200
300
400
0
1000
2000
3000
4000
5000
6000
7000
8000
Timesteps


Windy Grid World Environment
SARSA
Q-learning
Expected SARSA
N-step SARSA
e-soft Monte Carlo
Episodes
0
50
100
150
200
0
1000
2000
3000
4000
5000
6000
7000
8000
Timesteps

Q 5)