CS5180 – Ex6

Sudhendra Kambhamettu
NUID: 002786797

Q1.

Part a)

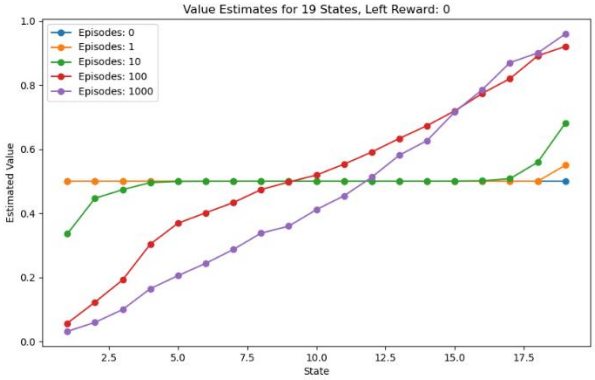
Q-learning is considered an off-policy control method because it learns the optimal policy independently of the agent's actions. It does this by estimating the value of the optimal action-value function, $Q^*(s, a)$, regardless of the policy being followed during the learning process. This characteristic enables Q-learning to evaluate the potential of all actions from a given state by utilizing the Bellman equation as a basis for updates, thereby learning the best action to take in each state without necessarily taking those actions. By updating its estimates based on the maximum future rewards predicted, rather than the rewards of the policy currently being followed, Q-learning can discover the optimal policy even when the agent is exploring the environment in a non-optimal manner.

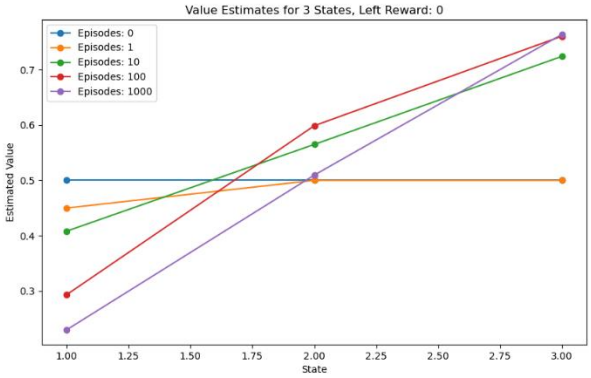
Part b)

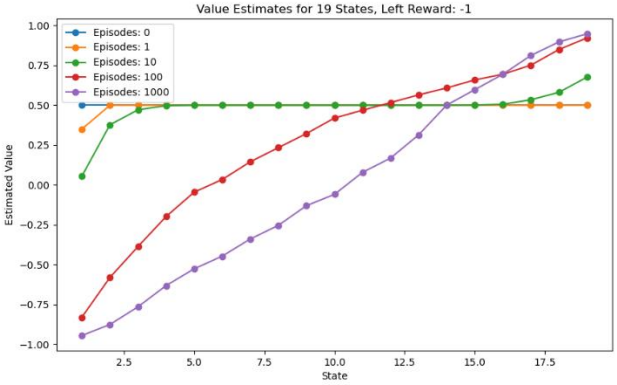
When action selection is greedy, Q-learning and SARSA still differ fundamentally due to their approaches to learning: Q-learning is off-policy, while SARSA is on-policy. Even with greedy action selection, Q-learning updates its action-value function based on the maximum reward that could be obtained from the next state, effectively learning the optimal policy independent of the agent's current strategy. In contrast, SARSA updates its action-value function based on the action actually taken from the next state, which, despite being greedy in this context, means it learns a policy consistent with its current behavior. Therefore, while both algorithms may make similar action selections under a greedy policy, their updates and consequently the learning process will differ. Q-learning aims at learning the optimal policy (what is the best possible action to take), whereas SARSA learns a policy that is more reflective of its current strategy, including the exploration strategy. This difference can lead to variations in the learning trajectory and the eventual policy each algorithm converges to, even if both are utilizing greedy action selection.

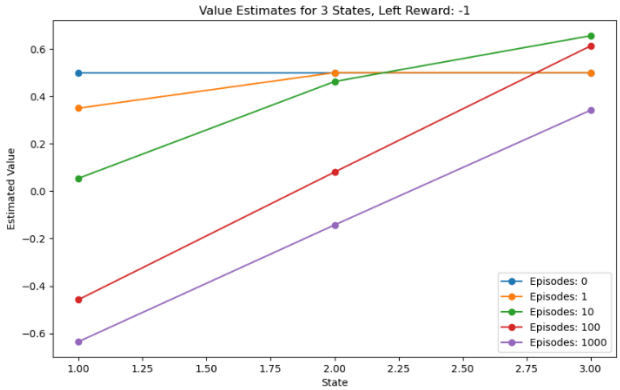
Q2.

Plots


Value Estimates for 19 States, Left Reward: 0
Episodes: 0
Episodes: 1
Episodes: 10
Episodes: 100
Episodes: 1000
Estimated Value
State


Value Estimates for 3 States, Left Reward: 0
Episodes: 0
Episodes: 1
Episodes: 10
Episodes: 100
Episodes: 1000
Estimated Value
State


Value Estimates for 19 States, Left Reward: -1
Episodes: 0
Episodes: 1
Episodes: 10
Episodes: 100
Episodes: 1000
Estimated Value
State


Value Estimates for 3 States, Left Reward: -1
Episodes: 0
Episodes: 1
Episodes: 10
Episodes: 100
Episodes: 1000
Estimated Value
State

Written

1. The larger random walk task with 19 states, as opposed to 5, was likely used in Example 7.1 to provide a more complex and nuanced environment for evaluating the learning algorithms. A larger state space introduces more granularity and a wider range of situations for the algorithm to learn from, making it a better test of the algorithm's capability to generalize and accurately estimate value functions over a diverse set of states. This complexity can help in distinguishing the performance of different approaches or parameter settings more clearly than a simpler task with fewer states.
2. A smaller walk with fewer states would likely have shifted the advantage to a different value of **n** because the complexity and the breadth of the state space directly impact how information propagates during learning. In a smaller state space, the updates would propagate more quickly, potentially favoring algorithms or settings that can adapt more rapidly to new information. This might result in different optimal settings for the learning rate, **α**, or the discount factor, **γ**, to balance the trade-off between exploration and exploitation efficiently.
3. The change in the left-side outcome from 0 to −1 in the larger walk introduces a penalty for reaching the left terminal state, making the task more challenging by adding a negative consequence to certain actions. This modification likely affects the learning dynamics by emphasizing the importance of avoiding negative outcomes, possibly requiring a different strategy or parameter tuning to optimize learning. This change could influence the optimal value of **n** in n-step methods or the learning rate in methods like TD(0) and Monte Carlo, as it alters the reward structure the algorithms need to adapt to.
4. Therefore, the introduction of a larger state space and the adjustment of the reward structure both play significant roles in determining the optimal parameters for learning algorithms. The complexity added by these factors can influence the best value of **n** by affecting how quickly and effectively the algorithms can learn from the environment, adjust their value estimates, and converge towards optimal policies.

Q3.

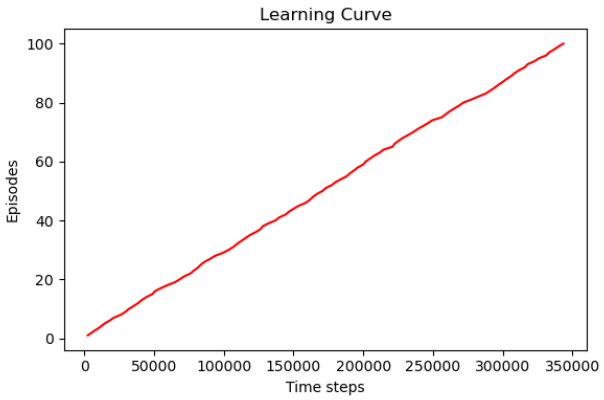
Part a)

Implementation in windy_env.py

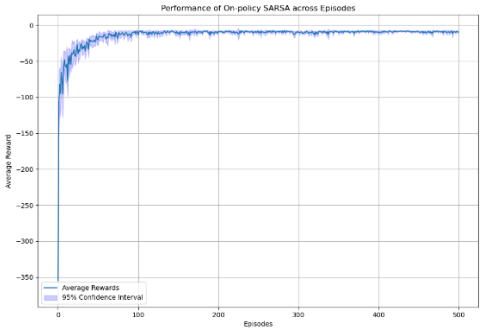
Part b)

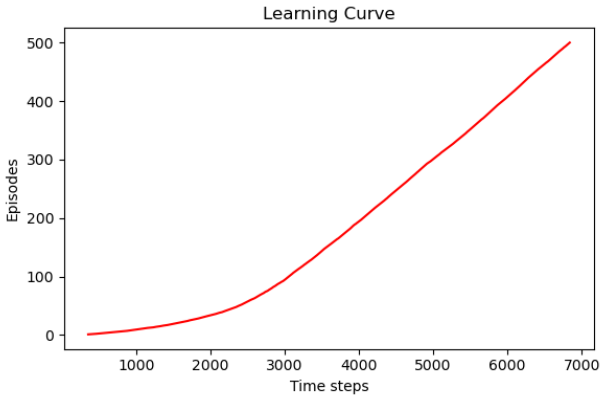
<u>On-policy Monte-Carlo control</u>

Note: Due to time constraints, I ran MC control for 100 episodes & it is evident that it performs worse than all TD based methods as outlined in the textbook.

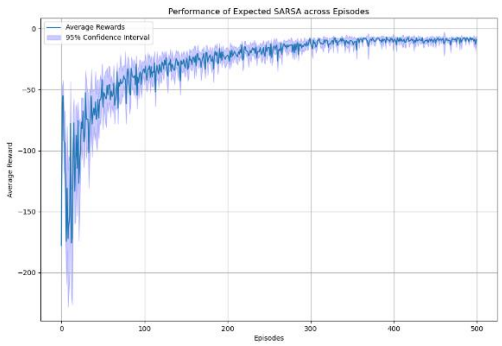


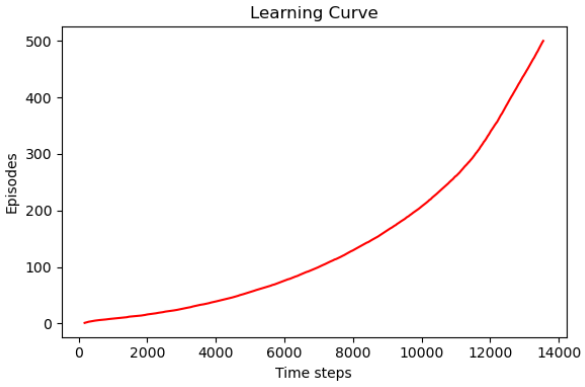
SARSA



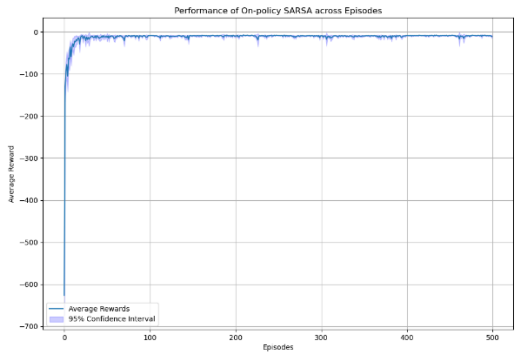


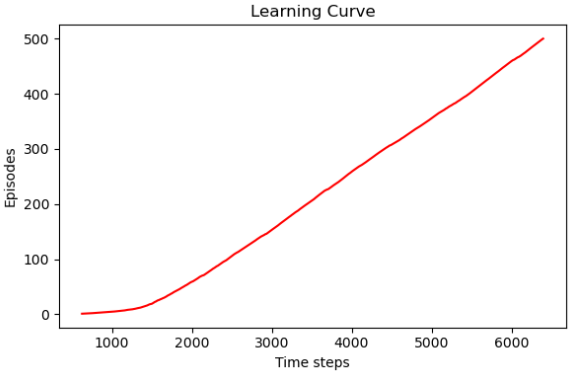
Expected SARSA



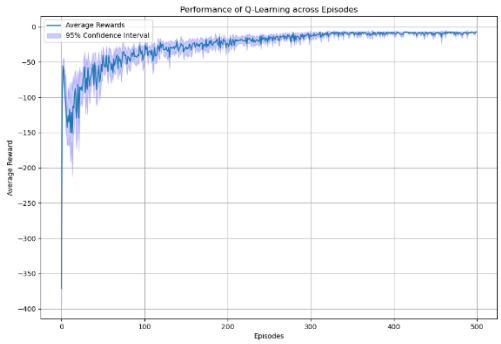


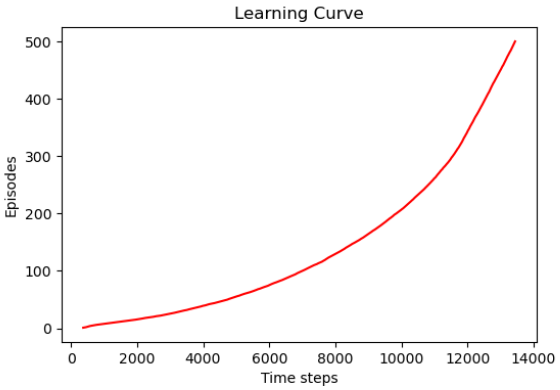
n-step SARSA





Q-Learning

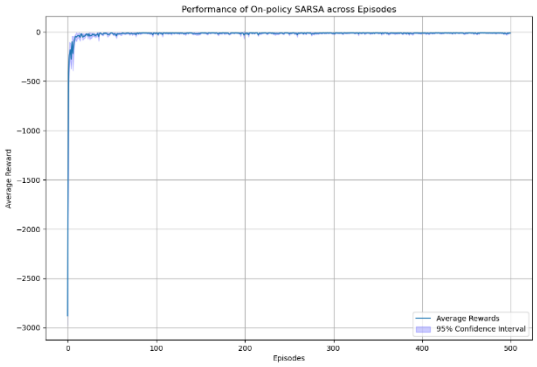


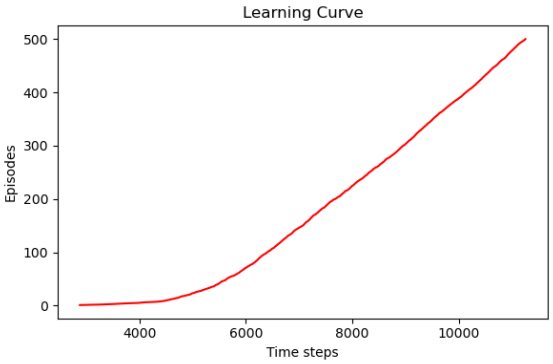


Part c)

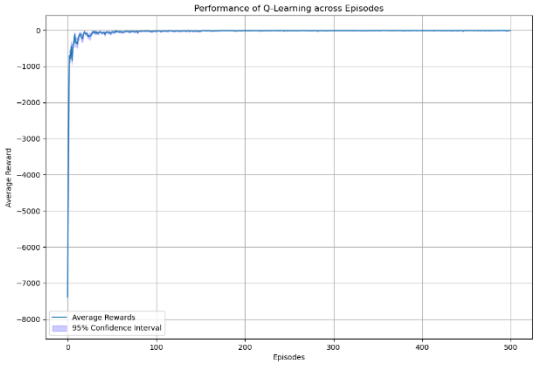
Without 9th action

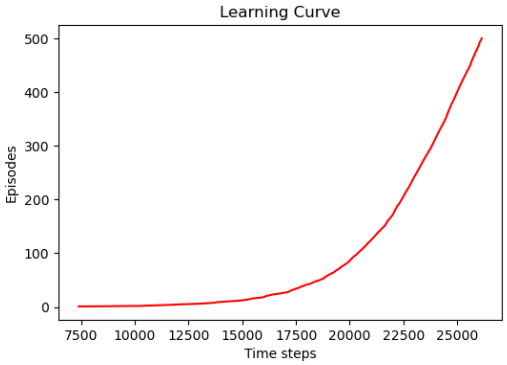
On-policy TD control SARSA





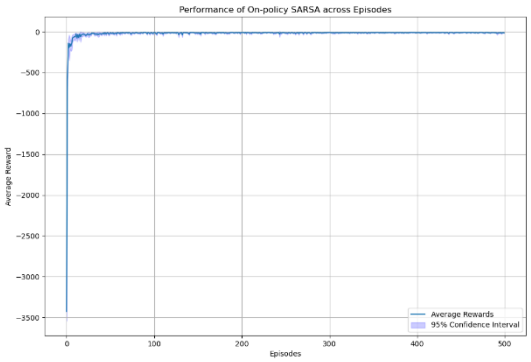
Off-policy TD control Q-learning

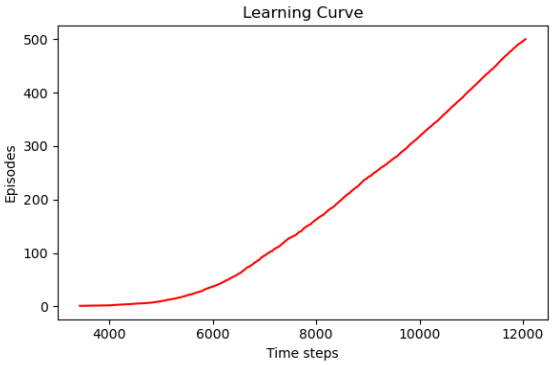




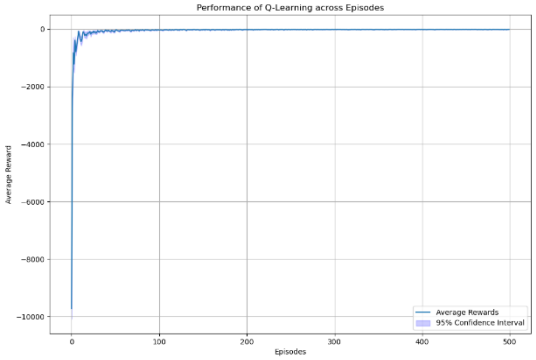
With 9th action

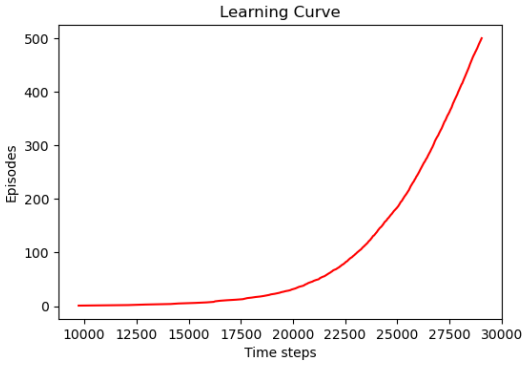
On-policy TD control SARSA





Off-policy TD control Q-learning

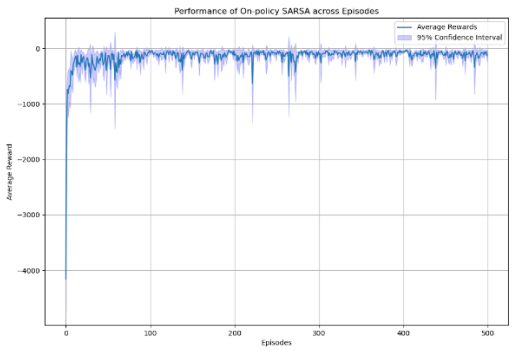


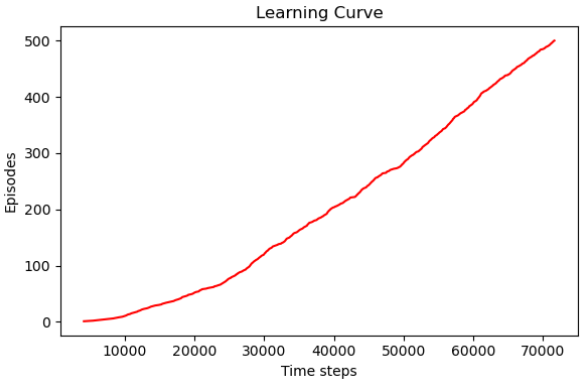


There isn't much of a difference when a 9th action is added.

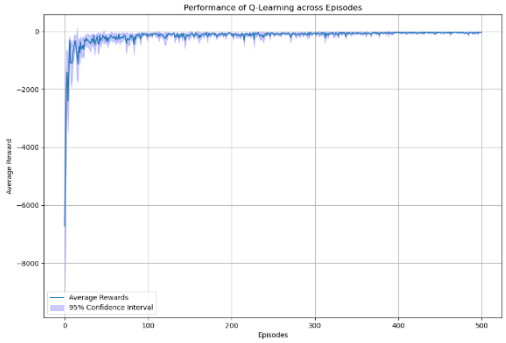
Part d)

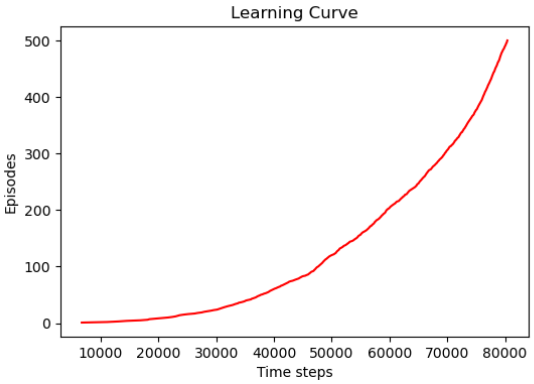
On-policy TD control SARSA





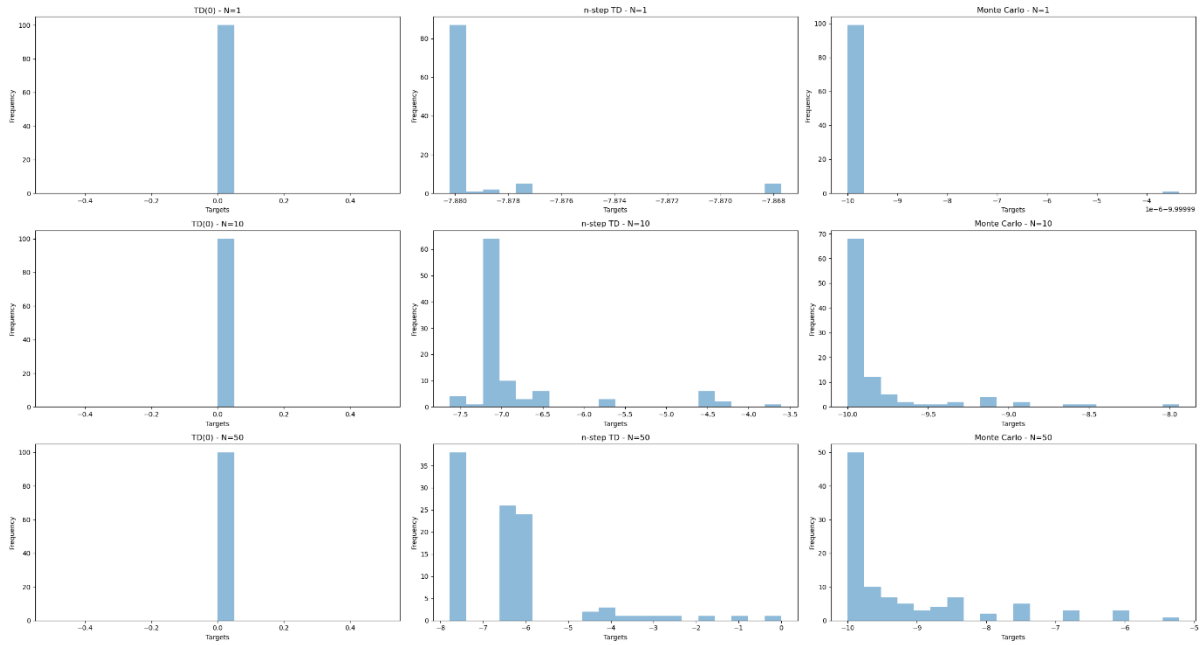
Off-policy TD control Q-learning





Q4.

Part a)



Part b)

Firstly looking at the overall findings, they indicate that while n-step TD captures more diverse outcomes with increasing N, TD(0) maintains consistency with low variance across training episodes, and Monte Carlo's variance increases with more training. These findings reflect the expected trade-off between bias and variance as the policy evolves and incorporates more information about the dynamics of the environment and more exploration.

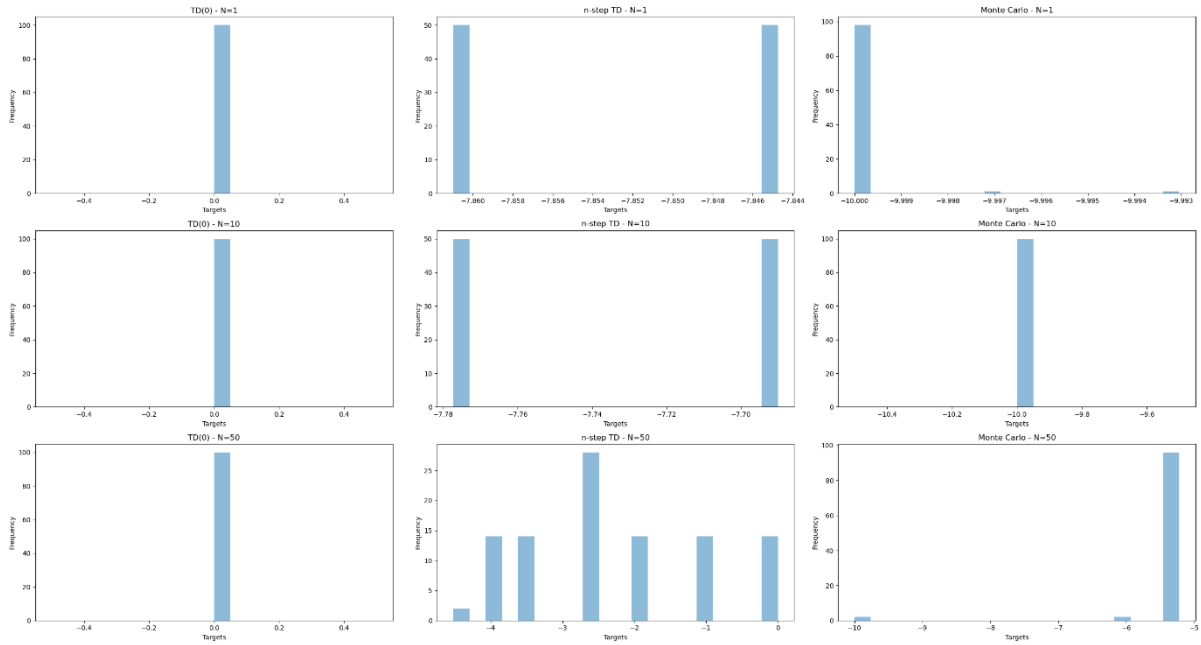
Individually,

TD(0): There appears to be little fluctuation as the learning objectives are grouped around a single value. This suggests that there may be a significant bias resulting from bootstrapping from previous estimated values, and that this bias may not alter with more training. Since TD(0) relies more on other value estimations than on actual returns, it is likely to have more bias and lower variance.

n-step TD: As the policy grows better, the variation in learning objectives rises, suggesting that the approach is collecting a wider range of outcomes and perhaps reducing bias. As a result, when the number of steps taken into account in the update grows, n-step TD may be able to reduce bias with greater variation.

Monte Carlo: Low variance is shown in the histograms for lower N values, which is unusual for Monte Carlo techniques. But there is an increase in variance at N = 50, which is consistent with the predicted behavior of full return Monte Carlo techniques instead of bootstrapping. As a result, Monte Carlo has low variance at first, increasing at N = 50 to demonstrate a decrease in bias as the policy begins to explore a wider range of behaviors with more training.

Part c)



All things considered, TD(0) exhibits minimal variation and consistent behavior at all training levels.

The convergence of learning objectives with more training is demonstrated by the n-step TD and Monte Carlo approaches, suggesting better policy refinement and maybe less bias.

As training increases, Monte Carlo shows more variation, suggesting more experimentation and learning from a wider range of events.

In conclusion, the control scenario shows tendencies that are similar to those of the preceding scenario, with the policy evolving toward more variance and convergence as demonstrated by the n-step TD and Monte Carlo approaches. The Monte Carlo approach, however, behaves slightly differently; at N=1, it displays unexpectedly low variance at first, but as N grows, it begins to resemble the predicted behavior more and more.