# Assignement: ex5

Ian Steenstra

October 2024

## 1 Problem 1

Yes, Monte Carlo may be better in some non-Markov situations.

For example, imagine a simple gambling game where there are two states: "Playing" and "Terminal"; as well as two actions: "Bet High" and "Bet Low". In the "Playing" state, regardless of the action chosen, the transition to "Terminal" occurs randomly, with a probability of 0.5. If the transition does not happen, the agent stays in the "Playing" state. The rewards in the "Playing" state are always zero. Only in the "Terminal" state is a reward given: +1 for a win or -1 for a loss. Whether the agent wins or loses is independent of the actions taken in the "Playing" state and is random. Because the "Playing" state rewards are always zero and transitions are independent of actions, TD learning cannot learn anything useful in this game. It will constantly update state values based on the next state's estimate, but that estimate will fluctuate randomly as it receives only winning (+1) or losing (-1) updates in the "Terminal" state. No consistent pattern will emerge for guiding action selection in the "Playing" state. On the other hand, Monte Carlo will correctly determine that both actions in the "Playing" state have the same expected return (zero), as wins and losses are equally likely.

## 2 Problem 2

- **(a)** Q-learning is considered an off-policy control method because its update target is derived from a different policy than the behavior policy.
- **(b)** Yes, if action selection is greedy, Q-learning and SARSA are the same algorithm. They will make the same selections and weight updates. Furthermore, if both use the same target and update rule, they become equivalent.

## 3 Problem 3

- **(a)** The conclusions about which algorithm is better would probably still hold for a wider range of $\alpha$ values, although the specific values $\alpha$ at which one is better than the other might change. It is unlikely that there is a single $\alpha$ value that would make either algorithm significantly better across all stages of learning.
- **(b)** The bump is likely due to overshooting updates early in learning at high $\alpha$ values. The initial values may affect the magnitude of the bump, but it's not solely due to initialization.

## 4 Problem 4

- **(a)** See code.
- **(b)** See Figures 1-3.

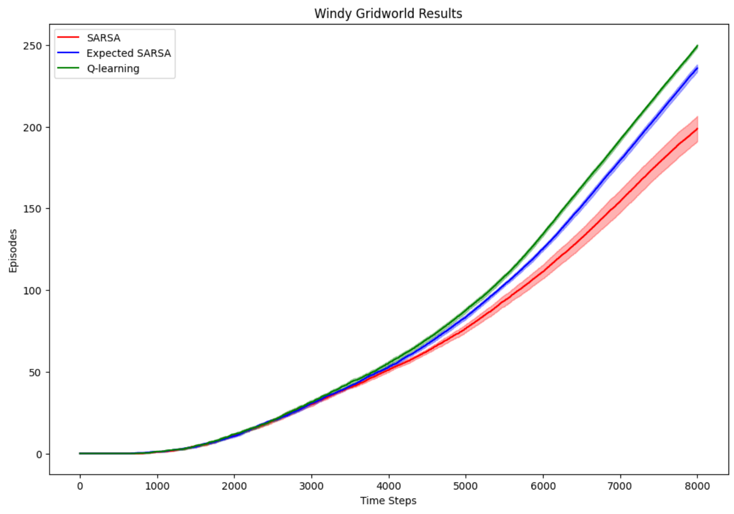

Figure 1: Problem 4(b): No Kings - 4 moves

## 5 Problem 5

- **(a)** See code and Figures 4-9.
- **(b)** During my tests, I saw the opposite of what I was expecting: the TD(0) method showed higher variance than the MC. Though, the TD(0) method did show higher bias, which was expected, as it varied further away from the true value. As N increased, the TD(0) targets got closer to

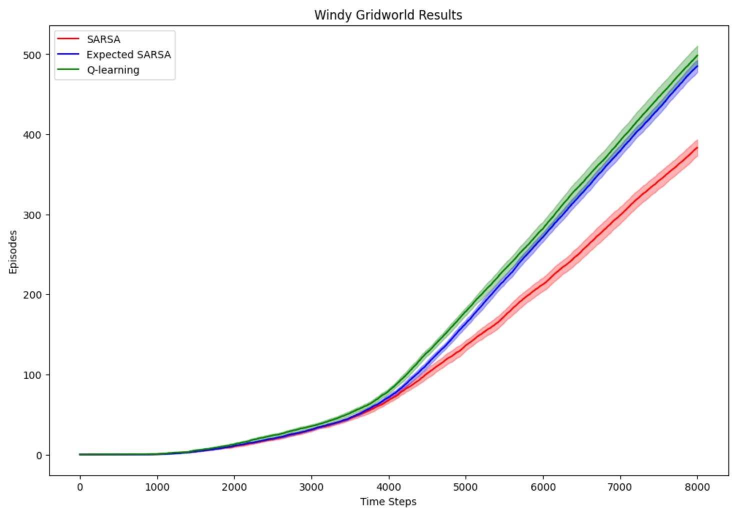

Figure 2: Problem 4(b): Kings - 8 moves

the true value, while the MC method remained basically at the true value even when N=1. The results may be indicative of an error on my side when coding, or a phenomenon of variability given the bootstrapping.

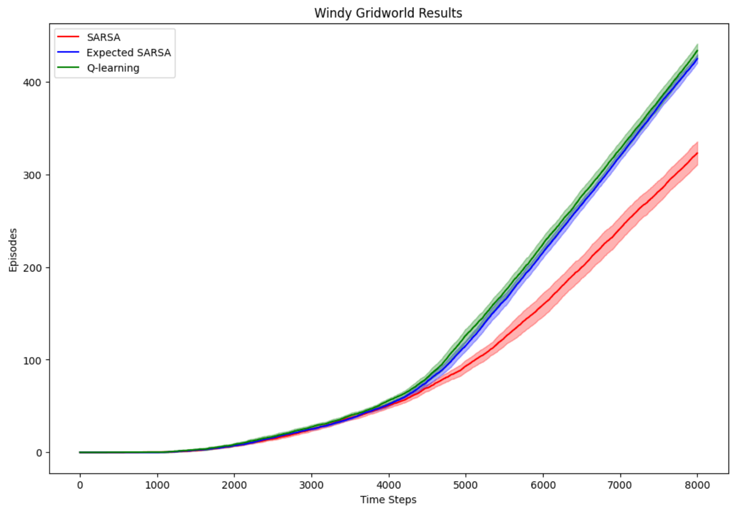

Figure 3: Problem 4(b): Kings w/ No Move - 9 moves

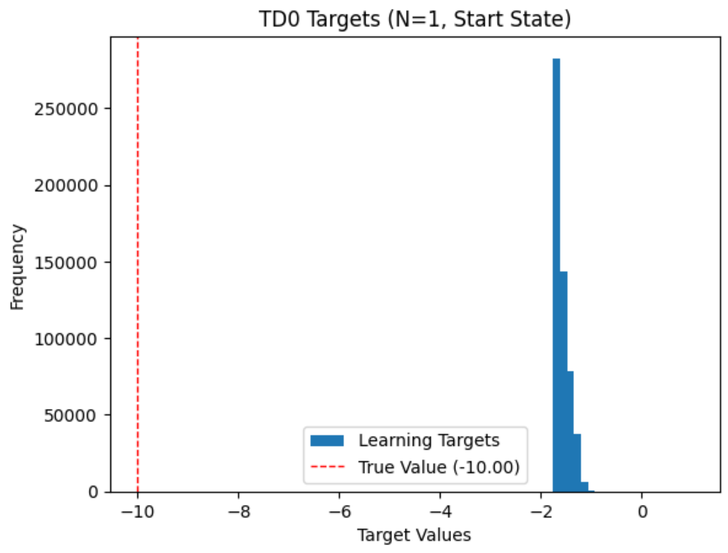

Figure 4: Problem 5(a): TD(0) Targets; N=1

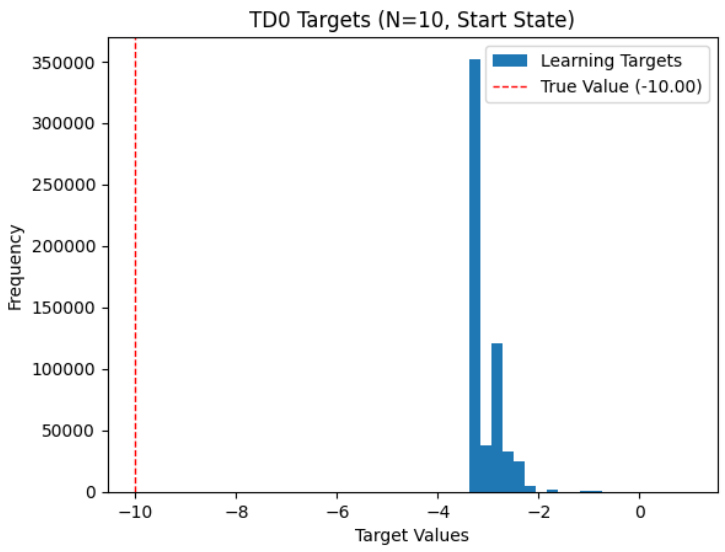

Figure 5: Problem 5(a): TD(0) Targets; N=10

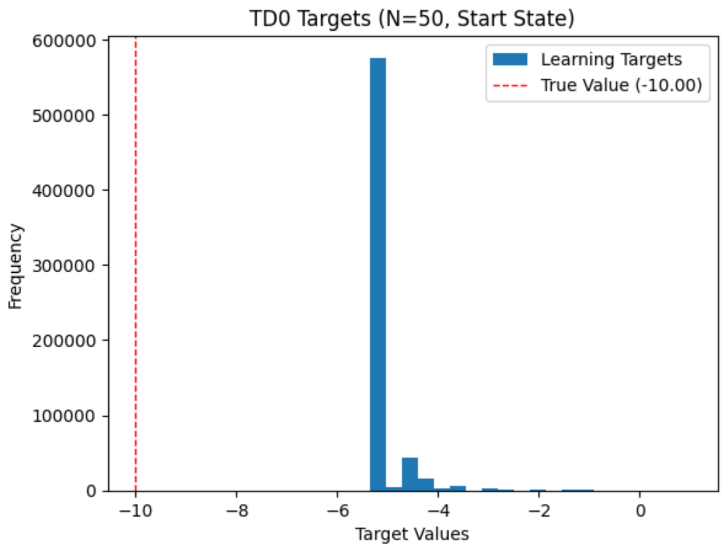

Figure 6: Problem 5(a): TD(0) Targets; N=50

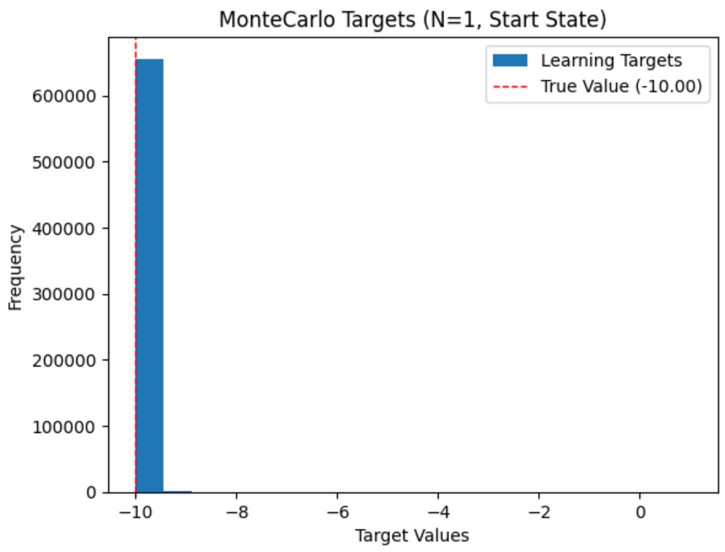

Figure 7: Problem 5(a): MC Targets; N=1

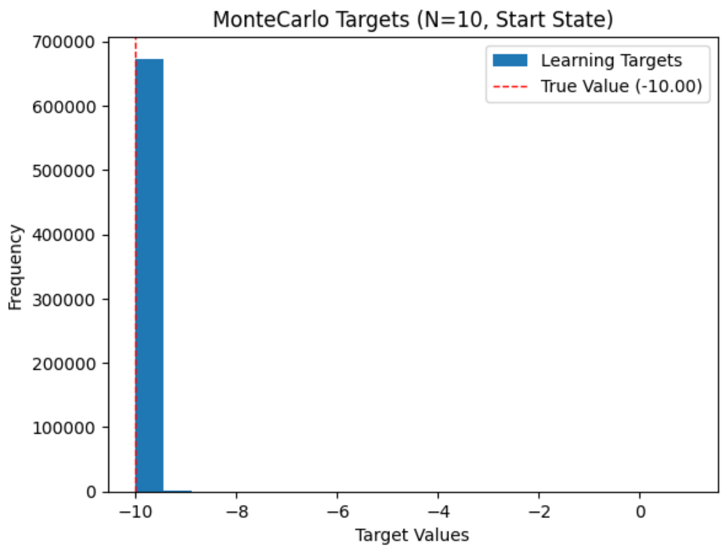

Figure 8: Problem 5(a): MC Targets; N=10

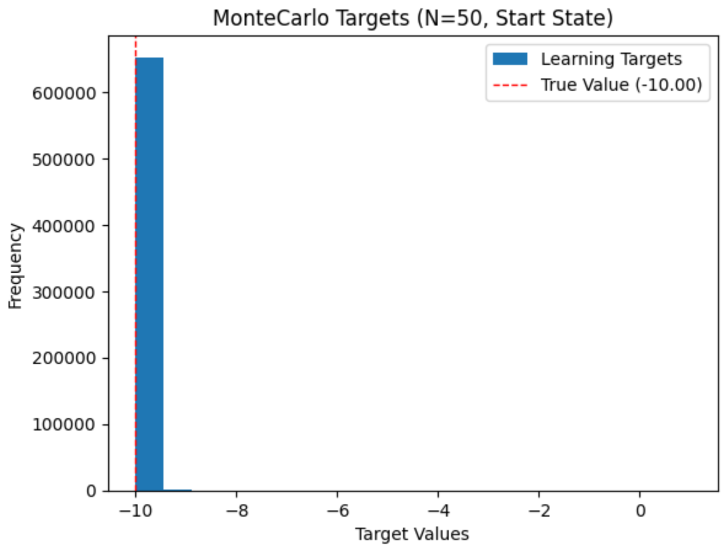

Figure 9: Problem 5(a): MC Targets; N=50