Reinforcement Learning

Assignment 03

Spring 2024

By

W.H. Sasinda C. Prabhashana-202395458

&

Premisha Premananthan - 202397583

[This document fulfills the requirements for the Reinforcement Learning Module (DSCI-6650-001) at Memorial University in Newfoundland, Canada]

PART 01

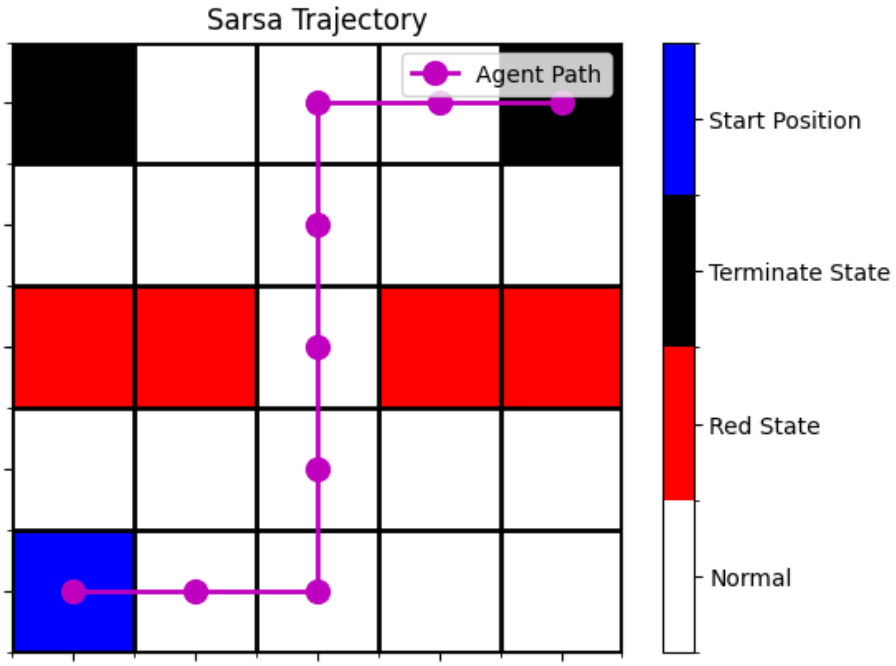


Figure 01: Sarsa Trajectory

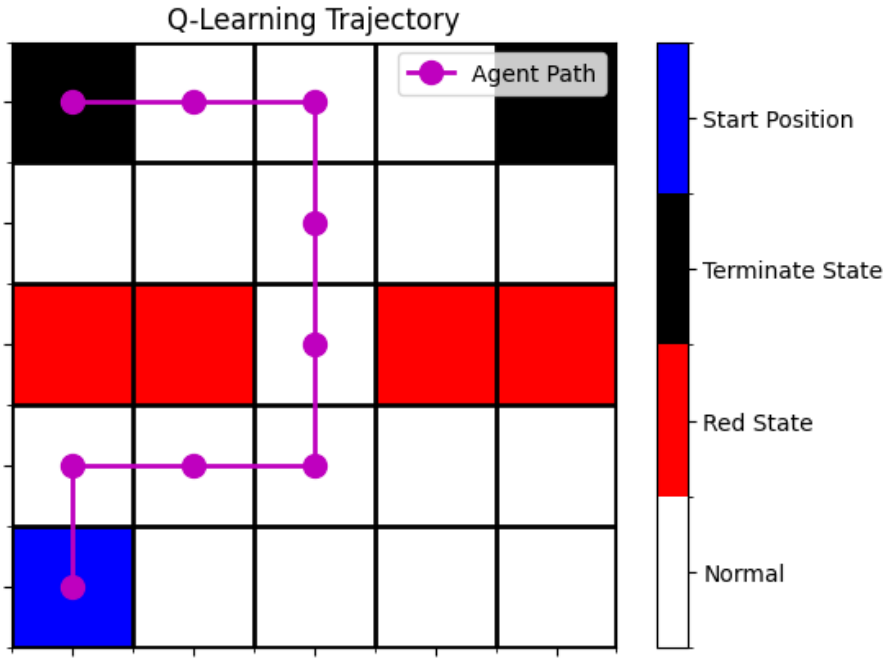
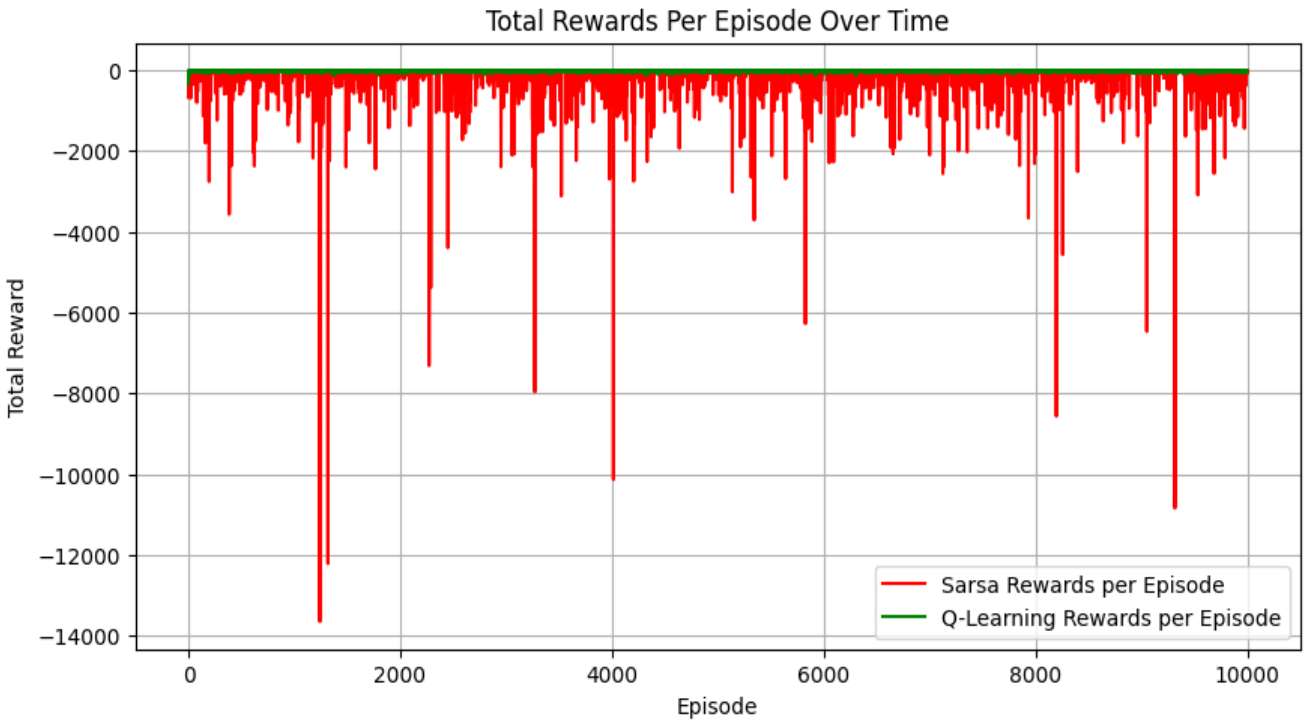


Figure 02: Q-Learning Trajectory



Discussion

The trajectory of the agent, as visualized in the 1st figure, demonstrates the path taken from the starting blue square to one of the terminal black squares. This trajectory highlights Sarsa's on-policy nature, characterized by a more exploratory approach. The agent occasionally took longer routes, reflecting the policy's iterative improvement as it balanced exploration and exploitation. The total rewards per episode, depicted in the subsequent plot, exhibited significant fluctuations initially. These fluctuations can be attributed to the agent frequently encountering high penalty red squares (-20 reward) during its exploratory phases. Over time, however, the sum of rewards improved as the policy became more refined and the agent learned to avoid the red squares more effectively. This gradual improvement, although with persistent variability, underscores Sarsa's inherent exploratory behavior and the consequent learning process that adapts the policy based on the actions taken during episodes.

The Q-learning algorithm produced a more direct and stable trajectory for the agent in the same grid world environment. The 2nd figure illustrates the Q-learning trajectory which indicates a more straightforward path to the terminal black squares. This outcome is expected given Q-learning's off-policy nature, which prioritizes the optimal future rewards regardless of the current policy's action. Consequently, the agent quickly learned to avoid the penalty states and converge on a more optimal path. The total rewards per episode for Q-learning, as shown in the relevant plot, displayed a markedly more stable trend compared to Sarsa. The rewards quickly converged to less negative values, demonstrating the algorithm's efficiency in learning the optimal policy.

The results from Sarsa and Q-learning offer insightful contrasts. While both algorithms successfully navigated the agent to the goal, their approaches and efficiencies differed. Sarsa's on-policy method led to more exploration, evident in the varied trajectories and fluctuating rewards. In contrast, Q-learning's off-policy strategy resulted in a more direct path and quicker convergence to stable rewards, showcasing its effectiveness in learning an optimal policy. These differences underscore the fundamental distinctions between on-policy and off-policy learning, with Sarsa excelling in exploratory environments and Q-learning demonstrating superior performance in environments where optimal policies can be learned more directly.

- **Are they different or similar? Why or why not? You may assume to use ϵ-greedy action selection for this task. How does the sum of rewards over an episode behaves for each of these two methods.**

The trajectories learned by Sarsa and Q-learning in the grid world problem exhibit both similarities and differences, primarily influenced by their on-policy and off-policy learning strategies, respectively. Both algorithms employed ϵ-greedy action selection to balance exploration and exploitation, resulting in eventual navigation to the terminal states while avoiding penalty states. However, Sarsa's on-policy nature led to more exploratory paths and greater initial variability in rewards, as it updates Q-values based on the actions taken by the current policy, which inherently promotes more exploration. In contrast, Q-learning's off-policy approach resulted in a more direct path to the goal and quicker convergence of rewards, as it updates Q-values using the maximum future rewards, thus focusing on the optimal policy with less exploratory deviation. The sum of rewards per episode for Sarsa fluctuated significantly in the beginning due to the agent's high exploration rate, but gradually improved as the agent learned to avoid high-penalty states. Despite this improvement, the variability persisted, reflecting the ongoing balance between exploration and exploitation. On the other hand, the sum of rewards for Q-learning displayed a more stable and less negative trend from the start, indicating a faster and more efficient learning process with stable rewards, thanks to its focus on maximizing future rewards and achieving quicker policy convergence.

PART 02

1.

```
Gradient Monte Carlo Value Function:
-0.00  0.01  0.05  0.12  0.24  0.52  0.00
-0.01  0.01  0.03  0.09  0.19  0.33  0.46
-0.04 -0.03  0.00  0.06  0.09  0.16  0.17
-0.10 -0.07 -0.02  0.00  0.02  0.07  0.07
-0.23 -0.15 -0.06 -0.04 -0.01  0.04  0.02
-0.45 -0.28 -0.15 -0.10 -0.06 -0.00 -0.01
 0.00 -0.46 -0.20 -0.10 -0.06 -0.03  0.00

Optimal Policy:
→ → → → → → x
→ → → → → ↑ ↑
↑ ↑ → → ↑ ↑ ↑
↑ → → ↑ ↑ ↑ ↑
↑ → ↑ ↑ → ↑ ↑
↑ → ↑ ↑ → ↑ ↑
x → → → → → ↓
```

2.

```
Semi-Gradient TD(0) Value Function:
 0.00  0.01  0.03  0.07  0.20  0.50  0.00
-0.01  0.00  0.02  0.06  0.12  0.26  0.47
-0.03 -0.02  0.00  0.03  0.08  0.15  0.20
-0.08 -0.06 -0.03  0.01  0.03  0.06  0.09
-0.21 -0.12 -0.07 -0.03 -0.00  0.02  0.04
-0.49 -0.26 -0.13 -0.06 -0.02  0.00  0.01
 0.00 -0.42 -0.17 -0.08 -0.03 -0.01  0.00

Optimal Policy:
→ → → → → → x
→ → → → → ↑ ↑
↑ → → → → ↑ ↑
↑ ↑ → → ↑ ↑ ↑
↑ ↑ → ↑ ↑ ↑ ↑
↑ ↑ → → → ↑ ↑
x → → → → → ↑
```

- **Exact Value Function**

```
Exact Value Function:
 0.00  0.01  0.03  0.08  0.20  0.46  0.00
-0.01  0.00  0.02  0.06  0.13  0.27  0.46
-0.03 -0.02  0.00  0.03  0.07  0.13  0.20
-0.08 -0.06 -0.03  0.00  0.03  0.06  0.08
-0.20 -0.13 -0.07 -0.03  0.00  0.02  0.03
-0.46 -0.27 -0.13 -0.06 -0.02  0.00  0.01
 0.00 -0.46 -0.20 -0.08 -0.03 -0.01  0.00

Optimal Policy:
→ → → → → → X
→ → → → → → ↑
↑ → → → → ↑ ↑
↑ ↑ → → ↑ ↑ ↑
↑ ↑ → → ↑ ↑ ↑
↑ → → → → ↑ ↑
X → → → → → ↑
```

Comments

The value function computed using the Gradient Monte Carlo method demonstrates some variance and discrepancies from the exact values. For example, the value at state (0, 5) is 0.52, compared to the exact value of 0.46, indicating a slight overestimation. Similarly, the value at state (4, 4) is 0.04, whereas the exact value is 0.02, showing a minor deviation. However, the Gradient Monte Carlo method still captures the overall trend of the value function reasonably well. For instance, the value at state (3, 3) is 0.00, which is consistent with the exact value function, indicating that central states are estimated with higher accuracy. The derived policy, while showing some inconsistencies near terminal states, reflects a reasonable approximation of the optimal actions.

On the other hand, the Semi-Gradient TD(0) method produces value estimates that are closer to the exact values, with reduced variance and higher accuracy. For instance, the value at state (0, 5) is 0.50, which is very close to the exact value of 0.46. Additionally, the value at state (4, 4) is 0.02, precisely matching the exact value, and the value at state (3, 3) is 0.01, also aligning closely with the exact value. The incremental updates using bootstrapped estimates in the TD(0) method result in lower variance, making it a more reliable method for approximating value functions. The policy derived from the TD(0) method is more consistent with the exact policy, providing reliable action choices near terminal states and across the grid. So, while the Semi-Gradient TD(0) method outperforms the Gradient Monte Carlo method in terms of value function accuracy and policy consistency.

References

[1]. R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction

----------END----------