Project-RL 3

Experimentation with simple bandit learning algorithms

Yeganeh Safari

Mahtab Mohammadi

INTRODUCTION

Reinforcement Learning (RL) is a key area in artificial intelligence that explores how agents can make decisions to maximize rewards in different environments. Two popular algorithms in RL are Sarsa and Q-learning, both of which help agents learn from their experiences. Sarsa is an on-policy algorithm, meaning it learns the value of the actions it is currently taking, while Q-learning is an off-policy algorithm, learning the value of the best possible actions regardless of the current policy. Both use strategies that balance exploring new actions and exploiting known ones to maximize rewards.

Another important method in RL is the Monte Carlo Method, which estimates the value of a state by averaging the results of multiple episodes that start from that state. It doesn't need a model of the environment and relies on complete episodes to calculate the value. Semi-Gradient TD(0) is another approach that combines aspects of Monte Carlo and dynamic programming. It updates the value of the current state based on the difference between its estimated value and the next state's value, using a linear approximation. In this report, we apply these methods to a grid world problem, where an agent must navigate a grid to reach a goal while avoiding penalties.

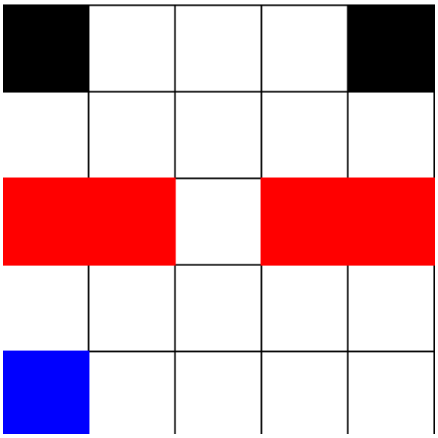
Mission

The main goal of this project is to use the Sarsa and Q-learning algorithms to help an agent find the best way to move from the start to the end of a grid while avoiding penalties. We will also look at the paths the agent takes based on the policies it learns and compare how well these two algorithms work by measuring the total rewards they collect over time. Since both methods use an epsilon-greedy strategy, we will explore how this approach impacts the agent's ability to try new actions and stick with what it already knows.

In this project, we want to show how Sarsa, which learns from the actions it is currently taking, and Q-learning, which learns the best possible actions even if it isn't following

them, deal with the challenge of balancing exploration and exploitation. We will also see how these methods help the agent learn the best actions in a changing environment.

Part 1: Consider a grid world problem where the agent starts at the blue square and moves to a neighbouring state with equal probability. If the agent moves to a red state, it receives a reward of −20 and goes back to the start, i.e., the blue square. A move between any two other states receives a reward of −1. A move that attempts to move outside of the grid receives a reward of −1. The black squares serve as a terminal state. Intuitively, you can see how the goal here is to pass through the opening in the red "wall" and get to one of the black squares and hence terminate the episode.



Use the Sarsa and Q-learning algorithms to learn the optimal policy for this task. Plot a trajectory of an agent utilizing the policy learned by each of the methods. 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.

Grid World Environment

We constructed a grid world environment where an agent must navigate from a starting position to one of the terminal states. The environment includes specific states that impose penalties, which the agent should avoid. The primary challenge for the agent is to find an optimal path that maximizes rewards while avoiding these penalties.

Learning Algorithms

We employed two reinforcement learning algorithms—Q-learning and Sarsa—to teach the agent how to navigate through the grid world. Both algorithms involve training the agent over multiple episodes, allowing it to explore the environment and learn from the rewards and penalties it encounters.

Policy Learning

Through the application of Q-learning and Sarsa, the agent developed policies that guided its actions within the grid. These policies were refined over time as the agent gathered more experience from the environment, ultimately allowing it to make more informed decisions.

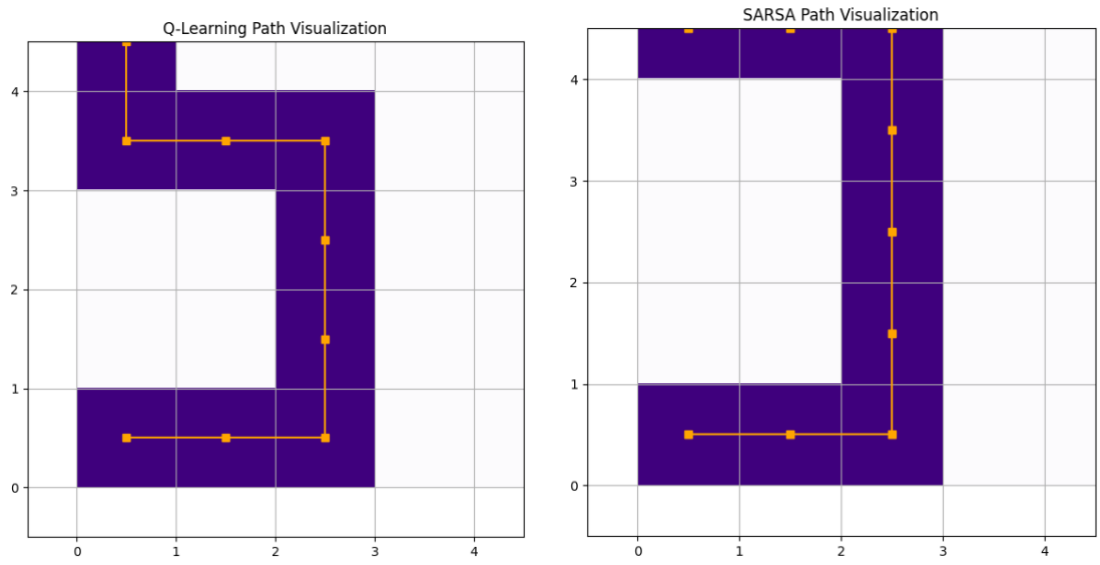
Performance Comparison

After training, we analyzed the performance of both algorithms by examining the total rewards accumulated over time. This comparison highlighted the differences between the on-policy approach of Sarsa and the off-policy nature of Q-learning.

Visualizing Results

The learned policies were then used to simulate the agent's movement within the grid. These simulations provided a visual representation of the paths taken by the agent under each policy, offering insights into the effectiveness of the learning algorithms in navigating the grid world environment.

The chart above shows how total rewards change over episodes for Sarsa and Q-learning. Sarsa has more and deeper dips, especially between episodes 300 and 400, because it updates based on the actions the agent actually takes, including random exploratory moves, which can lead to less optimal paths and more penalties. In contrast, Q-learning, which always updates towards the best possible future rewards, has a smoother learning curve with fewer drops, suggesting it finds a more stable and optimal policy. Despite these differences, both algorithms eventually learn effective strategies that help the agent reach the goal while avoiding penalty states.



The results above appear to have learned to avoid the red line (penalty states) and successfully navigates through the grid world, reaching the terminal state efficiently under the policies learned by both the Q-learning and Sarsa algorithms. The paths generated by both methods are similar, showing that both approaches can learn effective policies in this environment.

Conclusion

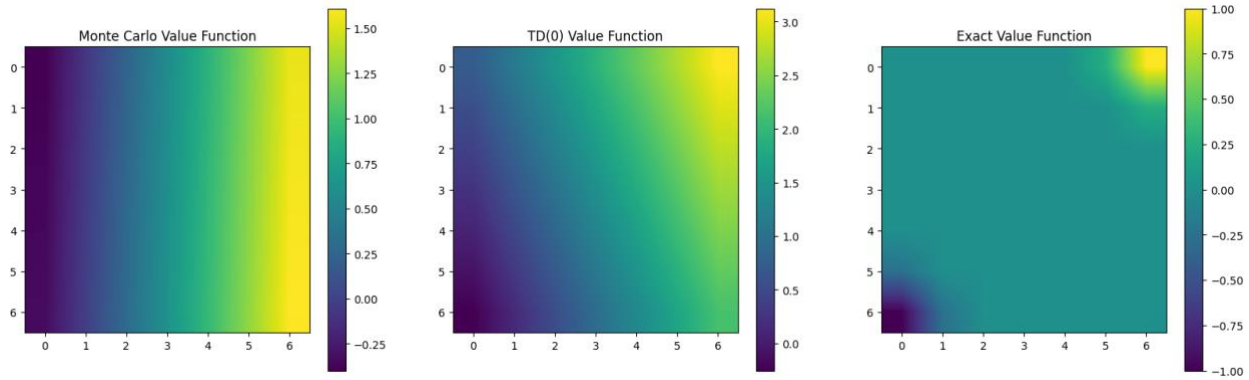
Both Q-learning and Sarsa effectively learned strategies for navigating the grid world while avoiding penalty states, though they did so in different ways. Sarsa, being on-policy, updated its actions based on actual moves made, leading to more conservative paths but with deeper dips in rewards, especially when exploration led to penalties. Q-learning, on the other hand, consistently aimed for the best possible future rewards, resulting in a smoother and more stable learning curve. Despite these differences, both algorithms ultimately guided the agent to successfully reach the terminal states while avoiding penalties, showing that each approach can develop effective policies in this environment.

Part 2: Consider a scenario where we have a random walk on a 7 × 7 grid. That is, we are equally likely to move up, down, left, or right. Suppose that we start the random walk at the precise center of the grid. We assume that the lower left and upper right corners are terminal states, with, respectively, rewards of −1 and 1. Rewards for transitions between two states are 0, if an attempt to transition outside the wall is made, the agent stays in the same spot and receives a reward of 0. Compute the value function for this "random walk" policy using (1) gradient Monte Carlo method and (2) the semi-gradient TD(0) method with an affine function approximation. How does it compare to the exact value function?

We set up a 7x7 grid where an agent starts in the middle and tries to reach one of two goal points: the lower-left or upper-right corners of the grid. The agent gets a reward of -1 for reaching the lower-left corner and +1 for reaching the upper-right corner. As it moves around, if the agent tries to go outside the grid, it stays in the same spot with no reward. The agent's movements are random, choosing to go up, down, left, or right with equal chances. This environment was designed to see how well the agent can learn to find the best path to a goal.

To help the agent learn, we used two different learning methods. The first method involved the agent exploring the grid, moving randomly until it reached a goal. After each complete journey, the agent updated what it had learned about the value of different states in the grid. The second method was more of a step-by-step approach, where the agent updated its understanding of the grid after every single move. By the end of training, both methods gave the agent a good idea of which paths in the grid are the most rewarding.

Once the training was done, we compared the results from the two learning methods with an exact calculation of the best possible values for each state in the grid. We created visual heatmaps to show how well each method did at figuring out the best paths. The heatmaps display the expected rewards from each point in the grid, and by looking at them, we could see how closely the agent's learning matched the exact best paths.



CONCLUSION

We explored how different reinforcement learning algorithms—Sarsa, Q-learning, Gradient Monte Carlo, and Semi-Gradient TD(0)—perform in various environments. The first experiment with Sarsa and Q-learning in a grid world showed that Q-learning was better at finding stable and direct paths, while Sarsa, which focuses on the current policy, sometimes chose safer but less efficient routes. In the second experiment with random walks, Gradient Monte Carlo provided a smooth overall estimate of values, whereas Semi-Gradient TD(0) responded more to immediate rewards, creating sharper value changes. Both methods closely matched the exact value function, but each had its own unique strengths. These experiments gave us valuable insights into how these algorithms behave in different scenarios, highlighting their individual benefits depending on the situation.