Report on SARSA & Q-Learning for Taxi-v3

1 Introduction

In this report, we compare the performance of two well-known reinforcement learning algorithms: SARSA (State-Action-Reward-State-Action) and Q-Learning. Both algorithms are tested on the Taxi-v3 environment, a classical OpenAI Gym problem, where an agent must navigate a grid to pick up and drop off passengers.

2 Methodology

2.1 Q-Learning

For Q-Learning, we implemented an epsilon-greedy policy. With a probability of $1 - \epsilon$, the agent selects the best action based on its current knowledge (Q-values). With probability ϵ, it explores randomly to gather more information.

The Q-values are updated using the following equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{1}$$

2.2 Q-Learning with Scheduling

In the scheduled version of Q-Learning, we decrease the exploration rate ϵ over time to favor exploitation as the agent learns more about the environment. Initially, ϵ is high to encourage exploration, but it decreases gradually to converge towards a stable policy.

2.3 SARSA

SARSA updates the Q-values based on the actual action taken by the agent, in contrast to Q-Learning, which updates based on the optimal future action, regardless of whether it was taken or not. The SARSA update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a) \right) \tag{2}$$

3 Results

We ran the algorithms for 6000 episodes to compare their performance. The rewards were tracked over time, and we evaluated the average reward over the last 100 episodes to monitor convergence.

Q-Learning initially took some time to converge, exhibiting instability during the early stages of training, especially when the agent explored risky actions. However, after optimizing the hyperparameters, particularly through the use of epsilon scheduling, the results improved significantly. The Q-Learning with exploration experienced a dip in performance before becoming unstable but ultimately converged correctly again, demonstrating a more robust learning process.

SARSA converged quickly and directly toward the solution, demonstrating consistent learning throughout the episodes. It rarely took extreme actions, which minimized the negative rewards early in training. In contrast to Q-Learning, SARSA's policy evolved steadily, with fewer spikes in performance, and did not require the optimization of hyperparameters to achieve these results.

4 Improvements and Challenges

To improve the performance of both algorithms, we tried resetting the agent to a previous state if it showed signs of "forgetting" what it had learned (i.e., when rewards dropped significantly after good performance). This method helped stabilize learning but introduced the challenge of selecting an appropriate fallback policy. Ultimately, while both models struggled initially, they converged to policies that yielded positive rewards over time. SARSA, in particular, exhibited more reliable learning, while Q-Learning required more fine-tuning of hyperparameters (like epsilon scheduling) to avoid instability.

5 Conclusion

In conclusion, through our exploration of Q-Learning and SARSA in the Taxi-v3 environment, we ultimately managed to develop an agent that effectively learned to navigate its surroundings and achieve its objectives. Although the project was challenging, particularly at the beginning when we observed fluctuations in rewards that left us uncertain despite correct implementation, we persevered. Over time, both algorithms adapted, leading to an agent that consistently performed well. Graphs and videos illustrating our results are available in the repository.