Lab 6 Report – Reinforcement Learning

Martyna Wielgopolan and Aditya Kandula
Group 9
Variant 4: Taxi-v3

Summer 2025

1. Introduction

Reinforcement Learning (RL) is a machine learning approach where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. In this lab, we implement the Q-Learning algorithm—a model-free RL method—on the Taxi-v3 environment using the Gymnasium library.

The Taxi-v3 task involves navigating a 5x5 grid to pick up and drop off passengers at specified locations, with rewards given for successful deliveries and penalties for invalid actions or delays. The environment has a discrete state and action space, making it suitable for tabular Q-Learning.

The goal is to train the agent to learn an optimal policy by updating a Q-table through trial and error. We monitor the agent's performance across episodes, analyze the effect of different hyperparameters, and evaluate the learned policy based on cumulative rewards and convergence behavior.

2. Q-Learning Overview

Q-Learning is a widely used model-free reinforcement learning algorithm that enables an agent to learn the value of actions in given states through direct interaction with the environment. It is based on the principle of estimating the optimal action-value function $Q(s,a)$, which represents the expected cumulative reward the agent can obtain by taking action and following the optimal policy thereafter.

Unlike supervised learning, Q-Learning does not require labeled data. Instead, the agent explores the environment, takes actions, receives feedback in the form of rewards, and updates its estimates of future rewards using the Q-value update rule:

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

where:

- α is the learning rate that controls how quickly the Q-values are updated,
- γ is the discount factor that determines the importance of future rewards,
- r is the immediate reward received after taking action a in state s,

- s' is the next state after taking action a.

To balance exploration and exploitation, Q-Learning typically uses an ϵ-greedy strategy: with probability ϵ, the agent explores by choosing a random action; with probability $1 - \epsilon$, it exploits the best-known action based on current Q-values.

As training progresses and the Q-table is updated across many episodes, the agent gradually converges to an optimal policy that maximizes long-term rewards. Q-Learning is particularly well-suited to problems with discrete state and action spaces, such as the `Taxi-v3` environment used in this lab.

3. Implementation Details

- **Environment:** `Taxi-v3`
- **Libraries used:** `gymnasium`, `numpy`, `matplotlib`
- **Q-table:** Initialized as a zero matrix of shape (500, 6)
- **Hyperparameters:**
  - Learning rate (α): 0.1
  - Discount factor (γ): 0.99
  - Exploration rate (ϵ): starts at 1.0, decays to 0.1
  - Episodes: 5000

4. Results and Training Metrics

To evaluate the effect of Q-Learning hyperparameters on training performance, multiple experiments were conducted using different values of the learning rate (α), discount factor (γ), and minimum exploration rate ($\epsilon_{\min}$). The results are visualized in the reward curves below.

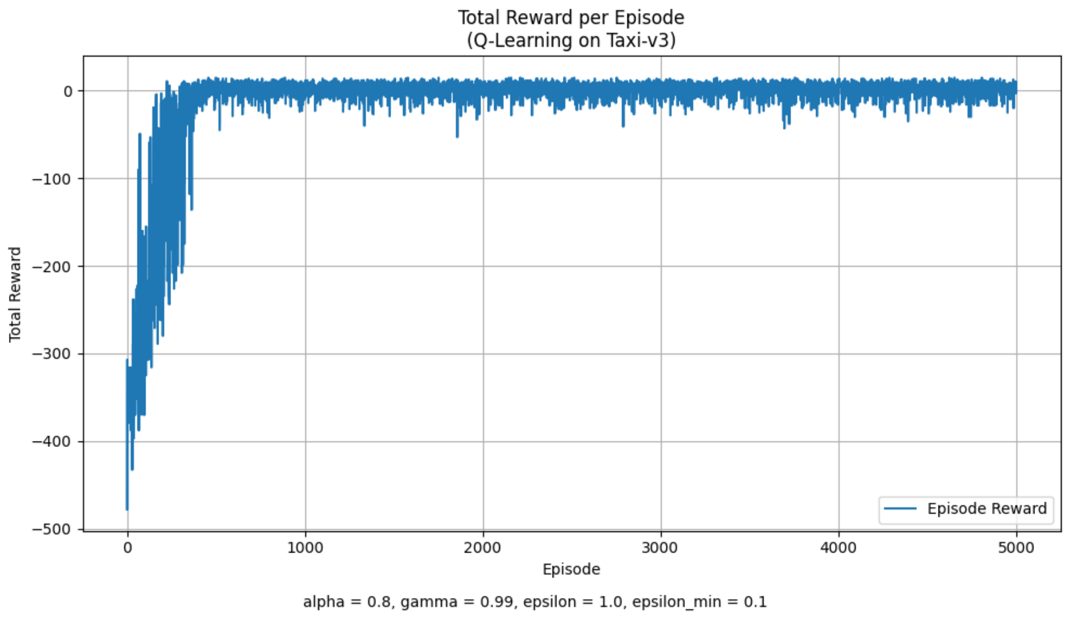

Figure 1: Episode rewards for $\alpha = 0.8$, $\gamma = 0.99$, $\epsilon_{\text{min}} = 0.1$

Analysis: A high learning rate caused fast convergence but introduced more variance. The agent learned quickly, but rewards fluctuated due to aggressive updates.

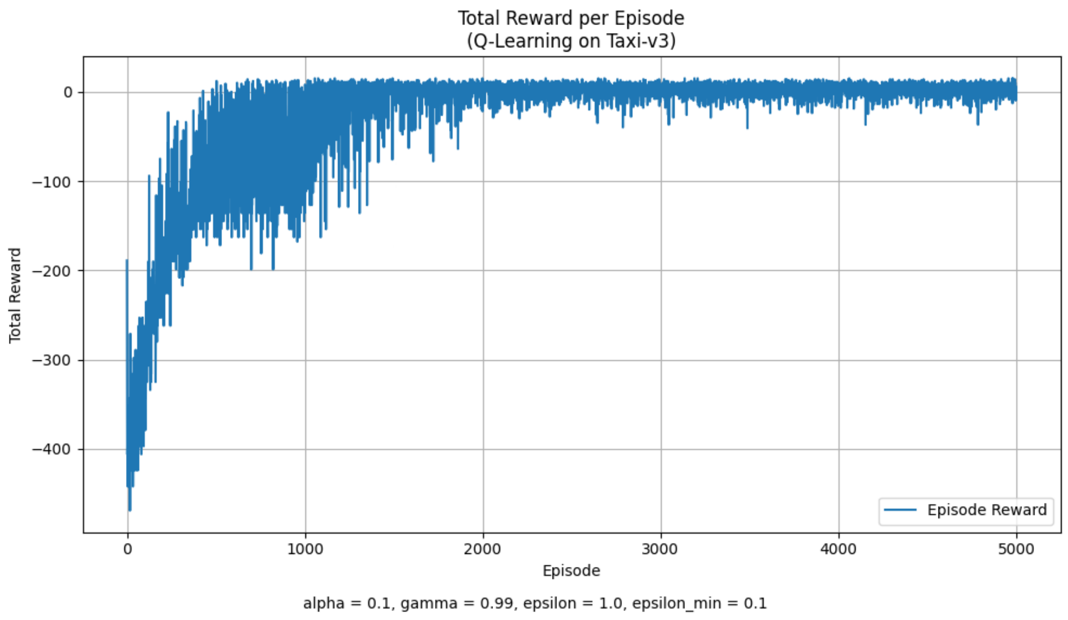

Figure 2: Episode rewards for $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon_{\text{min}} = 0.1$

Analysis: This standard setting provided stable and gradual learning. Performance improved steadily and converged to consistent positive rewards after around 2000 episodes.

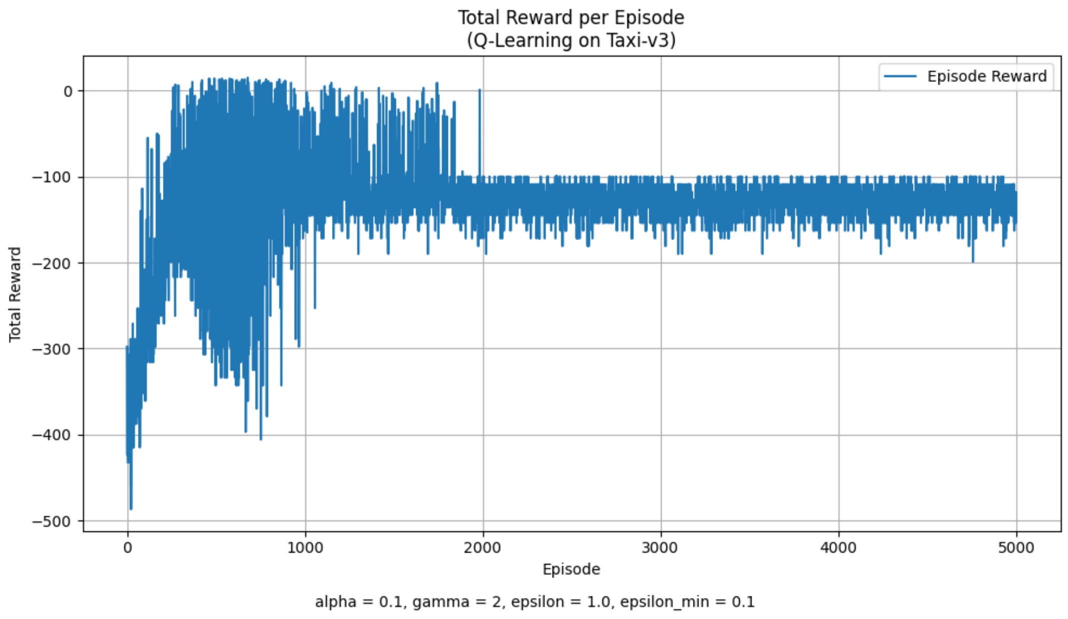

Figure 3: Episode rewards for $\alpha = 0.1$, $\gamma = 2.0$, $\epsilon_{\text{min}} = 0.1$

Analysis: Using $\gamma > 1$ destabilized the learning. The agent's performance degraded after initial progress, indicating divergence and an inability to converge on a stable policy.

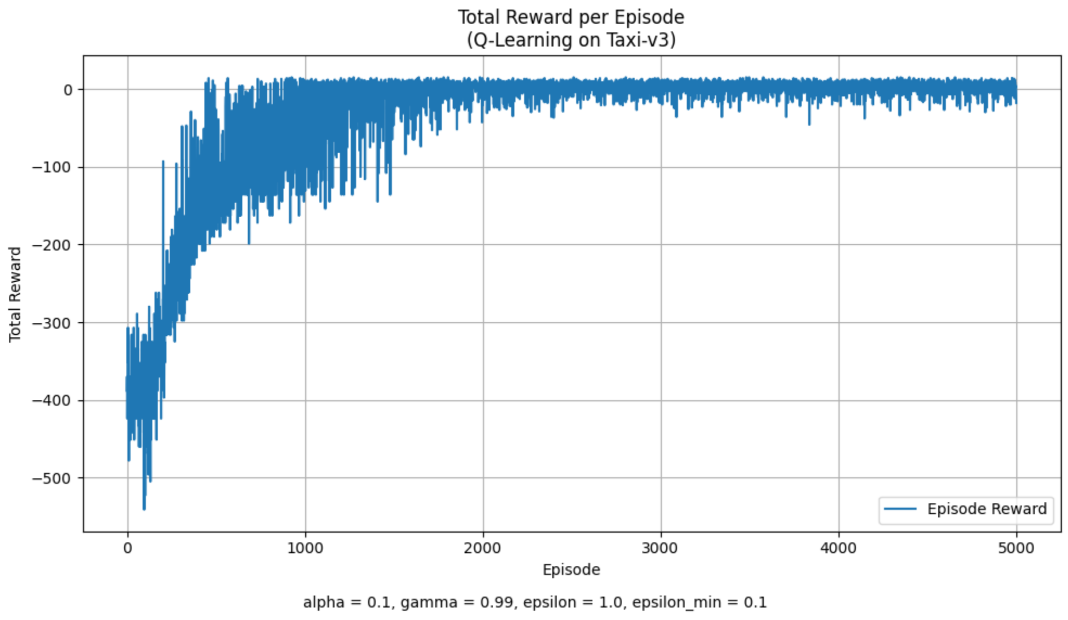

Figure 4: Episode rewards for $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon_{\text{min}} = 0.1$ (repeat)

Analysis: This run confirmed the stability of the baseline configuration. The agent again learned effectively with low variance and consistent reward growth.

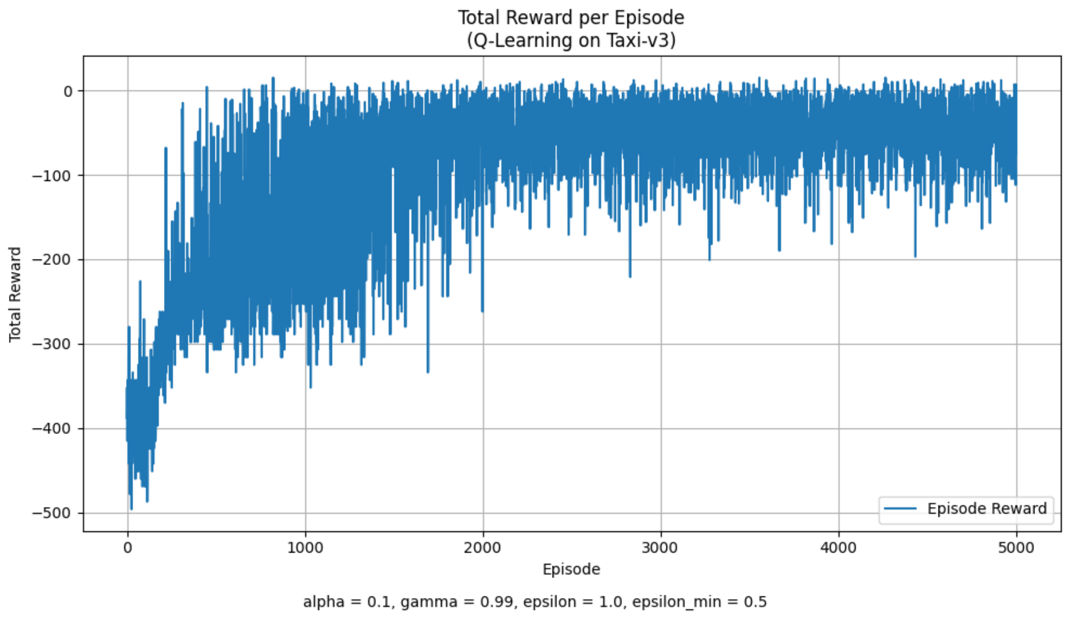

Figure 5: Episode rewards for $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon_{\text{min}} = 0.5$

Analysis: A higher minimum epsilon led to too much exploration even in later episodes. The agent failed to fully exploit its learned policy, resulting in noisy and suboptimal performance.

5. Observations

- The agent initially explored random actions and performed poorly.
- As ϵ decayed, the agent relied more on learned values, and performance improved.
- The average number of steps per successful episode reduced significantly after around 2000 episodes.

6. Experimentation

Additional experiments were conducted by adjusting the following:

- **Higher learning rate ($\alpha = 0.5$):** Converged faster, but less stable.
- **Lower discount factor ($\gamma = 0.8$):** Focused on immediate rewards, leading to suboptimal policies.

7. Conclusion

Q-Learning is an effective method for solving discrete reinforcement learning problems like Taxi-v3. Proper tuning of hyperparameters significantly affects convergence and performance.