دانشکده مهندسی برق

گروه کنترل

HW4 Exercise - Taxi V3

Reinforcement Learning

(Special Topics in Control)

Armin Attarzadeh

Iran University of Science and Technology

The codes and results are all presented in the notebook file.

Resources used:

- https://www.gymlibrary.dev/environments/toy_text/taxi/
- https://github.com/TheAthleticCoder/RL-on-OpenAI-Gym/tree/main
- https://github.com/VirajVaitha123/Q-learning-to-solve-OpenAI-Gyms-Taxi-v2/tree/main

According to the documents available on the gym site, the Taxi-V3 environment has 6 actions and 500 different modes.

It is supposed to achieve the optimal policy in the form of model-free in different ways . So that in the initial state, the taxi will move towards the passenger and take him to his destination in the shortest time.

There are two general approaches to solving this problem:

- Learning through Monte Carlo estimation
    - Off-Policy
    - On-Policy
- Learning through the Temporal Differential Method or TD
    - Q-Learning
    - SARSA

It was observed that in general, MC methods are very slow because they require the information of the entire episode to be updated, but TD methods perform the update at each step and converge faster to optimal behavior.

Part A : Solving with the Monte Carlo Method

✓ A1 – Monte Carlo –On-Policy

$$\epsilon = 0.2 \quad \gamma = 1$$

Due to the requirement for the agent to complete each episode, the speed of this algorithm is quite slow. After running 1,000 episodes, the Monte Carlo method did not achieve the expected convergence, and the performance deteriorated.

The average reward in the final episodes was -200

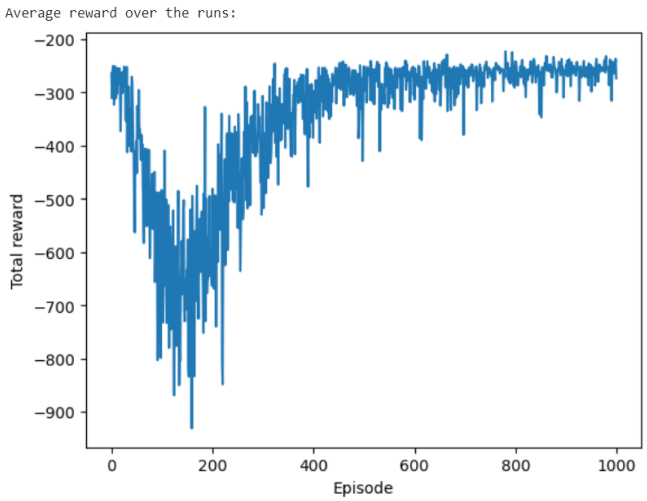

*Figure -**Error! No text of specified style in document.**-1 training plot for on-policy MC*

Even after increasing the number of episodes tenfold, the algorithm still failed to achieve positive rewards

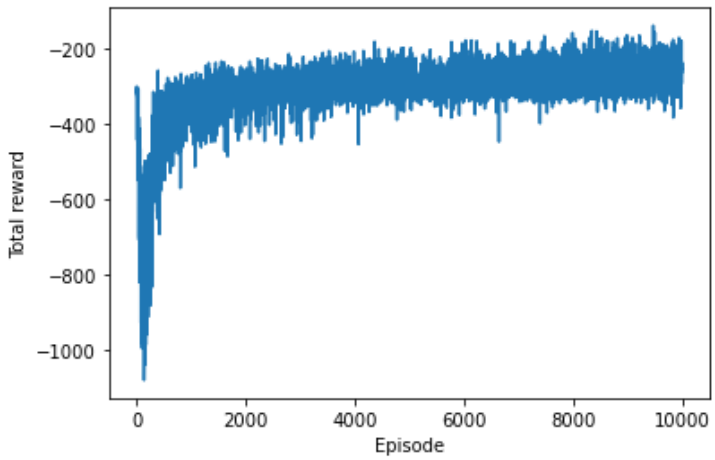

*Figure -**Error! No text of specified style in document.**-2 training plot for MC - 10K episodes*

Analysis: This may be due to the large state space, as data collection using a soft epsilon strategy did not sufficiently explore all states for updates. The performance indicated that the agent only performed well in certain states, while it struggled in most others. Thus, additional exploration is required. Although increasing the exploration rate by 0.5 addressed some exploration issues, variance remains a concern. The performance results will be detailed in Part C.

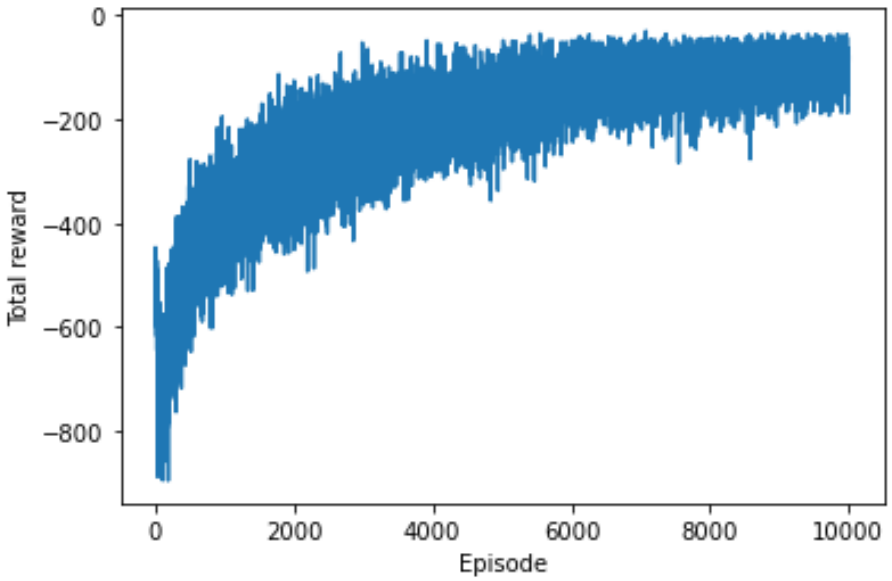

*Figure-**Error! No text of specified style in document.**-3 training plot for MC with more exploration*

- ✓ **A2- Monte Carlo Off-Policy**

$$\epsilon = 0.1 \quad \gamma = 1$$

In this algorithm, the behavior policy differs from the target policy. There are two methods for estimating the importance of sampling. During the execution phase, you can choose between ordinary or weighted commands and different modes.

Due to the lengthy nature of the learning algorithm in the Monte Carlo method, we only completed up to 1,000 episodes in Part A2, yielding the following results:

- By the Ordinary Method

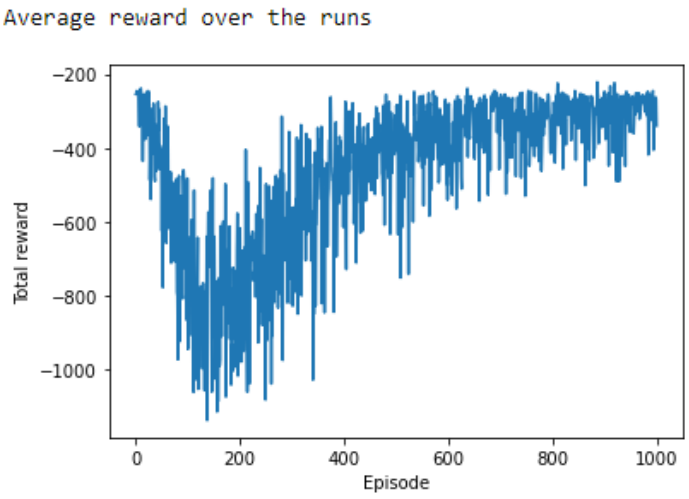

*Figure-**Error! No text of specified style in document.**-4 training plot for MC with ordinary sampling*

- Weighted Method

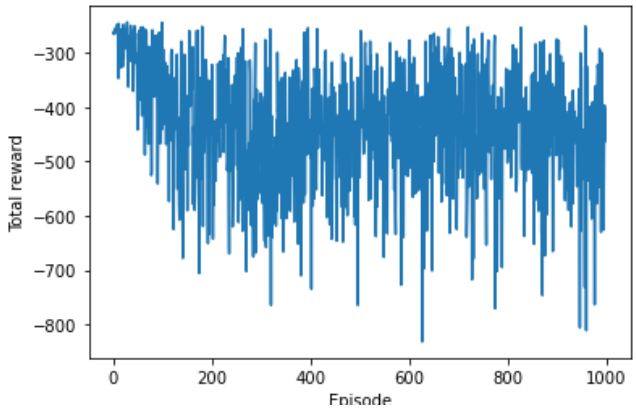

*Figure-**Error! No text of specified style in document.**-5 training plot for MC with weighted sampling*

The conditions of convergence and reward variance were not suitable in this section and it is better to use TD methods

Part B - TD Methods

The implemented methods remain model-free, but updates occur at every step instead of waiting for the end of the episode. This approach allows for significantly faster convergence.

✓ **B1- Problem Solving with Q-Learning**

$$\epsilon = 0.1 \quad , \gamma = 1 \, , \alpha = 0.5$$

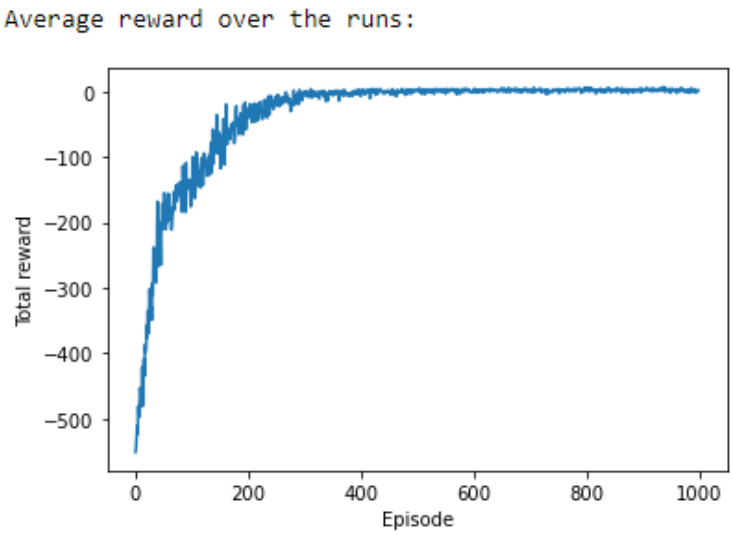

*Figure-**Error! No text of specified style in document.**-6 training plot for Q-learning*

The average rewards in the final episodes are often positive and greater than zero. In other words, in most cases, the episode was executed correctly.

✓ **B2- Problem Solving with SARSA**

When comparing SARSA with Q-learning, the speed of both methods was nearly identical. However, SARSA exhibited higher variance in rewards compared to Q-learning.

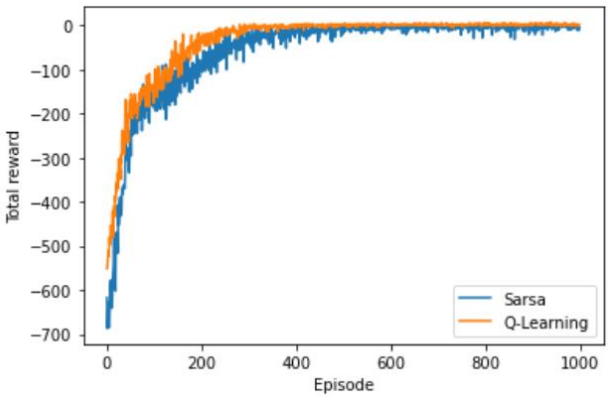

Figure -7 SARSA vs Q-learning

Part C- Implementation of Trained Policies and Conclusions

Videos related to each run are posted in the folder.

Evaluation: In ten different execution of the obtained policies, the TD methods exhibited lower variance in rewards. However, the Monte Carlo method occasionally failed to update certain modes effectively, leading to suboptimal performance in some cases. Additionally, the training phase with TD methods was significantly faster.

When comparing Q-learning and SARSA, the results indicate that the Q-learning algorithm had less variance in rewards. Its error percentage requirement was also lower than that of SARSA in the implementations.