Second Mini-Project, Monte Carlo and Temporal Difference control

Arya Ebrahimi
Reinforcement Learning Spring2023

Abstract **– In this mini-project, we investigate MC and TD methods to compute optimal policies and compare their performance in different problem settings. First, the MC first visit on-policy control algorithm is implemented and tested using the modified OpenAi gym TaxiEnv environment, and in the next part, Q-Learning and Sarsa algorithms are implemented and tested on the same environment.**

Index Terms – Monte Carlo control, Temporal difference, Q-Learning, Sarsa

INTRODUCTION

In the preceding mini-project, we gained an understanding of DP methods that utilized a comprehensive model of the world along with bootstrapping to enhance policies. Nevertheless, this model may not be accessible in numerous real-world scenarios. To address this issue, we employ sample-based methods as an alternative instead of relying on the complete model of the world.

Monte Carlo methods are a class of algorithms that rely on random sampling to estimate numerical results. The core idea behind Monte Carlo methods is to approximate unknown values by generating a large number of random samples and averaging their outcomes. These methods are particularly useful in situations where an analytical solution is either infeasible or difficult to derive. In the context of reinforcement learning, Monte Carlo methods are employed to estimate the value of a state or action by averaging the returns observed from multiple episodes or episodes.

On the other hand, Temporal Difference methods combine elements of Monte Carlo methods with bootstrapping techniques to estimate the values of states or actions. These methods aim to learn from experience by updating value estimates incrementally as new information becomes available. Unlike Monte Carlo methods, Temporal Difference methods do not require complete episodes to be simulated. Instead, they update value estimates based on the immediate reward received and the estimated value of the next state or action. This bootstrapping approach allows for more efficient learning, as the agent can start updating its estimates before reaching a terminal state.

Both Monte Carlo and Temporal Difference methods have their advantages and trade-offs. Monte Carlo methods provide unbiased estimates of state or action values but require complete episodes to be sampled, which can be computationally expensive. Temporal Difference methods, on the other hand, can update value estimates after each time step, making them more suitable for online learning tasks. However, they may introduce bias in the estimates due to the bootstrapping process.

ON-POLICY FIRST-VISIT MC CONTROL

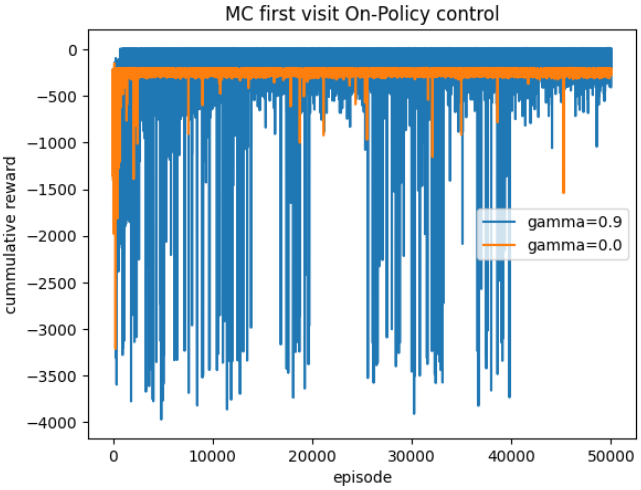


FIGURE 1: ON-POLICY FIRST-VISIT MC CONTROL USING TWO GAMMA VALUES.

It is evident that due to the high variance in Monte Carlo methods, a substantial number of episodes are required to achieve a desirable policy. In the case of the presented taxi environment, even after utilizing 50,000 episodes, satisfactory values have not been attained.

TEMPORAL DIFFERENCE CONTROL METHODS

In this section, we will examine two algorithms, Sarsa and Q-Learning, which are temporal difference control methods. Q-learning is an off-policy control version of TD(0), which uses the Bellman optimality equation. In contrast, Sarsa is

on-policy and operates according to the policy it is trying to learn.

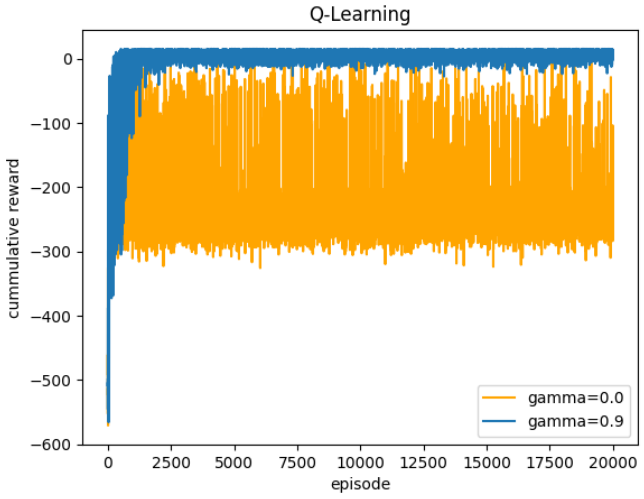


FIGURE 2: Q-LEARNING USING TWO GAMMA VALUES.

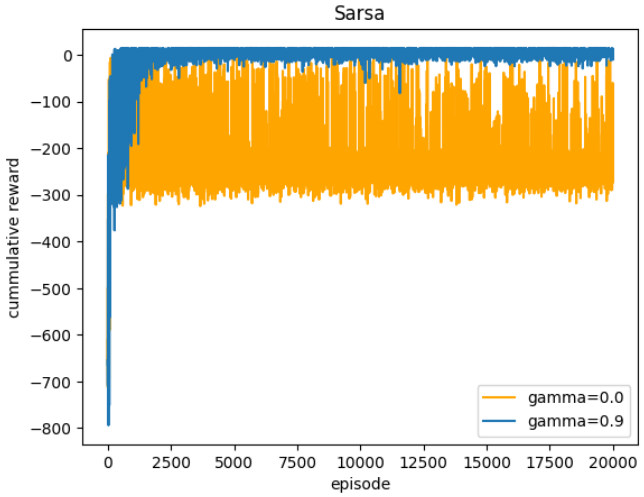


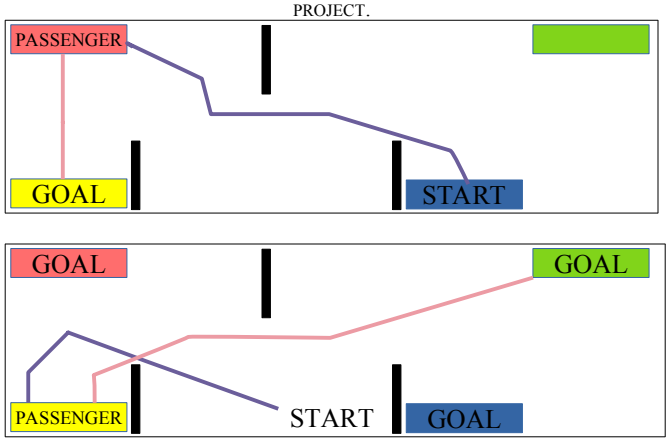
FIGURE 3: SARSA CUMULATIVE REWARDS DURING EPISODES USING TWO GAMMA VALUES.

As can be seen, in both algorithms, the version which $\gamma = 0$ does not converge well, Because it only considers the immediate rewards and neglects the discounted rewards towards terminal states. However, when gamma is higher, Sarsa and Q-learning both converge to the optimal action values, but Q-learning converges a little faster.

Q-learning is an off-policy reinforcement learning algorithm, meaning it learns the optimal action-value function by selecting the maximum-valued action at each state, regardless of the policy being followed. This allows Q-Learning to potentially explore and update its Q-values more efficiently, leading to faster convergence.

On the other hand, Sarsa is an on-policy algorithm that updates its action-value estimates based on the actions actually taken according to the current policy. Sarsa takes into account the exploration and exploitation trade-off in a more conservative manner by following a specific policy during the learning process. This cautious exploration can slow down the learning process and may require more iterations to converge to an optimal policy.

TABLE 1: THESE CHARTS ILLUSTRATE THE OPTIMAL POLICIES THE Q-LEARNING AGENT CALCULATES IN THE SPECIFIED CASES. THE PURPLE LINE SHOWS THE AGENT'S PATH UNTIL REACHING AND PICKING UP THE PASSENGER. THE PINK LINE SHOWS THE TAXI'S PATH TO THE GOAL LOCATION. THESE TESTS CAN BE SEEN IN THE RUNS FOLDER OF THE MINI-PROJECT.



As seen in the above charts, the Q-Learning agent uses the modified actions (southwest, southeast, northwest, and northeast) to utilize its optimal policy.

WRAPPER

As previously discussed, we have made modifications to the Gymnasium Taxi environment. Instead of the original 6 actions, we have implemented a new version with 10 actions. To achieve this, we created a wrapper class that alters the step function of the environment. Additionally, we changed the action space of the environment to Discrete(10).

Within the step function, there is a conditional statement that handles the different actions. If the action chosen falls within the range of 0 to 5 (the original actions), it behaves in the same way as before. However, if the action is between 6 and 9, it involves combining two of the previous actions. For instance, if the action 6 is selected, it combines the south and east actions to create a new action called "south east.