COMP 138 RL: Programming Assignment 2

Hannah Doherty

October 22, 2022

1 Exercise 5.12: the Problem Text

Consider driving a race car around a turn like those shown in Figure 1 (Figure 5.5 in the textbook). Apply a Monte Carlo control method to this task to compute the optimal policy from each starting state. Exhibit several trajectories following the optimal policy (but turn the noise off for these trajectories). [1]

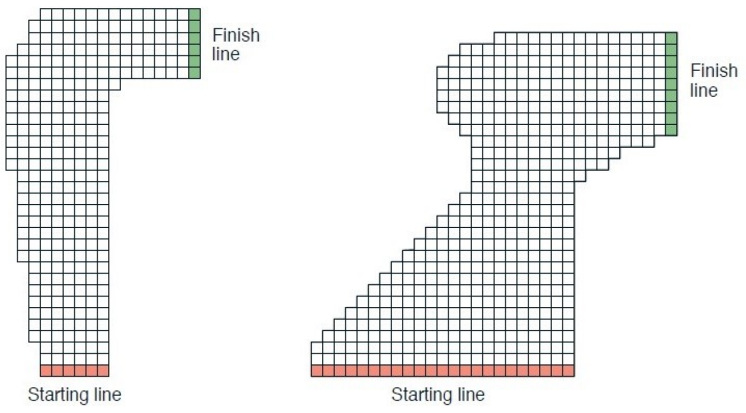

Figure 1: A couple of right turns for the racetrack task.

2 Background

In the race car problem, you want to go as fast as possible, but not so fast as to run off the track. In our simplified racetrack, the car is at one of a discrete set of grid positions, the cells in the diagram. The velocity is also discrete, a number of grid cells moved horizontally and vertically per time step. The actions are increments to the velocity components. Each may be changed by +1, -1, or 0 in each step, for a total of nine (3 X 3) actions. Both velocity components are restricted to be non-negative and less than 5, and they cannot both be zero except at the starting line. Each episode begins in one of the randomly selected start states with both velocity components zero and ends when the car crosses the finish line. The rewards are -1 for each step until the car crosses the finish line. If the car hits the track boundary, it is moved back to a random position on the starting line, both velocity components are reduced to zero, and the episode continues. Before updating the car's location at each time step, check to see if the projected path of the car intersects the track boundary. If it intersects the finish line, the episode ends; if it intersects anywhere else, the car is considered to have hit the track boundary and is sent back to the starting line. To make the task more challenging, with probability 0.1 at each time step the velocity increments are both zero, independently of the intended increments. [1]

3 Problem Statement

I will show how a race car can drive around those tracks in the Figure by using a Monte Carlo control method. Using this method, I will compute the optimal policy for multiple trajectories.

4 Problem Solution

4.1 Analysis

To solve the race car problem, I will generate an environment for the track, an agent to choose actions to take on the track (and the state it's currently at), a Monte Carlo Off-Policy Control algorithm using the pseudo-code in the textbook, and a visualizer for the track.
To generate the racetrack I created an array where the numerical value in the array represents the locations of the track, starting line, and finish line. Building the agent required using a policy to do actions in the state and the track. The visualizer used plt.imshow() to create an image that specifies the locations and trajectory of an agent moving through the track.
The difference between on-policy methods and off-policy methods is that on-policy estimate the value of the policy while using it for control. In off-policy methods, they separate these two functions. The behavior policy is used to generate behavior and can be unrelated to the target policy which is used to evaluated and improved. What's great about the separation in function with the off-policy methods is that the target policy can be deterministic (for example greedy), while the behavior policy can keep on trying out all possible actions. To summarize, these type of methods follow the behavior policy while also learning and improving the target policy.
Using this algorithm we generate episodes using ϵ-greedy policy (a behavior policy) with probability ϵ. This ensures the exploration of previously unexplored state-action values. The type of policy that this MC Off-Policy Control algorithm will follow in this assignment is deterministic (greedy) with a soft behavior policy being the epsilon-greedy policy with epsilon set to 0.1.
In Figure 2 (see below), the rosy pink dots represent the agent moving through

the track in a single episode and represents the trajectories of the optimal policy. The track is black, the starting line is green, and the finish line is red. When you look at these trajectories you can see the agent's starting point adjust sometimes as it learns how to get the highest reward possible in each episode.

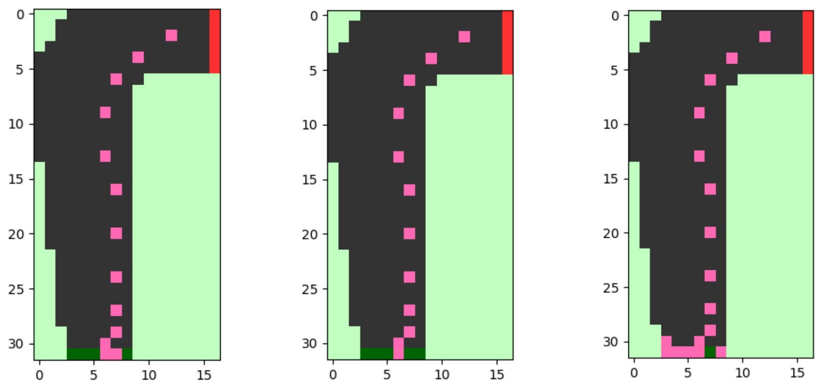

Figure 2: Several trajectories of the optimal policy for the racetrack problem.

In Figure 3 (see below), we see similar figures as shown in Figure 2 but this time it's on track 2. These figures represent the trajectories of the optimal policy again.

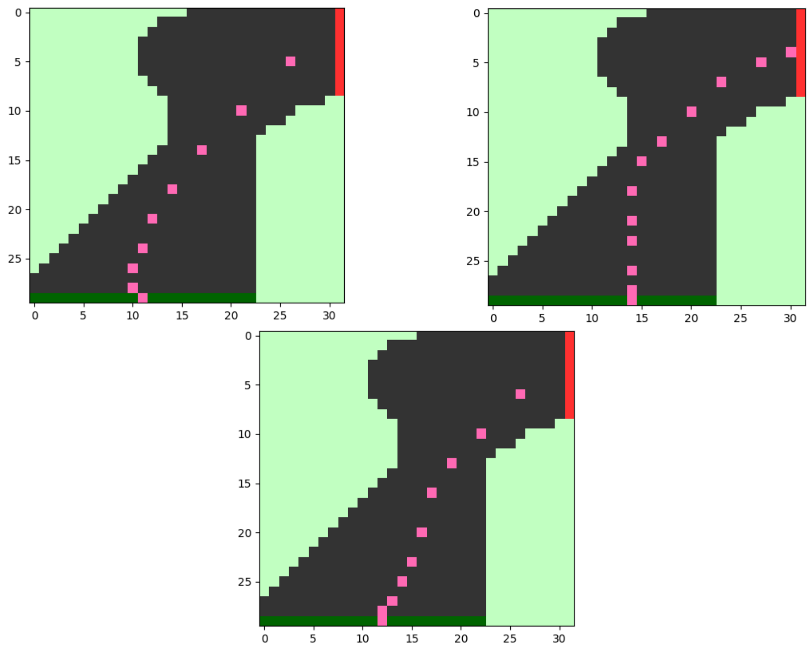

Figure 3: Several trajectories of the optimal policy for the racetrack problem.

5 Conclusion

In conclusion, we can see that the number of episodes is directly proportional to the reward given until it reaches an optimal reward amount and plateaus. In Figure 4 (shown below) we can see that there is a learning curve for the Monte Carlo Off-Policy Control Method. After several thousand episodes the rewards start to reach their optimal reward of around -40.
We have solved the racetrack problem with the Monte Carlo Off-Policy Control method. In the images/figures we see that the agent learns the optimal policy for each starting state. The plots below show the rewards for each episode. Track 2 seems to converge at an optimal solution faster than track 1. My theory is that that is because the ratio of acceptable spaces versus unacceptable spaces that send the agent back to the start is higher for track 2 than track 1. This means that it's easier for an agent to fail in track 1 than track 2.

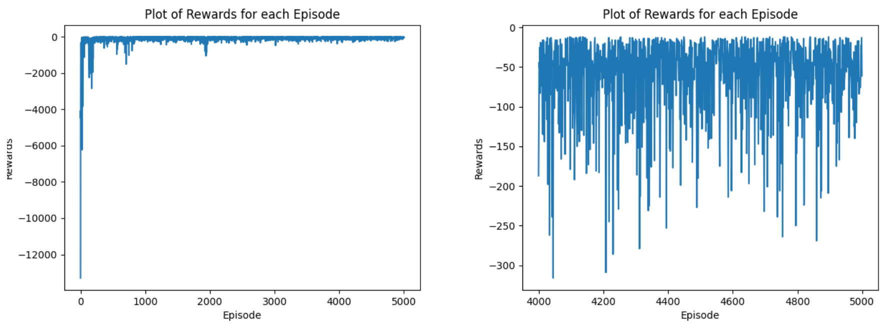

Figure 4: Track 1: Plot of Rewards per Episode

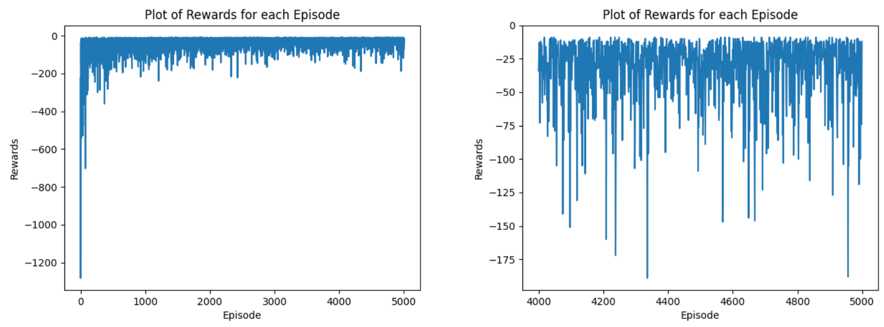

Figure 5: Track 2: Plot of Rewards per Episode

6 Extra Credit

6.1 Exercise 4.8

Problem Statement: Why does the optimal policy for the gambler's problem have such a curious form? In particular, for capital of 50 it bets it all on one flip, but for capital of 51 it does not. Why is this a good policy?
Solution: The reason that the optimal policy for the gambler's problem bets all the money at 50 is because that price has the highest probability to win. If the value function represents the probability then it is at it's highest probability of winning when it's at 50.

6.2 Exercise 5.2

Problem Statement: Suppose every-visit MC was used instead of first-visit MC on the blackjack task. Would you expect the results to be very different? Why or why not?
Solution: I would expect the result to be not be very different. Because blackjack doesn't have two of the same state in an episode, so both methods are basically the same. It also doesn't re-visit old states in an episode once it's visited. Using every-visit MC wouldn't have an effect on the value function.

References

[1] A. Sutton, R. Barto. *Reinforcement Learning.* The MIT Press, 2020.