COMP 138 RL: Homework 2

Ramtin Hosseini

October 13, 2020

1 Introduction

Monte Carlo method is a way of solving the reinforcement learning problems based on averaging sample returns. Advantages of Monte Carlo methods over dynamic programming methods are:

1. They can be used to learn optimal behavior directly from interaction with the environment, with no model of the environment's dynamics.
2. They can be used with simulation or sample models.
3. It is easy to focus Monte Carlo methods on a small subset of the states.
4. They may be less harmed by violations of the Markov property. This is because they do not update their value estimates on the basis of the value estimates of successor states.

In the programming assignment 5.12 of the book [1], we would like to leverage Monte Carlo method for driving a race car. We want to go as fast as possible, but not so fast as to run off the track. In the provided racetracks, 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 nonnegative 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.

2 Learning Methodology

In this section, we explore the off-policy Monte Carlo method. The algorithm has been provided in figure 1.
The distinguishing feature of on-policy methods is that they estimate the value of the policy while using it for control. In off-policy methods, these two functions are separated. The policy used to generate behavior, called the behavior policy, may, in fact, be unrelated to the policy that is evaluated and improved, called the target policy. Although in principle, the behavior policy can be unrelated to the target policy, but in practice, it is kept quite representative of the target policy.
The episodes are generated by using behavioral policy action which is an ϵ-greedy policy which with probability ϵ, it chooses an action uniform randomly and with probability 1- ϵ, it chooses the greedy action. This ensures the exploration of previously unexplored state-action values.
To update the Q-values, importance sampling has been deployed.

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Figure 1: Off-policy MC prediction

3 Experimental Results

I did my experiments for two maps. In addition, I considered 2 values for ϵ. One is 0.001 and the other one is 0.1. In the following subsections, we investigate all settings.

3.1 Map 1

The first map has been provided in figure 2.

Figure 2: Map 1

3.1.1 Case Study 1: Less Exploration; $\epsilon = 0.001$

The first scenario is that using less exploration. Thus I consider a small value for ϵ to increase the chance for exploitation. Figure 3 shows the average rewards with respect to episode number. After almost 50000 episodes, the algorithm converges. In this case, The reward function converges around -40.
There are 6 possible starting points. After training the model, I ran the experiments for each of them which are shown in figures 4 and 5. These two figures show how the model approaches from beginning to the end line. The obtained rewards for each of them are $-169, -14, -64, -10, -12, -164$ respectively.

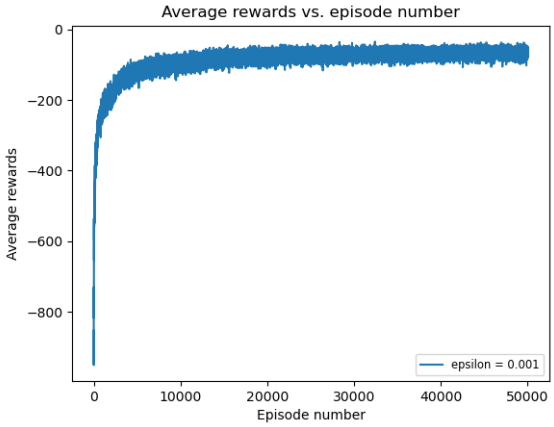

Figure 3: Average rewards vs. episode number

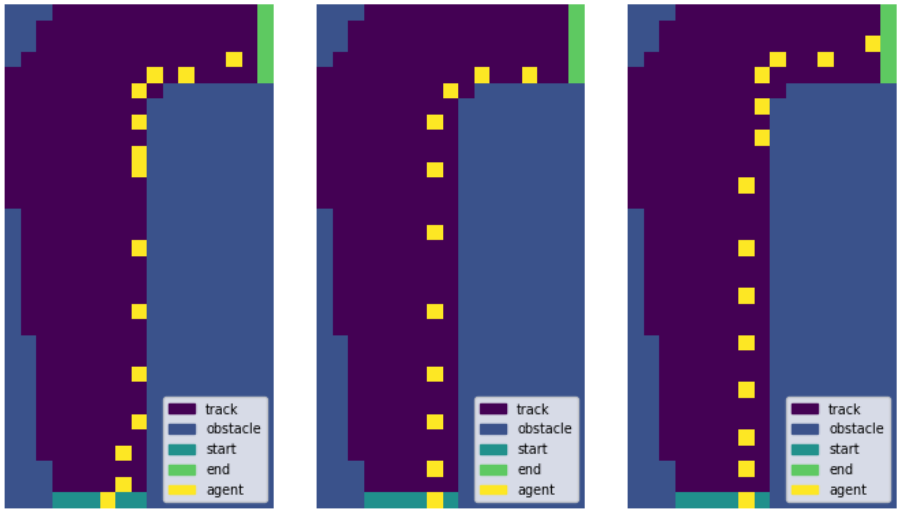

Figure 4: Approach of trained agent to finish line from different starting points

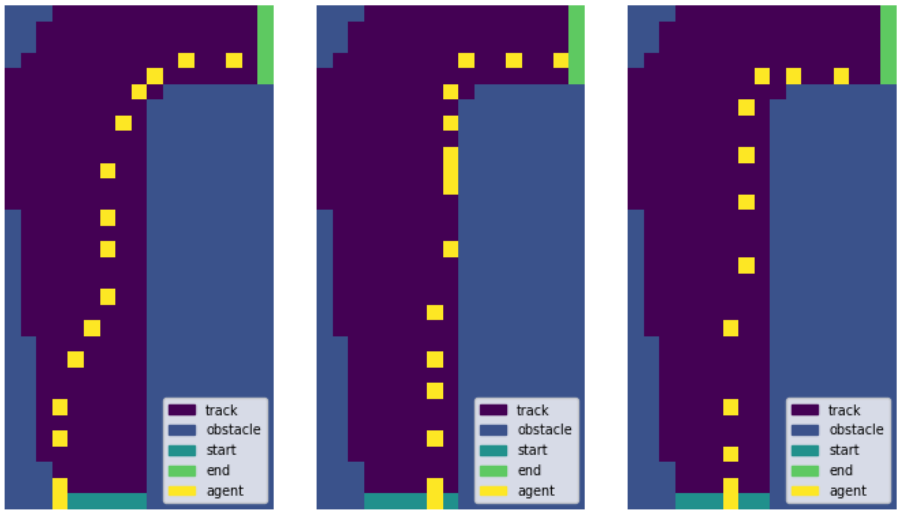

Figure 5: Approach of trained agent to finish line from different starting points

3.1.2 Case Study 2: More Exploration; $\epsilon = 0.1$

In the second scenario, I consider more exploration. Thus I consider a larger value for ϵ to decrease the chance of exploitation. Figure 5 shows the average

rewards with respect to episode number. After almost 30000 episodes, the algorithm converges. In this case, The reward function converges around -60. There are 6 possible starting points. After training the model, I ran the experiments for each of them which are shown in figures 6 and 7. These two figures show how the model approaches to the end line. The obtained rewards for each of them are $-213, -16, -12, -13, -13, -14$ respectively.

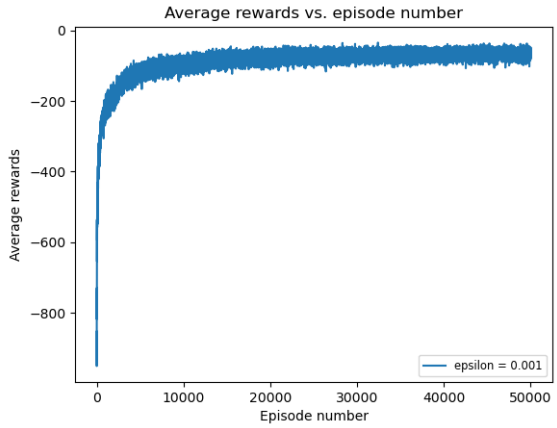

Figure 6: Average rewards vs. episode number

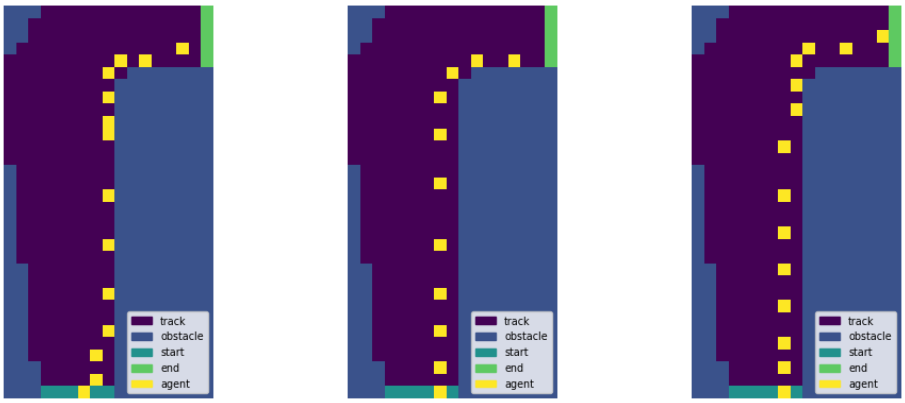

Figure 7: Approach of trained agent to finish line from different starting points

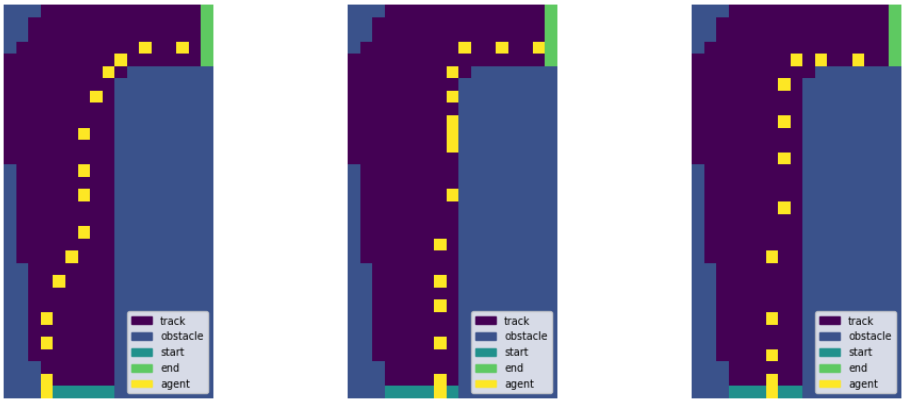

Figure 8: Approach of trained agent to finish line from different starting points

3.2 Map 2

The second map is provided in figure 9.

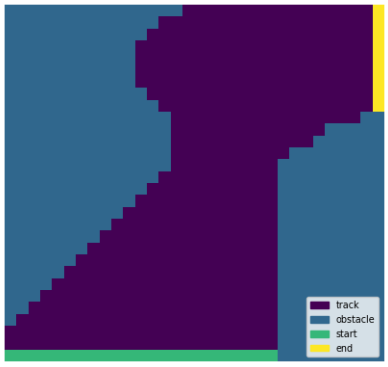

Figure 9: Map 2

3.2.1 Case Study 1: Less Exploration; $\epsilon = 0.001$

The first scenario is that using less exploration. Thus I consider a small value for ϵ to increase the chance for exploitation. Figure 10 shows the average rewards with respect to episode number. After almost 50000 episodes, the algorithm converges. In this case, The reward function converges around -20. There are 23 possible starting points. After training the model, I ran the experiments for each of them. Figures 11 and 12 depicts the approach of the trained agent from starting points 1 through 6 to the finish line. $-14, -15, -22, -15, -11, -11$ respectively.

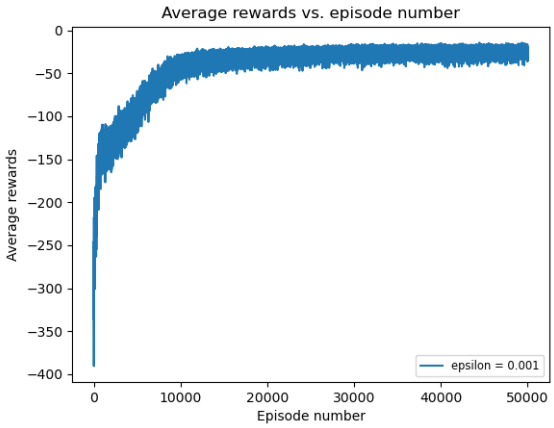

Figure 10: Average rewards vs. episode number

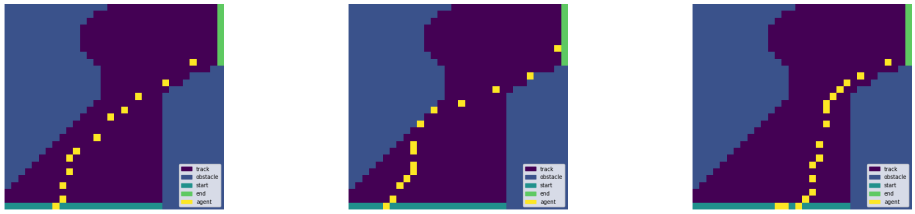

Figure 11: Approach of trained agent to finish line from different starting points

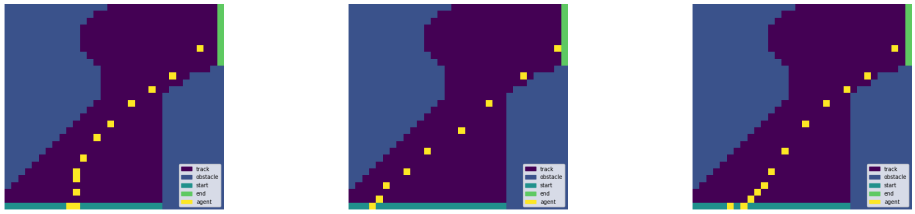

Figure 12: Approach of trained agent to finish line from different starting points

3.2.2 Case Study 2: More Exploration; $\epsilon = 0.1$

The second scenario is that using more exploration. Thus I consider a larger value for ϵ to decrease the chance for exploitation. Figure 13 shows the average rewards with respect to episode number. After almost 30000 episodes, the algorithm converges. In this case, The reward function converges around -22. There

are 23 possible starting points. After training the model, I ran the experiments for each of them. Figures 14 and 15 show the approach of the trained agent from starting points 1 through 6 to the finish line. $-12, -13, -11, -21, -16, -12$ respectively.

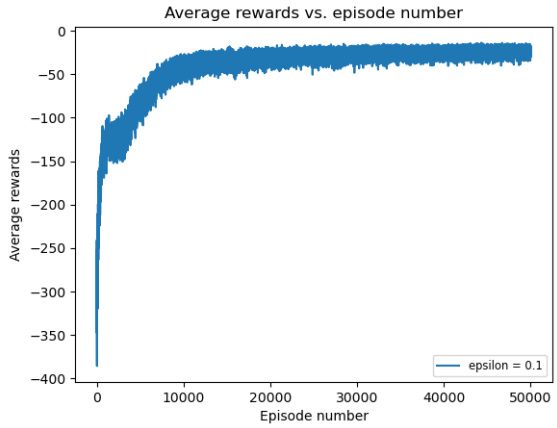

Figure 13: Average rewards vs. episode number

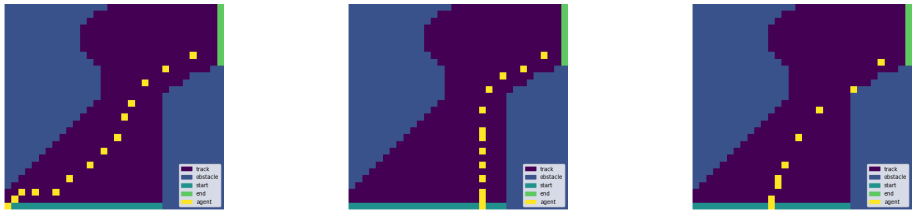

Figure 14: Approach of trained agent to finish line from different starting points

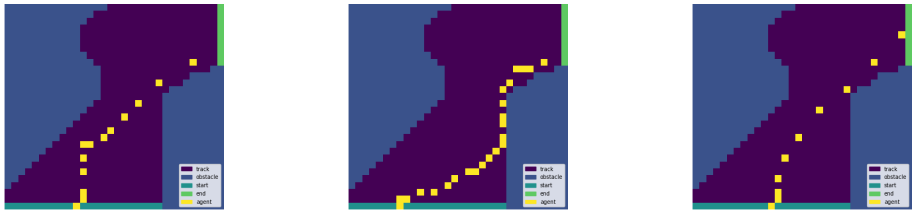

Figure 15: Approach of trained agent to finish line from different starting points

4 Extra Credit

The extra credits in my work are 3 folds:

1. Using 2 settings for ϵ.
2. Using 2 maps (one small and one large).
3. Visualizing the projection, rather than only plotting the rewards

All of them are included in the results sections.

5 Discussion

A Monte Carlo agent can successfully solve this task after roughly 50000 training episodes. In the first race track, overall when we have less exploration, the reward function is better, however in the second race track, the trend is opposite. One reason is that when the map is small, we require less exploration, however when it is larger, the need for exploration is more. It also shows the effect of choosing epsilon per environment. The second race track is bigger than the first one but it is easier to solve. However since the second race track is bigger, it requires more time for convergence.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.