RaceTrack Problem

Yi.zhou

October.14 2021

Introduction

According to the problem description, this assignment is to solve race car problem in a given map. Car wants to go as fast as possible, but not so fast as to run off the track. In the 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 one step, for a total of nine 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.

Goal

The propose of this assignment is to implement a RL program with Monte Carlo Method and find the optimal policy to reach the finish line. In this assignment, multiple policy will be tested, including basic soft policy, random policy, ϵ greed, and ϵ decay.

Methodology

Monte Carlo Method

Monte Carlo method has an advantage of only relying on the experience with the open model. Namely, I only need state, action, reward interacting with the environment and the method model of Monte Carlo.

In this Monte Carlo method, every state-action will be updated based on the previous episode which is a complete run of current policy. And the policy will be updated based the calculated Q value. The psydo-code is present below.

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

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

Repeat forever:
  $b \leftarrow$ any soft policy
  Generate an episode using b:
    $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1, T-2, \ldots$ down to 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 For loop
    $W \leftarrow W\frac{1}{b(A_t|S_t)}$

Figure 1: Off-policy MC control, for estimating $\pi \approx \pi\star$

Implementation

- First is to build the environment of race map. I used csv file to generate the simplified maps with reward. The start line is the one with reward 2 which doesn't make any effect in Q value but it's helpful building the environment. The race track has the reward 1, and the finish line has the reward 3, and 0s are the area that the car runs outside the track.

```
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0
```

Figure 2: an example of csv file of the map

- Second is to build the interaction function with the race map, it has the ability to store all the position a car has reached and the function to check where the car is to make sure it gets the corresponding reward. Also, the helper functions like acceleration is coded in the environment.
- The last is the Monte Carlo class that includes the Monte Carlo function and functions of create episode and visualization.

Result

The goal is to exhibit 5 complete run with optimal policy. Here are the parameters number I used for the example.

- policy: λ - greedy
- $\gamma = 1$
- $\lambda = 0.1$
- initial Q value = 0 (or any arbitrary value the lower the better)
- Here the screenshot of one exhibition

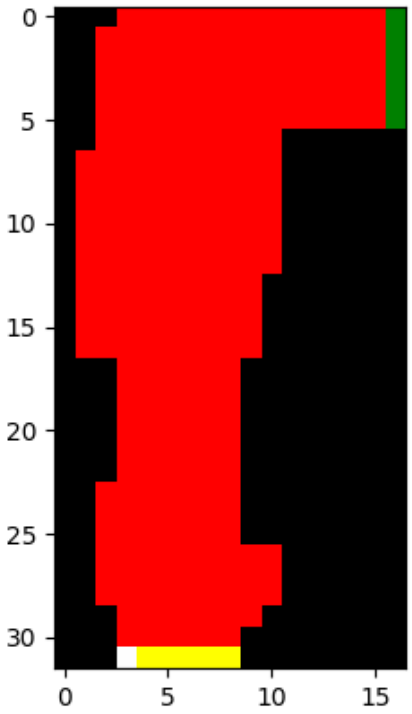

Figure 3: an example of exhibition

Furthermore

I have also made a plot of the episode and the steps with the policy I used. Policy tested include deterministic with $\epsilon = 0$, random with $\epsilon = 1$, $\epsilon - greedy$ with $\epsilon = 0.1$ and the decrease ϵ start from 0.99 to 0.1

Hypothesis

If the map is simple like the one above both $\epsilon - greedy$ and $\epsilon decay$ should present a better scale. Because in the simple map, it is not hard to explore a good path to the finish line. And $\epsilon decay$ will take longer to find out the optimal, since it shouldn't take too much exploration to find out a good path because as long as it goes to upright it should be able to reach the goal. However, obviously random and deterministic shouldn't present as good as $\epsilon - greedy$ or $\epsilon decay$, especially random action policy because the action policy is totally random and not updated by the MC method. Each of the result is shown below. I only run 200 episodes for deterministic policy it really takes too long to run 5000 episodes To make the process faster I set initial Q value to -100000 so that it will preform faster.

Result

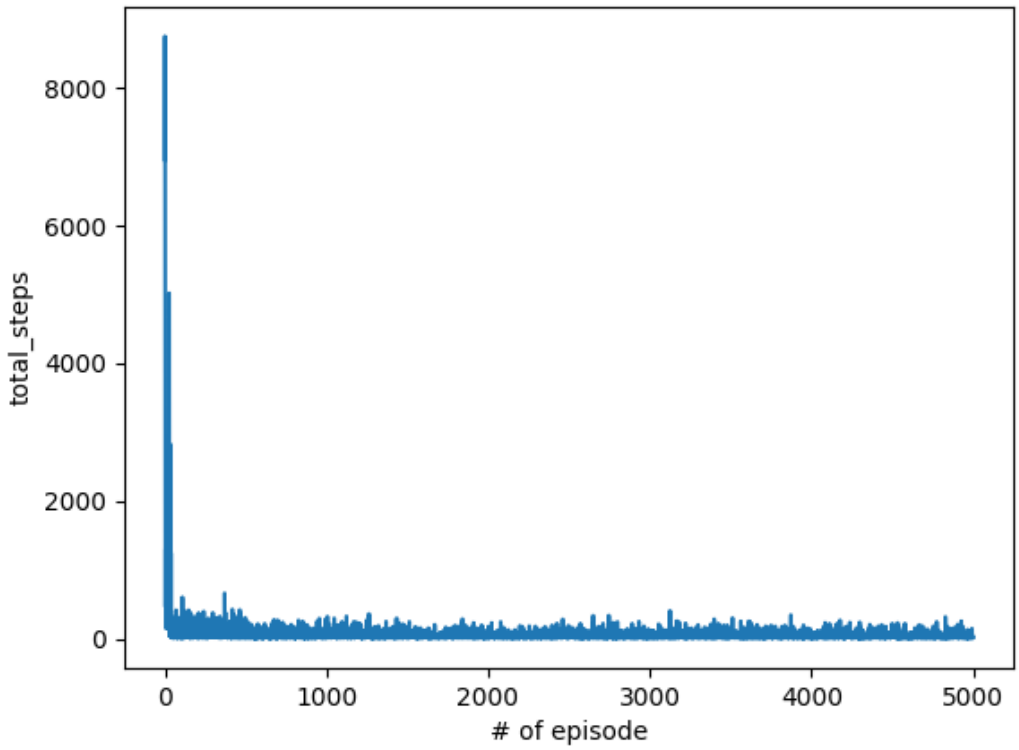

Figure 4: the plot of $\epsilon greedy$ with $\epsilon = 0.1$

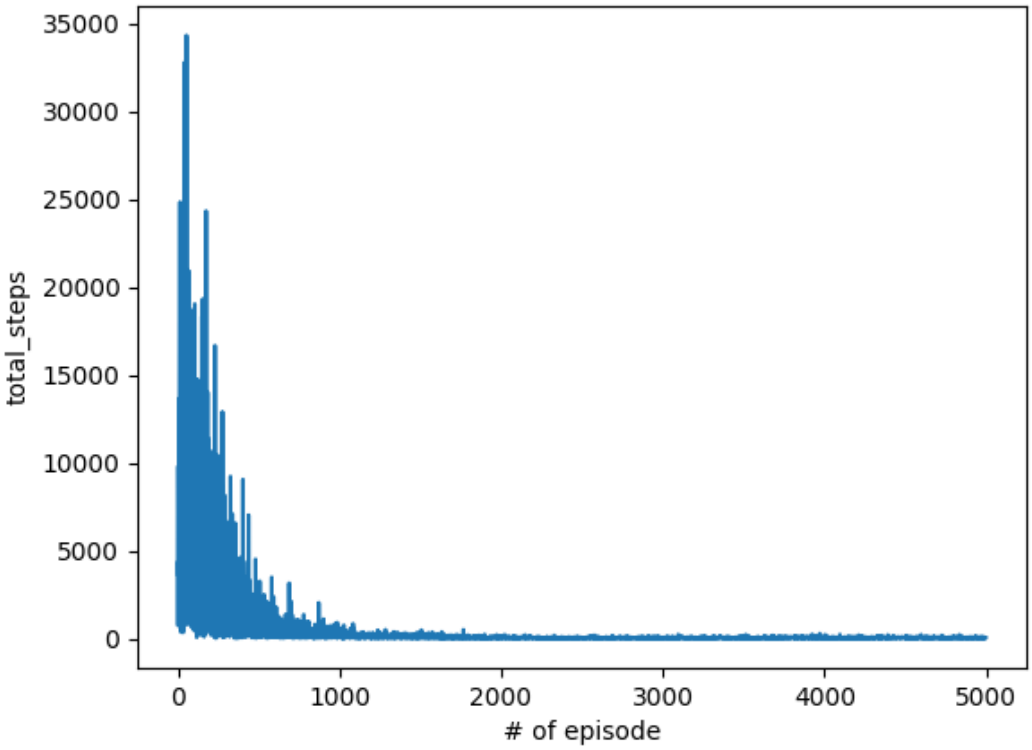

Figure 5: the plot of $\epsilon - decay$

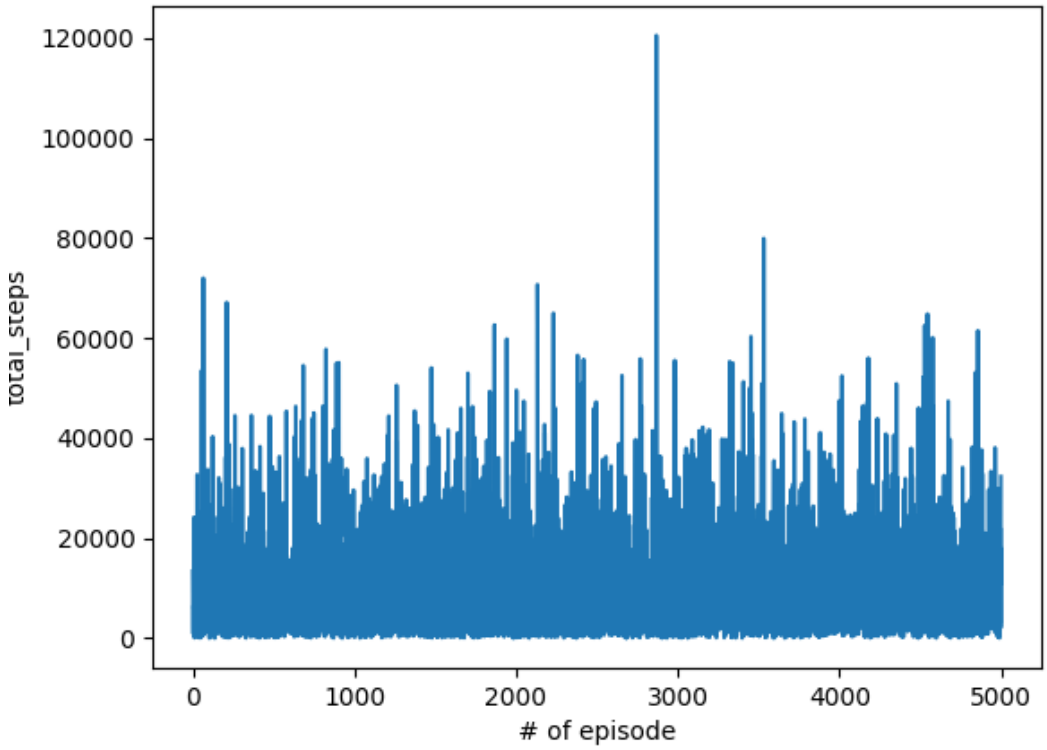

Figure 6: the plot of random policy

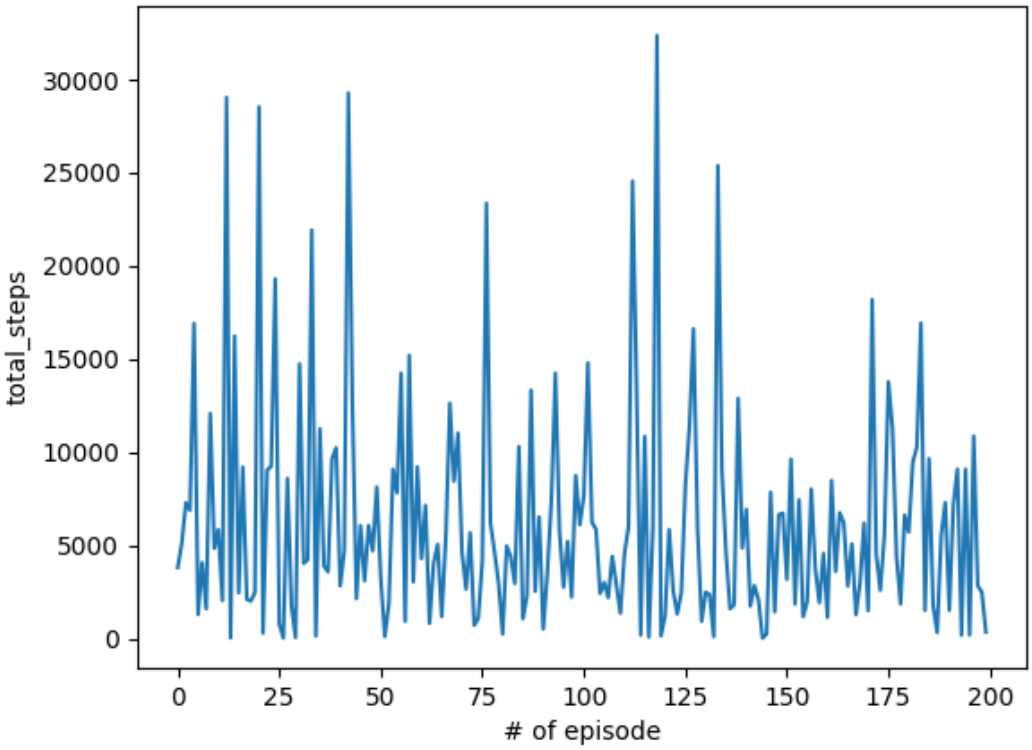

Figure 7: the plot of deterministic policy

Extra Credit

Exercise 5.1

Description

Consider the diagrams on the right in Figure 5.1. Why does the estimated value function jump up for the last two rows in the rear? Why does it drop off for the whole last row on the left? Why are the frontmost values higher in the upper diagrams than in the lower

Answer

- The reason of it is that when a player get a high sun for example a 20 or an ace, it is very possible to win the game, if dealer wants to win than

it has to beat it with a extremely high score which is unlikely to happen often.

- The last row happens when the dealer gets an ace, which means a play can only have an ace to draw which will cause value function drop since it is unlikely to happen
- The upper function is the one has the usable ace which reduces the chance to going bust than it's higher than the lower one.

Exercise 5.2

Description

What is the backup diagram for Monte Carlo estimation of q?

Answer

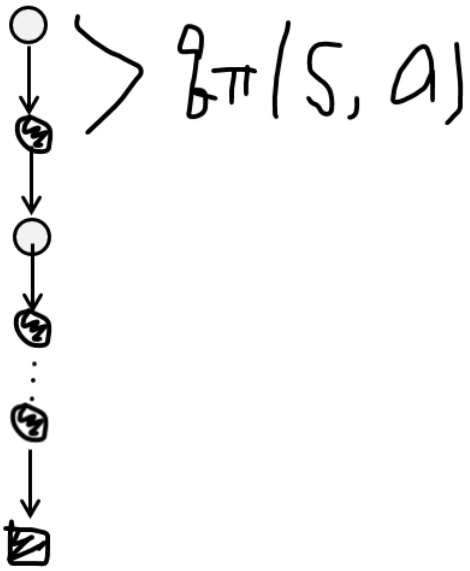

Figure 8: anwser of exercise5.2