Reinforcement Learning: Optimizing Lap Time on a Race Track

Emeka Oguike

October 18, 2021

1 Introduction

The Monte Carlo method for reinforcement learning learns directly from episodes of experience without any prior knowledge of MDP transitions. In this method, agents generate experienced samples and, then based on average returns, calculate value for each state-action pair. In this homework, we would be using On-policy first-visit Monte Carlo control ϵ-greedy policies to help an agent learn to navigate a race track from a starting location to a finish location.

2 Problem Statement

Consider driving a race car around a turn like those shown in Figure 5.5. 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 one step, for a total of nine 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. Apply a Monte Carlo control method to this task to compute the optimal policy from each starting state.

3 Experiment

The problem described above was simulated in a python environment. The race track was implemented as a character array where cells with the character 'B' denote the boundaries, 'S' the possible starting positions, 'F' the possible finish positions, and 't' the race track the agent is allowed to traverse.

The rules of the race were implemented by Umpire class, and the On-policy first-visit Monte Carlo control for ϵ-greedy policy was implemented by an Agent class. The results of the simulation include the paths traversed by the agent and the plot of reward vs episode number.

4 hypothesis

I expect agents with low epsilon values to have a higher, more constant reward vs episode plots in the steady state and traverse fewer paths on the race track if the agent start from a position with a straight path to the finish, but have lower less constant reward vs episode plots and traverse more paths if agents have to navigate a more complex path to the finish. Additionally, I expect that agents with higher epsilon values to navigate complex tracks more quickly than agents with smaller epsilon values because complex tracks would require more unusual actions from the agent to complete.

5 Results

Figure 1a and 1b depict the paths traversed by the agent and the returns vs episodes plots for an agent with epsilon value of 0.6. We can see that the agent traverses a wider variety of direction: vertical, diagonal and horizontal movements. For each episode, the agent is positioned in a random starting position in the track. As a result, the agent has to update it's policy for each new starting position. Hence a higher likelihood of exploration should help this agent correctly modify it's policy more quickly to finish the race. Figure 1a shows the path taken by the agent in the final episode of learning. Here we can see the agent traverses fewer cells in the track. This result looked too good to be true, so I ran the experiment again, and the results are shown in Figure 1c. While these two results are significantly different, they show the potential of exploration has to find good solutions.

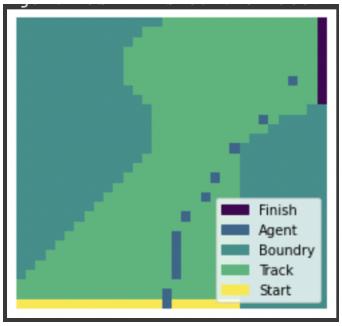

Figure 1a

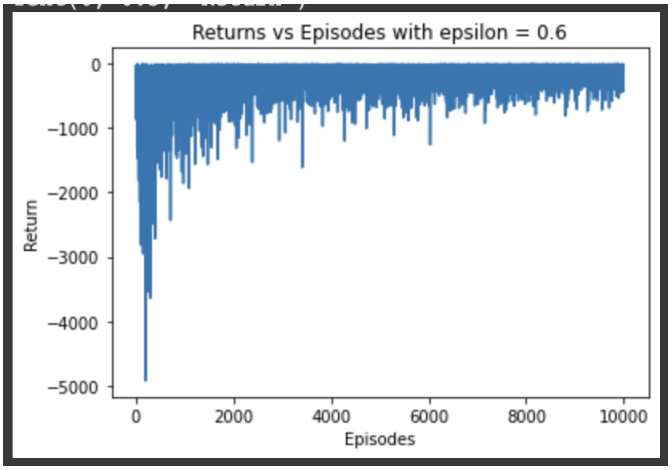

Figure 1b

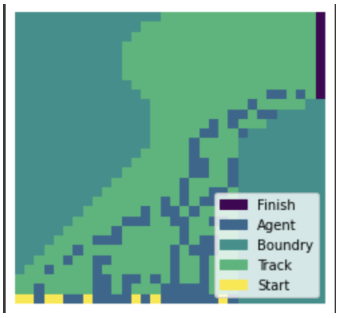

Figure 1c

Figure 2a and 2b depict the paths traversed by the agent and the returns vs episodes plots, respectively, for an agent with epsilon value of 0.2. From Figure 2a, since epsilon is fairly low, we can see that the agents movements are mostly diagonal because there isn't much exploration done. Also we can see the agent traverses more cells in the track, the agent is less sensitive to random changes in the starting positions after each episode. Hence it is slow to adjust its policy. This agent has not learnt to put much value on vertical movements initially because it is more greedy, and its greedy policy places more value on diagonal movements that work for a significant number of starting positions. As predicted, in Figure 2b, we see the returns vs episode plot has a less spiky steady state than that of the agent with epsilon value of 0.6 as this agent tends to explore with a comparatively lower probability of 0.2.

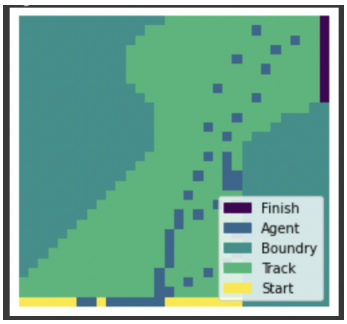

Figure 2a

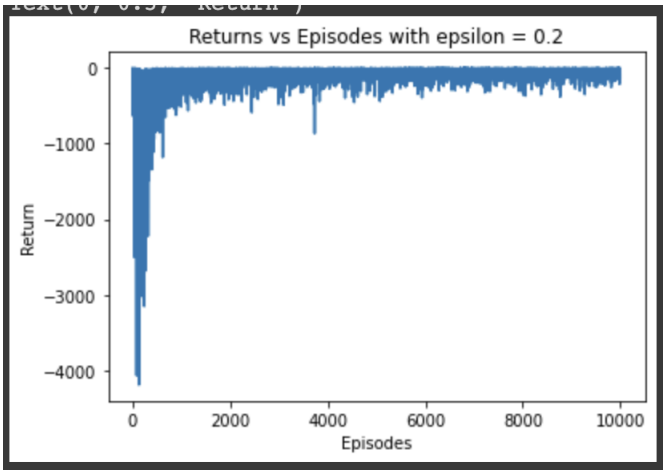

Figure 2b

Figure 3a and 3b depict the paths traversed by the agent and the returns vs episodes plots, respectively, for an agent with epsilon value of 0.05. Since epsilon is really low, we can see that, in Figure 3a, the agent still traverses a significant number of cells after the 10,000 episode. This is because the agent is a lot less likely to modify it's policy after a starting in a new position. as a result it needs more attempts: bumping into boundaries, restarting the episode, and rarely experimenting to change it's policy. This agent is very sensitive to the randoms starts for each episode.

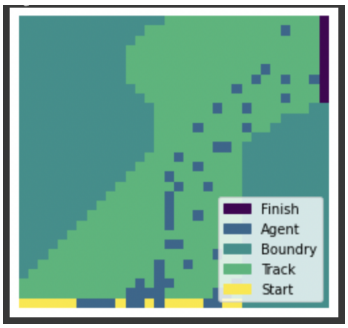

Figure 3a

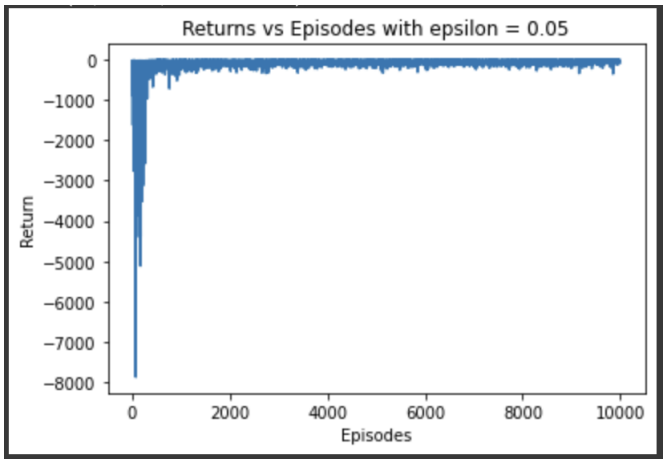

Figure 3b

6 Experimental Scenario

Some experimentation was done with two different tracks: one with an obstacle that limits the number of starting positions where diagonal movements get the agent to the finish positions and another with an additional obstacle in the

upper region of the track.

6.1 Results with Second Track

Figure 4a and 4b depict the paths traversed by the agent and the returns vs episodes plots, respectively, for an agent with epsilon value of 0.6 within a track with one obstacle. Only results for the epsilon value of 0.6 are shown. Lower epsilon values resulted in very long run times for each episode. In figure 4a we can see the agent traversed all the cells below the obstacle. Because the obstacle, agents have to find policies with more complex path geometries. As a result, the higher the epsilon value the more likely the agent is to explore such complex paths. With lower epsilon values, and hence less exploration, the agent spends more time with policies that do not help the agent navigate the complex paths needed to reach the finish positions.

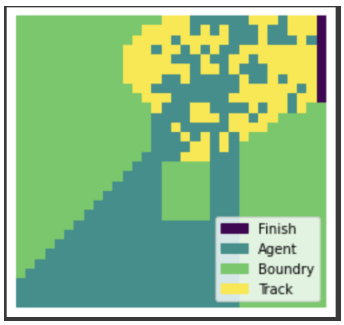

Figure 4a

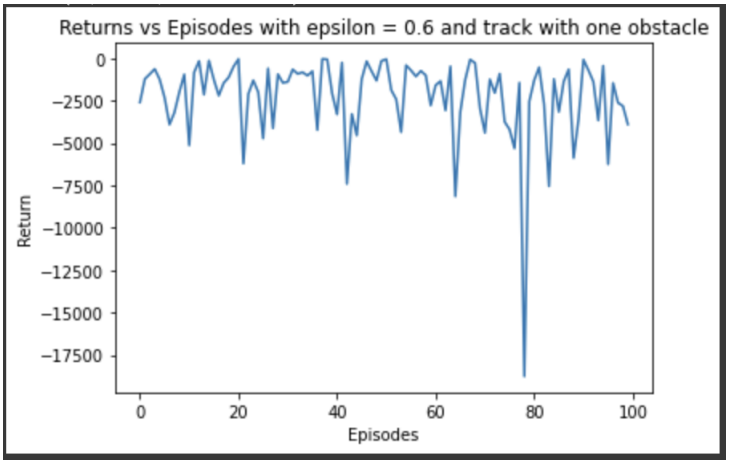

Figure 4b

6.2 Results with Third track

Figure 5a and 5b depict the paths traversed by the agent and the returns vs episodes plots, respectively, for an agent with epsilon value of 0.6 within a track with two obstacles. Only results for the epsilon value of 0.6 and 10 episodes are shown. Lower epsilon values, or higher episodes took a much longer time to run. In figure 5a, we can see that the agent traversed almost all the cells in the track in only ten episodes! Because the obstacle, agents have to find more policies with more complex path geometries. As a result, the higher the epsilon value the more likely the agent is to explore such complex paths.

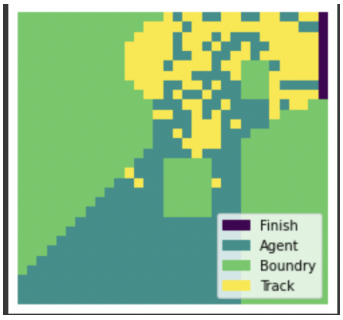

Figure 5a

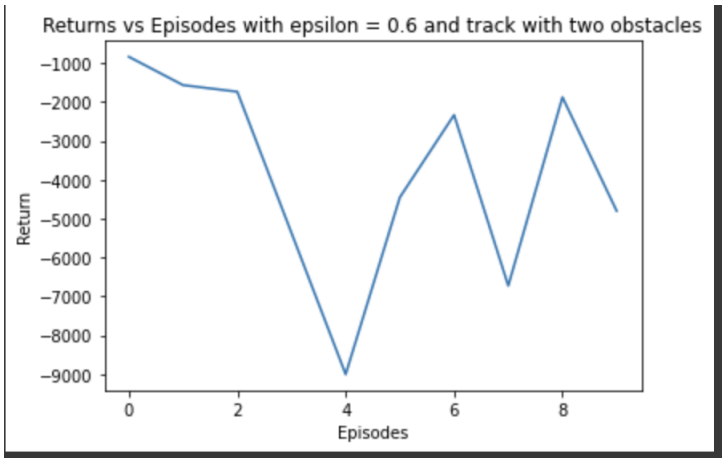

Figure 5b

7 Conclusion

The agents in the first track behaved as I had predicted. This homework problem revealed an interesting trade-off here: higher epsilon values require higher learning episodes to converge on an optimal returns; however it give the agent the best chance of stumbling on very good solutions. This benefit of agents with high likelihoods of exploration becomes apparent in the tracks with obstacles in the race tracks. Agents with lower epsilon values look extremely long computation times to finish each episode, while agents with high epsilon values were able to complete a significantly higher number of episodes in reasonable times.

This highlights the fact that, for a variety of difficult reinforcement learning tasks, high level of curiosity from agents not only raise the probability of finding feasible solutions but also cut computational time.