CIS 419/519: Homework 6

{Zhuheng Jiang}

Although the solutions are entirely my own, I consulted with the following people and sources while working on this homework: Yihang Xu, Yupeng Li, Yongxin Guo

https://www.cse.iitm.ac.in/ ravi/courses/Reinforcement

https://github.com/openai/gym/wiki/MountainCarContinuous-v0

I would like to use 3 out of my 6 late-days, thank you!

1 Reinforcement Learning I

This reward function does not communicate the goal to the agent effectively. The problem is that the reward will be given only after the robot get out of the maze, but the robot receives no feedback in the whole process of attempting to escape. This process is so long that the robot cannot learn effectively. It keeps trying its fortune aimlessly in the maze.

A better way can be giving the robot negative rewards when the robot get into the deadlock, and positive rewards when getting through a path.

2 Reinforcement Learning II

a. Since the GridWorld is a continuous work, only the interval between them is important. But if the work is modified into periodic, then the signs will matter as well.

b. According to the given equations, adding a constant to the reward leads to prem R:

$$\begin{aligned} R_t^{'} &= \sum_{k=0}^{\infty} \gamma^k (r_{t+k+1} + C) \\ &= \sum_{k=0}^{\infty} \gamma^k (r_{t+k+1}) + \sum_{k=0}^{\infty} \gamma^k C \\ &= R_t + \sum_{k=0}^{\infty} \gamma^k C \end{aligned}$$

Then we can take the prem R into the equation of value of a state, and modify the equation based on the fact that the gamma is a constant smaller than 1:

$$\begin{aligned} V^{\pi'}(s) &= E^{\pi}[R_t^{'} \mid s_t = s] \\ &= E^{\pi}[R_t + \sum_{k=0}^{\infty} \gamma^k C \mid s_t = s] \\ &= E^{\pi}[R_t + C \sum_{k=0}^{\infty} \gamma^k \mid s_t = s] \\ &= V^{\pi}(s) + \frac{C}{1-\gamma} \end{aligned}$$

From the above process the proposal can be proved.

c. From the part b, the constant K is:

$$K = \frac{C}{1-\gamma}$$

3 Random policy for the MountainCar gym environment

i. Observation space: a two dimensional space in which every element denotes one state of the mountain car. It is consisted of car position and car velocity information.
Action space: a discrete space in which every element denotes the car's action. Three actions in total: push left, no push and push right.

ii. Under random policy, the mountain car's mean reward after 10 episodes is -200.

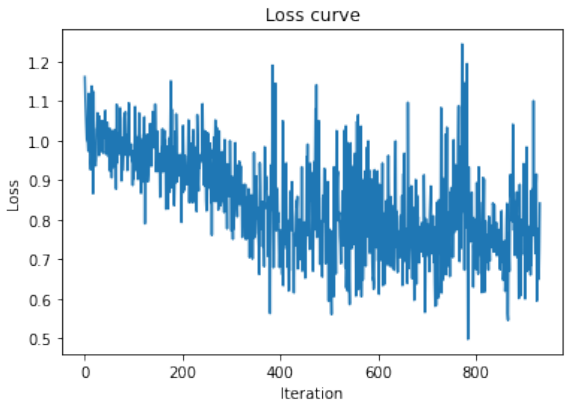
4 Train a Q-learner and generate expert trajectories

i. The original state of the car is continuous and infinite, which is not applicable to the learning model. The function discretize() is converting the raw observation into discrete state parameters and round them so that they can be used for model training.

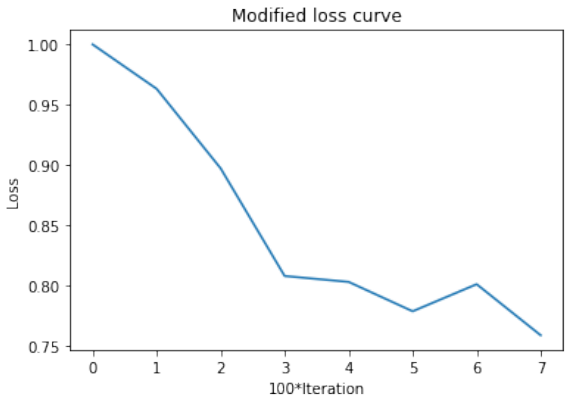
ii. Varying the discretization size will change the resolution of the mountain car's observation state.The interval after discretization will be bigger if bigger discretizations are applied. This can indirectly increase the quantity of possible obsearvation states.

5 Train an imitation policy

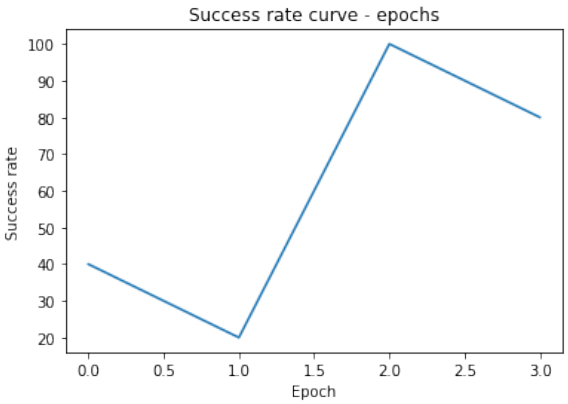
i. If the loss is recorded in every iteration, the curves will be really messy and oscillating, I chose to record the mean loss every 100 iterations and plot the graph to get a better result. I also recorded the detailed loss in every iteration.
The plots for 2 epochs and 20 episodes are attached below.These plots corresponds to pi_0:



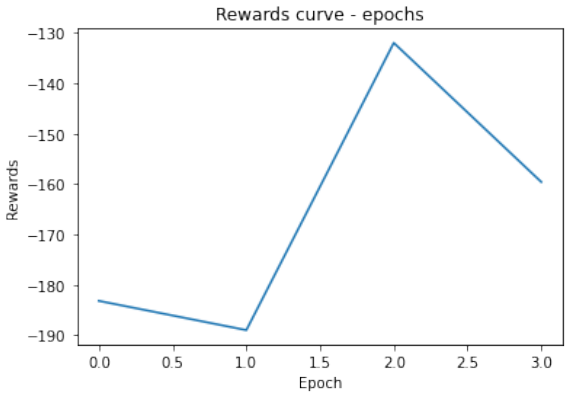
20 episodes Loss - iterations
(a)



20 episodes Sparse loss - 100 iterations
(b)

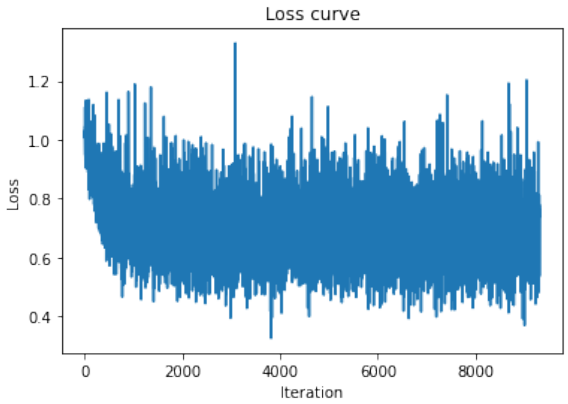


20 episodes Success rates - epochs
(c)

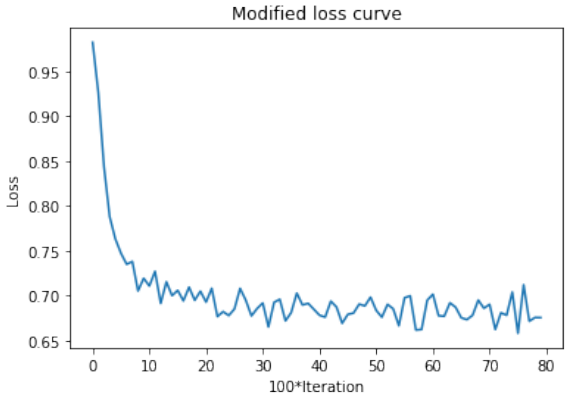


20 episodes Rewards - epochs
(d)

From the previous plots, we can find that the loss is gradually decreasing as iterates. But the epochs are so limited for the success rate and rewards curve.
Then the plots for 20 epochs and 2 episodes is attached in the next page, which corresponds to the pi_1:



2 episodes Loss - iterations
(e)

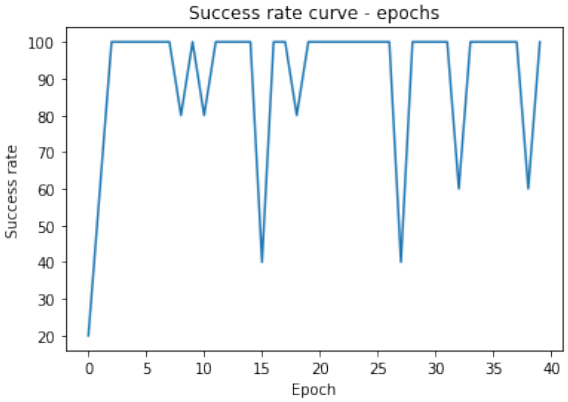


2 episodes Sparse loss - 100 iterations
(f)

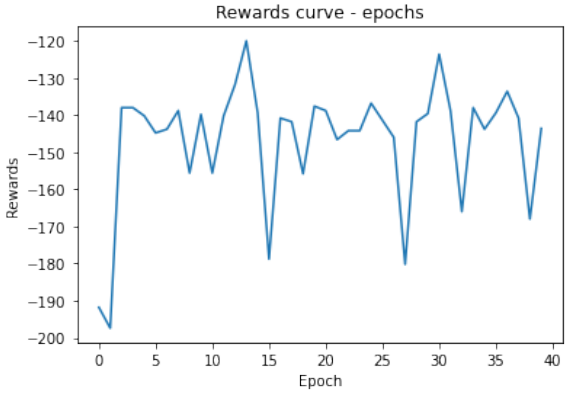
From the plots in this and the next pages, we can see the trend more clearly that the loss is efficiently decreasing with iterations. And the success rate can reach a high level rapidly, but may exist some fluctuations.

6 Implement DAgger

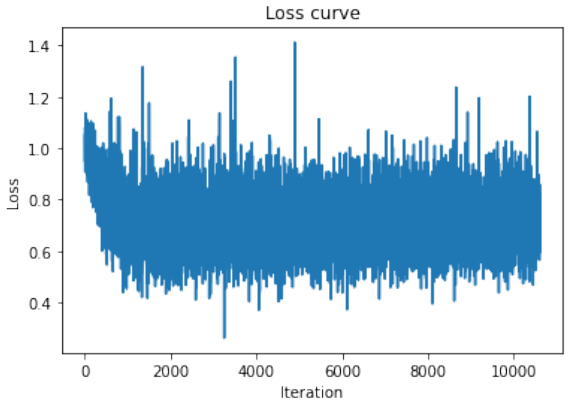
The curves for DAgger are attached. From the comparison between the curves from DAgger and non-DAgger we can find:
1. The loss decrease faster when executing the DAgger. It requires less iterations.
2. The success rate turns out to be more stable than the previous results.
3. The rewards grows faster and keeps more stable than the model without DAgger.



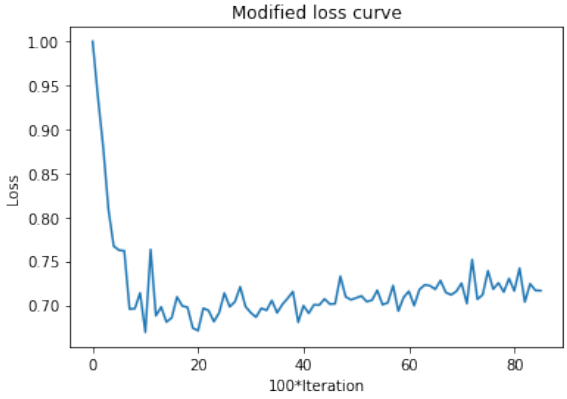
2 episodes Success rate - epochs
(g)



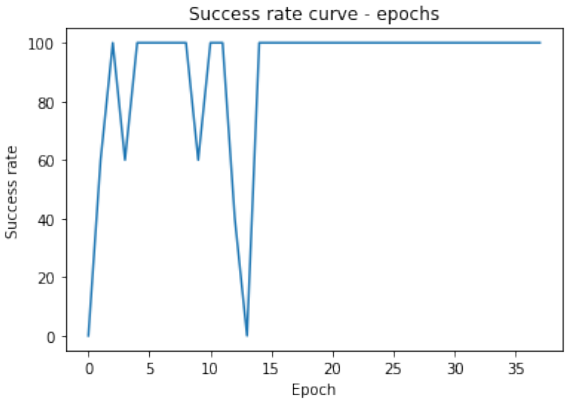
2 episodes Rewards - epochs
(h)



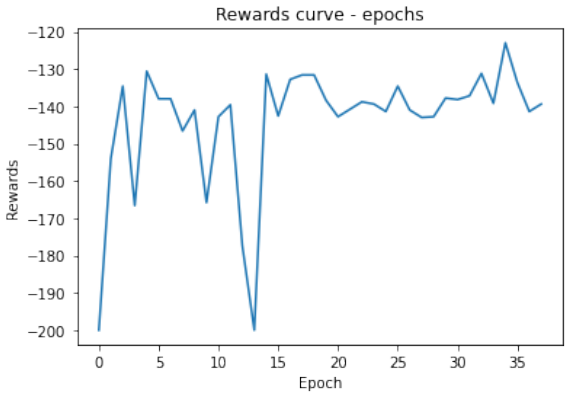
DAgger Loss - iterations
(i)



DAgger Sparse loss - 100 iterations
(j)



DAgger Success rates - epochs
(k)



DAgger Rewards - epochs
(l)