Reinforcement Learning - Assignment - Policy Gradient
Student ID: m10902121 Name: 李育誠

1. Project topic

Policy Gradient, Basketball-v2

2. Environment Configuration

Programming Language: Python,
Packages: Pytorch, Tkinter, Matplotlib, Numpy

The environment was made on me. I spend lots of time building up the environment, in order to make people observe the actions that the agent did in the real basketball domain. The grid world contains four elements, basket, basketball, opponents, and robot. Since the robot, or the system itself, can choose five actions in each state, up, down, left, and right. For my definition, dribbling is the agent with the ball. Each state is the matrix with digits, the number reflects the agent, ball, basket, and opponents.
For example, the agent without the ball can be recognised as the (Fig. 1), and the other picture (Fig. 2) is the agent with the ball.

0	0	3:opponent1	0	0	0
0	4:agent	0	3:opponent2	0	0
1.ball	0	0	0	0	2.basket
0	0	3:opponent3	0	0	0
0	0	0	3:opponent4	0	0

Fig.1 The enviroment of agent without ball

0	0	3:opponent1	0	0	0
0	4:agent	0	3:opponent2	0	0
0	0	0	0	0	2.basket
0	0	3:opponent3	0	0	0
0	0	0	3:opponent4	0	0

Fig.2 The enviroment of agent with ball the agent change the color to yellow

3. Training Results

Besides the rewards defined by the assignment, I define some rewards as well. If the agent stays in the field the reward is +1. If the agent is not with the ball and the agent is near the ball the reward is 2 * distance. The agent would keep bumping the wall at the beginning, therefore when the number of the training episodes is higher than 2000 the agent itself will become different. In my opinion, the situation is because of the law of large numbers, when the agent tried to choose the higher probability actions, it seems to have no chance to change the direction. However, when the number of trails is high the chance to change to the small probability comes across. However, after I trained for 1,2000

episodes the agent still hard to get the score properly. I thought the training loss is almost converged, so I decided to stop the training. And even though it is hard for training the agent to get the score. As my observation, the agent learns about how to approach the basket.

Environment Configuration

(1) WIDTH * HEIGHT = 9 * 6 and 5 Opponents

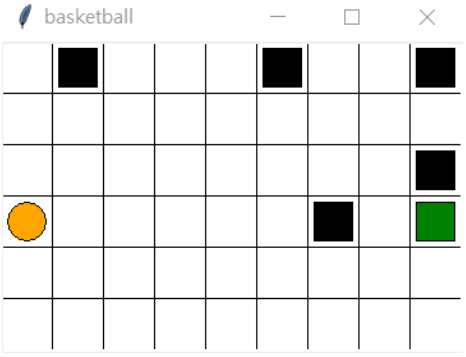


Fig.3 The configuration of 9 * 6

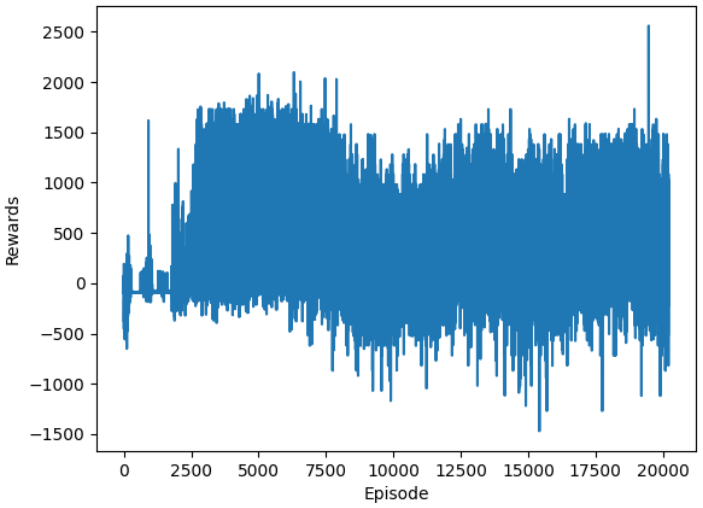


Fig.4 The training rewards through the whole episodes

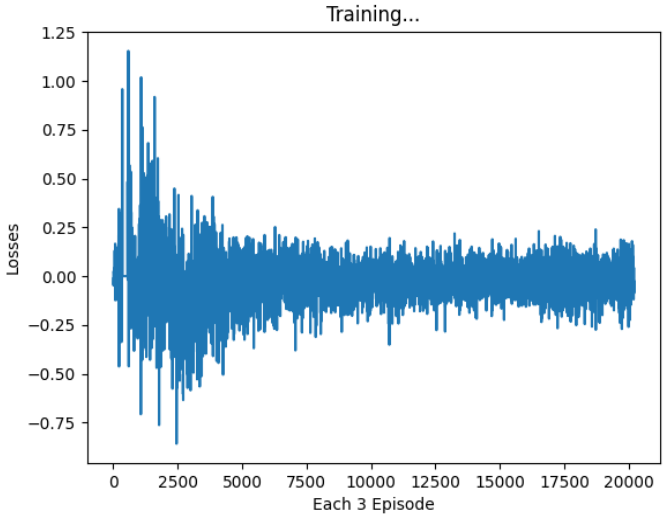


Fig.5 The training losses through the whole episodes

(2) WIDTH * HEIGHT = 12 * 10 and 10 Opponents

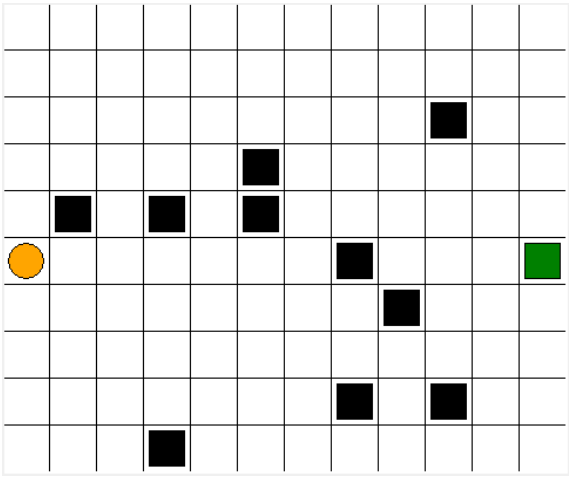


Fig.6 The configuration of 12 * 10

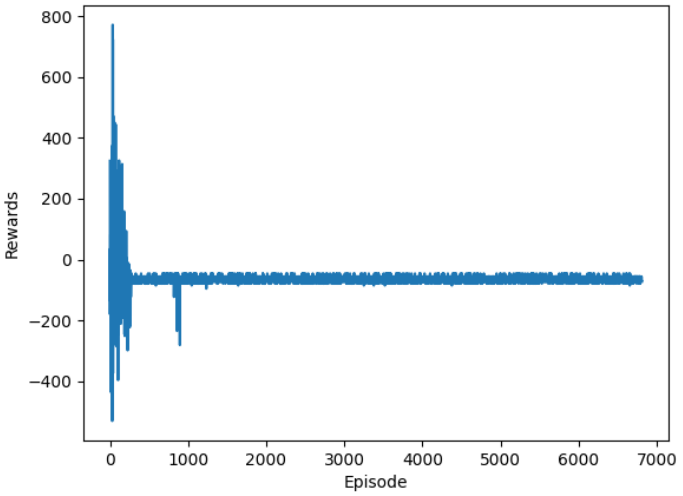


Fig.7 The training rewards through the whole episodes

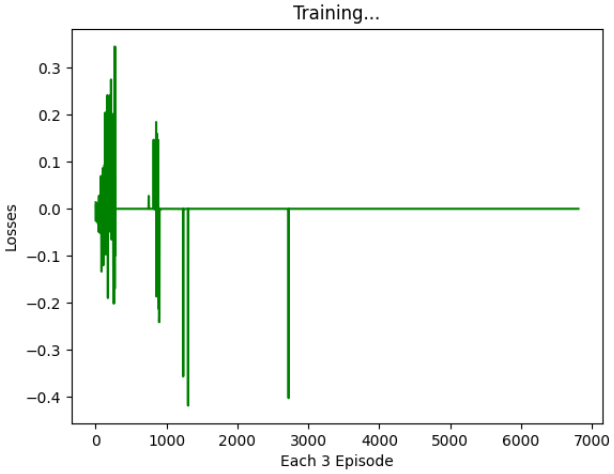


Fig.8 The training losses through the whole episodes

(3) WIDTH * HEIGHT = 12 * 10 and 10 Opponents

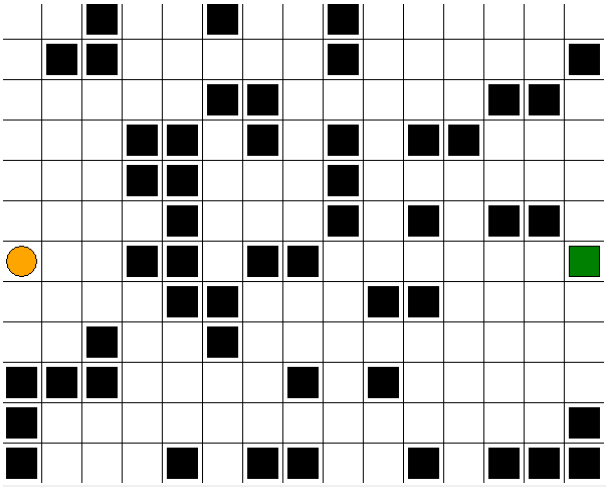


Fig.9 The configuration

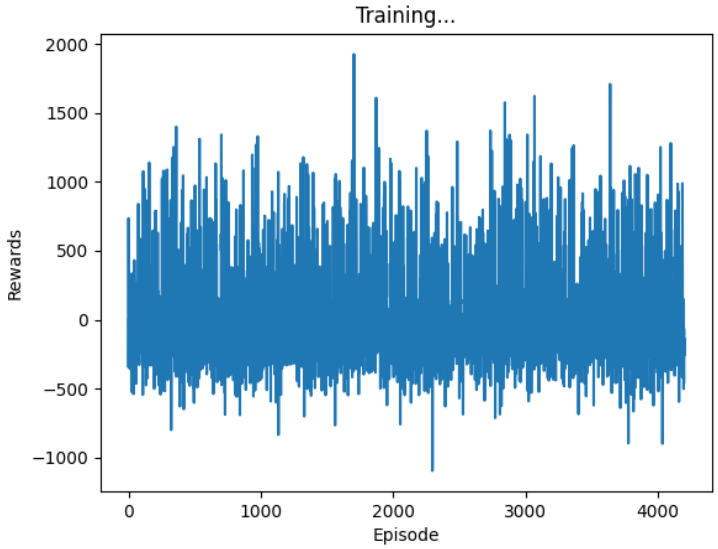


Fig.10 The training rewards through the whole episodes

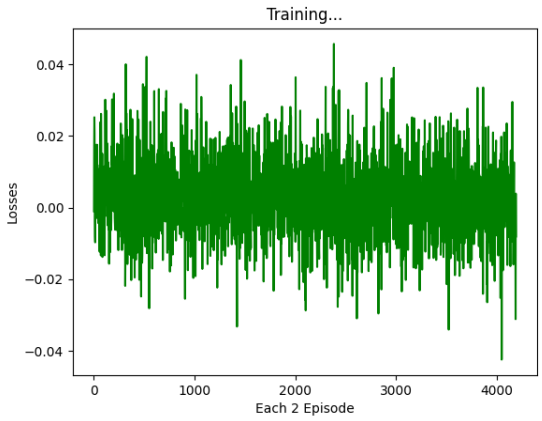


Fig.11 The training losses through the whole episodes

4. Approach

(1)Policy Gradient Introduction:

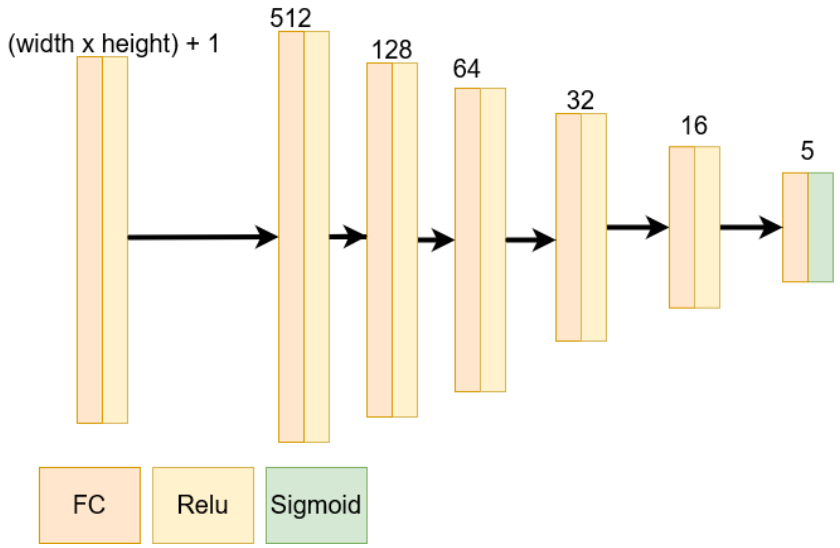
Theta is the neural network parameters, and the log function is the loss function for the network to update the parameters.

function REINFORCE
Initialise θ arbitrarily
for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$
for $t = 1$ to $T - 1$ **do**
$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
end for
end for
return θ

Algorithm 1.

(2) Neural Network Model

The input features is the map size and one with ball parameter. The outputs are the probability distribution of 5 actions. The model itself is to learn the state have different



5.Reflection and Discussion

I was not impressed about the results, since my network architecture is not well-designed compare to the state-of-art network architecture. It is interesting that I can design any kinds of different policy on my own to make sure the agent can train well for the huge episode. However, to well-designed the reward function is still a long way for me to learn. I think maybe I can choose to add the epsilon-greedy in this domain, because in the first configuration. It is hard for the agent to learn about how to find the good routes in each episode, and sometimes it even bumped into the walls. It make me feel stressed that I think I should change the reward function when I saw this situation. The other way of training is that I think I should train for more epsiodes because I only trained for 10,000 epsiodes which seems to be not enough in this domain.