Udacity

Deep Reinforcement Learning Nanodegree

Project 3—Collaboration and Competition

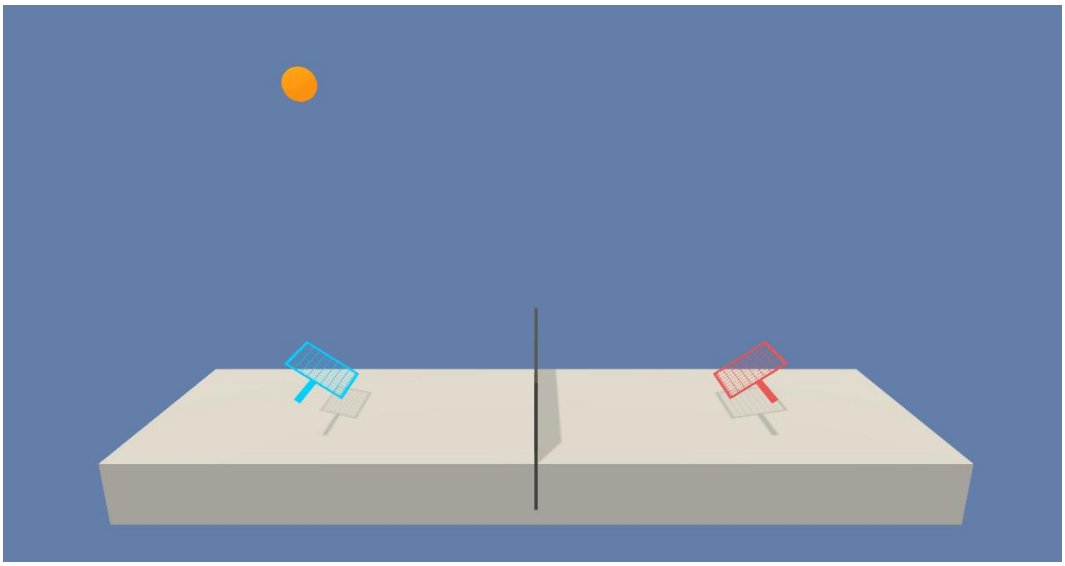
Liu Tianze

I. Introduction

In this project, the task is episodic and I used DDPG to train the agents to achieve the goal which the average score is at least +0.5 over 100 consecutive episodes.

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.



II. DDPG

For architecture, I implemented the DDPG codes which I written for project 2 and change a little for the multi-agent task.

In DDPG, there was an actor-critic architecture. The actor network was used to

determine the best action for a specific state through policy gradient method and the critic network was used to evaluate the policy determined by actor by TD (Temporal Difference) error. In other words, when the actor used policy gradient to find the best action through gradient ascent, the critic network would tell the actor whether the direction of gradient ascent was right or not.

The formulas used for updating weights for actor and critic networks were as follows:

Actor：

Update the actor policy using the sampled policy gradient：

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a \mid \theta^Q) \mid_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu) \mid_{s_i}$$

Critic：

$$Set\ y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$$

Update critic by minimizing the loss：$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

I also add noise (OU Noise) to their actions at training time in order to make DDPG policies explore better.

After many trial and error, I found that 2 FC layers was enough. For neurons of FC layers, I have experimented many different combinations. For example, 512/512, 512/256, 256/256, 256/128 and so on. I found that 256/256 could have the best results. Hence, too many or too few neurons would not result in a better performance.

The hyperparameters of DDPG were as follow:

Replay buffer size	500000
Batch size	64
Gamma	0.99
TAU	0.005
Learning rate of actor	0.0002
Learning rate of critic	0.0001
Weight decay	0

The agent used replay buffer as length 500,000 and it could have the ability hold important experiences during the training for the multi-agent task. After experimenting, I found batch size 64 was suitable for training. The soft update of target parameters (TAU) was 0.005. Furthermore, I wanted the agents to focus on the long term returns so the discount factor gamma was set to 0.99.

During this project, I used two DDPG agents (agent_1 and agent_2) and each

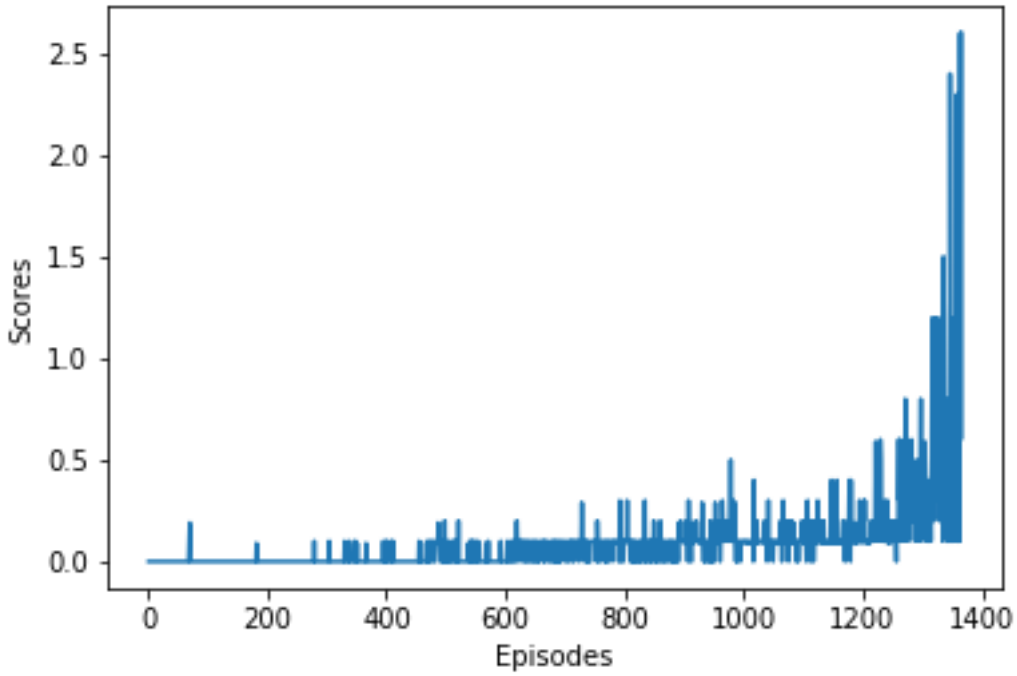
agent received its own observations as well as the observations of its co-player. That is to say, the states for each agent would be doubled (The state sizes were 24*2=48).

After each episode, in order to get a score for each agent, I added up the rewards that each agent received. It yielded 2 (potentially different) scores. I then took the maximum of these 2 scores and added to the deque which length was 100 for calculating the average score of recent 100 consecutive episodes.

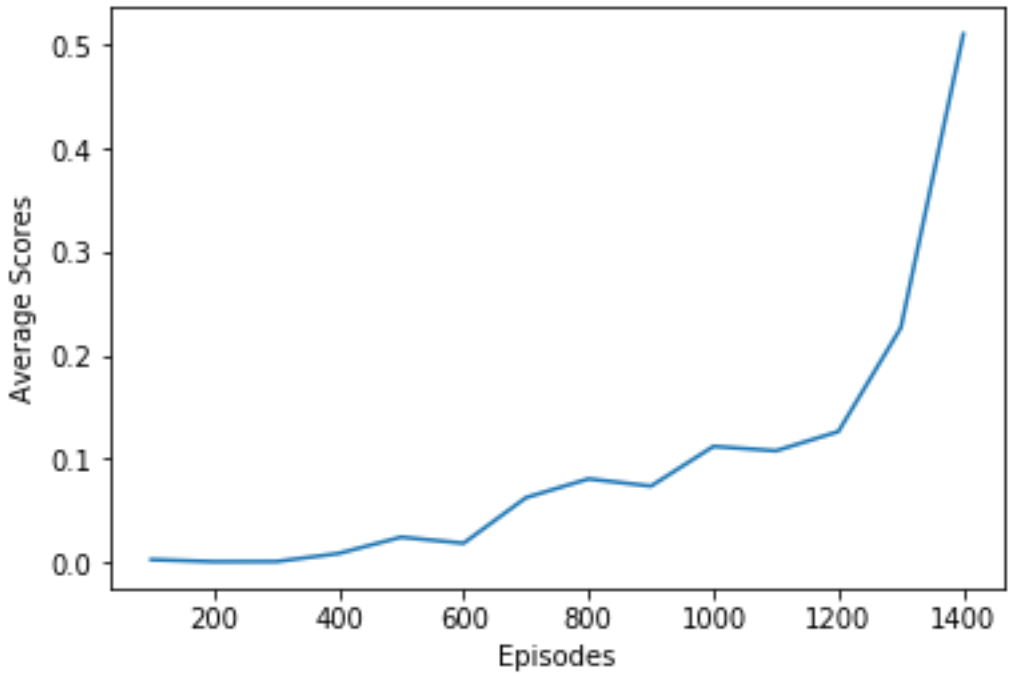
III. Results

From the results of my project, it achieved +0.5 scores in average below 1500 episodes. At first, the average score was around 0. After about 800 episodes, the average score was closed to +0.1. After 1300 episodes, the average score increased dramatically. Finally, it achieved the average score of +0.5 at 1365 episodes.

The training progress of DDPG model was as follow:



The average scores of training process of DDPG model was as follow:



IV. Further work

Although I have achieved the goal, there was still some room to improve. For example, I could use Multi-Agent DDPG (MADDPG) as my network, maybe it would converge faster because it used shared critic network. In addition, I could also use MA-D4PG as my network, maybe it would make me get better results. Furthermore, in order to improve my abilities and practical experiences, I can use Multi-Agent RL algorithms to try the optional project: Play Soccer.