Project 3 “Collaboration and Competition”

Dongsu Du

Part of the Project is based on

https://github.com/udacity/deep-reinforcement-learning/tree/master/ddpg-bipedal

1 Introduction

I decide to use multi agent ddpg model for this continuous control project since it is easy to implemented for multi agent learning. The basic structure of the whole project is list below.

1. Define the Network architecture in ***model.py***
2. Define the Agent in ***maddpg_agent.py***
3. Generate 2 agent in ***Tennis.py***, each agent has its own Actor and Critic model
4. Train the model in ***Tennis.py***
5. Modify the Network structure, Network parameter update frequency, and

1 Network Architecture

I tested to use both 3 and 2 hidden layers in this project. I first tried 3 layer network structure with 256/128/64. Then I tried 2 layer network structure with 256/128 hidden layers for both Actor and Critic network. Surprisingly, the 2 hidden layer network performance is better than 3 layers network. It starts to learn how to hit the ball within 30 episodes and keep improve its performance to achieve average 0.8 score (last 100 episodes).

Another important factor is the input of each network. The agent1 and agent2 should use as much as possible information in their network. Therefore, the input data for Critic network will be the States, States_next, and Actions of both agents. The input data for Actor Network is the States of both agents. But Agent_1 and agent_2 only predict its own action.

2 MADDPG Agent

I setup 2 agents for agent1 and agent 2. It increase the flexibility of the model and improved the performance. Instead, I can also setup only 1 agent to predict both agent 1 and agent 2's actions. But it will not be flexible enough to achieve good model, since the 2 agents have different reward function.

The update frequency is another factor that affect the model performance and training speed. I tested to update the network 5 times per timestep and 1 time update per timestep.

Batch size	LR_ACTOR	LR_CRITIC	GAMMA	TAU
128	1e-3	1E-3	0.99	9e-3

3 Important added noise

Since it is very difficult to find the action that can hit the ball. At the beginning of the training, we need to make the action more like a random action. Therefore added a large noise in the action is important. I tested with and without noise training. The training without added noise to action never reach a

4 Training process

The n_episodes is set to 2000. The training . The training process is defined in "Continuous Control.ipynb". The network will be saved if the mean score value in last 100 episodes >0.5.

5 Result

4.1. Training time depends on the frequency of network updates and the network structure. The speed of the training also depends on the learning rate. Please see table 1 for training speed on CPU (for 2 layers network), with different timestep and learning rate.

Hidden layers	256/128	256/128	256/128
Update frequency	5 per timesteps	5 per timesteps	1 per timesteps
Learning rate	1e-3	2e-4	1e-3
Episode to reach >0.5 average score	1380	cannot reach	1220
Total training time	~2.5 hours	NA	~0.5 hours

4.2. The scores for network 256/128. 5 updates per iteration. No noise added to action

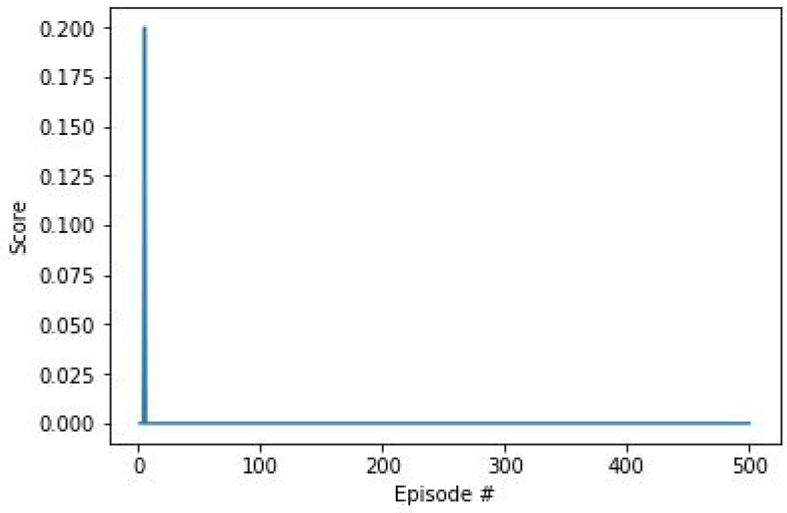

Figure 1: Network:256/128. No added noise. The agent never learn how to hit the ball.

4.4. The scores and moving average scores for network 256/128. 5 updates per iteration. With added noise

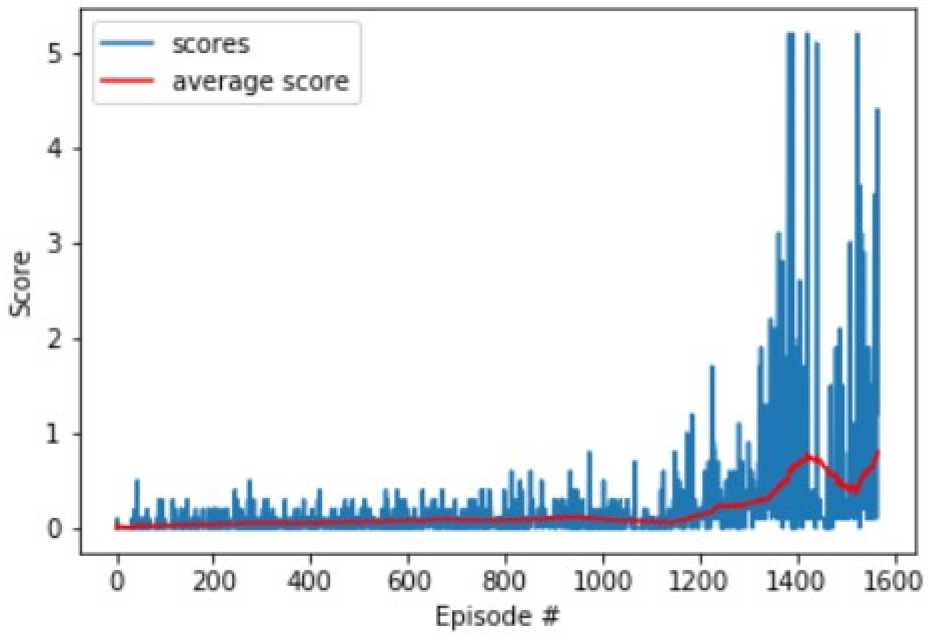

Figure 2: Network:256/128. Added gausian noise. The average score achieves 0.8 in 1500 espisodes.

4.5. The scores and moving average scores for network 256/128. 5 updates per iteration. LR = 2e-4. The learning is very slow and only reach 0.1 in 2000 Episodes.

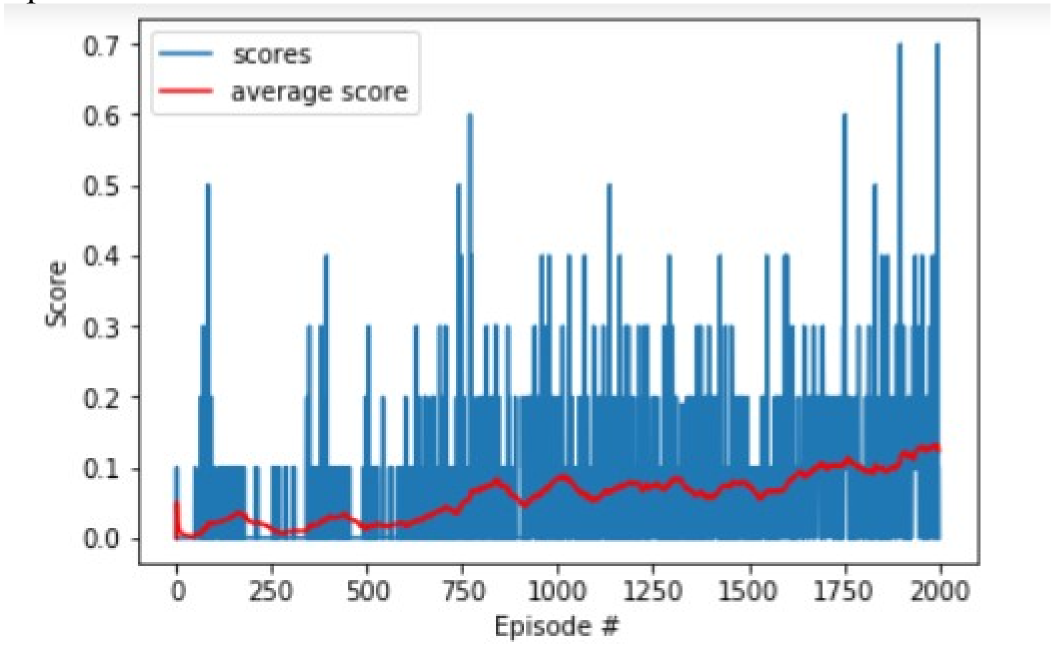

Figure 3: Network:256/128. Added gausian noise. Learning rate = 2e-4

4.6. The scores and moving average scores for network 256/128. 1 updates per iteration. LR = 1e-3. The learning speed is very fast and achieve 0.5 average score within 1200 episodes (0.5 hours)

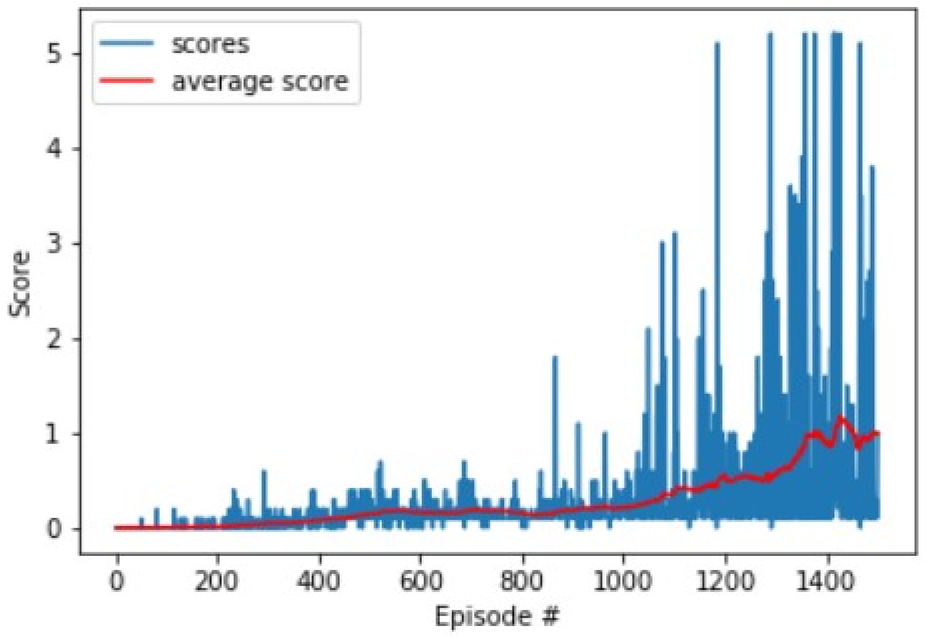

Figure 2: Network:256/128. Added gausian noise. Updatew 1 time per time step. The average score achieves >1 in 1500 espisodes.

5 Discussion

The Noise is very important in this model, because without initial noise, it is very difficult to find an action to hit the ball, then the agent will not learn anything. The noise decay make sure the noise is smaller and will not affect the agent's performance when it get trained well.