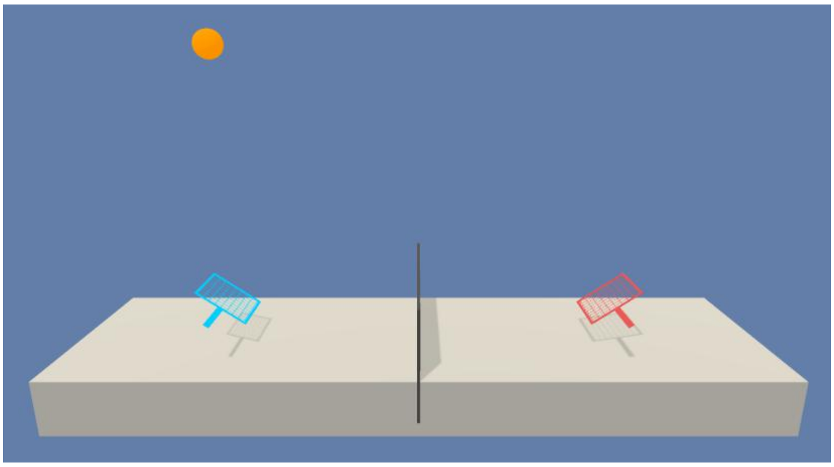
Deep Reinforcement Learning Nanodegree

Project 3 : Collaboration and Competition

Chenbo Gu

1. Problem Introduction

In this project, two agents controlling rackets are trying to bounce a ball over a net and keep the ball in play.



Reward:

An agent hits the ball over the net: reward +0.10 for this agent

An agent lets a ball hit the ground or hits the ball out of bounds: reward -0.01 for this agent

State:

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.

Action:

2 continuous actions available for each agent:

movement toward (or away from) the net, and jumping.

Target:

An average score of +0.5 over 100 consecutive episodes, after taking the maximum over both agents.

More detailed environment information can be found on the github website provided by Unity:
https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md

2. Training Algorithms

DDPG is mainly adopted for solving this problem. Many different parameter sets are tested. Many changes are made to the sample code provided by Udacity. Generally speaking, the agent is quite sensitive to the values of parameters. Large TAU works really well in this example.

2.1 DDPG with network weights updating every multiple steps

In folder: min_as_settings

Tennis_min.ipynb + ddpg_agent.py + model.py:

Train the less intelligent agent to achieve mean score >= 0.5 with 2 actors and 1 critic

I. Hyper-parameters

```
BUFFER_SIZE = int(1e6)    # replay buffer size
BATCH_SIZE = 2**10              # minibatch size
GAMMA = 0.99                    # discount factor

# the large soft updating rate of target parameters matters much
TAU = 3e-1                      # for soft update of target parameters,from 1e-3 to 3e-1
OPPO_TAU = 1e-1            # To turn down the function 'learning from the opponent', place 0 here
LR_ACTOR = 2e-4                 # learning rate of the actor
LR_CRITIC = 2e-4               # learning rate of the critic
WEIGHT_DECAY = 0.000    # L2 weight decay

# 4 is bad in this case
# the network weights in both actor and critic cases are updated every # time steps
net_update_every = 5
n_episodes = 1000
max_t = 1000
```

II. Model Structure

Actor:

```
state_size-->fc1-->ReLU-->fc2-->ReLU-->fc3-->tanh-->action_size
fc1_units=400
fc2_units=300
```

Critic:

```
state_size-->fc1-->ReLU-->fc2-->ReLU-->fc3--> state-value function (dim = 1)
                  action_size-->
```

2.2 DDPG with network weights updating every single step

In folder: copy_param_ref

Tennis_v6.10_original.ipynb + ddpg_agent.py + model.py:

Train the more intelligent agent to achieve mean score >= 0.5 with 2 actors and 1 critic

Some parameters are referred from:
https://github.com/marcelloaborges/Tennis-Collaboration-Continuous-Control/blob/master/Tennis.ipynb

The training speed is improved rapidly and the problem is solved in only several hundred episodes.

I. Hyper-parameters

BUFFER_SIZE = int(1e6) # replay buffer size
BATCH_SIZE = 2**7 # minibatch size
GAMMA = 0.99 # discount factor
TAU = 2e-1 # for soft update of target parameters
OPPO_TAU = 0 # To turn down the function 'learning from the opponent', place 0 here
LR_ACTOR = 1e-4 # learning rate of the actor
LR_CRITIC = 4e-4 # learning rate of the critic
WEIGHT_DECAY = 0 # L2 weight decay
net_update_every = 1

II.Model Structure

Actor:

state_size-->BN0-->fc1-->BN1-->ReLU-->fc2-->BN2-->ReLU-->fc3-->tanh-->action_size
fc1_units=512
fc2_units=256

Critic:

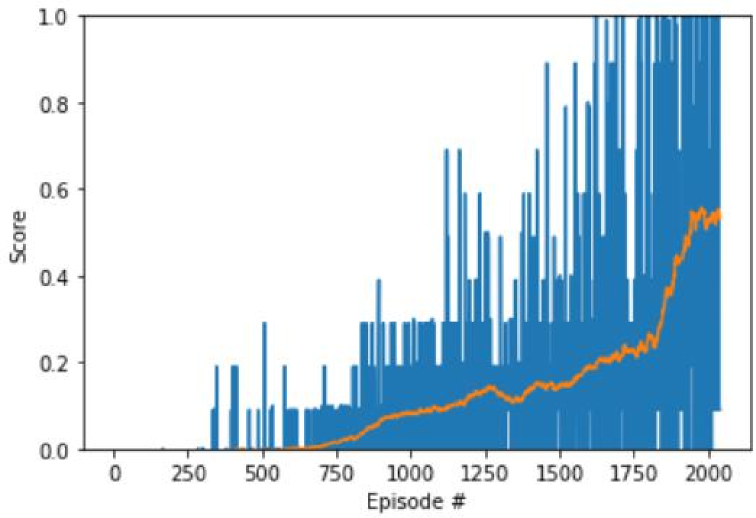
state_size-->BN0-->fc1-->BN1-->ReLU-->fc2-->BN2-->ReLU-->fc3-->state-value function
action_size-->

3. Results

3.1 **DDPG with network weights updating every multiple steps**

The setting takes around 1939 episodes to achieve satisfying results.

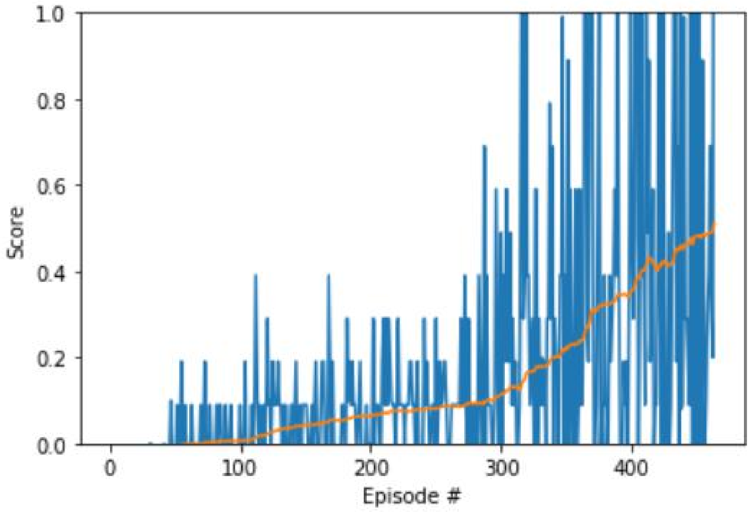
```
Episode 100	Average Score: -0.01
Episode 200	Average Score: -0.01
Episode 300	Average Score: -0.01
Episode 400	Average Score: -0.00
Episode 500	Average Score: -0.00
Episode 600	Average Score: 0.000
Episode 700	Average Score: 0.000
Episode 800	Average Score: 0.03
Episode 900	Average Score: 0.07
Episode 1000	Average Score: 0.08
Episode 1100	Average Score: 0.09
Episode 1200	Average Score: 0.12
Episode 1300	Average Score: 0.13
Episode 1400	Average Score: 0.14
Episode 1500	Average Score: 0.14
Episode 1600	Average Score: 0.19
Episode 1700	Average Score: 0.21
Episode 1800	Average Score: 0.26
Episode 1900	Average Score: 0.43
Episode 1939	Average Score: 0.50score = 0.5 achieved by the weaker agent by episode: 1939
Episode 2000	Average Score: 0.52
Episode 2040	Average Score: 0.53
```



3.2 DDPG with network weights updating every single step

The setting takes only 463 episodes to achieve satisfying results.

```
Episode 100     Average Score: 0.01
Episode 200     Average Score: 0.06
Episode 300     Average Score: 0.12
Episode 400     Average Score: 0.35
Episode 463     Average Score: 0.51
```



Validation:

Score (max over agents) from episode 1: 2.600000038743019

Score (max over agents) from episode 2: 2.600000038743019

Score (max over agents) from episode 3: 2.600000038743019

Score (max over agents) from episode 4: 2.7000000402331352

4. Future Work

There is still much to do. I am working on the version based on the proposed paper. The actors only receives local information and the critics use global information during training except rewards. In addition, the approach mentioned by Udacity to train the agent with single shared actor and critic is also interesting. I will keep updating the repository.