Collaboration and Competition

Introduction

The aim of this project was to implement and train agent/s to play table tennis. The two agents were provided control of rackets to bounce a ball over a net. A reward of +0.1 is given to an agent if it is able to hit the ball over the net. A penalty point of -0.01 is given when the ball hits the ground or his the ball out of the play area.

This is an episodic problem where the goal is for the agent to continuously keep the ball in the air. The observation space contains the position and velocity of the ball and racket.

To solve this problem, A multi-agent version of the Deep Deterministic Policy Gradient (DDPG) was applied. This is a model-free algorithm based on deterministic policy gradient that can operate over continuous action spaces Lillicrap et al. (2016). The code was adapted from Udacity's ddpg-bipedal exercise.

Network Architecture

The DDPG algorithm was adapted for a multi-agent version by using a shared observation and action space and feeding it into the learning function. To adapt to the shared observation and action space, the state size in the model was doubled. The units for the actor and critic model were reduced from 400 and 300 to 256 and 128. Additionally, the algorithm used clipped gradients but removed batch normalised from the network architecture. The code was adapted from my last project with continuous control.

HYPER PARAMETERS

Hyper parameters such as the soft update of target parameters, batch size, noise and time steps before updating the network were instrumental in achieving the required score for the project. The TAU parameters assisted in a larger factor in updating parameters and the batch size increase from 128 to 256 resulted in a smoother learning experience.

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 256        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-2              # for soft update of target parameters
LR_ACTOR = 1e-3         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0.0      # L2 weight decay

n_timestepupdate = 6    # time steps before update to network
n_updatelearn = 6       # number of times to update training
```

Optimising noise was the key factor to reaching the score with the score. Similarly, Lillicrap et al. (2016) used altered the noise process to enhance the exploration because of the very different time scales involved in certain environments. This may be due to the observation space and the environment units.

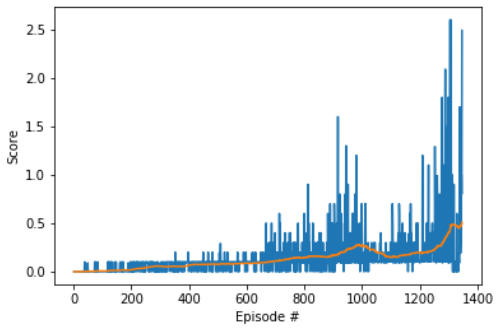
```
# parameters for noise
init_noise = 1          # initial noise level
noise_decay = 9999e-4   # weight decay
decay_step = 1          # decay after number of steps

# ddpg_agent.py lines 101 - 103
```

```
if timestep % decay_step == 0:
    self.eps *= noise_decay
actions += self.eps * self.noise.sample()
```

Results

The agent was able to reach a score of 0.5 over 1346 episodes and in 21 minutes.

```
Episode 100     Average Score: 0.0058   Time Cost per epi: 0.2584
Episode 200     Average Score: 0.0202   Time Cost per epi: 0.5235
Episode 300     Average Score: 0.0543   Time Cost per epi: 0.2639
Episode 400     Average Score: 0.0657   Time Cost per epi: 0.5328
Episode 500     Average Score: 0.0746   Time Cost per epi: 0.2623
Episode 600     Average Score: 0.0873   Time Cost per epi: 0.5303
Episode 700     Average Score: 0.1091   Time Cost per epi: 0.2619
Episode 800     Average Score: 0.1450   Time Cost per epi: 0.4547
Episode 900     Average Score: 0.1627   Time Cost per epi: 0.2674
Episode 1000    Average Score: 0.2735   Time Cost per epi: 0.4600
Episode 1100    Average Score: 0.1474   Time Cost per epi: 1.6596
Episode 1200    Average Score: 0.1948   Time Cost per epi: 0.4620
Episode 1300    Average Score: 0.4034   Time Cost per epi: 0.46912
Episode 1346    Average Score: 0.5064   Time Cost per epi: 14.8088
Environment solved in 1346 episodes!    Average Score: 0.5064
Total time taken: 1303.95
```



Future

For future improvements in this exercise, trialing algorithms such as PPO, A3c and D4PG that use distributed training methods to gain experience would be a worthwhile learning. With the goal of attempting to reach the goal score in fewer episodes. The search space in an environment in the real work can be vast, by using advanced algorithms to solve the problem or by simplifying it we'll be able to achieve faster results.

References

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2016). Continuous Control with Deep Reinforcement Learning. [online] Available at: https://arxiv.org/abs/1509.02971 [Accessed 9 Nov. 2018].