Project 3: Collaboration Competition

---

1 Introduction

For this project, I have trained an Agent using the Multi Agent Deep Deterministic Policy Gradient algorithm which is based on the Actor Critic Methods. For 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.

2 Dependencies

1. Download the environment from one of the links below. You need only select the environment that matches your operating system:
   - Linux: click here
   - Mac OSX: click here
   - Windows (32-bit): click here
   - Windows (64-bit): click here

   (_For Windows users_) Check out this link if you need help with determining if your computer is running a 32-bit version or 64-bit version of the Windows operating system.

   (_For AWS_) If you'd like to train the agent on AWS (and have not enabled a virtual screen), then please use this link to obtain the "headless" version of the environment. You will **not** be able to watch the agent without enabling a virtual screen, but you will be able to train the agent. (To watch the agent, you should follow the instructions to enable a virtual screen), and then download the environment for the **Linux** operating system above.)

   env = UnityEnvironment(file_name="/data/Tennis_Linux_NoVis/Tennis")

3 How to Execute this solution

Follow the solution in the file **Tennis-v3.ipynb** to train the agent to pick up the Yellow bananas.

4 Implementation Approach

The Multi Agent Deep Deterministic Policy Gradient algorithm with Experience Replay is implemented. The details of the algorithm are given in this research paper.

https://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf

The following steps were taken to implement the algorithm

4.1 Define the class for the Ornstein-Uhlenbeck Noise

We define a class for the Ornstein - Uhlenbeck Noise. The Ornstein-Uhlenbeck process, or OU process, is a stochastic process that models mean-reverting behavior, meaning it tends to revert to a long-term mean, while also experiencing random fluctuations.
The OU Noise class has been defined as given in the class example.

[Reference:
https://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process#:~:text=In%20mathematics%2C%20the%20Ornstein%E2%80%93Uhlenbeck,Ornstein%20and%20George%20Eugene%20Uhlenbeck.]

4.2 Define the class for Experience Replay

I used the example given in the class to create Experience Replay which is the Replay Buffer for state transitions along with actions, rewards.

4.3 Define the Actor and the Critic Neural Networks

I define the Actor and the Critic Neural Networks with 128 neurons in the first hidden layer and 128 neurons in the second hidden layer, with batch normalizations and the dropout implemented.

The following hyperparameters were considered for the DQN Agent training.

BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 64 # minibatch size
GAMMA = 0.99 # discount factor

TAU = **1e-3** # for soft update of target parameters
UPDATE_EVERY = **1** # how often to update the network
UPDATE_COUNT = **1** # how many times to update the network each update step!
LR_ACTOR = **1e-4** # learning rate of the actor
LR_CRITIC = **3e-4** # learning rate of the critic
WEIGHT_DECAY = **0** # L2 weight decay
SEED = **12345**

4.4 Defining the Agent

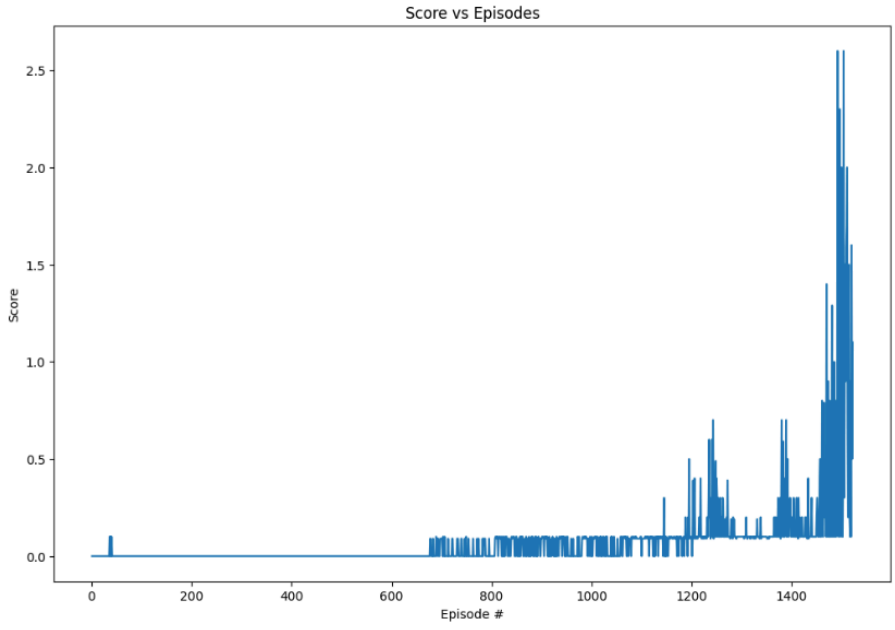
I have defined the agent as suggested in the MADDPG lab.

4.5 Main DDPG Function

Main MADDPG function (coded as DDPG) is defined as mentioned in the MARL class for the training of the algorithm.

4.6 Plot the Scores

Using the Matplotlib library, I plotted the scores. The score curve looks as below



5 Solution Details

1. Tennis-v3.ipynb : The main file that contains the OU Noise, Experience Replay Buffer, Actor and Critic Neural Networks, DDPG Agent.

2. checkpoint_actor.pth - Saved model weights for the Actor Neural network
3. checkpoint_critic.pth - Saved model weights for the Critic Neural Network
4. Readme.md
5. Report Continuous Control.pdf

6 Ideas for Future Work

I would like to explore the following as ideas for future work

a. Explore Hyperparameter tuning.
b. Explore Personalized Experience Replay.