Report for Udacity Reinforcement Learning Nanodegree - P3 Collab Compete

Project Setup and Learning Algorithm

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. The value for each action should be between -1 and +1.

After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores. The environment is considered solved, when the average (over 100 episodes) of the max score is at least +0.5.

The Deep Deterministic Policy Control (DDPG) has been used as the Reinforcement Learning Algorithm. DDPG agent uses an actor and a critic network to learn policies in high-dimensional, continuous action spaces. Here the critic is a value based learner and actor is a policy based learner. The actor network specifies an action based on the current state of the environment. The actor returns a deterministic actions which is the best action under the current state. The critic network calculates an advantage to evaluate the actions made by the actor.
My implementation of this project uses the Agent , OUNoise and the Replay Buffer class that I used in the Project 2 of this NanoDegree. To solve the multi agent problem, I initiated two different agents. Their actions are concatenated when passed into the environment. Also for each agent we save the next states and rewards corresponding to their action. The maximum of the score of two agents is considered as the score for this environment.

I found good success by using a relatively simple neural network that also happens to be faster to train. The actor neural network consists of 2 linear layers, the first with 400 neurons and the second with 300 neurons. The actor network takes as input the state size of 33 and the output layer has 4 neurons corresponding to the 4 actions. The Actor network uses RELU as the activation function for the linear layers. The final layer uses tanh as the activation function to squash outputs b/w -1 and +1.
The neural network used here is very similar to the one used for P2.

The Actor Network is shown below:

```
Actor network built: ModuleList(
  (0): Linear(in_features=33, out_features=400, bias=True)
  (1): Linear(in_features=400, out_features=300, bias=True)
  (2): Linear(in_features=300, out_features=4, bias=True)
)
```

The critic neural network consists of 2 linear layers, the first with 400 neurons and the second with 300 neurons. The action predictions from the action network are concatenated with the first layer of the critic neural network. The critic network takes as input the state size of 33 and the output layer has 1 neuron. The critic network uses RELU as the activation function for the linear layers.

```
,
Critic network built: ModuleList(
  (0): Linear(in_features=33, out_features=400, bias=True)
  (1): Linear(in_features=404, out_features=300, bias=True)
  (2): Linear(in_features=300, out_features=1, bias=True)
)
```

The mean square loss is used as the loss function to train the critic neural network. Adam is used as the optimizer with a learning rate of 1e-3. The actor network is also trained using the Adam Optimizer with a learning rate of 1e-3. Soft update with tau = 1e-2 is used to update the weights of the target actor and critic networks.

To do exploration, DDPG adds random noise to the action predictions from the actor network. After 300 episodes, add_noise is turned off and no exploration is done.

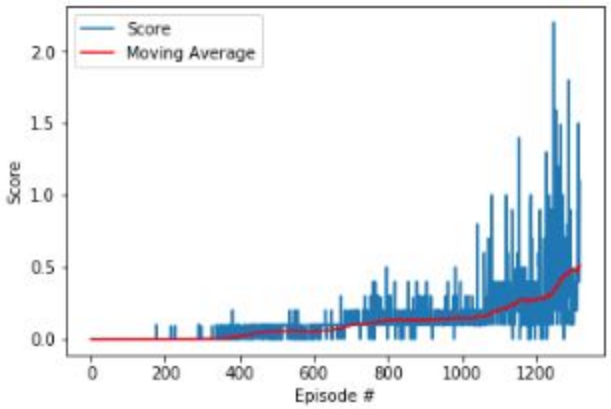
The state, action, reward, next state, done from each agent is stored in their replay memory. A batch size = 512 experiences are randomly picked up to train the neural network.

Training the Neural Network

The DDPG Network was trained for max of 2000 steps or till the average reward of 0.5 over 100 episodes was reached. It took the model 1316 episodes to reach an average reward > 0.5. The plot of score by epoch is shared below:

```
Episode 100     Best Score:  0.00      Moving average:  0.00
Episode 200     Best Score:  0.10      Moving average:  0.00
Episode 300     Best Score:  0.10      Moving average:  0.00
Episode 400     Best Score:  0.20      Moving average:  0.02
Episode 500     Best Score:  0.20      Moving average:  0.05
Episode 600     Best Score:  0.20      Moving average:  0.06
Episode 700     Best Score:  0.30      Moving average:  0.09
Episode 800     Best Score:  0.50      Moving average:  0.13
Episode 900     Best Score:  0.50      Moving average:  0.13
Episode 1000    Best Score:  0.50      Moving average:  0.15
Episode 1100    Best Score:  1.00      Moving average:  0.20
Episode 1200    Best Score:  1.40      Moving average:  0.27
Episode 1300    Best Score:  2.20      Moving average:  0.48

Environment solved in 1316 episodes!    Average Score: 0.51
```



Once training was finished, the best weights were saved. I then tested the trained agent on Unity Environment by playing games. Most games had a score > 0.5

Opportunities for Improvement

The performance of Multi Agent DDPG t can be further improved. Some of these improvements include:

1. Prioritized Experience Replay - The idea behind Prioritized Experience Replay is to sample events from the replay buffer based on importance probabilities instead of randomly. The intuition is that some past experiences that occur infrequently may be more important for learning. Past experiences with higher TD error delta could be given higher priority
2. Share information between the two agents - Current implementation trains the two agents independently. It will be good if Replay Buffer can be shared b/w them so they can learn from each others experience and train faster