Udacity Deep Reinforcement Learning Nanodegree
Project 2
Collaboration and Competition

In this project, we control 2 agents with rackets to attempt to pass a ball over the net. We use the Multi Agent Deep Deterministic Policy Gradient Algorithm to train the agents.

Tennis Environment

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 task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

- 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.
- This yields a single score for each episode.

The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.

DDPG Algorithm

DDPG is an actor critic method for reinforcement learning over continuous action spaces.
The actor component takes in a state vector S and outputs a vector of continuous actions A. The actor is a feedforward neural network with two copies, one being used as a target for the other one.
The critic component takes in a state vector S and vector A of continuous actions as input and outputs the Q value for the given state and action pair, i.e Q(s,a). It is implemented as a feedforward neural network with two copies, one being used as a target for the other one.
During training, in order to encourage exploration, we add Ornstein-Uhlenbeck noise to the agent at every step.

MADDPG Algorithm

MADDPG (multi-agent deep deterministic policy gradient) is the multi agent modification of the DDPG algorithm. In implementing MADDPG, I took inspiration from the

implementation created by OpenAI. Choosing the hyperparameters was non trivial and required a lot of trial and error.

All the agents share the same critic network during training, and use only the actor network during testing.

Network Architecture

Actor Network – Hidden Layer 1 : 256, Hidden Layer 2 : 256
Critic Network – Hidden Layer 1 : 256, Hidden Layer 2 : 256

We initialized the weights of all layers using Glorot Initialization scheme.

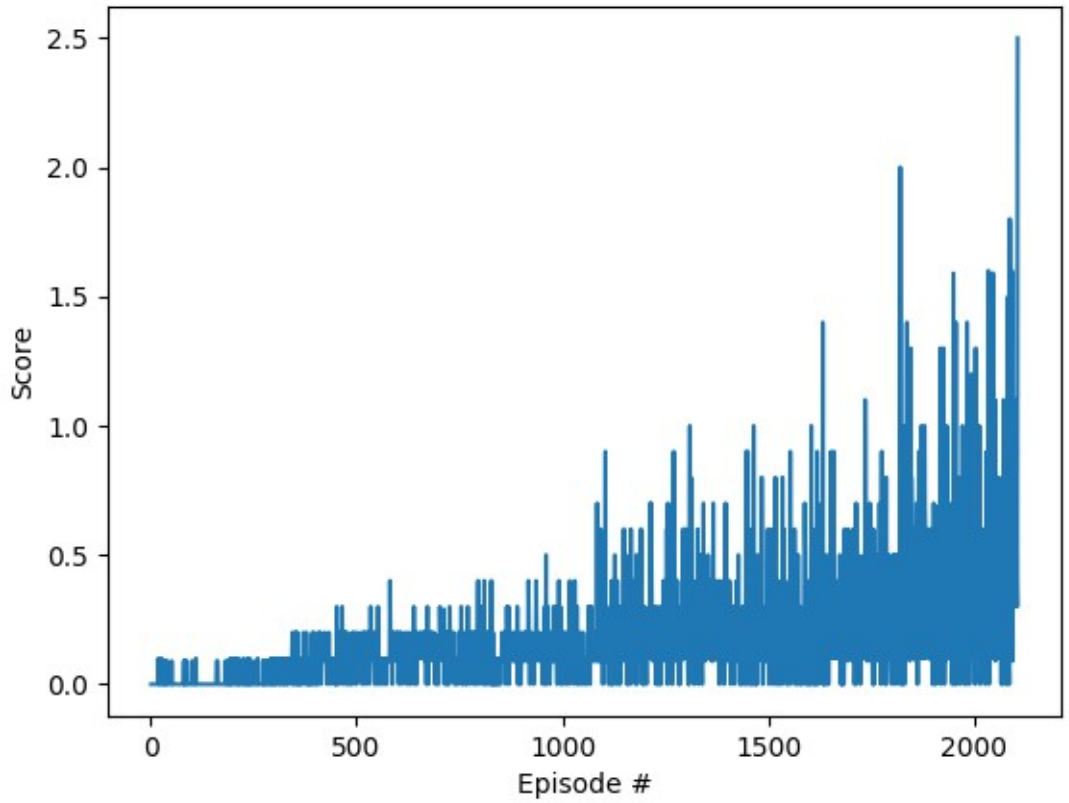
We use the following hyperparameters for training the agents -

```
# Parameters of the training
self.BUFFER_SIZE = int(1e6) # replay buffer size
self.BATCH_SIZE = 128 # minibatch size
self.GAMMA = 0.99 # discount factor
self.TAU = 0.2 # for soft update of target parameters
self.LR_ACTOR = 1e-4 # learning rate of the actor
self.LR_CRITIC = 1e-3 # learning rate of the critic
self.WEIGHT_DECAY = 0 # L2 weight decay
# no of updates per step
self.n_updates = 5
# track no of steps before update
self.step_no = 0
# updating after how many steps
self.update_rate = 1
```

For Ornstein-Uhlenbeck process, I used the default parameters, mu=0, theta=0.15, sigma=0.2

Results

DDPG solves the environment in around 128 episodes. The graph belows shows the average score plotted as a function of number of episodes. The file is saved as result.png.



The console output is logged in result.txt

Ideas for future work

As an intellectual curiosity, I wish to understand the role of random seeds in convergence of RL algorithms. (I faced a lot of issues with convergence with changing seed values).
We wish to explore the role of Batchnorm layers in convergence of MADDPG algorithm.
We also believe that Prioritzed Experience Replay would result in faster convergence.