# Udacity Deep Reinforcement Learning

# Project 3: Collaboration and Competition

Jihye Seo

March 24, 2020

**Introduction**

In this exercise, I have trained a pair of agents that plays tennis. The goal of the agent is to keep the ball in play. 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. The task is episodic, and in order to solve the environment, agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

We used DADDPG algorithms to train agents. While each agent had their own Actor network that provides policy for the next action, we trained critics with replay buffer data that includes experiences from other agents. Our agent had following environment.

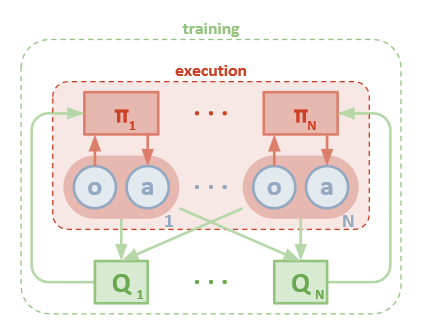
Unity Tennis Agent Environment:

* + Number of agents: 2
  + Size of action for each agent: 2, Value between [-1, 1]
  + Size of states for each agent: 24

I have started from the default setting from the last DDPG agent. With multiple wrapper functions that create and train actor and critic networks for each agents, I used MADDPG algorithms to train agents. I have tried different networks and frequency of learnings.

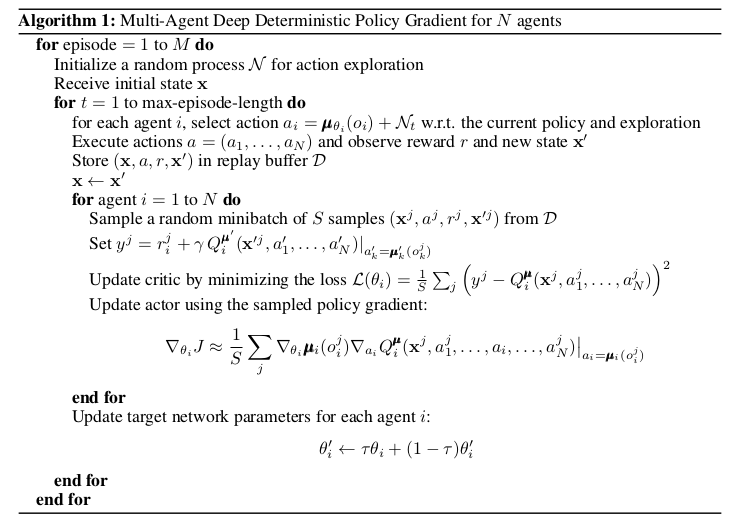
[**Multi-Agent Deep**](https://arxiv.org/pdf/1706.02275.pdf) **Deterministic Policy Gradient (MADDPG) Algorithm**

In this project I used MA[DDPG](https://arxiv.org/pdf/1509.02971.pdf) algorithm to train the actor and critic network. In previous project with DDPG, all agents were trained sharing experiences in single actor network and single critic. However, with MADDPG, each agent’s actor network was trained using its own observations, while each critic’s network was trained using the observations and actions from all the agents.



Ref: <https://arxiv.org/pdf/1706.02275.pdf>

The pseudo code for this algorithm is included below.



**Methods**

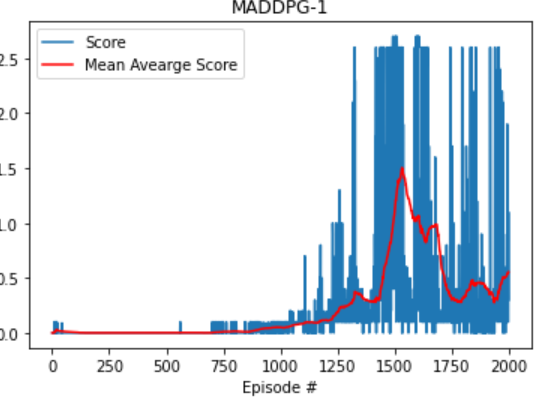
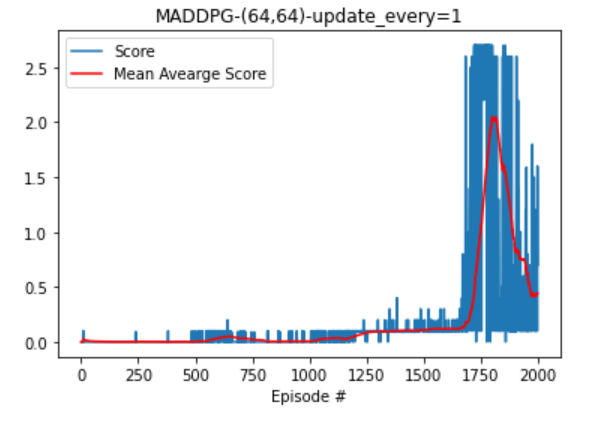
I tried the hyper parameter setting and Batch Normalization recommended in DDPG paper with little modifications based on last experiments. And tried different hidden network architecture, different learning numbers after experience replay addition.

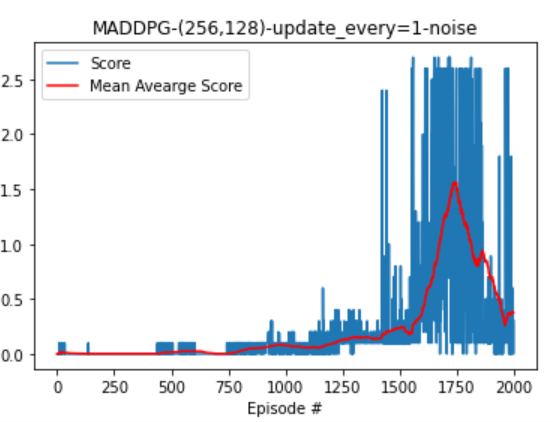
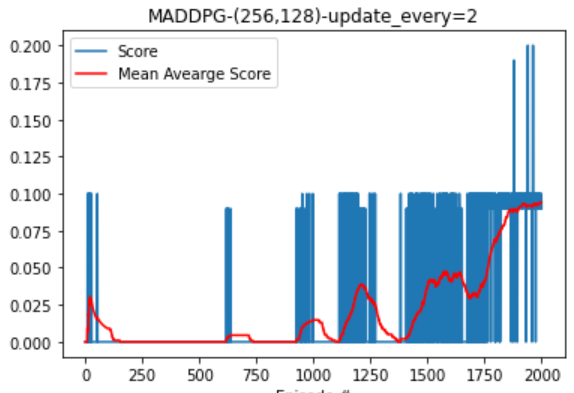
* + Batch norm: State input and all layers of Actor network, All layers before Critic
  + Hidden Layer network:1) (256,128) , 2) (64,64)
  + Initialization:
    - Uniform distribution between [-1/sqrt(f), 1/sqrt(f)), where f is fan-in of the layer.
    - Final layer of Actor [-3x10^-3, 3x10^3],
    - Final Layer of Critic [-3x10^-4, 3x10^4]
  + BUFFER\_SIZE = int(1e5) # replay buffer size
  + BATCH\_SIZE = 128 # minibatch size
  + GAMMA = 0.99 # discount factor
  + TAU = 1e-3 # for soft update of target parameters
  + LR\_ACTOR = 1e-4 # learning rate of the actor
  + LR\_CRITIC = 1e-3 # learning rate of the critic
  + WEIGHT\_DECAY = 0
  + UPDATE\_EVERY=1 or 2
  + OUnoise : scale=0.1, mu=0, theta=0.2, sigma=0.2

**Results**

I achieved goal around 1,350 episodes. Larger network and no Noise condition worked better for me. It took me more than 2 hours with AWS P2.xlarge instance so removing learning at the early stage and noise control such that the noise slows down as number of episode increase would be better.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Trial #** | **Hidden**  **Layer** | **Update**  **Every** | **Noise** | **Max Score** | **Mean**  **Score** | **First Episode goal Achieved** |
| **1** | **256,128** | **1** | **No** | **2.680** | **1.491** | **1350** |
| **2** | **64,64** | **1** | **No** | **2.724** | **2.034** | **1625** |
| **3** | **256,128** | **1** | **Yes** | **2.783** | **1.522** | **1514** |
| **4** | **256,128** | **2** | **No** | **0.201** | **0.0942** | **-** |

**Future Work**

Due to time limit I wasn’t able to finish up another experiments with other algorithm like PPO. Also I would like to see noise control effects on variances of the agents. By decreasing errors as episode increases, I may have able to see fast convergence and more stable behavior from our agents.