Report: Project 3

Deep Reinforced Learning Nanodegree: Collaboration and Competition with MA-DDPG

Introduction

This project solves for two agents to learn and play table-table tennis against each other. Implemented using MA-DDPG (Multi Agent Deep Deterministic Policy Gradient) algorithm. Each agent has two actor (local and target) and two critic (local and target) neural net models, and they share the replay buffer (experiences from the past - s a r s' d).

Environment

Tennis Unity Environment provided by Udacity and is used for this project.

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). 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.

State/Observation size : 24
Action space size : 2
Number of Agents : 2
Goal:
The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.

Learning Algorithm

MA-DDPG - Multi Agent Deep Deterministic Policy Gradient

MADDPG is an extension version of DDPG algorithm for multiple agents, an Actor-Critic algorithm that simultaneously learns a policy and an action-value function Q(s,a) for each agent. The actor takes a deterministic action following a policy and critic critics the actors action based on its action-value function.

Noise is added to the action intentionally to match up for a continuous action space use case through the output is deterministic.

Other helpful methods used to improve learning stability include:
Experience replay from memory, Local and Target networks for each Actor and Critic with a soft update feature, Gradient clipping at 1 to avoid parameter blow up, periodic learning, batch normalization.

Hyperparameters

GAMMA	0.99	Discount factor
TAU	2e-1	Soft update of target parameters
LR_ACTOR	7e-4	Learning rate of the actor
LR_CRITIC	7e-4	Learning rate of the actor
WEIGHT_DECAY	0	L2 weight decay
BATCH_SIZE	512	Batch size
BUFFER_SIZE	int(1e6)	Replay buffer size
learn_every	10	How often to update target network
num_learn	10	Number of updates at each step
OUNoise.theta	0.15	Ornstein-Uhlenbeck Noise parameter
OUNoise.mu	0.2	Ornstein-Uhlenbeck Noise parameter

Network Structure

Actor : NN (both agents)

a_FC1 : Linear (state_size - > 256)

a_FC1 : ReLU (Batch Normalization (a_FC1))

a_FC2 : Linear (a_FC1 -> 128)

a_FC2 : ReLU (Batch Normalization (a_FC2))

a_FC3 : tanh (128 (a_FC2) -> action_size)

Critic: NN (both agents)

c_FC1 : Linear (state_size - > 256)

c_FC1 : ReLU (c_FC1)

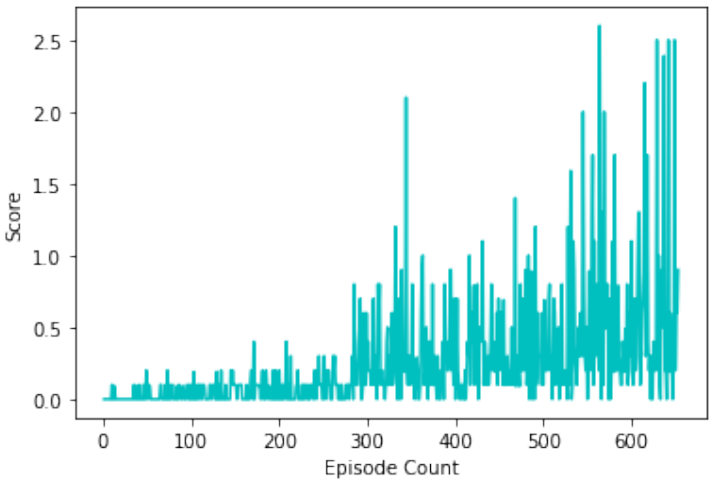
c_FC2 : ReLU (256 ([c_FC1 <concat> a_FC3]) 2 -> 128)

c_FC3 : Linear (128 (c_FC2) -> 1)

Result

The goal of reaching an average score > 0.5 over 100 consecutive episodes were reached in about 600 episodes. I kept it going till it reached an average score of 0.6.

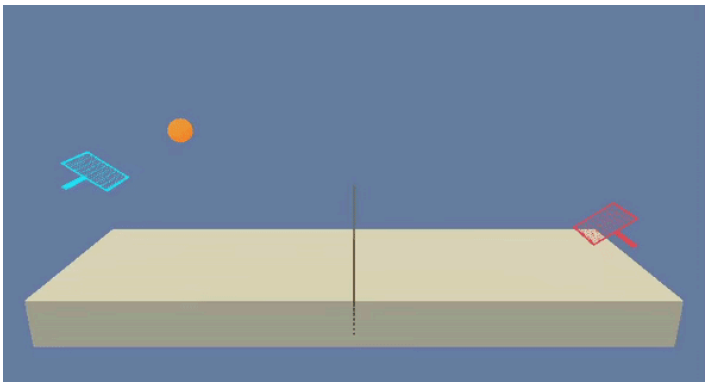
Score Vs Episode Plot



Result GIF

Untrained: https://media.giphy.com/media/Q4fut2cCiHnxBISXqQ/giphy.gif

Trained: https://media.giphy.com/media/RTbDwVdJddTIXxK7WR/giphy.gif



Future Work

- Try out the soccer environment that has 4 agents
- Use prioritized experience replay
- Use tensorboard to compare results form multiple hyper parameter choices
- Checkout out torch-summary