Project Report - Multi-Agent Collaboration & Competition

Introduction

Project Goal

This project aims to train two RL agents to play tennis in the Unity ML-Agents Tennis Environment. As in real tennis, the goal of each player is to keep the ball in play. When you have two equally matched opponents, you tend to see reasonably long exchanges where the players hit the ball back and forth over the net.

Project Overview

We'll work with an environment that is similar, but not identical to the Tennis environment on the Unity ML-Agents GitHub page.
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 moves 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 $+0.5$.

Algorithms

The algorithm used here is a Multi-Agent Deep Deterministic Policy Gradient (MADDPG). This algorithm has multiple competitive agents that are suitable for the game like a Tennis environment.

Model architecture

The MADDPG is composed of two networks, actor and critic. The details of these networks have been given below.

Actor Network

Layer	**Neurons**	**Activation**
Fully Connected Layer	($2 \times State\_Size$, 256)	ReLU
Fully Connected Layer	(256, 128)	ReLU
Output Layer	(128, $Action\_Size$)	TanH

Critic Network

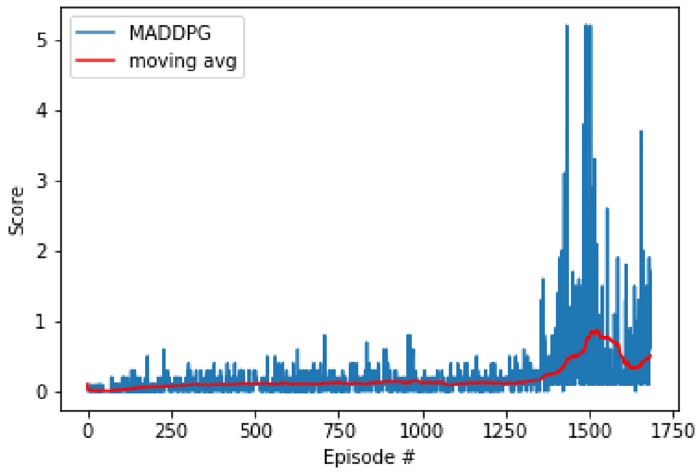
Layer	Neurons	Activation
Fully Connected Layer	$(2 \times State\_Size, 256)$	ReLU
Fully Connected Layer	$(256 + 2 \times Action\_Size, 128)$	ReLU
Output Layer	(128, 1)	Linear

Hyper-parameters

- $BUFFER\_SIZE = int(1e6)$
- $BATCH\_SIZE = 128$
- $LR\_ACTOR = 1e-3$
- $LR\_CRITIC = 1e-3$
- $WEIGHT\_DECAY = 0$
- $LEARN\_EVERY = 1$
- $LEARN\_NUM = 5$
- $GAMMA = 0.99$
- $TAU = 8e-3$
- $OU\_SIGMA = 0.2$
- $OU\_THETA = 0.15$
- $EPS\_START = 5.0$
- $EPS\_EP\_END = 300$
- $EPS\_FINAL = 0$

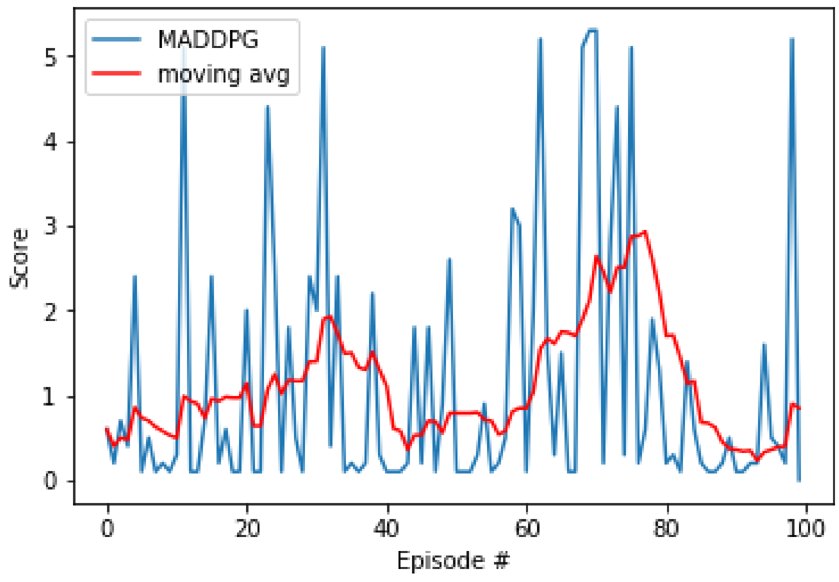
Results

Training



Training Log for the MADDPG algorithm

Inference



Inference score of 100 episodes for the MADDPG algorithm

Future Work

There are several future works we can pursue to improve the result.

- Address stability issues to produce more consistent results.
- Add prioritized experience replay.