Report

Introduction

This document contains a brief report of Unity Tennis problem.

Problem Description

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 goal is to reach score of +0.5 over 100 consecutive episodes for a single agent.

Method used

To solve this problem the MADDPG algorithm is used which is a multi-agent actor-critic method. The network architecture for actor and critic are as below.

Table 1 Actor Network

Layer	Activation Function	Size
Input	Relu	24
Fully connected 1	Relu	128
Fully connected 2	Relu	64
Fully connected 3	Relu	32
Fully connected 4	Tanh	16
Output	N/A	2

Table 2 Critic Network

Layer	Activation Function	Size
Input	Relu	28
Fully connected 1	Relu	128
Fully connected 2	Relu	64
Fully connected 3	Relu	32
Fully connected 4	N/A	16
Output	N/A	1

A stack of three consecutive observations is used as the environment state. For the actor network, the size of input layer and output put layer matches the state space size and action space size respectively.

For the critic network, the size of input matches the state space size plus the action size for each agent and the output layer has a size of 1.

Hyper-Parameters

The table below describes the hyper-parameters used.

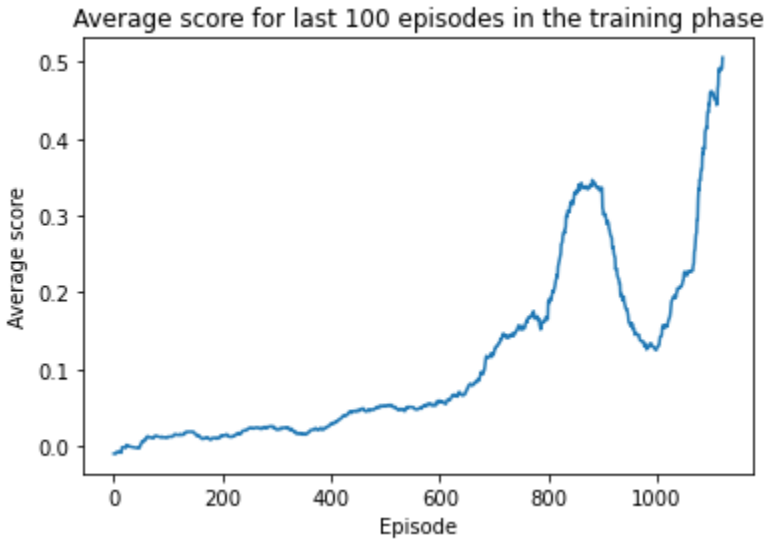
Table 3 Hyper parameters

Parameter	Value
Batch size	128
Learning Rate	0.0002
Gamma	0.99
Tau	0.001

Results

The following result shows the training progress for a random run of the code.

```
Episode 100       Average Score: 0.01
Episode 200       Average Score: 0.01
Episode 300       Average Score: 0.02
Episode 400       Average Score: 0.03
Episode 500       Average Score: 0.05
Episode 600       Average Score: 0.07
Episode 700       Average Score: 0.13
Episode 800       Average Score: 0.16
Episode 900       Average Score: 0.34
Episode 1000      Average Score: 0.12
Episode 1100      Average Score: 0.46
Episode 1122      Average Score: 0.51
Criteria reached after 1122 episodes
```



Ideas for future work

Although the current results show that the model is working well, there might be a few solutions to make the agent work even better or learn faster. I suggest trying the following ideas.

- Changing the network structure
    - Adding extra layers
    - Change layers size
- Try changing the hyper-parameters
- Using models and replay buffer for both agents
- Using prioritized replay buffer