Project: Collaboration and Competition

Submitter:

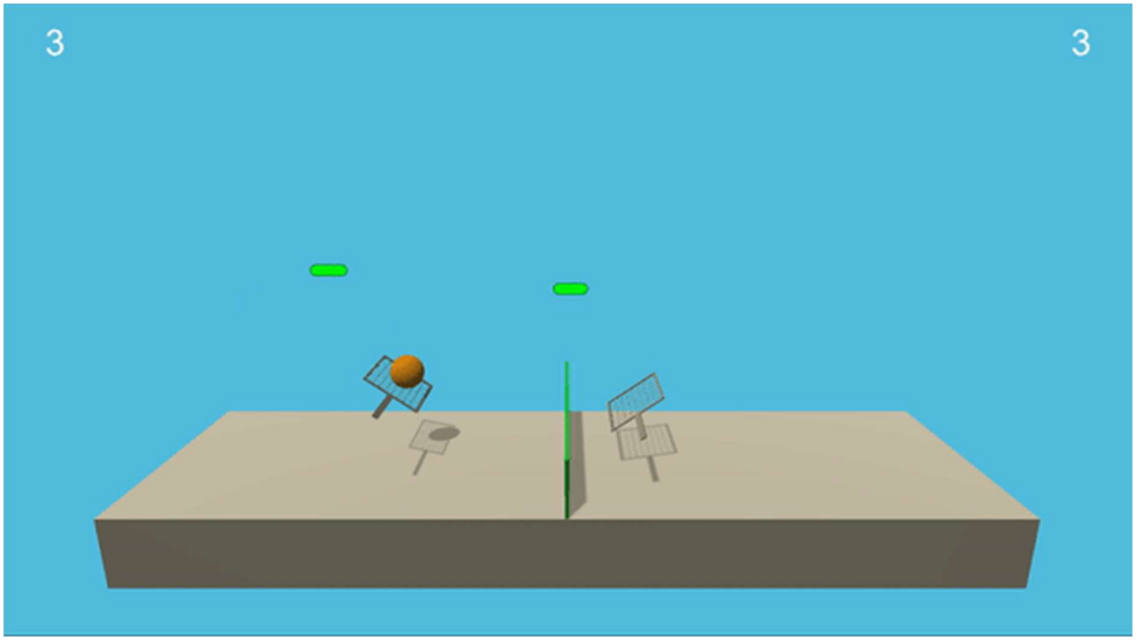
Monimoy Deb Purkayastha (monimoyd@yahoo.com | monimoyd@gmail.com)

Aim:

For fulfilment of Udacity Deep Reinforcement Learning Nano Degree

I. Project Overview

Problem Statement



In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, the agent 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.

Solving the environment

The task is episodic, and in order to solve the environment, the 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.

II. Solution approach

In this project I have used Multi Agent Deep Deterministic Policy Gradient (MADDPG) algorithm (https://arxiv.org/pdf/1706.02275.pdf) for training. This environment needs bot collaboration in certain scenarios as well as competition between agents in other scenarios. That is why extension of single agent by independent training them will not work in this case.

The primary principle used in MADDPG is that if we take actions of all the agents together, system is stationary even if policy changes.

MADDPG uses just like DDPG algorithm Actor and Critic principle. Critic of both the agents uses observation and actions from both the agents. But each of Agents actor uses its own observation independent of observation of the other agent. This allow both the agents to learn to collaborate in some scenarios and compete in other scenarios

MADDPG, just like DDPG uses experience replay where experience tuples (S,A,R,S`) are added to replay buffer and are randomly sampled from the replay buffer so that samples are not correlated.

MADDPG just like DDPG algorithm also uses separate target neural network for both Actor and Critic for each of the agent. As target values are determined for both the critic and actor networks, copy of both of these networks and soft update their weights are periodically updated to the respective target networks.

III. Methodology

In this project, I have used MADDPG algorithm. T uses both experience replay and a separate target Neural Network

For implementing MADDPG algorithm have used pytorch for implementing Neural Network.

For each of agent, there are four neural networks used

i. Local network for Actor
ii. Target network for Actor
iii. Local network for Critic
iv. Target network for Critic

For all the neural networks for each both the agents I have used 128x256x128 hidden layers.

I have used Adam optimizer and used the following hyper parameters:

Batch Size: 256
Discount Rate (gamma) : 0.995
Soft update of target parameters (Tau) : 0.001
Neural Network Initial Learning Rate of Actor: 0.0001
Neural Network Initial Learning Rate of Critic: 0.0002
Initial warmup episodes without learning: 400
Number of learning steps for environment step : 3
Noise Start rate: 1.0
Noise End rate: 0.1
Noise reduction rate: 0.999

To improve the learning I have used the following techniques

i. As the network tend to overfit after training for many episodes, I have implemented a method to reduce the learning rate by 0.7 if the mean over consecutive 100 episodes does not improve by 0.05 after 100 episodes. Once learning rate is changed, next time will be changed only after another 100 episodes. This approach helps in getting out of plateau region of learning curve. The same learning rate reduction is done both the actor and critic. Learning rate reduction mechanism is started only after initial 400 episodes

ii. Added initial warmup period of 400 episodes during, experience tuples are added only to the replay buffer and no learning happens. The learning starts only after initial warm up period

iii. Noise process Ornstein-Uhlenbeck with mean value of 0 and standard deviation (sigma) of 0.2 has been added to explore states.

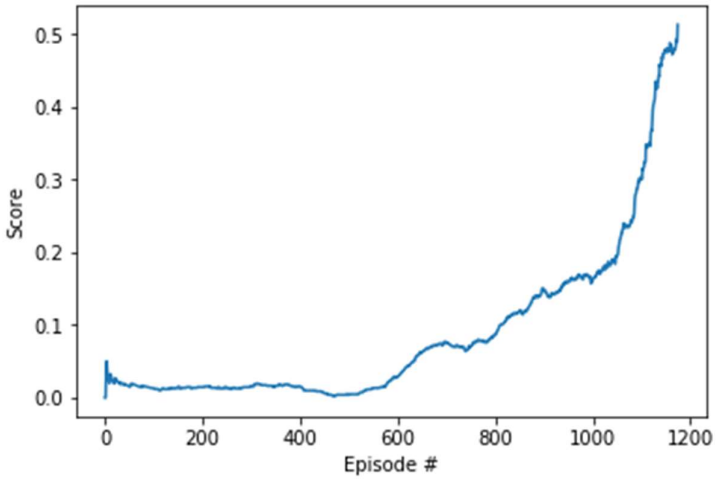
Also I gathered following metrics while performing training

i. **Score** : Mean of scores received by all the agents in each episode
ii. **Mean Score:** Mean score over last consecutive 100 episodes

IV. Results

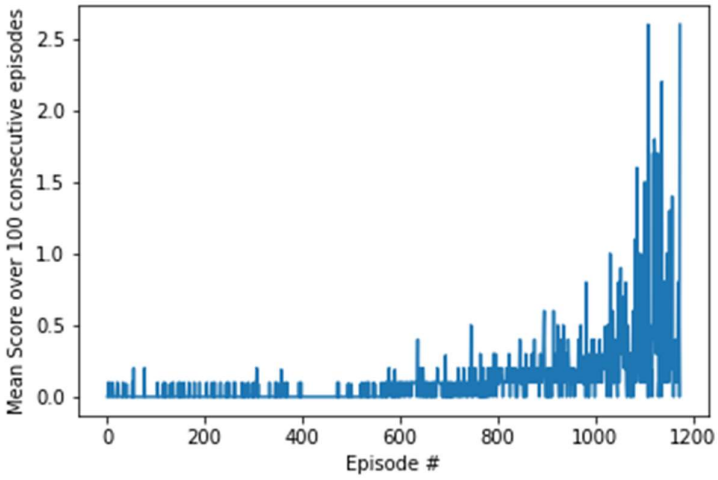
The environment is solved in **1174** episodes and achieved a score of **0.5131**

- **Plot of scores vs episodes**



Observation: The scores are initially flat but increases almost exponentially after 600 episodes. Initial flat scores can be explained as there is a warm up period of first 400 episodes when no learning happens but only

- **Plot for Average Score over 100 consecutive episodes vs episodes**



Observation: The mean scores over 100 episodes looks like right skewed histogram which is initially low but gradually increases after 600 episodes

V. Conclusion:

In this project I have used MADDPG algorithm with neural networks having hidden layer dimension of 128x256x128 for both Actor and Critic and for both the agents

i. I have trained the agent and achieved the desired mean score over 100 consecutive episodes of 0.5131 in 1174 episodes
ii. Scores (Rewards) earned are initially flat but increases exponentially after 600 episodes
iii. Mean score over 100 episodes is a right skewed histogram
iv. To improve Used various techniques like Learning rate reduction if the mean score over 100 episodes does not change by 0.05 for 100 episodes, also added initial warmup period of 400 episodes when no learning happens

Further Improvements

Further improvements to the project can be done using:

i. Prioritized Experience Replay
ii. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient (https://people.eecs.berkeley.edu/~russell/papers/aaai19-marl.pdf)