Deep Reinforcement Learning Collaboration and Competition Project

Project Overview

The goal of this project is to train 2 agents in the Unity Tennis environment to play tennis. 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, the agents must get an average score of +0.5 over 100 consecutive episodes, after taking the maximum over both agents.

Algorithms and Techniques

To train the Tennis agents I used MDDGP algorithm implementing centralized training and decentralized execution. Each agent receives its own, local observation. Thus it will be possible to simultaneously train both agents through self-play.

During training, the critic for each agent uses extra information like states observed and actions taken by all other agents. For the actor, there is one for each agent. Each actor has access to only its agents observations and actions. During execution time only actors are present and all observations and actions are used.

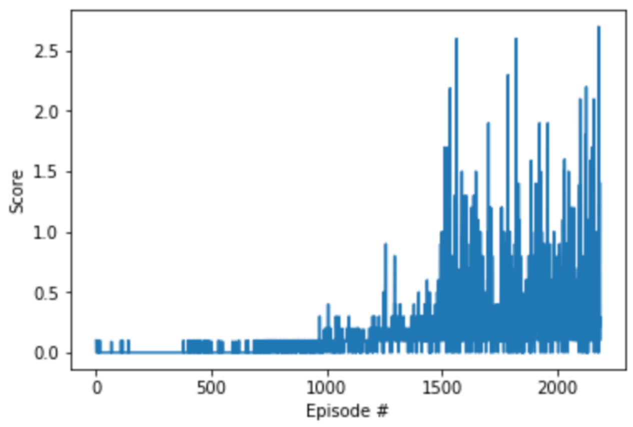
MADDPG (Multi-agent **DDPG**) class uses 2 **DDPG** agents similar to what was used in Udacity classroom for previous projects. Also **ReplayBuffer** is used as a shared buffer between agents. **MADDPG** combines **states, actions, rewards, next_states, dones** from both agents and adds them to shared **ReplayBuffer**. **MADDPG** act calls act for 2 **DDPG** agents.

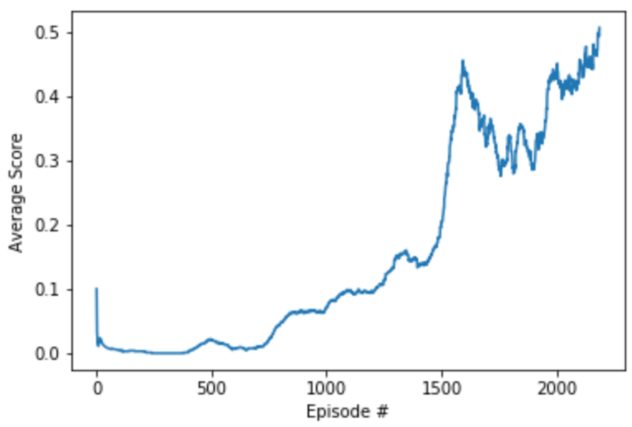
The Actor network consists of 3 linear layers. The first 2 linear layers are followed by Relu layer and the 3rd linear layer is followed by a Tanh layer as output layer. The sizes for the layers are 24*200, 200*150 and 150*2 respectively. The Critic has a similar structure with sizes of 52*200, 200*150 and 150*1.

Environment was solved in 2186 episodes and the network weights for 2 agents were saved as agent1_checkpoint_actor.pth, agent1_checkpoint_critic.pth, agent2_checkpoint_actor.pth and agent2_checkpoint_critic.pth. For different trials the number of episodes to solve the environment was usually around 2000.

Training results

The agents were trained in 2186 episodes with an average of 0.507 for last 100 episodes. The graphs for episodes scores and average scores are as follows:





Ideas for Future Work

During training I observed that changing hyper-parameters affects the way the training converge a lot but couldn't see a pattern how the changes improve or worsen the training. I'd like to change the parameters more systemically and observe how they affect the training.

Also I would like to try other methods and algorithms for multi-agent environments like multi-agent PPO or multi-agent DQN.