Solving the Multi-Agent Tennis Environment Through Self Play using a Deep Deterministic Policy Gradient Method

Jason Kutarnia

February 10, 2019

1 Learning Algorithm

This repository extends the Deep Deterministic Policy Gradient (DDPG) method to work in a multi-agent environment by utilizing the concept of self play. It is based on code provided in Udacity's DeepRL Nanodegree. Key features of this architecture include the definition of actor and critic networks as well as a replay memory which is uniformly sampled before each learning step. The introduction of the actor/critic architecture allows the DQN algorithm to solve continuous control problems. The actor network outputs the deterministic action the agent should take, while the critic network attempts to learn the value function. Noise is introduced to the actor network to improve convergence through increased exploration. The replay memory addresses the correlation problem between states occurring close together in time which keeps the network from reinforcing undesirable behaviors.

To solve the tennis environment, which consists of two computer controlled players volleying the ball back and forth, I trained a single DDPG agent to control both players using self play. During every step the experience of each player was added to the replay memory and subsequently a single actor network was used to output the next action for each of them. Sharing the actor/critic networks between players enables the agent to learn faster through self play. Parameters for the networks can be found in the *models.py* and *ddpg_agent.py* files.

2 Plot of Rewards

Figure 1 shows that the DDPG agent successfully solved the environment after 1669 episodes, which occurs when the agent's average performance over the last 100 episodes is at least 0.50.

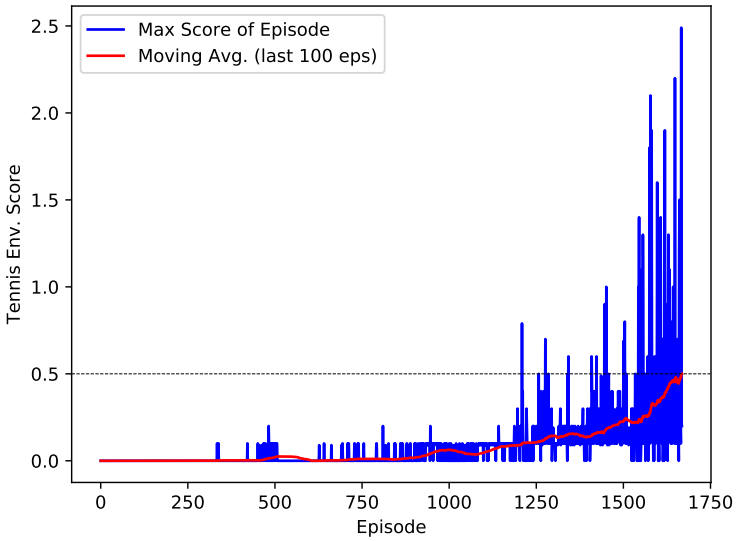

Figure 1: Plot of rewards for DDPG agent

3 Ideas for Future Work

In the paper "Benchmarking Deep Reinforcement Learning for Continuous Control" the authors discuss many additional algorithms which may be more effective for this task, including REINFORCE and Trust Region Policy Optimization. It would be interesting to adapt these approaches to the task of Multi-Agent Reinforcement Learning.