Project 3 - Collaboration and Competition from the Deep Reinforcement Learning Nanodegree of Udacity

October 24, 2019

1 Introduction

This is the report for Project 3 - Collaboration and Competition from the Deep Reinforcement Learning Nanodegree of Udacity. The task is to train two agents to 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 task is continuous, although the episodes end after a certain time or when an agent fails.

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.

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). More explicitly, after each episode, the rewards that each agent received (without discounting) are added up to get a score for each agent. This yields 2 (potentially different) scores; the maximum of these 2 scores is taken as the (single) score for each episode.

2 Learning Algorithm

At a high level the algorithm follows the usual approach of Reinforcement Learning, in which the agents interacts with the environment by choosing an action at every time step, where the environment is in a state; the environment in turn responds by providing rewards to the agents for this, and moving to the next state. The learning proceeds through a loss function, a reward function in this case, which the agents seeks to maximize by tweaking its internal parameters (weights), with the various techniques that we have seen in the course. The high level algorithm implemented in the notebook contains little more than a loop to implement this learning process.

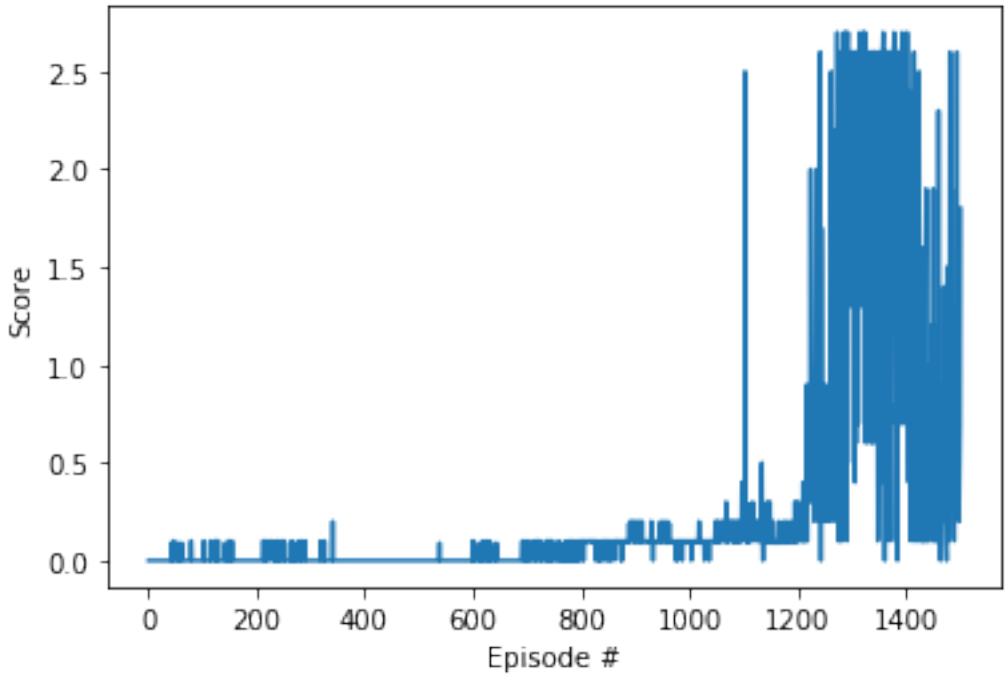
More insteresting is the learning process itself, the internal configuration of the agents provided (*ddpg_agent.py*) and the structure of the neural networks (*model.py*). We have used the same algorithm for both agents, based on the the modified Deep Deterministic Policy Gradient (DDPG) from Project 2, as implemented in the first file. This is an actor-critic architecture in which an actor network is used to approximate the policy function, and a critic is used for evaluating the policy function by using temporal differences (TD). This implementation of DDPG also has a replay buffer, and a noisy process (Ornstein-Uhlenbeck) is used to aid with the exploration of the observation space.

As mentioned, the underlying models for actor and critic have neural network architecture, in our case:

- an input layer
- a fully connected hidden layer with 128 units
- a relu activation function
- batch normalization
- a fully connected hidden layer with 128 units
- a relu activation function
- an output layer

Two of these networks (target and local) are used for both, actor and critic, with the updating of parameters proceeding via the soft update mechanism.

With this configuration the learning rate was already quite good, better than in the benchmark implementation, as can be seen in Figure 1.



although I have to admit that it took a fair bit of arbitrary tweaking of hyperparameters during the training of Project 2, for example I started with larger network sizes, learning rates, and shorter episodes. The full set of parameters for this final run is the one currently found in the files. This same set of parameters worked quite well for project 3.

3 Future

Since the learnign rate is already quite good, and it has already been quite difficult to get to this point, I am happy enough and have not experimented more with the configuration. I do realize that the possibility that was suggested in the clases was that the two agents shared the same critic network, this would allow also for collaboration and competition environments. However, frankly speaking, for this project the agents should be more concerned with the position and velocity of the ball, and less with that of the other agent, so I honestly do not believe that using that method will bring any benefit. I might use it for optional projects though.