Deep Reinforcement Learning – Project 3 : Collaboration and Competition

Melan Vijayaratnam

This document presents a technical description of the Collaboration and Competition project in the context of the Deep Reinforcement Learning Nanodegree from Udacity.

1 Summary

For this project, we will work with the Tennis environment.

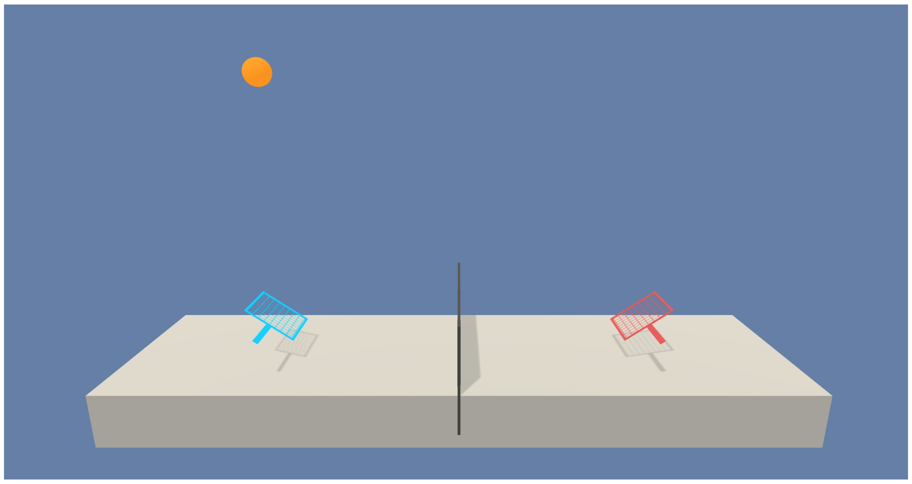

Figure 1: Unity ML-Agents Tennis environment

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 24 variables corresponding to position and velocity of ball and racket. Each action is a vector with two numbers, corresponding to movement toward or away from the net, and jumping. Every entry in the action vector should be a number between -1 and 1.

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. 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 signel **score** for each episode.

2 DDPG

2.1 Overview

DDPG (Lilicrap et al., 2015), short for **Deep Deterministic Policy Gradient**, is a model-free off-policy actor-critic algorithm, combining DPG with DQN. Recall that DQN (Deep Q-Network) stabilizes the learning of the Q-function by using experience replay and a frozen target network. The original DQN works in discrete space, and DDPG extends it to continuous space with the actor-critic framework while learning a deterministic policy.

In DDPG, we use 2 deep neural networks : one is the actor and the other is the critic:

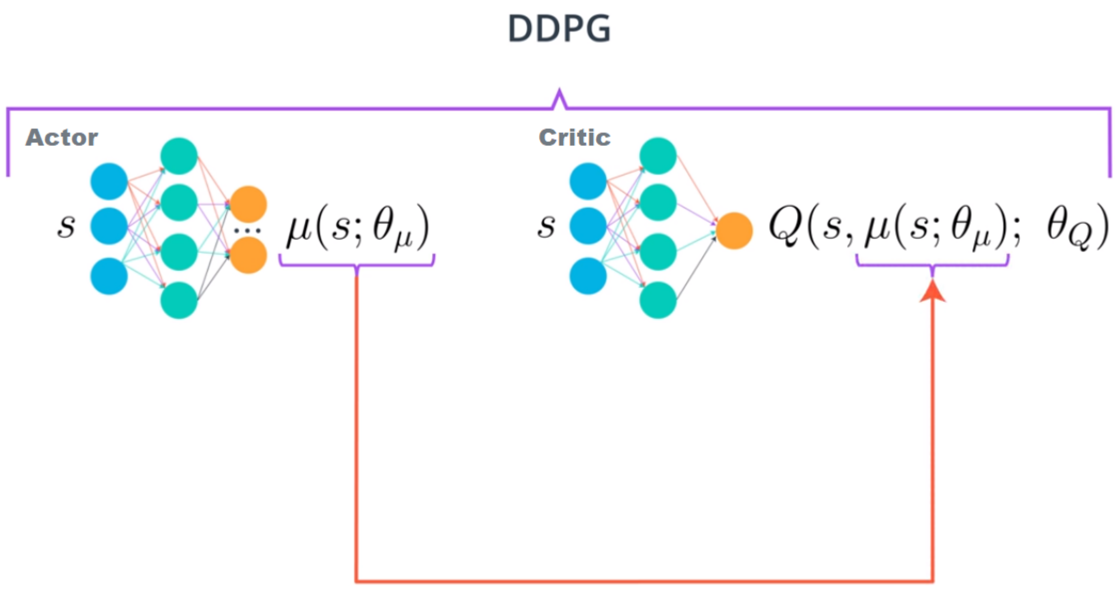

Figure 2: DDPG structure

2.2 Modification

The main components of DDPG have been explained in the previous report. In this report, we will focus on the modification made on the standard DDPG to solve the Tennis environment.

The main difference is that all agents share the same replay buffer memory. The main reason is that the agent may get caught in a tight loop of specific states that loop back through one another and detract from the agent exploring the full environment ”equally”. By sharing a common replay buffer, we ensure that we explore all the environment. Note that the agents still have their own actor and critic networks to train.

3 Implementation details

Because we have two models for each agent; the actor and critic that must be trained, it means that we have two set of weights that $\underline{\text{must be optimized separately}}$. Adam was used for the neural networks with a learning rate of 10^{-4} and 10^{-3} respectively for the actor and critic for each agent. * For $\mathcal{Q}$, I used a discount factor of $\gamma = 0.99$. For the soft target updates, the hyperparameter τ was set to 0.001.

The neural networks have 2 hidden layers with 250 and 100 units respectively. For the critic $\mathcal{Q}$, the actions were not included the 1st hidden layer of $\mathcal{Q}$. The final layer weights and biases of both the actor and critic were initialized from a uniform distribution $[-3 \times 10^{-3}, 3 \times 10^{-3}]$ and $[3 \times 10^{-4}; 3 \times 10^{-4}]$ to ensure that the initial outputs for the policy and value estimates were near zero. As for the layers, they were initialized from uniform distribution $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$ where f is the fan-in of the layer. We use the prelu[1] activation function for each layer.

[1]https://pytorch.org/docs/stable/nn.html#torch.nn.PReLU

4 What's next

Because of the symmetry of the environment, each agent mirrors the other one behavior. As such, there is no really such adversarial or collaborative relationship needed, thus the multi-agent reinforcement learning does not seem ideal in this setting.

A great challenge would be to tackle an environment in which the Multi-Agent Reinforcement Learning framework could apply, like the Soccer environment. With it, we can apply Multi-Agent DDPG (Lowe et al.) in which each of the actor takes a concatenation of all the states and actions of the other agents.