Project 2 Continous Control (Reacher Environment) Report

Udacity Deep Reinforcement Learning Nanodegree

Algorithm Used

This project uses the Deep Deterministic Policy Gradient (DDPG) algorithm to train two neural networks, an Actor and a Critic, to control 20 two-degree-of-freedom robotic arms – each following a moving goal position. The environment state space has 33 dimensions – position, rotation, velocity, angular velocities of an arm – and 4 actions corresponding to torque applicable to the two joints. Of course, each arm perceives its own version of these state and action vectors. However, because each arm follows the same rules (i.e., good state→action decisions for one arm apply to all arms), only one Actor/Critic pair is needed to control all arms.

Excellent performance was observed using identical neural network structures for the Actor and Critic. The 33-input environment state vector was input to a 400-element hidden layer, followed by a 300-element hidden layer, and finally a 4-element output layer to determine the actions to be taken. This network structure is depicted in Figure 1.

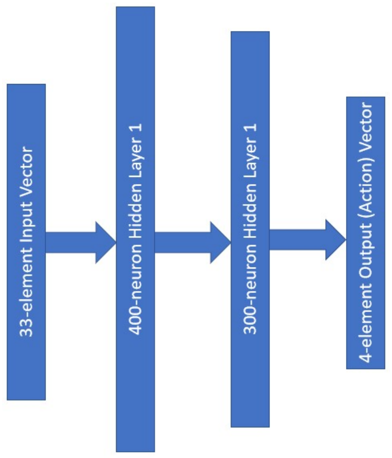
33-element Input Vector
400-neuron Hidden Layer 1
300-neuron Hidden Layer 1
4-element Output (Action) Vector

Figure 1. A representation of our Actor and Critic neural networks

These networks were trained over a maximum of 265 episodes while using soft updates with a tau value of 0.001 and minibatches of size 128, each helping to prevent action values from oscillating or diverging), and a modest discount factor of 0.99 seen to be useful in earlier projects.

Using these parameters, the environment was solved (i.e., achieved a score of at least +30 over 100 consecutive episodes over all 20 arms) in 205 episodes. A plot of the scores over these episodes is shown in Figure 2 below.

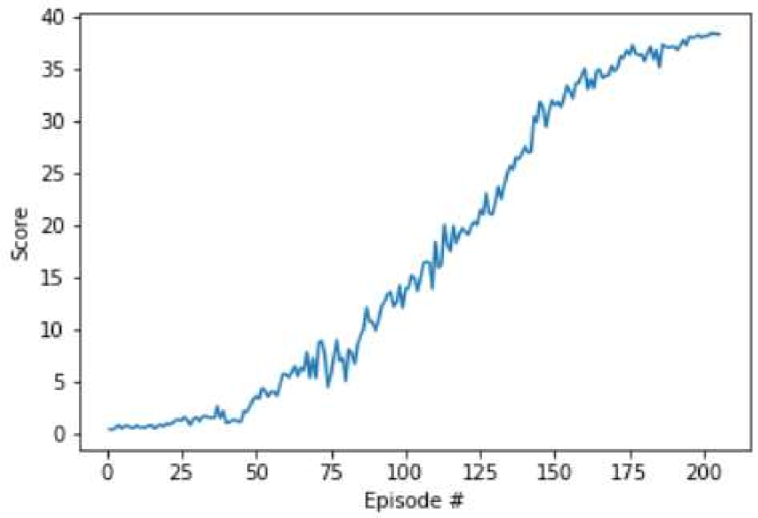

Figure 2. A plot of our 20-arm average score over training episodes

```
Episode 100    Average Score: 4.67   Score: 13.99
Episode 200    Average Score: 28.89  Score: 38.21
Episode 205    Average Score: 30.07  Score: 38.38
```

In this case, “Average Score” is the average of each arm over the course of 100 episodes.

Future Work

This algorithm would likely benefit from prioritizing experience replay. For example, the distance between an arm’s end effector at any time step and its goal position could be used to weight a priority queue in the ReplayBuffer class, or rather the (currently) random selection of such observations. In tandem with this prioritization, application of Trust Region Policy Optimization (TRPO) should be explored, which updates policies by taking the largest step possible to improve performace while satisfying a constraint in terms of KL-Divergence. In our case, robotic arms are often defined by Denavit-Hartenberg parameters. As such, according to the Jacobian matrix which defines relations between joint velocities and end effector velocities, a robotic arm can respond most quickly to any action when its joints are at 90°. For example, when a human is working with their hands, they can move their hands most effectively when their elbows are at a 90° angle, whereas a 180° angle has limited motion. Such kinematics should be considered when training our neural networks.