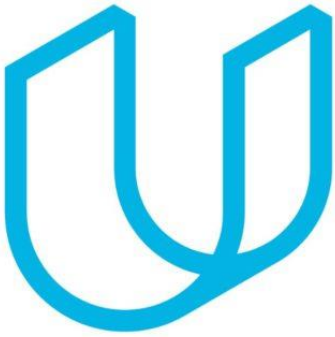

UDACITY

DEEP REINFORCEMENT LEARNING

POLICY-BASED METHODS

PROJECT: *CONTINUOUS CONTROL*

AUTHOR: ALBERTO GARCÍA GARCÍA

Introduction

The problem to be solved in this project consists in training a reinforcement learning agent controlling a double-jointed robotic arm. The robotic arm must reach a target location and stay within it, obtaining a positive reward of +0.1 for every step maintaining its position in the target location. The state space consists of 33 variables including information like the velocity or the rotation of the arm among others, while the action space consists of 4 variables representing the torque applicable to the arm joints. An image of such environment is presented in Fig. 1. The problem is considered to be solved when the agent's average score of the last 100 episodes is equal or greater than **30 points**.

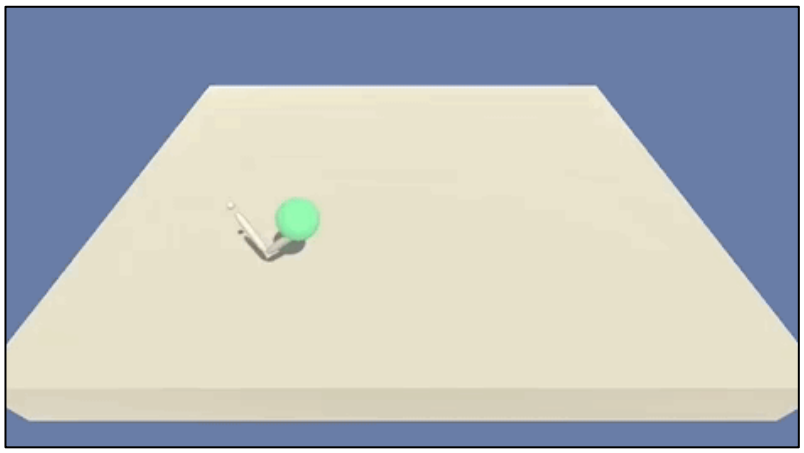

Fig. 1. Environment

This problem presents two different versions: training a single arm or training 20 of them. In this case the first option (just one arm) has been chosen. This document includes the details of the algorithm developed to train the agent, the conclusions obtained after the completion of the project and some future ideas to improve the current implementation.

Implementation

Since the problem environment presents a continuous action space, the Deep Deterministic Policy Gradient (**DDPG**) algorithm has been chosen to train the agent. DDPG is an off-policy reinforcement algorithm that combines elements of both policy-based and value-based methods, using an actor neural network to learn a deterministic policy and a critic neural network to approximate the value of the action (similar to the deep Q-network from Deep Q-Learning). Given that both of these targets change as the models learn from passed experiences (moving targets problem), this algorithm also includes target networks for both the actor and the critic in order to smooth the learning and increase stability. Furthermore, some noise is added to the actions during training to encourage some level of exploration. Thanks to these features, DDPG has been proved to be successful in tasks with continuous actions spaces.

In the solution provided, the actor and critic networks have been implemented using the PyTorch framework and present the following properties:

- The **actor network** receives a state as input and returns an action as output. It includes an input layer of 33 neurons (the state space size), five hidden layers of 128, 128, 64, 32 and 16 neurons respectively, and a final output layer of 4 neurons (the action space size). All the hidden layers include batch normalization and ReLU as activation function, while the output layer uses the hyperbolic tangent as activation function in order to keep the action values within the valid limits [-1, 1]. Adam has been the chosen optimizer, with a learning rate of 0.001.
- The **critic network** receives an action-state tuple as input and returns the estimated Q-value. It presents a shallower structure, with just one input layer of 33 neurons (again, the state space size, since the action tensor is concatenated after the first hidden layer activation), two hidden layers of 128 and 256 neurons respectively and a final output layer of one neuron (to provide the Q-value mapped to the state-action tuple). Once again, the hidden layers include batch normalization and ReLU as activation function, while the output layer does not use any activation function at all. In this network, Adam with a learning rate of 0.001 has also been chosen as optimizer.

These networks properties, along with an exhaustive fine-tuning of the rest of hyperparameters (the reward discount, the batch size, the τ parameter for soft updating the target networks, etc.) have been the key to solve the task. While in the second version of the problem there are 20 agents retrieving experiences in parallel and sharing the environment knowledge acquired, in the single-arm version there is just one agent exploring, which significantly hinders the learning process. Since only one agent is available in this version of the problem, such agent and its configuration need to be noticeably more complex than in the 20-agent version. In fact, it was not until more hidden layers were added to the actor network that the agent was able to achieve target scores (over 30 points).

Conclusion

The implementation described in the previous section is able to solve the problem in just **68 episodes** (averaging 30 points from episode 68 to 168), as shown in Fig. 2. Such performance demonstrates the effectiveness of the chosen algorithm.

Engaging with this project has been a great opportunity to explore the DDPG algorithm and its outstanding capabilities in tasks with continuous action spaces. Nevertheless, it is important to remark that the agent training has not been a trivial task and a careful fine-tuning of its hyperparameters has been required.

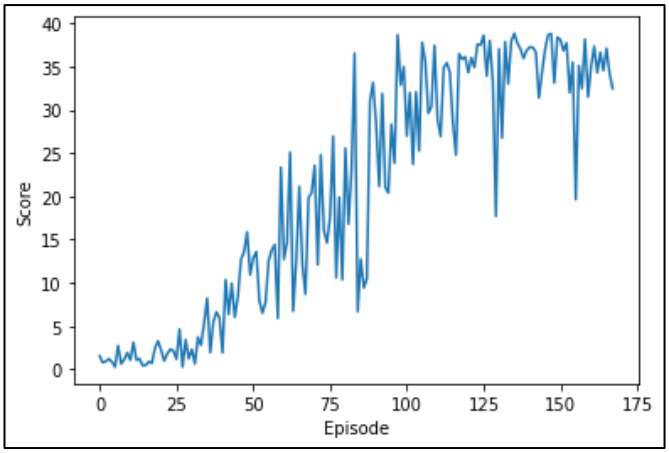

Fig. 2. Agent's score history

Future Work

Even though the agent has successfully learnt to solve the problem, there are some ideas that could have been explored to improve even more its current performance. The most straight forward extension that could be tried to increase the convergence of the algorithm would be adding prioritized experience replay, which would allow the agent to focus on experiences with higher impact. It could also be interesting to try other actor-critic methods or even policy-based methods (like Proximal Policy Optimization or Trust Region Policy Optimization) and compare their results with the DDPG algorithm.