Continuous Control Project
Deep Reinforcement Learning Nanodegree

Nontawat Pattanajak

11 January 2020

1. Introduction

This project aims to develop multiple agents to control (move) robot arms maintaining at the target location all the time. This task can be classified as a continuous environment. The reward will be given to the agent +1 every step that the robot arm is at the goal location. The environment space consists of 33 variables related to position, rotation, velocity, and angular velocities of the robot arms. The agents need to provide the values for four actions corresponding to joint torque, which they are ranging between -1 and 1 to maintain the robot arms being at the target position.

2. Algorithm

One of the most important conditions to consider in order to develop algorithm to control robot arm in this project is continuous action space. Even though Deep Q-Network (DQN) can solve problems with high dimensional observation space, it has limitation to handle continuous action space. This is because DQN finds the action by maximizing action-value function. When the action value can be any value, ranging between -1 and 1 for example, it requires optimization process to find an action which can maximize reward. If we still force to use DQN to solve this task by discretizing action space, it may cause a dimensionality issue. This will result in requiring high computational expense. Deep Deterministic Policy Gradient (DDPG) [1] which is actor-critic method, is considered to implement in this task. DDPG learns a policy that approximates optimal action. Therefore, it is suitable for solving the problem in continuous action space.

DDPG contains similar techniques to DQN such as Reply Buffer and Target Network. The agent of DDPG collects experiences from environment online and store the experiences to Reply Buffer. Then Reply Buffer generates batch data (in sampled uniformly random) to feed to Artificial Neural Networks which maps state to action. However, DDPG does not use argmax to provide a selected action, but it uses policy function.

The policy of DDPG is deterministic. For the untrained policy, the action is not accurate enough and the policy will not explore on-policy. The agent might select the same action all the time. To solve this problem, Gaussian noise is injected to the action.

3. Methodology

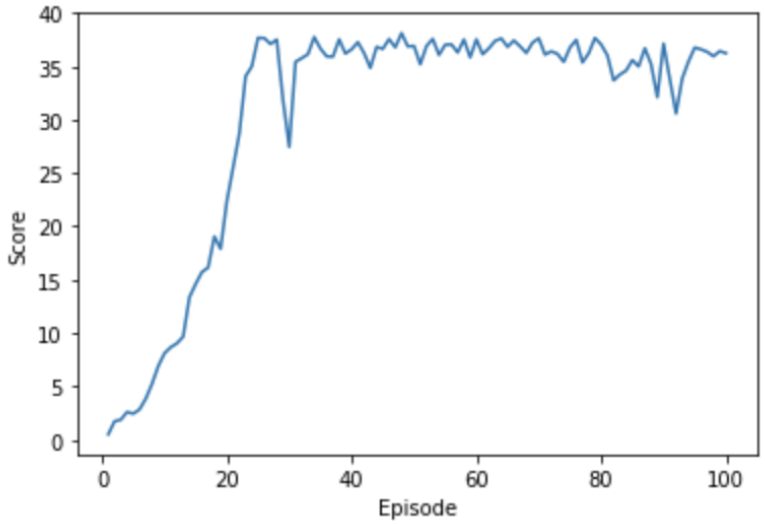
In order to develop agent's brain to solve this task, multi agents (20) are chosen. All agents are designed to have the same architecture. This is because they perform the same task. The Hyperparameters are shown in the list below.

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 128        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-3         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay

OU_SIGMA = 0.2          # Ornstein-Uhlenbeck noise parameter
OU_THETA = 0.15         # Ornstein-Uhlenbeck noise parameter
EPSILON = 1.0           # explore/exploit noise process added to act step
EPSILON_DECAY = 1e-6    # decay rate for noise process
```

4. Result

Training the agents to solve this task can achieve nearly 30 episodes with having score more than 30. The agents continue to be trained until episode 100 and they mostly have score more than 30, which is shown below.



```
Episode 10     Average Score: 3.62
Episode 20     Average Score: 14.66
Episode 30     Average Score: 33.24
Episode 40     Average Score: 36.38
Episode 50     Average Score: 36.80
Episode 60     Average Score: 36.69
Episode 70     Average Score: 37.00
Episode 80     Average Score: 36.46
Episode 90     Average Score: 35.02
Episode 100    Average Score: 35.20
```

5. Conclusion and Suggested Future Works

The agents in this project is developed based on DDPG. The architecture of algorithm consists of Artificial Neural Networks (ANN), Reply Buffer, Ornstein-Uhlenbeck Noise. The agents learn experience from environment and store in Buffer Reply. Then, Buffer Reply produces batch data and feeds to ANN. ANN learns the data and provides action. However, DDPG is a deterministic policy. In the early episode, the action provided by DDPG will not accurate enough. Adding Ornstein-Uhlenbeck Noise to the action can improve accuracy.

The ANN structure two hidden layers. Each layer contains 128 nodes. Batch Normalization is added after hidden layer one to improve the accuracy.

The result of training the agents in this project found that the agents can achieve score more than 30 in nearly 30 episodes and the score mostly remains more than 30 until finishing the training in 100 episodes.

There are still many ways to improve the accuracy by adding some techniques such as adding prioritized experience replay, or by using different algorithms such as Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), and Distributed Distributional Deterministic Policy Gradients (D4PG).

References

[1] T. Lillicrap et al, Continuous control with deep reinforcement learning, https://arxiv.org/pdf/1509.02971.pdf