Project2: Continuous Control

Mengheng Xue

July 2, 2019

1 Introduction

In this report, I will discuss the learning algorithm, along with the chosen hyper-parameters, and my model architectures for neural network I used to solve the continuous control problem.

2 Learning Algorithm

To solve this continuous control problem, I used Deep Deterministic Policy Gradients (DDPG) algorithm. DDPG is an approximate Actor-Critic Method, which extends DQN to work in continuous spaces. So the agent is composed of two neural networks, one as actor and the other as critic. Similar network architectures are used for both actors and critic which will be discussed in the following section.

2.1 Hyper-parameter Settings

- $BUFFER\_SIZE = 100000$. It defines the replay buffer size. Buffer is an object that contains tuples called experiences composed by *state, actions, rewards, next states* and *dones*, which is necessary information for learning.
- $BATCH\_SIZE = 128$. When the number of experiences in the replay buffer exceeds the batch size, the learning method is called.
- $TAU = 0.001$. It controls the model soft updates which is used for slowly changing the target networks parameters slowly, improving stability.
- $LR\_ACTOR = 0.00015$ and $LR\_CRITIC = 0.00015$. They are optimizer learning rates which control the gradient ascent step.
- $WEIGHT\_DECAY = 0.0001$. It is the L_2 regularization parameter of the optimizer.

2.2 Neural Network Architecture

2.2.1 Actor

Neural network for actor is consisted of three hidden layers with nodes $[600, 400, 200]$. The activation function for each hidden layer is ReLU, which will accelerate the training process. The activation function for output layer is a Tanh, which will make possible tackling continuous action spaces. This network takes the states as inputs and outputs the best calculated action for the input states.

2.2.2 Critic

For Critic, neural net structure is consisted of two hidden layers with nodes $[400, 300]$. Similar as Actor, each hidden layer using ReLU activation function. This network takes the states as inputs and outputs the action value function for the best action outputted by the Action network.

3 Performance

We can see that our model can solve the this problem in 186 episodes and achieve average score 30.23.

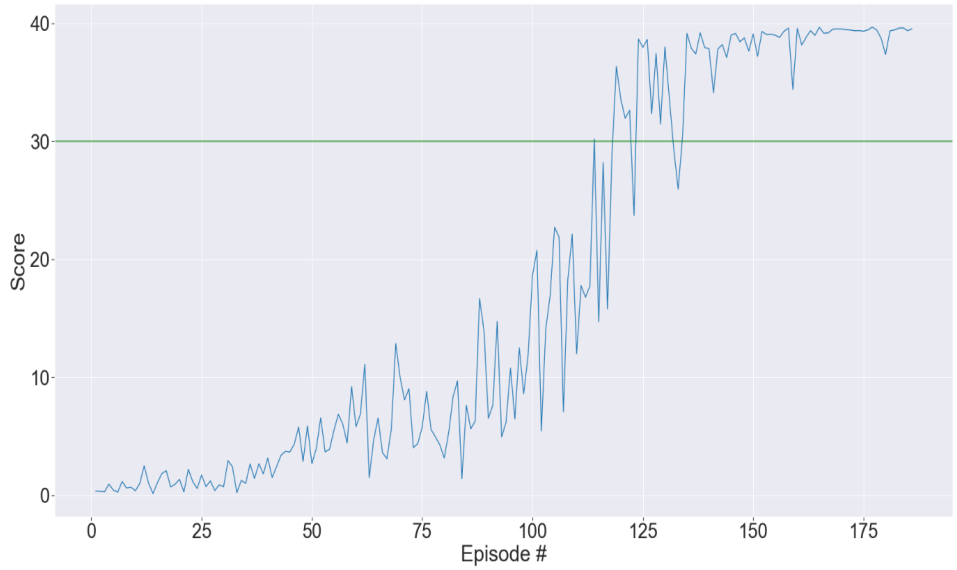

Fig. 1: Rewards per episode of our final model

4 Future Work

The advantages of policy-based method allows reinforcement learning algorithms to tackle continuous problems, approximating the real world. This motivates to create devices such as the physical double jointed arm mentioned in "The Environment - Real World", it would be interesting to create mechanisms that can, initially, operate in environments like project 1. Also, studying more about reward function designs and how to create Unity ml-agents simulations.

Regarding this project, besides solving the 20 arms and the Crawler environment, there are many improvements that can and should be implemented, prioritized experience replay is one of these. Also the D4PG algorithm, that showed state of art results should be used and compared.