Project: Continuous Control

Name: Zain Us Sami Ahmed Ansari

Introduction

For this project, I trained an agent to work with the reacher environment.

In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of my agent is to maintain its position at the target location for as many time steps as possible.

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

For this project I worked with the Single agent Environment.

Implementation

In this exercise the parameters that are tuned are as follows.

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate of the actor
LR_CRITIC = 1e-4        # learning rate of the critic
WEIGHT_DECAY = 0.0000   # L2 weight decay
```

The model architecture is defined through set of following variables

```
Number of agents: 1
Size of each action: 4
There are 1 agents. Each observes a state with length: 33
```

The parameters that define the network are as follows

"""Actor (Policy) Model."""

```
"""Initialize parameters and build model.
Params
======
    state_size (33): Dimension of each state
    action_size (4): Dimension of each action
    seed (1): Random seed
    fc1_units (128): Number of nodes in first hidden layer
    fc2_units (64): Number of nodes in second hidden layer
“""
```

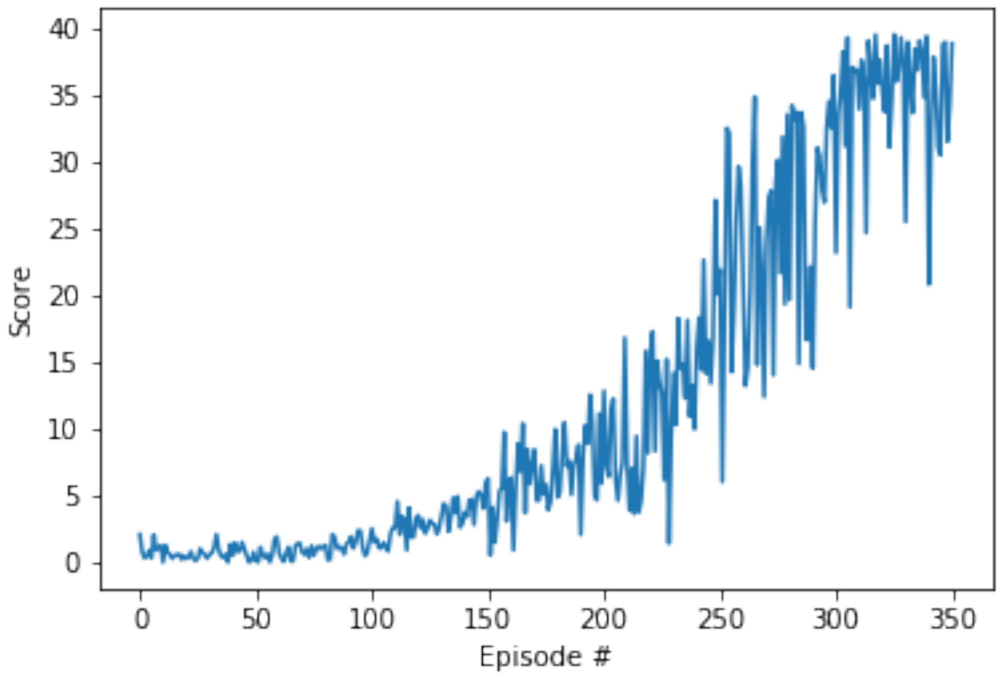
"""Critic (Value) Model.”""

```
    """Initialize parameters and build model.
    Params
    ======
        state_size (33): Dimension of each state
        action_size (4): Dimension of each action
        seed (1): Random seed
        fc1_units (128): Number of nodes in the first hidden layer
        fc2_units (128): Number of nodes in the second hidden layer
    “""
```

I have used two neural networks, the actor network and the critic network. The actor network contain 2 hidden layers of 128 and 64 units respectively. The output layer is a tanh activation layer.

The critic network has two hidden layers 128 and 128 units each.

Plot of Rewards



Future Work

As it is very evident from the plot of rewards that the standard deviation of this implementation is very high I want to try out different parameters and network architectures to reduce the standard deviation.

I would also want to try out algorithms like PPO, A3C and D4PG that use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience.

I would also like to try changing different actor, critic network sizes in DDPG.