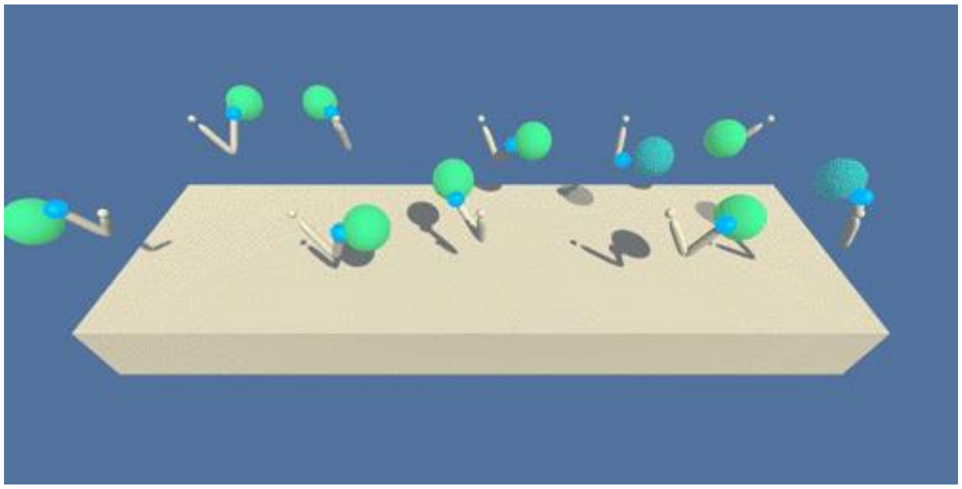
Project 2: Continuous Control

Introduction

For this project, you will work with the Reacher environment.



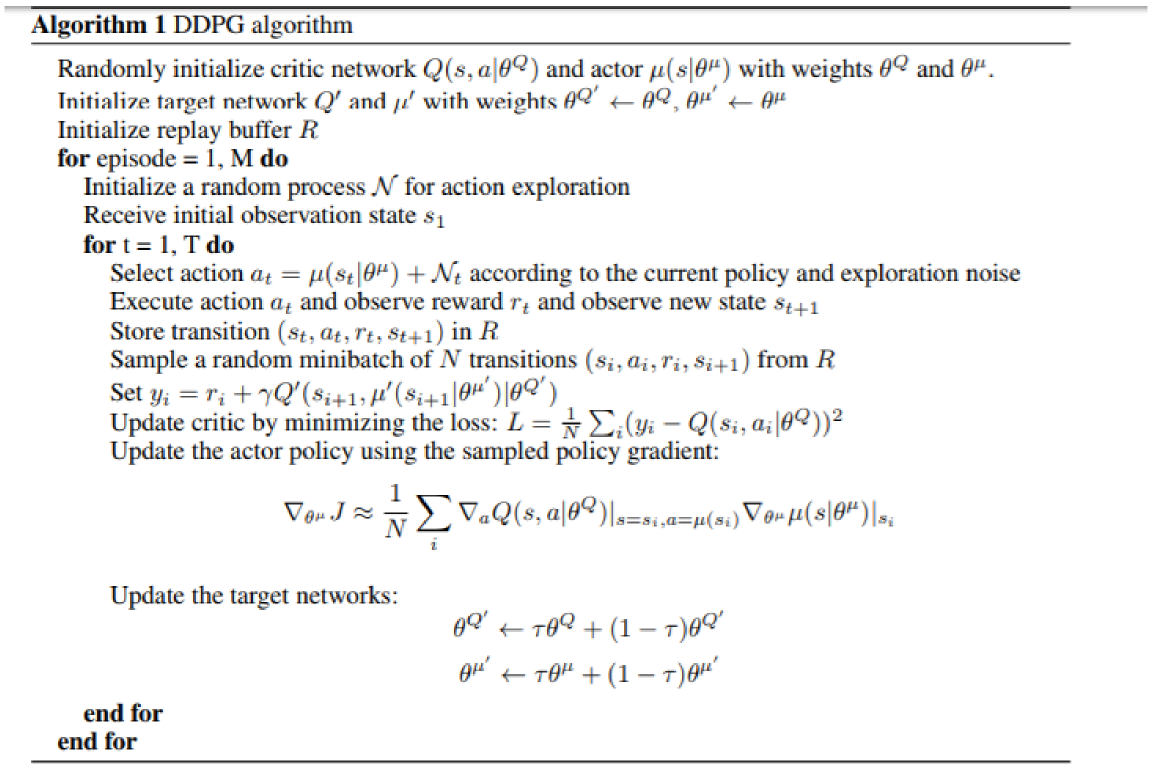
In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

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.

The task is episodic, and in order to solve the environment, your agent must get an average score of +30 over 100 consecutive episodes.

DDPG Algorithm

The Deep Deterministic Policy Gradient (DDPG) is an actor-critic algorithm. The input of the actor network is the current state, and the output is a single real value representing an action chosen from a continuous action space. The critic’s value model output is the estimated Q-value of the current state and of the action given by the actor. The deterministic policy gradient theorem provides the update rule for the weights of the actor network [1] & [2].



Algorithm 1 DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  for t = 1, T do
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:
        $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
    Update the target networks:
        $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
        $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for
```

DDPG Model

The Actor's model I used consisted of two hidden linear layers with 400 and 300 units. The output layer used was tanh. The Critic's model consisted of three hidden layers - 400, 300 & 100 units. Each layer is fully connected with RELU activations with an output layer consisting of 1. Batch normalization was applied on both the Critic's & Actor's model.

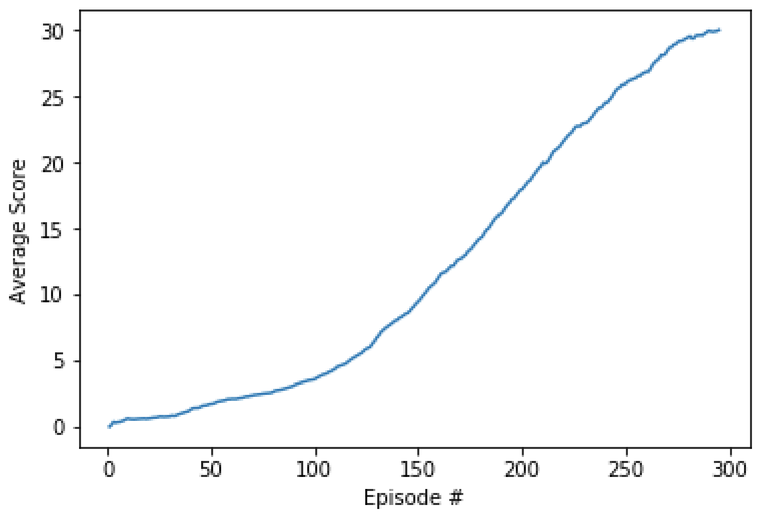
The hyperparameters that were used in the model consisted of:

```
BUFFER_SIZE = int(1e6)    # replay buffer size
BATCH_SIZE = 1024         # minibatch size
GAMMA = 0.99              # discount factor
SIGMA=0.05
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
n_episodes=300            # of episodes
```

Plot of Rewards

The DDPG algorithm achieved an average score of 30.03 in 295 episodes as seen below:

```
Environment solved in 295 episodes!     Average Score: 30.03
```



Improvements

The areas of improvement would be to continue to tune the hyper parameters and also to implement a PPO model.

References

[1] https://pemami4911.github.io/blog/2016/08/21/ddpg-rl.html

[2] https://arxiv.org/abs/1509.02971