Project: Continous Control

Submitter:

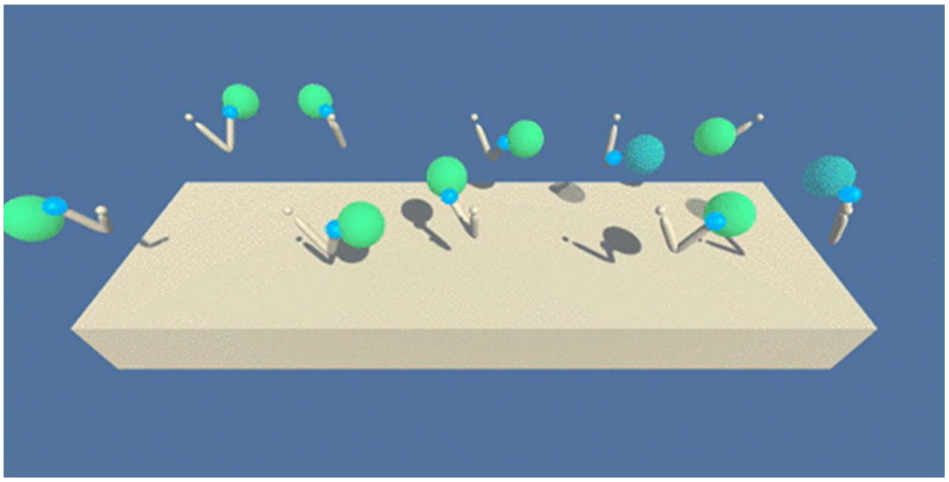
Monimoy Deb Purkayastha (monimoyd@yahoo.com | monimoyd@gmail.com)

Aim:

For fulfilment of Udacity Deep Reinforcement Learning Nano Degree

I. Project Overview

Problem Statement



In this project, agent is an acrobat arm that has two joints needs to be trained so that it tracks a balloon. As the balloon moves, the two joints are adjusted to track the balloon. a double-jointed arm can move to target locations. The environment gives a reward of +0.1 is provided for each step that the acrobat arm reaches the goal location. Thus, the goal of the 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 Unity environment is used to train and test the agent

For this project, there are two versions of Unity environment available:

- The first version contains a single agent.
- The second version contains 20 identical agents, each with its own copy of the environment

In this project I have used second version having 20 agents. The environment is considered solved if in 100 consecutive episodes the mean score is 30 or over

II. Solution approach

In this project I have used Deep Deterministic Policy Gradient (DDPG) algorithm for training. Deep Deterministic Policy Gradient (DDPG) is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

It uses Actor Critic approach where Actor function specifies action given the current state of the environments. Critic value function specifies a signal (TD Error) to criticize the actions made by the actor.

DDPG uses experience replay where experience tuples (S,A,R,S`) are added to replay buffer and are randomly sampled from the replay buffer so that samples are not correlated.

DDP also uses separate target neural network for both Actor and Critic. As target values are determined for both the critic and actor networks, copy of both of these networks and soft update their weights are periodically updated to the respective target networks.

III. Methodology

In this project, I have used DDPG algorithm. The DDPG uses both experience replay and a separate target Neural Network

For Deep Q learning I have used pytorch for implementing Neural Network.

There are four neural networks used

i. Local network for Actor
ii. Target network for Actor
iii. Local network for Critic
iv. Target network for Critic

For all the 4 neural network I have used 128x128 hidden layers.

I used Adam optimizer and used the following hyper parameters:

Batch Size: 128
Discount Rate (gamma) : 0.99
Soft update of target parameters (Tau) : 0.001
Neural Network Initial Learning Rate of Actor: 0.0001
Neural Network Initial Learning Rate of Critic: 0.0005
Initial Exploration rate (Epsilon): 1.0
Final Exploration rate (Epsilon): 0.05
Exploration Rate (Epsilon) decay factor=0.00002
Buffer Size: 100000

As the network tend to overfit after training for many episodes, I have implemented a method to reduce the learning rate by 0.7 if the mean over consecutive 10 episodes does not improve by 1.5 after 10 episodes. Once learning rate is changed, next time will be changed only after another 20 episodes. This approach helps in getting out of plateau region of learning curve.

Noise process Ornstein-Uhlenbeck with mean value of 0 and sigma of 0.2 has been added to explore states.

Also I gathered following metrics while performing training

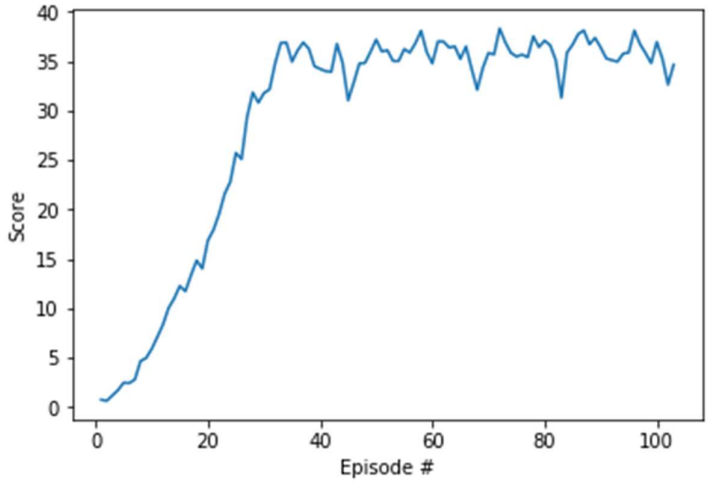
i. **Score** : Mean of scores received by all the agents in each episode
ii. **Mean Score:** Mean score over last consecutive 100 episodes
iii. **Steps:** Number of steps used in each episode

IV. Results

The environment is solved in just **103** episodes and achieved a score of **30.033**.

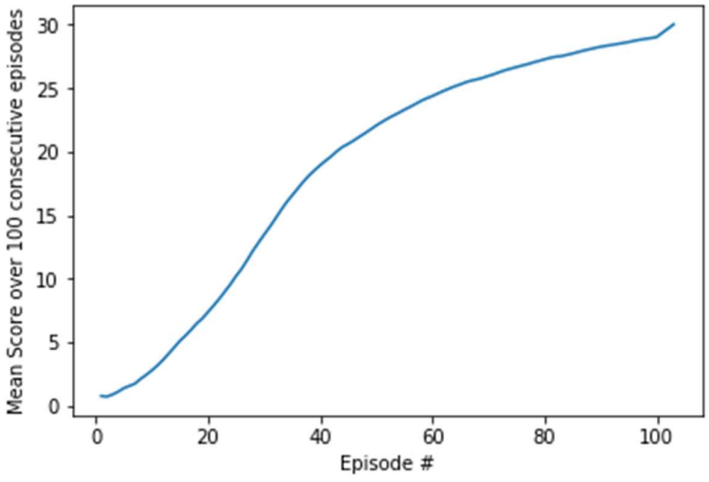
When I loaded the saved model and tested for 5 episodes, the mean score was **34.85** which is better than training averages

- **Plot of scores vs episodes**



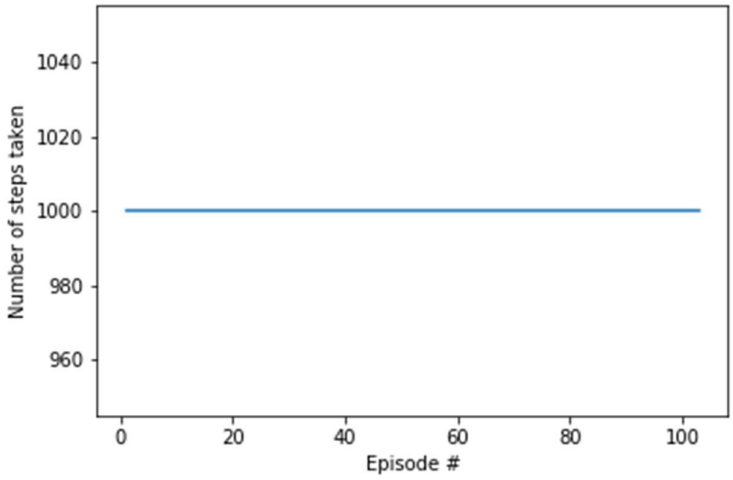
Observation: The scores gradually increase exponentially and then stabilizes

- **Plot for Average Score over 100 consecutive episodes vs episodes**



Observation: The mean scores over 100 episodes increases over as episodes increases, the slope is slightly more than linear.

- **Plot for number of steps vs episodes**



Observation: The number of steps are constant over episodes. Not very useful for analysis

V. Conclusion:

In this project I have used Deep Q Learning algorithm and neural networks with 64x64 dimensions for both Actor and Critic

i. I have trained the agent and achieved the goal of score of 30 over consecutive 100 episodes in just 103 episodes. Also, the test average score I loaded the model was 34.85
ii. Scores (Rewards) earned in each episode has exponentially increased initially before it stabilizes
iii. Mean score over 100 episodes were increasing as episodes increases

Further Improvements

Further improvements to the project can be done using:

i. Proximal Policy Optimization (PPO) algorithm (https://arxiv.org/pdf/1707.06347.pdf)
ii. Prioritized Experience Replay
iii. A3C (https://arxiv.org/pdf/1602.01783.pdf)
iv. D4PG (https://openreview.net/pdf?id=SyZipzbCb)