Contiuous Control Project

Introduction
In this project, I solved the Reacher environment with a single agent. I used DDPG (Deep Deterministic Policy Gradient) algorithm.

Reacher environment[1,2,3]
In this environment, a double-joint arm can move to target locations. The goal is that the agent must move its hand to the target location and needs to maintain its position at the target location as many time steps as possible. The agent reward function is +0.1 each step agent's hand is in target location. 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.

DDPG (Deep Deterministic Policy Gradient)[1,3]
DDPG is an algorithm that simultaneously 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.

Goal
The agent must get an average score of +30 over 100 consecutive episodes.

Learning Algorithm
I followed the code provided by Udacity repository DDPG Pendulum.[1] There are some experimentation that I did to improve the performance of the agent.

1. I changed the nodes of the Actor and Critic fully connected (fc) layers, such as 512/256, 128/64, 64/64, with 3 fc layers, 300/300/300, etc. I also tried asymmetric fc layers between actor and critic.[4] This attempt failed. The reward didn't increase even until 200 episodes.
2. I added batch normalization to the first fully connected layer both Actor & Critic.
3. I tried the sigma value of 0.1, 0.2, 0.3, 0.5 and added mu value to 0.01, 0.1, 0.2
4. I increased the batch size from 128 to 1028
5. I increased the max_t from 300 to 10000
   Until step 5, I still could not make my agent to learn. I had poor results around 1-2 until 200 episodes

   Below are sample of results from some of my experimentation:

```
Episode 50      Average Score: 0.24
Episode 51      Average Score: 0.24     Time Duration: 3.30
Episode 52      Average Score: 0.25     Time Duration: 3.34
Episode 53      Average Score: 0.25     Time Duration: 3.35
Episode 54      Average Score: 0.25     Time Duration: 3.41
Episode 55      Average Score: 0.26     Time Duration: 3.57
```

```
Episode 100     Average Score: 1.62
Episode 200     Average Score: 1.88
Episode 235     Average Score: 1.60
```

6. Then, I decreased the LR_ACTOR and LR_CRITIC from 1e-3 to 2e-4, used fc layers (both actor and critic) 200/200, added batch normalization to the first fc. I was able to improve the performance of the agent, however, the reward went down after it reached average score of 27 at around 400+ episodes.

```
Episode 100     Average Score: 2.56
Episode 200     Average Score: 4.91
Episode 300     Average Score: 15.11
Episode 400     Average Score: 25.32
Episode 500     Average Score: 25.05
Episode 600     Average Score: 23.87
Episode 660     Average Score: 20.69
```

7. With the help of a mentor, he mentioned that changing the Actor and Critic learning rates, together with the buffer size, will improve the agent performance.
   I increased the buffer size from 1e5 to 2e5

```
Episode 100     Average Score: 2.70
Episode 200     Average Score: 5.54
Episode 300     Average Score: 8.36
Episode 400     Average Score: 12.43
Episode 500     Average Score: 26.20
Episode 541     Average Score: 30.17
Environment solved in 541 episodes      Average Score30.17
```

8. By increasing the buffer size from 2e5 to 3e5, I got better result.

```
Episode 100     Average Score: 2.71
Episode 200     Average Score: 8.71
Episode 300     Average Score: 17.67
Episode 400     Average Score: 27.51
Episode 428     Average Score: 30.06
Environment solved in 428 episodes      Average Score30.06
```

<u>Hyperparameters:</u>

- BUFFER_SIZE= 3e5
  The replay buffer size
- BATCH_SIZE= 128
  Number of inputs processed per batch when running Stochastic Gradient Descent
- GAMMA= 0.99
  Discount factor of the Q-Learning Algorithm
- TAU: 1e-3
  To perform soft updates of the target network parameters
- LR_ACTOR: 2e-4
  Learning rate of the actor
- LR_CRITIC: 2e-4
  Learning rate of the critic

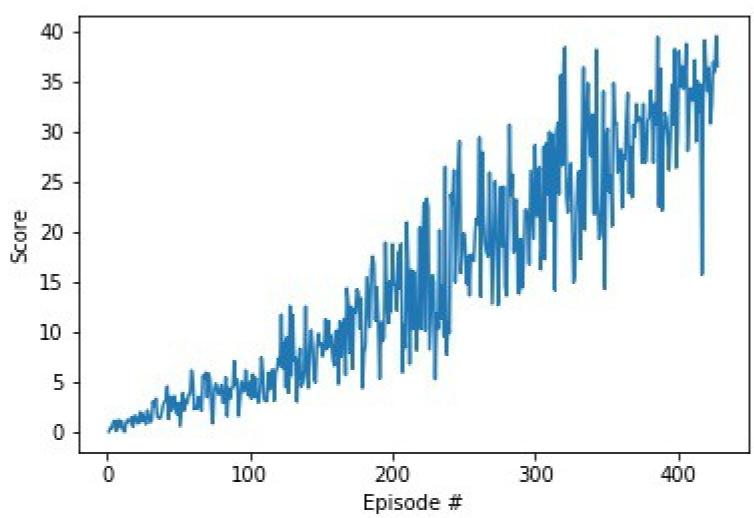
Plot of Rewards

By increasing the Actor and Critic learning rates to 2e-4, buffer size to 3e5, reducing sigma value to 0.1, I solved the environment in 428 episodes.

```
Episode 100     Average Score: 2.71
Episode 200     Average Score: 8.71
Episode 300     Average Score: 17.67
Episode 400     Average Score: 27.51
Episode 428     Average Score: 30.06
Environment solved in 428 episodes      Average Score30.06
```

7. Reward Plot

```
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(np.arange(1, len(scores)+1), scores)
plt.ylabel('Score')
plt.xlabel('Episode #')
plt.show()
```



Ideas for Future Work

This is an interesting project as I need to find the right hyperparameters to adjust in order to get the expected result.

For the future projects:

- Make a video output of this project
- Try higher buffer size to see if I can solve the environment faster
- Do the 20 agents training
- Try different algorithm, PPO

References

1. Deep Reinforcement Learning Nano Degree Udacity Course

2. https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md#reacher

3. Lillicrap. TP., Hunt, JJ., Pritzel, A., Continuous Control with Deep Reinforcement Learning. https://arxiv.org/pdf/1509.02971.pdf

4. Pinto, L., Andrychowicz, M., Welinder, P., et al. Asymmetric Actor Critic for Image-Based Robot Learning

   https://arxiv.org/pdf/1710.06542.pdf