Continuous Control

DQN can solve problems with high-dimensional observation spaces; it can only handle discrete and low-dimensional action spaces. Many tasks of interest, most notably physical control tasks, have continuous (real valued) and high dimensional action spaces. DQN cannot be straightforwardly applied to continuous domains since it relies on a finding the action that maximizes the action-value function, which in the continuous valued case requires an iterative optimization process at every step.

In order to solve our problem statement we will be using DDPG(Deep Deterministic Policy Gradient) which is an actor-critic algorithm that extends **DQN** to work in continuous spaces. Here, we use two deep neural networks, one as actor and the other as critic. Similar network architectures are used for both actor and critic.

During initial trials, my average rewards are very low (<10) even after I complete 1000 episodes. (As shown in below plots)Sometimes rewards used to increase till 200 episodes and by the time I think I am near to the solution, rewards used to decrease again. After some research and suggestions in slack channel I have used batch normalization in actor network which gave average reward of 20 and then I have implemented the same in both the networks through which I was able to solve the problem.

```
    return scores

In [8]: %time scores = ddpg()

c:\users\smarasan\appdata\local\continuum\anaconda2\envs\drlnd\lib\site-packages\torch\nn\functional.py:1374: UserWarning: nn.f
unctional.tanh is deprecated. Use torch.tanh instead.
  warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh instead.")

Episode 100     Average Score: 4.10
Episode 200     Average Score: 11.71
Episode 300     Average Score: 15.33
Episode 400     Average Score: 19.06
Episode 500     Average Score: 19.66
Episode 600     Average Score: 16.92
Episode 700     Average Score: 13.96
Episode 800     Average Score: 13.52
Episode 900     Average Score: 8.600
Episode 1000    Average Score: 5.55
Wall time: 3h 45min 20s
```

```
In [8]: %time scores = ddpg()
```

```
c:\users\smarasan\appdata\local\continuum\anaconda2\envs\drlnd\lib\site-packages
unctional.tanh is deprecated. Use torch.tanh instead.
  warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh instead.")
```

```
Episode 100	Average Score: 2.40
Episode 200	Average Score: 16.07
Episode 300	Average Score: 26.77
Episode 400	Average Score: 21.97
Episode 500	Average Score: 16.50
Episode 600	Average Score: 15.00
Episode 700	Average Score: 8.162
Episode 800	Average Score: 6.50
Episode 900	Average Score: 8.12
Episode 1000	Average Score: 8.17
Wall time: 4h 29min 46s
```

Hyper Parameters:

BUFFER_SIZE = int(1e5) # replay buffer size

BATCH_SIZE = 128   # minibatch size

GAMMA = 0.99   # discount factor

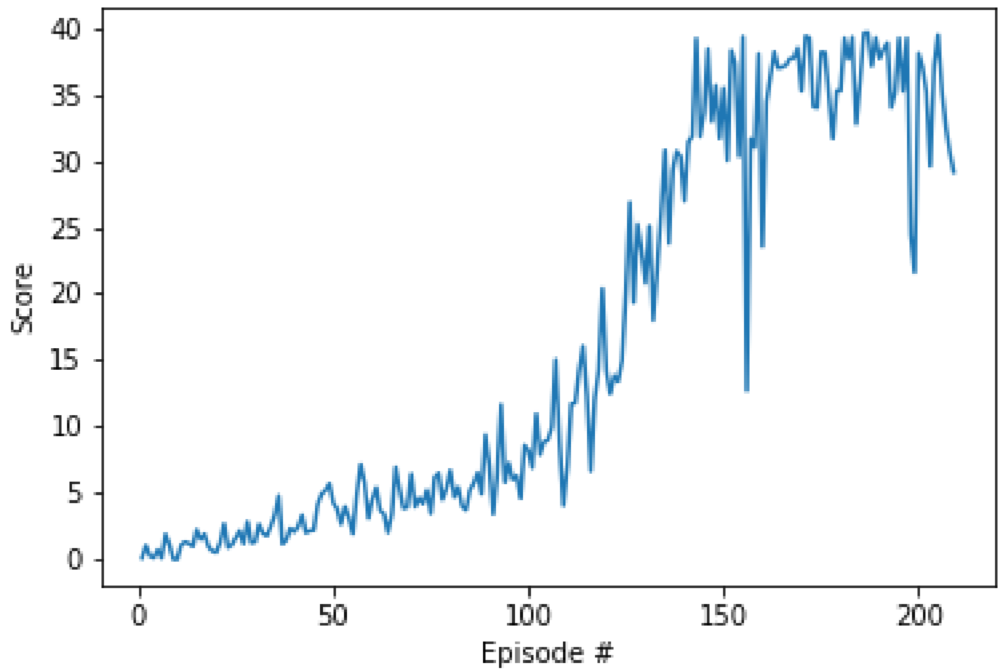
TAU = 1e-3   # for soft update of target parameters

LR_ACTOR = 2e-4   # learning rate of the actor

LR_CRITIC = 2e-4   # learning rate of the critic

WEIGHT_DECAY = 0   # L2 weight decay

Plot of Rewards:



Ideas for future work:

I want to try hyper tuning parameters more and also try algorithms (A3C, A2C, PPO) which were explained in the course. I would also like to go through one or two research papers thoroughly to see how they are using DDPG algorithms and their use cases.