Introduction

I have worked on the DDPG agent algorithm by tweaking the hyper parameters to achieve the objective of achieving average score of 30+ for over 100 episodes.

Learning Algorithm (Deep Deterministic Policy Gradients (DDPG)

I used Deep Deterministic Policy Gradients (DDPG) pendulum for this problem. a kind of actor-critic method.

Here is the algorithm:

At a high level DDPG algorithm can be broken into: Experience replay, Actor and critic network updates, target network updates and exploration

Experience replay: Since this activity is sequential but to optimize, we need independently distributed data. So we store them in a replay buffer and take random batches for training. the sample experience is used to update neural network parameters. we save all the experience tuples (state, action, reward, next_state) and store them during each trajectory the next-state Q values are calculated with the target value network and target policy network. Then, we minimize the mean-squared loss between the updated Q value and the original Q value

Actor and critic network updates: the target value network and target policy network as used to update the next state values. we try to minimize the mean-squared loss between the updated Q value and the original Q value

Target Network Updates: We make a copy of the target network parameters and have them slowly track those of the learned networks via soft updates
Exploration: since the problem is in continuous action spaces, exploration is done by adding noise to the action. In the DDPG paper mentioned in the reference, the authors use Ornstein-Uhlenbeck Process to add noise to the action output

Network Architecture

The network architecture for actor and critic has two fully connected hidden layers of 400 and 300 units. Adam was used as an optimizer for both actor and critic networks in the DDPG pendulum code.

Hyperparameters

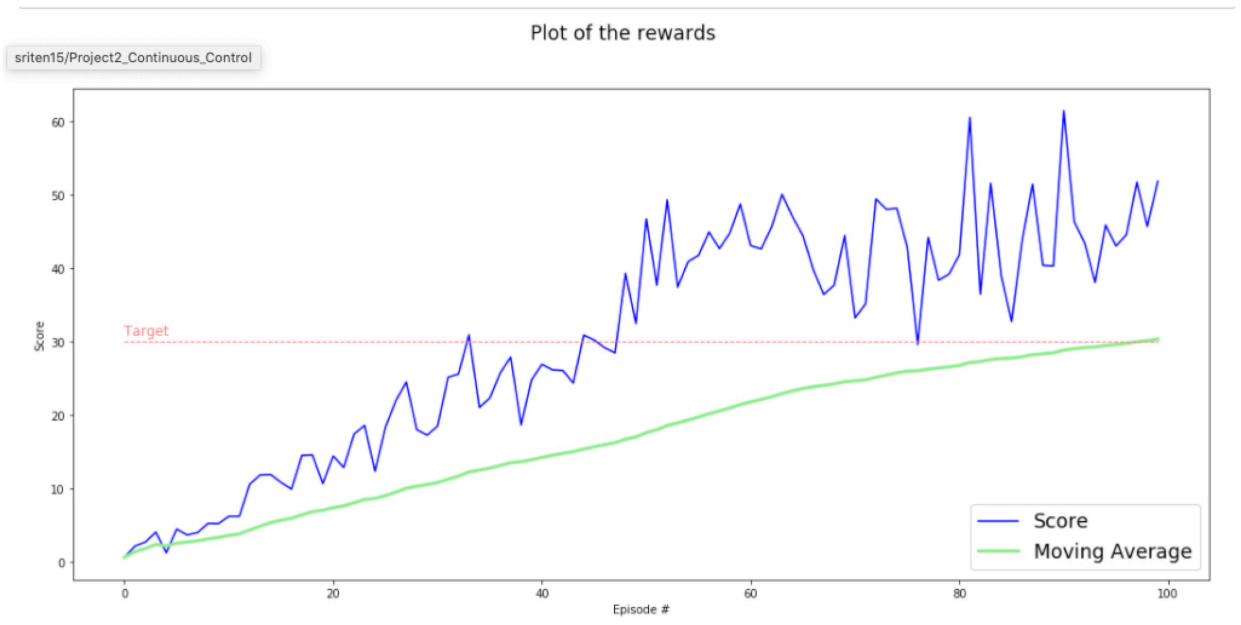
The important aspect of this project is to tweak the hyper parameters to achieve project objectives

Initially, I tried varying the network architecture but it didnt yield expected results. Then i switched backed to existing architecture in ddpg pendulum code and changed the actor and critic learning rate to 2x10-4 each and then further tuned the frequency to update the replay buffer, the final parameters that helped to achieve 30+ moving average after first 100 episodes are below:

```
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 = 0.00        # L2 weight decay
LEARN_FREQ_in_STEP = 5        # learning timestep interval
LEARN_COUNT   = 1        # number of learning passes
```

Conclusion

The problem was solved in 100 episodes for score of over 30+. The plot for rewards is shared below:



Future Work:
The above version shared is a single-agent one. Further i would like to expand my learnings into multi agent environment using DDPG. And try problem with recommended such as TRPO, TNPG and own version of PPO for multi agent problem,