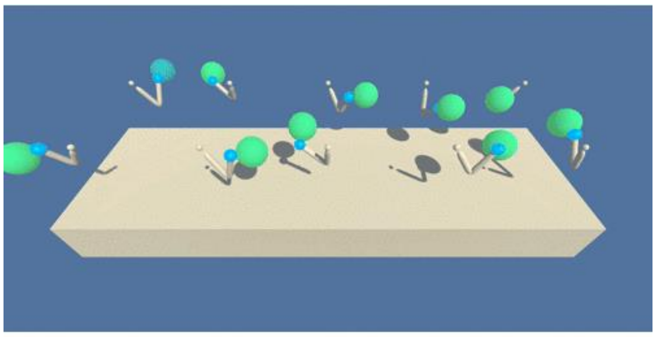
Report – Deep Deterministic Policy Gradient applied for Project 2 – Continuous Control

Objective : To Train a double-jointed arm to reach target locations. The Unity ML-Agents Reacher Environment is used in this Project.

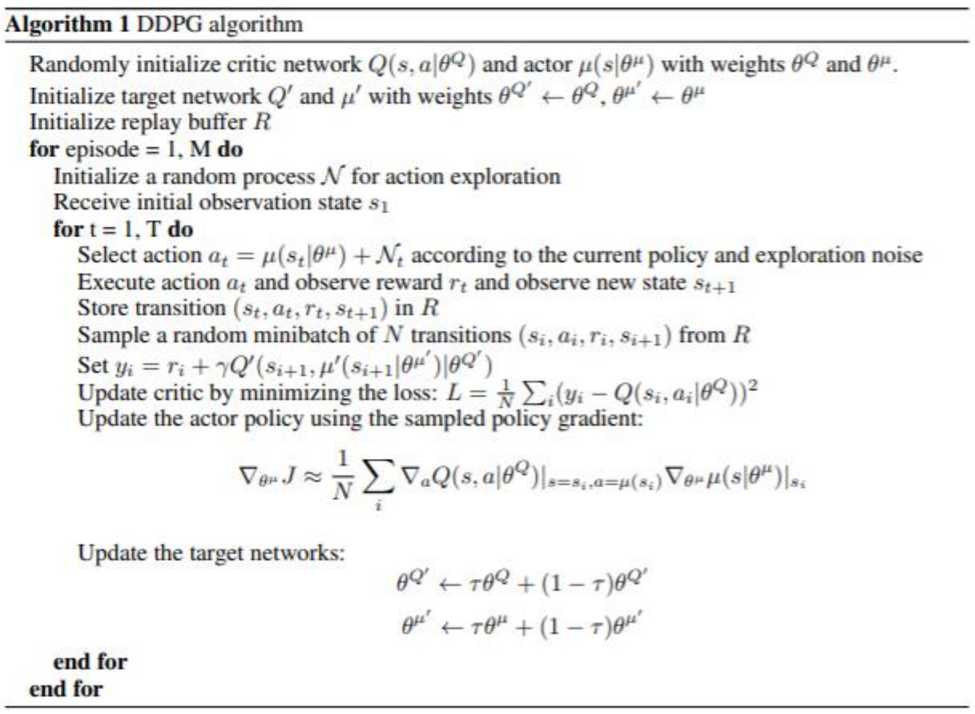

Unity ML-Agents Reacher Environment

Source : Udacity 2nd Project – The Environment-Introduction

Introduction : The state space has 33 variables corresponding to position, rotation, velocity and angular velocities of the arm. The action is defined by a vector with 4 numbers , corresponding to the applicable to two joints.

In this project, the first version containing a single agent is used.

Algorithm : Deep Deterministic Policy Gradient (DDPG) Algorithm is used in this project is referenced from the DDPG-Pendulum [1] for applying the actor-critic methods



```
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
```

Source : Algorithm [2]

The DDPG consists of an Actor and Critic both of which are Neural Networks. This algorithm comes under the Policy based approaches. The objective of the Actor is to approximate the policy in a deterministic way. The input of the Actor network is the state and output is the action taken in that state.

The objective of the Critic is to estimate the value of different state-action pairs i.e. Q-value of the optimal action value function (Based on Bellman's Equation). The weights of the actor and critic network are updated using the soft update strategy.

The learning process takes place by using mini-batches and a replay buffer is utilized here. It is finite sized cache R. The transitions were sampled from the environment according to the exploration policy [2]. Once the buffer gets full, then the oldest samples are discarded [2]. For this algorithm the replay buffer can be large which is beneficial for the learning process.

Model Architecture :

The actor neural network consists of 3 fully connected layers. The number of nodes in the first hidden layer is set as 300 and the second hidden layer consists 150 nodes. ReLU activations are used in the network for including non-linearity. The output layer is followed by the *tanh* activation function.

The critic neural network consists of 3 fully connected layers with two hidden layers with 400 hidden neurons in the first hidden layer followed by 200 neurons in the second hidden layer. ReLU activation functions are used in the network for including non-linearity.

Hyperparameters :

BUFFER_SIZE = int(1e6)

BATCH_SIZE = 128

GAMMA = 0.99

TAU = 1e-3

LR_ACTOR = 1e-4

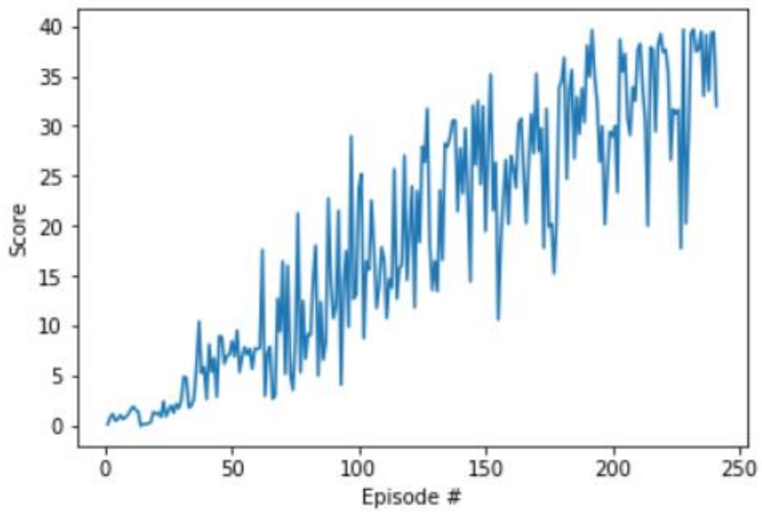
LR_CRITIC = 1e-4

WEIGHT_DECAY = 0.0001

The value of the learning rates for the actor and critic network were set to be the same value and a weight decay value of 0.0001. For these set of the hyperparameters the environment was solved in 241 episodes with an average score of 30.01.

Plot of Rewards per episode :

```
Environment solved in 241 episodes      Average Score attained: 30.01
```



The agent attains an average reward of 30.01 after crossing the 241 episodes.

Future scope for improving agent's performance :

Using Batch Normalizations will help in agent's performance. This will help in reducing the internal covariance shift during training and helps in effective learning [2].

Using prioritized experience replay in DDPG method improves in training time, training stability and final performance [3].

Algorithms like D4PG could be considered as other alternatives as mentioned in the Udacity coursework.

References

[1] Udacity- Deep Reinforcement Learning- Github- https://github.com/udacity/deep-reinforcement-learning/tree/master/ddpg-pendulum

[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra: Continuous control with deep reinforcement learning. ICLR (Poster) 2016

[3] Hou, Yuenan & Zhang, Yi. (2019). Improving DDPG via Prioritized Experience Replay (RL course report). 10.13140/RG.2.2.23694.41287.