Udacity Reinforcement Learning Nanodegree Project 2 - Continuous Control

Jayanth Nair, September 2020

Project Details:

Objective: Train a double-jointed arm to move to target locations in the Reacher Unity ML environment using Deep Reinforcement Learning

Reward System:

The reward system is as follows

- +0.1 for each step that the agent's hand is in the goal location

State Space:

The state space has 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm.

Action Space:

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.

Solution Criterion:

The task is considered solved when the agent scores on average +30 over 100 consecutive episodes.

Learning Algorithm(s)

Two learning algorithms were used to solve this network. Namely, the Deep Deterministic Policy Gradient (DDPG) and the Advantage Actor Critic (A2C).

1) ***DDPG Algorithm:***

   The DDPG algorithm consists of two separate networks, namely the actor and the critic. The actor converts a state into an action. This mapping will be deterministic. Given a state and action, the critic will provide a Q-value.

   “The gradients are calculated from Q which use actions produced by the actor, so the whole system is differentiable and could be optimized end to end with stochastic gradient descent (SGD)” - Maxim Lapan, Deep Reinforcement Learning Hands-on

Since the policy learnt is deterministic, in order to encourage exploration, noise is added to action selection using the Ornstein-Uhlenbeck process. To improve stability, a large replay buffer is used along with target networks for both actor and critic.

DDPG Network Architecture:

The actor network consists of 3 fully connected layers. The input layer consists of 128 units, the hidden layer consists of 64 units and the output layer consists of 4 units corresponding to the action_size parameter. The ReLU activation function is used for the input and hidden layer. The tanh activation function is used on the output layer -> clips actions between -1 and 1

The critic network also consists of 3 fully connected layers with a similar number of units as the actor network. However, in the hidden layer, an additional 4 units are connected to link the actions. The activation functions are the same as the actor network.

The weights for the actor and critic network are initialized as per Lillicrap et al. (2016)

Hyper-parameters:

- Replay Buffer Size - 10^6
- Batch Size - 512
- Discount Factor (Gamma) - 0.99
- Soft Update Annealing Factor (Tau) - 0.001
- Learning Rate for Actor - 0.0001
- Learning Rate for Critic - 0.0001
- L2 Weight Decay for Critic Optimizer - 0
- Time Steps Before Update - 20
- Number of Updates - 10

2) *A2C Algorithm:*

The A2C algorithm also consists of actor and critic networks. It is an improvement on policy gradient methods such as REINFORCE. The idea is to reduce the variance of policy gradients by introducing a state dependent baseline. Policy gradients are directly proportional to the predicted Q values. If the value of a state is subtracted from the predicted Q value, the policy gradient will then be proportional to the advantage (Q(s,a) - V(s) = A (s,a)).

The actor network approximates the policy while the critic predicts the value of the state. The DDPG algorithm is a type of A2C algorithm.

A2C Network Architecture:

TThe A2C network architecture used in this project handles continuous action spaces. The neural network contains of two common fully connected linear layers with three heads. The three heads predict the average action values, the variance of the actions and the value of the state. The average action value and variance is used to predict the action from a normal distribution.

The common body of the network contains an input layer of 128 units and a hidden layer of 128 units. Each of these layers use the ReLu activation function. The average action values head contains a fully connected layer of 128 units with an output layer of 4 units (corresponding to the action size). The output layer uses a Tanh activation function to ensure the action lies between -1 and 1. The action values variance head contains a fully connected layer of 128 units with an output layer of 4 units. The output layer uses a softplus activation function. The state value head consists of 1 fully connected layer of 128 units with an output layer with 1 unit.

An entropy bonus is added to encourage exploration.

Hyper-parameters:

- Discount Factor (Gamma) - 0.99
- Learning Rate - 0.00005
- Entropy Beta - 0.0001

Results

The two plots below show the performance of both algorithms. The DDPG algorithm solved the environment in 162 episodes while the A2C algorithm solved the environment in 440 episodes.

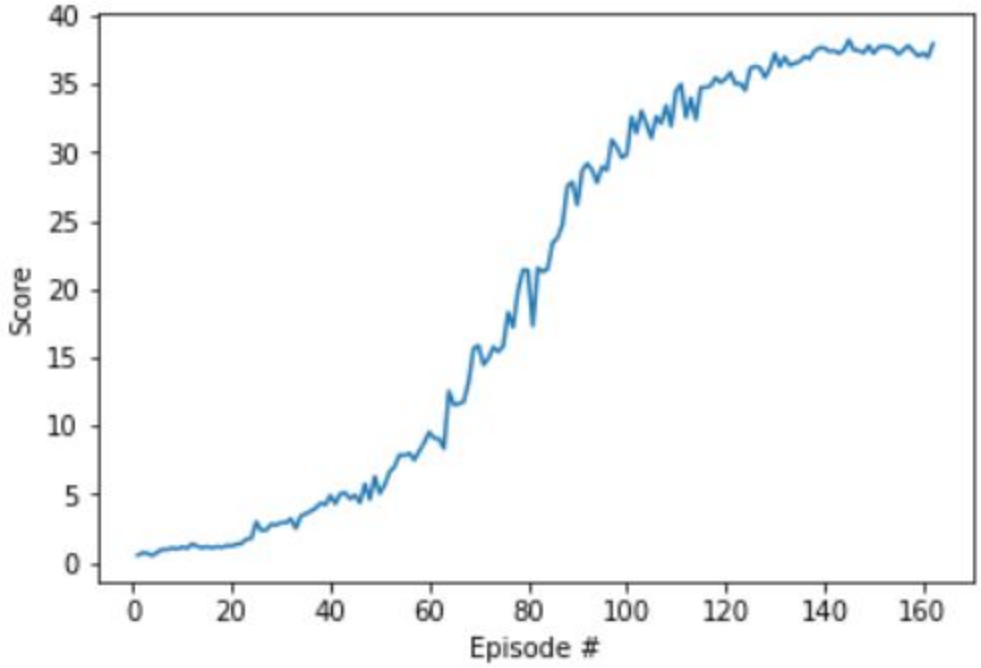

Figure 1 : DDPG Algorithm Performance

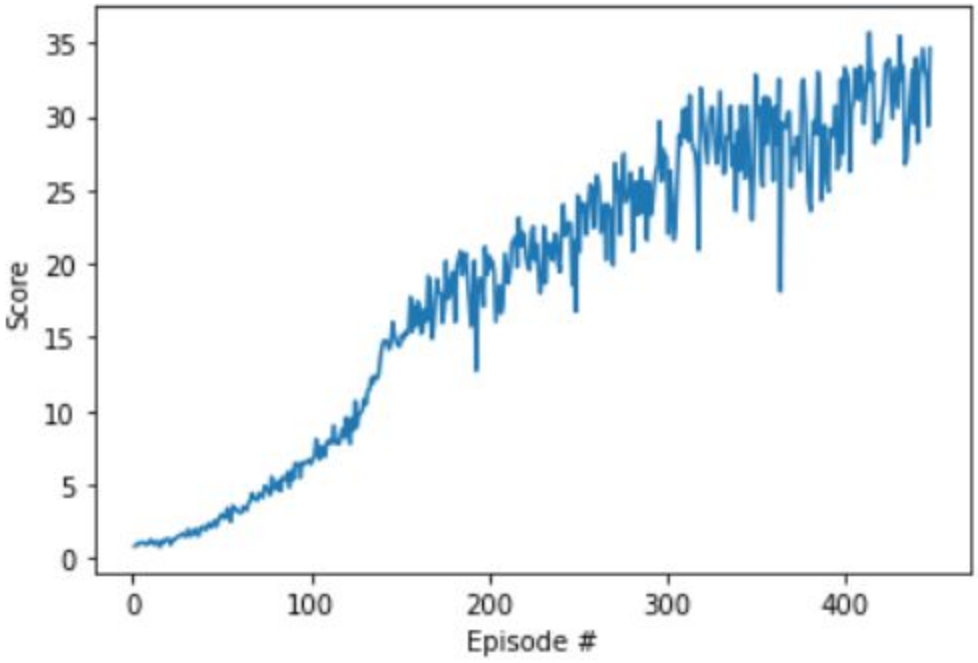

Figure 2 : A2C Algorithm Performance

Lessons Learnt, Next Steps and Room for Improvement

The DDPG algorithm is very unstable and relies heavily on early exploration for future success. Many of my attempts at training the algorithm ended in frustration with scores staying below 5 (significantly lower than the target score of 30). Upon further investigation, I realized the implementation of the OU noise in my code was the main culprit for the poor performance. This excellent thread (https://knowledge.udacity.com/questions/268909) helped me a lot. I modified the OU noise function as in the thread and the agent started exploring more and learning more effectively. Some promising next steps are

1. Try the D4PG algorithm which promises more stability
2. Tweaking the hyper-parameters to see if this environment can be solved in less than 162 episodes
3. Trying random noise instead of OU noise

References:

1. Udacity Course Material, Deep Reinforcement Learning Nanodegree
2. Deep Reinforcement Learning Hands-on by Maxim Lapan