Project 2: Continuous Control

Environment Details

We solve the second version of the problem, which contains 20 identical agents with its own environment. Using 20 agents helps with fast training of our DDPG network. All the data from the 20 agents share a common network and data from all of them are randomly used to update the network.

In this environment, a double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

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.

Learning algorithm

We solve the problem here with the DDPG algorithm. The actor model takes in as input the state variables and returns a vector of 4 variables representing each action. The critic model takes in as input the state variables and action and returns the value corresponding to the state-action pair. We pass the action directly into the hidden layer.

ACTOR NETWORK - Fully connected layer with two hidden layers.
Input size - state_size(33)
Output size - action_size(4)

The first hidden layer has 400 nodes and the second has 300 nodes. The activation function used for the first two layers are relu activation function. The final output from the last layer is passed through tanh activation layer as we want the output values to be within -1 to +1.

Critic NETWORK - Fully connected layer with two hidden layers.
Input size - state_size(33)
Output size - 1(corresponding to Q(s,a))

The first hidden layer has 400+4 nodes and the second has 300 nodes. The additional four is the action as input to the network. The activation function used for the first two layers is the relu activation function. The final output is not passed through any activation function as we want the value.

The hyperparameters for the problem are as follows,

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 512        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay
NUM_UPDATES = 1
UPDATE_EVERY_T = 1
```

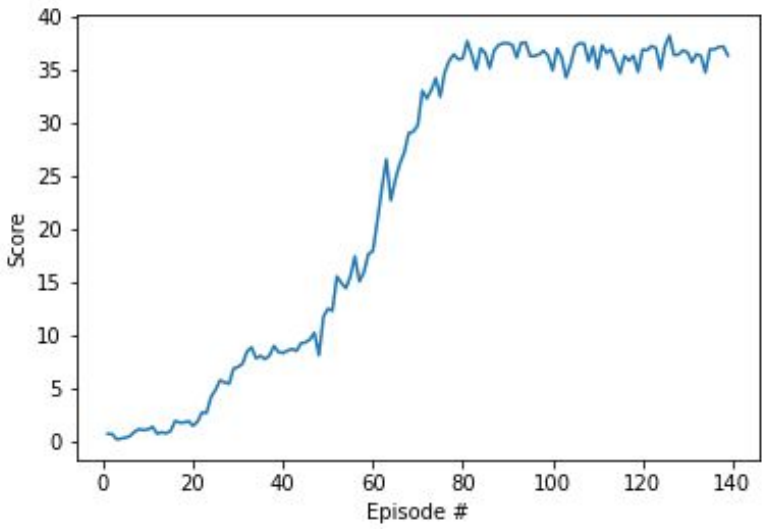
We have taken a batch size of 512 as we have 20 different agents giving us data and this ensures that we can at random take enough data from individual agents. And 512 is not too high that it would slow down learning.

We have taken a gamma as 0.99 since we care about current reward as well as future reward.

The learning rate for the actor and critic are kept low so we are able to maintain stability. We keep the learning rate for the actor even lower to affect us making a huge update that might prevent convergence.

We use the Ornstein-Uhlenbeck process to add noise to the action space. This ensures exploration of our agent while training.

Below is the plot of the score vs episode number for the trained agent. We solve the problem in 139 episodes.



Future Work

I would like to solve the problem with the A2C algorithm to see how well it compares to the DDPG algorithm.

Also try the solcving the crawler problem to see how well this performs.

In terms of improvement, I would like to implement the Twin-Delayed DDPG algorithm (TD3). The concept is simlar to the DDPG algorithm except we use two critics to estimate the optimal future values better. Here the two critics propose two different Q values which are then used.