Project Overview

For this project, we have used Actor-Critic Learning algorithm to train our model. The requirement states that: 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.

Algorithm in use

Deep Deterministic Policy Gradient aka DDPG algorithm continously improves the policy by exploring the environment and converging on the large action space keeping in account the Actor-critic architecture. The actor specifies action in the current state while critic tries to criticize the actions made by actor by using Temporal Difference error.

The goal is to achieve the increase in overall discounted reward which can be acieved by maximizing the action-value function to compute in a direction to change the current action taken by the actor. Several attempts were made to try adding noise to OUNoise parameters to the action space of the policy.However, the exploration part is not taken care by the agent.

Model and the Hyper-parameter used

The model architecture is 3 fully connected layers of an actor and critic neural network. The details for the same are in model.py. Description of the layers:

1. Layer1 takes the state_size as input and gives the output as fc1_units
2. Layer2 takes the fc1, ie the output of the layer1 as the input and outputs fc2_units as the ouput
3. Silimarly, layer3 takes the fc2_units as the input and gives action_size as the output.
4. Both for actor and critic the number of units used as a paremeter for fc1 and fc2 were 400 and 300 respectively.

Almost every thing holds true for the critic layer as well except for the action_size which is 1 for the critic. I did apply batch normalisation on the layer1 for both actor and the critic. Relu was used as activation function. And forward returns via tanh was done for actor and directly for the critic.

The set of hyper-parameter which worked to get to the desired results are:

1. BUFFER_SIZE = int(1e6) # replay buffer size
2. BATCH_SIZE = 128 # minibatch size
3. GAMMA = 0.99 # discount factor
4. TAU = 1e-3 # for soft update of target parameters
5. LR_ACTOR = 2e-4 # learning rate of the actor
6. LR_CRITIC = 2e-4 # learning rate of the critic
7. WEIGHT_DECAY = 0 # L2 weight decay

I tried different values for OUNoise related to theta and sigma, but the value which worked was 0.15 and 0.02 respectiveley

Results

The agent achieved Average Score of 30.25 on the episode 121. The plot for the same could be seen in the notebook

Ideas for future Work

1. Solve the crawler challenge - a more challenging environment to test the DRL skills learned.
2. Experiment with other algoritms like PPO, D4PG and try to achieve similar results.
3. Prioritized Experience Replay could be tried in place of random one, which could help agent to learn better.