Continuous Control with Deep Deterministic Policy Gradient

This report presents the description of the implementation of the Continuous Control in the Reacher Unity environment using Deep Deterministic Policy Gradient (DDPG).

This Reacher environment is a double-jointed arm which 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.

The barrier for solving the environment take into account the presence of many agents. In particular, the agents must get an average score of +30 (over 100 consecutive episodes, and over all agents). Specifically,

- After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 20 (potentially different) scores. We then take the average of these 20 scores.
- This yields an average score for each episode (where the average is over all 20 agents).

1. Learning Algorithm

The project used the Deep Deterministic Policy Gradient algorithm to solve the Reacher environment problem. DDPG was introduced as an actor-critic method that performs well in environments with a continuous action space, which is a known limitation of the popular DQN algorithm.

The algorithm consists of two different types of neural networks: an actor and a critic. The actor network is used as an approximate for the optimal deterministic policy. As a result, only one action is returned for each state, which is considered to be the best action at that state. The critic, on the other hand, learns how to evaluate the optimal action-value function by using actions from the actor.

The pseudocode of this actor critics method obtained from the DDPG paper is showed in below graph.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' 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

The hyperparameters combination which let arrive to the solution for average 30 reward are the following.

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 = 1e-4	# learning rate of the actor
LR_CRITIC = 1e-3	# learning rate of the critic
WEIGHT_DECAY = 0	# L2 weight decay

The DDPG algorithm uses 2 neural networks. The Actor Neural Network has the next characteristics.

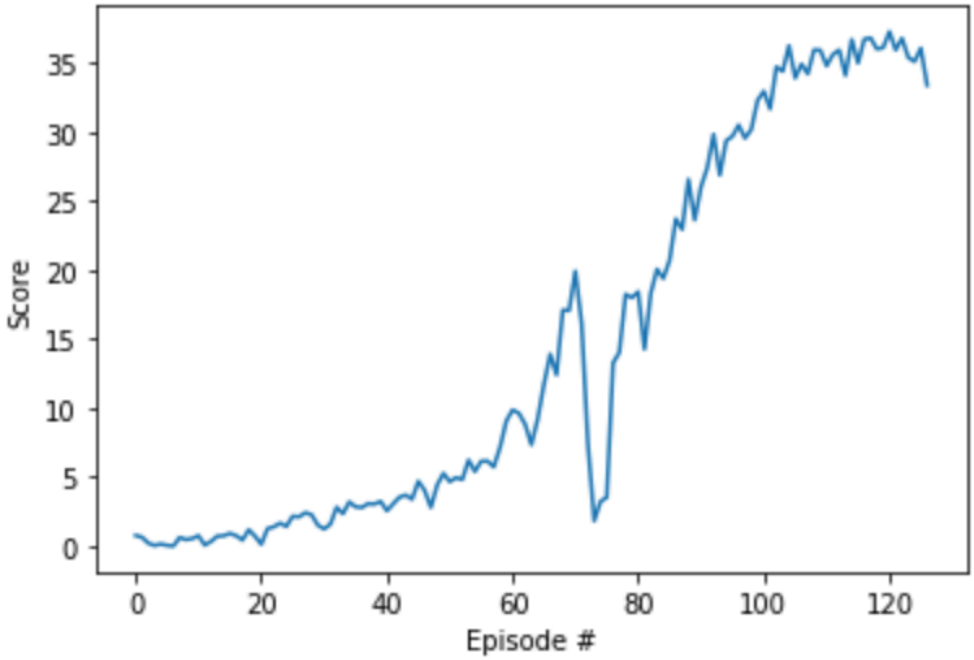
- The model has 3 fully connected layers.
- The first layer takes in the input state which passes through a Relu activation.
- The second layer take the output from the first layer and passes through a Relu activation.
- The third layer takes the output a single value representing the action chosen.
- The optimizer of this network is Adam.

The Critic Neural Network maps (state, action) pairs with Q-values and it has similar architecture than the Actor.

- The model has 3 fully connected layers.
- The first layer takes the state and passes through a Relu activation, then it takes the output from the first layer and concatenate with the action size.
- The second layer which passes through a Relu activation.
- The last layer calculates the Critic's output which estimates the Q-value of the current state and the action given by the actor.
- The optimizer used is Adam.

2. Plot of Rewards

In this project, the agent received an average of 30.05 for 127 episodes and the next graph presents the evolution of the scores to solve the problem.



3. Ideas for Future Work

Some improvements to be done to the actual project are the following.

- This problem will be solved using other similar algorithms as A3C, D4PG and compare with the actual results obtained by DDPG.
- Fine tuning of the model hyperparameters