Udacity Continuous Control Project

1.Learning Algorithm:

1.1.Introduction:

This project was solved using the DDPG algorithm : Deep Deterministic Policy Gradient, which is an Actor-Critic method. This algorithm can be considered as the DQN method for continuous action spaces.

So, it's an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. Let's see its key aspects.

1.2.Continuous action-spaces:

The DDPG algorithm is based on two key components:

- **Actor:** approximate the optimal policy deterministically.
- **Critic:** used to approximate the maximizer over Q-values of the next state.

The main difference between DDPG algorithm and the other actor-critic algorithms is the role of the critic network, which is generally used as a learned baseline in other AC methods.

And in this way, we can deal with actions having a continuous range, instead of just discrete actions like in DQN.

1.3.Replay Buffer:

Like DQN, DDPG uses also a replay buffer to address the problem of **moving targets.**

Actually, the TD target from the update rule of network weights is dependent on our approximation (current weights). And this can lead to harmful correlations!

The solution here is to decouple the TD target from weights' update. For this we will need two networks with identical architectures:

- **Primary Q-network**: its weights will be updated at every learning step
- **Target Q-network**: weights won't be updated at every learning step. The frequency of update is a hyperparameter.

This will be applied to both actor and critic networks.

1.4.Soft Updates:

The difference between DDPG and DQN is that in DDPG the target network weights are updated using a soft update strategy: Every step, mix in your regular network weights into your target network weights.

In practice, this update strategy will result in a faster convergence.

2.Hyperparameters:

This is the list of hyperparams used along with their values and definitions:

- **BUFFER_SIZE**: indicates the replay buffer size for experience replay, value used 1 000 000
- **BATCH_SIZE:** Minibatch size for training, value used 256
- **GAMMA:** Discount rate, value used 0.99
- **TAU**: soft update of target parameters, value used 0.001
- **LR_ACTOR**: Learning rate for actor network, value used 0.0001
- **LR_CRITIC**: Learning rate for critic network, value used 0.0003
- **WEIGHT_DECAY:** L2 weight decay, value used 0.0004
- **UPDATE_FREQ**: Frequency of updating the network, value used 20
- **EPSILON**: starting value of epsilon, value used 1.0
- **EPSILON_DECAY**: decay value of epsilon, value used: 10^(-6)

3.Network Architecture:

3.1 Actor network

- Input layer: Fully connected layer with 33 units (state size is 33)
- 1st Hidden layer: Fully connected layer of 128 units, with relu activation and batch normalization.

- 2nd Hidden layer: Fully connected layer of 128 units, with relu activation and batch normalization.
- Output layer: Fully connected layer of 4 units (action size is 4)

3.2 Critic network

- Input layer: Fully connected layer with 33 units (state size is 33)
- 1st Hidden layer: Fully connected layer of 128 units, with relu activation and batch normalization.
- 2nd Hidden layer: Fully connected layer of 128 units, with relu activation.
- Output layer: Fully connected layer of 1 unit (Maximizer of over Q-values)

4.Results:

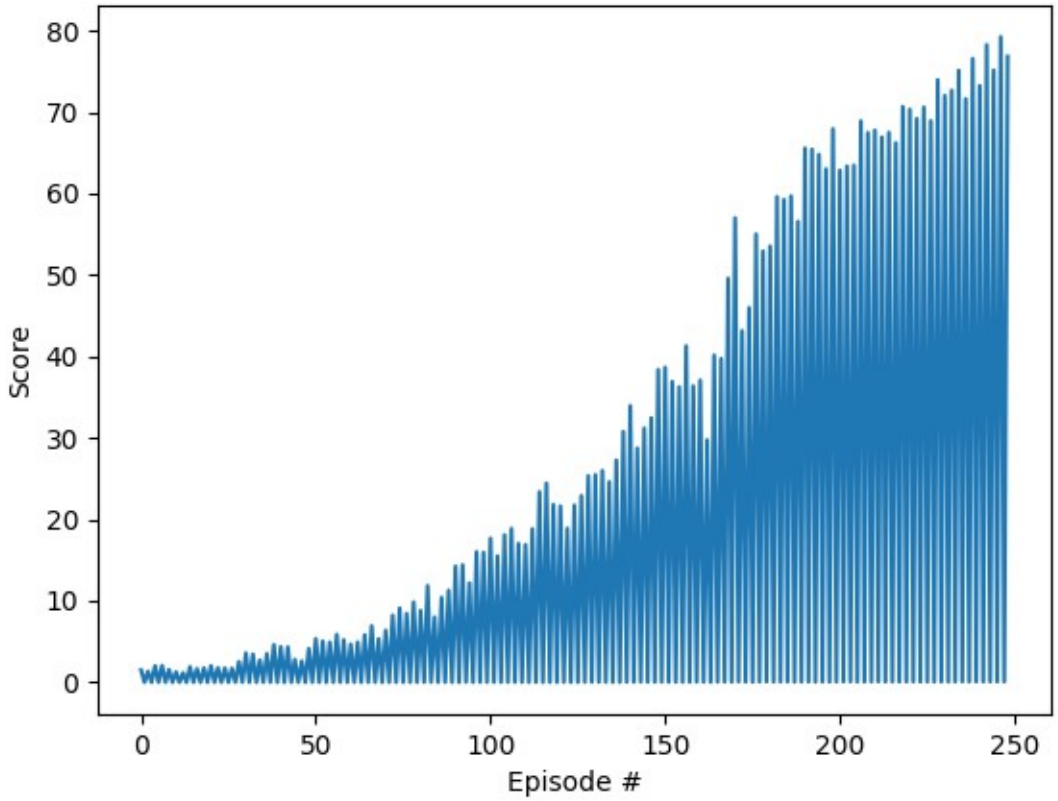
The figure below shows the score evolution regarding the number of episodes done by the agent. As shown here, the problem was solved after about 250 epsiodes.

PS: The environment is considered solved when all 20 agents must get an average score of +30 (over the 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).

The environment is considered solved, when the average (over 100 episodes) of those average scores is at least +30.



5.Ideas for future improvements:

DDPG algorithm can be improved via the use of a set of techniques like: The prioritized experience replay or extending DDPG to the bootstrapped actor-critic architecture to improve the efficiency of exploration.