Udacity Report

Table of Contents

1. Algorythm

① Hyperparameters

In ddpg_agent.py

Replay buffer size	1×10^6
Batch size	512
Discount factor	0.85
τ	1×10^-3
Learning rate of the actor	2×10^-4
Learning rate of the critic	2×10^-4
L2 weight decay	0

In model.py

Number of input nodes in the input layer	33(state size)
Number of output nodes in input layer	128
Number of input nodes in hidden layer	128
Number of output nodes of hidden layer	128
Number of input nodes in output layer	128
Number of output nodes in output layer	4(action size)

In Continuous_control.ipynb

Number of episode	1000
Max time steps	300

Next Pages ↓

② Learning structure

We designed a program based on the following pseudo-code of the DDPG algorithm.

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

③ Neural networks details

The model architecture consists of three layers: input layer, hidden layer, and output layer, and the number of nodes in each architecture has already been described in the "Hyperparameters" section.

For the final output, we adopted tanh as the activation function, considering that the required output obtained by the neural network is between -1 and 1.

Next page ↓

2. Consequence

Episode		229
Average Score		30.18

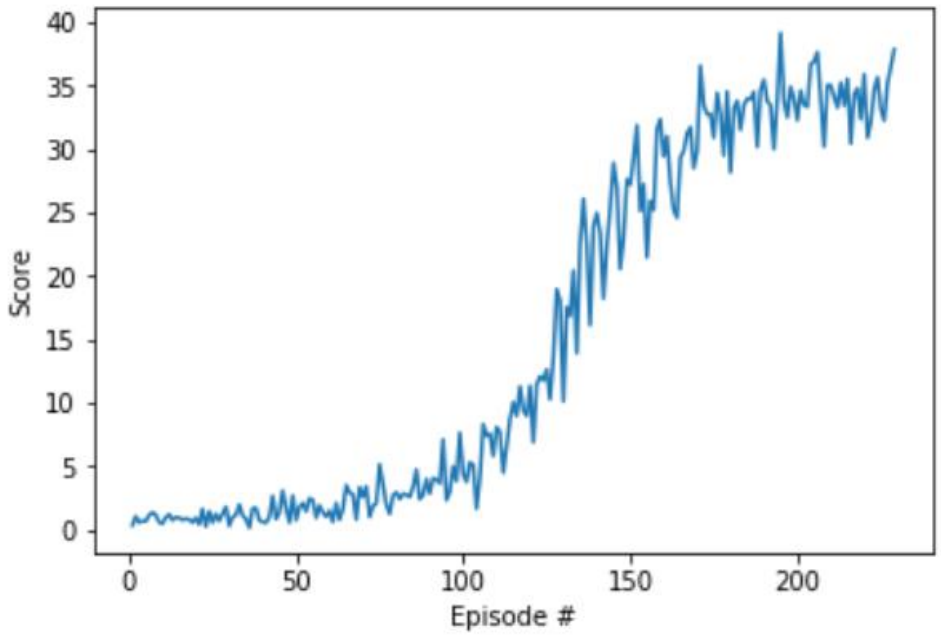

Fig.Average score plots of my Agents reward.

3. Future vision

I would like to try other algorithms such as PPO and A3C.