Deep Reinforcement Learning Project 2: Continuous Control

*Note: Sub-titles are not captured in Xplore and should not be used

Jaeyoung Lee
Seongnam-si, Korea
jylee0121@gmail.com

***Abstract*—This project shows how to solve continuous control using deep reinforcement learning algorithm. As continuous control algorithm DDPG is used for 33 input states from multi-agent Unity based Video Game .**

***Index Terms*—reinforcement learning, DDPG, multi-agent, Noisy Network**

I. Introduction

This project shows DDPG algorithms on 33 input state and improvement by Noisy Network. In Sec II over all algorithms are explained shortly. In Sec III all parameters used for individual experiment are described. Sec IV shows learning curve graphs and analyzes improvements.

II. Applied Algorithms

A. DDPG

DDPG [1] uses deep neural network for continuous control. Fig 1 shows overall DDPG pseudo code.

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

Fig. 1. DDPG Algorithm

B. Noisy Network for Exploration

For efficient exploration, Noisy Network [2] add small noise to weights and bias of last few layers. Additionally this network learns the standard deviation size of this noise, not just adding fixed values. This strategy replace naive Epsilon-Greedy. The paper said that this noise learned over training process remove useless and redundant explorations.

Identify applicable funding agency here. If none, delete this.

III. Parameters

A. Common Parameters

Table I shows the shared parameters over all experiments. Table II, III shows neural network configuration.

TABLE I
Shared Paramaters

Name	Value
Max episode	200
Ending condition	30+ points
μ for OUNoise	0.0
θ for OUNoise	0.15
σ for OUNoise	0.1
Experience replay buffer size	$1e^5$
Batch size	64
Discount factor	0.99
Learning rate for Actor	$1e^{-4}$
Learning rate for Critic	$3e^{-4}$
τ for soft target network update	$1e^{-3}$

TABLE II
Network Configuration for DDPG Actor

Name	Value
1st	fully connected 400
2nd	fully connected 300

TABLE III
Network Configuration for DDPG Critic

Name	Value
1st	fully connected 400
2nd	fully connected 300

B. Noisy Network

For Noisy Network, only initial sigma setup is needed.

IV. Results

A. DDPG

Vanilla DDPG got an average 30+ points over 100 episodes after 101 episodes. Fig 2 shows the result.

TABLE IV
PARAMETERS FOR NOISY NETWORK

Name	Value
Initial σ	0.07f

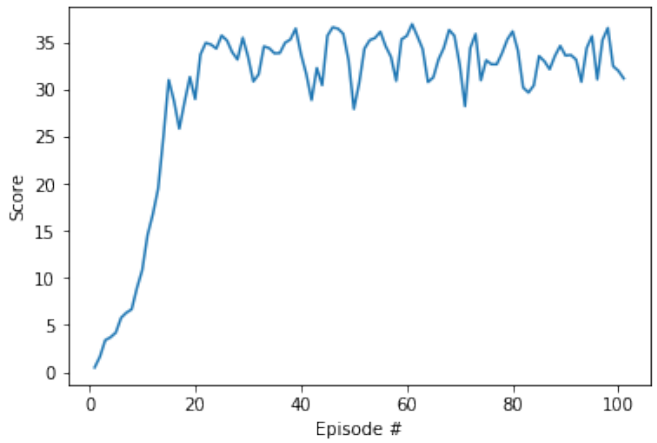

Fig. 2. Learning Curve of DDPG.

B. Noisy Network + DDPG

Fig 3 shows learning curve graph of Noisy DDPG method with red color. It reached 30+ points dramatically faster than vanilla DDPG in 71 episodes. The improvement is $101-71=$ 30 episodes.

C. Comparison and Analysis on the Results

Noisy DDPG improves RL system 30 episodes faster than Vanilla DDPG. Even though multiple tests and statistical analysis should be followed, saving about 30 episodes means certain improvement clearly.

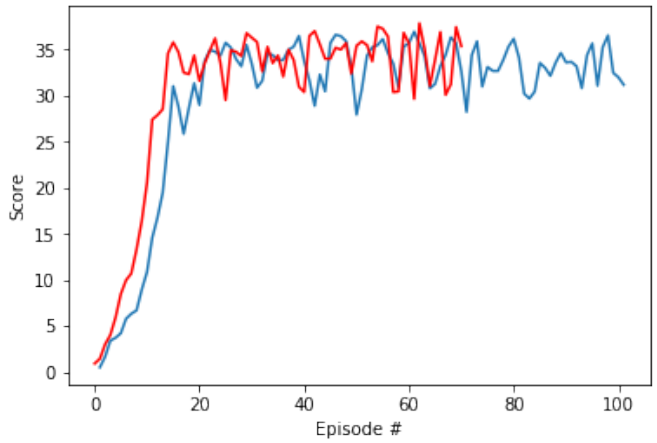

Fig. 3. Learning Curve Graph of Noisy DDPG.

V. FUTURE WORK

Even though this project is based on multi-agent simulator, multi-agent environment is not used fully. It means that the experiments are based on usage of Replay Buffer to remove state-correlation issue. As future work full A2C implementation removing replay buffer is considerable.

REFERENCES

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[2] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," *CoRR*, vol. abs/1706.10295, 2017.