REINFORCEMENT LEARNING
(CS3316)

Assignment Report

Assignment 5

Pendulum
Asynchronous Advantage Actor-Critic
Deep Deterministic Policy Gradient

Name: Kezhi Li ID: 520021911013
Date: 11 May 2023

Contents

1 Introduction

The mission of this assignment is solving the classical Pendulum problem in the Gymnasium environment developed by OpenAI. The system consists of a pendulum attached at one end to a fixed point, and the other end being free. The pendulum starts in a random position and the goal is to apply torque on the free end to swing it into an upright position, with its center of gravity right above the fixed point.

The state of the environment is described by the coordinates of the pendulum's end and the its angular velocity. The action is described by the torque applied to the pendulum. To make the pendulum's center of gravity right above the fixed point, it should be rewarded if the angle of pendulum from the right above place is small, and the velocity of the pendulum is also small.

Figure 1: Pendulum

2 Experiment

2.1 A3C and DDPG

Actor-Critic Algorithm uses Q value function named Critic to replace the unbiased estimation term v_t in the original policy gradient. $\Delta_\theta \log \pi_\theta(s_t, a_t)v_t$. Thus, the actual Q value function $Q^{\pi_\theta}(s, a)$ can be learned, and the variance of the original method can be largely decreased.

Furthermore, the Asynchronous Actor-Critic (A3C) exalted this algorithm to a parallel version, where we should maintain several instances of local agent and a global agent. As the local agents are executed asynchronously in parallel, the parameter of the global agent can be updated by all the local experience.

Instead, the Deep Deterministic Policy Gradient (DDPG) borrows the idea of experience replay and target Q network from DQN into the deterministic policy gradient method. DPG deterministically maps states to specific action instead of outputting a stochastic policy over all actions.

Both the two methods fused the kernel idea of value-based method and policy-based method in reinforcement learning. The A3C is an on-policy method, but the parallel asynchronous scheme makes it free of data correlation. Instead, the DDPG, as an off-policy method, leverage the data more efficiently by reasonably using the experience replay buffer and the target Q network.

The pseudo codes of A3C and DDPG are shown in Fig. 2 and Fig. 3 respectively.

Algorithm S2 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Figure 2: Pseudo code for A3C.

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

Figure 3: Pseudo code for DDPG.

2.2 Experimental Setup

The parameters chosen for DDPG actor and critic are shown in Tab. 1.

Table 1: Parameters for DDPG.

PARAMETERS	VALUES
Actor Learning Rate	0.001
Critic Learning Rate	0.002
Discount Factor γ	0.9
Updating Weight τ	0.01
Memory Length	10000
Sample Batch Size	64
Total Episodes	500

The parameters of A3C is almost the same with DDPG, but without the use of replay buffer. Besides, I trained A3C agent using 5 workers and with 2000 episodes.

The actor and critic network consist of multiple fully connected layers. The detailed implementation is shown in Fig. 4.

2.3 Results

I choose the 500th episode model for DDPG and 2000th model for A3C to show their results. The comparason is shown in Tab. 2. Mote that the average score is calculated by running the model in the environment for 50 times. It can be shown that DDPG has generally better results with less training episodes. The two different models are attached in the compressed file. If you are interested in playing them, you can run the "test_visual()" function with the name of the file.

Table 2: Comparason between A3C and DDPG best models

Model	Episode	AvgScore
A3C	2000	-225.2
DDPG	500	-166.9

Besides, the scores during training are shown in Fig. 5 and Fig. 6 for A3C and DDPG respectively. The score of A3C increased steadily, and it uses 2000 episodes to reach a reasonably good performance. Instead, the DDPG exalted score very quickly, with about 50 episodes. However, DDGP scores have a higher variance than A3C according to the pictures.

3 Conclusion

Both DDPG and A3C are classical and outstanding reinforcement learning algorithms. Both of them can find the good solution of the problem Pendulum. I am expecting their extraordinary show in the AI world nowadays.

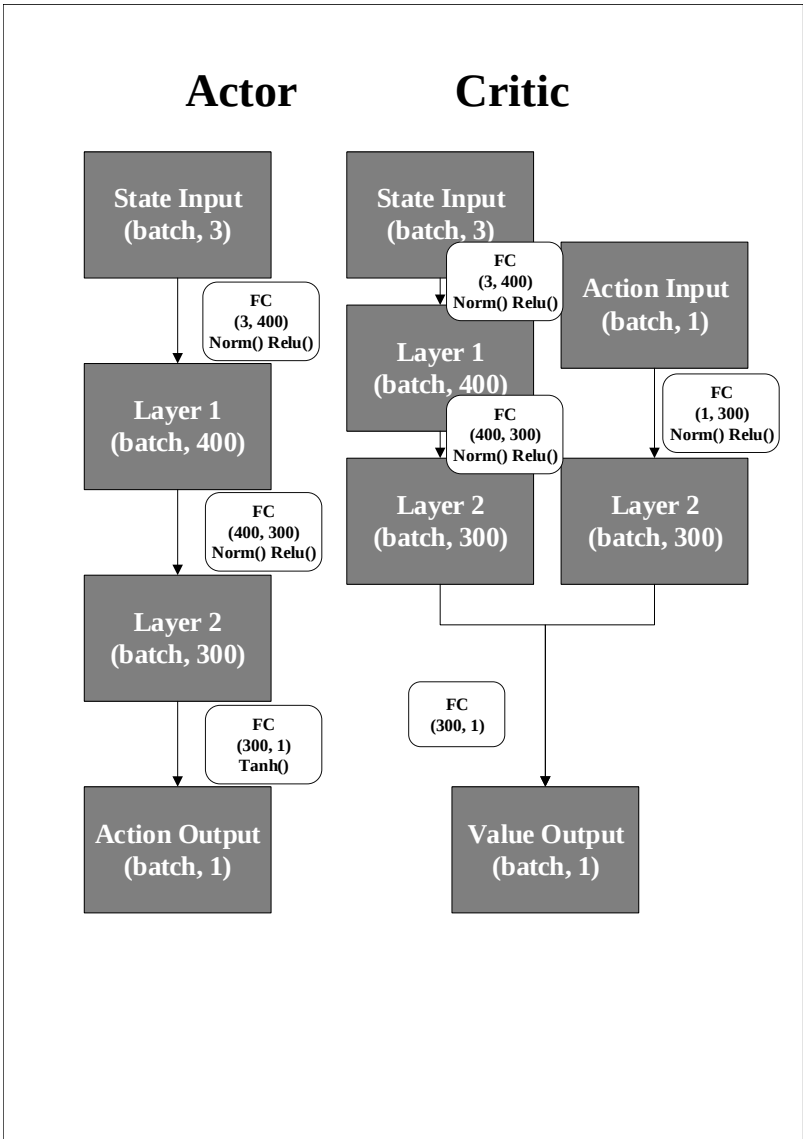

Figure 4: Actor and Critic network framework.

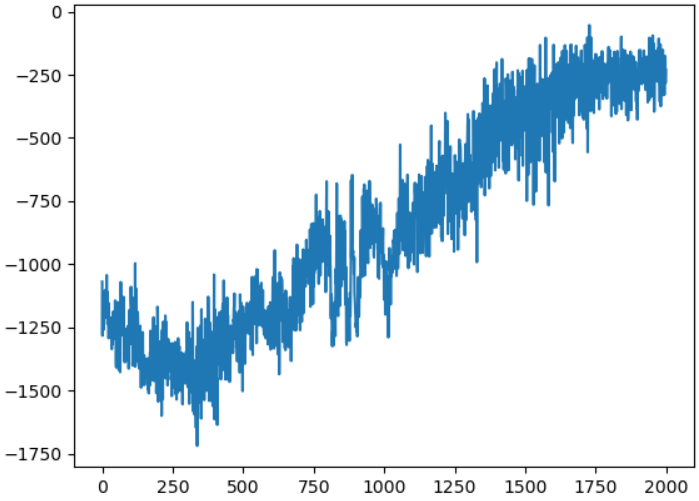

Figure 5: Scores of A3C during training.

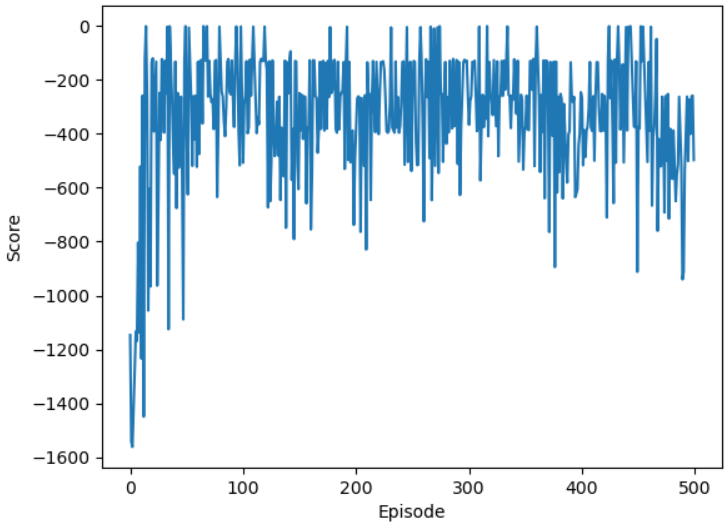

Figure 6: Scores of DDPG during training.