Course Project of Reinforcement Learning (2023), **xx**:xx 1–5

RESEARCH ARTICLE

RL Algorithms in Continous Action Space

Author: Yao Tang

Keywords: A3C, DDPG

Abstract
This study compares the performance of two popular reinforcement learning algorithms, A3C and DDPG, on the Pendulum environment in OpenAI Gym. Both algorithms were implemented using PyTorch and trained with the same hyperparameters and training setup. A3C outperforms DDPG in terms of both reward and convergence speed on Pendulum. However, DDPG exhibits more stability in training and produces less variance in performance. These results provide valuable insights for researchers and practitioners interested in applying reinforcement learning to solve control problems.

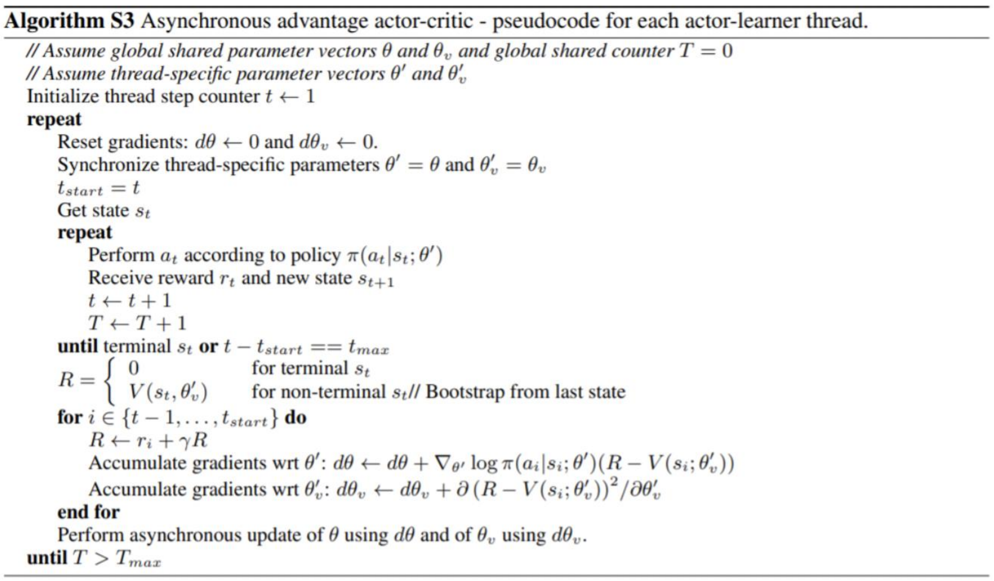
Contents

1. Introduction

Asynchronous Advantage Actor-Critic (A3C) and Deep Deterministic Policy Gradient (DDPG) are two popular reinforcement learning algorithms that have shown impressive performance on a variety of tasks. A3C is a variant of the Actor-Critic algorithm that uses multiple parallel workers to improve the efficiency of training. DDPG is a model-free, off-policy algorithm that combines ideas from actor-critic and deep Q-learning.

While both A3C and DDPG have been extensively studied, there is a lack of systematic comparison between these two algorithms in the literature. In this study, we aim to bridge this gap by conducting a comparative analysis of A3C and DDPG on the Gym environment, the Pendulum environment. We will compare the performance of these algorithms in terms of their training time, convergence speed, stability, and final reward.

The results of this study will provide insights into the relative strengths and weaknesses of A3C and DDPG, and help guide the selection of the appropriate algorithm for a given task. Moreover, this study will contribute to the broader goal of advancing the state-of-the-art in reinforcement learning and improving the ability of intelligent agents to solve real-world problems.



Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

```
// Assume global shared parameter vectors $\theta$ and $\theta_v$ and global shared counter $T = 0$
// Assume thread-specific parameter vectors $\theta'$ and $\theta'_v$
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 $\theta'$: $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        Accumulate gradients wrt $\theta'_v$: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of $\theta$ using $d\theta$ and of $\theta_v$ using $d\theta_v$.
until $T > T_{max}$
```

Figure 1. *Pseudocode of A3C.*

2. Preliminaries

2.1. A3C

A3C is a parallelized deep reinforcement learning algorithm that combines actor-critic methods. It uses multiple agents to explore the environment and update a global network asynchronously. By leveraging both value-based and policy-based approaches, A3C learns policies for sequential decision-making tasks. The algorithm benefits from efficient exploration and exploitation, thanks to its asynchronous nature. Parallel agents provide diverse experiences, reducing bias and improving exploration. A3C is suitable for large-scale systems and efficiently utilizes distributed computing resources. Overall, A3C is a powerful algorithm that achieves effective policy learning in complex environments through parallelization and actor-critic integration.

The detailed algorithm is shown in Fig.1.

2.2. DDPG

DPG (Deep Deterministic Policy Gradient) is an off-policy deep reinforcement learning algorithm that combines actor-critic methods. It is suitable for continuous action spaces and deterministic policies. DDPG uses a separate target network to stabilize learning. It employs a replay buffer to store experiences for efficient learning. The algorithm utilizes the actor network to select actions and the critic network to estimate the state-action value function. DDPG applies gradient updates to both networks to improve the policy and value function. It can handle high-dimensional state spaces and continuous actions. DDPG has been successful in tasks such as robotic control and continuous control problems.

3. Experiment Analysis

3.1. Setting

or the Pendulum-v1 environment, both DDPG and A3C algorithms were implemented with specific hyperparameters. The hidden dimension of the neural networks used in both algorithms was set to

```
Algorithm 1 DDPG algorithm
  Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
  Initialize target network $Q'$ and $\mu'$ 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 2. *Pseudocode of DDPG..*

64. The discount factor (gamma) was set to 0.9, which determines the weight of future rewards in the learning process. These settings were chosen to optimize the performance of the algorithms in the Pendulum-v1 environment. We show our setting in Table 1 as shown below.

Table 1. *DDPG and A3C Hyperparameters.*

Hyperparameters	**Values**
Environment	Pendulum-v1
Hidden Dimension	64
Discount Factor	0.9

3.2. Results

Comparing the performance of A3C and DDPG in terms of convergence speed, final reward after convergence, and stability:

- Convergence Speed:
  A3C: A3C utilizes parallelism and asynchronous updates, allowing for faster convergence compared to DDPG. DDPG: DDPG typically takes more time to converge compared to A3C due to its off-policy nature and the need for exploration.
- Final Reward after Convergence:
  DDPG tends to achieve higher final rewards after convergence. Its continuous action space exploration and the use of a deterministic policy enable DDPG to exploit the environment effectively once convergence is reached. A3C, although achieving competitive rewards, may not reach the same level of optimality as DDPG.
- Stability:
  DDPG is known for its stability during training. The deterministic nature of the policy and the off-policy learning scheme contribute to smoother updates and more stable training. A3C, on the

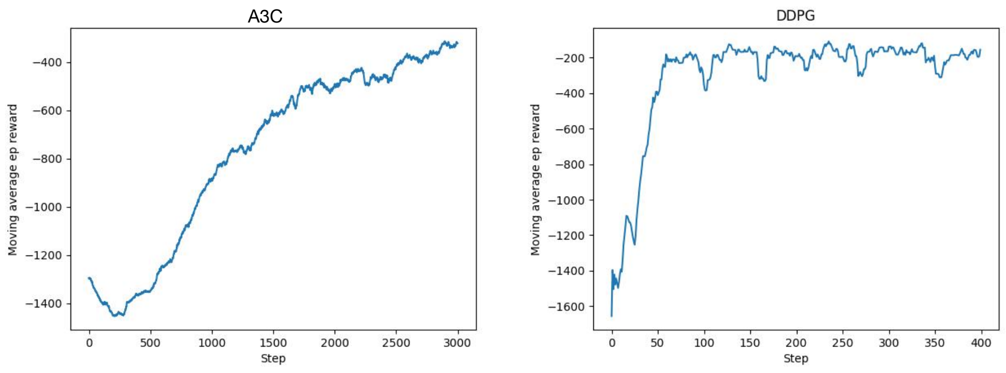


Figure 3. *Results of A3C and DDPG on Pendulum*$_v$1..

other hand, may exhibit more variability and fluctuation during training due to the asynchronous updates and exploration from multiple workers.

To summarize, A3C converges faster due to the parallel exploration from multiple workers, while DDPG achieves higher final rewards after convergence. DDPG's stability during training makes it more reliable, whereas A3C's performance may vary due to the asynchronous updates and exploration from multiple sources. The choice between A3C and DDPG depends on the trade-off between speed of convergence, final performance, and the desired level of stability for a given task.

3.3. Rethink

- Exploration vs. Exploitation:
  A3C's faster convergence is a result of its strong exploration capability through parallel workers. However, this exploration may limit its ability to exploit the environment fully, leading to slightly lower final rewards compared to DDPG.
- Stability-Performance Trade-off:
  DDPG's stability during training allows for reliable and consistent learning. It achieves higher final rewards due to its deterministic policy and off-policy learning scheme. However, DDPG's stability comes at the cost of slower convergence compared to A3C.

4. Conclusion

In conclusion, A3C and DDPG are two popular deep reinforcement learning algorithms that exhibit distinct characteristics and performance trade-offs.

A3C demonstrates faster convergence speed by leveraging parallelism and asynchronous updates from multiple workers. This parallel exploration enables A3C to cover a wider state-action space efficiently, leading to faster convergence. However, A3C may achieve slightly lower final rewards due to its emphasis on exploration, which can limit exploitation.

DDPG, on the other hand, showcases higher stability and final rewards after convergence. Its deterministic policy and off-policy learning scheme allow for effective exploitation of the environment. DDPG's stability during training ensures consistent and reliable learning, but it may take more time to converge compared to A3C.

The choice between A3C and DDPG depends on the specific requirements of the task. If faster convergence is desired, A3C can be a suitable choice. However, for tasks where stability and higher final rewards are crucial, DDPG may be more appropriate. It is essential to consider the trade-offs between exploration, exploitation, convergence speed, stability, and final performance when selecting the appropriate algorithm for a particular reinforcement learning problem.