Evaluating the Performance Metrics of Ppo, Dqn, and Ddpg in Continuous Control Tasks

Huacong Cui

Aberdeen Institute of Data Science and Artificial Intelligence, South China Normal University, Guangzhou, China

Abstract. Reinforcement learning (RL) has made significant progress about solving continuous control and discrete space issues. Each algorithm contains different properties so that they are applicable to various issues. This paper conducts a comparative analysis of three widely used RL algorithms, Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG) to explore and evaluate their performance in the continuous control Pendulum-v1 environment. This work implements each algorithm using standardized hyperparameters and analyzes its overall performance, convergence speed, and training stability using the same experimental setup. The results show that PPO performs better than DDPG and DQN in terms of stability, while DDPG exhibits the fastest convergence speed among the three. DQN performs poorly in continuous control due to its dependence on Q-maximization and discrete action enumeration, causing the large fluctuations during the convergence process. This work emphasizes the significance of environment-algorithm compatibility and offers experimental support for algorithm selection in continuous control applications

1 Introduction

One important branch of machine learning, reinforcement learning (RL), uses agent-environment interactions, trying to identify the best ways to maximize cumulative returns or accomplish certain objectives [1]. In contrast to supervised learning, which uses labeled datasets, reinforcement learning (RL) is a trial-and-error method in which the agent learns by being rewarded or penalized according to its behavior. When dealing with high-dimensional states and action spaces, traditional reinforcement learning techniques frequently encounter difficulties like dimensional catastrophe [2]. Exploring every potential state and action is computationally impossible in complex systems due to the exponential growth of the state and action spaces. It is time-consuming that conventional RL methods require careful feature engineering to represent states and actions effectively. Because of Deep Learning's (DL) quick progress, researchers have created Deep Reinforcement Learning (DRL) via the integration of DL with reinforcement learning [3]. DRL has produced impressive results in the domains of gaming and robot control by leveraging the feature extraction and

20223801068@m.scnu.edu.cn

representation capabilities of neural networks to tackle the decision-making problem in high-dimensional state and action spaces [4].

In DRL, a variety of algorithms have been proposed, such as Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG). These algorithms show their respective advantages and limitations in different application scenarios. For example, by approximating the Q-value function with neural networks, the DQN, which was suggested by Mnih et al., addresses the drawbacks of conventional Q-Learning in high-dimensional state spaces and is mostly utilized in discrete action spaces [5]. DDPG combines deterministic policy gradients with deep learning based on the Actor-Critic architecture, and was created by Lillicrap et al. specifically for solving problems in continuous action spaces [6]. Schulman et al proposed the PPO, which simplifies the policy optimization process, and introduced the clip policy, which works with both continuous and discrete action spaces, to increase algorithm's stability and efficiency [7]. Although the above algorithms have been applied in respective domains, there are still some differences in their performance in different tasks and environments. This paper will do a comparative performance analysis of the three algorithms, PPO, DQN, and DDPG so that the nature of the algorithms can be better understood [8]. Using a continuously controlled environment, Pendulum-v1, to assess the three algorithms' relative stability and rate of convergence [9]. Research anticipates that the experimental results will enhance theoretical research and practical application scenarios, enabling better decision-making when choosing and utilizing DRL algorithms.

2. Materials and Method

This section introduces the principles and formulas of the PPO, DQN and DPPG algorithms, and the environment Pendulum-v1 used in the experiment.

2.1 Proximal Policy Optimization

A policy gradient-based reinforcement learning method called Proximal Policy Optimization (PPO) was created to reduce the size of policy updates and enhance training stability [10]. By implementing a clipped surrogate target that halts significant policy changes, PPO aims to stop performance degradation. In particular, PPO ensures that the policy changes within a defined range by minimizing the differences between the previous and current policies [11].

PPO avoids the instability problems that are frequently present in conventional policy gradient approaches while combining the benefits of policy gradient methods by optimizing a clipped objective function to update policy parameters. And PPO uses an Actor-Critic architecture, in which the Critic network estimates the state value function to assist in computing the advantage function, while the Actor network creates the action policy [12].

The PPO objective function is defined as Equation (1)

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (1)$$

Where, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, representing the ratio of the probability of selecting action a_t under the current policy parameters θ to the probability under the old policy parameters old θ_{old}. $\hat{A}_t$ is the advantage function. ϵ is a hyperparameter that controls the range of policy updates (typically 0.1 or 0.2).$\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ is the clipping function that restricts $r_t(\theta)$ to the interval $[1-\epsilon,1+\epsilon]$.

2.2 Deep Deterministic Policy Gradient

An algorithm for off-policy reinforcement learning is called Deep Deterministic Policy Gradient (DDPG) [13]. It combines the advantages of Deep Q-Networks (DQN) and Deterministic Policy Gradient (DPG) [14]. The two primary parts of DDPG's Actor-Critic architecture are the Critic network, which assesses the Q-value of state-action pairs, and the Actor network, which learns a deterministic policy to produce actions. DDPG stabilizes training by using target networks for the Actor and Critic. These networks are updated by soft updates, and experience replay is used to eliminate data correlations. And maximizing the Q-value that the network of Critic predicts is the goal of the network Actor.

$$J(\theta^{\mu}) = \mathbb{E}_{s\sim\mathcal{D}}\left[Q\left(s, \mu(s|\theta^{\mu})\right)\right] \tag{2}$$

Where, θ^{μ} are the Actor network's parameters, $\mu(s|\theta^{\mu})$ is deterministic action output by the Actor, $Q\left(s, \mu(s|\theta^{\mu})\right)$ is Critic network's anticipated Q-value.

MSE between the target and predicted Q-values is minimized by the Critic network.

$$L(\theta^{Q}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[(Q(s,a|\theta^{Q}) - y)^2] \tag{3}$$

Where, θ^{Q} are the Critic network's parameters, y is the target Q-value, computed as Equation (4)

$$y = r + \gamma Q'(s', \mu'(s'|\theta^{\mu'})|\theta^{Q'}) \tag{4}$$

Where, the discount factor is γ, the target Critic and Actor networks are Q' and μ' .

To stabilize training, a soft update rule is used to the target networks.

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{5}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{6}$$

Where, τ is the soft update coefficient (typically a small value, e.g., 0.001).

The Actor network parameters are updated using the deterministic policy gradient, The calculation process is shown in Equation (7).

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_a Q(s,a|\theta^{Q})|_{a=\mu(s)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})\right] \tag{7}$$

2.3 Deep Q-Network

A reinforcement learning algorithm called Deep Q-Network (DQN) uses deep neural networks and Q-Learning to identify the state spaces with high dimensions. It approximates the Q-value function using a neural network, so it can learn policies in complex environments. DQN utilizes two approaches to stabilize training: experience replay stores past experiences and samples them randomly to break correlations, and target networks provide stable Q-value targets during updates. DQN is particularly effective for discrete action spaces and has been

widely used in tasks like playing Atari games [15]. In DQN, the Q value is updated to carry out the learning process. Specifically, the value is updated by the following Equations.

The first step is Q-Learning Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{8}$$

Where, the learning rate is α , γ is the discount factor, $\gamma \max_{a'} Q(s',a')$ is the next state's maximum Q-value.

The second step is DQN Loss Function

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} [(Q(s,a|\theta) - y)^2] \tag{9}$$

Where, the main parameters of network θ are modified by reducing this loss.
The third step is where the target Q-value y is:

$$y = r + \gamma \max_{a'} Q(s',a'|\theta^-) \tag{10}$$

This ensures that the target y remains stable for a period of time, even as the main network's parameters θ are updated.

Finally is Target Network Update:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{11}$$

Where the update method gradually blends the main network's parametersθinto the target network's parameters θ^-.

And τ is a small hyperparameter (e.g.,τ=0.001) that controls the rate of blending. This approach ensures smoother and more stable updates to the target network.

2.4 Pendulum v1

Pendulum-v1 is a continuous control problem. Maintaining equilibrium while swinging a pendulum upright is the aim. The environment simulates a basic pendulum hanging from a fixed location, and the agent's job is to regulate the pendulum's motion by applying torque. The reward function is intended to motivate the pendulum to remain upright with the least amount of effort. The state space is made up of the pendulum's angle and angular velocity, and the torque given to the pendulum is represented by the action space, which is continuous [16].

The following differential equation governs the motion of the pendulum.

$$\frac{d^2\theta}{dt^2} = \frac{1}{I}\left(mgl\sin(\theta) - \beta \frac{d\theta}{dt} + u \right) \tag{12}$$

The purpose of the reward function is to decrease control effort and encourage the pendulum to remain upright.

$$r = -\left(\theta^2 + 0.1\left(\frac{d\theta}{dt}\right)^2 + 0.001u^2 \right) \tag{13}$$

Where, θ^2 Penalizes deviations from the upright position, $\left(\frac{d\theta}{dt}\right)^2$ Penalizes high angular velocity, u^2 Penalizes large control actions.

3 Results

The study's goal is to evaluate how well the DDPG, DQN, and PPO algorithms perform in the pendulum-v1 environment. Based on the experiment results, differences in the algorithms can be discovered among convergence speed, stability, and average reward.

Convergence Speed: As shown in Figures 1,2,3, DDPG demonstrated the fastest convergence rate among the three algorithms, followed by DQN, with PPO exhibiting the slowest convergence.

DDPG's direct policy optimization and use of a replay buffer facilitate faster learning compared to DQN's value-based approach and PPO's more conservative policy updates.

Stability: Figure 1,2,3 illustrates the training stability of each algorithm. PPO had the most stable learning curve, followed by DDPG, while DQN displayed the highest variance. This observation confirms the expectations, as PPO's clipped objective function effectively limits policy updates, preventing drastic changes and promoting stable training. The utilization of target networks and the replay buffer makes DDPG relatively stable. DQN's instability is due to its reliance on value function approximation and the potential for overestimation bias.

Average Reward: PPO achieved the highest average reward across all algorithms, although its convergence speed is slow. Its stable update mechanism and exploration ability enable it to find better strategies after long-term training, so it can obtain higher average rewards.DDPG attained a lower average reward compared to PPO and It can directly optimize policies and effectively utilize environment-agent interaction through the Actor Critic framework. DQN mainly used for discrete action spaces got the lowest average reward.

Discussion: Overall, in the Pendulum-v1 environment, PPO has the best performance with the stable convergence process and ability to achieve high average rewards, making it the most appropriate algorithm among the three for this environment. Its conservative policy updates and strong exploration capabilities make it well-suited for continuous control tasks where stability and long-term performance are critical. However, DDPG's faster convergence speed suggests that it could be a valuable component in a mixed approach. Combining the advantages of PPO and DDPG could potentially develop an optimization algorithm that converges quickly while maintaining stability and achieving high rewards. In the early phases of training, DDPG might be utilized to quickly explore the policy space, followed by PPO to refine the policy and ensure stable, high-reward performance. On the contrary, DQN is more suitable for environments with discrete action spaces. Its performance is limited and extremely unstable in continuous control tasks like Pendulum-v1, which means it is generally not the first choice for such problems.

For tasks in discrete action spaces, DQN is still a very worthwhile algorithm to utilize, and its effective use of experience replay and target networks makes it perform well in these tasks.

Figure 1 Reward of PPO (Picture credit: Original)

Figure 2 Reward of DQN (Picture credit: Original)

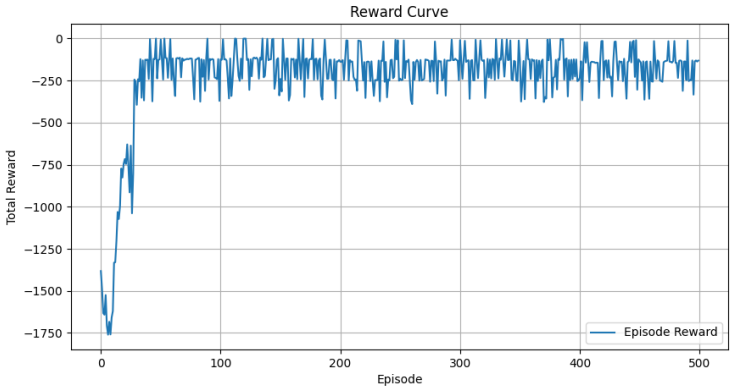

Figure 3 Reward of DDPG (Picture credit: Original)

4 Conclusion

The experiment in this research was carried out in the Pendulum-v1 environment, comparing the performance of three reinforcement learning algorithms—PPO, DQN, and DDPG—in continuous control spaces. The experiment results demonstrate that DDPG has the feature of rapid convergence, so It is suitable for tasks that require high-speed convergence. PPO proves to be the most appropriate algorithm for this environment because it maintains a stable convergence process and achieves the highest average reward among the three algorithms. However, DQN has disadvantages in handling continuous action spaces, which shows that it should be applied in discrete action environments.

This study's primary drawback is that it was only evaluated in a single environment. The people could explore more complex tasks and fusion approaches, such as combining the PPO and DDPG to develop optimized algorithms in future work. To utilize the stability of PPO and fast convergence of DDPG, which could create a more robust and efficient algorithm for continuous control tasks.

This research can make a contribution to a better understanding of RL algorithm selection in continuous control applications and provide additional theoretical and empirical support for practical applications. Researchers and related staff can make more informed decisions when choosing and designing RL algorithms for specific tasks.

References

1. M.A. Wiering and M. Van Otterlo, 'Reinforcement learning', Adapt. Learn. Optim., vol. 12, no. 3, pp. 729, 2012.
2. J. Garcia and F. Fernández, 'Safe exploration of state and action spaces in reinforcement learning', J. Artif. Intell. Res., vol. 45, pp. 515–564, 2012.
3. V. François-Lavet, P. Henderson, R. Islam, et al., 'An introduction to deep reinforcement learning', Found. Trends Mach. Learn., vol. 11, no. 3-4, pp. 219–354, 2018.
4. T. Zhang and H. Mo, 'Reinforcement learning for robot research: A comprehensive review and open issues', Int. J. Adv. Robot. Syst., vol. 18, no. 3, pp. 1–20, 2021.
5. V. Mnih, K. Kavukcuoglu, D. Silver, et al., 'Human-level control through deep reinforcement learning', Nature, vol. 518, no. 7540, pp. 529–533, 2015.
6. T.P. Lillicrap, J.J. Hunt, A. Pritzel, et al., 'Continuous control with deep reinforcement learning', arXiv preprint, arXiv:1509.02971, 2015.
7. J. Schulman, F. Wolski, P. Dhariwal, et al., 'Proximal policy optimization algorithms', arXiv preprint, arXiv:1707.06347, 2017.
8. A. Awasthi, 'Evaluating reinforcement learning algorithms for LunarLander-v2: A comparative analysis of DQN, DDQN, DDPG, and PPO', 2025.
9. S. Zhu, S. Liu, S. Feng, et al., 'An optimization method for the inverted pendulum problem based on deep reinforcement learning', in Proc. J. Phys.: Conf. Ser., 2022, vol. 2296, no. 1, pp. 012008.
10. Y. Peng, G. Chen, M. Zhang, et al., 'Proximal evolutionary strategy: Improving deep reinforcement learning through evolutionary policy optimization', Memetic Comput., vol. 16, no. 3, pp. 445–466, 2024.
11. Y. Wang, H. He, and X. Tan, 'Truly proximal policy optimization', in Proc. Uncertainty Artif. Intell., 2020, pp. 113–122.

12. Z. Fan, R. Su, W. Zhang, et al., 'Hybrid actor-critic reinforcement learning in parameterized action space', arXiv preprint, arXiv:1903.01344, 2019.
13. M. Sewak and M. Sewak, 'Deterministic policy gradient and the DDPG: Deterministic-policy-gradient-based approaches', in Deep Reinforcement Learning: Frontiers of Artificial Intelligence, Springer, 2019, pp. 173–184.
14. E.H.H. Sumiea, S.J. Abdulkadir, M.G. Ragab, et al., 'Enhanced deep deterministic policy gradient algorithm using grey wolf optimizer for continuous control tasks', IEEE Access, vol. 11, pp. 139771–139784, 2023.
15. J. Zhu, F. Wu, and J. Zhao, 'An overview of the action space for deep reinforcement learning', in Proc. Int. Conf. Algorithms, Comput. Artif. Intell., 2021, pp. 1–10.
16. U. Demircioğlu, H. Bakir, and R. Bakir, 'An investigation of pendulum control using reinforcement learning: Comparison of different agents', 2024.