Exploring Deep Deterministic Policy Gradient (DDPG) : A Case Study in the Continous Mountain Car Environment

MEMUDU Alimatou Sadia
Msc Data Science and Artificial Intelligence
University of Cote d'Azur

Lee Hyelim
Msc Data Science and Artificial Intelligence
University of Cote d'Azur

***Abstract*—This report presents an investigation into the application of the Deep Deterministic Policy Gradient (DDPG) algorithm in the context of the continuous mountain car environment. The aim of this study is to explore the effectiveness of DDPG in solving this challenging reinforcement learning problem. Through implementation and experimentation, various hyperparameters were tuned, and their impacts on the behavior of the agent, total rewards achieved, as well as the losses incurred by the critic and actor networks were analyzed. The findings provide insights into the sensitivity of DDPG to different hyperparameters and their influence on the performance of the agent in the continuous mountain car environment.**

***Index Terms*—DDPG, Continous action space, mountain car, reinforcement learning, DPG, DQN.**

I. Introduction

Reinforcement Learning (RL) algorithms have demonstrated remarkable capabilities in enabling agents to learn optimal behaviors through interaction with their environments. Among these algorithms, Deep Deterministic Policy Gradient (DDPG) has emerged as a powerful technique for addressing tasks with continuous action spaces, offering promising solutions to challenging problems such as the continuous mountain car environment. Introduced by Timothy P. Lillicrap et al. [2], researchers from Google Deepmind, DDPG combines key elements from two prominent RL approaches: Deep Q-Networks (DQN) and Deterministic Policy Gradient (DPG).

DDPG leverages the insights from DQN's value-based methods, such as experience replay and target networks, while incorporating the policy-based approach of DPG to learn deterministic policies for continuous action spaces. By combining these techniques, DDPG enables agents to efficiently learn optimal policies in environments with continuous action spaces, such as pendulum , MovingGriper environment as shown in this paper [2]. However in this study , we would investigate the efficacy of the DDPG algorithm in the continuous mountain car environment, where a car must navigate a hilly terrain by applying continuous actions, aiming to reach the mountain top.

By implementing the DDPG algorithm and conducting thorough experimentation, we aim to understand the impact of various hyperparameters on the behavior and performance of the agent, shedding light on the factors influencing the effectiveness of DDPG in this challenging RL task.

II. Continuous Mountain Car Environment

The continuous mountain car environment is deterministic Markov Decision Process (MDP) also a classic problem in reinforcement learning introduced in 1990 by Andrew William Moore [4]. This environment is designed to test the ability of agents to navigate through a challenging terrain with continuous states and actions. The task of the agent in this environment typically involves controlling a car to reach the top of a hill (mountain) by applying continuous actions.

Here's a breakdown of the states, actions, and rewards in this environment:

- **States**: The state space typically consists of two continuous variables. The state s is represented as a tuple $(x, \dot{x})$.
  - Position (x): The position of the car along the horizontal axis. It can range from -1.2 to 0.6.
  - Velocity ($\dot{x}$): The velocity of the car. It can range from -0.07 to 0.07.
- **Actions**: The action space is also continuous, allowing the agent to apply a continuous force to the car. The action a is represented as a single real-valued number representing the acceleration or deceleration of the car where:
  - -1.0: Maximum deceleration, causing the car to accelerate in the opposite direction of its current motion.
  - 0.0: No acceleration, allowing the car to coast without any applied force.
  - 1.0: Maximum acceleration, causing the car to accelerate in the direction of its current motion.
- **Rewards**:The agent receives a negative reward of -1 for each time step until the goal state is reached. Upon reaching the goal state, the agent receives a reward of 0.

The objective of the agent in the continuous mountain car environment is to learn a policy that maximizes the cumulative reward over time by efficiently reaching the goal state (top of the hill). This challenging task demands a delicate balance between exploration (trying different actions) and exploitation (leveraging learned policies) to navigate through a continuous state space and action space while conserving energy.

III. Deep Deterministic Policy Gradient (DDPG)

DDPG is a model-free, off-policy actor-critic algorithm [1] designed to tackle tasks with continuous action spaces. It combines elements from both value-based (Deep Q-learning (DQN) [3]) and policy-based (Deterministic Policy Gradient (DPG) [5]) methods to effectively learn optimal policies.

A. The evolution of Deep Deterministic Policy Gradient

The evolution of DDPG stems from a lineage of reinforcement learning algorithms as illustrated in figure 1. Initially, Reinforce, introduced by Ronald J. Williams in 1992 [6], laid the groundwork for policy gradient methods by directly optimizing the policy parameters to maximize expected rewards. Actor-Critic methods, such as those pioneered by Barto et al. [1] , extended Reinforce by incorporating value function estimation to guide policy updates, laying the baseline for DPG algorithms. While Actor-Critic methods addressed some challenges, they still lacked stability and sample efficiency in learning. Subsequently, Deep Q-Networks (DQN) an extension of Q-learning [3] demonstrated significant advancements in RL by efficiently handling discrete action spaces using deep neural networks . However, DQN's applicability was restricted to discrete spaces, necessitating solutions for continuous action domains. Deterministic Policy Gradient (DPG) algorithms addressed this challenge by focusing on learning deterministic policies, offering a solution for continuous action spaces [5]. Nonetheless, DPG methods lacked the sample efficiency observed in DQN. Combining the strengths of both approaches, Timothy P. Lillicrap et al. [2] proposed DDPG. By integrating the deep neural network architecture of DQN with the deterministic policy gradient theorem DPG, DDPG effectively tackled the challenges associated with learning in continuous action spaces.

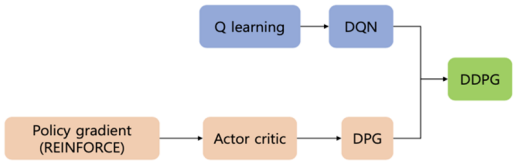

Fig. 1. Origin of Deep Deterministic Policy Gradient (DDPG) source

Below, we provide an explicit explanation of the core components of the DDPG algorithm 3 and their respective roles:

B. Actor Network

The actor network is responsible for learning a deterministic policy that maps states to actions.It takes the current state as input and outputs a deterministic action, representing the agent's policy. By updating the parameters of the actor network, DDPG aims to maximize the expected cumulative reward over time.

C. Critic Network

The critic network evaluates the actions suggested by the actor by estimating the Q-value of the current state-action pair. It takes both the current state and the action chosen by the actor as input and outputs the estimated Q-value. By learning the critic network, DDPG aims to approximate the optimal action-value function, which represents the expected return from taking a particular action in a given state.

The DPG algorithm maintains a parameterized actor function $\mu(s|\theta^{\mu})$ which specifies the current policy by deterministically mapping states to a specific action. The critic $Q(s,a)$ is learned using the Bellman equation as in Q-learning. The actor is updated by following the applying the chain rule to the expected return from the start distribution J with respect to the actor parameters

$$\begin{aligned}\nabla_{\theta^{\mu}} J &\approx E_{s_t\sim\rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s,a|\theta^{Q})|_{s=s_t,a=\mu(s_t|\theta^{\mu})}] \\ &= E_{s_t\sim\rho^{\beta}}[\nabla_a Q(s,a|\theta^{Q})|_{s=s_t,a=\mu(s_t)} \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})|_{s=s_t}]\end{aligned}$$

D. Replay Buffer

Replay Buffer is a key technique used in DDPG to improve sample efficiency and stabilize training. It involves storing experiences (state, action, reward, next state) in a replay buffer during interaction with the environment. During training, experiences are sampled from the replay buffer to update the critic and actor networks, breaking temporal correlations and reducing the impact of sequential data.

E. Target Networks

Target networks are slow-moving copies of the actor and critic networks used to provide more stable target values during training. They are periodically updated with the parameters of the main networks, typically through a soft update mechanism as shown below to reduce the likelihood of divergence and improves the stability of the learning process.

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \quad with\ \tau \ll 1.$$

It greatly improves the stability of learning.

F. The Ornstein-Uhlenbeck Exploration Noise

This noise is added to encourage exploration in the action space. It prevents the agent from getting stuck in local optima and facilitates the discovery of potentially better policies. The Ornstein-Uhlenbeck process is defined by the following stochastic differential equation:

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t$$

Here, dx_t represents the change in the state at time t, μ is the mean, θ is the rate of mean reversion, σ is the volatility parameter, dt is an infinitesimally small time interval, and dW_t is a Weiner process.

To incorporate this into the formula for $\mu'(s_t)$, we add the noise term $\mathcal{N}$, which is drawn from the Ornstein-Uhlenbeck process, to the output of the actor network $\mu(s_t|\theta_t^{\mu})$:

$$\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N}$$

Therefore, the noise term $\mathcal{N}$ at time t can be expressed as $(\mu - x_t) + \theta(\mu - x_t) + \sigma dW_t$.

With the provided code snippet, the Ornstein-Uhlenbeck noise process is implemented to generate the noise term $\mathcal{N}$ for exploration.

By combining these techniques, DDPG algorithm enables agents to efficiently learn optimal policies in environments with continuous action spaces, such as the continuous mountain car environment as displayed in the figure 2.

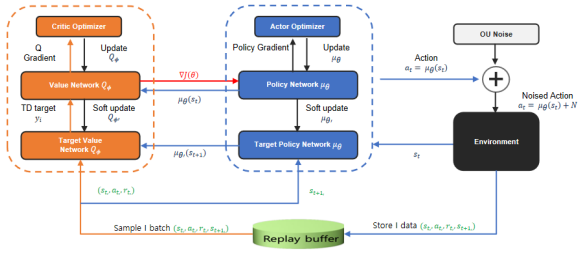

Fig. 2. Interaction between DDPG core components.
source

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. 3. Algorithm of Deep Deterministic Policy Gradient (DDPG) [2]

IV. Experiments

The aim of this study is to investigate the impact of the tuning some hyperparameter of the DDPG algorithm on the performance of the agent in a continuous mountain car environment. Specifically, we focus on the following hyperparameters :

- **Ornstein-Uhlenbeck Noise Parameters**:
  - **theta**: determines the speed of mean reversion in the Ornstein-Uhlenbeck process. Higher theta value leads to faster return to the mean and a lower value allows for prolonged exploration.
  - **sigma**: controls the intensity or amplitude of the noise added to the actions. Higher sigma value leads to larger fluctuations in actions and a lower value restricts action variability, promoting conservative exploration.
- **Discount Factor (Gamma)**: determines the importance of future rewards relative to immediate rewards. It influences the agent's tendency to favor long-term rewards over short-term gains.
- **Soft Target Update Rate (Tau)**: indicates the rate at which the target network parameters are updated towards the main network parameters. This parameter controls the degree of stability in learning.

The baseline code for this project is a Jupyter notebook provided by an AI engineer, where DDPG algorithm is implemented on a Pendulum environment, as demonstrated here. We adapted this code to suit our specific study case which is the continuous mountain car, extensively modifying the agent class and fine-tuning hyperparameters code section. The source code of this study's implementation was programmed in python using Pytorch framework and available on Google Colab here.

V. Results

Our results include plots depicting the total reward, as well as the actor and critic loss curves.By monitoring the trends and convergence of the critic and actor loss over the episodes, we assess the learning progress of the DDPG algorithm.

A. Hyper parameter set 1

The optimal performance is obtained with the following parameters: theta = 0.15, sigma = 0.3, gamma = 0.95, tau = 0.01. The total rewards is 94.94 and the agent reaches the goal within 4 seconds. The agent's behavior is initially seen as moving in the opposite direction of the goal (deceleration), gaining momentum, and applying momentum and accelerate on the car velocity to reach the hill of the mountain.

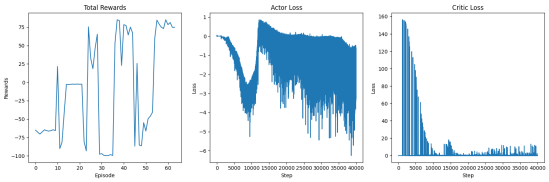

Fig. 4. The result with the best hyper parameters

B. Hyper parameter set 2

The second best performance is yielded when the parameters are: theta = 0.15, sigma = 0.1, gamma = 0.95, tau = 0.005. The total rewards is 93.65 and the agent reaches the hill of the mountain within 2 seconds. Similar behavior to parameter set 1 above appears, but exploration is taken less than set 1, so the time to reach the goal appears to be reduced.

Both cases of good performance have in common that the same theta(0.15) and gamma(0.95) were applied.

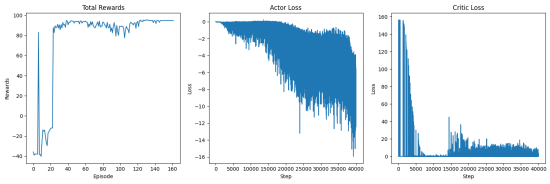

Fig. 5. The result with the second best hyper parameters

C. Hyper parameter set 3

The worst performance is yielded when the parameters are: theta = 0.15, sigma = 0.3, gamma = 0.05, tau = 0.005. The total rewards is -99.89. When the agent goes from top to bottom, it decelerates rather than accelerates, like a movement that seems to forcibly prevent it from reaching the goal. The agent was unable to accelerate to reach the hill of the mountain.

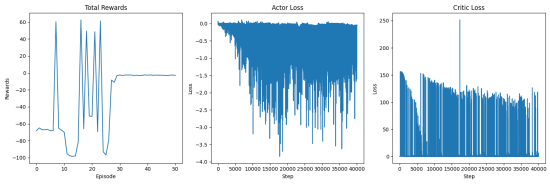

Fig. 6. The result with the worst hyper parameters

D. Hyper parameter set 4

The another bad case is yielded with the following parameters are: theta = 0.25, sigma = 0.3, gamma = 0.95, tau = 0.01. The total rewards is -0.22. when the reward is close to 0, the agent is oscillating with little movement back and forth, and not gaining enough momentum (velocity) to reach its goal.

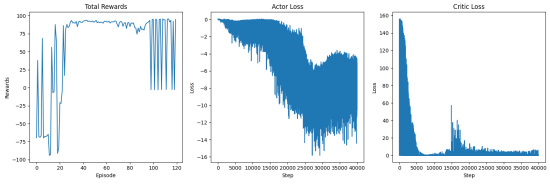

Fig. 7. The result with the bad hyper parameters

With these results, we derived that some specific hyper parameters can contribute to performance of DDPG algorithm in this environment. Since parameters, rewards do not have a linear correlation, hyperparameter tuning through gridsearch or Bayesian optimization is essential when applying DDPG. Moreover , in all cases, the actor loss decreases which indicate that the actor is improving in selecting actions that lead to higher rewards. On the other hand, critic loss also decreases which indicate the critic network is getting better at predicting the expected future rewards given a state-action pair.

VI. Conclusion

Our investigation into applying DDPG in the continuous Mountain Car environment reveals its effectiveness in tackling challenging reinforcement learning tasks. By merging features from DQN and DPG , DDPG learns optimal policies in continuous action spaces. Its key features include Actor-Critic networks, replay buffer, soft target update, and exploration using OU-noise. Leveraging deep neural network architecture and the deterministic policy gradient theorem, DDPG effectively addresses challenges in learning within continuous action spaces, providing a comprehensive framework for agents to excel in dynamic environments.

Through experimentation and hyperparameter tuning, we have observed the impact of parameters such as theta, sigma, gamma, and tau on the behavior and performance of the agent. By monitoring the trends in total rewards, actor and critic loss curves, we have gained a deeper understanding of how these hyperparameters influence the learning progress of the DDPG algorithm.

Our results show the adaptability of DDPG in navigating the complex terrain of the Mountain Car environment, with combination of specific theta and gamma values contributing to enhanced performance. By shedding light on the sensitivity of DDPG to different hyperparameters, we have laid the groundwork for further exploration and optimization of this algorithm in challenging RL tasks.

References

[1] Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics **SMC-13**(5), 834–846 (1983). https://doi.org/10.1109/TSMC.1983.6313077

[2] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–33 (02 2015). https://doi.org/10.1038/nature14236

[4] Moore, A.W.: Efficient memory-based learning for robot control (1990), https://api.semanticscholar.org/CorpusID:60851166

[5] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. 31st International Conference on Machine Learning, ICML 2014 **1** (06 2014)

[6] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**, 229–256 (1992). https://doi.org/10.1007/BF00992696, https://doi.org/10.1007/BF00992696