Application of Reinforcement Learning in Games

Lunyuan Hu[1][a,*], Ruike Peng[2][b] and Junpeng Yang[3][c]
[1]Finance, Southwestern University of Finance and Economics, No.555, Liutai Avenue, Chengdu, China
[2]Faculty of Data Science, City University of Macau, Xu Risheng Yin Road in Taipa, Macau, China
[3]Data science of Qingdao University, Qingdao University, Hong Kong Middle Road in Qingdao, Qingdao, China

Keywords: Reinforcement Learning, Q-Learning, Policy Gradient, Actor-Critic.

Abstract: With the advancement of technology, non-player characters (NPCs) can respond to players' behaviors more intelligently, thereby enhancing the fun and challenge of games. In this regard, reinforcement learning is an algorithm suitable for training agents and improving the playability of games. This paper explores the innovation of artificial intelligence in the field of games, focusing on the application of reinforcement learning algorithms such as Q-Learning, policy gradient, and Actor-Critic in game development and deeply analyzes the practical application of reinforcement learning in classic games and the challenges it faces, such as overfitting problems, and explores the prospects for future algorithm improvements. Through reinforcement learning, game content and gameplay can be enriched and optimized, so that characters and environments can present more realistic and natural behavior patterns. In the future, with the continuous improvement of hardware performance and algorithms, Q-learning, policy gradient and Actor-Critic are expected to achieve personalization and dynamic adjustment in more complex game environments, promoting the continuous innovation and development of the game industry.

1 INTRODUCTION

With the swift advancements in internet technology, Artificial Intelligence (AI) has gradually infiltrated into all industries, now assuming a pivotal role in people's daily lives and emerging as a formidable force propelling the progression of industrial innovation. At present, the game industry is developing rapidly and becoming an indispensable part of people's life, especially for teenagers, so it is no surprise that the game industry has also ushered in a profound change caused by artificial skills technology. Since the birth of the game, has gone through many changes, each change to the user to bring a new experience, enrich the daily life of people. Today, games have evolved from simple electronic pixel entertainment into interactive, immersive, and life-of-the-plot Entertainment, and the introduction of artificial intelligence technology, let the game have more rapid development, not only greatly enriching the game's content and play, but also for players to bring an unprecedented game experience, so that players have more options.

As a comprehensive discipline, artificial intelligence encompasses a broad spectrum of fields, encompassing computer science, control theory, information theory, neurophysiology, psychology, linguistics, and numerous others. Its development has gone through many stages, from simple decision-making based on rules, to complex strategy based on machine learning, to intelligent self-adaptation based on deep learning. Significant progress in reinforcement learning has been achieved in recent years, fueled by the rapid advancements in technologies such as big data, cloud computing, and deep learning. At the same time, the application of reinforcement learning technology in the gaming industry has become increasingly widespread, bringing many new development opportunities and challenges to the gaming industry. The application of reinforcement learning technology has greatly enriched the content and play method of the game, widened the development idea, design method and

[a] https://orcid.org/0009-0001-9878-7889
[b] https://orcid.org/0009-0004-8786-3729
[c] https://orcid.org/0009-0006-7214-8022

Hu, L., Peng, R. and Yang, J.
Application of Reinforcement Learning in Games.
DOI: 10.5220/0013234900004558

In *Proceedings of the 1st International Conference on Modern Logistics and Supply Chain Management (MLSCM 2024)*, pages 129-134
ISBN: 978-989-758-738-2

development method of the game, and greatly improved the user experience. The further application of reinforcement learning technology in various types of games has made the interaction between non-player characters (npcs) and players more diverse, make the role and the environment in the game can present a more real, natural and intelligent behavior pattern, enriched the game play and plot. Non-player characters (npcs) are no longer simply agents that perform the actions of the developer. Instead, npcs are agents that react to the choices and actions of the player. Not only does this change enhance the game's interest and challenge, but it also facilitates players to encounter more authentic human interaction and emotional resonance within the game.

Therefore, in-depth learning and exploration of the relevant applications of reinforcement learning technology in games not only helps us better understand the nature and characteristics of this cutting-edge technology, also can provide the powerful support and the impetus for the game industry innovation and the development. This article will analyze and expound the application of reinforcement learning technology in games from many angles, in order to provide useful reference and enlightenment for scholars and practitioners in related fields. The purpose of this paper is to discuss the application status, principle, method and future development trend of the game field of reinforcement learning technology. This article introduces some mainstream reinforcement learning algorithms such as Q-learning, policy gradient and their application in real-world development cases, to introduce the impact of reinforcement learning technology on the game field and change. The challenges and problems of reinforcement learning are discussed, and the future trend of artificial intelligence and reinforcement learning is predicted.

2 Q-LEANING

2.1 Q-learning Principle

Q-learning is a reinforcement learning algorithm, which belongs to the model-free, off-policy and value-based learning methods. It uses the Bellman equation for iterative updates and learns the optimal policy by interacting with the environment. Its goal is to maximize the expected cumulative return by estimating the action value function Q (s, a) to select the optimal action. In practical applications, neural networks are often used to approximate the Q-function, such as Deep Q - Network (DQN). The main elements include：1. State (State, S): Describes the situation or position of the agent in the environment. 2. Action (Action, A): Possible operations that the agent can perform in a specific state. 3. Reward (Reward, R): Immediate feedback given by the environment to the agent to evaluate the quality of the action. 4. Q-value: Represents the sum of expected future rewards after performing an action in a certain state. 5. Policy: Determines the action that the agent should take in a given state.

2.2 The Application of Q-learning in Games

Q-learning, as a model-free reinforcement learning method, plays a significant role in games. The following are the application processes and principles of it in three games: Snake, Flappy Bird, and T-Rex Runner.

2.2.1 Snake Game

State Representation: Covers information such as the position coordinates of the snake, the position coordinates of the food, the direction of the snake's head, and the length of the snake's body. Action Selection: There are only four directions of movement: up, down, left, and right. Q-learning Implementation: Firstly, initialize the Q-table, and all Q-values are zero. Use the ϵ - greedy strategy to select actions to strike a balance between exploring new actions and exploiting known high-value actions. After performing the selected action, receive the reward and new state feedback from the environment, and update the Q-values according to the Q-learning formula. Through multiple iterations of training, the snake can gradually learn the optimal movement strategy. For example, in the paper by Yuhang Pan et al. (Pan, 2023), DQN was used to play the Snake game. The target Q value was calculated using $yt = rt+1 + \gamma \max Q (st+1, a; \theta -)$, and the neural network parameters were updated by minimizing the network function $L(\theta) = 1/2 (yt - Q(st, at; \theta))2$. Effective results were obtained and it was proved that DQN performed better than PPO in the Snake game.

2.2.2 Flappy Bird Game

State Representation: Defined by pixel input and the actions taken (rising, natural falling). Q-learning Implementation: Adopt DQN to approximate the Q-function. Its structure includes multiple convolutional layers, pooling layers, and fully connected layers. For action selection, the ε - greedy strategy is used, and

the exploration probability ε gradually decreases as the training progresses. After performing the action, calculate the reward based on the state of the bird, update the Q-values using the Q-learning formula, and adjust the network weights through backpropagation.

2.2.3 T-Rex Runner Game

Environment Setup: Preprocess the game screen image, such as converting RGB to grayscale, removing irrelevant objects, erosion, and dilation operations, etc., and stack multiple frames of historical images as input. Q-learning Implementation: Initialize the weights of the policy network to random values and set relevant parameters. At the same time, set the target network and initialize it the same as the policy network. In the training loop, select actions according to the ϵ - greedy strategy. After performing the action, store the transitions in the experience replay pool. Sample from the replay pool, use the target network to calculate the expected return to optimize the weights of the policy network, and update the target network when reaching a certain number of steps. For example, in Yue Zheng's paper (Zheng, 2019). The original image undergoes a series of preprocessing operations, including RGB to grayscale conversion, removal of irrelevant objects, erosion and dilation operations to reduce noise, and finally is resized to 84×84. To capture the movement of the dinosaur, the last four frames of historical images undergo the same preprocessing and are stacked into a data point as the input. Overall, in these three games, Q-learning continuously optimizes the decision-making strategy of the agent by relying on the reward signal and the learning of the state-action value. The integration of deep learning techniques, especially DQN, provides strong support for handling complex game inputs and decision spaces.

2.3 Problems and Optimization Strategies

2.3.1 Experience Replay

Experience replay is a technique used in reinforcement learning to improve training efficiency and stability. In traditional Q-learning, experiences of consecutive frames are often highly correlated, which can hinder the training process and lead to inefficient training. The purpose of experience replay is to solve this problem by decorrelating these experiences to improve the training effect. Specifically, in experience replay, the experience (s, a, r, s') is stored in the replay memory at each frame. The replay memory has a certain size and saves some recent experiences. It updates continuously like a queue to ensure that the experiences in the memory are related to the recent actions and the corresponding Q functions. When it is necessary to update the Deep Q Network (DQN), instead of using the current consecutive experiences, a batch of experiences is uniformly sampled from the replay memory. The advantage of this is that the sampled experiences are no longer highly correlated, making the training more stable and efficient. Through experience replay, the model can better learn from diverse historical experiences, avoid being misled by recent consecutive correlated experiences, and help converge to a better policy faster.

For example, in the paper by Kvein Chen (Chen, 2015), during the training of Flappy Bird, he found that the traditional Q-learning method had a strong correlation, resulting in biased training results. Therefore, he used the experience replay method in the training process and updated the DQN, eventually obtaining weakly correlated results.

2.3.2 Target Network

The target network (Target Network) is a mechanism introduced in reinforcement learning, especially when using algorithms such as Q-learning, to increase training stability. During the training process of Q-learning, a neural network (such as the Deep Q Network DQN) is usually used to approximate the Q function. When updating the Q value, it is necessary to calculate the target value $y_i = E_{s\sim\varepsilon}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a]$which involves the evaluation of all possible actions (a') under the current state (s') by the Q function and selects the maximum value. However, if the same network is directly used to calculate both the current Q value and the target value simultaneously, it may lead to training instability because the parameters of the network are constantly updated and the target value will also fluctuate, making training difficult to converge. The target network ({Q}(s, a)) is introduced to tackle with the problem. The parameters of the target network may be different from the Q network although they have the structure. During the training process, the parameters of the target network are updated to synchronize with the parameters of the current DQN only after the DQN is updated a certain number of times (such as C times). The parameter update approach of the target network is as follows: During the training process, when the DQN is updated every

C time, the parameters of the target network will be updated. Specifically, the parameters of the target network will be updated to the parameters of the current DQN. This update method can make the target network relatively stable for a period of time, thereby providing a more stable reference for calculating the target value, helping to reduce training fluctuations, and improve the stability and convergence speed of training.

3 POLICY GRADIENT

3.1 Principle of Policy Gradient

The goal of Q-Learning and Actor-Critic methods is to learn the optimal value function in order to directly or indirectly select actions to maximize long-term rewards. Policy gradient is a reinforcement learning algorithm that learns a policy probability density function. It learns from policy functions to maximize expected long-term returns.

When the policy is usually expressed as a parameterized function $\pi_\theta(a|s)$, where θ is the policy parameter, s express state, and a express action. The policy function outputs the selecting probability each possible action in a given state. The policy gradient method aim to maximize the expected long-term return $J(\theta)$, which is usually defined as: $J(\theta)=E\tau\sim \pi_\theta[\Sigma\, t\infty=0\ \gamma_t r_t]$ where $\tau=(s_0,a_0,r_0,s_1,a_1,r_1\ldots)$ is a sample path, r_t is the reward at time step t, and γ is the discount factor.

3.2 Application of Policy Gradient in Game

With the development of artificial intelligence technology, the MARL application in complex game environments has attracted increasing attention. In their paper 'Multiagent Simulation on Hide and Seek Games Using Policy Gradient Trust Region Policy Optimization', Hani'ah Wafa and Judhi Santoso (2020) used the TRPO algorithm to optimize the agent's strategy (Hani'ah and Judhi, 2020). The article describes the system architecture and neural network architecture in detail, pointing out that the agents use the same strategy to make decisions in the same team. Through training, the agent's behavior goes through multiple stages, from initial exploration to interaction with objects in the environment, and finally forms a more complex strategy. In the "Hide and Seek" game environment, each game round consists of 80 time steps, of which the preparation phase is 0.4 time steps, during which the hider can choose to hide or manipulate the environment to avoid being discovered by the seeker. The agent can perform three types of actions: moving to another location, moving the location of an object, and locking the location of an object. The hider will receive a 10-point reward when he is not discovered, and the seeker will also receive a 10-point reward when he finds the hider. In addition, the game environment has different scene configurations, such as quadrant scenes and random wall scenes, which affect the initial position of the hider and the number of rooms. This study provides a new perspective for MARL methods in the game field and shows the potential for efficient learning in complex game scenarios. Pengcheng Dai et al. explored the impact of strategy complexity on the learning effect of agents in their paper 'Distributed Actor–Critic Algorithms for Multiagent Reinforcement Learning Over Directed Graphs' (Dai, et al. 2023). Malloy et al's CLDAC encourages agents to learn simpler strategies through information theoretic constraints to reduce the risk of overfitting (Malloy et al. 2021). Pengcheng Dai et al. used distributed computing to optimize strategies to enable agents to perform better in complex environments (Dai, et al. 2023).

3.3 Future Directions and Challenges of Policy Gradient Algorithms in Game Applications

For policy gradient, how to effectively coordinate and cooperate in a multi-agent environment is an important challenge. In future research, the MARL field can further explore and expand existing methods so that agents can better collaborate in dynamic environments. Malloy et al. mentioned extending CLDAC to more complex MARL structures (Malloy et al. 2021), while Pengcheng Dai et al. mentioned that after combining the advantages of distributed computing, the research aims to develop a policy gradient algorithm that can operate effectively in a distributed environment, so that agents can learn and optimize strategies through local information exchange without central control (Dai et al. 2023). Hani'ah Wafa and Judhi Santoso and Pengcheng Dai et al. both mentioned that one of the main challenges facing multi-agent systems is the "curse of dimensionality", that is, as the number of agents increases, the complexity of the system increases significantly (Wafa, Santoso, 2020). The behavior and decision of each agent will increase the variables of the system, making the overall learning process more complicated.

4 ACTOR-CRITIC

4.1 Overview of Actor-Critic Algorithms

Actor-critic algorithm is widely used in games. It combines strategy gradient and value function learning, and combines the advantages of the two algorithms, through the cooperation of actor neural network and critic neural network, the complex strategy and strategy optimization problems in the game are effectively solved. In actor critic algorithms, strategy is a function that selects the next action based on the current operating environment state. The Actor part is responsible for learning the strategy, which is a parameterized strategy function π (a | S) that takes the states as input, and output in a given state to take each action a probability distribution or specific action (in the deterministic strategy). The goal of an Actor is to find a strategy that maximizes the cumulative reward, or Return. The value function is used to estimate the expected cumulative reward (or expected reward) for taking an action in a given state. The Critic section is responsible for learning the value function, which is usually a parameterized function, such as a neural network, that takes a state and (in some cases) an action as input and outputs an estimate of the value of that state or state-action pair. This value estimate is used to guide Actor on how to update their strategies. The Actor section updates its policy parameters using the policy gradient method. The strategy gradient method is based on the principle of gradient rise (for the maximization problem) by calculating the gradient of the cumulative reward on the strategy parameters and updating the parameters along the gradient direction, to make the strategy more likely to generate high cumulative rewards in the future.

4.2 Application of Actor-Critic Algorithm

To use Actor-critic algorithm into these games, Di Yuan; Zirui Luo; Xinyi Yao; Jing Xue describes and analyzes how to put Actor-Critic algorithm in use in the Snake Game in detail in the article 'a Multi-Agent Actor-Critic Based Approach Applied to the Snake Game'. This algorithm is not only suitable for the shared reward settings, but also for the personalized settings where the agent can not understand the global state. Nowadays, it has been widely accepted that game simulation environment is the research and verification environment of artificial intelligence. In this paper, the Snake Game real-time combat platform is chosen as the research environment of reinforcement learning algorithm, and the performance of Q-learning algorithm and actor-critic algorithm is compared. In order to improve the actor-critic algorithm, a better experience playback method is introduced and its performance in this environment is compared.

4.3 Actor-Critic Algorithm's Problems and Solutions

Although Actor-critic algorithm is widely used now, it still has many shortcomings, such as: because actor network and critic Network train at the same time, therefore, it leads to poor training stability, over-sensitivity to the influence of super-parameters to find the optimal solution, and higher cost of collecting sample data, when dealing with large-scale state space and complex environment, the sample utilization rate is low, and the actor network and critic network are used simultaneously, the paper introduces the target network to calculate the loss of critic network to reduce the instability, using more advanced optimizer, adjusting the contribution of samples according to different weights, experience playback and introducing neural network to optimize, to some extent, the efficiency and utilization of the algorithm are improved.

5 CONCLUSION

The paper explores the application of reinforcement learning in games, including Q-learning, policy gradient, and Actor-Critic algorithms. Q-learning learns the optimal policy through interaction with the environment and has specific applications in games such as Snake, Flappy Bird, and T-Rex Runner. Meanwhile, it faces problems such as experience correlation and training stability, which can be optimized through strategies such as experience replay, target networks, and Double Q-learning. The policy gradient directly learns the policy function to maximize the long-term return and has important applications in multi-agent reinforcement learning, but faces challenges such as coordination and cooperation and the "curse of dimensionality" in the multi-agent environment. The Actor-Critic algorithm combines policy gradient and value function learning and has wide applications in games, but there are problems such as poor training stability, which can be solved by introducing target networks and other methods. Q-learning, policy gradient, and Actor-

Critic each have their unique application scenarios and methodologies in reinforcement learning. Choosing the appropriate method usually depends on the characteristics of the problem, such as the type of action space and maximizing the long-term return.

In the future, with the improvement of hardware performance and algorithm improvement, reinforcement learning can be applied in more complex and computationally intensive game environments. Reinforcement learning will be able to handle more complex and realistic environments, such as real-time strategy games and VR games. And game developers can create more challenging and personalized game experiences based on agents. This personalized experience can be dynamically adjusted according to the player's skill level and preferences to enhance the attractiveness and playability of the game. Although the generalization ability of current reinforcement learning algorithms outside the training environment is still limited, future research can focus on improving the generalization ability of the algorithm to allow agents to perform better in diverse game environments.

AUTHORS CONTRIBUTION

All the authors contributed equally and their names were listed in alphabetical order.

REFERENCES

Pan, Y., Yi, S., Ma, Q., Bowen, G., Junyi, D., & Zijun, T. 2023. Playing the Snake Game with Reinforcement Learning. Cambridge Explorations in Arts and Sciences.

Zheng, Y. 2019. Reinforcement Learning and Video Games. ArXiv, abs/1909.04751.

Chen, K. 2015. Deep Reinforcement Learning for Flappy Bird.

Wafa, H., & Santoso, J. 2020. Multiagent simulation on hide and seek games using policy gradient trust region policy optimization. In *Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality* (pp. 1-6). IEEE.

Dai, P., Yu, W., Wang, H., & Baldi, S. 2023. Distributed actor–critic algorithms for multiagent reinforcement learning over directed graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10), 7210-7221.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 6382–6393).

Yuan, D., Luo, Z., Yao, X., and Xue, J. 2023. A Multi-Agent Actor-Critic Based Approach Applied to the Snake Gam. 2023 42nd Chinese Control Conference (CCC), Tianjin, China, 2023, pp. 8356-8361, doi: 10.23919/CCC58697.2023.10241182.

Fachada, N., Barreiros, F. F., Lopes, P., and Fonseca, M. 2023. Active Learning Prototypes for Teaching Game AI. 2023 IEEE Conference on Games (CoG), Boston, MA, USA, 2023, pp. 1-4.

Gao, T., Li, J., and Mi, Q. 2023. From Game AI to Metaverse AI: Realistic Simulation Training as a New Development Direction for Game Artificial Intelligence. 2023 International Conference on Emerging Techniques in Computational Intelligence (ICETCI), Hyderabad, India, 2023, pp. 130-137.

Xia, B., Ye, X., and Abuassba, A. O. M. 2020. Recent Research on AI in Games. 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020, pp. 505-510.