Improved Deep Reinforcement Learning Algorithms and Applications in Quantitative Trading in Extreme Stock Markets

Deguan Cui[1*], Ailin Deng[2], Zhengxun Xia[3], Liqi Zeng[1]

{deguan.cui, ailin.deng, zhengxun.xia, liqi.zeng}@transwarp.io

[1]Data Intelligence Department,Transwarp Technology(shanghai) Co.,ltd., Guangzhou,510665, China

[2]Data Intelligence Department,Transwarp Technology(shanghai) Co.,ltd., Shanghai,200233, China

[3] Big Data Research Institute,Transwarp Technology(shanghai) Co.,ltd., Nanjing,210000, China

Abstract: Aiming at the issue that existing deep reinforcement learning algorithms cannot achieve satisfactory returns in the quantitative trading process, particularly during extreme stock market conditions such as sharp declines, we propose an improved TD algorithm based on immediate reward R_t, which we name as RTD. We further extend RTD to multi-step conditions (MRTD) and apply it to enhance algorithms such as DQN and DDPG. We then utilize these two improved algorithms in the stock quantitative trading process. Experimental results demonstrate that our proposed algorithms can respond to market changes more efficiently, resulting in improved accuracy of investment strategies and higher investment return rates, even during extreme market conditions. For instance, when the market declines by more than 10%, and the two specified stocks decline by nearly 5.09% and 13.30%, we can still obtain return rates of 2.78% and 4.19% respectively, which confirms the effectiveness of our proposed algorithms.

Keywords: Deep reinforcement learning, TD algorithm, quantitative trading, extreme markets

1. INTRODUCTION

Since the beginning of 2022, many stock markets around the world have experienced sharp declines or violent fluctuations. Taking the Chinese stock market as an example, the A-share Shanghai Composite Index dropped from around 3,651 in early January to a low of 3,252 at the end of March, resulting in a drop of more than 10% in just three months, with a maximum drawdown of more than 17%, particularly in early March to mid-April. Similar conditions are prevalent in other stock markets, resulting in heavy losses for investors. To help people avoid such losses in extreme markets, quantitative trading is one of the key and promising means to solve the problem [1-3]. However, currently, there are few relevant studies on how to develop a quantitative trading model that can effectively handle such conditions [4-7].

Quantitative trading has always been a hot and challenging topic in financial time series data research, and in recent years, it has demonstrated satisfactory investment returns in stock markets [8-12]. With the rapid development of high-performance computing, big data, and

ICIDC 2023, June 02-04, Nanchang, People's Republic of China

DOI 10.4108/eai.2-6-2023.2334605

deep learning technologies over the past few years, reinforcement learning has also gained wider attention and faster development. The combination of reinforcement learning and deep learning has led to breakthroughs in the field of deep reinforcement learning, demonstrating strong learning and decision-making capabilities in various areas [13-18]. As a result, many researchers have introduced deep reinforcement learning methods into stock quantitative trading [17-21].

As early as 2000, scholars began applying reinforcement learning algorithms to quantitative stock trading[22-25]. Relevant research in recent years includes: In 2019, NN.Y.Vo et al.[26] proposed a Deep Response Portfolio (DRIP) model, which contains a multivariate bidirectional long-term short-term memory neural network for predicting stock returns. In the same year, Tianxin Liang et al.[27] reviewed the current researches of reinforcement learning models commonly used in the financial field, and analyzed the difficulties and challenges in the application of reinforcement learning technology in the financial field. Also in 2019, Yuefeng Cen et al.[28] introduced an near-end reinforcement learning method (PPO) into the stock price forecasting process. By 2020, Brando.I.V. et al.[29] proposed a DRL-based decision support framework for stock market, which identifies trades by learning trading rules. In the same year, Huang G et al.[30] proposed a portfolio optimization framework with a short-selling mechanism in the continuous action space. By adding a short-selling mechanism and designing an arbitrage mechanism, the existing depth of the reinforcement learning portfolio model has been improve. In 2021, Zhidong Liu et al.[31] first used a clustering algorithm to extract market state features in different time periods, and then used the Almgren-Chriss optimal transaction execution framework article to construct optimal transaction execution. The reinforcement learning model incorporates the market state feature vector into the input variable, and solves it through the Deep Deterministic Policy Gradient (DDPG) in reinforcement learning.

The literature shows that the main focus in recent years has been on building a suitable deep reinforcement learning framework to achieve quantitative trading. However, there is still a lack of effective research on how to improve investment returns and reduce losses in extreme stock market conditions. Addressing the problem of building a deep reinforcement learning framework and method that can avoid risks in sharp market declines, and provide stability and anti-risk abilities to the model, is a pressing issue that requires urgent attention. Additionally, existing reinforcement learning-based models lack a timely response mechanism for predicting stock market volatility during market turbulence. Therefore, improving the response efficiency of the models to market changes is another critical area that needs to be studied in-depth.

Therefore, this paper proposes a deep reinforcement learning framework for quantitative stock trading. Firstly, we introduce an improved TD algorithm and use it to enhance classic reinforcement learning algorithms such as DQN and DDPG. Subsequently, we utilize the improved DQN algorithm and the improved DDPG algorithm to guide the quantitative trading process. Experimental results show that the improved algorithm proposed in this paper can generate positive returns even in periods of stock market crashes, providing valuable insights for quantitative investment transactions.

2. TD ALGORITHM AND SERIES IMPROVED ALGORITHMS

In order to improve the return or reduce the loss of quantitative trading in extreme stock markets, this paper proposes a series improved algorithms and applies them to quantitative trading.

2.1 Improvement TD algorithm(RTD)

The TD algorithm is a basic algorithm in reinforcement learning. It enables the model to update the cumulative expected return in the current state using the results of one or several steps during the training process and guide the update of model parameters. The TD algorithm integrates Monte Carlo sampling and the idea of Bootstrapping in dynamic programming, which uses the value function of the next state to estimate the current value function. This integration allows for parameter updates in a single step and greatly improves training speed, thereby expanding the application scenarios of reinforcement learning. Many well-known reinforcement learning algorithms, such as DQN, AC, A2C, TRPO, PPO, and DDPG, are either improved or extended based on the TD algorithm.

The calculation formula of TD algorithm is as follows[32]:

$$\mathrm{V}(\mathrm{s_t}) \leftarrow \mathrm{V}(\mathrm{s_t}) + \alpha(\mathrm{R_t} + \gamma * \mathrm{V}(\mathrm{s_{t+1}}) - \mathrm{V}(\mathrm{s_t})) \qquad (1)$$

In the formula: $V(s_t)$ is the value function of state st , $V(s_{t+1})$ is the value function of the next state(s_{t+1}), $\mathrm{R_t}$ is the immediate reward obtained after performing the action in the St state, and γ is the return discount rate. $R_t + \gamma * V(s_{t+1})$ is TD Target and $\delta_t = R_t + \gamma * V(s_{t+1}) - V(s_t)$ is TD Error.

When performing an action in the state St to get the immediate reward Rt and transform to the next state s_{t+1}, the existing algorithm just uses the information of s_{t+1} in the calculation, and does not make full use of the immediate reward Rt, resulting in possible bias with the actual value $\mathrm{V}(s_{t+1})$ larger. Therefore, this paper proposes an improved TD algorithm, which uses the immediate return Rt in state St to improve the value of $\mathrm{V}(s_{t+1})$, thereby optimizing the model parameter update process, making the model training process more stable and more accurate. The specific calculation process is as follows:

Let r_t' be the model predicts of immediate reward from state S_t to state S_{t+1}. It can be calculated as:

$$r_t' = V(s_t) - V(s_{t+1}) \qquad (2)$$

Then use r_t' and rt to calculate the correction coefficient β_t, as shown in the following formula(assume that r_t' is not equals to 0, otherwise β_t is 1):

$$\beta_t = r_t / r_t' \qquad (3)$$

Therefor:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t + \beta_t * \gamma * V(s_{t+1}) - V(s_t)] \qquad (4)$$

As R_t is used in the improved TD algorithm, we can named the algorithm as RTD.

2.2 Improved multi-step TD algorithm (MRTD)

Since the transition between the states of each step is a random process, there may be a large error when just using one-step immediate reward. In order to make the correction value more robust, multi-step immediate reward needs to be used to correct the , as shown in figure 1.

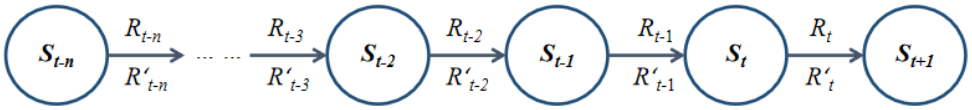


Figure 1: Actual reward and predicted reward at different steps

By definition, we can get:

$$\begin{aligned}
V(s_t) \leftarrow V(s_t) + \alpha[R_t &+ \beta_t * \gamma * V(s_{t+1}) \\
&- V(s_t)] \\
= V(s_t) + \alpha[R_t &+ \beta_t * \gamma * (R_{t+1} \\
&+ \gamma * V(s_{t+2})) - V(s_t)] \\
= V(s_t) + \alpha[R_t &+ \beta_t * \gamma * R_{t+1} \\
&+ \beta_t * \gamma^2 * V(s_{t+2})) \\
&- V(s_t)] \\
= V(s_t) + \alpha[R_t &+ \beta_t * \gamma * R_{t+1} \\
&+ \beta_t * \gamma^2 * (R_{t+2} + \gamma \\
&* V(s_{t+3})) - V(s_t)] \\
= V(s_t) + \alpha[R_t &+ \beta_t * \gamma * R_{t+1} \\
&+ \beta_t * \gamma^2 * R_{t+2} + \beta_t \\
&* \gamma^3 * V(s_{t+3}) - V(s_t)] \\
= V(s_t) + \alpha[R_t &+ \beta_t \\
&* \sum_{i=1}^{2} \gamma^i * R_{t+i} + \beta_t \\
&* \gamma^{2+1} * V(s_{t+2+1}) \\
&- V(s_t)]
\end{aligned} \tag{5}$$

Similarly, it can be inferred that:

$$\begin{aligned}
V(s_t) \leftarrow V(s_t) + \alpha * [R_t &+ \beta_t \\
&* \sum_{i=1}^{m} \gamma^i * R_{t+i} + \beta_t * \gamma^{m+1} \\
&* V(s_{t+m+1}) - V(s_t)]
\end{aligned} \tag{6}$$

Let $t \leftarrow t$ -n, according to the above formula:

$$V(s_{t-n}) \leftarrow V(s_{t-n}) + \alpha[R_{t-n} \tag{7}$$

$$+\beta_{t-n}\sum_{i=1}^{m}\gamma^{i}*R_{t-n+i}$$
$$+\beta_{t-n}*\gamma^{m+1}$$
$$*V(s_{t-n+m+1})$$
$$-V(s_{t-n})]$$

When $m = n$, we can get:

$$\begin{aligned}V(s_{t-n}) \leftarrow V(s_{t-n}) &+\alpha[R_{t-n}+\beta_{t-n}\\ &*\sum_{i=1}^{n}\gamma^{i}*R_{t-n+i}+\beta_{t-n}*\gamma^{n+1}\\ &*V(s_{t+1})-V(s_{t-n})]\\ =V(s_{t-n})+\alpha[&R_{t-n}+\\ &\textstyle\sum_{i=1}^{n}\gamma^{i}*R_{t-n+i}+(\beta_{t-n}-\\ &1)*\textstyle\sum_{i=1}^{n}\gamma^{i}*R_{t-n+i}+\\ &\beta_{t-n}*\gamma^{n+1}*V(s_{t+1})-\\ &V(s_{t-n})]\end{aligned} \tag{8}$$

In addition, before the improvement, we can get:

$$\begin{aligned}V(s_{t-n}) \leftarrow V(s_{t-n})&+\alpha[R_{t-n}+\\ \textstyle\sum_{i=1}^{n}\gamma^{i}*&R_{t-n+i}+\gamma^{n+1}*\\ V(s_{t+1})-&V(s_{t-n})]\end{aligned} \tag{9}$$

Comparing formula (8) and (9), we can see that if we use the correction coefficient from *t-n* to correct the TD error, we also need to correct the cumulative real-time discounted reward from *t-n* to *t*.

$$\begin{aligned}\gamma^{n+1}*V(s_{t+1}) \leftarrow (\beta_{t-n}-1)&*\\ \textstyle\sum_{i=1}^{n}\gamma^{i}*&R_{t-n+i}+\\ \beta_{t-n}*\gamma^{n+1}*&V(s_{t+1})\end{aligned} \tag{10}$$

The value on the left of "←" is the corrected value, and the value on the right side is the state function value predicted by the model at the state .

When n is large or tends to infinity, the second term of the plus sign on the right side of the above formula is close to (or tends to) 0. Then, the corrected result is mainly related to the actual immediate reward of the previous steps and the correction coefficient of each step. Then, the TD target is:

$$
\begin{aligned}
TD\ target\ &= R_{t-n} + \sum_{i=1}^{n} \gamma^i * R_{t-n+i} \\
&\quad + (\beta_{t-n} - 1) \\
&\quad * \sum_{i=1}^{n} \gamma^i * R_{t-n+i} \\
&\quad + \beta_{t-n} * \gamma^{n+1} \\
&\quad * V(s_{t+1}) \\
&\approx R_{t-n} + \sum_{i=1}^{n} \gamma^i * R_{t-n+i} \\
&\quad + (\beta_{t-n} - 1) \\
&\quad * \sum_{i=1}^{n} \gamma^i * R_{t-n+i} \\
&= R_{t-n} + \beta_{t-n} * \sum_{i=1}^{n} \gamma^i * R_{t-n+i}
\end{aligned}
\tag{11}
$$

It can be seen that the correction of the state value function of n steps in the past history is the correction coefficient of the corresponding time multiplied by the expectation of the return of the next state.

From state *t-n* to state *t*, the correction values of in the right side of formula (1) are respectively:

$$
\begin{aligned}
V(s_{t+1}) &\leftarrow \beta_t * \gamma * V(s_{t+1})/\gamma \\
&= \beta_t * V(s_{t+1})
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
V(s_{t+1}) &\leftarrow [(\beta_{t-1} - 1) * \sum_{i=1}^{1} \gamma^i * R_{t-1+i} \\
&\quad + \beta_{t-1} * \gamma^2 * V(s_{t+1})]/\gamma^2 \\
&= (\beta_{t-1} - 1) * R_{t-1}/\gamma + \beta_{t-1} \\
&\quad * V(s_{t+1})
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
V(s_{t+1}) &\leftarrow [(\beta_{t-2} - 1) * \sum_{i=1}^{2} \gamma^i * R_{t-1+i} \\
&\quad + \beta_{t-2} * \gamma^3 * V(s_{t+1})]/\gamma^3 \\
&= (\beta_{t-2} - 1) * \sum_{i=1}^{2} \gamma^i * R_{t-1+i}/V(s_{t+1}) \\
&\quad + \beta_{t-2} * \gamma^3 *
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
V(s_{t+1}) \leftarrow [(\beta_{t-n} - 1) * \sum_{i=1}^{n} \gamma^i * R_{t-n+i} \\
+\beta_{t-n} * \gamma^{n+1} * V(s_{t+1})]/\gamma^{n+1} \\
= (\beta_{t-n} - 1) \\
* \sum_{i=1}^{n} \gamma^i * R_{t-n+i}/\gamma^{n+1} \\
+ \beta_{t-n} * V(s_{t+1})
\end{aligned}
\tag{15}
$$

Let g_{t-n} as:

$$
\begin{aligned}
g_{t-n} = (\beta_{t-n} - 1) \\
* \sum_{i=1}^{n} \gamma^i * R_{t-n+i}/\gamma^{n+1} \\
+\beta_{t-n} * V(s_{t+1})
\end{aligned}
\tag{16}
$$

We can find that if β_{t-n}, β_{t-n+1}, …, β_t are used to update $V(s_{t+1})$, R_{t-n}, R_{t-n+1}, …, R_t need to be used at the same time, then:

$$
V'(s_{t+1}) = \sum_{i=t-n}^{t} g_i/(n+1) \tag{17}
$$

Considering that there are different weights of the correction value of continuous multiple steps, the attenuation factor is added here (the value range is (0,1]), indicating that the farther away from the current state, the smaller the weight of the correction factor. As shown in the following.

$$
V'(s_{t+1}) = (\sum_{i=t-n}^{t} \varphi^i * g_i)/(\sum_{i=t-n}^{t} \varphi^i) \tag{18}
$$

On this basis, the TD algorithm is improved as follows.

$$
\begin{aligned}
V(s_t) \leftarrow V(s_t) + \alpha[R_t + \gamma * V'(s_{t+1}) \\
-V(s_t)]
\end{aligned}
\tag{19}
$$

Where *n* is the step size, and when *n* = 1, it is the improved single-step TD algorithm(shorted for SRTD).When *n* > 1, it is an improved multi-step TD algorithm(shorted for MRTD).

In addition, the TD algorithm is the abbreviation of TD(λ) algorithm. When λ=0, the TD algorithm is called TD(0) algorithm. Generally, unless otherwise specified, the TD algorithm refers specifically to TD(0) algorithm. The RTDs in this paper is mainly for TD(0) algorithm, but the idea is also applicable to scenarios where λ is not 0.

Further more, the RTDs in this paper can not only be used to improve the state value function $V(s_{t+1})$,it can also be used to improve the state-action value function $Q(s_{t+1}, a_{t+1})$ in the same way.

2.3 Improved DQN algorithm

In reinforcement learning, two typical implementations of TD algorithm are SARSA algorithm and Q-Learning algorithm. SARSA algorithm is an On-Policy TD algorithm, while the Q-Learning algorithm is an Off-Policy algorithm. These two algorithms are mainly used in scenarios where the state space is discrete. When the state space is very large or tends to be infinite, the problem of dimensional disaster will occur. In response to the problem, scholars have proposed a method of value function approximation, that is, using a function to approximate the values in formula (1). Among them, the more classic one is the DQN algorithm proposed by Deepmind's scientists in 2015[33-34].

The DQN algorithm is an improvement to the Q-learning algorithm, so the RTDs proposed in this paper can also be used to improve the DQN algorithm. The specific process is shown in tabel 1.

Table 1: Algorithm 1: Improved DQN algorithm

1. First, for all states $s \in S$ and all actions $a \in A(s)$, initialize the parameters α and γ and the maximum episode N, construct neural network $Q(s,a;\boldsymbol{w})$ and initialize its parameters $\boldsymbol{w}$.Initialize φ and steps n in formula (18). Initialize lists buffer list Bs,which is used to save multi-step correct values β.

2. For each episode from 1 to N, do:

Initialize the starting state s_0 according to the existing strategy, and select action a in state s_0 according to the ε-greedy strategy;

For each step t in the same episode,do:

①According to the ε-greedy strategy, choose the action in state s_t, get the immediate reward r_t and the next state s_{t+1};

②Calculate the estimated value of the action value function in state s_t according to the network parameters:

$$q_t = Q(s_t, a_t; \boldsymbol{w}_t) \tag{20}$$

③Use the current network parameters to calculate the estimated value of the action value function in state s_{t+1}:

$$q_{t+1} = \max_a Q(s_{t+1}, a_{t+1}; \boldsymbol{w}_t) \tag{21}$$

④Use formula (2) to calculate the predicted value of the immediate reward from state s_t to state s_{t+1} :

$$r_t' = q_t - q_{t+1} \tag{22}$$

⑤Calculate β_t, and append a transition data (s_t, a_t, r_t, s_{t+1}, β_t) to the end of Bs. If the length of Bs is greater than n, pop the first value of Bs.

⑥Use formula (16) 、(17)、(18) to calculate the predicted value $Q^*(s_{t+1})$:

$$Q^*(s_{t+1}, a_{t+1}; \boldsymbol{w}_t) = \beta_t * q_{t+1} \tag{23}$$

⑦Calculate the value of TD Target:

$$y_t = r_t + \gamma \cdot Q^*(s_{t+1}, a_{t+1}; \boldsymbol{w}_t) \quad (24)$$

⑧Calculate the gradient of the action-value network:

$$\boldsymbol{d}_t = \frac{\partial Q(s_t, a_t; \boldsymbol{w}_t)}{\partial \boldsymbol{w}} \quad (25)$$

⑨Update the parameters of the network according to the gradient descent method:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha * (q_t - y_t) * d_t \quad (26)$$

End loop;

End loop.

2.4 Improved DDPG algorithm

The above improvements to the TD algorithm are implemented within the framework of the DQN algorithm. Since the DQN algorithm is a Value-Based algorithm, it is generally suitable for scenarios with limited action space (discrete space). When the action space is very large or continuous, the method of value function approximation cannot obtain an accurate action policy. Therefore, scholars propose to directly parameterize the search strategy and express the strategy as a function of the state, and then the method based on the Policy-Based appears[35-36]. On the basis of Value-Based and Policy-Based methods, in order to make full use of the advantages of both, the AC (Actor-Critic) algorithm integrating the two and a series of related improved algorithms (such as A2C algorithm[37], A3C algorithm[38], etc.) appeared.Under the framework of the AC algorithm, the policy gradient update formula can be written as:

$$g = E[\int H_t \nabla_\theta log\pi_\theta(a_t|s_t) d_t] \quad (27)$$

In the formula, H_t is Value-Based state value function, which is used to evaluate the pros and cons of the current action and plays the role of Critic. $\nabla_\theta log\pi_\theta(a_t|s_t)$ is Policy-Based policy function, which is used to guide the choice of strategy and plays the role of Actor. Among them, there are many different calculation methods of H_t. According to the calculation method, the AC algorithm can be divided into the Monte Carlo-based AC algorithm, the advantage function-based AC algorithm, and the TD-based AC algorithm, and so on. When calculated with TD error, H_t can be written as:

$$H_t = r_t + \gamma * V^\pi(s_{t+1}) - V^\pi(s_t) \quad (28)$$

Here, the AC algorithm is called TD-AC algorithm. For example, drive from Guangzhou to Wuhan, passing through Changsha City. As shown below.

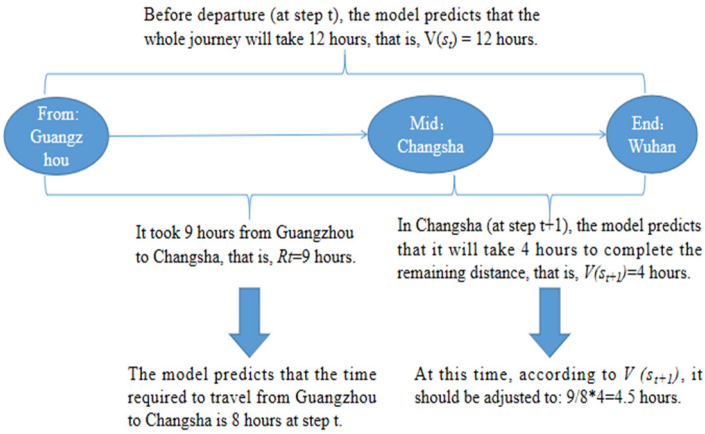


Figure 2: Example of TD error

Before departure, the model predicts that the entire journey will take 12 hours, but when arriving in Changsha, it actually took 9 hours. At this point, the model predicts that it will take another 4 hours from Changsha to Wuhan. In this example, both the 12 hours from Guangzhou to Wuhan and the 4 hours from Changsha to Wuhan are estimated by the same model. It can be inferred that using the same model to estimate the time from Guangzhou to Changsha would take 12-4=8 hours. Therefore, based on this model, it was estimated that it would take a total of 13 hours from Guangzhou to Wuhan (9+4). Therefor, $V(s_t)$ is 12 hours, R_t is 9 hours, and $V(s_{t+1})$ is 4 hours. However, in reality, the estimate for $V(s_{t+1})$ should have been adjusted based on the available information. For example, the time needed from Changsha to Wuhan should be adjusted proportionally, which would be 9/8*4=4.5 hours. The TD algorithm update model parameters based on 4 hours, but in reality, updating the model parameters based on 4.5 hours is more reasonable.

It can also be improved with the RTDs proposed in this paper. However, the outputs of the AC series algorithms are the probability density functions of the action, not the deterministic action. They are stochastic policy gradient algorithms. In addition to the AC algorithm and its variants mentioned, popular stochastic policy gradient algorithms also include TRPO[39], PPO[40], and so on. The convergence performance of this type of algorithm is relatively good, but the main problems are, on the one hand, that it needs to sample the action according to the probability density function of the action, which has a large uncertainty. On the other hand, it cannot be directly applied to the scene of continuous action space. In response to these problems, scholars have proposed the DDPG (Deep Deterministic Policy Gradient) algorithm based on the AC algorithm[41-42]. The output of the DDPG algorithm is a deterministic action that reduces randomness and errors. The DDPG algorithm has been widely concerned and applied due to its excellent performance in computational efficiency and accuracy. Since the DDPG algorithm is an improvement of the Actor-Critic algorithm, it is also composed of an Actor network and a Critic network. The Actor network output is a deterministic action, and the Critic network is used to evaluate the Actor's output action.Similarly, the DDPG algorithm can be improved with the RTDs proposed in this paper, as shown in table 2.

Table 2: Algorithm 2. Improved DDPG algorithm

1. Initialize the Critic network $Q(s,a|\boldsymbol{\theta}^{Q})$ parameters and Actor network $\mu(s|\boldsymbol{\theta}^{\mu})$ parameters. Initialize the target network parameters as $\boldsymbol{\theta}^{Q'}=\boldsymbol{\theta}^{Q}$ and $\boldsymbol{\theta}^{\mu'}=\boldsymbol{\theta}^{\mu}$. Initialize γ and the maximum training episode number N.Initialize φ and steps n in formula (18). Initialize lists buffer list Bs,which is used to save the multi-step correct values β.

2. For each episode from 1 to N, do:

Initialize the starting state s_0 according to the existing strategy;

For each step t=1..T, do:

①Sampling an action a_t according to the Actor network:

$$a_t=\mu(s_t|\boldsymbol{\theta}^{\mu})+\Omega_t \tag{29}$$

②Perform action a_t in state s_t, get a reward r_t, and the next state s_{t+1};

③ Using the Critic network $Q(s,a|\boldsymbol{\theta}^{Q})$ to estimated value of state s_t $Q'(s_t,\mu'(s_t|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'})$, the estimated value of the next state $Q'(s_{t+1},\mu'(s_{t+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'})$;

④Calculate β_t, append the transition data $(s_t,a,r_t,s_{t+1},\beta_t)$to the end of Bs. If the length of Bs is greater than n, pop the first value of Bs.

⑤Sampling a mini batch from Bs, use formula (16) 、(17)、(18) to update the predicted value $Q'(s_{t+1},\mu'(s_{t+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'})$;.Note that the data in a mini batch can be Calculated concurrently;

⑥Update the estimated value of the next state-action function and calculate the value of the Critic target network y_i:

$$y_i=r_i+\beta_t*\gamma*Q'(s_{t+1},\mu'(s_{t+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'}) \tag{30}$$

⑦Update Critic network parameters:

$$\delta_t=\frac{1}{N}\sum_i^N(y_i-Q(s_i,a_i|\boldsymbol{\theta}^{Q})) \tag{31}$$

$$\boldsymbol{\theta}^{Q}{}_{t+1}=\boldsymbol{\theta}^{Q}{}_{t+1}+\delta_t*\alpha^{Q}*\nabla_{\boldsymbol{\theta}^{Q}}Q(s_t,a_t|\boldsymbol{\theta}^{Q}) \tag{32}$$

⑧Update policy network parameters:

$$\nabla_{\boldsymbol{\theta}^{\mu}}|_{s_t}\approx\frac{1}{N}\sum_{i=1}^{N}\nabla_a Q(s,a|\boldsymbol{\theta}^{Q})|_{s=s_t,a=\mu(s_t)}\nabla_{\boldsymbol{\theta}^{\mu}}\mu(s|\boldsymbol{\theta}^{\mu})|_{s_t} \tag{33}$$

$$\boldsymbol{\theta}^{\mu}{}_{t+1}=\boldsymbol{\theta}^{\mu}{}_{t+1}+\beta_t*\delta_t*\alpha^{\mu}*\nabla_{\boldsymbol{\theta}^{\mu}}|_{s_t} \tag{34}$$

⑨Update the target network parameters:

$$\boldsymbol{\theta}^{Q'}=\tau\boldsymbol{\theta}^{Q}+(1-\tau)\,\boldsymbol{\theta}^{Q'} \tag{35}$$

$$\boldsymbol{\theta}^{\mu'}=\tau\boldsymbol{\theta}^{\mu}+(1-\tau)\,\boldsymbol{\theta}^{\mu'} \tag{36}$$

End loop;

End loop.

3. EXPERIMENTS AND ANALYSIS

In this paper, the improved DQN algorithm and the improved DDPG algorithm are used in stock quantitative trading process, and compared with the algorithms before the improvement. The experiment data originate from www.baostock.com, a free and open source securities data platform.

3.1 Experimental data

The data used in this paper is the daily exchange data of Chinese stock market since January 1, 2000, of which 22 years of data from January 4, 2000 to December 31, 2021 are used as the training set (5332 trading days in total), including the daily trading information of all stocks and indexes of Chinese stock market during this period. The data for a total of 3 months January 4, 2022, to March 31, 2022,is used as the test set (a total of 58 trading days) . The dataset includes 13 features，such as exchange date, stock code, opening price today, highest price, lowest price, closing price today, closing price yesterday, transaction quantity, transaction amount, turnover rate, resumption status, transaction status, change percentage, etc. Among them with 9 continuous features and 2 discrete features (re-weight status, transaction status), except exchange date and stock code.We first use all the stock data to train the model, and for convenience of description, we take China Merchants Bank(sh.600036) and Lepu Medical(sz.300003) as examples to illustrate the effects of the models in this paper.The reason why we choose these two stocks is that one is from the A-share main board and the other is from the GEM.The volatility of the main board stocks is small, while the volatility of the GEM stocks is large, representing different risk levels and income levels.

Since the dimensions of the variables in the original data are not uniform, it is first necessary to convert the continuous variable data into normalized data with a mean of 0 and a standard deviation of 1. At the same time, discrete features such as month, week, and day (that is, the number of the month) , which extracted from the exchange date, will be converted to One-hot type (52 features are obtained), together with the features such as weight restoration status and transaction status. The above 9+52 feature variables are combined to 61 vectors as environmental features.

3.2 Markov Decision Process Model

This paper takes stock quantitative trading as an example to build a deep reinforcement learning model, takes stock investors as the agent, and takes the stock market as the system. The Markov decision process model (S, A, P, R, γ) is shown in the following.

(1) **State space S**: The 61 environmental features obtained above, combining the agent's self information (including total market value, total assets, available amount, number of shares, current profit and loss, total profit and loss, total profit and loss ratio, available Amount, position ratio and other 9 features), a total of 70 features are used as the environmental state variables of the model.

(2) **State transition matrix P**: Since the stock market is a large and complex nonlinear system, its state transition matrix P cannot be obtained. When the agent takes an action, the environment moves to the next state (including the state of the environment and the state of the agent). In this experiment, the next state can be obtained through simulated sampling and

calculation, in which the stock price can be uniformly sampled between the highest price and the lowest price on the second day, and then other environmental variables are calculated in equal proportions.Agent's own state information can be obtained by corresponding calculation after taking action.

(3) **Action space** $\boldsymbol{A}$: including the three actions of buying, selling and holding. The dimension of action space is 1-dimensional space. When the value of action a is less than 0, it is a selling operation, and when it is greater than 0, it is a buying operation, and when it is equal to 0, it is holding operation.

(4) **Reward function** $\boldsymbol{R}$: the reward function of the model is designed as the proportion of the income brought by the operation (total assets after investment/total assets before investment -1), and in order to ensure the safety of funds, when losses occur, give a large penalty, amplifies the loss value by a factor of 10. which is:

$$r_t = \begin{cases} r_t, & \text{if } r_t >= 0 \\ 10 * r_t, & \text{if } r_t < 0 \end{cases} \tag{37}$$

(5) **Discounted rate** γ: $\gamma \in [0, 1]$.

3.3 Neural networks

In this paper, two groups of improved models are designed to compare the effectiveness, as follows.

3.3.1 DQN and improved DQN networks

The improvement of the algorithm in this paper only involves the update process of specific parameters, not the network structure. In order to compare the effectiveness of algorithms before and after the improvement, the DQN algorithm and the improved DQN algorithm use the same network structure. Since the stock data is time series data, its network model mainly adopts the LSTM layers + MLP layers, as shown in figure 3.

The network includes three LSTM layers, one BatchNormalization(BN) layer, and two fully connected layers. The input dimension of the network is [30, 71], where 30 indicates the past 30 trading days, 70 is the number of features, and 1 is the action dimension. In order to unify the network input, the action shape is expanded into a [30,1] tensor, all 1 when the action is buying, all -1 when the action is selling, and all 0 when the corresponding action is holding. Each LSTM layer adopts the RELU activation function, and its output is processed by the BN layer. The normalized data is connect to the fully connected layer. The first fully connected layer also uses the RELU activation function, and the second fully connected layer is the output layer and does not require an activation function. In terms of main parameter settings, the learning rate of the Actor network is 0.0001, the learning rate of the Critic network is 0.0002, and the value of τ is 0.01.

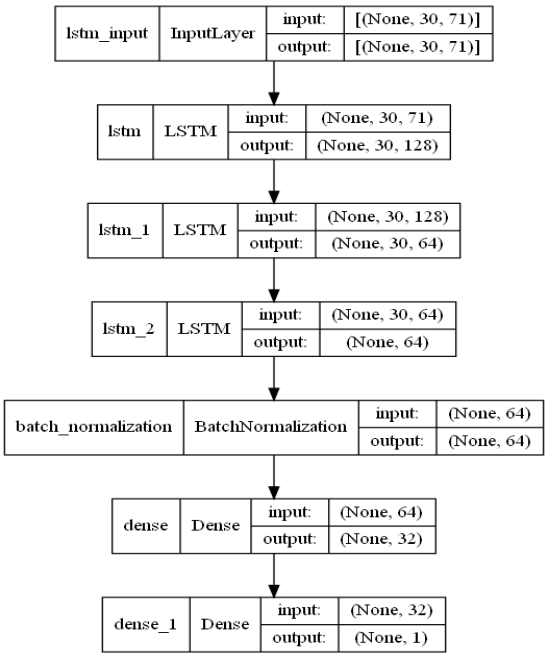


Figure 3: Network of DQN and improved DQN

3.3.2 DDPG and improved DDPG networks

Since the DDPG model includes two networks, Actor and Critic, in order to make multiple algorithms comparable, both Actor and Critic networks here adopt a structure similar to the above

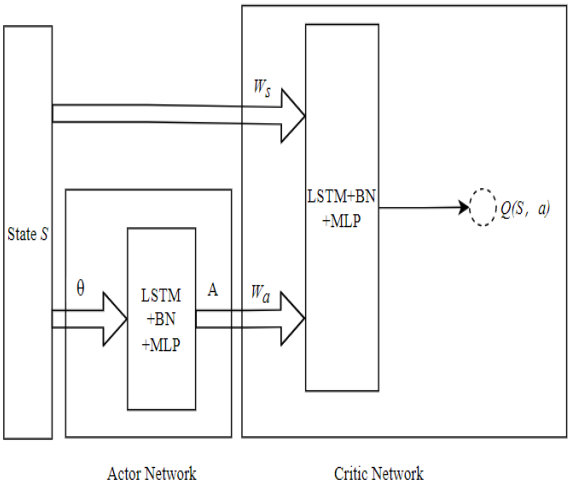
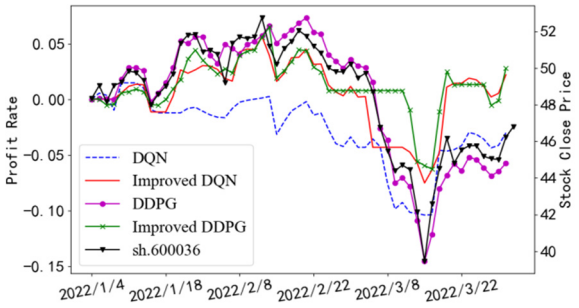


Figure 4: DDPG structure and improved DDPG structure

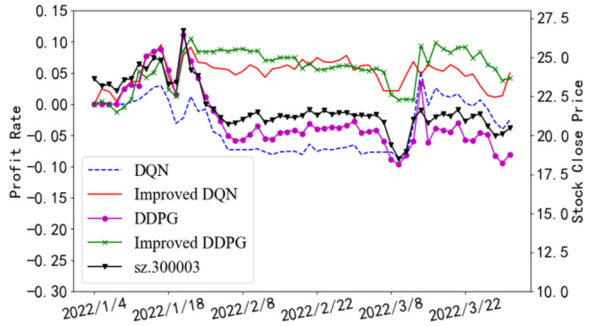
DQN model. The differences are:①Actor network input is 30*70 tensor (without action input);②Actor network output is a continuous value between -1 and 1, which is expanded to 30*1. After the tensor is combined with the state tensor, a 30*71 tensor is input into the Critic network.The structure is shown in figure 4.

3.4 Experimental results and discusses

Since reinforcement learning aims to maximize returns rather than predict trends, and therefore performance cannot be evaluated using traditional metrics such as RMSE, MAE, and R-square. Instead, this study evaluates model performance based on final returns. We use data from China Merchants Bank (sh.600036) and Lepu Medical (sz.300003) as examples, with dates ranging from January 4, 2022 to March 30, 2022, and calculate the cumulative return for each day. The results are presented in Figure 5.



(a) sh.600036



(b) sz.300003

Figure 5: Comparison of cumulative returns of different models on each trading day

For sh.600036, Fig.4 shows that the DQN model failed to implement an effective investment strategy to obtain due investment returns in the early stage of the stock rising cycle, and failed to take effective actions to avoid risks in the later stage of stock price decline. The improved DQN model can better follow the rising cycle of stocks, significantly improve investment returns, and can take effective action strategies in the falling cycle of stocks to avoid more

losses, indicating that the improved DQN model is added to the current operation. The correction of immediate return can effectively improve the return on investment. The DDPG model can fit the trend of stocks very well, and actively invest in the rising cycle of stocks to get higher returns, but still take active action strategies in the downward cycle of stocks, and there is also a large loss. The improved DDPG model can get higher returns in the stock up cycle, and take the correct action strategy (stop trading) for a longer period of time in the stock down cycle, avoid investment losses, and thus get higher returns.And we can get similar conclusion on stock sz.300003.

Further more, the investment return indicators under each model are counted, as shown in table 3.

For Sh.600036, through historical data query, it can be seen that the stock price was 48.71 RMB before the opening on 2021-1-4, and the closing price on 2022-3-30 was 46.23, a decrease of 5.09%, and the average daily increase and decrease of 0.06% . It can be seen from table 3 that the Average daily Cumulative Return of the DQN model is the smallest, the loss ratio reaches 2.75%, and the return at the end of the test period (2022-3-30) is -3.02%, the standard deviation of return is 0.0317%. It can be seen that the investment strategy based on the DQN model can effectively reduce investment losses. The average daily cumulative rate of return of the improved DQN model is 0.53%, and the final rate of return is 2.19%, which can obtain a positive rate of return when the stock falls as a whole, which shows the effectiveness of the improved DNQ model proposed in this paper. The average daily cumulative rate of return of the DDPG model is 0.24%, and the final rate of return is -5.75%, which shows that the overall rate of return of the DDPG model is relatively poor. The standard deviation of the daily cumulative rate of return is 0.0560%, and the maximum drawdown reaches 21.87%. It can be seen that the model's return is very volatile. The average daily cumulative return of the improved DDPG model is 1.25%, the final return is 5.09%, the standard deviation of the daily return is only 0.0242%, and the maximum drawdown ratio is only 12.75%. Both performance and benefits of improved DDPG model are better than the DDPG model.For sz.300003, we can find that,the improved DQN model the improved DDPG model are also better than the model before improvement on each performance index, which also shows the effectiveness of the improved algorithm.

Further more, to analyze the reasons for the differences in the effectiveness of different models and the trading characteristics of each model (such as the number of trading days, the average daily position ratio and its standard deviation, the maximum buying ratio, the maximum selling ratio, etc.), as shown in Table 4.

Table 3: Statistical comparison of cumulative investment returns

Stock	Model	Average of Daily Cumulative Return	Standard Deviation of Daily Cumulative Return	Cumulative Return in the End	Maximum Drawdown	Sharpe Ratio
	DQN	-2.75%	0.0317%	-3.02%	11.88%	-1.18
	Improved DQN	0.53%	0.0307%	2.19%	13.18%	-0.16

Sh.600036	DDPG	0.24%	0.0560%	-5.75%	21.87%	-0.14
	Improved DDPG	1.25%	0.0242%	2.78%	12.75%	0.09
	Sh.600036	-0.06%	0.0238%	-5.09%	25.54%	-0.45
Sz.300003	DQN	-3.43%	0.0403%	-2.53%	11.59%	-1.09
	Improved DQN	5.01%	0.0220%	3.44%	8.42%	1.52
	DDPG	-2.38%	0.0484%	-8.04%	20.77%	-0.69
	Improved DDPG	5.74%	0.0310%	4.19%	9.81%	1.82
	Sz.300003	-0.13%	0.0369%	-13.30%	27.34%	-1.32

Table 4: Comparison of transaction characteristics of different models

Stock	Model	Exchange Days	Average Daily Position Ratio	Standard Deviation of Daily Position Ratio	Maximum Buying Ratio	Maximum Selling Ratio	Days of More than 95% position Ratio
Sh.600036	DQN	57	96.52%	4.38%	100.00%	100.00%	52
	Improved DQN	47	75.15%	18.39%	100.00%	100.00%	24
	DDPG	55	94.52%	6.44%	100.00%	100.00%	52
	Improved DDPG	42	63.55%	33.46%	100.00%	100.00%	20
Sz.300003	DQN	57	96.44%	4.41%	100.00%	100.00%	51
	Improved DQN	50	72.23%	19.22%	100.00%	100.00%	18
	DDPG	55	95.02%	6.40%	100.00%	100.00%	52
	Improved DDPG	45	72.61%	21.45%	100.00%	100.00%	19

In table 4, the DQN model and the DDPG model have traded on 57 and 55 trading days respectively in the 57 trading days for sh.600036, 51 and 52 for sz.300003.However, for sh.600036, the number of trading days of the improved DQN model and the improved DDPG model are significantly smaller than those unimproved models, and the number of days of more than 95% of positions is also less, the same to sz.300003. Combined with the above figure 5, it can be seen that although the number of transactions in the improved DDPG model has decreased, the trading accuracy has been greatly improved. And finally, in a falling stock market, the model can still get better returns.

4. CONCLUSIONS

This paper proposes an improved TD algorithm (RTD) and extends it to multi-step conditions (MRDT). Based on this, we propose improved versions of the DQN and DDPG algorithms. We then demonstrate the effectiveness of these algorithms by applying them to the Chinese stock market under extreme conditions. Experimental results show that our improved algorithms can significantly improve returns and reduce trading risk in quantitative stock trading scenarios. In the future, we plan to further extend these algorithms to multi-agent scenarios and explore their applications in more complex scenarios, such as portfolio investment strategies.

REFERENCES

[1] Tang L, Wang H, Li L, et al. Quantitative models in emission trading system research: A literature review[J]. Renewable and Sustainable Energy Reviews, 2020, 132:110052.

[2] Farmer J D, Foley D. The economy needs agent-based modelling[J]. Nature, 2009, 460.

[3] Rosenberger J. Computer Scientists Beat U.S. Stock Market[J]. Communications of the ACM, 2010, 53(8):P.20.

[4] Yanshuang Li, Xintian Zhuang, Jian Wang, Weiping Zhang Community structure and systemic risk analysis of China's stock market under extreme market [J] Journal of Northeast University (NATURAL SCIENCE EDITION), 2020,41 (10): 1500-1508.

[5] Gan-Hua Wu, et al. Statistics of extreme events in Chinese stock markets[J]. Chinese Physics B, 2014.

[6] Jca B, Mg B, Hz A. Coherence, extreme risk spillovers, and dynamic linkages between oil and China's commodity futures markets[J]. Energy, 2021.

[7] T Lyócsa, N Todorova, Vrost T. Predicting risk in energy markets: Low-frequency data still matter[J]. Applied Energy, 2021.

[8] Cagliero L,Garza P,Attanasio G, et al. Training ensembles of faceted classification models for quantitative stock trading[J].Computing, 2020(4):1-13.

[9] Staff S. How computers helped stampede the stock market: Many analysts blame automated program trading for the worldwide plunge — But the machines only did what they were told[J]. IEEE Spectrum, 2013, 24(12):30-33.

[10] Mu G H , Zhou W X , Chen W , et al. Order flow dynamics around extreme price changes on an emerging stock market[J]. New Journal of Physics, 2010.

[11] Fengzhao Yang, Yangyong Zhu. An Efficient Method for Similarity Search on Quantitative Transaction Data[J]. Computer Research and Development, 2004(02):361-368.

[12] Xiaoxu Zhang, Zhentao Gao, Lei Wu, Xin Li, Mingjing Lu. Stock price prediction based on hybrid quantum classical neural network model[J] Journal of University of Electronic Science and technology, 2022,51 (01): 16-23.

[13] Volodymyr M , Koray K , David S , et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529-33.

[14] Luo B , Wu H N , Huang T , et al. Reinforcement learning solution for HJB equation arising in constrained optimal control problem[J]. Neural Networks, 2015, 71:150-158.

[15] Ying H, Zheng Z, Yu F R, et al. Deep Reinforcement Learning-based Optimization for Cache-enabled Opportunistic Interference Alignment Wireless Networks[J]. IEEE Transactions on Vehicular

Technology, 2017, 66(11):10433-10445.
[16] Zhu B, Bedeer E, Nguyen H H, et al. UAV Trajectory Planning in Wireless Sensor Networks for Energy Consumption Minimization by Deep Reinforcement Learning[J]. IEEE Transactions on Vehicular Technology, 2021.
[17] Jung S, Yun W J, Shin M, et al. Orchestrated Scheduling and Multi-Agent Deep Reinforcement Learning for Cloud-Assisted Multi-UAV Charging Systems[J]. IEEE Transactions on Vehicular Technology, 2021, PP(99):1-1.
[18] Arulkumaran K, Deisenroth M P, Brundage M , et al. A Brief Survey of Deep Reinforcement Learning[J]. IEEE Signal Processing Magazine, 2017, 34(6).
[19] Shah D ,Campbell W , Zulkernine F H . A Comparative Study of LSTM and DNN for Stock Market Forecasting[C]// IEEE International Conference on Big Data. IEEE, 2018.
[20] Neeharika C H , Riyazuddin Y M , Vijayalakshmi D . Performance of Stock Price Prediction using Deep Learning by using AI and Machine Learning Technique[J]. High Technology Letters, 2021.
[21] Moody J, Wu L, Liao Y, et al. Performance functions and reinforcement learning for trading systems and portfolios[J]. Journal of Forecasting, 1998, 17(5-6):441-470.
[22] Lee J W. Stock price prediction using reinforcement learning[C]// IEEE International Symposium on Industrial Electronics. IEEE, 2001:690-695.
[23] Huang G, Zhou X, Song Q. Deep reinforcement learning for portfolio management based on the empirical study of chinese stock market. 2021.
[24] Li H, Dagli C H, D Enke. Dynamic Programming and Reinforcement Learning (ADPRL 2007) Short-term Stock Market Timing Prediction under Reinforcement Learning Schemes. 2013.
[25] Pendharkar P C, Cusatis P. Trading financial indices with reinforcement learning agents[J]. Expert Systems with Applications, 2018, 103:1-13.
[26] NN Y Vo, He X , Liu S , et al. Deep learning for decision making and the optimization of socially responsible investments and portfolio[J]. Decision Support Systems, 2019, 124:113097.
[27] Tianxin Liang, Xiaoping Yang, Liang Wang, Zhenyuan Han.Research and development of financial transaction system based on Reinforcement Learning[J] Journal of software, 2019,30 (03): 845-864 DOI:10.13328/j.cnki. jos. 005689.
[28] Yuefeng Cen, Chenguang Zhang , Gang Cen, Cheng Zhao. Stock price prediction method based on near end reinforcement learning[J].Control and decision making, 2021,36 (04): 967-973 DOI:10.13195/j.kzyjc. 2019.1245.
[29] Brando I V ,Costa J , Praciano B , et al. Decision support framework for the stock market using deep reinforcement learning[C]// 2020 Workshop on Communication Networks and Power Systems (WCNPS). 2020.
[30] Huang G, Zhou X, Song Q. Deep reinforcement learning for portfolio management based on the empirical study of chinese stock market. 2021.
[31] Zhidong Liu, Zhiyuan Zhao. Clustering of intra day state characteristics and optimizing transaction execution in China's financial market[J].Systems Science and mathematics, 2021,41 (06): 1648-1668.
[32] Sutton R, Barto A. Reinforcement Learning:An Introduction[M]. MIT Press, 1998.
[33] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J]. Computer Science, 2013.
[34] Volodymyr M , Koray K , David S , et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529-33.
[35] Pablo, Escandell-Montero, Delia, et al. Online fitted policy iteration based on extreme learning

machines[J]. Knowledge-Based Systems, 2016, 100(May 15):200-211.
[36] MD Awheda, Schwartz H M. Exponential moving average based multiagent reinforcement learning algorithms[J]. Artificial Intelligence Review, 2016, 45(3).
[37] Vamvoudakis K G, Lewis F L. Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem[J]. Automatica, 2010, 46(5):878-888.
[38] Fernando C, D Banarse, Blundell C, et al. PathNet: Evolution Channels Gradient Descent in Super Neural Networks[J]. 2017.
[39] Schulman J , Levine S , Moritz P, et al. Trust Region Policy Optimization[C]// ICML. 2015.
[40] Schulman J, Wolski F , Dhariwal P, et al. Proximal Policy Optimization Algorithms[J].2017.
[41] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. Computer ence, 2015.
[42] Arulkumaran K,Deisenroth M P, Brundage M, et al. A Brief Survey of Deep Reinforcement Learning[J]. IEEE Signal Processing Magazine, 2017, 34(6).