Proceedings of the 4th International Conference on Mathematical Physics and Computational Simulation
DOI: 10.54254/2753-8818/2026.35156

Stock Prediction Based on Value Function Reinforcement Learning Algorithm

Ningyu Cai

School of Information Management and Information System, Northeastern University at Qinhuangdao, Qinhuangdao, China
202411618@stu.neuq.edu.com

Abstract. This paper applies three algorithms in deep reinforcement learning (DQN, Double DQN, Dueling Double DQN) to stock prediction and trading decisions. Using the historical trading data of Kweichow Moutai (600519.SS) from 2018 to 2023 as the experimental sample, a systematic stock trading reinforcement learning environment is constructed. Subsequently, the prediction performance and trading returns of the three algorithms are compared. Ultimately, the cumulative returns of the first two algorithms are 15.2% and 23.5% respectively, while that of Dueling Double DQN is 32%. In terms of maximum drawdown, Dueling Double DQN is only 8%, significantly lower than 18% and 12% of DQN and Double DQN respectively. Regarding the win rate, Dueling Double DQN reaches 65%, higher than 52% and 58% of the other two algorithms. In terms of average total reward, Dueling Double DQN reaches 1645.72, far exceeding 848.56 and 1196.89 of DQN and Double DQN respectively. The results of the experiment show that due to the adoption of the value-advantage separation network structure, Dueling Double DQN outperforms Double DQN and basic DQN in terms of cumulative return, maximum drawdown, and win rate. Therefore, the accuracy of stock prediction and the stability of trading decisions have been greatly improved.

Keywords: Reinforcement learning, DQN algorithm, Double DQN, Dueling Double DQN, Stock prediction

1. Introduction

As a fundamental component of the financial market, the price fluctuations of the stock market have the most direct and significant impact on investors' decisions and investment returns. Therefore, accurate stock price predictions not only assist investors in making buying and selling decisions, optimizing asset allocation to achieve excess returns [1], but also supply an essential basis for risk management and market supervision, and are of great significance for maintaining the overall stability and efficiency of the financial market [2]. Thus, accurate stock predictions are of great significance for the stability of the financial market and the profitability of investors. However, it is undeniable that stock prices are affected by various factors such as macroeconomics, industry policies, and market sentiment, presenting obvious nonlinear dynamic characteristics [3]. Therefore, traditional time series analysis (such as ARIMA, GARCH models) and traditional machine learning

DOI: 10.54254/2753-8818/2026.35156

methods based on supervised learning (such as support vector machines, random forests, etc.) cannot fully explore the inherent laws of price changes, and their prediction accuracy and decision utility are naturally limited [4]. These methods usually focus on single-step predictions of static datasets and are difficult to effectively depict and utilize the sequential dependence of market state evolution and the long-term cumulative effects of decisions.

Reinforcement Learning (RL) is a machine learning paradigm which based on trial-and-error learning and studying the optimal sequential decision-making strategy through sustaining interaction between an intelligent agent and the environment [5]. Unlike traditional supervised learning that learns "what is" from fixed labeled data, the core of reinforcement learning relies on learning "how to act" to maximize long-term cumulative rewards [6]. This characteristic makes it naturally applicable to typical sequential decision-making problems such as stock trading, which have delayed returns and high uncertainty [7]. The DQN algorithm was the first to combine deep learning with Q-learning, effectively settling the problem of state space explosion in conventional Q-learning. Also, it is an excellent attempt at high-dimensional state decision-making [6]. Subsequently, Double DQN directly and ingeniously improved the overestimation defect of Q values in DQN, and Dueling Double DQN further split the value function, separating the state value function from the action-dependent advantage function for learning, thus being able to more stably evaluate the relative value of actions in different states, and is therefore considered to be able to more effectively capture and utilize the long-term trends and relative value information of the stock market [8, 9].

This paper uses three reinforcement learning algorithms (DQN, Double DQN, Dueling Double DQN) as tools, takes real stock data as the research object, and conducts a systematic and rigorous comparative analysis of the behavior of several algorithms in stock prediction and trading decisions. Thus, naturally and reasonably, it discusses the applicable occasions and respective advantages and disadvantages of different algorithms, and accordingly proposes suitable technical methods for stock prediction.

2. Research methods

This paper uses Python as the development language and implements the programming and experiments of the algorithms based on the PyCharm development environment. The core dependent libraries include numpy, pandas, torch, matplotlib, yfinance, etc. All codes are run in PyCharm.

Through the value function of deep reinforcement learning algorithms, the elementary purpose can be explained: to adapt the state-action value function Q(s, a), that is, "the accumulative reward of performing action a in state s" The proposed method uses a neural network to fit the Q value and introduces an experience playback mechanism to eliminate data correlation, while strictly separating the target network from the evaluation network, thus effectively suppressing training oscillations. It is in the form of formula (1), that is, the Bellman optimality equation:

$$y = r + \gamma max a^{'} Qtarget\left(s^{'}, a^{'}\right) \tag{1}$$

In this formula: y: Target Q value, which is the target value of the current state-action pair (s, a), is used to update the Q value of the evaluation network.; γ: Discount factor, a constant with a value range of [0, 1], used to measure the weight of future rewards. The closer γ gets to 1, the more the agent values long-term rewards; the closer γ gets to 0, the more it values short-term rewards. r: Immediate reward, which is the instant reward value fed back by the environment after the agent

DOI: 10.54254/2753-8818/2026.35156

carrys out action a in the current state s. Qtarget: Target Q network, used to stabilize training, with parameters periodically synchronized from the evaluation network to avoid frequent fluctuations in target values. maxa′Qtarget(s′,a′): The maximum Q value of the destination network for the next state s′, where s′ is the new state reached after carrying out action a, and a′ represents all probable actions in s′. Taking the maximum Q value represents the optimal future value. Double DQN provides a very clear solution to the problem of overestimation of Q-values in DQN: it splits "selecting actions" and "evaluating values" into two networks, using the evaluation network choose the top action and the destination network to assess the value of the chosen action, in order to make the decision more reliable [10]. The specific formula is shown in (2):

$$y = r + \gamma \cdot Qtarget\left(s^{'}, argmaxa^{'}Qeval\left(s^{'}, a^{'}\right)\right) \tag{2}$$

In this formula:y, r, γ: represent the target Q value, immediate reward, and discount factor respectively; Qeval: the evaluation Q network, which updates parameters in real time and is used to choose the top action; s', a': represent the next state and all possible actions in the next state respectively; argmaxa'Qeval(s', a'): use the evaluation network to choose the top action a* under s', rather than directly taking the maximum Q value, to settle the overestimation problem of Q values in DQN; Qtarget(s', a*): use the destination network to calculate the Q value of the top action a*, as an estimate of future value, achieving the separation of "action selection and value estimation". Dueling Double DQN goes further on this basis, decomposing the Q value into state value V(s) and advantage value A(s, a) for separate learning, thereby more fully and accurately capturing the stock trend [9]. The specific formula is shown in (3):

$$Q\left(s,a\right) = V\left(s\right) + A\left(s,a\right) - \left| A \right| 1\sum a^{'}A\left(s,a^{'}\right) \tag{3}$$

In this formula: Q(s, a): State-action value function, representing the total value of the agent performing action a in state s, which is the final output Q value. V(s): State value function,which is only associated to state s, representing the inherent value of state s itself, independent of specific actions. A(s, a): Advantage function, indicating the advantage (additional value) of performing action a in state s compared to other actions, that is, A(s, a) = Q(s, a) - V(s). |A|: The size of the action space, a constant, representing the total number of all available actions in state s. |A|1∑a' A(s, a'): The mean of the advantage function, used for centralization constraints to solve the identifiability problem of V(s) and A(s, a), ensuring that the mean of A(s, a) is 0, allowing V(s) to accurately represent the state value. a': All available actions in state s, used to calculate the mean of the advantage function.

3. Research process and data

3.1. Data set selection and preprocessing

This thesis uses the historical trading data of Kweichow Moutai (600519.SS) in the CSI 300 from January 1, 2018 to December 31, 2023 as the sample. The data is obtained from Yahoo Finance and extracted using the yfinance library. The original data has six basic variables: date, opening price, highest price, lowest price, closing price, and trading volume. Therefore, the effective trading day

DOI: 10.54254/2753-8818/2026.35156

data used in this paper totals 1,489 records. To enhance the training effect of the model, hierarchical preprocessing was conducted on the data: Firstly, feature engineering was carried out, calculating the 5-day moving average, 10-day moving average, 14-day relative strength index, and daily return rate from the raw data, thereby rationally constructing a 9-dimensional feature vector. Secondly, data normalization was performed, using MinMaxScaler to normalize all features to the range of [-1, 1]. Thirdly, state construction was done, using a 30-day sliding window to construct the state space.

3.2. Construction of the stock trading environment

In this paper, the StockTradingEnv stock trading environment was constructed. Firstly, the action space of the agent was defined as three actions (0 = sell, 1 = hold, 2 = buy). Secondly, the state space was set as a 30-day by 9-dimensional feature matrix (30 days by 9-dimensional features). Correspondingly, the reward function of the environment was designed: the reward for the action of sell was the difference value between the selling price and the buying price, magnified by 100 times; the reward for the hold action was the difference value between the closing price of the current day and the day before, magnified by 10 times; and there was no immediate reward for the buy action.

3.3. Model training and experimental settings

The three algorithms adopted the same training parameters: the rate of learning was set to 0.001, the discount factor was 0.95, the rate of initial exploration was 1.0, the decay coefficient was 0.995, the rate of minimum exploration was 0.01, the batch size was 32, the capacity of the experience replay pool was 10,000, the training was conducted for 100 rounds, with the target network updated every 10 rounds, thus naturally ensuring the stability of the training process. At last, this paper use the test set to examine the generalization ability of the obtained model and to objectively assess the behavior of the three algorithms.

4. Research results

Table 1. Comparison of performance parameters of the three algorithms

Algorithm Type	Average Total Reward	Cumulative Return Rate	Maximum Drawdown	Win Rate
DQN	848.56	15.2%	18%	52%
Double DQN	1196.89	23.5%	12%	58%
Dueling Double DQN	1645.72	32.1%	8%	65%

The experimental results (Table 1) show that there is a clear progressive relationship in the performance of the three algorithms. Therefore, it can be clearly stated that Dueling Double DQN is the best, Double DQN is second, and DQN is the worst [9]. This is also well confirmed by the analysis of the training process: DQN has the largest reward fluctuation and still has significant oscillations in the later stage of training, thus being the most sensitive to stock price noise and having the most prominent Q-value overestimation problem. In contrast, Double DQN has significantly reduced reward fluctuations and improved training stability, while Dueling Double DQN has the fastest reward growth rate and the smallest fluctuations, indicating that its value-advantage separation structure is more conducive to capturing the long-term trends of stock prices.

DOI: 10.54254/2753-8818/2026.35156

5. Conclusion

Firstly, since reinforcement learning algorithms are naturally suitable for stock prediction and trading decisions, the win rates of the three algorithms discussed in this paper are all higher than random trading, thus effectively demonstrating the effectiveness of reinforcement learning in capturing the nonlinear characteristics of stock prices and learning the optimal trading strategy.

Secondly, as the performance of the algorithms improves in a clear and reasonable manner with the degree of improvement, the defects of the DQN basic model can be analyzed: namely, Q-value overestimation and sensitivity to noise, resulting in low trading returns and large fluctuations. Double DQN solves the overestimation problem by splitting the "action selection" and "value evaluation" processes, thereby improving the robustness of decision-making. Dueling Double DQN combines the advantages of both, and it is the most suitable model for stock prediction among the three algorithms. Thirdly, since Dueling Double DQN performs best in terms of cumulative return, maximum drawdown, and win rate, it naturally becomes the best tool to provide investors with scientific and reliable trading advice, and thus opens up a new path for stock prediction.

References

[1] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. The Journal of Finance, 25(2), 383–417.
[2] China Securities Regulatory Commission. (2018). Guiding opinions on regulating the asset management business of financial institutions.
[3] Jiang, Z., et al. (2020). A deep reinforcement learning framework for the financial portfolio management problem. Applied Soft Computing, 93, 106389.
[4] Zhang, L., & Li, J. (2021). Research on stock trading strategies based on reinforcement learning. Application Research of Computers, 38(7), 2025–2029.
[5] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.
[6] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.
[7] Jiang, Y., Xu, X., & Liang, J. (2020). Stock price prediction using deep reinforcement learning. IEEE Access, 8, 151992–152006.
[8] Liu, Y., Wang, Y., & Long, J. (2022). A comparative study of DQN variants for algorithmic trading. Applied Soft Computing, 123, 108952.
[9] Wang, Z., Schaul, T., Hessel, M., et al. (2016). Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning (pp. 1995–2003).
[10] van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 30(1).