Sock Market Prediction using Q-Learning-Based Machine Learning Model

G Kamalesh
Electrical Engineering Dept.
Indian Institute of Technology Bombay
Mumbai, India
20D070029@iitb.ac.in

Samarth Bansal
Electrical Engineering Dept.
Indian Institute of Technology Bombay
Mumbai, India
200070073@iitb.ac.in

Umang Babu
Electrical Engineering Dept.
Indian Institute of Technology Bombay
Mumbai, India
200070085@iitb.ac.in

***Abstract*— Machine Learning is the important tool in this century to be used in optimizing and predicting many outcomes, and it makes stock market to be the best place where you can use Machine Learning models. Stock Market is very volatile and involves prior experience and real time information to trade. But, it is very difficult to predict the pattern of stocks exactly. And to Trade in stock market using automated Model, we need to design them using deep reinforcement learning (DRL) techniques to forecast stock values and price movement specifically a deep Q-Network (DQN). An algorithm based on a research paper has been implemented to train the model for stock market. The result has been contrasted with results of model trained using time-series data.**

Keywords*— *Optimizing, reinforcement, deep Q-Network

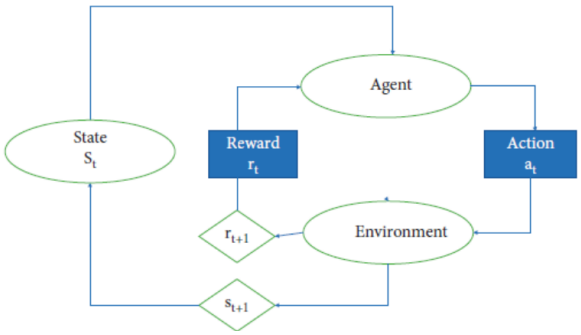
I. Introduction

The stock market makes it challenging for investors and financial analysts to accurately predict stock prices because they are influenced by factors like direct economic indicators like fluctuations in supply and demand, commodity price indexes, and investor behaviour. It makes difficult to build prediction model just based on correlation stock prices, let alone to construct a model for stock price prediction based on this correlation because it is frequently obscure. Older models and assumptions are not appropriate because they do not include volatility involved with stocks. Multiple theories have been introduced to predict price but updated or discarded with time because new factors been discovered that influence stock price. Out of many papers, it has been established that deep learning model can predict the stocks more accurately. This is also divide by type of approaches like Model-based and non-policy approaches, such as TD-learning and Q-learning and are utilized in the process of resolving problems pertaining to discrete area optimization.

Multiple research papers discussing method to forecast stock prices accurately using Q-learning has been surveyed, and a model based on one of the paper has been implemented in later section of the report. And the following sections contain the result and observation for the models.

II. Literature Survey

Reinforcement learning involves an agent that receives information from its environment and adjusts its behavior accordingly to maximize rewards. The agent takes actions that impact the environment and receives immediate feedback in the form of rewards or punishments. The goal is to find a decision-making policy that results in the highest possible reward. This process is influenced by four key factors, including rewards, environment, policies



(rules for behavior), and value functions (estimations of the worth of different states and actions). The reward function is the ultimate objective for the agent, while the value function helps determine the best actions to take. At a high level, the reinforcement learning procedure involves the agent taking action in response to the current state. The system then receives a signal in the form of a reward, which directs the behavior of the agent in the subsequent time step and adjusts the state using a probability function. This process is iterative, with the agent continually learning and adapting its behavior based on the feedback it receives from the environment. The ultimate goal is to find a decision-making policy that maximizes the total reward over time.

Reinforcement learning aims to maximize the agent's reward in a limited number of actions by mapping environmental states to actions. The Markov decision process (MDP) is a mathematical model used to describe the reinforcement learning problem. A MDP is represented by the tuple (S, A, ρ, f), where S is the set of all possible environmental states, st ∈ S denotes the current state at time t, A is the set of possible actions the agent can take, at ∈ A denotes the agent's action at time t, ρ: S × A ⟶ R is the reward function which summarizes the benefits that a possible action at can obtain in a given state st, and rt ∼ ρ(st, at) is a reward or punishment table. The probability distribution function for state transitions is f: S × A × S ⟶ [0, 1], and st+1 ∼ f(st, at) denotes the probability of successfully transitioning from state st to state st+1.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{T-1} \gamma^k r_{t+k+1}. \quad (1)$$

$$Q^{\pi}(s,a) = E\left[R_t | s_{t=s}, a_{t=a}, \pi\right]. \quad (2)$$

$$Q^{\pi}(s,a) = \max_{\pi} E\left[R_t | s_{t=s}, a_{t=a}, \pi\right]. \quad (3)$$

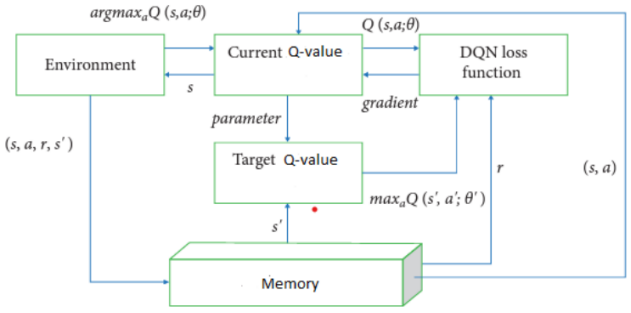
$$Q^{*}(s,a) = E_{s' \sim S}\left[r + \gamma \max_{a'} Q(s',a') | s,a\right]. \quad (4)$$

Fig. 1 The above plots shows the stock price of Dabur-NSE observed during the period of Jul 2012 to Jul 2014 and split in train and test data by date marker on Jun 2013 and same data will be used for both DQN and Double DQN models

The Q-value function is typically solved using the iterative Bellman equation (5) until it converges to the optimal Q-value function Q∗. The optimal policy for which Q* function converges is given by equation (6) In practice, iterating the Bellman equation for large sample spaces is impractical due to the vast number of calculations required.

III. Method

Deep Q-Network (DQN) is a type of deep reinforcement learning algorithm that combines the traditional Q-learning algorithm with deep neural networks. It was introduced by a team of researchers from Google DeepMind in 2015, and since then has become one of the most popular algorithms in the field of deep reinforcement learning. It is based on Q-learning model, the major difference here is that instead of using q-table we use neural networks for updating Q-value, which maps input states to action and Q-value. The agent takes action depending on the current state of the environment and receives reward from the environment.
The experience replay is used to learn from previous experiences and is used to store the previous states, actions, rewards, and next states. The data from the replay memory is sampled randomly and fed to the train network in small batch sizes to avoid overfitting. In deep Q-learning, the Convolutional Neural Network (known as Q-Network) is used to learn the expected future reward Q-value function (Q(st,at)). One major difference between the Deep Q-Network and the basic Q-learning algorithm is a new Target-Q-Network, which is given by: Double Deep Q-Network In DQN there is an issue of overestimation because it takes the max of all action in each step, as the loss accumulates over time it leads to overestimation.



Double deep q-Network(DDQN) addresses this problem by using two neural networks with the same structure as DQN: the main network and the target network. At each step, all the action-value pairs for all possible actions in the current state are taken from the main Q-Network, which is updated at each time step. Then, an argmax is taken over all the state-action values of such possible actions, and the state-action value which maximizes the value is selected.

This selected state-action pair is used to update the main Q-Network with the corresponding value taken from the target Q-Network. This process helps to overcome both the problems of overestimation and instability in Q-values. The Q-target for updating the main Q-Network is calculated using the above equation, which takes into account the reward received at the current step, the discount factor gamma, and the state-action value corresponding to the selected action at the next state from the target Q-Network. By using DDQN, the Q-values are estimated more accurately, which can lead to better performance in the reinforcement learning task.

IV. Results and Observations

Training the model with input data of stock prices, and as we know in RL models, it has two parameters, which it tries to minimize and maximize the loss and reward parameter respectively. The output of our trained model has been shown below.

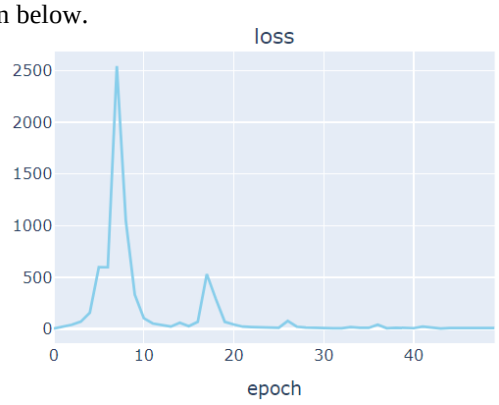


Fig. 2. This figure shows the changes in the loss versus number of epoch for the DQN based RL model

DQN: train s-reward 18, profits 389, test s-reward 4, profits 140

Fig. 3. This figure shows the reward for the agent versus number of epoch for the DQN based RL model

We have used the Dabur-NSE Stock prices over the period of time and split the data in two halves and former part was used for training and later part for testing the trained models

As we observe the fig[2] that loss reduced significantly when number of epochs crossed 20 which is important for such models which are meant to deployed in real-world and involve finance. Same observations can be made from fig[3] for our model making reward based on number of epoch, this training also has to cross certain number of epochs to be sure that the model is making profit(reward is in positive), for this model, minimum number of epochs is greater than 25.

The profit has been around 140 for DQN based models at the end of epochs. This will make the model stable and appropriate for further improvements can be done according to different kind of stocks.

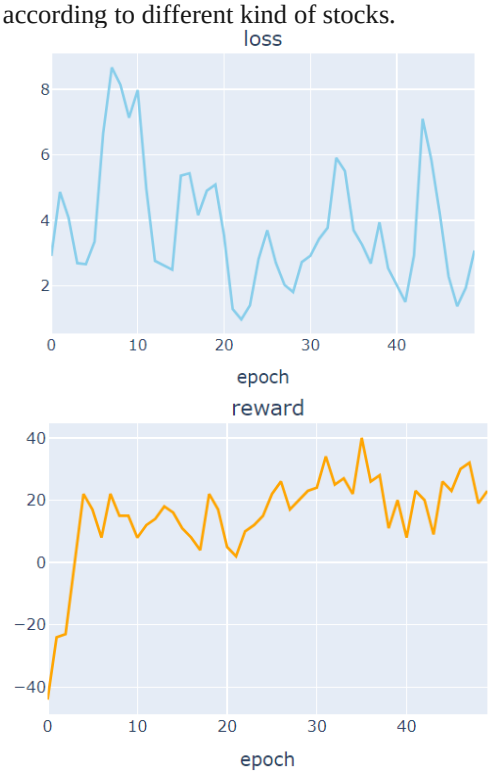


Fig. 4. The loss been plotted against number of epochs for The Double DQN based RL model

Double DQN: train s-reward 37, profits 239, test s-reward 24, profits 214

The above results observed from Double DQN for reward and loss and total profit it has made after cycle of epoch. According to the output a profit of 214 has been made that is approximately twice the profit made using model trained based on DQN and the minimum number of epochs needed for positive reward and loss threshold is around 10, which is significantly less than DQN based model.

V. Conclusions

We were able to predict stock prices using reinforcement learning, out of which dueling double deep q-network performed best compared to other models. Based on the level of accuracy of results that we got, this model worked as planned and can be further improved to be deployed for real-world financial market. We are planning to work on it further by making portfolio management system using bandit algorithms, and integrate it with this model, to make the function as automated trading bot.

VI. Statement of Contribution

G Kamalesh was responsible for the literature survey and choosing appropriate method to be implemented, comparing the result with observed data in discussion with other group members.

Samarth Bansal was responsible for searching and pre-processing of the datasets, finding code references and observation of data results and the performance of models in discussion with other group members.

Umang Babu was mainly responsible for building the RL model and optimizing functions, transforming different parameters and collecting the results with the help from other members

References

[1] Bajpai, S. (2021). Application of deep reinforcement learning for Indian stock trading automation. ArXiv. /abs/2106.16088B. Rieder, *Engines of Order: A Mechanology of Algorithmic Techniques*. Amsterdam, Netherlands: Amsterdam Univ. Press, 2020.

[2] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling Network Architectures for Deep Reinforcement Learning. ArXiv. /abs/1511.06581

[3] Junhao Zhang, Yifei Lei, "Deep Reinforcement Learning for Stock Prediction", Scientific Programming, vol. 2022, Article ID 5812546, 9 pages, 2022. https://doi.org/10.1155/2022/5812546

[4] C. Yoon. Double Deep Q NetworksTackling maximization bias in Deep Q-learning, 2019. Available online at https://towardsdatascience.com/double-deep-q-networks-905dd8325412