Indian Institute of Technology Bombay

Reinforcement Learning To Optimise Stock Trading Strategy And Thus Maximise Investment Return

Finsearch, Finance Club

Team Members:
Hrishkesh S
Jatin Kumar
Sangeetha D
Oviya S

Date of Submission:
August 2024

Mentor name:
Likhita Sree

Contents

Chapter 1

Introduction

Predicting market movements is a challenging yet crucial task in financial trading. This study trains an RL agent using historical market data, evaluates its performance on a testing dataset, and fine-tunes the model for improved results. While RL holds promise, careful application in real-time trading is essential due to market volatility. A comparison of returns and risks between RL and traditional models like ARIMA and LSTM will offer insights into their practical use. Segmenting the data effectively will further enhance the analysis, drawing on relevant research for guidance.

1.1 Objectives

Our main objective is to evaluate and compare the performance of traditional algo- rithms like ARIMA and LSTM models with a reinforcement learning approach in the context of financial forecasting and trading.
(a) Evaluating the effectiveness of traditional forecasting methods, such as ARIMA and LSTM, compared to reinforcement learning (RL) algorithms.
(b) Conducting a comparative study between LSTM and RL using data from the past six weeks, revealing that RL outperforms LSTM in terms of predictive accuracy and overall performance.
(c) Analyzing the strengths and limitations of both approaches, with a focus on balancing returns and risk factors. This analysis offers a comprehensive comparison of their practicality in real-world trading scenarios.

Chapter 2

2.1 Literature Review

2.2 Time Series Forecasting: ARIMA vs LSTM

2.2.1 ARIMA

The Autoregressive Integrated Moving Average (ARIMA) model is an advanced version of the Autoregressive Moving Average (ARMA) model, integrating the concepts of Autoregressive (AR) and Moving Average (MA) processes to provide a unified framework for analyzing time series data. The ARIMA model is characterized by the notation ARIMA(p, d, q), where the parameters are defined as follows:

- **AR (Autoregression):** This component involves constructing a regression model that examines the relationship between a given observation and previous observations from a specified number of time steps in the past, denoted as p. The autoregressive aspect captures the influence of past values on the current value.
- **I (Integrated):** The "integrated" component focuses on transforming the time series data into a stationary series. This is achieved by differencing the observations at different time points, with d representing the number of differences required to achieve stationarity. The goal is to stabilize the mean of the series.
- **MA (Moving Average):** This part of the model accounts for the relationship between observations and the residual errors obtained when applying a moving average model to lagged observations. The parameter q denotes the order of the moving average component, reflecting the number of past residuals used to model the current observation.

2.2.2 LSTM

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) designed to address the limitations of conventional RNNs, particularly their difficulty in remembering information over extended sequences. LSTMs are capable of retaining information from earlier stages for future use, making them well-suited for tasks requiring long-term memory.

An LSTM network consists of interconnected cells that function as data storage units. These cells are structured to facilitate the flow of information through a series of gates, which regulate the addition, retention, or removal of data. The primary components of an LSTM cell include:

- **Forget Gate:** This gate determines which information should be discarded from the cell state. It outputs values between 0 and 1, where 1 indicates that the information should be retained, and 0 means it should be ignored.
- **Memory Gate:** Comprising an input gate (sigmoid function) and a tanh layer, the memory gate is responsible for selecting the data to be stored in the cell state. The sigmoid function filters the incoming data, while the tanh layer creates a candidate for updating the cell state.
- **Output Gate:** This gate decides the output of the LSTM cell based on the current state of the cell and the processed data. It filters the data to determine what should be output from the cell.

The gates within each LSTM cell are formed by sigmoidal neural network layers, which produce values ranging from 0 to 1. These values determine whether data segments should be passed along the network, with values closer to 0 blocking the passage and values closer to 1 allowing it.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment. The objective of RL is to maximize cumulative rewards by selecting the best actions in various states. This learning process involves trial and error, with the agent receiving feedback in the form of rewards or penalties based on its actions.

One of the main advantages of using RL in stock trading is its adaptability to changing market conditions. RL algorithms are capable of learning from historical data and adjusting their strategies accordingly. They can incorporate a wide array of information, including technical indicators, news sentiment, and social media trends, to make well-informed trading decisions. This adaptability often makes RL a superior alternative to traditional trading algorithms.

However, RL in stock trading also presents several challenges. These include the need for careful risk management, handling noisy and uncertain market data, and avoiding overfitting to historical data. Despite these challenges, when trained accurately, RL models can outperform traditional algorithms by developing more effective trading strategies.

2.4 Findings from Recent Studies

Recent research has explored the application of Deep Q-Networks (DQN) in stock trading, as discussed in the study referenced in [1]. The authors evaluate the performance of DQN using extensive real-world datasets and highlight its unique advantage of facilitating direct stock trading without requiring additional optimization steps. This contrasts with other supervised learning methods that often need further optimization.

Remarkably, even with a relatively small dataset of only a few hundred samples, variants of reinforcement learning algorithms based on Q-learning have demonstrated the ability to develop trading strategies that yield positive returns on average. This finding underscores the potential of reinforcement learning techniques, particularly DQN, in the field of stock trading.

Chapter 3

3.1 Data utilized:

This study used two different datasets - one to train and one to test. We kept the training dataset the same throughout using NIFTY 50 data from January 2010 to June 2019. To test, we checked our model with two separate datasets to get a full picture across different time periods. The first test dataset looked at the newest six weeks of NIFTY 100 data giving us a quick view. The second dataset had NIFTY 50 data from last year letting us see how well the model worked over a longer time.

3.2 Data processing

In our data prep step, we use the Min-Max scaling method from scikit-learn to normalize our data. This limits the values to fall between 0 and 1. We then use this normalized data to train our model . But when we figure out profits during training, we undo the scaling with the same method. This brings the data back to its original scale, which lets us calculate profits .

3.3 The benchmark models, ARIMA and LSTM

In finance, ARIMA and LSTM are commonly used as benchmark models when testing reinforcement learning (RL) strategies. These two models are chosen because they have been around for a while and have also been proven effective in time series analysis. ARIMA is often used because it is simple and easier to interpret, which can be helpful in understanding market dynamics. Similarly, LSTM is utilized because it is good at capturing complex, nonlinear patterns in data. By comparing RL strategies to these two standard models, researchers can better understand the effectiveness of RL strategies in finance, and if they offer a significant improvement. This experiment can also speak to the robustness and reliability of RL techniques in the highly turbulent and complicated financial domain like stock trading.

3.4 Our Implementation using the LSTM Algorithm

Our implementation consists of two distinct phases: an LSTM-based prediction model and a Q-learning-based Reinforcement Learning (RL) strategy.

LSTM Model Implementation The Long Short-Term Memory (LSTM) model was implemented to predict future prices based on past data. We first normalized the training and testing data using MinMaxScaler to scale the prices between 0 and 1, ensuring that the LSTM could process the data effectively. We then prepared the data by creating sequences of inputs with a window size of 1, where each sequence's corresponding output was the next price point.

We constructed an LSTM model with one LSTM layer of 50 units, followed by a Dense layer for the output. The model was compiled with the 'adam' optimizer and 'mean_squared_error' loss function. We trained the model for **50 epochs** with a batch size of 32. After training, we made predictions on both the training and testing data, and then inversely transformed these predictions back to the original price scale.

Q-Learning Model Implementation For the RL strategy, we normalized the data again and defined the state function, which considers the previous 5 time steps as the state. The actions were defined as either '**Hold**' (0) or '**Buy**' (1). A Q-table was created to store Q-values for each state-action pair, and it was initialized with Xavier initialization.

An epsilon-greedy policy was used for action selection to balance exploration and exploitation. The Bellman equation was implemented to update the Q-values during training. We trained the RL model for 30 epochs, updating the target Q-table every 10 epochs and decaying epsilon to gradually shift from exploration to exploitation.

3.5 The Model Specifications

State Space The state space in the RL model is defined by the previous 5 time steps of normalized price data, creating a state vector that captures the recent market behavior. This lookback window allows the model to make decisions based on short-term trends.

Action Space The action space consists of two actions:

- Hold (0): The agent does nothing and keeps the current number of shares
- Buy (1): The agent buys one additional share. These actions are designed to simulate a simple trading strategy where the agent can either buy more of the asset or hold its current position

Rewards The reward is defined as the next normalized price point. This means that the agent is incentivized to choose actions that will lead to higher future prices. During the training, the reward for each action is calculated based on the next state (next time step) price, aiming to maximize the return by choosing the optimal action at each step.

The RL model utilizes a replay buffer to store past experiences (state, action, reward, next state), allowing the agent to learn from a random sample of experiences during each training iteration. This helps break the correlation between consecutive experiences and stabilizes learning.

Furthermore, the LSTM model's predictions are used to simulate a trading strategy where profits are accumulated over time based on the predicted price changes. The profits and returns are calculated, allowing for a comparative analysis of the LSTM-based trading strategy and the Q-learning-based RL strategy.

3.6 Results and Conclusion

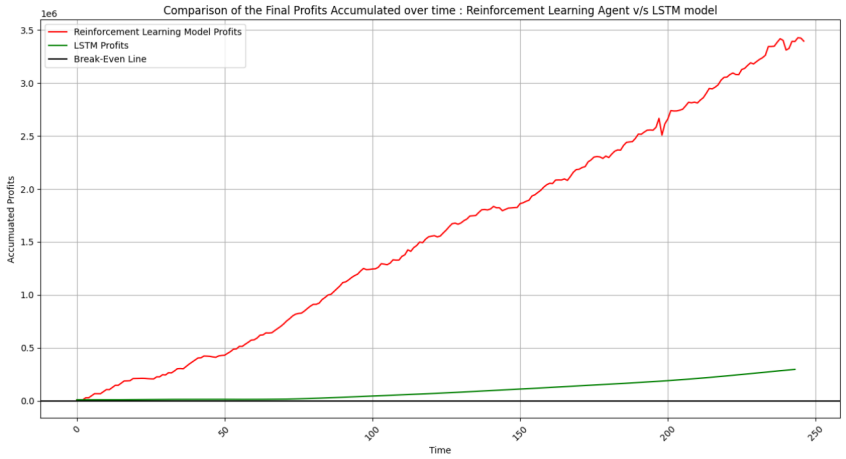

Figure 3.1: Comparison of Performances on 1 year Data

The above plot shows the comparison of the profits generated over a span of the past 6 weeks by both LSTM and the RL Agent.Here also the agent outperforms the LSTM model by a significant margin as it can adapt to dynamic conditions as we have incorporated the Q table which can be updated during iterations and thus getting the better results.

From the plots provided above, it is clear that the RL model outperforms the given LSTM model by a large margin of the final profit.We choose to compare a LSTM model with our Deep Q learning agent for the final comparison. The results were promising as the LSTM model was outperformed by our RL agent by a large margin.

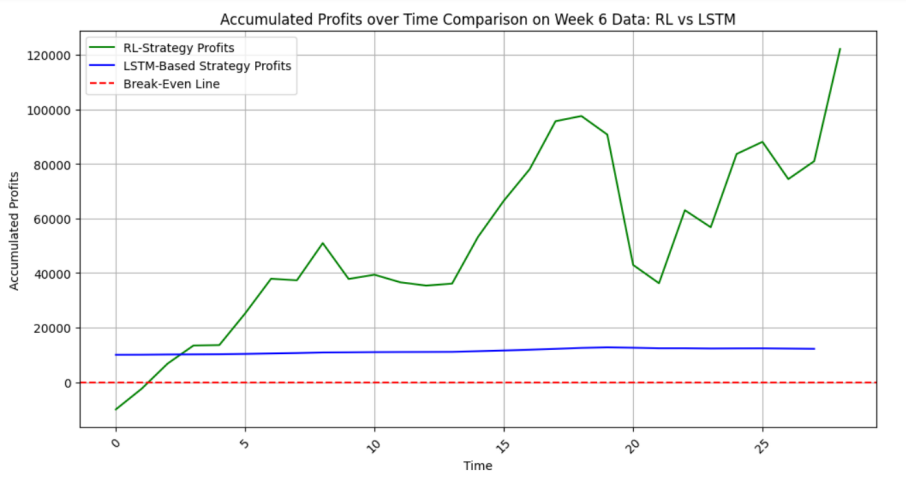

Figure 3.2: Comparison of Performances on 6 weeks data

3.7 References

- Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, Massachusetts, 2014, 2015.
- https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
- https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average
- https://ieeexplore.ieee.org/document/9964190
- https://dl.acm.org/doi/pdf/10.1145/3383455.3422540
- https://www.bioinf.jku.at/publications/older/2604.pdf
- https://www.analyticsvidhya.com/blog/2021/04/q-learning-algorithm-with-step-by-step-implementa
- https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecastingusing-time-series-analysis
- https://www.ijrte.org/wp-content/uploads/papers/v8i6/F8405038620.pdf
- https://arxiv.org/html/2407.09557v1