Smart Stock Market Prediction Using Q-Learning

Arbaz Attar, Shashank Gopalakrishna, Kathirvel Manivannan Vimaladevi

1 Abstract

This report presents the final findings of our project, which leverages Q-Learning, a reinforcement learning technique, to develop a stock trading agent capable of making decisions in a highly dynamic and volatile market environment. By training on historical stock price data, the agent learns to select from three possible actions—Buy, Sell, or Hold—to maximize cumulative rewards. Our approach integrates market simulation and dynamic decision-making to capture the complexities of real-world trading. The results indicate that Q-Learning can effectively model stock trading decisions, offering a promising method for enhancing financial decision-making processes.

2 Introduction

The objective of this project was to design and implement a Q-Learning-based trading agent capable of making informed decisions in a simulated stock market environment. By using historical stock price data, the agent is trained to determine the best possible actions (Buy, Sell, Hold) based on past market behavior. Unlike traditional machine learning approaches, which typically rely on supervised learning, reinforcement learning provides an environment where the agent learns through interaction, making it ideal for complex systems like the stock market.

Stock market prediction is a challenging task due to the volatile and stochastic nature of financial markets. Many traditional models struggle to adapt to sudden market changes. By incorporating Q-Learning, our approach allows the agent to dynamically adjust its strategy, learning from both successes and failures. This method holds potential for improving financial decision-making systems and guiding future research into autonomous trading systems.

3 Background and Literature Review

3.1 Reinforcement Learning in Financial Markets

Reinforcement learning (RL) has become increasingly popular in the financial sector, especially for its ability to model sequential decision-making problems. It has been applied to a wide range of financial tasks, including portfolio optimization, asset pricing, and algorithmic trading. RL techniques, particularly deep Q-Learning, have shown significant promise in stock price prediction and trading strategy development. Studies have demonstrated the potential of RL to outperform traditional methods like moving averages, regression models, and heuristic-based strategies. Notable applications include designing intelligent trading agents capable of automatically adapting to changing market conditions, thereby reducing human bias and error.

3.2 Q-Learning Basics

Q-Learning is a foundational reinforcement learning algorithm that employs a model-free approach to solve decision-making problems. By learning an action-value function (Q-function), the agent determines which actions to take to maximize cumulative future rewards. In the context of stock trading, Q-Learning helps agents learn to make decisions that align with long-term profitability. Since the stock market is highly unpredictable, Q-Learning's ability to adapt to unseen states through exploration and exploitation is particularly valuable.

3.3 Tools Used

- **Python**: Programming language for model implementation.
- **Libraries**: NumPy, Pandas, TensorFlow, and Matplotlib for data handling, training, and visualization.
- **Simulation Environment**: OpenAI Gym, will be customized for stock market simulation.
- **Data Source**: Historical stock prices from Yahoo Finance.

4. Model Implementation

We processed historical stock prices from **S&P 500 Finance**, normalized the data, and designed an environment where the agent could "learn" by interacting with the market. Through preprocessing and initial tests, we ensured the compatibility of our model with the chosen Q-Learning framework.

Environment

The trading environment simulates a stock market scenario for the Q-Learning agent. The environment comprises the following components:

State Space:

- The state is defined as an **n-day sliding window** of normalized closing prices, representing short-term trends in stock market data.
- Example: If $n = 5$, the state for day t is:

$$[\text{price}_{t-4}, \text{price}_{t-3}, \text{price}_{t-2}, \text{price}_{t-1}, \text{price}_{t}]$$

Action Space:

- The agent has three possible actions:

  - **Buy**: Purchase one unit of stock at the current price.
  - **Sell**: Sell one unit of stock at the current price.
  - **Hold**: Take no action.

- The environment ensures that the agent cannot sell stock if it holds none, avoiding invalid states.

Rewards:

- The reward is calculated based on profit:

$$\text{Reward (Sell Action)} = \text{Sell\_Price} - \text{Buy\_Price}$$

- For Hold or invalid actions, Reward = 0.

Episode Initialization:

- Each episode represents a single trading session and begins with:
  - The agent holding zero stock.
  - A random starting index in the dataset for diverse training scenarios.

Terminal Condition:

- An episode ends when:
  - The agent has iterated through the entire dataset.
  - The agent reaches a pre-defined maximum number of trading steps.

Market Constraints:

- Transaction costs, such as brokerage fees, were not included in this simulation but can be added for real-world scenarios.

- No constraints on the number of shares bought or sold were applied.

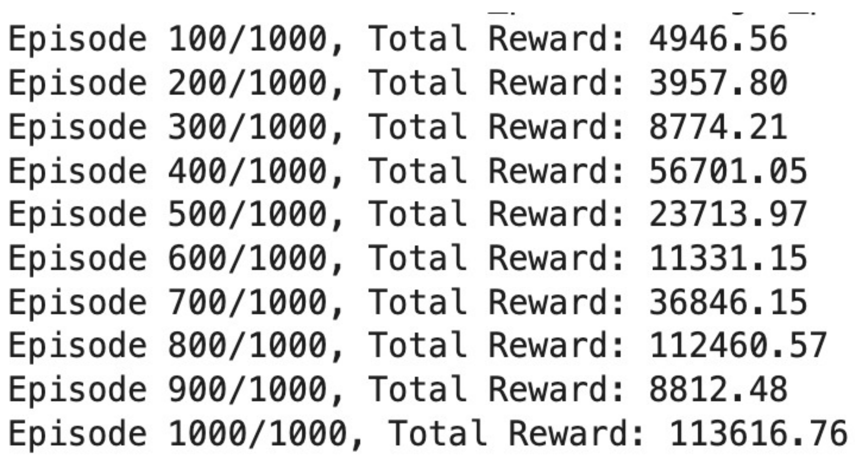
5. Metrics and Evaluation

To evaluate the performance of our Q-Learning model, we focused on several key metrics that reflect the agent's decision-making and learning efficiency:

1. **Cumulative Rewards**: This metric represents the total rewards accumulated per episode. It was visualized using Matplotlib to observe how well the agent learned to maximize its rewards across episodes, tracking both short-term and long-term performance trends.

2. **Q-Value Stability**: To ensure the Q-values were converging correctly and not oscillating, we measured the Mean Squared Error (MSE) between the old and updated Q-values. This allowed us to track the stability of the learning process and confirm that the agent was gradually improving its value estimates.

3. **Exploration vs. Exploitation Ratio**: We monitored how the agent balanced exploration (trying new actions) versus exploitation (selecting the best-known actions). By analyzing the shift in this ratio over time, we gained insights into how well the agent transitioned from exploring the environment to exploiting learned strategies as it became more confident in its actions.



```
Episode 100/1000, Total Reward: 4946.56
Episode 200/1000, Total Reward: 3957.80
Episode 300/1000, Total Reward: 8774.21
Episode 400/1000, Total Reward: 56701.05
Episode 500/1000, Total Reward: 23713.97
Episode 600/1000, Total Reward: 11331.15
Episode 700/1000, Total Reward: 36846.15
Episode 800/1000, Total Reward: 112460.57
Episode 900/1000, Total Reward: 8812.48
Episode 1000/1000, Total Reward: 113616.76
```

Figure 1:

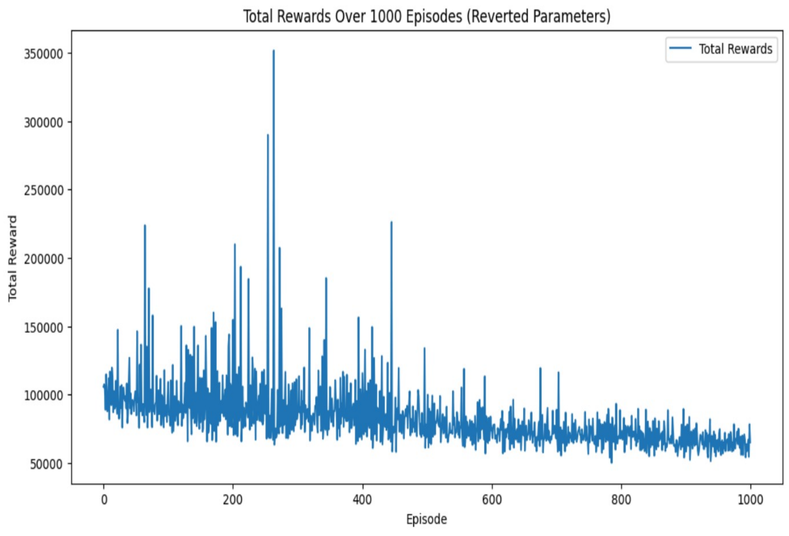


Figure 2: Total Rewards over 1000 episodes

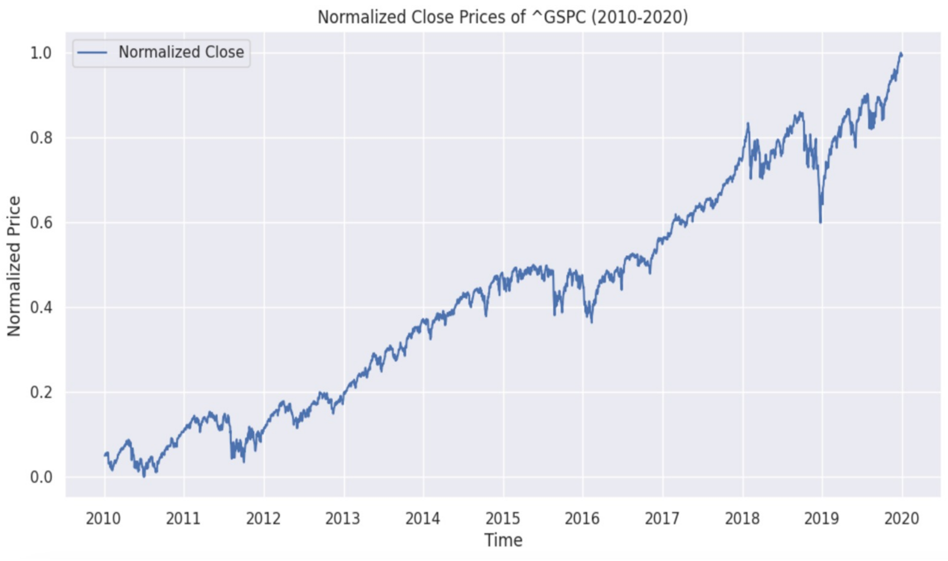


Figure 3: Normalised Close Prices

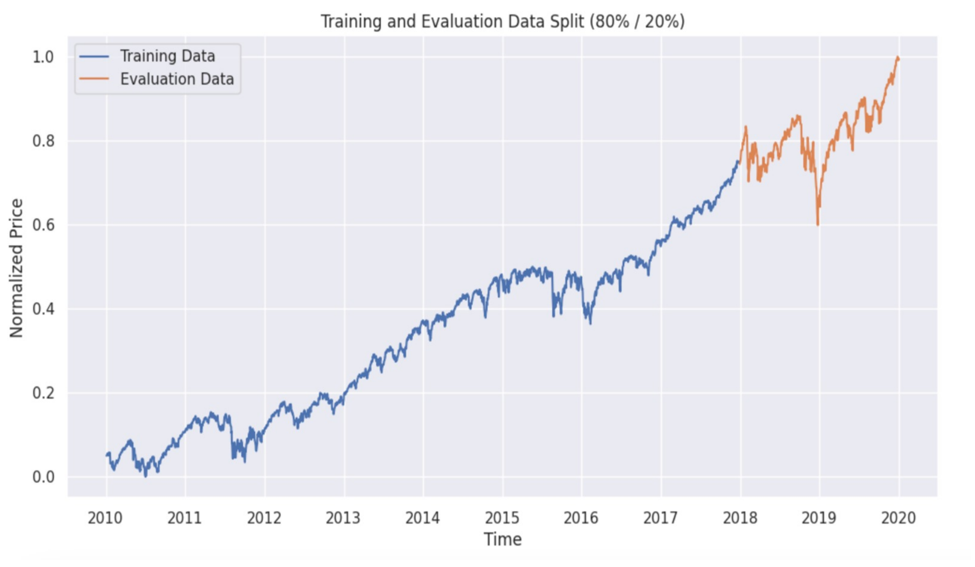


Figure 4: Training and Evaluation Datasplit

6. Key Results

Observations

- Initial runs with **Alpha = 0.1, Epsilon = 1.0** showed high rewards and epsilon decayed over episodes.

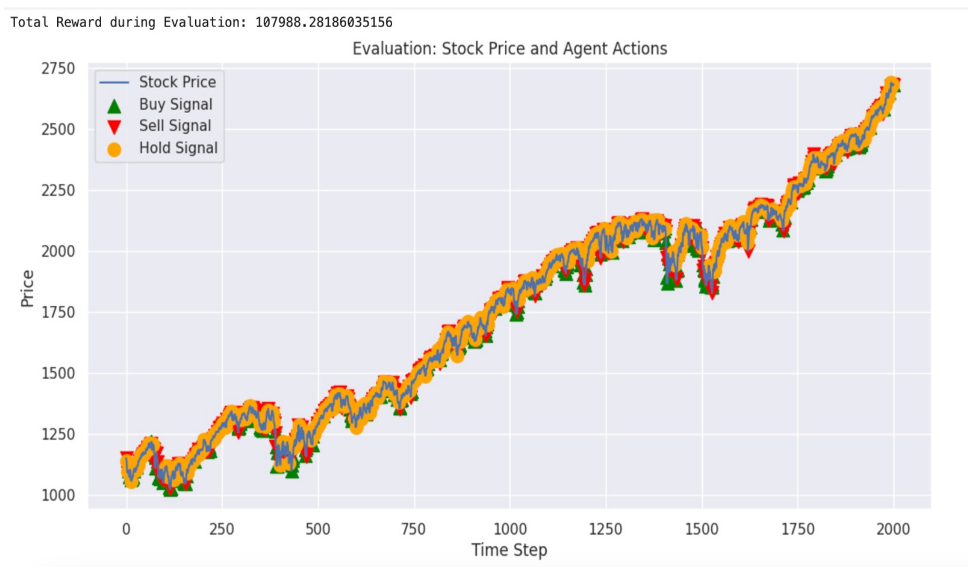
Evaluation Phase

- **Total Reward During Evaluation: $113616.76**.
- Buy signals aligned with upward price trends.(Depicted in Figure 4)
- Sell signals aligned with declines, demonstrating the agent's ability to identify profitable trends. (Depicted in Figure 4)

7. Challenges and Lessons Learned

Challenges

- Balancing exploration vs. exploitation for optimal trading strategies. Achieving the right trade-off was difficult, as excessive exploration led to instability while too much exploitation caused missed opportunities for better strategies.
- Fine-tuning hyperparameters for both short- and long-term profitability. Parameters like learning rate, discount factor, and exploration rate had to be carefully adjusted to optimize both immediate rewards and long-term gains.
- Managing the volatility and unpredictability of the stock market. Market noise often led to suboptimal decisions, requiring constant model adjustments.



```
Total Reward during Evaluation: 107988.28186035156
```

Figure 5: Buy, Sell Indicators marked

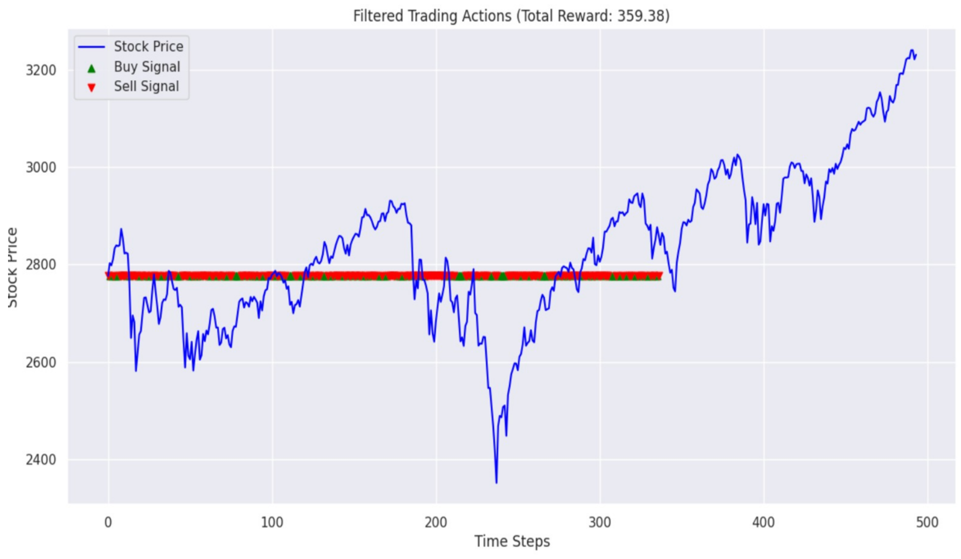


Figure 6: Filtered Trading Actions

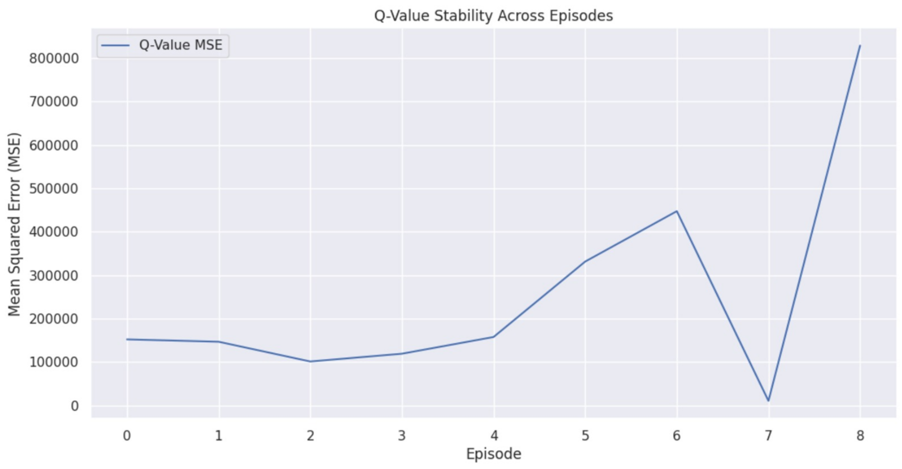


Figure 7: Q Value Stability

- Designing an effective reward function. It was challenging to balance immediate rewards with long-term profitability to ensure that the model learned profitable strategies.

Key Learnings

- Lower learning rates provided stability, while higher rates offered faster adaptability but also increased the risk of instability.
- Higher exploration rates during initial training phases significantly enhanced learning, allowing the model to explore a variety of strategies.
- Exploration strategies, such as epsilon-greedy, greatly impacted the model's overall profitability. Gradual decay of exploration helped transition the model towards exploitation as training progressed.
- Continuous fine-tuning and iterative refinement were necessary for maintaining optimal performance in the dynamic stock market environment.

8. Completion Summary

What We Have Done

- **Data and Training:**
  1. We trained a Q-Learning model using S&P 500 stock data from Yahoo Finance.
  2. The model successfully learned to make Buy, Sell, and Hold decisions, with a focus on cumulative rewards.
- **Metrics Evaluated:**
  1. Cumulative Reward: The model achieved $113616.76 during evaluation (final test phase with no random actions).
  2. Exploration vs. Exploitation: We tuned epsilon to shift from exploration (trying random actions) to exploitation (using learned strategies).

3. Q Value Stability: Q Value Stability was computed and the graph was charted using MSE(Mean Square Error) of the Q Values stored in tables.
4. Random Policy: Random Policy model was also implemented as a baseline model and compared against Q-Learning Policy model. Q-Learning model was clearly performing much better.

- **Deliverables Created:**
1. A fully trained Q-Learning model capable of making optimized stock trading decisions.
2. Graphs illustrating the Buy, Sell, and Hold actions made by the agent, aligned with corresponding stock price trends. These visualizations helped demonstrate how the model's actions responded to market movements, providing clarity on its decision-making process and overall performance.

9. Proposed Improvements

- **Hybrid Parameter Tuning**:
  - Start with higher exploration (**Epsilon = 0.2**) and learning rate (**Alpha = 1.0**).
  - Decay these parameters over time to stabilize training.
- **Additional Features**:
  - Use technical indicators like moving averages and Relative Strength Index (RSI).

10. Conclusion

Key Achievements

- Successfully implemented a Q-Learning agent that makes Buy, Sell, and Hold decisions based on historical market data, simulating real-world market conditions.
- Achieved a Total Reward of 113,616.76 during evaluation, demonstrating the agent's ability to adapt to market fluctuations and maximize profitability.
- Effectively balanced exploration and exploitation, enabling continuous learning while maintaining profitability.
- Utilized key evaluation metrics such as cumulative rewards, Q-value stability, and exploration-exploitation ratio to track the agent's learning process and performance, ensuring consistent improvement.

Recommendations

- Implement hybrid parameter tuning for better long-term performance and adaptability to changing market conditions.

- Incorporate additional features like sentiment analysis or external indicators to further enhance the agent's decision-making capabilities.
- Broaden the training scenarios to improve the agent's ability to generalize and perform well under varying market conditions.

11.Acknowledgement

We express our sincere gratitude to Dr. Xin Zhang for the invaluable feedback and mentorship, which significantly propelled this research forward. We also wish to acknowledge the collective efforts and dedication of the authors, Arbaz Attar, Shashank Gopalakrishna, and Kathirvel Manivannan Vimaladevi, whose hard work made this research possible.

12.Bibliography

Related Research Publications

- "A study of forecasting stocks price by using deep Reinforcement Learning" - Razib Hayat Khan, Jonayet Miah, Md Minhazur Rahman, Md Maruf Hasan, Muntasir Mamun - 2023 IEEE World AI IoT Congress (AIIoT).
- J. W. Lee, E. Hong and J. Park, "A Reinforcement Learning Based Approach to Design of Intelligent Stock Trading Agents", Proceedings of the 2004 IEEE International Engineering Management Conference, pp. 1289-1292, 2004.
- M. S. Roobini, K. Babu, J. Joseph and G. Ieshwarya, "Predicting Stock Price using Data Science technique", Proceedings of the Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), pp. 1013-1020, 2022.
- A. Pastore, U. Esposito and E. Vasilaki, "Modelling stock-market investors as Reinforcement Learning agents", Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems, pp. 1-6, 2015.