Final Project:
Stock Market Simulation and Reinforcement Learning

submiters: Itay Asher - 208199232
Shay Rozin- 206946519

Abstract

In this project, we aimed to enhance our Deep Q-Learning (DQN) stock-trading agent by investigating the impact of introducing short-selling capabilities. The core question we addressed was whether expanding the agent's action space—by allowing it to execute short positions in addition to buying and selling—would improve its performance in volatile and dynamic market conditions, despite the increased complexity of exploration. By adding short-selling, the agent is equipped to profit not only from rising markets but also from market declines, potentially maximizing returns. However, our results indicate that while this expanded action set increases the agent's ability to adapt and optimize its decisions during downturns, it also complicates the learning process. The larger exploration space requires more sophisticated exploration strategies and increases the risk of suboptimal actions during training. Nonetheless, the inclusion of short-selling shows promise in enhancing the agent's overall performance.

Introduction

In recent years, the application of machine learning to financial markets has gained significant attention, for example the use of reinforcement learning techniques like Deep Q-Learning (DQN) for automated stock trading. DQN, a model-free algorithm that approximates the optimal policy for decision-making through interactions with an environment, has the potential to be successful when applied to trading scenarios where an agent must learn to buy, sell, or hold based on market data. We saw in class and in our midterm project that DQN can approximate good results on states he hasn't visited yet by using the neural network. Which can be useful when dealing with an environment like stock trading where it is likely that the agent will encounter unfamiliar states frequently.

The goal of our project is to expand the range of actions available to the agent, providing more opportunities for it to generate profit by short-selling in addition to the traditional long-selling. Short-selling is a technique whereby an agent profits from declining asset prices by selling borrowed assets and repurchasing them at a lower price. While this action introduces new possibilities for profit in bear markets, it also expands the agent's action space, increasing the complexity of the exploration process. This introduces a key research question: Can the addition of short-selling in a DQN-based trading agent lead to better performance across varying market conditions, despite the increased complexity of exploration?

This question is significant as it addresses a gap in the given reinforcement learning-based trading strategy, which overlooks the advantages of short-selling. By exploring this direction, we aim to better understand whether more sophisticated trading strategies can be effectively learned in dynamic and unpredictable markets. The novelty of our approach lies in expanding the agent's decision-making capabilities, thus broadening the scope of financial scenarios it can profit from.

MAIN FINDINGS

The main findings of our project indicate that expanding the DQN agent's action space to include short-selling improved its ability to adapt to volatile market conditions. By introducing short-selling, the agent was able to profit from both rising and falling markets, resulting in faster increases in average state value and longer episodes. Additionally, the agent's sensitivity to transaction costs was demonstrated, as higher broker fees for short-selling led to a decrease in the frequency of these actions, affecting overall performance.

Data

In this project we are given a dataset of the Yandex company from the Russian stock market for the period 2015-2016. The data set contains minute by minute bars (M1) which includes the close price, open price, high price and low price of each minute. In addition to that the data also adds the volume of transactions in each minute.

Yandex is a leading Russian multinational corporation specializing in Internet-related products and services. Founded in 1997, Yandex is best known for its search engine, which holds a significant market share in Russia and several other countries. The company operates a diverse array of services, including online advertising, cloud computing, navigation, mobile applications, and e-commerce. Between 2015 and 2017, Yandex focused on enhancing its core services while exploring new markets and innovations. The company also navigated a complex economic environment marked by sanctions and currency fluctuations, which impacted many businesses in Russia.

In our project we preprocess the data in several ways. First, we filter out every minute where the change falls below a specified threshold ($\Delta < 10^{-8}$). Then, we stitch together all the remaining bars open and close by setting the open price of each bar to match the close price of the previous bar.

This preprocessing is useful for two primary reasons. First, filtering out minute changes below a certain threshold (e.g., $<10^{-8}$) reduces noise by eliminating insignificant fluctuations that are unlikely to impact the agent's decision-making. Such small movements often reflect random market noise rather than meaningful trends, which could distract the model from learning important patterns. Second, aligning the open price of each remaining bar to the close price of the previous bar ensures continuity in the data. This step prevents artificial gaps and inconsistencies between bars, creating a smoother price trajectory for the agent to learn from. Together, these steps result in cleaner, more meaningful data for training.

While this preprocessing improves the clarity of the data for the agent, it does introduce some deviations from real-world trading conditions. In actual markets, price movements are continuous, and there can be minor fluctuations that, although small, may still influence decision-making. Additionally, real-world price data may contain gaps and inconsistencies due to various factors such as low liquidity or market events, which are smoothed over in this approach. These simplifications make the environment easier for the agent to learn from but may reduce its ability to handle more complex or erratic market behavior when deployed in a live trading scenario.

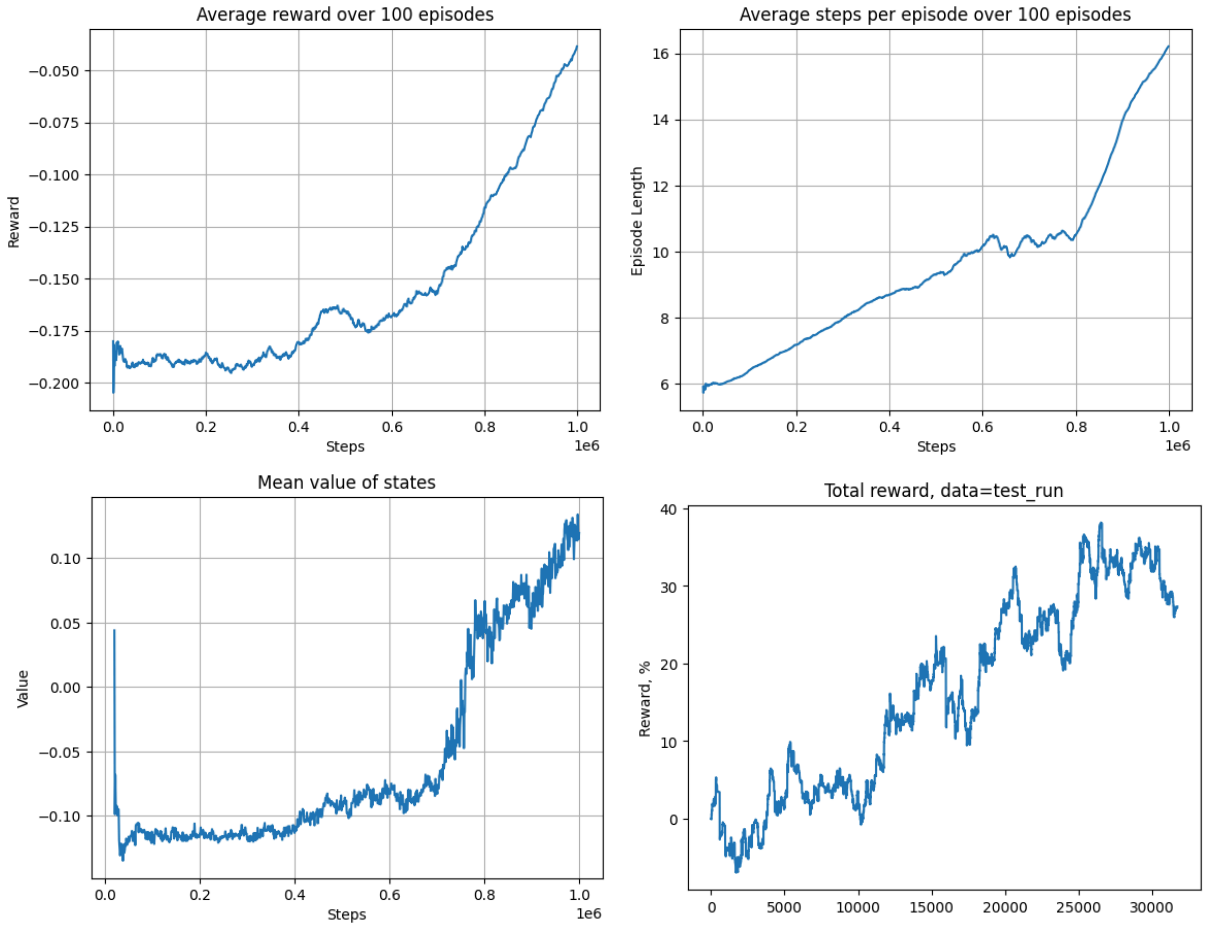
Experiments and Results

First, in Part 1 we implemented the basic environment with a basic reward function and started experimenting with the DQN parameters to see which gave the better result. We found a couple of important parameters needed adjustment:

- TOTAL STEPS - We found that the longer the training the better result we got.
- EPSILON STEPS - We found that with this large TOTAL STEPS letting the agent explore half the time and exploit the other worked best
- REWARD STEPS - We found that value of 1 made the agent choose immediate reward and a larger value made the agent made the learning too slow therefore we settled on REWARD STEPS = 2 as we got the best results with it and wanted the agent to give a little less importance to immediate reward

For the rest of the experiments we use fixed parameters (see 1 in appendix).
With the parameters settled we moved to inspect the reward function. We found that even though we got a positive mean reward the reward function wasn't good enough as we couldn't find a way to represent the reward as percent profit which encouraged us to move and implement part 2.

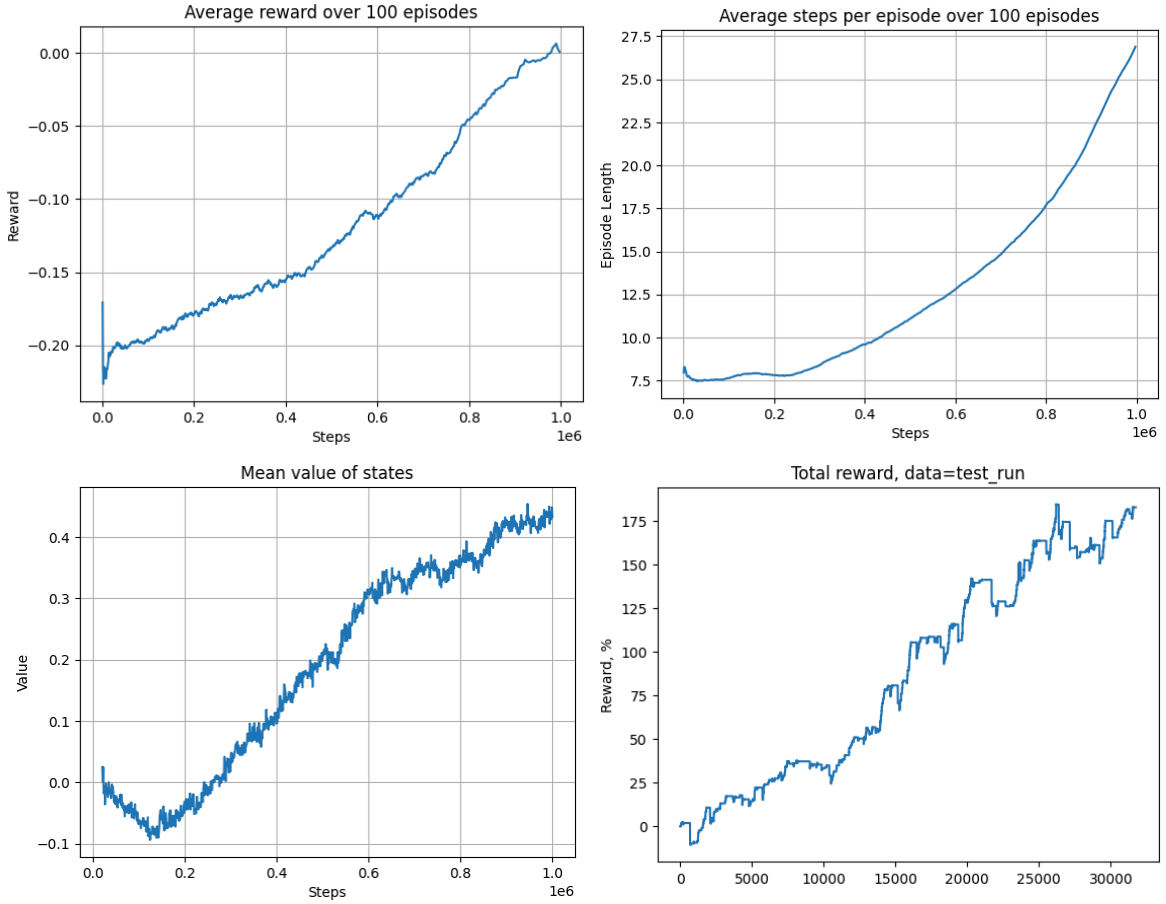
In part 2, we moved to implement an initial money pool in our environment. This helped us formulate a more reliable reward function by translating each loss or win into a percentage of the initial pool, which in order resulted in a better performance. In addition to that, we implemented a dynamic system where the agent can buy the maximum of stocks he can afford with his given money, making each action to be more risky but potentially more profitable.



Now, with the setup finished, we moved to implement the short-selling option. We added 2 new actions to the action space - Borrow_short and Return_short, we altered the observation space by adding a new holding position for every state and finally we added a new hyperparameter as the broker fee for short deals so we can simulate a simplified version of the interest of shorts in the real world by making the broker fee higher than the long deals commision.

Since we wanted to see the implications of this implementation, we conducted the same experiment with the same parameters as before and recorded the results with a fee the same as the long-deals commission.
In the results we observed a couple of differences:

- The average steps per episode reached higher length faster - This because the nature of short-selling inherently involves higher risk and uncertainty, the agent must carefully assess not only when to enter a short position but also when to exit. This cautious approach may result in longer trade durations as the agent weighs its options more thoroughly.
- The mean state value reached higher values faster - This is a direct result of our help in giving the agent more ways to profit in more situations and also the nature of shorts which results in higher profits for a transaction



We also noticed that the number of short transactions in contrast to longs were about 55%-45% in favor of short-selling. So, we wanted to see if increasing the broker-fee will make the agent choose less short actions. Therefore we also conducted couple of experiments with a higher fees to see the implications of this new hyper-parameter in our environment:
With fee = 0.5%, we saw a distribution of approximately 40-60 in favor of long-selling, indicating that the agent chose less shorts as they were more expensive.
With fee = 1.0%, we saw even lower reward where the agent made approximately only 2% short-deals

Total reward - 0.5%

Total reward - 1.0%

Discussion and Conclusions

In this project, we explored the effect of expanding our DQN agent's action space to include short-selling, in addition to the traditional buying and selling actions. This introduced complexity in the exploration process, as the agent had to balance between profit opportunities in both rising and falling markets.

Our main findings suggest that the addition of short-selling did indeed enhance the agent's performance in volatile market conditions. Specifically, the agent was able to make profit during market downturns, as evidenced by the increase in the total reward when running on the same data. The short-selling capability allowed the agent to profit in a broader range of market scenarios, leading to a more robust performance overall.

However, there were trade-offs. The increased complexity in exploration resulted in longer training times and a higher risk of suboptimal actions, especially early in the learning process. Additionally, when we introduced higher broker fees for short-selling, the agent adapted by reducing its short-selling frequency, showing a sensitivity to transaction costs.

Further research could focus on making the environment more similar to the real world by adding more complex concepts such as interest on holding a borrowed shares in short-selling.

Appendix

1:

```
BATCH_SIZE = 64
BARS_COUNT = 50
TARGET_NET_SYNC = 5_000
DEFAULT_STOCKS = "CLAI/data/YNDX_160101_161231.csv"
DEFAULT_VAL_STOCKS = "CLAI/data/YNDX_150101_151231.csv"

GAMMA = 0.98

REPLAY_SIZE = 200_000
REPLAY_INITIAL = 20_000

REWARD_STEPS = 2

LEARNING_RATE = 0.0002

STATES_TO_EVALUATE = 1_000
EVAL_EVERY_STEP = 1_000

EPSILON_START = 1.0
EPSILON_STOP = 0.1
EPSILON_STEPS = 1_000_000

TOTAL_STEPS = 1_000_000
CHECKPOINT_EVERY_STEP = 50_000
VALIDATION_EVERY_STEP = 25_000
```