Trading Strategy and Energy Maze

Tianyi Zhu
903880510

Abstract

This report presents the solution to the homework assignment involving two main tasks. The first task implements a trading strategy based on historical stock price data using Python. The second task solves an energy optimization problem in a maze, where the goal is to determine the minimum initial energy required to traverse the maze without the energy dropping below 1.

1 Introduction

The homework assignment consists of two distinct problems:

- **Task 1: Trading Strategy** - This task involves designing a trading algorithm based on historical stock price data, generating buy and sell signals, and analyzing the cumulative profit and loss (P&L) over time.
- **Task 2: Energy Maze** - This task requires solving an energy optimization problem where the objective is to find the minimum initial energy needed to traverse a maze without dropping below a critical energy level.

2 Task 1: Trading Strategy

2.1 Problem Description

The goal of Task 1 is to implement a trading strategy that analyzes stock price movements and generates trading signals. The strategy uses the following rules:

- Buy 10 shares if the price has increased for three consecutive days.
- Sell all shares if the price has decreased for two consecutive days.
- Sell any remaining shares on the last day.

2.2 Methodology

The strategy was implemented using Python with the Numpy and Pandas libraries. The input data is read from a CSV file containing the columns `Date` and `Close`. The algorithm iterates through the price data, applying the trading rules to generate signals, update positions, and calculate the account value.

2.3 Results and Analysis

The cumulative P&L over the entire period was calculated and visualized. Figure 1 shows the smoothed cumulative P&L plot, including markers for buy and sell signals.

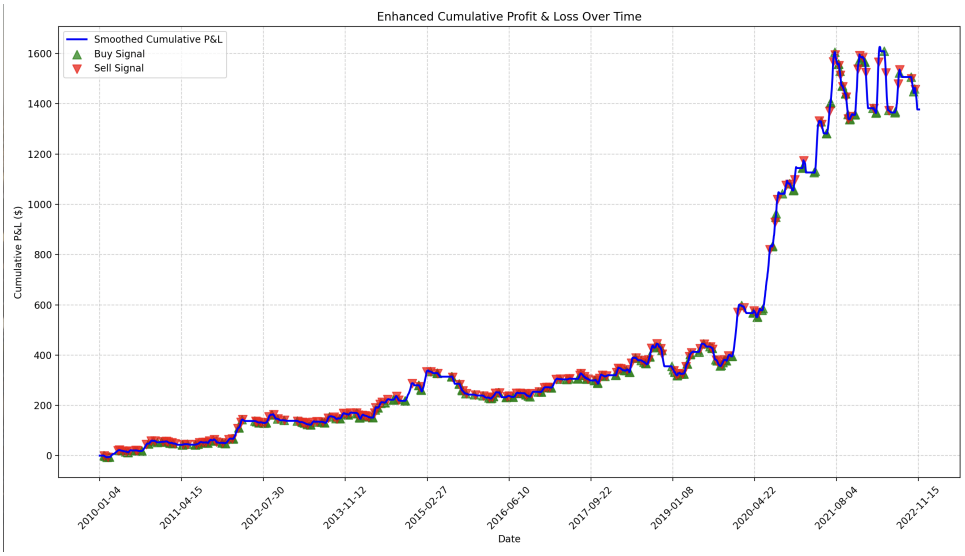

Figure 1: Enhanced Cumulative Profit & Loss Over Time

The final cumulative profit/loss was $1377.38. The results were saved to a CSV file named `trading_results.csv`.

2.4 Unit Test

Unit tests were conducted to validate the implementation. The tests checked:

- Correct generation of trading signals based on the example provided in the assignment.
- Accurate calculation of positions over time.
- Proper calculation of the final account value.

The tests passed successfully, indicating that the implementation meets the requirements.

3 Task 2: Energy Maze

3.1 Problem Description

Task 2 involves solving an optimization problem where the goal is to find the minimum initial energy required to traverse a maze from the top-left to the bottom-right corner without the energy level dropping below 1. The maze contains both positive and negative values, representing energy gains and losses, respectively.

3.2 Methodology

A dynamic programming approach was employed to compute the minimum initial energy required. The dp table stores the minimum energy needed at each cell to ensure the energy never drops below 1 during traversal.

The algorithm initializes the bottom-right corner and iteratively fills the dp table from bottom-right to top-left. The solution also considers edge cases with significant negative cumulative losses.

3.3 Results

The default maze and a custom user-defined maze were tested. The results are as follows:

- For the default maze:

$$\text{Minimum Initial Energy} = 7$$

- For a user-defined maze with negative values:

$$\text{Minimum Initial Energy} = 15$$

3.4 Analysis

The example maze provided in the assignment was tested, and the minimum initial energy required was determined to be 7. The solution was also tested with a generic maze input to ensure that the implementation is flexible and handles arbitrary inputs correctly.

3.5 Unit Test

Unit tests were performed to verify the correctness of the solution. The tests included:

- Validation using the example maze from the assignment.
- Testing with a generic maze input to ensure the solution handles arbitrary cases.

The tests passed successfully, confirming the correctness of the implementation.

4 Appendix: Terminal Result Screenshot

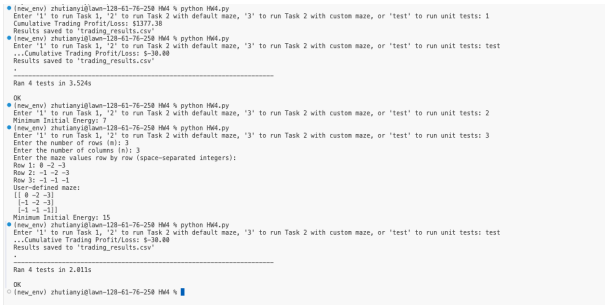

Figure 2: terminal Screenshot

The screenshot shown all the result run sucessfully in terminal, and all unit test got passed.

4.1 Usage

To execute the code, run the following command in the terminal:

```
python HW4.py
```

The user can select:

- Enter '1' to run Task 1 (Trading Strategy).
- Enter '2' to run Task 2 with the default maze.
- Enter '3' to run Task 2 with a custom maze input.
- Enter 'test' to run all unit tests.