Assignment: Using Simulated Annealing and Genetic Evolution to optimize a strategy for buying and selling stocks

Barry Denby

October 20, 2021

1 Introduction

NOTE: read the whole assignment brief first before implementing it contains very important information

In this assignment you will be tasked with doing four things:

- Create a simulation of a sample stock market based on 5 years of S&P 500 stock data with a portfolio that can take a maximum of 10 stocks at any time. You will simulate buying and selling of stocks with the aim of producing the largest return value at the end of 5 years of trading
- A mapping onto the simanneal library that will use Simulated Annealing algorithm to determine the best strategy
- A mapping onto the deap library that will use genetic evolution to determine the best strategy
- Using both mappings to determine the best strategy in a number of situations different stock metrics

In terms of effort the vast majority of your time will be spent on the first part in creating the simulation. The mappings to both libraries will not take long, and the experimentation to find the best strategy will take about 3 to 5 minutes per run using either annealing or evolution.

It is recommended that when you have completed your simulations that you test them with the test case given below. if you get the same results at the end then your simulation is matching the simulation that we are using. **Remember if you produce a bad simulation your results from numerical optimisation cannot be trusted.**

Read the details of the simulation carefully and if you have any questions do not hesitate to ask. You will need to have full clarity on the simulation in order for implementation to be correct and by consequence your numerical optimisation to be correct

2 Simulation Details

The reason why simulations of this nature is needed is that strategies for trading stocks can be trialled with virtual money without risk of losing money. With the stock market there are two things that we are trying to predict at all times.

1. When is the best time to purchase a stock
2. When is the best time to sell a stock

In the ideal case you should purchase a stock when it's value is lowest and sell it when its value is highest. The difference in share price between when the stock was purchased and when it was sold is the profit or loss that was made on that stock.

Thus a lot of people are highly interested in modelling the stock market to see if there is a clear cut strategy that can determine in advance when the right time to buy and sell will occur in order to maximise profit. The two main methods that are used are:

1. Techincal Analysis
2. Fundamental Analysis

Techincal analysis which is used here basically looks at the historical stock price and tries to use that alone to predict how the stock will change. Fundamental analysis requires that you look at the profile of a company (assets etc) and their balance sheets to see if they are worth investing in. We will not use the latter approach.

In this particular case we will have a portfolio of 10 slots. For each slot on day one you will start with an amount of $2,000. meaning you will have a maximum of $20,000 to invest in this version of the stock market. On day 1 (Day 128 in simulation time, will explain this later) you will purchase as many shares as possible in the 10 most desirable stocks. Every 30 days you will then decide if you should sell any of these stocks.

Should one or more stocks be sold you will then purchase alternative stocks using a similar mechanism to day 1 to ensure you still have 10 stocks purchased at all times. The process will then repeat where you hold onto those shares for another 30 days and determine again if you need to replace shares.

The reason for the 30 day limit is to ensure that when you apply Simulated Annealing and Genetic Algorithms to this problem it will complete all iterations/generations in 3 to 5 minutes. If this was reduced to every day the numerical optimisation could easily take several hours to complete.

Once you apply optimisation to this problem using either approach you should be aiming to get a final portfolio value between $120,000 and $170,000 after the 5 year period of the dataset has completed.

3 Classes

Discussion of the classes that you will need in order to write the simulation

3.1 StockFrame

A class to represent a stock and it's associated values on each given day in the dataset. This definition is provided for you in the base script and should not be modified.

- __init__: constructor for the class that takes in a number of properties that are set on the stock for this particular day
- self.closing: The value of a single share of this stock on this day.
- self.volume: How many shares of this stock were traded on this day.
- self.sma_ema: A collection of four Simple Moving Averages, for the last 8, 16, 32, and 64 days. also 4 Exponential Moving Averages, for the same time periods
- self.sma_ema_slope: not to be used but present in the data
- self.sma_ema_curvature: not to be used but present in the data
- self.profit_loss: The profit and loss of the stock compared to the value of the stock 1, 2, 4, 8, 16, 32, 64, and 128 ago.

3.2 Portfolio

A class to represent a portfolio of stocks. Will also be responsible for simulating the stock market that we have as well as purchasing and selling stocks.

- __init__: takes in six parameters in addition to self:
  - stocks: How many stocks are in the dataset
  - days: How many days worth of stocks are in the dataset
  - initial_funds: The starting funds for each slot in the portfolio
  - stock_frames: The list of stock frames containing all stock data
  - stock_names: The ticker names of all stocks e.g. AAPL for Apple
  - total_stocks: How many slots this portfolio supports

  This function constructs and initialises a portfolio. It should take a copy or reference of all six parameters provided and then construct lists to store the funds available for each slot (defaulted to initial_funds), a list to store how many shares of a stock was purchased, a list to store the share price of the stock when it was purchased, and finally a list of indexes to the stock itself. You should also initalise a list of 8 weights for a stock along with two thresholds one for purchasing a stock and one for selling a stock.
- haveStock(self, index): checks to see if the provided stock index exists in the list of stock indexes. If yes return true, false otherwise
- MACWeightsValues(self, weights, values): takes two lists exactly 8 elements long and multiplies each list element pairwise before adding it to a total that is returned.

- purchaseStocks(self, day, buy_indexes): takes in the current day of the simulation and a list of tuples containing a buy weight and a stock index. This list should be in reverse order from highest buy weight to lowest buy weight. The function should first remove any stocks from the list that have already been purchased after this for each empty slot the maximum amount of shares of the most desirable stock should be purchased (you may need to use math.floor() here as partial shares are not permitted). Once a stock has been purchased it should be removed from the list. Stocks should be purchased until either all slots or filled or buy_indexes is emptied. Note a stock should only be purchased if it has a closing value of 0.01 or higher.
- reset(self): This should reset the portfolio to as it was on day one with all initial funds present and no stocks purchased.
- sellStock(self, day, slot): sells the stock in the given slot for whatever value the relevant stock has on that day. it should add these funds to the total funds available for that slot.
- shouldBuy(self, day): goes through all stocks on a given day and determines if they would be desirable to purchase. It will go through all stocks for a given day (the offset (day * self.stocks) will be useful to you to select the right set of stock frames for a given day) and by using MACWeightsValues on the buy weights and the profilt and loss values of a stock it should calculate a buy weight. if this is above the required threshold a tuple containing the stock index and the buy weight should be added to a list. Once evaluation of all stocks is finished this list should be sorted in reverse order of buy weight (highest weight first). An attempt to purchase stocks should then be made
- shouldSell(self, day): goes through the list of stocks currently held and applies the buy weights to the profit and loss of each stock for the current day. if any of the stocks fall below the sell threshold it should be sold.
- simulate(self): simulates the stock market starting from day 128 to ensure we have enough data for profit_loss to calculate correctly. every 30 days it should decide if any stocks should be sold. If stocks are sold it should purchase new stocks in the sold stock's place.
- totalValue(self, day): determines the total value of the portfolio on the given day. for each slot take the funds remaining for that slot and add in the value of the shares for that stock on the given day.
- updateBuyThreshold(self, threshold): sets a new buy threshold on this portfolio will be used by both optimisation methods
- updateBuyWeights(self, weights): takes the list of 8 buy weights and sets them this portfolio will be used by both optimisation methods.
- updateSellThreshold(self, threshold): sets a new sell threshold on this portfolio. will be used by both optimisation methods

4 Suggestion for how you should proceed and other relevant information

The recommendation here would be to make sure the portfolio works first before attempting to do any optimisation. You've been provided with a dataset (processed-stock-data-frames.dat) and also the ticker names for all stocks (stock-names.txt) and also functions for reading in this data into the relevant stock frames (readPreprocessedData() and readStockNames()) respectively.

The data you have has 505 stocks in total and covers 1259 days worth of data (about 5 years total, weekends are not included as the stock market doesn't operate on weekends) each stock. Each group of 505 stock frames represents a single day of the stock market one frame per stock, thus for each day you wish to access you will need to offset by 505 stock frames e.g. if you want day 100 you will need to offset by 505 * 100 to get the correct starting point for that day.

For the purposes of testing you will need to get to the point of simulating the entire stock market first before you can verify it is working correctly. Simulation should start at day 128 and run until the end of the dataset. This is to ensure you have enough data available for the profit and loss metrics to work correctly.

The sample case provided here uses buy weights of 1.0 for all 8 weights in the list with a buy threshold of 1.0 and a sell threshold of 0.0. If you run this from day 128 you should get a starting portfolio value of 20,000 and an ending portfolio value of 73,010.12 on the last day of the dataset. If you put in print statements at the time a stock is purchased and before a stock is sold in purchaseStock() and sellStock() respectively you should get the following output.

```
Day 128 purchased 24 shares in TRIP at 80.95999908447266 per share
Day 128 purchased 133 shares in MU at 14.970000267028809 per share
Day 128 purchased 54 shares in FB at 37.01900100708008 per share
Day 128 purchased 64 shares in BBY at 30.860000610351562 per share
Day 128 purchased 66 shares in SEE at 30.270000457763672 per share
Day 128 purchased 104 shares in GT at 19.209999084472656 per share
Day 128 purchased 108 shares in UAA at 18.517499923706055 per share
Day 128 purchased 62 shares in TSN at 31.829999923706055 per share
Day 128 purchased 128 shares in HBI at 15.602499961853027 per share
Day 128 purchased 73 shares in INCY at 27.329999923706055 per share
Day 158 selling 66 shares in SEE at 27.149999618530273 per share
Day 188 selling 62 shares in TSN at 27.989999771118164 per share
Day 188 purchased 31 shares in ALGN at 57.09000015258789 per share
Day 188 purchased 3 shares in CMG at 539.6300048828125 per share
Day 248 selling 24 shares in TRIP at 76.37000274658203 per share
Day 248 selling 64 shares in BBY at 22.780000686645508 per share
Day 248 selling 31 shares in ALGN at 55.22999954223633 per share
Day 248 purchased 12 shares in ILMN at 151.58999633789062 per share
Day 248 purchased 43 shares in AAL at 34.04999923706055 per share
Day 248 purchased 11 shares in ALXN at 156.2100067138672 per share
Day 308 selling 3 shares in CMG at 500.1499938964844 per share
Day 368 selling 73 shares in INCY at 47.13999938964844 per share
Day 368 purchased 15 shares in UHS at 108.06999969482422 per share
Day 368 purchased 78 shares in NFX at 44.0 per share
Day 398 selling 43 shares in AAL at 37.959999084472656 per share
Day 398 selling 11 shares in ALXN at 161.3699951171875 per share
Day 398 selling 104 shares in GT at 24.829999923706055 per share
Day 398 selling 78 shares in NFX at 39.470001220703125 per share
Day 398 purchased 18 shares in AVGO at 88.40499877929688 per share
Day 428 purchased 54 shares in MNST at 32.786598205566406 per share
Day 428 purchased 74 shares in LUV at 34.59000015258789 per share
Day 428 purchased 28 shares in VRTX at 108.16000366210938 per share
Day 488 selling 133 shares in MU at 28.989999771118164 per share
Day 578 selling 74 shares in LUV at 36.810001373291016 per share
Day 578 purchased 43 shares in NFLX at 89.5071029663086 per share
Day 578 purchased 25 shares in SWKS at 107.6500015258789 per share
Day 638 selling 12 shares in ILMN at 199.9499969482422 per share
Day 638 selling 54 shares in FB at 86.05999755859375 per share
Day 638 selling 18 shares in AVGO at 111.44999694824219 per share
Day 638 selling 54 shares in MNST at 46.30329895019531 per share
Day 638 selling 25 shares in SWKS at 79.06999969482422 per share
Day 638 selling 108 shares in UAA at 45.01499938964844 per share
Day 638 selling 128 shares in HBI at 29.059999465942383 per share
Day 638 selling 28 shares in VRTX at 125.47000122070312 per share
Day 668 selling 15 shares in UHS at 130.32000732421875 per share
Day 668 purchased 77 shares in ATVI at 31.969999313354492 per share
Day 698 purchased 152 shares in NVDA at 30.399999618530273 per share
Day 698 purchased 3 shares in AMZN at 647.8099975585938 per share
Day 728 purchased 846 shares in AMD at 2.9800000190734863 per share
Day 758 selling 77 shares in ATVI at 28.1200008392334 per share
Day 758 selling 43 shares in NFLX at 87.4000015258789 per share
Day 758 selling 152 shares in NVDA at 25.729999542236328 per share
Day 758 selling 3 shares in AMZN at 507.0799865722656 per share
Day 758 selling 846 shares in AMD at 1.8300000429153442 per share
Day 758 purchased 84 shares in NEM at 25.780000686645508 per share
Day 758 purchased 62 shares in TSN at 60.16999816894531 per share
Day 758 purchased 65 shares in O at 60.47999954223633 per share
Day 788 purchased 17 shares in WYNN at 94.13999938964844 per share
Day 788 purchased 542 shares in AMD at 2.859999895095825 per share
Day 788 purchased 200 shares in FCX at 10.140000343322754 per share
Day 788 purchased 137 shares in NVDA at 35.38999938964844 per share
Day 788 purchased 56 shares in INFO at 34.95000076293945 per share
Day 818 purchased 92 shares in OKE at 40.08000183105469 per share
Day 818 purchased 19 shares in MLM at 187.85000610351562 per share
Day 848 selling 56 shares in INFO at 32.29999923706055 per share
Day 908 selling 65 shares in O at 64.47000122070312 per share
Day 908 selling 200 shares in FCX at 9.9399995803833 per share
Day 908 selling 19 shares in MLM at 173.9499969482422 per share
Day 908 purchased 571 shares in CHK at 7.349999904632568 per share
Day 908 purchased 114 shares in MU at 17.479999542236328 per share
Day 908 purchased 61 shares in WMB at 29.760000228881836 per share
Day 908 purchased 60 shares in WDC at 54.939998626708984 per share
Day 938 selling 84 shares in NEM at 35.380001068115234 per share
Day 938 selling 62 shares in TSN at 69.73999786376953 per share
Day 938 selling 571 shares in CHK at 5.789999961853027 per share
Day 938 selling 17 shares in WYNN at 94.18000030517578 per share
Day 938 purchased 42 shares in AKAM at 69.91000366210938 per share
Day 938 purchased 34 shares in NFLX at 126.56999969482422 per share
Day 938 purchased 24 shares in FFIV at 134.66000366210938 per share
Day 998 selling 24 shares in FFIV at 133.72000122070312 per share
Day 998 purchased 67 shares in CSX at 48.599998474121094 per share
Day 998 purchased 12 shares in URI at 127.05999755859375 per share
Day 1028 selling 61 shares in WMB at 28.920000076293945 per share
Day 1028 selling 92 shares in OKE at 54.33000183105469 per share
Day 1028 purchased 11 shares in INCY at 149.24000549316406 per share
Day 1028 purchased 281 shares in NRG at 17.860000610351562 per share
Day 1088 selling 42 shares in AKAM at 48.45000076293945 per share
Day 1088 selling 12 shares in URI at 106.31999969482422 per share
Day 1088 selling 11 shares in INCY at 122.2699966430664 per share
Day 1088 purchased 13 shares in ALGN at 148.6199951171875 per share
Day 1088 purchased 5 shares in COO at 241.6199951171875 per share
Day 1088 purchased 13 shares in ADSK at 111.44999694824219 per share
Day 1178 selling 60 shares in WDC at 85.76000213623047 per share
Day 1208 selling 542 shares in AMD at 11.380000114440918 per share
Day 1208 purchased 109 shares in NTAP at 56.40999984741211 per share
```

A good idea here would be to test the purchasing of 10 stocks on day 128 before adding in selling behaviour. Once you have verified that your simulation works correctly then you can proceed to apply simulated annealing and genetic algorithms to it.

5 Notes

You are required to submit this assignment by 2021-12-19 (Sunday 19th of December) by 23:55. You are required to submit two separate components to the Moodle

- A single python file containing all of your code.
- A PDF containing documentation of your experiments along with graphs **If you do not provide documentation your code will not be marked.**

There are also a few penalties you should be aware of

- Code that fails to compile will incur a 30% penalty before grading. At this stage you have zero excuse to produce non compiling code. I should be able to open your project and be able to compile and run without having to fix syntax errors.
- The use of libraries outside the python SDK and permitted libraries (simanneal, deap, matplotlib) will incur a 20% penalty before grading. You have all you need in the standard SDK. I shouldn't have to figure out how to install and use an external library to get your app to work

- The standard late penalties will also apply

You should be aware that I will remove marks for the presence of bugs anywhere in the code and this will incur a deduction of between 1% and 15% depending on the severity. If you have enough of these bugs it is entirely possible that you may not score very many marks overall. I want robust bug free code for this that closely matches the graphs above

Also note that the percentage listed after the bracket is the maximum mark you can obtain if you complete that many brackets without error.

6 Tasks

1. Write the simulation using the detail above and test it works by correlating your simulation with the output above in Section 4 (50%)
2. Map the simulation so it can run using simulated annealing to compute the ideal strategy (65%)
3. Map the simulation so it can run using genetic algorithms to compute the ideal strategy (80%)
4. Generate strategies using both simulated annealing and genetic algorithms for the following situations. You must detail what the best strategy is used (buy weights and thresholds) for all, write a small discussion of the strategy (time taken, any interesting effects, total discussion should be no more than 1,000 words). Using matplotlib provide a graph of the value of the portfolio after each day (100%)
   - Strategies produced by three seperate runs of simulated annealing
   - Strategy produced by three seperate runs of genetic algorithms