A Discrete-Time-Markov-Chain Based Trading Algorithm

Gabriel Kagan

Abstract

Approximately 15 trillion dollars[1] is invested into the stock market, with every investor attempting to maximize profit. The economist John Maynard Keynes once said, "the market will be subject to waves of optimistic and pessimistic sentiment, which are unreasoning, and yet in a sense legitimate where no solid basis exists for a reasonable calculation".[2] I will attempt to refute Keynes assertion by constructing a portfolio that algorithmically buys and sells NYSE (New York Stock Exchange) securities with a goal of maximizing profit. I will construct a portfolio of various stocks and apply a MDP (Markov Decision Process), which will develop a heuristic for transacting securities.

Objective

The goal for the project was to develop a trading algorithm that executes daily trading decisions to maximize profits while managing a portfolio of stocks. We sought to achieve this by analyzing historical time-series data of stocks and extracting various features/technical indicators. I then map the values of the indicators to different states based on the probabilistic distributions of the indicators. This allows the generation of a state transition probability matrix that provides transition information between states and then develop a heuristic-based trading strategy that makes transaction decisions using the transition probabilities.

To validate the effectiveness of the algorithm I will simulate the trading strategy on a validation dataset to measure performance in terms of profit.

Assumptions

History does not repeat itself, but it does rhyme. Based on the historical data, I sought to develop an algorithm that would execute trades using information extracted from past performance of the stock in the market. It is an extremely daunting task to predict even near-exact price of a stock in the future. The approach used in this trading system was to ask not the price of the stock in the future or whether the price in the future will increase or decrease but it is to quantify how likely is it that the trader will see a 'desirable' state in the near future and then make a trading decision based on this knowledge. The states in the Discrete Time Markov Chain were constructed using the differences in daily closing price and daily short term moving average. Let's call this parameter δ. The closing price is the price of a stock at 4:30PM, when the market closes. The daily moving average is the average price the stock was sold at on that day, using a moving period of 50 days. If the closing price is greater than that day's moving average, this implies that the stock performed well on that day, has a high buy rating and therefore it is desirable to buy. If the contrary were true, then the stock price had not performed well on the day. Examining the historical data, the δ values computed for every day in the time-series were mapped to states based on the probability distribution of this parameter (Using percentiles of the distribution). Logically, the states that fall at the left-tail of the distribution will be highly undesirable and the scale of desirability progressively increases toward the right-tail. The Chapman-Kolmogrov equation allowed for the computation of steady state probabilities which informed the probability of being in any state in the long run. Using the future transition probabilities, I developed rules for trading. If the probability of desirable states was high, this would imply a buy order. If the probability of the least desirable states was high, this would imply a sell order. And if

the probability of desirable and non-desirable states were similar, this would imply a hold decision.

Modeling

Markov Property

The Markov Property[3] is defined as a condition on a sequence of random variables X_1,X_2,... thought to represent the state of a system over time (n representing time).
A sequence X_1,X_2,X_3,... is to be a Markov Chain if

$$P(X_{n+1} = j | X_n = i = \alpha_1, \cdots, X_{k_m} = \alpha_m)$$

$$(\textit{where the indices } k_1, \cdots, \textit{ are all strictly less than } n)$$

is a number entirely determined by n,i, and j. This number is called the transition probability from i to j at time n. If the number is also independent of n, then we say the Markov chain is homogeneous and the transition probability is denoted p(i,j).In the case of stock prediction, financial derivatives, such as options, are essential in initiating the calculation of transition probabilities. Let us think of the states as being labeled from 1 through N. Then, the transition probability requires NxN Numbers $p(i, j)$.
Given any i, we have

$$\sum_j p(i, j) = 1$$

The total probability formula says that

$$P(X_{n+1} = j) \sum_k P(X_{n+1} = j | X_n = k) P(X_n = k)$$
$$= \sum_k p(k, j) P(X_n = k)$$

This has a linear algebra interpretation: if we introduce a transition probability matrix with entries $p = (p(i,j))_{ij}$, then to go from the distribution of X_n to the distribution of X_{n+1}, all we need to do is multiply the "vector" of probabilities by the adjoint matrix p. This observation leads to what is known as the Chapman-Kolmogorov equation, which allows us to compute probabilities of the state several time steps into the future.

The Chapman-Kolmogorov Equation

The Chapman-Kolmogorov[4] equations provide a method for computing these n-step transition probabilities:

$$p_{ij}^{(n)} = \sum_{k=0}^{M} p_{ik}^{(m)} p_{kj}^{(n-m)}, \qquad \text{for all } i = 0, 1, ..., M,$$

$$j = 0, 1, ..., M,$$

$$\text{and any } m = 1, 2, ..., n-1,$$

$$n = m+1, m+2, ...$$

These equations point out that in going from state i to state j in n steps, the process will be in some state k after exactly m (less than n) states. Thus,$p_{ik}^{(m)} p_{kj}^{(n-m)}$ is just the conditional probability that, given a starting point of state i, the process goes to state k after m steps and then to state j in $n-m$ steps. Therefore, summing these conditional probabilities over all possible k must yield $p_{ij}^{(n)}$.

Strategy

1. Develop a trading algorithm that executes daily trading decisions to maximize profits while managing a portfolio of stocks.

2. Analyze historical time-series data of stocks and extract features/technical indicators; map the values of the indicators to different states based on the probabilistic distributions of the indicators.

3. Generate state transition probability matrices for every indicators states and develop a heuristic-based trading strategy that makes transaction decisions using the transition probabilities.

4. Simulate the trading strategy on a validation data-set to measure performance in terms of profit.

Algorithm

For any given data entry, the algorithm first computes the δ value for the day, maps the value to it's corresponding state and then accesses the transition probability matrix for this parameter's states. This is the primary piece of information required for the trader to make a decision.

Based on the state's transition probabilities, the algorithm goes over a set of predetermined rules to choose an action. A detailed description of the rules is included below:

1. If the difference between the highest transition probability among the desirable states and the highest transition probability among the undesirable states is greater than some ϵ, then a buy decision is supported.

2. If the sum of the probabilities of the desirable states is greater than the sum of the probabilities of the undesirable states, then buy decision is supported.

3. If the highest probability among the undesirable states is greater than the highest probability among the desirable states, a state of 'caution' is activated wherein a sell action will most likely be chosen based on the combination of other rules satisfied too. If the contrary were true, a state of 'confidence' is activated which supports a buy decision similarly.

4. If Moving Average Convergence Divergence(MACD) is greater than the corresponding value on the 'signal graph', a buy decision is supported.

5. If the difference between the sum of the probabilities of the desirable states and the sum of the probabilities of the undesirable states is greater than some γ, a buy decision is supported.

Finally, based on the combination of rules satisfied, an action to either 'buy', 'sell' or 'hold' will be chosen. For more details, please refer the code section.

The algorithm has two hyperparameters. *'Lookahead'* which determines the number of days between successive state transitions to be considered while computing the transition matrix. Particularly, for a *'lookahead'* value of 4, the algorithm will compute the transition matrix for the states that the current state will end up in, 4 days ahead. Next, there is *'max buys'* which is the maximum position for the trader any point in time.

Simulation

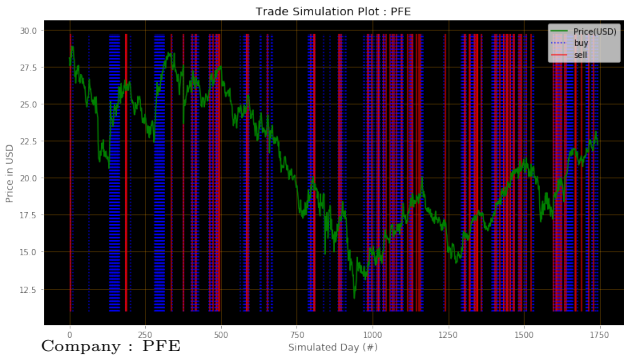
The data was split into three components. The first 70 percent of the data, the 'source data' was used to compute the transition matrix. The next 15 percent of the data was used to tune the algorithm's hyperparameters. The remaining portion of the data, 'the validation

data', was used to validate the performance of the trader(analogous to real-time trading).

At the start of the simulation, the trader was provided 2.5 million Dollars to start trading. This number was chosen to ensure that the trader is not restrained in it's actions throughout the simulation run. For instance, 'GOOGL' stocks are highly priced and hence if the trader were given just 100,000 Dollars, it would not be able to maintain a high position for most of the run. The transaction volume was fixed at 1000 units.

Results

Training-time Results



Company : PFE

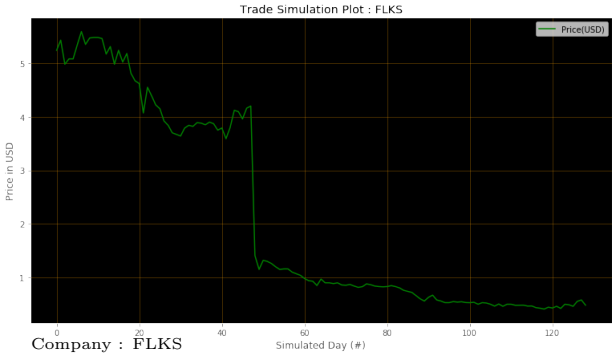
Account Balance : 2627577.539949598 USD

Holdings : 0

Next Price : 22.360001

Net Present Value : 2627577.539949598

Net Profits : 144646.24396639995



Company : FLKS

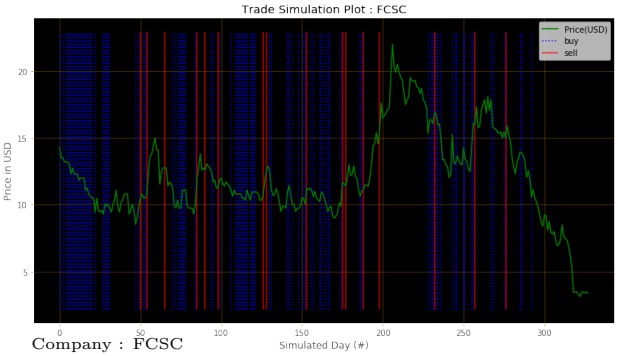
Account Balance : 2500000.0 USD

Holdings : 0

Next Price : 0.451

Net Present Value : 2500000.0

Net Profits : 0



Company : FCSC

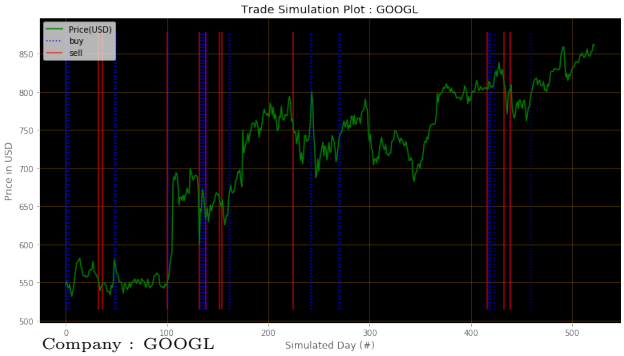
Account Balance : 2577759.520004804 USD

Holdings : 0

Next Price : 3.25

Net Present Value : 2577759.520004804

Net Profits : 80303.51999999997



Company : GOOGL

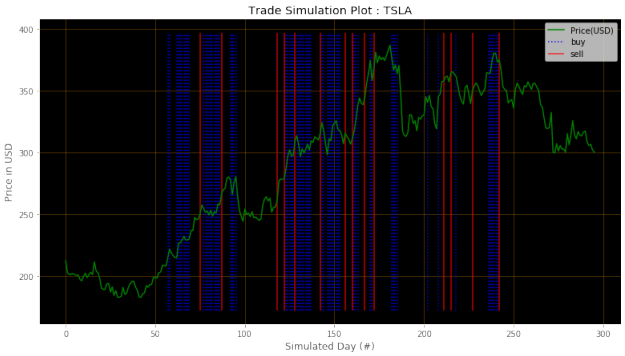
Account Balance : 3462309.2510588006 USD

Holdings : 0

Next Price : 863.75

Net Present Value : 3462309.2510588006

Net Profits : 1003114.5792843999



Company : TSLA

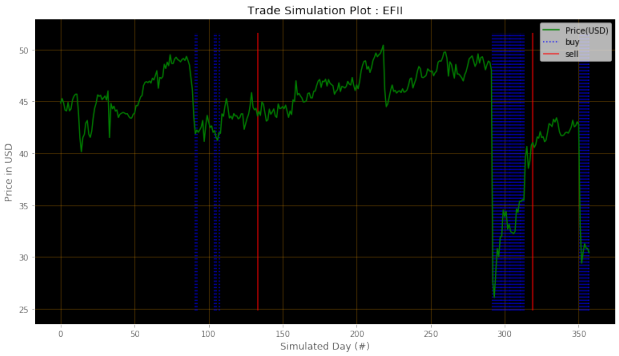
Account Balance : 3048015.388994 USD

Holdings : 0

Next Price : 312.0

Net Present Value : 3048015.388994

Net Profits : 595544.4130899998



Company : EFII

Account Balance : 2683101.1449603974 USD

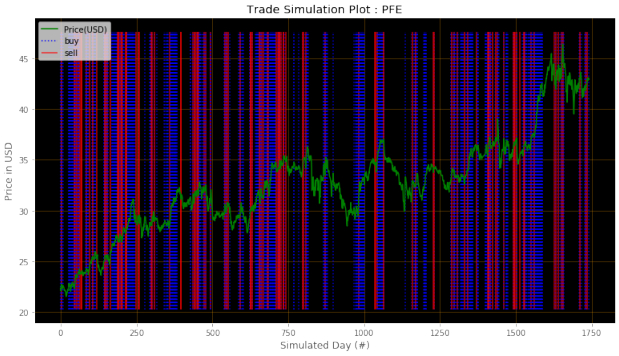
Holdings : 0

Next Price : 32.27

Net Present Value : 2683101.1449603974

Net Profits : 185866.71297480003

Validation-time Results



Company : PFE

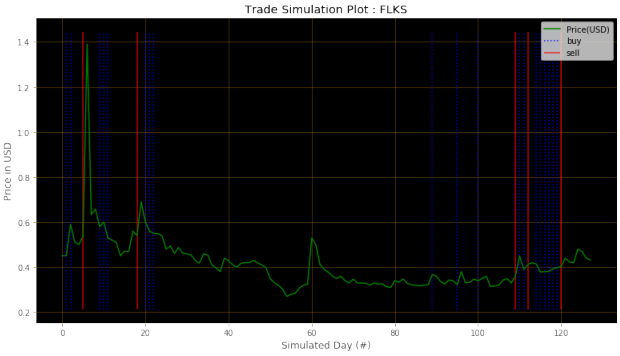
Account Balance : 2773254.8850420006 USD

Holdings : 0

Next Price : 42.790001000000004

Net Present Value : 2773254.8850420006

Net Profits : 310307.11708039965



Company : FLKS

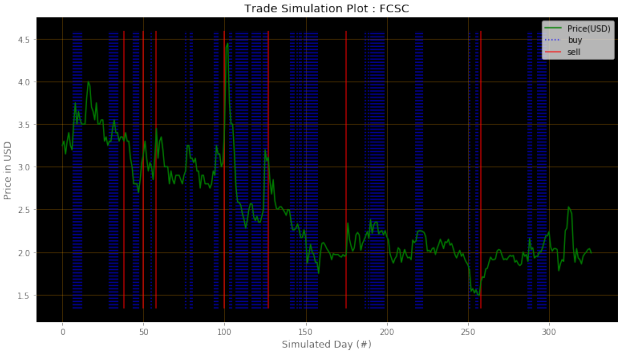
Account Balance : 2502428.3264 USD

Holdings : 0

Next Price : 0.47

Net Present Value : 2502428.3264

Net Profits : 2449.0575999999996



Company : FCSC

Account Balance : 2512396.7559999996 USD

Holdings : 0

Next Price : 1.91

Net Present Value : 2512396.7559999996

Net Profits : 12914.483999999988

Company : GOOGL

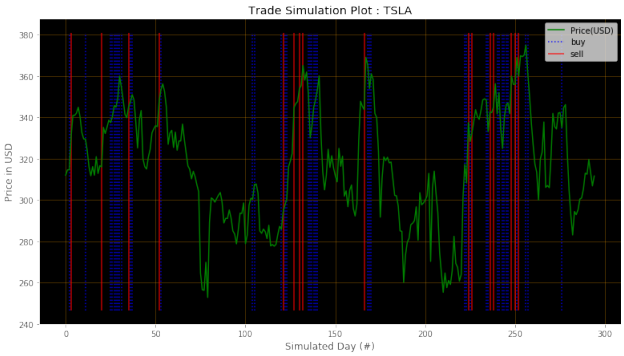
Account Balance : 3202749.9182183994 USD

Holdings : 0

Next Price : 1208.900024

Net Present Value : 3202749.9182183994

Net Profits : 732190.2620648004



Company : TSLA

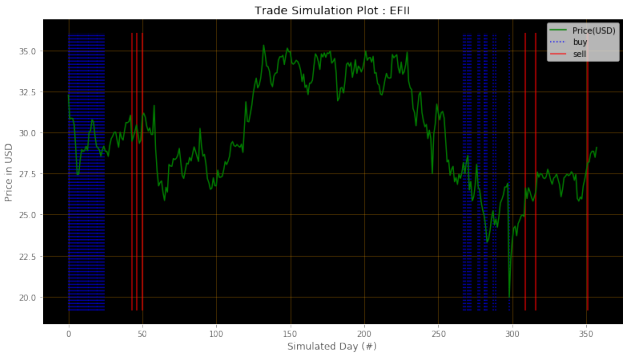
Account Balance : 2873948.79406 USD

Holdings : 0

Next Price : 316.200012

Net Present Value : 2873948.79406

Net Profits : 407450.4420360001



Company : EFII

Account Balance : 2535831.7169963983 USD

Holdings : 0

Next Price : 29.5

Net Present Value : 2535831.7169963983

Net Profits : 38383.877008400006

Conclusion

Based on the results, one can see that the algorithm smartly executes trades and definitively yields profits. For example, if one were to look at the 'GOOGL' portfolio, the training set yields a profit of roughly $1 million and the simulation run for validation net an astounding $732,000 dollar profit. For the stocks chosen, the algorithm visibly makes the right decisions based on the data that it is given. Even if one looks at the visuals with the most basic level of scrutiny, they can see that on average, the algorithm recommends one sell before a major price drop or buy before a price spike. 'GOOGL' is not necessarily the best example for this, because as a blue chip stock, it is pretty stable. Regardless, this buy-sell efficiency is reflected across volatile stocks as well. Penny stocks, which typically fluctuate wildly and are by no means stable, were also bought and sold by the algorithm. 'FLKS', which trades for $0.45 a share, is shown to have a massive change in stock value through its timeline. Despite this, the algorithm made a $2450 profit when trading this in the validation dataset. Although investing $2.5 million into penny stocks for a mere $2450 profit is neither impressive nor even adequate by any means, its important to note that the algorithm did not lose money on the penny stock holding, which would be both easy to do and catastrophic given a stock this volatile. An interesting aside regarding volatility is that given the stock

data for 'TSLA', a high priced 'stable' stock, our algorithm still made a $400,000 profit on 'TSLA' in the validation run, where the stock rather unusually wildly fluctuated in value. Despite evident safeguards against volatility, the model cannot predict and protect holdings in event of a crash. First of all, most of the data gathered has come from a period of relative economic prosperity, so it is unlikely that there is data that reflects a change from incredibly high stock value to very low to support the algorithm. Secondly, a crash is just that. In the Markov Chain model, a crash would be represented as a change in state from very high to very low. So in terms of the distribution of our parameter, *Closing Price - Short Term Moving Average*, the crash would be analogous to a state change from the right tail of the distribution to the absolute left. As a spontaneous, one-time event, a crash is difficult to account for, and would not necessarily be predicted by the model, which reflects more long term patterns, unless it was signalled by some distinctive pattern.

Effectiveness and Limitations

The model works on the premise that trends in stock price changes repeat over time, and during these instances of repetition, the trader needs to make intelligent buy/sell decisions to maximize their profit. The optimistic results of our simulation confirm that trends in stock price changes do repeat in various instances and profitable trading strategies can be devised by analyzing past data.

The algorithm requires rigorous back testing before implementing in real time. The ideal back test will have sample data from a relevant time period of a duration that reflects a variety of market conditions. In this way, one can better judge whether the results of the back test represent a fluke or sound trading. The tuning procedure must also be continuously repeated when new data is available.

References

(1) HowStuffWorks, If all the money in the U.S. only totals $6 trillion how can the New York Stock Exchange have stocks valued at $15 trillion?

(2) Keynes, J. M. *The general theory of employment, interest, and money*; Springer, 2018.

(3) Durrett, R. *Probability: theory and examples*; Cambridge university press, 2019; Vol. 49; p 274.

(4) Durrett, R. *Probability: theory and examples*; Cambridge university press, 2019; Vol. 49; p 284.