Group Members:
Taejin Hwang
Eric Lin
Steve Shi
Peter Veerman

EE126 Financial Modeling Project Report

I. Abstract

Financial modelling is a complex process whereby the historical data of a multitude of given features of a financial market is converted into a forecast for the market's future performance. There are many implementations of this process in use today that are based on Machine Learning techniques, most notably forms of linear or non-linear regression. However, the use of Hidden Markov Models have been shown to produce excellent results in time-series prediction as well. Our project focuses on the performance and viability of using Hidden Markov Models for stock market prediction applications.

II. Introduction

Hidden Markov Models, or HMMs, are statistical tools that can be used to infer the most likely sequence of *states* of a "hidden" generative process given *observations* that characterize the states of the process. The description of the states being hidden simply means that a user cannot observe which state the process is in at a given time. HMMs rely on the assumption that the hidden *states* are governed by a Markov process, and that the *observations* have a probabilistic dependence on the underlying states. HMMs can effectively model many real-world stochastic processes, and therefore see usage in a wide variety of fields such as natural language processing, computational biology, and artificial intelligence.

Stock market analysis and predictions is another field in which HMMs can be a powerful tool. A given stock has many observable characteristics (symbols), such as their daily opening prices, closing prices, highs, lows, etc. The value of these symbols are driven by dozens of unpredictable and uncontrollable factors, including but not limited to global politics, economic conditions, and even public perception of a company's policies and ethics. This system lends itself well to being modeled as an HMM. Let us define the following variables:

- N = the number of states in the state space S
- M = the number of possible symbols per state
- L = the number of observations
- O = the observation sequence, O_i = observation i, $1 < i < L$
- q_i = the current state at time t, $1 < i < N$
- v_k = A possible symbol for a state, $1 < k < M$
- A = the transition probability matrix, $A_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]$
- B = the symbol emission probability matrix, $B_j(k) = P[O_i = v_k \mid q_t = S_j]$
    - Distribution assumed Gaussian

- π = the initial state probability, $\pi_i = P[q_1 = S_i]$

The HMM can be represented with the parameters **$\lambda = (A, B, \pi)$.**

III. Training the HMM

HMMs should be trained on previous data in order to make the model as accurate as possible. There are three problems that need to be solved in the process of training an HMM, each one addressed by a different algorithm.

1. The first problem to be tackled is: Given the observation sequence, what are the most likely HMM parameters λ? This problem is solved using the Baum-Welch algorithm.
2. The next problem is: Given the observations and the HMM parameters, what is the probability of observing the given sequence of emissions? This problem is tackled using the Forward-Backward algorithm.
3. Finally, Given all of that information, what is the most likely sequence of hidden states? The last problem is solved using the Viterbi algorithm.

In our project, we leave the training and calculations in the hands of the open source python library *hmmlearn*, which uses all three of these algorithms in its implementation. We train on the earliest 95% of a given market data and use the last 5% for testing our HMM implementation.

IV. Implementation

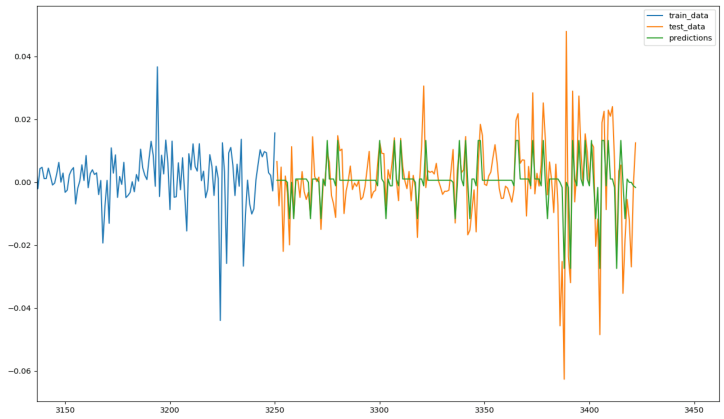
We decided to have the Hidden Markov Chain States describe movement in the day to day price of a given stock. We played around with the number of states in order to find the number that gave us the most optimal predictions. Based on all of these states, some days we will see large increases while other days we may see slight drops. But the point is that we assume there is an associated probability or a form of randomness from moving from one state to another. In order to forecast the observations at the next time step t+1, assume we are at hidden state S_t at time t. Since we assume that the emission probability distribution of each symbol of every state is Gaussian, we can estimate that the emission from S_t will be about the mean of the Gaussian distribution of each symbol. Luckily, we can easily get these means using *hmmlearn*'s means_ method. This is our prediction for time t+1, and we then augment the training data with the actual data for time t+1 and then retrain the HMM. This allows us to repeat the forecasting process and predict the t+2, t+3, etc. day's data using all of the information gained up until the day before the next prediction.

Another challenge we faced was getting the data to fit our model. We wanted to make our states represent linear buckets but after noticing that the majority of our data was concentrated within rises and drops of at most 10%, we decided to find the absolute difference of 99% of the data after sorting. In addition to this, we wanted to keep the data stationary and less susceptible to outliers so we decided to take the log, but we ran into many errors in which we would take the log of 0 since we were mostly observing changes that were either positive or negative. To combat this, we applied a linear transform that ensured all of the data was positive and then from there we were able to customize the number of states much more smoothly. However, we realized in the end through observations that containing 99% of the data was often ineffective so we ended up removing it from our code.

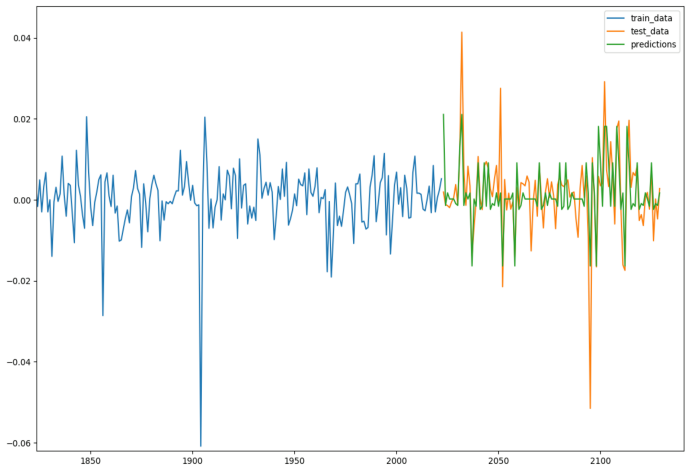
The observations that we are considering in our model that come directly from the stock are the daily closing prices, opening prices, highs, lows, and volume. In addition, we have to multiply the stock price by the cumulative product of the stock splits. This is because the splits drastically change the price of the stock along with the number of shares of that stock, but we don't consider the number of shares in our dataset. However, in our first HMM models we only considered the opening price in order to test the implementation in our code. In addition to the individual stocks, we planned on running the HMM prediction method on the average values of the symbols across each of the 14 US stock market sectors, and applying that additional information to our individual stock predictions. Our original idea was to organize every stock into several related sectors (similar to the actual market sectors), weight the effect of each sector trend by the stock's estimated involvement in each sector, and apply the weighted information into our stock predictions. After realizing each stock was already categorized into a specific sector, we decided to just settle with the simpler model after struggling with the implementation of the first model. We hoped that the addition of the general trend of a stock's respective market sector will help our predictions be more accurate. However, we ran into problems along the way implementing the sector idea, mainly trying to tackle the problem of each stock's time series data having different lengths and starting points. In the end, we did not add this into our project.

As mentioned in the previous section, in order to test the effectiveness of our HMM stock market predictor, we pretend the last 5% of the stock market data hasn't been recorded yet and train on the first 95%. Then using the method described in the first paragraph of this section, we predict the stock market data for the last 5%. Finally, we examine the effectiveness of our prediction by calculating the difference between our predicted data and the actual data using the Mean Absolute Error (MAE) formula provided by the library sklearn.metrics in their mean_absolute_error method. We performed this calculation on various stocks, using the stock's actual observed data and the predicted data in the last 5% of the market history. We then compare the results with the Support Vector Regression (SVR) and MAE results found in the referenced research paper.[1]

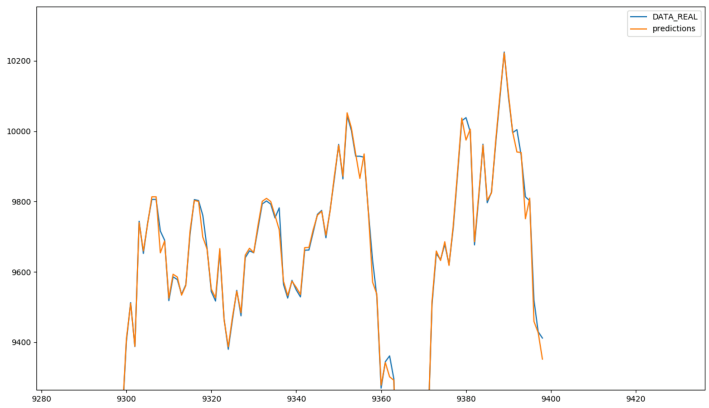
IV. Results



(Figure 1, Estimated vs. Actual Open Price Delta for GOOGL)



(Figure 2, Estimated vs. Actual Open Price Delta for VRSK)



(Figure 3, Estimated vs. Actual Open Price for APPL)

Stock	HMM	HMM (Ref.)	SVR (Ref.)
GOOGL	0.0063	0.0057	0.0071
APPL	0.0069	0.0053	0.0052
VRSK	0.0051	N/A	N/A

(Table 1, MAE Results for Open Price Estimation)

Our results show promising indications that our HMM model is able to provide a fairly accurate prediction of short term stock performance in upcoming days, given the history of the stock market.

V. Conclusion

Our group faced many challenges during the course of the project. Before we started the project, none of our group members had very much experience working with HMMs or any experience with stock markets. Therefore, we had to learn and research through the power of Google. Thankfully, our searching led us to several useful tools and research papers, although many of these we needed to do additional research just to understand their usefulness. One example would be learning how to use *hmmlearn*, the python library we used for our HMM prediction. There were many nuances between the *hmmlearn* implementations we learned from in our research sources and our own implementation, since the sources we learned from weren't using *hmmlearn* in the same way as us.[2] Another challenge that we faced was that our original ideas for our financial prediction project were too ambitious to be implemented with our limited knowledge on the topics required. The majority of our time was spent just trying to learn how to create and then actually implement a working HMM test example on Google stock's closing prices. We ended up having to cut a majority of our planned features in our HMM stock prediction project.

Over the course of this project, we learned a lot about what HMMs do, the probability and math behind their design, and their effectiveness in many real-world applications. We learned the importance of setting realistic goals and timelines for our projects, but at the same time we learned how to improvise and adapt to the resources available to us to set achievable ones. Finally, we learned how complex and uncertain well used stock market prediction methods and algorithms can be, even with the large amount of real-world use and research already conducted in the field.

Our project has left lots of room to be built and improved upon in the future. Incorporating a wider variety of stocks and different sectors can help determine whether or not our results and conclusions drawn from them are indeed valid. Attempts to predict further than one day into the future without retraining the HMM model can be experimented with, but based on our research seems unlikely to produce favorable results. Also, further investigation into the effect of different parameters, including but not limited to the number of HMM states and the inclusion of our idea of applying the weighted sector trends to an individual stock's prediction.

References and Sources:

https://github.com/ayushjain1594/Stock-Forecasting/blob/master/Final_Report.pdf [1]

https://www.quantstart.com/articles/market-regime-detection-using-hidden-markov-models-in-qstrader?fbclid=IwAR3W7P8dI-tRQGKqk3P2K4NGqHAgrcHn1XN7kLTq6ZdKhmS2kkZNVPQjg9U [2]

https://editorialexpress.com/cgi-bin/conference/download.cgi?db_name=SILC2016&paper_id=38

https://www.cs.sfu.ca/~anoop/students/rzhang/rzhang_msc_thesis.pdf

http://www.blackarbs.com/blog/introduction-hidden-markov-models-python-networkx-sklearn/2/9/2017

https://hmmlearn.readthedocs.io/en/latest/tutorial.html

http://www.cs.cmu.edu/~bdhingra/papers/stock_hmm.pdf

https://www.youtube.com/watch?v=0u80UvCAKVo

https://stackoverflow.com/questions/45258306/predicting-future-emissions-from-fitted-hmm-model