Formatting instructions for NIPS 2017

Forest Kobayashi
Department of Mathematics
Harvey Mudd College
Claremont, CA 91711
`fkobayash@hmc.edu`

Jacky Lee
Department of Mathematics
Department of Computer Science
Harvey Mudd College
Claremont, CA 91711
`jaclee@hmc.edu`

Abstract

Stock data is very difficult to analyze using classical methods, in large part due to heavy involvement of humans in stock pricing. In this paper, we examine a new approach to analyzing time-series stock data, particularly in the context of grouping correlated stocks.

1 Introduction

Here is a rough table of contents for our paper, so that the reader has a rough idea of where we'll be going:

1. Motivation (challenges to modelling stock data)
2. Long-term goal for the model / ideal model architecture
3. Classification
4. Findings
5. Conclusion

2 Background

There are many reasons why predicting stock prices is difficult, but for our model today, we will focus on just two:

1. The behavior of a given stock is influenced by hundreds of hidden variables — e.g., the current state of geopolitics, governmental fiscal policies (for instance, the US Treasury interest rate), the performance of stocks in various other markets, etc. Hence, the price of a stock at any given moment is an emergent phenomenon, influenced by many small movements of markets around the world. Further, these connections themselves are in a constant state of flux — as the market evolves, technology improves, and policies are rewritten, the weight each of these variables has in influencing pricing will wax and wane.
2. With the exception of some forms of algo trading, most trading strategies are being created and implemented by humans. That is, often humans are the ones who perform analysis of market information, and make decisions based off the findings. But humans are not naturally predisposed to rigorous quantitative analysis, and hence markets do not always behave rationally. As an example, consider `Bitcoin`.[1] Thus, an accurate financial model will need to have some method of predicting human psychology.

[1] Citation: Gary Evans

2nd Course in Mathematics of Big Data (MATH189 2018), Claremont, CA, USA.

Using modern machine learning techniques, it is possible to control for some effects of (2) by using previous stock data. However, less work has been done to model the effects of (1). So, for today's model, we will focus on methods of preprocessing time series stock data so as to control for some of the effects of (1).

3 The Model

3.1 Goals

Our ultimate goal is to be able to use past stock data to train our model. Then given some stock trend data for some stock, use our model to predict the upcoming trends and return a vector of decisions. An example of the output is shown below.

$$\mathbf{y} = \Big\langle P_{buy}(\mathbf{x}), P_{sell}(\mathbf{x}), P_{short}(\mathbf{x}), P_{nothing}(\mathbf{x}) \Big\rangle$$

Here, $P_{buy}(\mathbf{x})$ represents how confident the model is in recommending the user to buy the stock. Similarly, $P_{sell}(\mathbf{x})$ represents selling the stock, $P_{short}(\mathbf{x})$ represents shorting the stock, and $P_{nothing}(\mathbf{x})$ represents doing nothing. All these probabilities should sum to 1.

In order to reach this sort of conclusion, we want our model to compensate for the effects of (1) in a clever way. In traditional stock trading practices, there are a few guidelines that are used for similar determinations:

1. Someimtes industries follow trends together. For instance, if we see a price drop in one oil stock, we might expect to see drops in other oil stocks.
2. When interest rates go up, yield-bearing financial assests increase in value, while the value of equity stocks might decrease.
3. If an earnings report reveals that a stock overestimated its expected profits, the value of the stock will decrease.
4. In times of high uncertainty, money might be pulled out of equity assets and moved to yield-bearing financial assets. Hence, stock prices might go down.
5. Bad PR can result in a temporary dip in stock pricing.

These are external factors that cannot be determined purely from the stock data itself. Often, most of this information comes from news outlets, press releases, and other media-sharing forums. So, we need our model to perform the following:

1. Scrape real-time text data from news websites, financial websites, and (possibly) academic journals in the case that we're looking at something like a technology stock.
2. Perform some sort of Natural Language Processing and Sentiment Analysis to determine how this information might affect the market if / when it becomes widely publicized.
3. Somehow combine this information with time-series stock data to make a prediction of how the market might behave in the short and long term.
4. Make a trading decision based on this information.

We propose the following prototype model.

3.2 Prototype Architecture

Before we get into it, we want to stress that this is a heavily compartmentalized model we built to try and capture how we want information to flow and be processed. In reality, we'd want to collapse some of the nodes into a single process, and remove some of the overall complexity to make training easier. We are not saying this would be easy to implement and / or train without significant computing power, just that this model could accomplish what we desire.

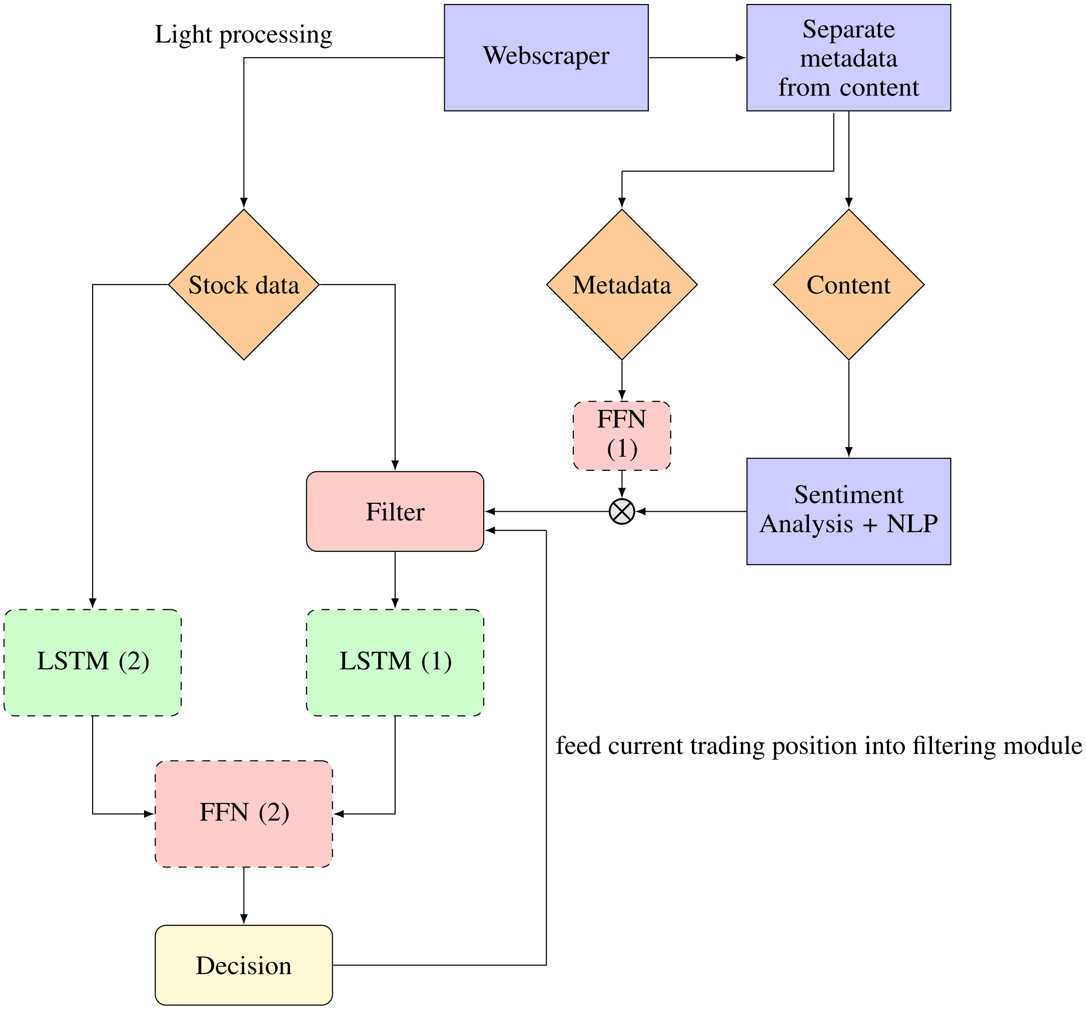

Figure 1: Model overview

Of course, as we just said, this is not the optimal architecture. But let's walk through the rationale before discounting it entirely:

1. We scrape two types of data: time series stock data, as well as news / financial / academic articles.
2. To preprocess the stock data, we extract the time series and convert it into an array, abstracting the data in some manner so that we can make all inputs the same size. We embed the stock symbol itself with a word vector, so that the network can learn to differentiate behaviors of different stocks. This will also be employed in the subsequent filtering stage.
3. To preprocess the text data, we first separate out the metadata from the content, and clean things like html tags out of the body (depending on how we're doing the scraping). We perform Natural Language Processing methods on the input data (e.g., sentiment analysis), to try and quantify whether the article represents good news or bad news for some stock.
4. We feed the metadata into a feedforward network that is trained to determine the bias of the source from which each datum was obtained. The $\tanh()$ function will be used, because we want to be able to "flip" the input from some organization if it consistently predicts the opposite of what actually happens.
5. At the $\otimes$ junction, we do elementwise multiplication of the output from FFN (1) with the output from our sentiment analysis of our articles. These will then be sorted by the stocks they apply to, and we will take the harmonic mean of these values to obtain a single predictive value for the behavior of the stock in the near future.

6. Next, we feed this into a filter, which will choose a few families of stocks for LSTM (1) to focus on. The filter might include some neural architecture itself, and will incorporate information about the network's current assests.

7. We take the most significant stocks output by the filter, along with some sentiment data about it, and feed this heavily into LSTM (1), which will be a relatively deep network.

8. Simultaneously, we feed the unprocessed stock data (without selecting a noteworthy subset) into LSTM (2), which will be a much shallower network, the idea being that LSTM (2) might pick up on noteworthy trends that the news articles missed.

9. Finally, this is input into a small feedforward network that aggregates the processed and unprocessed inputs, and returns decisions to buy, sell, short, or do nothing with respect to some particular stock. The process is then repeated, and ideall the network would turn a profit.

For today's paper, we focused on the filtering stage. In particular, we wanted to identify ways of classifying families of noteworthy stocks that LSTM (1) could make predictions on, and identify abstract relationships between pricing of various stocks. Our rationale was this: most of the other components of the diagram are famous problems that others have already performed lost of work on. That is, using LSTMs to predict stock pricing is, for the most part, a solved problem. Implementations and hyperparameters may differ, of course, but underneath, the structure of the model is largely the same. We saw the filtering process as the component we could perhaps make advancments on.

4 Methods

4.1 Data Scraping

We gathered intraday data for ten randomly-selected S&P 500 stocks using the Alpha Vantage API (`https://github.com/RomelTorres/alpha_vantage`). A portion of the data for one stock is shown below.

```
                     1. open  2. high   3. low   4. close  5. volume
date
2018-05-08 09:30:00  1058.54  1058.54  1055.00  1055.1600    26407.0
2018-05-08 09:31:00  1054.99  1058.16  1054.99  1056.9248     6384.0
2018-05-08 09:32:00  1057.41  1058.95  1057.41  1058.5700     7160.0
2018-05-08 09:33:00  1058.30  1058.64  1056.93  1058.6400     6448.0
2018-05-08 09:34:00  1058.80  1060.00  1058.50  1059.9600     5548.0
```

For each stock, we computed the arithmetic mean of the high and low as a heuristic for summarizing the price of the stock during that minute. For our minimal working prototype, we decided to ignore the data for the volume of trading, the open price, and the close price, since stock prices vary quite a bit and we felt that the high and low prices would be sufficient in giving us an estimate of how the stock was preforming at that moment in time. However, when composing a more polished model in the future, we will be sure to include this information.

4.2 Polynomial Fitting on Windowed Data

We then analyzed the data by running a sliding window of 20 minutes across the prices, with each window having a large overlap with the previous (to make the data smoother). Each consecutive window was only shifted by 2 minutes. We then fit a cubic polynomial to each window, extracting the constant, linear, and quadratic coefficients for further analysis.

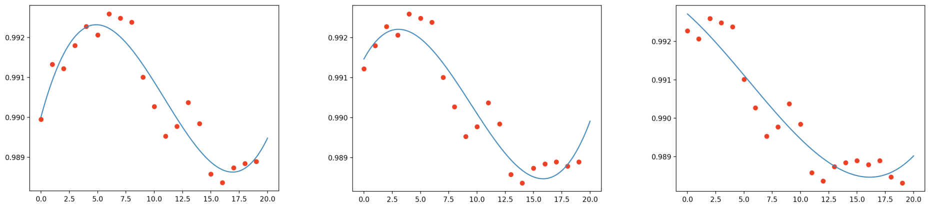

Figure 2: Sliding windows of data fitted with polynomials

4.3 Visualizing the Coefficients

After performing this sliding window analysis on each of the stocks, we then plotted the coefficients of our polynomial fits in $\mathbb{R}^3$ to visualize our data. We were hoping that we'd gain insight by examining the structure of these trajectories through time. Some examples are shown below.

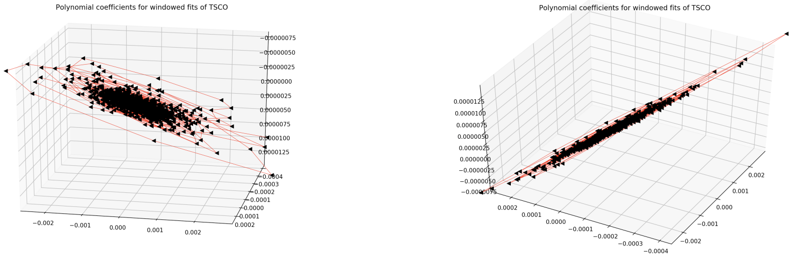

Figure 3: Visualization of the lower order coefficients in $\mathbb{R}^3$

The distribution of the coefficients in three dimensional space seems to be a multivariate Gaussian distribution. Furthermore, it seems to be very flat, appearing to be a line if viewed from the correct angle.

We experimented with different window sizes and different amounts of shifting between consecutive windows. We found that in general, varying the window sizes did not affect the shape of our graphs much but the amounts of shifting affected the smoothness and trajectories of our graphs. If the amount of shifting was large, such as if it were half the size of a window, then the trajectory would be very rough and edgy. If this value were to increase even more such that there was no overlap between windows, the trajectory would be random, as shown below.

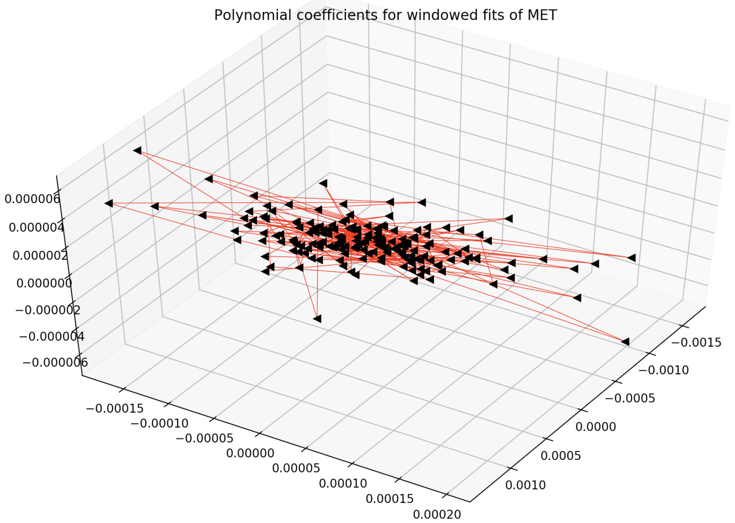

Figure 4: Visualization of the lower order coefficients in $\mathbb{R}^3$ without overlapping windows

Although it is hard to see on the diagrams above, we noticed that the trajectory of coefficients seemed to be rotating about the center of the distribution. Thus, we wanted to examine the following questions about this representation of the data:

1. Why is the distribution so flat? Are the coefficients maybe not as independent from one another as we believed?
2. Why might we be seeing this periodic behavior? And can we use Fourier analysis to extract meaningful information about it?

For the first, we have a few theories: first, for a degree N polynomial, the coefficients are really only controlled by $N - 1$ variables: the roots. Hence, our solutions might be constrained to some hyperplane. Second, it could be that regularization is constraining the coefficients to the plane, preventing any one term from getting too large. So, satisfied with this answer for the time being, we moved on to fourier analysis.

4.4 Frequency Analysis