Stock Prediction Based on Past History and Twitter Sentiment Cross Analysis

Joseph, A Jude
UCSC Undergraduate
Bioinformatics B.S Program
Data Collection, Programming, and Analyzation

Mangalore, Aman
UCSC Undergraduate
Computer Science B.S Program
Data Collection and Analysis

Problem

The efficient market hypothesis is an investment theory that states that it is “impossible” to “beat” the stock market because market efficiency causes existing stock prices to always incorporate and reflect all relevant information. In other words, there is no way for an investor to purchase an undervalued stock or sell stocks for inflated prices on their own. The market makes it so that the only way for an investor to gain a higher return is to bet on a riskier investment. So the question becomes can we decrease the amount of risk that accompanies the investors’ aim to accumulate a higher profit?

Methods

Approach

How an investor calculates his or her choice to invest is due to numerous economic factors but may be influenced by emotions as well. In recent years, the Human Emotion theory as it applies to the stock market has been a popular topic amongst psychologists. Andrew Lo, a finance professor at Massachusetts Institute of Technology, and Dmitry Repin, a neuroscientist at Boston University, published a research article featured in *The Journal of Cognitive Neuroscience* which suggests that emotions may influence many if not all Wall Street trader’s decisions.

Based on this information, it might not be impossible to produce a low error predictive model of a stock. First, a machine learning model will be built to analyze previous history on particular stock prices. Then, using the Twitter API, Tweepy, tweets surrounding a particular company or business can be gathered. Sentiment analysis can be performed on that data and classified as having an overall positive or negative polarity. Synthesizing both results may allow us to predict the trajectory - a increase or decrease by some value - of that company's stock.

Rationale

The machine learning method that is implemented to predict stock prices based on previous history is a Long Short Term Memory Recurrent Neural Network(RNN). This type of deep learning approach will allow for better retention of memory through a large series of past dates and stocks compared to regular regression. The RNN starts with one hidden layer with four 'memory'(LSTM) cells that optimize relations to previous results. The task is then simplified in a sense to a time series prediction problem.

The sentiment analysis will be done through a public library called *Textblob* that uses similar techniques for learning involving : part-of-speech tagging, classification, language translation and detection, etc. *Textblob* mainly utilizes the Natural Language ToolKit(NLTK) , a leading platform for building programs with human language data, and pattern.en, a module for a part-of-speech tagger, for accurate sentiment analysis. This will produce a better representation of tweets attained since *Textblob* gathers large data from numerous different avenues than if built from scratch.

Plan

Hypothesis

The main questions are :

- What is the relationship between public opinion and stocks involving a particular business?
- How accurately can a RNN model predict the price of a stock on a given day with only one hidden layer?
- In what other ways can we use tweet analysis to potentially improve prediction in the future?

Experimental Design

To approach this problem, we first had to figure out how to get the general public's opinion on the company. We found that Twitter is a place that people talk about their opinions on a subject, and so we looked for every tweet within a timeframe that contained 'Google'. We used a Textblob API which uses NLTK to figure out if a tweet is positive or negative based on a threshold, and combined the overall sentiment for each day, repeated for five days. We then compared the graph of trends of public opinion vs. the trends of the stock price to see how they compare.

Implementation Outcome

Framework(s)

Many core frameworks were utilized for this experiment some of which have been already introduced such as *Textblob* and *Tweepy*. Another vital framework utilized is *Keras*, a deep learning library that uses Theano and TensorFlow. Those are two of the top numerical platforms used for Deep Learning research and development, but they can sometimes be difficult to use for creation of models. The *Keras* python library provides a convenient way to create a variety of deep learning models on top of Tensorflow and Theanos. It contains a high-level neural networks API that is easily extendible to build models and works with Python.

Another significant framework necessary for building matrices and datasets was *Pandas* , an open source library providing high-performance data structures and analysis tools. *Pandas* was used to simplistically read and structure stock data from a csv file into a dataset which could be separated into training and testing segments.

Finally, the *scikit-learn* framework was utilized for data mining and analysis. S-learn features classification, regression, and regression algorithms that can include SVMs, gradient boosting, k-means, and more and it is designed to be used with numerical values from numpy. We ended up using S-learn primarily for normalizing data, manipulating matrices, and error analysis.

Previous Work

Our primary motivation to do this project came from a stock prediction challenge shown here: https://github.com/llSourcell/predicting_stock_prices. This source gave us the idea to predict on something that isn't supposed to be predictable (the stock market) based on sources that should influence the market on a day-to-day basis. This also provided us with a skeleton for our Twitter querying model.

We also used a keras tutorial in order to get our general layout. This tutorial: http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ showed us how we can use an RNN for time series prediction.

Challenges

Some challenges we ran into involved using multiple hidden layers. When we tried to implement more than one, we seemed to run into ridiculously high error rates. We tried to use a base implementation for these hidden layers experimenting activation by relu, sigmoid, and tanh function similar to assignment four in the class, to see which gives us the best performance.

One of our major challenges involved trying to get tweets from years ago with *Tweepy*. We originally had our model to use tweets from many years ago, but we learned that the Twitter API only gives us access to tweets from up to seven days ago. Even then, we had a lot of trouble retrieving this information. We programmed a system where we would acquire tweets from the past week, between days, and then splitting them into two separate csv files - one column with dates and one with the public opinion.

Similarly, we ran into some issues with extracting csv file data when we wanted to use it for training. *Keras* is sensitive to formatting for its models and neural network APIs so we had to use *panda* to help us separate the file properly before transforming it. Finally, we split the original data set into quarters: three of which were used for training the model and the last for testing.

Knowledge Gained

Our project expresses various methods that we learned in class, but we were able to get a lot more practice applying these methods to a real scenario. We gained more knowledge about tools (similar to *TensorFlow*) we could use to build the models. Studying *Keras* and *Tweepy* taught us that many public libraries we use are very high level and abstract, meaning we as the programmers have the ability to create fairly complex models without having to create everything from scratch. This makes it easier for us to repeatedly train and test new models and see what works most effectively for us.

Experiments

Results

Our results from our RNN model return Fig. 1.0 with 7.43 root mean squared error deviation between predicted and real stock prices for our training set and a 8.10 root mean squared error deviation for our test set. As you can see in the graph, the training and test predictions are slightly shifted. This was done purposefully so that we can compare the model produced to the true data.

Our Twitter sentiment analysis returned distinct tweets with varying polarity which we then compiled together to achieve an average score across the timestamps on each. The resulting trend can be seen in Fig. 1.2. The overall sentiment analysis of all the tweets acquired in the past few days were positive but deviated in variance of negative tweets. This can be visualized in the drop at position 3 and sudden turn at position 4.

Compared side by side with the last corresponding dates to our first model we see a bit of similarity with the curvature of each graph. Though this is no accurate indication of a relationship between public opinion and stock this indicates that we may be able to gain more significant results with further testing.

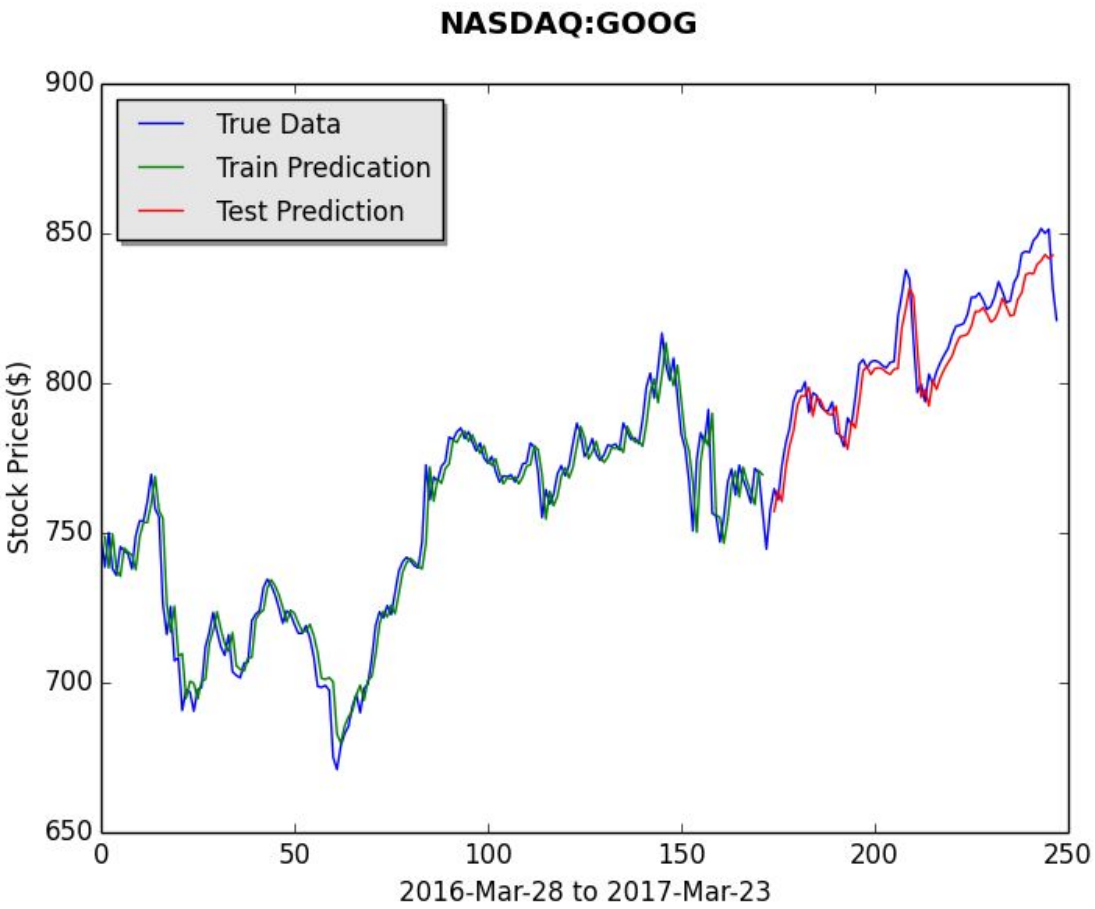

Figure 1.0

dates were converted into floating point numbers for graphing purposes

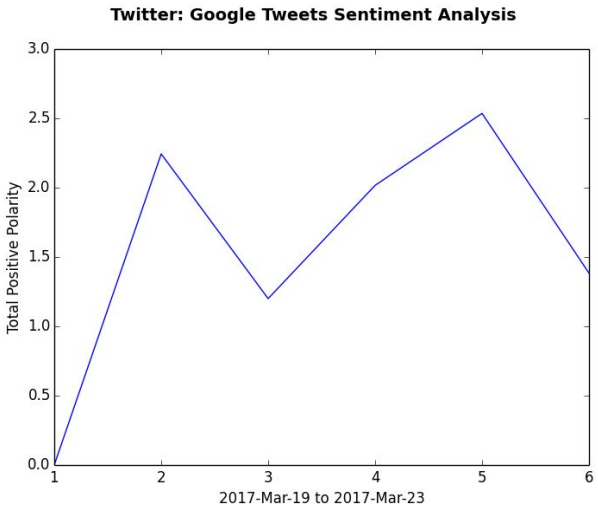

Figure 1.2

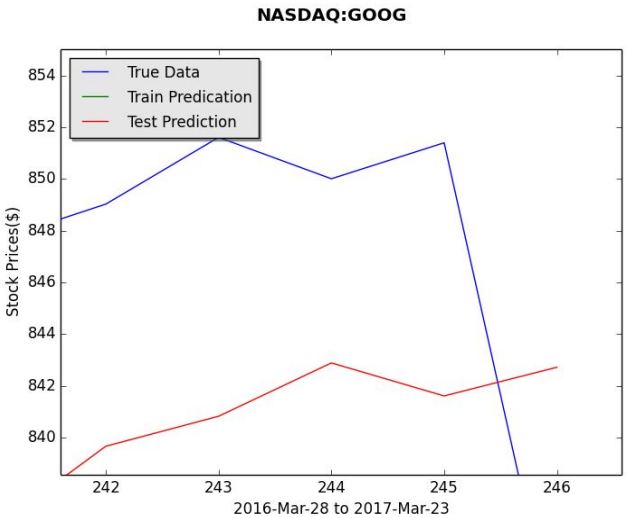

Figure 1.3

Conclusion and Evaluation:

Although we knew we weren't going to get perfectly accurate predictions, we wanted to tackle this project in order to visualize the correlation between stock prices and public opinion of the company - or at least show some rough relationship existed. Our ideal goal was to calculate know how much influence the public opinion has, and how much it could potentially be weighted as a feature for price prediction. One of the main difficulties with acquiring a better solution was the lack of public opinion data obtained through the Twitter API which limited our potential.

As a result in our project, we assumed our sentiment for each tweet is completely accurate, which is not true because there may be a lots of skewed data - random tweet that have nothing to do with Google as an actual business - yet our model will assign a value to these tweets regardless.

Moving forward, the first step to continuing this project is to get a significant range of tweets, rather than only within the week, in addition to more accurate sentiments from them. A potential solution could be to search through each word in the tweet itself, and see how relevant it is to the company's finances. Tweets that are more relevant will have a higher influence to prediction while tweets with low relevance will affect the model less.