Investigating Correlation and Predicting Stock Movements with LSTM Neural Networks

Brian Barry.....bfbarry@ucsd.edu
Patrick Garrett.....pgarrett@ucsd.edu
Kelvin Murillo .....k1murill@ucsd.edu
Cogs 118B Final Project
University of California, San Diego

Introduction and Motivation

The stock market and its price movements are one of the most captivating sources of data for those attempting to capture the kinds of patterns and trends present in the chaotic and dynamic world of economics. Because a stock's (also referred to as security) value can change throughout time, it can be expressed as time series data with different time scales and is a perfect case for methods such as Long Short Term Memory (LSTM) neural networks. Because businesses at a macroeconomic scale often interact, there will be some companies whose price movements –along with other profitability measures– are related, thus making correlation analysis a possible method for grouping stocks in coherent ways. We will investigate stocks with high correlation, as well as apply an LSTM neural network to predict certain securities' prices over a period of 100 days. While looking at intraday market information is possible, we decided to only look at data on the interday scale.

We chose to tackle this particular problem because we are personally interested in the subject of quantitative security analysis. In the bigger picture of machine learning and economics, this problem is important as understanding what methods are most appropriate for a task such as a price prediction reveals intrinsic properties about the market's chaotic behavior that weaker technical indicators fail to do.

Related work

PCA, as covered in class, is a method for dimensionality reduction that transforms the data's dimensions based on their correlation measure. Because we originally intended to cluster some stocks, and there are often many dimensions to deal with, PCA would be an appropriate first step to organize the data. PCA can be applied in several interesting ways for security and portfolio analysis. It seems that the general case for PCA on price information in this context indicates that using "two principal

components is enough to explain most of the total variation", where the first component can be interpreted as a "market" component accounting for general variation in that dimension, and the second one representing other underlying structures of the security. (Pasini et. al.).

Neural networks, as discussed in class, are an effective machine learning framework for complex pattern recognition. There are a variety of different neural networks appropriate for different tasks based on the data at hand, and because we are using data that changes over time we chose a Recurrent Neural Network (RNN) known as LSTM. RNN's are neural networks that loop over a certain amount of time, which allows them to incorporate information from the past to infer what is happening in the present. However, the longer the time scale becomes, RNN's are not as effective as they, in a sense, "forget" certain long term dependencies (Bengio et. al.).

LSTM counteracts this pitfall by in effect learning these long term dependencies. They are passed into a "cell" that remains in the artificial neuron and decides whether information needs to be incorporated or not in each successive iteration of the neural network by passing the information through an activation gate. A study applying LSTM to predict security prices by Nelson et. al. showed that compared to other methods, LSTM was among the safest. Along with prices, they used a number of technical indicators –which are other quantitative measures of a security's assets such as the simple moving average – as input to their algorithm. Their study showed an average 55.9% accuracy rate on a security's price movement.

Methods

Data Collection:

Data was retrieved from a Kaggle dataset containing stock information from 1970-2018. For individual price data, we used the Yahoo Finance library.

PCA and Correlation matrix:

To perform PCA analysis on the trends of the stock market, we chose to analyze the closing prices for an abundance of stocks. We wished to see which stocks the most correlated and how well the resulting PCA components could explain price movements. First, we imported the stock market data as a large CSV file. The stocks were represented by their market ID "tickers". We utilized the individual stocks tickers to cross-reference which industry the company was in. For this project, we chose to only analyze tech company stocks, but we designed it to handle any industry chosen. Once the dataset was purged of all other industries, we used pythons data frame structure to clean and analyze the data further. To accurately perform PCA it is important to have stocks that represent a continuous time series. Because the original dataset went all the way back to the 1970s and many of the stocks are no longer around today, we chose to only analyze stocks after January 1st, 2010. The remaining stocks

were further cleaned so that only tech stocks that have complete data from January 1st, 2010 to present-day remain. Finally, the closing prices of the remaining stocks were converted into percent increase/decrease, so that we could compare their relative values instead of absolute price points. To do this we used the function:
Price increase % = $(P_{cur} - P_{prev})/P_{prev}$
For PCA, the price increase percent values were normalized by dividing by the highest price percent increase.

Prediction

We then normalized the time series data from the Yahoo Finance library, using several methods. Calling yf.download automatically loads the time series into a data frame, and because we only care about High, Low, and Closing price of the data we remove the 3,5 and 6 columns. We imported time series from 4 years of stocks from Google (GOOGL), Adobe (ADBE), and Amdocs (DOX). The following normalization methods were used: standard scaler, min-max scaling, robust scaling. The latter is the most efficient as it takes into account outliers, and was thus chosen as the final normalization method.

The data was split into training and testing with 90% and 10%, respectively. This preprocessing is part of the split_data function.

For price prediction, we used the LSTM from Tensorflow's Keras library. To build the model we set the dropout rate (layers.Dropout) to 20%. A neural network's drop out randomly sets certain input units to zero upon training in order to avoid overfitting. Our first LSTM layer is 128 units, followed by 64 units. These were unit sizes used in similar projects, and upon manipulation were decided to be the best. Then we add dense layers to the model to integrate the inputs under an activation function. We found the tanh activation function to be less sensitive to outliers. The loss is based on Mean Squared Error (MSE), and the optimizer chosen is the Adam optimizer. With regards to time-series data, this optimizer is more appropriate because it takes into account the moving averages of the parameters, which is an important indicator for time series.

When building the network we chose the layer parameters [3,7,1] because there are 3 inputs (Open, High, and Closing price), a time window of 7 days (how long the LSTM looks back in time at a given point), and 1 output.

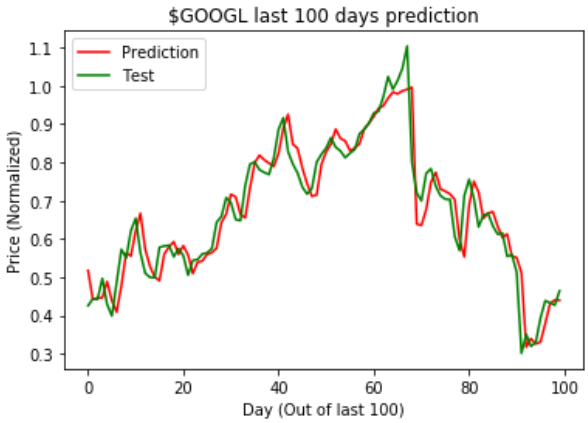
To fit the data to the model, we found that a large batch size and a large epoch size helped fit the data better (epochs = 500, batch_size = 500).

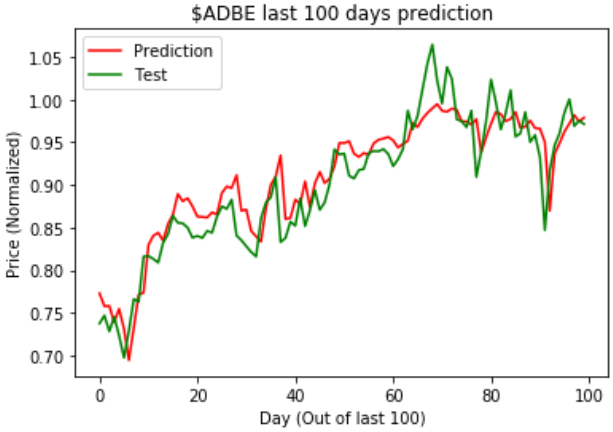
Results

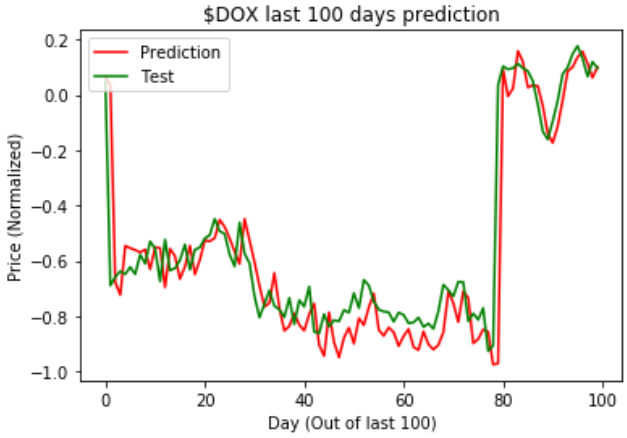
We used mean squared error (MSE) to evaluate our LSTM model. Knowing the accuracy of LSTM this was not surprising. The following chart shows the MSE for training and testing.

```
===TRAINING SCORES===
GOOGL: 0.00337 MSE
ADBE: 0.00065 MSE
DOX: 0.00843 MSE
===TEST SCORES===
GOOGL: 0.00326 MSE
ADBE: 0.00095 MSE
DOX: 0.02196 MSE
```

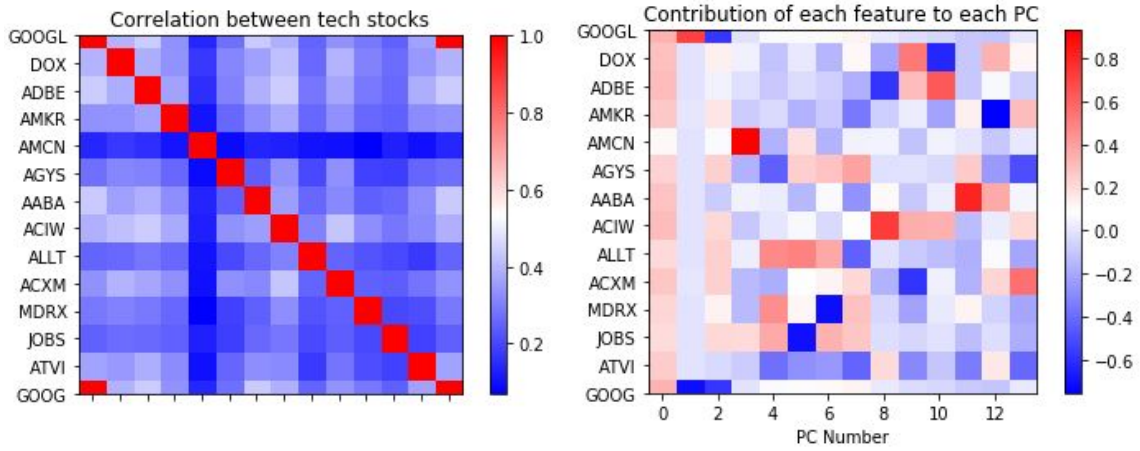
The following are the predicted prices at the last 100 days trained on the previous 900.







We found that the stocks which were most correlated to other tech stocks were 1) GOOGL, 2) ADBE, 3) ACIW, 4) DOX. We looked at these stocks afterwards with the LSTM.



Discussion

For PCA, there are a vast number of other measures that could have been used besides just prices alone. However based on our time and available resources we decided to proceed with just prices, which is perhaps why the PCA analysis was not as informative as we would have liked. One way to carry out dimensionality reduction could have been based on certain measures like “beta” (a measure of systematic risk), return on assets, return on equity, and others. Doing this might be better because they might represent more unique qualities with uncorrelated properties. Because the two go well together, we had also initially planned on performing some variant of clustering algorithms such as k-means after dimensionality reduction to possibly find clusters of stocks that behaved the same way. It would be beneficial to use the clusters of similar stocks to train an LSTM model because the clusters represent stocks that are highly correlated. We then used correlation as another method, but were not able to bridge the gap between highly correlated stocks and using their relationship to predict prices, so we ended up investigating these two topics (PCA/correlation and LSTM prediction) in slightly different ways.

Another improvement that could have been made would have been to denoise the time series more rigorously before data processing. One such method is Singular Value Decomposition (SVD), which is effective for deterministic time series.

If everything worked perfectly, we would have explored different more mathematically involved methods for analyzing the stock market. Because the markets are chaotic and often random in nature, a hidden Markov model could have made interesting contributions. Areas from dynamical systems research such as empirical

dynamical modeling with Taken's theorem is also known for giving very accurate results on time series data.

There are other areas of machine learning that can be useful for this subject. We have been focusing on quantitative analysis on this project (looking purely at numerical data for predictions) but machine learning assisted fundamental analysis is also possible. This has been done with sentiment analysis on how people feel about a certain company's value.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166. doi: 10.1109/72.279181

Nelson, David M. Q., et al. "Stock Markets Price Movement Prediction with LSTM Neural Networks." 2017 International Joint Conference on Neural Networks (IJCNN), 3 July 2017, doi:10.1109/ijcnn.2017.7966019.

Pasini, G. (2017). Principal Component Analysis For Stock Portfolio Management. International Journal of Pure and Applied Mathematics, 115(1). doi: 10.12732/ijpam.v115i1.12