Stock Market Movement Prediction Using Graph Convolutional Networks

Dylan Loe, Sung-lin Chang, and Jason Chau
University of California, San Diego

Abstract

In this project, we aim to produce a tool that will be able to predict the stock movement of a company. The output will be a binary output where it will indicated whether we are bullish or bearish on a stock. In our pursuit of making this tool, we will incorporate graph convolutional networks to take advantage of the interconnected features of stocks.

1 Introduction

Stock markets are important aspects of many economies and investors are constantly looking for the most successful companies from which to buy and the least successful companies whose shares should be sold. Stock market prediction is a certainly lucrative domain to which machine learning methods can be applied, and recent advancements in the field of artificial intelligence are heavily aiding this prediction. Powerful new types of neural network models called graph convolutional networks (GCNs) can effectively learn from data contained within a network structure [1]. GCNs facilitate ML approaches on graphs similarly to the way traditional CNNs conveniently operate on structured data like images. Whereas before one might derive features from the graph's inherent properties and use those for a machine learning task, one now has access to powerful algorithms that can learn directly from the graph itself. Stock markets consist of highly self-dependent data and have an inherent network structure that can be appropriately exploited using these techniques.

Human analysts typically employ either fundamental or technical analysis to make their market predictions. Fundamental analysis assesses companies' rate of growth through compiled metrics like a price-to-earnings ratio; if a company is perhaps overvalued, sell it, as its stock price may likely drop soon. Technical analysis tries to predict stock market movement through trends and solely concerns stock price information; how did a market behave previously, and when might it repeat this behavior? While human analysts will not be replaced by machines just yet, stock markets are driven by many different complicated sources of information and are becoming too complex for a single investor to fully understand. For instance, the 2008 financial crash, the current financial crisis fueled by the pandemic, and the recent mania caused by Reddit's r/wallstreetbets demonstrate the ways intricate societal interactions heavily influence the stock market [9]. This complexity makes it difficult as well as time consuming to predict stock prices using only handcrafted technical indicators and previous market data. Stock markets are also well known to be very noisy sources of data, so much so that there is theory that stock asset prices are actually random walks themselves [10]. A competing theory called the efficient market hypothesis posits that markets actively reflect all current information, and no knowledge about companies can yield a long-term advantage over other investors [11]. Despite this seemingly difficult task, several machine learning methods have been employed to solve this problem.

Stock market prediction can be divided into two categories: price prediction and movement prediction. Price prediction is a regression problem where a real number is predicted for each stock at each day, while movement prediction is a binary classification problem predicting whether a stock moves up or down on each day. Movement prediction is more feasible than price prediction and is the task on which we shall focus for this project.

Previous approaches to this task use statistical models like ARIMA to make time series predictions, and traditional machine learning algorithms like logistic regression and support vector machines were also utilized for classification; more recent approaches use graph convolutional networks followed by a form of recurrent neural network, useful to model sequential data, to make predictions [2].

2 Related Work

There exists multiple ways to construct a graph for the model based on stock data. A knowledge graph can be constructed using extrinsic data about the stock market that models the type of knowledge a professional investor might use in making predictions [2]. For example, one could make an industry graph where the edges are 1 if the companies represented by the nodes are in the same industry and 0 otherwise. Ye et al. construct three knowledge graphs based on industry, shareholding, and topicality information to incorporate relationships between companies in their model using a GCN to learn informative features for corporations followed by a GRU that helps model the temporal dependencies in the predictions [3]. Their methodology allows them to incorporate multiple graphs and sources of information at once into their model for prediction. However, this involves the collection of specific data that is not necessarily publicly available or easy to collect, so we forego this approach in favor of using a correlation graph. The interdependence of the stock price movements allows one to examine the cross correlation between the time series data and connect an edge between two companies if the correlation exceeds a threshold value and provides a method to create a graph modeling the stock market with more freely available and collectable stock price data [4, 5].

3 Methods

In order to model the stocks as a set of nodes and edges between them in a network, we construct a graph based on the cross correlation between the time series data we collect for each stock's closing price. We define the cross correlation c_{ij} between stock price time series x_i and x_j as follows where t ranges from day 0 to day N-1:

$$c_{ij} = \frac{\sum_t [(x_i(t) - \overline{x_i})(x_j(t) - \overline{x_j})]}{\sqrt{\sum_t (x_i(t) - \overline{x_i})^2} \sqrt{\sum_t (x_j(t) - \overline{x_j})^2}} \tag{1}$$

where $\overline{x_i}$ and $\overline{x_j}$ are the average values of the time series, and which corresponds to the sample Pearson correlation coefficient from our data [5]. An edge is formed between two stocks if their correlation exceeds a threshold value, and after some experimentation on the effects the threshold has on the sparsity of the graph, we use a threshold value of 0.4, which gives an average node degree of about 5 [4].

$$A_{ij} = \begin{cases} 1, & \text{if } c_{ij} > 0.4 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Making one correlation graph over the entire period our data stretches could result in capturing spurious correlations, so we difference the time series data once to get its increments ($X_t = S_t - S_{t-1}$) and apply a logarithm ($X'_t = log(X_t)$) to reduce the effect of variance [4].

We use a graph convolutional neural network to make daily predictions on our data. The adjacency matrix used in the GCN is usually preprocessed by adding an identity matrix to it to ensure that a node's own features are being included in the aggregation that becomes its features in the next iteration; however, since each stock has perfect correlation with itself, the main diagonal of the adjacency matrix of our correlation graph is guaranteed to be all ones and already contains self-loops [1]. The representation of the graph is then normalized using the equation

$$\hat{A} = D^{-1/2} A D^{-1/2} \tag{3}$$

where D is the diagonal node degree matrix of A [1]. Each day has four features (opening, low, and high price, and trading amount), and stock movement on day i is predicted from these features from the past p days, where p, a lag variable, is usually set to 5 as there are that many trading days within a week. Therefore, our feature matrix X is $\in \mathbb{R}^{n \times 4p}$ where n is the number of stocks. Our target labels Y are 1 if closing price is greater than that day's opening price, and 0 otherwise. Two graph

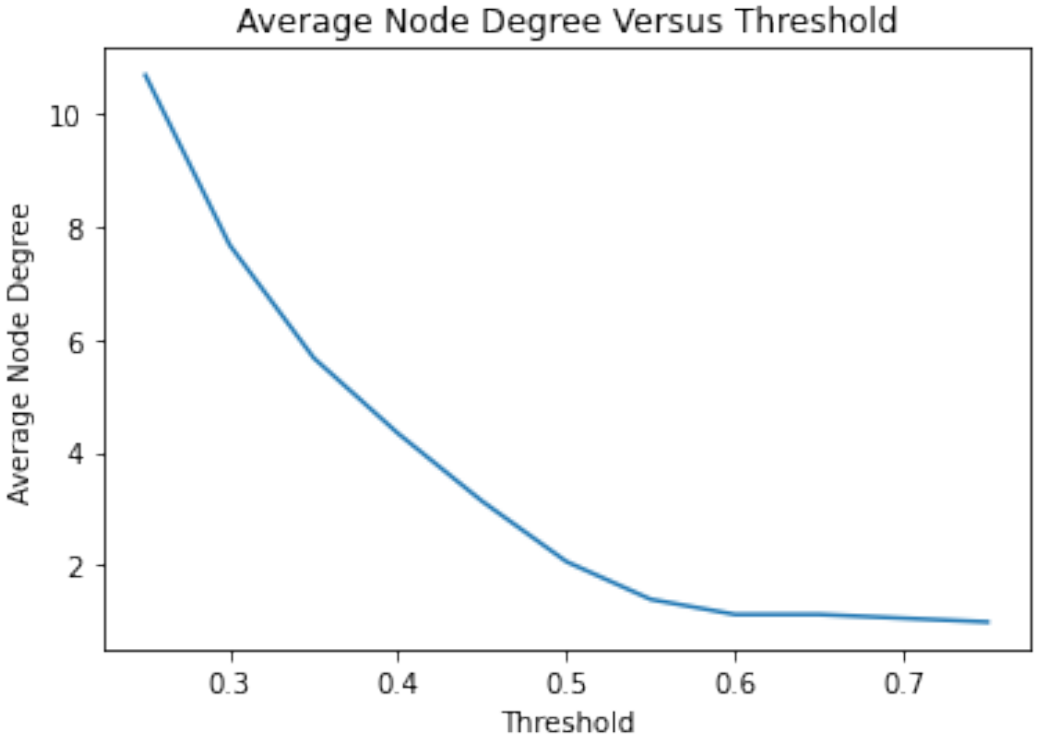

Figure 1: Graph of average node degree vs threshold

convolutional layers are used, followed by a fully connected layer, to obtain the probability of stock increase for each stock on a particular day.

$$\begin{aligned} H^{(i+1)} &= \sigma(\hat{A}H^{(i)}W), \\ H^{(0)} &= X, \\ \hat{Y} &= \sigma(H^{(2)}W) \end{aligned} \tag{4}$$

The network is trained using binary cross-entropy loss for 100 epochs using a learning rate of .001, number of hidden units of 32, and Adam as the optimizer after some hyperparameter tuning. We adapted Thomas Kipf's Pytorch implementation of graph convolutional networks for our vanilla GCN model [8].

4 Data

For this project, we collected stock price data from the 30 stocks that comprise the Dow Jones Market Index over a 12-month period (127 trading days) ranging from January 2020 to January 2021 using Yahoo Finance's API. We chose this data because we did not want to compare pre-pandemic stock prices to post-pandemic prices, as market conditions varied drastically due to the pandemic, but we still wanted to work with as much data as we possibly could given this constraint. Our model features consist of opening price, low price, high price, and trading amount. The features are min-max normalized to deal with discrepancies of scale between stock price (in the 100s) and trading amount (in the 10000s). Our labels for each day will be 1 if closing price is greater than that day's opening price and 0 otherwise. We split the data into a 70-30 train-test split for prediction.

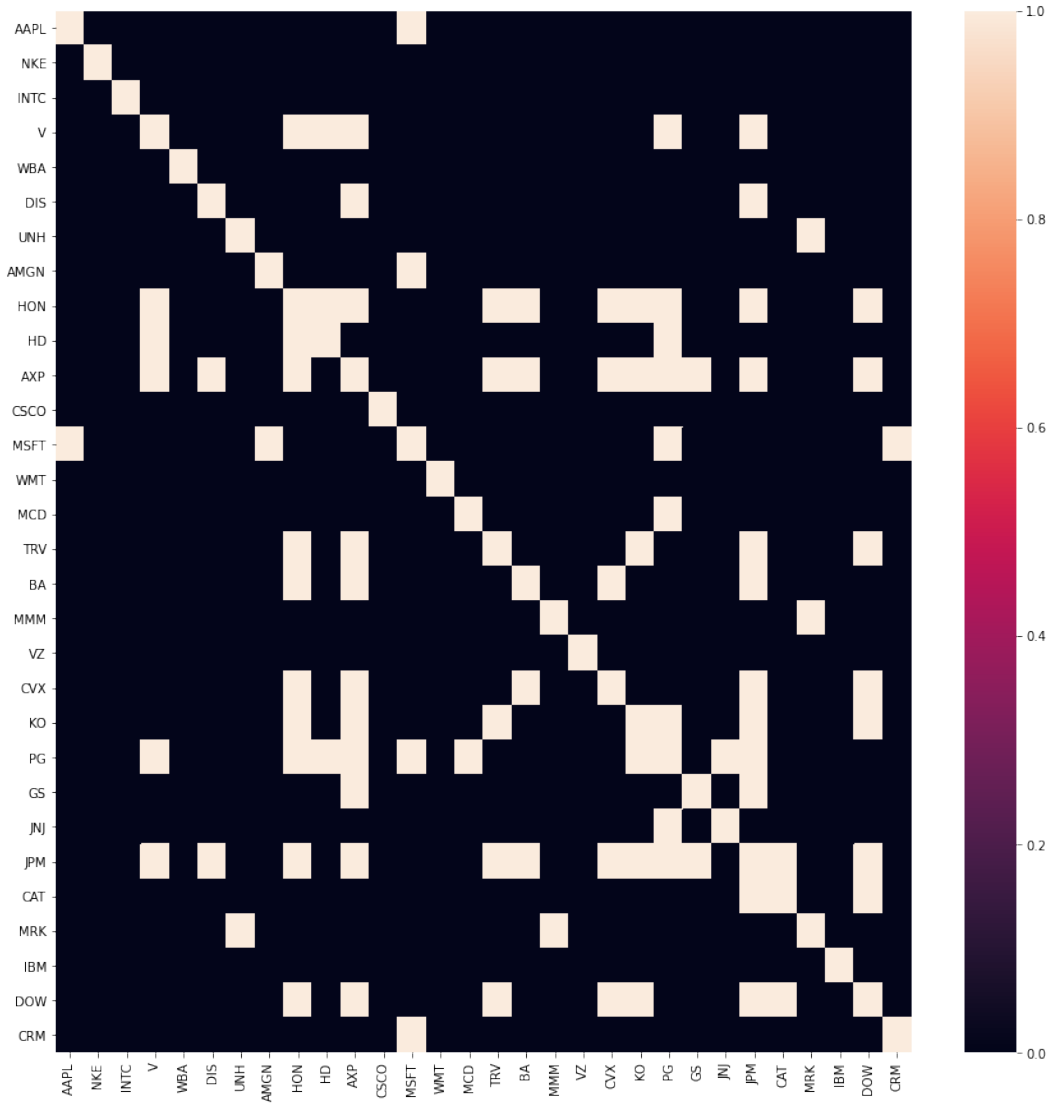

Figure 2: Adjacency matrix representation of correlation graph

Model	Accuracy
ARIMA	60
Fully Connected Network	60
GCN	60

Table 1: Summary of results of models on Dow Jones data. Accuracy is averaged over all stocks and days.

5 Results

We compare the results of our model using graph convolutional networks to two baseline models, an ARIMA model and a fully connected network that does not incorporate a graph trained on the same combination of hyperparameters as the GCN [7]. The ARIMA model performs well at 60% accuracy over the test set. The fully connected network performs similarly to the GCN at 60%, perhaps because our original graph was not of very good quality and doesn't add too much to our models' ability to predict stock movement. This is also accuracy averaged over the days and stocks

in our test set, and the average varied greatly between them. There may be certain days that have harder prediction tasks than others.

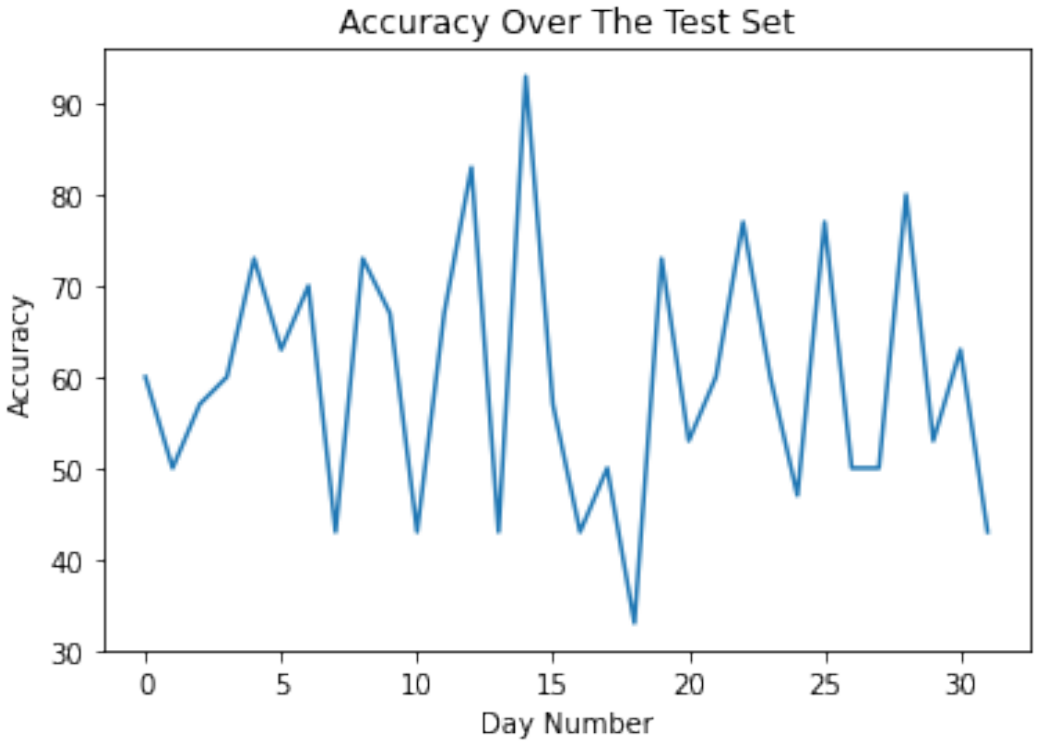

Figure 3: Graph of accuracy of GCN model on the test set over each day.

6 Conclusion

Although our models performed similarly, we show that graph convolutional networks are one method through which one can predict how stock markets move. If we had access to a better graph than a correlation graph and more data over more stocks, the graph convolutional network would perhaps use that to outperform the other two approaches.

7 References

[1] Kipf, T., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks.

[2] Jiang, W. (2020). Applications of deep learning in stock market prediction: recent progress.

[3] Ye, J., Zhao, J., Ye, K., & Xu, C. (2020). Multi-Graph Convolutional Network for Relationship-Driven Stock Movement Prediction.

[4] Patil, P. (2019). Stock Market Prediction Using Ensemble of Graph Theory, Deep Learning, and Machine Learning Models.

[5] Tse C. K., Liu J., and Lau F. C. M. (2010). A network perspective of the stock market. Journal of Empirical Finance, vol. 17, no. 4, pp. 659–667.

[6] Y. Xu & S. B. Cohen. (2018). Stock movement prediction from tweets and historical prices. ACL, pp. 1970–1979.

[7] Loukas, S. (2020). Time-Series Forecasting: Predicting Stock Prices Using An ARIMA Model. Medium. https://towardsdatascience.com/time-series-forecasting-predicting-stock-prices-using-an-arima-model-2e3b3080bd70.

[8] Kipf, T. (2020). Pygcn. GitHub. https://github.com/tkipf/pygcn.

[9] Matsunaga, D., Suzumura, T. and Takahashi, T. (2019). Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window AnalysisCoRR, abs/1909.10660.

[10] Smith, T. (2020). Random Walk Theory. Investopedia. https://www.investopedia.com/terms/r/randomwalktheory.asp.

[11] Downey, L. (2021). Efficient Market Hypothesis (EMH). Investopedia. https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp.