Predicting Stock Trend Using GNN

Zhiluo Chen [1, †, *], Zeyu Huang [2, †], Yukang Zhou [3, †]

[1] Washington University in St.Louis, 1 Brookings Drive, St. Louis, MO 63130, USA

[2] Computer Science, Nanjing University, Nanjing City, Jiangsu Province, 210046, China

[3] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100080, China

* Corresponding Author Email: Chenzhiluo@wustl.edu

†These authors contributed equally to this work and should be considered co-first authors.

Abstract. In stock market, numerous ways have been used to predict the future asset return. Traditional time series models and neural network models are both popular in practice. Many of them have been shown that they do not make full use of the property of long-term dependency while with the help of graph structure, one can turn time series into complex network and keep the long-term dependency property. To make use of the property of the long memory of financial market in prediction, our framework first applies the visibility method to turn stock price data into graph structure, and then applies graph neural network to make graph classification task to let the network learn the overall topological structure of the graphs. The result shows that with the use of such property, our model can successfully forecast the future stock trend.

Keywords: Graph Structure; Predicting Stock Trend; GNN Model.

1. Introduction

Stock return in traditional financial theory follows Brownian Motion. Investor cannot get insights from the past information. However, in practice, there are lots of evidence showing that, in fact, the market doesn't follow Brownian Motion. There is lots of information in the past that can be used to predict the future.

Hurst index is one type of the evidence. Empirical study shows that, in lots of financial assets, their historical movement indicates a Hurst index level over 0.5, which means past information might have a slow decaying or long-range correlations [1].

Evidence has shown that such long-range correlations cannot be modeled by traditional financial model such as ARMA and GRACH [2]. But luckily, following the further research of Hurst Index, complex network is one of the models that can maintain such property. With the help of the visibility graph, time series can be transformed into a complex graph while maintaining its property of long-range correlations.

In this paper, we propose a method to make stock trend prediction with the combination of the visibility graphs and Graph Neural Network together. To avoid the noise of stock return series, rather than simply using stock return as our label, the future information of the close, high, low and open of each stock is used to decide whether the stock is in an upward trend or downward trend. Second, the close series of each stock is turned into visibility graphs. Finally, the graph classification task is applied on the visibility graphs using Graph Neural Network.

The data used is from China Securities Index 300 from 2010 to 2022, which contains 300 most popular stocks in China, to test the power of our model. The result shows that our model can make prediction of future trend.

Section 2 discusses the related literature of stock market prediction, graph neural network. Section 3 introduces our method and Section 4 discusses our experiment settings. Finally, Section 5 draws some conclusions about our work.

2. Related Work

2.1 Stock Market Prediction

Financial market has long been drawing close attention from the world. Stock market prediction models, the most crucial part, take overall consideration of possible factors to forecast the profits and stock's prices. However, the intrinsic complexity of stock market, the chaotic property as well as the long-term dependency of price series are key challenges for market prediction models.

Traditionally, statistical models and machine learning models are the two mainstream research directions. Yule first introduced an autoregressive (AR) model, which employs time series analysis to give some clues about the market prices [3]. Furthermore, a more advanced model, autoregressive moving average (ARMA), was proposed to make insightful predictions on financial market, which was one of the most commonly used statistical models [4]. On the grounds of the market price series' intrinsic complexity, a plenty of nonlinear models were proposed to improve the performances, including nonlinear autoregressive exogenous (NARX) model, and quantile autoregressive (QAR) model [5][6]. However, evidence has shown that such time series model cannot capture the long-term dependency well enough [2].

Lots of machine learning methods were also employed in stock market prediction, providing stronger capability to express the complicated hidden rules of the stock market. Random forest (RF) algorithm was applied to analyse the market behaviors and make forecasting [7]. However, the method is not sophisticated enough to express the stock market. Thenozhi and Chand conducted investigations on forecasting global stock market prices, showing that the support vector machine (SVM) method outperformed regression models [8].

The rise of deep learning method gave birth to extensive research on applying deep learning method on stock price prediction. The greater expressive power enables the deep learning method to go beyond other traditional methods. Artificial neural networks (ANNs) were investigated by Lin and Yu to make prediction and find trading strategy for the Taiwan Weighted Index [9]. Nonetheless, ANN couldn't make full use of the long-term dependency and the model is not sufficiently sophisticated [10]. Some more advanced deep learning models were applied in research. RNN model is one of the most commonly used frameworks to analyse stock trading time series, which showed significantly improvement in performance [11]. Long short-term memory (LSTM) model further promoted the model's capability to deal with data series of long-term dependencies [12]. Yu and Li implemented and compared LSTM model with other models, which showed LSTM's out-performance over other models [13].

2.2 Graph Neural Network

The technique of graph embedding as well as the graph neural network (GNN) approach got mature and stronger recently. Instead of directly learning from the stock price time series, it can be easier to extract and learn the latent features and properties from graph representations. As for the stock price prediction, by transforming the raw price series into graph data, the long-term dependencies of price series, and the chaotic properties become easier to handle, since graph data conveniently adds direct link between nodes and provides more knowledge about the latent structure of points in price series.

The transformation from time series into complex network domains could be performed by visibility graphs (VG) method, which has been proven to be able to calculate the Hurst Index, meaning that the long-term dependency characteristic is still maintained after such transformation [14].

A recent work followed this idea. The price series is first transformed into graph, then passed through a dual stage attention-based RNN (DARNN) encoder and decoder to incorporate attention and node weights. Finally, a cross asset attention network (CAAN) is used to utilize the interrelationship among stocks and make the prediction [15]. Nevertheless, it is far from practice and the model's inefficiency is one of its fatal weaknesses.

Recently, Wijesinghe and Wang proposed a novel GNN models called GraphSNN and proved that GraphSNN is intrinsically more expressive than traditional GNNs [16]. Inspired by this work and the virtues mentioned above of transforming time series into graph, we propose our approach.

3. Method

In this section, we propose our model for the stock market prices prediction. The model consists of two main parts: a VG module, and a GNN module.

The input data are stock price series. The VG module is first applied to transform the raw market price series into corresponding time series graphs, in order to deal with the evolving and chaotic property of stock price series. Next, the GNN module takes the converted graphs as input, trains, learns, and makes predictions based upon them.

3.1 Label Module

Asking a good question to machine is vital. Especially for the financial data since there are full of noise. Instead of using the raw return or return in the future as our label, we proposed a method to identify whether an individual stock is in an upward trend or downward trend by using future return and their topological structure. The method is inspired by a common setting that although forecasting future is difficult, an investor can often identify trend by graph in the past with the local highest and local lowest. We set an algorithm close to such idea. The algorithm distinguishes the knee point of stocks by finding local highest points and local lowest points using future and past price data. Then the algorithm deliver label to each day by the sign of price change between those knee points.

The following graph shows the results of our label. The sign of the slope of the trend line indicates the label, for which positive sign means upper trend while negative sign means downward trend. The trend line is undefined at the last part of the figure due to the reason that the future information is not enough for the algorithm to identify the trend it is in.

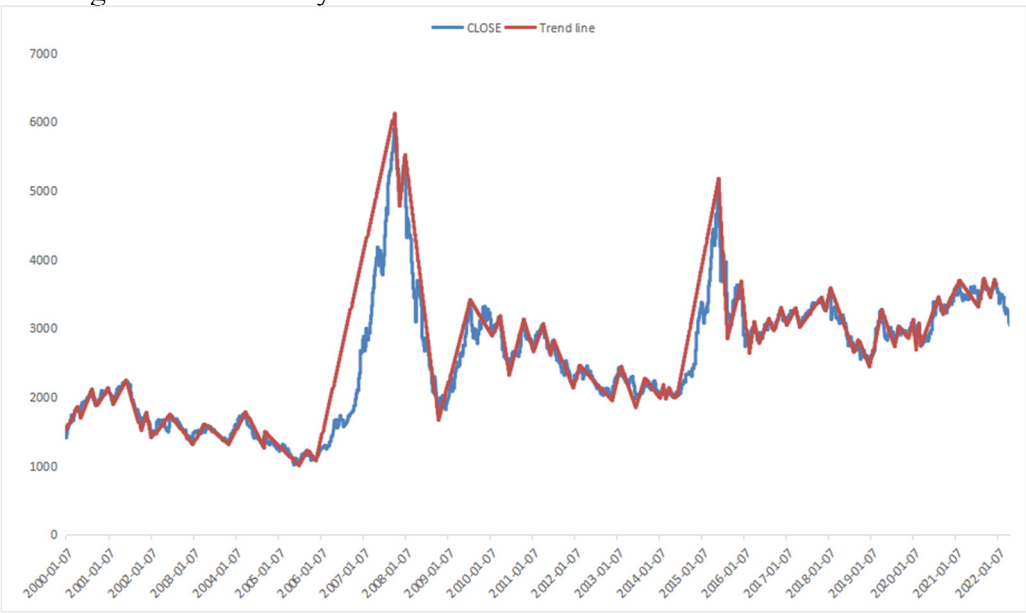


Figure 1. Label and stock price

3.2 Visibility Graph Module

As discussed above, the visibility graph is helpful when we want to make use of the long-range dependence features of financial markets. It is also an algorithm free of parameters. With our particular interest, we select the visibility graph as our method to turn time series into graphs. Among different version of VGs, we select the normal VG as our algorithm.

Visibility graph is a graph which shows the visibility relationship between different vertices of several obstacles. It connects the start node, the vertices of all obstacles and the target nodes to each

other to build the graph. The edge of visibility graph connects two points that can see each other. Considering the graph below, it is clear that each pair of vertices have a straight line between them if the line connects two vertices and dose not intersect with obstacles.

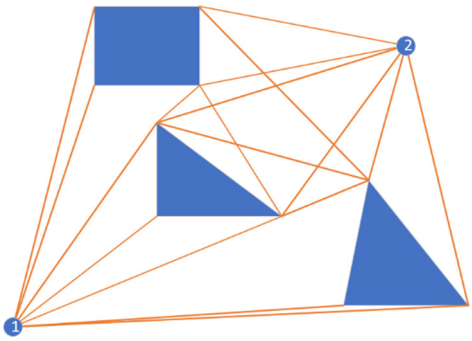
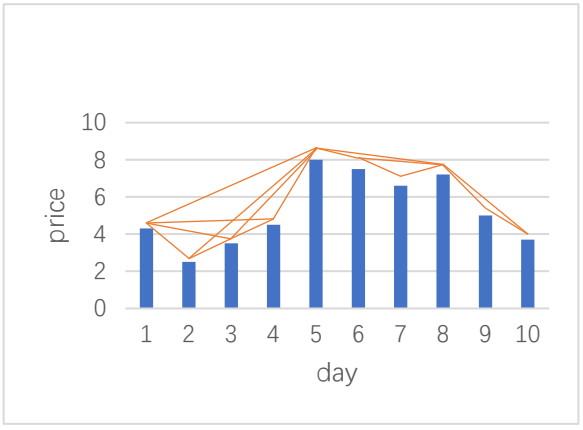


Figure 2. Visibility graph with obstacles

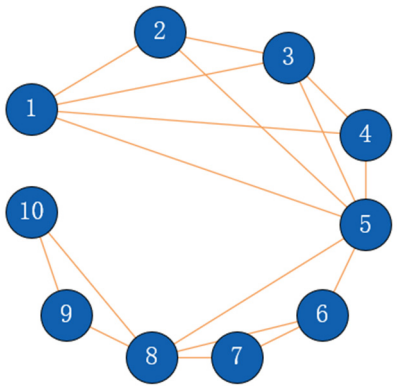
Histogram is used to show a series of stock price in selected time. Each vertex in the stock price graph corresponds to a data point in time series graph. For the stock price graph, p represents price and d represents day. Select any 3 points in the graph, $(d_i,\ p_i)$, $(d_k,\ p_k)$ and $(d_j,\ p_j)$ where $d_i<d_k<d_k$. $(d_i,\ p_i)$ and $(d_j,\ p_j)$ can be connected with a line which dose not intersect with any other columns if and only if they satisfy the formula below:

$$p_k < p_i + \frac{d_k - d_i}{d_j - d_i}(p_j - p_i) \tag{1}$$

In this way, we get corresponding Visibility graph. The Figure 3.a and 3.b is an example for how visibility graph turns time series into graph.



(a)



(b)

Figure 3. Stock price graph containing 10 temporal points and corresponding time series graph

3.3 Graph Neural Network Module

In GNN module, we implemented a GraphSNN. The transformed time series graph is passed into the GraphSNN, together with the labels generated by the algorithm mentioned above. Wijesinghe and Wang proved that GraphSNN is more expressive than Weisfeiler-Lehman algorithm (1-WL) in testing non-isomorphic graphs, while it is reported that traditional popular GNNs are at most as powerful as 1-WL [16][17]. As a result, GraphSNN has more expressive intrinsic power in describing different substructures.

The GraphSNN framework is constituted by k layers. Let $\{G_1, G_2, \ldots, G_n\}$ denotes the set of converted price graphs for each stock price series (n in total) from VG module, $\{Adj_1, Adj_2, \ldots, Adj_n\}$ indicates the corresponding adjacent matrix, and let $h^t_{v(i)} \in R^m$ denote the hidden state of vertex $v(i) \in G_i$ at layer $t > 0$ (m denotes the size of hidden state). $h^t_{v(i)}$ is the

output of GraphSNN layer t for $v(i)$, and $h^0_{v(i)} \in R^M$ represents the initial feature of $v(i)$ (M denotes the size of initial feature). These $h^t_{v(i)}$ are stacked together to get H^t_i.

$$H^t_i = stack([h^t_{v(i)};], axis = 0), \forall v(i) \in G_i \quad (2)$$

Then the GraphSNN layer $t(t > 0)$ for G_i formally writes as:

$$H^t_i = \text{MLP}^t(\text{AGGREGATE}^t(H^{t-1}_i, Adj_i)) \quad (3)$$

MLP^t is the Multiple-Layer Perception unit for layer t, and AGGREGATE^t is GraphSNN's aggregation function for layer t. Before giving the definition of AGGREGATE^t, some concepts need to be introduced first.

Let $G_i = (V_i, E_i, H_i)$, where V_i and E_i is the vertex and the edge set respectively, and H_i denotes the initial features set. The set of neighbours of a vertex v is denoted as $N(v) = \{u \in V | (v, u) \in E\}$. The neighbourhood sub-graph of v, denoted as S_v, represents the induced sub-graph of G by vertex set $N(v) \cup \{v\}$. The overlap sub-graph of two adjacent vertices $v, u \in V$, denoted as S_{vu}, is defined as $S_{vu} = S_v \cap S_u$

The key idea of GraphSNN is the structural coefficient and defining aggregation function [16]. For vertex v and its neighbour u, the structural coefficient A_{vu} is defined as $A_{vu} = \omega(S_v, S_{vu})$, where ω is a function that maps a pair of neighbourhood sub-graph and overlap sub-graph to a real number. In GraphSNN, the structural coefficients, or ω is given as:

$$A_{vu} = \omega(S_v, S_{vu}) = \frac{|E_{vu}|}{|V_{vu}||V_{vu}-1|}|V_{vu}|^\lambda \quad (4)$$

where $\lambda > 0$.

Then the AGGREGATE^t is defined as follow:

$$\text{AGGREGATE}^t(v(i); H^{t-1}_i, Adj_i) = \gamma^t(\sum_{u \in N(v)}(\tilde{A}_{vu}) + 1)h^{t-1}_{v(i)} + \sum_{u \in N(v)}(\tilde{A}_{vu} + 1)h^{t-1}_{v(i)} \quad (5)$$

where γ^t is a learnable parameter.

As a result, the hidden state of $v(i)$ at layer t is calculated by:

$$h^t_{v(i)} = \text{MLP}^t(\text{AGGREGATE}^t(v(i); H^{t-1}_i, Adj_i)) = \text{MLP}^t(\gamma^t(\sum_{u \in N(v)}(\tilde{A}_{vu}) + 1)h^{t-1}_{v(i)} + \sum_{u \in N(v)}(\tilde{A}_{vu} + 1)h^{t-1}_{v(i)}) \quad (6)$$

and the output of GraphSNN layer t for G_i, H^t_i is calculated by the formula (2).

After we get all the H^t_i for a G_i, a concatenation and then a vertex-wise summation is applied on them. We get $H_i \in R^{M+k*m}$ as below:

$$H_i = sum(concat(H^t_i \ \forall t \in [0,1,\ldots,k], axis = 1), axis = 1) \quad (7)$$

After that, a linear transformation is applied on H_i using a learnable matrix $W_2 \in R^{(M+k*m)\times 2}$, and the final output of GraphSNN is calculated by:

$$P_{pro} = softmax(H_i W_2, axis = 0) \quad (8)$$

The prediction is made based on the two values, one of which indicates the probability of the stock price to go up and the other indicates the inverse.

4. Experiments

4.1 Data

We obtain the component stocks of CSI-300 index at 2021.12.31 as well as their historical data, including daily open, high, low, and close from 2010.01.04 to 2022.06.24. The daily data is obtained from Wind, a financial platform with enriched finance data. The price adjustment, according to the dividend, stock split and other events, is made on the raw daily quote data in general for the continuity of price graph. CSI300 index includes the biggest 300 market value stocks in China in recent, and is often used to represent the overall large market value and market performance of China securities. These stocks are often the most liquid stocks in China. We filter out the stock that is public before 2021.06.01 for the best of following experiments. The open, high, low, and close are all used in label

procedure, while only the close is used to generate the visibility graph and served as the input of the graph neural network.

4.2 Setting

We intentionally set our test fold limits to the data after 2020 Jan and our train data is limited before 2020 Jan to avoid future information. We use the label algorithm mentioned above to identify whether at each time point it is on an upward trend or downward trend, and then filter out the data that the trend sign is undefined. For each stock, we take a rolling window of 42 days of close series to generate visibility graph and then generate its adjacent matrix to predict the trend. For each stock, we run a ten-fold cross-validation then store the test set performance and train set performance. We define our learning target y_t as follow:

$$\mathrm{la}(x_t) = \begin{cases} 1, & x_t \text{ in upward trend} \\ -1, & x_t \text{ in downward trend} \end{cases} \tag{9}$$

$$y_t = \begin{cases} 1(\text{upward trend}), & \mathrm{la}(p_{t+1}^c) = 1 \\ 0(\text{downward trend}), & \mathrm{la}(p_{t+1}^c) = -1, \end{cases} \tag{10}$$

where $\mathrm{la}(\cdots)$ denotes the label algorithm mentioned above. p_t^c represents the close of a stock at time t. y_t is simply the transformation of the $\mathrm{la}(p_{t+1}^c)$. We collect 70000 + samples in total and nearly 10% for the test set. We evaluate our experiment performance using accuracy metric and the loss function is cross entropy.

Table 1. Statistic description of CSI300 index datasets. The label for training and test is almost balanced. There are 688515 samples in total and 617700 of them are assigned to training set and the rest 70815 are assigned to test set.

Dataset		Sample	upward trend sign		downward trend sign	
			Sample	ratio	Sample	ratio
Train	2010.1-2020.12	617700	323336	52.35%	294364	47.65%
Test	2021.1-2022.6	70815	34430	48.62%	36385	51.38%

For the graph neural network setting, the total epochs are set to 500. We choose Adam as our optimizer and its initial learning rate is set to be 0.009 used in the model and the L2 penalization rate is 0.009. In addition, we set the dropout rate to 0.6 for each MLP layers in the module and the dropout rate to 0.5 for the concatenation of output. We follow the advice of the original author of the GraphSNN and set the number of MLP layer to 2 for the best of graph classification task.

4.3 Result

The test performance is shown on the following chart. Given the fact that signal to noise ratio is quite low in stock markets, our model achieves a high accuracy rate.

Table 2. Statistic description of the test set result. The mean of accuracy of each fold.

		1	2	3	4	5	6
Accuracy	fold	88.72%	86.53%	86.12%	87.69%	83.25%	89.17%
		7	8	9	10	mean	std
		86.85%	87.94%	88.79%	84.34%	86.94%	1.95%

As the chart shows, the standard deviation of the accuracy is 1.95% and the average performance is 86.94%. The model successfully identifies the trend using graph neural network.

5. Conclusion

The result shows that by exploiting the long-term dependency characteristics, the stock trend can be predicted in the future. The combination of visibility graph and graph neural network is beneficial to the understanding of movements of stocks.

For the possible future work, there are mainly three aspects of improvement. First, the visibility graph is not the only way to model the characteristics of long-term dependency, and other transforming methods are also worth considering. Second, a better labeling function could be designed. Instead of using a post experience label, the label can be generated through learning the different structure characteristics of the time series graph structure through an embedding procedure. Third, the cross information of each stock could be used since the cross data is also informative in practice.

References

[1] A. Lo, Long-term memory in stock market prices, Econometrical., vol. 1279–131.

[2] Mandelbrot Benoît, & Hudson, R. L. (2008). The (mis)behavior of markets: A fractal view of financial turbulence. Basic Books.

[3] Yule, G.U. (1995). The Foundations of Econometric Analysis: On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers (Philosophical Transactions of the Royal Society of London, A, vol. 226, 1927, pp. 267–73).

[4] Young, P.J., & Shellswell, S. (1972). Time series analysis, forecasting and control. IEEE Transactions on Automatic Control, 17, 281-283.

[5] Lin, T., Horne, B.G., Tiňo, P., & Giles, C.L. (1996). Learning long-term dependencies in NARX recurrent neural networks. IEEE transactions on neural networks, 7 6, 1329-38.

[6] Li, L., Leng, S., Yang, J., & Yu, M. (2016). Stock Market Autoregressive Dynamics: A Multinational Comparative Study with Quantile Regression. Mathematical Problems in Engineering, 2016, 1-15.

[7] Stock Market Prices Prediction using Random Forest and Extra Tree Regression. (2019). International Journal of Recent Technology and Engineering.

[8] Thenmozhi, M., & Chand, G.S. (2015). Forecasting stock returns based on information transmission across global markets using support vector machines. Neural Computing and Applications, 27, 805-824.

[9] Lin, T., & Yu, C. (2009). Forecasting Stock Market with Neural Networks.

[10] Huynh, H.D., Dang, L.M., & Duong, D. (2017). A New Model for Stock Price Movements Prediction Using Deep Neural Network. Proceedings of the Eighth International Symposium on Information and Communication Technology.

[11] Wang, Q., Xu, W., Huang, X., & Yang, K. (2019). Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. Neurocomputing, 347, 46-58.

[12] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9, 1735-1780.

[13] Yu, S., & Li, Z. (2018). Forecasting Stock Price Index Volatility with LSTM Deep Neural Network.

[14] Lacasa, L., Luque, B., Luque, J., & Nuño, J.C. (2009). The visibility graph: A new method for estimating the Hurst exponent of fractional Brownian motion. EPL (Europhysics Letters), 86, 30001.

[15] Wu, J., Xu, K., Chen, X., Li, S., & Zhao, J. (2022). Price graphs: Utilizing the structural information of financial time series for stock prediction. Inf. Sci., 588, 405-424.

[16] Wijesinghe, A., & Wang, Q. (2022). A New Perspective on "How Graph Neural Networks Go Beyond Weisfeiler-Lehman?". ICLR.

[17] Leman, A. (2018). THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN.