Static-Dynamic Graph Neural Network for Stock Recommendation

Yanshen He
Central South University
Changsha, China
heyanshen@csu.edu.cn

Qiutong Li
Central South University
Changsha, China
qiutonglee@csu.edu.cn

Feng Wu
Central South University
Changsha, China
204712213@csu.edu.cn

Jianliang Gao
Central South University
Changsha, China
gaojianliang@csu.edu.cn

ABSTRACT

Stock prediction is a hot topic of research in the field of *Fintech*. Stocks are not independent of each other. But, existing studies ignore the relations between stocks or simply utilize stock spatial dependencies based on predefined graphs. The predefined graphs may miss some potential relations and are not suitable for depicting the dynamic relations between stocks. To address both problems for stock recommendation, we propose the static-dynamic graph neural network (SDGNN). In SDGNN, a graph learning module is designed to learn the static and dynamic graphs. This module employs a data-driven approach which makes the model uncover potential relations between stocks. Furthermore, to enable each stock node to obtain more information from more important neighbor nodes, we develop a graph interaction module. It implements interactions between the static graph and the dynamic graph. Experiments demonstrate that our proposed model significantly outperforms the current state-of-the-art methods on two real-world stock datasets.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Social and professional topics → Economic impact**.

KEYWORDS

Graph neural networks, graph structure learning, stock recommendation, spatial-temporal graphs

ACM Reference Format:
Yanshen He, Qiutong Li, Feng Wu, and Jianliang Gao. 2022. Static-Dynamic Graph Neural Network for Stock Recommendation. In *34th International Conference on Scientific and Statistical Database Management (SSDBM 2022), July 6–8, 2022, Copenhagen, Denmark.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3538712.3538733

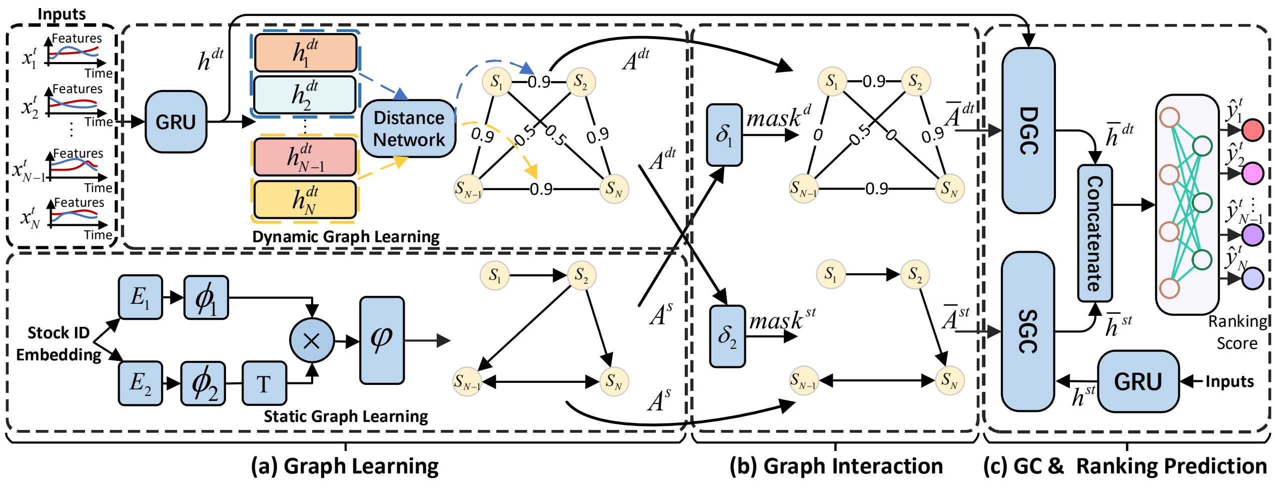
1 INTRODUCTION

With the development of financial markets, stocks are gaining more and more attention due to their potentially high returns [8]. However, attributed to the low signal-to-noise ratio of stock data, the task of stock prediction is still very challenging.

In reality, there are two types of relations between stocks. One is the static relation, which is difficult to change over time. For example, the upstream and downstream relations between stocks are static relations. Another one is the dynamic relation, which is likely to change in the short term. For example, two stocks can be linked together by a hot spot, but the hot spot generally lasts for a short period of time. When the hot spot fades, the relation between the two stocks disappears. To exploit stock relations, some researchers collect sector-industry, supplier-consumer, and ownership relations to build predefined graphs [3]. However, the structures of predefined graphs are determined by human experts. They may introduce subjective factors and miss potential relations between stocks. Besides predefined graphs are not suitable for describing dynamic stock relations. To capture dynamic stock relations, some researchers use the information of time series to dynamically infer stock relations [1]. But they fail to fuse the information of dynamic and static graphs. The dynamic graphs are insufficient to fully describe the relations between stocks. So there are two major **challenges**. One is how to capture the potential static and dynamic relations between stocks and reduce the influence of human subjective factors on graph structures. Another one is how to fuse the information of the two graphs.

To tackle the challenges mentioned above, we propose the **S**tatic-**D**ynamic **G**raph **N**eural **N**etwork (SDGNN). In SDGNN, we design a graph learning module to learn the static graph and dynamic graph. This module employs a data-driven approach to learn the two graphs so that the two graphs can automatically capture potential stock relations not discovered by human experts. Then, we design a graph interaction module to implement interactions between the static graph and dynamic graph. It makes the structural information of the two graphs converge with each other. After that, we develop the static graph convolution (SGC) module and the dynamic graph convolution (DGC) module to handle the spatial dependencies on the static graph and dynamic graph, respectively. Finally, the ranking prediction module concatenates outputs of two graph convolution modules and projects them to ranking scores. Our main contributions are as follows:

Figure 1: The overall framework of the proposed model



- We propose a novel graph framework to learn both static and dynamic graph structures by a data-driven learning approach for stock recommendation.
- We propose a novel model that captures the potential static and dynamic relations. And it achieves the interaction of the structural information of the static and dynamic graph.
- In experiments, our proposed model significantly outperforms all baselines in terms of each metric on two real-world datasets.

2 PRELIMINARY

Let a stock set $S = \{S_1, S_2, \cdots, S_N\}$ denotes N individual stocks. $X^t = \{X_1^t, X_2^t, \cdots, X_N^t\} \in \mathbb{R}^{N\times L\times U}$ is historical sequence data of S at date t. L is the length of time series and U is the number of raw features (such as opening price, closing price, etc.).

Definition 2.1 (Stock Recommendation). The ground-truth raking score is defined as return ratio $y_i^{t+1} = (p_i^{t+1} - p_i^t)/p_i^t$, where p_i^t is the closing price of stock S_i at date t. Given X^t, we aim to learn a function f which projects X^t to ranking scores and get a ranking list $\hat{y}^t = \{\hat{y}_1^t, \hat{y}_2^t, \cdots, \hat{y}_N^t\}$. It recommends top-N stocks to investors. In experiments, we choose N as 3, 5, 10, and 30.

3 METHODOLOGY

3.1 Feature Extraction

GRU. It is widely used to process sequential data. Therefore, we employ the GRU network to encode stock history sequences. Given the historical sequence of S_i ,$X_i^t = \{x_i^{t-L+1}, \cdots, x_i^t\}$, we input X_i^t into the GRU network. The formula is as follows:

$$h_i^t = GRU(h_i^{t-1}, x_i^t) \tag{1}$$

where $h_i^t \in \mathbb{R}^{U_h}$ is the hidden state of sequence X_i^t, and U_h is the number of hidden units in GRU.

3.2 Graph Learning Module

Static Graph Learning. Some researchers utilize the graph learning approach to learn the static graph of variables [10]. And they experimentally verify that the learned graph performs significantly better than predefined graphs. Inspired by them, we attempt to employ the graph learning approach to learn the static graph of stocks. The process can be described as follows:

$$M_i = \phi_i(E_i) = \tanh(\alpha E_i \theta_i) \tag{2}$$

$$A^s = \varphi(M_1 M_2{}^T) = ReLU(\tanh(\alpha(M_1 M_2{}^T))) \tag{3}$$

where A^s is the static graph; $E_i \in \mathbb{R}^{N\times U_e}$ is a stock ID embedding dictionary that is randomly initialized and learnable during training; $\theta_i \in \mathbb{R}^{U_e\times U_e}$ are learnable parameters; tanh and $ReLU$ are activation functions; and α is a hyper-parameter for controlling the saturation rate of the activation function. Because the market capitalization of any two stocks is asymmetric, static stock relations should be directed. We use Equation 3 to control the graph directed.
Dynamic Graph Learning. Predefined graphs fail to describe the dynamic relations between stocks. In order to capture the dynamic relations between stocks, we input hidden states of time series into a distance network to calculate the dynamic correlation strength. The dynamic correlation strength is used to describe the dynamic relation between stocks. The calculation details of the distance network are shown in the following equation:

$$A_{ij}^{dt} = f_d(h_i^{dt}, h_j^{dt}) = \frac{h_i^{dt} \cdot h_j^{dt}}{\|h_i^{dt}\| \cdot \|h_j^{dt}\|} \tag{4}$$

$h_i^{dt} \in \mathbb{R}^{U_h}$ is calculated by GRU and the hidden state of time series of S_i at date t; $\|\cdot\|$ is the Euclidean norm; f_d is the distance network; and A_{ij}^{dt} is the correlation strength between h_i^{dt} and h_j^{dt}. The more similar the historical time series of S_i and S_j at date t, the greater the value of A_{ij}^{dt}.

3.3 Graph Interaction Module

In order to fuse the information between the two graphs, the model implements the interaction between the two graphs. So that in the process of graph convolution, each node can obtain more information from more important neighbor nodes. The calculation details are illustrated as follows:

$$\begin{aligned} mask^d &= \delta_1(A^s) = where(A^s + (A^s)^T + I, 1, 0) \\ mask_i^{st} &= \delta_2(A_i^{dt}) = where(A_i^{dt} \geq min(topk(A_i^{dt})), 1, 0) \end{aligned} \quad (5)$$

$$\begin{aligned} \bar{A}^{dt} &= A^{dt} \odot mask^d \\ \bar{A}^{st} &= A^s \odot mask^{st} \end{aligned} \quad (6)$$

where I is an identity matrix; $where(C, 1, 0)$ is a function whose output is a matrix. If C_{ij} is greater than 0 the value at the corresponding position of the output matrix is 1, otherwise 0; $topk(\cdot)$ returns the top k largest values of a vector; k is a hyper-parameter; $min(\cdot)$ returns the minimum value in a vector; and $\odot$ denotes the element-wise production. We use Equation 5 to 6 to calculate two masks and utilize them to modify edges on the static and dynamic graph. When there is no edge between S_i and S_j in A^s, we modify the correlation strength A_{ij}^{dt} and A_{ji}^{dt} to 0. If the correlation strength from S_i to S_j in A^{dt} can not be ranked in the top k compared to the correlation strength of S_i with all their adjacent nodes, we will remove the edge from S_i to S_j in A^s.

3.4 Graph Convolution Module

Static Graph Convolution (SGC) Module. To handle the diffusion of node information on the static graph, we develop a graph convolution which is illustrated as follows:

$$\bar{h}^{st} = (\beta h^{st} + (1-\beta)\tilde{\bar{A}}^{st} h^{st})W_1 + b_1 \quad (7)$$

where $h^{st} \in \mathbb{R}^{N \times U_h}$ is the output of GRU; β is a hyper-parameter; $\tilde{\bar{A}}^{st} = (\tilde{D}^t)^{-1}(\bar{A}^{st} + I)$, and $\tilde{D}_{ii}^t = 1 + \sum_{k=0}^N \bar{A}_{ik}^{st}$; $W_1 \in \mathbb{R}^{N \times U_h}$ are learnable parameters; and $b_1 \in \mathbb{R}^{U_h}$ is learnable bias. According to Equation 7, we use β to retain part of the original hidden state of stock time series. This method can avoid stock nodes from losing their unique information in the process of information propagation.

Dynamic Graph Convolution (DGC) Module. In the dynamic graph, the value of edge is the correlation strength. To obtain the aggregation weights for dynamic graph convolution, we firstly normalize $\bar{A}^{dt}$ by each row. The normalization process is demonstrated in Equation 8 in which we use the softmax function for normalization.

$$\lambda_{ij}^t = \frac{\exp(\bar{A}_{ij}^{dt})}{\sum_{k=0}^N \exp(\bar{A}_{ik}^{dt})} \quad (8)$$

$$\bar{h}_i^{dt} = \sum_{k=0}^N \lambda_{ik}^t h_k^{dt} \quad (9)$$

λ_{ij}^t represents the aggregated weight from S_j to S_i. Secondly, according to the aggregation process shown in Equation 9, we use the aggregation weights to get the new hidden state $\bar{h}_i^{dt}$.

3.5 Ranking Prediction Module

Predicted Values. We formulate the stock recommendation as a regression task. In the ranking prediction module, we concatenate $\bar{h}^{dt}$ with $\bar{h}^{st}$ and then employ a multi-layer perception (MLP) to calculate ranking scores $\hat{y}^{t+1}$.

$$\hat{y}^{t+1} = (ReLU([\bar{h}^{dt}; \bar{h}^{st}]W_1' + b_1'))W_2' + b_2' \quad (10)$$

where $W_1' \in \mathbb{R}^{2U_h \times U_h}$ and $W_2' \in \mathbb{R}^{U_h \times 1}$ are learnable parameters of fully connected layers; and $b_1' \in \mathbb{R}^{U_h \times 1}$ and $b_2' \in \mathbb{R}$ are learnable biases.

Loss Function. Give the ground truth y_i^{t+1}, y_j^{t+1} and predicted ranking scores $\hat{y}_i^{t+1}, \hat{y}_j^{t+1}$, if $y_i^{t+1} > y_j^{t+1}$, we expect $\hat{y}_i^{t+1} > \hat{y}_j^{t+1}$. So we combine both pointwise regression loss and pairwise ranking-aware loss as follows:

$$\begin{aligned} L(\hat{y}^{t+1}, y^{t+1}) &= ||\hat{y}^{t+1} - y^{t+1}||^2 \\ &+ \mu \sum_{i=0}^N \sum_{j=0}^N max(0, -(\hat{y}_i^{t+1} - \hat{y}_j^{t+1})(y_i^{t+1} - y_j^{t+1})) \end{aligned} \quad (11)$$

where μ is a hyper-parameter to balance the two loss terms.

4 EXPERIMENTS

4.1 Datasets

CSI 300 & CSI 100 [11]: CSI 300 consists of the 300 largest and most representative stocks in the Shanghai and Shenzhen A-shares. CSI 100 consists of the largest 100 stocks selected from the CSI 300.

Features: We use last 60-day raw features of stocks which include opening prices, highest prices, lowest prices, closing prices, trading volumes, volume-weighted average prices. We use stock data of CSI 300 and CSI 100 from 01/01/2007 to 12/31/2020 and split them into the training set, validation set and test set in chronological order. we repeats every experiments for ten different runs and take the average performance of models on the test set when the model performs best on the validation set.

4.2 Experimental Setup

Evaluation Metrics. Following the previous research [11], we evaluate the prediction results by two widely used metrics: **Information Coefficient (IC)** and **Rank IC**, which are defined below:

$$IC(y^t, \hat{y}^t) = corr(y^t, \hat{y}^t) \quad (12)$$

$$Rank\ IC(y^t, \hat{y}^t) = corr(rank_{y^t}, rank_{\hat{y}^t}) \quad (13)$$

where $corr(\cdot)$ is Pearson correlation coefficient, $rank_{y^t}$ and $rank_{\hat{y}^t}$ are the rankings of labels and predictions respectively from high to low. Besides we also use another metric, **precision@N**, to evaluate the precision of top N predictions of models. Given N equals 30 and 15 labels among the top 30 predictions are positive, Precision@30 equals 0.5. For comparison with existing studies, we also set the N as 3, 5, 10, 30 to evaluate models.

Parameter Settings. Models are implemented with PyTorch framework and optimized by Adam with a learning rate of 0.0004. To obtain the optimal hyper-parameters, we use the grid search to search for parameters. For the proposed framework, the embedding size U_e of E_i is searched within (32, 40, 64); the hidden size U_h of GRUs is searched within (32, 64, 128); the saturation rate α ranges from 0.5 to 5; the parameter β is searched from 0 to 0.8; k in Equation 5 ranges from 10 to 100; and μ ranges from 0.1 to 0.5.

4.3 Baselines

- **MLP**: a 3-layer multi-layer perceptron with 512 units per layer.
- **SFM** [6]: a recurrent neural network (RNN) variant that can memorize information of time series at different frequencies.

Table 1: Evaluation results on the datasets

Model	CSI 100						CSI 300					
	IC	Rank IC	Precision@N(↑)				IC	Rank IC	Precision@N(↑)			
	(↑)	(↑)	3	5	10	30	(↑)	(↑)	3	5	10	30
MLP	0.071	0.067	56.53	56.17	55.49	53.55	0.082	0.079	57.21	57.10	56.75	55.56
SFM	0.081	0.074	57.79	56.96	55.92	53.88	0.102	0.096	59.84	58.28	57.89	56.82
GRU	0.103	0.097	59.97	58.99	58.37	55.09	0.113	0.108	59.95	59.28	58.59	57.43
LSTM	0.097	0.091	60.12	59.49	59.04	54.77	0.104	0.098	59.51	59.27	58.40	56.98
ALSTM	0.102	0.097	60.79	59.76	58.13	55.00	0.115	0.109	59.51	59.33	58.92	57.47
ALSTM+TRA	0.107	0.102	60.27	59.09	57.66	55.16	0.119	0.112	60.45	59.52	59.16	58.24
GATs	0.096	0.090	59.17	58.71	57.48	54.59	0.111	0.105	60.49	59.96	59.02	57.41
HIST	0.120	0.115	61.87	60.82	59.38	56.04	0.131	0.126	61.60	61.08	60.51	58.79
SDGNN-D	0.121	0.114	61.83	60.88	59.23	56.03	0.132	0.127	61.47	61.25	60.60	58.95
SDGNN-S	0.125	0.119	62.16	61.16	59.53	56.37	0.133	0.129	62.09	61.49	60.87	59.26
SDGNN-I	0.125	0.119	62.18	61.35	59.53	56.27	0.136	0.131	61.93	61.64	61.17	59.50
SDGNN	**0.126**	**0.120**	**62.49**	**61.41**	**59.81**	**56.39**	**0.137**	**0.132**	**62.23**	**61.76**	**61.18**	**59.56**

- **GRU** [2]: a gated recurrent unit (GRU) for stock recommendation.
- **LSTM** [5]: a long short-term memory (LSTM) network.
- **ALSTM** [4]: it utilizes temporal attention layer to extract information from previous hidden state at each step of LSTM.
- **ALSTM+TRA** [7]: an extension of ALSTM that models multiple trading patterns using temporal routing adaptor (TRA).
- **GATs** [9]: it aggregates time series embeddings computed by GRU using graph attention networks (GATs).
- **HIST** [11]: a model handles spatial dependencies between stocks based on a stock-concept bipartite graph.

4.4 Results

Overall Performance. IC and RankIC are the most important metrics in the task of stock recommendation. In Table 1, our model performs best on IC and rankIC. For example, the IC and RankIC values are 0.126, 0.120 and 0.137, 0.132 on the CSI 100 and CSI 300 datasets, respectively. Besides, our model achieves the highest Precision@N. This result verifies the superiority of our model.

Impact of Graph Learning Module. SDGNN-D is a variant of SDGNN, w/o the dynamic graph learning module and the graph interaction module. SDGNN-S is a variant of SDGNN, w/o the static graph learning module and the graph interaction module. This is because some potential relations between stocks are missing in predefined graphs. The learned graph can better capture these potential stock relations.

Impact of Graph Interaction Module. SDGNN-I is a variant of SDGNN, which combines SDGNN-D and SDGNN-S. By comparing SDGNN-S and SDGNN-I we find that they perform about the same. However, by comparing SDGNN-I and SDGNN, we find that the graph interaction module enables the model to perform better in each metric, especially Precision@3 which improves 0.31 and 0.30 in CSI 100 and CSI 300, respectively. This is because the graph interaction module is able to highlight important nodes.

5 CONCLUSION

In this paper, we propose the static-dynamic graph neural network for stock recommendation. To fully exploit relations between stocks, our proposed model learns the static and dynamic graph by the data-driven learning approach. Then to highlight important neighbor nodes on graphs, SDGNN enables the two graphs to interact. Furthermore, we develop graph convolution module to handle spatial dependencies on graphs. Experiment results show that our proposed model outperforms all the baselines on two datasets.

ACKNOWLEDGMENTS

The work is supported by the National Natural Science Foundation of China under Grant No.: 61873288.

REFERENCES

[1] Rui Cheng and Qing Li. 2021. Modeling the Momentum Spillover Effect for Stock Prediction via Attribute-Driven Graph Attention Networks. In *Proceedings of Association for the Advancement of Artificial Intelligence*, Vol. 35. 55–62.

[2] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Neural and Evolutionary Computing*. 1–9.

[3] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *Association for Computing Machinery's Transactions on Information Systems* 37, 2 (2019), 1–30.

[4] FENG FULI, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and CHUA TAT SENG. 2019. Enhancing Stock Movement Prediction with Adversarial Training. In *Proceedings of International Joint Conference on Artificial Intelligence*. 5843–5849.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[6] Hao Hu and Guo-Jun Qi. 2017. State-frequency memory recurrent neural networks. In *International Conference on Machine Learning*. 1568–1577.

[7] Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. 2021. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *Proceedings of the Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining*. 1017–1026.

[8] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. *Proceedding of Association for the Advance of Artificial Intelligence* (2021), 497–504.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *The Institute for Catastrophic Loss Reduction* (2018), 1–12.

[10] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining*. 753–763.

[11] Wentao Xu, Weiqing Liu, Lewen Wang, Yingce Xia, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. HIST: A Graph-based Framework for Stock Trend Forecasting via Mining Concept-Oriented Shared Information. *arXiv preprint arXiv:2110.13716* (2021).