Stock Predictor with Graph Laplacian-based Multi-task Learning

Jiayu He[1], Nguyen H. Tran[1], and Matloob Khushi[1,2]

[1] The University of Sydney, Australia
jihe5893@uni.sydney.edu.au, nguyen.tran@sydney.edu.au
[2] University of Suffolk, Ipswich, UK
matloob.khushi@sydney.edu.au

Abstract. The stock market is a complex network that consists of individual stocks exhibiting various financial properties and different data distribution. For stock prediction, it is natural to build separate models for each stock but also consider the complex hidden correlation among a set of stocks. We propose a federated multi-task stock predictor with financial graph Laplacian regularization (FMSP-FGL). Specifically, we first introduce a federated multi-task framework with graph Laplacian regularization to fit separate but related stock predictors simultaneously. Then, we investigate the problem of graph Laplacian learning, which represents the association of the dynamic stock. We show that the proposed optimization problem with financial Laplacian constraints captures both the inter-series correlation between each pair of stocks and the relationship within the same stock cluster, which helps improve the predictive performance. Empirical results on two popular stock indexes demonstrate that the proposed method outperforms baseline approaches. To the best of our knowledge, this is the first work to utilize the advantage of graph Laplacian in multi-task learning for financial data to predict multiple stocks in parallel.

Keywords: Federated Learning · Multi-task Learning · Graph Learning · Stock Prediction.

1 Introduction

Deep learning based stock prediction modeling has been intensively studied in recent years [17, 27]. From the point of view of market analysis, stocks exhibit highly different properties [9]. It is natural to build separate models for a group of stocks and select portfolios based on the prediction [19]. In fact, most researches build prediction models independently, which ignore the dynamic relationship among different stocks, instead of learning models simultaneously.

It is well known that the stock market is a complex network [20]. The price movement of an individual stock is correlated to its historical behavior and also highly depends on other stocks, namely the *inter-series relationship*. For example, investors often assess the performance of an individual stock by exploring the relative impact of each company in its supply-chain network. The financial market shows a hierarchical structure [18], where stocks in different groups (clusters) respond to the same economic factor in a different manner; on the other hand, stocks in the same cluster always demonstrate strong similarities when responding to information, which is referred to as the *intra-cluster relationship*.

Therefore, the main objective of stock prediction is to build prediction models for different stocks and utilize both inter-series and intra-cluster relationships among stocks. The first challenge is building prediction models to extract the temporal dependencies of time series. To address the challenge, we introduce a federated multi-task learning method [11, 25] to learn separate models for each stock simultaneously. Federated multi-task learning is able to handle the diversity of different tasks and build the best model for each task in parallel. On the other hand, in order to model both inter-series and intra-cluster relationships, we present a graph Laplacian [22] learning optimization problem with stylized financial constraints. Here, we consider an undirected weighted graph to represent the stocks network, where each stock task is treated as one node, and the edges represent the dependencies between pairs of stocks. To capture the intra-cluster relationships, we introduce a k-cluster Laplacian constraint to learn a graph with exact k connected groups. The learned graph is added to the multi-task learning framework as a regularization term to control the relationship between tasks. Then, we learn the graph and stock prediction models in an alternating fashion.

The main contribution of this work can be summarized as follows:

- We propose the federated multi-task learning with estimates from stock market data to predict a set of stocks simultaneously.
- We propose the formulation to learn a $k-$cluster graph with rank constraints to captures inter-series and intra-cluster dependencies between stocks.
- We propose the first stock prediction framework that utilizes the advantage of graph Laplacian learning.

To show the effectiveness of our proposed methods, we compare the prediction results with baseline approaches over two popular stock indexes.

2 Related works

This work is highly related to multi-task learning and graph learning. Multi-task learning algorithms have been intensively studied and have a wide range of applications, such as healthcare, wireless networks [5, 8]. In Federated multi-task learning (FMTL) [25], given datasets that are distributed over multiple clients, the goal is to learn separate models for each clients. Each model works best for each client. Stratified Model [26] is introduced in a similar manner while the objective function is minimized by the alternating direction methods of multipliers [4]. However, the critical limit of both FMTL and Stratified Model is that they are unable to solve non-convex objective functions. Recently, a unified framework for FMTL (FedU) [11] had success in solving multi-task learning applications in both convex and non-convex objective functions. It should be noted that FedU treats Laplacian in the regularization term as prior information; however, the relationships between tasks are often unknown in the field of the stock market. Thus, considering the natural diversity of stocks distribution, the objective of our work is to fit separate tasks for each stock and estimate the Laplacian simultaneously.

From the perspective of financial engineering, graph learning [20] is an increasingly important problem that carries out graph signal processing and machine learning tasks. Given the observations of each node, the goal of graph learning is to learn the optimal matrix, which represents the relationship between each pair of nodes. The graph structure is usually embedded by a Laplacian matrix [22]. Among all methods, learning graphs under smoothness assumption [16] has gained popularity. [16] assumes that observations change smoothly between connected nodes and shows that the optimal Laplacian matrix can be found by minimizing the Dirichlet energy. Motivated by that, [12] adds a variable to approximate the observations, which allows some noise in the observations. Although the above methods have achieved promising results on graph learning tasks, they are not designed to learn graphs with clustering sub-class; therefore, they can not be directly used for financial tasks. Recently, [21] proposes a graph-based clustering method to perform clustering on nodes by adding a constrained Laplacian rank to the objective function. However, the above method works in a two-stages process. Specifically, an initial estimate of graph is needed in the first stage, then it projects the initial estimate onto a rank-constrained Laplacian. A disadvantage of the two-stage process is that the final Laplacian estimate is not directly learned from the data. Furthermore, the results depend on the initial graph's estimate. Recent work [6] investigate the graph Laplacian as a candidate to capture the relationship of stocks from a probabilistic perspective. However, these methods have not been studied for stock prediction, and the advantages of graph learning in financial markets remain open for further research.

3 Methods

3.1 Problem Formulation

Suppose there are N stocks, each of them consists of $m+1$ time series. We specify one time series as the target series, while the other series are used as exogenous series. We use $\boldsymbol{Y} = \{Y^1, Y^2, \ldots, Y^N\} \in R^{N\times T}$ to denote the observations of all target series, where T is the length of window size. For example, given a dataset with N stocks, we use the closing price of each stock as the target series. Then, for each stock, we use $X = (x_1, x_2, \ldots, x_T) \in R^{m\times T}$ to denote the observations of its exogenous series, such as hand-crafted technical indicators. Given the previous values of the target series, i.e., $Y^i = (y_1^i, y_2^i, \ldots, , y_T^i)$, and the historical observations of the exogenous series, i.e., $X^i = (x_1^i, x_2^i, \ldots, x_T^i)$, the problem is to build a stock predictor, $F_i(\boldsymbol{w}_i)$, which can predict the price movement $y_{T+p}^{i,\text{binary}}$ of each stock in the next p time step:

$$\hat{y}_{T+p}^{i,\text{binary}} = F_i(\boldsymbol{w}_i | X^i, Y^i),$$

where $y_{T+p}^{i,\text{binary}} = \text{sign}(y_{T+p}^{i} - y_{T}^{i})$.

Also, we consider a undirected weighted graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}, \boldsymbol{A}\}$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the node set representing stocks, $\mathcal{E} \subseteq \{u, v \in \mathcal{V}\}$ is the edge set representing all possible connections between pairs of nodes. $\boldsymbol{A} \in R_{+}^{N\times N}$ is a symmetric weighted adjacency matrix that satisfies $A_{ii} = 0, A_{ij} > 0$ if $\{i, j\} \in \mathcal{E}$ and $A_{ij} = 0$, otherwise. The graph Laplacian matrix is defined as $\boldsymbol{L} \triangleq \boldsymbol{D} - \boldsymbol{A}$. $\boldsymbol{D} \triangleq \text{diag}(\boldsymbol{A}\boldsymbol{1})$ is the degree matrix, $\boldsymbol{1} \in R^{N}$ is the all-ones vector.

3.2 Federated Multi-task Stocks Predictor

From the point of view of market analysis, a market predictor is used to predict a market movement by extracting the dynamic historical temporal information and utilizing the related economic or public events. However, different stocks are not necessarily influenced by the same events or information. Different assets exhibit various properties and data distribution [9]. Thus, it is natural to fit personalized models to each stock. On the other hand, the price movement of an individual stock is usually related to other stocks besides its own information [15]. In this work, to fit separated models for each stock and consider the connection between stocks, we introduce the Federated Multi-task Learning with Financial Graph Laplacian Regularization entitled "Federated Multi-task Stocks Predictor with Financial Graph learning (FMSP-FGL)".

The introduced FMSP-FGL follows the module of FMTL [11, 25]. In this work, we fit separate models to each stock (node) to capture the temporal dynamics for prediction and use the Laplacian matrix as a regularization term to consider the inter-series structure by using the following formulation:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{L}} \quad & \sum_{i=1}^{N} F_i(\boldsymbol{w}_i) + \alpha \, \text{Tr}(\boldsymbol{W}^T \boldsymbol{L} \boldsymbol{W}) + \beta ||\boldsymbol{L}||_F^2 \\ \text{s.t.} \quad & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, L_{ij} = L_{ji} \leq 0, \forall i \neq j, \\ & \text{diag}(\boldsymbol{L}) = \boldsymbol{1}, \\ & \text{rank}(\boldsymbol{L}) = N - k. \end{aligned} \tag{1}$$

where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N]^T \in R^{N\times d}$ is a collective matrix whose i-th column is the weight vector for the i-th stock predictor (task), α and β are two positive regularization parameters, and k represents number of stocks clusters (groups). In addition, $\text{Tr}(\cdot)$ and $||\cdot||_F$ denote the trace and Frobenius norm of a matrix, respectively; $\text{diag}(\cdot)$ and $\text{rank}(\cdot)$ denote the diagonal vector and rank of a matrix, respectively. Here, $F_i(\boldsymbol{w}_i) = E[l(\boldsymbol{w}_i|X^i, Y^i)]$ represents the expected negative log-likelihood loss corresponding to i-th task's sample and weights. Specifically, we fit separated models for different stocks because different stocks have non-i.i.d. distribution, which means that $F_i(\boldsymbol{w}_i)$ and $F_j(\boldsymbol{w}_j)$ should be personalized when $i \neq j$.

It should be noted that we use the Laplacian to measure the hidden structure of the given weights matrix $\boldsymbol{W}$, and minimize the Dirichlet energy [16] to find the optimal Laplacian as follow,

$$\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W}) = \frac{1}{2}\sum_{i,j} A_{ij}||\boldsymbol{w}_i - \boldsymbol{w}_j||^2,$$

where $||\boldsymbol{w}_i - \boldsymbol{w}_j||^2$ is the squared Euclidean distances between two stock predictors. The learned Laplacian matrix contains the correlated structures among all stock tasks, i.e., $-L_{ij} = A_{ij} \geq 0$ measures the conditional dependency between two tasks, and $L_{ij} = A_{ij} = 0$ iif $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ are independent. The Frobenius norm is added as a penalty term in the objective function to control the edge weights of the Laplacian matrix. Moreover, we enforce a k-cluster graph Laplacian to capture the intra-cluster relationship between stocks in the same group. The above constraints of the Laplacian matrix are designed considering the stylized facts of financial tasks, as discussed in Section 3.3.

The proposed optimization problem (1) is not jointly convex in $\boldsymbol{L}$ and $\boldsymbol{W}$. Therefore, we adopt an alternating optimization method [25, 28], in which $\boldsymbol{L}$ or $\boldsymbol{W}$ is optimized alternatively with the other variable fixed until it converges. The whole scheme of the algorithm is shown in Algorithm 1. The algorithm consists of the model weights updating part (lines $2-12$) and the graph Laplacian updating part (lines 13-14). First, we fix $\boldsymbol{L}$ and solve the following optimization problem with respect to $\boldsymbol{W}$:

$$\min_{\boldsymbol{W}} \quad \sum_{i=1}^{N} F_i(\boldsymbol{w}_i) + \alpha\,\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W}). \tag{2}$$

At the second part, we solve the graph learning optimization problem with respect to $\boldsymbol{L}$ given $\boldsymbol{W}$ as:

$$\begin{aligned} \min_{\boldsymbol{L}} \quad & \alpha\,\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W}) + \beta||\boldsymbol{L}||_F^2 \\ \text{s.t.} \quad & \boldsymbol{L}\mathbf{1} = \mathbf{0}, L_{ij} = L_{ji} \leq 0, \forall i \neq j, \\ & \mathrm{diag}(\boldsymbol{L}) = \mathbf{1}, \\ & \mathrm{rank}(\boldsymbol{L}) = N - k. \end{aligned} \tag{3}$$

Specifically, when $\boldsymbol{L}$ is fixed, in each global iteration (line 1), each task performs R local updates first. The updated models are sent to their related task to perform the Laplacian regularization, which determines the correlated structure. For each local stock predictor, $F_i(\boldsymbol{w}_i)$, the goal is to learn the temporal dynamics of time series and predict future values of each stock. To achieve this goal, we adopt N modules of DARNN [24]. We choose DARNN due to its capability of selecting the most relevant exogenous input features and exploit the temporal dependencies in predicting target series. We follow the network structure of DARNN with some modifications for stock time series prediction as discussed in section 4.1. It should be noted that the proposed problem (1) reduces to a decentralized version of FedU [11] when $\boldsymbol{L}$ is fixed. However, FedU does not consider the variation of the correlated structure among different tasks.

Algorithm 1 FMSP-FGL: Federated Multi-task Stocks Predictor with Financial Graph learning

```
Input: Data $\{X^i, Y^i\}$, initial $\boldsymbol{w}_i^{(0)}$, for $i = 1, \dots, N$ tasks, initial matrix $\boldsymbol{L}$, learning rate $\mu$, regularization parameter $\alpha$, and number of local and global iteration $R$ and $T_{\max}$.
1:  for $t = 1, 2, \dots, T_{\max}$ do
2:      Step to update $\boldsymbol{W}$:
3:          for task $i \in \{1, \dots, N\}$ in parallel do
4:              initialize local model $\boldsymbol{w}_{i,0}^{(t)} := \boldsymbol{w}_i^{(t)}$
5:              for $r = 0, \dots, R-1$ do
6:                  compute mini-batch gradient $\nabla F_i(\boldsymbol{w}_{i,r}^{(t)})$
7:                  update local task weights $\boldsymbol{w}_{i,r+1}^{(t)} := \boldsymbol{w}_{i,r}^{(t)} - \mu \nabla F_i(\boldsymbol{w}_{i,r}^{(t)})$
8:              end for
9:          end for
10:         for task $i \in \{1, \dots, N\}$ in parallel do
11:             $\boldsymbol{w}_i^{(t+1)} := \boldsymbol{w}_{i,R}^{(t)} + \alpha\mu R \sum_{j \neq i} L_{ij}(\boldsymbol{w}_{i,R}^{(t)} - \boldsymbol{w}_{j,R}^{(t)})$
12:         end for
13:     Step to update Graph Laplacian $\boldsymbol{L}$:
14:         Solve the problem (3) to update $\boldsymbol{L}$
15: end for
```

When $\boldsymbol{W}$ is fixed, the problem (3) is non-convex and non-differentiable due to the constraint $\text{rank}(\boldsymbol{L}) = N - k$, which enforce a k-cluster graph Laplacian. We solve the problem by using optimization relaxation and alternating optimization methods as presented in section 3.3.

3.3 Graph Laplacian Interpretation for Financial Tasks

Learning graphs from data is a fundamental problem to capture the hidden relationship between different assets [6]. To uncover the conditional dependencies between stock prediction tasks, we propose the problem (3) with constraints considering the stylized financial facts.

The first constraint in (3), $\boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, L_{ij} = L_{ji} \leq 0, \forall i \neq j$, follows the definition of a positive semidefinite Laplacian matrix. This constraint implies that $\boldsymbol{L}$ only represents non-negative relationships, which meets the assumption that assets are always positively dependent [9]. The second constraint, $\text{diag}(\boldsymbol{L}) = \boldsymbol{1}$, controls the degree of the nodes to avoid isolated nodes. This constraint meets the fact that there is no independent asset in the financial market as all assets are treated as a complex network [20]. Thus, in our setting, the correlation between two nodes (tasks) can be measure as $-(L_{ij}/\sqrt{L_{ii}L_{jj}}) = -L_{ij}, \forall i \neq j$. In practice, the stock market is a well-defined complex network [18] where hierarchical structures can be detected, assets in different clusters have different reaction to market information. More interestingly, the intra-cluster assets demonstrate much more lead-lag correlations than inter-group components [15]. Thus, we add the rank constraint, $\mathrm{rank}(L) = N - k$, to enforce the graph to have k connected clusters to learn the intra-cluster relationship.

The proposed problem (3) is non-convex due to the rank constraint, $\mathrm{rank}(\boldsymbol{L}) = N - k$, which is that the sum of the k smallest eigenvalues of $\boldsymbol{L}$ is equal to zero, i.e., $\sum_{i=1}^{k} \sigma_i(\boldsymbol{L}) = 0$. According to Fan's theorem [13],

$$\sum_{i=1}^{k} \sigma_i(\boldsymbol{L}) = \min_{\boldsymbol{V} \in R^{N\times k}, \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}} \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{L}\boldsymbol{V}),$$

thus we have the problem (3) equivalent to the relaxed version as following,

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{V}\in R^{N\times k}} \quad & \alpha\,\mathrm{Tr}(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{L}\boldsymbol{W}) + \beta||\boldsymbol{L}||_F^2 + \lambda\,\mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{L}\boldsymbol{V}) \\ \text{s.t.} \quad & \boldsymbol{L}\mathbf{1} = \mathbf{0}, L_{ij} = L_{ji} \le 0, \forall i \ne j, \\ & \mathrm{diag}(\boldsymbol{L}) = \mathbf{1}, \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}, \end{aligned} \tag{4}$$

where λ is a regularization parameter. Note that when λ is large enough, the optimal solution will enforce the sum of k smallest eigenvalues of $\boldsymbol{L}$ to be zero. Then we rewrite the problem (4) into two convex sub-problems in an alternating fashion. When $\boldsymbol{L}$ is fixed, we have the sub-problem for $\boldsymbol{V}$:

$$\min_{\boldsymbol{F}\in R^{N\times k}, \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}} \quad \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{L}\boldsymbol{V}), \tag{5}$$

whose solution is given by the k collective eigenvectors of $\boldsymbol{L}$ corresponding to the k smallest eigenvalues according to Fan's theorem [13].

For a fixed $\boldsymbol{V}$, we have the following sup-problem for $\boldsymbol{L}$:

$$\begin{aligned} \min_{\boldsymbol{L}} \quad & \alpha\,\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W}) + \beta||\boldsymbol{L}||_F^2 + \lambda\,\mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{L}\boldsymbol{V}) \\ \text{s.t.} \quad & \boldsymbol{L}\mathbf{1} = \mathbf{0}, L_{ij} = L_{ji} \le 0, \mathrm{diag}(\boldsymbol{L}) = \mathbf{1}. \end{aligned} \tag{6}$$

We can rewrite the sub-problem (6) as a quadratic program by using *half-vec operator* and *duplication matrix* [2]. Specifically, given the vectorization of $\boldsymbol{L}$, denoted as $\mathrm{vec}(\boldsymbol{L}) \in R^{N^2}$, we introduce the *half-vec operator* $\mathrm{vech}(\cdot)$. Then $\mathrm{vech}(\boldsymbol{L}) \in R^{N(N+1)/2}$ denotes the vector obtained from $\mathrm{vec}(\boldsymbol{L})$ by eliminating all supradiagnoal elements of $\boldsymbol{L}$. Now, notice that $\boldsymbol{L}$ is symmetric, there exists a unique constant matrix $\boldsymbol{D}_N \in R^{N^2\times N(N+1)/2}$, called the duplication matrix [2], that transforms, for symmetric $\boldsymbol{L}$, $\mathrm{vech}(\boldsymbol{L})$ into $\mathrm{vec}(\boldsymbol{L})$, that is,

$$\boldsymbol{D}_n\,\mathrm{vech}(\boldsymbol{L}) = \mathrm{vec}(\boldsymbol{L}) \quad (\boldsymbol{L} = \boldsymbol{L}^T). \tag{7}$$

Now, together with the facts that, $\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{L}\boldsymbol{W}) = \mathrm{vec}(\boldsymbol{W}\boldsymbol{X}^T)^T\,\mathrm{vec}(\boldsymbol{L})$, and $||\boldsymbol{L}||_F^2 = \mathrm{vec}(\boldsymbol{L})^T\,\mathrm{vec}(\boldsymbol{L})$, we can rewrite the sup-problem (6) as:

$$\begin{aligned} \min_{\mathrm{vech}(\boldsymbol{L})} \quad & \left[\alpha\,\mathrm{vec}(\boldsymbol{W}\boldsymbol{W}^T) + \lambda\,\mathrm{vec}(\boldsymbol{V}\boldsymbol{V}^T)\right]\boldsymbol{D}_N\,\mathrm{vech}(\boldsymbol{L}) \\ & + \beta\,\mathrm{vech}(\boldsymbol{L})^T\boldsymbol{D}_N^T\boldsymbol{D}_N\,\mathrm{vech}(\boldsymbol{L}) \\ \text{s.t.} \quad & \boldsymbol{G}\,\mathrm{vech}(\boldsymbol{L}) \le h, \\ & \boldsymbol{A}\,\mathrm{vech}(\boldsymbol{L}) = b, \end{aligned} \tag{8}$$

Algorithm 2 Graph learning algorithm to solve $\boldsymbol{L}$ in (3)

```
    Input: Model weights $\boldsymbol{W}$, initial $\boldsymbol{L}, \alpha, \beta, \lambda$.
1: while not converge do
2:     Update $\boldsymbol{V}^{(l+1)}$ by solving the problem (5) fixing $\boldsymbol{L}$ at $\boldsymbol{L}^{(l)}$
3:     Update $\boldsymbol{L}^{(l+1)}$ by solving the problem (8) fixing $\boldsymbol{V}$ at $\boldsymbol{V}^{(l)}$
4: end while
    Output: Graph Laplacian $\boldsymbol{L}$.
```

where the constraints in the problem (8) handle the inequality and equality constraints in the problem (6). Problem (8) is convex and can be solved efficiently by convex programming languages [3], e.g. *CVXPY* [10]. Algorithm 2 summarized the implementation to solve the problem (3).

4 Experiments

In this section, we first introduce the stock dataset, parameters setting, and performance evaluation metrics. In order to show the effectiveness of the proposed model, we then compare FMSP-FGL with several cutting-edge approaches. Finally, we then use a step-by-step justification to demonstrate its capability of capturing the inter-series association as well as the intra-cluster relationship.

4.1 Experiment Settings

We choose two prominent stock indexes, namely DJIA, and SP500, which contain 30, 500 constituent stocks, respectively. We collect the constituent stocks' time series data from Jan-3-2017 to Dec-31-2020 from Yahoo! Finance.[3] For SP500, we collect the data of 55 stocks among 11 GICS [1] sectors. For each sector, stocks with top 5 market capitalization are selected. The frequency of the data collection is day-by-day. Each data sample contains 5 features: the opening price, highest price, lowest price, closing price, and volume (OHLCV). As previous works [14, 17], we select OHLV as well as 8 popular technical indicators[4] as the exogenous series, and use closing price as the target series. We pre-processed the collected time series data by calculating the relative percentage change of each stocks on each day with respect to its observations 5 days ago. We aim to predict the next trading day price direction (up/down) of stocks given the percentage changes of stocks over 5 consecutive days. In our experiment, the first 90% data are used for training, and the following 10% are used as the test set. We select four metrics in evaluation, i.e., Accuracy (Acc), Precision (Pre), Recall and F1 scores.

[3] In total, we collect 29 stocks from DJIA, because the stock, DOW, was listed after 2017.

[4] Technical indicators: Moving Average Convergence Divergence, Average Directional Movement Index, Awesome Oscillator, Money Flow Index, Upper Bollinger Bands, Lower Bollinger Bands, Chaikin Money Flow, On-balance Volume Mean Range.

There are two important parameters in the proposed algorithm, i.e., the graph learning regularization parameters $\{\alpha, \beta\}$ and the number of clusters k in (1). In practice, the regularization parameters setting are carried out on different ratio $\frac{\beta}{\alpha}$ to maximize the prediction performance. We fix α as 1, the ratio was determined by conducting a grid search to achieves the highest test accuracy. To determine the number of cluster k of all tasks, we use a pre-defined sector classification list, Global Industry Classification Standard (GICS) [1], as an advantage of prior domain knowledge. We denote the number of unique sectors of a dataset as $k_{\max}$, then we choose k from the finite set $k = \{1, 2, ..., k_{\max}\}$.

FMSP-FGL contains N modules of DARNNs [24], where N is the number of constituent stocks of each dataset. DARNN is used as our basic predictive module due to its capability of capturing dynamic temporal dependencies. We modify the output of DARRN as a single scalar value to perform movement prediction, with a negative log-likelihood loss function. The dimension of the hidden state and cell state are fixed as 256. All DARNN have the same window size $T = 5$, and prediction step $p = 1$. We treat the prediction of one stock as one single task. Each task shares the same hyper-parameters setting. The size of the minibatch is 128. The number of global iteration is 50, and the number of local iteration is 2. The learning rate is 0.001. Considering that λ should be large enough as we discuss in section 3, we start λ with a small value 2, and double the value if k is larger than the number of zero eigenvalues of $\boldsymbol{L}$. All tasks are trained with stochastic gradient descent. All experiments are repeated five times, and the average performance is reported. A Tesla V100 GPU is used for training. All experiments are implemented by PyTorch [23] version 1.7.

4.2 Results

To show the effectiveness of our proposed method, we compare our algorithm with the following baselines methods: Local Model (training one separate model for each stock), Federated multi-task learning framework (FedU) [11], also, we compare to FMSP with state-of-the-art graph learning methods, i,e., SigRep [12], CLR [21]. Considering the fact that FedU requires a prior graph for training, we use the sector classification list, GICS, as the prior information, where stocks in the same sector share the same weight connection. We denote Local as single task learning (STL) and the others as multi-task learning algorithms (MTL).

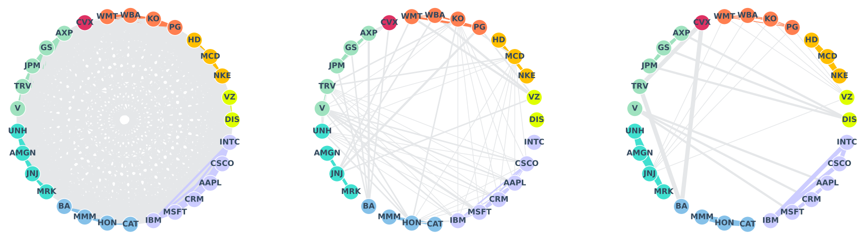
The price movement prediction results of FMSP and baseline methods are summarized in Table 1. The results show that federated multi-task learning algorithms have better performance over all evaluation metrics than single task algorithm, Local, which confirms the effectiveness of multi-task learning. The reason is that multi-task learning with Laplacian regularization can fit personalized but related models for each stock. The algorithm, FedU, with graph pre-defined setting outperforms Local, which confirms that the prior domain knowledge can be used to increase predictive performance. We conclude that federated learning algorithms can improve the overall prediction performance.

We observe that the proposed FMSP-FGL has the best performance across two datasets. The accuracy and precision of FMSP-FGL are constantly higher

Table 1: Prediction results. All models predict price direction (up/down) on the next day.

Algorithm		DJIA				SP500			
		Acc	Pre	Recall	F1	Acc	Pre	Recall	F1
STL	Local	70.61	78.91	71.76	75.16	69.05	69.45	77.96	73.46
MTL	FedU	71.61	72.88	78.64	75.65	70.82	70.78	79.84	75.04
	FMSP-SigRep	75.45	74.58	**85.50**	**79.67**	72.45	71.73	82.26	76.64
	FMSP-CLR	74.76	74.65	83.45	78.80	74.13	73.20	**83.49**	78.00
	FMSP-FGL*	**75.84**	**77.92**	79.56	78.73	**74.70**	**74.09**	82.94	**78.27**

than others, which is preferred for stock prediction. It shows evidence that the proposed graph learning algorithm can help with learning the correlated structure of different tasks thus improving overall prediction performance. One interesting point to note is that the recall rate of FMSP-FGL is lower than the other baseline, which can be seen as a trade off between accuracy and recall. In stock markets, the goal of predictor is to maintain a higher level of precision. To demonstrate the effectiveness of our graph learning algorithm, we visualize the learned graph of the percentage change of the closing price with $k = 8$ (number of sectors) in Fig. 1. The figure shows the SigRep [12] is unable to learn financial data, which leads to many possibly fake connections and fails to capture a meaningful network. The CLR [21] learns a k-cluster graph; however, the graph contains isolated nodes, namely, *DIS, INTC*, which contradicts the financial network theory in the real-world [7]. The proposed algorithm returns a k-cluster graph meaningful representation with much less fake connections. Precisely, the graph captures the prior GICS sector classification information (node-colored edges) as well as the inter-cluster connections (grey-colored edges) learned from the data without a two-stage process. Together with the prediction results, it shows the financial graph learning algorithm can help with stocks prediction.



(a) SigRep [12] (b) CLR [21] (c) Algorithm 2 (proposed)

Fig. 1: Graph learning visualization. The graph Laplacian is learned directly from the percentage change of each stocks in the DJIA dataset.

We study the sensitivity of FMSP-FGL with respect to the ratio of regularization parameters $\frac{\beta}{\alpha}$. We plot the number of edges of the learned graph and the prediction performance of the proposed algorithm versus different ratios $\frac{\beta}{\alpha}$ over the DJIA dataset in Fig. 2. As the ratio increase, we observe that both the number of edges and the prediction accuracy tends to increase. The intuitions behind this observation are as follows. When the ratio increases, the Frobenius norm of the learned Laplacian matrix in(2) tends to be small. Because we set the diagonal of Laplacian to be **1** in the constraint, the number of edges tends to increase with small values to enforce the Frobenius norm to be small. Next, we see that the proposed algorithm outperforms FedU when the number of edges of the learned graph exceeds the one of the prior graph provided by GICS. These results show that FMSP-FGL is able to learn a graph that captures the complex connection between different stocks, which contains more useful and unobserved information for prediction compared to the prior graph.

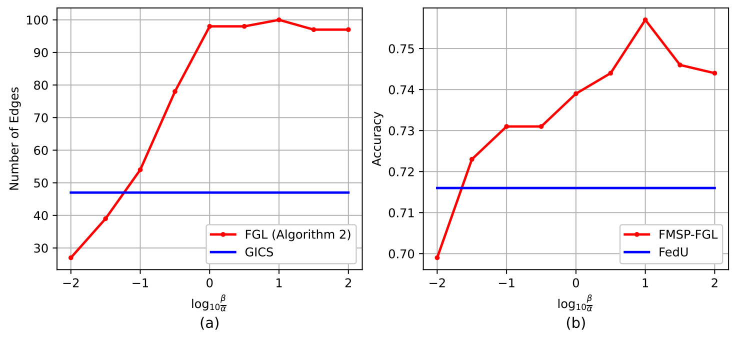


Fig. 2: (a) Number of edges in the graphs learned by FGL (Algorithm 2), and (b) prediction performance of FMSP-FGL, for different ratios $\frac{\beta}{\alpha}$ over the DJIA dataset

5 Conclusion

In this paper, we first design a multi-task stock predictor to fit separated but related tasks simultaneously. Then a graph-based clustering optimization problem with rank constraint and degree control constraint is presented to capture both the inter-series and intra-cluster relationship between stock tasks. We transform the proposed non-convex optimization problem into a relaxed convex problem and solve it alternatively. Empirical results on two stocks dataset show our methods outperform competing approaches. To our best knowledge, the proposed technique is the first to apply graph Laplacian learning with meaningful financial interpretations on multi-task stock prediction.

References

1. Gics, https://www.msci.com/our-solutions/indexes/gics. Last accessed 7 Oct 2021
2. Abadir, K.M., Magnus, J.R.: Matrix algebra, vol. 1. Cambridge University Press (2005)
3. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
4. Boyd, S., Parikh, N., Chu, E.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc (2011)
5. Brisimi, T.S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I.C., Shi, W.: Federated learning of predictive models from federated electronic health records. International journal of medical informatics **112**, 59–67 (2018)
6. Cardoso, J.V.d.M., Palomar, D.P.: Learning undirected graphs in financial markets. arXiv preprint arXiv:2005.09958 (2020)
7. Cardoso, J.V.d.M., Ying, J., Palomar, D.P.: Algorithms for learning graphs in financial markets. arXiv preprint arXiv:2012.15410 (2020)
8. Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H.V., Cui, S.: A joint learning and communications framework for federated learning over wireless networks. IEEE Transactions on Wireless Communications **20**(1), 269–283 (2020)
9. Cont, R.: Empirical properties of asset returns: stylized facts and statistical issues. Quantitative finance **1**(2), 223 (2001)
10. Diamond, S., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research **17**(83), 1–5 (2016)
11. Dinh, C.T., Vu, T.T., Tran, N.H., Dao, M.N., Zhang, H.: Fedu: A unified framework for federated multi-task learning with laplacian regularization. arXiv preprint arXiv:2102.07148 (2021)
12. Dong, X., Thanou, D., Frossard, P., Vandergheynst, P.: Learning laplacian matrix in smooth graph signal representations. IEEE Transactions on Signal Processing **64**(23), 6160–6173 (2016)
13. Fan, K.: On a theorem of weyl concerning eigenvalues of linear transformations i. Proceedings of the National Academy of Sciences of the United States of America **35**(11), 652 (1949)
14. He, J., Khushi, M., Tran, N.H., Liu, T.: Robust dual recurrent neural networks for financial time series prediction. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). pp. 747–755. SIAM (2021)
15. Hou, K.: Industry information diffusion and the lead-lag effect in stock returns. The Review of Financial Studies **20**(4), 1113–1138 (2007)
16. Kalofolias, V.: How to learn a graph from smooth signals. In: Artificial Intelligence and Statistics. pp. 920–929. PMLR (2016)
17. Li, C., Song, D., Tao, D.: Multi-task recurrent neural networks and higher-order markov random fields for stock price movement prediction: Multi-task rnn and higer-order mrfs for stock price classification. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1141–1151 (2019)
18. Mantegna, R.N.: Hierarchical structure in financial markets. The European Physical Journal B-Condensed Matter and Complex Systems **11**(1), 193–197 (1999)
19. Markowitz, H.M.: Portfolio selection. Yale university press (1968)
20. Marti, G., Nielsen, F., Bińkowski, M., Donnat, P.: A review of two decades of correlations, hierarchies, networks and clustering in financial markets. Progress in Information Geometry pp. 245–274 (2021)

21. Nie, F., Wang, X., Jordan, M.I., Huang, H.: The constrained laplacian rank algorithm for graph-based clustering. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
22. Oellermann, O.R., Schwenk, A.J.: The laplacian spectrum of graphs (1991)
23. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
24. Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., Cottrell, G.: A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971 (2017)
25. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.: Federated multi-task learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4427–4437. Curran Associates Inc. (2017)
26. Tuck, J., Barratt, S., Boyd, S.: A distributed method for fitting laplacian regularized stratified models. arXiv preprint arXiv:1904.12017 (2019)
27. Yoo, J., Soun, Y., Park, Y.c., Kang, U.: Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 2037–2045 (2021)
28. Zhang, Y., Yeung, D.Y.: A convex formulation for learning task relationships in multi-task learning. arXiv preprint arXiv:1203.3536 (2012)