Knowledge Graph Guided Simultaneous Forecasting and Network Learning for Multivariate Financial Time Series

Shibal Ibrahim[1], Wenyu Chen[1], Yada Zhu[2], Pin-Yu Chen[2], Yang Zhang[2], Rahul Mazumder[1]
[1]Massachusetts Institute of Technology, USA [2]IBM Research, USA
shibal@mit.edu, wenyu@mit.edu, yzhu@us.ibm.com, Pin-Yu.Chen@ibm.com, Yang.Zhang2@ibm.com, rahulmaz@mit.edu

ABSTRACT

Financial time series forecasting is challenging due to limited sample size, correlated samples, low signal strengths, among others. Additional information with knowledge graphs (KGs) can allow for improved prediction and decision making. In this work, we explore a framework `GregNets` for jointly learning forecasting models and correlations structures that exploit graph connectivity from knowledge graphs. We propose novel regularizers based on KG relations to guide estimation of correlation structure. We develop a pseudo-likelihood layer that can learn the error residual structure for any multivariate time-series forecasting architecture in deep learning APIs (e.g. Tensorflow/Keras). We evaluate our modelling and algorithmic proposals in small sample regimes in real-world financial markets with two types of knowledge graphs. Our empirical results demonstrate sparser connectivity structures, runtime improvements and high-quality predictions.

CCS CONCEPTS

• **Applied computing** → **Economics**; **Forecasting**; • **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

multivariate time-series, precision matrix, sparsity, knowledge graphs, graph neural networks, financial markets

ACM Reference Format:
Shibal Ibrahim[1], Wenyu Chen[1], Yada Zhu[2], Pin-Yu Chen[2], Yang Zhang[2], Rahul Mazumder[1]. 2021. Knowledge Graph Guided Simultaneous Forecasting and Network Learning for Multivariate Financial Time Series. In *KDD MLF 2021, August, 2021.* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Time series analysis is a problem central to finance and statistics [47]. In this paper, we consider the problem of modeling and forecasting in multivariate time-series panel data $\{\boldsymbol{y}_t\}_{t\in[T]}$ where, $\boldsymbol{y}_t \in \mathbb{R}^N$ is a vector of panel measurements at time $t \in [T] := \{1, \ldots, T\}$. Our study is motivated by challenging financial time-series forecasting problems (e.g., volatility, volume forecasting) characterised by relatively large panel-sizes, limited sample sizes and low-signal regimes. There is a rich body of work on multivariate time-series in statistics/econometrics for modeling and forecasting purposes mainly using linear modeling approaches. Earlier work [6, 14, 43] has noted that time-series modeling and forecasting in low-signal-regimes poses outstanding challenges. Common unregularized time-series models, e.g., vector autoregressive (VAR), may lead to overfitting and poor out-of-sample performance, therefore additional regularization methods (e.g., using sparsity) have been proposed—see for eg., [14, 43] and references therein.

A powerful parallel body of exciting recent progress includes Neural Network (NN) based approaches to model nonlinearities in time-series data e.g. recurrent neural networks (long short-term memory networks (LSTMs) [28], gated recurrent units [13]), convolutional neural networks [25], dynamic graph convolutional networks [37, 38, 49, 51] etc. Both these approaches—high-dimensional linear model based regularization techniques and those based on NN—have their respective strengths, but are yet to be studied together under one umbrella. To this end, we present a new framework that brings them together in a flexible and modular fashion.

Knowledge Graph Data. In addition to the time-varying panel data $\{y_t\}_{t\in T}$, we also make use of alternative data in the form of a Knowledge graph (KG) that contains relevant information regarding important links/associations among the time-series components. For example, in the context of finance, KGs may reflect relationships among different entities (e.g. based on industry classification, supply-chain links, products and competitors, etc). Earlier work has targeted learning of event/relation-based knowledge graph embeddings from textual data for stock prediction/explanation [12, 15]. Others have directly used financial KGs in the form of graph adjacency matrices in the context of graph convolutional networks (GCNs) [11, 22, 40]. It is worth noting that due to the low-signal to noise ratio in financial forecasting problems, additional data in the form of KGs can lead to improved stock prediction as shown by the aforementioned works. Our proposed framework `GregNets` [1] extends earlier work on KG-aided forecasting within a systematic statistical and computational framework.

Simultaneous mean and network learning. Our modeling framework has two key components: (a) the conditional mean function and (b) the correlation structure of the errors. Both (a) and (b) are trained jointly; and as we explain below, we use KG to assist the learning of (b) and/or (a). Specifically, the conditional mean is a model for the panel and is given by $\boldsymbol{f}_t := \mathbb{E}(\boldsymbol{y}_t|\mathcal{F}_{t-1})$, where $\mathcal{F}_{t-1}$ denotes the filtration corresponding to the time-series up to time point $t-1$. The function $\boldsymbol{f}_t$ is flexible and modular: for example, this can be modeled via sparse VAR models [14, 24, 29] to allow for Granger Causality interpretations or GCNs which directly take

[1]This stands for Graph Regularized Networked Time Series.

KDD MLF 2021, August, 2021,

into account KG-information. GCNs have been conventionally studied for time-series forecasting in the context of traffic networks [37, 49, 51] where the number of samples T are 2 orders of magnitude higher than the number of nodes N. We differ in that we use these models in the context of a joint mean and covariance learning framework and perform this joint learning in very limited sample regimes in the context of financial applications. We model the unexplained residuals $\boldsymbol{\epsilon}_t = \boldsymbol{y}_t - \boldsymbol{f}_t$ via a multivariate Gaussian graphical model [27]. We model the precision matrix [34] or partial correlations of the errors to be sparse—the sparsity structure is partially (not fully) informed by KGs. Our framework `GregNets` extends the framework of [6] in multiple ways: we propose faster algorithms (100×–250×) for joint training, incorporate flexible GCN models for the panel component, and use KGs to inform the partial correlation matrices.

Joint training. Jointly training the conditional mean function and error correlation structure leads to algorithmic challenges. When $\boldsymbol{f}_t$ is a sparse linear model (e.g., sparse VAR), we present a joint proximal gradient descent algorithm [8, 42] to simultaneously learn $\boldsymbol{f}_t$ and a sparse partial correlation matrix for $\boldsymbol{\epsilon}_t$ via a pseudolikeihood approach that results in improvements over the recently proposed algorithm of [6]. When $\boldsymbol{f}_t$ corresponds to a GCN or other generic temporal (graph) neural networks, we embed the partial correlation learning into our proposed pseudo-likelihood layer, append it to the standard temporal graph neural network layers, and train the time-series parameters as well as the partial correlation simultaneously via backpropagation by standard stochastic gradient descent (SGD) or its adaptive variant methods (e.g., Adam [32]).

Contributions. The key contributions of our work can be summarized as follows: (1) we propose a joint modeling and training framework `GregNets` to learn the conditional mean function for the panels and the error correlation structure for the innovations — both of these incorporate KGs in the form of structured priors. (2) Joint training leads to computational challenges. We address this by developing a pseudo-likelihood layer that incorporates the KG information and can be easily appended to any existing multivariate time-series forecasting architecture trainable with (stochastic) gradient method. (3) We demonstrate the usefulness of `GregNets` for S&P500 and S&P1500 stock volatilities time-series analysis in terms of improved prediction, reduced model complexity, faster optimization and enhanced interpretability.

2 RELATED WORK

There is an impressive literature on financial time-series modeling; we discuss work that is most closely related to the main topic of the paper. Recall that our goal is to simultaneously learn a model for $\boldsymbol{f}_t$ and a partial correlation structure for the error $\boldsymbol{\epsilon}_t$ within a joint training framework, using information from KG, in the context of financial time-series forecasting. Below we discuss related work as they pertain to the different individual components of our general model.

The partial correlation learning sub-problem is related to the rich literature on high-dimensional Gaussian graphical models [27, 34], and central to the portfolio optimization in finance [39]. Given a data matrix, common approaches to learn a corresponding ℓ_1-sparse precision matrix are the graphical lasso [23], row-by-row estimation methods [41], the pseudo-likelihood framework [9, 45]. Recently, [4, 48] propose the MTP_2 method to learn a precision matrix with nonpositive entries. For other methods on covariance matrix estimation, see [20] for a nice overview. The above methods focus on learning correlation matrices; and assume that the time-series panel component $\boldsymbol{f}_t \equiv \boldsymbol{0}$.

VAR is a popular linear modeling tool for modeling the dynamics of panel time series data, with wide applications in various domains [30, 47]. However, the number of parameters in VAR grows quadratically with the number of time series components, hence regularization is called for. Common methods to reduce the complexity of VAR models include ℓ_1-regularization [26], canonical correlation analysis, factor models [10], among others—see [43] for further discussions. Apart from VAR-based models, there are some other traditional statistical time series models in financial econometrics [5, 21, 46]. All of these models learn $\boldsymbol{f}_t$; but they do not directly learn the correlation structure of the residuals.

There is a large body of recent literature on GCNs. The fundamental GCN idea was formulated by [33] and originally studied for author citation networks; see [2, 3] amongst others. The idea has been extended to time-series modelling by combining graph convolution operation with recurrent/1D convolution architectures to capture dynamics in [37, 38, 49, 51]. These networks are interesting from ideas' standpoint and have been primarily explored in the context of traffic forecasting datasets where the number of samples is very large compared to number of nodes. Some recent financial literature uses variation of these modelling ideas in the context of stock prediction [22, 40]; see also [31] for a nice review of different applications of deep learning for stock market prediction.

Our work is most related to [6] who consider joint learning of an ℓ_1-sparse VAR model in addition to a sparse partial correlation matrix of residuals $\boldsymbol{\epsilon}_t$. However, our work differs in that (i) we use KG information to aid in the learning of the partial correlation matrix and/or a NN-based model for $\boldsymbol{f}_t$, (ii) we allow for a GNN model for $\boldsymbol{f}_t$ while [6] considers VAR models, (iii) our proposed algorithm for joint training is more general; and when specialized to the specific task considered in [6], our proposed algorithms are found to be much faster.

Our proposals are evaluated on knowledge graphs, some of which are close in spirit to those used in financial econometrics literature. Existing modelling approaches with financial knowledge graphs have explored different forms of connectivity information such as industry-sector classification [22] and first and second order relationships from text data [12, 22, 40]. We explore a new knowledge graph, proposed in [36], constructed on the basis of aggregated co-search of financial information by analysts, which is described in more detail in section 5.1. We also consider knowledge graph created using one type of financial indicator (e.g. returns) to improve prediction/correlation structure estimation for another financial indicator (e.g. volatilities).

3 GREGNETS : STATISTICAL FRAMEWORK

We present the general modeling framework for `GregNets`.

Model. Given multivariate time-series data $\{\boldsymbol{y}_t\}_1^T$, we consider the following model

$$\boldsymbol{y}_t = \boldsymbol{f}(\boldsymbol{y}_{t-1}, \cdots, \boldsymbol{y}_{t-p}) + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{-1}) \tag{1}$$

where, the conditional mean function $\boldsymbol{f}_t = \mathbb{E}(\boldsymbol{y}_t|\mathcal{F}_{t-1})$ is given by $\boldsymbol{f}(\{\boldsymbol{y}_t\}_{t-1}^{t-p}) \in \mathbb{R}^N$ which depends upon $\boldsymbol{y}_t$ for the past p time-points; and the error vectors $\{\boldsymbol{\epsilon}_t\}_1^T$ are independent and assumed to follow a multivariate Gaussian distribution with stationary covariance matrix $\Sigma = \boldsymbol{C}^{-1} \in \mathbb{R}^{N\times N}$, where $\boldsymbol{C}$ denotes the precision matrix. Both $\boldsymbol{f}(\cdot)$ and $\boldsymbol{C}$ are unknown and need to be estimated from the data under suitable structural constraints. In addition to the data $\{\boldsymbol{y}_t\}$, we will also be using knowledge graphs to inform the estimation of $\boldsymbol{f}$ and $\boldsymbol{C}$, as discussed below.

Illustration: We present some examples of the different components of (1) that can be addressed by `GregNets`. The conditional mean function $\boldsymbol{f}$ can be taken to be a linear model (e.g. VAR) or some nonlinear NN-based models (e.g. LSTMs, GCNs). We will also use KGs to incorporate prior information into some of these models. At the same time, we learn the precision matrix $\boldsymbol{C}$ through the partial correlation $\boldsymbol{\rho}$ under a pseudo-likelihood framework (See Section 4.2). This can avoid lack of scalability issues arising from the likelihood method and is more amenable to joint training with the deep learning APIs (e.g. Tensorflow [1]). Further details are presented in Section 4.

As an illustration, we can consider the target variable $\boldsymbol{y}_t$ to be stock returns, stock volatilities, trading volume or bond yields of N different companies over a certain time horizon[2].

KG based regularization: A key challenge in learning model (1) lies in suitably constraining the number of parameters associated with $\boldsymbol{f}$ and $\boldsymbol{C}$, within our joint learning framework. To this end, we use the structure of KGs to aid in the learning of $\boldsymbol{C}$. KGs can also be used to constrain $\boldsymbol{f}$ when it is taken to be a GCN. Figure 1 shows a schematic picture of using KG within our joint learning framework. Specifically, given a knowledge graph $\boldsymbol{G}$, we extract a sparse connectivity information matrix $\mathcal{E}$ from $\boldsymbol{G}$ (Section 4.3.1), and create masking matrices based on $\mathcal{E}$ and $\boldsymbol{G}$ (Section 4.3.2). The masking matrices will be used in some regularizer $\Omega_{\boldsymbol{\rho}}$, which further guides the learning of $\boldsymbol{\rho}$. In Section 4.3.3, we also discuss how to extend this to using multiple KGs. At the same time, the KG can also be used as an input for some graph neural network architectures to guide the learning of the time series part $\boldsymbol{f}$. In the next section, we will provide details of the components of `GregNets` and show the details of using the KG information.

4 LEARNING THE MODEL COMPONENTS

A principal challenge in learning model (1) lies in its joint training. Here, we discuss the details of the individual components: the mean function $\boldsymbol{f}$ (Section 4.1), the long-term contemporaneous residual correlation structure $\boldsymbol{C}$ (Section 4.2), KG-based prior regularization (Section 4.3). The joint training algorithm is discussed in Section 4.4.2.

[2]As discussed in Section 5, we need to adjust these raw time-series values to remove common systematic risks associated with either the whole market or a certain market sectors [6, 19], before appealing to model (1).

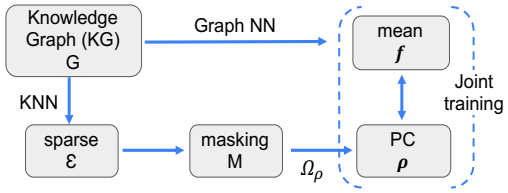

Figure 1: The use of knowledge graph in our model framework. Here, PC denotes partial correlation matrix $\boldsymbol{\rho}$ corresponding to $\boldsymbol{C}$ (see Section 4.2).

4.1 Learning the conditional mean function $\boldsymbol{f}$

We discuss two major classes of the conditional mean function $\boldsymbol{f}$: the sparse VAR model; and the NN models (e.g. LSTMs, GCNs and their variants).

Sparse Vector Autoregression (VAR). Vector autoregression models the function as a linear combination of all the time series at different lags upto p. This can be written as:

$$\boldsymbol{f}(\boldsymbol{y}_{t-1}, \cdots, \boldsymbol{y}_{t-p}) = \boldsymbol{A}^{(1)}\boldsymbol{y}_{t-1} + \boldsymbol{A}^{(2)}\boldsymbol{y}_{t-2} + \cdots + \boldsymbol{A}^{(p)}\boldsymbol{y}_{t-p} \tag{2}$$

The number of parameters in the VAR models are of order n^2p and typically the parameter tensor $\mathcal{A} := [\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(p)}] \in \mathbb{R}^{N\times N\times p}$ is penalized with either ridge or lasso regularizers. If a good pre-estimator of the tensor is available, then the regularizer can be modified to the adaptive lasso [27] which takes the form: $\Omega_{\boldsymbol{f}}(\boldsymbol{f}) = \lambda_{\mathcal{A}}\|\mathcal{B} \odot \mathcal{A}\|_1$, where $\mathcal{B}$ is the element-wise reciprocal of the pre-estimator $\hat{\mathcal{A}}$ and $\odot$ is the element-wise multiplication operation. The model complexity of the forecasting component $\boldsymbol{f}$ grows quickly as we consider higher time lags in $\boldsymbol{f}$ with VAR models. This curse of high-dimensionality, coupled with limited sample sizes may lead to severe overfitting in high-dimensional VARs (even if one imposes ℓ_1-based sparsity). Knowledge Graphs can help reduce the large model complexity in VARs in a transparent way with the use of graph NN-based approaches, as we discuss next.

Graph Convolutional Networks: The knowledge graph information can be encoded as adjacency matrices in a different class of models which are referred to as GCNs [33]. In a special case for time-series forecasting (single-layered GCN with a linear function on the graph Laplacian), this family nicely restricts the large number of parameters associated with higher-order VARs. GCNs process past time-samples of multivariate time-series with graph operations, e.g. graph convolutions, before *feeding* them into a linear model of significantly reduced model complexity. This model compression aspect of graph-based linear models over high-dimensional VARs appears to lead to improved forecasting performance in financial prediction tasks with large panels and limited data.

A single-layered GCN still falls under the linear model class. We write the conditional mean function $\boldsymbol{f}$ for GCN as $\boldsymbol{f}_t = \hat{\boldsymbol{G}}\boldsymbol{X}_t\boldsymbol{w}$, where $\boldsymbol{X}_t = [\boldsymbol{y}_{t-1}, \cdots, \boldsymbol{y}_{t-p}] \in \mathbb{R}^{N\times p}$ is the input data, $\hat{\boldsymbol{G}} = \hat{\boldsymbol{D}}^{1/2}(\boldsymbol{G}+\boldsymbol{I})\hat{\boldsymbol{D}}^{1/2}$ is the normalized graph matrix apriori defined based on the knowledge graph, $\hat{D}_{ii} = \sum_j(\boldsymbol{G}+\boldsymbol{I})_{ij}$, and $\boldsymbol{w} \in \mathbb{R}^p$ is the weight vector to be estimated. We impose ℓ_2 regularization on $\boldsymbol{w}$, i.e. $\Omega_{\boldsymbol{f}}(\boldsymbol{f}) = \lambda_{\boldsymbol{w}}\|\boldsymbol{w}\|_2^2$. A single-layered GCN is a highly structured instance of a high-dimensional VAR model; the model

complexity for the forecasting component reduces from N^2p for VAR to p in GCN because of parameter sharing and availability of additional (weighted) graph connectivity. The limited data sample scenarios greatly benefit from this model compression as long as the knowledge graph has informative connectivity structure.

Higher-order Graph Convolutional Network. It has been shown in literature that higher-order GCNs improve the predictive performance in many classification tasks [2, 3]. These networks learn from powers of adjacency matrix by combining multiple graph scales. We consider the N-GCN model given in [2] that feeds each power of the adjacency matrix into parallel GCN layers and processes the outputs through a fully connected layer. Both GCN and N-GCN models can be stacked to form two-layered networks with an intermediate non-linear activation function as done by [2, 33]. Note that we adapt the GCN and N-GCN architectures to cater to time-series forecasting, where we use only the previous time-steps as covariates and aggregate loss over the (time) sample dimension. Both GCN and N-GCN models have very few parameters to learn and hence they serve as good candidate models to estimate conditional mean function $\boldsymbol{f}$ in limited sample settings.

Temporal (Graph) Neural Network Models. We consider some temporal neural network architectures e.g. LSTMs, temporal GCNs which model the temporal dependence in the time-series. Multivariate LSTMs are good at capturing long short-term dependencies but lack the capacity to exploit graph-structured information and therefore tend to underperform when compared to the temporal GCNs. There are many temporal graph convolution variants. For instance, T-GCN follows graph convolution layers by gated recurrent units [51]; along the same spirit, a multitude of architectures such as dynamic graph convolutional networks [38], diffusion convolutional network [37], spatio-temporal graph convolutional networks [49] etc. model dynamic dependence with recurrent/convolutional layers with some intermediate graph convolution operations. All these models are highly over-parameterized and our empirical investigation showed they don't work well in the small sample regimes where $T \leq N$ and N is even moderate as we consider.

4.2 Learning the error precision matrix

The nonzero pattern of the precision matrix $\boldsymbol{C} := ((c_{ij}))$ is closely related to Gaussian graphical model [41], which corresponds to the nonzero edges of the partial correlation matrix. Specifically, the partial correlation ρ_{ij} between ϵ_{it} and ϵ_{jt} has the expression:

$$\rho_{ij} = \text{corr}(\epsilon_{it}, \epsilon_{jt} | \{\epsilon_{kt} : k \neq i, j\}) = -\frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}}. \tag{3}$$

The graphical lasso [23, 50] which considers an ℓ_1-regularized negative log-likelihood criteria, is a popular method to estimate $\boldsymbol{C}$ under the assumption that it is sparse. However, this leads to a semi-definite optimization problem, which can be hard to scale for large panels. Furthermore, as our goal is to jointly learn $\boldsymbol{f}$ (possibly a GCN) and $\boldsymbol{C}$, using a deep-learning API, we do not pursue this approach — instead, we use a pseudo-likelihood-based approach [9] outlined below.

For a multivariate Gaussian distribution, the conditional distribution of ϵ_{it} given $\{\epsilon_{kt} : k \neq i\}$ is given by:

$$\epsilon_{it} | \{\epsilon_{kt} : k \neq i\} \sim \mathcal{N}\left(\sum_{k \neq i} \rho_{ik} \sqrt{\frac{c_{kk}}{c_{ii}}} \epsilon_{kt}, \frac{1}{c_{ii}} \right). \tag{4}$$

The pseudo-likelihood [9] framework makes use of (4) to approximate the negative log-likelihood of $\boldsymbol{\epsilon}_t$ in (1) by the following expression (ignoring constants)

$$PL(\boldsymbol{\rho}, \boldsymbol{c}) = \sum_{t,i} \left\{ \log c_{ii} - c_{ii} \left(\epsilon_{it} - \sum_{k \neq i} \rho_{ik} \sqrt{\frac{c_{kk}}{c_{ii}}} \epsilon_{kt} \right)^2 \right\}. \tag{5}$$

To learn a sparse $\boldsymbol{\rho}$ (or sparse $\boldsymbol{C}$), one can use a sparsity inducing penalty on $\boldsymbol{\rho}$ [27]. Additionally, as we discuss in Section 4.3 we can also use KG-based information to guide sparsity pattern discovery in $\boldsymbol{\rho}$.

Following [6, 45], we consider a slight reformulation of (5) leading to the weighted loss function

$$\ell(\boldsymbol{\rho}, \boldsymbol{f}, \boldsymbol{c}; \boldsymbol{y}_{t-p}^t) = \sum_{i=1}^{N} w_i \left(\epsilon_{it} - \sum_{h=1, h \neq i}^{N} \rho_{ih} \sqrt{\frac{c_{hh}}{c_{ii}}} \epsilon_{ht} \right)^2, \tag{6}$$

where w_i's are nonnegative weights, and $\boldsymbol{c} \in \mathbb{R}^N$ denotes the diagonal vector of $\boldsymbol{C}$. When the weights w_i are taken to be equal to c_{ii}, this loss (6) is equivalent to the pseudo-likelihood function (5) when c_{ii}'s are fixed. Therefore, we provide a statistical interpretation for (6) as an extension of pseudo-likelihood loss. Following [6], we will use $w_i = 1/N$ for our loss function. With the weighted loss (6), we reduce the learning of $\boldsymbol{C}$ to the learning of $\{\boldsymbol{\rho}, \boldsymbol{c}\}$. However, since we learn $\boldsymbol{\rho}, \boldsymbol{c}$ (and $\boldsymbol{f}$) at the same time, the optimal solution of $\{c_{ii}\}$'s to this formulation does not necessarily maximize the original pseudo-likelihood loss (5), because of the missing logarithmic terms in (6). However, according to (4), c_{ii}^{-1} is the conditional variance of ϵ_{it} given $\boldsymbol{\epsilon}_{-it}$. We make use of this relationship in Algorithm 1 while updating the parameter c_{ii}'s.

4.3 Using knowledge graph information

In a nutshell, KGs provide us useful alternative connectivity/similarity information across time-series components. This can help in reducing the number of parameters in model (1) in the form of sparsity-priors on the components $\boldsymbol{C}$ and $\boldsymbol{f}$ (when $\boldsymbol{f}$ is a GCN). This allows us to obtain good forecasting for the financial time-series applications discussed in our experiments especially for a large number of panels with limited training data.

4.3.1 Nearest neighbors based KGs. Given a KG[3], we create a symmetric weight matrix $\boldsymbol{G}$ so that each entry G_{ij} measures the strength of similarity between i and j in the knowledge graph. However, the weight matrix $\boldsymbol{G}$ given by the knowledge graph is not always sparse. We use a "K-nearest neighbor" (KNN)-scheme to extract a sparse connectivity structure $\mathcal{E}$ from the dense graph. Specifically, given a pre-specified neighborhood sparsity-level K, for any company i, we can define its neighborhood $N_K(i)$, as those companies j's that have top K weights G_{ij} among $j \neq i$. We denote by

$$\mathcal{E} = \{(i, j) : i \in N_K(j) \text{ or } j \in N_K(i)\} \subseteq [N] \times [N]$$

[3]E.g., the KG can denote the symmetric matrix containing the co-searched companies' (on EDGAR) fractions—with high values implying higher pairwise similarity.

the set of edges obtained by these K-nearest neighbors induced by the graph G. In our experimental section (Section 5), we apply nearest neighbor method to extract important connectivity from two different graphs—the EDGAR cosearch KG and the returns partial correlation, and demonstrate the gains of using KGs.

In passing we note that there may be other ways to extract a sparse-matrix $\mathcal{E}$ from $\boldsymbol{G}$. For example, different clustering techniques can help identify the clusters of companies, and $\mathcal{E}$ can be defined as the set of edges joined by the companies within the same cluster. In our experience, these methods were found to be outperformed by the KNN method.

4.3.2 Masking matrices. After we extract the sparse connectivity structure $\mathcal{E}$ from $\boldsymbol{G}$, we define two types of masking matrices (hard/soft) as follows. The hard masking matrix is defined as

$$M_{ij}^{\text{hard}} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E} \\ \infty, & \text{otherwise} \end{cases}. \tag{7}$$

If $G_{\max}$ denotes the maximum entry of $\boldsymbol{G}$, we define the soft masking matrix $\boldsymbol{M}^{\text{soft}}$ as follows

$$M_{ij}^{\text{soft}} = \begin{cases} \frac{G_{\max}}{G_{ij}}, & \text{if } (i,j) \in \mathcal{E} \\ \infty, & \text{otherwise} \end{cases}. \tag{8}$$

These masking matrices — $\boldsymbol{M} \in \{\boldsymbol{M}^{\text{hard}}, \boldsymbol{M}^{\text{soft}}\}$ — are used to impose modified regularization penalty on $\boldsymbol{\rho}$. As the masking entry becomes larger, the penalty imposed on the corresponding entry of $\boldsymbol{\rho}$ increases. A large entry essentially zeros out the corresponding element in $\boldsymbol{\rho}$, reducing model complexity. With these additional masking weights, we re-define our new masked regularizers as follows:

$$\begin{aligned} &\text{Lasso:} && \Omega_{\boldsymbol{\rho}}(\boldsymbol{\rho}) = \lambda_\rho \left\| \boldsymbol{M} \odot \boldsymbol{\rho} \right\|_1 \\ &\text{Adaptive Lasso:} && \Omega_{\boldsymbol{\rho}}(\boldsymbol{\rho}) = \lambda_\rho \left\| \boldsymbol{M} \odot \boldsymbol{r} \odot \boldsymbol{\rho} \right\|_1, \end{aligned} \tag{9}$$

where $\boldsymbol{r}$ is the element-wise reciprocal of the prior knowledge/pre-estimator of $\boldsymbol{\rho}$.

In our experience, the masked regularizers were found to be very promising candidates from the perspective of learnability in small sample settings. The model complexity for partial correlation only grows as $O(KN)$ as opposed to $O(N^2)$ without any knowledge graph. This reduced model complexity for $\boldsymbol{\rho}$ becomes even more crucial when the conditional mean function $\boldsymbol{f}$ is over-parameterized. The reduced parameter space for $\boldsymbol{C}$ leads to improvements in predictive performance and sparsity of the network structure induced by $\boldsymbol{C}$—as we consistently observe in our empirical results.

4.3.3 Using multiple knowledge graphs. Sometimes we may have multiple KGs encoding complementary side-information on the panel components. We present a simple but useful strategy to extend `GregNets` to handle multiple knowledge graphs. For simplicity, we assume that we have weight matrices $\boldsymbol{G}^{(1)}$ and $\boldsymbol{G}^{(2)}$ from two knowledge graphs. Let $\mathcal{E}^{(k)}$ be the sparse structure extracted from $\boldsymbol{G}^{(k)}$, $k = 1, 2$, and $\tilde{\boldsymbol{G}}^{(k)}$ be the graph matrix of $\boldsymbol{G}^{(k)}$ restricted to the edges $\mathcal{E}^{(k)}$, i.e. for $(i,j) \in \mathcal{E}^{(k)}, \tilde{G}_{ij}^{(k)} = G_{ij}^{(k)}$; for $(i,j) \notin \mathcal{E}^{(k)}$, $\tilde{G}_{ij}^{(k)} = 0$. For a given weight $\alpha \in (0,1)$, we define the new graph matrix $\boldsymbol{G}$ as the convex combination of the restricted matrices, i.e. $\boldsymbol{G} = (1-\alpha)\tilde{\boldsymbol{G}}^{(1)} + \alpha\tilde{\boldsymbol{G}}^{(2)}$, and we define the sparse structure $\mathcal{E}$ as the union of $\mathcal{E}^{(1)}$ and $\mathcal{E}^{(2)}$. With the new graph $\boldsymbol{G}$ and the sparse structure $\mathcal{E}$, we can create the soft and hard masking matrices and their corresponding masked regularizers induced by the new graph. Note that hard masking matrix only looks at the union of the individual components; and remains the same for all $\alpha \in (0,1)$. The soft masking matrix considers a weighted combination of the constituent graphs, and changes with α. In the experiments section (Section 5), we demonstrate that the combined EDGAR cosearch KG and return PC graph outperforms either of them individually.

4.4 Joint Training

Finally, we present the joint training algorithm. This is given by the following optimization problem:

$$\min_{\boldsymbol{\rho}, \boldsymbol{f}, \boldsymbol{c}} \; L(\boldsymbol{\rho}, \boldsymbol{f}; \boldsymbol{c}) := \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{\rho}, \boldsymbol{f}; \boldsymbol{c}, \boldsymbol{y}_{t-p}^t) + \Omega_f(\boldsymbol{f}) + \Omega_\rho(\boldsymbol{\rho}) \tag{10}$$

where, the optimization variables are $(\boldsymbol{\rho}, \boldsymbol{f}, \boldsymbol{c})$, with

$$\begin{aligned} \ell(\boldsymbol{\rho}, \boldsymbol{f}, \boldsymbol{c}; \boldsymbol{y}_{t-p}^t) = \sum_{i=1}^{N} \frac{1}{N} \Big(& y_{it} - f_i(\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_{t-p}) \\ & - \sum_{h=1, h\neq i}^{N} \rho_{ih} \sqrt{\frac{c_{hh}}{c_{ii}}} \left(y_{ht} - f_h(\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_{t-p}) \right) \Big)^2, \end{aligned} \tag{11}$$

and Ω_f, Ω_ρ are regularizers that control the model complexity of the conditional mean and the partial correlation structure. Ω_f can be an explicit regularizer (e.g., ℓ_1-penalty) as in the case of VAR model or an implicit regularization (e.g. dropout) in the context of GCNs. Ω_ρ are regularizers with/without KG-based masking that induce the sparsity structure of $\boldsymbol{\rho}$ (i.e. that of $\boldsymbol{C}$). We first outline our algorithm for joint optimization when the conditional mean function is taken to be the pth-order VAR and the partial correlation regularizers are taken to be as those in (9). We denote this model as VAR-PC (VAR with partial correlation). In Section 4.4.2, we describe our joint optimization setup when conditional mean function is any multivariate time-series forecasting model with deep learning APIs via the implementation of a pseudo-likelihood layer.

4.4.1 Proximal Gradient Descent Algorithm for optimizing VAR-PC. We develop a custom algorithm for (10) when $\boldsymbol{f}$ is a VAR model (2) parameterized by the tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N \times N \times p}$, and the corresponding $\Omega_{\boldsymbol{f}}(\boldsymbol{f})$ becomes $\Omega_{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{A}})$. In brief, our algorithm is alternately updating $\boldsymbol{w} = (\boldsymbol{\mathcal{A}}, \boldsymbol{\rho})$ by proximal gradient descent [8, 42] and a model-based update for $\boldsymbol{c}$ (in line 5 of Algorithm 1). For the latter, we make use of the fact that each c_{ii} is the reciprocal of the conditional variance of ϵ_{it} given $\boldsymbol{\epsilon}_{-it}$ (see (4); and also Section 4.2 for further details). The algorithm is detailed in Algorithm 1.

It is important to highlight here that the VAR-PC model is considered in [6] and constitutes a special case of the modeling framework without any KG input. They propose a generalized active shooting algorithm to solve the joint optimization problem with VAR forecasting component. Their optimization algorithm is not scalable for even moderate number of multivariate time-series. In fact, our experimentation with their available `nets` package[4] in R demonstrates our algorithm is 100×−250× faster than their solver in optimizing the VAR-PC problem for $N = 403$ firms in SP500 and $T = 504$

[4] https://cran.r-project.org/web/packages/nets/index.html

Algorithm 1 Algorithm for optimizing VAR-PC

Require: Learning rate γ, Initialization $\mathcal{A}^{[0]}$, $\boldsymbol{\rho}^{[0]}$, $\boldsymbol{C}^{[0]}$

1: **for** $k = 0, 1, 2 \ldots$ **do**

2: Update $\boldsymbol{w}^{[k+1]} = (\mathcal{A}^{[k+1]}, \boldsymbol{\rho}^{[k+1]})$ by solving

$$\min_{\mathcal{A}, \boldsymbol{\rho}} \frac{\gamma}{2} \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|^2 + \Omega_{\mathcal{A}}(\mathcal{A}) + \Omega_{\boldsymbol{\rho}}(\boldsymbol{\rho}),$$

where $\tilde{\boldsymbol{w}} = \boldsymbol{w}^{[k]} - \gamma \nabla_{\boldsymbol{w}} L(\mathcal{A}^{[k]}, \boldsymbol{\rho}^{[k]}, \boldsymbol{c}^{[k]})$

3: Compute $\epsilon_{it}^{[k+1]} = y_{it} - f_{it}(\boldsymbol{y}_{t-p}^{t-1}; \mathcal{A}^{[k+1]})$

4: Compute $u_{it}^{[k+1]} = \epsilon_{it}^{[k+1]} - \sum_{j \neq i} \rho_{ij}^{[k+1]} \sqrt{\frac{c_{jj}^{[k]}}{c_{ii}^{[k]}}} \epsilon_{jt}^{[k+1]}$

5: Compute $c_{ii}^{[k+1]} = 1/\mathrm{Var}(u_{it}^{[k+1]})$

6: **end for**

samples[5], and their final objective values are about $1\% - 17\%$ higher than the objective values obtained by our algorithm.

4.4.2 Joint optimization with Pseudo-Likelihood Layer in Keras (or Tensorflow). For joint learning of a general multivariate time-series NN forecasting model e.g. GCNs along with the error correlation structure, we define a pseudo-likelihood layer that has two sets of trainable parameters $\{\boldsymbol{\rho}, \boldsymbol{c}\}$ to learn the correlation structure. The layer takes the prediction $\boldsymbol{f}$, the response variable $\boldsymbol{y}$, and the knowledge graph masking matrix $\boldsymbol{M}$ and sums the weighted loss ℓ given in (11) with the knowledge-graph based regularization $\Omega(\boldsymbol{\rho})$. This loss is backpropagated to the conditional mean function $\boldsymbol{f}$ as training proceeds. However, this loss definition alone is not sufficient to arrive at the interpretable correlation matrix with backpropagation as there is a non-identifiability problem with the correct scales of $\boldsymbol{c}$ in the loss (11). We again use the fact that c_{ii} is the reciprocal of the conditional variance of ϵ_{it} given ϵ_{-it} and use this to update the vector $\boldsymbol{c}$ during backpropagation stage of the algorithm and let the deep learning API update the partial correlation via the backpropagated gradients. This ensures the interpretation of the two components isn't lost during training. This allows use of any SGD optimizer and its variants to jointly estimate both the conditional mean and the error correlation structure.

There are some caveats in using the SGD for joint optimization that need to be addressed to ensure the partial correlation structure is both symmetric and sparse at the conclusion of the optimization procedure. Without any explicit symmetry constraint on the partial correlation structure, small numerical differences in auto-differentiation for ρ_{ij} and ρ_{ji} can cause a small drift in the symmetry of the partial correlation structure. Therefore, we impose a symmetry constraint on the partial correlation elements by symmetrizing the matrix $\boldsymbol{\rho}$ after each gradient update via $\boldsymbol{\rho}^{(k)} = 0.5(\boldsymbol{\rho}^{(k)} + (\rho^{(k)})^\top)$. It is known that although SGD methods with Lasso regularizers shrink the parameters during training, they don't precisely produce sparse parameters. This means the true sparse structure of the partial correlation is not recovered. Therefore at convergence of the stochastic optimization algorithm for the joint optimization, we set the parameters for the conditional mean function f and vector $\boldsymbol{c}$ as non-trainable and apply a proximal gradient descent on the matrix $\boldsymbol{\rho}$ to recover the sparse solution. This allows us to compare the gains in the sparsity with the KG-based regularizers over the vanilla lasso regularizer for better interpretation.

[5]For one hyperparameter setting $(\lambda_{\mathcal{A}}, \lambda_{\boldsymbol{\rho}})$, our algorithm takes a few seconds to one minute, while theirs takes at least an hour

5 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of `GregNets` on a real-world financial application, and demonstrate the benefits of using knowledge graph information. In Section 5.1, we first introduce the time series data sets and the associated KGs. All the datasets used in our experiments are available in the public domain. We then introduce the experimental setup in Section 5.2 and discuss the results in Section 5.3.

5.1 Data

Stock Volatilities Time-Series. We evaluate the combination of time-series prediction with KG-information in the context of market volatilities. The particular empirical application has been studied by multiple works [6, 16–18]. We consider two different financial markets S&P500 and S&P1500 and define the daily volatility, as given in [6, 16, 44], using the daily high and low stock prices:

$$\tilde{\sigma}_{it}^2 = 0.361(\log p_{it}^{\text{high}} - \log p_{it}^{\text{low}})^2 \tag{12}$$

where p_{it}^{high} and p_{it}^{low} denote the maximum and minimum price of stock i on day t. There are some valuable insights in a large body of literature regarding the influence of common factors in the network on the sparsity level of correlation structure. In summary, [7] shows that it is necessary to remove the market-wide and sector-wide volatility factors to evaluate the idiosyncratic behavior in terms of correlation of firms. Following [6], we condition on the corresponding market index: S&P500 or S&P1500 and 9 sector indices[6]. This reduces the number of companies to 403 for S&P500 and 1,072 for S&P1500 markets. Our target variable $\boldsymbol{y}$ is the idiosyncratic volatility residuals computed via least squares adjustment.

Knowledge Graph using EDGAR Cosearch. We follow [36] to generate a KG between firms by collecting cosearch of peer firms by users of the EDGAR website (https://www.sec.gov). Analysts collectively search for financial data on economically-related or similar firms. [36] explored how the cosearch of peer firms by users of EDGAR explains a degree of similarity between firms, which is superior to standard industry classification of peer firms in the context of cross-sectional regression of monthly stock returns. This motivates us to use cosearch to extract meaningful information about graph connectivity and construct a knowledge graph. Specifically, for each pair of firms (i, j), the number of unique users searching for both firms i and j is used to define a daily co-search proportion, which we aggregate annually. Although these search peers may not reflect judgement of an individual user, they reflect the collective view of similarity between firms across all users. We evaluate the effectiveness of this KG within the context of `GregNets` when used to learn (a) $\boldsymbol{C}$ using KG-guided masked regularizers and (b) $\boldsymbol{f}$ via GCNs with KG-guided adjacency matrices.

[6]Consumer Discretionary (XLY), Consumer Staples (XLP), Energy (XLE), Financials (XLF), Health Care (XLV), Industrials (XLI), Materials (XLB), Information Technology (XLK), and Utilities (XLU).

Returns partial correlation graph as KG (PC KG):. In addition to the EDGAR co-search KG, we test our `GregNets` framework using another KG created from open-source data. This is based on a different financial time series—the stock return, defined as

$$r_{it} = p_{it}^{\text{close}}/p_{i,t-1}^{\text{close}} - 1, \tag{13}$$

where p_{it}^{close} denotes the close price of stock i on day t. Following the same procedure used for processing of volatilities, we remove the market-wide and sector-wide factors of returns to get the idiosyncratic return residuals. We then compute the Ledoit-Wolf estimator of the covariance matrix [35], take the inverse, and get the partial correlation matrix via (3). We expect the conditional correlation structure of idiosyncratic return residuals may provide some information into the structure of volatility residuals. Therefore, we consider this as our second example of KG.

5.2 Experimental Setup

Our experimental setup considers two financial markets S&P500 and S&P1500, using daily stock volatilities residuals from 2013-2016. This period was selected because EDGAR cosearch data is only publicly available till 2017 second quarter. We set 2013-2014 as the training period, 2015 as hyperparameter validation period and 2016 for final model evaluation. There are 504 days in the training period (as compared to the 403 companies for SP500 and 1,072 companies for SP1500) to evaluate our proposed methodology on small sample size regime. We evaluate our modelling approach in terms of R^2 performance and sparsity of partial correlation structure. We follow similar R^2 definition as used in [6] where the computation of R^2 uses zero-mean prediction in the denominator (as the market and sector factors have been accounted for). However, we consider the joint R^2 of the conditional mean and the partial correlation structure.

We evaluate the utility of KG-based regularizers (constructed from two different knowledge graphs, in particular EDGAR Cosearch-based KG and returns PC KG) by comparison with various regularizers that do not use additional knowledge graph connectivity information. We perform the optimization for VAR-PC using the algorithm proposed in Section 4.4.1 and in this case the baseline scenario refers to non-masked regularized VAR-PC model optimized via our own algorithm. We also evaluate the effectiveness of the joint learning framework for NN-PC with Knowledge graphs in the context of general multivariate time-series models (e.g. LSTMs, GCNs, N-GCNs, T-GCNs) with Keras using the optimization strategy outlined in section 4.1.

5.3 Discussion of Results

VAR-PC using cosearch graph. We present our results for different regularizers in Table 1 for prediction of daily volatilities' residuals defined in Section 5.1. We compare the joint learning framework with the two stage approach with VAR-ridge, followed by Ledoit-Wolf estimator for precision matrix on the residuals. We see large gains in R^2 with the joint framework over this baseline. We also compare our proposed KG-based regularizers with their vanilla lasso/adaptive lasso counterparts. We observe that KG-based regularizers outperform both in terms of improvement in R^2 and sparsity of the estimated error correlation structure.

In Fig. 2, we display the test R^2's of VAR-PC with soft masking of the cosearch graph with different number of nearest neighbors K, as well as the sparsity of the learned partial correlation $\boldsymbol{\rho}$ for S&P1500. The dashed lines correspond to VAR-PC Lasso without using any KG. We note that we start to see improvement in R^2 by using KG from $K = 75$; also, the KG-masked partial correlation structure is much sparser than the one learned with vanilla Lasso.

Moreover, in Fig. 3, we show the sparsity patterns of partial correlation along with the blocks of 9 sectors for two different

[7] $\Omega_{\boldsymbol{\rho}}(\boldsymbol{\rho})$ denotes the type of regularizers specified in (9).
[8] Two-stage algorithm with ridge penalty on VAR and Ledoit-Wolf estimator [35] applied on the residuals.

Table 1: Evaluating different KG-masked regularizers for VAR-PC in predicting daily volatilities for S&P1500.

Model	KG Mask	$\Omega_{\boldsymbol{\rho}}(\boldsymbol{\rho})$ [7]	R^2(%)	nnz($\boldsymbol{\rho}$)(%)
VAR-Ridge[8]	N/A	Ledoit-Wolf	20.97	100.0
VAR-PC	None	Lasso	38.63	7.45
	Hard		38.90	7.38
	Soft		**39.45**	**4.91**
VAR-PC	None	Adaptive Lasso	37.61	4.72
	Hard		38.05	3.37
	Soft		**38.58**	**3.14**

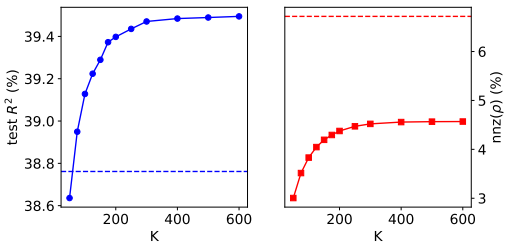

Figure 2: Test R^2 and nonzeros of partial correlation $\boldsymbol{\rho}$ with varying nearest neighbors K for VAR-PC Lasso with soft masking of the cosearch graph for S&P1500 volatilities. Dashed lines correspond to Lasso without KG information.

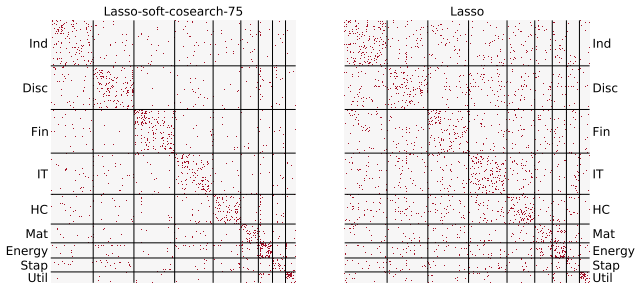

Figure 3: Sparsity patterns (binary matrix) of partial correlation $\boldsymbol{\rho}$ with sector blocks, illustrated on a submatrix formed by 40% random companies from S&P1500. Left panel: Lasso with soft masking of cosearch graph with $K = 75$; right panel: Lasso without any KG information.

algorithms, with/without use of cosearch KG information. For illustration purpose, we sample 40% random companies in S&P1500 from each sector (433 companies in total) to plot the heatmap. The figure indicates that even after removal of market and sector trends, the volatility residuals still tend to have more partial correlation connections within the sectors. In Fig. 5, we compare the test R^2 for each sector for both VAR-PC and N-GCN-PC models, and compare these R^2 under the two cases of using and not using KG information for correlation estimation. We also include the two-stage baseline model VAR-Ridge for comparison.

VAR-PC using the combined graphs. In Fig. 4, we present the test R^2 of VAR-PC using convex combination of returns PC graph and the cosearch graph, as well as some properties of the combined graph. In the plot, we take $\alpha \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$ to show the performance of the different weighted graphs with both hard and soft masking matrices, where $\alpha = 0$ corresponds to the return PC graph, while $\alpha = 1$ corresponds to the cosearch graph. For both S&P500 and S&P1500 markets, we see a flat curve for hard masking when $\alpha \in (0, 1)$, above the R^2 of both end points, which implies using the union of the sparse structure helps improve the performance. For the soft masking case, the best combination ($\alpha = 0.9$) outperforms the test R^2 from hard masking. For S&P500, the test R^2 is 35.90%, which is even better than any results using the cosearch graph alone. For S&P1500, the test R^2 using the combination with $K = 75$ is comparable to the cosearch graph masking with $K = 125$. Compared to soft masking on cosearch graph, soft masking with $\alpha = 0.9$ may have almost the same effect on the edges from the cosearch graph, but it also allows for the edges from the return PC graph (with high penalties). Therefore, the superior performance of the combined graph over the cosearch graph implies that the strong partial correlation connections are mainly from the cosearch graph, but the return PC graph contains some important weak connections that can help improve the prediction.

Multivariate (Graph) Neural Networks with partial correlation using KGs. We present `GregNets` results for the joint learning framework for various multivariate time-series models with partial correlation in Table 2. For each model, we display the two cases

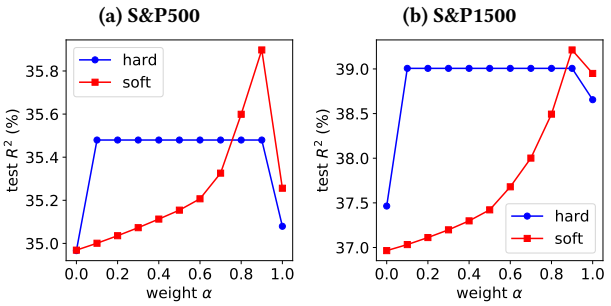

Figure 4: Test R^2 of our model using different convex combination of the cosearch graph and returns' PC graph for Lasso with soft masking and $K = 75$. $\alpha = 0$ corresponds to cosearch graph; $\alpha = 1$ corresponds to the returns' PC graph. "hard" and "soft" denote different masking strategies.

Table 2: Evaluating multiple forecasting models for joint learning with knowledge-graph based regularizers for daily volatilities for S&P1500.

Model	KG-mask	R^2(%)	nnz(ρ)(%)
VAR-PC	None	38.63	7.45
VAR-PC	Hard/Soft	**39.46**	**4.91**
LSTM-PC	None	34.96	38.95
LSTM-PC	Hard/Soft	**35.62**	**24.71**
T-GCN-PC	None	34.68	36.65
T-GCN-PC	Hard/Soft	**36.44**	**18.36**
GCN-PC	None	42.75	28.71
GCN-PC	Hard/Soft	**44.38**	**10.58**
N-GCN-PC	None	43.25	28.75
N-GCN-PC	Hard/Soft	**44.83**	**13.47**

corresponding to whether KG-based masking is used or not in defining the regularizer. We consistently see a gain in performance in terms of R^2 and sparser precision-matrix estimates with the use of KG-based regularizers. We also highlight that VAR-PC models outperform both LSTM-PC and temporal graph convolution networks (T-GCN-PC) perhaps due to over-parameterization in these recurrent NN architectures; hence these models tend to overfit severely when there are few samples to learn from. We see significant gains in predictive performance with simpler graph convolution architectures such as GCNs and N-GCNs. This confirms the effectiveness of EDGAR cosearch graph as an adjacency matrix in the context of stock volatility prediction with GCNs. We similarly observe better predictive performance by learning from graph scales with N-GCN as observed in other applications (e.g. citation datasets [2, 3]).

6 CONCLUSION

In summary, we propose a general framework `GregNets` for jointly learning multivariate time-series models (e.g. VARs, GCNs, LSTMs etc.) with their correlation structures using knowledge graphs with our proposed pseudo-likelihood layer. Based on empirical evidence, our approach leads to improved prediction, reduced model complexity, computational efficiency and interpretability.

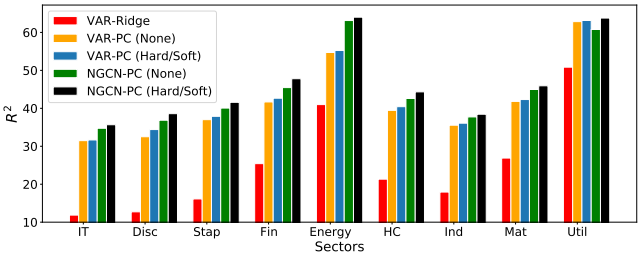

Figure 5: Sector-wise R^2 in predicting daily volatilities for S&P1500.

ACKNOWLEDGMENTS

This work is supported by the MIT-IBM Watson AI Lab. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

[2] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee. 2019. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification. In *UAI*.

[3] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing *(Proceedings of Machine Learning Research, Vol. 97)*, K. Chaudhuri and R. Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 21–29.

[4] Raj Agrawal, Uma Roy, and Caroline Uhler. 2019. Covariance matrix estimation under total positivity for portfolio selection. *arXiv preprint arXiv:1909.04222* (2019).

[5] J. Bai and S. Ng. 2008. Forecasting economic time series using targeted predictors. *Journal of Econometrics* 146, 2 (2008), 304–317.

[6] M. Barigozzi and C. Brownlees. 2019. Nets: Network estimation for time series. *Journal of Applied Econometrics* 34, 3 (2019), 347–364.

[7] M. Barigozzi and M. Hallin. 2015. Generalized dynamic factor models and volatilities: recovering the market volatility shocks. *The Econometrics Journal* 19, 1 (Nov. 2015), C33–C60.

[8] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2, 1 (2009), 183–202.

[9] J. Besag. 1975. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society: Series D (The Statistician)* 24, 3 (1975), 179–195.

[10] Rong Chen, Han Xiao, and Dan Yang. 2021. Autoregressive models for matrix-valued time series. *Journal of Econometrics* 222, 1 (2021), 539–560.

[11] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. 2018. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1655–1658.

[12] D. Cheng, F. Yang, X. Wang, Y. Zhang, and L. Zhang. 2020. Knowledge Graph-Based Event Embedding Framework for Financial Quantitative Investments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 2221–2230.

[13] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[14] Richard A Davis, Pengfei Zang, and Tian Zheng. 2016. Sparse vector autoregressive modeling. *Journal of Computational and Graphical Statistics* 25, 4 (2016), 1077–1096.

[15] Shumin Deng, Ningyu Zhang, Wen Zhang, Jiaoyan Chen, Jeff Z. Pan, and Huajun Chen. 2019. *Knowledge-Driven Stock Trend Prediction and Explanation via Temporal Convolutional Network*. Association for Computing Machinery, New York, NY, USA, 678–685. https://doi.org/10.1145/3308560.3317701

[16] F. Diebold and K. Yilmaz. 2015. Financial and Macroeconomic Connectedness: A Network Approach to Measurement and Monitoring.

[17] F. X. Diebold and K. Yilmaz. 2014. On the network topology of variance decompositions: Measuring the connectedness of financial firms. *Journal of Econometrics* 182, 1 (Sept. 2014), 119–134.

[18] R. Engle, G. Gallo, and M. Velucchi. 2012. Volatility Spillovers in East Asian Financial Markets: A Mem-Based Approach. *Review of Economics and Statistics - REV ECON STATIST* 0 (02 2012), 222–223.

[19] Jianqing Fan, Yingying Fan, and Jinchi Lv. 2008. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics* 147, 1 (2008), 186–197.

[20] J. Fan, Y. Liao, and H. Liu. 2016. An overview of the estimation of large covariance and precision matrices.

[21] J. Fan and Q. Yao. 2017. *The elements of financial econometrics*. Cambridge University Press.

[22] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T. Chua. 2019. Temporal relational ranking for stock predictions. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019).

[23] J. Friedman, T. Hastie, and R. Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9, 3 (2008), 432–441.

[24] André Fujita, Joao R Sato, Humberto M Garay-Malpartida, Rui Yamaguchi, Satoru Miyano, Mari C Sogayar, and Carlos E Ferreira. 2007. Modeling gene expression regulatory networks with the sparse vector autoregressive model. *BMC systems biology* 1, 1 (2007), 1–11.

[25] J. Gamboa. 2017. Deep Learning for Time-Series Analysis. *ArXiv* abs/1701.01887 (2017).

[26] Fang Han, Huanran Lu, and Han Liu. 2015. A direct estimation of high dimensional stationary vector autoregressions. *Journal of Machine Learning Research* (2015).

[27] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. 2015. *Statistical learning with sparsity: the lasso and generalizations*. CRC press.

[28] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[29] Douglas Holtz-Eakin, Whitney Newey, and Harvey S Rosen. 1988. Estimating vector autoregressions with panel data. *Econometrica: Journal of the econometric society* (1988), 1371–1395.

[30] Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. 2010. Estimation of a structural vector autoregression model using non-gaussianity. *Journal of Machine Learning Research* 11, 5 (2010).

[31] Weiwei Jiang. 2020. Applications of deep learning in stock market prediction: recent progress. *arXiv preprint arXiv:2003.01859* (2020).

[32] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[33] T. N. Kipf and M. Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[34] Steffen L Lauritzen. 1996. *Graphical models*. Vol. 17. Clarendon Press.

[35] Olivier Ledoit and Michael Wolf. 2004. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis* 88, 2 (2004), 365–411.

[36] C. M. C. Lee, P. Ma, and C. C. Y. Wang. 2015. Search-based peer firms: Aggregating investor perceptions through internet co-searches. *Journal of Financial Economics* 116, 2 (May 2015), 410–431.

[37] Y. Li, R. Yu, C. Shahabi, and Y. Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[38] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (Jan. 2020), 107000. https://doi.org/10.1016/j.patcog.2019.107000

[39] Harry Markowitz. 1959. Portfolio selection.

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

[41] N. Meinshausen and P. Bühlmann. 2006. High-dimensional graphs and variable selection with the lasso. *The annals of statistics* 34, 3 (2006), 1436–1462.

[42] Yu Nesterov. 2013. Gradient methods for minimizing composite functions. *Mathematical Programming* 140, 1 (2013), 125–161.

[43] William B Nicholson, Ines Wilms, Jacob Bien, and David S Matteson. 2020. High dimensional forecasting via interpretable vector autoregression. *Journal of Machine Learning Research* 21, 166 (2020), 1–52.

[44] M. Parkinson. 1980. The Extreme Value Method for Estimating the Variance of the Rate of Return. *The Journal of Business* 53, 1 (1980), 61–65.

[45] J. Peng, P. Wang, N. Zhou, and J. Zhu. 2009. Partial correlation estimation by joint sparse regression models. *J. Amer. Statist. Assoc.* 104, 486 (2009), 735–746.

[46] J. H. Stock and M. W. Watson. 2016. Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. In *Handbook of macroeconomics*. Vol. 2. Elsevier, 415–525.

[47] Ruey S Tsay. 2005. *Analysis of financial time series*. Vol. 543. John wiley & sons.

[48] Yuhao Wang, Uma Roy, and Caroline Uhler. 2020. Learning high-dimensional gaussian graphical models under total positivity without adjustment of tuning parameters. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2698–2708.

[49] B. Yu, H. Yin, and Z. Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[50] M. Yuan and Y. Lin. 2007. Model selection and estimation in the Gaussian graphical model. *Biometrika* 94, 1 (2007), 19–35.

[51] L. Zhao, Y. Song, M. Deng, and H. Li. 2018. Temporal Graph Convolutional Network for Urban Traffic Flow Prediction Method. *ArXiv* abs/1811.05320 (2018).