# Graph-based Inference and Learning with Applications in Finance


Xingyue (Stacy) Pu
Wolfson College
University of Oxford

A thesis submitted for the degree of
*Doctor of Philosophy*

Trinity 2023

# Acknowledgements

At the outset of this thesis, I would like to express my deepest and most heartfelt gratitude to my supervisor, Xiaowen Dong. His unmatched guidance, boundless patience, consistent support, and steadfast belief in my capabilities have served as the foundational pillars upon which this academic endeavour stands. My sincere appreciation extends to the distinguished members of the Oxford-Man Institute (OMI): Stephen Roberts, Stefan Zohren, Mihai Cucuringu and Anthony Ledford. Their invaluable expertise, mentorship, and constructive feedback were pivotal in the completion of this thesis. I am immensely grateful to OMI and Man AHL for providing the essential PhD funding and facilities.

I would also like to thank my viva examiners, Daniel Palomar and Ismail Ceylan, for their immense expertise, meticulous examination, and valuable feedback. I am deeply appreciative of my collaborators, Chao Zhang, Dino Sejdinovic, and Siheng Chen, whose insights and dedication have been pivotal to this work. I would also like to extend my gratitude to my peers: Zihao Zhang, Bryan Lim, Daniel Poh, Henry Kenlay, Zhi Yin Cong, Kieran Wood, Peer Nagy, Ammar Mahran, Konstantinos Ntagiantas, Ray Fung, Wee Ling Tan, Anran Hu, Yutong Lu, Branson Guo, Qi Jin, Yanzhao Yang, Li Zhang, Ning Zhang and many others. The intellectual conversations with them were not only enriching, but also paved the way for enduring friendships. Special thanks to Zhongmin Qian and Liang Zhao of OSCAR, and Yu Wang of SJTU, for their hospitality during the pandemic.

Most importantly, I offer my deepest gratitude to my mother, other family members, and all my close friends, who have steadfastly stood by my side during the most demanding moments. I dedicate this work to the memory of my father and my grandfather, who regrettably passed away during my PhD journey, but whose memories and support have guided me throughout this endeavour.

# Statement of Originality

I declare that this thesis is my own work and is submitted for the degree of Doctor of Philosophy at the University of Oxford. No part of this thesis has been submitted for any other qualification.

The main body of this thesis was co-authored with my primary supervisor Prof. Xiaowen Dong. In other collaborations, I made substantial contributions to the main content of the thesis (Chapters 2, 3, 4, 5). This thesis contains the following publications:

Chapter 2: **Pu, X.**, Chau, S.L., Dong, X., and Sejdinovic, D. *Kernel-Based Graph Learning From Smooth Signals: A Functional Viewpoint*. IEEE Transactions on Signal and Information Processing over Networks, vol. 7, pp. 192-207, 2021.

Chapter 3: **Pu, X.**, Cao, T., Zhang, X., Dong, X., and Chen, S. *Learning to Learn Graph Topologies*. 35th Conference on Neural Information Processing Systems (NeurIPS), 2021.

Chapter 4: **Pu, X.**, Zohren, S., Roberts, S., and Dong, X. *Learning to Learn Financial Networks for Optimising Momentum Strategies*, submitted to Risk.net, available at: https://doi.org/10.48550/arXiv.2308.12212.

Chapter 5: **Pu, X.**, Zhang, C., Cucuringu, M., and Dong, X. *Graph Neural Networks for Forecasting Realised Volatility with Nonlinear Spillover Effects*. International Journal of Forecasting, vol. 41, pp. 377-397, 2025.

# Abstract

Graph machine learning has emerged as a powerful tool for modelling intricate interconnections between data entities. These methods have been applied to diverse domains, from social science to molecular biology, offering unprecedented insights. However, a significant challenge is the need for an explicit and accurate graph as input, which is not always possible to obtain. This has led to the development of the discipline of Graph Learning, which focuses on inferring graph topologies from data observed on nodes of the graph.

Modern graph learning faces some challenges, such as incomplete datasets or complex topological properties that are difficult to incorporate into model specifications. To address these challenges, this thesis investigates model-based graph learning from various perspectives and proposes two novel methodologies: Kernel Graph Learning (KGL) and Learning to Learn Graph Topologies (L2G). KGL is developed from a functional perspective, offering a robust model that can handle noisy and missing node data by jointly inferring graphs and data distribution that affected by the graphs. L2G, on the other hand, employs a deep learning architecture to transform optimisation of graph adjacency matrices into a parametric functional mapping task, ensuring faster inference and precision in learning graphs with certain topological properties.

In terms of the practical implications, we focus on Quantitative Finance, where markets, institutions and assets are deeply interconnected. Although contemporary finance studies use graph representation to visualise the economic ties, the potential of graph machine learning and its ability to enhance forecasting is rarely explored. Learning a financial graph from a rich source of data also lacks exploration. This thesis aims to bridge this gap. By learning financial networks of assets from pricing data, we improve portfolio construction with better profitability. Additionally, a

customised graph neural network model, GNNHAR, is developed to examine the non-linear volatility spillover effect in equity graphs and improve realised volatility forecasting.

# Contents

# List of Figures

# Chapter 1

# Introduction

## Contents



## 1.1 Background

In the rapidly evolving domain of machine learning, graph-based methods have emerged as powerful and prominent subfield for modelling intricate relationships between data entities [30, 69, 95, 231, 215]. The importance of graph machine learning lies in its versatility and applicability across a number of domains, from social network analysis [215, 80, 199, 135] to molecular biology [116, 71], offering insights that were previously unattainable. By leveraging the inherent interconnections of data entities in a graph structure, this approach not only enhances predictive accuracy and inference capabilities but also unveils deeper and more intricate patterns and relationships, adding immense value to both academic research and real-world applications.

A significant challenge arises when applying graph machine learning: the necessity of an explicit graph as input. Such an explicit graph, which encodes relationships between data entities into edges connecting nodes, is not always readily available or explicitly given. The importance of revealing and understanding these graphs cannot be overstated, as they visualise the topological structure that captures underlying relations between data entities and provide a must-have building block for graph

machine learning to model and analyse complex systems, leading to more meaningful inferences and accurate predictions. This brings us to the heart of the matter: **Graph Learning**. This discipline is dedicated to learning a graph topology that reveals the strength of interconnections (i.e. edges) between data entities (i.e. nodes) from node data. This problem has been extensively studied in the literature from diverse perspectives, including physics [175, 102, 87], statistics [13, 79], and graph signal processing (GSP) [68, 148, 112]. Rather than methods that immediately reveal existing edges, such as illustrating a connection when two individuals are friends, graph learning adopts a data-driven strategy. In this approach, edges are not directly observed but are inferred based on node observations.

In the realm of data-driven graph learning, we face specific challenges. A fundamental assumption is that these models rely on a complete and high-quality dataset for all entities. However, practical data collection can fall short due to situations like sensor failures or budget constraints. Moreover, data observations might not be aligned, as in the example that different sensors might collect data at different timestamps.

As we address data quality, possibly through statistical imputation, it is important to understand the intertwined relationship between graphs and data. This intertwined relationship is characterised by two main directions - the data distribution being influenced by the graph structure and the graph structure being inferred from the data realisations. Existing graph learning formulations have predominantly focuses on the extraction of graph structures from underlying data observations. However, in instances of low data quality, it is imperative to recognise that the inherent structure of the graph can reciprocally inform data inference, given the joint relationship between data distribution and graph topology. This concurrent joint relationship often remains underexplored, underscoring the necessity for a novel approach that concurrently infers both of the graph and the data. This joint relationship between data and graphs presents a modelling challenge faced by data-driven graph learning. Our exploration in Chapter 2 is motivated by the desire to understand graph learning from a functional viewpoint. We aim to comprehend and model the joint relationship in a way that allows for inferences about both data and graphs at the same time. From a functional perspective, such a joint relationship can be approximated by a linear combination of kernel functions and evaluated with observed data points. We present the formulation in Chapter 2.

In addition to the input data, the literature has not extensively explored certain vital considerations regarding the potential outcomes (i.e., the learned graphs)

of graph learning frameworks. One such consideration is the enforcement of flexible topological properties on the resulting graphs that go beyond the commonly chosen sparsity. This sparsity has been explicitly integrated into the optimisation problems, as an additional convex regularisation function, of many existing methods [79, 129, 42, 185, 167, 112]. While sparsity is undoubtedly an important property, assisting in highlighting key connections in the resulting graphs, there are potentials in learning graphs with properties inspired by specific application domains (e.g., presence of communities [76], scale-free property [14]). These properties might provide significant advantages and deserve more in-depth exploration. However, they have not yet been effectively transformed into regularisers for integration into existing optimisation frameworks. This gap has prompted our exploration into deep learning methods, leveraging their universal approximation capabilities. As detailed in Chapter 3, we propose an unrolling neural network with a layer of a topological difference variational autoencoder to replace the structural regularisation functions in traditional graph learning. Through training with graphs that have the same topological characteristics, our proposed neural network is expected to learn a mapping from node observations to the underlying graph topology with intended structural property.

The lack of a readily available or useful graph representation is particularly pronounced in the application field of **Quantitative Finance**. Assets, whether they are stocks, bonds, commodities, or derivatives, do not operate in isolation. Numerous studies highlight the existence, significance, and persistence of relationships between financial assets [48, 83, 94]. They are deeply interconnected by a range of economic, geopolitical, and market-driven factors [48, 156, 164, 123]. For instance, stock prices of companies within the same industry may influence each other due to shared market risks [156, 28], and a country's currency value can impact the global commodities market [47]. While these pairwise examinations offer valuable insights, they might not fully capture the complex dynamics of the interconnected financial ecosystem. Such analyses risk overlooking the broader network effects and potential cascading influences that spread through the extensive web of asset interconnections [15].Furthermore, modern finance studies do not often represent asset interconnections through graph representations, like the graph adjacency matrix or the Laplacian matrix. These matrix representations typically serve as inputs for graph machine learning models. This oversight leads to a significant research gap, overlooking the predictive and inferential capabilities of graph machine learning within quantitative

finance. This gap presents an excellent opportunity for exploration. By creating effective methods to infer financial graphs and leveraging graph-based techniques, we can achieve deeper insights and make more informed financial decisions. In this thesis, we delve into two quantitative finance topics, demonstrating how graph machine learning, and in particular financial graph learning, can enhance portfolio construction (Chapter 4) and risk management (Chapter 5).

## 1.2 Related Work

### 1.2.1 Graph Learning

Graph learning aims to infer edge interconnections from data on every node entity and represent them in matrix forms, such as the graph adjacency matrix or the graph Laplacian matrix [67, 148]. This data-driven strategy differs from those techniques using domain knowledge that identify inherent relationships, like quantifying the number of friends a user possesses in a social network [127]. We summarise graph learning into two methodologies found in the literature: (1) pairwise metrics, (2) model-based optimisation derived from statistical, physical or signal processing models.

**Pairwise metrics** This approach calculates a distance or similarity score between pairs of node observations using specific metric functions, such as cosine similarity [209]. The score measures the extent to which two nodes are dissimilar or similar to each other. Then, some algorithms are applied to highlight the most important connections for each node, e.g. $k$-nearest neighbours ($k$-NN) [51] and the minimal spanning tree [70, 27]. The most popular $k$-NN connects a node to its $k$ closest nodes based on the smallest pairwise distances (or highest similarity scores) derived from the observations. While this method offers flexibility in choosing distance or similarity metrics, it remains heuristic. This is because the neighbourhood search relies on pairwise comparisons of node observations. Most $k$-NN variants emphasise rapid approximate search algorithms (ANN) [21, 7, 158, 65]. Recent iterations employ deep reinforcement learning to optimise search efficiency explicitly [219, 234].

**Model-based optimisation** In contrast to pairwise metrics, model-based optimisation incorporates the global properties of the node data directly into an objective function. Specifically, it formulates an regularised convex optimisation w.r.t. graph adjacency matrix or Laplacian matrix, whose objective function reflects the inductive graph-data interactions under different model assumptions [129, 192, 72, 69, 126,

112, 66, 23, 125, 232, 224]. These models are main formulated from three perspective: probabilistic and statistics, physics and signal processing [68].

Statistical and probabilistic models assume the node observations are realisations from a probability distribution over the variables, determined by the structure of a hidden graph. Such a graph structure encodes the conditional independence relationship among random variables represented by the vertices. Early works, in the context of probabilistic graphical models, solve an $\ell_1$ regularised log-likelihood maximisation to obtain a sparse precision matrix [13, 55, 145, 79]. Precision matrices are further restricted to Laplacian matrices [129, 192, 72, 59, 225], while more complex structural priors such as community structure are explored [124].

Physically motivated Models defines the graph-data interactions based on an underlying physical phenomenon or process on the graph. An example is network diffusion or cascades, where the node observations are a result of the diffusion behaviour on a graph [201, 167, 187, 185].

From a graph signal processing (GSP) perspective [67, 66, 112], graph learning involves estimating a graph adjacency or Laplacian matrix with a model assumption that graph signals mainly consists of low frequency components in the graph spectral domain, i.e. low-pass graph signals [173]. It is therefore expected that graph signals have a slow variation over the resulting graph. The variation is measured by the Laplacian quadratic term [67, 148].

The model-based optimisation formulated from these graph learning models is often convex, but non differentiable. Thus, the numeric algorithm to solve the optimisation requires specific designs, including proximal gradient descent [176], alternating direction method of multiplier (ADMM) [29], block-coordinate decent methods [79] and primal-dual splitting [112].

Many of the above models emphasise inferring graphs from data but overlook the potential of concurrently inferring both graphs and data. With a low-quality dataset, the graph structure can guide data inference due to their joint relationship. Such oversight can compromise efficiency, particularly with noisy or incomplete datasets. In Chapter 2, we introduce a model to elucidate the joint relationship between data and graph structures.

Furthermore, existing literature primarily explores sparsity for the outcome topologies of graph learning, often ignoring other potentially significant properties, such as the presence of community structure. These properties, while crucial, have been under-researched due to complexities to integrate them into an optimisation. To address this deficiency, Chapter 3 presents a deep learning framework that replace

the conventional structural regularisation methods with a neural network tailored to capture specific graph topologies.

### 1.2.2 Graph-based Inference in Finance

Graph-based methods have brought innovative viewpoints and techniques in finance to discover the complex interconnections of financial systems, thus drawing increasing attentions.

Initial endeavours applied social network visualisation and analysis techniques to uncover the complex interconnections in financial systems. These networks capture various relationships, such as capital or transaction flows, among various financial entities such as banks, firms, and countries. The primary objective of these analyses is pinpointing systemic vulnerabilities by understating how small financial turbulence can grow and spread, leading to bigger crises [183, 106, 152]; and thus prepare for possible financial downturns, often referred to as contagion risk [196]. Additionally, they shed light on the emergence of rival stock exchanges and the birth of new financial markets [36].

Graph-based techniques have also enhanced investment strategies, with two primary frameworks emerging. The first framework utilises the covariance or correlation matrix of assets to minimise the portfolio risk, aiming for diversification. This aligns with the modern portfolio theory [146, 178]. The challenge here is the precise estimation and accurate forecasting of the correlation or covariance matrix between assets [161, 163, 197, 202]. It is essential to understand that while correlation matrices provide valuable insights into asset relationships, they do not directly translate to graph topologies where edges have only non-negative values. A valid graph topology with only non-negative values is the prerequisite for many downstream learning and inference algorithms, e.g. information propagation, spectral clustering, and computation of minimal spanning trees, among others.

The second framework of graph-based investment delves into identifying nature and fundamental connections, such as industry similarities [156, 28], geographical proximity [164, 123], news co-mentions [209], and supply-demand chain [50, 151]. These studies investigate the cross-predictive potential of returns between assets based on these connections. An example is the network momentum strategy [3, 220]. This strategy operates on the premise that an asset's future returns in a financial network can be predicted from the historical returns of its connected assets, a concept termed as momentum spillover [3].

Beyond the cross-predictability of returns, identifying the nature and fundamental connections also helps explain a phenomenon known as volatility spillover. This refers to the idea that a considerable shock in a particular asset (or market) not only escalates its subsequent volatility but also influences the volatility of other interconnected assets (or markets) [10, 31, 58, 211]. Such a spillover can be understood as a form of network effect. The underlying notion is that the ensuing panic, post a substantial shock, tends to permeate through a densely interlinked network of financial assets (or markets) [117, 184, 86].

In this thesis, we focus on the two aforementioned subareas: network momentum strategy and realised volatility forecasting, demonstrating how graph machine learning can enhance existing frameworks developed for these topics.

**Network Momentum Strategies** Momentum as a risk premium in finance refers to the persistent abnormal returns demonstrated by the propensity of winning assets to continue winning and losing assets to continue losing [8, 17, 107]. The momentum premium, interestingly, is not confined to individual assets. There are instances where the performance of one asset appears to lead a similar performance of another, suggesting a propagation of momentum across assets. Such a phenomenon was first documented as *momentum spillover* by Gebhardt et al. [82].

Many studies have identified momentum spillover in the equity returns of firms linked by economic and fundamental ties. A prime example is the supply-demand chain, where firms closely linked along the chain often experience similar cash flow shocks [49, 151]. This similarity, stemming from direct trading relationships or shared market influences, contributes to cross-predictability in returns. The phenomenon of momentum spillover is not limited to a single firm or industry. It permeates across firms and industries that share similar fundamentals. This becomes particularly evident when we consider contexts that focus on the similarity of firms. For instance, momentum spillover has been observed among firms that belong to the same industry [156, 91], operate in the same market segments [50], or are located in the same region [164]. In the realm of technological innovation, firms sharing the similar patents exhibit a lead-lag effect, where the returns of technology-linked firms strongly predict the returns of the target firms [132]. Moreover, alternative news data and online search data has been used to identify firm similarities [131], and sell-side analyst coverage can serve as a strong and versatile proxy for fundamental linkages between firms [3], thereby capturing the unified phenomenon of momentum spillover. These exam-

ples underscore the pervasive nature of momentum spillover across diverse economic and fundamental linkages.

Building upon the concept of momentum spillover in pairwise relationships, further studies have broadened the scope to consider more complex network effects within a universe of stocks. In these networks, firms are not just economically and fundamentally linked in node pairs, but are part of a larger, interconnected system. This network perspective allows for a more comprehensive understanding of how momentum can propagate through a system of interconnected assets, beyond just pairwise connections. For example, Yamamoto et al. [222] modified the customer momentum strategy in [50] by applying network theory, specifically using edge betweenness centrality to calculate the importance of supplier-customer relationships.

Based on the understanding of the interconnections between assets and momentum spillover, we can derive a unique momentum signal for a target asset. This is achieved by averaging the time-series momentum characteristics of its connected assets, with the weights determined by the strength of the connections. We refer to this type of momentum signal as *Network Momentum*.

Constructing financial networks, in which nodes represent stocks and edges signify their relationships based on factors such as industry similarity, geographical proximity, news mentions, and supply-demand dynamics, is essential for effective network momentum signals. However, this endeavour often requires the use of expensive and specialised databases. Although historical pricing data presents a more cost-effective alternative, it introduces significant challenges in the realms of machine learning and finance. While the literature has explored the learning of financial networks from pricing data [34, 35, 56], there remains a marked disconnect in integrating graph learning with investment decision modelling. To bridge this gap, Chapter 4 introduces an end-to-end graph learning framework for constructing network momentum strategies.

**Realised Volatility Forecasting** Graphs are involved into realised volatility because of the discovery of volatility spillover effect, which is referred to as the phenomenon that some big shocks of a specific asset (or market) may have an influence on the volatilities of other assets (or markets). The authors [31] documented that the VIX of the U.S. market plays an important role in forecasting the volatilities of other global assets markets. [58] examined the cross-asset spillover effects from stocks, currencies, and commodities to improve the prediction of RV of crude oil. [25, 134] utilises the commonality in risk structures to improve the forecasting of

future volatility. [228] employed various financial networks to harness the spillover effect and enhance the accuracy of individual volatility forecasts.

There is a number of studies dedicated to incorporating the spillover effect into volatility modelling, e.g. BEKK-GARCH ([73]) and VAR-GARCH ([139]). In terms of modelling RV, [213] employed Vector Autoregression (VAR) to obtain the multivariate volatility forecasts for stock market indices. However, in high-dimensional scenarios, the aforementioned models may deliver poor out-of-sample forecasts due to the curse of dimensionality, as emphasised by [33]. Most recently, [228] introduced graph-based methods to capture the volatility spillover effects, and proposed a parsimonious model to augment HAR via neighbourhood aggregation on a graph that represents a financial network, denoted as Graph HAR (or GHAR). In these graphs, each asset is modelled as a node and an edge connecting two nodes encodes the existence of the spillover effect between their volatilities. GHAR leverages neighbourhood aggregation to generate a new covariate over the graph for each underlying asset and enhance the accuracy of individual volatility forecasts.

Most previous studies have focused on linear relationships across assets or markets, as seen in [58, 213, 228]. While there have been instances where nonlinear volatility spillover effects were identified and analysed [44, 212], these findings have not been incorporated into prediction models. In Chapter 5, we propose a graph neural network that illustrates how it can model nonlinear volatility spillover and enhance predictive accuracy.

## 1.3 Contributions

This thesis contributed to the technical advancement of model-based graph learning, and on the application side, using graph machine learning to achieve a better investment strategy and risk management in quantitative finance.

### 1.3.1 Graph Learning

The thesis presents significant advancements in the domain of graph learning with the introduction of two novel frameworks. These frameworks address the issues related to input data sources and output graphs accordingly, as previously discussed in Section 1.1.

In Chapter 2, we introduce **Kernel Graph Learning** (KGL). This approach revisits the graph signal processing-based graph learning model from a functional perspective. Central to this is a kernel-based model, which is designed to capture

the intricate joint relationship between graphs and graph signals. This model is able to jointly infer the missing signals and graph Laplacian matrix simultaneously. KGL stands out by seamlessly integrating covariates that describe characteristics from the node side, such as additional node features, and at the same time the observation side, like temporal dynamics. Another feature of KGL is its capability at learning a robust graph structure, even with heavily noisy and incomplete graph signals.

Addressing the underexplored topic of how to learn a graph with complex topological properties, Chapter 3 introduces the **Learning to Learn Graph Topologies** (L2G) framework. L2G stands as a deep learning architecture that reformulates optimisation-based graph learning, transforming it into a task of learning a parametric functional mapping between graph signals and their topologies. Central to this approach is a neural network model. Its forward propagation rules are derived from the iterative steps of the primal-dual-splitting algorithm originally for solving the convex optimisation of model-based graph learning. A notable feature of L2G is its integration with an effective variational auto-encoder (VAE) module, which facilitates the learning of sophisticated topological regularisers. The framework not only provides flexibility in selecting loss functions but also ensures faster inference. Our research underscores a significant enhancement in the precision of learning graph topologies, such as scale-free and community graphs, setting the stage for an in-depth exploration of graph properties.

### 1.3.2 Quantitative Finance

**Investment Strategy** We introduce the **Learning to Learn Financial Networks for Optimising Momentum Strategies** (L2GMOM) framework in Chapter 4. This framework infers financial networks from readily available pricing data, thereby eliminating the need for expensive datasets and reducing human biases. Most importantly, L2GMOM adopted an end-to-end deep learning architecture as in L2G in Chapter 3, allowing it to simultaneously optimise portfolio performance and learn the financial networks. This ensures that the financial networks are tailored to enhance portfolio performance. The proposed methods have showcased robust empirical results in terms of profitability and risk management, suggesting significant monetization potential.

**Risk Management** In Chapter 5, we delve deep into the development and application of a custom graph neural network model, **GNNHAR**, specifically tailored for examining the non-linear volatility spillover effect in the financial graph, with an aim

to more accurately forecast the realised volatility. GNN's flexible training framework allows us to adopt the task-specific loss function Quasi-Likelihood (QLIKE) which optimises for the precision of realised volatility predictions. Our findings underscore that models trained with QLIKE outstrip those trained with the conventional MSE in terms of predictive accuracy. Rigorous tests of our model's robustness were conducted across diverse market conditions, employing different data-splitting approaches, and within varied universes. In every experiment, the model exhibited a superior prediction accuracy. This chapter emphasises the immense potential of graph-based machine learning models, particularly GNN, in refining the predictive accuracy in quantitative finance tasks.

## 1.4 Thesis Outline

This thesis adopts an *Oxford integrated thesis format* [1], where the primary chapters comprise first-author papers that have either been accepted or are currently under review for publication.

Chapter 2 and Chapter 3 serve as the technical pillars, representing the major contributions to graph learning. These have been published as follows:

- Chapter 2: **Pu, X.**, Chau, S.L., Dong, X., and Sejdinovic, D. *Kernel-Based Graph Learning From Smooth Signals: A Functional Viewpoint*. IEEE Transactions on Signal and Information Processing over Networks, vol. 7, pp. 192-207, 2021.

- Chapter 3: **Pu, X.**, Cao, T., Zhang, X., Dong, X., and Chen, S. *Learning to Learn Graph Topologies*. 35th Conference on Neural Information Processing Systems (NeurIPS), 2021.

Chapter 4 and Chapter 5 dedicate to applications in quantitative finance:

- Chapter 4: **Pu, X.**, Zohren, S., Roberts, S., and Dong, X. *Learning to Learn Financial Networks for Optimising Momentum Strategies*, submitted to the 4th ACM International Conference on AI in Finance, 2023. Available at arXiv: https://arxiv.org/abs/2308.12212.

[1] For details of *integrated thesis format*, please refer to https://www.mpls.ox.ac.uk/graduate-school/information-for-postgraduate-research-students/submitting-your-thesis

- Chapter 5: **Pu, X.**, Zhang, C., Cucuringu, M., and Dong, X. *Graph Neural Networks for Forecasting Realised Volatility with Nonlinear Spillover Effects*, submitted to International Journal of Forecasting, 2023. Available at SSRN: https://ssrn.com/abstract=4375165.

# Chapter 2

# Graph Learning from a Functional Viewpoint

## Contents



## 2.1 Introduction

Modelling based on graphs has recently attracted an increasing amount of interest in machine learning and signal processing research. On the one hand, many real-world data are intrinsically graph-structured, e.g. individual preferences in social networks or environmental monitoring data from sensor networks. This makes graph-based

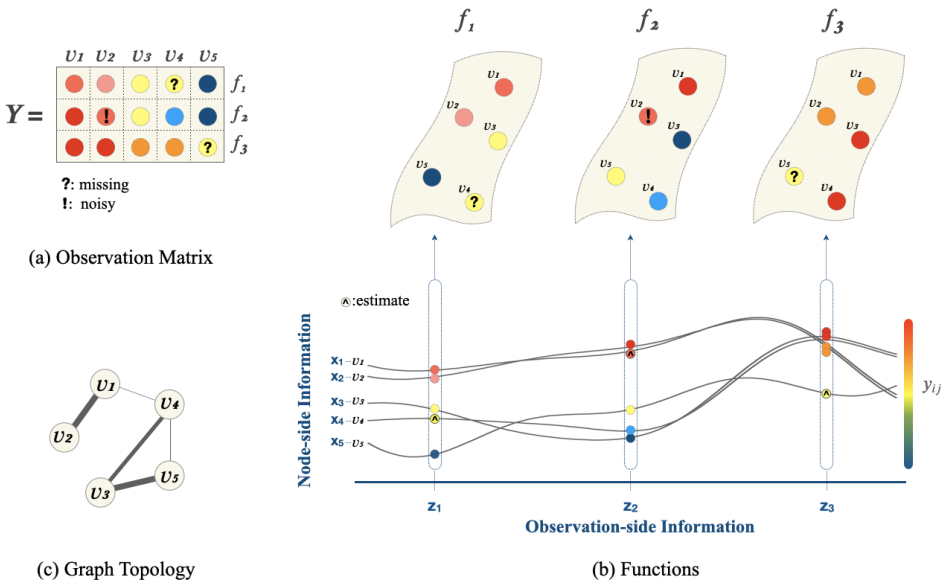



Figure 2.1: A functional viewpoint of graph learning. (a) The observation matrix with missing and noisy entries; (b) Each row of the observation matrix is modelled as samples obtained at a collection of fixed locations (considered as nodes in a graph) from an underlying function $f$ (top); Values at each node are determined by the underlying function, as well as node-side information $\mathbf{x}$ and observation-side information $\mathbf{z}$ (bottom); (c) The learned graph topology.

methods a natural approach to analysing such structured data. On the other hand, graphs are an effective modelling language for revealing relational structure in complex domains and may assist in a variety of learning tasks. For example, knowledge graphs improve the performance in semantic parsing and question answering [22]. Despite their usefulness, however, a graph is not always readily available or explicitly given. The problem of graph learning therefore concerns the construction of a topological structure among entities from a set of observations on these entities.

Methodologies to learn a graph from the structured data include naïve methods such as $k$-nearest neighbours ($k$-NN), and approaches from the literature of probabilistic graphical models (PGMs) and more recently graph signal processing (GSP) and graph neural networks (GNNs). The basic idea of $k$-NN is to connect a node to $k$ other nodes with the smallest pairwise distances in terms of the observations [7, 158, 21, 65]. In PGMs, a graph expresses the conditional dependence with edges between random variables represented by nodes [130]. The GSP literature, on the other hand, focuses on algebraic and spectral characteristics of the graph signals [191, 160, 180], which are defined as observations on a collection of nodes. The GSP-based graph learning

methods (see [148, 68] for two recent reviews) further fall into two distinct branches, i.e. those based on the diffusion processes on graphs [201, 167, 187, 185] and those based on smoothness measures of graph signals [112, 66, 113, 72, 42, 23]. Very recently, GNNs have attracted a surging interest in the machine learning community which leads to a number of approaches to graph inference [120, 77].

While many of the above methods can effectively learn a meaningful graph from observations, there is a lack of consideration of the prior information, e.g. node-side covariates, which may be available for the task at hand. Those covariates that provide valuable side information should be integrated into the graph learning framework. Taking an example of measuring temperature records in different locations in a country, where nodes represent weather stations, the latitude, longitude and altitude of each station are useful node-side information. One major benefit is to lessen the reliance of the above models on the quality of the observations. Heavily corrupted or even missing records can be predicted by such prior information, which in turn helps improve the efficiency in graph inference.

Besides the node-side prior information, the observation-side dependency is also largely ignored in the literature. One example are temperature records collected at different timestamps, which are clearly related and could largely affect the evaluation of the strength of relation between stations. Another example is that of a recommender system, where the item ratings collected from different individuals are largely affected by the social relationship between them.

To tackle the above issues, we revisit the graph signal observations from a functional viewpoint and propose a framework for learning undirected graphs by considering additional covariates on both the node- and observation-side. This allows us to capture dependency structure beyond the graph signals which leads to more effective and graph and signal recovery. More specifically, as shown in Figure 5.1, the $ij$-th entry of the graph-structured data matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$, which contains $n$ graph signals collected on $m$ nodes, can be viewed as some potentially noisy or missing observation of $f_i(j)$, i.e. the $i$-th function evaluated at $j$-th node. To model the node-side information, we introduce a covariate $\mathbf{x} \in \mathcal{X}$ that can explain the variations in a graph signal, e.g. a vector that contains the latitude, longitude and altitude of stations in the aforementioned temperature example. To model the observation-side information, we also introduce a generic covariate $\mathbf{z} \in \mathcal{Z}$. For example, $\mathbf{z}$ could be the timestamp at which the temperature record is collected. Observation-side dependency hence arises due to $f_i$ depending on $\mathbf{z}_i$. Combining the two, the function underlying the graph signals takes the form of $f_i(j) = f(\mathbf{z}_i, \mathbf{x}_j)$.

Formally, we define a function $f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ in a reproducing kernel Hilbert space (RKHS) with a product kernel $\kappa_{\otimes} = \kappa_{\mathcal{Z}} \otimes \kappa_{\mathcal{X}}$ on $\mathcal{Z} \times \mathcal{X}$, where $\otimes$ denotes the Kronecker product. The function draws a connection between graph signals and both node- and observation-side covariates. At the same time, by modifying the classical Laplacian quadratic form that is widely adopted in the literature, we propose a novel smoothness measure that takes into account the observation-side dependency introduced by $\mathbf{z} \in \mathcal{Z}$, which has a nice interpretation associated with a Kronecker product graph.

The adoption of kernel methods in our framework is motivated by the need for a flexible representation of the intricate and potentially non-linear associations between graph signals and a broad range of covariates, both on the node and observation sides. The complexity of these relationships is underscored by empirical findings in diverse data analysis. For example, research has demonstrated that geographical coordinates, including altitude and longitude, exhibit a non-linear relationship with land surface temperature (LST) [136, 170]. In particular, studies in certain regions of China have revealed that topographic factors such as elevation and slope are negatively correlated with LST, highlighting the intricacy of these relationships [170]. Furthermore, the evolving dynamics of climate change have complicated the relationship between temporal factors such as timestamps or seasons and temperature, deviating from simple linear associations [61, 208]. These findings offer valuable insights to further formulate our motivating example, where temperature, modelled as a graph signal, is integral to inferring the structure of weather station networks. Geographical factors, in this context, act as node-side covariates, and temporal factors as observation-side covariates reinforcing the necessity for kernel methods in our model to adeptly capture and represent these complex, non-linear interactions. This rationale forms the cornerstone of our decision to employ kernel methods in our analytical model.

Our key contribution is the Kernel Graph Learning (**KGL**) framework, which allows us to learn the graph Laplacian matrix $\mathbf{L}$ by jointly inferring the function $f$ and optimising the smoothness over $\mathbf{L}$ of the graph signals modelled and refined by $f$ with node- and observation-side prior information.

In addition, we provide several extensions of **KGL** for the scenario of a partially observed $\mathbf{Y}$ with known missing value positions, and that of observations without either node-side or observation-side information. The learning problem is effectively solved via a block coordinate descent algorithm, which has a theoretical guarantee of convergence. We show that **KGL** can effectively recover the groundtruth graph from

the two-side dependent data and outperform the state-of-the-art smoothness-based graph learning methods in both synthetic and real-world experiments.

In summary, the main contributions of our work are as follows:

- A novel graph-based regularisation based on a smoothness measure of graph signals with observation-side dependency;
- A graph learning framework that integrates node- and observation-side covariates from a functional viewpoint;
- An efficient method for denoising and imputing missing values in the observed graph signals as a byproduct of the graph learning framework.

### 2.1.1 Related Work

In this section, we survey the classical methods of learning a graph from a number of different perspectives. From each perspective, we highlight the most related work that considers one or more aspects of the 1) node-side information, 2) observation-side dependency, and 3) noisy and missing data.

**$k$-Nearest Neighbours Methods** The $k$-nearest neighbours ($k$-NN) connects a node to $k$ other nodes with the smallest pairwise distances in terms of the observations. It is flexible with different choices of distance metrics and yet heuristic since the neighbourhood search is based on pairwise comparison of observations on nodes. The majority of the $k$-NN variants focuses on fast approximate search algorithms (ANN) [21, 7, 158, 65] and recent variants apply deep reinforcement learning to explicitly maximise search efficiency [219, 234]. By comparison, the model-based methods, e.g. PGMs and GSP, directly integrate global properties of the observations into learning objectives.

**Probabilistic Graphical Models** In the field of PGMs, the inverse covariance matrix $\Theta$ is often regarded as an undirected graph that parameterises the joint distribution of random variables representing nodes. There is a rich literature on effective algorithms to estimate a sparse $\Theta$ from Gaussian-distributed data by solving an $\ell_1$-regularised log-likelihood maximisation [13, 55, 149, 104, 145], including the widely used graphical Lasso [79] and G-ISTA [176]. The recent state-of-the-art algorithm BigQUIC [105] scales to millions of nodes with performance guarantees. Besides computational improvements, models based on attractive Gaussian Markov Random Fields (GMRFs) [129, 192, 72, 59] further restrict the off-diagonal entries of $\Theta$ to

be non-positive, which is equivalent to learning the Laplacian matrix of the corresponding graph with non-negative edge weights. The most related extensions of the graphical Lasso were proposed in [195, 89], which simultaneously learn two dependency structures in the matrix-variate Gaussian data. While their work focuses on estimating covariance matrices, our work focuses on recovering a graph topology from data.

**Structural Equation Models** Structural equation models (SEMs) are another type of models (similar to PGMs) that is widely used to learn a directed acyclic graph (DAG) that encodes the conditional dependence of random variables [233, 85, 162]. Based on SEMs, the authors in [108] proposed a block coordinate descent algorithm to solve the joint optimisation problem of denoising the data and learning a directed graph. The joint learning framework is further extended to time series [109], where the structural vector autoregressive models (SVARMs) replace the linear SEMs to handle temporal dependency. The main difference from our work is that their denoising function is an identity mapping without side information as covariates. The work in [162] also considers the temporal dependency in learning a DAG with SVARMs, but does not consider the denoising scenario.

**Graph Signal Processing** In the context of GSP, every observation on a collection of nodes is defined as a graph signal. GSP-based graph learning models have seen an increasing interest in the literature [148, 67] and further fall into two distinct branches. The first branch assumes graph signals are outcomes of diffusion processes on graphs and reconstructs a graph from signals according to the diffusion model [201, 167, 187, 185]. The other branch constructs a graph by promoting the global smoothness of graph signals, which is defined by the Laplacian quadratic form [112, 66] or more generally via total variation [23]. Smoothness-based methods are related to GMRFs by recognising that the Laplacian quadratic form is closely related to the log-likelihood of the precision matrix defined as the graph Laplacian. Our work can be regarded as an extension to smoothness-based graph construction.

In the literature of smoothness-based GSP graph learning, the authors in [66, 23] adopt a two-step learning algorithm to learn an undirected graph while denoising graph signals. They simply assume an identity mapping between the actual graph signals and noisy observations, which is different from our work that considers side information. The most related work is proposed in [206], which uses kernel ridge regression with observation-side covariates to infer graph signals. However, their

work mainly focuses on data prediction and graph learning is only a byproduct in their approach. In Section 2.4.1, we will show, both theoretically and empirically, that their method uses a smoothness term that imprecisely incorporates the observation-side dependency in the learned graph structure, leading to an inferior performance in learning a graph.

In terms of the observation-side dependency, there exist some GSP graph learning models that consider temporal dependency in graph signals. A so-called spatiotemporal smoothness was proposed in [142, 141] to transform the graph signals using a temporally weighed difference operator. If every timestamp is equally important, the operator is equivalent to a prepossessing step to make the time series observed on each node stationary. It should be noted that there is another branch of research assuming that the temporal dependency in graph signals originates in the dynamic changes in the edges [115, 221], and therefore the problem is formulated as learning a dynamic series of graphs, which is different from the goal of our chapter.

**Graph Neural Networks** A new branch of graph learning models is developed from the perspective of GNNs. Essentially, GNNs discover the patterns in graph-structured data in a hierarchical manner [231, 215, 30]. The activations at intermediate layers, e.g. the $l$-th layer, can be interpreted as a new representation for the nodes in the embedding space that incorporates the information from a specifically defined neighbourhood of the nodes. The authors in [93, 120] thus defined the strength of connectivity between nodes $i$ and $j$ based on the pairwise similarity of their embeddings $h_i^{(l)}$ and $h_j^{(l)}$ at the $l$-th layer of the GNN architecture. The authors in [218] extended this method to construct a directed graph in the process of training a GNN that deals with time series data. The main goal of these methods is to improve the performance of node-related tasks (e.g. classification or prediction) and graph learning is only a byproduct, whose performance is often not guaranteed. The recent works in [111, 223, 41, 77] start to incorporate an additional loss for recovering graphs while training the GNNs. However, a significant limitation of most GNN-based methods is that they typically require a large volume of training data and the learned connectivity is often less explainable compared to PGM and GSP methods.

## 2.2 Preliminaries

### 2.2.1 Smoothness-Based Graph Learning in GSP

Observing a data matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$ whose $ij$-th entry $y_{ij}$ corresponds to the observation on the $j$-th node in the $i$-th graph signal, we are interested in constructing an undirected and weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$. The node set $\mathcal{V}$ represents a collection of variables, where $|\mathcal{V}| = m$. The edge set $\mathcal{E}$ represents the relational structure among them to be inferred. The structure is completely characterised by the weighted adjacency matrix $\mathbf{W}$ whose $jj'$-th entry is $w_{jj'}$. If two nodes $j$ and $j'$ are connected by an edge $e_{jj'} \in \mathcal{E}$ then $w_{jj'} > 0$, else if $e_{jj'} \notin \mathcal{E}$ then $w_{jj'} = 0$. The graph Laplacian matrix is defined as $\mathbf{L} = \text{diag}(\mathbf{W1}) - \mathbf{W}$, where $\mathbf{1}$ denotes the all-one vector. $\mathbf{L}$ and $\mathbf{W}$ are equivalent and complete representations of the graph on a given set of nodes.

In the literature of GSP, one typical approach of constructing a graph from $\mathbf{Y}$ is formulated as minimising the variation of signals on graphs as measured by the Laplacian quadratic form[1] [191, 112, 66]:

$$\min_{\mathbf{L} \in \mathcal{L}} \ \text{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top) + \lambda \Omega(\mathbf{L}) \tag{2.1}$$

where $\mathcal{L} = \{\mathbf{L} | \mathbf{L1} = 0, \mathbf{L}_{jj'} = \mathbf{L}_{j'j} \leq 0, \forall j \neq j'\}$ defines the space of valid Laplacian matrices, and $\Omega(\mathbf{L})$ is a regularisation term with a hyperparameter $\lambda > 0$. Equivalently, the problem can be formulated using the weighted adjacency matrix $\mathbf{W}$ such that

$$\min_{\mathbf{W} \in \mathcal{W}} \ \frac{1}{2} \sum_{i=1}^{n} \sum_{j,j'} w_{jj'} (y_{ij} - y_{ij'})^2 + \lambda \Omega(\mathbf{W}) \tag{2.2}$$

where $\mathcal{W} = \{\mathbf{W} | \text{diag}(\mathbf{W}) = \mathbf{1}, w_{jj'} = w_{jj'} \geq 0, \forall j \neq j'\}$ defines the space of valid weighted adjacency matrices. Popular choices of regularisation include the sum barrier $\Omega(\mathbf{L}) = ||\mathbf{L}||_F^2$ and $\Omega(\mathbf{L}) = |\text{Tr}(\mathbf{L}) - m|$ (often added as a constraint such that $\text{Tr}(\mathbf{L}) = m$) to prevent trivial solutions where all edge weights are zero and meanwhile controlling the variations of edge weights [66], or the log-barrier $\Omega(\mathbf{W}) = \mathbf{1}^\top \log(\mathbf{W1})$ to prevent isolated nodes and promote connectivity [112].

With a fixed Frobenius norm for $\mathbf{Y}$, a small value of the objective in Eq.(2.1) implies that $\mathbf{Y}$ is smooth on $\mathcal{G}$ in the sense that neighbouring nodes have similar

[1] We acknowledge that the conventional form of the Laplacian quadratic in GSP literature is $\text{Tr}(\mathbf{Y}^\top \mathbf{L} \mathbf{Y})$, where each column of $\mathbf{Y}$ corresponds to a graph signal. In our case, $\mathbf{Y}$ has two-side dependency such that either a column or a row may be regarded as a graph signal. The term $\text{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top)$ measures the smoothness of row vectors over a column graph. This formulation is however consistent with the statistical modelling convention where each column in $\mathbf{Y}$ is often regarded as a random variable and the graph of main interest is the column graph.

observations. The authors in [66] further propose a probabilistic generative model of the noise-free smooth observations $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]^\top$ such that

$$\mathbf{y}_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{L}^\dagger), \quad i = 1, 2, \ldots, n \tag{2.3}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo-inverse of a matrix. This leads to a graph learning framework which solves an optimisation problem similar to Eq.(2.1).

### 2.2.2 Kronecker Product Kernel Regression

Taking a functional viewpoint on the generation of graph-structured data matrix, we can make use of the well-studied formalism of Kronecker product kernel ridge regression to infer the latent function [179]. Specifically, we consider $f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ to be an element of a reproducing kernel Hilbert space (RKHS) corresponding to the product kernel function $\kappa_\otimes = \kappa_\mathcal{Z} \otimes \kappa_\mathcal{X}$ on $\mathcal{Z} \times \mathcal{X}$.

A kernel function can be expressed as an inner product in a corresponding feature space, i.e. $\kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}_{i'}) = \langle \phi_\mathcal{Z}(\mathbf{z}_i), \phi_\mathcal{Z}(\mathbf{z}_{i'}) \rangle_{\mathcal{H}_\mathcal{Z}}$ where $\phi_\mathcal{Z} : \mathcal{Z} \to \mathcal{H}_\mathcal{Z}$ and $\kappa_\mathcal{X}(\mathbf{x}_i, \mathbf{x}_{i'}) = \langle \phi_\mathcal{X}(\mathbf{x}_i), \phi_\mathcal{X}(\mathbf{x}_{i'}) \rangle_{\mathcal{H}_\mathcal{X}}$, where $\phi_\mathcal{X} : \mathcal{X} \to \mathcal{H}_\mathcal{X}$. An explicit representation of feature maps $\phi_\mathcal{X}$ and $\phi_\mathcal{Z}$ is not necessary and the dimension of mapped feature vectors could be high and even infinite. By the representer theorem, the function $f$ that fits the data $\mathbf{Y}$ takes the form

$$f(\mathbf{z}, \mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}) \kappa_\mathcal{X}(\mathbf{x}_j, \mathbf{x}) \tag{2.4}$$

where $a_{ij} \in \mathbb{R}$ are the coefficients to be learned, and the estimated value $\hat{y}_{ij} = f(\mathbf{z}_i, \mathbf{x}_j)$. Denoting the corresponding kernel matrices as $\mathbf{K}_z$ and $\mathbf{K}_x$, where $(\mathbf{K}_z)_{ii'} = \kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}_{i'})$ and $(\mathbf{K}_x)_{jj'} = \kappa_\mathcal{X}(\mathbf{x}_j, \mathbf{x}_{j'})$, we have the matrix form

$$\hat{\mathbf{Y}} = \mathbf{K}_z \mathbf{A} \mathbf{K}_x \tag{2.5}$$

where $\hat{\mathbf{Y}}$ is an approximation to $\mathbf{Y}$ and the coefficient matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ has the $ij$-th entry as $a_{ij}$. We assume the observation $\mathbf{Y}$ is a noisy version of $\hat{\mathbf{Y}}$, where the noise is *i.i.d.* normally distributed random variables such that $\epsilon_{ij} = \mathbf{Y}_{ij} - \hat{\mathbf{Y}}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where $\sigma_\epsilon^2$ measures the noise level. This leads to a natural choice of the sum of squared errors as the loss function.

A standard Tikhonov regulariser is often added to the regression model to reduce overfitting and penalise complex functions, which is defined in our case as

$$\begin{aligned} ||f||^2_{\mathcal{H}_\otimes} &= \mathrm{vec}(\mathbf{A})^\top (\mathbf{K}_x \otimes \mathbf{K}_z) \mathrm{vec}(\mathbf{A}) \\ &= \mathrm{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) \end{aligned} \tag{2.6}$$

where vec$(\cdot)$ is the vectorisation operator for a matrix. We now arrive at the following optimisation problem to infer the function $f(\mathbf{z}, \mathbf{x})$ that approximates the observation matrix $\mathbf{Y}$ such that

$$\min_{\mathbf{A}} \ ||\mathbf{Y} - \mathbf{K}_z \mathbf{A} \mathbf{K}_x||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) \tag{2.7}$$

where the hyperparameter $\lambda > 0$ controls the penalisation of the complexity of the function to be learned.

To have a better understanding of how this model is expressive for the two-side dependency, we show that the objective in Eq.(2.7) can be derived from a Bayesian viewpoint. In the vector form, i.e. $\mathbf{a} = \mathrm{vec}(\mathbf{A})$ and $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, we assume that both the data likelihood $p(\mathbf{y}|\mathbf{a})$ and the prior $p(\mathbf{a})$ follow a Gaussian distribution:

$$\mathbf{y}|\mathbf{a} \sim \mathcal{N}((\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a}, \sigma_\epsilon^2 \mathbf{I}_{nm}), \tag{2.8a}$$

$$\mathbf{a} \sim \mathcal{N}(\mathbf{0}_{nm}, \mathbf{K}_x^\dagger \otimes \mathbf{K}_z^\dagger), \tag{2.8b}$$

where $\mathbf{0}_{mn}$ is a zero-vector of length $mn$. Notice that $\mathbf{K}_x$ and $\mathbf{K}_z$ (and their Kronecker product) can be either singular or non-singular matrices, depending on the kernel choice. For the sake of simplicity, we use the pseudo-inverse notation throughout the chapter. Now, the marginal likelihood of $\mathbf{y}$ is

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}_{mn}, \mathbf{K}_x \otimes \mathbf{K}_z + \sigma_\epsilon^2 \mathbf{I}_{nm}) \tag{2.9}$$

from which we can see that the covariance structure of $\mathbf{y}$ can be understood as a combination of $\mathbf{K}_x$ and $\mathbf{K}_z$. Specifically, in the noise-free scenario where $\sigma_\epsilon^2 = 0$, the covariance matrix over the rows of $\mathbf{Y}$ is

$$\mathrm{Cov}_r[\mathbf{Y}] = E[\mathbf{Y}^\top \mathbf{Y}] = \mathrm{Tr}(\mathbf{K}_z)\mathbf{K}_x \tag{2.10}$$

Similarly, the covariance matrix over the columns of $\mathbf{Y}$ is

$$\mathrm{Cov}_c[\mathbf{Y}] = E[\mathbf{Y}\mathbf{Y}^\top] = \mathrm{Tr}(\mathbf{K}_x)\mathbf{K}_z \tag{2.11}$$

The proof for Eq.(2.10) and Eq.(2.11) can be found in [64]. In Appendix 2.5, we show that the maximisation of the log-posterior of the coefficient vector $\mathbf{a}$ leads to the objective in Eq.(2.7).

## 2.3 Kernel Graph Learning

### 2.3.1 A Smoothness Measure for Dependent Graph Signals

Given $n$ dependent graph signals observed on $m$ nodes, we want to infer a graph Laplacian matrix $\mathbf{L} \in \mathbb{R}^{m \times m}$ that captures the invariant relationship between node pairs. The invariance here is understood in the sense that $\mathbf{L}$ does not change due to the observation-side dependency. This is motivated by the fact that the classical Laplacian quadratic in Eq.(2.1) as a smoothness measure is no longer applied for non-*i.i.d.* signals. By recognising a Kronecker product covariance structure in a noise-free version of $\mathbf{y}$ from Eq.(2.9), we define a novel measure of smoothness for the graph signals with observation-side dependency over an invariant graph (with graph Laplacian $\mathbf{L}$) as

$$\mathbf{y}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\mathbf{y} \tag{2.12}$$

where $\mathbf{y}$ is modelled by the Kronecker product kernel regression in Section 2.2.2 and thus the measure can be approximated, based on Eq.(2.5), by

$$\begin{aligned} \hat{\mathbf{y}}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\hat{\mathbf{y}} &= \mathrm{vec}(\hat{\mathbf{Y}})^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\mathrm{vec}(\hat{\mathbf{Y}}) \\ &= \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) \end{aligned} \tag{2.13}$$

One interpretation of the above smoothness term is associated with a Kronecker product graph, if we further assume the observation-side dependency corresponds to a graph such that $\mathbf{K}_z^\dagger = \mathbf{L}_z$. Such an assumption is typically used in defining kernel functions on graphs [193]. Now, Eq.(2.12) turns into a standard Laplacian quadratic form such that

$$\mathbf{y}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\mathbf{y} = \mathbf{y}^\top \mathbf{L}_\otimes \mathbf{y} \tag{2.14}$$

where $\mathbf{L}_\otimes = \mathbf{L} \otimes \mathbf{L}_z$. We further show in Appendix 2.5 that $\mathbf{L}_\otimes$ is a Laplacian-like operator on which a notion of frequency of $\mathbf{y}$ in the context of graph Fourier transform can be defined.

Furthermore, $\mathbf{L}_\otimes$ brings another interpretation for the smoothness term. It can be viewed as a Laplacian-based regulariser which can be added to the problem of

inferring a function $f$ that fits the graph signals in Eq.(2.7):

$$\begin{aligned} ||f||^2_{\mathcal{H}_\mathcal{M}} &= \langle f, \mathbf{L}_\otimes f\rangle_{\mathcal{H}_\mathcal{M}} \\ &= \langle f, (\mathbf{L} \otimes \mathbf{K}_z^\dagger) f\rangle_{\mathcal{H}_\mathcal{M}} \\ &= \text{vec}(\hat{\mathbf{Y}})^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger) \text{vec}(\hat{\mathbf{Y}}) \\ &= \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) \end{aligned} \tag{2.15}$$

where $\mathcal{M}$ denotes a compact manifold[2].

The difference between $\mathbf{K}_x$ and $\mathbf{L}$ is worth clarifying, as they both represent the relationship between nodes in the graph. $\mathbf{K}_x$ is obtained from node-side covariates $\mathbf{x} \in \mathcal{X}$ and can be viewed as an initial estimate of the dependency between nodes obtained from prior information, e.g. node feature in addition to graph signals. By contrast, $\mathbf{L}$ can be interpreted as the refined estimate, i.e. the graph of interest, obtained by learning from the observed graph signals.

### 2.3.2 Learning Framework

We propose a joint learning framework for inferring the function $f$ that fits the graph signals as in Eq.(2.7) as well as the underlying graph $\mathbf{L}$ to capture the relationship between the nodes as in Eq.(2.13). This relationship is disentangled from the observation-side dependency of non-*i.i.d.* graph signals with the notion of smoothness introduced in Section 2.3.1. We name this framework Kernel Graph Learning (**KGL**) which aims at solving the following problem:

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \quad & J(\mathbf{L}, \mathbf{A}) = ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x||^2_F + \lambda \text{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & \quad + \rho \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||^2_F \\ \text{s.t.} \quad & \text{Tr}(\mathbf{L}) = m \end{aligned} \tag{2.16}$$

where $\mathcal{L} = \{\mathbf{L}|\mathbf{L}\mathbf{1} = 0, \mathbf{L}_{jj'} = \mathbf{L}_{j'j} \leq 0, \forall j \neq j'\}$, and $||\cdot||_F$ denotes the Frobenius norm. The first two terms correspond to the functional learning part where the hyperparameter $\lambda > 0$ controls the complexity of the function $f$ for fitting $\mathbf{Y}$. The last two terms and the constraints can be viewed as a graph learning model in Eq.(2.1) with the fitted values of $\mathbf{Y}$ as input graph signals and the sum barrier as the graph regulariser. The hyperparameter $\rho > 0$ controls the relative importance between fitting the function and learning the graph, and $\psi > 0$ controls the distribution of

[2]We refer the interested reader to [20] for the theorem of manifold regularisation with the Laplace-Beltrami operator.

edge weights. The trace constraint acts as a normalisation term such that the sum of learned edge weights equals the number of nodes. The model is compatible with constraints that enforce other properties on the learned graph, e.g. the log barrier introduced in Section 2.2.1. This chapter is mainly based on one of the choices for the constraints in order to maintain focus on the general framework.

### 2.3.3 Optimisation: Alternating Minimisation

We first recognise that Eq.(2.16) is a biconvex optimisation problem, i.e. convex w.r.t $\mathbf{A}$ while $\mathbf{L}$ is fixed and vice versa. This motivates an iterative block-coordinate descent algorithm that alternates between minimisation in $\mathbf{A}$ and $\mathbf{L}$ [203, 88]. In this section, we derive the update steps of $\mathbf{A}$ and $\mathbf{L}$ separately, propose the main algorithm in Algorithm 1, and prove its convergence.

#### Update of A

The update of coefficients $\mathbf{A}$ can be regarded as solving a Laplacian-regularised kernel regression [20]. Given $\mathbf{L}$, the optimisation problem of Eq.(2.16) becomes

$$\begin{aligned}\min_{\mathbf{A}}\ &||\mathbf{Y}-\mathbf{K}_z\mathbf{A}\mathbf{K}_x||_F^2+\lambda\mathrm{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top)\\ &+\rho\mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z)\end{aligned} \tag{2.17}$$

and, after dropping constant terms,

$$\begin{aligned}\min_{\mathbf{A}}\ J_{\mathbf{L}}(\mathbf{A}) =&\mathrm{Tr}(\mathbf{A}^\top\mathbf{K}_z^2\mathbf{A}\mathbf{K}_x^2)-2\mathrm{Tr}(\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z\mathbf{Y})\\ &+\lambda\mathrm{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top)+\rho\mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z).\end{aligned} \tag{2.18}$$

Denote $\mathbf{a}=\mathrm{vec}(\mathbf{A})$ and $\mathbf{y}=\mathrm{vec}(\mathbf{Y})$, we obtain a dual-form for $J_{\mathbf{L}}(\mathbf{A})$ such that

$$\begin{aligned}J_{\mathbf{L}}(\mathbf{a}) =&\mathbf{a}^\top(\mathbf{K}_x^2\otimes\mathbf{K}_z^2)\mathbf{a}-2\mathbf{a}^\top(\mathbf{K}_x\otimes\mathbf{K}_z)\mathbf{y}\\ &+\lambda\mathbf{a}^\top(\mathbf{K}_x\otimes\mathbf{K}_z)\mathbf{a}+\rho\mathbf{a}^\top\big((\mathbf{K}_x\mathbf{L}\mathbf{K}_x)\otimes\mathbf{K}_z\big)\mathbf{a}\\ =&\mathbf{a}^\top\Big(\mathbf{K}^2+\lambda\mathbf{K}+\rho\mathbf{S}\otimes\mathbf{K}_z\Big)\mathbf{a}-2\mathbf{a}^\top\mathbf{K}\mathbf{y}\end{aligned} \tag{2.19}$$

where $\mathbf{K}=\mathbf{K}_x\otimes\mathbf{K}_z$, and $\mathbf{S}=\mathbf{K}_x\mathbf{L}\mathbf{K}_x$ for simplicity. We prove in Appendix 2.5 that $\mathbf{K}^2+\lambda\mathbf{K}+\rho\mathbf{S}\otimes\mathbf{K}_z$ is positive semi-definite thus Eq.(2.19) is an unconstrained quadratic programme. The gradient of $J_{\mathbf{L}}(\mathbf{a})$ w.r.t. $\mathbf{a}$ is

$$\nabla J_{\mathbf{L}}(\mathbf{a})=\Big(\mathbf{K}^2+\lambda\mathbf{K}+\rho\mathbf{S}\otimes\mathbf{K}_z\Big)\mathbf{a}-\mathbf{K}\mathbf{y}. \tag{2.20}$$

Strictly speaking, the matrix $\mathbf{K}^2+\lambda\mathbf{K}+\rho\mathbf{S}\otimes\mathbf{K}_z$ may contain zero eigenvalues which makes it not invertible. However, a majority of popular kernel functions for $\mathbf{K}_x$

and $\mathbf{K}_z$ are positive-definite, e.g. the RBF kernel. Since Kronecker product preserves positive definiteness, $\mathbf{K}$ and hence the whole matrix is positive-definite and invertible. Setting $\nabla J_{\mathbf{L}}(\mathbf{a}) = \mathbf{0}$ and cancelling out $\mathbf{K}$, we have:

$$\begin{aligned} &\Big(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z\Big)\mathbf{a} - \mathbf{K}\mathbf{y} = \mathbf{0} \\ \implies &\Big(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{K}(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n)\Big)\mathbf{a} - \mathbf{K}\mathbf{y} = \mathbf{0} \\ \implies &\Big(\mathbf{K} + \lambda\mathbf{I}_{mn} + \rho\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n\Big)\mathbf{a} = \mathbf{y} \end{aligned} \tag{2.21}$$

Denote $\mathbf{H} = \mathbf{K} + \lambda\mathbf{I}_{mn} + \rho\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n$, where $\mathbf{H}$ has a dimension of $nm \times nm$. We can therefore obtain a closed-form solution such that

$$\mathbf{a} = \mathbf{H}^{-1}\mathbf{y} \tag{2.22}$$

where the inverse of $\mathbf{H}$ requires $\mathcal{O}(n^3m^3)$.

To further reduce the complexity, we make use of the Kronecker structure and matrix tricks. We first recognise $\mathbf{H}$ as

$$\mathbf{H} = \left(\left(\rho\mathbf{L} + \lambda\mathbf{K}_x^{-1}\right) \oplus \mathbf{K}_z\right)(\mathbf{K}_x \otimes \mathbf{I}_n)$$

where $\oplus$ is the Kronecker sum. With the eigendecomposition $\mathbf{K}_x = \mathbf{Q}_x\mathbf{\Lambda}_x\mathbf{Q}_x^\top$, $\mathbf{K}_z = \mathbf{Q}_z\mathbf{\Lambda}_z\mathbf{Q}_z^\top$ and $\rho\mathbf{L} + \lambda\mathbf{K}_x^{-1} = \mathbf{U}_x\mathbf{D}_x\mathbf{U}_x^\top$, we have

$$\mathbf{H} = (\mathbf{U}_x \otimes \mathbf{Q}_z)(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)\left(\mathbf{U}_x^\top \otimes \mathbf{Q}_z^\top\right)\left(\mathbf{Q}_x\mathbf{\Lambda}_x\mathbf{Q}_x^\top \otimes \mathbf{I}_n\right). \tag{2.23}$$

Here, $\mathbf{D}_x \oplus \mathbf{\Lambda}_z$ is an $mn \times mn$ diagonal matrix with entries being all the pairwise sums of eigenvalues in $\mathbf{D}_x$ and $\mathbf{\Lambda}_z$. Inversion of that matrix is thus $O(mn)$. We can now obtain cheap inversion with

$$\begin{aligned} \mathbf{H}^{-1}\mathbf{y} = &\left(\mathbf{Q}_x\mathbf{\Lambda}_x^{-1}\mathbf{Q}_x^\top \otimes \mathbf{I}_n\right)(\mathbf{U}_x \otimes \mathbf{Q}_z)(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1} \\ &\cdot \operatorname{vec}\left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right). \end{aligned} \tag{2.24}$$

The operation $(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1} \operatorname{vec}\left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right)$ is simply rescaling each term in the $mn$-vector with the corresponding diagonal entry of $(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1}$. If we denote $\mathbf{d}_x$ and $\mathbf{d}_z$ column vectors containing the diagonal entries this can be expressed as $\operatorname{vec}(\mathbf{B})$ with

$$\mathbf{B} = \left(\mathbf{1}_n\mathbf{d}_x^\top + \mathbf{d}_z\mathbf{1}_m^\top\right)^{\circ -1} \circ \left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right),$$

where $\circ$ denotes the Hadamard product and $(\cdot)^{\circ -1}$ denotes entrywise inversion. Remaining operations are now direct and give the closed-form solution as

$$\begin{aligned} \mathbf{A} &= \mathbf{Q}_z\mathbf{B}\mathbf{U}_x^\top\mathbf{Q}_x\mathbf{\Lambda}_x^{-1}\mathbf{Q}_x^\top \\ &= \mathbf{Q}_z\mathbf{B}\mathbf{U}_x^\top\mathbf{K}_x^{-1} \\ &= \mathbf{Q}_z\left[\left(\mathbf{1}_n\mathbf{d}_x^\top + \mathbf{d}_z\mathbf{1}_m^\top\right)^{\circ -1} \circ \left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right)\right]\mathbf{U}_x^\top\mathbf{K}_x^{-1}. \end{aligned} \tag{2.25}$$

Notice that this solution requires only matrix multiplications and inversions and eigen-decompositions on $m \times m$ or $n \times n$ matrices, giving an overall computational cost of $\mathcal{O}\left(n^3 + m^3 + nm^2 + n^2m\right)$.

When $\mathbf{K}_x$ and $\mathbf{K}_z$ are not invertible, we suggest using the gradient descent to avoid the inverse of a large matrix of dimension $nm \times nm$. The update step using Eq.(2.20) is:

$$\mathbf{a}^{(\tau+1)} = \mathbf{a}^{(\tau)} - \gamma \nabla J_{\mathbf{L}}(\mathbf{a}^{(\tau)}) \tag{2.26}$$

where $\gamma > 0$ is the step size.

### Update of L

Given $\mathbf{A}$, the optimisation problem of Eq.(2.16) becomes

$$\begin{aligned} \min_{\mathbf{L} \in \mathcal{L}} \;& J_{\mathbf{A}}(\mathbf{L}) = \rho \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi ||\mathbf{L}||_F^2 \\ \text{s.t} \;& \mathrm{Tr}(\mathbf{L}) = m \end{aligned} \tag{2.27}$$

which is a constrained quadratic programme w.r.t. $\mathbf{L}$. By taking $\mathbf{P} = \mathbf{K}_z^{1/2}\mathbf{A}\mathbf{K}_x$, the problem fits in the learning framework in Eq.(2.1). We use the package CVXPY [62] to solve this problem.

**Algorithm 1** Kernel Graph Learning (**KGL**)

**Input:** Observation $\mathbf{Y}$, node-side kernel matrix $\mathbf{K}_x$, observation-side kernel matrix $\mathbf{K}_z$, hyper-parameters $\rho$, $\lambda$ and $\psi$, tolerance level $\epsilon$.

**Initialisation:** $t = 0, \mathbf{A} = \mathbf{0} \in \mathbb{R}^{n \times m}$
**while** $|\mathbf{L}^{(t)} - \mathbf{L}^{(t-1)}| > \epsilon$ and $|\mathbf{A}^{(t)} - \mathbf{A}^{(t-1)}| > \epsilon$ **do**
  Update $\mathbf{L}^{(t)} = \arg\min J_{\mathbf{A}^{(t-1)}}(\mathbf{L})$ by solving Eq.(2.27)
  Update $\mathbf{A}^{(t)} = \arg\min J_{\mathbf{L}^{(t)}}(\mathbf{A})$ by
  (i) using the closed-form solution in Eq.(2.25), **if** $\mathbf{K}_x$ and $\mathbf{K}_z$ are invertible; or
  (ii) updating $\mathrm{vec}(\mathbf{A}^{(t)})$ with gradient descent in Eq.(2.26), **otherwise**
  $t = t + 1$
**end while**
**return** $\mathbf{L}^{(t)}$, $\mathbf{A}^{(t)}$

The overall KGL framework is presented in Algorithm 1. The convergence for each update step of $\mathbf{A}^{(t)}$ and $\mathbf{L}^{(t)}$ is guaranteed in solving the respective convex optimisation of Eq.(2.18) and Eq.(2.27). It should be noted that the step size $\gamma$ in Eq.(2.26) needs to be set appropriately for the gradient descent to converge. We suggest a $\gamma \leq 10^{-4}$ from empirical results. The choices of hyperparameters $\lambda$, $\rho$ and $\psi$ are discussed in Section 2.4.6. We now provide the following statement on convergence of Algorithm 1.

**Lemma 2.3.1.** *The sequence* $\{J(\mathbf{L}^{(t)}, \mathbf{A}^{(t)})\}$ *generated by Algorithm 1 converges monotonically and the solution obtained by Algorithm 1 is a stationary point of Eq.(2.16).*

*Proof.* We follow the convergence results of the alternate convex search in [88] and that of a more general cyclic block-coordinate descent algorithm in [203]. By recognising that Eq.(2.18) and Eq.(2.27) are quadratic programmes (with Lemma 2.5.1 in Appendix 2.5), the problem of Eq.(2.16) is a bi-convex problem with all the terms differentiable and the function $J(\mathbf{L}, \mathbf{A})$ continuous and bounded from below. Theorem 4.5 in [88] states that the sequence $\{J(\mathbf{L}^{(t)}, \mathbf{A}^{(t)})\}$ generated by Algorithm 1 converges monotonically. Theorem 4.1 in [203] states that the sequence $\{\mathbf{L}^{(t)}\}$ and $\{\mathbf{A}^{(t)}\}$ generated by Algorithm 1 are defined and bounded. Furthermore, according to Theorem 5.1 in [203], every cluster point $\{\mathbf{L}^{(t)}, \mathbf{A}^{(t)}\}$ is a coordinatewise minimum point of $J$ hence the solution is a stationary point of Eq.(2.16).

□

Our empirical results suggest that after only 10 iterations or less, the sequence $\{\mathbf{L}^{(t)}, \mathbf{A}^{(t)}\}$ does not change more than the tolerance level. The computational complexity of **KGL** in Algorithm 1 is dominated by the step of updating $\mathbf{A}$. It requires $\mathcal{O}(n^3+m^3+nm^2+n^2m)$ to compute the closed-form solution of $\mathbf{A}$ or $\mathcal{O}(n^3m^3)$ to compute the gradient in Eq.(2.20) if the closed-form solution is not applied when $\mathbf{K}_z$ and $\mathbf{K}_x$ are not invertible. Updating $\mathbf{L}$ requires $\mathcal{O}(m^2)$. Overall, for $T$ iterations that guarantee the convergence of Algorithm 1, it requires either $\mathcal{O}(T(n^3 + m^3 + nm^2 + n^2m))$ or $\mathcal{O}(T(n^3m^3))$ operations, depending on whether $\mathbf{K}_z$ and $\mathbf{K}_x$ are chosen to be invertible. We note that one could readily appeal to large-scale kernel approximation methods for further reduction of computational and storage complexity of the **KGL** framework, and hence broaden its applicability to larger datasets. There are two main approaches to large-scale kernel approximations and both can be applied to **KGL**. The former focuses on kernel matrix approximations using methods such as Nyström sampling [126], while the latter deals with the approximation of the kernel function itself, using methods such as Random Fourier Features [78]. Nonetheless, this chapter focuses on the modelling perspective, and we will leave the algorithmic improvement as a future direction.

### 2.3.4 Special Cases of Kernel Graph Learning

**Independent observations** It is often assumed that graph signals are *i.i.d.* hence there exists no dependency along the observation side. This is equivalent to setting

$\mathbf{K}_z = \mathbf{I}_n$ in our framework. We refer to this special case of **KGL** as Node-side Kernel Graph Learning (**KGL-N**):

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \; & ||\mathbf{Y} - \mathbf{A}\mathbf{K}_x||_F^2 + \lambda \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & + \rho \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \quad & \mathrm{Tr}(\mathbf{L}) = m \end{aligned} \tag{2.28}$$

**No node-side information** It also may be the case that no node-side information is available for the problem at hand. In this case, we can simply set $\mathbf{K}_x = \mathbf{I}_m$ in **KGL**, leading to to Observation-side Kernel Graph Learning (**KGL-O**):

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \; & ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}||_F^2 + \lambda \mathrm{Tr}(\mathbf{A}^\top\mathbf{K}_z\mathbf{A}) \\ & + \rho \mathrm{Tr}(\mathbf{A}\mathbf{L}\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \quad & \mathrm{Tr}(\mathbf{L}) = m \end{aligned} \tag{2.29}$$

In the cases of one-side **KGL**, the optimisation again follows the alternating minimisation, i.e. to solve for **KGL-N** or **KGL-O**), one can simply set $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$ in Algorithm 1. It should be noted, however, that the update step of $\mathbf{A}$ requires less computational cost when either $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$. Indeed, the objective function of **KGL-N** can be decomposed into the sum according to $n$ functions such that

$$J_{\mathbf{L}}(\{\mathbf{a}_i\}_{i=1}^n) = \sum_{i=1}^{n} \Big( ||\mathbf{y}_i - \mathbf{K}_x\mathbf{a}_i||_2^2 + \mathbf{a}_i^\top(\lambda\mathbf{K}_x + \rho\mathbf{S})\mathbf{a}_i \Big) \tag{2.30}$$

where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]^\top$ and $\mathbf{S} = \mathbf{K}_x\mathbf{L}\mathbf{K}_x$. Consequently, the update step can be parallelised.

### 2.3.5 Learning with Missing Observations

By modifying the least-squares loss in **KGL**, we propose an extension to jointly learn the underlying graph and function from graph-structured data with missing values. We encode the positions of missing values with a mask matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ such that $\mathbf{M}_{ij} = 0$ if $\mathbf{Y}_{ij}$ is missing, and $\mathbf{M}_{ij} = 1$ otherwise. Now, we only need to minimise the least-squares loss over observed $\mathbf{Y}_{ij}$ in the functional learning part, which leads to the formulation:

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \; & ||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_{\mathcal{Z}}\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & + \rho \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \quad & \mathrm{Tr}(\mathbf{L}) = m. \end{aligned} \tag{2.31}$$

This formulation also applies to one-side kernel graph learning, i.e. **KGL-N** or **KGL-O**, with $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$.

The optimisation problem in Eq.(2.31) is a bi-convex problem and alternating minimisation can still be applied. The update step of $\mathbf{L}$ remains the same as in Eq.(2.27), but the gradient in the update step of $\mathbf{a} = \text{vec}(\mathbf{A})$ (Step 4. in Algorithm 1) becomes

$$\nabla J_{\mathbf{L}}(\mathbf{a}) = \Big(\mathbf{K}\text{diag}(\mathbf{m})\mathbf{K} + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z\Big)\mathbf{a} - \mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) \tag{2.32}$$

where $\mathbf{m} = \text{vec}(\mathbf{M})$. The detailed derivation of the gradient is provided in Appendix 2.5. We further assume $\mathbf{K}$ is invertible, which is a mild assumption as we have many choices of kernel functions for $\mathbf{K}_x$ and $\mathbf{K}_z$ to be invertible. Also noting $\mathbf{S} \otimes \mathbf{K}_z = \mathbf{K}(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n)$, the gradient becomes

$$\nabla J_{\mathbf{L}}(\mathbf{a}) = \Big(\text{diag}(\mathbf{m})\mathbf{K} + \lambda\mathbf{I}_{nm} + \rho(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n)\Big)\mathbf{a} - \text{vec}(\mathbf{M} \circ \mathbf{Y}). \tag{2.33}$$

One can either derive a close-form solution or use gradient descent based on Eq.(2.33).

## 2.4 Experiments

### 2.4.1 General Settings

**Groundtruth Graphs** Random graphs of $m$ nodes are drawn from the Erdös-Rényi (ER), Barabási-Albert (BA) and stochastic block model (SBM) as groundtruth, which are denoted as $\mathcal{G}_{\text{ER}}$, $\mathcal{G}_{\text{BA}}$ and $\mathcal{G}_{\text{SBM}}$, respectively. The parameters of each network model are chosen to yield an edge density of 0.3. The edge weights are randomly drawn from a uniform distribution $\mathbf{W}_{ij} \sim \mathcal{U}(0, 1)$. The weighted adjacency matrix is set as $\mathbf{W} = (\mathbf{W} + \mathbf{W}^\top)/2$ for symmetry and normalised such that the sum of edge weights is equal to $m$ for ease in comparison. The graph Laplacian $\mathbf{L}$ is calculated from $\mathbf{L} = \text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W}$.

**Groundtruth Data** We generate mild noisy data $\mathbf{Y} \in \mathbb{R}^{n \times m}$ from $\mathbf{Y} = \mathbf{K}_z\mathbf{A}\mathbf{K}_x + \mathbf{E}$, where $\mathbf{a} = \text{vec}(\mathbf{A})$ is drawn from $\mathbf{a} \sim \mathcal{N}(\mathbf{0}_{mn}, \mathbf{K}_x^\dagger \otimes \mathbf{K}_z^\dagger)$ according to Eq.(2.8b). Every entry of the noise matrix $\mathbf{E}$ is an *i.i.d.* sample from $\mathbf{E}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$. To test the proposed model against different levels of noises, we vary the value of $\sigma_\epsilon^2$ in Section 2.4.2. For all other synthetic experiments, we add a mild-level noise with $\sigma_\epsilon^2 = 0.01$. We choose $\mathbf{K}_x = (\mathbf{I} + \lambda\mathbf{L})^{-1}$, as it is a popular method to generate smooth signals in related work [112, 206]. We consider both dependence and independence along the observation side:

- **Independent Data**: $\mathbf{K}_z = \mathbf{I}_n$;
- **Dependent Data**: $\mathbf{K}_z$ is obtained from an RBF kernel evaluated on synthetic observation-side information $\mathbf{z} = [0, 1, 2, \ldots, n-1]^\top$, which can be interpreted as the time-stamps of a discrete-time Markov chain. The bandwidth parameter is chosen according to the median heuristic [81].

**Model Candidates** To have a fair comparison, the models are divided into two groups. The first group contains the baseline models that cannot deal with observation-side dependence:

- **GL** (Eq.(14) in [112]): the GSP graph learning model in Eq.(2.1) with $\Omega(\mathbf{L}) = ||\mathbf{L}||_F^2$.
- **GL-2step** (Eq.(16) in [66]): a two-step GSP graph learning framework with an identity mapping as denoising function. From a modelling perspective, Eq.(13) in [23] proposed a similar model with more constraints on edge weights and a different optimisation algorithm. We treat them as the same kind of techniques.
- **KGL-N** (proposed model in Eq. (2.28)): $\mathbf{K}_z = \mathbf{I}_n$ in **KGL**.

The second group is examined with observation-side dependent data:

- **KGL-Agnostic** (Eq.(18) in [207]): As discussed in Section 2.1.1, the joint learning model in [207] considered the observation-side kernel, but did not use the observation-side dependence in learning the graph. We denote their model as **KGL-Agnostic** with our notations:

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \; & ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}||_F^2 + \lambda \mathrm{Tr}(\mathbf{A}^\top \mathbf{K}_z \mathbf{A}) \\ & + \rho \mathrm{Tr}(\mathbf{K}_z \mathbf{A}\mathbf{L}\mathbf{A}^\top \mathbf{K}_z) + \psi ||\mathbf{L}||_F^2 \end{aligned} \tag{2.34}$$

- **KGL** (proposed model in Eq. (2.16)): the main learning framework.
- **KGL-O** (proposed model in Eq.(2.29)): To have a fair comparison to **KGL-Agnostic**, we also assume the graph is agnostic to the model (i.e. $\mathbf{K}_x = \mathbf{I}_m$ as model input).

For each method, we determine the hyperparameters via a grid search, and report the highest performance achieved by the best set of hyperparameters.

**Evaluation Metrics** Average precision score (APS) and normalised sum of squared errors ($\mathrm{SSE}_{\mathcal{G}}$) are used to evaluate the graph estimates, and out-of-sample mean squared error ($\mathrm{MSE}_y$) is used to evaluate the estimated entries of graph-structured data matrix that were not observed (or missing). The APS is defined in a binary classification scenario for graph structure recovery, which automatically varies the threshold of weights above which the edges are declared as learned edges. An APS score of

1 indicates that the algorithm can precisely detect the ground-truth edges and non-edges. The $\text{SSE}_{\mathcal{G}}$ is defined over learned adjacency matrix $\hat{\mathbf{W}}$ and the groundtruth adjacency matrix $\mathbf{W}_0$ as

$$\text{SSE}_{\mathcal{G}} = \frac{||\hat{\mathbf{W}} - \mathbf{W}_0||_F^2}{||\mathbf{W}_0||_F^2}.$$

The out-of-sample $\text{MSE}_y$ of data matrix is defined with a mask matrix $\mathbf{M}$ (same as in Eq.(2.31)), where $\mathbf{M}_{ij} = 0$ indicates $\mathbf{Y}_{ij}$ is a missing entry:

$$\text{Out-of-sample MSE}_y = \frac{||(\mathbf{1}\mathbf{1}^\top - \mathbf{M}) \circ (\hat{\mathbf{Y}} - \mathbf{Y})||_F^2}{||\mathbf{1}\mathbf{1}^\top - \mathbf{M}||_F^2}$$

where $\hat{\mathbf{Y}} = \mathbf{K}_z\mathbf{A}\mathbf{K}_x$ and $\mathbf{A}$ is obtained from model estimates. Similarly, we are interested in the training $\text{MSE}_y$ for analysing overfitting:

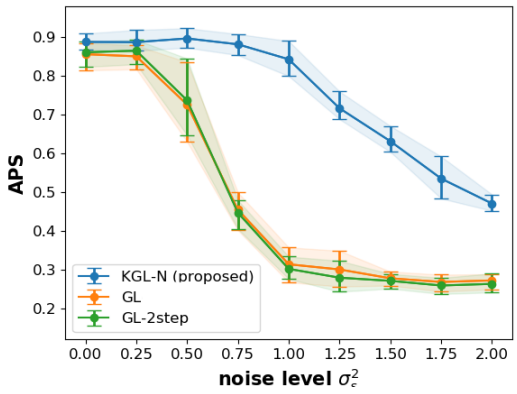
$$\text{Training MSE}_y = \frac{||\mathbf{M} \circ (\hat{\mathbf{Y}} - \mathbf{Y})||_F^2}{||\mathbf{M}||_F^2}.$$

### 2.4.2 Learning a Graph from Noisy Graph Signals

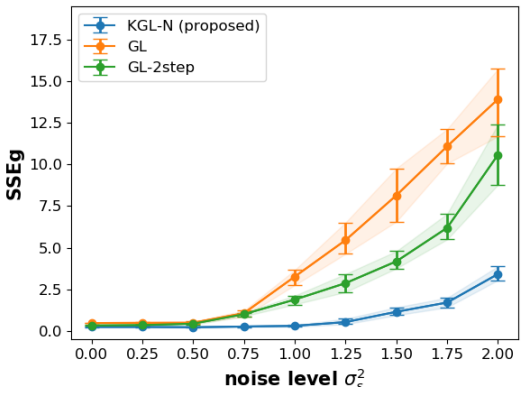
In order to evaluate the performance of the proposed model in learning a graph from noisy data, we add noise to the groundtruth data such that $\mathbf{Y} = \mathbf{K}_z\mathbf{A}\mathbf{K}_x + \mathbf{E}$, where every entry of the noise matrix $\mathbf{E}$ is an *i.i.d.* sample from $\mathbf{E}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$. We vary the noise level $\sigma_\epsilon^2$ from 0 to 2 against which we plot the evaluation metrics in Figure 2.2. Under the same settings of the noise level, random graph and model candidate, we repeat the experiment for 10 times and report the mean (the solid curves) as well as the 5th and 95th percentile (the error bars) of the evaluation metrics.

From Figure 2.2, Figure 2.3 and Figure 2.4, the proposed models outperform the baseline models in terms of all evaluation metrics. Specifically, for $\mathcal{G}_{\text{SBM}}$, when the data are independent, the performance of **KGL-N** drops slowly as the noise level increases, while that of **GL** and **GL-2step** drops quickly when noise level goes above 0.5. It is worth mentioning that the curves of two almost overlap in terms of APS. This indicates the identity mapping in **GL-2step** as denoising function does not help much in recovering graph structure, although it yields slightly smaller $\text{SSE}_{\mathcal{G}}$ than **GL** with the noise level greater then 0.75.

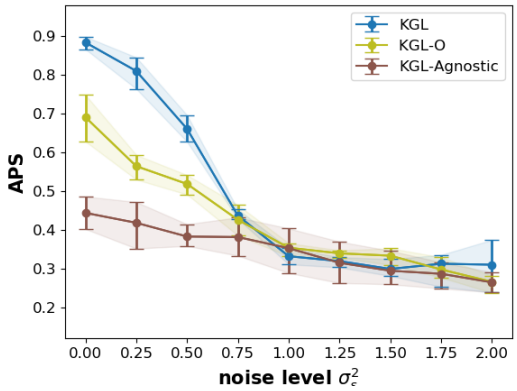
For the dependent data, the proposed model **KGL** achieves a high performance when the noise level is less than 0.75. Even without the node-side information $\mathbf{K}_x$ as model input in **KGL** (note that the groundtruth data are generated in a consistent way in [207] proposing **KGL-agnostic**), the proposed method (**KGL-O**) can still learn a meaningful graph with slightly worse performance compared to **KGL**.



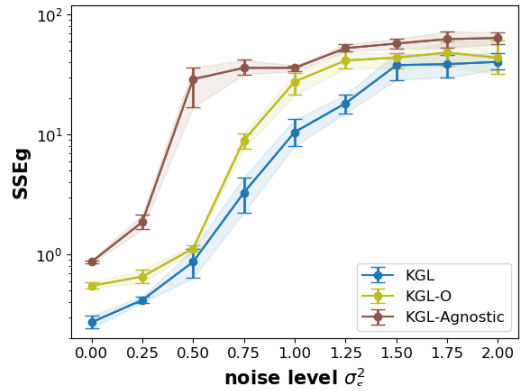

(a) $\mathcal{G}_{\text{SBM}}$, independent data




(b) $\mathcal{G}_{\text{SBM}}$, independent data




(c) $\mathcal{G}_{\text{SBM}}$, dependent data




(d) $\mathcal{G}_{\text{SBM}}$, dependent data

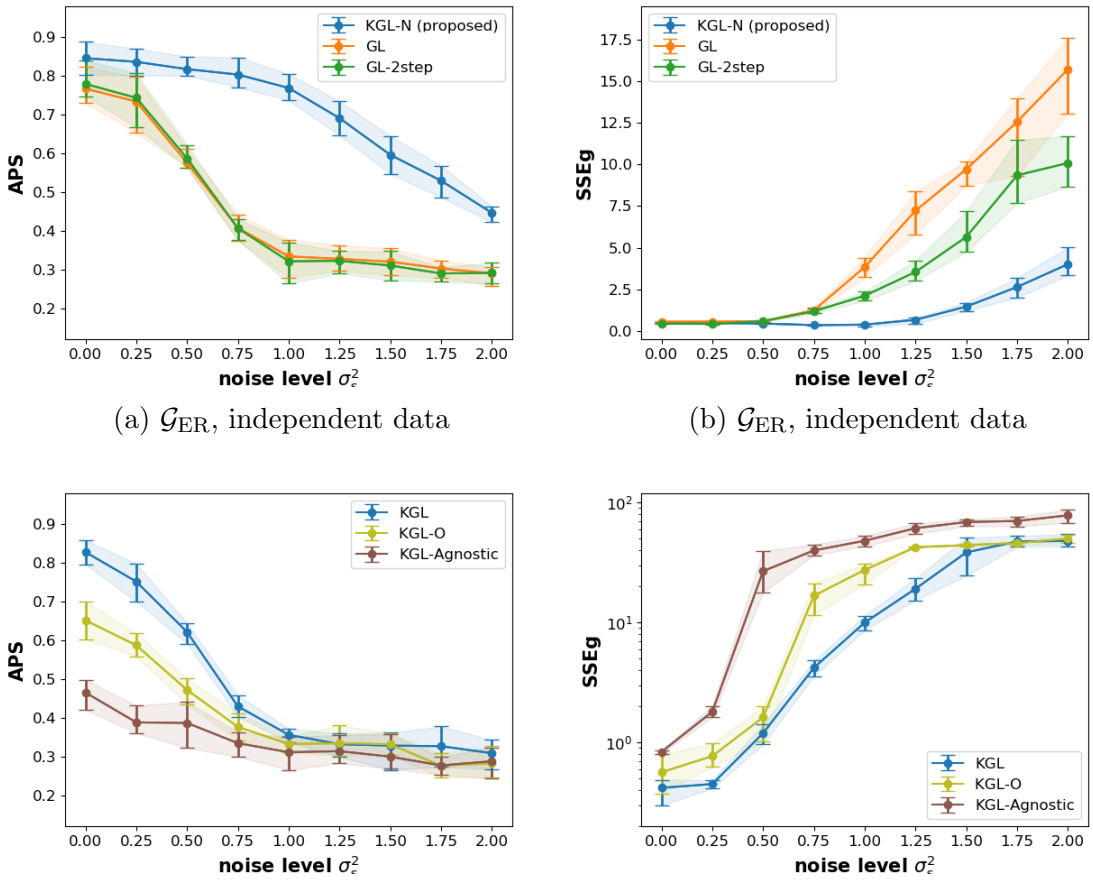
Figure 2.2: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{SBM}}$ in terms of APS and $\text{SSE}_{\mathcal{G}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels.

By contrast, **KGL-agnostic** cannot recover the groundtruth graph to a satisfying level even with little noise ($\sigma_\epsilon^2 = 0$), as its smoothness term does not capture the dependence structure on the observation side.

From Figure 2.3 and Figure 2.4, we see that $\mathcal{G}_{\text{BA}}$ is slightly more difficult to recover from data, but an improvement can nevertheless be seen in the proposed models from the baselines when the noise level is low.

### 2.4.3 Learning a Graph from Missing Graph Signals

To examine the performance of learning a graph from incomplete data with **KGL** described in Section 2.3.5, we generate the mask matrix $\mathbf{M}$ indicating missing entries, where $\mathbf{M}_{ij} \overset{i.i.d.}{\sim} Bernoulli(1-r)$ and $r$ is the missing rate, i.e. $\mathbf{M}_{ij}$ has a probability of




(a) $\mathcal{G}_{\text{ER}}$, independent data

(b) $\mathcal{G}_{\text{ER}}$, independent data

(c) $\mathcal{G}_{\text{ER}}$, dependent data

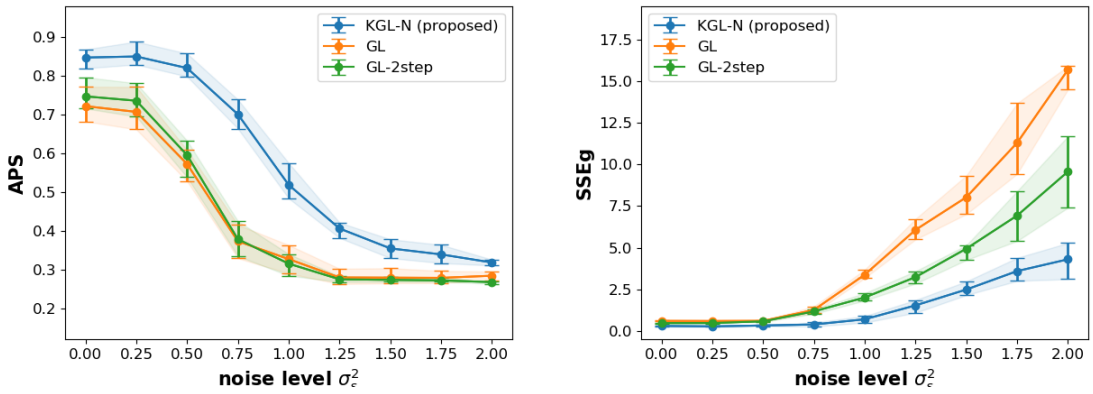
(d) $\mathcal{G}_{\text{ER}}$, dependent data

Figure 2.3: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{ER}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels.

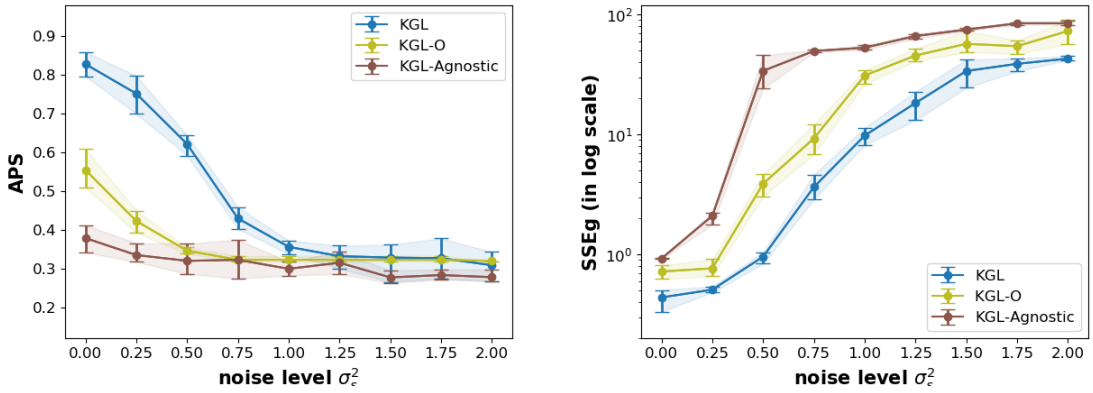
$r$ to be missing and has a value of 0. The preprocessed data with 0 replacing missing entries is $\mathbf{Y}_m = \mathbf{Y} \circ \mathbf{M}$, which is a natural choice in practice for the model candidates that cannot directly deal with missing entries, as the mean value of the entries in $\mathbf{Y}$ is 0 by design. We use $\mathbf{Y}_m$ as the input in the baseline models **GL** and **GL-2Step**. For **KGL-Agnostic** in Eq.(2.34), we also add a mask matrix $\mathbf{M}$ in the least-squares loss term to have a fair comparison.

We vary $r$ from 0 to 0.9, against which we plot the evaluation metrics, APS and $\text{SSE}_{\mathcal{G}}$, in Figure 2.5. Similar to the noisy scenario, we repeat the experiments 10 times under the same settings of missing rate, random graph and model candidate and report the mean (the solid curves) as well as the 5th and 95th percentile (the error bars) of the evaluation metrics.

For $\mathcal{G}_{\text{SBM}}$, the proposed methods **KGL** and **KGL-N** can recover the groundtruth




(a) $\mathcal{G}_{\text{BA}}$, independent data (b) $\mathcal{G}_{\text{BA}}$, independent data

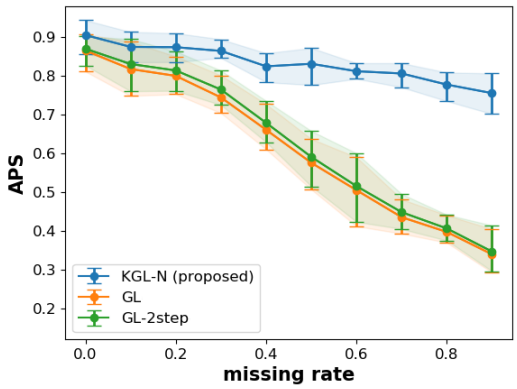



(c) $\mathcal{G}_{\text{BA}}$, dependent data (d) $\mathcal{G}_{\text{BA}}$, dependent data

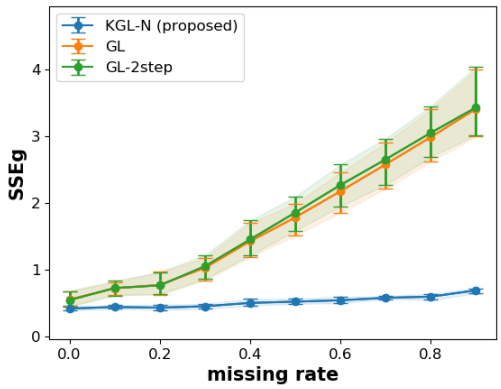
Figure 2.4: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{BA}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels.

graphs reasonably well even when there are 80% missing entries. For the independent data scenario, the performance of the baseline models with the preprocessed data $\mathbf{Y}_m$ drops steeply as the missing rate increases. Although it can still recover the groundtruth graphs with a high APS and low $\text{SSE}_{\mathcal{G}}$ when the missing rate is less than 20%, the performance is not as good as **KGL-N**. By contrast, for **KGL-N**, the APS only drops by 0.1 from no missing entries to around 90% missing entries, while $\text{SSE}_{\mathcal{G}}$ only increases by around 0.05.

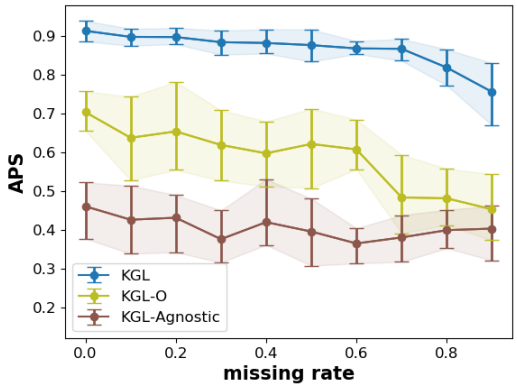
For the dependent data scenario, all three model candidates can deal with missing data directly by adding a mask matrix in the least-squares loss term. Consequently, their curves are relatively stable as the missing rate increases from 0 to 80%. However, the accuracy levels at which each of the model stabilises are different. The proposed method **KGL**, aware of both node-side and observation-side information, achieves



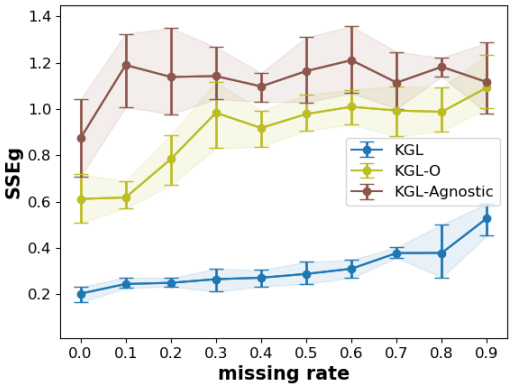

(a) $\mathcal{G}_{\text{SBM}}$, independent data




(b) $\mathcal{G}_{\text{SBM}}$, independent data




(c) $\mathcal{G}_{\text{SBM}}$, dependent data

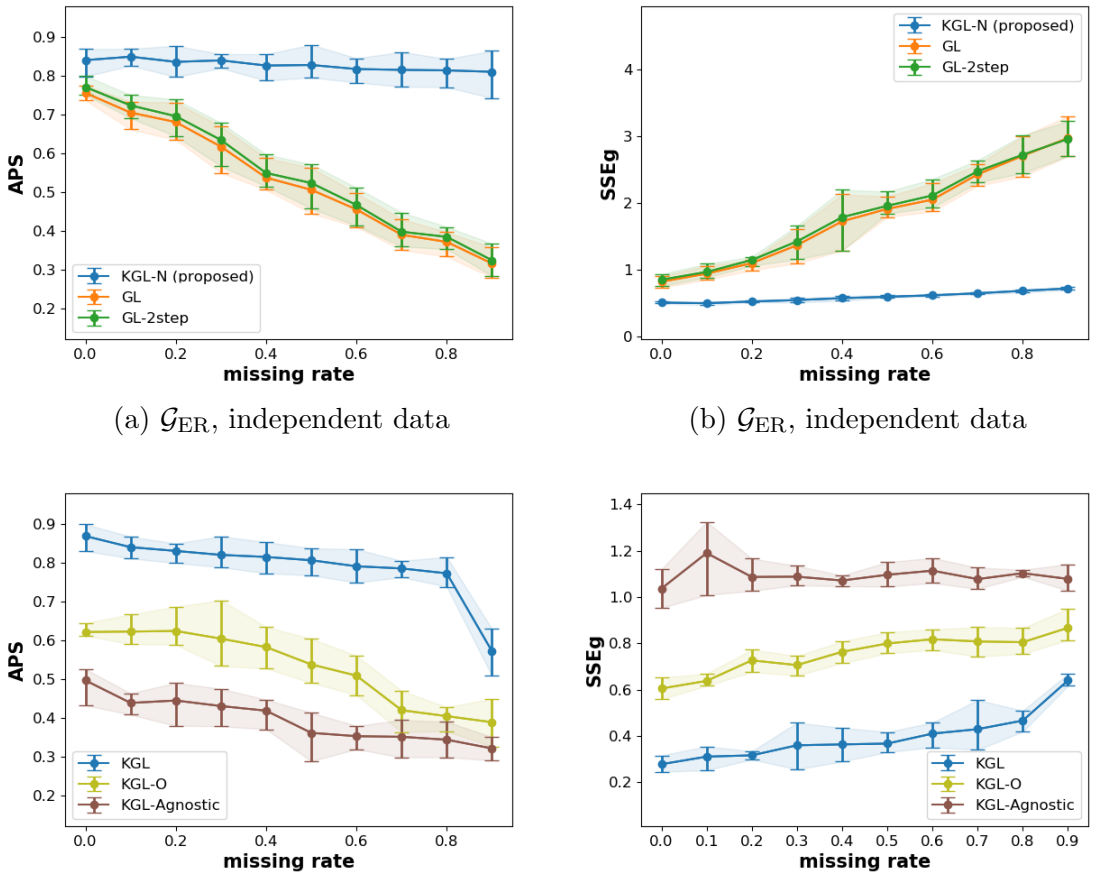



(d) $\mathcal{G}_{\text{SBM}}$, dependent data

Figure 2.5: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{SBM}}$ in terms of APS and $\text{SSE}_{\mathcal{G}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$.

the highest APS and lowest $\text{SSE}_{\mathcal{G}}$. By contrast, **KGL-O**, without access to the node-side information, can still recover a meaningful graph but with less correct edges and less accurate edge weights when the missing rate is less than 0.6. The performance of **KGL-Agnostic** is not as good as **KGL-O** because the smoothness term in Eq.(2.34) is not able to disentangle the influence of the observation-side dependency from the graph structure.

The performance of **KGL** does not differ much for different random graph models. Still, there is an improvement in recovering $\mathcal{G}_{\text{BA}}$ compared to the baseline graph learning models, as can be seen in Figure 2.7c and Figure 2.7d.




(a) $\mathcal{G}_{\text{ER}}$, independent data

(b) $\mathcal{G}_{\text{ER}}$, independent data

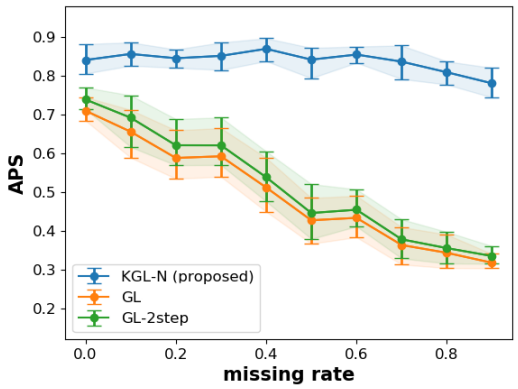
(c) $\mathcal{G}_{\text{ER}}$, dependent data

(d) $\mathcal{G}_{\text{ER}}$, dependent data

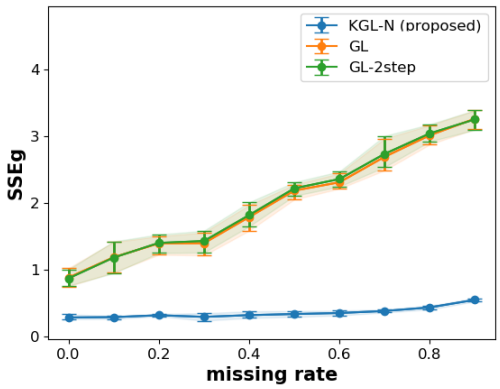
Figure 2.6: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{ER}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$.

### 2.4.4 Graph-Structured Matrix Completion

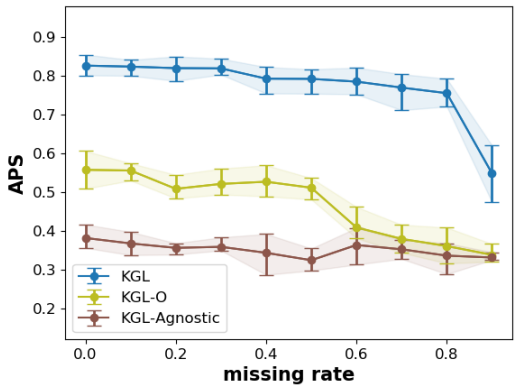
In the experiment of learning a graph with incomplete data matrix in Section 2.4.3, we are also interested in the performance of matrix completion. In Figure 2.8, we plot the $\text{MSE}_y$ of the missing entries (i.e. the out-of-sample $\text{MSE}_y$) that is averaged over all types of graphs against the varying missing rate. The proposed methods lead to much smaller errors compared to baseline models, for both independent and dependent data. It should be noted that **GL** does not offer a mechanism for inferring missing data, hence is not included in this experiment.



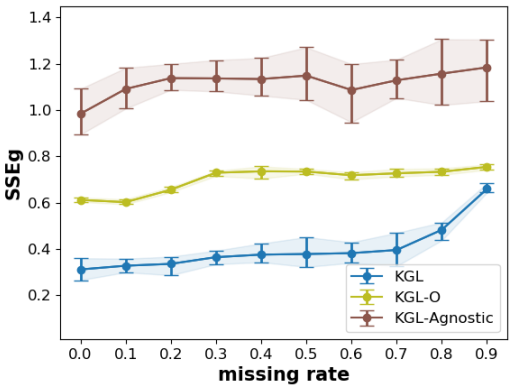

(a) $\mathcal{G}_{\text{BA}}$, independent data




(b) $\mathcal{G}_{\text{BA}}$, independent data

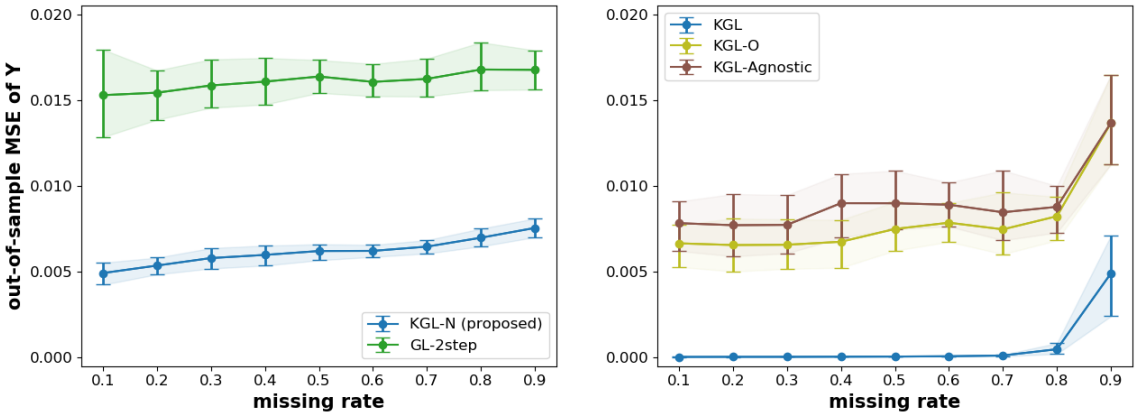



(c) $\mathcal{G}_{\text{BA}}$, dependent data




(d) $\mathcal{G}_{\text{BA}}$, dependent data

Figure 2.7: The performance of recovering groundtruth graphs $\mathcal{G}_{\text{BA}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$.




(a) independent data (b) dependent data

Figure 2.8: The MSE of recovering missing entries in $\mathbf{Y}_m$. The results show the mean of 30 random experiments of 3 different types of graphs, i.e. 10 for each graph type.

### 2.4.5 Learning a Graph of Different Sizes

The graph learning performance of the proposed method varies with the size of graphs and number of observations. As shown in Figure 2.9, a hundred observations are sufficient to recover a small graph with $m = 20$ nodes with a high accuracy. When the graph size increases, the number of the observations required for **KGL** to achieve a high APS increases roughly exponentially. On the other hand, when the number of observations increases, the variance of APS of the learned graphs decreases.

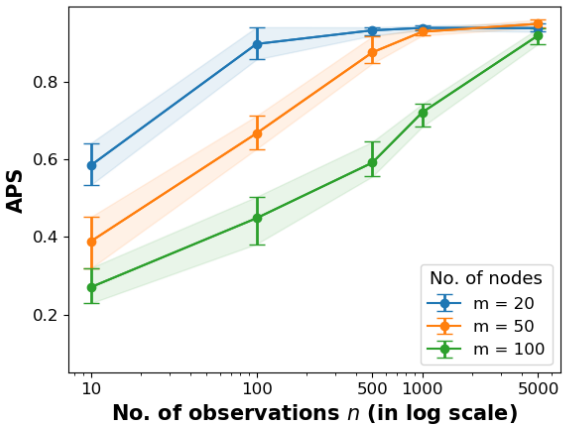
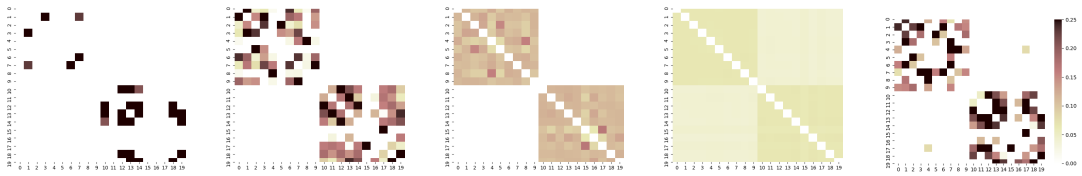
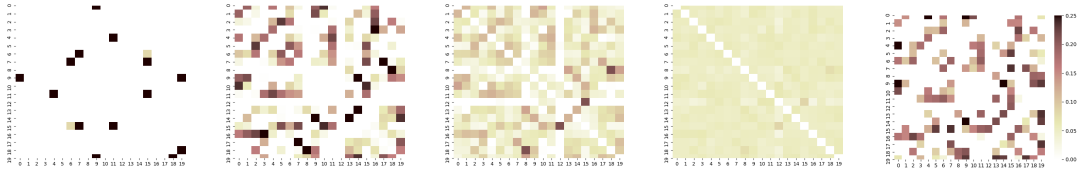


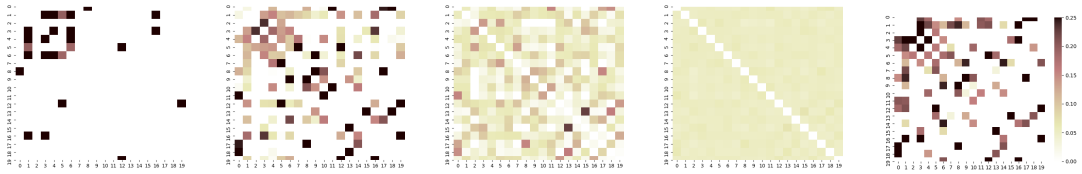

Figure 2.9: The performance of learning graphs of different sizes $m$ against varying number of observations $n$ with the proposed model **KGL**. The results show the mean of 30 random experiments of 3 different types of graphs, i.e. 10 for each graph type.



(a) $\psi = 10^{-7}$ (APS = 0.39) (b) $\psi = 10^{-5}$ (APS = 0.86) (c) $\psi = 10^{-3}$ (APS = 0.80) (d) $\psi = 10^{-1}$ (APS = 0.52) (e) Groundtruth $\mathcal{G}_{\text{SBM}}$



(f) $\psi = 10^{-7}$ (APS = 0.33) (g) $\psi = 10^{-5}$ (APS = 0.90) (h) $\psi = 10^{-3}$ (APS = 0.78) (i) $\psi = 10^{-1}$ (APS = 0.48) (j) Groundtruth $\mathcal{G}_{\text{ER}}$



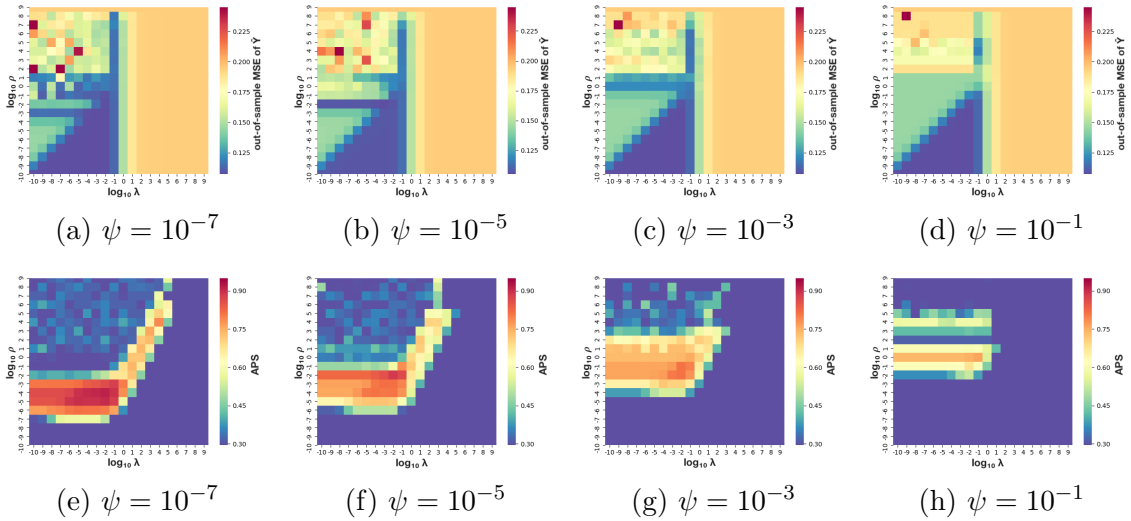
(k) $\psi = 10^{-7}$ (APS = 0.40) (l) $\psi = 10^{-5}$ (APS = 0.89) (m) $\psi = 10^{-3}$ (APS = 0.80) (n) $\psi = 10^{-1}$ (APS = 0.67) (o) Groundtruth $\mathcal{G}_{\text{BA}}$

Figure 2.10: Graph sparsity with respect to $\psi$. The first row (a)-(d): the learned $\mathcal{G}_{\text{SBM}}$; the second row (f)-(i): the learned $\mathcal{G}_{\text{ER}}$; the third row (k)-(n): the learned $\mathcal{G}_{\text{BA}}$, all from **KGL** with $\lambda = 10^{-1}$, $\rho = 10^{-2}$ and a fixed $\psi$. The respective groundtruth graphs are shown in the last column.

### 2.4.6 Impact of Regularisation Hyperparameters

Three hyperparameters are involved in **KGL** and its variants. As introduced in Section 2.3, $\lambda > 0$ controls the complexity of the functional learning and prevents overfitting; $\rho > 0$ controls the relative importance of graph learning compared to functional learning, and at the same time determines the smoothness of the predicted data $\hat{\mathbf{y}}$ over $\mathbf{L} \otimes \mathbf{K}_z^{\dagger}$; Finally, $\psi > 0$ controls the Frobenius ($\ell_2$) norm of the graph Laplacian which, together with the trace ($\ell_1$) constraint, bears similarity to an elastic net regularisation [235]. The larger the $\psi$, the less sparse the graph with more uniform edge weights. The accuracy in learning the graph Laplacian $\mathbf{L}$ and inferring the data matrix $\mathbf{Y}$ is determined by the combination of these three hyperparameters, which is not straightforward to visualise and analyse at the same time. Fortunately, we may still gain some insights by examining their distinct effects separately.

Firstly, $\psi$ should be chosen according to the prior belief of the graph sparsity




(a) $\psi = 10^{-7}$ (b) $\psi = 10^{-5}$ (c) $\psi = 10^{-3}$ (d) $\psi = 10^{-1}$

(e) $\psi = 10^{-7}$ (f) $\psi = 10^{-5}$ (g) $\psi = 10^{-3}$ (h) $\psi = 10^{-1}$

Figure 2.11: The out-of-sample MSE for data matrix $\mathbf{Y}$ (the first row) and the APS of the learned graph (the second row) with respect to $\lambda$ and $\rho$, with 80% entries of $\mathbf{Y}$ as training sample from **KGL** with a fixed $\psi$.

defined as the number of edges with non-zero weights. As shown in Figure 2.10, when $\psi \to 0$, the learned graph contains only a few most significant edge. Due to the constraint on the sum of edge weights, i.e. $\text{tr}(\mathbf{L}) = m$, the total weights $m$ are allocated to a few significant edges when $\psi$ is small. When $\psi \to \infty$, the learned graph becomes fully connected with equal edge weights. In the synthetic experiment where we have knowledge of the groundtruth graph, the best accuracy is obtained when the sparsity coincides with the groundtruth graph.

The value of $\psi$, on the other hand, has little effect on the accuracy of predicting the missing entries in $\mathbf{Y}$, as the update step of $\mathbf{A}$ does not involve the term with $\psi$. As shown in Figure 2.11(a)-(d), the out-of-sample $\text{MSE}_y$ is determined by the combination of $\lambda$ and $\rho$. We first notice that the error is the same when $\lambda > 10^2$, showing that we overly penalise the function complexity in this case. Indeed, when $\lambda \to \infty$, the function is overly smooth such that the entries of the coefficient matrix $\mathbf{A}$ are all zero leading to the entries of prediction $\hat{\mathbf{Y}}$ being all zero as well. This also happens when $\rho \to \infty$, where the vector form of the prediction $\hat{\mathbf{y}}$ is forced to be overly smooth on $\mathbf{L} \otimes \mathbf{K}_z^\dagger$. In particular, when $\mathbf{K}_z^\dagger = \mathbf{I}_n$, every row vector of $\hat{\mathbf{Y}}$ (i.e. the predicted graph signal) has constant entries, as a result of minimising the term $\text{Tr}(\hat{\mathbf{Y}}^\top \mathbf{K}_z^\dagger \hat{\mathbf{Y}} \mathbf{L})$ to zero. On the other hand, when $\mathbf{K}_z^\dagger \neq \mathbf{I}_n$, all the entries in $\hat{\mathbf{Y}}$ has constant values such that this term is minimised to zero.

The ranges of values of $\lambda$ and $\rho$ for which the out-of-sample $\text{MSE}_y$ is the smallest generally coincide with that for which the APS is the largest in Figure 2.11, e.g. when

$\psi = 10^{-5}$, $\lambda = 10^{-2}$ and $\rho = 10^{-2}$. However, the out-of-sample $\text{MSE}_y$ is generally small when $\lambda < 0.1$ and $\rho < 0.1$. This is understandable as the function could be very complex with little penalisation, but this does not guarantee good performance in recovering the groundtruth graph.

## 2.5 Supplementary Derivation

**Kronecker product kernel regression** Taking a Bayesian viewpoint, we provide an interpretation of Eq.(2.7). From Eq.(2.8), maximising the log-posterior of the coefficient vector $\mathbf{a}$ leads to the objective in Eq.(2.7):

$$\begin{aligned}
&\max_{\mathbf{a}} \log p(\mathbf{a}|\mathbf{y}) \\
=& \max_{\mathbf{a}} \log p(\mathbf{y}|\mathbf{a}) + \log p(\mathbf{a}) \\
=& \max_{\mathbf{a}} -\big(\mathbf{y} - (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a}\big)^\top \big(\mathbf{y} - (\mathbf{K}_z \otimes \mathbf{K}_z\mathbf{a}\big) \\
&\qquad - \lambda \mathbf{a}^\top (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a} \\
=& \min_{\mathbf{a}} ||\mathbf{y} - (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a}||_2^2 + \lambda \mathbf{a}^\top (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a} \\
=& \min_{\mathbf{A}} ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x||_F^2 + \lambda \text{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top)
\end{aligned}$$

where $\lambda$ is some constant parameter proportional to the variance of the noise $\sigma_\epsilon^2$ in Eq.(2.8a).

**Kronecker Product Laplacian-like Operator** We define a Laplacian-like operator with a Kronecker product structure $\mathbf{L}_\otimes = \mathbf{L} \otimes \mathbf{L}_z$ for the data matrix with both node-side and observation-side dependency. Although $\mathbf{L}_\otimes$ may have positive off-diagonal entries and thus may not be a valid graph Laplacian matrix, it satisfies the following properties and a notion of frequency of $\mathbf{y}$ can be defined upon $\mathbf{L}_\otimes$:

- $\mathbf{L}_\otimes$ is symmetric and $\mathbf{L}_\otimes \cdot \mathbf{1} = \mathbf{0}$;
- $\mathbf{L}_\otimes$ admits the eigendecomposition

  $$\begin{aligned}
  \mathbf{L}_\otimes &= \mathbf{L} \otimes \mathbf{L}_z \\
  &= (\mathbf{U} \otimes \mathbf{U}_z)(\mathbf{\Lambda} \otimes \mathbf{\Lambda}_z)(\mathbf{U} \otimes \mathbf{U}_z)^\top
  \end{aligned}$$

  where $\mathbf{U}$ and $\mathbf{\Lambda}$, and $\mathbf{U}_z$ and $\mathbf{\Lambda}_z$, are the eigenvector and eigenvalue matrices of the Laplacian matrices $\mathbf{L}$ and $\mathbf{L}_z$, respectively, and $\mathbf{U}_\otimes = (\mathbf{U} \otimes \mathbf{U}_z)$ is an orthogonal matrix and $\mathbf{\Lambda}_\otimes = (\mathbf{\Lambda} \otimes \mathbf{\Lambda}_z)$ is a diagonal matrix with real entries.

We can also obtain a two-dimensional graph Fourier transform $\check{\mathbf{Y}}$ of $\mathbf{Y}$ as in [154]:

$$\text{vec}(\check{\mathbf{Y}}) = (\mathbf{U} \otimes \mathbf{U}_z)^\top \text{vec}(\mathbf{Y}).$$

**Proof of positive semi-definiteness**

**Lemma 2.5.1.** *The matrix* $\mathbf{C} = \mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z$ *is positive semi-definite.*

*Proof.* To prove $\mathbf{C}$ is positive semi-definite (p.s.d.), it suffices to show that the matrices $\mathbf{K}^2$, $\mathbf{K}$ and $\mathbf{S} \otimes \mathbf{K}_z$ are p.s.d.

As $\mathbf{K}_x$ and $\mathbf{K}_z$ are kernel matrices constructed by pairwise evaluations from two reproducing kernels $\kappa_{\mathcal{X}}$ and $\kappa_{\mathcal{Z}}$, they are p.s.d. The Kronecker product $\mathbf{K}$ is p.s.d, which is easy to prove from the eigendecomposition:

$$\begin{aligned} \mathbf{K} &= \mathbf{K}_x \otimes \mathbf{K}_z \\ &= (\mathbf{U}_x \mathbf{\Lambda}_x \mathbf{U}_x^\top) \otimes (\mathbf{U}_z \mathbf{\Lambda}_z \mathbf{U}_z^\top) \\ &= (\mathbf{U}_x \otimes \mathbf{U}_z)(\mathbf{\Lambda}_x \otimes \mathbf{\Lambda}_z)(\mathbf{U}_x \otimes \mathbf{U}_z)^\top \end{aligned} \tag{2.35}$$

where $\mathbf{U} = (\mathbf{U}_x \otimes \mathbf{U}_z)$ is an orthogonal matrix and $\mathbf{\Lambda} = (\mathbf{\Lambda}_x \otimes \mathbf{\Lambda}_z)$ is a diagonal matrix with non-negative real entries.

Next, $\mathbf{K}^2 = \mathbf{K}\mathbf{K}$ is also p.s.d., as it is a product of commuting matrices and $\mathbf{K}^2$ preserves symmetry [150].

Finally, denote the column vectors of $\mathbf{K}_x$ as $[\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_m]$, the weighted adjacency matrix as $\mathbf{W}$, and the non-negative edge weight between node $j$ and $j'$ as $w_{jj'}$. We have

$$\mathbf{S} = \mathbf{K}_x \mathbf{L} \mathbf{K}_x = \sum_{j \neq j'} w_{jj'} (\mathbf{k}_j - \mathbf{k}_{j'})(\mathbf{k}_j - \mathbf{k}_{j'})^\top$$

where $w_{jj'} \geq 0$. $\mathbf{S}$ can then be viewed as a weighted covariance matrix, which is symmetric and p.s.d. Therefore, $\mathbf{S} \otimes \mathbf{K}_z$ is p.s.d. following the same argument as for $\mathbf{K}$.

□

**Derivation of Gradient in Eq.(2.33)** In this section, we show the derivation of Eq.(2.33), i.e. the gradient for updating $\mathbf{A}$ in the missing values scenario. Recall that the objective function is

$$\begin{aligned} J_{\mathbf{L}}(\mathbf{A}) =& ||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z \mathbf{A} \mathbf{K}_x)||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) \\ &+ \rho \mathrm{Tr}(\mathbf{A} \mathbf{K}_x \mathbf{L} \mathbf{K}_x \mathbf{A}^\top \mathbf{K}_z). \end{aligned}$$

With the following standard linear algebra identities for any matrices $\mathbf{D}, \mathbf{E}, \mathbf{F}$

- $||\mathbf{D} \circ \mathbf{E}||_F^2 = \mathrm{Tr}\big((\mathbf{D} \circ \mathbf{E})^\top (\mathbf{D} \circ \mathbf{E})\big) = \mathrm{Tr}\big(\mathbf{E}^\top (\mathbf{D} \circ \mathbf{E})\big)$
- $\mathrm{vec}(\mathbf{D} \circ \mathbf{E}) = \mathrm{vec}(\mathbf{D}) \circ \mathrm{vec}(\mathbf{E})$
- $\mathrm{Tr}(\mathbf{D}^\top \mathbf{E}) = \mathrm{vec}(\mathbf{D})^\top \mathrm{vec}(\mathbf{E})$

- $\text{vec}(\mathbf{DEF}) = (\mathbf{F}^\top \otimes \mathbf{D})\text{vec}(\mathbf{E})$

the first term of $J_{\mathbf{L}}(\mathbf{A})$ becomes

$$\begin{aligned}
&||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2 \\
=&\text{Tr}\Big((\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top\big(\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)\big)\Big)
\end{aligned}$$

By dropping constant terms, we have

$$\begin{aligned}
&||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2 \\
=&\text{Tr}\Big( - 2(\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top(\mathbf{M} \circ \mathbf{Y})\Big) \\
&+ \text{Tr}\Big((\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top\big(\mathbf{M} \circ (\mathbf{K}_z\mathbf{A}\mathbf{K}_x)\big)\Big) \\
=& - 2\text{vec}(\mathbf{A})^\top(\mathbf{K}_x \otimes \mathbf{K}_z)\text{vec}(\mathbf{M} \circ \mathbf{Y}) \\
&+ \text{vec}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top\big(\text{vec}(\mathbf{M}) \circ \text{vec}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x)\big) \\
=& - 2\mathbf{a}^\top\mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top\mathbf{K}\big(\mathbf{m} \circ (\mathbf{K}\mathbf{a})\big)
\end{aligned}$$

where $\mathbf{a} = \text{vec}(\mathbf{A})$, $\mathbf{m} = \text{vec}(\mathbf{M})$ and $\mathbf{K} = \mathbf{K}_x \otimes \mathbf{K}_z$. Putting it back to the objective and recognising the fact that $\mathbf{d} \circ \mathbf{e} = \text{diag}(\mathbf{d})\mathbf{e}$ for two vectors $\mathbf{d}$ and $\mathbf{e}$, we have

$$\begin{aligned}
J_{\mathbf{L}}(\mathbf{a}) =& - 2\mathbf{a}^\top\mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top\mathbf{K}\big(\mathbf{m} \circ (\mathbf{K}\mathbf{a})\big) \\
&+ \lambda\mathbf{a}^\top\mathbf{K}\mathbf{a} + \rho\mathbf{a}^\top(\mathbf{S} \otimes \mathbf{K}_z)\mathbf{a} \\
=& - 2\mathbf{a}^\top\mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top\mathbf{K}\text{diag}(\mathbf{m})\mathbf{K}\mathbf{a} \\
&+ \lambda\mathbf{a}^\top\mathbf{K}\mathbf{a} + \rho\mathbf{a}^\top(\mathbf{S} \otimes \mathbf{K}_z)\mathbf{a}
\end{aligned}$$

where $\mathbf{S} = \mathbf{K}_x\mathbf{L}\mathbf{K}_x$. We thus obtain the gradient for deriving Eq.(2.32) such that

$$\begin{aligned}
\nabla J_{\mathbf{L}}(\mathbf{a}) =& - \mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{K}\text{diag}(\mathbf{m})\mathbf{K}\mathbf{a} \\
&+ \lambda\mathbf{K}\mathbf{a} + \rho\big(\mathbf{S} \otimes \mathbf{K}_z\big)\mathbf{a}.
\end{aligned}$$

## 2.6 Conclusions

In this chapter, we have revisited the smooth graph signals from a functional viewpoint and proposed a kernel-based graph learning framework that can integrate node-side and observation-side covariates. Specifically, we have designed a novel notion of smoothness of graph signals over the Kronecker product of two graph Laplacian matrices and combined it with a Kronecker product kernel regression of graph signals in order to capture the two-side dependency. We have shown the effectiveness and efficiency of the proposed method, via extensive synthetic experiments, demonstrating

its usefulness in learning a meaningful topology from noisy, incomplete and dependent graph signals. Although we have proposed a fast implementation exploiting the Kronecker structure of kernel matrices, the computational complexity remains cubic in the maximum of the number of nodes and the number of signals. Hence, a natural future direction is to further reduce the computational complexity with the state-of-the-art methods for large-scale kernel-based learning, such as random Fourier features. Another interesting direction is to develop a generative graph learning model based upon the framework presented here, using connections between Gaussian processes and kernel methods.

# Chapter 3

# Graph Learning with Unrolled Neural Network

## Contents



## 3.1 Introduction

Graphs are an effective modelling language for revealing relational structure in high-dimensional complex domains and may assist in a variety of machine learning tasks. However, in many practical scenarios an explicit graph structure may not be readily available or easy to define. Graph learning aims at learning a graph topology from observation on data entities and is therefore an important problem studied in the literature.

Model-based graph learning from the perspectives of probabilistic graphical models [130] or graph signal processing [191, 180, 160] solves an optimisation problem over the space of graph candidates whose objective function reflects the inductive graph-data interactions. The data are referred as to the observations on nodes, node features or graph signals in literature. The convex objective function often contains a graph-data fidelity term and a structural regulariser. For example, by assuming

that data follow a multivariate Gaussian distribution, the graphical Lasso [13] maximises the $\ell_1$ regularised log-likelihood of the precision matrix which corresponds to a conditional independence graph. Based on the convex formulation, the problem of learning an optimal graph can be solved by many iterative algorithms, such as proximal gradient descent [176], with convergence guarantees.

These classical graph learning approaches, despite being effective, share several common limitations. Firstly, handcrafted convex regularisers may not be expressive enough for representing the rich topological and structural priors. In the literature, only a limited number of graph structural properties can be modelled explicitly using convex regularisers, e.g. the $\ell_1$ penalty for enforcing sparsity. More complex structures such as scale-free and small-world properties have been largely overlooked due to the difficulty in imposing them via simple convex optimisation. Secondly, graph structural regularisers might not be differentiable despite being convex, which perplexed the design of optimisation algorithms. Thirdly, iterative algorithms might take long to converge, as the search space grows quadratically with the graph size. Fourthly, tuning penalty parameters in front of structural regularisers and the step size of iterative algorithms are laborious.

To address the above limitations, we propose a novel functional learning framework to learn a mapping from node observations to the underlying graph topology with desired structural property. Our framework is inspired by the emerging field of *learning to optimise* (L2O) [39, 153]. Specifically, as shown in Figure 3.1, we first unroll an iterative algorithm for solving the aforementioned regularised graph learning objective. To further increase the flexibility and expressiveness, we design a topological difference variational autoencoder (TopoDiffVAE) to replace the proximal projector of structural regularisation functions in the original iterative steps. We train the model in an end-to-end fashion with pairs of data and graphs that share the same structural properties. Once trained, the model can be deployed to learn a graph topology that exhibits such structural properties.

The proposed method learns topological priors from graph samples, which are more expressive in learning structured graphs compared to traditional methods using analytic regularisers. Compared to deep neural networks, unrolling an iterative algorithm that is originally designed for graph learning problem introduces a sensible inductive bias that makes our model highly interpretable. We test the effectiveness and robustness of the proposed method in learning graphs with diverse topological structures on both synthetic and real-world data.

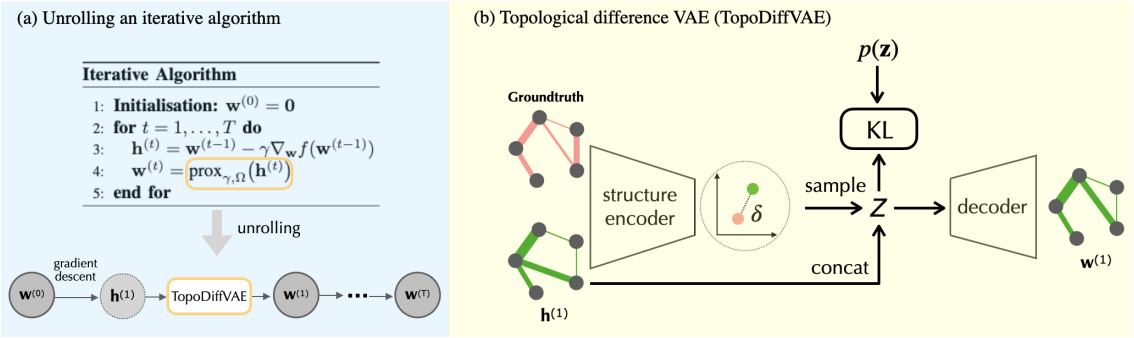



Figure 3.1: An illustration of the proposed framework.

The main contributions are as follows. Firstly, we propose a novel and highly interpretable neural networks to learn a data-to-graph mapping based on algorithmic unrolling. Secondly, we propose an effective TopoDiffVAE module that replaces the proximal operators of structural regularisers. To the best of our knowledge, this constitutes the first attempt in addressing the difficult challenge of designing convex functions to represent complex topological priors, which in turn allows for improved flexibility and expressiveness. Finally, the proposed method improves the accuracy of graph learning by 80% compared to traditional iterative solvers.

### 3.1.1 Related Work

**Graph Learning.** Broadly speaking there are two approaches to the problem of graph learning in the literature: model-based and learning-based approaches. The model-based methods solve an regularised convex optimisation whose objective function reflects the inductive graph-data interactions and regularisation that enforces embody structural priors [129, 192, 72, 59, 66, 79, 112, 23]. They often rely on iterative algorithms which require specific designs, including proximal gradient descent [176], alternating direction method of multiplier (ADMM) [29], block-coordinate decent methods [79] and primal-dual splitting [112]. Our model unrolls a primal-dual splitting as an inductive bias, and it allows for regularisation on both the edge weights and node degrees.

Recently, learning-based methods have been proposed to introduce more flexibility to model structural priors in graph learning. GLAD [190] unrolls an alternating minimisation algorithm for solving an $\ell_1$ regularised log-likelihood maximisation of the precision matrix. It also allows the hyperparameters of the $\ell_1$ penalty term to differ element-wisely, which further enhances the representation capacity. Our work differs from GLAD in that (1) we optimise over the space of weighted adjacency matrices

that contain non-negative constraints than over the space of precision matrices; (2) we unroll a primal-dual splitting algorithm where the whole proximal projection step is replaced with TopoDiffVAE, while GLAD adds a step of learning element-wise hyperparameters. Furthermore, GLAD assumes that the precision matrices have a smallest eigenvalue of 1, which affects its accuracy on certain datasets. Deep-Graph [19] uses a CNN architecture to learn a mapping from empirical covariance matrices to the binary graph structure. We will show in later sections that a lack of task-specific inductive bias leads to an inferior performance in recovering structured graphs.

**Learning to optimise.** Learning to optimise (L2O) combines the flexible data-driven learning procedure and interpretable rule-based optimisation [39]. Algorithmic unrolling is one main approach of L2O [153]. Early development of unrolling was proposed to solve sparse and low-rank regularised problems and the main goal is to reduce the laborious iterations of hand engineering [90, 194, 155, 1]. Another approach is Play and Plug that plugs pre-trained neural networks in replacement of certain update steps in an iterative algorithm [174, 230]. Our framework is largely inspired by both approaches, and constitutes a first attempt in utilising L2O for the problem of graph learning.

## 3.2 Learning to Learn graph topologies

### 3.2.1 From Solving an Optimisation to Learning a Functional Mapping

Traditional model-based graph learning is formulated as to solve a convex optimisation w.r.t. the graph adjacency or Laplacian matrix. In this section, we introduce how we convert this approach to learning a functional mapping from graph signals to graph adjacency or Laplacian matrices, with algorithm unrolling.

Let $G = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ be an undirected weighted graph, where $\mathcal{V}$ is the node set with $|\mathcal{V}| = m$, $\mathcal{E}$ is the edge set and $\mathbf{W}$ is the weighted adjacency matrix whose $ij$-th entry embodies the similarity between node $i$ and node $j$. The combinatorial graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ is a diagonal matrix of node degrees. In many scenarios, we have access to a structured data matrix, which is denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^\top \in \mathbb{R}^{m \times n}$, where each column $\mathbf{x}_i$ can be considered as a signal on the graph $G$. The goal of graph learning is to infer such a graph $G$ that is fully represented by $\mathbf{L}$ or $\mathbf{W}$ from $\mathbf{X}$. Specifically, we solve a generic optimisation

problem

$$\min_{\mathbf{L}} \operatorname{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) + \Omega(\mathbf{L}), \tag{3.1}$$

where the first term is the so-called Laplacian quadratic form, measuring the variation of $\mathbf{X}$ on the graph, and $\Omega(\mathbf{L})$ is a convex regulariser on $\mathbf{L}$ to reflect structural priors. Since $\mathbf{W}$ is symmetric, we introduce the vector of edge weights $\mathbf{w} \in \mathbb{R}_+^{m(m-1)/2}$ and reparameterise Eq.(3.1) based on $\operatorname{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) = \sum_{i,j} \mathbf{W}_{ij} ||\mathbf{x}_i - \mathbf{x}_j||_2^2 = 2\mathbf{w}^\top \mathbf{y}$, where $\mathbf{y} \in \mathbb{R}_+^{m(m-1)/2}$ is the half-vectorisation of the Euclidean distance matrix. Now, we optimise the objective w.r.t $\mathbf{w}$ such that

$$\min_{\mathbf{w}} 2\mathbf{w}^\top \mathbf{y} + \mathcal{I}_{\{\mathbf{w} \geq 0\}}(\mathbf{w}) + \Omega_1(\mathbf{w}) + \Omega_2(\mathcal{D}\mathbf{w}), \tag{3.2}$$

where $\mathcal{I}$ is an indicator function such that $\mathcal{I}_{\mathcal{C}}(\mathbf{w}) = 0$ if $\mathbf{w} \in \mathcal{C}$ and $\mathcal{I}_{\mathcal{C}}(\mathbf{w}) = \infty$ otherwise, and $\mathcal{D}$ is a linear operator that transforms $\mathbf{w}$ into the vector of node degrees such that $\mathcal{D}\mathbf{w} = \mathbf{W}\mathbf{1}$. The reparameterisation reduces the dimension of search space by a half and we do not need to take extra care of the positive semi-definiteness of $\mathbf{L}$.

In Eq.(3.2), the regularisers $\Omega$ is split into $\Omega_1$ and $\Omega_2$ on edge weights and node degrees respectively. Both formulations can be connected to many previous works, where regularisers are mathematically handcrafted to reflect specific graph topologies. For example, the $\ell_1$-regularised log-likelihood maximisation of the precision matrix with Laplacian constraints [129, 124] can be seen as a special case of Eq.(3.1), where $\Omega(\mathbf{L}) = \log\det(\mathbf{L} + \sigma^2\mathbf{I}) + \lambda \sum_{i \neq j} |\mathbf{L}_{ij}|$. The $\ell_1$ norm on edge weights enforces the learned conditional independence graph to be sparse. The author in [112] considers the log-barrier on node degrees, i.e. $\Omega_2(\mathcal{D}\mathbf{w}) = -\mathbf{1}^\top \log(\mathcal{D}\mathbf{w})$, to prevent the learned graph with isolated nodes. Furthermore, $\Omega_1(\mathbf{w}) = ||\mathbf{w}||_2^2$ is often added to force the edge weights to be smooth [66, 112].

Formally, we aim at learning a neural network $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ that maps the data term $\mathbf{y}$ to a graph representation $\mathbf{w}$ that share the same topological properties and hence belong to the same graph family $\mathcal{G}$. With training pairs $\{(\mathbf{y}_i, \mathbf{w}_i)\}_{i=1}^n$, we consider a supervised learning framework to obtain optimal $\mathcal{F}_{\boldsymbol{\theta}}^*$ such that

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{w} \sim \mathcal{G}}[\mathcal{L}(\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y}), \mathbf{w})], \tag{3.3}$$

where $\mathcal{L}(\widehat{\mathbf{w}}, \mathbf{w})$ is a loss surrogate between the estimation $\widehat{\mathbf{w}} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y})$ and the groundtruth. Once trained, the optimal $\mathcal{F}_{\boldsymbol{\theta}}^*(\cdot)$ implicitly carries topological priors and can be applied to estimate graphs from new observations. Therefore, our framework promotes learning to learn a graph topology. In the next section, we elaborate on the architecture of $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ and the design of $\mathcal{L}(\cdot, \cdot)$.

### 3.2.2 Unrolling Primal-dual Iterative Algorithm

The data-to-graph mapping $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ is modelled with unrolling layers (Section 3.2.2) inherited from a primal-dual splitting algorithm that is originally proposed to solve the model-based graph learning.

**Algorithm 2** PDS

**Input:** $\mathbf{y}$, $\gamma$, $\alpha$ and $\beta$.

1: **Initialisation:** $\mathbf{w}^{(0)} = \mathbf{0}, \mathbf{v}^{(0)} = \mathbf{0}$
2: **while** $|\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}| > \epsilon$ **do**
3: $\quad \mathbf{r}_1^{(t)} = \mathbf{w}^{(t)} - \gamma(2\beta\mathbf{w}^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{v}^{(t)})$
4: $\quad \mathbf{r}_2^{(t)} = \mathbf{v}^{(t)} + \gamma\mathcal{D}\mathbf{w}^{(t)}$
5: $\quad \mathbf{p}_1^{(t)} = \text{prox}_{\gamma,\Omega_1}(\mathbf{r}_1^{(t)}) = \max\{\mathbf{0}, \mathbf{r}_1^{(t)}\}$
6: $\quad \mathbf{p}_2^{(t)} = \text{prox}_{\gamma,\Omega_2}(\mathbf{r}_2^{(t)})$, where $\left(\text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2)\right)_i = \frac{r_{2,i} - \sqrt{r_{2,i}^2 + 4\alpha\gamma}}{2}$
7: $\quad \mathbf{q}_1^{(t)} = \mathbf{p}_1^{(t)} - \gamma(2\beta\mathbf{p}_1^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{p}_2^{(t)})$
8: $\quad \mathbf{q}_2^{(t)} = \mathbf{p}_2^{(t)} + \gamma\mathcal{D}\mathbf{p}_1^{(t)}$
9: $\quad \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{r}_1^{(t)} + \mathbf{q}_1^{(t)}$
10: $\quad \mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \mathbf{r}_2^{(t)} + \mathbf{q}_2^{(t)}$
11: **end while**
12: **return** $\widehat{\mathbf{w}} = \mathbf{w}^{(T)}$

**Algorithm 3** Unrolling Net (L2G)

**Input:** $\mathbf{y}$, $T$, $\texttt{enhancement}^{(t)} \in \{\texttt{TRUE}, \texttt{FALSE}\}$

1: **Initialisation:** $\mathbf{w}^{(0)} = \mathbf{0}, \mathbf{v}^{(0)} = \mathbf{0}$
2: **for** $t = 0, 1, \ldots, T$ **do**
3: $\quad \mathbf{r}_1^{(t)} = \mathbf{w}^{(t)} - \gamma^{(t)}(2\beta^{(t)}\mathbf{w}^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{v}^{(t)})$
4: $\quad \mathbf{r}_2^{(t)} = \mathbf{v}^{(t)} + \gamma^{(t)}\mathcal{D}\mathbf{w}^{(t)}$
5: $\quad$ if $\texttt{enhancement}^{(t)}\texttt{=TRUE}$, $\mathbf{p}_1^{(t)} = \text{TopoDiffVAE}(\mathbf{r}_1^{(t)})$; else, $\mathbf{p}_1^{(t)} = \text{prox}_{\gamma,\Omega_1}(\mathbf{r}_1^{(t)}) = \max\{\mathbf{0}, \mathbf{r}_1^{(t)}\}$.
6: $\quad \mathbf{p}_2^{(t)} = \text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2^{(t)})$, where $\left(\text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2)\right)_i = \frac{r_{2,i} - \sqrt{r_{2,i}^2 + 4\alpha^{(t)}\gamma^{(t)}}}{2}$
7: $\quad \mathbf{q}_1^{(t)} = \mathbf{p}_1^{(t)} - \gamma^{(t)}(2\beta^{(t)}\mathbf{p}_1^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{p}_2^{(t)})$
8: $\quad \mathbf{q}_2^{(t)} = \mathbf{p}_2^{(t)} + \gamma^{(t)}\mathcal{D}\mathbf{p}_1^{(t)}$
9: $\quad \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{r}_1^{(t)} + \mathbf{q}_1^{(t)}$
10: $\quad \mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \mathbf{r}_2^{(t)} + \mathbf{q}_2^{(t)}$
11: **end for**
12: **return** $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(T)} = \widehat{\mathbf{w}}$

Our model leverages the inductive bias from model-based graph learning. Specifically, we consider a special case of the formulation in Eq. (3.2) such that

$$\min_{\mathbf{w}} \; 2\mathbf{w}^\top \mathbf{y} + \mathcal{I}_{\{\mathbf{w}\geq 0\}}(\mathbf{w}) - \alpha \mathbf{1}^\top \log(\mathcal{D}\mathbf{w}) + \beta||\mathbf{w}||_2^2 \tag{3.4}$$

where $\Omega_1(\mathbf{w}) = \beta||\mathbf{w}||_2^2$ and $\Omega_2(\mathcal{D}\mathbf{w}) = -\alpha\mathbf{1}^\top \log(\mathcal{D}\mathbf{w})$, both of which are mathematically handcrafted topological priors. To solve this optimisation problem, we consider a forward-backward-forward primal-dual splitting algorithm (PDS) in Algorithm 2. The algorithm is specifically derived from Algorithm 6 in [122] by the authors in [115]. In Algorithm 2, Step 3, 5 and 7 update the primal variable that corresponds to $\mathbf{w}$, while Step 4, 6 and 8 update the dual variable in relation to node degrees. The forward steps are equivalent to a gradient descent on the primal variable (Step 3 and 7) and a gradient ascent on the the dual variable (Step 4 and 8), both of which are derived from the differentiable terms in Eq.(3.4). The backward steps (Step 5 and 6) are proximal projections that correspond to two non-differentiable topological regularisers, i.e. the non-negative constraints on $\mathbf{w}$ and the log barrier on node degrees in Eq.(3.4). The derivations for Step 5 and Step 6 in Algorithm 2 are as follows.

**Lemma 3.2.1.** *The proximal projection step for primal variable (i.e. Step 5) in Algorithm 2 is* $prox_{\gamma\Omega_1}(r_1^{(t)}) = \max\{0, r_1^{(t)}\}$.

*Proof.* From Algorithm 6 in [122], the proximal projection step for primal variable (i.e. Step 5 in Algorithm 2) is $p_1^{(t)} = \text{prox}_{\gamma\Omega_1}(r_1^{(t)})$. $\Omega_1(r_1^{(t)}) = 0$ if $r_1^{(t)} \geq 0$ else $\Omega_1(r_1^{(t)}) = \infty$. We have $\gamma\Omega_1(r_1^{(t)}) = \Omega_1(r_1^{(t)})$ and hence $\text{prox}_{\gamma\Omega_1}(r_1^{(t)}) = \text{prox}_{\Omega_1}(r_1^{(t)}) = \max\{0, r_1^{(t)}\}$. □

**Lemma 3.2.2.** *The proximal projection step for dual variable (i.e. Step 6) in Algorithm 2 is* $prox_{\gamma\Omega_2^*}(r_2^{(t)}) = \frac{r_2^{(t)} - \sqrt{r_2^{(t)2} + 4\alpha\gamma}}{2}$.

*Proof.* From Algorithm 6 in [122], the proximal projection step for dual variable for our case is $p_2^{(t)} = \text{prox}_{\gamma\Omega_2^*}(r_2^{(t)})$, where $\Omega_2^*$ is the conjugate of $\Omega_2$. Denote $h(r_2^{(t)}) = \gamma\Omega_2(\frac{r_2^{(t)}}{\gamma})$. We have $h(r_2^{(t)}) = \gamma(-\alpha 1^T \log(\frac{r_2^{(t)}}{\gamma})) = \gamma(-\alpha 1^T \log(r_2^{(t)}) + \alpha 1^T \log\gamma) = \gamma(\Omega_2(r_2^{(t)})+\text{constant})$. By the affine rule of proximal operators, $\text{prox}_h(r_2^{(t)}) = \text{prox}_{\gamma\Omega_2}(r_2^{(t)})$. Meanwhile, by Theorem 6.9 of [18], $\text{prox}_{\gamma\Omega_2}(r_2^{(t)}) = \frac{r_2^{(t)} + \sqrt{r_2^{(t)2} + 4\alpha\gamma}}{2}$. By Theorem 6.12 of [18], if $h(r_2^{(t)}) = \gamma\Omega_2(\frac{r_2^{(t)}}{\gamma})$, then $\text{prox}_h(r_2^{(t)}) = \gamma\text{prox}_{\frac{1}{\gamma}\Omega_2}(\frac{r_2^{(t)}}{\gamma})$. We thus have $\text{prox}_h(r_2^{(t)}) = \text{prox}_{\gamma\Omega_2}(r_2^{(t)}) = \gamma\text{prox}_{\frac{1}{\gamma}\Omega_2}(\frac{r_2^{(t)}}{\gamma})$. Plugging it back to the Moreau decomposition leads to $\text{prox}_{\gamma\Omega_2^*}(r_2^{(t)}) = r_2^{(t)} - \gamma\text{prox}_{\frac{1}{\gamma}\Omega_2}(\frac{r_2^{(t)}}{\gamma}) = r_2^{(t)} - \text{prox}_{\gamma\Omega_2}(r_2^{(t)}) = r_2^{(t)} - \frac{r_2^{(t)} + \sqrt{r_2^{(t)2} + 4\alpha\gamma}}{2} = \frac{r_2^{(t)} - \sqrt{r_2^{(t)2} + 4\alpha\gamma}}{2}$. □

It should be note that $\ell_1$ norm that promotes sparsity is implicitly included, i.e. $2\mathbf{w}^\top\mathbf{y} = 2\mathbf{w}^\top(\mathbf{y} - \frac{\lambda}{2}) + \lambda||\mathbf{w}||_1$ given $\mathbf{w} \geq 0$. We choose to unroll PDS as the main network architecture of $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$, since the separate update steps of primal and dual variables allow us to replace of proximal operators that are derived from other priors.

The proposed unrolling network is summarised in Algorithm 3, which unrolls the updating rules of Algorithm2 for $T$ times and is hypothetically equivalent to running the iteration steps for $T$ times. In Algorithm 3, we consider two major substitutions to enable more flexible and expressive learning.

Firstly, we replace the hyperparameters $\alpha$, $\beta$ and the step size $\gamma$ in the PDS by layer-wise trainable parameters (highlighted in blue in Algorithm 3), which can be updated through backpropagation. Note that we consider independent trainable parameters in each unrolling layer, as we found it can enhance the representation capacity and boost the performance compared to the shared unrolling scheme [90, 153].

Secondly, we provide a layer-wise optional replacement for the primal proximal operator in Step 5. When learning complex topological priors, a trainable neural network, i.e. the TopoDiffVAE (highlighted in red in Algorithm 3) takes the place to enhance the learning ability. TopoDiffVAE will be defined in Section 3.2.3. It should be noted that such a replacement at every unrolling layer might lead to overly complex model that is hard to train and might suffer from overfitting, which depends on the graph size and number of training samples. Empirical results suggest TopoDiffVAE takes effects mostly on the last few layers. Comprehensibly, at first several layers, L2G quickly search a near-optimal solution with the complete inductive bias from PDS, which provides a good initialisation for TopoDiffVAE that further promotes the topological properties. In addition, we do not consider a trainable function to replace the proximal operator in Step 6 of Algorithm 2, as it projects dual variables that are associated to node degrees but do not draw a direct equality. It is thus difficult to find a loss surrogate and a network architecture to approximate such an association.

### 3.2.3 Unrolled Networks with Topological Enhancement

The proximal operator in Step 5 of Algorithm 2 follows from $\Omega_1(\mathbf{w})$ in Eq. (3.4). Intuitively, an ideal regulariser in Eq. (3.4) would be an indicator function $\Omega_1 = \mathcal{I}_{\mathcal{G}}(\cdot)$, where $\mathcal{G}$ is a space of graph sharing same topological properties. The consequent proximal operator projects $\mathbf{w}$ onto $\mathcal{G}$. However, it is difficult to find such an oracle function. To adaptively learn topological priors from graph samples and thus achieve a better expressiveness in graph learning, we design a topological difference variational autoencoder (TopoDiffVAE) to replace the handcrafted proximal operators in Step 5 of Algorithm 2.

Intuitively, the goal is to learn a mapping between two graph domains $\mathcal{X}$ and $\mathcal{Y}$. The graph estimate after gradient descent step $\mathbf{r}_1^{(t)} \in \mathcal{X}$ lacks topological properties while the graph estimate after this mapping $\mathbf{p}_1^{(t)} \in \mathcal{Y}$ is enhanced with the topological properties that are embodied in the groundtruth $\mathbf{w}$, e.g. graphs without and with the scale-free property. We thus consider a conditional VAE such that $\mathcal{P} : (\mathbf{r}_1^{(t)}, \mathbf{z}) \to \mathbf{w}$, where the latent variable $\mathbf{z}$ has an implication of the topological difference between $\mathbf{r}_1^{(t)}$ and $\mathbf{w}$. In Algorithm 3, $\mathbf{p}_1^{(t)}$ is the estimate of the groundtruth $\mathbf{w}$ from $\mathcal{P}$. When we talk about the topological difference, we mainly refer to binary structure. This is because edge weights might conflict with the topological structure, e.g. an extremely large edge weight of a node with only one neighbour in a scale-free network. We want to find a continuous representation $\mathbf{z}$ for such discrete and combinatorial information that allows the backpropagration flow. A straightforward solution is to embed both

graph structures with graph convolutions networks and compute the differences in the embedding space. Specifically, the encoder and decoder of the TopoDiffVAE $\mathcal{P}$ are designed as follows.

**Encoder.** We extract the binary adjacency matrix of $\mathbf{r}_1^{(t)}$ and $\mathbf{w}$ by setting the edge weights less than $\eta$ to zero and those above $\eta$ to one. Empirically, $\eta$ is a small value to exclude the noisy edge weights, e.g. $\eta = 10^{-4}$. The binary adjacency matrix are denoted as $\mathbf{A}_r$ and $\mathbf{A}_w$ respectively. Following [119], the approximate posterior $q_\phi(\mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w})$ is modelled as a Gaussian distribution whose mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma} = \boldsymbol{\sigma}^2\mathbf{I}$ are transformed from the topological difference between the embedding of $\mathbf{A}_r$ and $\mathbf{A}_w$ such that

$$\boldsymbol{\delta} = f_{\text{emb}}(\mathbf{A}_w) - f_{\text{emb}}(\mathbf{A}_r) \tag{3.5a}$$

$$\boldsymbol{\mu} = f_{\text{mean}}(\boldsymbol{\delta}), \quad \boldsymbol{\sigma} = f_{\text{cov}}(\boldsymbol{\delta}), \quad \mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \tag{3.5b}$$

where $f_{\text{mean}}$ and $f_{\text{cov}}$ are two multilayer perceptrons (MLPs). The embedding function $f_{\text{emb}}$ is a 2-layer graph convolutional network[121] followed by a readout layer $f_{\text{readout}}$ that averages the node representation to obtain a graph representation, that is,

$$f_{\text{emb}}(\mathbf{A}) = f_{\text{readout}}\Big(\psi\big(\mathbf{A}\psi(\mathbf{A}\mathbf{d}\mathbf{h}_0^\top)\mathbf{H}_1\big)\Big), \tag{3.6}$$

where $\mathbf{A}$ refers to either $\mathbf{A}_r$ and $\mathbf{A}_w$, $\mathbf{h}_0$ and $\mathbf{H}_1$ are learnable parameters at the first and second convolution layer that are set as the same in embedding $\mathbf{A}_r$ and $\mathbf{A}_w$. $\psi$ is the non-linear activation function. Node degrees $\mathbf{d} = \mathbf{A}\mathbf{1}$ are taken as an input feature. This architecture is flexible to incorporate additional node features by simply replacing $\mathbf{d}$ with a feature matrix $\mathbf{F}$.

The rationale behind the specific choices of GCN and MLP in the proposed encoder lies in several key considerations. Firstly, GCNs are capable of aggregating local neighbourhood features while maintaining permutation invariance, a fundamental process for capturing the characteristics of graph topologies. This ability is particularly essential for the encoding purpose, i.e., accurately extracting topological representations of both the ground truth and estimated graphs, and simplifying the process of calculating their differences in a vector representation. Secondly, the preference for GCNs over other GNN architectures is justified by their simple yet effective approach in representing essential topologies, meeting a core requirement of this task. Admittedly, more complex architectures like Transformers or advanced GNN variants might offer additional expressive power, they also introduce greater

computational complexity and a higher risk of overfitting. This is particularly relevant considering the need to train the encoder on limited training samples of specific properties, where a simpler yet effective GCN architecture provides an optimal balance. While GCNs are excellent for processing graph-structured data, the task of encoding mean and variance may not directly involve the graph structure, but rather the features extracted from it, making MLPs more suitable for such tasks where the input is a regular, fixed-size feature vector. It is particularly suitable for encoding the mean and variance within the latent space proposed for TopoDiffVAE due to its capability to map high-dimensional data to a lower-dimensional space efficiently.

**Latent variable sampling.** We use Gaussian reparameterisation trick [119] to sample $\mathbf{z}$ such that $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We also restrict $\mathbf{z}$ to be a low-dimensional vector to prevent the model from ignoring the original input in the decoding step and degenerating into an auto-encoder. Meanwhile, the prior distribution of the latent variable is regularised to be close to a Gaussian prior $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to avoid overfitting. We minimise the Kullback–Leibler divergence between $q_\phi(\mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w})$ and the prior $p(\mathbf{z})$ during training as shown in Eq.(3.8).

**Decoder.** The decoder is trained to reconstruct the groundtruth graph representations $\mathbf{w}$ (the estimate is $\mathbf{p}_1^{(t)}$) given the input $\mathbf{r}_1^{(t)}$ and the latent code $\mathbf{z}$ that represents topological differences. Formally, we obtain an augmented graph estimation by concatenating $\mathbf{r}_1^{(t)}$ and $\mathbf{z}$ and then feeding it into a 2-layer MLP $f_{\text{dec}}$ such that

$$\mathbf{p}_1^{(t)} = f_{\text{dec}}(\text{CONCAT}[\mathbf{r}_1^{(t)}, \mathbf{z}]). \tag{3.7}$$

**Loss Function.** By stacking the unrolling module and TopoDiffVAE, we propose an end-to-end training scheme to learn an effective data-to-graph mapping $\mathcal{F}_{\boldsymbol{\theta}}$ that minimises the loss from both modules such that

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\mathbf{w}\sim\mathcal{G}}\left[\sum_{t=1}^{T}\left\{\tau^{T-t}\frac{||\mathbf{w}^{(t)}-\mathbf{w}||_2^2}{||\mathbf{w}||_2^2} + \beta_{\text{KL}}\text{KL}\Big(q_\phi(\mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w})||\mathcal{N}(\mathbf{0}, \mathbf{I})\Big)\right\}\right] \tag{3.8}$$

where $\mathbf{w}^{(t)}$ is the output of the $t$-th unrolling layer and $\mathbf{r}_1^{(t)}$ is the output of Step 3 in Algorithm 3, $\tau \in (0, 1]$ is a loss-discounting factor added to reduce the contribution of the normalised mean squared error of the estimates from the first few layers. This is similar to the discounted cumulative loss introduced in [190]. The first term represents the graph reconstruction loss, while the second term is the KL regularisation term in the

TopoDiffVAE module. When the trade-off hyper-parameter $\beta_{\text{KL}} > 1$, the VAE framework reduces to $\beta$-VAE [32] that encourages disentanglement of latent factors $\mathbf{z}$. The trainable parameters in the overall unrolling network $\mathcal{F}_{\boldsymbol{\theta}}$ are $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{unroll}}, \boldsymbol{\theta}_{\text{TopoDiffVAE}}\}$, where $\boldsymbol{\theta}_{\text{unroll}} = \{\alpha^{(0)}, \ldots, \alpha^{(T-1)}, \beta^{(0)}, \ldots, \beta^{(T-1)}, \gamma^{(0)}, \ldots, \gamma^{(T-1)}\}$ and $\boldsymbol{\theta}_{\text{TopoDiffVAE}}$ includes the parameters in $f_{\text{emb}}$, $f_{\text{mean}}$, $f_{\text{cov}}$ and $f_{\text{dec}}$. The optimal mapping $\mathcal{F}_{\boldsymbol{\theta}}^{\star}$ is further used to learn graph topologies that are assumed to have the same structural properties as training graphs.

## 3.3 Experiments

**General settings.** Random graphs are drawn from the Barabási-Albert (BA), Erdös-Rényi (ER), stochastic block model (SBM) and Watts–Strogatz (WS) model to have scale-free, random, community-like, and small-world structural properties respectively. The parameters of each network model are chosen to yield an edge density in $[0.05, 0.1]$. Edge weights are drawn from a log-normal distribution $\log(\mathbf{W}_{ij}) \sim \mathcal{N}(0, 0.1)$. The weighted adjacency matrix is set as $\mathbf{W} = (\mathbf{W} + \mathbf{W})^{\top}/2$ to enforce symmetry. We then generate graph-structured Gaussian samples (similar to [129]) with

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^{-1}), \ \mathbf{K} = \mathbf{L} + \sigma^2 \mathbf{I}, \ \text{where } \sigma = 0.01. \tag{3.9}$$

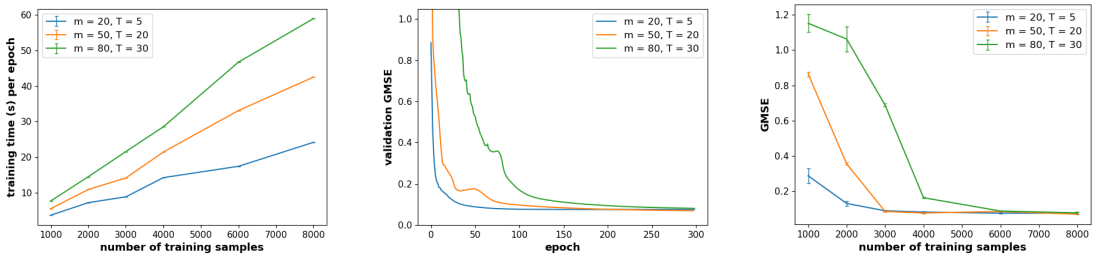
The out-of-sample normalised mean squared error for graph recovery (GMSE) and area under the curve (AUC) are reported to evaluate the prediction of edge weights and the recovery of the binary graph structure. Specifically, GMSE $= \frac{1}{n}\sum_{i=1}^{n} ||\hat{\mathbf{w}}_i - \mathbf{w}_i||_2^2 / ||\mathbf{w}_i||_2^2$. The mean and 95% confidence interval are calculated from 64 test samples.

We include the following baselines in the comparison: 1) **primal-dual splitting algorithm** [112, 122] that is unrolled into a neural network in our model; 2) the iterative algorithm of **ADMM** (details in Appendix). For PDS and ADMM, we tune the hyperparameters on training set via grid search based on GMSE. For the state-of-the-art models that share the idea of learning to optimise, we compare against 3) **GLAD**[1] [190] and 4) **Deep-graph**[2] [19] that are previously introduced in Section 2.1.1. Besides the proposed 5) **learning to learn graph topologies (L2G)** framework that minimises the objective in Eq.(3.8) to obtain a data-to-graph mapping, we also evaluate the performance of ablation models without TopoDiffVAE in replacement of

[1]https://github.com/Harshs27/GLAD
[2]https://github.com/eugenium/LearnGraphDiscovery

the proximal operator. This model is equivalent to unrolling PDS directly while allowing parameters varying across the layers, which is referred to as 6) **Unrolling** in the following sections. We also consider 7) **Recurrent Unrolling** with shared parameters across the layers, i.e $\alpha^{(0)} = \alpha^{(1)} = \cdots \alpha^{(T-1)}$, $\beta^{(0)} = \beta^{(1)} = \cdots \beta^{(T-1)}$ and $\gamma^{(0)} = \gamma^{(1)} = \cdots \gamma^{(T-1)}$. The loss function for Unrolling and Recurrent Unrolling reduces to the first term in Eq.(3.8), i.e. without the KL divergence term of TopoDiffVAE. We also release the code for implementation [3].

For the experiments in Section 3.3, we train L2G and Unrolling with the Adam optimiser [118]. The learning rate is exponentially-decayed from an initial value of $10^{-2}$ at a rate of 0.95. The number of training samples is changed with the graph size $m$. For $m = 20$, we use 4000 for training and 1000 for validation. Figure 3.2c shows how many training samples are needed (at different graph size). All the metrics reported in the chapter are computed from 64 test samples that are additionally generated. The batch size is set to 32. We run all the experiments on a remote machine of Amazon EC2 G4dn-2xlarge instance[4]. Figure 3.2a shows training time per epoch with different number of samples. Once trained, the computational cost of applying L2G to learn a graph from new observations is negligible. The source code is included in the supplementary file.




(a) training time per epoch in seconds v.s. number of training samples

(b) validation GMSE v.s. number of epochs

(c) test GMSE v.s. number of training samples

Figure 3.2: Training L2G for different size of graphs $m$ with different number of unrolls $T$

**Ability to recover graph topologies with specific structures.** We evaluate the accuracy of recovering groundtruth graphs of different structural properties from synthetic data. We train an L2G for each set of graph samples that are generated

[3]https://github.com/xpuoxford/L2G-neurips2021
[4]https://aws.amazon.com/ec2/instance-types/g4/

Table 3.1: GMSE* in graph reconstruction

| model/graph type | Scale-free (BA) | Random sparse (ER) | Community (SBM) | Small-world (WS) |
|---|---|---|---|---|
| **Iterative algorithm:** | | | | |
| ADMM | 0.4094 ± .0120 | 0.3547 ± .0120 | 0.3168 ± .0226 | 0.2214 ± .0151 |
| PDS | 0.4033 ± .0072 | 0.3799 ± .0085 | 0.3111 ± .0147 | 0.2180 ± .0117 |
| **Learned to optimise:** | | | | |
| GLAD[190]* (NMSE) | 1.1230 ± .1756 | 1.1365 ± .1549 | 1.4231 ± .2625 | 0.9999 ± .0001 |
| Deep-graph** [19] | 0.8423 ± .0026 | 0.8179 ± .0269 | 0.8931 ± .0103 | 0.8498 ± .0017 |
| **Proposed:** | | | | |
| Recurrent Unrolling | 0.3018 ± .0080 | 0.2885 ± .0075 | 0.2658 ± .0108 | 0.2007 ± .0081 |
| Unrolling | 0.1079 ± .0047 | 0.0898 ± .0067 | 0.1199 ± .0064 | 0.1028 ± .0073 |
| L2G | **0.0594 ± .0044** | **0.0746 ± .0042** | **0.0735 ± .0051** | **0.0513 ± .0060** |

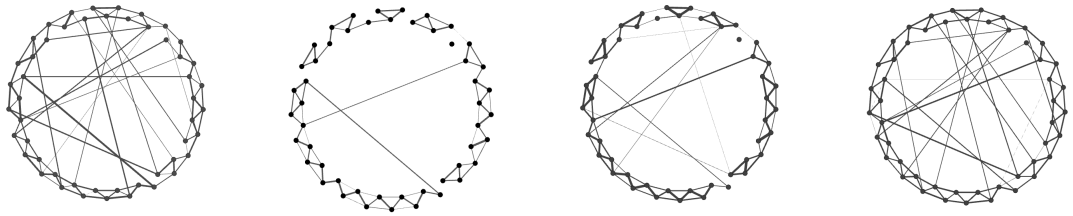
* For GLAD, we report test NMSE instead of GMSE so as to follow their original setting in [190]. GLAD is sensitive to the choice of $\sigma^2$ when generating samples (see Figure 3.6).
** GMSE may not favour Deep-graph as it learns binary structures only (see AUC in Table 3.2).

Table 3.2: Structural Metrics of recovered binary graph structure

| metric / model | groundtruth | PDS | Deep-Graph [19] | Unrolling | L2G |
|---|---|---|---|---|---|
| **Scale-free (BA):** | | | | | |
| AUC | - | 0.934±.009 | 0.513±.017 | 0.993±.001 | **0.999±.000** |
| KS test score* | 95.31% | 65.63% | 59.38% | 93.75% | **95.31%** |
| **Community (SBM):** | | | | | |
| AUC | - | 0.926±.006 | 0.586±.021 | 0.989±.002 | **0.993±.002** |
| community score | 0.463±.002 | 0.481±.002 | 0.343±.006 | 0.458±.001 | **0.469±.003** |
| **Small-world (WS):** | | | | | |
| AUC | - | 0.913±.007 | 0.529±.010 | 0.984±.005 | **0.991±.000** |
| average shortest path | 2.334±.028 | 2.656±.204 | 1.136±.008 | 2.328±.040 | **2.331±.041** |
| clustering coefficient | 0.323±.018 | 0.514±.035 | 0.906±.004 | 0.433±.016 | **0.382±.014** |

* KS test score is the percentage of test graphs whose degree sequence passes the Kolmogorov -Smirnov test for goodness of fit of a power-law distribution, i.e. $p$-value ¿ 0.05.

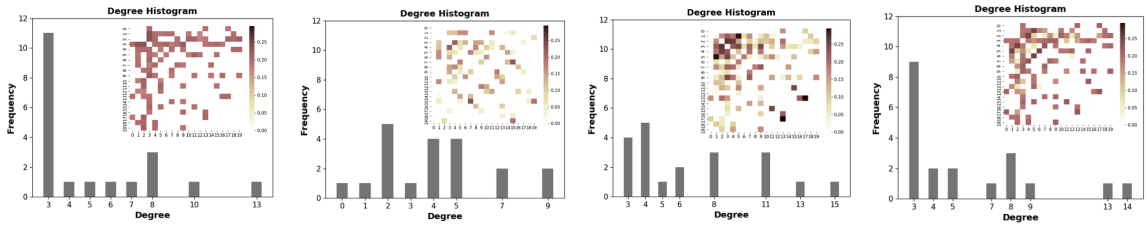
from one particular random graph model, and report GMSE between the groundtruth graphs and the graph estimates on test data in Table 3.1.



(a) groundtruth (b) PDS (c) Unrolling (d) L2G

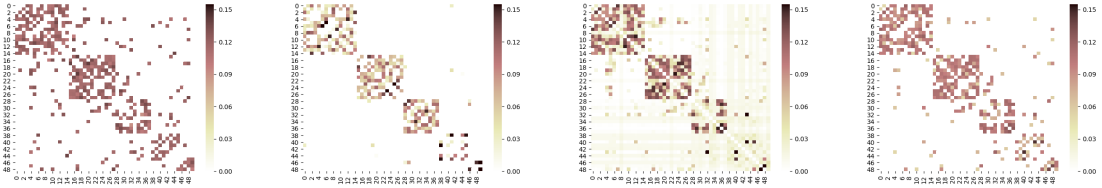
Figure 3.3: The ability to recover small-world network

The proposed L2G reduces the GMSE by over 70% from the iterative baseline PDS on all types of graphs. Significant drops from PDS to Recurrent Unrolling and to Unrolling are witnessed. The scenario of recovering the scale-free networks sees biggest improvement on GMSE, as they are relatively hard to learn with handcrafted iterative rules. As shown in Figure 3.4 in Appendix, the power-law degree distribution




(a) groundtruth (b) PDS (c) Unrolling (d) L2G

Figure 3.4: The ability to recover scale-free networks



(a) groundtruth (b) PDS (c) Unrolling (d) L2G

Figure 3.5: The ability to recover community-structured networks

associated with such networks is preserved in graph estimates from L2G, while PDS and Unrolling have difficulties to recognise small-degree nodes and nodes that are connected to many other nodes, respectively. PDS and Deep-Graph also fail the KS test, which measures how close the degrees of graph estimates follow a power-law distribution (see Table 3.2 where a $p$-value less than 0.05 rejects the null hypothesis that the degrees follow a power-law distribution). We also test the models on SBM and WS graphs. We deliberately increase the rewiring probability in generating WS graphs in Figure 3.3 and add inter-community edges on SBM graphs in Figure 3.5 to increase the learning difficulties in both cases. PDS may effectively detect the $k$-NN structure behind the construction of WS and the communities (block diagonal structure of the adjacency matrix) in SBM, but largely ignored the new edges from the procedures above. By contrast, L2G can precisely predict the additional rewired edges and inter-community edges.

**Ability to recover binary graphs** Deep-graph [19] is a state-of-the-art learning-based method, but its output is binary graph structure that is different from our setting. In order to have a fair comparison, we redo the experiments with another synthetic dataset with binary groundtruth graphs. The results are shown in Table 3.3 as an addition to the results of recovering weighted graph in the main body (see Table 3.1 and Table 3.2). Both results lead to the same conclusion that Deep-graph

Table 3.3: A table of GMSE and Structural Metrics in recovering binary graphs

| metric / model | groundtruth | Deep-graph | L2G |
|---|---|---|---|
| **Scale-free (BA):** | | | |
| GMSE | - | $0.802 \pm .003$ | $0.060 \pm .002$ |
| AUC | - | $0.565 \pm .101$ | $0.999 \pm .000$ |
| KS test score | 96.15% | 68.32% | 96.15% |
| **Community (SBM):** | | | |
| GMSE | - | $0.901 \pm .016$ | $0.080 \pm .010$ |
| AUC | - | $0.701 \pm .088$ | $0.992 \pm .001$ |
| community score | $0.472 \pm .002$ | $0.311 \pm .006$ | $0.479 \pm .005$ |
| **Small-world (WS):** | | | |
| GMSE | - | $0.873 \pm .010$ | $0.056 \pm .004$ |
| AUC | - | $0.519 \pm .023$ | $0.997 \pm .000$ |
| average shortest path | $2.23 \pm .028$ | $1.111 \pm .008$ | $2.225 \pm .041$ |

[3] has a less satisfactory performance compared to the proposed L2G.

**Comparisons with SOTA.** GLAD[190] and Deep-graph[19] are the few state-of-the-art learning-based methods worth comparing to, but there are notable differences in their settings that affect experimental results. As discussed in Section 2.1.1, the objective of GLAD[190] is to learn the precision matrix where the diagonal entries are considered. The output of Deep-graph[19] is an undirected and unweighted (i.e. binary) graph structure. By comparison, the product of generic graph learning adopted in our work is a weighted adjacency matrix, where the diagonal entries are zeros due to no-self-loop convention.

For a fair comparison, we provide the following remarks. Firstly, we stick to their original loss function NMSE for training GLAD[190] and reporting the performance on test data in Table 3.1. The difference between GMSE and NMSE is that the latter considers the $\ell_2$ loss on diagonal entries of the precision matrix $\mathbf{\Theta}$ in GLAD[190]. Note that GMSE equals NMSE when applied to adjacency matrices which are the focus of the present chapter. The experimental results show a far inferior performance of GLAD[190] if their NMSE is replaced with our GMSE.

$$\text{GMSE} = \frac{1}{n}\sum_{i=1}^{n}\frac{||\hat{\mathbf{w}}_i - \mathbf{w}_i||_2^2}{||\mathbf{w}_i||_2^2}, \quad \text{NMSE(GLAD [190])} = \frac{1}{n}\sum_{i=1}^{n}\frac{||\hat{\mathbf{\Theta}}_i - \mathbf{\Theta}_i||_2^F}{||\mathbf{\Theta}_i||_2^F} \tag{3.10}$$

Secondly, a potential limitation of GLAD[190] is that it typically requires good conditioning of the precision matrix, i.e. a large $\sigma^2$ in Eq.(3.9). On the other hand, in the graph signal processing and graph learning literature [66, 112], smooth graph signals follow a Gaussian distribution where the precision matrix is taken as the graph Laplacian matrix, which is a singular matrix with at least one zero eigenvalue. A large $\sigma^2$ is more likely to destroy the graph topological structure behind the

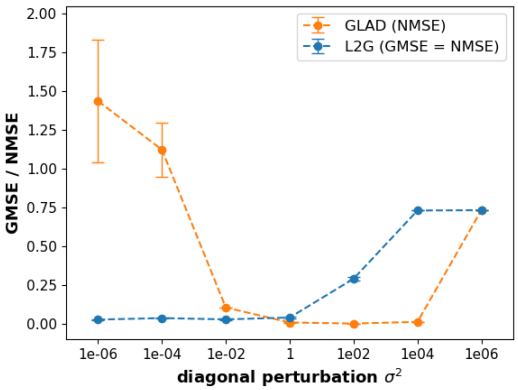
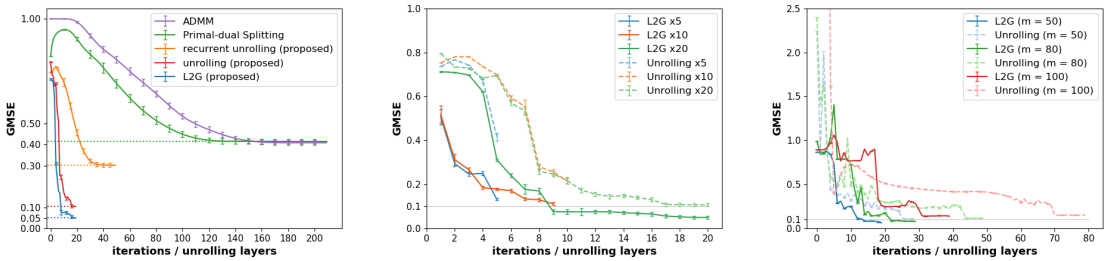



Figure 3.6: GMSE v.s. $\sigma^2$ for L2G/GLAD.

data. Given that our objective is to learn a graph topology with non-negative edge weights, we report results in Table 3.2 using a small $\sigma = 0.01$. In Figure 3.6, we see that GLAD[190] performs badly with small diagonal perturbation $\sigma^2$, while our L2G method performs stably with different $\sigma^2$. Admittedly, GLAD[190] outperforms L2G in terms of GMSE/NMSE under $1 \leq \sigma^2 \leq 10^4$ in Figure 3.6. However, this could be due to the fact that during training GLAD makes use of the groundtruth precision matrix which contains information of $\sigma^2$, while L2G does not require and hence is not made aware of this information.

Thirdly, we generate another synthetic dataset with binary groundtruth graphs to have a fairer comparison with Deep-graph[19] and report the experimental results in Table 3.3 of Appendix 3.3. We follow the original setting in the chapter for training and inference. For both binary graphs or weighted graphs, Deep-graph[19] has achieved inferior performance, showing that CNN has less expressiveness than the inductive bias introduced by model-based graph learning.

**Analysis of model behaviour.** Figure 3.7a shows the proposed models require far less iterations to achieve a big improvement on accuracy than the baseline models. Here, the number of iterations is referred to as the number of unrolling layers $T$ in Algorithm 3. This means the proposed models can be used to learn a graph from data with significant saving in computation time once trained. In Figure 3.7b, we vary $T$ in both L2G and Unrolling. With the same $T$, L2G with TopoDiffVAE replacing the proximal operator in Unrolling achieves a lower GMSE at each layer. This suggests that TopoDiffVAE can help the unrolling algorithm to converge faster to a good solution (in this case a graph with desired structural properties).




(a) compared with baselines (b) different unroll layers (c) learning graphs with different sizes

Figure 3.7: Convergence of number of iterations

In Figure 3.7c, we train L2G and Unrolling on different sizes of graphs, where $m$ is the number of nodes. A larger graph requires more iterations to converge in both models, but the requirement is less for L2G than for Unrolling. One advantage of applying Unrolling is that it can be more easily transferred to learn a graph topology of different size. This is because the parameters learned in Unrolling are not size-dependent. Figure 3.7c shows that with a large number of iterations (which will be costly to run), Unrolling can learn a effective data-to-graph mapping with a little sacrifice on GMSE. However, L2G significantly outperforms Unrolling in terms of preserving topological properties, which are hard to be reflected by GMSE. Unrolling might fail at detecting edges that are important to structural characteristics. For example, as shown in Figure 3.3c and 3.3d in Appendix, rewired edges in WS graphs are missing from the results of Unrolling but successfully learned by L2G.

In Figure 3.7c of Section 3.3, we show that L2G performs better in learning graphs with $m = 100$ nodes. The main challenge of scaling to large graphs (i.e. millions of nodes) lies in the memory required for large training samples including both graphs and data on nodes. Fortunately, we show that the proposed methods can be pre-trained and then transferred to real-world data (see the real-world applications in Section 3.3). We can use a similar trick to pretrain an Unrolling model and then transferred to evaluate a graph with more nodes. This is because the learnable parameters of the Unrolling model ($\boldsymbol{\theta}_{\text{unrolling}}$) are not dependent on graph size. More extensive tests on this is one of the main directions for future work.

## 3.4 Conclusions

In this chapter, we utilise the technique of learning to optimise to learn graph topologies from node observations. Our method is able to learn a data-to-graph mapping,

which leverages the inductive bias provided by a classical iterative algorithm that solves structural-regularised Laplacian quadratic minimisation. We further propose a topological difference variational autoencoder to enhance the expressiveness and flexibility of the structural regularisers. Limitations of the current method include its focus on unrolling an algorithm for learning undirected graph, and scalability to learning large-scale graphs. We leave these directions as future work.

# Chapter 4

# Financial Application I: Network Momentum Strategies

## Contents



## 4.1 Introduction

Financial networks represent interconnections among financial entities such as institutions, markets and assets, and are constructed to facilitate the analysis of various downstream tasks. These tasks primarily focus on systemic risk assessment and market structure analysis, and more recently on alpha research and portfolio optimisation [3, 132]. Network momentum is a type of network alpha signals, which is based on the idea that the future return of one company in the financial network can be predicted

from the past returns of companies to which it is connected, a phenomenon known as momentum spillover [3].

The construction of financial networks is crucial to the success of network momentum signals. Nodes often represent individual stocks, while edges capture the relations between them and are constructed based on various financial or economic factors. These factors include, but are not limited to, industry or business similarity [156, 28], geographical proximity [164, 123] news co-mentions [209] and supply-demand links [48, 151]. However, the construction of these networks often relies on acquiring expensive databases from domain experts with deep understanding of finance and economics, making it costly for small-sized, non-profit or academic institutions. Alternatively, we may adopt the reasonable assumption that momentum spillover is subtly echoed in the historical pricing data, which are more readily accessible. Therefore, if an effective model could be developed to learn the relationship between companies from readily available data, it could significantly lessen the dependence on costly data acquisition and specialised financial knowledge. Nevertheless, creating such a model remains a significant open challenge in the field of machine learning and quantitative finance.

Another limitation of the literature is that the construction of financial networks or graphs[1] is often detached from the subsequent downstream tasks. For example, analysts tend to focus more on economic justification when constructing these graphs, giving less consideration to whether they would lead to optimal portfolio performance. However, it is possible that the method of graph construction, such as deducing a custom sales ratio formula to determine the connection strength between two companies [48], may not help with achieving optimal portfolio performance. In other words, the graph may not be effectively optimised with respect to portfolio performance. This underscores the need for a task-specific graph construction framework.

In response to these considerations, we propose an end-to-end machine learning framework capable of simultaneously learning financial graphs and optimising the trading signals from momentum features built from these networks. Our framework leverages the concept of *learning to learn graph topologies* (L2G) [172], which has been introduced in Chapter 3. Traditional graph learning methods typically employ convex optimisation to solve for an optimal graph given observed data on nodes [68, 66]. L2G reformulated this process into learning a parametric mapping, where node features

[1]In this chapter, we use *graph* and *network* interchangeably, and *neighbours* and *peers* interchangeably to represent the assets one is connected to. In the literature, the connected assets are also sometimes referred to as *linked* assets.

are input and the graph serves as the output, by unrolling the optimisation algorithm into a forward propagation of a neural network. Taking advantage of the flexibility in neural architecture design and training, we propose to incorporate into L2G an additional layer for constructing network momentum features, and train the model with loss being portfolio performance metrics in an end-to-end fashion.

We test the proposed strategy, *Learning to Learn Financial Networks for Optimising Momentum Strategies* (L2GMOM), on 64 continuous future contracts of marco assets, spanning commodities, equities, fixed income funds, and foreign currencies. In a simplified portfolio construction using real market data of these future contracts, we have observed a Sharpe ratio of 1.74 during a 20-year backtest period from 2000 to 2020. It is important to acknowledge that while these results are promising, they necessitate careful considerations when applied in real-world trading scenarios, particularly with regard to potential associated costs. Another noteworthy aspect emerging from our experiments is the ability to uncover momentum spillover of marco assets through the learned financial graphs. This capability enhances the interpretability of our methodology for portfolio construction. Furthermore, our model demonstrates computational efficiency, enabling rapid inference in milliseconds and eliminating the daily need for graph optimisation.

### 4.1.1 Contributions

The major contributions of our work are threefold: Firstly, we present a novel methodology that uses easily accessible market data to infer a network to describe the momentum spillover between marco assets in the futures market. This approach eliminates the costly requirement for alternative datasets and reduces human bias. Secondly, we propose an end-to-end learning framework that is capable of both inferring financial networks and simultaneously optimising the downstream task of portfolio construction. This ensures the financial networks are directly optimised in relation to portfolio performance.

## 4.2 Preliminary

### 4.2.1 Individual Momentum Strategy

The concept of individual asset momentum, proposed by [156], explores the profitability by longing past winner assets and shorting past underperformers. A time-series

(TSMOM) momentum can be therefore constructed, with the daily return defined as

$$r_{t:t+1}^{\text{portfolio}} := \frac{1}{N_t} \sum_{i=1}^{N_t} x_{i,t} \frac{\sigma_{\text{tgt}}}{\sigma_{i,t}} r_{i,t:t+1} \tag{4.1}$$

where $N_t$ is the number of assets at day $t$, $r_{i,t:t+1}$ is the daily return of asset $i$. With the target annualised volatility $\sigma_{\text{tgt}}$, the asset return is scaled by its annualised realised volatility $\sigma_{i,t}$. Here, $\sigma_{\text{tgt}} = 0.15$ following literature standards [138], and $\sigma_{i,t}$ is estimated from an exponential weighted moving standard deviation with a 60-day span on daily returns. The position, or trading signal $x_{i,t} \in [-1, 1]$ at day $t$ for asset $i$, specifies the strategy. The core of momentum strategies is the construction of the trading signal $x_{i,t}$. It often consists of two steps - trend estimation and position sizing. We illustrate this with the following examples.

**Annual Return** The authors in [157] proposed to use past 1-year asset return $r_{i,t-252:t}$ as a trend estimation, and the direction as the position $x_{i,t} = \text{sgn}(r_{i,t-252:t})$.

**MACD** The authors in [17] constructed trend estimation using the volatility normalised moving average crossover divergence (MACD) indicators such that

$$\text{MACD}(i, t, S, L) = m(i, t, S) - m(i, t, L) \tag{4.2}$$

$$\text{MACD}_{\text{norm}}(i, t, S, L) = \frac{\text{MACD}(i, t, S, L)}{\text{std}(p_{i,t-63:t})} \tag{4.3}$$

$$y_{i,t}(S, L) = \frac{\text{MACD}_{\text{norm}}(i, t, S, L)}{\text{std}(\text{MACD}_{\text{norm}}(i, t - 252, S, L))} \tag{4.4}$$

$$x_{i,t} = \frac{1}{3} \sum_{k=1}^{3} \phi(y_{i,t}(S_k, L_k)) \tag{4.5}$$

where $m(i, t, J)$ is the exponentially weighted moving average of the prices of asset $i$ at time $t$, with a scale $J$ such that the smoothing factor $\alpha = 1/J$. Here $\text{std}(p_{i,t-63:t})$ is the 63-day rolling standard deviation of the prices of asset $i$. Similarly, the denominator of Eq.(4.4) is the 252-day rolling standard deviation of the normalised MACD. A typical choice for time span is $(S_k, L_k) \in \{(8, 24), (16, 48), (32, 96)\}$ and $\phi(y) = \frac{y \exp(-y^2/4)}{0.89}$ is a position scaling function.

**Machine Learning-based TSMOM** Recent literature managed to construct the individual time-series momentum from machine learning models. The authors in [138] proposed a general supervised learning framework such that

$$x_{i,t} = \text{sgn}(y_{i,t}) \quad \text{and} \quad y_{i,t} = f_{\boldsymbol{\theta}}(\mathbf{u}_{i,t}). \tag{4.6}$$

It can also directly model the position:

$$x_{i,t} = f_{\boldsymbol{\theta}}(\mathbf{u}_{i,t}) \tag{4.7}$$

The model specifications for $f$ are diverse and often considered in relation to the nature of the input feature $\mathbf{u}$. The authors in [138] compared linear models and deep learning models, such as Lasso Regression, MLP, WaveNet, and LSTM, using input features such as the MACD indicators and volatility-scaled returns from diverse lookback windows. In a similar vein, Wood et al. [214] considered an attention-based deep learning model. Tan et al. [200] proposed advanced deep learning to effectively model for time-series and cross-sectional momentum.

### 4.2.2 Network Momentum Strategy

Network momentum strategy takes into account the interconnections of assets, assuming an asset's future performance can be influenced by the past performance of its linked assets. A network or graph is defined to represent economic links between assets, modelled by a graph adjacency matrix $\mathbf{A}_t$ at time $t$. A general framework of network momentum strategies starts by defining a *network* or *graph* to represent certain economic links between two assets. The network is mathematically modelled by a graph adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$ between $N_t$ assets at time $t$, where the entries $a_{ii,t} = 0$ and $a_{ij,t} \geq 0$. If $a_{ij,t} > 0$, there exists an economic link that shows certain similarity between asset $i$ and asset $j$. Some studies [156, 28] defined binary graph such that $a_{ij,t} \in \{0, 1\}$, while others [222, 3] also quantified the strength of the interconnections using edge weights.

The network momentum of asset $i$ at time $t$ is defined through the average of trend estimation propagated from its connected assets in $\mathbf{A}_t$:

$$y_{i,t} = \frac{1}{|\mathcal{N}_t(i)|} \sum_{j \in \mathcal{N}_t(i)} a_{ij,t} y_{j,t} \tag{4.8}$$

where $\mathcal{N}_t(i)$ is the neighbourhood set of asset $i$ at time $t$ containing all its connected assets. In the literature, different economic links were explored. Some examples of $\mathbf{A}$ include:

- *Industry Linkage.* $a_{ij,t} = 1$ if stock $i$ and $j$ belong to the same industry, based on industrial classification codes such as two-digit SIC [156] and GICS [28].
- *Industrial Segment.* A conglomerate firm contain multiple industrial segments that corresponds to standalone firms. Cohen and Lou [50] predicted return of the target conglomerate from its linked standalone firms.

- *Supply-demand Relations* [151, 222]. Cohen and Frazzini [48] define a customer link $a_{ij,t} = 1$ for a target firm $i$ if it sells more then 10% its products or services to the firm $j$. Yamamoto et al. [222] further modified the binary link by using centrality statistics from network theory as edge weights to show the relevant importance of customers. Menzly and Ozbas [151] considered supplier links together with customer links, where the edge weights are defined as the flow of goods.
- *Geographic Location.* $a_{ij,t} = 1$ if two companies have headquarters in the same location [164, 123].
- *Patent Similarity.* Lee et al. [132] define a technology proximity score $a_{ij,t}$ to measure the similarity of patent portfolios between two firms.
- *News co-occurrence*: Wan et al. [209] defines $a_{ij,t}$ as the cosine similarity of the sentiment representations from the news of two firms.
- *Analyst co-coverage.* Ali and Hirshleifer [3] defined $a_{ij,t}$ as the number of sell-side analysts who cover stock $i$ and $j$ in the same report at time $t$. The two firms that are both covered by a sell-side analyst share similar economic and business fundamentals. Ali and Hirshleifer [3] finds that such a connected-firm momentum factor generates a monthly alpha of 1.68% and the co-coverage network more effectively encapsulates the information contained within other links such as industry, geographic location and customer/supplier chain.

While the above networks can effectively capture some economic similarity between companies and potentially generate alpha signals, they often rely on expensive databases from domain experts with deep understanding of finance. In this chapter, we propose learning the graph adjacency matrix $\mathbf{A}_t$ from readily accessible historical returns that directly optimises momentum strategy.

### 4.2.3 Learning to Learn Graph Topologies

In Chapter 3, we introduce a deep learning model *Learning to Learn Graph Topologies* (L2G). L2G reformulates the optimisation problem of GSP-based graph learning [66, 112, 148] into a task of learning a parametric mapping. A neural network with a forward propagation step is proposed by unrolling the iterative PDS optimisation steps (as in Algorithm 3). The hyperparameters that influence the graph's topological characteristics transform into learnable parameters, which are trained using a loss function comparing graph estimates against ground truth graphs.

In the context of this Chapter, each financial asset represents a node, and their historical pricing data act as node observations. With graph learning, we can reveal

the interconnections between assets. Learning to Learn Graph Topologies (L2G) provides further inspiration, as we can now use a loss function associated with portfolio performance to determine the optimal graph structure.

## 4.3 Learning to Learn Network Momentum

### 4.3.1 L2GMOM

In this section, we introduce *Learning to Learn Financial Networks for Optimising Momentum Strategies* (L2GMOM), a solution specifically designed to learn financial networks from readily available data and optimise them directly for an ideal portfolio performance.

As introduced in Section 4.2.3, L2G reformulates optimisation-based graph learning into an unrolling neural network. By leveraging the inherent modularity of neural networks, where different layers can be easily stacked for forward propagation, we propose to incorporate an additional layer into L2G for directly constructing network momentum. Given a feature matrix $\mathbf{V}_t$ captured from $N_t$ assets at time $t$, the initial step of L2GMOM involves employing the L2G layer (Algorithm 3), as defined in Eq.(4.9b), to learn a graph adjacency matrix $\mathbf{A}_t$. This matrix illustrates the interconnections among the assets. Subsequently, $\mathbf{A}_t$ has a normalisation step, as outlined in Eq.(4.9c). The normalisation aims to offset the dominance of nodes with many connections. In line with the machine learning-based TSMOM approach [138] and traditional network momentum in Eq.(4.8), we propose a linear layer to estimate the momentum trend of asset $i$, derived from a linear combination of individual momentum features $\mathbf{U}_t$ of its connected assets in $\tilde{\mathbf{A}}_t$. $\tilde{\mathbf{A}}_t\mathbf{U}_t$ has an interpretation of a network momentum features and the learned parameters $\boldsymbol{\theta}$ are the weights of combining different momentum features. It is worth mentioning that $\tilde{\mathbf{A}}_t$ from Eq.(4.9c) contains only non-negative edges, as a result of enforcing the constraints of Eq.(3.4). In summary, these are the forward propagation rules for the L2GMOM neural network:

$$\mathbf{h}_t = \text{vech}(\mathbf{H}_t), \text{ where } \mathbf{H}_{ij,t} = ||\mathbf{V}_{i,:,t} - \mathbf{V}_{j,:,t}||_2^2 \tag{4.9a}$$

$$\mathbf{A}_t = \mathbf{L2G}(\mathbf{h}_t, L, \texttt{enhancement}^{(t)} = \texttt{False};\ \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \tag{4.9b}$$

$$\tilde{\mathbf{A}}_t = \mathbf{D}_t^{-1/2}\mathbf{A}_t\mathbf{D}_t^{-1/2}, \text{ where } \mathbf{D}_{ii,t} = \sum_j \mathbf{A}_{ij,t},\ \mathbf{D}_{ij,t} = 0 \tag{4.9c}$$

$$(\mathbf{L2GMOM}) \quad \mathbf{y}_t = \tilde{\mathbf{A}}_t\mathbf{U}_t\boldsymbol{\theta} + b \tag{4.9d}$$

where $\mathbf{y}_t$ is the model output, a $N_t$-dimensional vector of trend estimations for every assets, such that $\mathbf{y}_t = [y_{1,t}, y_{2,t}, \ldots, y_{N_t,t}]^T$. We further take the trading positions $\mathbf{x}_t = \text{sgn}(\mathbf{y}_t)$ for constructing a network momentum strategy. The trainable parameters include $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}, b)$.

Taking one step further, we also model the direct position $\mathbf{x}_t$ instead of the trend estimation $\mathbf{y}_t$. We name the model **L2GMOM_SR**, since we further use Sharpe Ratio (SR) to optimise for the positions, which are consistent with the literature [138]. Since the resulting positions $\mathbf{x}_t$ lie within the range of $[-1, 1]$ in TSMOM, we simply introduce a non-linear activation function, tanh, in the linear layer to ensure the same bahaviour:

$$(\textbf{L2GMOM\_SR}) \quad \mathbf{x}_t = \tanh\Big(\tilde{\mathbf{A}}_t \mathbf{V}_t \boldsymbol{\theta} + b\Big) \tag{4.10}$$

The use of L2GMOM_SR showcases our model's flexibility, accommodating different loss functions. L2GMOM_SR not only optimises the portfolio positions, but also the graph learning layer, ensuring the network is well optimised w.r.t. Sharpe ratio.

### 4.3.2 Input Features

We consider the following eight momentum features calculated from pricing data for $\mathbf{U}_t$:

- **Volatility-scaled returns** over the past 1-day, 1-month, 3-month, 6-month and 1-year periods, denoted as $\frac{r_{t-\Delta:t}}{\sigma_t \sqrt{\Delta}}$ for asset $i$, where $\Delta = \{1, 21, 63, 126, 252\}$ days. The daily volatility $\sigma_t$ is estimated using an exponential weighted moving standard deviation with a 60-day span on daily returns. This method gives more weight to recent observations, using a smoothing factor $\alpha = \frac{2}{60+1}$. The 60-day span refers to the window over which the weights decay, making this a responsive measure of market risk.

- **MACD** indicators with three combinations of short and long time scales $(S_k, L_k) \in \{(8, 24), (16, 48), (32, 96)\}$ in Eq.(4.4).

Therefore, $\mathbf{U}_t \in \mathbb{R}^{N_t \times 8}$ is the matrix containing these eight features for every asset.

For $\mathbf{V}_t$, we adopt a 252-day lookback window and stack the individual features together. This is to ensure the L2G layer can capture a stable similarity from longer time-series, providing a robust foundation for learning. As a result, we obtain a representation $\mathbf{V}_t \in \mathbb{R}^{N_t \times 8\delta}$, where $\delta = 252$.

In order to mitigate the influence of outliers, we also winsorise the data by capping/flooring it to be within 5 times its exponentially weighted moving (EWM) standard deviations from its EWM average – computed using a 252-day half-life, as adopted by Lim et al. [138].

### 4.3.3 Loss Functions

The parameters in L2GMOM to be estimated include graph learning parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ and momentum feature parameters $(\boldsymbol{\theta}, b)$. Note that in Chapter 3, we observed that allowing varying hyperparameters $(\alpha, \beta, \gamma)$ in each unrolling layer would significantly increase the performance of graph learning. Therefore, we adopt the same approach. The coefficients are learned by minimising with respect to different loss functions associated with portfolio performance, depending on whether the output is $\mathbf{y}_t$ or $\mathbf{x}_t$.

- Mean Squared Error (MSE) for training L2GMOM:

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{\theta}, b) = \frac{1}{|\Omega|} \sum_{(i,t)\in\Omega} \left( y_{i,t} - \frac{r_{i,t:t+1}}{\sigma_{i,t}} \right)^2 \tag{4.11}$$

- Negative Sharpe Ratio for training L2GMOM_SR:

$$\mathcal{L}_{\text{SR}}(\boldsymbol{\theta}, b) = -\frac{\sum_{(i,t)\in\Omega} R_{i,t} \times \sqrt{252}}{\sum_{(i,t)\in\Omega} R_{i,t}^2 - (\sum_{(i,t)\in\Omega} R_{i,t})^2} \tag{4.12}$$

  where $R_{i,t} = x_{i,t} \frac{\sigma_{\text{tgt}}}{\sigma_{i,t}} r_{i,t:t+1}$ is the volatility-normalised return of asset $i$ at time $t$.

## 4.4 Performance Evaluation

### 4.4.1 Backtest Setup

#### Datasets

Our dataset contains 64 ratio-adjusted continuous future contracts obtained from the Pinnacle Data Crop CLC Database[2]. We computed the momentum feature in Section 4.3.2 from the daily prices from 1990 to 2020, although different assets could have different time spans when the prices are available. Table 4.1 lists all the assets used in backtest.

[2] https://pinnacledata2.com/clc.html

Table 4.1: The Pinnacle Universe

| Ticker | Description | Period | Ticker | Description | Period |
|---|---|---|---|---|---|
| **Commodities:** | | | **Equities:** | | |
| BC | BRENT CRUDE OIL, composite | 2010-2020 | AX | GERMAN DAX INDEX | 1999-2020 |
| BG | BRENT GASOIL, composite | 2010-2020 | CA | CAC40 INDEX | 2000-2020 |
| CC | COCOA | 1990-2020 | EN | NASDAQ, MINI | 2001-2020 |
| CL | CRUDE OIL | 1990-2020 | ER | RUSSELL 2000, MINI | 2004-2020 |
| CT | COTTON #2 | 1990-2020 | ES | S&P 500, MINI | 1999-2020 |
| DA | MILK III, composite | 1999-2020 | HS | HANG SENG INDEX | 1999-2020 |
| GI | GOLDMAN SAKS C. I. | 1995-2020 | LX | FTSE 100 INDEX | 1991-2020 |
| JO | ORANGE JUICE | 1990-2020 | MD | S&P 400, MINI, electronic | 1994-2020 |
| KC | COFFEE | 1990-2020 | SC | S&P 500, composite | 1996-2020 |
| KW | WHEAT | 1990-2020 | SP | S&P 500, day session | 1990-2020 |
| LB | LUMBER | 1990-2020 | XU | DOW JONES EUROSTOXX 50 | 2003-2020 |
| MW | WHEAT, MINN | 1990-2020 | XX | DOW JONES STOXX 50 | 2004-2020 |
| NR | NATURAL GAS | 1990-2020 | YM | DOW JONES, MINI ($5.00) | 2004-2020 |
| SB | SUGAR #11 | 1990-2020 | **Fixed Income:** | | |
| W_ | WHEAT, CBOT | 1990-2018 | AP | AUSTRALIAN PRICE INDEX | 2010-2020 |
| ZA | PALLADIUM, electronic | 1990-2020 | DT | EURO BOND (BUND) | 1991-2020 |
| ZB | RBOB, electronic | 1990-2020 | FB | T-NOTE, 5yr composite | 1990-2020 |
| ZC | CORN, electronic | 1990-2020 | GS | GILT, LONG BOND | 1991-2020 |
| ZF | FEEDER CATTLE, electronic | 1990-2020 | TU | T-NOTES, 2yr composite | 1992-2020 |
| ZG | GOLD, electronic | 1990-2020 | TY | T-NOTE, 10yr composite | 1990-2020 |
| ZI | SILVER, electronic | 1990-2020 | UB | EURO BOBL | 2000-2020 |
| ZK | COPPER, electronic | 1990-2020 | US | T-BONDS, composite | 1990-2020 |
| ZL | SOYBEAN OIL, electronic | 1990-2020 | **Currencies:** | | |
| ZM | SOYBEAN MEAL, electronic | 1990-2020 | AN | AUSTRALIAN $$, day session | 1990-2020 |
| ZN | NATURAL GAS, electronic | 1992-2020 | BN | BRITISH POUND, composite | 1990-2020 |
| ZO | OATS, electronic | 1990-2020 | CB | CANADIAN 10YR BOND | 1996-2020 |
| ZP | PLATINUM, electronic | 1990-2020 | CN | CANADIAN $$, composite | 1990-2020 |
| ZR | ROUGH RICE, electronic | 1990-2020 | DX | US DOLLAR INDEX | 1990-2020 |
| ZS | SOYBEANS electronic | 1990-2020 | FN | EURO, composite | 1990-2020 |
| ZT | LIVE CATTLE, electronic | 1990-2020 | JN | JAPANESE YEN, composite | 1990-2020 |
| ZU | CRUDE OIL, electronic | 1990-2020 | MP | MEXICAN PESO | 1997-2020 |
| ZW | WHEAT electronic | 1990-2020 | NK | NIKKEI INDEX | 1992-2020 |
| ZZ | LEAN HOGS, electronic | 1990-2020 | SN | SWISS FRANC, composite | 1990-2020 |

### Strategy Candidates

We compare the proposed strategies against the following reference benchmarks:

- **Long Only** takes a consistent long position $x_{i,t} = 1$ with daily volatility scaling in Eq.(4.1)
- **MACD** [17] takes positions from Eq.(4.5).
- **LinReg** [138] stands for Linear Regression with the eight *momentum features* as covariates, and the signals were constructed following Eq.(4.6) .
- **GLinReg** stands for Graph Linear Regression. This is an ablation strategy. We learn graphs separately with Eq.(3.4) and we fit them in Eq.(4.9d). This is to demonstrate whether the performance gain is from the end-to-end learning architecture.

We name the strategies constructed from the proposed models as **L2GMOM** and **L2GMOM_SR**. The positions of L2GMOM are the signs of the trend estimation in Eq.(4.9d), and L2GMOM_SR directly outputs the positions.

### Optimisation Details

The models were trained every 5 years, involving coefficient optimisation and hyperparameter tuning using data up to the re-trained time point. These optimised models were then used to generate trading signals for the subsequent 5 years, using out-of-sample data. To ensure an adequate number of training samples, the first backtest period covered 1990-1999 for training and 2000-2004 for testing. Similarly, the second period included 1990-2004 for training and 2005-2009 for testing, and so on. Thus, we had a 20-year aggregated out-of-sample period from 2000 to 2020. For each backtest period, the most recent 10% of training data was set aside as a validation set for hyperparameter tuning. The hyperparameters included the number of unrolling times $L$ in Algorithm 3, batch size, and optimiser parameters. Linear models such as LinReg and GLinReg have analytical solutions. For L2GMOM and L2GMOM_SR, we used stochastic gradient descent with the `Adam` optimiser in a mini-batch approach. Early stopping was activated when the validation loss failed to improve beyond a certain threshold, which was determined using the validation set for convergence. To mitigate the randomness introduced by the `Adam` optimiser, we conducted 5 runs with random initial values and averaged the model outputs to create an ensemble.

### 4.4.2 Portfolio Performance

In evaluating the portfolio performance, we use the following annualised metrics in Table 4.2:

- **Profitability**: expected return and hit rate (the percentage of days with positive returns across the test period)
- **Risk**: daily volatility (vol.), downside deviation, maximum drawdown (MDD) and MDD duration (the percentage of days with MDD across the test period).
- **Overall Performance**: Sharpe ratio (expected return/vol.), Sortino ratio (expected return/downside deviation), Calmar ratio (expected return/MDD), and the average profits over the average loss $\left(\frac{\text{Avg. P}}{\text{Avg. L}}\right)$.

In order to have a better comparison between different strategies, we apply an additional layer of volatility scaling at the portfolio level to an annualised volatility target of 15% and report the above evaluation metrics. In Figure 4.1a and Figure 4.1b, we plot the daily cumulative returns of raw signals and signals rescaled to 15% annual volatility of main strategies.

L2GMOM and GLinReg both outperform LinReg in terms of portfolio returns in both raw signals and signals rescaled to 15% target volatility. This suggests that considering the network effect in assets can greatly enhance profitability. L2GMOM also outperforming GLinReg indicates that optimising the graph with respect to portfolio performance can yield higher returns, which supports our conjecture that the separation of network construction in network momentum strategies may adversely affect the portfolio performance. In terms of portfolio risk, L2GMOM has a lower downside deviation and shorter MDD duration in both raw signals and signals rescaled to 15% target volatility, indicating that it manages risk more effectively. L2GMOM also outperforms GLinReg in terms of volatility, suggesting that optimising the graph with respect to portfolio performance can also reduce risk.

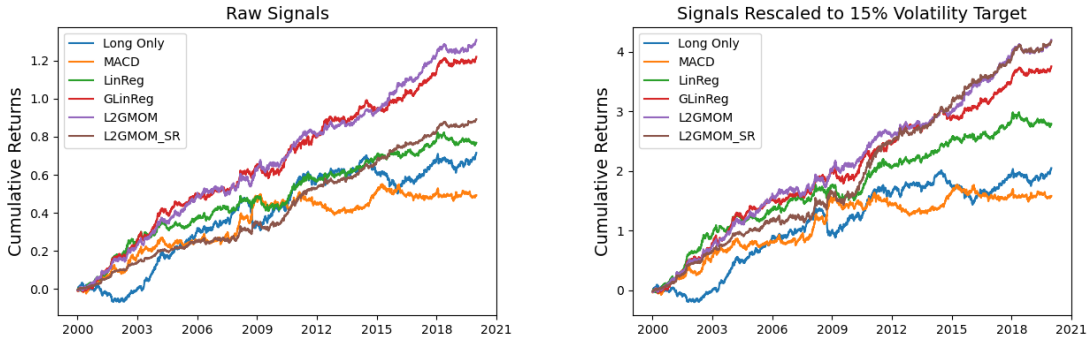
L2GMOM_SR has the lowest volatility, downside deviation, MDD, and MDD duration in the raw signal, indicating that optimising with the Sharpe ratio can significantly reduce risk. It is not surprising to observe a higher Sharpe ratio in L2GMOM_SR, which is consistent with the results in machine learning TSMOM [138]. Note that, the returns of L2GMOM_SR drops in the raw signals, but the return is the highest in the volatility-scaled signals.

Table 4.2: Portfolio Performance Metrics

| | return | vol. | Sharpe | downside deviation | MDD | MDD duration | Sortino | Calmar | hit rate | $\frac{\text{Avg. P}}{\text{Avg. L}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Raw Signals:** | | | | | | | | | | |
| Long Only | 0.036 | 0.053 | 0.671 | 0.037 | 0.200 | 17.2% | 0.951 | 0.173 | 53.9% | 0.955 |
| MACD | 0.025 | 0.047 | 0.533 | 0.033 | 0.121 | 27.5% | 0.766 | 0.200 | 52.9% | 0.977 |
| LinReg | 0.043 | 0.042 | 1.029 | 0.028 | 0.080 | 11.5% | 1.541 | 0.538 | 53.6% | 1.031 |
| GLinReg | 0.060 | 0.049 | 1.227 | 0.033 | 0.085 | 7.7% | 1.806 | 0.713 | **54.3%** | 1.034 |
| L2GMOM | **0.065** | 0.047 | **1.400** | 0.030 | 0.066 | 7.2% | **2.197** | **0.942** | 50.7% | **1.086** |
| L2GMOM_SR | 0.045 | **0.032** | 1.394 | **0.023** | **0.052** | **6.2%** | 1.911 | 0.817 | 52.0% | 1.051 |
| **Signals Rescaled to 15% Target Volatility:** | | | | | | | | | | |
| Long Only | 0.117 | 0.150 | 0.792 | 0.095 | 0.633 | 17.1% | 1.229 | 0.177 | 53.9% | 0.969 |
| MACD | 0.102 | 0.150 | 0.694 | 0.094 | 0.342 | 23.7% | 1.085 | 0.279 | 52.9% | 0.997 |
| LinReg | 0.174 | 0.150 | 1.189 | 0.092 | 0.285 | 11.9% | 1.892 | 0.623 | 53.6% | 1.049 |
| GLinReg | 0.200 | 0.150 | 1.365 | 0.093 | 0.253 | 8.9% | 2.149 | 0.822 | **54.3%** | 1.048 |
| L2GMOM | 0.224 | 0.150 | 1.525 | **0.089** | 0.186 | 7.2% | 2.531 | 1.201 | 50.7% | **1.099** |
| L2GMOM_SR | **0.255** | 0.150 | **1.744** | 0.094 | **0.173** | **5.7%** | **2.716** | **1.495** | 52.0% | 1.091 |

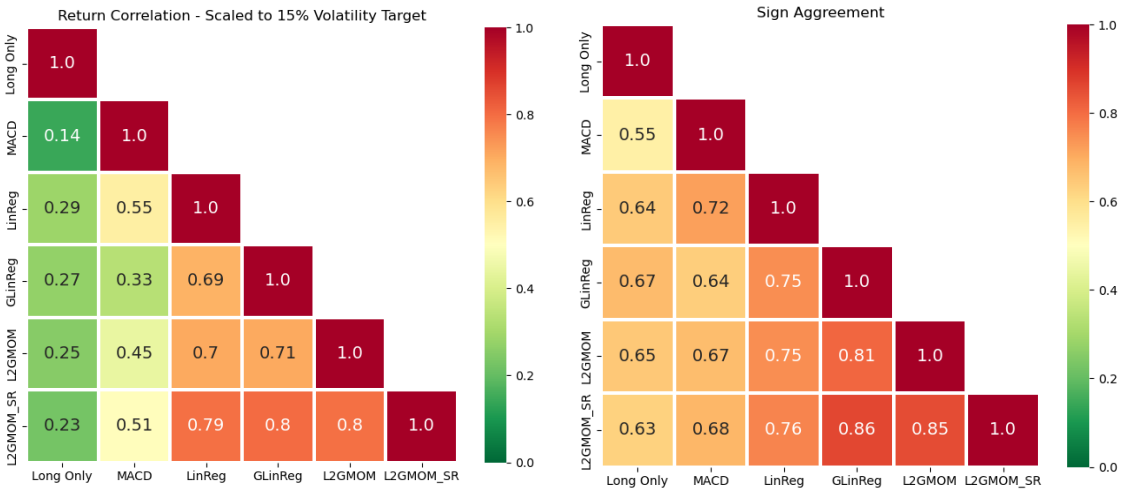
[a] Best performance of each metric is **bold**.

[b] We have a total of six strategies to compare: the proposed strategies (L2GMOM/L2GMOM_SR), one ablation strategy (GLinReg) and three reference strategies (Long Only/MACD/LinReg)




(a) Raw signals (b) Signals scaled to 15% volatility target

Figure 4.1: The cumulative daily returns of the proposed strategies (L2GMOM/L2GMOM_SR), one ablation strategy (GLinReg) and three reference strategies (Long Only/MACD/LinReg)




(a) Return Correlations (b) Sign Agreement

Figure 4.2: A diversification analysis on the correlations and sign agreement of daily returns between the proposed strategies (L2GMOM/L2GMOM_SR), one ablation strategy (GLinReg) and three reference strategies (Long Only/MACD/LinReg).

### 4.4.3 Diversification Analysis

To evaluate whether the signals constructed from the proposed models are different from the reference baselines, we calculated the *correlation* of daily returns between candidate signals in Figure 4.2a. Besides, we calculate the percentage by how much the signals of two strategies have the same direction, also known as *sign agreement* in Figure 4.2b. Note that LineReg, GLinReg, L2GMOM and L2GMOM_SR have the

exactly same input momentum features.

From Figure 4.2a and Figure 4.2b, we draw the following conclusions. The L2GMOM strategy, which incorporates graph learning and network momentum, shows moderate diversification potential with the model-free MACD strategy, as evidenced by their moderate return correlation and position sign agreement. Comparing L2GMOM with LinReg, which uses the same input but different momentum calculation methods, shows less diversification potential due to higher correlation and agreement. However, L2GMOM has better portfolio performance, showing it could be a better alternative strategy. Finally, L2GMOM, GLinReg, and L2GMOM_SR, all of which consider graph effects but differ in optimisation methods, show limited diversification potential due to high return correlations and sign agreements. Despite their different modelling approaches, the similar inputs and graph-based methods used by these strategies result in relatively high correlations, suggesting limited diversification benefits when combined in a portfolio.

### 4.4.4 Turnover Analysis

Following the convention of literature [138], we calculate the cost-adjusted Sharpe ratio from the adjusted returns:

$$\tilde{r}^{\text{portfolio}}_{i,t:t+1} := \frac{1}{N_t}\sum_{i=1}^{N_t}\left(x_{i,t}\frac{\sigma_{\text{tgt}}}{\sigma_{i,t}}r_{i,t:t+1} - c\cdot\sigma_{i,t}\left|\frac{x_{i,t}}{\sigma_{i,t}} - \frac{x_{i,t-1}}{\sigma_{i,t-1}}\right|\right) \tag{4.13}$$

where $c$ is the turnover cost in bps, varying from 0.5 to 5 bps. As ex-ante costs rise, all strategies see a drop in cost-adjusted Sharpe ratios. LinReg, a machine learning model, is most sensitive, turning negative at costs over 3 bps. GLinReg, which includes graph learning, fares better, maintaining a higher positive Sharpe ratios at 2 bps. The proposed models, L2GMOM and L2GMOM_SR, also stay positive at 3 bps, showing resilience to increasing costs. Thus, graph learning strategies like GLinReg, L2GMOM, and L2GMOM_SR offer better cost resilience than LinReg. The Long Only and MACD strategies, which are model-free, show a relatively gradual decrease in Sharpe ratio as ex-ante costs increase. To further reduce the turnover of machine learning strategies, [138] proposed including a turnover regularisation. This is an interesting avenue for future analysis.

### 4.4.5 Network Analysis

Figure 4.4 are example daily networks obtained from the proposed L2GMOM and L2GMOM_SR, specifically the output of Eq.(4.9c). We deliberately chose two differ-

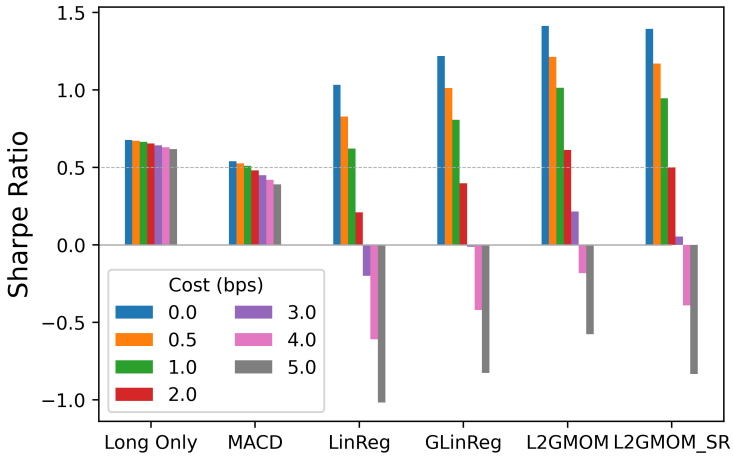
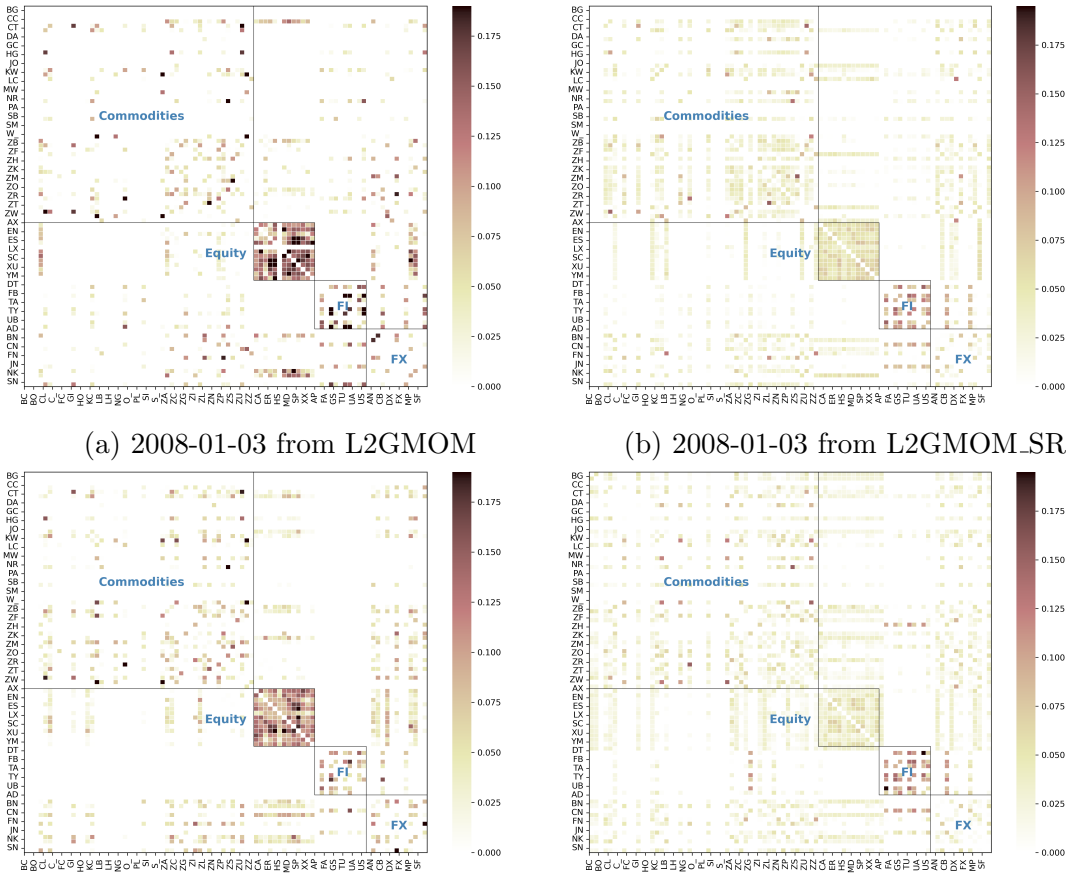



Figure 4.3: Cost-adjusted sharpe ratio of the proposed strategies (L2GMOM/L2GMOM_SR), one ablation strategy (GLinReg) and three reference strategies (Long Only/MACD/LinReg)

ent dates to exhibit, 2008-01-03 for the turbulent period and 2010-01-04 for the calm period. Note that there were some assets without any connections, because the input features for them were missing. The four subfigures demonstrate a strong community structure corresponding to four asset classes: commodities, equities, fixed income (FI), and foreign exchange (FX). The equities form a dense cluster, indicating a high degree of interconnection among equity assets. Inter-class connections are also observed, particularly between commodities and the other three asset classes. However, there is a notable absence of connections between equities and fixed income. This is reasonable given that these two asset classes often exhibit negative correlations and are used to hedge each other, suggesting no momentum spillover according to the strategy.

By comparing Figure 4.4a and 4.4c, we can also capture some temporal variations in the momentum network structure. During calm periods (e.g. 2010-01-04), there are fewer inter-class connections between fixed income and the other assets. However, in turbulent times (e.g. Global Financial Crisis covering 2008-01-03), these connections increase, indicating a shift in momentum propagation under different market conditions.

The graphs obtained from the L2GMOM_SR strategy are denser with more similar edge weights, demonstrating the strategy's tendency in reducing volatility. One possible explanation is that, by increasing the connections between assets, the strategy ensures that each asset receives momentum information passed from a broader range of assets, rather than relying on a few, thus enhancing the robustness and stability




(a) 2008-01-03 from L2GMOM

(b) 2008-01-03 from L2GMOM_SR

(c) 2010-01-04 from L2GMOM

(d) 2010-01-04 from L2GMOM_SR

Figure 4.4: A momentum spillover analysis on the learned networks from L2GMOM or L2GMOM_SR for two different dates: 2008-01-03 (turbulent period) and 2010-01-04 (calm period).

of the strategy.

## 4.5 Conclusions

In this chapter, we introduce an end-to-end graph machine learning framework for developing network momentum strategies in the futures market of macro assets, specifically proposing two model variants: L2GMOM and L2GMOM_SR. This framework effectively learns the graph adjacency matrix to describe the network effect of momentum spillovers between macro assets from their market data, while simultaneously optimising portfolio performance. A key contribution of the proposed framework is to reduce the dependence on alternative datasets, which are less accessible for macro assets like commodities compared to equities. For futures contracts of commodities, acquiring alternative data to capture relationships beyond the company level as in equities proves more challenging and expensive.

The backtest of the proposed model on a simplified portfolio construction across 64 continuous futures contracts of macro assets demonstrates promising results. It shows an enhancement in profitability compared to traditional time series momentum strategies. Additionally, our models offer high interpretability, yielding explicit networks as a byproduct. These networks not only illustrate the similarities in the momentum feature space of these assets but are also optimised for portfolio performance. This provides a profound insight into the unexpected connections, e.g. between the 10-year Treasury Notes and Lean Hogs, that can be instrumental in further economic and financial research. Additionally, analysis of these learned networks reveals a community structure and temporal variations consistent with established financial principles.

Admittedly, there are inherent limitations to the proposed framework. Firstly, our model is data-driven, and its efficacy in capturing meaningful momentum signals from other asset classes, such as equities, is not confirmed. Secondly, the portfolio construction procedure in our study is simplified to underscore the modelling innovation. For practical application in real-world trading scenarios, many other factors, including potential costs and liquidity constraints, must be considered. Adjusting for these factors could potentially reduce portfolio performance.

Consequently, there are several paths for future research. Investigating turnover regularisation as a potential enhancement to the proposed framework is also worth considering. Additionally, a thorough analysis of trading costs and limitations is

recommended. This could provide a deeper understanding of the practical advantages of this method in the real financial market.

# Chapter 5

# Financial Application II: Realised Volatility Forecasting

## Contents



## 5.1 Introduction

Modelling and forecasting stock return volatility plays a crucial role in the theory and practice of finance. Extensive attention has been devoted to this subject within the

literature, encompassing numerous ARCH, GARCH, and stochastic volatility models. Due to the availability of high-frequency data, realized volatility (RV), calculated from the sum of squared intraday returns, has gained popularity in recent years. For example, [52] put forward the Heterogeneous Autoregressive (HAR) for predicting daily RVs using various lagged RV components over different time horizons. While these methods provided valuable insights into the dynamic dependence of volatilities, they neglected the volatility spillover effect among assets, as highlighted by [25].

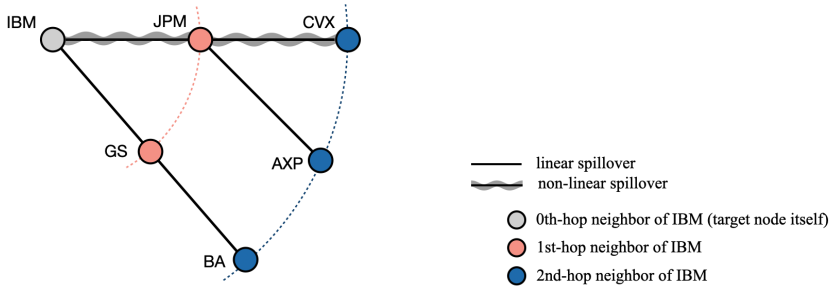
The *volatility spillover effect* is referred to as the phenomenon that some big shocks of a specific asset (or market) may have an influence on the volatilities of other assets (or markets). Therefore, the discovery of volatility spillover effects is expected to benefit the understanding and forecasting of volatilities. For example, [31] documented that the VIX of the U.S. market plays an important role in forecasting the volatilities of other global assets markets. [58] examined the cross-asset spillover effects from stocks, currencies, and commodities to improve the prediction of RV of crude oil. [25, 134] utilises the commonality in risk structures to improve the forecasting of future volatility.

There is a number of studies dedicated to incorporating the spillover effect into volatility modelling, e.g. BEKK-GARCH ([73]) and VAR-GARCH ([139]). In terms of modelling RV, [213] employed Vector Autoregression (VAR) to obtain the multivariate volatility forecasts for stock market indices. However, in high-dimensional scenarios, the aforementioned models may deliver poor out-of-sample forecasts due to the curse of dimensionality, as emphasised by [33]. Most recently, [228] introduced graph-based methods to capture the volatility spillover effects, and proposed a parsimonious model to augment HAR via neighbourhood aggregation on a graph that represents a financial network, denoted as Graph HAR (or GHAR). In these graphs, each asset is modelled as a node and an edge connecting two nodes encodes the existence of the spillover effect between their volatilities. GHAR leverages neighbourhood aggregation to generate a new covariate over the graph for each underlying asset and enhance the accuracy of individual volatility forecasts.

One natural question following GHAR is whether there exists any spillover effect between nodes that is beyond one step, a.k.a. *multi-hop neighbours* (see detailed definitions in Section 5.2.1). For example, as illustrated in Figure 5.1, for the target node (i.e. IBM), in addition to the spillover effect of 1st-hop neighbours (i.e. JPM and GS), we are also interested in whether there is any spillover effect from 2nd-hop neighbours (i.e. AXP, CVX, and BA). To the best of our knowledge, spillover effects

from multi-hop neighbours have not yet received much attention in the volatility modelling literature.[1]

Figure 5.1: An illustration of multi-hop and nonlinear volatility spillover.




*Note:* The target node represents the volatility of IBM. The connections are only for illustration, and hence not necessarily consistent with our experiments.

In addition to multi-hop effects, another interesting question is whether the volatility spillover is *nonlinear*. Most of the previous studies focus on the linear relationships across assets or markets, such as [58, 213, 228]. In a few scenarios, the existence of nonlinear volatility spillover effects has also been discovered and examined. For example, [44] documented the existence of significant nonlinear spillover effects among four major markets, i.e. the U.S., Canada, Japan, and the U.K., via a nonlinear causality test proposed by [11]. [212] attempted to capture the nonlinear relationship between the volatilities of stocks and crude oil, by incorporating the asymmetric effect of oil prices and regime shifts.

While the incorporation of multi-hop neighbours expands the set of features and the potential presence of nonlinear spillover effects introduces new functional forms to describe volatility dynamics, it is also worth emphasising that the choice of *estimation criterion (EC)*, representing the objective function for estimating model parameters, plays a crucial role. This follows the perspective that a statistical forecasting model typically consists of three essential components: (i) feature set, (ii) model specification, and (iii) EC. Traditional econometric models, such as GARCH, commonly employ conditional quasi-likelihood (QLIKE) based on normal distributions for parameter estimation. Conversely, models focused on forecasting realized volatilities,

[1]The 2nd-hop connections have been studied in the context of cascading effects of financial networks, e.g. [2], where the shocks that occur to an individual firm would propagate through the rest of the economy. Consequently, the downstream firms more than one hop away may also suffer from the impact.

such as HAR, often utilise the mean squared error (MSE) as their EC. Therefore, an important question arises as to whether a preferred EC exists[2], especially when combined with the aforementioned aspects, namely the effect of multi-hop neighbours and non-linear relationships.

In the present work, we explore these three questions using graph neural networks (GNNs). GNNs are a class of deep learning models designed for performing inference on graphs and graph-structured data. They are capable of learning node and graph-level representations that are useful for a wide range of tasks involving graph analysis, such as node classification, node regression, and graph clustering. GNNs have demonstrated successful applications in various financial domains, including stock movement prediction ([40, 181]), credit risk prediction ([210, 137]) and payment fraud detection ([143, 144]).

The revised paragraph below addresses the comment suggesting that the claims in Chapters 4 and 5 be moderated. It aims to provide a more balanced viewpoint, highlighting both the novel aspects and the limitations of the applications discussed.

In this chapter, we have developed a GNN-based framework, specifically utilising Graph Convolutional Neural Networks (GCNs), to examine the topological characteristics of volatility spillover effects[3]. By replacing the linear neighbourhood aggregation in the GHAR model in [229] with nonlinear operations, our model can autonomously learn the nonlinear spillover effects. Additionally, the multi-layer setup of GCNs allows exploration of this nonlinearity in a multi-hop context, i.e., spillover to neighbours more than one hop away in the financial network.

The advantage of our model lies in its flexibility to accommodate various error criteria (EC) during the training phase. Additionally, the preference for GCNs over more complex architectures like Transformers or advanced GNN variants is further justified by their efficiency. While these advanced GNN models might offer additional expressive power, they also bring greater computational complexity and a higher risk of overfitting. This is particularly worth noting, considering that daily realised volatility is susceptible to overfitting due to non-stationary distribution and limited training data size. Therefore, the GCN architecture offers an optimal balance for the requirements of our study.

The main contributions of our work are summarised as follows. First, we examine the spillover effect from multi-hop neighbours in the financial graph, and observe that

[2][45] conducted an empirical evaluation of the influence of various EC on linear models and found that using the QLIKE yields slightly better forecasts.

[3]A contemporaneous study by [38] employed GNN for intraday volatility forecasting, but their method faced limitations in terms of interpretability and bench-marking challenges.

the multi-hop spillover effect is not necessary, as long as 0-hop and 1-hop are included. Second, we establish that the proposed GNN model with nonlinear operations significantly improves the forecasting performance of GHAR, indicating the existence of nonlinear spillover effects on 1-hop neighbours. Third, compared to MSE-trained models, models employing QLIKE as the EC generally achieve substantial improvements in predictive accuracy. Overall, our proposed GNN model trained with QLIKE exhibits an average forecast error in MSE (resp. QLIKE) approximately 13% (resp. 4%) lower than that of the standard HAR model. Additionally, we examine the robustness of our proposed models across various market conditions, an alternative data-splitting scheme, and an alternative universe, consistently observing enhanced prediction accuracy across all experimental settings.

The remainder of the chapter is organised as follows. Section 5.2 contains preliminaries on the mathematical definitions of graphs, a brief review of GNN models, and two baseline models (HAR, GHAR). In Section 5.3, we introduce the proposed model (GNNHAR), evaluation criterion, and forecast evaluation approaches. Section 5.5 outlines the experimental setup and provides the key out-of-sample results across various forecast horizons and market regimes. In this section, we also conduct an extensive analysis concerning the impact of QLIKE, nonlinearity, and multi-hop neighbours. In Section 5.6, we perform several robustness tests. We conclude our work and highlight future research directions in Section 5.7.

## 5.2 Preliminaries

In this section, we summarise the preliminary concepts and models. In particular, we provide the mathematical definitions of graphs and multi-hop neighbours in Section 5.2.1. In Section 5.2.2, we briefly review two popular graph neural networks, that inspired our work. Section 5.2.3 revisits the baseline model HAR for forecasting realized volatilities, while Section 5.2.4 reviews another baseline model GHAR.

### 5.2.1 Graph definitions

**Definition 5.2.1** (Graphs)**.** *A graph $\mathcal{G}$ is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of $N$ nodes and $\mathcal{E}$ is a set of edges, where $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge connecting node $v_i$ and node $v_j$.*

**Definition 5.2.2** (Adjacency matrix)**.** *An adjacency matrix $\mathbf{A}$ is a square matrix whose dimension is $N \times N$, where $\mathbf{A}[i, j]$ represents the connection between $v_i$ and*

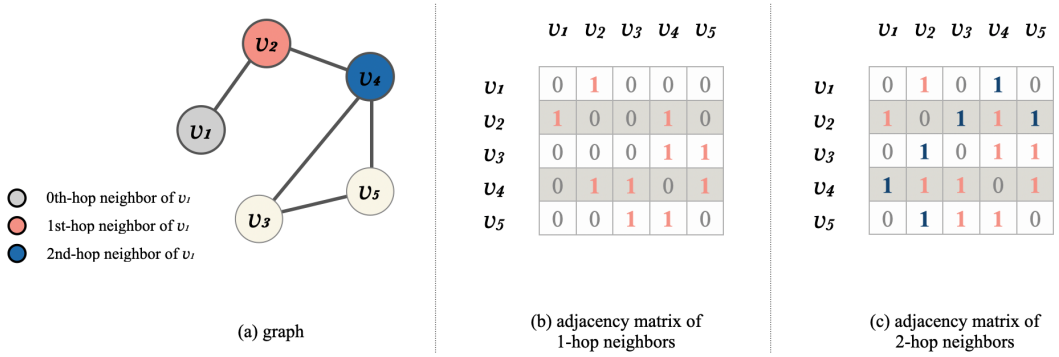
$v_j$ *in the graph* $\mathcal{G}$*. An adjacency matrix can be weighted, where* $\mathbf{A}[i,j] \geq 0$, $\forall i,j$ *represents the strength/intensity of the connection between nodes* $v_i$ *and* $v_j$*. If* $\mathbf{A}[i,j] \in \{0,1\}$, $\forall i,j$*, the graph is a binary graph. The diagonal elements of* $\mathbf{A}$ *are all 0 since edges from a node to itself are typically not considered in graphs. In this article, we mainly consider* ***binary*** *graphs without self-connections.*

**Definition 5.2.3** ($K$-hop neighbours)**.** *Following [75], we use the* $K$*-hop neighbours of node* $v$ *to represent all the neighbours that have distance from node* $v$ *less than or equal to* $K$*, based on the shortest path distance (SPD) kernel. In contrast,* $k$*th-hop neighbours represent the neighbours with exact distance* $k$ *from node* $v$*. Finally, we denote* $Q^K_{v,\mathcal{G}}$ *as the set of* $K$*-hop neighbours of node* $v$ *in graph* $\mathcal{G}$*.*

**Example 1** (A graph with 5 nodes)

In Figure 5.2(a), we plot an example graph with 5 nodes and 5 undirected edges, where the node $v_1$ is coloured as a target node. Nodes $v_2$ and $v_4$ are the 1st-hop and 2nd-hop neighbours of $v_1$, respectively. Figure 5.2(b) shows its adjacency matrix. Figure 5.2(c) is the adjacency matrix of the graph in (a) when considering 2-hop neighbours, where we write $Q^1_{v_1,\mathcal{G}} = \{v_1, v_2\}$ and $Q^2_{v_1,\mathcal{G}} = \{v_1, v_2, v_4\}$.

Figure 5.2: Illustration of a graph and its corresponding adjacency matrices with multi-hop neighbours.




## 5.2.2 A brief review on GNNs

Graph neural networks (GNNs) are a class of deep learning models designed for performing inference on graphs. The main idea is to learn a vector representation for every node defined on a graph, while preserving both graph topology structure and node content information ([217]). The node representations, for example, can be

further applied to node classification or regression. To this end, many GNN variants utilise the idea of **neighbourhood aggregation** in developing the layerwise forward propagation rules. In essence, neighbourhood aggregation effectively generates a node $v$'s representation by aggregating its own feature vector $\boldsymbol{h}_v \in \mathbb{R}^D$ and the feature vectors of its connected nodes $\boldsymbol{h}_u \in \mathbb{R}^D$, where $u \in Q^1_{v,\mathcal{G}}$. Common examples of aggregation functions include sum, mean, and maximum. Early attempts of GNNs, see [182] and [54], update node representations by aggregating neighbourhood information recursively until a stable equilibrium is reached. More efficiently, a novel notion of convolution operator can be defined on irregular graphs to process neighbourhood aggregation in parallel, so-called graph convolution.[4] A considerable number of GNN variants and architectures are built from different graph convolution operators. We provide a brief introduction to a specific GNN architecture that is relevant to our volatility forecasting models.

**Graph Convolutional Network** (GCN) was introduced by [121]. It approximates the graph convolution with the following layer-wise propagation rule[5]

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{O}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{O}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{\Theta}^{(l)}\right), \tag{5.1}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A}+\boldsymbol{I}_N$ is the adjacency matrix of the graph $\mathcal{G}$ with added self-connections, and $\tilde{\boldsymbol{O}}$ is a diagonal matrix with $\tilde{\boldsymbol{O}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$. $\tilde{\boldsymbol{O}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{O}}^{-\frac{1}{2}}$ is the normalised adjacency matrix, introduced to stabilise the training of the GNN models. $\boldsymbol{\Theta}^{(l)} \in \mathbb{R}^{D^{(l)}\times D^{(l+1)}}$ is the layer-specific trainable weight matrix. $\boldsymbol{H}^{(l)} \in \mathbb{R}^{N\times D^{(l)}}$ is the matrix of node representations at $l$-th layer. $\boldsymbol{H}^{(0)}$ is the input node features. $\sigma(\cdot)$ denotes a nonlinear activation function, such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.

When addressing various research problems, the above GNN layers can be combined with other deep learning layers in an end-to-end learning framework. Additionally, the exploration of multi-hop effects can be achieved by straightforwardly stacking multiple GNN layers within a model. A model that incorporates $K$-layer GNN layers is commonly referred to as a $K$-layer GNN model.

**Definition 5.2.4** (Receptive field)**.** *In a GNN model, the receptive field of a target node is the set of nodes of the graph that determine its representations; see [75, 4].*

[4]Convolution operation has been widely applied to regular grid data, e.g. image pixels. Recently, it has been extended to graph-structured data. More details can be found in [191].

[5]The GCN propagation rule approximates the graph convolution with the first-order Chebyshev spectral polynomials (ChebyNet). It alleviates the gradient vanishing/exploding and stabilises the training in ChebyNet by introducing a normalisation step on $\boldsymbol{A}$. More details about ChebyNet can be found in [57].

**Proposition 5.2.1.** *After $K$ layers of graph convolution in a GNN model, every node representation is determined by the information from the nodes within $K$ hops; see [75].*

The above proposition states that the size of receptive field of every node is associated with the number of layers in a GNN model. It is found that [4] when $K$ is unnecessarily large, any two nodes could easily have highly overlapping receptive fields, and consequently attain highly similar node representations, which leads to the problem of over-smoothing (see [133, 37]). Therefore, a large $K$ does not always help, but on the contrary, it may lead to indistinguishable node representations, and thus weaken the forecasting or classification accuracy.

### 5.2.3 Forecasting RV with HAR

Let $P_{i,t}$ denote the price of asset $i$ and $r_{i,t} = \log(P_{i,t}/P_{i,t-1})$ be its log-return at day $t$. The standard approach for modelling return data is to use the decomposition

$$r_{i,t} = \mu_{i,t} + X_{i,t}, \tag{5.2}$$

where $\mu_{i,t}$ denotes the (conditional) mean of the return, $X_{i,t}$ is a diffusion term which may be modelled as

$$X_{i,t} = \sigma_{i,t}\varepsilon_{i,t}, \quad \varepsilon_{i,t} \sim \mathrm{IID}(0,1), \tag{5.3}$$

where $\sigma_{i,t}$ is often referred to as the volatility function, and $\varepsilon_{i,t}$ is assumed to be independent of $\sigma_t$.

[5, 16] showed that the sum of squared intraday returns is a consistent estimator of the unobserved $\sigma^2_{i,t}$. Therefore, the daily RV for a particular asset $i$ at day $t$ is defined as

$$RV_{i,t} = \sum_{l=1}^{M} r^2_{i,t(l)}, \tag{5.4}$$

where $r_{i,t(l)}$ is the $l$-th $\Delta$-min log returns during day $t$, i.e. $r_{t(l)} = \log p_{t(l\Delta)} - \log p_{t((l-1)\Delta)}$, and $p_{t(l\Delta)}$ is the price at time $l\Delta$ at day $t$. We refer to $\boldsymbol{RV}_t = (RV_{1,t}, \ldots, RV_{N,t})'$ as the vector of cross-sectional realized volatilities. In this article, we consider 5-min windows in a trading day, following [140].[6]

[52] proposed a Heterogeneous Autoregressive Regression (HAR) model for modelling and forecasting the RV where the lagged daily, weekly, and monthly volatility

[6]We also adopt the subsampling averaging method (see [189, 6, 205]) to improve the above $RV$ estimation, which uses all $\Delta$-minute returns, not just non-overlapping ones.

components are incorporated as features. For a given asset $i$, its RV of day $t$ is modelled as

$$\textbf{HAR}: RV_{i,t} = \alpha + \beta_d RV_{i,t-1} + \beta_w RV_{i,t-5:t-2} + \beta_m RV_{i,t-22:t-6} + u_{i,t}, \tag{5.5}$$

where $RV_{i,t-5:t-2} = \frac{1}{4}\sum_{k=2}^{5} RV_{i,t-k}$, $RV_{i,t-22:t-6} = \frac{1}{17}\sum_{k=6}^{22} RV_{i,t-k}$ denote the weekly and monthly lagged RV, respectively. The choice of a daily, weekly, and monthly lag is aiming to capture the long-memory dynamic dependencies observed in most RV series.

### 5.2.4 Graph HAR (GHAR)

[228] augmented the HAR model to capture the volatility spillover effect via neighbourhood aggregation on graphs. Denote $\boldsymbol{V}_{:t-1} = [\boldsymbol{RV}_{t-1}, \boldsymbol{RV}_{t-5:t-2}, \boldsymbol{RV}_{t-22:t-6}] \in \mathbb{R}^{N\times 3}$, GHAR is defined as

$$\begin{aligned}\textbf{GHAR}(\boldsymbol{A}): \quad \boldsymbol{RV}_t &= \boldsymbol{\alpha} + \beta_d \boldsymbol{RV}_{t-1} + \beta_w \boldsymbol{RV}_{t-5:t-2} + \beta_m \boldsymbol{RV}_{t-22:t-6} \\ &\quad + \gamma_d \boldsymbol{W}\cdot\boldsymbol{RV}_{t-1} + \gamma_w \boldsymbol{W}\cdot\boldsymbol{RV}_{t-5:t-2} + \gamma_m \boldsymbol{W}\cdot\boldsymbol{RV}_{t-22:t-6} + \boldsymbol{u}_t, \\ &= \boldsymbol{\alpha} + \boldsymbol{V}_{:t-1}\boldsymbol{\beta} + \boldsymbol{W}\boldsymbol{V}_{:t-1}\boldsymbol{\gamma} + \boldsymbol{u}_t,\end{aligned} \tag{5.6}$$

In the GHAR, $\boldsymbol{\alpha} \in \mathbb{R}^N, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^3$ are parameters to be estimated. $\boldsymbol{W} = \boldsymbol{O}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{O}^{-\frac{1}{2}}$ is the normalised adjacency matrix without self-connections, where $\boldsymbol{O} = \text{diag}\{n_1, \ldots, n_N\}$ and $n_i = \sum_j \boldsymbol{A}[i,j]$, $\forall i$.[7]

The authors in [228] constructed different types of graphs and concluded that adjacency matrices obtained through Graphical LASSO effectively capture the relationships between individual volatilities, thereby enhancing forecasting accuracy.

**Graphical LASSO** (GLASSO), proposed by [79], is a sparsity-penalised maximum likelihood estimator for the precision matrix $\boldsymbol{\Theta}$ (i.e. the inverse of the covariance matrix). It assumes the input vector of $N$ nodes is drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. If the $ij$-th entry of the precision matrix is zero, the returns of $i$-th asset and $j$-th asset are conditionally independent. Therefore, the adjacency matrix $\boldsymbol{A}$ from GLASSO is defined as $\boldsymbol{A}[i,j] = 1$ if $\boldsymbol{\Theta}[i,j] \neq 0$; otherwise $\boldsymbol{A}[i,j] = 0$.

One key advantage of Graphical LASSO is its ability to estimate the conditional independence of assets based on historical returns. Additionally, it offers stability in estimation even in high-dimensional settings where the number of assets exceeds the

[7] It is worth noting that for GHAR, the normalisation of $\boldsymbol{W}$ does not impact the forecasting performance directly. However, it does assist with evaluating the relative effect of 0th-hop neighbours in comparison to 1st-hop neighbours.

number of returns. Based on these compelling results, we adopt Graphical LASSO to establish the volatility graph in our study.

## 5.3 Proposed methodology

To investigate the presence of multi-hop and nonlinear effects in modelling volatility spillover, we propose a new class of forecasting models based on the GNNs in Section 5.4.1. Furthermore, we highlight the significance of using various criteria for the estimation of model coefficients in Section 5.4.2. In Section 5.4.3, we introduce the forecast evaluation methods and emphasise the differences between estimation criteria and forecast evaluations.

## 5.4 GHAR with multi-hop (GHAR2Hop)

It is important to highlight that HAR can be interpreted as a model that only considers the 0th-hop neighbours, i.e. the target node itself, while the GHAR takes into account both the 0th-hop and 1st-hop neighbours. In order to explore the potential benefits of multi-hop neighbours in enhancing volatility forecasting, we delve into the investigation of whether they provide additional predictive power. To address this novel question, we consider the following model.

$$\textbf{GHAR2Hop}(\boldsymbol{A}): \quad \boldsymbol{RV}_t = \boldsymbol{\alpha} + \boldsymbol{V}_{:t-1}\boldsymbol{\beta} + \boldsymbol{W}\boldsymbol{V}_{:t-1}\boldsymbol{\gamma} + \textsc{Hop}2(\boldsymbol{A})\boldsymbol{V}_{:t-1}\boldsymbol{\delta} + \boldsymbol{u}_t, \quad (5.7)$$

where $\textsc{Hop}2(\boldsymbol{A})$ maps the raw adjacent matrix (for 1st-hop neighbours) to the adjacent matrix of 2nd-hop neighbours. Specifically, $\textsc{Hop}2(\boldsymbol{A}) = \text{XOR}(\boldsymbol{A}^2 \wedge (\neg \boldsymbol{A}), \boldsymbol{I}_N)$. $\boldsymbol{A}^2[i,j]$ has a non-zero if it is possible to go from node $i$ to node $j$ in 2 or fewer steps, $\neg \boldsymbol{A}$ excludes the 1st-hop neighbours, and XOR confirms the diagonal of 2nd-hop adjacent matrix to be zero. For a visual representation and further details, we refer the reader to Example 1 and Figure 5.2. In our experiments, we use the normalized adjacent matrix of 2nd-hop neighbours and estimate (5.7) through OLS.

### 5.4.1 GNN-enhanced HAR (GNNHAR)

As introduced in (5.6), GHAR in [228] assumes a linear relationship between the volatilities of two connected assets. However, if the spillover effect is nonlinear, linear models are misspecified and are likely to generate less accurate forecasts. Additionally, GHAR considers only the 0th-hop and 1st-hop neighbours, and this lack of consideration for multi-hop neighbours may lead to incomplete information and less

accurate predictions. In light of the abilities of GNNs discussed in Section 5.2, we propose the following GNN architecture for modelling the volatility spillover effect, allowing for nonlinearity and multi-hop neighbours to improve prediction accuracy.

$$\mathbf{GNN}(\boldsymbol{H}^{(l)}, \boldsymbol{A}): \quad \boldsymbol{H}^{(l+1)} = \mathrm{ReLU}\left(\boldsymbol{O}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{O}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{\Theta}^{(l)}\right), \tag{5.8}$$

where $\boldsymbol{W} = \boldsymbol{O}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{O}^{-\frac{1}{2}}$ is the normalised adjacency matrix, used to avoid numerical instabilities and exploding/vanishing gradients during the training phrase. Note that $\boldsymbol{H}^{(0)} = \boldsymbol{V}_{:t-1} \in \mathbb{R}^{N\times 3}$, which is the matrix composed of the past daily, weekly and monthly volatilities. $\boldsymbol{H}^{(l)} \in \mathbb{R}^{N\times D^{(l)}}$ is a matrix of node representations at the $l$-th layer of GNN, where $D^{(l)}$ is the dimension of node representations. $\boldsymbol{\Theta}^{(l)} \in \mathbb{R}^{D^{(l)}\times D^{(l+1)}}$ is a matrix of trainable parameters.

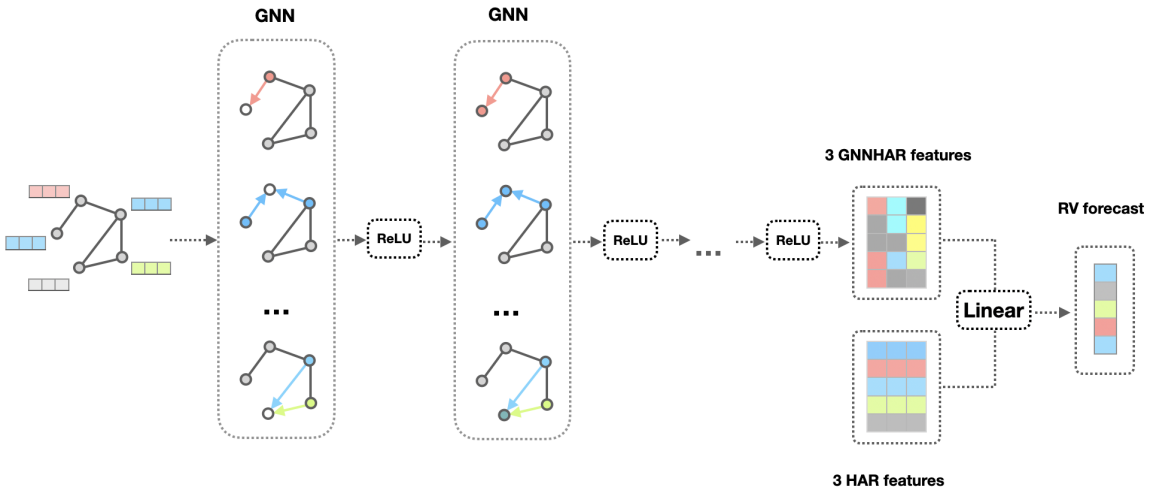



Figure 5.3: An illustration of the GNNHAR model.

In contrast to the GCN architecture shown in (5.1), our proposed GNN propagation rule does not include self-connections, i.e. the diagonal elements in $\boldsymbol{A}$ are zeros. We conjecture that the mechanism of an individual stock's past volatility on its future volatility differs from the spillover effect. As a result, we apply the above GNN propagation in (5.8) solely to model the spillover effect, while the impact of a stock's own past volatility is modelled using the same linear model as in HAR.[8] This allows for a clear and straightforward explanation of the performance gain of our proposed model compared to the baseline models, HAR and GHAR.

[8]For a study on the non-linearity between a stock's past volatility and its future volatility, we refer readers to [229], which suggested that introducing nonlinearity does not result in additional predictive power when modelling daily RV.

We introduce a GNN-enhanced HAR model, referred to as GNNHAR1L in (5.9), by replacing the linear neighbourhood aggregation in GHAR (i.e. the term $\boldsymbol{W}\boldsymbol{V}_{:t-1}\boldsymbol{\gamma}$ in (5.6)) with the proposed GNN layer in (5.8). It is worth noting that the main difference between GNNHAR1L and GHAR is that GNNHAR1L uses a graph convolutional layer with a nonlinear activation function, in the form of

$$\begin{aligned} & \boldsymbol{H}^{(1)} = \mathrm{GNN}(\boldsymbol{V}_{:t-1},\ \boldsymbol{A}) \\ \mathbf{GNNHAR1L}(\boldsymbol{A}): \quad & \boldsymbol{RV}_t = \boldsymbol{\alpha} + \boldsymbol{V}_{:t-1}\boldsymbol{\beta} + \boldsymbol{H}^{(1)}\boldsymbol{\gamma} + \boldsymbol{u}_t. \end{aligned} \tag{5.9}$$

As introduced in Section 5.2, the nonlinear multi-hop effects can be explored by stacking multiple layers of GNN. We denote the 2-layer and 3-layer models as GNNHAR2L and GNNHAR3L respectively. Specifically,

$$\begin{aligned} & \boldsymbol{H}^{(2)} = \mathrm{GNN}(\boldsymbol{H}^{(1)},\ \boldsymbol{A}) \\ \mathbf{GNNHAR2L}(\boldsymbol{A}): \quad & \boldsymbol{RV}_t = \boldsymbol{\alpha} + \boldsymbol{V}_{:t-1}\boldsymbol{\beta} + \boldsymbol{H}^{(2)}\boldsymbol{\gamma} + \boldsymbol{u}_t. \end{aligned} \tag{5.10}$$

$$\begin{aligned} & \boldsymbol{H}^{(3)} = \mathrm{GNN}(\boldsymbol{H}^{(2)},\ \boldsymbol{A}) \\ \mathbf{GNNHAR3L}(\boldsymbol{A}): \quad & \boldsymbol{RV}_t = \boldsymbol{\alpha} + \boldsymbol{V}_{:t-1}\boldsymbol{\beta} + \boldsymbol{H}^{(3)}\boldsymbol{\gamma} + \boldsymbol{u}_t. \end{aligned} \tag{5.11}$$

Our empirical analysis indicates that each node in the volatility spillover graphs for the components of the DJIA30 index, chosen by GLASSO, is connected to other nodes within a maximum of three steps (i.e. the graph has a diameter of length 3, which is the size of the longest shortest pairwise path distance in the graph).[9] Consequently, by employing a 3-layer GNN, we can guarantee that the volatility representation of each asset encompasses information from all other assets. Hence, there is no requirement to investigate beyond a 3-layer GNN. Nevertheless, it is worth noting that for different universes or graphs, the number of GNN layers may need to be re-evaluated according to the distribution of SPDs.

### 5.4.2 Estimation criterion

The standard HAR model described in (5.5) is often estimated via ordinary least squares (OLS). In other words, the estimation criterion (EC) for its in-sample training is the MSE. When the errors $u_{i,t}$ in (5.5) are independent, homoscedastic, and normally (Gaussian) distributed, the OLS estimator is consistent under the asymptotic sense. Nonetheless, given the stylised facts of RV (such as spikes, heteroskedasticity, and so on), the OLS estimator may not be an ideal choice and a better estimator

[9]Note that the hyperparameter that determines the sparsity of GLASSO graph estimates is chosen by cross-validation on the GLASSO objective function.

may be available. For example, [98] proved that the likelihood-based estimator is asymptotically efficient, although the likelihood-based estimator can also be vastly inferior if the underlying statistical model is misspecified. [46] empirically compared various estimation criteria on HAR and found that simple weighted least squares can yield substantial improvements to the predictive ability of the standard HAR.

Meanwhile, QLIKE has served as a commonly employed metric for estimating traditional econometric models including GARCH. When $\varepsilon_t$ in (5.3) has a density (typically unknown), we can utilise the conditional likelihood based on normal density to estimate the models. Specifically, assuming $\varepsilon_t \sim \mathcal{N}(0,1)$, the conditional Gaussian likelihood function after ignoring constants is $-\frac{1}{T}\sum_t [\log(\sigma_t^2) + X_t^2/\sigma_t^2]$. It was demonstrated by [96, 74] that the conditional Gaussian QLIKE estimator is always consistent, even when $\varepsilon_t$ deviates from a normal distribution.

Utilising the flexibility of neural networks and the stochastic gradient descent algorithms, we are able to investigate whether different estimation criteria would result in disparate model predictions. Specifically, our primary focus revolves around the following estimation criteria: MSE and QLIKE, defined as follows

- MSE:
$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\#\mathcal{T}_{train}}\sum_{t\in\mathcal{T}_{train}}\left(RV_{i,t}-\widehat{RV}_{i,t}^{(F)}\right)^2, \tag{5.12}$$
- QLIKE:
$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\#\mathcal{T}_{train}}\sum_{t\in\mathcal{T}_{train}}\left[\frac{RV_{i,t}}{\widehat{RV}_{i,t}^{(F)}}-\log\left(\frac{RV_{i,t}}{\widehat{RV}_{i,t}^{(F)}}\right)-1\right], \tag{5.13}$$

where $\widehat{RV}_{i,t}^{(F)}$ represents the predicted value of $RV_{i,t}$ by a specific model $F$. $N$ is the number of stocks in our universe, $\mathcal{T}_{train}$ is the training period, and $\#\mathcal{T}_{train}$ is the length of the training period.

Lower values are preferred for both measures. For clarity, we will use $F_M$ ($F_Q$) to denote the model $F$ trained with MSE (QLIKE). To the best of our knowledge, adopting QLIKE as the estimation criterion to optimise volatility models, especially those grounded on neural networks, has not yet drawn considerable attention within the literature.

Figure 5.4: A comparison of the MSE and QLIKE loss functions.

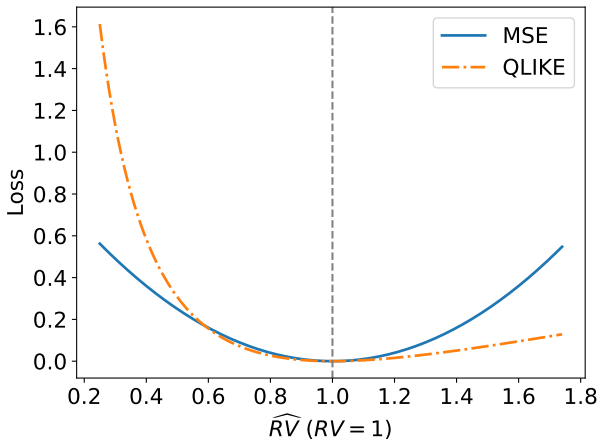



Figure 5.4 displays the aforementioned EC for different forecasts $\widehat{RV}$ when $RV = 1$. Notably, the QLIKE function exhibits asymmetry and imposes a higher penalty on under-predictions. This feature becomes particularly significant during turbulent periods, as the volatility forecasts tend to be smaller than the actual shocks. By placing emphasis on under-predictions, models trained with QLIKE have the potential to achieve improved prediction accuracy during such turbulent periods.

### 5.4.3 Forecast evaluation approaches

Regarding the performance of forecasts in out-of-sample tests, we continue to employ MSE and QLIKE as our evaluation methods. However, it is important to distinguish between the concept of forecast loss (FL) and the estimation criterion (EC), as they serve distinct purposes. FL assesses the performance of RV forecasts during out-of-sample testing, while EC is utilized for model estimation within the in-sample period.

In order to determine the significance of the performance improvement compared to the baseline models, we employ two commonly used statistical tests found in the literature. As suggested by [169], QLIKE demonstrates greater statistical power than MSE in the Diebold-Mariano (DM) test. Consequently, our focus in the analysis of the out-of-sample results is primarily on QLIKE.

- **Model Confidence Set (MCS)** was proposed by [100] to identify a subset of models with significantly superior performance from model candidates, at a given level of confidence. The MCS procedure renders it possible to make statements about the statistical significance from multiple pairwise comparisons. For additional details, we refer to the studies of [99, 100].

- **Diebold-Mariano (DM) test** was proposed by [63] to examine whether there are significant differences between two time-series forecasts. The DM test was further modified by [101], to account for serial dependence in forecasts. In addition to comparing errors for each individual stock, we also follow [92] to compare the cross-sectional average of prediction errors from two models. Further details of the DM test are available in [63].

## 5.5 Empirical analysis

In this section, we first introduce the data and provide details regarding the implementation. Subsequently, we present the main findings and conduct a stratified analysis to evaluate the performance across different market regimes.

### 5.5.1 Setup

The intraday data of Dow Jones Industrial Average (DJIA) components are obtained from the LOBSTER database.[10] The time period under consideration is from July 1, 2007 to Jun 30, 2021.[11] Following [24], we include only those stocks among the DJIA components that traded continuously throughout the entire period. As a result, 27 stocks are included in the final sample, and their ticker symbols are summarised in 5.1, where we also present summary statistics for the volatility estimates. Additionally, for robustness checks, we consider a larger universe of S&P100 components. Further details regarding this analysis can be found in Section 5.6.2.

Our out-of-sample forecast comparisons are based on the RV forecasts for the set of models introduced in Sections 5.2 and 5.3. All models are re-calibrated every month based on a rolling sample window of the past 1000 days, following [26, 198, 166]. Specifically, we use 36-month data for model training, and the recent 12-month data as the validation set to tune the hyperparameters and prevent overfitting.[12] Finally, testing data are the samples in the following month; they are out-of-sample in order to provide objective assessments of the model performance. To this end, in aggregate, we obtain a 10-year out-of-sample period, that is, from July 1, 2011 to June 30, 2021.

The parameters in $\text{HAR}_M$ and $\text{GHAR}_M$ are estimated by OLS using both the training and validation data, as there is no requirement for hyperparameter tuning.

[10] https://lobsterdata.com/

[11] The LOBSTER database contains data from June 27, 2007, up until the day before yesterday

[12] To examine the impact of validation dataset, we perform a robustness check for GNNHAR models in Section 5.6.1, and we conclude that the other choice of validation data does not significantly alter our findings.

To estimate the parameters in the proposed GNNHARs, we adopt the Adam optimiser ([119]).[13] When QLIKE is chosen as the EC, there are no available estimators in closed form. Therefore, we also employ Adam to optimise $\text{HAR}_Q$ and $\text{GHAR}_Q$ using both the training and validation data. Given the stochastic nature of the optimiser[14], we employ an ensemble approach to enhance the robustness of GNNHAR models and QLIKE-trained linear models (see [92, 229]). We train multiple models with random initialisation and obtain final predictions by averaging the outputs of all networks.

Following the convention of stochastic optimisation ([119]), we set the batch size to 32.[15] The learning rate for Adam is set to be $10^{-3}$. We stop the training procedure early if there is a sign of overfitting, that is, the training loss keeps dropping but validation loss increases beyond a tolerance level.

One-day forecasting is not the only time horizon of interest to practitioners. Following the convention established in the literature ([198, 228]), we also examine whether the proposed methods can be applied to various forecasting horizons, e.g. one week or one month. The weekly and monthly target volatility are defined as $\boldsymbol{RV}_{t:t+h} = \sum_{k=0}^{h} \boldsymbol{RV}_{t+k}$, where $h = 4$ and 21, respectively.

### 5.5.2 Main results

We begin our empirical analysis by comparing the out-of-sample performance of the competing models under consideration. Table 5.2 presents the ratio of forecast losses for each model relative to the $\text{HAR}_M$ model (i.e. HAR estimated by OLS).

Table 5.2 first highlights the consistent improvement of the GHAR model over the standard HAR model in both forecast losses (FL), implying the importance of graph information. Furthermore, the first two columns of Table 5.2, which represent results for the 1-day horizon, demonstrate that our proposed GNNHAR model with a single hidden layer ($\text{GNNHAR1L}_M$), further improves the performance of the linear model $\text{GHAR}_M$. This finding underscores the significance of incorporating nonlinearity when modelling the spillover effect. However, it is worth noting that the performance starts to decline when additional GNN layers are added, particularly with three layers.

[13] Adam is a popular stochastic optimisation algorithm for deep learning models and is very efficient to find the local minimum, especially with those non-convex and less smooth loss functions.

[14] The stochastic optimisation algorithms might be ended up with different local minima with different initial values.

[15] Mini-batch training is believed to improve generalisation performance, see [147].

Table 5.2: Out-of-sample forecast losses.

| | 1-Day | | 1-Week | | 1-Month | |
|---|---|---|---|---|---|---|
| | MSE | QLIKE | MSE | QLIKE | MSE | QLIKE |
| $\text{HAR}_M$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $\text{GHAR}_M$ | 0.927 | 0.983 | 0.904 | 0.987 | 0.975* | 1.036 |
| $\text{GNNHAR1L}_M$ | 0.907 | 0.979 | 0.940 | 0.943 | 1.021 | 0.968 |
| $\text{GNNHAR2L}_M$ | 0.967 | 0.977 | 1.034 | 0.953 | 1.134 | 1.032 |
| $\text{GNNHAR3L}_M$ | 1.210 | 0.982 | 1.014 | 0.961 | 1.046 | 0.958 |
| $\text{HAR}_Q$ | 0.927 | 0.981 | 0.939 | 0.945 | 1.069 | 0.986 |
| $\text{GHAR}_Q$ | 0.886 | 0.983 | 0.842* | 0.936 | 1.151 | 0.954† |
| $\text{GNNHAR1L}_Q$ | 0.867* | 0.961† | 0.855 | 0.913* | 1.179 | 0.965 |
| $\text{GNNHAR2L}_Q$ | 0.879 | 0.959* | 0.873 | 0.920 | 1.736 | 0.947* |
| $\text{GNNHAR3L}_Q$ | 0.894 | 0.963 | 1.185 | 0.942 | 1.502 | 0.971 |

*Note:* The table reports the ratios of forecast losses of various models compared to the standard $\text{HAR}_M$ model over the 1-day, 1-week, and 1-month horizons, respectively. The model with the lowest average out-of-sample loss is marked with an asterisk (*). A dagger (†) indicates models that yield as accurate forecasts as the best model at the 5% significance level based on the Model Confidence Set (MCS) test.

When considering models trained with QLIKE, the results for the 1-day horizon reveal that $\text{HAR}_Q$ achieves better forecasts than its counterpart $\text{HAR}_M$. $\text{GNNHAR1L}_Q$ further improves the predictive accuracy of $\text{GNNHAR1L}_M$ and yields the best (resp. second best) out-of-sample performance in terms of MSE (resp. QLIKE). Specifically, at the daily forecast horizon, $\text{GNNHAR1L}_Q$ has about 13% (resp. 4%) lower average forecast error in MSE (resp. QLIKE) compared to the standard $\text{HAR}_M$ model. In addition, the MCS test indicates that both $\text{GNNHAR1L}_Q$ and $\text{GNNHAR2L}_Q$ are included in the subset of best models, based on the QLIKE forecast loss. Interestingly, $\text{GNNHAR3L}_Q$ delivers worse out-of-sample performance than GNNs with one or two layers, yet still outperforms its counterpart trained with MSE. These findings suggest that QLIKE might serve as a more effective in-sample estimation criterion than MSE. In the subsequent sections, we will provide further analysis to delve into these results.

The results for weekly and monthly horizons presented in Table 5.2 demonstrate that models incorporating graph information (including GHAR and various GNNHAR models) exhibit significantly superior forecast accuracy compared to the HAR model over longer horizons, up to one week. Specifically, when examining the QLIKE loss for the 1-week forecast horizon, we observe that $\text{GNNHAR1L}_Q$ achieves the best out-of-sample performance. However, as the prediction horizon extends, the ratios approach or even exceed one, particularly for MSE. This suggests that longer-term forecasting becomes less sensitive to graph information. Additionally, we notice that

the discrepancy between the ratios based on MSE and QLIKE becomes more pronounced over longer horizons. One possible explanation is that the QLIKE loss is generally less impacted by extreme observations in the testing samples (see [168]). This is particularly relevant considering that such extreme observations may occur more frequently over longer horizons.

### 5.5.3 Market regimes

To assess the stability of performance across different market regimes, we perform a stratified out-of-sample analysis on two sub-samples: relatively calm periods when the RV of the S&P500 ETF index is below the 90% quantile of its entire sample distribution, and the turbulent periods when the RV is above its 90% quantile (see [166, 228]).

The results presented in Table 5.3 demonstrate that the enhancements achieved through the introduction of nonlinearity and the selection of QLIKE as the EC are generally consistent across different market regimes. Specifically, when considering calm days and the daily forecast horizon, the models GNNHAR1L$_M$ and GNNHAR2L$_M$ appear to be the most effective based on the MSE loss. On the other hand, when evaluating accuracy in terms of QLIKE, the models GNNHAR1L$_Q$ and GNNHAR2L$_Q$ provide the most precise forecasts. This outcome is expected since the volatility process tends to be more stable during calm periods. Consequently, if the forecast user has a specific preference for a particular loss function, it would be advisable to optimise the model parameters accordingly. In other words, for stationary time series, the alignment of the training loss (i.e. EC) and the testing loss (i.e. FL) may produce improved forecasts.

Nevertheless, when examining turbulent days and the daily forecast horizon, models trained with QLIKE exhibit greater percentage improvements compared to those trained with MSE across both losses. For instance, the average forecast MSE (QLIKE) loss of GNNHAR1L$_Q$ is approximately 13% (2%) lower than GNNHAR1L$_M$. This suggests that models trained with QLIKE may possess unique characteristics distinct from their MSE-trained counterparts during turbulent periods. This intriguing discovery will be further explored and analysed in the subsequent section.

In addition, when considering longer forecast horizons and periods of calmness, GNNHAR1L$_M$ produces significantly more accurate out-of-sample forecasts relative to other models in terms of MSE. Regarding the QLIKE accuracy, GNNHAR1L$_Q$ outperforms other models for the weekly horizon, while GNNHAR2L$_M$ emerges as the top-performing model for the monthly horizon. When transitioning to the volatile

periods, we continue to observe the superiority of QLIKE-trained models (especially $\text{GHAR}_Q$) over MSE-trained models, with the exception being the monthly forecast horizon and considering MSE as the FL.

### 5.5.4 Impact of evaluation criterion

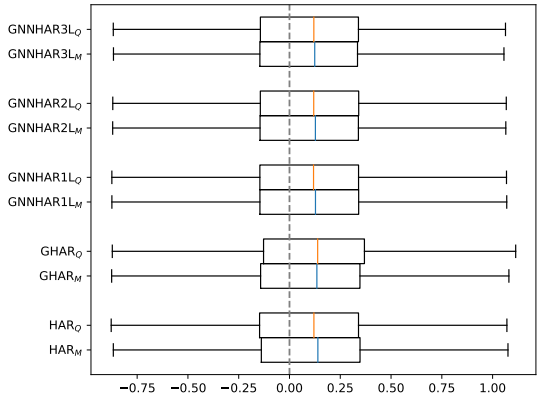
As previously mentioned, QLIKE deals with over- and under-predictions differently, which may account for the overall better performance of QLIKE-trained models compared to MSE-trained models. In light of this observation, we examine the forecast errors ($\widehat{RV}_{i,t}^{(F)} - RV_{i,t}$) and forecast ratios ($\widehat{RV}_{i,t}^{(F)} / RV_{i,t}$) over the entire testing period and various sub-periods.[16]

Figure 5.5 presents the box plots for forecast errors and ratios of various models. From subplots (a-b), we observe that in general, all models tend to exhibit a bias towards over-predictions (i.e. positive errors or ratios greater than 1) rather than under-predictions, aligning with the findings of [46]. Subplots (c-d) further unveil that this over-prediction tendency is primarily observed during calm periods. Conversely, subplots (e-f) indicate that these models are more inclined to under-predict volatilities during turbulent periods. This observation is not surprising, as the models do not explicitly incorporate any exogenous variables to aid in detecting changes in market conditions.

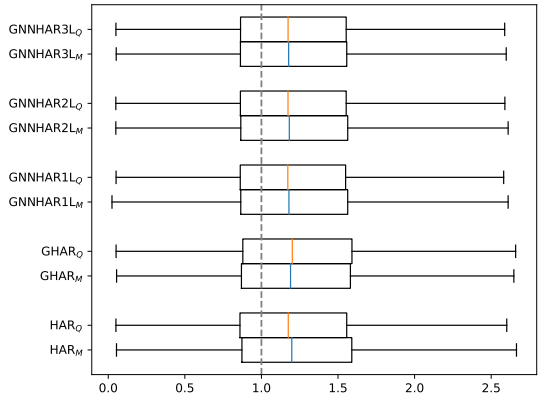
Furthermore, subplots (a-b) demonstrate that the bulk of the forecast errors (resp. ratios) of QLIKE-trained models is generally closer to zero (resp. one) compared to MSE-trained models. Specifically, subplots (c-d) reveal that QLIKE-trained models exhibit a reduced tendency to over-predict during calm periods, while subplots (e-f) suggest that they are less prone to excessive under-prediction during turbulent periods, when compared to the MSE-trained models.

[16] It is worth noting that the MSE loss is solely dependent on the forecast error, while QLIKE exclusively relies on the forecast ratio, as corroborated by [168].

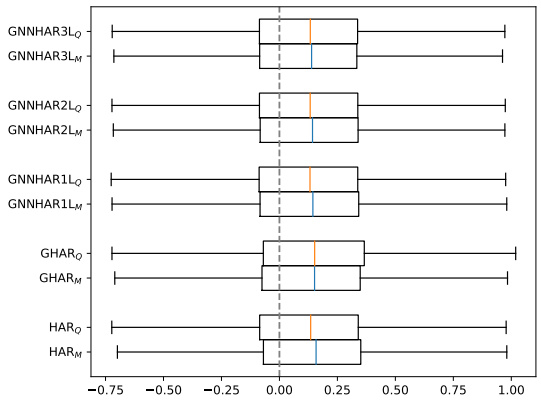
Figure 5.5: Grouped box plots for models trained with MSE or QLIKE.



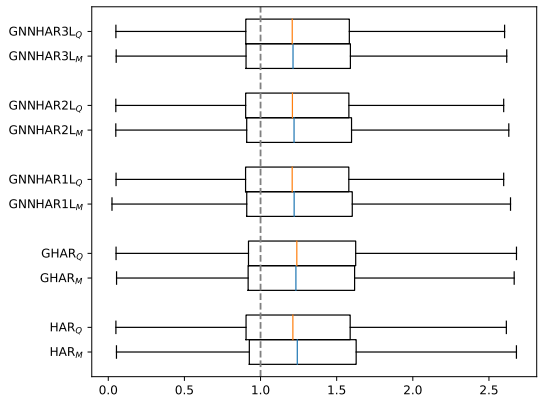

(a) Forecast errors.



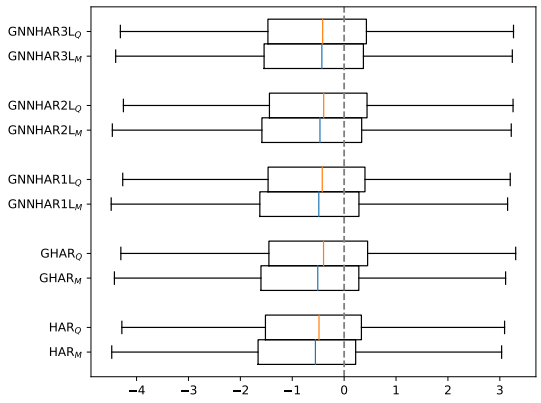

(b) Forecast ratios.



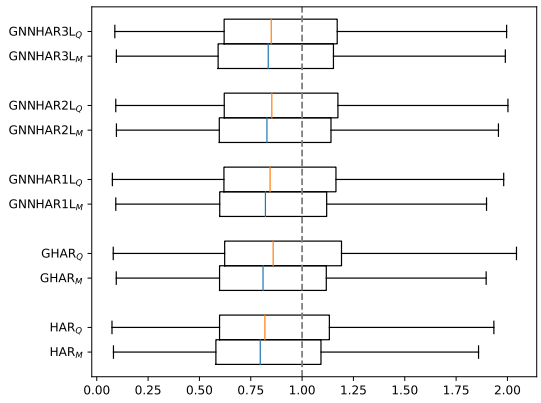

(c) Forecast errors during calm days.




(d) Forecast ratios during calm days.




(e) Forecast errors during turbulent days.

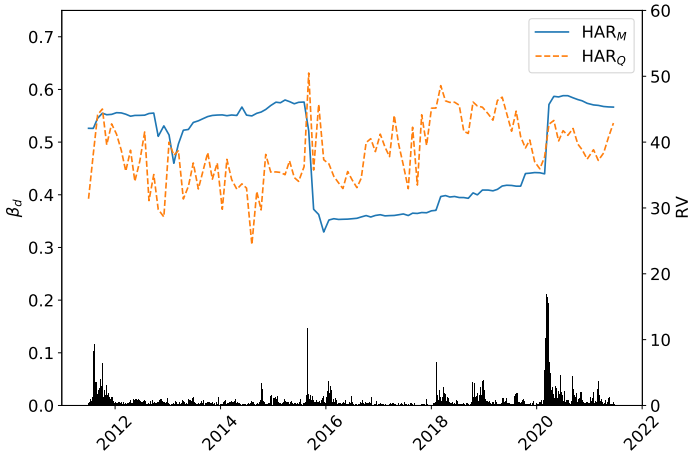



(f) Forecast ratios during turbulent days.

*Note:* This figure presents a box plot illustrating five summary statistics: the median, Q1 and Q3 quantiles, and two whiskers. Each group consists of two sets of box plots, with the top (resp. bottom) set representing models utilising QLIKE (resp. MSE) as EC.

In order to gain further insights from these findings, we present the trajectories of $\beta_d$ in the HAR models estimated using MSE or QLIKE in Figure 5.6. As anticipated, there are substantial temporal variations in the rolling estimates of both models. In general, the estimates of $\beta_d$ in $\text{HAR}_Q$ exhibit greater variability compared to those in $\text{HAR}_M$, which can be attributed to the stochastic nature of the optimisation algorithm employed in $\text{HAR}_Q$. However, the estimates of $\beta_d$ in $\text{HAR}_M$ reveal two prominent changes occurring during Dec 2015 - Feb 2016 and March 2020 - April 2020, albeit in different directions.[17] On the other hand, the patterns of $\beta_d$ in $\text{HAR}_Q$ are comparatively more stable, exhibiting an increasing trend during turbulent periods. This suggests that QLIKE-trained models have the ability to swiftly adapt to market changes and assign greater importance to observations associated with recent significant events. Future studies exploring the relationship between different estimators of HAR are therefore recommended.

Figure 5.6: Trajectories of $\beta_d$ in HAR trained with different losses.




*Note:* The left $y$-axis represents the estimated values of $\beta_d$ every month, while the right $y$-axis represents the daily RV of S&P500 ETF shown in bar-charts.

### 5.5.5 Impact of nonlinearity

To examine the necessity of nonlinear relations, we provide the following analysis that sheds light on the competitive performance of these models, particularly during volatile periods. Inspired by [43], we introduce, for each day $t$, the following metric to evaluate the Fraction of Variance of model $F$ which is Unexplained (FVU) by the

[17] These two periods correspond to significant market changes, namely the Chinese stock market turbulence and the Covid-19 pandemic.

standard $\text{HAR}_M$ model[18]

$$\text{FVU}_t = \frac{\sum_{i=1}^{N}\left(\widehat{RV}_{i,t}^{(F)} - \widehat{RV}_{i,t}^{(\text{HAR}_M)}\right)^2}{\sum_{i=1}^{N}\left(\widehat{RV}_{i,t}^{(F)} - \overline{RV}_{t}^{(F)}\right)^2}, \tag{5.14}$$

where $\overline{RV}_t^{(F)}$ is the average forecast RV of model $F$ across stocks on day $t$. At one extreme, $\text{FVU}_t = 0$ means that the $\text{HAR}_M$'s RV forecasts explain all of the variation in the predicted RVs provided by $F$; whereas, at the other extreme, $\text{FVU}_t = 1$ denotes that $\text{HAR}_M$ explains none of this variation.

Table 5.4 displays the Fraction of Variance Unexplained (FVU) of each model in comparison to $\text{HAR}_M$. It is worth noting that nonlinear models, particularly those with multiple hidden layers, exhibit higher FVU values, as anticipated. In addition, the results for 1-week and 1-month horizons in Table 5.4 suggest that the nonlinearity in volatility models seems to strengthen as the forecasting horizons increase. It is important to mention that the distinction between GHAR and GNNHAR1L lies in the presence of an additional hidden layer with a nonlinear activation function in GNNHAR1L. Consequently, the extra FVUs observed in GNNHAR1L can be considered as a measure of the degree of nonlinearity.

Table 5.4: FVU compared to $\text{HAR}_M$.

| | 1-Day | | 1-Week | | 1-Month | |
|---|---|---|---|---|---|---|
| | Bottom | Top | Bottom | Top | Bottom | Top |
| $\text{HAR}_M$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| $\text{GHAR}_M$ | 0.044 | 0.061 | 0.054 | 0.099 | 0.066 | 0.092 |
| $\text{GNNHAR1L}_M$ | 0.077 | 0.165 | 0.117 | 0.244 | 0.178 | 0.300 |
| $\text{GNNHAR2L}_M$ | 0.080 | 0.205 | 0.114 | 0.304 | 0.207 | 0.441 |
| $\text{GNNHAR3L}_M$ | 0.079 | 0.300 | 0.130 | 0.246 | 0.218 | 0.272 |
| $\text{HAR}_Q$ | 0.033 | 0.056 | 0.068 | 0.139 | 0.184 | 0.263 |
| $\text{GHAR}_Q$ | 0.077 | 0.128 | 0.108 | 0.216 | 0.228 | 0.779 |
| $\text{GNNHAR1L}_Q$ | 0.060 | 0.134 | 0.102 | 0.244 | 0.216 | 0.886 |
| $\text{GNNHAR2L}_Q$ | 0.060 | 0.184 | 0.118 | 0.379 | 0.283 | 1.391 |
| $\text{GNNHAR3L}_Q$ | 0.070 | 0.212 | 0.163 | 0.764 | 0.292 | 1.236 |

*Note:* The table reports the fraction of variance unexplained of multiple models compared by the baseline HAR, across different market regimes.

By comparing the first column and second column in Table 5.4, we observe higher FVU scores during turbulent days, regardless of the choice of EC. This suggests

[18]In fact, $\text{FVU}_t = 1 - R^2(\widehat{RV}_{i,t}^{(F)}, \widehat{RV}_{i,t}^{(\text{HAR}_M)})$, where $R^2$ is the coefficient of determination between the predicted RVs from the target model and the baseline model.

nonlinear spillover effects are most likely to exist in turbulent periods, rather than calm periods. In light of the results in Table 5.3, it can be inferred that a suitable level of model nonlinearity, such as that exhibited by GNNHAR1L, leads to improved predictive power during turbulent days. However, we find that overly complex models, such as GNNHAR3L, are unable to outperform the linear baseline. As a result, GNNHAR1L shows significant promise as a model for capturing nonlinearity, while avoiding the overfitting problem.

### 5.5.6 Impact of multi-hop neighbours

We utilise the DM test to evaluate the statistical significance of 2nd-hop neighbours by comparing the performance of GNNHAR2L and GNNHAR1L. Here, a positive (resp. negative) DM test value indicates the superiority of the GNNHAR1L (resp. GNNHAR2L) model. A $p$-value less than a given significance level $a$ rejects the null hypothesis that GNNHAR2L and GNNHAR1L have the same forecasting power at the $1-a$ confidence level.[19]

Figure 5.7 illustrates the main results from the above hypothesis test. In terms of individual stocks, GNNHAR2L$_M$ is only superior to GNNHAR1L$_M$ in forecasting AXP's volatilities, at the 5% confidence level. When considering the cross-sectional performance, the $p$-value is around 75%, from which we cannot reject the null hypothesis. This suggests that once the impact from itself and 1st-hop neighbours have been taken into account, 2-hop neighbours are not deemed necessary. The comparison between GNNHAR2L$_Q$ and GNNHAR1L$_Q$ indeed supports these findings.

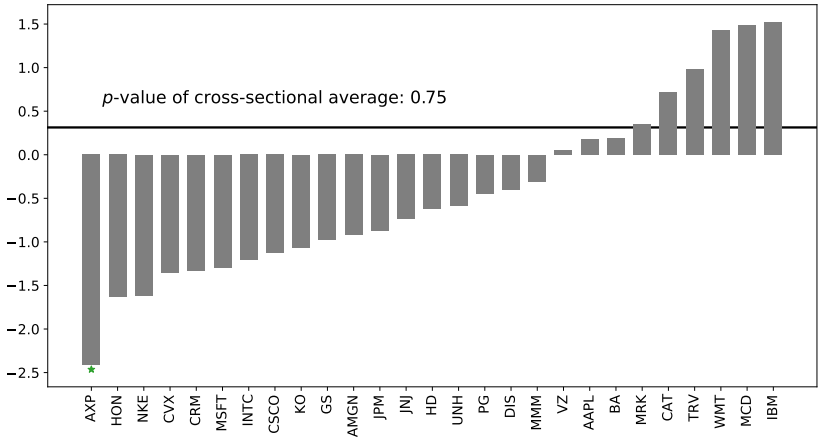
GNNs are known to suffer from the problem of **over-smoothing**, which is defined as the high similarity of node representations obtained at the output layer of GNNs, see [133]. The high similarity is often observed when stacking with multiple GNN layers that are more than necessary. With $K$ layers, every node receives information from its $K$-hop neighbours.[20] When $K$ is large, node representations obtained from GNN information propagation become indistinguishable and weaken the forecasting accuracy.

Following the convention in the GNN literature (e.g. [37]), we use the Mean Average Distance (MAD) to measure the similarity of node representations and identify whether there is any sign of over-smoothing in our GNNHAR models. MAD takes as

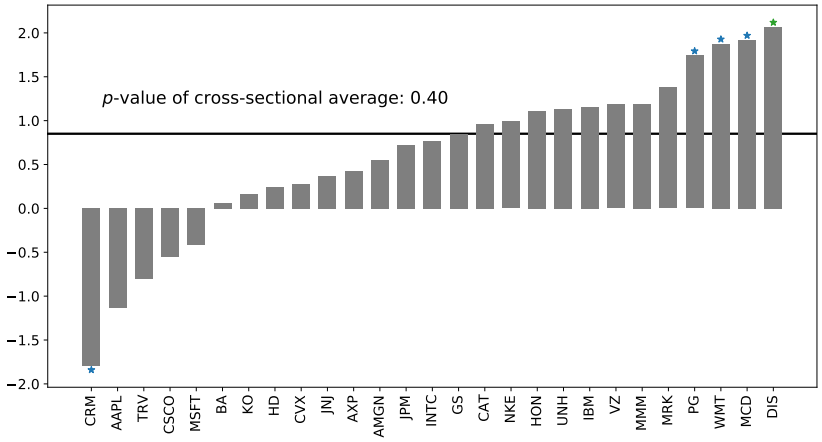
[19]We also conduct the same test to compare linear multi-hop graph models, i.e. GHAR and GHAR2Hop (see Appendix 5.4) and the conclusions are similar.

[20]This is also known as the receptive field of GNN. More details have been introduced in Section 5.2.

Figure 5.7: DM test between GNNHAR2L and GNNHAR1L.




(*a*) GNNHAR2L$_M$ vs GNNHAR1L$_M$




(*b*) GNNHAR2L$_Q$ vs GNNHAR1L$_Q$

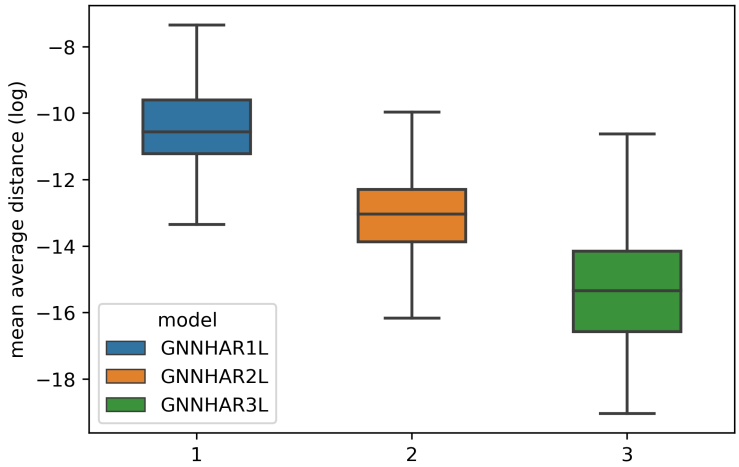
*Note:* A positive (negative) number indicates superiority for the GNNHAR1L (GNNHAR2L) model. The $y$-axis represents the DM test values based on QLIKE between GNNHAR2L and GNNHAR1L, while the $x$-axis lists the stock symbols. Stars indicate the $p$-value, with orange, green, and blue representing significance at the 1%, 5%, and 10% levels, respectively. The horizon line represents the cross-sectional DM test value and its corresponding $p$-value.

input the node representations $\boldsymbol{H} \in \mathbb{R}^{N \times D}$ obtained at the final layer of GNN, that is $\boldsymbol{H} = \text{GNN}(\boldsymbol{V_{:t-1}}, \boldsymbol{A})$ in (5.9), and is defined as follows [21]

$$\text{MAD} = \frac{\sum_{i=1}^{N} \bar{d}_i}{\sum_{i=1}^{N} \mathbb{1}_{\bar{d}_i > 0}}, \quad \text{where } \bar{d}_i = \frac{\sum_{j=1}^{N} \bar{\boldsymbol{D}}_{ij}}{\sum_{j=1}^{N} \mathbb{1}_{\bar{\boldsymbol{D}}_{ij} > 0}}. \tag{5.15}$$

$\bar{\boldsymbol{D}}$ is the masked cosine distance matrix, i.e. $\bar{\boldsymbol{D}} = \boldsymbol{D} \circ \boldsymbol{A}$, where $\circ$ denotes the Hadamard product (element-wise multiplication), and $\boldsymbol{D}_{ij} = 1 - \frac{\boldsymbol{H}[i,:] \cdot \boldsymbol{H}[j,:]}{\|\boldsymbol{H}[i,:]\| \|\boldsymbol{H}[j,:]\|}$. In the above definition, $\bar{d}_i$ is the average distance between the representations of node $i$ and its connected nodes. Overall, MAD represents an average level of how a node representation is similar to the representations of its connected neighbours in a graph.

Figure 5.8: Smoothness of GNNHARs.




*Note:* A small value of mean average distance (MAD) indicates a high similarity between node representations at the output layer of GNN.

In Figure 5.8, three boxes represent GNNHAR models with 1, 2, and 3 GNN layers trained with MSE.[22] Each box corresponds to the MAD values on a logarithmic scale, calculated across all out-of-sample samples. As the number of GNN layers increases, there is a decrease in log MAD that corresponds to an increase in smoothness. The 3-layer GNNHAR has the lowest MAD score, suggesting potential over-smoothing of node representations. Specifically, the rows of $\text{GNN}(\boldsymbol{V_{:t-1}}, \boldsymbol{A})$ from GNNHAR3L in (5.9), become too similar to provide any node specific predictive information. This partially explains the inferior performance of GNNHAR3L, as shown in Table 5.2.

[21] $\boldsymbol{H}$ is the (unweighted) average of the hidden representations obtained from GNNHARs in our ensemble set.

[22] Similar results (unreported) are observed for GNNHARs trained with QLIKE.

## 5.6 Robustness tests

After presenting the main empirical results and analyzing the model performance across different market periods, we shift our focus to evaluating the robustness of the proposed models by considering two aspects: (i) an alternative validation set size, and (ii) a larger universe.

### 5.6.1 Alternative validation set size

Our main analysis is based on rolling samples of 4 years, with the first approximately 3 years as training data, and the recent 1 year as validation data. Using a smaller validation data set, such as 1 month, does not significantly alter our findings, as shown in Table 5.5.

Table 5.5: Out-of-sample forecast losses under a smaller validation data set.

| | 1-Day | | 1-Week | | 1-Month | |
|---|---|---|---|---|---|---|
| | MSE | QLIKE | MSE | QLIKE | MSE | QLIKE |
| $HAR_M$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $GHAR_M$ | 0.927 | 0.983 | 0.904 | 0.987 | 0.975* | 1.031 |
| $GNNHAR1L_M$ | 0.942 | 0.978 | 0.931 | 0.945 | 1.008 | 0.975 |
| $GNNHAR2L_M$ | 0.984 | 0.984 | 1.005 | 0.956 | 1.138 | 1.033 |
| $GNNHAR3L_M$ | 1.078 | 1.002 | 1.035 | 0.954 | 1.068 | 0.958 |
| $HAR_Q$ | 0.936 | 0.986 | 0.945 | 0.944 | 1.218 | 0.959 |
| $GHAR_Q$ | 0.942 | 0.982 | 0.993 | 0.945 | 1.174 | 0.954 |
| $GNNHAR1L_Q$ | 0.889* | 0.967* | 0.875† | 0.912 | 1.226 | 0.961 |
| $GNNHAR2L_Q$ | 0.896 | 0.968† | 0.861* | 0.907* | 1.510 | 0.925* |
| $GNNHAR3L_Q$ | 1.152 | 0.981 | 1.060 | 0.929 | 1.572 | 0.972 |

*Note:* The table reports the out-of-sample losses of various models using 47 months as training data and the recent 1 month as validation data. The model with the lowest average out-of-sample loss is marked with an asterisk (*). A dagger (†) indicates models that yield as accurate forecasts as the best model at the 5% significance level based on the MCS test.

### 5.6.2 Larger universe

To further assess the robustness of our findings and ascertain that they are not specific to the stocks under current consideration, we repeat the out-of-sample analysis using a larger data set, including the components of the S&P100 index.[23] The experimental

[23]Details about the data are provided in Table 5.1.

setups and the hyperparameter choices in GNNHAR remain the same as those described in Section 5.5.1. As illustrated in Table 5.7, in the volatility spillover graphs for the S&P100 index components, each node is connected to other nodes within a maximum of 5 steps. Consequently, we extend our analysis to include 4-layer and 5-layer versions of the GNNHAR model.

Table 5.6: Out-of-sample forecast losses on S&P100.

| | 1-Day | | 1-Week | | 1-Month | |
|---|---|---|---|---|---|---|
| | MSE | QLIKE | MSE | QLIKE | MSE | QLIKE |
| $\text{HAR}_Q$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $\text{GHAR1L}_Q$ | 0.948 | 0.988 | 0.909 | 0.994 | 0.972* | 0.986 |
| $\text{GNNHAR1L}_Q$ | 0.963 | 0.986 | 0.951 | 0.944 | 1.027 | 1.092 |
| $\text{GNNHAR2L}_Q$ | 1.072 | 0.988 | 1.031 | 0.954 | 1.092 | 1.000 |
| $\text{GNNHAR3L}_Q$ | 1.061 | 0.986 | 1.029 | 0.959 | 0.992 | 0.967 |
| $\text{GNNHAR4L}_Q$ | 1.047 | 0.992 | 1.042 | 0.975 | 1.079 | 0.978 |
| $\text{GNNHAR5L}_Q$ | 1.090 | 0.997 | 1.057 | 0.986 | 1.109 | 1.038 |
| $\text{HAR}_Q$ | 0.949 | 0.983 | 0.937 | 0.947 | 1.171 | 0.991 |
| $\text{GHAR}_Q$ | 0.919 | 0.984 | 0.850* | 0.922 | 1.154 | 0.939* |
| $\text{GNNHAR1L}_Q$ | 0.917† | 0.969 | 0.858 | 0.916 | 1.231 | 1.017 |
| $\text{GNNHAR2L}_Q$ | 0.915* | 0.969 | 0.909 | 0.915* | 1.206 | 0.941† |
| $\text{GNNHAR3L}_Q$ | 0.938 | 0.966* | 1.178 | 0.968 | 1.523 | 0.946 |
| $\text{GNNHAR4L}_Q$ | 0.985 | 0.970 | 1.165 | 0.972 | 1.563 | 0.971 |
| $\text{GNNHAR5L}_Q$ | 0.951 | 0.968 | 1.193 | 0.975 | 1.741 | 0.989 |

*Note:* The table reports the ratios of forecast losses of various models compared to the standard $\text{HAR}_M$ model over the 1-day, 1-week, and 1-month horizons, respectively. The model with the lowest average out-of-sample loss is marked with an asterisk (*). A dagger (†) indicates models that yield as accurate forecasts as the best model at the 5% significance level based on the MCS test.

Table 5.7: Frequency (in percentage) of the shortest path distance.

| SPD | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| DJIA | 57.7 | 41.8 | 0.5 | 0.0 | 0.0 |
| S&P100 | 24.3 | 61.2 | 12.0 | 2.2 | 0.3 |

*Note:* For example, in the case of S&P100, 12% of pairs of nodes have their shortest path distance of size 3.

The out-of-sample forecasting performance on the volatilities of S&P100 components is presented in Table 5.6. Firstly, we observe that GHAR consistently enhances forecasting accuracy compared to the traditional HAR model. Additionally, the nonlinear variant, GNNHAR1L, further improves upon the performance of GHAR over

the 1-day horizon. Generally, as we increase the number of layers in the GNNHAR models, their forecasting performance tends to decline. Nevertheless, we still observe the benefits of training models with the QLIKE loss function. In summary, the findings presented in Table 5.6 align closely with those observed for DJIA30, providing consistent results across both data sets.

## 5.7 Conclusions

In this chapter, we present a model GNNHAR for predicting realised volatilities (RVs). Our model incorporates volatility spillover effects in the U.S. equity market. The major experiment results indicate that the information from multi-hop neighbours in the financial network does not significantly enhance the prediction of the target stock's volatility. Nevertheless, the integration of nonlinear spillover effects appears to contribute to improved RV forecasting accuracy. Further, our results suggest that using QLIKE as the training loss function, rather than the more traditional MSE, can improve the accuracy of volatility predictions. QLIKE-trained nonlinear models also show increased robustness in turbulent market conditions compared to more stable periods. Extensive evaluation tests and an alternative setting support the reliability and effectiveness of our proposed method.

One interesting direction is to further investigate why utilising QLIKE instead of MSE, as the evaluation criterion, improves forecasting accuracy. While [98] asserted the asymptotic efficiency of the likelihood-based estimator, our settings differ from theirs in that they assumed the likelihood function is in conjunction with the forecasting loss. Conversely, [168] claimed that MSE is more sensitive to extreme observations than QLIKE, but there is a lack of theoretical underpinnings on how this might improve the predictive powers in various market conditions.

Another interesting direction to explore is the robustness of the proposed methods when applied to different approaches in constructing financial graphs, such as those based on supply-chain ([103]) and analyst co-coverage ([3]). It would be valuable to investigate whether these graphs provide unique information content and have the potential to enhance performance.

Table 5.1: Summary statistics of realized volatility.

| Ticker | Mean | Std | Min | 25% | 50% | 75% | Max | DJIA | S&P100 |
|---|---|---|---|---|---|---|---|---|---|
| AAPL | 2.30 | 3.39 | 0.07 | 0.70 | 1.25 | 2.46 | 38.30 | ✓ | ✓ |
| ABT | 1.41 | 1.95 | 0.12 | 0.57 | 0.89 | 1.50 | 34.32 | | ✓ |
| ACN | 1.72 | 2.79 | 0.14 | 0.58 | 0.92 | 1.76 | 54.88 | | ✓ |
| ADBE | 2.53 | 3.34 | 0.16 | 0.93 | 1.54 | 2.76 | 45.55 | | ✓ |
| ADP | 1.41 | 2.51 | 0.10 | 0.49 | 0.78 | 1.39 | 44.36 | | ✓ |
| AMGN | 1.91 | 2.34 | 0.16 | 0.82 | 1.27 | 2.14 | 33.44 | ✓ | ✓ |
| AMT | 2.16 | 3.83 | 0.19 | 0.68 | 1.11 | 2.10 | 53.19 | | ✓ |
| AMZN | 3.22 | 4.48 | 0.11 | 1.02 | 1.84 | 3.59 | 62.14 | | ✓ |
| AXP | 3.19 | 6.32 | 0.12 | 0.64 | 1.15 | 2.67 | 91.45 | ✓ | ✓ |
| BA | 2.69 | 5.00 | 0.13 | 0.78 | 1.35 | 2.60 | 90.65 | ✓ | ✓ |
| BAC | 4.93 | 11.48 | 0.10 | 1.01 | 1.81 | 3.68 | 135.30 | | ✓ |
| BDX | 1.37 | 1.84 | 0.13 | 0.54 | 0.86 | 1.48 | 28.52 | | ✓ |
| BMY | 1.77 | 2.20 | 0.08 | 0.72 | 1.14 | 1.93 | 30.75 | | ✓ |
| BSX | 3.15 | 4.39 | 0.20 | 1.14 | 1.92 | 3.35 | 55.28 | | ✓ |
| C | 5.48 | 14.6 | 0.15 | 0.99 | 1.82 | 3.94 | 257.34 | | ✓ |
| CAT | 2.79 | 4.00 | 0.15 | 0.94 | 1.58 | 2.89 | 45.26 | ✓ | ✓ |
| CB | 1.82 | 3.66 | 0.07 | 0.44 | 0.75 | 1.62 | 61.54 | | ✓ |
| CI | 3.65 | 6.92 | 0.19 | 1.01 | 1.75 | 3.28 | 164.21 | | ✓ |
| CMCSA | 2.35 | 3.57 | 0.13 | 0.78 | 1.29 | 2.47 | 43.26 | | ✓ |
| CME | 3.07 | 5.49 | 0.18 | 0.84 | 1.38 | 2.72 | 68.79 | | ✓ |
| COP | 3.12 | 5.18 | 0.16 | 0.98 | 1.71 | 3.26 | 75.84 | | ✓ |
| COST | 1.44 | 2.11 | 0.0 | 0.51 | 0.79 | 1.44 | 26.30 | | ✓ |
| CRM | 4.00 | 4.93 | 0.22 | 1.44 | 2.41 | 4.64 | 61.67 | ✓ | ✓ |
| CSCO | 1.98 | 2.92 | 0.14 | 0.70 | 1.13 | 2.09 | 43.74 | ✓ | ✓ |
| CVS | 1.99 | 3.15 | 0.13 | 0.70 | 1.17 | 2.03 | 53.28 | | ✓ |
| CVX | 2.03 | 3.51 | 0.13 | 0.61 | 1.07 | 2.04 | 48.07 | ✓ | ✓ |
| D | 1.44 | 2.56 | 0.1 | 0.56 | 0.85 | 1.40 | 40.39 | | ✓ |
| DHR | 1.6 | 2.41 | 0.14 | 0.54 | 0.95 | 1.67 | 29.78 | | ✓ |
| DIS | 1.89 | 3.04 | 0.12 | 0.60 | 1.01 | 1.88 | 40.56 | ✓ | ✓ |
| DUK | 1.32 | 2.20 | 0.06 | 0.50 | 0.78 | 1.32 | 36.07 | | ✓ |
| FIS | 1.89 | 3.48 | 0.15 | 0.59 | 0.97 | 1.74 | 62.40 | | ✓ |
| FISV | 1.71 | 2.82 | 0.15 | 0.58 | 0.93 | 1.69 | 53.36 | | ✓ |
| GE | 3.08 | 5.54 | 0.09 | 0.68 | 1.43 | 3.05 | 77.33 | | ✓ |
| GILD | 2.36 | 2.67 | 0.23 | 1.03 | 1.55 | 2.64 | 33.62 | | ✓ |
| GOOG | 1.94 | 2.72 | 0.11 | 0.64 | 1.08 | 2.07 | 30.36 | | ✓ |
| GS | 3.24 | 6.27 | 0.19 | 0.92 | 1.49 | 2.81 | 112.41 | ✓ | ✓ |
| HD | 2.11 | 3.59 | 0.15 | 0.62 | 1.02 | 2.01 | 48.22 | ✓ | ✓ |
| HON | 1.85 | 3.25 | 0.1 | 0.52 | 0.97 | 1.84 | 49.64 | ✓ | ✓ |
| IBM | 1.38 | 2.33 | 0.11 | 0.47 | 0.75 | 1.34 | 30.22 | ✓ | ✓ |
| INTC | 2.29 | 3.12 | 0.14 | 0.86 | 1.39 | 2.44 | 42.90 | ✓ | ✓ |
| INTU | 2.00 | 2.81 | 0.15 | 0.75 | 1.22 | 2.15 | 38.91 | | ✓ |

| Ticker | Mean | Std | Min | 25% | 50% | 75% | Max | DJIA | S&P100 |
|---|---|---|---|---|---|---|---|---|---|
| ISRG | 3.19 | 4.31 | 0.22 | 1.10 | 1.81 | 3.38 | 46.66 | | ✓ |
| JNJ | 0.92 | 1.56 | 0.06 | 0.35 | 0.54 | 0.90 | 24.74 | ✓ | ✓ |
| JPM | 3.46 | 7.04 | 0.15 | 0.74 | 1.36 | 2.82 | 108.17 | ✓ | ✓ |
| KO | 0.99 | 1.68 | 0.07 | 0.37 | 0.58 | 1.00 | 25.00 | ✓ | ✓ |
| LLY | 1.59 | 2.29 | 0.13 | 0.61 | 0.98 | 1.70 | 35.90 | | ✓ |
| LMT | 1.64 | 2.59 | 0.12 | 0.56 | 0.94 | 1.64 | 35.79 | | ✓ |
| LOW | 2.71 | 4.20 | 0.17 | 0.88 | 1.45 | 2.77 | 73.32 | | ✓ |
| MA | 2.86 | 4.60 | 0.13 | 0.73 | 1.31 | 2.79 | 52.20 | | ✓ |
| MCD | 1.17 | 2.15 | 0.08 | 0.39 | 0.61 | 1.13 | 37.57 | ✓ | ✓ |
| MDT | 1.5 | 2.19 | 0.13 | 0.59 | 0.93 | 1.57 | 36.66 | | ✓ |
| MMM | 1.43 | 2.25 | 0.08 | 0.46 | 0.81 | 1.49 | 31.11 | ✓ | ✓ |
| MO | 1.41 | 2.25 | 0.06 | 0.52 | 0.84 | 1.43 | 39.67 | | ✓ |
| MRK | 1.65 | 2.45 | 0.12 | 0.58 | 0.92 | 1.74 | 30.99 | ✓ | ✓ |
| MS | 5.74 | 14.30 | 0.20 | 1.25 | 2.18 | 4.33 | 286.91 | | ✓ |
| MSFT | 1.82 | 2.51 | 0.11 | 0.67 | 1.09 | 1.92 | 30.64 | ✓ | ✓ |
| NFLX | 5.53 | 5.69 | 0.36 | 2.14 | 3.78 | 6.75 | 72.86 | | ✓ |
| NKE | 2.03 | 3.00 | 0.14 | 0.74 | 1.15 | 2.02 | 47.87 | ✓ | ✓ |
| NVDA | 5.14 | 6.03 | 0.40 | 1.83 | 3.2 | 5.96 | 72.25 | | ✓ |
| ORCL | 1.90 | 2.84 | 0.08 | 0.66 | 1.14 | 2.05 | 44.23 | | ✓ |
| PEP | 1.02 | 1.78 | 0.06 | 0.37 | 0.58 | 1.01 | 28.18 | | ✓ |
| PFE | 1.55 | 2.07 | 0.14 | 0.59 | 0.95 | 1.67 | 26.54 | | ✓ |
| PG | 1.00 | 1.76 | 0.09 | 0.38 | 0.58 | 0.98 | 31.60 | ✓ | ✓ |
| PNC | 3.64 | 7.52 | 0.16 | 0.79 | 1.38 | 3.04 | 141.27 | | ✓ |
| QCOM | 2.46 | 3.39 | 0.10 | 0.81 | 1.49 | 2.77 | 42.15 | | ✓ |
| SBUX | 2.45 | 3.90 | 0.18 | 0.71 | 1.24 | 2.48 | 63.45 | | ✓ |
| SO | 1.19 | 1.98 | 0.12 | 0.47 | 0.72 | 1.22 | 36.40 | | ✓ |
| SYK | 1.67 | 2.61 | 0.08 | 0.62 | 0.98 | 1.76 | 49.51 | | ✓ |
| T | 1.49 | 2.55 | 0.08 | 0.47 | 0.76 | 1.39 | 32.03 | | ✓ |
| TGT | 2.46 | 4.02 | 0.11 | 0.76 | 1.24 | 2.34 | 53.02 | | ✓ |
| TJX | 2.33 | 3.34 | 0.16 | 0.76 | 1.24 | 2.53 | 55.49 | | ✓ |
| TMO | 1.89 | 2.74 | 0.16 | 0.71 | 1.14 | 1.99 | 40.82 | | ✓ |
| TRV | 2.04 | 4.09 | 0.11 | 0.49 | 0.81 | 1.76 | 57.95 | ✓ | |
| TXN | 2.33 | 3.02 | 0.16 | 0.84 | 1.41 | 2.57 | 48.68 | | ✓ |
| UNH | 2.70 | 4.34 | 0.16 | 0.78 | 1.35 | 2.57 | 52.54 | ✓ | ✓ |
| UNP | 2.53 | 3.94 | 0.14 | 0.83 | 1.39 | 2.52 | 45.94 | | ✓ |
| UPS | 1.58 | 2.35 | 0.10 | 0.51 | 0.88 | 1.72 | 31.67 | | ✓ |
| USB | 3.20 | 6.88 | 0.13 | 0.62 | 1.16 | 2.64 | 95.38 | | ✓ |
| VZ | 1.40 | 2.36 | 0.10 | 0.50 | 0.77 | 1.33 | 34.19 | ✓ | ✓ |
| WFC | 4.05 | 8.89 | 0.11 | 0.73 | 1.39 | 3.24 | 106.81 | | ✓ |
| WMT | 1.18 | 1.76 | 0.11 | 0.45 | 0.67 | 1.18 | 27.18 | ✓ | ✓ |

*Note:* The table reports summary statistics for the daily realized volatility of stocks in DJIA30 or S&P100. The statistics are averaged across each trading day.

Table 5.3: Stratified out-of-sample forecast losses.

| | 1-Day | | 1-Week | | 1-Month | |
|---|---|---|---|---|---|---|
| | MSE | QLIKE | MSE | QLIKE | MSE | QLIKE |
| | | | Panel A: Bottom 90% | | | |
| $HAR_M$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $GHAR_M$ | 0.961 | 0.998 | 0.949 | 1.001 | 0.967 | 1.027 |
| $GNNHAR1L_M$ | 0.943* | 0.998 | 0.883* | 0.960† | 0.923* | 0.924† |
| $GNNHAR2L_M$ | 0.944† | 0.990 | 0.901 | 0.954† | 0.946† | 0.921* |
| $GNNHAR3L_M$ | 0.957 | 0.987 | 0.911 | 0.965 | 0.937† | 0.930† |
| $HAR_Q$ | 1.010 | 0.984 | 1.005 | 0.955† | 1.159 | 0.942† |
| $GHAR_Q$ | 0.989 | 1.007 | 1.076 | 1.001 | 1.257 | 1.084 |
| $GNNHAR1L_Q$ | 0.967 | 0.978* | 0.944 | 0.943* | 1.478 | 0.977 |
| $GNNHAR2L_Q$ | 0.976 | 0.979† | 0.985 | 0.947† | 1.433 | 0.973 |
| $GNNHAR3L_Q$ | 0.970 | 0.980† | 1.062 | 0.957 | 1.662 | 0.969 |
| | | | Panel B: Top 10% | | | |
| $HAR_M$ | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| $GHAR_M$ | 0.916 | 0.910 | 0.897 | 0.959 | 0.976* | 1.043 |
| $GNNHAR1L_M$ | 0.895 | 0.903 | 0.949 | 0.908 | 1.033 | 1.007 |
| $GNNHAR2L_M$ | 1.102 | 0.915 | 1.056 | 0.951 | 1.157 | 1.131 |
| $GNNHAR3L_M$ | 1.293 | 0.958 | 1.030 | 0.952 | 1.059 | 0.982 |
| $HAR_Q$ | 0.900 | 0.965 | 0.928 | 0.925 | 1.059 | 1.024 |
| $GHAR_Q$ | 0.852 | 0.867† | 0.804* | 0.799* | 1.149 | 0.841* |
| $GNNHAR1L_Q$ | 0.834* | 0.879 | 0.841 | 0.848 | 1.143 | 0.955 |
| $GNNHAR2L_Q$ | 0.848 | 0.862* | 0.924 | 0.861 | 1.773 | 0.886 |
| $GNNHAR3L_Q$ | 0.868 | 0.882 | 1.205 | 0.909 | 1.483 | 0.973 |

*Note:* The table reports stratified losses during trading days with the bottom 90% (Panel A) and the top 10% (Panel B) RV of the S&P500 ETF index over the 1-day, 1-week, and 1-month horizons, respectively. The model with the lowest average out-of-sample loss is marked with an asterisk (*). A dagger (†) indicates models that yield as accurate forecasts as the best model at the 5% significance level based on the MCS test.

# Chapter 6

# Conclusions

## Contents



In this thesis, we delved into graph-based learning and proposed two innovative frameworks to address the challenges of model-based graph learning. We also demonstrated how graph-based learning and inference can enhance investment strategies and risk management in quantitative finance.

## 6.1 Graph Learning

In this thesis, we investigate model-based graph learning from two new perspectives - functional viewpoints and deep learning. Based on each of them, we proposed a novel framework. In Chapter 2, Kernel Graph Learning (KGL) revisits the graph signal processing-based graph learning model from a functional perspective, offering robust graph learning even with noisy and missing data. L2G in Chapter 3, on the other hand, employs a deep learning architecture to transform optimisation-based graph learning into a parametric function mapping task, ensuring faster inference and precision in learning intricate graph topologies.

In particular, the goal of KGL is to address the quality of the input signals for graph learning. Many studies in the field often operate under the assumption that a complete data set is available for all the entities being studied, covering every vertex in the graph. However, in many practical scenarios, such as environmental networks, we face instances where only partial data is accessible. This might be due to difficulties in data acquisition from certain sensors, or perhaps the prohibitive costs associated with comprehensive observations. The challenge lies in designing

graph-learning methods that can effectively handle such incomplete data sets, and understanding how these data gaps might impact learning performance. It is crucial to note a bi-directional relationship between the input data and the output graph - data distribution affected by the graph structure, and meanwhile the graph structure inferred from data realisation. The model specification of KGL can well describe such a bidirectional relationship. Our extensive synthetic experiments provide empirical backing to our claims, showcasing the method's efficacy and efficiency. The results underscore its capability to unearth meaningful topologies even when faced with noisy, incomplete, and interdependent graph signals.

The goal of L2G is to more accurately capture flexible and intricate topological properties. In modern model-based graph learning, the challenge is also to ensure that these properties are representative of inherent data structures, especially when such characteristics cannot be straightforwardly expressed through a regularisation function. L2G offers a unique advantage; it can discern a data-to-graph mapping, harnessing the inductive bias presented by a classical iterative algorithm focused on solving structural-regularised Laplacian quadratic minimisation. Such inductive biases can stem from a variety of sources, whether empirical studies, domain expertise, or historical data, making the model's interpretation more precise, its accuracy improved, and computations potentially faster. Yet, understanding that there are scenarios where topological priors might not be immediately available, our method is designed to explore a broader spectrum of graph configurations. To push the boundaries of what these structural regularises can achieve, we introduce a topological difference variational autoencoder. This addition not only amplifies the expressiveness of the structural regularises but also imbues them with greater flexibility, ensuring that even in the absence of clear priors, the intrinsic structures within the data are unveiled and employed effectively.

In light of these technical advances in model-based graph learning, future research can focus on several directions:

**Scalability** In Chapter 2, we provided an efficient implementation for KGL that leverages the Kronecker structure of kernel matrices. Despite this, the computational complexity still scales cubically with the number of nodes or signals. This cubic complexity implies that as the size of the graph increases, the time and resources required for computation also increase, potentially exponentially. This makes it challenging to apply KGL to large graphs of thousands of nodes. To address this issue, we suggest that future research should focus on enhancing computational efficiency. One

potential approach is the adoption of advanced techniques in large-scale kernel-based methods, such as random Fourier features [78]. This technique has the potential to reduce computational complexity, thereby making the application of KGL to larger graphs more feasible.

With L2G, the challenge lies in the quadratic expansion of the variable space as the graph size increases. This expansion makes the application of L2G to very large graphs (with millions of nodes) constrained by the computational budget. It should be noted that this issue of quadratic complexity is not unique to L2G but is common to most existing model-based graph learning models [67, 66, 114]. In literature, attempts have been made to reduce the complexity using heuristic edge pre-selection [113, 165]. In a related area of estimating large inverse covariance matrices, a recent attempt, BigQUIC, can solve a one-million dimensional sparse graph from an objective function of Gaussian MLE optimisation using a single machine in a reasonable amount of time [105]. This approach is based on a novel block-coordinate descent method, with the blocks chosen via a clustering scheme. It could provide an inspiration for learning a graph representation matrix (such as adjacency or Laplacian matrices) by solving a matrix optimisation in a more scalable manner.

**Temporal Dynamics** In many real-world scenarios, it has been observed that graphs are evolving over time, e.g. social networks [186], financial systems of cash flows and assets [159] and biological interactomes [171]. The main approach of this thesis, similar to the majority of existing research, is to model the static snapshots of graph structures. These models are typically based on a predefined set of node observations. Recently, there is an increasing need for advanced methodologies capable of adaptively tracking and integrating changes in graph topologies over time [188, 84, 12, 148]. Such methods would more effectively capture the inherent dynamism of these networks.

A significant portion of research on learning dynamic graphs has been confined to the domain of discrete-time dynamics, formulated as a time series of graph snapshots [97, 221]. This approach, while useful, falls short in representing more complex, real-world scenarios. For instance, in social networks, graphs are not only continuous where edges may emerge at any moment, but also evolving with new nodes joining incessantly. Recent advancements have begun to address continuous-time scenarios, offering a deeper understanding of these dynamic networks [53]. Nevertheless, there remains a substantial gap in research, particularly in terms of scalability and applicability across various scenarios where dynamics differ. This gap highlights the need for

further exploration and development of models that are both versatile and efficient in capturing the complexities of continuously evolving graph structures.

**Downstream Task Integration** Graph machine learning aims to learn a vector representation for every node or the graph as a whole and make a better inference. As a subproblem, graph learning aims to uncover the intricate connections between data entities. Currently, a significant gap persists between the process of graph learning and its subsequent utilisation in tasks such as graph-based prediction or inference. Depending on the application, distinct graph topological properties might be emphasised, influencing their effectiveness. For instance, while a social network is expected to be sparse, showcasing prominent hubs, a financial network might ideally highlight communities of assets, aiding in diversifying investments.

Our research pioneers the fusion of deep learning and model-based graph learning. Leveraging the layered architecture intrinsic to deep learning, we adeptly integrate graph learning layers with downstream inference layers, leading to an end-to-end training approach as detailed in Chapter 4. However, challenges persist: deep learning optimisation is stochastic and difficult to converge, and data scarcity in many domains complicates the use of large training datasets. Exploring solutions to these issues remains a promising avenue for future endeavours.

## 6.2 Quantitative Finance

This thesis delves deeply into the practical potential of graph-based learning and inference, providing a novel perspective to comprehend the intricate interconnections within financial systems and enhancing the forecasting of financial time series on that foundation. Our study prominently highlights two pivotal subareas: the network momentum strategy and realised volatility forecasting.

In chapter 4, we introduce a novel end-to-end graph machine learning framework which encompasses two model variants: L2GMOM and L2GMOM_SR. Notably, these models go beyond merely enhancing the profitability and risk management of traditional momentum strategies; they provide profound insights into the underlying asset interconnections. These insights bring a richer understanding of network momentum strategies. The resulting networks unvocer community structures and temporal variations, aligning well with established financial concepts.

Transitioning to chapter 5, we introduce GNNHAR, a graph neural network tailored for modelling and forecasting the realised volatility of U.S. equities. Our ex-

amination brings to light the pivotal role of nonlinear spillover network effects in significantly boosting forecasting accuracy. Furthermore, we underscore the significance of the loss function choice in modelling; opting for QLIKE over the traditional MSE enhances forecasting precision, especially during turbulent market phases. This showcases the flexibility of incorporating task-specific criterion in graph machine learning models.

There are several paths for future research.

**Modelling Considerations** In this thesis, we introduce a couple of graph machine learning models designed to infer the intricate relationships between the returns and volatility of assets. These returns and volatility possess a dynamic structure that can change over time. It is worth exploring if alternative graph machine learning models, such as those presented in [177, 227], can more adeptly capture both the nonlinearity and temporal dynamics for graph-based financial inferences. Another path of exploration is tailoring models specifically for financial tasks, especially investment decisions. For instance, when predicting asset returns to construct a portfolio, one might desire a portfolio with buy/sell decisions that have low correlations to established portfolios, aiming for diversification. Incorporating a de-correlation regularisation [204] into the modelling objective could be beneficial. Similar considerations, like turnover regularisation [138], can help reduce turnover rates, enhancing real-world trading profitability.

**Financial Interpretability** A thorough theoretical analysis, drawing from both finance and economics, on network momentum derived from pricing data would be invaluable. Such a study can delve deeper into the dynamics and significance of the intricate interrelationships prevalent in financial markets. In Chapter 4, we presented an end-to-end framework tailored to understand the financial network. This framework excels in both inferring financial networks and simultaneously optimising the ensuing task of portfolio construction. We ensure that these financial networks are meticulously adjusted in alignment with portfolio performance. However, an in-depth investigation into the network's optimality, the reasoning for employing the Sharpe ratio in training the financial networks, and the occurrence of tenuous connections within equities is essential. Pertaining to volatility forecasting, as underscored in Chapter 5, the pronounced superiority of QLIKE over MSE is noteworthy and deserves additional scrutiny. In the realm of graph machine learning, research exists that offers interpretable explanations for predictions made by GNN-based models on

molecular data [226]. It would be intriguing to assess the applicability of this research to financial tasks, particularly in determining if it can align with foundational or economic principles.

**Data and Ethical Issues** In the realm of financial machine learning, two primary challenges are data access and ethical concerns. The financial sector thrives on detailed, up-to-date data, but often, this information is kept private for business reasons, making research difficult [60, 110]. Additionally, using personal financial details can lead to privacy issues or unfair practices, emphasising the need for ethical handling. To tackle the data challenge, we have proposed Kernel Graph Learning (KGL) in Chapter 2 for noisy and incomplete data. Besides, we have innovated with graph learning in Chapter 4, a method to replace the need of obtaining hard-to-source datasets with learning from readily available pricing data. Furthermore, combining traditional economic ideas with machine learning, as suggested by Israel et al. [110], can lead to simpler yet effective models when there are a small size of data. It is worth noting that certain financial tasks, such as asset pricing and return prediction, can be re-calibrated more frequently. This shift from low-frequency to high-frequency data, available down to milliseconds, paves the way for deploying highly parameterised machine learning models, a principle that equally applies to graph machine learning.

**From Prediction to Casual Inference** In applying machine learning to quantitative finance, a pivotal distinction exists between prediction and causal inference. While machine learning models excel at prediction, understanding causality is often more relevant in finance for decision-making and policy implications [9, 216]. Consider the concept of network momentum. We have demonstrated that the graph adjacency matrices that represent the historical co-movement pattern in momentum features can enhance predictions on asset returns in Chapter 4. This leads to a subsequent question: Can the upward trend of past 1-day returns for Crude Oil and past 6-month returns for Cotton cause an upward trend in the future 1-day return of the S&P 100 stock market index? Building on this, when we look at methods to depict these causal relationships, graph machine learning offers some promising avenues. In this field, directed acyclic graphs (DAGs) serve as valuable tools to illustrate causal relationships. Both optimisation-based models [233] and deep learning models [128] have been proposed to learn DAGs from data. Exploring these methods might offer insightful future directions, especially in determining their efficacy in causal inference

regarding momentum spillover phenomena and a broader range of financial applications.

# Bibliography


[1] Pierre Ablin, Thomas Moreau, Mathurin Massias, and Alexandre Gramfort. Learning step sizes for unfolded sparse coding. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Cascades in networks and aggregate volatility. Technical report, National Bureau of Economic Research, 2010.

[3] Usman Ali and David Hirshleifer. Shared analyst coverage: Unifying momentum spillover effects. *Journal of Financial Economics*, 136(3):649–675, June 2020. ISSN 0304405X. doi: 10.1016/j.jfineco.2019.10.007.

[4] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2020.

[5] Torben G Andersen, Tim Bollerslev, Francis X Diebold, and Heiko Ebens. The distribution of realized stock return volatility. *Journal of Financial Economics*, 61(1):43–76, 2001.

[6] Torben G Andersen, Tim Bollerslev, and Nour Meddahi. Realized volatility forecasting and market microstructure noise. *Journal of Econometrics*, 160(1): 220–234, 2011.

[7] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468, 2006.

[8] Clifford S. Asness, Tobias J. Moskowitz, and Lasse Heje Pedersen. Value and Momentum Everywhere. *The Journal of Finance*, 68(3):929–985, 2013.

[9] Vladimir A. Atanasov and Bernard S. Black. Shock-Based Causal Inference in Corporate Finance and Accounting Research, December 2016.

[10] Lieven Baele. Volatility spillover effects in european equity markets. *Journal of Financial and Quantitative Analysis*, 40(2):373–401, 2005.

[11] Zhidong Bai, Wing-Keung Wong, and Bingzhi Zhang. Multivariate linear and nonlinear causality tests. *Mathematics and Computers in simulation*, 81(1): 5–17, 2010.

[12] Brian Baingana and Georgios B. Giannakis. Tracking Switched Dynamic Network Topologies From Information Cascades. *IEEE Transactions on Signal Processing*, 65(4):985–997, February 2017. ISSN 1941-0476. doi: 10.1109/TSP.2016.2628354.

[13] Onureena Banerjee, Laurent El Ghaoui, and Alexandre D'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008. ISSN 15324435. doi: 10.1145/1390681.1390696.

[14] Albert-László Barabási. Scale-Free Networks: A Decade and Beyond. *Science*, 325(5939), July 2009. doi: 10.1126/science.1173299.

[15] Albert-László Barabási. *Network Science*. Cambridge University Press, 2015. ISBN 978-1-107-07626-6.

[16] Ole E Barndorff-Nielsen and Neil Shephard. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(2):253–280, 2002.

[17] Jamil Baz, Nicolas Granger, Campbell R. Harvey, Nicolas Le Roux, and Sandy Rattray. Dissecting Investment Strategies in the Cross Section and Time Series. *Available at SSRN: https://ssrn.com/abstract=2695101*, December 2015.

[18] Amir Beck. *First-Order Methods in Optimization*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2017. ISBN 1-61197-498-4.

[19] Eugene Belilovsky, Kyle Kastner, Gaël Varoquaux, and Matthew B Blaschko. Learning to discover sparse graphical models. In *International Conference on Machine Learning*, pages 440–448, 2017.

[20] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[21] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[22] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

[23] Peter Berger, Gabor Hannak, and Gerald Matz. Efficient graph learning from noisy and incomplete data. *IEEE Transactions on Signal and Information Processing over Networks*, 6:105–119, 2020.

[24] Tim Bollerslev, Andrew J Patton, and Rogier Quaedvlieg. Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192(1):1–18, 2016.

[25] Tim Bollerslev, Benjamin Hood, John Huss, and Lasse Heje Pedersen. Risk everywhere: Modeling and managing volatility. *Review of Financial Studies*, 31 (7):2729–2773, 2018.

[26] Tim Bollerslev, Andrew J Patton, and Rogier Quaedvlieg. Modeling and forecasting (un) reliable realized covariances for more reliable financial decisions. *Journal of Econometrics*, 207(1):71–91, 2018.

[27] Giovanni Bonanno, Guido Caldarelli, Fabrizio Lillo, and Rosario N. Mantegna. Topology of correlation-based minimal spanning trees in real and model markets. *Physical Review E*, 68(4):046130, October 2003. doi: 10.1103/PhysRevE.68.046130.

[28] Leslie Boni and Kent L. Womack. Analysts, Industries, and Price Momentum. *Journal of Financial and Quantitative Analysis*, 41(1):85–109, March 2006. ISSN 0022-1090, 1756-6916. doi: 10.1017/S002210900000243X.

[29] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers Inc, 2011.

[30] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[31] Daniel Buncic and Katja IM Gisler. Global equity market volatility spillovers: A broader role for the united states. *International Journal of Forecasting*, 32 (4):1317–1339, 2016.

[32] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $\beta$-VAE. In *Proceedings of the 31th International Conference on Neural Information Processing Systems*, 2017.

[33] Laurent AF Callot, Anders B Kock, and Marcelo C Medeiros. Modeling and forecasting large realized covariance matrices and portfolio choice. *Journal of Applied Econometrics*, 32(1):140–158, 2017.

[34] José Vinícius de M. Cardoso, Jiaxi Ying, Sandeep Kumar, and Daniel P. Palomar. Estimating Normalized Graph Laplacians in Financial Markets. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[35] José Vinícius de Miranda Cardoso and Daniel P. Palomar. Learning Undirected Graphs in Financial Markets. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 741–745, November 2020. doi: 10.1109/IEEECONF51394.2020.9443573.

[36] Nicola Cetorelli and Stavros Peristiani. Prestigious stock exchanges: A network analysis of international financial centers. *Journal of Banking & Finance*, 37(5): 1543–1551, May 2013. ISSN 0378-4266. doi: 10.1016/j.jbankfin.2012.06.011.

[37] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[38] Qinkai Chen and Christian-Yann Robert. Multivariate realized volatility forecasting with graph neural network. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 156–164, 2022.

[39] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

[40] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. 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*, pages 1655–1658, 2018.

[41] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Deep iterative and adaptive learning for graph neural networks. *arXiv preprint arXiv:1912.07832*, 2019.

[42] Sundeep Prabhakar Chepuri, Sijia Liu, Geert Leus, and Alfred O Hero. Learning sparse graphs under smoothness prior. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6508–6512, 2017.

[43] Alex Chinco, Adam D Clark-Joseph, and Mao Ye. Sparse signals in the cross-section of returns. *Journal of Finance*, 74(1):449–492, 2019.

[44] Taufiq Choudhry, Fotios I Papadimitriou, and Sarosh Shabi. Stock market volatility and business cycle: Evidence from linear and nonlinear causality tests. *Journal of Banking & Finance*, 66:89–101, 2016.

[45] Fabrizio Cipollini, Giampiero M Gallo, and Alessandro Palandri. Realized variance modeling: Decoupling forecasting from estimation. *Journal of Financial Econometrics*, 18(3):532–555, 2020.

[46] Adam Clements and Daniel PA Preve. A practical guide to harnessing the har volatility model. *Journal of Banking & Finance*, 133:106285, 2021.

[47] Kenneth W. Clements and Renée Fry. Commodity currencies and currency commodities. *Resources Policy*, 33(2):55–73, June 2008. ISSN 0301-4207. doi: 10.1016/j.resourpol.2007.10.004.

[48] Lauren Cohen and Andrea Frazzini. Economic Links and Predictable Returns. *The Journal of Finance*, 63(4):1977–2011, 2008. ISSN 1540-6261. doi: 10.1111/j.1540-6261.2008.01379.x.

[49] Lauren Cohen and Andrea Frazzini. Economic Links and Predictable Returns. *The Journal of Finance*, 63(4):1977–2011, 2008.

[50] Lauren Cohen and Dong Lou. Complicated firms. *Journal of Financial Economics*, 104(2):383–400, May 2012.

[51] Michael Connor and Piyush Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):599–608, July 2010. ISSN 1941-0506. doi: 10.1109/TVCG.2010.9.

[52] Fulvio Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196, 2009.

[53] Mario Coutino, Elvin Isufi, Takanori Maehara, and Geert Leus. State-space network topology identification from partial observations. *IEEE Transactions on Signal and Information Processing over Networks*, 6:211–225, 2020. doi: 10.1109/TSIPN.2020.2975393.

[54] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International Conference on Machine Learning*, pages 1106–1114. PMLR, 2018.

[55] Alexandre D'Aspremont, Onureena Banerjee, and Laurent El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008. ISSN 08954798. doi: 10.1137/060670985.

[56] Jose Vinicius de Miranda Cardoso, Jiaxi Ying, and Daniel Palomar. Graphical Models in Heavy-Tailed Markets. In *Advances in Neural Information Processing Systems*, volume 34, pages 19989–20001. Curran Associates, Inc., 2021.

[57] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.

[58] Stavros Degiannakis and George Filis. Forecasting oil price realized volatility using information channels from other asset classes. *Journal of International Money and Finance*, 76:28–49, 2017.

[59] Zengde Deng and Anthony Man-Cho So. A fast proximal point algorithm for generalized graph laplacian learning. *2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5425–5429, 2020.

[60] Vasant Dhar. Should You Trust Your Money to a Robot? *Big Data*, 3(2):55–58, June 2015. ISSN 2167-6461. doi: 10.1089/big.2015.28999.vda.

[61] Grace J. Di Cecco and Tarik C. Gouhier. Increased spatial and temporal autocorrelation of temperature under climate change. *Scientific Reports*, 8(1):14850, October 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-33217-0.

[62] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17 (83):1–5, 2016.

[63] Francis X Diebold and Roberto S Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995.

[64] Shanshan Ding, R Dennis Cook, et al. Matrix variate regressions and envelope models. *Journal of the Royal Statistical Society Series B*, 80(2):387–408, 2018.

[65] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586, 2011.

[66] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning Laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

[67] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representative perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[68] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, May 2019. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2018.2887284.

[69] Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal Processing Magazine*, 37(6):117–127, 2020. doi: 10.1109/MSP.2020.3014591.

[70] C. Dussert, G. Rasigni, M. Rasigni, J. Palmari, and A. Llebaria. Minimal spanning tree: A new approach for studying order and disorder. *Physical Review B*, 34(5):3528–3531, September 1986. doi: 10.1103/PhysRevB.34.3528.

[71] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2224–2232, Cambridge, MA, USA, December 2015. MIT Press.

[72] Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[73] Robert F Engle and Kenneth F Kroner. Multivariate simultaneous generalized arch. *Econometric Theory*, 11(1):122–150, 1995.

[74] Jianqing Fan, Lei Qi, and Dacheng Xiu. Quasi-maximum likelihood estimation of GARCH models with heavy-tailed likelihoods. *Journal of Business & Economic Statistics*, 32(2):178–191, 2014.

[75] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are K-hop message passing graph neural networks. In *Advances in Neural Information Processing Systems*, 2022.

[76] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5): 75–174, February 2010. ISSN 03701573. doi: 10.1016/j.physrep.2009.11.002.

[77] L Franceschi, M Niepert, M Pontil, and X He. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97. PMLR, 2019.

[78] Deena P Francis and Kumudha Raimond. Major advancements in kernel function approximation. *Artificial Intelligence Review*, 2020.

[79] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[80] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Transactions on Recommender Systems*, 1(1):3:1–3:51, March 2023. doi: 10.1145/3568022.

[81] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[82] William R Gebhardt, Soeren Hvidkjaer, and Bhaskaran Swaminathan. Stock and bond market interaction: Does momentum spill over? *Journal of Financial Economics*, 2005.

[83] William R Gebhardt, Soeren Hvidkjaer, and Bhaskaran Swaminathan. Stock and bond market interaction: Does momentum spill over?$. *Journal of Financial Economics*, 2005.

[84] Georgios B. Giannakis, Yanning Shen, and Georgios Vasileios Karanikolas. Topology identification and learning over graphs: Accounting for nonlinearities and dynamics. *Proceedings of the IEEE*, 106(5):787–807, 2018. doi: 10.1109/JPROC.2018.2804318.

[85] Georgios B. Giannakis, Yanning Shen, and Georgios Vasileios Karanikolas. Topology Identification and Learning over Graphs: Accounting for Nonlinearities and Dynamics. *Proceedings of the IEEE*, 106(5):787–807, May 2018. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2018.2804318.

[86] Paul Glasserman and H Peyton Young. How likely is contagion in financial networks? *Journal of Banking & Finance*, 50:383–399, 2015.

[87] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1019–1028, Washington DC USA, July 2010. ACM. ISBN 978-1-4503-0055-1. doi: 10.1145/1835804.1835933.

[88] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: A survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. ISSN 14322994. doi: 10.1007/s00186-007-0161-1.

[89] Kristjan Greenewald, Shuheng Zhou, and Alfred Hero III. Tensor graphical lasso (TeraLasso). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(5):901–931, 2019.

[90] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 399–406, 2010.

[91] Klaus Grobys, Joni Ruotsalainen, and Janne Äijö. Risk-managed industry momentum and momentum crashes. *Quantitative Finance*, 18(10):1715–1733, 2018. ISSN 1469-7688. doi: 10.1080/14697688.2017.1420211. URL https://doi.org/10.1080/14697688.2017.1420211.

[92] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *Review of Financial Studies*, 33(5):2223–2273, 2020.

[93] Vedran Hadziosmanovic, Yongbin Li, Xiao Liu, Stuart Kim, David Dynerman, and Loic Royer. Graph learning networks. In *ICML 2019 Work. Learn. Reason. with Graph-Structured Represent.*, 2019.

[94] Daniel Haesen, Patrick Houweling, and Jeroen Van Zundert. Momentum spillover from stocks to corporate bonds. *Journal of Banking & Finance*, 79: 28–41, June 2017. ISSN 03784266. doi: 10.1016/j.jbankfin.2017.03.003.

[95] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological Deep Learning: Going Beyond Graph Data. *Available at arXiv: 2206.00606*, May 2023.

[96] Peter Hall and Qiwei Yao. Inference in ARCH and GARCH models with heavy-tailed errors. *Econometrica*, 71(1):285–317, 2003.

[97] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 205–213, 2017.

[98] Peter Reinhard Hansen and Elena-Ivona Dumitrescu. How should parameter estimation be tailored to the objective? *Journal of Econometrics*, 230(2):535–558, 2022.

[99] Peter Reinhard Hansen, Asger Lunde, and James M Nason. Choosing the best volatility models: the model confidence set approach. *Oxford Bulletin of Economics and Statistics*, 65:839–861, 2003.

[100] Peter Reinhard Hansen, Asger Lunde, and James M Nason. The model confidence set. *Econometrica*, 79(2):453–497, 2011.

[101] David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2): 281–291, 1997.

[102] Xinran He, Ke Xu, David Kempe, and Yan Liu. Learning influence functions from incomplete observations. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2073–2081, December 2016.

[103] Bernard Herskovic, Bryan Kelly, Hanno Lustig, and Stijn Van Nieuwerburgh. Firm volatility in granular networks. *Journal of Political Economy*, 128(11): 4097–4162, 2020.

[104] Cho-Jui Hsieh, Inderjit S Dhillon, Pradeep K Ravikumar, and Mátyás A Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems*, pages 2330–2338, 2011.

[105] Cho-Jui Hsieh, Matyas A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[106] Wei-Qiang Huang, Xin-Tian Zhuang, Shuang Yao, and Stan Uryasev. A financial network perspective of financial institutions' systemic risk contributions. *Physica A: Statistical Mechanics and its Applications*, 456:183–196, August 2016. ISSN 0378-4371. doi: 10.1016/j.physa.2016.03.034.

[107] Brian Hurst, Yao Hua Ooi, and Lasse Heje Pedersen. A Century of Evidence on Trend-Following Investing. *The Journal of Portfolio Management*, 2017.

[108] Vassilis N Ioannidis, Yanning Shen, and Georgios B Giannakis. Semi-blind inference of topologies and signals over graphs. In *2018 IEEE Data Science Workshop (DSW)*, pages 165–169. IEEE, 2018.

[109] Vassilis N Ioannidis, Yanning Shen, and Georgios B Giannakis. Semi-blind inference of topologies and dynamical processes over dynamic graphs. *IEEE Transactions on Signal Processing*, 67(9):2263–2274, 2019.

[110] Ronen Israel, Bryan T. Kelly, and Tobias J. Moskowitz. Can Machines 'Learn' Finance? *Journal of Investment Management*, 18(2), January 2020. doi: 10.2139/ssrn.3624052.

[111] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11313–11320, 2019.

[112] Vassilis Kalofolias. How to Learn a Graph from Smooth Signals. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 920–929. PMLR, May 2016.

[113] Vassilis Kalofolias and Nathanaël Perraudin. Large scale graph learning from smooth signals. In *International Conference on Learning Representations*, 2018.

[114] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

[115] Vassilis Kalofolias, Andreas Loukas, Dorina Thanou, and Pascal Frossard. Learning time varying graphs. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2826–2830, 2017.

[116] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, August 2016. ISSN 1573-4951. doi: 10.1007/s10822-016-9938-8.

[117] Mervyn A King and Sushil Wadhwani. Transmission of volatility between stock markets. *Review of Financial Studies*, 3(1):5–33, 1990.

[118] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[119] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[120] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697, 2018.

[121] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[122] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal? Dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.

[123] George M. Korniotis and Alok Kumar. State-Level Business Cycles and Local Return Predictability. *The Journal of Finance*, 68(3):1037–1096, 2013. ISSN 1540-6261. doi: 10.1111/jofi.12017.

[124] Sandeep Kumar, Jiaxi Ying, Jose Vinicius de Miranda Cardoso, and Daniel Palomar. Structured graph learning via laplacian spectral constraints. In *Advances in Neural Information Processing Systems*, pages 11651–11663, 2019.

[125] Sandeep Kumar, Jiaxi Ying, José Vinícius De M. Cardoso, and Daniel P. Palomar. A unified framework for structured graph learning via spectral constraints. *The Journal of Machine Learning Research*, 21(1):785–844, 2020.

[126] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[127] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA, April 2010. Association for Computing Machinery. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772751.

[128] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-Based Neural DAG Learning. *Proceedings of the Eighth International Conference on Learning Representations ( (ICLR 2020)*, February 2020. doi: 10.48550/arXiv.1906.02226.

[129] Brenden Lake and Joshua Tenenbaum. Discovering structure by learning sparse graphs. *Proceedings of the annual meeting of the cognitive science society*, 32 (32), 2010.

[130] Steffen L Lauritzen. *Graphical Models*. Clarendon Press, 1996.

[131] Charles M. C. Lee, Paul Ma, and Charles C. Y. Wang. The Search for Peer Firms: When Do Crowds Provide Wisdom?, November 2016.

[132] Charles M.C. Lee, Stephen Teng Sun, Rongfei Wang, and Ran Zhang. Technological links and predictable returns. *Journal of Financial Economics*, 132(3): 76–96, June 2019. ISSN 0304405X. doi: 10.1016/j.jfineco.2018.11.008.

[133] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[134] Sophia Zhengzi Li and Yushan Tang. Automated volatility forecasting. *Available at SSRN 3776915*, 2021.

[135] Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, September 2023. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.126441.

[136] Penghong Liang, Xiangping Wang, Han Sun, Yanwen Fan, Yulian Wu, Xin Lin, and Jinfeng Chang. Forest type and height are important in shaping the altitudinal change of radial growth response to climate change. *Scientific Reports*, 9(1):1336, February 2019. ISSN 2045-2322. doi: 10.1038/s41598-018-37823-w.

[137] Ting Liang, Guanxiong Zeng, Qiwei Zhong, Jianfeng Chi, Jinghua Feng, Xiang Ao, and Jiayu Tang. Credit risk and limits forecasting in e-commerce consumer lending service via multi-view-aware mixture-of-experts nets. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 229–237, 2021.

[138] Bryan Lim, Stefan Zohren, and Stephen Roberts. Enhancing Time Series Momentum Strategies Using Deep Neural Networks. *The Journal of Financial Data Science*, 1(4), September 2020.

[139] Shiqing Ling and Michael McAleer. Asymptotic theory for a vector ARMA-GARCH model. *Econometric Theory*, 19(2):280–310, 2003.

[140] Lily Y Liu, Andrew J Patton, and Kevin Sheppard. Does anything beat 5-minute RV? A comparison of realized measures across multiple asset classes. *Journal of Econometrics*, 187(1):293–311, 2015.

[141] Yueliang Liu, Wenbin Guo, Kangyong You, Lei Zhao, Tao Peng, and Wenbo Wang. Graph learning for spatiotemporal signal with long short-term characterization. *arXiv preprint arXiv:1911.08018*, 2019.

[142] Yueliang Liu, Lishan Yang, Kangyong You, Wenbin Guo, and Wenbo Wang. Graph learning based on spatiotemporal smoothness for time-varying graph signal. *IEEE access : practical innovations, open solutions*, 7:62372–62386, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2916567.

[143] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2077–2085, 2018.

[144] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4424–4431, 2019.

[145] Zhaosong Lu. Adaptive first-order methods for general sparse inverse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2000–2016, 2009. ISSN 08954798. doi: 10.1137/080742531.

[146] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 0022-1082. doi: 10.2307/2975974.

[147] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[148] Gonzalo Mateos, Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro. Connecting the Dots: Identifying Network Structure via Graph Signal Processing. *IEEE Signal Processing Magazine*, 36(3):16–43, May 2019. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2018.2890143.

[149] Rahul Mazumder and Trevor Hastie. The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, 6(August):2125–2149, 2012. ISSN 19357524. doi: 10.1214/12-EJS740.

[150] AR Meenakshi and C Rajian. On a product of positive semidefinite matrices. *Linear algebra and its applications*, 295(1-3):3–6, 1999.

[151] Lior Menzly and Oguzhan Ozbas. Market Segmentation and Cross-predictability of Returns. *The Journal of Finance*, 65(4):1555–1580, 2010.

[152] Camelia Minoiu and Javier A. Reyes. A network analysis of global banking: 1978–2010. *Journal of Financial Stability*, 9(2):168–184, June 2013. ISSN 15723089. doi: 10.1016/j.jfs.2013.03.001.

[153] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[154] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[155] Thomas Moreau and Joan Bruna. Understanding the learned iterative soft thresholding algorithm with matrix factorization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[156] Tobias J. Moskowitz and Mark Grinblatt. Do Industries Explain Momentum? *The Journal of Finance*, 54(4):1249–1290, August 1999.

[157] Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, May 2012. ISSN 0304-405X. doi: 10.1016/j.jfineco.2011.11.003.

[158] Marius Muja and David G Lowe. FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATION. In *International Conference on Computer Vision Theory and Applications*, volume 1, pages 331–340, 2009.

[159] Anna Nagurney and Stavros Siokos. *Financial networks: Statics and dynamics*. Springer Science & Business Media, 2012.

[160] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[161] Szilárd Pafka and Imre Kondor. Estimated correlation matrices and portfolio optimization. *Physica A: Statistical Mechanics and its Applications*, 343:623–634, November 2004. ISSN 0378-4371. doi: 10.1016/j.physa.2004.05.079.

[162] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605, 2020.

[163] Ester Pantaleo, Michele Tumminello, Fabrizio Lillo, and Rosario N. Mantegna. When do improved covariance matrix estimators enhance portfolio optimization? An empirical comparative study of nine estimators. *Quantitative Finance*, 11(7):1067–1080, July 2011. ISSN 1469-7688. doi: 10.1080/14697688.2010.534813.

[164] Christopher A Parsons, Riccardo Sabbatucci, and Sheridan Titman. Geographic Lead-Lag Effects. *The Review of Financial Studies*, 33(10):4721–4770, October 2020.

[165] Dimosthenis Pasadakis, Matthias Bollhöfer, and Olaf Schenk. Sparse quadratic approximation for graph learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11256–11269, 2023. doi: 10.1109/TPAMI.2023.3263969.

[166] Razvan Pascalau and Ryan Poirier. Increasing the information content of realized volatility forecasts. *Journal of Financial Econometrics*, 2021.

[167] Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, Dominique Pastor, and Michael G Rabbat. Characterization and inference of graph diffusion processes from observations of stationary signals. *IEEE transactions on Signal and Information Processing over Networks*, 4(3):481–496, 2017.

[168] Andrew J Patton. Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1):246–256, 2011.

[169] Andrew J Patton and Kevin Sheppard. Evaluating volatility and correlation forecasts. In *Handbook of Financial Time Series*, pages 801–838. Springer, 2009.

[170] Xiaoxue Peng, Wenyuan Wu, Yaoyao Zheng, Jingyi Sun, Tangao Hu, and Pin Wang. Correlation analysis of land surface temperature and topographic elements in Hangzhou, China. *Scientific Reports*, 10(1):10451, June 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-67423-6.

[171] Teresa M Przytycka, Mona Singh, and Donna K Slonim. Toward the dynamic interactome: it's about time. *Briefings in bioinformatics*, 11(1):15–29, 2010.

[172] Xingyue Pu, Tianyue Cao, Xiaoyun Zhang, Xiaowen Dong, and Siheng Chen. Learning to Learn Graph Topologies. In *Advances in Neural Information Processing Systems*, volume 34, pages 4249–4262, 2021.

[173] Raksha Ramakrishna, Hoi-To Wai, and Anna Scaglione. A User Guide to Low-Pass Graph Signal Processing and Its Applications: Tools and Applications. *IEEE Signal Processing Magazine*, 37(6):74–85, November 2020. ISSN 1558-0792. doi: 10.1109/MSP.2020.3014590.

[174] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them All–Solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.

[175] Manuel Gomez Rodriguez, Jure Leskovec, David Balduzzi, and Bernhard Schölkopf. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(1):26–65, April 2014. ISSN 2050-1242, 2050-1250. doi: 10.1017/nws.2014.3.

[176] Benjamin Rolfs, Bala Rajaratnam, Dominique Guillot, Ian Wong, and Arian Maleki. Iterative thresholding algorithm for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2012.

[177] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *ICML 2020 Workshop on Graph Representation Learning*, June 2020.

[178] Mark Rubinstein. Markowitz's "Portfolio Selection": A Fifty-Year Retrospective. *The Journal of Finance*, 57(3):1041–1045, 2002. ISSN 0022-1082.

[179] Yunus Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, Citeseer, 2012.

[180] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[181] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. Deep attentive learning for stock movement prediction from social media text and company correlations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8415–8426, 2020.

[182] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[183] Stefano Schiavo, Javier Reyes, and Giorgio Fagiolo. International trade and financial integration: A weighted network analysis. *Quantitative Finance*, 10 (4):389–399, April 2010. ISSN 1469-7688. doi: 10.1080/14697680902882420.

[184] G William Schwert. Stock volatility and the crash of'87. *Review of Financial Studies*, 3(1):77–102, 1990.

[185] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):467–483, 2017.

[186] Vedran Sekara, Arkadiusz Stopczynski, and Sune Lehmann. Fundamental structures of dynamic social networks. *Proceedings of the national academy of sciences*, 113(36):9977–9982, 2016.

[187] Rasoul Shafipour, Santiago Segarra, Antonio G Marques, and Gonzalo Mateos. Identifying the topology of undirected networks from diffused non-stationary graph signals. *arXiv preprint arXiv:1801.03862*, 2018.

[188] Yanning Shen and Georgios B. Giannakis. Online identification of directional graph topologies capturing dynamic and nonlinear dependencies. In *2018 IEEE Data Science Workshop (DSW)*, pages 195–199, 2018. doi: 10.1109/DSW.2018.8439119.

[189] Kevin Sheppard. Financial econometrics notes. *University of Oxford*, pages 333–426, 2010.

[190] Harsh Shrivastava, Xinshi Chen, Binghong Chen, Guanghui Lan, Srinvas Aluru, Han Liu, and Le Song. GLAD: Learning Sparse Graph Recovery. In *International Conference on Learning Representations*. arXiv, December 2019.

[191] David Shuman, Sunil Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 3(30):83–98, 2013.

[192] Martin Slawski and Matthias Hein. Estimation of positive definite M-matrices and structure learning for attractive Gaussian Markov random fields. *Linear Algebra and its Applications*, 473:145–179, 2015.

[193] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines*, pages 144–158. Springer, 2003.

[194] Pablo Sprechmann, Roee Litman, Tal Ben Yakar, Alexander M Bronstein, and Guillermo Sapiro. Supervised sparse analysis and synthesis operators. In *Advances in Neural Information Processing Systems*, volume 26, pages 908–916, 2013.

[195] Oliver Stegle, Christoph Lippert, Joris M Mooij, Neil D Lawrence, and Karsten Borgwardt. Efficient inference in matrix-variate gaussian models with\iid observation noise. In *Advances in Neural Information Processing Systems*, pages 630–638, 2011.

[196] Martin Summer. Financial Contagion and Network Analysis. *Annual Review of Financial Economics*, 5(1):277–297, 2013. doi: 10.1146/annurev-financial-110112-120948.

[197] Ruili Sun, Tiefeng Ma, Shuangzhe Liu, and Milind Sathye. Improved Covariance Matrix Estimation for Portfolio Risk Measurement: A Review. *Journal of Risk and Financial Management*, 12(1):48, March 2019. ISSN 1911-8074. doi: 10.3390/jrfm12010048.

[198] Efthymia Symitsi, Lazaros Symeonidis, Apostolos Kourtis, and Raphael Markellos. Covariance forecasting in equity markets. *Journal of Banking & Finance*, 96:153–168, 2018.

[199] Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep Representation Learning for Social Network Analysis. *Frontiers in Big Data*, 2, 2019. ISSN 2624-909X.

[200] Wee Ling Tan, Stephen Roberts, and Stefan Zohren. Spatio-Temporal Momentum: Jointly Learning Time-Series and Cross-Sectional Strategies. *The Journal of Financial Data Science*, 5(3), February 2023.

[201] Dorina Thanou, Xiaowen Dong, Daniel Kressner, and Pascal Frossard. Learning heat diffusion graphs. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):484–499, 2017.

[202] Vincenzo Tola, Fabrizio Lillo, Mauro Gallegati, and Rosario N. Mantegna. Cluster analysis for portfolio optimization. *Journal of Economic Dynamics and Control*, 32(1):235–258, January 2008. ISSN 0165-1889. doi: 10.1016/j.jedc.2007.01.034.

[203] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[204] Gerhard Tutz and Jan Ulbricht. Penalized regression with correlation-based penalty. *Statistics and Computing*, 19(3):239–253, September 2009. ISSN 1573-1375. doi: 10.1007/s11222-008-9088-5.

[205] Rasmus Varneskov and Valeri Voev. The role of realized ex-post covariance measures and dynamic model choice on the quality of covariance forecasts. *Journal of Empirical Finance*, 20:83–95, 2013.

[206] Arun Venkitaraman, Saikat Chatterjee, and Peter Händel. Predicting graph signals using kernel regression where the input signal is agnostic to a graph. *IEEE Transactions on Signal and Information Processing over Networks*, 5(4): 698–710, 2019.

[207] Arun Venkitaraman, Pascal Frossard, and Saikat Chatterjee. Kernel regression for graph signal prediction in presence of sparse noise. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5426–5430. IEEE, 2019.

[208] Miklós Vincze, Ion Dan Borcia, and Uwe Harlander. Temperature fluctuations in a changing climate: An ensemble-based experimental approach. *Scientific Reports*, 7(1):254, March 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-00319-0.

[209] Xingchen Wan, Jie Yang, Slavi Marinov, Jan-Peter Calliess, Stefan Zohren, and Xiaowen Dong. Sentiment correlation in financial news networks and associated market movements. *Scientific Reports*, 11(1):3062, February 2021.

[210] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607. IEEE, 2019.

[211] Yudong Wang, Zhiyuan Pan, and Chongfeng Wu. Volatility spillover from the US to international stock markets: A heterogeneous volatility spillover GARCH model. *Journal of Forecasting*, 37(3):385–400, 2018.

[212] Yudong Wang, Yu Wei, Chongfeng Wu, and Libo Yin. Oil and the short-term predictability of stock return volatility. *Journal of Empirical Finance*, 47:90–104, 2018.

[213] Ines Wilms, Jeroen Rombouts, and Christophe Croux. Multivariate volatility forecasts for stock market indices. *International Journal of Forecasting*, 37(2): 484–499, 2021.

[214] Kieran Wood, Sven Giegerich, Stephen Roberts, and Stefan Zohren. Trading with the Momentum Transformer: An Intelligent and Interpretable Architecture, November 2022.

[215] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph Neural Networks in Recommender Systems: A Survey. *ACM Computing Surveys*, 55 (5):97:1–97:37, December 2022. ISSN 0360-0300. doi: 10.1145/3535101.

[216] Tao Wu, Xiangyun Gao, Sufang An, and Siyao Liu. Time-varying pattern causality inference in global stock markets. *International Review of Financial Analysis*, 77:101806, October 2021. ISSN 1057-5219. doi: 10.1016/j.irfa.2021.101806.

[217] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[218] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. *arXiv preprint arXiv:2005.11650*, 2020.

[219] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, 2017.

[220] Qi Xu and Yang Ye. Commodity network and predictable returns. *Journal of Futures Markets*, n/a(n/a). ISSN 1096-9934. doi: 10.1002/fut.22420.

[221] Koki Yamada, Yuichi Tanaka, and Antonio Ortega. Time-varying graph learning based on sparseness of temporal variation. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5411–5415, 2019.

[222] Rei Yamamoto, Naoya Kawadai, and Hiroki Miyahara. Momentum Information Propagation through Global Supply Chain Networks. *The Journal of Portfolio Management*, 47(8):197–211, July 2021. ISSN 0095-4918, 2168-8656. doi: 10.3905/jpm.2021.1.264.

[223] Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4054–4061, 2019.

[224] Jiaxi Ying, José Vinícius de Miranda Cardoso, and Daniel Palomar. Nonconvex Sparse Graph Learning under Laplacian Constrained Graphical Model. In *Advances in Neural Information Processing Systems*, volume 33, pages 7101–7113. Curran Associates, Inc., 2020.

[225] Jiaxi Ying, José Vinícius de M. Cardoso, and Daniel P. Palomar. A Fast Algorithm for Graph Learning under Attractive Gaussian Markov Random Fields. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pages 1520–1524. IEEE, 2021.

[226] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating Explanations for Graph Neural Networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[227] Jiaxuan You, Tianyu Du, and Jure Leskovec. ROLAND: Graph Learning Framework for Dynamic Graphs. *Proceedings of 28th ACM SIGKDD*, August 2022.

[228] Chao Zhang, Xingyue Stacy Pu, Mihai Cucuringu, and Xiaowen Dong. Graph-based methods for forecasting realized covariances. *Available at SSRN*, 2022.

[229] Chao Zhang, Yihuang Zhang, Mihai Cucuringu, and Zhongmin Qian. Volatility forecasting with machine learning and intraday commonality. *Journal of Financial Econometrics, forthcoming*, 2023.

[230] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep CNN denoiser prior for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3929–3938, 2017.

[231] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[232] Licheng Zhao, Yiwei Wang, Sandeep Kumar, and Daniel P. Palomar. Optimization Algorithms for Graph Laplacian Estimation via ADMM and MM. *IEEE Transactions on Signal Processing*, 67(16):4231–4244, August 2019. ISSN 1941-0476. doi: 10.1109/TSP.2019.2925602.

[233] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pages 9472–9483, 2018.

[234] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[235] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.