

ABSTRACT

WANG, JIANIAN. Two Tales of One Story: when Causal Graph Meets Graph Neural Network. (Under the direction of Dr. Rui Song).

Representing complex relational data, graph data could contain both the individual node feature and structural information. Both associated with the graph structure, causal structure discovery approaches and graph neural network methods have different perspectives. On the one hand, causal structure discovery approach primarily focuses on the interpretation problem, aiming to estimate a causal graph based on observational data and understand the underlying causal relationship. On the other hand, graph neural network is tailored for the prediction task, where the graph data is served as an input to provide node-specific feature and relational insights. Telling one story in two tales, both methodologies deep dive into the graph structure data and many works have been done to adapt to unique characteristics of the graph structure. However, there are still limitations in each line of the method. In this dissertation, we conduct comprehensive literature review and propose new methodologies to overcome the limitations in both causal structure learning methods and graph neural network models.

In Chapter 1, we focus on causal structure discovery methodologies, and develop a new framework to model the time-varying causal graph. Though many causal structure learning approaches are developed to uncover the hidden causal structure utilizing deep-learning approaches, they often have a hidden assumption that the causal relationship remains unchanged over time, which may not hold in real life. Without assuming a static causal structure, in this chapter, we propose a framework to model the time-varying causal graph by integrating the basis approximation method into the score-based causal discovery approach based on dynamic linear structural equation model. We provide an algorithm capable of estimating causal graphs using historical information and also making accurate predictions for the future leveraging information spanning all time-periods. Additionally, we introduce definitions of the time-varying causal effect, which is estimated based on the causal graph estimates. To show the effectiveness of the proposed method, we conduct extensive numerical experiments and demonstrate its superior performance compared to the benchmarks in the synthetic dataset.

In Chapter 2, we further extend the dynamic causal structure discovery approach to accommodate time-varying and time-lagged causal relations. While our previous approach, introduced in the preceding chapter, assumed that only variables at the same time stamp were causally related, many real-world scenarios involve time-lagged relationships, where past observations may causally influence others. In this chapter, we develop a dynamic causal structure discovery based the dynamic structural vector autoregression model to account for

the time-lagged relationships. We propose the dynamic causal effect to represent treatment variable's effect on the outcome variable, which is applicable to diverse scenarios where causal relations could be both time-varying and time-lagged. Furthermore, we propose an algorithm to provide an estimate of the underlying causal graph, and conduct simulations to illustrate the effectiveness of the proposed method. Additionally, we apply the proposed approach to analyze real COVID-19 data and provide causal estimates on how policy restriction's effect changes.

In Chapter 3, we shift the focus on the graph neural network methods and conduct a comprehensive literature review on the graph neural network methods specifically on the financial application. To model the graph data, many works have been done utilizing machine-learning methods. Among them, graph neural network (GNN) methods could achieve state-of-art performance on various tasks. However, the complex nature of financial systems may result in multiple data sources and complicated graph structures, which imposes challenges on feature processing, graph construction, and graph neural network modeling. Since financial systems process unique characteristics and receive great attention, in this chapter, we provide a comprehensive summary of the GNN methodology developed for financial tasks. We first systemically categorize the commonly-used financial graphs based on graph characteristics and present the GNN models according to their graph types. We provide a comprehensive list of financial applications that GNN methods are applied and summarize various aspects of information for each application. Additionally, we propose some challenges that could be future directions of research.

In Chapter 4, we develop a causality-guided graph neural network approach, which incorporates the causal structure into graph neural network modeling, disentangling the spurious variable and causal features. While GNNs excel at forecasting tasks by leveraging structural relationships and node features in graph data, recent research has revealed their susceptibility exploiting spurious features in the modeling step, potentially undermining model's generalization ability. To enhance models' interpretation and generalization abilities, we propose a causality-guided graph neural network approach, integrating causal structure learning into GNN models. Leveraging the graph data, we first estimate a causal graph, which is then feed into the GNN model to adaptively learn the attention weights for each variable to differentiate the spurious variables. We develop an algorithm which could produce both a causal graph estimate and prediction results, providing causal insights on the data, while the forecast outcome could serve as a quantitative metric for causal graph estimation. Additionally, we conduct simulation analysis to demonstrate the effectiveness of the proposed method. We also apply the proposed method to real-world datasets, where it shows substantial improvement on prediction accuracy compared to benchmark methods and also provides a causal graph for interpretation.

© Copyright 2024 by Jianian Wang

All Rights Reserved

Two Tales of One Story: when Causal Graph Meets Graph Neural Network

by
Jianian Wang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina
2024

APPROVED BY:

Dr. Wenbin Lu

Dr. Shu Yang

Dr. Marie Davidian

Dr. Rui Song
Chair of Advisory Committee

BIOGRAPHY

The author was born in the city of Jinhua, located in the Zhejiang province of China. In 2018, she earned a Bachelor of Science degree in Statistics from University of Hong Kong, graduating with First Class Honours. Driven by enthusiasm for quantitative data analysis, she embarked on a new chapter in her academic journey and was admitted to North Carolina State University to pursue a Ph.D. degree in Statistics. Guided by Dr. Rui Song, she conducted research on causal structure.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude and appreciation to all those who have helped me throughout this academic journey. Their support, guidance, and encouragement have been invaluable throughout the years of pursuing the doctoral degree.

First and foremost, I would like to extend my sincere gratitude to my supervisor, Dr. Rui Song, for her steadfast support and insightful guidance, throughout my Ph.D. journey. In the past few years, Dr.Song has consistently supported me with constructive feedback, professional expertise, and gentle encouragement. She generously dedicated her time and effort to my academic growth even after leaving the department. Working with Dr.Song has been an honor, and her mentorship will certainly continue to inspire me.

I would like to thank my dissertation committee members, Dr. Wenbin Lu, Dr. Marie Davidian, and Dr. Shu Yang, for dedicating their time and providing invaluable feedback. Their insightful guidance and constructive suggestions have played a crucial role in my academic development.

Furthermore, I wish to extend my appreciation to faculty members and staff in the Statistics department. It has been a privilege to belong to such a welcoming and supportive community. I am especially thankful for the kindness and assistance extended by Lanakila Alexander, Alison McCoy, and Dr. Charlie Smith. Their warmth and helpfulness have greatly enriched my doctoral studies.

Lastly, I am profoundly grateful to my family and friends for their unwavering love, understanding, and encouragement. Their sincere support and constant encouragement serve as the cornerstone for my pursuit of this degree. Without their support, the completion of this dissertation would not have been achievable.

TABLE OF CONTENTS

List of Tables	vii
List of Figures	viii
Chapter 1 Dynamic Causal Structure Discovery	1
1.1 Introduction	1
1.2 Statistical framework	3
1.2.1 Graph terminology	3
1.2.2 Assumptions	4
1.3 Related work	4
1.3.1 Methods on causal structure learning	4
1.3.2 Dynamic causal effects	13
1.4 Proposed methods	16
1.4.1 Dynamic linear structural equation model	16
1.4.2 Dynamic causal effects	17
1.4.3 Basis approximation for varying coefficient modeling	18
1.4.4 Constrained causal structural learning	20
1.5 Simulation Studies	21
1.5.1 Simulation settings	21
1.5.2 Basis selection	22
1.5.3 Benchmark methods and evaluation metrics	23
1.5.4 Results	25
1.6 Discussion	27
Chapter 2 Dynamic Causal Structure Discovery with Time-Lagged Dependency	30
2.1 Introduction	30
2.2 Related work	32
2.2.1 Causal discovery with time-lagged dependencies	32
2.2.2 Time-lagged causal effects	35
2.3 Proposed method	36
2.3.1 Dynamic structural vector autoregression model	36
2.3.2 Basis approximation and constrained causal structural learning	37
2.4 Dynamic causal effect estimation	39
2.4.1 Assumptions	39
2.4.2 Dynamic causal effect	39
2.5 Simulation analysis	41
2.5.1 Simulation setup	41
2.5.2 Evaluation metric and benchmark methods	41
2.5.3 Implementation details	41
2.5.4 Results	42
2.6 Real data analysis	46
2.7 Discussion	49

Chapter 3	Review on Graph Neural Network Methods on Financial Application	51
3.1	Introduction	51
3.2	Graph categorization	54
3.2.1	Graph-related definition	54
3.2.2	Graph categorization by construction methods	55
3.2.3	Graph categorization by graph types	56
3.3	Feature processing	60
3.3.1	Sequential numerical data	61
3.3.2	Textual information	63
3.4	Graph neural network models	64
3.4.1	Homogeneous graph	66
3.4.2	Directed graph	67
3.4.3	Bipartite graph	68
3.4.4	Multi-relation graph	69
3.4.5	Dynamic graph	71
3.5	Application	72
3.5.1	Stock movement prediction	72
3.5.2	Loan default risk prediction	75
3.5.3	Recommender system of e-commerce	78
3.5.4	Fraud detection	79
3.5.5	Event prediction	81
3.6	Challenges	82
3.6.1	Graph evaluation methods	82
3.6.2	Explainability	83
3.6.3	Task type	83
3.6.4	Data availability	83
3.6.5	Scalability	84
Chapter 4	Causality Guided Graph Neural Network	85
4.1	Introduction	85
4.2	Related work	87
4.3	Proposed methodology	88
4.3.1	Graph-based causal structure discovery	88
4.3.2	Causality guided graph neural network	89
4.4	Simulation study	91
4.4.1	Synthetic data generation	91
4.4.2	Benchmark methods	92
4.4.3	Results	93
4.5	Real data analysis	96
4.6	Discussion	99
References		101
APPENDICES		111
Appendix A	Appendix for Chapter 1	112

A.1	Proof of Theorem 1.4.1	112
A.2	Corollary of Theorem 1.4.1	114
Appendix B	Appendix for Chapter 2	115
B.1	Proof of Theorem 2.4.1	115
Appendix C	Appendix for Chapter 3	117
C.1	Acronyms	117

LIST OF TABLES

Table 1.1	The empirical comparison of the estimated causal graph for the synthetic data in the dynamic LSEM setup. The number in parenthesis denotes the standard deviation.	25
Table 1.2	The empirical comparison of the estimated causal graph for the synthetic data in the dynamic LSEM setup, when $m = 1, T = 100$. The number in parenthesis denotes the standard deviation.	27
Table 2.1	The empirical comparison of estimated contemporaneous weight matrix \mathbf{B}_t in the dynamic SVAR model setup. The number in parenthesis denotes the standard deviation.	43
Table 2.2	The empirical comparison of estimated time-lagged weight matrix \mathbf{W}_t in the dynamic SVAR model setup. The number in parenthesis denotes the standard deviation.	44
Table 3.1	Summary of financial-related graphs	59
Table 3.2	Summary of stock movement prediction literature	73
Table 3.3	Summary of loan default risk prediction literature	76
Table 3.4	Summary of literature on recommendation system of e-commerce	79
Table 3.5	Summary of fraud detection prediction literature	80
Table 3.6	Summary of event prediction literature	82
Table 4.1	Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and the spurious scale $\gamma_s = -1$	95
Table 4.2	Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and the spurious scale $\gamma_s = -0.3$	96

LIST OF FIGURES

Figure 1.1	Demonstration for causal structure learning problem. Based on a data matrix, the causal structure learning problem aims to learn a DAG that could represent the causal relationships between variables. Each node in the DAG represents a variable in the data matrix and each edge represents a causal relationship.	2
Figure 1.2	Causal strength estimates with different number of knots under scenario 1 with cosine function in the dynamic LSEM setup using the proposed method.	23
Figure 1.3	Estimated causal strength using the proposed method, DAGGNN, NOTEARS, DYNOTEARS and ANOCE for Scenario S1 under different time-varying coefficient functions. The solid line denotes the estimated coefficient in the ten time stamps and the dashed line denotes the one-step ahead prediction.	26
Figure 1.4	Estimated causal graphs at multiple time-stamps using the proposed method and the benchmarks for Scenario 1 based on the dynamic LSEM setup. The horizontal axis denotes the number of timestamps.	28
Figure 1.5	Estimated causal graphs at multiple time-stamps using the proposed method and benchmarks for dynamic LSEM setup, Scenario 2. The horizontal axis denotes the number of timestamps.	29
Figure 2.1	Estimated causal strength using the proposed method and benchmark methods in the dynamic SVAR model setup. The solid line denotes the estimated coefficient in the ten time stamps and the dashed line denotes the one-step ahead prediction.	44
Figure 2.2	Estimated causal graphs at multiple time-stamps using the proposed method and the benchmark based on the dynamic SVAR model setup.	45
Figure 2.3	Average reported new cases and mobility with respect to time. The shaded region represents the time period that the contact restriction policy started to implement.	46
Figure 2.4	Estimated causal graph at March and July. Each node represents a variable and the arrows represent the discovered causal relations. Red color represents positive causal relations and blue color represents negative causal relations.	48
Figure 2.5	Estimated dynamic causal effect of the contact restriction policy on reported new cases. The shaded region represents the time period when the policy starts to implement.	49

Figure 3.1	Workflow for stock movement prediction task using GNN methodology. The graph construction and feature processing steps present stock information in a graph and a feature matrix, which is then used as the input for the GNN model. In the graph, nodes are connected if there exist some relationships between stocks, such as supplier, competitor, shake-holder, etc. A multi-layer perception layer (MLP) is used to output the price prediction result.	52
Figure 3.2	Graph categorization based on graph characteristics. Each color of the circle represents a node type and each color of the line represents an edge type. Arrows represent directed edges. A homogeneous graph is a graph with one type of node and one type of edge. A directed graph is a graph with directed edges. A bipartite graph is a graph with two types of nodes and edges only exist between nodes of different types. A multi-relation graph has edges with different types. A dynamic graph is a sequence of graphs.	57
Figure 3.3	Feature processing for sequential features and textual information. For sequential numerical features, recurrent neural network (RNN) based approaches are commonly used to capture the temporal dependencies. For text features, it is often processed utilizing natural language processing (NLP) methods including word embedding, sentence embedding, and language models, to convert the unstructured data to structured ones.	61
Figure 3.4	Graph neural network models for different graph types. The term Conv denotes graph convolution process. The term MLP denotes the multi-layer perception. The term Agg denotes the aggregation process. The term Split denotes data splitting according to its characteristics.	66
Figure 4.1	Colored MNIST(Arjovsky et al. 2019) data set for graph classification task, where the coloring pattern is different in the training and testing data. . .	87
Figure 4.2	Different causal graphs used in the data generation process, when $\gamma_s = -1$.	92
Figure 4.3	Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and spurious scale different γ_s	94
Figure 4.4	Test accuracy for the proposed method when the graph input is randomly generated and causally estimated, with different number of variable d , and spurious scale different γ_s	95
Figure 4.5	Citation graph for pubmed data. Nodes represent the scientific publications and are linked if there are citation relationships. Darker nodes indicates the paper has more citations.	97
Figure 4.6	Snapshot of the Pubmed data, where each column is a binary matrix indicating the existence of a word. The last column is the binary label of whether the paper is about the type-2 diabetes.	97
Figure 4.7	Estimated causal graph based on the Pubmed data. The text represents the root words, processed to minimize variations in word forms.	98

Figure 4.8	A subgraph of the estimated causal graph focusing on the outcome variable. The red node denotes the outcome variable, and the green node denotes the potential spurious variable.	99
Figure 4.9	Prediction accuracy on testing data.	99

CHAPTER

1

DYNAMIC CAUSAL STRUCTURE DISCOVERY

1.1 Introduction

To represent the causal relationships between variables, a directed acyclic graph (DAG) (VanderWeele and Robins 2007) is widely utilized in many areas, such as social sciences, epidemics, and genetics (Pearl 2009). The right side of Figure 1.1 shows a simple DAG, where X_2 and X_3 are causes of X_4 , while X_1 also causally affect X_2 , and X_3 . Based on the causal graph, we could calculate various types of causal effects to illustrate the causal relations better. Learning the underlying causal graph from the observed data is the major goal for the causal structure discovery problem and there are many recent works that focus on this area.

Score-based approach is commonly used to recover the hidden causal graph. Based on the fact that each graph could be scored using some score functions, such as Bayesian information criterion (BIC) (Schwarz 1978), it focuses on finding the DAG which could yield the optimal score. For instance, Chickering (2002) formulates the structure learning problem as a combinatorial optimizing problem and optimize the score by searching over the graph set with a greedy search method. Owing to the constraint that the graph must be acyclic, searching over the graph space is a combinatorial problem which is NP-hard (Chickering et al. 2004), and the

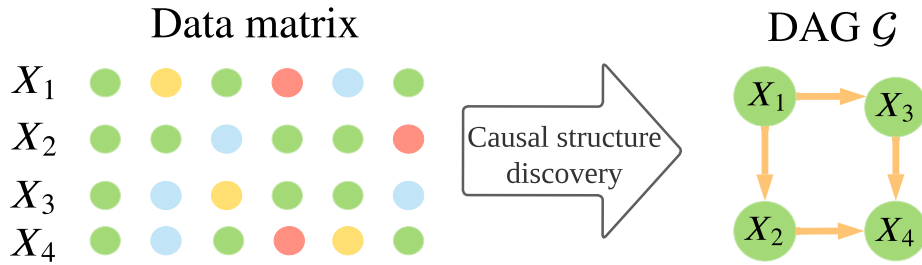


Figure 1.1: Demonstration for causal structure learning problem. Based on a data matrix, the causal structure learning problem aims to learn a DAG that could represent the causal relationships between variables. Each node in the DAG represents a variable in the data matrix and each edge represents a causal relationship.

intractable problem poses challenges on optimization due to the computational difficulties in large-scale data. To overcome this challenge caused by acyclicity constraint, Zheng et al. (2018) convert the combinatorial optimization problem into a continuous optimization problem by setting the acyclicity constraint as a function of the adjacency matrix, so that black-box solvers could be used to obtain the optimal graph efficiently. Motivated by this constraint setup, Yu et al. (2019) apply the variational auto-encoder model to allow non-linear relationships, and Cai et al. (2020) provide a decomposition of the indirect effect, to categorize the interactions between mediators better. Pamfil et al. (2020) further extend the algorithm proposed by Zheng et al. (2018) by considering both contemporaneous and time-lagged relationships.

Obtaining a causal graph using observations across time, the above literature often has an assumption that the causal relation is fixed across time. However, the causal relations may not be static and may change with time. For instance, putting on a mask against wildfire smoke may be effective at the beginning, but will become less effective after long time of exposure (Wu et al. 2020a). Failing to capture the dynamic pattern of the causal graph, the conventional approaches may have large bias assuming the same causal structure among all observations.

The rest of this chapter is organized as follows. We first introduce the graph terminology and assumptions in Section 1.2. We then provide a literature review on both causal structure learning methods and dynamic causal effects in Section 1.3. In Section 1.4, we briefly discuss the limitation of the current method and introduce our proposed method. In Section 1.5, we provide simulation studies to evaluate the performance of the proposed method and the existing approaches.

1.2 Statistical framework

In this section, we introduce the statistical framework of causal structure discovery, containing both graph terminology and widely-accepted assumptions.

1.2.1 Graph terminology

A graph \mathcal{G} is defined by a pair: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_p\}$ is a set of p nodes and \mathcal{E} is a set of edges, where $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge joining node v_i and node v_j . An adjacency matrix \mathbf{A} is an $p \times p$ matrix, where \mathbf{A}_{ij} represents the connection status between node v_i and node v_j . For an unweighted graph, the adjacency matrix could be a binary matrix where $\mathbf{A}_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and 0 otherwise. For instance, the graph \mathcal{G} in Figure 1.1 has $p = 4$ nodes, where the nodes are $\mathcal{V} = \{X_1, X_2, X_3, X_4\}$ and edges are $\mathcal{E} = \{(X_1, X_2), (X_1, X_3), (X_2, X_4), (X_3, X_4)\}$.

A directed graph is a graph where the edges have orientations and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge pointing from node v_i to (\rightarrow) node v_j . The skeleton of a directed graph is the graph that replaces all directed edges with undirected edges. Node v_i is a parent of v_j if there is a directed edge from v_i to v_j and v_j is the descendent of v_i . Directed edges could be used to represent relations that have directions while undirected edges could represent mutual relations. For instance, we could construct an undirected edge between nodes X_1, X_2 to represent their friendship. And a directed edge pointing from X_1 to X_2 could be used to represent a transaction made from X_1 to X_2 . For the graph \mathcal{G} in Figure 1.1, X_1 is the parent node for X_2 and X_2 is the descendent of X_1 .

A dynamic graph is defined as a sequence of graphs $\mathcal{G}^{seq} = \{\mathcal{G}_1, \dots, \mathcal{G}_T\}$, where $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, for $i = 1, \dots, T$, where $\mathcal{V}_i, \mathcal{E}_i$ are the set of nodes and edges for i^{th} graph in the sequence respectively.

A path from v_i to v_j in \mathcal{G} is a sequence of distinct vertices, $\{a_0, \dots, a_L\} \subset \mathcal{V}$, such that $a_0 = v_i$, and $a_L = v_j$. A directed path from v_i to v_j is a path between v_i and v_j where all edges are directed toward v_j . A directed cycle is defined if there exists a directed path from v_i to v_j and a directed path from v_j to v_i for any $v_i, v_j \in \mathcal{V}, i \neq j$. A directed acyclic graph is defined as a directed graph that does not contain directed cycles. The graph \mathcal{G} in Figure 1.1 is a directed acyclic graph.

A triple (v_i, v_j, v_k) in a graph \mathcal{G} is unshielded if $e_{ij}, e_{jk} \in \mathcal{E}$, but v_i, v_k is not adjacent. A v-structure (v_i, v_j, v_k) is an unshielded triple in a \mathcal{G} where the edges are oriented as $(v_i \rightarrow v_j \leftarrow v_k)$.

1.2.2 Assumptions

In this section, we list some commonly-used assumptions in the causal discovery literature (Spirtes et al. 2000a).

Causal sufficiency A set \mathcal{V} of variables is causally sufficient for a population if and only if in the population every common cause of any two or more variables in \mathcal{V} is in \mathcal{V} , or has the same value for all units in the population.

Causal Markov condition Each vertex is independent of its non-descendants in the graph conditional on its parents in the graph. In other words, we have

$$P(v_1, v_2, \dots, v_n) = \prod_{i=1}^n P(v_i | pa(i)), \quad (1.1)$$

for vertices $\{v_1, \dots, v_p\} \in \mathcal{V}$, where $pa(i)$ are the parent nodes for v_i and $P(\cdot)$ is a probability function .

For instance, in Figure 1.1, X_4 is independent of X_1 given X_2, X_3 , under the causal Markov condition.

Causal faithfulness Any conditional independence relations in the population are entailed by the Causal Markov Assumption.

With the causal faithfulness assumption, we could conduct conditional independence tests to obtain information of the underlying graph structure. Take Figure 1.1 as an example, if a statistical test indicates that X_4 is independent of X_1 given X_2, X_3 , we could conclude that there is no direct edge between X_1 and X_4 .

1.3 Related work

In this section, we review the literature on causal structure learning and dynamic causal effect. The rest of this section is organized as follows. In Section 1.3.1, we categorize causal structure learning methods into the constraint-based approach, structural equation model (SEM) based approach, and score-based approach. In Section 1.3.2, we summarize the literature focusing on the dynamic causal effects.

1.3.1 Methods on causal structure learning

Assuming that the observed i.i.d. data is generated from a hidden causal graph, three major types of methods are developed for causal structural learning to recover the hidden causal structure: constraint-based approach, structural equation model (SEM) based approach, and

score-based approach (Heinze-Deml et al. 2018).

Constraint-based approach

The first line of work, the constraint-based approach utilizes series of conditional independence tests to learn the underlying causal structure.

A well-known example of the constraint-based approach is the PC algorithm (Spirtes et al. 2000a). Starting with a complete undirected graph, the PC algorithm conducts three steps to find the underlying DAG: obtain the skeleton, determine the v-structure and determine the rest edge orientations. In the first step, by conducting conditional independence tests on the observed data, an edge between v_i, v_j is deleted if v_i is independent of v_j given some subset S in the remaining nodes and S is saved in the separation set $sepset(v_i, v_j)$. Specifically, at l^{th} iteration, it tests the conditional independence between v_i, v_j given the following S ,

$$|S| = l, S \subseteq \{adj(v_i) \setminus v_j\} \cup \{adj(v_j) \setminus v_i\},$$

where $|S|$ denotes the size of the set S and $adj(v_i)$ denotes the nodes that are adjacent to node v_i , and $\setminus v_j$ denotes subtracting node v_j from the set. By doing so, this search step will stop if the size of S reaches the degree of the underlying DAG, which ensures the algorithm's efficiency when the underlying graph is sparse. In the second step, the algorithm loops through all unshielded triples to determine the v-structure, in which an unshielded triple (v_i, v_j, v_k) is oriented to be a v-structure $(v_i \rightarrow v_j \leftarrow v_k)$ if and only if $v_j \notin sepset(v_i, v_j)$. In the third step, the orientations of the rest of the edges are determined based on some orientation rules. Assuming the causal sufficiency, the causal faithfulness and Gaussian distribution for the data, the PC algorithm could obtain a completed partially directed acyclic graph (CPDAG) that represents the underlying DAG.

As shown by Kalisch and Bühlman (2007), the PC algorithm is asymptotically consistent as the sample size goes to infinity and performs computationally well under high-dimensional sparse graphs. However, the PC algorithm is order-dependent and its output may vary if the order of the variables changes. The order dependence of the PC algorithm may result in non-stable results and Colombo et al. (2014) modifies the PC algorithm and makes it order-independent in both the skeleton and v-structure determination steps. Another order-independent modification of the PC algorithm is the fast causal inference algorithm (FCI), which drops the causal sufficiency assumption. Allowing the hidden latent variables, the FCI learns a Markov equivalence class of the true DAG (Zhang 2008), where the Markov equivalence class refers to DAGs describing the same conditional independence information. The FCI has the same first step as the PC algorithms, while more conditional independence tests are conducted due to the latent

variables and additional orientation rules are posed in the third step. With extra conditional independence tests, FCI is not computationally feasible for large graphs and rapid fast causal inference algorithm (RFCI) is proposed with lower computation complexity (Colombo et al. 2012). Avoiding some time-costly conditional independence tests and shrinking the size of the separation set, RFCI is much faster than FCI for sparse graphs.

Structural equation model based approach

The second type of approach is the structural equation model based approach, which imposes assumptions on data distribution while utilizing the structural equation models. Unlike the constraint-based approach which often assumes Gaussian noise distribution, this line of method drops the Normal assumption and shows that the graph is identifiable under some conditions. For instance, ICA-LiNGAM (Linear Non-Gaussian Acyclic Model) (Shimizu et al. 2006) utilizes the linear structural equation model to estimate the adjacency matrix \mathbf{A} from the observed data, while assuming non-Gaussian noise to guarantee the model identification.

The linear structural equation model specifies the relationship between variables through linear equations (Drton et al. 2011):

$$x_k = \sum_{j \in pa(k)} a_{kj} x_j + e_k, \quad k = 1, \dots, p,$$

where x_k is the observed value for k^{th} variable, p is the number of variables, a_{kj} is the coefficient between k^{th} and j^{th} variable, $pa(k)$ represent the parent nodes of k^{th} variable and e_k is the respective error term.

Without loss of generality, we have

$$\mathbf{x}_i = \mathbf{A}^T \mathbf{x}_i + \mathbf{e}_i,$$

where $\mathbf{x}_i \in \mathbb{R}^p$ is a data vector for i^{th} observation, $\mathbf{A} \in \mathbb{R}^{p \times p}$ is a adjacency matrix and $\mathbf{e}_i \in \mathbb{R}^p$ is a noise for i^{th} observation. Since the adjacency matrix could be permuted as an upper-triangular matrix based on causal orderings, the adjacency matrix is invertible which guarantees the existence of $(\mathbf{I} - \mathbf{A})^{-1}$, where \mathbf{I} denotes the identity matrix. Thus we could write:

$$\mathbf{x}_i = (\mathbf{I} - \mathbf{A}^T)^{-1} \mathbf{e}_i,$$

which could be seen as an independent component analysis (ICA) model, assuming the error is independent and non-Gaussian. As shown by Comon (1992), the adjacency matrix \mathbf{A} is identifiable since the observed data is a linear combination of non-Gaussian independent

components and the adjacency matrix is invertible. With the identification guarantee, ICA-LiNGAM utilizes the independent component analysis (ICA) algorithm to obtain the linear model decomposition and permute the order to get the estimate of the adjacency matrix.

Since the ICA method is an iterative search method, it may obtain a local optimal result instead of finding the global optima, and Direct-LiNGAM (Shimizu et al. 2011) is proposed to guarantee the convergence to the correct solution within a fixed number of steps. Using pairwise regressions, Direct-LiNGAM finds the variables that are mostly independent of the residuals and removes their effect based on least square regression. Then LiNGAM is utilized to find the causal orderings of the residuals which are proven as the same as the ordering of variables. With guaranteed convergence, Direct-LiNGAM has shown better performance than ICA-LiNGAM in numerical experiments.

Score-based approach

The third major approach, the score-based approach is based on the fact that every graph could be scored using some score functions, such as BIC (Bayesian information criterion) (Schwarz 1978) and the likelihood-based Bayesian scoring metric (BDe)(Heckerman et al. 1995), and it focuses on finding the DAG which could yield the optimal score.

Searching in a large space of graphs to find the graph with the optimal score, Chickering (2002) propose and theoretically justify a two-phase greedy search algorithm: greedy equivalence search (GES). In the first phase, edges are added until reach the local maximum and in the second phase, edges deletions are conducted. By comparing the scores after each addition and deletion phase, GES is able to obtain the graph with the optimal score efficiently and consistently. However, GES requires a long time when variable sizes are large, and to speed up the computing process, fastGES (Ramsey et al. 2017) is proposed. By decomposing the graph's score to the sum of graph fragments' scores, fastGES stores and updates the scores of fragments without calculating the graph's score at each updating step. With this decomposing technique and paralleling the scoring step, fastGES is able to find the underlying graph rapidly in a high-dimensional setup.

The previously mentioned scored-based approach searches for the graph that optimizes the score over the graph set. Owing to the constraint that the graph must be acyclic, searching over the graph space is a combinatorial problem which is NP-hard even with causal Markov condition and causal faithfulness assumptions (Chickering et al. 2004). The intractable problem poses challenges to optimization due to the computational difficulties in large-scale data. One approach to handle the scalability issue is to use approximate search algorithms. For example, Nie et al. (2014) develops a sample-based approach to provide approximate solutions and Scanagatta et al. (2015) utilizes an approximate score function to guide the exploration over

the search space to reduce time cost.

Recently, there's another approach that formulate the acyclic constraint as a smooth equality constraint by proving that a matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$ is a DAG if and only if

$$h(\mathbf{A}) = \text{trace}(e^{\mathbf{A} \circ \mathbf{A}}) - p = 0, \quad (1.2)$$

where \circ is the element-wise multiplication and $e^{\mathbf{A}}$ is the matrix exponential of \mathbf{A} (Zheng et al. 2018). By doing so, the acyclic constraint could be replaced with an equality constraint $h(\mathbf{A})$, which is smooth and has an easy-to-calculate gradient:

$$\nabla h(\mathbf{A}) = (e^{\mathbf{A} \circ \mathbf{A}})^T \circ 2\mathbf{A}.$$

With this algebraic representation of the acyclic constraint, the constrained optimization problem (left) could then be converted to a continuous programming problem (right):

$$\begin{array}{ccc} \min_{\mathbf{A} \in \mathbb{R}^{p \times p}} F(\mathbf{A}) & \implies & \min_{\mathbf{A} \in \mathbb{R}^{p \times p}} F(\mathbf{A}) \\ \text{subject to } \mathcal{G}(\mathbf{A}) \in \text{DAG} & & \text{subject to } h(\mathbf{A}) = 0, \end{array}$$

where \mathbf{A} is a $p \times p$ adjacency matrix, $F(\cdot)$ is the score function, $\mathcal{G}(\mathbf{A})$ is a graph induced by the adjacency matrix \mathbf{A} , and $h(\cdot)$ is the equality constraint function defined in Equation (1.2).

Based on this continuous programming problem setup, Zheng et al. (2018) propose the NOTEARS algorithm (Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning) with the Linear Structural Equation Model (LSEM):

$$\mathbf{X} = \mathbf{X}\mathbf{A} + \mathbf{E},$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a p dimensional data matrix of n i.i.d. observations, \mathbf{A} denotes the weighted adjacency matrix, and \mathbf{E} denotes the noise matrix.

For the score function, the NOTEARS algorithm includes both the least square loss $\ell(\mathbf{A}; \mathbf{X}) = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{A}\|_F^2$ and the ℓ_1 penalty term, since minimizing the least square loss could yield a consistent estimator of the DAG (Aragam et al. 2015) and the penalty term could help the algorithm to learn a sparse DAG. The score function $F(\cdot)$ is defined as follows:

$$\begin{aligned} F(\mathbf{A}) &= \ell(\mathbf{A}; \mathbf{X}) + \lambda \|\mathbf{A}\|_1 \\ &= \frac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_1, \end{aligned}$$

where λ is the penalty parameter, and $\|\cdot\|_F$ is the Frobenius norm, .

With the augmented Lagrangian method, the constraint optimization problem could then be converted into the unconstrained problem:

$$\min_{\mathbf{A} \in \mathbb{R}^{p \times p}} F(\mathbf{A}) + \frac{\rho}{2} |h(\mathbf{A})|^2 + \alpha h(\mathbf{A}),$$

where $\rho > 0$ is a penalty parameter and α is the Lagrangian multiplier. With the reformulation of the problem, the optimization problem could be solved using black-box solvers and NOTEARS could return local optimums which are highly close to the global minimizer.

Since NOTEARS has assumed a linear structural equation model (LSEM) structure, it may fail to capture more complex distribution of the real data. With the same spirit of forming the acyclic constraint as a function of adjacency matrix, DAG-GNN (Yu et al. 2019) applies a deep-generative model, that is, variational auto-encoder (VAE) which generalizes the LSEM structure to better represent the data distribution.

Based on the LSEM structure, we have

$$\mathbf{x}_i = \mathbf{A}^T \mathbf{x}_i + \mathbf{e}_i,$$

where $\mathbf{x}_i \in \mathbb{R}^p$ is a data vector for i^{th} observation, $\mathbf{A} \in \mathbb{R}^{p \times p}$ is an adjacency matrix and $\mathbf{e}_i \in \mathbb{R}^p$ is a noise for i^{th} observation.

Then, we could consider the \mathbf{x}_i as a function of the generated error \mathbf{e}_i as follows:

$$\mathbf{x}_i = (\mathbf{I} - \mathbf{A}^T)^{-1} \mathbf{e}_i,$$

where \mathbf{I} denotes the identity matrix. Since the adjacency matrix \mathbf{A} is upper-triangular when we sort the graph nodes in the topological order, the existence of the inverse for matrix $(\mathbf{I} - \mathbf{A}^T)$ is guaranteed.

Motivated by the graph neural network structure, the decoder of the VAE is written as:

$$\mathbf{x}_i = f_2[(\mathbf{I} - \mathbf{A}^T)^{-1} f_1(\mathbf{e}_i)],$$

where f_1 and f_2 are user-defined functions. When f_2 is invertible, we could have the following representation:

$$f_2^{-1}(\mathbf{x}_i) = \mathbf{A}^T f_2^{-1}(\mathbf{x}_i) + f_1(\mathbf{e}_i),$$

which could be seen as a generalization of the LSEM structure.

Respectively, the encoder is written as

$$\mathbf{e}_i = f_4[(\mathbf{I} - \mathbf{A}^T)f_3(\mathbf{x}_i)],$$

where f_3 and f_4 are user-defined functions. In the DAG-GNN, the f_1, f_4 are set to be the identity mapping, and f_2, f_3 are set to be the multi-layer perceptron (MLP), so that the likelihood $p(\mathbf{x}_i|\mathbf{e}_i)$ and the variational posterior $q(\mathbf{e}_i|\mathbf{x}_i)$ are factored Gaussians. In this way, the nonlinearity of the data distribution could be captured by the MLP layer.

For the score function of DAG-GNN, since the marginal log-likelihood is intractable due to the true posterior density $p(\mathbf{e}_i|\mathbf{x}_i)$, the score function is related to the evidence lower bound (ELBO) (Kingma and Welling 2013), which could approximate the marginal log-likelihood:

$$L_{ELBO} = \frac{1}{n} \sum_{i=1}^n \{-D_{KL}\{q(\mathbf{e}_i|\mathbf{x}_i)||p(\mathbf{e}_i)\} + \mathbb{E}_{q(\mathbf{e}_i|\mathbf{x}_i)}\{\log p(\mathbf{x}_i|\mathbf{e}_i)\}\},$$

where D_{KL} represents the KL divergence and the evidence lower bound is differed from the marginal log-likelihood by the KL divergence of the variational posterior from the true posterior $D_{KL}\{q(\mathbf{e}_i|\mathbf{x}_i)||p(\mathbf{e}_i|\mathbf{x}_i)\}$.

As for the acyclic constraint, a new form is proposed by Yu et al. (2019), since the matrix exponential may not be practically appealing. It has been proven that a matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$ is a DAG, if and only if

$$h_1(\mathbf{A}) = \text{trace}[(\mathbf{I} + \alpha \mathbf{A} \circ \mathbf{A})^p] - p = 0, \quad \text{for any } \alpha > 0.$$

Based on this new form of acyclic constraint, the DAG-GNN approach solves the following optimization problem:

$$\begin{aligned} & \min_{\mathbf{A} \in \mathbb{R}^{p \times p}} -L_{ELBO} \\ & \text{subject to } \text{trace}[(\mathbf{I} + \alpha \mathbf{A} \circ \mathbf{A})^p] - p = 0, \end{aligned}$$

and obtain competitive performance for both linear and non-linear generated cases.

However, DAG-GNN ignores the causal relationship among variables, when treatment and outcome variables are specified. To incorporate the topological order for treatment and outcome variable, ANOCE (ANalysis Of Causal Effects) is proposed by Cai et al. (2020) which extends DAG-GNN by incorporating an identification constraint and enables the decomposition

of the indirect effect. With a linear structural equation model, we could have

$$\mathbf{x}_i = [a_i, \mathbf{m}_i^T, y_i]^T = \mathbf{A}^T \mathbf{x}_i + \mathbf{e}_i,$$

where a_i is the treatment variable for i^{th} observation, $\mathbf{m}_i \in \mathbb{R}^p$ is the p -dimensional mediator variable, that is, the intermediate variables between treatment variable and outcome variable, for i^{th} observation, and y_i is the outcome variable for i^{th} observation. Taking the topological order between the variables into consideration, we could further write:

$$\mathbf{x}_i = \begin{bmatrix} a_i \\ \mathbf{m}_i \\ y_i \end{bmatrix} = \mathbf{A}^T \begin{bmatrix} a_i \\ \mathbf{m}_i \\ y_i \end{bmatrix} + \mathbf{e}_i = \begin{bmatrix} 0 & \mathbf{0}_{p \times 1}^T & 0 \\ \boldsymbol{\alpha} & \mathbf{B}_m^T & 0 \\ \gamma & \boldsymbol{\beta}^T & 0 \end{bmatrix} \begin{bmatrix} a_i \\ \mathbf{m}_i \\ y_i \end{bmatrix} + \mathbf{e}_i,$$

where γ is a scalar, $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\mathbf{0}_{p \times 1}$ are $p \times 1$ vectors, and \mathbf{B}_m is a $p \times p$ matrix, since the treatment variable a_i has no parent node and the outcome variable y_i has no decedent.

To reflect the structure of the adjacency matrix, an additional identity constraint is added:

$$h_2(\mathbf{A}) = \sum_{i=1}^{p+2} |\mathbf{A}_{i,1}| + \sum_{j=2}^{p+2} |\mathbf{A}_{p+2,j}| = 0,$$

where $\mathbf{A}_{i,j}$ represents the $(i, j)^{th}$ element for the adjacency matrix A .

The optimization problem could then be written as

$$\begin{aligned} \min_{\mathbf{A} \in \mathbb{R}^{p \times p}} -L_{ELBO} &= -\frac{1}{p} \sum_{i=1}^p [-D_{KL}\{q(\mathbf{e}_i|\mathbf{x}_i)||p(\mathbf{e}_i)\} \\ &\quad + \mathbb{E}_{q(\mathbf{e}_i|\mathbf{x}_i)}\{\log p(\mathbf{x}_i|\mathbf{e}_i)\}] \\ \text{subject to } &trace[(\mathbf{I} + \alpha \mathbf{A} \circ \mathbf{A})^{p+2}] - (p+2) = 0, \\ \text{and } &\sum_{i=1}^{p+2} |\mathbf{A}_{i,1}| + \sum_{j=2}^{p+2} |\mathbf{A}_{p+2,j}| = 0, \end{aligned}$$

where D_{KL} represents the KL divergence, $q(\mathbf{e}_i|\mathbf{x}_i)$ denotes the variational posterior of \mathbf{e}_i , $p(\mathbf{e}_i)$ denotes the prior distribution of e_i , $\log p(\mathbf{x}_i|\mathbf{e}_i)$ denotes the log-likelihood function, \mathbf{e}_i denotes the i^{th} latent variable, and \mathbf{x}_i denotes the i^{th} data vector.

Based on the LSEM model, the previous approaches assume that there are only contemporaneous causal relationships and omit the potential time-lagged causal dependencies. To take both contemporaneous and time-lagged causal relationships into consideration, Pamfil et al. (2020) extend the NOTEARS approach and propose the DYNOTEARS algorithm. The data is modeled using the SVAR (Structural Vector AutoRegressions) model (Demiralp and Hoover

2003) as follows:

$$\begin{aligned} \mathbf{x}_{m,t}^T &= \mathbf{x}_{m,t}^T \mathbf{W} + \mathbf{x}_{m,t-1}^T \mathbf{A}_1 + \dots + \mathbf{x}_{m,t-d}^T \mathbf{A}_d + \mathbf{z}_{m,t}^T \\ &\text{for } t \in d, \dots, T, m \in 1, \dots, M, \end{aligned} \quad (1.3)$$

where $\mathbf{x}_{m,t} \in \mathbb{R}^p$ denotes the m^{th} data vector at t^{th} timestamp, p represents the number of variables in the dataset, T represents the number of timestamps, M represents the number of samples at each time stamp, matrices \mathbf{W} and $\mathbf{A}_i, i \in 1, \dots, d$ denote the weighted adjacency matrices, d is the autoregressive order, and $\mathbf{z}_{m,t}$ denotes the error vector which is independent within and across time.

By assuming that this network structure is constant across time, Equation (1.3) could be written in the following matrix form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}_1\mathbf{A}_1 + \dots + \mathbf{Y}_d\mathbf{A}_d + \mathbf{Z}, \quad (1.4)$$

where \mathbf{X} is an $n \times p$ matrix whose rows are $\mathbf{x}_{m,t}$, $n = M(T + 1 - d)$ denotes the effective sample size, and matrices $\mathbf{Y}_1, \dots, \mathbf{Y}_d$ are the respective lagged data matrix.

Let $\mathbf{Y} = [\mathbf{Y}_1 | \dots | \mathbf{Y}_d] \in \mathbb{R}^{n \times pd}$ be concatenated matrix for the lagged data, and let $\mathbf{A} = [\mathbf{A}_1^T | \dots | \mathbf{A}_d^T]^T \in \mathbb{R}^{pd \times p}$ denote the weights for the across-time dependencies. The Equation (1.4) could be written in the following compact form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}\mathbf{A} + \mathbf{Z}.$$

Since the future data won't affect the past observations, the edges in \mathbf{A} only go forward in time, which guarantees the acyclicity of the matrix \mathbf{A} . Thus, it suffices to require that \mathbf{W} is acyclic to ensure the whole causal network is acyclic.

In order to estimate the weighted adjacency matrices \mathbf{W} and \mathbf{A} , given the data matrices \mathbf{X} and \mathbf{Y} , the DYNOTEARS solves the following regularized optimization problem:

$$\begin{aligned} &\min_{\mathbf{W}, \mathbf{A}} f(\mathbf{W}, \mathbf{A}) \text{ s.t. } \mathbf{W} \text{ is acyclic} \\ &\text{with } f(\mathbf{W}, \mathbf{A}) = \ell(\mathbf{W}, \mathbf{A}) + \lambda_{\mathbf{W}} \|\mathbf{W}\|_1 + \lambda_{\mathbf{A}} \|\mathbf{A}\|_1, \\ &\ell(\mathbf{W}, \mathbf{A}) = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{W} - \mathbf{Y}\mathbf{A}\|_F^2, \end{aligned}$$

where $\lambda_{\mathbf{W}}, \lambda_{\mathbf{A}}$ denotes the regularization weights for the weight matrices, $\|\cdot\|_1$ denotes the element-wise ℓ_1 norm.

As shown by (Zheng et al. 2018), the acyclicity constraint could be formulated to an equality constraint $h(\mathbf{W}) = 0$ as in Equation (1.2). Then the problem could be converted into an

unconstrained problem as follows:

$$\min_{\mathbf{W}, \mathbf{A}} f(\mathbf{W}, \mathbf{A}) + \frac{\rho}{2} |h(\mathbf{W})|^2 + \alpha h(\mathbf{W}),$$

where $\rho > 0$ is a penalty parameter and α is the Lagrangian multiplier.

Limitation of the current causal structural learning methods

The current causal structural learning methods discussed in Section 1.3.1 have assumed that the causal relation is constant across time and treat observations across time as duplicates with the identical distribution. However, this assumption may not hold in real applications and the causal relations may change with time. For instance, Wu et al. (2020b) has shown that putting on a mask against wildfire smoke may be effective at the beginning, but will become less effective after a long time of exposure. Failing to capture the dynamic pattern of the causal graph, the conventional approaches may have large bias assuming the same causal structure among all observations.

1.3.2 Dynamic causal effects

In this section, we review the literature on the dynamic causal effects. The related work could be divided into two categories: dynamic causal effect in longitudinal studies and dynamic Granger causality in time-series analysis.

Dynamic causal effects in longitudinal studies

Considering the data structure of repeated treatments and covariates, this line of literature often aims to estimate the treatment effect based on the structural nested mean model (SNMM) (Robins 1994).

Assume n independent subject's measurements are taken at time points t_0, t_1, \dots, t_K . Let A_k denote the binary treatment variable where 0 represents no treatment, X_k denote other covariate variables and Y_k denote the outcome variables measured at time t_k . By assuming the treatment at or after t_k won't affect the outcome at t_k , the SNMM models the effect of the "blip" of treatment variables, that is, holding all future treatments fixed at level 0 as follows (Vansteelandt and Joffe 2014):

$$\begin{aligned} & g(\mathbb{E}[Y_{k+1}^{\bar{a}_k, \mathbf{0}} | \bar{X}_k = \bar{x}_k, \bar{Y}_k = \bar{y}_k, \bar{A}_k = \bar{a}_k]) - g(\mathbb{E}[Y_{k+1}^{\bar{a}_{k-1}, \mathbf{0}} | \bar{X}_k = \bar{x}_k, \bar{Y}_k = \bar{y}_k, \bar{A}_k = \bar{a}_k]) \\ & = \gamma_k(\bar{x}_k, \bar{y}_k, \bar{a}_k; \psi), \end{aligned}$$

where overbar denotes the history of the respective variables, e.g. $\bar{A}_k = (A_1, \dots, A_k)$, $Y_k^{(\bar{a}_{k-1})}$ denote the potential outcome at time t_k if the individual had the treatment history \bar{a}_{k-1} , $(\bar{a}_k, \mathbf{0})$ denotes the counterfactual history as the history that agrees with a_k through time t_k and is 0 thereafter, $g(\cdot)$ denotes a known link function, and $\gamma_k(\cdot)$ denotes a known function that must satisfy that $\gamma_k(\bar{x}_k, \bar{a}_{k-1}, \mathbf{0}, \psi) = 0$.

Instead of conditioning on all the past variables, Boruvka et al. (2018) generalize the SNMM framework that only requires a subset of the past variables. Since the treatment's effect on the outcome variable may not be intermediate, the lag- m effect is proposed as:

$$\mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 1, a_{k+1}^{a_k=1}, \dots, a_{k+m-1}^{a_k=1})} | S_{mk}(\bar{A}_{k-1})] - \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 0, a_{k+1}^{a_k=0}, \dots, a_{k+m-1}^{a_k=0})} | S_{mk}(\bar{A}_{k-1})], \quad m = 1, 2, \dots,$$

where $a_{k+1}^{a_k=1}$ denotes the potential treatment at time $k+1$ had the treatments a_k been allocated at level 1. $S_{mk}(\bar{A}_{k-1})$ is a subset of the history variable $H_k(\bar{A}_{k-1})$, where

$$H_k(\bar{A}_{k-1}) = (X_1, A_1, Y_2^{(A_1)}, \dots, X_k^{(\bar{A}_{k-1})}, A_{k-1}, Y_k^{(\bar{A}_{k-1})}),$$

and $X_k^{(\bar{A}_{k-1})}$ denotes potential measurements at time k had the sequence of treatments \bar{A}_{k-1} been allocated.

Under the positivity, consistency, and sequential ignorability assumptions (Boruvka et al. 2018), the lag- m effect could be simplified as follows if $S_{mk}(\bar{A}_{k-1}) = H_k = (\bar{X}_k, \bar{Y}_k, \bar{A}_{k-1})$:

$$\begin{aligned} & \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 1, a_{k+1}^{a_k=1}, \dots, a_{k+m-1}^{a_k=1})} | S_{mk}(\bar{A}_{k-1})] - \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 0, a_{k+1}^{a_k=0}, \dots, a_{k+m-1}^{a_k=0})} | S_{mk}(\bar{A}_{k-1})] \\ &= \mathbb{E}[Y_{k+m} | A_k = 1, H_k] - \mathbb{E}[Y_{k+m} | A_k = 0, H_k]. \end{aligned}$$

The SNMM is also extended by Wu et al. (2020a) to allow the causal effect to change with both time and locations:

$$\mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 1)} | \bar{X}_k = \bar{x}_k, \bar{A}_{k-1} = \bar{a}_{k-1}] - \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 0)} | \bar{X}_k = \bar{x}_k, \bar{A}_{k-1} = \bar{a}_{k-1}] = \gamma_{k,m}(\bar{x}_k, \bar{a}_k; \psi(s)),$$

where $\psi(s)$ denotes the location-related parameters. By doing so, the spatially varying structure nested mean model could consider both time-varying and spatially-varying effects.

Dynamic Granger causality in time-series analysis

There is another line of work estimating time-varying Granger causality (Granger 1969) in time-series analysis. Widely used in time-series analysis, Granger causality is defined if the predictability is improved by incorporating covariates. For instance, variable X is causing

variable Y , if the $\sigma^2(Y|A, X) < \sigma^2(Y|A)$, where X are other covariate variables and $\sigma^2(Y|A)$ denotes the variance of the prediction error fitting Y using A .

A common approach to estimate the Granger causality is to fit a regression-form model and obtain the estimated regression coefficients which are interpreted as the Granger causality. For instance, Du et al. (2019) propose a hierarchical regression model to estimate the regression coefficient and the time-lag simultaneously, to take the time-lagged dependencies into consideration. Dropping the fixed variance assumption, Huang et al. (2019) allows both causal strengths and noises variances to vary over time and use auto-regressive models to capture the time dependencies:

$$\begin{cases} x_{i,t} &= \sum_j b_{ij,t} x_{j,t} + e_{i,t}, & e_{i,t} \sim N(0, \exp(h_{i,t})) \\ b_{ij,t} &= \alpha_{ij,0} + \sum_{p=1}^{p_l} \alpha_{ij,l} b_{ij,t-p} + \epsilon_{ij,t} \\ h_{i,t} &= \beta_{i,0} + \sum_{q=1}^{q_l} \beta_{i,q} h_{i,t-q} + \eta_{i,t}, \end{cases} \quad (1.5)$$

where $x_{i,t}$ is the i^{th} variable at time t for the time series X_t , $b_{ij,t}$ denotes the causal coefficient between i^{th} and j^{th} variable ($x_{i,t}, x_{j,t}$) at time t , $e_{i,t}$ denotes the respective error term at time t , $h_{i,t}$ represents the volatility of the error term $e_{i,t}$, \exp denotes the exponential function, $\alpha_{ij,l}, \beta_{i,q}$ captures the time-dependencies of the causal coefficient $b_{ij,t}$ and the volatility $h_{i,t}$, and $\epsilon_{ij,t}, \eta_{i,t}$ denote the errors.

Without loss of generality, we could write the first equation of Equation (1.5) as a generalization of the linear structural equation model:

$$\mathbf{X}_t = \mathbf{X}_t \mathbf{B}_t + \mathbf{E}_t,$$

where \mathbf{X}_t is the data matrix, \mathbf{B}_t is the causal coefficient matrix and \mathbf{E}_t is the error matrix at time t . Based on Equation (1.5), Huang et al. (2019) estimate the time-varying Granger causality $b_{ij,t}$ using the stochastic approximation expectation maximization (SAEM) algorithm (Delyon et al. 1999), while allowing the variance of the causality to vary with time.

Limitation of the current dynamic causal effects

The current two lines of the dynamic causal effects both have limitations for estimating the causal effects of the dynamic causal graphs. The dynamic causal effect in longitudinal studies mainly estimates the treatment effect without assuming a structural model to describe the relations between variables. Without a structural model, it may fail to provide detailed information on how the treatment variable affects the outcome variable via causal paths. On the other hand, the dynamic Granger causality focuses on the time-varying coefficient, which could be seen as

the estimated edge weight between two variables in the causal graph. It may fail to measure the total causal effects between the treatment and outcome variable, since the treatment variable may affect the outcome via multiple causal paths which could not be represented in a single edge weight.

1.4 Proposed methods

To address the limitations in Section 1.3.1 and 1.3.2, we propose a new framework to model the dynamic causal graph where the causal relations are allowed to be time-varying. In Section 1.4.1, we propose a dynamic linear structural equation model to represent the dynamic causal relations and allow both changes in the causal strength and the causal graph structure. In Section 1.4.2, we present the form of the commonly-used causal effects based on the dynamic linear structural equation model. In Section 1.4.3, we incorporate the basis approximation method into the score-based causal discovery approach to capture the dynamic pattern of the causal graphs. In Section 1.4.4, we introduce an algorithm that could provide both past-time estimates and future-time predictions on the causal graphs.

1.4.1 Dynamic linear structural equation model

Specifying the relationship between variables through linear equations, the linear structural equation model (LSEM) (Drton et al. 2011) could be expressed as follows:

$$\mathbf{X} = \mathbf{XA} + \mathbf{E},$$

where $\mathbf{X} \in \mathbb{R}^{m \times p}$ is a data matrix for m observation and p variables, $\mathbf{A} \in \mathbb{R}^{p \times p}$ is an adjacency matrix of the DAG that characterizing the causal relationship of \mathbf{X} and $\mathbf{E} \in \mathbb{R}^{m \times p}$ is the noise matrix.

Since the causal relations among the variables may not be static, we extend the linear structural equation model to allow the causal structure to change with time and propose the following dynamic linear structural equation model (dynamic LSEM):

$$\mathbf{X}_t = \mathbf{X}_t \mathbf{B}_t + \mathbf{E}_t, \quad \text{for } t = 1, \dots, T, \quad (1.6)$$

where $\mathbf{X}_t, \mathbf{B}_t, \mathbf{E}_t$ denotes the $m \times p$ data matrix, $p \times p$ weighted adjacency matrix, $m \times p$ error matrix at t^{th} timestamp, and T denotes the number of the timestamps. It is worth mentioning that we assume that there are no time-lagged dependencies in this dynamic linear structural equation model, and it is a direction for future work to take the time-lagged causal relations

into consideration.

1.4.2 Dynamic causal effects

As proposed by (Cai et al. 2020), we could decompose the data matrix $\mathbf{X}_t \in \mathbb{R}^{m \times p}$ into the treatment variable, A_t , mediator variable \mathbf{M}_t and the outcome variable Y_t , $\mathbf{X}_t = [A_t, \mathbf{M}_t^T, Y_t]$. Then the Equation 1.6 could be written as:

$$\begin{aligned} [A_t, \mathbf{M}_t^T, Y_t] &= [A_t, \mathbf{M}_t^T, Y_t] \mathbf{B}_t + \mathbf{E}_t \\ &= [A_t, \mathbf{M}_t^T, Y_t] \begin{bmatrix} 0 & \boldsymbol{\alpha}_t & \gamma_t \\ \mathbf{0}_{(p-2) \times 1} & \mathbf{C}_t & \boldsymbol{\beta}_t \\ 0 & \mathbf{0}_{1 \times (p-2)} & 0 \end{bmatrix} + \mathbf{E}_t, \end{aligned} \quad (1.7)$$

where γ_t is a scalar, $\boldsymbol{\alpha}_t^T, \boldsymbol{\beta}_t, \mathbf{0}_{(p-2) \times 1}$ are $(p-2) \times 1$ vectors, \mathbf{C}_t is a $(p-2) \times (p-2)$ matrix, and $\mathbf{E}_t = [\epsilon_{A_t}, \boldsymbol{\epsilon}_{\mathbf{M}_t}, \epsilon_{Y_t}]$ is the error matrix. Since treatment has no parent node and the outcome variable won't affect the treatment or mediators, the first column and the last row of the weight matrix \mathbf{B}_t are all zero.

As presented in Pearl (2009), the natural direct effect, natural indirect effect, and total effect are:

$$\begin{aligned} DE_t &= E(Y_t | do(A_t = a+1, M_t = m^{(a)})) - E(Y_t | do(A_t = a)) \\ IE_t &= E(Y_t | do(A_t = a, M_t = m^{(a+1)})) - E(Y_t | do(A_t = a)), \\ TE_t &= E(Y_t | do(A_t = a+1)) - E(Y_t | do(A_t = a)) \\ &= E(Y_t | do(A_t = a+1, M_t = m^{(a)})) - E(Y_t | do(A_t = a)) \\ &\quad - [E(Y_t | do(A_t = a+1, M_t = m^{(a)})) - E(Y_t | do(A_t = a+1))] \\ &= DE_t + IE_t, \end{aligned} \quad (1.8)$$

where DE_t denotes the direct effect, IE_t denotes the indirect effect, TE_t denotes the total effect at t^{th} time stamp, $do(A_t = a)$ denotes the physical interventions by replacing A_t by a constant a , while keeping the rest of the model unchanged, and $m^{(a)}$ denotes to setting M_t to the value that it would be when $A_t = a$. The second to last equality suggests that the total effect is equal to the difference between the direct effect and the indirect effect of the reverse transition, i.e., intervening the treatment variable from $(a+1)$ to a . The last equality holds because that reversal of transition is the same as changing the sign of the indirect effect.

With the no-unmeasured confounder assumption, we assume the following:

(A1) At each time point t , the effect of the treatment A_t on the outcome Y_t is unconfounded, i.e. $Y_t(A_t = a; M_t = m) \perp A_t$;

(A2) At each time point t , the effect of the treatment A_t on the mediators M_t is unconfounded,

i.e. $M_t(A_t = a) \perp A_t$;

(A3) At each time point t , the effect of the mediators M_t on the outcome Y_t is unconfounded given the treatment A_t , i.e. $Y_t(A_t = a; M_t = m) \perp M_t | A_t$,

where $Y_t(A_t = a)$ denotes the potential outcome, i.e., the value that outcome Y_t would obtain if the treatment variable A_t has been a .¹

As shown in Equation (1.7), we could write

$$\begin{cases} A_t &= \epsilon_{A_t} \\ \mathbf{M}_t &= \boldsymbol{\alpha}_t^T A_t + \mathbf{C}_t^T \mathbf{M}_t + \boldsymbol{\epsilon}_{\mathbf{M}_t}^T \\ Y_t &= \gamma_t A_t + \boldsymbol{\beta}_t^T \mathbf{M}_t + \epsilon_{Y_t}, \end{cases} \quad (1.9)$$

where $\gamma_t, \epsilon_{A_t}, \epsilon_{Y_t}$ is a scalar, $\boldsymbol{\alpha}_t^T, \boldsymbol{\beta}_t, \boldsymbol{\epsilon}_{\mathbf{M}_t}^T$ are $(p-2) \times 1$ vectors, \mathbf{C}_t is a $(p-2) \times (p-2)$ matrix. Since the matrix, \mathbf{C}_t^T is upper-triangular when we sort the graph nodes in the topological order, the existence of the inverse for matrix $(\mathbf{I} - \mathbf{C}_t^T)$ is guaranteed.

Theorem 1.4.1. Under Assumptions A1-A3 and Equation (1.9), we could have the following causal effects:

The direct effect is $DE_t = \gamma_t$, the indirect effect is $IE_t = \boldsymbol{\beta}_t^T (\mathbf{I} - \mathbf{C}_t^T)^{-1} \boldsymbol{\alpha}_t^T$, and the total effect is $TE_t = \gamma_t + \boldsymbol{\beta}_t^T (\mathbf{I} - \mathbf{C}_t^T)^{-1} \boldsymbol{\alpha}_t^T$.

The proof could be found in the Appendix Chapter A.

1.4.3 Basis approximation for varying coefficient modeling

To obtain estimates for the dynamic causal effects, we try to estimate the weighted adjacency matrix \mathbf{B}_t under the dynamic linear structural equation model.

The dynamic linear structural equation model could be seen as a varying coefficient model (Hastie and Tibshirani 1993), where the coefficient \mathbf{B}_t is a function of time t . Nonparametrically modeling the coefficients may lead to the curse of dimensionality, since the complexity grows when the number of time stamps is large (Fan and Zhang 2008). Instead, due to its ease of implementation and fast convergence, we utilize the basis approximation method (Huang et al. 2002) to estimate the varying coefficient model, which conducts global smoothing on the

¹This assumption may be too strong to hold in the real-life scenario, since it requires that A_t is the only common causes for Y_t and M_t . It could be a future direction to relax this assumption to improve this model's generalization abilities.

coefficients. Then, each element of the coefficient matrix could be formulated as follows:

$$B_{mn,t} \simeq \sum_{k=1}^K F_k(t) \gamma_{mnk}, \quad (1.10)$$

$$m = 1, \dots, p, \quad n = 1, \dots, p, \quad t = 1, \dots, T,$$

where $B_{mn,t}$ denotes the $(m, n)^{th}$ element of the weighted adjacency matrix \mathbf{B}_t , t represents t^{th} timestamp, $F_k(\cdot)$ is k^{th} basis function, K is the number of basis, p is the number of covariates, T is the number of timestamps and γ_{mnk} are the respective coefficients of $B_{mn,t}$ for k^{th} basis. Note that although the number of basis is finite in the current implementation, the approximation error may be ignorable since the number of basis is close to the maximum number of basis, which is equal to the length of the time span. Thus, we utilize equal signs in the following discussions.

Then the dynamic LSEM model in Equation (1.6) could be written as

$$\begin{aligned} \mathbf{X}_t &= \begin{bmatrix} X_{t11} & \dots & X_{t1p} \\ \vdots & \ddots & \vdots \\ X_{tm1} & \dots & X_{tmp} \end{bmatrix} \\ &= \begin{bmatrix} X_{t11} & \dots & X_{t1p} \\ \vdots & \ddots & \vdots \\ X_{tm1} & \dots & X_{tmp} \end{bmatrix} \begin{bmatrix} \sum_{k=1}^K F_k(t) \gamma_{11k} & \dots & \sum_{k=1}^K F_k(t) \gamma_{1pk} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^K F_k(t) \gamma_{p1k} & \dots & \sum_{k=1}^K F_k(t) \gamma_{ppk} \end{bmatrix} + \mathbf{E}_t \\ &= \begin{bmatrix} F_1(t)X_{t11} & \dots & F_1(t)X_{t1p} & \dots & F_K(t)X_{t11} & \dots & F_K(t)X_{t1p} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ F_1(t)X_{tm1} & \dots & F_1(t)X_{tmp} & \dots & F_K(t)X_{tm1} & \dots & F_K(t)X_{tmp} \end{bmatrix} \\ &\quad \begin{bmatrix} \gamma_{111} & \dots & \gamma_{1p1} \\ \vdots & \ddots & \vdots \\ \gamma_{p11} & \dots & \gamma_{pp1} \\ \vdots & \ddots & \vdots \\ \gamma_{11K} & \dots & \gamma_{1pK} \\ \vdots & \ddots & \vdots \\ \gamma_{p1K} & \dots & \gamma_{ppK} \end{bmatrix} + \mathbf{E}_t, \quad t = 1, \dots, T, \end{aligned}$$

and we could denote it as

$$\mathbf{X}_t = \mathbf{D}_t \mathbf{\Gamma} + \mathbf{E}_t, \text{ where } t = 1, \dots, T,$$

$$\mathbf{D}_t = \begin{bmatrix} F_1(t)X_{t11} & \dots & F_1(t)X_{t1p} & \dots & F_K(t)X_{t11} & \dots & F_K(t)X_{t1p} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ F_1(t)X_{tm1} & \dots & F_1(t)X_{tmp} & \dots & F_K(t)X_{tm1} & \dots & F_K(t)X_{tmp} \end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix} \gamma_{111} & \dots & \gamma_{1p1} \\ \vdots & \ddots & \vdots \\ \gamma_{p11} & \dots & \gamma_{pp1} \\ \vdots & \ddots & \vdots \\ \gamma_{11K} & \dots & \gamma_{1pK} \\ \vdots & \ddots & \vdots \\ \gamma_{p1K} & \dots & \gamma_{ppK} \end{bmatrix},$$

where \mathbf{X}_t denotes the $m \times p$ data matrix, T denotes the number of the timestamps, and \mathbf{E}_t

denotes $m \times p$ error matrix at t^{th} timestamp.

Generally, we could express the data matrix as

$$\mathbf{X} = \mathbf{D}\mathbf{\Gamma} + \mathbf{E}, \quad (1.11)$$

where $\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_T \end{bmatrix}$, $\mathbf{D} = \begin{bmatrix} D_1 \\ \vdots \\ D_T \end{bmatrix}$ and $\mathbf{E} = \begin{bmatrix} E_1 \\ \vdots \\ E_T \end{bmatrix}$.

1.4.4 Constrained causal structural learning

A natural candidate to solve the basis approximation of the dynamic LSEM is the regression-based method. However, the regression-based method often imposes assumptions on the noise distribution, and this linear structure may fail to capture more complex data distribution. To overcome these limits, we adopt the deep generative model proposed by Yu et al. (2019), where a variational auto-encoder (VAE) is utilized.

Specifically, we could reformulate Equation (1.11) in an LSEM format:

$$\begin{bmatrix} \mathbf{X} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{X} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{p \times p} & \mathbf{0}_{p \times pK} \\ \mathbf{\Gamma} & \mathbf{0}_{pK \times pK} \end{bmatrix} + \begin{bmatrix} \mathbf{E} & \mathbf{E}' \end{bmatrix},$$

which implies that

$$\begin{aligned} & \begin{bmatrix} \mathbf{X} & \mathbf{D} \end{bmatrix} \left\{ \mathbf{I}_{(p+pK) \times (p+pK)} - \begin{bmatrix} \mathbf{0}_{p \times p} & \mathbf{0}_{p \times pK} \\ \mathbf{\Gamma} & \mathbf{0}_{pK \times pK} \end{bmatrix} \right\} \\ &= \begin{bmatrix} \mathbf{X} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{p \times p} & \mathbf{0}_{p \times pK} \\ -\mathbf{\Gamma} & \mathbf{I}_{pK \times pK} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E} & \mathbf{E}' \end{bmatrix}, \end{aligned}$$

where $[\mathbf{E}, \mathbf{E}']$ is a noise matrix which are then treated as latent variables in the variational autoencoder and MLP layers are utilized in both encoder and decoder structure.

Equivalently, we could write:

$$\begin{bmatrix} \mathbf{X} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{E} & \mathbf{E}' \end{bmatrix} \begin{bmatrix} \mathbf{I}_{p \times p} & \mathbf{0}_{p \times pK} \\ -\mathbf{\Gamma} & \mathbf{I}_{pK \times pK} \end{bmatrix}^{-1},$$

and we could treat the $[\mathbf{E} \ \mathbf{E}']$ as latent variables that generate the data matrix.

To ensure that the obtained graph at each time point is a DAG, we could express the acyclic

constraint (Yu et al. 2019) as:

$$h_1(\mathbf{\Gamma}) = \sum_{t=1}^T \{ |tr[(\mathbf{I}_p + \alpha \mathbf{B}_t \odot \mathbf{B}_t)^p] - p| \} = 0,$$

where $tr(\cdot)$ represents the trace of the matrix, \odot is the element-wise multiplication, \mathbf{B}_t could be written as a function of $\mathbf{\Gamma}$ as in Equation (1.10), p is the number of variables, T is the number of time-stamps and α is a hyperparameter.

To preserve the topological order for the treatment variable and the output variable, we could add an identifiable constraint (Cai et al. 2020) by

$$h_2(\mathbf{\Gamma}) = \sum_{i=1}^p \sum_{j=1}^{p+pK} |\gamma_{i,j}| + \sum_{i=p+1}^{pK} \sum_{j=p+1}^{pK} |\gamma_{i,j}| = 0,$$

where $\gamma_{i,j}$ is the $(i, j)^{th}$ element for matrix $\mathbf{\Gamma}$, so that the treatment variable would have no parent nodes and the outcome variable would have no child nodes.

For the objective function, since the marginal log-likelihood is intractable due to the true posterior density, we utilize evidence lower bound (L_{ELBO}) (Kingma and Welling 2013) as the score function, which is differed from the marginal log-likelihood by the Kullback-Leibler divergence of the variational posterior from the true posterior.

Thus, we could obtain the underlying causal graph by solving

$$\begin{cases} \min_{\mathbf{\Gamma}} -L_{ELBO}, \\ \text{subject to } h_1(\mathbf{\Gamma}) = 0 \text{ and } h_2(\mathbf{\Gamma}) = 0. \end{cases} \quad (1.12)$$

1.5 Simulation Studies

In this section, we evaluate the performance of the proposed method by conducting the simulation studies and comparing its performance with some benchmark methods listed in Section 1.3.1.

1.5.1 Simulation settings

The data is generated based on the dynamic LSEM model specified in Equation (1.6) with Gaussian errors, with 5 variables ($p = 5$), 10 time stamps ($T = 10$), 30 observations ($m = 30$) and 30 realizations. We consider the following two scenarios of data:

- Scenario 1 (S1): The true underlying causal graph has only one edge ($A \rightarrow Y$), and the strength of the causal relation change with time;
- Scenario 2 (S2): The true underlying causal graph is generated from the Erdos-Renyi model with an expected degree as 4 and the strength of the causal relations are randomly assigned to be time-varying or static.

We consider the following two functions to describe the time-varying causal relations:

- Function 1 (F1): Cosine function: $f_1(t) = \cos(\frac{t}{4\pi}) * 0.8$;
- Function 2 (F2): Quadratic function: $f_2(t) = \frac{-10+(5-t)^2}{20}$.

1.5.2 Basis selection

For the proposed method, to obtain an accurate basis approximation for the varying coefficient model, it is essential to select a basis system and determine the number of basis. In the following experiments, we utilize the B-spline as a basis system, since it has local support which is shown to improve the computational efficiency (Fey et al. 2018).

A B-spline with the order of zero is defined as follows (Edwards et al. 2019):

$$f_{i,0}(x) = \begin{cases} 0 & \text{if } x < k_{(i-1)} \text{ or } x \geq k_i, \\ 1 & \text{otherwise,} \end{cases}$$

where $f_{i,0}(x)$ denotes i^{th} B-spline function of order zero, $i = 1, \dots, k$, k_i denotes the i^{th} knots that characterize the B-spline, k denotes the number of the knots.

Then, B-splines with higher orders could be recursively defined as follows:

$$f_{i,r}(x) = v_{i,r} f_{i,r-1}(x) + (1 - v_{i+1,r}) f_{i+1,r-1}(x),$$

where r denotes the degree of the basis function, and

$$v_{i,r} = \begin{cases} \frac{x - k_{i-1}}{k_{i+r-1} - k_{i-1}}, & k_{i+r-1} \neq k_{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

As commonly used in the literature, we utilize the order-2 B-spline with equally spaced knots, where the number of knots is selected using cross-validation (Huang et al. 2002).

As shown in Figure 1.2, increasing the number of knots from 1 to 2 would lead to better estimates for the causal strength, while similar estimates would be obtained when there are more than 2 knots.

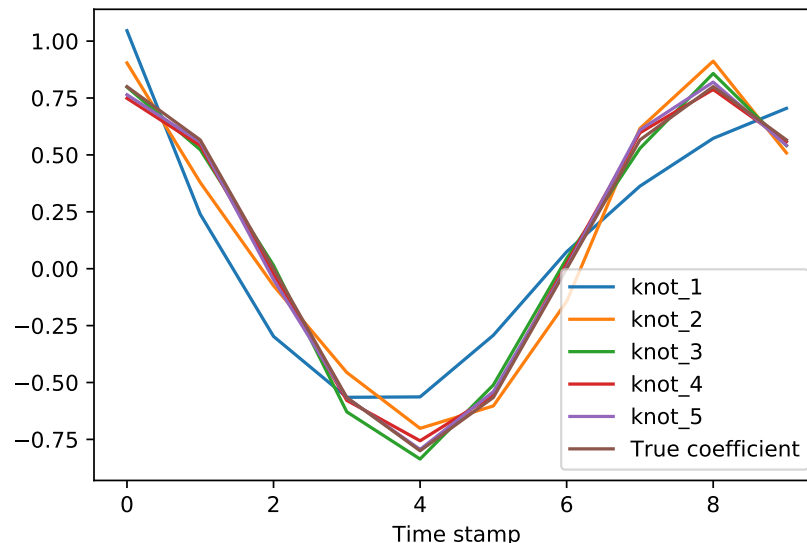


Figure 1.2: Causal strength estimates with different number of knots under scenario 1 with cosine function in the dynamic LSEM setup using the proposed method.

1.5.3 Benchmark methods and evaluation metrics

To evaluate the performance of the proposed method, we utilize the proposed method to identify the hidden causal graph and compare it with commonly used causal discovery approaches: NOTEARS (Zheng et al. 2018), ANOCE (Cai et al. 2020), and DYNOTEARS (Pamfil et al. 2020). To better assess the the proposed models’ ability to capture the dynamic pattern, we also utilize the CD-NOD (Zhang et al. 2017a) as an benchmark, which could conduct causal discovery for non-stationary process.²

Since both benchmark approaches are developed for the static causal graph, for each timestamp, we generate estimates for the causal graph using the current timestamp’s data and use the most-recent estimate as the one-step ahead prediction. As commonly used in the causal discovery literature, we remove the edges if the edge weight is lower than 0.2 to reduce

²The CD-NOD method is developed in the setup where there are no replicates and have long time-spans. Thus, we cannot apply it in the forementioned scenarios and tested it in the scenario where $m = 1, T = 100$.

noise. We evaluate the graph estimates using the following three metrics: false discovery rate (FDR), true positive rate (TPR), and the structural Hamming distance (SHD). To evaluate the time-varying strength of the causal relations, we utilize the mean squared error (MSE) as an evaluation metric. It is the higher the better for the TPR, while for the other three metrics, a lower value indicates better results.

Implementation details

In this section, we present the implementation details for the proposed method and the benchmark methods as follows:

- The computation is done by the processor Intel(R) Core(TM) i7-8550U CPU. The dataset and the code are publicly available at the repository <https://github.com/jackie31425/Dynamic-Causal-Structure-Discovery-and-Causal-Effect-Estimation/>.
- Proposed method: The proposed method is implemented based on PyTorch (Paszke et al. 2017) using Adam (Kingma and Ba 2014) optimizer to optimize the loss function. The batch size is set to be 10, training epoch is 200, and the number of hidden nodes is the square of the number of variables. The maximum iteration number for searching parameters is 100, the initial learning rate is 0.003 which decays with a decay rate of 1.0.
- ANOCE (Cai et al. 2020): ANOCE provides a decomposition of the indirect effect, to categorize interactions between mediators. We implement the ANOCE with the default hyper-parameters to estimate the causal graph at each time-stamp using each-times' data. Their code is available at the repository <https://github.com/hengruicai/ANOCE-CVAE>.
- NOTEARS (Zheng et al. 2018) converts the combinatorial optimization problem into a continuous optimization problem by setting the acyclicity constraint as a function of the adjacency matrix. We implement the NOTEARS with the default hyper-parameters to estimate the causal graph at each time-stamp using each-times' data. Their code is available at the repository <https://github.com/xunzheng/notears>.
- DYNOTEARS (Pamfil et al. 2020) extends the NOTEARS model to handle the scenario where the replicates are no longer independent but have interactions. We implement the DYNOTEARS with the default hyper-parameters to estimate the causal graph at each time-stamp using lag-1 data. Their code is available at the repository <https://github.com/mckinsey/causalnex/blob/develop/causalnex/structure/dynotears.py>.

- CD-NOD (Zhang et al. 2017b) proposes the constraint-based procedure to detect the changing causal structure. We implement the CD-NOD in Matlab (Inc. 2022) with the default hyper-parameters and set the number of iterations to be 1000. Their code is available at <https://github.com/Biwei-Huang/>.

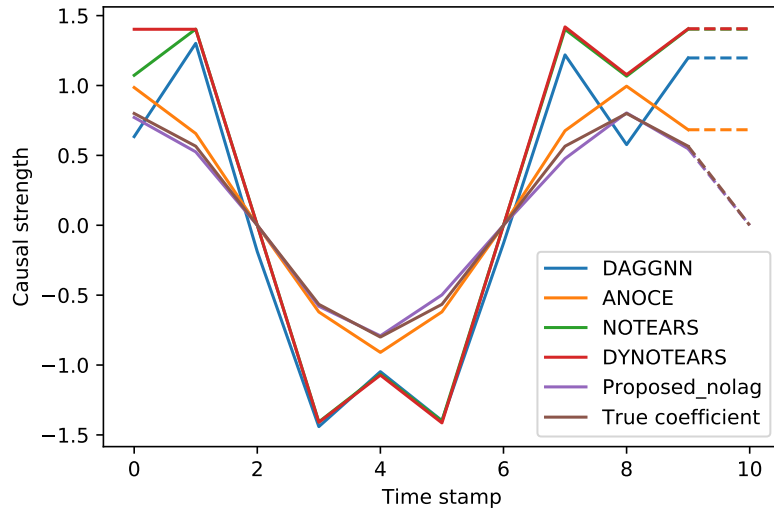
1.5.4 Results

The results for the proposed method and the benchmark methods are shown in Figures 1.3,1.4,1.5 and Table 1.1,1.2. The findings are summarized as follows.

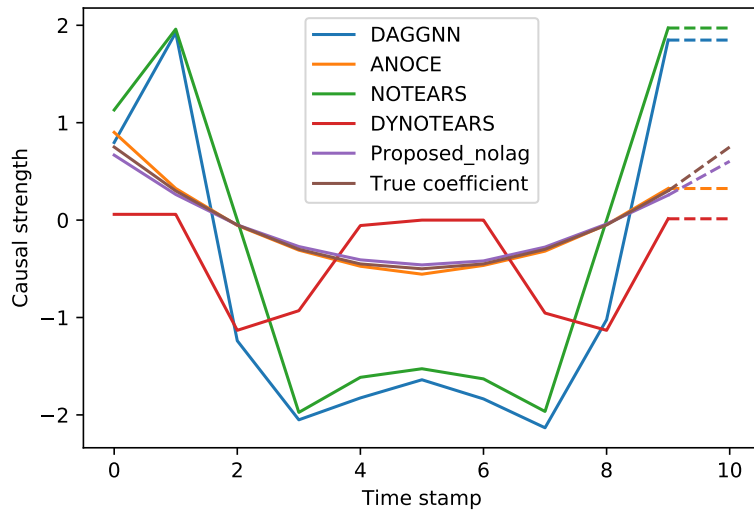
As shown in Figure 1.3, the proposed method could generate more accurate estimates for the causal strength in both cosine and quadratic function setups. Moreover, the proposed method is able to obtain an accurate prediction for the next time stamp, while the other benchmark methods fail to do so.

Table 1.1: The empirical comparison of the estimated causal graph for the synthetic data in the dynamic LSEM setup. The number in parenthesis denotes the standard deviation.

Methods	Metric	S1F1	S1F2	S2
DAGGNN	FDR	0.83(0.01)	0.80(0.01)	0.84(0.03)
	TPR	0.82(0.04)	0.90(0.03)	0.48(0.03)
	SHD	3.36(0.16)	3.38(0.15)	5.10(0.17)
	MSE	0.65(0.04),	2.69(0.26)	0.14(0.02)
ANOCE	FDR	0.54(0.03)	0.54(0.03)	0.36(0.02)
	TPR	1.00(0.00)	1.00(0.00)	0.84(0.03)
	SHD	1.83(0.13)	1.81(0.13)	2.83(0.17)
	MSE	0.06(0.02),	0.02 (0.01)	0.15(0.01)
NOTEARS	FDR	0.00(0.00)	0.00(0.00)	0.50(0.02)
	TPR	1.00(0.00)	1.00(0.00)	0.25(0.03)
	SHD	0.00(0.00)	0.00(0.00)	3.02(0.17)
	MSE	0.37(0.00),	1.51 (0.01)	0.52(0.01)
DYNOTEARS	FDR	0.00(0.00)	0.50(0.04)	0.36(0.02)
	TPR	1.00(0.00)	0.51(0.04)	0.25(0.00)
	SHD	0.00(0.00)	0.70(0.06)	3.03(0.01)
	MSE	0.66(0.00),	0.54(0.01)	0.50(0.01)
Proposed	FDR	0.00 (0.00)	0.00 (0.00)	0.12(0.01)
	TPR	1.00(0.00)	0.94(0.03)	0.85 (0.02)
	SHD	0.01(0.01)	0.06(0.03)	1.12(0.11)
	MSE	0.00(0.00)	0.01 (0.00)	0.08(0.01)



(a) Cosine function F1



(b) Quadratic function F2

Figure 1.3: Estimated causal strength using the proposed method, DAGGNN, NOTEARS, DYNOTEARS and ANOCE for Scenario S1 under different time-varying coefficient functions. The solid line denotes the estimated coefficient in the ten time stamps and the dashed line denotes the one-step ahead prediction.

Also, as shown in Table 1.1, we could conclude that the proposed method could also capture the hidden graph structure, since the proposed method obtains better results in terms of FDR, SHD, and MSE in all the scenarios while having comparable TPR with benchmark methods. We could also notice that, ANOCE could obtain relatively accurate estimates for the time-varying

causal strength, but fails to capture the whole graph structure since it only utilizes the current time information. NOTEARS, on the other hand, could detect the signals correctly but fail to generate accurate estimates for the causal strength. Moreover, as demonstrated in Figure 1.4,1.5, the proposed method could accurately capture the entire causal graph structure, while the other methods estimate wrong edges.

Table 1.2 shows the empirical results of the estimated causal graph for the synthetic data in the dynamic LSEM setup, when $m = 1, T = 100$. The causal strength changes with time following $f(t) = \cos(\frac{t}{30\pi}) * 0.8$. Since the other methods are not designed for no-replicate scenarios, we only present the results for the proposed method and CD-NOD (Zhang et al. 2017b). As shown in this table, our method has shown superior performance in all metrics, indicating its ability to correctly estimate the dynamic causal structure with long-time and no-replicate scenario.

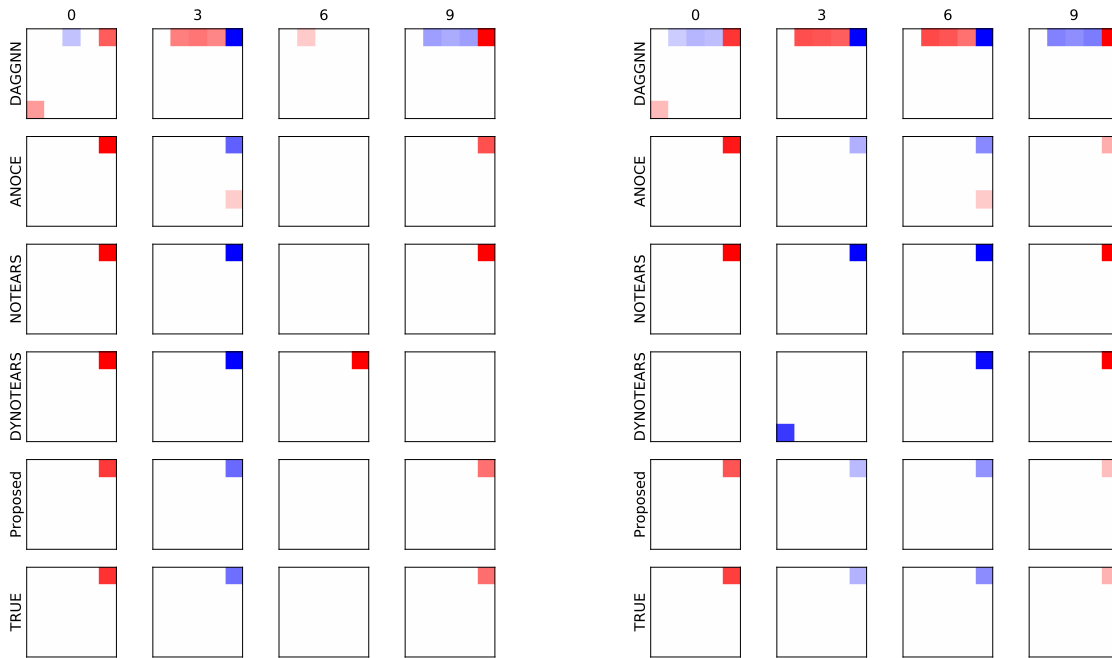
Table 1.2: The empirical comparison of the estimated causal graph for the synthetic data in the dynamic LSEM setup, when $m = 1, T = 100$. The number in parenthesis denotes the standard deviation.

Metric	CD-NOD	Proposed
FDR	0.77(0.02)	0.01 (0.00)
TPR	0.34(0.03)	0.94(0.00)
SHD	3.09(0.22)	0.07(0.00)
MSE	0.41(0.01),	0.03(0.00)

1.6 Discussion

In this paper, we develop a new framework to model the dynamic causal graph where causal relations are allowed to vary with time. We propose a dynamic causal structure discovery approach based on the dynamic LSEM model and propose the respective dynamic causal effect. We present an algorithm that could accurately estimate the dynamic casual graphs using all-time information and demonstrate its effectiveness via simulation studies. To conclude, we briefly discuss some limitations and possible future directions.

Firstly, we have made the assumption that there are no temporal causal dependencies, so that only information at the current time-stamp would be causally related. However, this assumption may not hold in real-life scenarios, since there may be lagged dependencies between variables. A possible approach is to extend the structural vector autoregressive model (Pamfil



(a) Cosine function F1

(b) Quadratic function F2

Figure 1.4: Estimated causal graphs at multiple time-stamps using the proposed method and the benchmarks for Scenario 1 based on the dynamic LSEM setup. The horizontal axis denotes the number of timestamps.

et al. 2020) to the dynamic fashion so that we could consider both time-lagged dependencies and the time-varying causal graphs at the same time. Secondly, we utilize the basis approximation method to obtain an estimate for the time-varying causal strength, which may lead to an estimation error in the number of basis. It may be better to incorporate the basis estimation bias into the loss function so that the method could be generalized to datasets with long time spans.

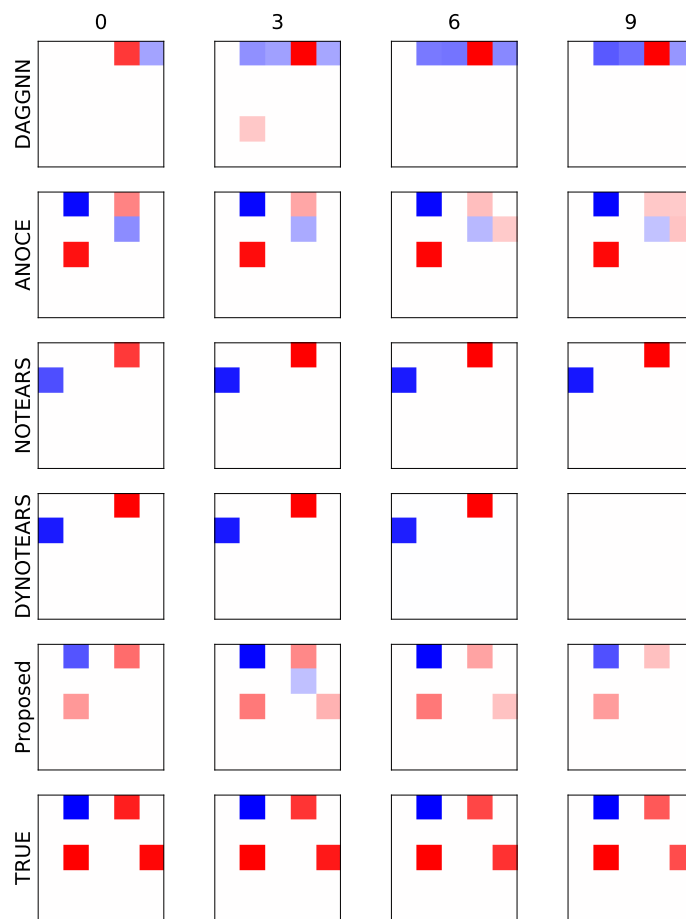


Figure 1.5: Estimated causal graphs at multiple time-stamps using the proposed method and benchmarks for dynamic LSEM setup, Scenario 2. The horizontal axis denotes the number of timestamps.

CHAPTER

2

DYNAMIC CAUSAL STRUCTURE DISCOVERY WITH TIME-LAGGED DEPENDENCY

2.1 Introduction

To represent the causal relationships between variables, a directed acyclic graph (DAG) (VanderWeele and Robins 2007) is widely utilized in many areas, such as social sciences, epidemics, and genetics (Pearl 2009). Based on the causal graph, we could calculate causal effects to illustrate the causal relations better. However, learning a faithful representation of the underlying causal graph is challenging, and there are many recent works that focus on the causal structure discovery problem.

Based on the fact that each graph could be scored using some score functions, such as BIC (Schwarz 1978), score-based approaches focus on finding the DAG which could yield the optimal score. For instance, Chickering (2002) formulate the structure learning problem as a combinatorial optimizing problem and optimize the score by searching over the graph set with a greedy search method. Owing to the constraint that the graph must be acyclic, searching over the graph space is a combinatorial problem which is NP-hard (Chickering et al. 2004), and the

intractable problem poses challenges on optimization due to the computational difficulties in large-scale data. To overcome this challenge caused by acyclicity constraint, Zheng et al. (2018) convert the combinatorial optimization problem into a continuous optimization problem by setting the acyclicity constraint as a function of the adjacency matrix so that black-box solvers could be used to obtain the optimal graph efficiently. Motivated by this constraint setup, Yu et al. (2019) apply the variational auto-encoder model to achieve better performance, and Cai et al. (2020) provide a decomposition of the indirect effect, to categorize interactions between mediators.

Obtaining a causal graph using observations across time, the above literature often has a hidden assumption that the causal relation is fixed across time. However, as shown in the previous chapter, the causal relations may not be static and may change with time and there are many challenges considering both dynamic and time-lagged causal relations in a dynamic way. Firstly, to our best knowledge, the causal effect is not well-defined in the dynamic causal graphs, when the causal structure is subject to changes and time-lagged dependencies. Secondly, there are no off-the-shelf models on the time-varying causal graphs to represent the dynamic causal relations among the variables. Thirdly, allowing time-varying and time-lagged causal strength, i.e., weights that quantify the causal relations, may lead to the curse of dimensionality, since the complexity grows with the number of time stamps.

In this study, we develop a new framework to model the dynamic causal graph and propose definitions of the time-varying causal effect, where there may also exist across-time dependencies. By incorporating the autoregressive model structure, we are able to identify both contemporaneous and time-lagged causal relationships while allowing them to vary with time. We provide an algorithm that could estimate the causal graphs using historical information and also make accurate predictions for the future leveraging information spanning all time-periods. Our main contributions could be summarized as follows:

- We propose the dynamic causal effect to represent treatment variable's effect on the outcome variable, providing a quantitative representation of time-varying causal relationships. The proposed dynamic causal effect is applicable to diverse scenarios where causal relations could be both time-varying and time-lagged.
- We propose a dynamic causal structure discovery approach which integrates the basis approximation method into the score-based method based on the proposed dynamic structural vector autoregression model. The proposed method could accommodate time-varying, time-lagged, and static causal relations.
- We propose an algorithm that could provide both past-time estimates and future-time predictions on the causal graphs, and conduct simulations to demonstrate the usefulness

of the proposed method. We apply the proposed method to the real COVID 19 data analysis and provide causal estimates on how policy restriction's effect changes.¹

The rest of this chapter is organized as follows. We first conduct a literature review on causal structure learning methods and causal effect estimation with time-lagged dependencies in Section 2.2. In Section 2.3, we introduce our proposed method and in Section 2.4, we propose the dynamic causal effect. Then in Section 2.5, we provide simulation studies to evaluate the performance of the proposed method and compare with existing methods. Moreover, we provide real data analysis to validate the performance of the proposed method on the real-world dataset in Section 2.6. Lastly, we summarize the chapter and discuss some future directions in Section 2.7.

2.2 Related work

Focusing on modeling both time-varying and time-lagged causal relationships, our work is closely related with the two lines of work: causal discovery with time-lagged dependencies and time-lagged causal effect.

2.2.1 Causal discovery with time-lagged dependencies

Based on the LSEM model, the previous approaches, such as NOTEARS (Zheng et al. 2018) and DAGGNN (Yu et al. 2019) assume that there are only contemporaneous causal relationships and omit the potential time-lagged causal dependencies. To take both contemporaneous and time-lagged causal relationships into consideration, Pamfil et al. (2020) extend the NOTEARS approach and propose the DYNOTEARS algorithm.

The data is modeled using the SVAR (Structural Vector AutoRegressions) model (Demiralp and Hoover 2003) as follows:

$$\begin{aligned} \mathbf{x}_{m,t}^T &= \mathbf{x}_{m,t}^T \mathbf{W} + \mathbf{x}_{m,t-1}^T \mathbf{A}_1 + \cdots + \mathbf{x}_{m,t-d}^T \mathbf{A}_d + \mathbf{z}_{m,t}^T \\ &\text{for } t \in d, \dots, T, m \in 1, \dots, M, \end{aligned} \tag{2.1}$$

where $\mathbf{x}_{m,t} \in \mathbb{R}^p$ denotes the m^{th} data vector at t^{th} timestamp, p represents the number of variables in the dataset, T represents the number of timestamps, M represents the number of samples at each time stamp, matrices \mathbf{W} and $\mathbf{A}_i, i \in 1, \dots, d$ denote the weighted adjacency

¹The dataset and the code are publicly available at <https://github.com/jackie31425/Dynamic-Causal-Structure-Discovery-and-Causal-Effect-Estimation/>

matrices, d is the autoregressive order, and $\mathbf{z}_{m,t}$ denotes the error vector which is independent within and across time.

By assuming that this network structure is constant across time, Equation (2.1) could be written in the following matrix form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}_1\mathbf{A}_1 + \dots + \mathbf{Y}_d\mathbf{A}_d + \mathbf{Z}, \quad (2.2)$$

where \mathbf{X} is an $n \times p$ matrix whose rows are $\mathbf{x}_{m,t}$, $n = M(T + 1 - d)$ denotes the effective sample size, and matrices $\mathbf{Y}_1, \dots, \mathbf{Y}_d$ are the respective lagged data matrix.

Let $\mathbf{Y} = [\mathbf{Y}_1 | \dots | \mathbf{Y}_d] \in \mathbb{R}^{n \times pd}$ be concatenated matrix for the lagged data, and let $\mathbf{A} = [\mathbf{A}_1^T | \dots | \mathbf{A}_d^T]^T \in \mathbb{R}^{pd \times p}$ denote the weights for the across-time dependencies. The Equation (2.2) could be written in the following compact form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}\mathbf{A} + \mathbf{Z}.$$

Since the future data won't affect the past observations, the edges in \mathbf{A} only go forward in time, which guarantees the acyclicity of the matrix \mathbf{A} . Thus, it suffices to require that \mathbf{W} is acyclic to ensure the whole causal network is acyclic.

In order to estimate the weighted adjacency matrices \mathbf{W} and \mathbf{A} , given the data matrices \mathbf{X} and \mathbf{Y} , the DYNOTEARS solves the following regularized optimization problem:

$$\begin{aligned} & \min_{\mathbf{W}, \mathbf{A}} f(\mathbf{W}, \mathbf{A}) \text{ s.t. } \mathbf{W} \text{ is acyclic} \\ & \text{with } f(\mathbf{W}, \mathbf{A}) = \ell(\mathbf{W}, \mathbf{A}) + \lambda_{\mathbf{W}} \|\mathbf{W}\|_1 + \lambda_{\mathbf{A}} \|\mathbf{A}\|_1, \\ & \ell(\mathbf{W}, \mathbf{A}) = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{W} - \mathbf{Y}\mathbf{A}\|_F^2, \end{aligned}$$

where $\lambda_{\mathbf{W}}, \lambda_{\mathbf{A}}$ denotes the regularization weights for the weight matrices, $\|\cdot\|_1$ denotes the element-wise ℓ_1 norm.

As shown by (Zheng et al. 2018), the acyclicity constraint could be formulated to an equality constraint $h(\mathbf{W}) = 0$ as follows:

$$h(\mathbf{W}) = \text{trace}(e^{\mathbf{W} \circ \mathbf{W}}) - p = 0,$$

where \circ is the element-wise multiplication and $e^{\mathbf{W}}$ is the matrix exponential of \mathbf{W} (Zheng et al. 2018). Then the problem could be converted into an unconstrained problem as follows:

$$\min_{\mathbf{W}, \mathbf{A}} f(\mathbf{W}, \mathbf{A}) + \frac{\rho}{2} |h(\mathbf{W})|^2 + \alpha h(\mathbf{W}),$$

where $\rho > 0$ is a penalty parameter and α is the Lagrangian multiplier.

This structural vector autoregressions model (SVAR) has made the i.i.d. assumption, assuming each sample is identically and independently distributed (Spirtes et al. 2000b). However, in real-world scenarios, samples may have associations and the data generation process may be affected by other samples. To take samples' interaction into account, Fan et al. (2023) conduct causal structure learning on the dynamic graph, where samples' interaction are presented in adjacency matrices. Then, the problem is formulated into the following structural equation model:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{A} \boxtimes \mathbf{Y}\mathbf{P} + \mathbf{E},$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a data matrix for n observation and p variables, $\mathbf{W} \in \mathbb{R}^{p \times p}$ represent weighted adjacency matrix of the DAG that characterizing the causal relationship of the p variables, $\mathbf{Y} = [\mathbf{Y}_1 | \dots | \mathbf{Y}_d] \in \mathbb{R}^{n \times pd}$ be concatenated matrix for the lagged data, and $\mathbf{P} = [\mathbf{P}_1^T | \dots | \mathbf{P}_d^T]^T \in \mathbb{R}^{pd \times p}$ denote the weights for the across-time dependencies, $\mathbf{A} = [\mathbf{A}_1 | \dots | \mathbf{A}_d] \in \mathbb{R}^{n \times nd}$ is the concatenated matrix for the time-lagged interaction data, d denotes the order of the autoregressive model, $\mathbf{A} \boxtimes \mathbf{Y} = [\mathbf{A}_1 \mathbf{Y}_1 | \dots | \mathbf{A}_d \mathbf{Y}_d] \in \mathbb{R}^{n \times pd}$ and $\mathbf{E} \in \mathbb{R}^{n \times p}$ is the noise matrix.

Since the future data won't affect the past observations, the edges in \mathbf{P} only go forward in time, which guarantees the acyclicity of the matrix \mathbf{P} . Thus, it suffices to require that \mathbf{W} is acyclic to ensure the whole causal network is acyclic. To ensure the acyclicity of the \mathbf{W} , the acyclicity constraint (Zheng et al. 2018) is imposed:

$$h(\mathbf{W}) = \text{trace}(e^{\mathbf{W} \circ \mathbf{W}} - p) = 0,$$

where \circ is the element-wise multiplication and $e^{\mathbf{W}}$ is the matrix exponential of \mathbf{W} . (Zheng et al. 2018)

Similar to NOTEARS, to estimate the weighted adjacency matrices \mathbf{W} and \mathbf{P} , given the data matrices \mathbf{X} , \mathbf{Y} and the interaction adjacency matrix \mathbf{A} , the GraphNOTEARS solves the following regularized optimization problem:

$$\begin{aligned} & \min_{\mathbf{W}, \mathbf{P}} f(\mathbf{W}, \mathbf{P}) \text{ s.t. } \mathbf{W} \text{ is acyclic} \\ & \text{with } f(\mathbf{W}, \mathbf{P}) = \ell(\mathbf{W}, \mathbf{P}) + \lambda_{\mathbf{W}} \|\mathbf{W}\|_1 + \lambda_{\mathbf{P}} \|\mathbf{P}\|_1, \\ & \ell(\mathbf{W}, \mathbf{P}) = \frac{1}{2n} \|\mathbf{X} - \mathbf{X}\mathbf{W} - \mathbf{A} \boxtimes \mathbf{Y}\mathbf{P}\|_F^2, \end{aligned}$$

where $\lambda_{\mathbf{W}}, \lambda_{\mathbf{A}}$ denotes the regularization weights for the weight matrices, $\|\cdot\|_1$ denotes the

element-wise ℓ_1 norm.

2.2.2 Time-lagged causal effects

Considering the data structure of repeated treatments and covariates, this line of literature often aims to estimate the treatment effect based on the structural nested mean model (SNMM) (Robins 1994).

Assume n independent subject's measurements are taken at time points t_0, t_1, \dots, t_K . Let A_k denote the binary treatment variable where 0 represents no treatment, X_k denote other covariate variables and Y_k denote the outcome variables measured at time t_k . By assuming the treatment at or after t_k won't affect the outcome at t_k , the SNMM models the effect of the "blip" of treatment variables, that is, holding all future treatments fixed at level 0 as follows (Vansteelandt and Joffe 2014):

$$\begin{aligned} & g(\mathbb{E}[Y_{k+1}^{(\bar{a}_k, \mathbf{0})} | \bar{X}_k = \bar{x}_k, \bar{Y}_k = \bar{y}_k, \bar{A}_k = \bar{a}_k]) - g(\mathbb{E}[Y_{k+1}^{(\bar{a}_{k-1}, \mathbf{0})} | \bar{X}_k = \bar{x}_k, \bar{Y}_k = \bar{y}_k, \bar{A}_k = \bar{a}_k]) \\ & = \gamma_k(\bar{x}_k, \bar{y}_k, \bar{a}_k; \psi), \end{aligned}$$

where overbar denotes the history of the respective variables, e.g. $\bar{A}_k = (A_1, \dots, A_k)$, $Y_k^{(\bar{a}_{k-1})}$ denote the potential outcome at time t_k if the individual had the treatment history \bar{a}_{k-1} , $(\bar{a}_k, \mathbf{0})$ denotes the counterfactual history as the history that agrees with a_k through time t_k and is 0 thereafter, $g(\cdot)$ denotes a known link function, and $\gamma_k(\cdot)$ denotes a known function that must satisfy that $\gamma_k(\bar{x}_k, \bar{a}_{k-1}, \mathbf{0}, \psi) = 0$.

Instead of conditioning on all the past variables, Boruvka et al. (2018) generalize the SNMM framework that only requires a subset of the past variables. Since the treatment's effect on the outcome variable may not be intermediate, the lag- m effect is proposed as:

$$\mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 1, a_{k+1}^{a_k=1}, \dots, a_{k+m-1}^{a_k=1})} | S_{mk}(\bar{A}_{k-1})] - \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 0, a_{k+1}^{a_k=0}, \dots, a_{k+m-1}^{a_k=0})} | S_{mk}(\bar{A}_{k-1})], \quad m = 1, 2, \dots,$$

where $a_{k+1}^{a_k=1}$ denotes the potential treatment at time $k+1$ had the treatments a_k been allocated at level 1. $S_{mk}(\bar{A}_{k-1})$ is a subset of the history variable $H_k(\bar{A}_{k-1})$, where

$$H_k(\bar{A}_{k-1}) = (X_1, A_1, Y_2^{(A_1)}, \dots, X_k^{(\bar{A}_{k-1})}, A_{k-1}, Y_k^{(\bar{A}_{k-1})}),$$

and $X_k^{(\bar{A}_{k-1})}$ denotes potential measurements X_k at time k had the sequence of treatments \bar{A}_{k-1} been allocated.

Under the positivity, consistency, and sequential ignorability assumptions (Boruvka et al.

2018), the lag- m effect could be simplified as follows if $S_{mk}(\bar{A}_{k-1}) = H_k = (\bar{X}_k, \bar{Y}_k, \bar{A}_{k-1})$:

$$\begin{aligned} & \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 1, a_{k+1}^{a_k=1}, \dots, a_{k+m-1}^{a_k=1})} | S_{mk}(\bar{A}_{k-1})] - \mathbb{E}[Y_{k+m}^{(\bar{a}_{k-1}, 0, a_{k+1}^{a_k=0}, \dots, a_{k+m-1}^{a_k=0})} | S_{mk}(\bar{A}_{k-1})] \\ &= \mathbb{E}[Y_{k+m} | A_k = 1, H_k] - \mathbb{E}[Y_{k+m} | A_k = 0, H_k]. \end{aligned}$$

2.3 Proposed method

2.3.1 Dynamic structural vector autoregression model

In Section 1.4.1, we extend the LSEM to allow the causal relations to vary with time, assuming that there are no temporal causal dependencies so that only same-time information would be causally related. However, this assumption may not hold in real-life, since there may exist time-lagged dependencies where the past observations could causally affect the current observations.

To take the time-lagged dependency into account, the structural vector autoregressive (SVAR) model (Demiralp and Hoover 2003) has been proposed:

$$\mathbf{X} = \mathbf{X}\mathbf{B} + \mathbf{Z}\mathbf{W} + \mathbf{E},$$

where $\mathbf{X} \in \mathbb{R}^{m \times p}$ is a data matrix for m observation and p variables, $\mathbf{B} \in \mathbb{R}^{p \times p}$ is an adjacency matrix of the DAG that characterizing the causal relationship of X , $\mathbf{Z} = [\mathbf{Z}_{(1)}, \dots, \mathbf{Z}_{(d)}] \in \mathbb{R}^{m \times p d}$ denotes the time-lagged data matrix, d denotes the order of the autoregressive model, $\mathbf{Z}_{(i)}$ denotes the lag- i data matrix, $\mathbf{W} \in \mathbb{R}^{p d \times p}$ denotes the weight matrix that characterizing the time-lagged causal dependencies and $\mathbf{E} \in \mathbb{R}^{m \times p}$ is the noise matrix.

Although taking into account the across-time causal relationships, the structural vector autoregressive model has assumed that the network structure is fixed across time. To relax this assumption to make the model more general and realistic, we extend the SVAR model to allow time-varying causal structure:

$$\mathbf{X}_t = \mathbf{X}_t \mathbf{B}_t + \mathbf{Z}_t \mathbf{W}_t + \mathbf{E}_t, \quad (2.3)$$

$$(2.4)$$

where $\mathbf{X}_t \in \mathbb{R}^{m \times p}$ is the data matrix for m observation and p variables at t^{th} time stamp, $\mathbf{B}_t \in \mathbb{R}^{p \times p}$ that characterizes the causal relationship of X_t , $\mathbf{Z}_t \in \mathbb{R}^{m \times p d}$ denotes the time-lagged data matrix for \mathbf{X}_t , d denotes the order of the autoregressive model, $\mathbf{W}_t \in \mathbb{R}^{p d \times p}$ denotes the

weight matrix that characterizing the time-lagged causal dependencies at t^{th} time stamp and $\mathbf{E}_t \in \mathbb{R}^{m \times p}$ is the noise matrix at t^{th} time stamp.

2.3.2 Basis approximation and constrained causal structural learning

Similar to Section 1.4.3, we could utilize the basis approximation approach to solve the curse of the dimensionality problem and obtain the estimates for the causal strength.

Applying basis approximation, Equation (2.4) could be written as

$$\begin{aligned} \mathbf{X}_t &= \mathbf{X}_t \mathbf{B}_t + \mathbf{Z}_t \mathbf{W}_t + \mathbf{E}_t, \\ &= \mathbf{D}_t \mathbf{\Gamma} + \mathbf{G}_t \mathbf{T} + \mathbf{E}_t, \text{ where } t = d + 1, \dots, T \end{aligned} \quad (2.5)$$

$$\mathbf{D}_t = \begin{bmatrix} F_1(t)X_{t11} & \dots & F_1(t)X_{t1p} & \dots & F_K(t)X_{t11} & \dots & F_K(t)X_{t1p} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ F_1(t)X_{tm1} & \dots & F_1(t)X_{tm p} & \dots & F_K(t)X_{tm1} & \dots & F_K(t)X_{tm p} \end{bmatrix}, \mathbf{G}_t = \begin{bmatrix} F_1(t)Z_{t11} & \dots & F_1(t)Z_{t1(pd)} & \dots & F_K(t)Z_{t11} & \dots & F_K(t)Z_{t1(pd)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ F_1(t)Z_{tm1} & \dots & F_1(t)Z_{tm(pd)} & \dots & F_K(t)Z_{tm1} & \dots & F_K(t)Z_{tm(pd)} \end{bmatrix},$$

$$\mathbf{\Gamma} = \begin{bmatrix} \gamma_{111} & \dots & \gamma_{1p1} \\ \vdots & \ddots & \vdots \\ \gamma_{p11} & \dots & \gamma_{pp1} \\ \vdots & \ddots & \vdots \\ \gamma_{11K} & \dots & \gamma_{1pK} \\ \vdots & \ddots & \vdots \\ \gamma_{p1K} & \dots & \gamma_{ppK} \end{bmatrix}, \text{ and } \mathbf{T} = \begin{bmatrix} \tau_{111} & \dots & \tau_{1p1} \\ \vdots & \ddots & \vdots \\ \tau_{(pd)11} & \dots & \tau_{(pd)p1} \\ \vdots & \ddots & \vdots \\ \tau_{11K} & \dots & \tau_{1pK} \\ \vdots & \ddots & \vdots \\ \tau_{(pd)1K} & \dots & \tau_{(pd)pK} \end{bmatrix}.$$

Stacking all the time points, we could have

$$\mathbf{X} = \mathbf{D}\mathbf{\Gamma} + \mathbf{G}\mathbf{T} + \mathbf{E},$$

$$\text{where } \mathbf{X} = \begin{bmatrix} \mathbf{X}_{d+1} \\ \vdots \\ \mathbf{X}_T \end{bmatrix}, \mathbf{D} = \begin{bmatrix} \mathbf{D}_{d+1} \\ \vdots \\ \mathbf{D}_T \end{bmatrix}, \mathbf{G} = \begin{bmatrix} \mathbf{G}_{d+1} \\ \vdots \\ \mathbf{G}_T \end{bmatrix} \text{ and } \mathbf{E} = \begin{bmatrix} \mathbf{E}_{d+1} \\ \vdots \\ \mathbf{E}_T \end{bmatrix}.$$

We could reformulate Equation (2.5) in a LSEM format:

$$\begin{aligned} [\mathbf{X} \quad \mathbf{D} \quad \mathbf{G}] &= [\mathbf{X} \quad \mathbf{D} \quad \mathbf{G}] \begin{bmatrix} \mathbf{0}_{p \times p} & \mathbf{0}_{p \times pK} & \mathbf{0}_{p \times pdK} \\ \mathbf{\Gamma} & \mathbf{0}_{pK \times pK} & \mathbf{0}_{pK \times pdK} \\ \mathbf{T} & \mathbf{0}_{pdK \times pK} & \mathbf{0}_{pdK \times pdK} \end{bmatrix} \\ &+ [\mathbf{E}, \mathbf{E}'], \end{aligned}$$

which implies that

$$[\mathbf{X} \ \mathbf{D} \ \mathbf{G}] = [\mathbf{E} \ \mathbf{E}'] \begin{bmatrix} \mathbf{I}_{p \times p} & \mathbf{0}_{p \times pK} & \mathbf{0}_{p \times pdK} \\ -\mathbf{\Gamma} & \mathbf{I}_{pK \times pK} & \mathbf{0}_{pK \times pdK} \\ -\mathbf{T} & \mathbf{0}_{pdK \times dK} & \mathbf{I}_{pdK \times pdK} \end{bmatrix}^{-1},$$

where $[\mathbf{E}, \mathbf{E}']$ is a noise matrix which are then treated as latent variables in the variational autoencoder and MLP layers are utilized in both encoder and decoder structure.

Since the future observations won't affect the past observations, there won't be edges pointing from the future data to the past data, which guarantees the acyclicity of \mathbf{W}_t . Thus it is sufficient to require \mathbf{B}_t to be acyclic to guarantee the acyclicity of the entire causal graph. Thus, we impose the acyclicity constraint as follows:

$$h_1(\mathbf{\Gamma}) = \sum_{t=1}^T \{ |tr[(\mathbf{I}_p + \alpha \mathbf{B}_t \odot \mathbf{B}_t)^p] - p| \} = 0,$$

where $tr(\cdot)$ represents the trace of the matrix, \odot is the element-wise multiplication, \mathbf{B}_t could be written as a function of $\mathbf{\Gamma}$ as in Equation (1.10), p is the number of variables, T is the number of time-stamps and α is a hyperparameter.

Also, as required by the assumptions in Section 2.4.1, the treatment won't be affected by other variables while the outcome variable won't affect the other variables in the same time stamp, the identifiable constraint could be written as:

$$h_2^*(\mathbf{\Gamma}, \mathbf{T}) = \sum_{i=1}^{pK} |\Gamma_{i,0}| + \sum_{i=1}^K \sum_{j=1}^p |\Gamma_{i^*p,j}| + \sum_{i=1}^{pdK} |T_{i,0}| = 0,$$

where $\Gamma_{i,j}, T_{i,j}$ are the $(i, j)^{th}$ element for matrix $\mathbf{\Gamma}, \mathbf{T}$ respectively.

Similar to Section 1.4.4, we could obtain the estimated causal graph by solving

$$\begin{cases} \min_{\mathbf{\Gamma}, \mathbf{T}} -L_{ELBO} & = -\frac{1}{p} \sum_{i=1}^p [-D_{KL}\{q(\mathbf{e}_i|\mathbf{x}_i)\|p(\mathbf{e}_i)\} \\ & \quad + \mathbb{E}_{q(\mathbf{e}_i|\mathbf{x}_i)}\{\log p(\mathbf{x}_i|\mathbf{e}_i)\}] \\ \text{subject to} & h_1(\mathbf{\Gamma}) = 0 \text{ and } h_2^*(\mathbf{\Gamma}, \mathbf{T}) = 0, \end{cases}$$

where D_{KL} represents the KL divergence, $q(\mathbf{e}_i|\mathbf{x}_i)$ denotes the variational posterior of \mathbf{e}_i , $p(\mathbf{e}_i)$ denotes the prior distribution of e_i , $\log p(\mathbf{x}_i|\mathbf{e}_i)$ denotes the log-likelihood function, \mathbf{e}_i denotes the i^{th} latent variable, and \mathbf{x}_i denotes the i^{th} data vector.

2.4 Dynamic causal effect estimation

2.4.1 Assumptions

As commonly-used in the causal discovery literature (Spirtes et al. 2000a; Cai et al. 2020), in this work, we have made the following assumptions.

Let overbar denotes the history of the respective variables, e.g. $\bar{A}_t = (A_1, \dots, A_t)$, $Y_t(\bar{a}_{t-1})$ denote the potential outcome at time t if the individual had the treatment history \bar{a}_{t-1} , and H_t denotes the history up to time t , $H_t = (\bar{M}_t, \bar{Y}_t, \bar{A}_t)$.

Causal sufficiency A set \mathcal{V} of variables is causally sufficient for a population if and only if in the population every common cause of any two or more variables in \mathcal{V} is in \mathcal{V} , or has the same value for all units in the population.

Causal Markov condition Each vertex is independent of its non-descendants in the graph conditional on its parents in the graph. In other words, we have

$$P(v_1, v_2, \dots, v_n) = \prod_{i=1}^p P(v_i | pa(i)), \quad (2.6)$$

for vertices $\{v_1, \dots, v_p\} \in \mathcal{V}$, where $pa(i)$ are the parent nodes for v_i and $P(\cdot)$ is a probability function .

Consistency: The observed data are equal to the potential outcomes as follows:

$$Y_t = Y_t(\bar{A}_t), M_t = M_t(\bar{A}_t), A_t = A_t(\bar{A}_t), \text{ for each } t \leq T$$

Sequential ignorability: For each $t \leq T$, the potential outcome $Y_t = Y_t(\bar{A}_t)$, are independent of A_t conditional on H_t .

Based on the sequential ignorability assumption, the underlying treatment probabilities $p_t(1|H_t)$, $t = 1, \dots, T$, are some unknown constants and thus the treatment variable A_t would have no parent node. Also, based on the consistency assumption, the outcome variable won't affect mediator variables at the same time-stamp.

2.4.2 Dynamic causal effect

In this section, we propose the dynamic causal effect based on the dynamic structural vector autoregression (dynamic SVAR) model, assuming the causal sufficiency, causal faithfulness, consistency, and sequential ignorability assumptions. ²

Without loss of generality, we could decompose the data matrix X_t into the treatment variable, A_t , mediator variable M_t and the outcome variable Y_t , $X_t = [A_t, M_t^T, Y_t]$ and then

²Note that the dynamic SVAR model would reduce to the dynamic LSEM model when the order $p = 0$, so that the dynamic causal effect could be applied in both scenarios.

Equation (2.4) could be written as:

$$\begin{aligned}
& [A_t, M_t^T, Y_t] \\
&= [A_t, M_t^T, Y_t] \mathbf{B}_t + \mathbf{E}_t \\
&\quad + [A_{t-1}, M_{t-1}^T, Y_{t-1}, \dots, A_{t-d}, M_{t-d}^T, Y_{t-d}] \mathbf{W}_t \\
&= [A_t, M_t^T, Y_t] \begin{bmatrix} 0 & \boldsymbol{\alpha}_t & \gamma_t \\ \mathbf{0}_{(p-2) \times 1} & \mathbf{C}_t & \boldsymbol{\beta}_t \\ 0 & \mathbf{0}_{1 \times (p-2)} & 0 \end{bmatrix} + \mathbf{E}_t \\
&\quad + \sum_{i=1}^d [A_{t-i}, M_{t-i}^T, Y_{t-i}] \begin{bmatrix} 0 & \boldsymbol{\alpha}_{t-i} & \gamma_{t-i} \\ \mathbf{0}_{(p-2) \times 1} & \mathbf{C}_{t-i} & \boldsymbol{\beta}_{t-i} \\ 0 & \mathbf{d}_{t-i} & f_{t-i} \end{bmatrix},
\end{aligned} \tag{2.7}$$

where $\gamma_t, f_{t-1}, \gamma_{t-i}$ are scalars, $\boldsymbol{\alpha}_t^T, \boldsymbol{\beta}_t, \mathbf{0}_{(p-2) \times 1}, \mathbf{d}_{t-i}^T$ are $(p-2) \times 1$ vectors, \mathbf{C}_t is a $(p-2) \times (p-2)$ matrix, and $\mathbf{E}_t = [\epsilon_{A_t}, \epsilon_{M_t}, \epsilon_{Y_t}]$ is the error matrix. Based on the assumption, we could know that the treatment variable has no parent node and the outcome variable has no child node at the same time-stamp, which results in the zeros in the weight matrix.

Based on the literature (Boruvka et al. 2018; Wu et al. 2020a; Robins 1994; VanderWeele and Robins 2007), we propose the dynamic causal effect as follows:

Theorem 2.4.1. Under assumptions in Section 2.4.1 and Equation (2.7), we could have the following dynamic causal effect:

$$\begin{aligned}
& \mathbb{E}[Y_{t+1}(\bar{a}_t, a) | H_t] - \mathbb{E}[Y_{t+1}(\bar{a}_t, 0) | H_t] \\
&= \mathbb{E}[Y_{t+1}(\bar{a}_t, a) | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t] - \\
&\quad \mathbb{E}[Y_{t+1}(\bar{a}_t, 0) | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t] \\
&= (\gamma_{t+1} + \boldsymbol{\beta}_{t+1}^T (\mathbf{I} - \mathbf{C}_{t+1}^T)^{-1} \boldsymbol{\alpha}_{t+1}^T) a,
\end{aligned}$$

where overbar denotes the history of the respective variables, e.g. $\bar{A}_t = (A_1, \dots, A_t)$, $Y_t(\bar{a}_{t-1})$ denote the potential outcome at time t if the individual had the treatment history \bar{a}_{t-1} , a is a treatment value and H_t denotes the history up to time t , $H_t = (\bar{M}_t, \bar{Y}_t, \bar{A}_t)$.

The proof could be found in the Appendix Chapter B.

2.5 Simulation analysis

2.5.1 Simulation setup

The data are generated based on the dynamic SVAR setup specified in Equation (2.4) with Gaussian errors, where each graph has 5 variables ($p = 5$) and 30 observations ($m = 30$). We consider 10 time stamps ($T = 10$) and generate 30 realizations. The order of the autoregressive model is $d = 1$ and the time-lagged weight matrix $\mathbf{W}_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \forall t$. The contemporaneous weight matrix \mathbf{B}_t has been designed with only one edge ($A \rightarrow Y$), whose weight changes with time as the following two different functions:

- (F1) Cosine function: $f_1(t) = \cos(\frac{t}{4\pi}) * 0.8$;
- (F2) Quadratic function: $f_2(t) = \frac{-15+(5-t)^2}{25}$.

2.5.2 Evaluation metric and benchmark methods

To evaluate the performance of the proposed method, we utilize the proposed method to identify the hidden causal graph and compare it with the commonly used causal discovery approaches: DAGGNN (Yu et al. 2019), NOTEARS (Zheng et al. 2018), DYNOTEARS Pamfil et al. (2020) and ANOCE (Cai et al. 2020). Since the benchmark approaches are developed for the static causal graph, for each timestamp, we generate estimates for the causal graph using the current timestamp’s data and use the most-recent estimate as the one-step ahead prediction. We use all-time data to generate historical estimates and the future predictions using the proposed method.

We evaluate the graph estimates using the following three metrics: false discovery rate (FDR), true positive rate (TPR), and structural Hamming distance (SHD). To evaluate the time-varying strength of causal relations, we use mean squared error (MSE) as an evaluation metric. It is the higher the better for the TPR, while for the other three metrics, a lower value indicates a better result. As commonly used in the causal discovery literature (Cai et al. 2020; Zheng et al. 2018), we remove the edges if the edge weight is lower than 0.2 to reduce noise.

2.5.3 Implementation details

In this section, we present the implementation details for the proposed method and the benchmark methods as follows:

- Proposed method: The proposed method is implemented based on PyTorch (Paszke et al. 2017) using Adam (Kingma and Ba 2014) optimizer to optimize the loss function.

The batch size is set to be 10, training epoch is 200, and the number of hidden nodes is the square of the number of variables. The maximum iteration number for searching parameters is 100, the initial learning rate is 0.003 which decays with a decay rate of 1.0. The degree of the autoregressive model is set to be 1.

- ANOCE (Cai et al. 2020): ANOCE provides a decomposition of the indirect effect, to categorize interactions between mediators. We implement the ANOCE with the default hyper-parameters to estimate the causal graph at each time-stamp using each-times' data. Their code is available at the repository <https://github.com/hengruicai/ANOCE-CVAE>.
- NOTEARS (Zheng et al. 2018) convert the combinatorial optimization problem into a continuous optimization problem by setting the acyclicity constraint as a function of the adjacency matrix. We implement the NOTEARS with the default hyper-parameters to estimate the causal graph at each time-stamp using each-times' data. Their code is available at the repository <https://github.com/xunzheng/notears>.
- DYNOTEARS (Pamfil et al. 2020) extend the NOTEARS model to handle the scenario where the replicates are no longer independent but have interactions. We implement the DYNOTEARS with the default hyper-parameters to estimate the causal graph at each time-stamp using lag-1 data. Their code is available at the repository <https://github.com/mckinsey/causalnex/blob/develop/causalnex/structure/dynotears.py>.

2.5.4 Results

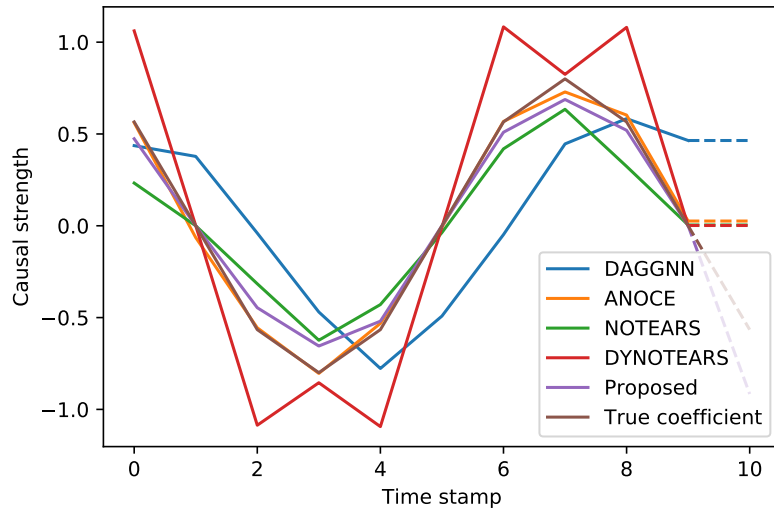
The results for the proposed method and the benchmark methods are shown in Figures 2.1, 2.2 and Table 2.1, 2.2. The findings are summarized as follows.

Table 2.1: The empirical comparison of estimated contemporaneous weight matrix \mathbf{B}_t in the dynamic SVAR model setup. The number in parenthesis denotes the standard deviation.

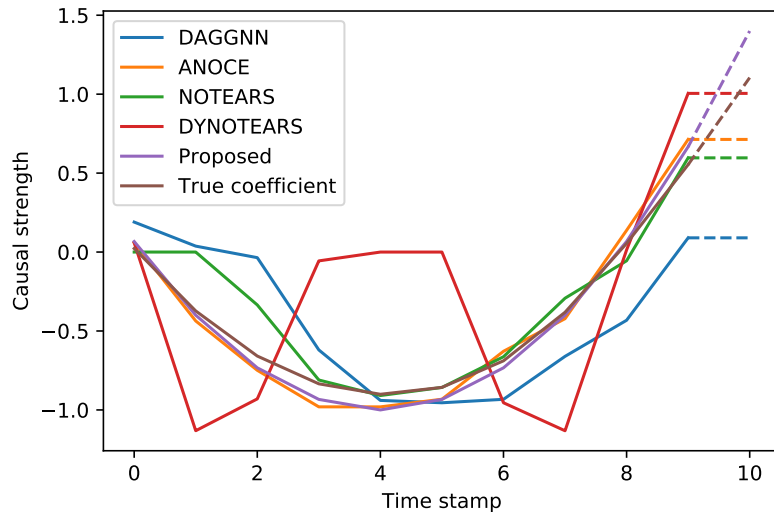
Methods	Metric	F1	F2
DAGGNN	FDR	0.79(0.02)	0.80(0.01)
	TPR	0.83(0.02)	0.76(0.02)
	SHD	3.05(0.13)	3.09(0.10)
	MSE	0.24(0.02)	0.24(0.02)
ANOCE	FDR	0.77(0.01)	0.77(0.01)
	TPR	0.82(0.02)	0.83(0.02)
	SHD	2.53(0.10)	2.61(0.10)
	MSE	0.07(0.01),	0.10 (0.01)
NOTEARS	FDR	0.36(0.03)	0.39(0.03)
	TPR	0.61(0.02)	0.7(0.02)
	SHD	0.82(0.06)	0.80(0.06)
	MSE	0.10(0.01)	0.13(0.01)
DYNOTEARS	FDR	0.05 (0.01)	0.31(0.01)
	TPR	0.98(0.01)	0.70(0.01)
	SHD	0.05(0.01)	0.32(0.01)
	MSE	0.15(0.00)	0.40(0.01)
Proposed	FDR	0.13 (0.02)	0.05 (0.01)
	TPR	0.98(0.00)	0.98(0.01)
	SHD	0.32(0.05)	0.05(0.01)
	MSE	0.02(0.01)	0.02 (0.02)

Table 2.2: The empirical comparison of estimated time-lagged weight matrix \mathbf{W}_t in the dynamic SVAR model setup. The number in parenthesis denotes the standard deviation.

Methods	Metric	F1	F2
DYNOTEARS	FDR	0.40 (0.01)	0.36(0.01)
	TPR	0.27(0.01)	0.27(0.01)
	SHD	3.64(0.02)	3.75(0.02)
Proposed	FDR	0.28(0.02)	0.26(0.01)
	TPR	0.95(0.03)	0.98(0.00)
	SHD	2.20(0.01)	1.93(0.08)

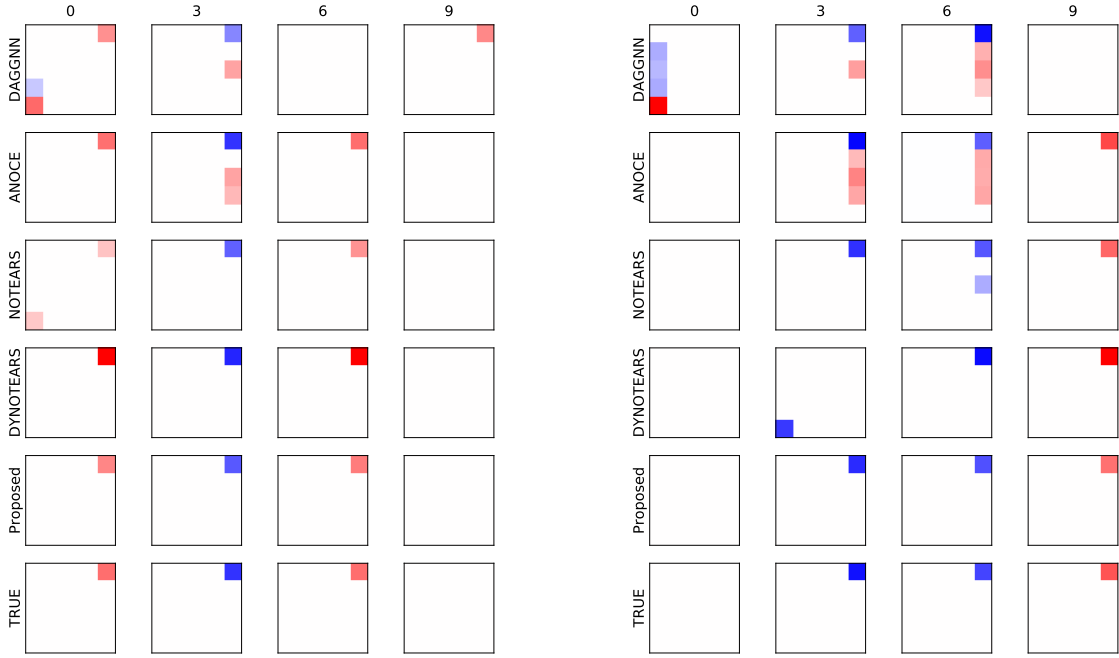


(a) Cosine function F1



(b) Quadratic function F2

Figure 2.1: Estimated causal strength using the proposed method and benchmark methods in the dynamic SVAR model setup. The solid line denotes the estimated coefficient in the ten time stamps and the dashed line denotes the one-step ahead prediction.



(a) Cosine function F1

(b) Quadratic function F2

Figure 2.2: Estimated causal graphs at multiple time-stamps using the proposed method and the benchmark based on the dynamic SVAR model setup.

As shown in Figure 2.1, the proposed method could generate more accurate estimates for the causal strength in both cosine and quadratic function setups. Moreover, the proposed method is able to obtain an accurate prediction for the next time stamp, while the other benchmark methods fail to do so.

Also, as shown in Table 2.1, we could conclude that the proposed method could also capture the hidden graph structure since the proposed method obtains better results in terms of FDR, SHD, and MSE in most of the scenarios while having comparable TPR with benchmark methods. The estimated graph, i.e., Figure 2.2 also shows that the proposed method could estimate the underlying causal graph better.

Table 2.2 presents the metrics for the estimated time-lagged weight matrix in the dynamic SVAR model setup. Since both ANOCE and NOTEARS are not capable to capture the time-lagged dependency, we compare the performance of the proposed method with DYNOTEARS. Also, since \mathbf{W}_t is not time-varying in the simulation, we omit the MSE metric in the table. As

shown in Table 2.2, our method outperforms the DYNOTEARS for all three metrics in both scenarios.

2.6 Real data analysis

Starting in 2019, the coronavirus disease (COVID-19) has spread globally and caused many human lives to be lost. There are many literature studying the factors that affect the COVID-19 transmission, such as policy restriction (Chinazzi et al. 2020) and people’s awareness (Li et al. 2020a). However, the effects of these factors may be time-varying. For instance, the contact restriction policy may be implemented for several months, but its effect may decrease since people may not obey the rules in the later stage. In this section, we aim to learn a dynamic causal graph based on the COVID-19 data to study the effects of policy intervention on the COVID-19 cases.

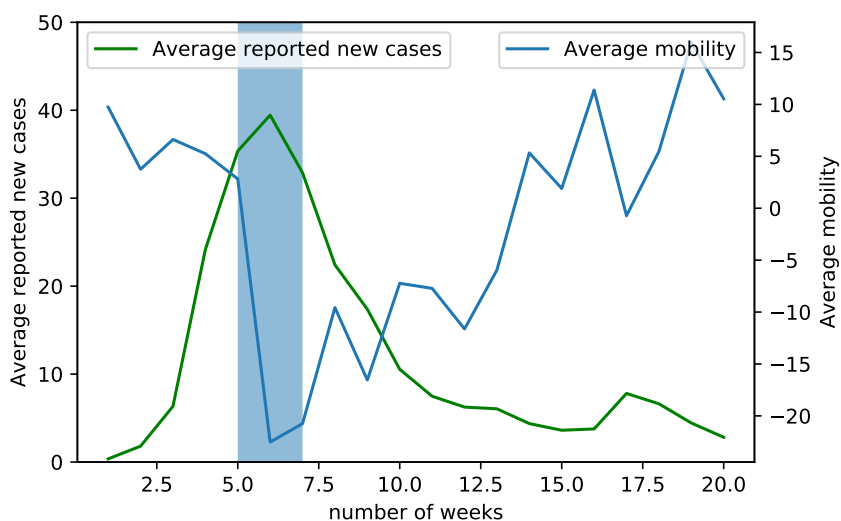


Figure 2.3: Average reported new cases and mobility with respect to time. The shaded region represents the time period that the contact restriction policy started to implement.

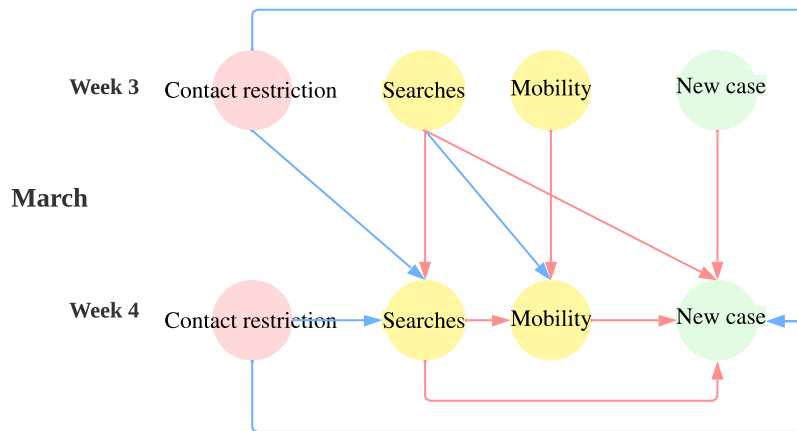
We collect weekly aggregated data on 27 districts in Germany from February 15 to July 8, 2020 (Steiger et al. 2021). Summarized by Steiger et al. (2021), the possible factors for COVID-19 could be categorized into five categories: mobility, awareness, weather, intervention, and socio-demographic factors. Since weather and socio-demographic factors will not be influenced by the policy intervention, we don’t include these factors and select contact restriction policy as

the treatment variable, average mobility and searches for corona as the mediator variables, and reported new cases as the outcome variable. The dataset has 4 variables ($p = 4$), 27 observations ($m = 27$) and 20 time stamps ($T = 20$). As shown in Figure 2.3, after the contact restriction policy beginning to implement, people tend to travel less since the mobility variable has a sudden drop. Also, the average reported new cases shows a decreasing trend after the policy begins to implement.

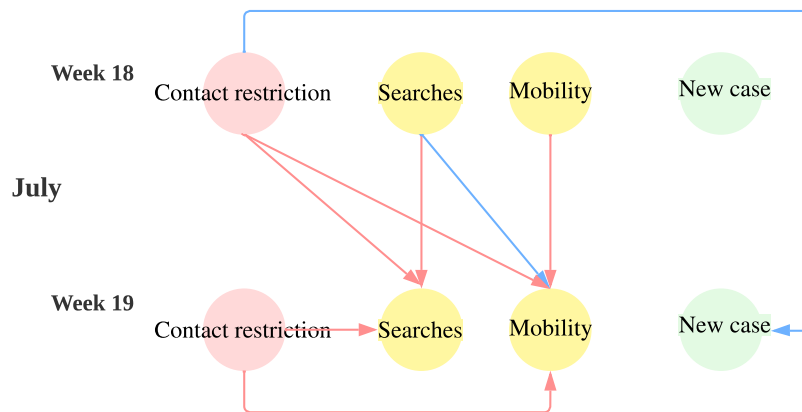
We apply the two proposed methods to the real data and compare their ELBO loss. Since it is shown in the literature that these factors may have a lagged effect of 5 days Steiger et al. (2021), the proposed method based on autoregressive model is applied with order $d = 1$. This proposed model with time-lagged dependency has a smaller ELBO loss (0.02) compared to that of the model without time-lagged dependency (0.06), which also matches with the literature.³ Thus, we present the results of the proposed method that is based on autoregressive model, since it has better performance and more intuitive.

Figure 2.4 presents the estimated causal graph at March and July. We could see that in both graphs, the contact restriction policy could reduce the reported new cases for the next week and the mediator variables, i.e., average mobility and searches for corona, could positively affect themselves in the next week. We could also notice that, in March, implementing the contact restriction policy could indirectly reduce mobility as shown in Figure 2.4a. But in July, this policy doesn't have negative effects on mobility and would increase mobility as in Figure 2.4b, which may indicate that the policy may not properly be implemented.

³This proposed model with time-lagged dependency has the log-likelihood loss of 0.002 and MSE of $8.06 * 10^{-6}$.



(a) Estimated causal graph at March



(b) Estimated causal graph at July

Figure 2.4: Estimated causal graph at March and July. Each node represents a variable and the arrows represent the discovered causal relations. Red color represents positive causal relations and blue color represents negative causal relations.

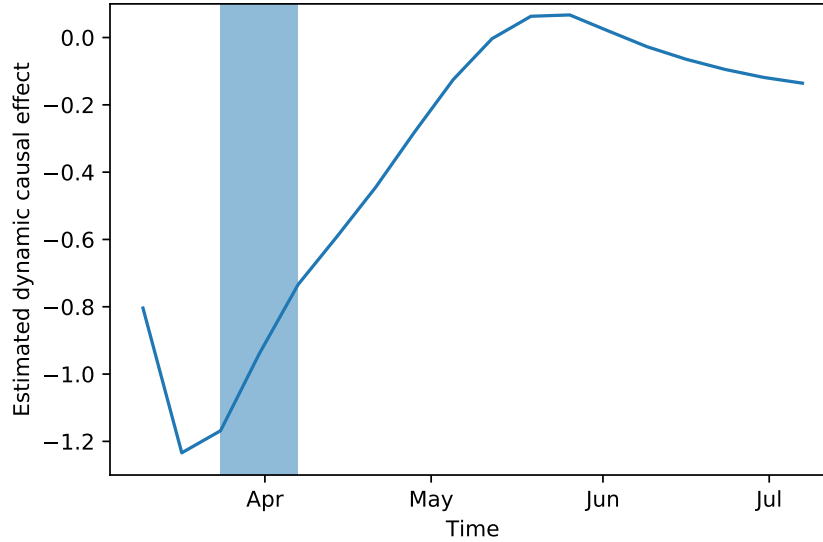


Figure 2.5: Estimated dynamic causal effect of the contact restriction policy on reported new cases. The shaded region represents the time period when the policy starts to implement.

Figure 2.5 presents the estimated dynamic causal effect of the contact restriction policy on the reported new cases. We could see that implementing this contact restriction policy could reduce the spread of COVID-19 since the estimated effect is negative. But this effect is diminishing as time goes, since the actual policy implementation may become worse. Also, Figure 2.5 suggests that it may be better for the government to implement the contact restriction policy earlier to reduce the spread, since the estimated dynamic causal effect is larger in March.

2.7 Discussion

In this chapter, we develop a new framework to model the dynamic causal graph where causal relations are allowed to be both time-varying and time-lagged. We propose a score-based dynamic causal structure discovery approach and propose an algorithm that could provide estimates on both dynamic casual graphs and dynamic causal effects. To conclude, we briefly discuss some limitations and possible future directions.

Firstly, we apply the basis approximation method to estimate the time-varying casual strength and may get less accurate results when the true relationship is not continuous. This issue may be resolved using more basis, but it may increase computation complexity.

Secondly, the basis approximation method may lead to estimation error since the number

of basis is finite in the implementation. It may be better to incorporate the basis estimation bias into the loss function to generalize the method to datasets with long time spans.

Thirdly, background information is utilized to determine the order of the autoregressive model, which may not be available in some scenarios. A possible approach is to utilize the cross-validation method to determine the appropriate order, but it may also increase the computational time.

CHAPTER

3

REVIEW ON GRAPH NEURAL NETWORK METHODS ON FINANCIAL APPLICATION

3.1 Introduction

As the data collection techniques grow, graph data are commonly collected in many areas including social sciences, transportation systems, chemistry, and physics (Wu et al. 2020b). Representing complex relational data, graph data contain both the individual node information and the structural information. Recently, there are growing interests in developing machine-learning methods to model the graph data of various domains. Among them, graph neural network (GNN) methods could achieve great performance on various tasks, including node classification, edge prediction, and graph classification (Kipf and Welling 2017; Zhang and Chen 2018; Xu et al. 2018). Performing node aggregation and updates, graph neural network models extend the deep learning methodology to graphs and are gaining popularity.

A financial system is a complex system with many components and sophisticated relations, which may be frequently updated. To represent the relational data in the financial domain, graphs are commonly constructed, including the transaction network (Weber et al. 2019), user-item review graph (Dou et al. 2020), and stock relation graph (Feng et al. 2019). By converting the financial task into a node classification task, GNN methods are commonly utilized since

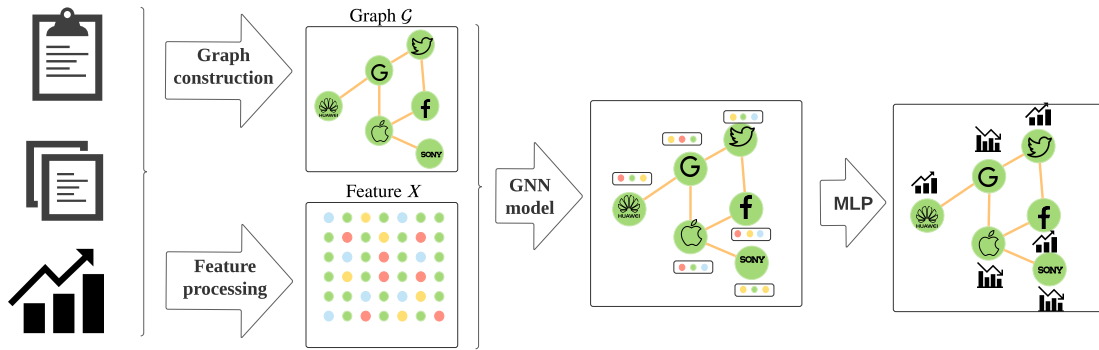


Figure 3.1: Workflow for stock movement prediction task using GNN methodology. The graph construction and feature processing steps present stock information in a graph and a feature matrix, which is then used as the input for the GNN model. In the graph, nodes are connected if there exist some relationships between stocks, such as supplier, competitor, shake-holder, etc. A multi-layer perceptron layer (MLP) is used to output the price prediction result.

it performs well among graph modeling methods (Liu et al. 2019). For instance, GNN could be utilized in a stock prediction task, by formulating it as a node classification task, where each node represents a stock and edges represent relations between companies. Figure 3.1 demonstrates the workflow of a stock prediction task using GNN methods. However, the complex nature of financial systems may result in multiple data sources and complicated graph structures, which imposes challenges on feature processing, graph construction, and graph neural network modeling. Represented as numerical sequences or textual information, financial data need to be processed with caution to keep the temporal pattern or semantic meanings. Also, the multi-facet nature of financial relations make it hard to construct a graph to capture the relations. Moreover, the financial-related graph is often heterogeneous or time-varying, which impose challenges on existing graph neural network models. What’s more, to reflect some financial patterns (e.g. device aggregation pattern, see Section 3.5 for details), GNN methods may need to be modified such as changing losses and adding additional layers. Since financial systems process unique characteristics and receive great attention, it is of significant importance to discuss and summary the GNN methodology developed for financial tasks.

There are several recent reviews on graph neural networks. Among them, Wu et al. (2020b) present a comprehensive review on graph neural networks and categorize the GNNs into four categories: recurrent graph neural networks, convolutional graph neural networks, graph auto-encoders, and spatial-temporal graph neural networks. Zhou et al. (2020) provide a taxonomy on GNN models based on graph type, training methods, and propagation steps. There is also literature focusing on limited types of GNNs. Zhang et al. (2019) focus on graph convolutional

networks (GCN) and introduce two taxonomies to group the existing GCNs. Lee et al. (2019) survey the literature on graph attention models and provided detailed examples on each type of method. However, the aforementioned reviews focus on the general methodology and provide little details for applications, seldom mentioning the financial application. Without covering GNN models developed based on financial contexts, the reviewed models may not be applicable to financial tasks due to the complexity of financial data. On the other side, review papers focusing on the financial domain haven't covered GNN methodologies in detail yet. Ozbayoglu et al. (2020) summarize the machine learning and deep learning models in the financial field, without mentioning the GNN methodologies. Huang et al. (2020) survey the financial deep learning models in the finance and bank industry, and the GNN models are not covered. Jiang (2021) review stock prediction-related machine-learning mythologies and mention GNN models very briefly. In summary, existing GNN surveys focus on modeling methodology and do not emphasize the financial application of GNN methods, while surveys on financial applications don't cover the GNN models in detail. To fill the gap, in this survey, we provide a systematic and comprehensive review of graph neural network methods in the financial application.

In this chapter, we present a thorough survey on graph neural network models with financial application. We provide a comprehensive review of graph neural networks and summarize the corresponding methods. This survey has contributions as follows.

- We systemically categorize the commonly-used financial graphs based on graph characteristics and provide a thorough list of graphs. Graphs are categorized into five groups: homogeneous graph, directed graph, bipartite graph, multi-relation graph, and dynamic graph. We also present the GNN models according to their graph types, so that this review could serve as a guide for implementing GNNs on real-life datasets.
- We provide a comprehensive list of financial applications that GNN methods are applied. We categorized the applications into five categories: stock movement prediction, loan default risk prediction, recommender system of e-commerce, fraud detection, and event prediction.
- We summarize various aspects of information for each application, including features, graphs, GNN models, and available codes. A GitHub¹ page is built to document the collection of information. This work could be considered as a resource to understand, implement and develop GNN models on multiple financial tasks.

¹Github link: <https://github.com/jackieD14/Graph-models-in-finance-application>

- We identify five challenges and discuss the recent progress. We also suggest future directions for these problems.

The rest of the paper is organized as follows. Section 3.2 classifies financial graphs into different categories based on its characteristics. Section 3.3 summarizes the commonly-used feature processing techniques for each node in the graph. Section 3.4 presents the GNN methodology used for each graph type. Section 3.5 provides a collection of application areas. Section 3.6 proposes some challenges that could be future directions of research.

3.2 Graph categorization

When preparing the data, how to construct the graph to represent the structural information is essential and the type for the constructed graph could determine the follow-up modeling methodology. In this section, we present the categorization of the graph based on its construction methods and graph types. Table 3.1 presents a comprehensive list of graphs for financial tasks.

3.2.1 Graph-related definition

In this section, we provide some graph-related definitions for better understanding of this article.

Definition 3.2.1 (Graph). A graph \mathcal{G} is defined by a pair: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ 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 joining node v_i and node v_j .

Definition 3.2.2 (Adjacency matrix). An adjacency matrix \mathbf{A} is an $n \times n$ matrix, where \mathbf{A}_{ij} represents the connection status between node v_i and node v_j . For an unweighted graph, the adjacency matrix could be a binary matrix where $\mathbf{A}_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and 0 otherwise.

Definition 3.2.3 (Undirected graph and Directed graph). A undirected graph is a graph where the edges are undirected. A directed graph is a graph where the edges have orientations. $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge pointing from node v_i to node v_j .

Remark. Undirected graph has a symmetric adjacency matrix, i.e., $\mathbf{A}_{ij} = \mathbf{A}_{ji}$.

Definition 3.2.4 (Bipartite graph). A Bipartite graph is a graph whose nodes could be divided into two non-empty and disjoint sets \mathcal{U}, \mathcal{W} , such that every edge connects a node in \mathcal{U} and a node in \mathcal{W} .

Definition 3.2.5 (Homogeneous graph and Heterogeneous graph). In a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we can assign a type to each node and edge; in this case, the graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where each node $v_i \in \mathcal{V}$ is associated with its type $a_i \in \mathcal{A}$, and each edge $e_{ij} \in \mathcal{E}$ is associated with its type $r_{ij} \in \mathcal{R}$. A homogeneous graph is a graph whose nodes are of the same type and edges are of the same type. Otherwise, the graph is heterogeneous.

Definition 3.2.6 (Multi-relation graph). A Multi-relation graph is a graph where edges have different types.

Definition 3.2.7 (Dynamic graph). A dynamic graph is defined as a sequence of graphs $\mathcal{G}^{seq} = \{\mathcal{G}_1, \dots, \mathcal{G}_T\}$, where $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, for $i = 1, \dots, T$, where $\mathcal{V}_i, \mathcal{E}_i$ are the set of nodes and edges for i^{th} graph in the sequence respectively.

3.2.2 Graph categorization by construction methods

In this section, we elaborate on frequently-used graph construction methods so that researchers could better understand, choose and construct graph data.

Data-based construction

Some types of data could be naturally represented as a graph since they contain relations among data objects. For instance, Wang et al. (2019) construct a user-relationship graph, where users are linked by an edge if they are labeled classmates, friends, or workmates in the data. Liu et al. (2018) construct an account-device network, where nodes are either accounts or devices. Edges connect an account node to a device node if the account has activities on the device. This type of construction method is based on the nature of data and could be used on data where relations are clearly defined.

Knowledge-based construction

Sometimes, data may not contain relational information, but relations could be found in knowledge bases. A knowledge base is a collection of descriptive data and contains numerous entities and their relations. For example, Wikidata (Vrandečić and Krötzsch 2014) is one of the largest open-domain knowledge bases which provides support for Wikipedia, an online encyclopedia. A graph could be built utilizing the relations extracted from knowledge bases. In order to predict stock movement, Feng et al. (2019) extract company relations from Wikidata, such as supplier, provider, partner, etc, and constructed a company-based relation network. This type of graph-construction method brings new information into the graph by utilizing the knowledge bases, which may improve modeling performance. However, it may take some

effort to process the complicated data structure and the massive amount of information of knowledge bases.

Similarity-based construction

There also exist cases that neither the data nor knowledge bases contain relations, but there may be some hidden relationships in the data. To mine the underlying relationship, a commonly-used approach is to calculate a similarity measure of the features for different observations and construct relations if the similarity value is greater than a threshold. For instance, Li et al. (2020b) construct a stock correlation graph based on the cosine similarity of stocks' historic market price. Two stocks are then connected if the absolute value of their correlation is larger than a threshold. This type of construction method could be easily implemented and understood. However, in this type of construction, feature information is represented both in the graph adjacency matrix and the feature matrix. The overlapping information may lead to doubts that whether the graph representation is still necessary. Also, how to set the threshold is an issue, and justification may be needed for the selected similarity threshold value.

3.2.3 Graph categorization by graph types

In this section, we categorize graphs into five categories based on their characteristics and provide examples in the financial context. Since different types of graphs may impose various challenges on modeling technology, we discuss GNN methods for each graph type in section 3.4 to provide solutions respectively. Figure 3.2 provides visualization for each graph type: homogeneous graph, directed graph, bipartite graph, multi-relation graph, and dynamic graph. It is also worth mentioning that a graph may be categorized into multiple types.

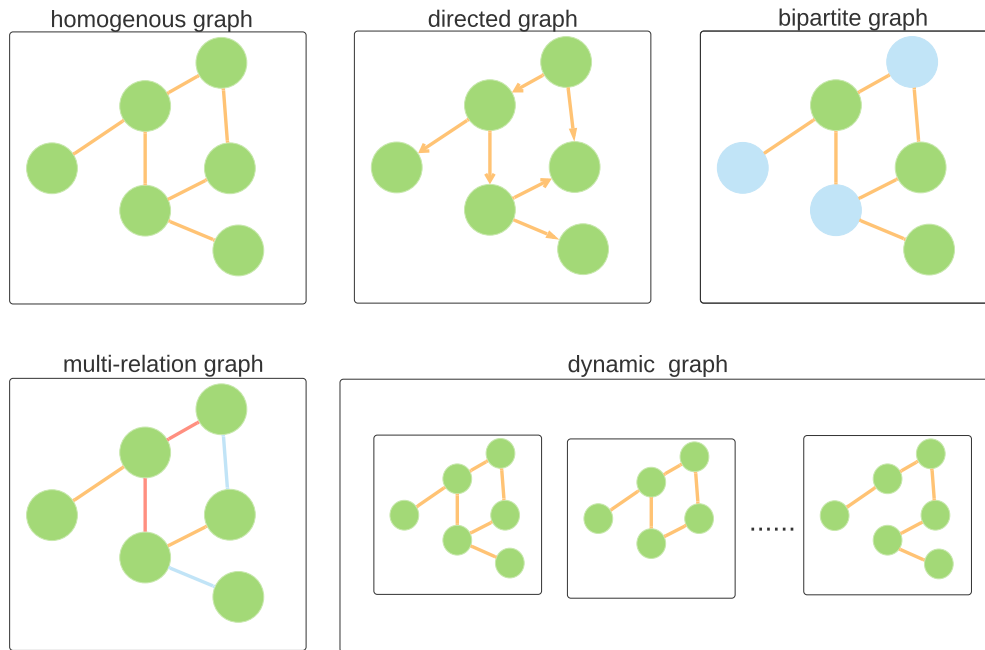


Figure 3.2: Graph categorization based on graph characteristics. Each color of the circle represents a node type and each color of the line represents an edge type. Arrows represent directed edges. A homogeneous graph is a graph with one type of node and one type of edge. A directed graph is a graph with directed edges. A bipartite graph is a graph with two types of nodes and edges only exist between nodes of different types. A multi-relation graph has edges with different types. A dynamic graph is a sequence of graphs.

Homogeneous graph

A homogeneous graph is a graph with one type of node and one type of edge. For instance, Liou et al. (2021) construct a financial news co-occurrence graph, where two companies are connected if they are tagged in the same news articles. Li et al. (2020c) build a transaction network, where nodes are accounts and are connected when there exist transactions between them. This type of graph has a relatively simple structure and the majority of the GNN methods could be applied to model this type of graph.

Directed graph

A directed graph is a graph where edges have orientations. For instance, Cheng et al. (2019) construct a guarantee network where nodes are the companies and edges represent the guarantee

relationship. Since the guarantor has the obligation to pay the debt for the borrower, but not the other way round, this type of guarantee relationship is one-sided and could be represented in a directed edge. In general, a directed graph may have an asymmetric adjacency matrix and thus cannot be semi-definite. Since some GNN methodologies are developed for semi-definite adjacency matrices, they may not be suitable for the directed graph.

Bipartite graph

A bipartite graph is a graph with two types of nodes and edges only exist between nodes of different types. For instance, Liu et al. (2018) construct an account-device network in a fraud detection task. Nodes could be either accounts or devices, with edges connecting them if the account has activities on the device. Li et al. (2019a) extract a user-item network using the rating data, where nodes could be either a user or an item. An edge exists if a user has rated the item. A bipartite graph is commonly used, when data can be divided into two groups and the interaction between two groups matters. A bipartite graph could be seen as a specific case of a multi-relation graph discussed in the next session and GNN methods developed for the multi-relation graph are applicable on a bipartite graph as well.

Multi-relation graph

Sometimes, edges may have multiple types to represent different relations between nodes. For instance, Wang et al. (2019) construct a user-relation graph, where nodes are the users of an e-commerce platform. There are multiple edge types representing various relationships including friendship, workmates and classmates. Dou et al. (2020) build a review graph where users are represented as nodes in the graph. Three types of relations between users are defined to capture their behavioral patterns: reviewing the same product, having the same star rating, and having similar texts. With multiple edge types, this type of graph contains more information about the relationship between nodes, and thus how to capture this information is critical when developing the GNN methodology.

Dynamic graph

A dynamic graph is a sequence of graphs, where each graph could have an adjacency matrix and feature matrix. For example, Cheng et al. (2020b) construct a temporal guarantee network, representing the guarantee relationship in each time step. This type of graph is commonly used to represent the changes in both relations and features, as time goes. Since both nodes and edges could appear and disappear, it is hard to perform some graph operations that require

fixed dimensions of matrices. Thus, capturing the dynamically of the graph is challenging and requires a more sophisticated methodology.

Table 3.1: Summary of financial-related graphs

Graph	Construction method ²	Graph type ³	Application ^{4,5}	Reference
Sector-industry stock relation network	Knowledge	Multi	Stock	Feng et al. (2019)
Wiki company-based relation network	Knowledge	Multi	Stock	Feng et al. (2019); Sawhney et al. (2020a); Ying et al. (2020)
Supplier, customer, partner and shareholder relation graph	Knowledge	Multi	Stock	Matsunaga et al. (2019)
Corporation shareholder network	Knowledge	Homo	Stock	Chen et al. (2018)
Stock correlation graph	Similarity	Multi	Stock	Li et al. (2020b)
Stock earning call graph	Knowledge	Bipartite	Stock	Sawhney et al. (2020b)
News co-occurrence graph	Similarity	Homo	Stock	Liou et al. (2021)
User-relation graph	Data	Homo	Loan	Wang et al. (2019)
User-app graph	Data	Bipartite	Loan	Wang et al. (2019)
User-nickname graph	Data	Bipartite	Loan	Wang et al. (2019)
User-address graph	Data	Bipartite	Loan	Wang et al. (2019)
Guarantee network	Data	Directed	Loan	Cheng et al. (2019, 2020a)
Temporal guarantee network	Data	Dynamic	Loan	Cheng et al. (2020b)
Temporal small business entrepreneur network	Data	Dynamic	Loan	Yang et al. (2020)
Alipay user and applet graph	Data	Dynamic	Loan	Hu et al. (2020)
User relationship graph	Data	Multi	Loan	Liang et al. (2021)

² *Knowledge*, *Similarity*, *Data* denotes knowledge-based, similarity-based and data-based construction respectively.

³ *Multi* denotes multi-relation graph and *Homo* denotes homogeneous graph.

⁴ *Stock* denotes stock movement prediction. *Loan* denotes loan default risk prediction. *E-comm* denotes recommender system of e-commerce. *Fraud* denotes fraud detection.

⁵ Details for financial applications could be found in Section 3.5.

Table 3.1 (Continued)

Auto relation network	Data	Multi	Loan	Xu et al. (2021)
Loan application event graph	Data	Directed	Loan	Harl et al. (2020)
Borrower relations' network	Similarity	Directed	Loan	Lee et al. (2021)
Xianyu comment graph	Similarity	Homo	E-comm	Li et al. (2019a)
Yelp review network	Data	Bipartite	E-comm	Zhang et al. (2020); Dou et al. (2020)
Amazon review network	Data	Bipartite	E-comm	Zhang et al. (2020); Dou et al. (2020); Kudo et al. (2020)
E-commerce user-item network	Data	Bipartite	E-comm	Li et al. (2019b)
Taobao user-item network	Data	Bipartite	E-comm	Li et al. (2020d)
Device sharing graph	Data	Bipartite	Fraud	Liang et al. (2019)
JD Finance anti-fraud graph	Data	Multi	Fraud	Lv et al. (2019)
Transaction records graph	Data	Multi	Fraud	Rao et al. (2020)
Iqiyi user network	Data	Multi	Fraud	Zhu et al. (2020)
CMU simulated user activity network	Similarity	Homo	Fraud	Jiang et al. (2019)
Alipay one-month account-device network	Data	Bipartite	Fraud	Liu et al. (2018)
Alipay one-week account-device network	Data	Bipartite	Fraud	Liu et al. (2019)
Account-registration graph	Data	Dynamic	Fraud	Rao et al. (2021)
Bitcoin-alpha graph	Data	Directed	Fraud	Zhao et al. (2021)

3.3 Feature processing

With diverse data sources in the financial field, node features are commonly formatted as sequential numerical features or textual information. These data formats impose challenges on the feature processing step since GNN methods could not be directly applied to these data formats. In this section, we summarize the commonly-used feature processing techniques and how they solve these challenges.

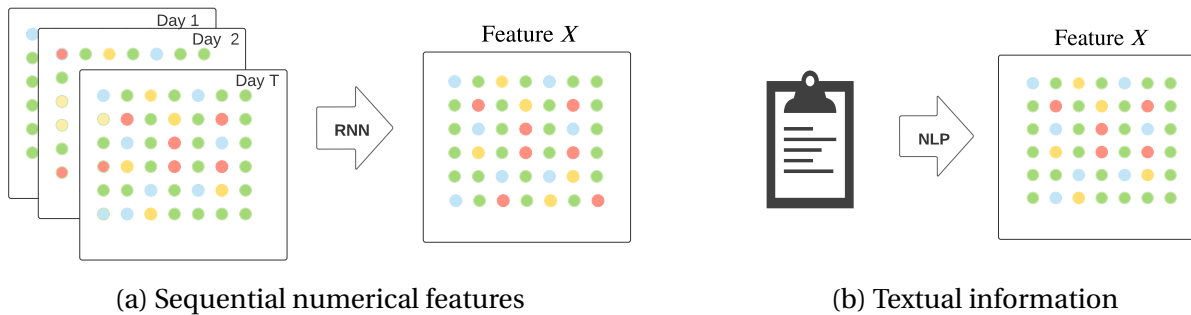


Figure 3.3: Feature processing for sequential features and textual information. For sequential numerical features, recurrent neural network (RNN) based approaches are commonly used to capture the temporal dependencies. For text features, it is often processed utilizing natural language processing (NLP) methods including word embedding, sentence embedding, and language models, to convert the unstructured data to structured ones.

3.3.1 Sequential numerical data

Updating information as time goes, the financial industry has a rich source of time-series data. Indexed using timestamps, features could be seen as sequential which requires appropriate modeling. Consider the feature matrix at time s , $X^s \in \mathbb{R}^{n \times p \times l}$, where n is the number of nodes, p is the dimension of features at each time point, l is the length of the sequence. For example, in a stock prediction task, we have a stock relation graph, with n stocks as nodes. The feature matrix X^s represents the p features in the past l days from time s for these n stocks. To encode the numerical sequence for each node, a recurrent neural network (RNN) is frequently used due to its superior performance in predicting time-series data. The literature could be summarized into two lines of work, long short-term memory (LSTM) based approach and gated recurrent unit (GRU) based approach.

LSTM based approach

As a special form of recurrent neural network, long short-term memory(LSTM) (Hochreiter and Schmidhuber 1997) is capable to capture the long-term dependencies and avoid the vanishing

gradient problem. Using memory cells and gate units, it has the following expression:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c), \\
c_t &= f_c \circ c_{t-1} + i_t \circ \tilde{c}_{t-1}, \\
h_t &= o_t \circ \tanh(c_t),
\end{aligned}$$

where $x_t \in \mathbb{R}^D$ is the input vector at time t and D is the number of features, $f_t, i_t, o_t, \tilde{c}_t, c_t, h_t$ denotes the forget gate, output gate, cell input, cell state and hidden state vectors respectively, $W_f, W_i, W_o, W_c, U_f, U_i, U_o, U_c$ are trainable weight matrices and b_f, b_i, b_o, b_c are trainable bias vectors, $\sigma(\cdot)$ represents the sigmoid activation function, and \circ denotes the element-wise product. The hidden state of the LSTM on day t is denoted by: $h_t = LSTM(x_t, h_{t-1}), s-l \leq t \leq s$.

Since the LSTM updates the hidden state to capture the structural information, a common approach to encode the historical data is generating sequential embedding E^s using the last hidden state of LSTM, $E^s = LSTM(X^s) \in \mathbb{R}^{n \times u}$, where u is the dimension of the output feature. Then the encoded pricing information is used as input to the graph neural network. For instance, Chen et al. (2018) used the generated sequential embedding E^s as the input feature matrix for graph convolutional network and Feng et al. (2019) utilized E^s as the input feature matrix for their proposed temporal graph convolutional layer. Using the last hidden state as an input feature, this type of method could capture the information in the past days while having an appropriate format to feed into the GNN model.

GRU based approach

Gated recurrent unit (GRU) (Cho et al. 2014) is another variant of RNN models. It also applies gating mechanism and has fewer parameters. It has the following structure:

$$\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\
\tilde{h}_t &= \tanh(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h), \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t,
\end{aligned}$$

where $x_t \in \mathbb{R}^D$ is the input vector at time t for stock i and D is the number of features, $r_t, z_t, \tilde{h}_t, h_t$ denotes the reset gate, update gate, candidate activation and hidden state vectors

respectively, $W_r, W_z, W_h, U_r, U_z, U_h$ are trainable weight matrices and b_r, b_z, b_h are trainable bias vectors, $\sigma(\cdot)$ represents the sigmoid activation function, and \circ denotes the element-wise product. The hidden states of the GRU on day t is denoted by: $h_t = \text{GRU}(x_t, h_{t-1})$.

Utilizing GRU to encode the past numerical information, we could obtain the hidden state for each day. Since past days' impact on the current-day representation may differ, an attention mechanism is frequently used to assign weights differently. For instance, Sawhney et al. (2020a) use an additive attention mechanism to aggregate the hidden states across time. Cheng et al. (2020a) utilize the concatenated attention method to incorporate different importance of time. The attention mechanism aggregates the hidden states of past days and assigns different weights across time. To get the feature representation at time s , it rewards the influential days when aggregating the hidden states from time $s - l$ to time s , and thus take temporal dependencies into account. The obtained node representation is then used as a feature in the graph neural network model.

3.3.2 Textual information

In the financial industry, a large amount of information is of textual form, including financial news, financial statements, and customer reviews. How to translate the texts into vector representation while preserving semantic information, is essential. The following section summarizes the commonly-used natural language processing (NLP) methodology to convert the unstructured texts into a vector form.

Word embedding and sentence embedding

Word embedding methods are widely used to represent a word as a fixed-length vector. Then, to learn the sentence representation, a recurrent neural network model is often utilized to capture local semantic information. For instance, to embed the news headlines, Li et al. (2020b) encode the word as word embedding using GloVe (Pennington et al. 2014). Then LSTM method is applied with an attention mechanism to create the sentence representation. However, since a word may have multiple meanings, word embedding methods may cause problems by assigning the same vector to words with different meanings. Thus, there are also approaches to embed at a sentence level in order to alleviate the problem. For instance, Sawhney et al. (2020a) generate sentence-level embedding for each Tweet using Universal sentence encoders (Cer et al. 2018).

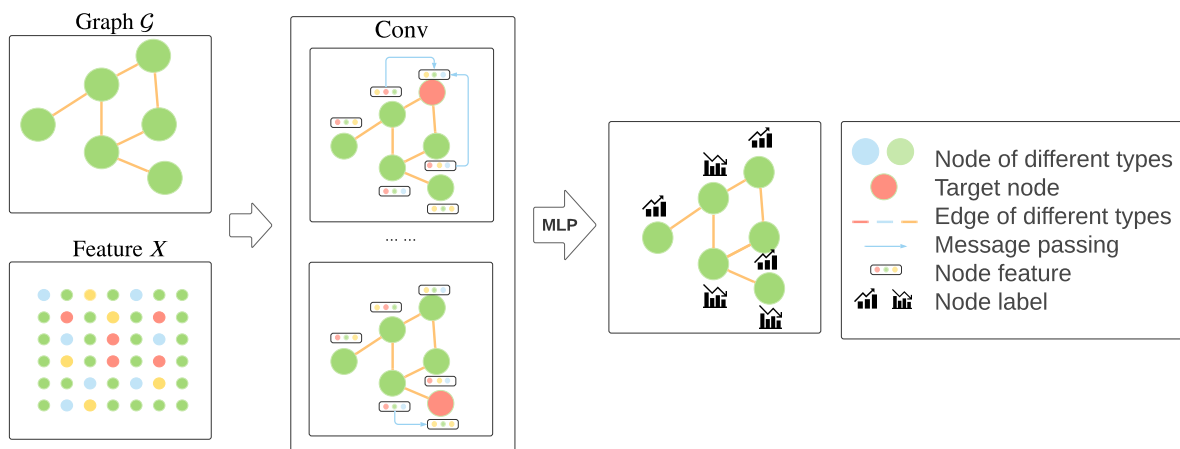
Language model

Instead of focusing on generating vectors for words, language models focus on capturing the pattern of languages to predict the word based on its surrounding words. Taking the contexts

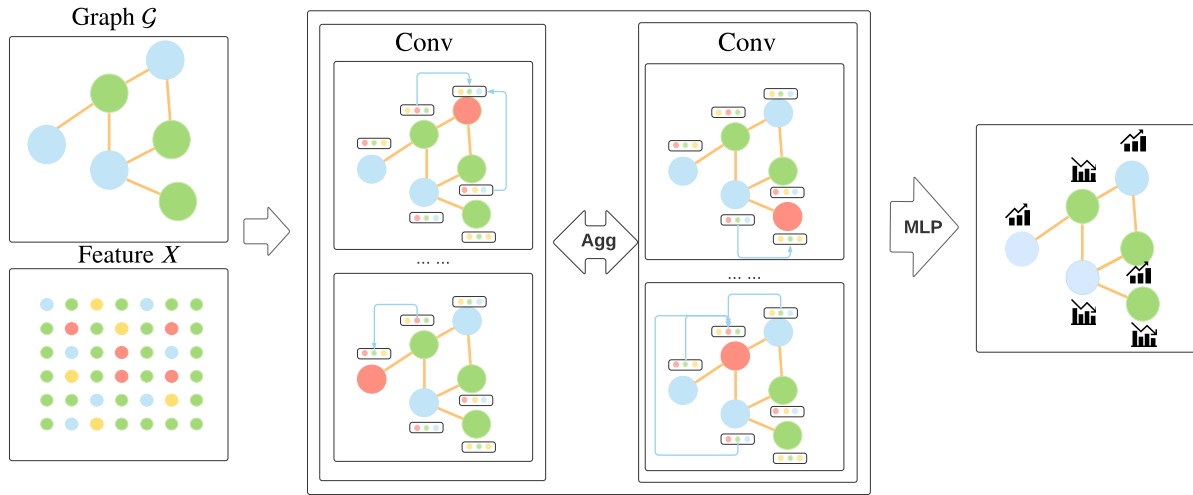
into account, language models achieve the state of art performance on many NLP tasks and are widely used in the literature. For instance, Liou et al. (2021) use bidirectional encoder representations from transformers (BERT) model (Devlin et al. 2019) to encode the entire news article and generated a news embedding. Without modifying the model architecture, the pre-trained BERT model is able to be fine-tuned and produce state-of-art performance. For example, FinBERT (Araci 2019), a BERT model pre-trained specific to the financial domain, is utilized to encode the text scripts in company earning calls (Sawhney et al. 2020b).

3.4 Graph neural network models

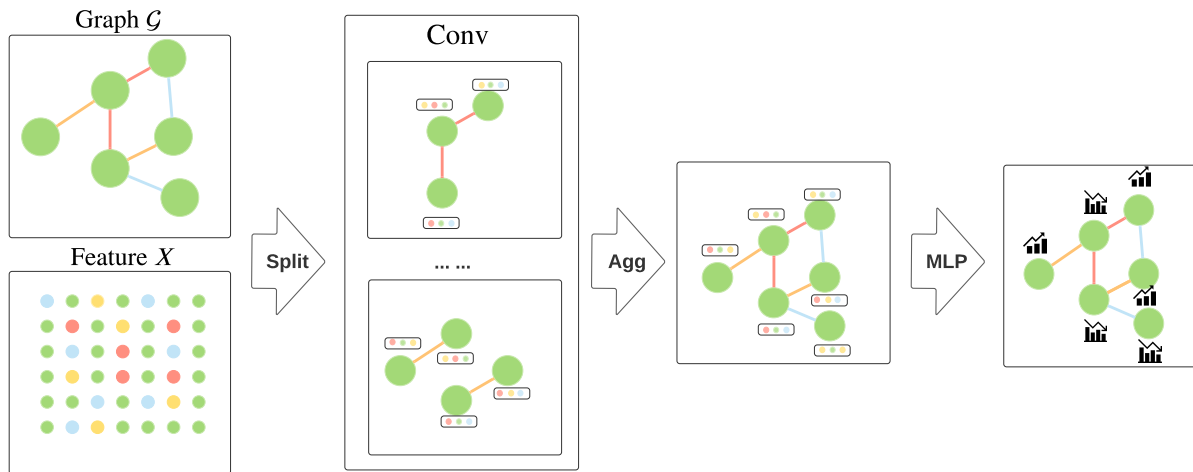
Proposed by Gori et al. (2005), a graph neural network is a neural network model capable of processing graphs. Unlike network embedding methods whose major aim is to generate a vector to represent each node, graph neural network models are designed for a variety of tasks, including node classification, edge prediction, and graph classification. Due to its wide application and superior performance, graph neural network models have drawn great attention. In this section, we present the commonly-used GNN models for each type of graphs, since the methodology may vary with different graph characteristics. In the supplementary materials, we present a figure demonstrating the major GNN methodology used for each graph type.



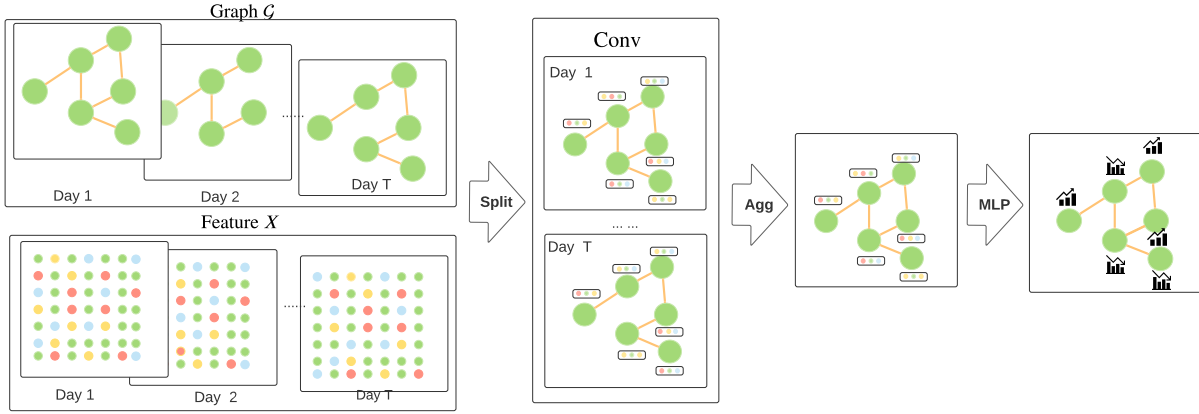
(a) Graph neural network models for homogeneous graphs. With a graph adjacency matrix and a feature matrix as inputs, the graph convolutional process generates a representation for each target node by aggregating the information of its neighbors. A multi-layer perceptron layer is then applied to generate the predicted labels.



(b) Graph neural network models for bipartite graphs. For nodes having the same type, the graph convolutional process updates their node representations using their own information and their neighbors' information. Aggregating functions are utilized to generate the node representations from the representations for both node types. A multi-layer perceptron layer is then applied to generate the predicted labels.



(c) Graph neural network models for multi-relation graphs. A multi-relation graph is split into several sub-graphs in which edges are of the same type. Within each sub-graph, the graph convolutional process takes place to update the node representations. Then between-graph aggregation is utilized to obtain the final node representations. A multi-layer perceptron layer is then applied to generate the predicted labels.



(d) Graph neural network models for dynamic graphs. A dynamic graph is often a sequence of graphs ordered by time. The graph convolutional process generates node representations for graph at each time stamp. To get the most recent representation, node representations are aggregated using functions including the recurrent neural network. A multi-layer perception layer is then applied to generate the predicted labels.

Figure 3.4: Graph neural network models for different graph types. The term Conv denotes graph convolution process. The term MLP denotes the multi-layer perception. The term Agg denotes the aggregation process. The term Split denotes data splitting according to its characteristics.

3.4.1 Homogeneous graph

Proposed by Kipf and Welling (2017), graph convolutional network (GCN) is a widely-used graph neural network model and could encode both local graph structure and node features. Extending the convolution concepts to graphs, graph convolution could be seen as message passing and information propagation. Aggregating neighbors' feature information, graph convolutional networks could be represented with the following layer-wise propagation rule:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l W^l),$$

where \tilde{A} is the adjacency matrix with added self-connections, $\tilde{D} = \text{diag}(\sum_j \tilde{A}_{ij})$ W^l is trainable weight matrix of l^{th} layer, H^l is the node hidden feature matrix in the l^{th} layer and $\sigma(\cdot)$ is the activation function. With its relatively simple model structure and great performance, the GCN model is often used as a benchmark method to compare with. Among the reviewed literature, over half of them have applied the GCN method as a benchmark method.

While GCN equally treats the neighbors of the target node, it often occurs that some neighbors may be more influential than others. Considering various impacts of the neighbor nodes,

Veličković et al. (2018) propose graph attention networks (GAT) and it is able to assign different weights to nodes in the same neighborhood as follows:

$$h_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^l W^l h_j^l\right),$$

$$\alpha_{ij}^l = \frac{\exp(\text{LReLU}(a^T [W^l h_i^l \| W^l h_j^l]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LReLU}(a^T [W^l h_i^l \| W^l h_k^l]))},$$

where h^l is the hidden feature vector for node i in the l^{th} layer, W^l is trainable weight matrix, a is a learnable vector, \mathcal{N}_i is the neighborhoods of node i , α_{ij}^l represents attention coefficient of node j to i at l^{th} layer, $\sigma(\cdot)$ is the activation function, $\|$ denotes vector concatenation, and LReLU denotes the leaky ReLU activation function. It is also worth mentioning that, since GAT is able to learn the weights of the neighboring node, we could interpret the learned attention weights as a relative importance measure, to better understand the model. Similar to GCN, GAT is also often used as a benchmark method in the reviewed papers with about 40% coverage.

3.4.2 Directed graph

A undirected graph has a symmetric adjacency matrix that guarantees a semi-definite Laplacian matrix, which lays the foundation for applying GCN. The directed graph, on the other hand, may has asymmetric adjacency matrix and could be handled by spatial-based GNN methods (Wu et al. 2020b), such as GAT. In practice, there is not much work developing methodologies for directed graph, since it could be processed by spatial-based GNN methods or making the adjacency matrix symmetric. However, there is also a line of work developing GNN methodology to predict sequences of events represented as a directed graph.

A sequence of events could be naturally formed as a directed graph, where each node is one type of event and an edge points from one event to the following one. Given the graph, which could be seen as a partial sequence of events, how to encode the features and predict the rest of the sequence is a challenge. The aforementioned GCN and GAT models aim at representation learning and are used to produce a single output instead of outputting a sequence. To fill the gap, Li et al. (2016) propose a gated graph neural network (GGNN) that could produce sequential outputs. It applies gated recurrent unit (GRU) as a recurrent function and is constructed as

follows:

$$\begin{aligned}
a_t &= A^T h_{t-1} + b, \\
r_t &= \sigma(W_r a_t + U_r h_{t-1}), \\
z_t &= \sigma(W_z a_t + U_z h_{t-1}), \\
\tilde{h}_t &= \tanh(W_h a_t + U_h (r_t \circ h_{t-1})), \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t,
\end{aligned}$$

where h_t is the updated event representation at t^{th} step, a_t contains information transferred from both directions' edges, r_t, z_t, \tilde{h}_t denotes the reset gate, update gate, and candidate activation vectors at t^{th} respectively, $W_r, W_z, W_h, U_r, U_z, U_h$ are trainable weight matrices, b is a trainable bias vector, $\sigma(\cdot)$ is the sigmoid function, \circ denotes the element-wise product, and \tanh denotes the hyperbolic tangent function. Incorporating the adjacency matrix A , GGNN aggregates the structural information in every propagation step. Unrolling the recurrence function to a fixed number, the GGNN ensures convergence without constraining the parameters.

3.4.3 Bipartite graph

The aforementioned methods focus on homogeneous graphs whose nodes and edges are all of one type. In real-life applications, graphs could be heterogeneous. As a running example, in a spam detection task, we could have a user-item network, where nodes are either users or items. An edge e_{ij} denotes that user i has rated on item j . This type of graph is well known as a bipartite graph \mathcal{G} with the following notation: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{U, W\}$ is a set of nodes and could be divided into two non-empty and disjoint sets \mathcal{U}, \mathcal{W} . \mathcal{E} is a set of edges, where every edge joins a node in \mathcal{U} and a node in \mathcal{W} .

Based on the characteristics of a bipartite graph, the commonly-used methodologies could be categorized into a framework as follows. In each iteration, the edge information h_{uw}^l is updated aggregating its previous hidden state h_{uw}^{l-1} and hidden states of the two nodes it links (h_u^{l-1}, h_w^{l-1}) , as shown in Equation (3.1). At each iteration, a user node u aggregates information from its rated items $h_{\mathcal{N}(u)}^l$ and its past hidden state h_u^{l-1} , as in Equation (3.3). While the representation of the rated items $h_{\mathcal{N}(u)}^l$ is updated aggregating the information from the linking edges h_{uw}^l and the item nodes h_w^{l-1} , as in Equation (3.2). Then, the item node w is

updated respectively as shown in the following equations:

$$h_{uw}^l = \sigma(W_E^l \cdot \text{AGG}_E(h_{uw}^{l-1}, h_u^{l-1}, h_w^{l-1})), \quad (3.1)$$

$$h_{\mathcal{N}(u)}^l = \sigma(W_{\mathcal{N}(U)}^l \cdot \text{AGG}_U(\text{AGG}_{UW}(h_w^{l-1}, h_{uw}^l))), \quad \forall w \in \{\mathcal{N}(u)\}, \quad (3.2)$$

$$h_u^l = \text{concat}(W_U^l \cdot h_u^{l-1}, h_{\mathcal{N}(u)}^l), \quad (3.3)$$

$$h_{N(w)}^l = \sigma(W_{\mathcal{N}(W)}^l \cdot \text{AGG}_W(\text{AGG}_{WU}(h_u^{l-1}, h_{uw}^l))), \quad \forall u \in \{\mathcal{N}(w)\}, \quad (3.4)$$

$$h_w^l = \text{concat}(W_W^{l-1} \cdot h_w^{l-1}, h_{N(w)}^l), \quad (3.5)$$

where e_{uw} represents the edge representation for edge linking node u and w , $h_{uw}^l, h_u^l, h_w^l, h_{\mathcal{N}(u)}^l, h_{\mathcal{N}(w)}^l$ are hidden states at l^{th} layer, $W_E^l, W_U^l, W_{\mathcal{N}(U)}^l, W_{N(W)}^l, W_W^l$ are trainable weight matrices at l^{th} layer, $\text{AGG}_E(\cdot), \text{AGG}_U(\cdot), \text{AGG}_{UW}(\cdot), \text{AGG}_W(\cdot), \text{AGG}_{WU}(\cdot)$ are user-chosen aggregation functions, $\sigma(\cdot)$ is the activation function, and concat denotes the vector concatenation.

There exist many modeling methodologies for bipartite graphs that could fit into the above framework. For instance, Zhang et al. (2020) propose a similar model structure as the framework and apply the attention mechanism as the aggregation function, since different items may have different impacts when learning users' representations. Instead of using all neighbors, Li et al. (2019a) utilize a sampling technique when aggregating neighbors' information in each iteration. There are also literature using the above framework as a building block and combining clustering methodology to learn a hierarchical representation of the graph, since hierarchical representation with various GNN models could achieve satisfactory performance. For instance, Li et al. (2019b) utilize the node embedding generated from the framework to cluster users into different communities and make a recommendation based on both community information and user information. Specifically, the user information is decomposed into two orthogonal spaces representing community-level information and individualized user preferences. Li et al. (2020d) treat the framework as a GNN module and stack it in a hierarchical fashion. With the embedding generated from the framework, clustering algorithms are performed to generate a coarsened graph which is used as an input for the next GNN layer.

3.4.4 Multi-relation graph

Instead of a simple homogeneous graph where all nodes and edges have the same type, in real life, there may exist multiple relations between nodes. For example, in a malicious account detection task, we could construct an Amazon review network. Users are the nodes in the graph and there are three relations between users: reviewing the same product, having the same star rating, and having similar texts. The multi-relation graph is denoted as $\mathcal{G} : \mathcal{G} = (\mathcal{V}, \mathcal{E}_{1:R})$, where \mathcal{V} is the set of nodes, $\mathcal{E}_{1:R}$ is the set of edges, R is the number of node types, $e_{i,j}^r \in \mathcal{E}_r$ is an edge

between node i, j with a relation $r \in \{1, \dots, R\}$.

A frequently used approach is to transform the heterogeneous graph into multiple homogeneous graphs by extracting R subgraphs $\{\mathcal{G}^r = (\mathcal{V}, \mathcal{E}_r), r = 1, \dots, R\}$ from it. Each subgraph \mathcal{G}^r only preserves one type of edge and thus is homogeneous. This line of work could be unified in a two-step framework. The first step implements the sub-graph aggregation to aggregate neighbor information in each sub-graph as shown in Equation (3.6). The second step is to conduct the inter-relation aggregation to aggregate relation-specific embeddings as Equation (3.7),

$$h_{i,r}^l = f(\text{AGG}_r\{h_{j,r}^{l-1}\}), \quad \forall j \text{ s.t. } (i, j) \in \mathcal{E}_r, \quad (3.6)$$

$$h_i^l = g(\text{AGG}\{h_{i,(1:R)}^l, h_i^{l-1}\}), \quad (3.7)$$

where $h_{i,r}^l$ is the subgraph-specific embedding of node i in subgraph r in l^{th} layer, h_i^l is the general embedding of node i in l^{th} layer, $\text{AGG}_r(\cdot)$ is the aggregation function in subgraph r , $\text{AGG}(\cdot)$ is the inter-relation aggregation function, and $f(\cdot), g(\cdot)$ are user-defined functions.

There are multiple methods that could be categorized into the above two-step framework. For instance, in a fraud classification task, Liu et al. (2018) observe that fraudsters tend to congregate in topology and thus use weighted sum for within-relation aggregation to capture this congregation pattern. They then apply an attention mechanism for inter-relation aggregation to learn the significance for each sub-graph as follows:

$$h_{i,r}^l = \sigma(\text{Weighted mean}\{h_{j,r}^{l-1}\}), \quad \forall j \text{ s.t. } (i, j) \in \mathcal{E}_r,$$

$$h_i^l = \sigma(X_i W + \text{Attention}\{h_{i,(1:R)}^l\}),$$

where X_i is the feature vector for node i , W is a trainable matrix, $\sigma(\cdot)$ is the activation function, and Attention denotes the attention aggregator.

To incorporate neighbors' information, Dou et al. (2020) use the mean aggregator for within-relation aggregation. To reduce the computational cost and keep the relational importance information, they apply a pre-calculated parameter p_r^l as the weight in the intra-relation aggregation step. They also test several aggregating functions when aggregating relation-specific embeddings with the following structure:

$$h_{i,r}^l = \sigma(\text{Mean}\{h_{j,r}^{l-1}\}), \quad \forall j \text{ s.t. } (i, j) \in \mathcal{E}_r,$$

$$h_i^l = \sigma(h_i^{l-1} + \text{AGG}\{h_{i,r}^l \cdot p_r^l\}), \quad \forall r \in (1, \dots, R),$$

where p_r^l is a pre-trained weight.

Since different relations provide various facets of user characteristics, relationship-specific embedding may have different statistical properties, which may cause trouble when aggregating them in a lower-level space. To deal with that, Wang et al. (2019) project the relation-specific node embedding to higher spaces using multi-layer perception (MLP) and concatenate them with relation-level attention:

$$h_{i,r}^l = \text{MLP}\{h_{i,r}^{l-1}\}, \text{ where } h_{i,r}^1 = \text{Attention}\{x_{j,r} : \forall j \text{ s.t. } (i, j) \in \mathcal{E}_r\},$$

$$h_i^l = \text{Concatenation with attention}\{h_{i,(1:R)}^l\},$$

where Attention denotes the attention aggregator.

3.4.5 Dynamic graph

The previously mentioned neural network models generally focus on a static graph. However, in real-life settings, a graph may be dynamically evolving since relations may be updated with time. For example, in order to predict the loan default risk, a guarantee network needs to be updated, adding newly-constructed guarantee relationships and removing companies that have fully paid the loan. With the rapid development of graph neural network methodologies on static graphs, there emerges a trend to extend GNN models to a dynamic setting. Consider a sequence of T graphs $\mathcal{G}^{seq} = \{\mathcal{G}_1, \dots, \mathcal{G}_T\}$, where $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ represent the graph at i^{th} time point. The feature matrices are represented as $X = \{\mathbf{X}_1, \dots, \mathbf{X}_T\}$ and the adjacency matrices are $A = \{\mathbf{A}_1, \dots, \mathbf{A}_T\}$.

To capture the sequential pattern in the dynamic graph, a common approach is to train a GNN to generate the node embedding at each time stamp and then utilize a recurrent neural network to aggregate the information. For example, Cheng et al. (2020b) obtain the node embeddings at each time step by training a GCN with multi-head attention. Then, they utilize the GRU to capture the sequential pattern with a temporal attention layer to capture the temporal variation over timestamps. Similarly, Yang et al. (2020) first aggregate node and edge information in each snapshot and then employ a LSTM operator to capture the temporal variations in the node embeddings.

In the aforementioned methods, a graph neural network is learned for feature aggregation and an RNN model is trained to capture the sequential pattern of the node embeddings. However, in reality, a node may appear and disappear, which may worsen the performance of the RNN model when updating the node representation. In a guarantee network, for example, a company that has borrowed loans could disappear from the graph after it pays all the debts and could appear again backing up other companies' loans. To overcome the limitation, Pareja et al. (2020) proposed EvolveGCN which utilizes a recurrent neural network to evolve the GCN

parameters instead of updating the node embeddings. For each time point t , a GCN model is constructed as follows to fit the graph \mathcal{G}_t :

$$H_t^{l+1} = \sigma(\tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}} H_t^l W_t^l),$$

where \tilde{A}_t is the adjacency matrix with added self-connections, $\tilde{D}_t = \text{diag}(\sum_j \tilde{A}_{ij})$ is the degree matrix, W_t^l is trainable weight matrix of l^{th} layer, H_t^l is the matrix of activation in the l^{th} layer, and $\sigma(\cdot)$ is the activation function.

To update the weight matrix W_t^l , Pareja et al. (2020) propose two methods. The first method considers W_t^l as a hidden state of the dynamics and update it using a GRU model, as shown in Equation (3.8). The second method treats W_t^l as an output state which is updated using a LSTM method, as shown in Equation (3.9). The structure for both methods is as follows:

$$W_t^l = \text{GRU}(H_t^l, W_{t-1}^l), \quad (3.8)$$

$$W_t^l = \text{LSTM}(W_{t-1}^l). \quad (3.9)$$

Compared to the second method, the first method incorporates the updated node embedding in the recurrent neural network and it may lead to better performance when node features are informative.

3.5 Application

In this section, we have detailed some financial applications that the GNN methods have been commonly applied on. We have also summarized features, graphs, methods, evaluation metrics, and baselines used in each financial application in the supplementary materials.

3.5.1 Stock movement prediction

Though there are still debates on whether stocks are predictable, stock prediction receives great attention and there are rich literature on predicting stock movements using machine learning methods. However, the task of stock prediction is challenging due to the volatile and non-linear nature of the stock market. Traditionally, there are two major approaches to handle the task: technical analysis and fundamental analysis (Sawhney et al. 2020a). Technical analysis utilizes numerical features such as closing prices and trading volumes, while the fundamental analysis approach includes non-numerical information, such as news and earning calls. The limitation of these non-graph approaches is that they often have a hidden assumption that the stocks are independent. To take the dependence into account, there is an increasing trend to

represent the stock relations in a graph where each stock is represented as a node and an edge would exist if there are relations between two stocks. Predicting multiple stocks' movements could then be formed as a node classification task and GNN models could be utilized to make the prediction.

In this section, we summarize the literature on stock prediction applying GNN methods, where table 3.2 presents the key features and graphs used in this application. There also exist challenges to apply the GNN methods in the stock prediction task. Unlike other fields where the benchmark graphs are available, to the best of our knowledge, there is no off-the-shelf graph representing inter-stock relations. With abundant relations existing in the financial system, it becomes challenging to obtain and select the relation for graph construction. Moreover, owing to the volatility of the stock market, how to model the sequential features and capture the temporal patterns are also critical. Also, the financial industry has rich data sources including financial statements, news and pricing information, which impose difficulty on modeling the data.

Table 3.2: Summary of stock movement prediction literature

Reference	Feature ⁶	Graph	Method	Evaluation metric	Baseline method
Chen et al. (2018)	7-day open, high, low, close prices and volume for CSI listed companies	Corporation shareholder network	LSTM + GCN	Accuracy	LSTM, GCN
Feng et al. (2019)	5/10/20/30 days moving average of close price for SP500/NYSE listed companies	Wiki company-based relation network & sector-industry stock relation network	RSR	MSE, IRR, MRR	SFM, LSTM, GBR, GCN
Matsunaga et al. (2019)	5/10/20/30 days moving average of close prices for Nikkei 225 listed companies	Supplier, customer, partner and shareholder relation graph	RSR	Return ratio, Sharpe ratio	Model with subsets of relations

⁶The acronym and the respective full form could be found in the Appendix Chapter C.

Table 3.2 (Continued)

Ying et al. (2020)	5/10/20/30 days moving average of close prices and stock description documents for SP500 or NYSE listed companies	Wiki based network	company-relation	TRAN	MSE, MRR, IRR	RSR, LSTM	GCN,
Li et al. (2020b)	News of 500/100 TPX listed companies	Stock graph	correlation	LSTM-RGCN	Accuracy	Naïve Bayes, LR, RF, HAN, transformer, S-LSTM	
Sawhney et al. (2020a)	Price and social media information for companies listed in the SP500 index or NYSE or NASDAQ markets	Wiki based network	company-relation	MAN-SF	Accuracy, F1, MCC	ARIMA, RF, TSLDA, HAN, StockNet, LSTM+GCN	
Sawhney et al. (2020b)	Text and audio features of earning calls for companies in the SP500 index	Stock graph	earning call	VoITAGE	MSE, R-squared	LSTM, HAN, MDRM, HTML	
Liou et al. (2021)	News and attributes for stock tags	News occurrence	co-graph	HAN	Accuracy, MCC	RF	

There are multiple ways to construct the stock relational graph. For instance, believing that correlation on historical prices reflects the inter-stock relation, Li et al. (2020b) construct the graph using the correlation matrix of historic data to predict the movement of Tokyo stock price index. On the other hand, Matsunaga et al. (2019) borrow information from knowledge bases and construct supplier, customer, partner, and shareholder relational graphs. With multiple ways of graph construction, there doesn't exist a "best" graph due to the lack of graph evaluation methods. Future work could be done to design a graph evaluation method to help researchers

better construct a relational graph.

To effectively process the sequential data and incorporate related corporations' information Chen et al. (2018) propose a joint model using LSTM and GCN to predict the stock movement. However, Chen et al. (2018)'s approach assumes that the relations between stocks are static, which may not reflect the reality. Instead, Feng et al. (2019) propose a temporal graph convolution layer to capture the stock relations in a time-sensitive manner, so that the strength of relation could be evolving over time. The relations are then updated based on historical pricing sequences and the proposed method obtained better performance compared to GCN. Believing that stock description documents also contain information reflecting the changes in companies' effect, Ying et al. (2020) capture the temporal relation by both sequential features and stock document attributes with a time-aware relational attention network.

The aforementioned methods focus on capturing the temporal dependencies, while Sawhney et al. (2020a) focus on fusing data from different sources. Sawhney et al. (2020a) propose a multipronged attention network to jointly learn from historical price, social media, and inter stock relations. Encoded pricing and textual information are used as node feature inputs to GAT, where the graph information comes from the Wiki company-based relations. The attention mechanisms is applied to allocate different weights on various data sources and latent correlations may learned via the attention layers.

3.5.2 Loan default risk prediction

For commercial banks and financial regularity institutions, monitoring and assessing the default risk is at the heart of risk controlling process. As one of the credit risks, default risk is the probability that the borrower fails to pay the interest and principal on time. With a binary outcome, loan default prediction could be seen as a classification problem and is commonly addressed utilizing user-related features with classifiers including neural network (Turiel and Aste 2020) and gradient boosted trees (Ma et al. 2018). Since the probability that a borrower defaults may be influenced by other related individuals, there is plenty of literature forming a graph to reflect the interactions between borrowers. With the rapid growth of GNN methods, GNN methods are widely applied on the graph structure for loan default predicting problems. There are currently three lines of work focusing on various types of loans: guarantee loans, e-commerce loans, and other loans.

Table 3.3: Summary of loan default risk prediction literature

Reference	Feature	Graph	Method	Evaluation metric	Baseline method
Wang et al. (2019)	/	Multiple user networks	SemiGNN	AUC, KS	Xgboost, LINE, GCN, GAT
Cheng et al. (2019)	Active loan behavior, historical behavior, and user profile	Guarantee network	HGAR	AUC, Precision@k	GF, DW, node2vec, AANE, SNE, GAT
Cheng et al. (2020a)	Customer profile, loan information, guarantee profile, and loan contract	Guarantee network	TRACER	F1, Precision@k	LR, GBDT, DNN
Cheng et al. (2020b)	Loan behavior and company profile	Temporal guarantee network	DGANN	AUC	GF, GCN, node2vec, GAT, SEAL, RNN, GRNN
Yang et al. (2020)	Credit-related features, spatial features, and temporal features	Temporal small-business entrepreneur network	ST-GNN	AUC, KS	GBDT, GAT, STAR
Hu et al. (2020)	User credit exposure features	Alipay user and applet graph	AMG-DP	AUC, KS	MLP, Xgboost, node2vec, GraphSAGE, GAT, HAN, SemiGNN
Liang et al. (2021)	Device information and trading features	User relationship graph	MvMoE	AUC	GBDT
Xu et al. (2021)	User profile and transaction summary	Auto loan network	GRC	Precision, recall, F1	SVM, MLP, GCN

Table 3.3 (Continued)

Lee et al. (2021)	Loan information, credit history, and soft information	Borrower's relation / network	Accuracy, precision, recall, F1, AUC	SVM, Xgboost, GCN	RF, MLP,
-------------------	--	-------------------------------	--------------------------------------	-------------------	----------

The guarantee loan allows small entrepreneurs to back each other in order to increase their credibility. It has a debt obligation contract that specifies that if one corporation fails to pay the debt, its guarantor needs to pay for it. A guarantee network naturally arises where each node is a company and directed edges represent the guarantee relationship. To learn a better representation of the network, Cheng et al. (2019) utilize the graph attention layer and design a objective function, so that vertices with similar structures will be closer in the learned feature space. Since the guarantee relations changes with time, Cheng et al. (2020b) forms a dynamic guarantee network to represent the dynamics. A recurrent graph neural network layer is developed to learn the temporal pattern and attentional weights are learned for each time point via an attention architecture.

Unlike guarantee loans that the loan information could be naturally represented in a directed graph, other loan types may not have a clear graph structure and researchers need to construct the graph based on interactive information. For instance, Xu et al. (2021) construct a user relation graph where users are connected by various relationships, such as social connections, transactions, and device usage. However, the interactive graph may also contain noisy data, which may be irrelevant. Since the massive interactive information may be noisy and the impacting supply-chain information is deficient, Yang et al. (2020) extract supply-chain relations while predicting loan defaults. Forming the interaction data as a graph, Yang et al. (2020) formulate the supply chain mining task as a link prediction task and thus construct a supply chain network, which is then used to predict the default probability with GNN methodology.

With the rise of e-commerce, e-commerce consumer lending service is gaining popularity to enhance consumers' purchasing power. Able to obtain information from multiple facets, the e-commerce platform could have multi-view data and multi-relation networks, which may require sophisticated modeling methodology. For instance, in order to predict the default probability for each consumer with multi-view data, Liang et al. (2021) utilize a hierarchical attention mechanism to encode the features on each view. Exploring multiplex relations, Hu et al. (2019) propose an attributed multiplex graph-based model with relation-specific layer and

attention mechanism to jointly model multiple relations. To simultaneously model the labeled and unlabeled data, Wang et al. (2019) proposed a semi-supervised graph neural network approach and obtain interpretable results.

3.5.3 Recommender system of e-commerce

With the rapid growth of e-commerce, customers gradually get used to shopping online and are exposed to a numerous range of products. To alleviate the burden of users choosing the appropriate item, the recommender system is developed to suggest products to users based on predicted item ratings. Presenting the user and item information in a graph, GNN methodologies are widely used in recommender system related tasks including click rate prediction and fake review detection.

To accurately predict users' preferences and recommend the appropriate item, it is vital to exploit information for users, items, and their interactions. A widely-used representation of that information is a bipartite graph, where nodes are of two types, user and item, and edges represent that there are relationships between user and item nodes. Noticing that the community the user belongs to may affect the shopping decision, for example, a user belongs to the traveler group may purchase travel-related items, Li et al. (2019b) combine the bipartite graph modeling algorithm with clustering techniques to reflect the community impact and the individual preference. However, Li et al. (2019b)'s approach only considers the hierarchy in the user side, while items may also have hierarchical information. To fill the gap, Li et al. (2020d) stack several GNN modules hierarchically to capture the hierarchical structure in both user and item perspectives. Embeddings learned from the GNN layer are clustered and used as an input for the next GNN layer, which could preserve the high-order hierarchical connections.

The recommender system is mainly based on the past history of the user, including its rating and reviews on the item. However, fake ratings and feedback may be posted by the fraudsters to seek financial benefits. To detect fraudulent reviews on the e-commerce platform, Kudo et al. (2020) construct a directed and signed comment graph with a signed graph convolutional network approach. Compared to Kudo et al. (2020)'s approach which only considers the comment graph, Li et al. (2019a) integrate both the bipartite user-item graph and a comment graph to capture the local and global context of the comments. Noticing that camouflage behaviors of the fraudsters may deteriorate the performance of fraud detection mechanism and has seldom been considered by prior works, Dou et al. (2020) propose a model against both feature and relation camouflage. For each node, only informative neighbors are selected for the next aggregation step, utilizing a similarity measure and a reinforcement learning mechanism. While the above literature focus on the fraud review detection side, there are also works that

accomplish both fraud review detection and item recommendation tasks. For example, Zhang et al. (2020) propose a GCN-based framework that performs both item recommendation and fraud detection in an end-to-end manner, while each of the tasks is beneficial for the other one.

Table 3.4: Summary of literature on recommendation system of e-commerce

Reference	Feature	Graph	Method	Evaluation metric	Baseline method
Li et al. (2019a)	Features for item, user and comment	Xianyu comment graph	GAS	AUC, F1, recall	GBDT
Zhang et al. (2020)	Behavioral features	Yelp review network, Amazon review network	GraphRfi	MAE, RMSE, precision, recall, F1	RCF, GCMC, PMF, ICF, MF
Dou et al. (2020)	Behavioral features	Yelp review network, Amazon review network	CARE-GNN	AUC, recall	GCN, GAT, RGCN, GraphSAGE, SemiGNN
Kudo et al. (2020)	Behavioral features	Amazon review network	GCNEXT	AUC	RGCN, SIDE, SGCN
Li et al. (2019b)	Purchasing power, number of transactions, item category, and shipping costs	E-commerce user-item network	Bi-HGNN	Accuracy, AUC, F1	GraphSAGE, Diffpool
Li et al. (2020d)	Click and transaction logs	Taobao user-item network	HiGNN	AUC	DIN, GE

3.5.4 Fraud detection

Including payment fraud, identity theft, financial scam and insurance fraud, financial fraud has a variety of types and has an increasing trend (Kurshan and Shen 2020). Observing that fraudsters tend to have abnormal connectivity with other users, there is a trend to present users' relations in a graph and thus, the fraud detection task could be formulated as a node

classification task.

Aiming to detect the malicious accounts, who may attack the online services to seek excessive profits, Liu et al. (2018) find out that fraudsters have two patterns: device aggregation and activity aggregation. Due to economic constraints, attackers tend to use limited number of devices and perform activities in a limited time, which may be reflected in the local graph structure. With this observation, Liu et al. (2018) propose a variant of GCN and use sum operators to capture the aggregation pattern. While the malicious accounts may be aggregated together, Liu et al. (2019) argue that the normal accounts could also be connected with the malicious account and may be mislabeled as malicious, which adds noisy signals to the graph. Taking into account that the graph could be noisy and nodes may have different impact, Liu et al. (2019) propose an adaptive path layer to adaptively select the important neighbor nodes that contribute most to the target node. Applying the GNN methodologies proposed by Liu et al. (2018), Liang et al. (2019) stack multiple adaptive path layers to aggregate neighbors' features and have great performance in a insurance fraud detection task.

Table 3.5: Summary of fraud detection prediction literature

Reference	Feature	Graph	Method	Evaluation metric	Baseline method
Liang et al. (2019)	Insurance claim history, shipping history and shopping history	Device sharing graph	/	F1, DE	GBDT, node embeddings
Lv et al. (2019)	Purchase history of 1000 commodity categories	JD Finance anti-fraud graph	AutoGCN	AUC	GCN, GAT, GCN with attention
Rao et al. (2020)	Individual risk features	Transaction records graph	xFraud	AUC	LR, DNN, GAT, GCN, HGT
Zhu et al. (2020)	User features	Iqiyi user network	HMGNN	Precision, recall, F1, AUC	LR, Xgboost, MLP, GCN, GAT, ASGCN, mGCN
Jiang et al. (2019)	User behavioral features and content based features	Simulated user activity network	/	Accuracy, precision, recall	RF, SVM, LR, CNN

Table 3.5 (Continued)

Liu et al. (2018)	User activities	Alipay one-month account-device network	GEM	F1, AUC, precision-recall	Connected sub-graph, GBDT, GCN
Liu et al. (2019)	User activities	Alipay one-week account-device network	GeniePath	Accuracy	MLP, node2vec, GCN, GraphSAGE, GAT
Rao et al. (2021)	Registration profile and transaction features	Account-registration graph	DHGReg	Precision	MLP, GCN, GAT
Zhao et al. (2021)	/	Bitcoin-alpha graph	GAL	Precision, recall, F1, AUC	GCN, GAT, GraphSAGE, DOMINANT

The above literature focus on a bipartite graph, where nodes are either account or devices, and there are also literature utilizing other types of graphs. For instance, Jiang et al. (2019) construct a homogeneous user network connecting user nodes based on their similarity of behaviors and apply the GCN model. Rao et al. (2021) focus on a dynamic graph of registration records and implement GCN layers on structural and temporal subgraphs.

Besides the literature developing GCN-based methodologies, there are also fraud-detection related literature applying GNN models with other structures. Since GCN may suffer from over-smoothing problem and have shallow model structure, Lv et al. (2019) propose to replace the graph convolutional matrix with auto-encoder to increase the depth of the neural network. Zhao et al. (2021) argues that GNN models with random-walk based losses have poor performance in anomaly detection task, since nodes of the same label may not be closer. They then propose a loss function which leads to better performance and has bounded prediction error.

3.5.5 Event prediction

Financial events, including revenue growth, acquisition and bankruptcy, could provide valuable information on market trends and could be used to predict future stock movement. Therefore, it draws great attention on how to predict next financial event based on past events and currently GGNN model is often used to accomplish the task. For example, given a sequence of financial events of Chinese listed companies, Yang et al. (2019) aim to predict the next event type and construct an event graph where each node is a financial event and edges are weighted using

the frequency of the event pairs. Harl et al. (2020) transform a binary classification problem into an event prediction task by dividing the loan application process into several events and predicting whether the next event would be accepting or rejecting the application. Utilizing the GGNN model, they obtain the predictions with high accuracy.

Table 3.6: Summary of event prediction literature

Reference	Feature	Graph	Method	Evaluation metric	Baseline method
Harl et al. (2020)	Event features	Loan application event graph	GGNN	Accuracy	/
Yang et al. (2019)	Financial news	Financial graph	GGNN	Accuracy, precision, recall, F1	PMI, LSTM, DW,

3.6 Challenges

3.6.1 Graph evaluation methods

To justify the inclusion of a graph, a commonly-used evaluation method is to compare the outcome for a graph-based machine learning method with a graph-free machine learning method (Feng et al. 2019; Li et al. 2020b). However, there is little discussion on comparing different graphs’ effects and quality, while existing literature discussing graph comparisons are often inadequate. For example, Liang et al. (2019) visualize the structural patterns in different graphs, concluding that the device sharing graph is more appropriate based on the observed patterns. Without presenting the evaluation metric for each graph, the graph comparison based on the visualization may not be adequate. The problem may be more severe in similarity-based graph construction since a threshold needs to be set to determine whether an edge exists. Different threshold values may lead to completely different graphs and thus affect the model performance. Thus, justification on threshold setup during graph construction is of great importance. Some efforts have been made to tackle this problem, such as utilizing a reinforcement learning approach to automatically select the optimal threshold (Dou et al. 2020). More attention may need to be drawn to develop a framework to assess the graph quality systematically.

3.6.2 Explainability

Combining both graph structural information and feature information, GNN models are often complicated and it is challenging to make an interpretation. Recently, there are some literature focusing on the explainability of GNN models. GNNExplainer (Ying et al. 2019), for example, is proposed to provide an interpretable explanation on trained GNN models such as GCN and GAT. Model explainability in financial tasks is of great importance, since understanding the model could benefit decision-making and reduce economic losses. However, there is little literature studying the explainability of GNN models in a financial application, which often accompanies by heterogeneous and dynamic graphs. Current literature focuses on relatively simpler graphs. For example, Li et al. (2019c) extend the GNNExplainer to a weighted directed graph and apply it on a Bitcoin transaction graph. Rao et al. (2020) propose an explainable fraud prediction system that could operate on heterogeneous graphs consisting of different node and edge types. More work could be done on the explainability of GNN models with edge-attributed graphs and dynamic graphs, which are not yet considered.

3.6.3 Task type

Tasks of GNN models are commonly classified into three categories: node-level task, edge-level task, and graph-level task (Wu et al. 2020c). However, the GNN methods applied on financial applications are mostly focused on the node-level task. For example, the stock movement prediction task is often formulated as a node classification task, where stocks are represented as nodes. There exist some literature focus on other types of task, for instance, Yang et al. (2020) aim to mine the underlying supply chain network and formulate it as a link prediction task, but this line of work is rare. Since there are rich literature developing GNN methodologies on edge prediction or graph classification tasks, there are rich opportunities on applying recently-developed GNN methods on financial fields if financial tasks could be formulated into an edge or graph level task.

3.6.4 Data availability

To pursue reproducibility, it is a common practice to release both data sources and codes when publishing the paper. However, in the reviewed papers, only about 24% of them release the code. Since the financial tasks are commonly based on real-world problems, data may have some restrictions due to the privacy obligation of the related corporations. Lack of open-source codes and datasets, it is hard to reproduce previous works and compare their methodologies in the latter literature. Thus, it is of great value to construct benchmark datasets, so that methods

could be compared on the same data.

3.6.5 Scalability

In the real-world financial scenario, commercial data are often of large scales. For instance, Yang (2019) utilize data from a popular e-commerce platform and it contains about 483 million nodes with 231 million edges. How to improve the scalability of GNNs is vital but challenging. Computing the Laplacian matrix becomes hard with millions of nodes and for a graph of irregular Euclidean space, optimizing the algorithm is also difficult. Sampling techniques may partially solve the problem with the cost of losing structural information. Thus, how to maintain the graph structure and improve the efficiency of GNN algorithms are worth further exploration.

CHAPTER

4

CAUSALITY GUIDED GRAPH NEURAL NETWORK

4.1 Introduction

Representing complex relational data, graph data contain both the individual node information and the structural information. Both related with the graph structure, causal structure discovery approaches and graph neural network methods have different perspectives. On the one hand, causal structure discovery primarily focuses on the interpretation problem, aiming to estimate a causal graph and understand the underlying causal relationship (Spirtes et al. 2000a; Shimizu et al. 2006; Yu et al. 2019). On the other hand, graph neural network is tailored for the prediction task, where the graph data is served as an input to provide node-specific feature and relational information (Kipf and Welling 2017; Veličković et al. 2018; Hamilton et al. 2017). Telling one story in two tales, these two methods have their strengths and also respective drawbacks.

Assuming that the observed i.i.d. data is generated from a hidden causal graph, many methods are developed for causal structural learning to recover the hidden causal structure, such as the PC algorithm (Spirtes et al. 2000a), ICA-LiNGAM (Linear Non-Gaussian Acyclic Model) (Shimizu et al. 2006), and NOTEARS (Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning) (Zheng et al. 2018). To cater for complex

scenarios, there are many methodologies that are developed where i.i.d. assumptions no longer hold. For instance, DYNOTEARS (DYnamic NOTEARS) algorithm (Pamfil et al. 2020) could capture time-lagged dependencies where the past observations could causally affect the current observations, and ACD (Amortized Causal Discovery) (Löwe et al. 2022) leverages shared dynamics from multiple time series samples, to enable the model to infer causal relations. Obtaining an estimated causal graph as the outcome, the above literature uses simulation data to validate the accuracy of the causal graph estimate. However, in real-world scenarios, it is hard to evaluate the quality of the causal graph without a given golden label, which adds to the uncertainty of the learned causal structure (Cheng et al. 2024). Also, interpreting the casual graph imposes challenges for general public (Shen et al. 2020), particularly when numerous causal relations exist, and the causal graph lacks application in the downstream tasks.

Utilizing graph representation of the relational information and the nodes' feature, graph neural network (GNN) achieves the state-of-art performance for forecasting tasks, including node classification, edge prediction and graph classification (Xu et al. 2018; Veličković et al. 2018; Hamilton et al. 2017). This success is majorly due to graph neural network's ability of aggregating and updating neighboring nodes' features to generate a representation for each nodes in an end-to-end manner. However, recent works (Knyazev et al. 2019; Sui et al. 2022) have shown that graph neural network are susceptible to exploiting spurious features in the modeling step, which may harm model's generalization ability (Geirhos et al. 2020). Relying on the spurious feature, the GNN may learn to assign a large weight to the spurious feature to accomplish the task, instead of struggling to learn actual important ones. This phenomenon could lead to worse performance on out-of-distribution data, where the spurious information is no longer correlated with the outcome data. For instance, as shown in Figure 4.1, considering the graph classification task, where we aim to classify the type of a digit in a picture, where the picture category of digit highly correlates with a pre-defined color in their background, e.g. 1 is always green. The GNN may capture the spurious feature, i.e. color, when making the classification, which may lead to poor performance when we reverse the coloring pattern in the test data, e.g. coloring 1 as red. To fill the gap, in this project, we aim to develop a causality-guided graph neural network approach, which incorporates the causal structure into graph neural network modeling, disentangling the spurious variable and causal features. We will first estimate a causal graph, utilizing both structural information and node features, which could provide causal insights for the audience to better understand the model. The estimated causal graph will then be incorporated into the graph neural network, where the GNN model adaptively learn the attention weights for each variable to differentiate the spurious variables. The major contributions could be summarized as follows:

- We propose a causality-guided graph neural network approach, integrating causal struc-



Figure 4.1: Colored MNIST(Arjovsky et al. 2019) data set for graph classification task, where the coloring pattern is different in the training and testing data.

ture learning into GNN models to obtain better interpretation and generalization abilities.

- We develop an algorithm which could generates both a causal graph estimate and prediction results. It could provide causal insights on the data, while the forecast outcome could serve as a quantitative metric for causal graph estimation.
- We conduct extensive numerical experiments to demonstrate the effectiveness of the proposed method. We apply the proposed method to real-world datasets, which shows substantial improvement on prediction accuracy compared to benchmark methods.

4.2 Related work

Recently, there are many efforts to employ the causal mechanism into graph neural network models. It could be categorized into two approaches: attention-based methods and causal graph based methods.

The first line of approach utilizes the attention mechanisms to capture the underlying causal features, without estimating a causal graph. For instance, CausalGNN (Wang et al. 2022), an attention-based dynamic GNN model for spatial-temporal epidemic forecasting, adds a causal module into the framework via ordinary differential equations. Based on Susceptible-Infected-Removed (SIR) model (Loli Piccolomini and Zama 2020), the four states (susceptible, infected , recovered and dead) are concatenated into the feature matrix and then encoded using multi-layer perceptions. This model is specifically designed for epidemic forecasting, which is hard to generalize to other scenarios. Being able for general applications, Causal attention learning (CAL) (Sui et al. 2022) uses attention modules to discover the underlying causal patterns and improves the prediction accuracy. Features are classified into causal and shortcut features, by the learned attention weights, where no causal discovery assumption or

methods are implemented.

The second line of approach, incorporates causal graph estimation into the modeling process. To conduct feature selection, Sizochenko et al. (2016) utilize the causal structure discovery to provide physical insight for investigating the biological activity of various compounds. Based on the causal graph, they extracted two features that are directly linked to the outcome variable and constructed models respectively. This feature extraction approach only preserves a small portion of the causal relationships in the causal graph, while omitting the majority of the causal mechanism. Instead, to preserve more information, Causal Temporal Graph Convolutional Neural Networks (CTGCN) (Langbridge et al. 2023), employ the entire estimated causal graph as the adjacency matrix. This approach tackles the challenge that the structural information between nodes may not available, but ignores the potential noise in features.

4.3 Proposed methodology

4.3.1 Graph-based causal structure discovery

Based on the linear structural equation model, the structural vector autoregressions model (SVAR) consider both time-lagged and contemporaneous causal relationships. Assuming different samples are independent and won't affect each other, the SVAR model (Demiralp and Hoover 2003) could be expressed as follows:

$$\mathbf{X} = \mathbf{X}\mathbf{B} + \mathbf{Y}_1\mathbf{W}_1 + \dots + \mathbf{Y}_d\mathbf{W}_d + \mathbf{E}, \quad (4.1)$$

where $\mathbf{X} \in \mathbb{R}^{m \times p}$ is a data matrix for m observation and p variables, matrices $\mathbf{Y}_1, \dots, \mathbf{Y}_d$ are the respective lagged data matrix, $\mathbf{B}, \mathbf{W}_1, \dots, \mathbf{W}_d \in \mathbb{R}^{p \times p}$ are adjacency matrices of the DAG that characterizing the causal relationship of \mathbf{X} , d denotes the degree of the SVAR model and $\mathbf{E} \in \mathbb{R}^{m \times p}$ is the noise matrix.

Let $\mathbf{Y} = [\mathbf{Y}_1 | \dots | \mathbf{Y}_d] \in \mathbb{R}^{m \times pd}$ be concatenated matrix for the lagged data, and let $\mathbf{W} = [\mathbf{W}_1^T | \dots | \mathbf{W}_d^T]^T \in \mathbb{R}^{pd \times p}$ denote the weights for the across-time dependencies. The Equation 4.1 could be written in the following compact form:

$$\mathbf{X} = \mathbf{X}\mathbf{B} + \mathbf{Y}\mathbf{W} + \mathbf{E}.$$

This structural vector autoregressions model (SVAR) has make the i.i.d. assumption, assuming each sample is identically and independently distributed (Spirtes et al. 2000b). However, in real-world scenarios, samples may have associations and the data generation process may be affected by other samples. For instance, when writing a review, a customers' review may be

affected by previous reviews that he has seen (Fan et al. 2023).

To take the samples' interaction into consideration, we incorporate the adjacency matrix, which specifies the relationship between samples, into the models:

$$\mathbf{X} = \mathbf{X}\mathbf{B} + \mathbf{A}\mathbf{Y}\mathbf{W} + \mathbf{E}, \quad (4.2)$$

where $\mathbf{X} \in \mathbb{R}^{m \times p}$ is a data matrix for m observation and p variables, $\mathbf{B} \in \mathbb{R}^{p \times p}$ represent weighted adjacency matrix of the DAG that characterizing the causal relationship of the p variables, $\mathbf{Y} = [\mathbf{Y}_1 | \dots | \mathbf{Y}_d] \in \mathbb{R}^{m \times pd}$ be concatenated matrix for the lagged data, and $\mathbf{W} = [\mathbf{W}_1^T | \dots | \mathbf{W}_d^T]^T \in \mathbb{R}^{pd \times p}$ denote the weights for the across-time dependencies, $\mathbf{A} \in \mathbb{R}^{m \times m}$ denotes the normalized adjacency matrix of the m observations and $\mathbf{E} \in \mathbb{R}^{m \times p}$ is the noise matrix.

Since the future data won't affect the past observations, the edges in \mathbf{W} only go forward in time, which guarantees the acyclicity of the matrix \mathbf{W} . Thus, it suffices to require that \mathbf{B} is acyclic to ensure the whole causal network is acyclic. To ensure the acyclicity of the \mathbf{B} , we impose the acyclicity constraint (Zheng et al. 2018):

$$h(\mathbf{B}) = \text{trace}(e^{\mathbf{B} \circ \mathbf{B}} - p) = 0$$

Thus, we solve the following regularized optimization problem, to estimate the weighted adjacency matrices \mathbf{W} and \mathbf{B} , given the data matrices \mathbf{X} and \mathbf{Y} :

$$\begin{aligned} \min_{\mathbf{B}, \mathbf{W}} f(\mathbf{B}, \mathbf{W}) &= \|\mathbf{X} - \mathbf{X}\mathbf{B} - \mathbf{A}\mathbf{Y}\mathbf{W}\|^2 + \lambda_{\mathbf{W}} \|\mathbf{B}\|_1 + \lambda_{\mathbf{W}} \|\mathbf{B}\|_1, \\ \text{s.t. } h(\mathbf{B}) &= 0 \end{aligned}$$

where $\lambda_{\mathbf{W}}, \lambda_{\mathbf{A}}$ denotes the regularization weights for the weight matrices, $\|\cdot\|_1$ denotes the element-wise ℓ_1 norm.

4.3.2 Causality guided graph neural network

To reduce spurious variables' effect, previous works have shown that removing the spurious variables may lead to performance improvement (Sizochenko et al. 2016). However, only keeping variables that are directed linked to the outcome variable may omit too much information, since other variables may have an indirect effect may also be valuable in prediction. To preserve more information from the causal graph, we propose the following attention mechanism to

adaptively learn the weights for each feature:

$$\mathbf{M}_c, \mathbf{M}_t = \sigma(\text{MLP}(\hat{\mathbf{B}})),$$

where $\mathbf{M}_c, \mathbf{M}_t \in \mathbb{R}^{p \times p}$ denotes the mask matrix for the causal features and spurious features, σ denotes the sigmoid function, MLP denotes a fully connected layer, and $\hat{\mathbf{B}}$ is the estimated causal graph. Note that, the $\mathbf{M}_c, \mathbf{M}_t$ satisfy the equation that

$$\mathbf{M}_{c,ij} + \mathbf{M}_{t,ij} = 1, \quad \text{for } i, j = 1, \dots, p,$$

where $\mathbf{M}_{c,ij}, \mathbf{M}_{t,ij}$ denotes the i, j th element of $\mathbf{M}_c, \mathbf{M}_t$ respectively. By doing so, the spurious feature may have large large weights in the \mathbf{M}_t , while having little weights in \mathbf{M}_c , while the causal features may have opposite pattern.

With the casual-graph based mask matrix, we utilize two separate graph neural network layers to obtain representations of the spurious features and the other features (Sui et al. 2022):

$$\begin{aligned} h_c &= f(\text{GConv}(\mathbf{A}, \mathbf{X} \odot \mathbf{M}_c)), z_c = \phi(h_c), \\ h_t &= f(\text{GConv}(\mathbf{A}, \mathbf{X} \odot \mathbf{M}_t)), z_t = \phi(h_t), \end{aligned}$$

where h_c, h_t denotes the representation utilizing causal features and spurious features respectively, \odot denotes the element-wise product, GConv denotes the graph convolutional layer, f, ϕ are fully connected layers, m is the sample size and z_c, z_t denotes the predicted outcome utilizing causal features and spurious features respectively.

To disentangle the effect of spurious features and causal features on the prediction, we apply different supervised classification loss functions on the predicted outcomes:

$$\begin{aligned} L_c &= -\frac{1}{m} \sum y^T \log(z_c), \\ L_t &= -\frac{1}{m} \sum KL(y_{unif}, z_t) \end{aligned}$$

where we utilize the actual label y to obtain the classification loss for the causal features, while utilize uniform labels y_{unif} to generate the KL divergence for the spurious features, since the spurious features may not be necessary for the prediction task.

Following Sui et al. (2022)’s approach, we also define the intervene loss as follows:

$$z_i = \phi(h_c + h_t),$$

$$L_i = -\frac{1}{m} \sum y^T \log(z_i),$$

where z_i represents the prediction from intervened features, where utilizing features from both spurious and causal sides. Combining losses of all aspects, the loss function aim to optimize is as follows:

$$L = L_c + \lambda_t L_t + \lambda_i L_i,$$

where λ_t, λ_i represents hyper-parameters that determine the strength of disentanglement and intervention.

4.4 Simulation study

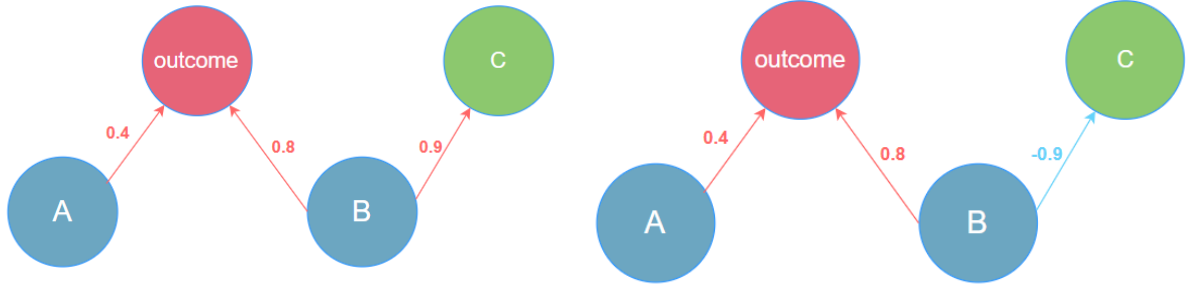
4.4.1 Synthetic data generation

We generate the training data based on the following SVAR model:

$$\mathbf{X} = \mathbf{X}\mathbf{B} + \mathbf{A}\mathbf{Y}\mathbf{W} + \mathbf{E}, \quad (4.3)$$

where \mathbf{X} is a data matrix for 50 observations ($m = 50$) observation with 10/15 variables ($p = 10/15$), \mathbf{Y} is the lag-5 data matrix ($d = 5$) and \mathbf{E} is the noise matrix $\mathbf{E} \sim \exp(1)$. The adjacency matrix A is randomly generated with density 0.1, and the weighted matrix W is generated using Erdős–Rényi(ER) (Newman et al. 2001) model with 5 expected edges, where the weights are randomly sampled from $(0.5, 1) \cup (-1, -0.5)$. The outcome variable is transformed into a binary variable by setting a threshold value, which equals to the sample mean.

After generating the weighted matrix W , we mutate it so that it could contain spurious variable, i.e. it has no outgoing edges and has a parent which also have direct link with the target variable. For instance, in Figure 4.2, the variable C is a spurious variable. To illustrate the negative effect of relying on the spurious variable to make predictions, we create an out-of-distribution setup for the training and testing data. Specifically, let w_t, w_s denotes the causal strength between the spurious variable and its parent node in the training and testing data respectively, it satisfies $w_t = w_s * \gamma_s$, where γ_s denotes the spurious scale, representing the different spurious effects in the dataset. As demonstrated in Figure 4.2, the causal strength between variable B and C has been reversed in the testing data, and $\gamma_s = -1$.



(a) Causal graph to generate the training data. (b) Causal graph to generate the testing data.

Figure 4.2: Different causal graphs used in the data generation process, when $\gamma_s = -1$.

4.4.2 Benchmark methods

To evaluate the performance of the proposed method, we evaluated the prediction results using accuracy. We adopt the following graph neural network methods as baselines: graph convolutional network (GCN) (Kipf and Welling 2017), and graph attention networks (GAT) (Veličković et al. 2018). The proposed method is based on these baseline methods and applied the respective graph convolutional layers. To evaluate the graph inputs' effect, we also evaluate the performance of the proposed method, when the graph input is randomly generated. The implementation details is summarized as follows:

- The computation is done by the processor Intel(R) Core(TM) i7-8550U CPU and the results are evaluated based on 50 realizations.
- GCN (Kipf and Welling 2017): Graph convolutional network (GCN) is a widely-used graph neural network model and could encode both local graph structure and node features by message passing and information propagation. The number of layer is 2 and the number of hidden nodes is 4,8 when the number of variables $d = 10, 15$ respectively. The rectified linear unit function (ReLU) is used as the activation function. The learning rate is 0.01, number of epoch is 300 and the algorithm is early stopped if there is no improvement in the recent 50 epoches. The pytorch implementation of GCN is available at the repository <https://github.com/kipf/pygcn>.
- GCN-proposed: The proposed method is implemented based on PyTorch (Paszke et al. 2017) using Adam (Kingma and Ba 2014) optimizer to optimize the loss function. The graph convolutional layer is based on the GCN model with the same hyper-parameters. We set the $\lambda_t = \lambda_i = 0.2, \lambda_B = \lambda_W = 0.01$ and use a 2-layer multi-layer perception to model the estimated causal graph, with the same number of hidden nodes and ReLU as the activation function.

- GCN-random: The graph input is generated using Erdős–Rényi(ER) Newman et al. (2001) model with 5 expected edges, where the weights are randomly sampled from $(0.5, 1) \cup (-1, -0.5)$. The method utilizes the randomly-generated graph as the graph input and utilize the same hyperparameters with the GCN-proposed method.
- GAT (Veličković et al. 2018): To consider various impacts of the neighbor nodes, Veličković et al. (2018) propose graph attention networks (GAT) and it is able to assign different weights to nodes in the same neighborhood. . The number of layer is 2, the number of heads is 2, the number of hidden nodes is 4,4 when the number of variables $d = 10, 15$ respectively. The exponential linear unit (ELU) (Clevert et al. 2016) function is used as the activation function. The learning rate is 0.005, number of epoch is 300 and the algorithm is early stopped if there is no improvement in the recent 30 epoches. The pytorch implementation of GAT is available at the repository <https://github.com/gordicaleksa/pytorch-GAT>.
- GAT-proposed: The proposed method is implemented based on PyTorch (Paszke et al. 2017) using Adam (Kingma and Ba 2014) optimizer to optimize the loss function. The graph convolutional layer is based on the GAT model with the same hyper-parameters. We set the $\lambda_t = \lambda_i = 0.2$, $\lambda_B = \lambda_W = 0.01$ and use a 2-layer multi-layer perception to model the estimated causal graph, with the same number of hidden nodes and ReLU as the activation function.
- GAT-random: The graph input is generated using Erdős–Rényi(ER) Newman et al. (2001) model with 5 expected edges, where the weights are randomly sampled from $(0.5, 1) \cup (-1, -0.5)$. The method utilizes the randomly-generated graph as the graph input and utilize the same hyperparameters with the GAT-proposed method.

4.4.3 Results

The results for the proposed method and the benchmark methods are shown in Figure 4.3,4.4 and Table 4.1,4.2. The findings are summarized as follows.

As shown in in Figure 4.3, in general, the proposed methods could improve the prediction accuracy, where GCN method benefits more from the proposed method. It may due to the fact that GCN treats all neighbors equally and might be more susceptible for spurious features. Also, when the spurious scale decreases, we could see that the performance of both benchmark methods and proposed methods have lifted, since the distributions of training and testing data become similar.

Figure 4.4 compares the performance of the proposed method, when the graph input is randomly generated and causally estimated. From this figure, we could see that the graph quality could affect the prediction accuracy, where the causally estimated graph could lift the prediction performance. Thus the prediction outcome may serve as a quantitative metric for causal graph estimation. The improvement is larger when there are more variables, since the true graph would be more likely to deviate from the randomly-generated graph.

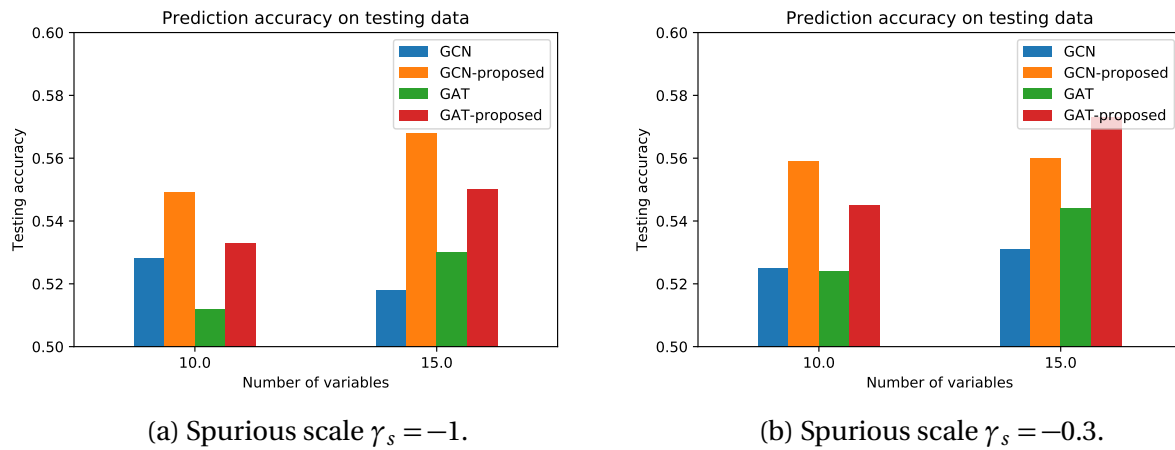
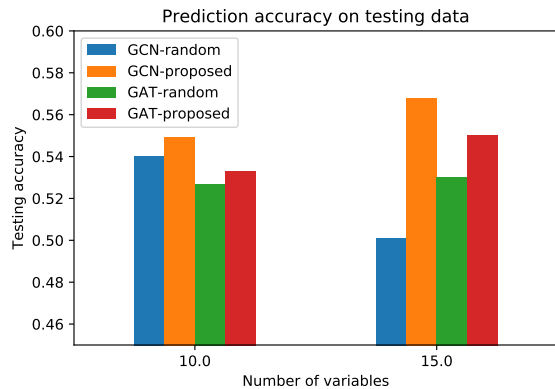
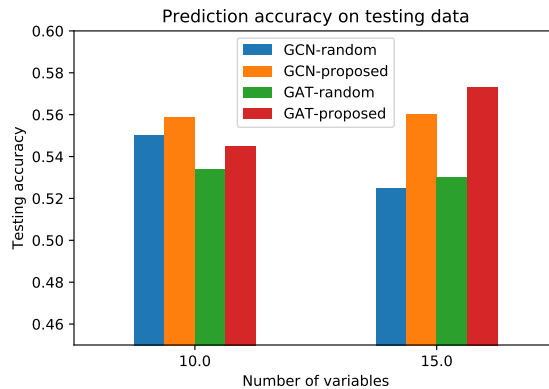


Figure 4.3: Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and spurious scale different γ_s .



(a) Spurious scale $\gamma_s = -1$.



(b) Spurious scale $\gamma_s = -0.3$.

Figure 4.4: Test accuracy for the proposed method when the graph input is randomly generated and causally estimated, with different number of variable d , and spurious scale different γ_s .

Table 4.1: Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and the spurious scale $\gamma_s = -1$.

d	Accuracy	Method
10	0.528(0.019)	GCN
10	0.540(0.019)	GCN-random
10	0.549 (0.020)	GCN-proposed
15	0.518 (0.019)	GCN
15	0.501 (0.018)	GCN-random
15	0.568 (0.019)	GCN-proposed
10	0.512(0.024)	GAT
10	0.527(0.018)	GAT-random
10	0.533 (0.023)	GAT-proposed
15	0.530 (0.020)	GAT
15	0.530 (0.020)	GAT-random
15	0.550 (0.020)	GAT-proposed

Table 4.2: Test accuracy for the benchmark methods and the proposed method, when the data is generated with different number of variable d , and the spurious scale $\gamma_s = -0.3$.

d	Accuracy	Method
10	0.525(0.019)	GCN
10	0.550(0.019)	GCN-random
10	0.559 (0.020)	GCN-proposed
15	0.531 (0.020)	GCN
15	0.525 (0.017)	GCN
15	0.560 (0.018)	GCN-proposed
10	0.524(0.024)	GAT
10	0.534(0.024)	GAT-random
10	0.545 (0.019)	GAT-proposed
15	0.544 (0.018)	GAT
15	0.530 (0.018)	GAT-random
15	0.573 (0.018)	GAT-proposed

4.5 Real data analysis

Characterizing the citation relationships between the scientific publications from Pubmed database, the Pubmed (Sen et al. 2008) dataset consists of 19,717 academic papers related to type-1 and type-2 diabetes. It contains the structural information, where each node is a publication which are connected if they are cited by other papers. And the node feature is the 500-dimension long vector, representing the existence of the word for vocabularies in the pre-specified word directories. To reduce the sparsity in the dataset, we remove the features that have a missing rate less than 80% and filter out the publications that has a feature coverage less then 60%. We then have a dataset with $m = 997$ publications and $p = 58$ binary features representing word existence, where each publication has a binary label of whether it is about the type-2 diabetes. Figure 4.5 shows the citation graph ¹ for the data and Figure 4.6 shows a snapshot of how data looks like. The data is then randomly split into training, validation, and testing sets with a ratio of 4:4:2.

¹For the presentation clarity, nodes without any links are excluded from the graph, although they remain included in the experimental setup.

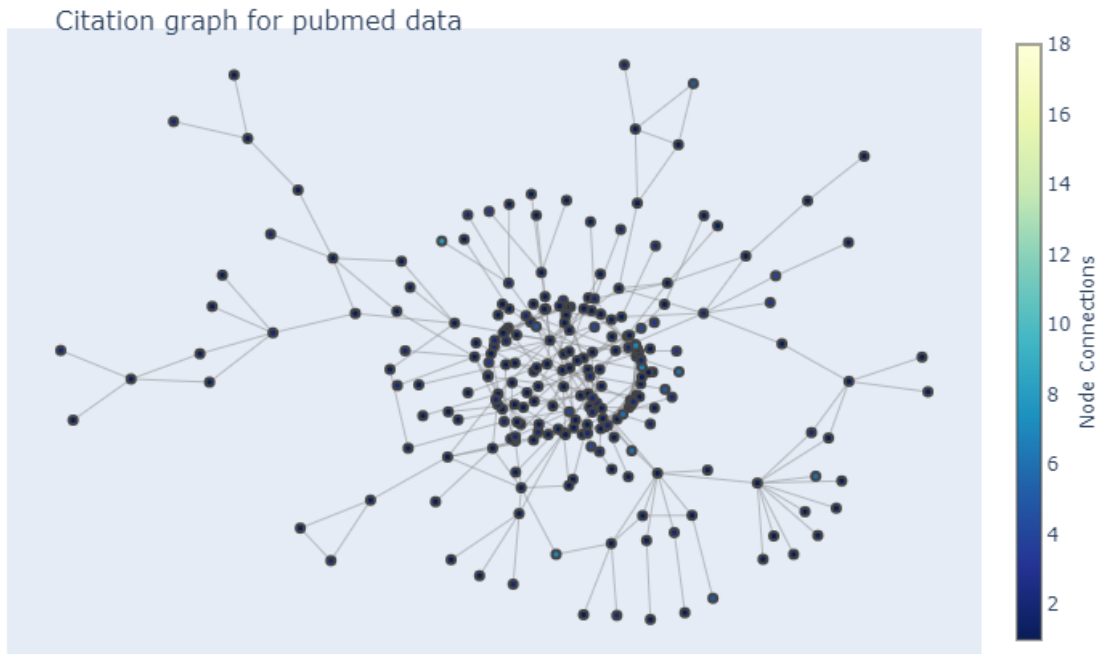


Figure 4.5: Citation graph for pubmed data. Nodes represent the scientific publications and are linked if there are citation relationships. Darker nodes indicates the paper has more citations.

	w- studi	w- 2	w- 1	w- type	w- result	w- control	w- increas	w- signific	w- 0	w- patient	...	w- chang	w- dure	w- observ	w- 10	w- higher	w- 9	w- data	w- rat	w- howev	label
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1
1	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	...	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1
2	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	...	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1
4	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1

Figure 4.6: Snapshot of the Pubmed data, where each column is a binary matrix indicating the existence of a word. The last column is the binary label of whether the paper is about the type-2 diabetes.

By applying the proposed method to the processed-data, we could estimate the causal graph ², as shown in Figure 4.7. Figure 4.8 shows a subgraph focusing on the outcome variable. From the visualization, we could observe that, to classify the type of the diabetes disease, the words "type" and "2" and useful. Moreover, since the type 2 diabetes is characterised by

²Since there are no time-series data from the citation network, we utilize the NOTEARS approach to generate the causal graph estimates, where the threshold is set at 0.2.

pancreatic β -cell dysfunction which could be caused by physical inactivity (Chatterjee et al. 2017), the term "active" and "cell" could contribute to the classification. However, there are also potential spurious variable, such as "patient" and "conclusion". While these words may follow the word "type", they have no path connecting them to the outcome variable.

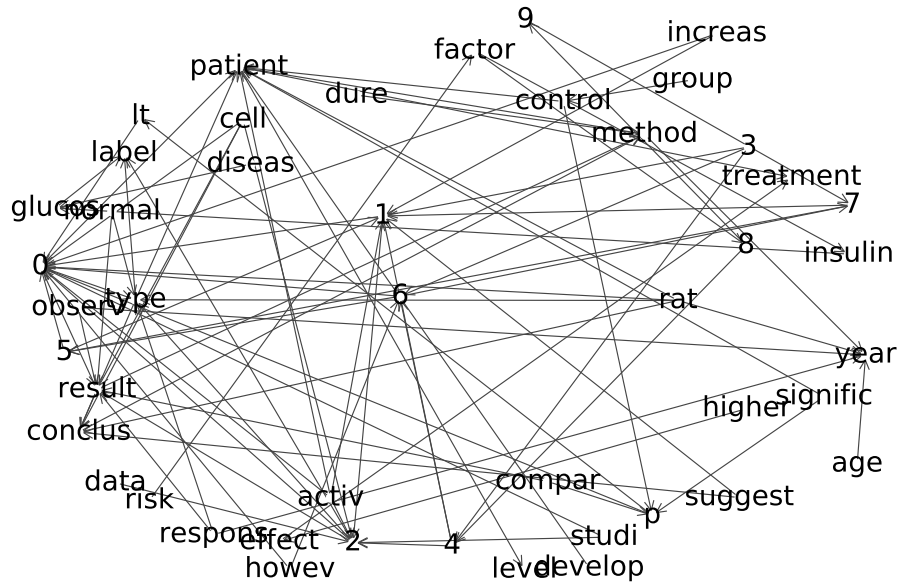


Figure 4.7: Estimated causal graph based on the Pubmed data. The text represents the root words, processed to minimize variations in word forms.

To further validate the effectiveness of the proposed method, we apply the graph convolutional network (GCN)(Kipf and Welling 2017) and compare its performance with our proposed method in the prediction task. As shown in Figure 4.9, the proposed method could lift the accuracy to 83%, while the GCN achieves an accuracy of 75%, slightly better than the actual class proportion (69%). Thus, we could conclude that, by leveraging the causal information, the proposed method could enhance the prediction performance.

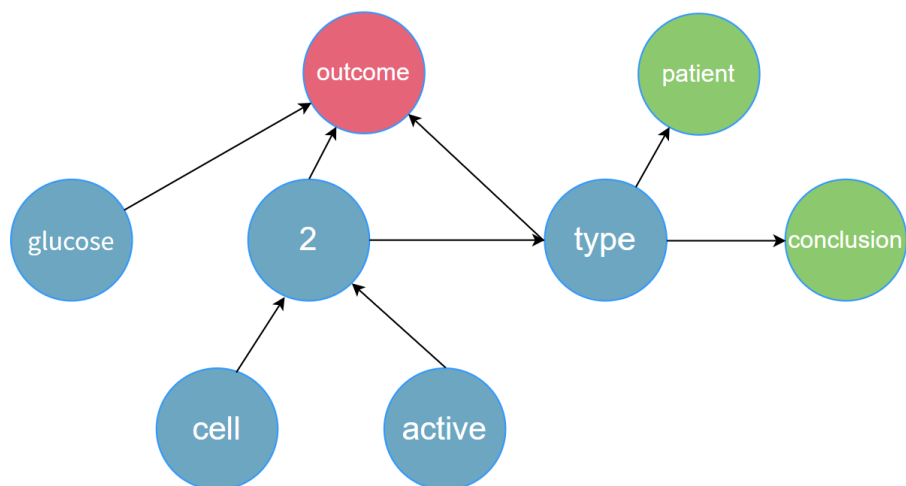


Figure 4.8: A subgraph of the estimated causal graph focusing on the outcome variable. The red node denotes the outcome variable, and the green node denotes the potential spurious variable.

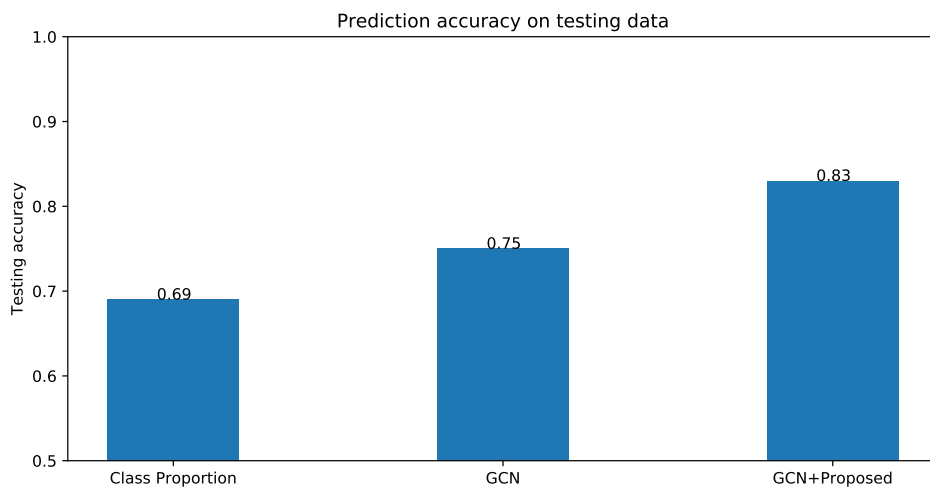


Figure 4.9: Prediction accuracy on testing data.

4.6 Discussion

In this chapter, we develop a causality-guided graph neural network approach, which incorporates the causal structure into graph neural network modeling, disentangling the spurious variable and causal features. To conclude, we briefly discuss some limitations and possible

future directions.

Firstly, in the current approach, we first apply the graph-based causal structure learning approach to obtain an estimate of the underlying causal graph and then utilize it in the later graph neural network step. It may be better to develop the approach in an end-to-end manner, where we simultaneously generate the causal graph estimate and the prediction. More efforts could be taken to add a causal module into the graph neural network models and modify the loss function accordingly.

Secondly, although taking the time-lagged relationship into consideration, the current approach has assumed a static causal structure, where the underlying causal graph won't change with over. As discussed in the previous chapter, allowing a time-varying causal structure may be worth exploring.

Thirdly, how to construct the graph is essential for the graph neural network and could affect the model's performance. In this work, efforts have been taken to use the causal information to disentangle spurious features, while there may still be noises in the given graph data. It could be a valuable direction to apply the causal structure to construct a graph representing node relationships instead of feature relationships.

REFERENCES

- Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.
- Aragam, B., Amini, A. A., and Zhou, Q. (2015). Learning directed acyclic graphs with penalized neighbourhood regression. *arXiv preprint arXiv:1511.08963*.
- Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.
- Boruvka, A., Almirall, D., Witkiewitz, K., and Murphy, S. A. (2018). Assessing time-varying causal effect moderation in mobile health. *Journal of the American Statistical Association*, 113(523):1112–1121.
- Cai, H., Song, R., and Lu, W. (2020). Anoce: Analysis of causal effects with multiple mediators via constrained structural learning. In *International Conference on Learning Representations*.
- Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.
- Chatterjee, S., Khunti, K., and Davies, M. J. (2017). Type 2 diabetes. *The lancet*, 389(10085):2239–2251.
- Chen, Y., Wei, Z., and Huang, X. (2018). Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1655–1658.
- Cheng, D., Li, J., Liu, L., Liu, J., and Le, T. D. (2024). Data-driven causal effect estimation based on graphical causal modelling: A survey. *ACM Computing Surveys*, 56(5):1–37.
- Cheng, D., Niu, Z., and Zhang, Y. (2020a). Contagious chain risk rating for networked-guarantee loans. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2715–2723.
- Cheng, D., Tu, Y., Ma, Z., Niu, Z., and Zhang, L. (2019). Risk assessment for networked-guarantee loans using high-order graph attention representation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5822–5828. International Joint Conferences on Artificial Intelligence Organization.
- Cheng, D., Wang, X., Zhang, Y., and Zhang, L. (2020b). Risk guarantee prediction in networked-loans. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4483–4489. International Joint Conferences on Artificial Intelligence Organization. Special Track on AI in FinTech.

- Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554.
- Chickering, M., Heckerman, D., and Meek, C. (2004). Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5:1287–1330.
- Chinazzi, M., Davis, J. T., Ajelli, M., Gioannini, C., Litvinova, M., Merler, S., Pastore y Piontti, A., Mu, K., Rossi, L., Sun, K., et al. (2020). The effect of travel restrictions on the spread of the 2019 novel coronavirus (covid-19) outbreak. *Science*, 368(6489):395–400.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.
- Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). arxiv 2015. *arXiv preprint arXiv:1511.07289*, 2.
- Colombo, D., Maathuis, M. H., et al. (2014). Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782.
- Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321.
- Comon, P. (1992). Independent component analysis.
- Delyon, B., Lavielle, M., and Moulines, E. (1999). Convergence of a stochastic approximation version of the em algorithm. *Annals of statistics*, pages 94–128.
- Demiralp, S. and Hoover, K. D. (2003). Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and statistics*, 65:745–767.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., and Yu, P. S. (2020). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 315–324.
- Drton, M., Foygel, R., and Sullivant, S. (2011). Global identifiability of linear structural equation models. *The Annals of Statistics*, 39(2):865–886.
- Du, S., Song, G., and Hong, H. (2019). Collective causal inference with lag estimation. *Neuro-computing*, 323:299–310.
- Edwards, M. C., Meyer, R., and Christensen, N. (2019). Bayesian nonparametric spectral density estimation using b-spline priors. *Statistics and Computing*, 29(1):67–78.

- Fan, J. and Zhang, W. (2008). Statistical methods with varying coefficient models. *Statistics and its Interface*, 1(1):179.
- Fan, S., Zhang, S., Wang, X., and Shi, C. (2023). Directed acyclic graph structure learning from dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7512–7521.
- Feng, F., He, X., Wang, X., Luo, C., Liu, Y., and Chua, T.-S. (2019). Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2):1–30.
- Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877.
- Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.
- Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.
- Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438.
- Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.
- Harl, M., Weinzierl, S., Stierle, M., and Matzner, M. (2020). Explainable predictive business process monitoring using gated graph neural networks. *Journal of Decision Systems*, pages 1–16.
- Hastie, T. and Tibshirani, R. (1993). Varying-coefficient models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(4):757–779.
- Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243.
- Heinze-Deml, C., Maathuis, M. H., and Meinshausen, N. (2018). Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hu, B., Zhang, Z., Shi, C., Zhou, J., Li, X., and Qi, Y. (2019). Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 946–953.

- Hu, B., Zhang, Z., Zhou, J., Fang, J., Jia, Q., Fang, Y., Yu, Q., and Qi, Y. (2020). Loan default analysis with multiplex graph learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2525–2532.
- Huang, B., Zhang, K., Gong, M., and Glymour, C. (2019). Causal discovery and forecasting in nonstationary environments with state-space models. In *International conference on machine learning*, pages 2901–2910. PMLR.
- Huang, J., Chai, J., and Cho, S. (2020). Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14:1–24.
- Huang, J. Z., Wu, C. O., and Zhou, L. (2002). Varying-coefficient models and basis function approximations for the analysis of repeated measurements. *Biometrika*, 89(1):111–128.
- Inc., T. M. (2022). Matlab version: 9.13.0 (r2022b).
- Jiang, J., Chen, J., Gu, T., Choo, K.-K. R., Liu, C., Yu, M., Huang, W., and Mohapatra, P. (2019). Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 109–114. IEEE.
- Jiang, W. (2021). Applications of deep learning in stock market prediction: recent progress. *Expert Systems with Applications*, page 115537.
- Kalisch, M. and Bühlman, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3).
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.
- Knyazev, B., Taylor, G. W., and Amer, M. (2019). Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32.
- Kudo, W., Nishiguchi, M., and Toriumi, F. (2020). Gcnext: graph convolutional network with expanded balance theory for fraudulent user detection. *Social Network Analysis and Mining*, 10(1):1–12.
- Kurshan, E. and Shen, H. (2020). Graph computing for financial crime and fraud detection: Trends, challenges and outlook. *International Journal of Semantic Computing*, 14(04):565–589.
- Langbridge, A., O’Donncha, F., Ba, A., Lorenzi, F., Lohse, C., and Ploennigs, J. (2023). Causal temporal graph convolutional neural networks (ctgcn). *arXiv preprint arXiv:2303.09634*.

- Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., and Koh, E. (2019). Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6):1–25.
- Lee, J. W., Lee, W. K., and Sohn, S. Y. (2021). Graph convolutional network-based credit default prediction utilizing three types of virtual distances among borrowers. *Expert Systems with Applications*, 168:114411.
- Li, A., Qin, Z., Liu, R., Yang, Y., and Li, D. (2019a). Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711.
- Li, C., Chen, L. J., Chen, X., Zhang, M., Pang, C. P., and Chen, H. (2020a). Retrospective analysis of the possibility of predicting the covid-19 outbreak from internet searches and social media data, china, 2020. *Eurosurveillance*, 25(10):2000199.
- Li, C., Jia, K., Shen, D., Shi, C.-J. R., and Yang, H. (2019b). Hierarchical representation learning for bipartite graphs. In *IJCAI*, pages 2873–2879.
- Li, W., Bao, R., Harimoto, K., Chen, D., Xu, J., and Su, Q. (2020b). Modeling the stock relation with graph network for overnight stock movement prediction. In *no. CONF*, pages 4541–4547.
- Li, X., Liu, S., Li, Z., Han, X., Shi, C., Hooi, B., Huang, H., and Cheng, X. (2020c). Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4731–4738.
- Li, X., Saúde, J., Reddy, P., and Veloso, M. (2019c). Classifying and understanding financial data using graph neural network. In *AAAI Workshop on Knowledge Discovery from Unstructured Data in Financial Services 2020*.
- Li, Y., Zemel, R., Brockschmidt, M., and Tarlow, D. (2016). Gated graph sequence neural networks. In *Proceedings of ICLR'16*.
- Li, Z., Shen, X., Jiao, Y., Pan, X., Zou, P., Meng, X., Yao, C., and Bu, J. (2020d). Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1677–1688. IEEE.
- Liang, C., Liu, Z., Liu, B., Zhou, J., Li, X., Yang, S., and Qi, Y. (2019). Uncovering insurance fraud conspiracy with network learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1181–1184.
- Liang, T., Zeng, G., Zhong, Q., Chi, J., Feng, J., Ao, X., and Tang, J. (2021). 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.
- Liou, Y.-T., Chen, C.-C., Tang, T.-H., Huang, H.-H., and Chen, H.-H. (2021). Finsense: An assistant system for financial journalists and investors. In *Proceedings of the 14th International Conference on Web Search and Data Mining*.

- Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., and Qi, Y. (2019). Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431.
- Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., and Song, L. (2018). Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2077–2085.
- Loli Piccolomini, E. and Zama, F. (2020). Monitoring italian covid-19 spread by a forced seird model. *PloS one*, 15(8):e0237417.
- Löwe, S., Madras, D., Zemel, R., and Welling, M. (2022). Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Conference on Causal Learning and Reasoning*, pages 509–525. PMLR.
- Lv, L., Cheng, J., Peng, N., Fan, M., Zhao, D., and Zhang, J. (2019). Auto-encoder based graph convolutional networks for online financial anti-fraud. In *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 1–6. IEEE.
- Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., and Niu, X. (2018). Study on a prediction of p2p network loan default based on the machine learning lightgbm and xgboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, 31:24–39.
- Matsunaga, D., Suzumura, T., and Takahashi, T. (2019). Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660*.
- Newman, M. E., Strogatz, S. H., and Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications. *Physical review E*, 64(2):026118.
- Nie, S., Mauá, D. D., De Campos, C. P., and Ji, Q. (2014). Advances in learning bayesian networks of bounded treewidth. *Advances in neural information processing systems*, 27.
- Ozbayoglu, A. M., Gudelek, M. U., and Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*, page 106384.
- Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Georgatzis, K., Beaumont, P., and Aragam, B. (2020). Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. PMLR.
- Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., and Leiserson, C. (2020). Evolvegn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5363–5370.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.
- Pearl, J. (2009). Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146.

- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Ramsey, J., Glymour, M., Sanchez-Romero, R., and Glymour, C. (2017). A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics*, 3(2):121–129.
- Rao, S., Zhang, S., and Zhang, C. (2021). Suspicious massive registration detection via dynamic heterogeneous graph neural networks. In *AAAI Workshop on Deep Learning on Graphs 2021*.
- Rao, S. X., Zhang, S., Han, Z., Zhang, Z., Min, W., Chen, Z., Shan, Y., Zhao, Y., and Zhang, C. (2020). xfraud: Explainable fraud transaction detection on heterogeneous graphs. *arXiv preprint arXiv:2011.12193*.
- Robins, J. M. (1994). Correcting for non-compliance in randomized trials using structural nested mean models. *Communications in Statistics-Theory and methods*, 23(8):2379–2412.
- Sawhney, R., Agarwal, S., Wadhwa, A., and Shah, R. (2020a). 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.
- Sawhney, R., Khanna, P., Aggarwal, A., Jain, T., Mathur, P., and Shah, R. (2020b). Voltage: Volatility forecasting via text-audio fusion with graph convolution networks for earnings calls. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8001–8013.
- Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. (2015). Learning bayesian networks with thousands of variables. *Advances in neural information processing systems*, 28.
- Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, pages 461–464.
- Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.
- Shen, X., Ma, S., Vemuri, P., and Simon, G. (2020). Challenges and opportunities with causal discovery algorithms: application to alzheimer’s pathophysiology. *Scientific reports*, 10(1):2975.
- Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A., and Jordan, M. (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10).
- Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., and Bollen, K. (2011). Directlingam: A direct method for learning a linear non-gaussian structural equation model. *The Journal of Machine Learning Research*, 12:1225–1248.

- Sizochenko, N., Gajewicz, A., Leszczynski, J., and Puzyn, T. (2016). Causation or only correlation? application of causal inference graphs for evaluating causality in nano-qsar models. *Nanoscale*, 8(13):7203–7208.
- Spirtes, P., Glymour, C., Scheines, R., Kauffman, S., Aimale, V., and Wimberly, F. (2000a). Constructing bayesian network models of gene expression networks from microarray data.
- Spirtes, P., Glymour, C. N., and Scheines, R. (2000b). *Causation, prediction, and search*. MIT press.
- Steiger, E., Mussgnug, T., and Kroll, L. E. (2021). Causal graph analysis of covid-19 observational data in german districts reveals effects of determining factors on reported case numbers. *PloS one*, 16(5):e0237277.
- Sui, Y., Wang, X., Wu, J., Lin, M., He, X., and Chua, T.-S. (2022). Causal attention for interpretable and generalizable graph classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1696–1705.
- Turiel, J. and Aste, T. (2020). Peer-to-peer loan acceptance and default prediction with artificial intelligence. *Royal Society open science*, 7(6):191649.
- VanderWeele, T. J. and Robins, J. M. (2007). Four types of effect modification: a classification based on directed acyclic graphs. *Epidemiology*, 18(5):561–568.
- Vansteelandt, S. and Joffe, M. (2014). Structural nested models and g-estimation: the partially realized promise. *Statistical Science*, 29(4):707–731.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.
- Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.
- Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S., and Qi, Y. (2019). A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607. IEEE Computer Society.
- Wang, L., Adiga, A., Chen, J., Sadilek, A., Venkatramanan, S., and Marathe, M. (2022). Causal-gnn: Causal-based graph neural networks for spatio-temporal epidemic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 12191–12199.
- Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., and Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.
- Wu, L., Yang, S., Reich, B. J., and Rappold, A. G. (2020a). Estimating spatially varying health effects in app-based citizen science research. *arXiv preprint arXiv:2005.12017*.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020b). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.

- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020c). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.
- Xu, B., Shen, H., Sun, B., An, R., Cao, Q., and Cheng, X. (2021). Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4537–4545.
- Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? In *International Conference on Learning Representations*.
- Yang, H. (2019). Aligraph: A comprehensive graph neural network platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3165–3166.
- Yang, S., Zhang, Z., Zhou, J., Wang, Y., Sun, W., Zhong, X., Fang, Y., Yu, Q., and Qi, Y. (2020). Financial risk analysis for smes with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4661–4667.
- Yang, Y., Wei, Z., Chen, Q., and Wu, L. (2019). Using external knowledge for financial event prediction based on graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2161–2164.
- Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894.
- Ying, X., Xu, C., Gao, J., Wang, J., and Li, Z. (2020). Time-aware graph relational attention network for stock recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2281–2284.
- Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR.
- Zhang, J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896.
- Zhang, K., Huang, B., Zhang, J., Glymour, C., and Schölkopf, B. (2017a). Causal discovery from nonstationary/heterogeneous data: Skeleton estimation and orientation determination. In *IJCAI: Proceedings of the Conference*, volume 2017, page 1347. NIH Public Access.
- Zhang, M. and Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.
- Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.
- Zhang, S., Yin, H., Chen, T., Hung, Q. V. N., Huang, Z., and Cui, L. (2020). Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 689–698.

- Zhang, S., Zhou, D., Yildirim, M. Y., Alcorn, S., He, J., Davulcu, H., and Tong, H. (2017b). Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 570–578. SIAM.
- Zhao, T., Deng, C., Yu, K., Jiang, T., Wang, D., and Jiang, M. (2021). Gnn-based graph anomaly detection with graph anomaly loss. In *The Second International Workshop on Deep Learning on Graphs: Methods and Applications (DLG-KDD'20)*.
- Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31.
- Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.
- Zhu, Y.-N., Luo, X., Li, Y.-F., Bu, B., Zhou, K., Zhang, W., and Lu, M. (2020). Heterogeneous mini-graph neural network and its application to fraud invitation detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 891–899. IEEE.

APPENDICES

APPENDIX

A

APPENDIX FOR CHAPTER 1

A.1 Proof of Theorem 1.4.1

In this section, we present the detailed proof of Theorem 1.4.1.

With Equation (1.7) in Section 1.4.2, we could write:

$$\begin{cases} A_t &= \epsilon_{A_t} \\ M_t &= (I - C_t^T)^{-1} \{ \alpha_t^T A_t + \epsilon_{M_t}^T \} \\ Y_t &= \gamma_t A_t + \beta_t^T (I - C_t^T)^{-1} \{ \alpha_t^T A_t + \epsilon_{M_t}^T \} + \epsilon_{Y_t} \\ &= \{ \gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T \} A_t + \{ \beta_t^T (I - C_t^T)^{-1} \epsilon_{M_t}^T + \epsilon_{Y_t} \}, \end{cases} \quad (\text{A.1})$$

Following the results in Pearl (2009), we could have the following results:

- Under assumption A_1 , we have $E(Y_t | do(A_t = a)) = E(Y_t | A_t = a)$;
- Under assumption A_2 , we have $E(M_t | do(A_t = a)) = E(M_t | A_t = a)$;
- Under assumption A_3 , we have $E(Y_t | do(A_t = a, M_t = m)) = E(Y_t | A_t = a, M_t = m)$;

Based on Equation (A.1), Equation (1.8) and Assumptions A1-A3 in Section 1.4.2, we have

$$\begin{aligned}
DE_t &= E(Y_t | do(A_t = a + 1, M_t = m^{(a)})) - E(Y_t | do(A_t = a)) \\
&= E(Y_t | A_t = a + 1, M_t = m^{(a)}) - E(Y_t | A_t = a) \\
&= \{\gamma_t(a + 1) + \beta_t^T m^{(a)}\} - \{\gamma_t a + \beta_t^T m^{(a)}\} \\
&= \gamma_t, \\
IE_t &= E(Y_t | do(A_t = a, M_t = m^{(a+1)})) - E(Y_t | do(A_t = a)) \\
&= E(Y_t | A_t = a, M_t = m^{(a+1)}) - E(Y_t | A_t = a) \\
&= \{\gamma_t(a) + \beta_t^T m^{(a+1)}\} - \{\gamma_t a + \beta_t^T m^{(a)}\} \\
&= \beta_t^T (I - C_t^T)^{-1} \{\alpha_t^T (a + 1 - a)\}, \\
&= \beta_t^T (I - C_t^T)^{-1} \alpha_t^T, \\
TE_t &= E(Y_t | do(A_t = a + 1)) - E(Y_t | do(A_t = a)) \\
&= DE + IE \\
&= \gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T.
\end{aligned} \tag{A.2}$$

A.2 Corollary of Theorem 1.4.1

Corollary A.2.0.1. It is also worth mentioning that, the total effect is equivalent to the lag-1 effect proposed by (Boruvka et al. 2018) under consistency and sequential ignorability assumption:

$$\begin{aligned} \text{lag-1 effect} &= E[Y_t(\bar{A}_{t-1}, A_t = a + 1) | H_{t-1}] \\ &\quad - E[Y_t(\bar{A}_{t-1}, A_t = a) | H_{t-1}] \\ &= \gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T, \end{aligned}$$

where $Y_t(A_t)$ denotes the potential outcome, overbar denotes the history of the respective variables, e.g. $\bar{A}_t = (A_1, \dots, A_t)$ and $H_t = (\bar{A}_t, \bar{M}_t, \bar{Y}_t)$ denotes the information up to time t . The proof could be found in the supplementary materials.

Following the definition of Boruvka et al. (2018), under the following consistency and sequential ignorability assumption, we could have

$$\begin{aligned} \text{lag-1 effect} &= E[Y_t(\bar{A}_{t-1}, A_t = a + 1) | H_{t-1}] - E[Y_t(\bar{A}_{t-1}, A_t = a) | H_{t-1}] \\ &= E[Y_t(\bar{A}_{t-1}, A_t = a + 1) | H_{t-1}, A_t = a + 1] - E[Y_t(\bar{A}_{t-1}, A_t = a) | H_{t-1}, A_t = a] \\ &= E[Y_t(\bar{A}_{t-1}, A_t) | H_{t-1}, A_t = a + 1] - E[Y_t(\bar{A}_{t-1}, A_t) | H_{t-1}, A_t = a] \\ &= E(Y_t | A_t = a + 1, H_{t-1}) - E(Y_t | A_t = a, H_{t-1}) \\ &= E(\{\gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T\} A_t + \{\beta_t^T (I - C_t^T)^{-1} \epsilon_{M_t}^T + \epsilon_{Y_t}\} | A_t = a + 1, H_{t-1}) \quad (\text{A.3}) \\ &\quad - E(\{\gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T\} A_t + \{\beta_t^T (I - C_t^T)^{-1} \epsilon_{M_t}^T + \epsilon_{Y_t}\} | A_t = a, H_{t-1}) \\ &= \{\gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T\} (a + 1 - a) \\ &= \gamma_t + \beta_t^T (I - C_t^T)^{-1} \alpha_t^T \\ &= TE, \end{aligned}$$

where $Y_t(A_t)$ denotes the potential outcome, and overbar denotes the history of the respective variables, e.g. $\bar{A}_t = (A_1, \dots, A_t)$, $H_t = (\bar{A}_t, \bar{M}_t, \bar{Y}_t)$ denotes the information up to time t . The consistency assumption assumes that the observed data is equal to the potential outcomes as follows: $Y_t(\bar{A}_t) = Y_t$, for $t = 1, \dots, T$. The sequential ignorability assumes that the potential outcome $Y_t(\bar{A}_t)$ is independent of A_t given H_{t-1} for $t = 1, \dots, T$. The second equality is due to the sequential ignorability assumption, the fourth equality is due to the consistency assumption and the sixth equality is due to the no-unmeasured confounder assumption.

APPENDIX

B

APPENDIX FOR CHAPTER 2

B.1 Proof of Theorem 2.4.1

Based on the Equation (2.7), we could then write

$$\left\{ \begin{array}{l}
 A_t = \epsilon_{A_t} \\
 M_t = \boldsymbol{\alpha}_t^T A_t + \mathbf{C}_t^T M_t + \epsilon_{M_t}^T \\
 \quad + \sum_{i=1}^d \{ \boldsymbol{\alpha}_{t-i}^T A_{t-i} + \mathbf{C}_{t-i}^T M_{t-i} + \mathbf{d}_{t-i}^T Y_{t-i} \} \\
 = (\mathbf{I} - \mathbf{C}_t^T)^{-1} [\boldsymbol{\alpha}_t^T A_t + \epsilon_{M_t}^T \\
 \quad + \sum_{i=1}^d \{ \boldsymbol{\alpha}_{t-i}^T A_{t-i} + \mathbf{C}_{t-i}^T M_{t-i} + \mathbf{d}_{t-i}^T Y_{t-i} \}] \\
 Y_t = \gamma_t A_t + \boldsymbol{\beta}_t^T M_t + \epsilon_{Y_t} \\
 \quad + \sum_{i=1}^d \{ \gamma_{t-i} A_{t-i} + \boldsymbol{\beta}_{t-i}^T M_{t-i} + f_{t-i} Y_{t-i} \}
 \end{array} \right. \quad (\text{B.1})$$

Let overbar denotes the history of the respective variables, e.g. $\bar{A}_t = (A_1, \dots, A_t)$, $Y_t(\bar{a}_{t-1})$ denote the potential outcome at time t if the individual had the treatment history \bar{a}_{t-1} , and H_t denotes the history up to time t , $H_t = (\bar{M}_t, \bar{Y}_t, \bar{A}_t)$.

Under Assumptions in Section 2.4.1 and Equation (B.1), the dynamic causal effect could be written as

$$\begin{aligned}
& \mathbb{E}[Y_{t+1}^{(\bar{a}_t, a)} | H_t] - \mathbb{E}[Y_{t+1}^{(\bar{a}_t, 0)} | H_t] \\
&= \mathbb{E}[Y_{t+1}^{(\bar{a}_t, a)} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t] \\
&\quad - \mathbb{E}[Y_{t+1}^{(\bar{a}_t, 0)} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t] \\
&= \mathbb{E}[Y_{t+1}^{(\bar{a}_t, a)} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = a] \\
&\quad - \mathbb{E}[Y_{t+1}^{(\bar{a}_t, 0)} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = 0] \\
&= \mathbb{E}[Y_{t+1} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = a] \\
&\quad - \mathbb{E}[Y_{t+1} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = 0] \\
&= \mathbb{E}[\gamma_{t+1} A_{t+1} + \boldsymbol{\beta}_{t+1}^T M_{t+1} + \epsilon_{Y_{t+1}} \\
&\quad + \sum_{i=1}^d \{\gamma_{t+1-i} A_{t+1-i} + \boldsymbol{\beta}_{t+1-i}^T M_{t+1-i} + f_{t+1-i} Y_{t+1-i}\} \\
&\quad | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = a] \\
&\quad - \mathbb{E}[\gamma_{t+1} A_{t+1} + \boldsymbol{\beta}_{t+1}^T M_{t+1} + \epsilon_{Y_{t+1}} \\
&\quad + \sum_{i=1}^d \{\gamma_{t+1-i} A_{t+1-i} + \boldsymbol{\beta}_{t+1-i}^T M_{t+1-i} + f_{t+1-i} Y_{t+1-i}\} \\
&\quad | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = 0] \\
&= \mathbb{E}[\gamma_{t+1} A_{t+1} + \boldsymbol{\beta}_{t+1}^T M_{t+1} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = a] \\
&\quad - \mathbb{E}[\gamma_{t+1} A_{t+1} + \boldsymbol{\beta}_{t+1}^T M_{t+1} | \bar{M}_t = \bar{m}_t, \bar{Y}_t = \bar{y}_t, \bar{A}_t = \bar{a}_t, A_{t+1} = 0] \\
&= \mathbb{E}[(\gamma_{t+1} + \boldsymbol{\beta}_{t+1}^T (\mathbf{I} - \mathbf{C}_{t+1}^T)^{-1} \boldsymbol{\alpha}_{t+1}^T) A_{t+1} | H_t, A_{t+1} = a] \\
&\quad - \mathbb{E}[(\gamma_{t+1} + \boldsymbol{\beta}_{t+1}^T (\mathbf{I} - \mathbf{C}_{t+1}^T)^{-1} \boldsymbol{\alpha}_{t+1}^T) A_{t+1} | H_t, A_{t+1} = 0] \\
&= (\gamma_{t+1} + \boldsymbol{\beta}_{t+1}^T (\mathbf{I} - \mathbf{C}_{t+1}^T)^{-1} \boldsymbol{\alpha}_{t+1}^T) a,
\end{aligned}$$

where the second equality is due to the sequential ignorability assumption, the third equality is due to the consistency assumption, the fourth and fifth equality are due to the Equation (B.1).

APPENDIX

C

APPENDIX FOR CHAPTER 3

C.1 Acronyms

ACRONYMS

- AANE** Accelerated Attributed Network Embedding. 76
- AMG-DP** Attributed Multiplex Graph based loan Default Prediction approach. 76
- ARIMA** AutoRegressive Integrated Moving Average. 74
- ASGCN** Adaptive Sampling Graph Convolutional Network. 80
- AUC** Area Under the Curve. 76, 77, 79, 80, 81
- AutoGCN** Auto-encoder based Graph Convolutional Networks. 80
- Bi-HGNN** Bipartite Hierarchical bipartite Graph Neural Network. 79
- CARE-GNN** CAouflage-REsistant GNN. 79
- CNN** Convolutional Neural Network. 80
- CSI** China Securities Index. 73
- DE** Detection Expansion. 80
- DGANN** Dynamic Graph-based Attention Neural Network. 76
- DHGReg** Dynamic Heterogeneous Graph Neural Network. 81
- Diffpool** Differentiable graph pooling. 79
- DIN** Deep Interest Network. 79
- DNN** Deep Neural Network-based model. 76, 80
- DOMINANT** Deep anOMaly detectIoN on Attributed NeTworks. 81
- DW** DeepWalk. 76, 82
- GAL** Graph Anomaly Loss. 81
- GAS** GCN-based Anti-Spam model. 79
- GAT** Graph Attention Network. 76, 79, 80, 81
- GBDT** Gradient Boosting Decision Tree. 76, 79, 80, 81

GBR Graph-Based Ranking. 73

GCMC Graph Convolutional Matrix Completion. 79

GCN Graph Convolutional Network. 73, 74, 76, 77, 79, 80, 81

GCNEXT Graph Convolutional Network with Expedited Balance Theory. 79

GE Graph Embedding-based method. 79

GEM Graph Embeddings for Malicious accounts. 81

GF Graph Factorization. 76

GGNN Gated Graph Neural Network. 82

GraphRfi GCN-based user Representation learning framework. 79

GraphSAGE SAmples and aggreGatE. 76, 79, 81

GRC Graph neural network with a Role-constrained Conditional random field. 76

GRNN Graph Recurrent Neural Network. 76

HAN Hierarchical Attention Networks. 74, 76

HAN Hybrid Attention Network. 74

HGAR High-order Graph Attention Representation. 76

HGT Heterogeneous Graph Transformer. 80

HiGNN Hierarchical bipartite Graph Neural Network. 79

HMGNN Heterogeneous Mini-Graphs Neural Network. 80

HTML Hierarchical Transformer-based Multi-task Learning. 74

ICF Item-based Collaborative Filtering. 79

IRR Investment Return Ratio. 73, 74

KS Kolmogorov-Smirnov distance. 76

LINE Large-scale Information Network Embedding. 76

LR Linear Regression. 74, 76, 80

LSTM Long Short Term Memory. 73, 74, 82

LSTM-RGCN LSTM Relational Graph Convolutional Network. 74

MAE Mean Absolute Error. 79

MAN-SF Multipronged Attention Network for Stock Forecasting. 74

MCC Matthew's Correlation Coefficient. 74

MDRM Multimodal Deep Regression Model. 74

MF Matrix Factorization. 79

mGCN modified version of Graph Convolutional Network. 80

MLP Multi-Layer Perceptron. 76, 77, 80, 81

MRR Mean Reciprocal Rank. 73, 74

MSE Mean Squared Error. 73, 74

MvMoE Multi-view-aware Mixture-of-Experts network. 76

NASDAQ NASDAQ stock exchange. 74

Nikkei 225 Nikkei Stock Average. 73

NYSE New York Stock Exchange. 73, 74

PMF Probabilistic Matrix Factorization. 79

PMI model based on Pairwise Mutual Information. 82

Precision@k Precision of the top k nodes. 76

RCF Robust Collaborative Filtering model. 79

RF Random Forest. 74, 77, 80

RGCN Relational Graph Convolutional Network. 79

RMSE Root Mean Squared Error. 79

RNN Recurrent Neural Network. 76

RSR Relational Stock Ranking. 73, 74

SEAL learning from Subgraphs, Embeddings and Attributes for Link prediction. 76

SemiGNN Semisupervised attentive Graph Neural Network. 76, 79

SFM State Frequency Memory. 73

SGCN Signed Graph Convolutional Network. 79

SIDE Signed Directed network Embedding. 79

S-LSTM Sentence-state LSTM. 74

SNE attributed Social Network Embedding. 76

SP500 Standard and Poor's 500. 73, 74

STAR Spatio-Temporal Attentive Recurrent neural network. 76

ST-GNN Spatial-Temporal aware Graph Neural Network. 76

SVM Support Vector Machine. 76, 77, 80

TPX Tokyo Stock Price Index. 74

TRACER TempoRal Attention Contagion chain Enhanced Rating model. 76

TRAN Time-aware graph Relational Attention Network. 74

TSLDA Topic Sentiment Latent Dirichlet Allocation. 74

VoITAGE Volatility forecasting via Text-Audio fusion with Graph convolution networks for Earnings calls. 74

Xgboost eXtreme gradient boosting. 76, 77, 80