University of California
Santa Barbara

Learning Representations for Information-rich Graphs

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Zexi Huang

Committee in charge:

Professor Ambuj Singh, Chair
Professor Yu-Xiang Wang
Professor Xifeng Yan

June 2023

The Dissertation of Zexi Huang is approved.

Professor Yu-Xiang Wang

Professor Xifeng Yan

Professor Ambuj Singh, Committee Chair

April 2023

Learning Representations for Information-rich Graphs

by

Zexi Huang

To my family for their unconditional love and support.

Acknowledgements

The five-year PhD study at UCSB has been the most exciting and fruitful part of my life. I would love to take this opportunity to thank all the people who supported, motivated, and guided me through the journey.

First and foremost, my heartfelt gratitude goes to my PhD advisor, Prof. Ambuj Singh. It is due to his guidance, encouragement, and enlightenment that I become the person that I always aspire to be. And his vision, optimism, and wisdom will continue benefiting me in the future. Great thanks to Prof. Yu-Xiang Wang and Prof. Xifeng Yan for being part of my PhD committee and providing their expertise and valuable mentorship.

I feel fortunate to have collaborated with some of the brightest researchers in the world: Arlei Silva, Mert Kosan, Sourav Medya, Sayan Ranu, Wei Ye, Manu Kondapaneni, Marianne Arriola, and Saurabh Sharma. I am also grateful for having a great group of labmates: Rachel, Richika, Chandana, Ashwini, Haraldur, Omid, Hongyuan, Furkan, Yuning, Sikun, Nikunj, Christos, Rasta, Kha-Dinh, and Sean. Special thanks to Arlei Silva, Mert Kosan, and Sourav Medya for being my greatest mentors and closest friends during my PhD study. It is hard to imagine what would have happened without them.

I also want to thank my colleagues and friends during my internships at Amazon: my manager Kannan Shah, my mentors Han Wezenberg and Alfredo Nantes, and my teammates Eric, Derek, Qian, Dushyanta, Vel, Gary, Puifai, and Gautham. They showed me the value of data science in the industry and enabled me to make a real-world impact.

I am blessed to have great company and support from my roommates, Liang, Yuke, Youfu, and Lianke. Life under a pandemic would have been a disaster without them.

Finally, committing to PhD study in a foreign country means being far away from home and my dearest ones. I miss my family and this dissertation is dedicated to them.

Curriculum Vitæ

Zexi Huang

Education

2023	Ph.D. in Computer Science, University of California, Santa Barbara.
2022	M.S. in Computer Science, University of California, Santa Barbara.
2018	B.Eng. in Computer Science and Technology, University of Electronic Science and Technology of China

Publications

Refereed conference papers:

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, Ambuj Singh. *Global Counterfactual Explainer for Graph Neural Networks*. Best paper at the Machine Learning on Graphs Workshop (MLoG) and top 10 best papers at ACM International Conference on Web Search and Data Mining (WSDM), 2023.

Zexi Huang, Arlei Silva, Ambuj Singh. *POLE: Polarized Embedding for Signed Networks*. ACM International Conference on Web Search and Data Mining (WSDM), 2022.

Zexi Huang, Arlei Silva, Ambuj Singh. *A Broader Picture of Random-walk Based Graph Embedding*. ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD), 2021.

Under review and working papers:

Zexi Huang, Mert Kosan, Arlei Silva, Ambuj Singh. *Link Prediction without Graph Neural Networks*. Under review, 2023.

Zexi Huang, Manu Kondapaneni, Arlei Silva, Ambuj Singh. *Multiscale Community Detection with Pointwise Mutual Information*. Working paper, 2023.

Marianne Arriola, Mert Kosan, Zexi Huang, Saurabh Sharma, Ambuj Singh. *Multiscale Anomaly Detection with Graph Autoencoders*. Working paper, 2023.

Wei Ye, Zexi Huang, Yunqi Hong, Ambuj Singh. *Graph Neural Diffusion Networks for Semi-supervised Learning*. Under review, 2022.

Science Internships

Applied Scientist, Amazon Summer 2022

Project: Stochastic Inventory Management for Print-On-Demand and Graph-based Text Classification for Content Intelligence

Mentor: Alfredo Nantes

Applied Scientist, Amazon Summer 2021

Project: Graph-based Fraud Detection in Kindle Direct Publishing
Mentor: Han Wezenberg

Applied Scientist, Amazon Summer 2020

Project: Graph-based Fraud Detection in Kindle Direct Publishing
Mentor: Han Wezenberg

Research Assistant, Nanyang Technological University Fall 2017 - Winter 2018

Project: Transfer Learning for Community Detection in Multiplex Networks
Advisor: Sinno Jialin Pan

Teaching Experience

Lead Teaching Assistant, Computer Science Department, UCSB 2020-2021

Teaching Assistant, Data Structures and Algorithms, UCSB Fall 2019

Instructor, LMU/UCSB Nanotech PhD Exchange and Symposium Spring 2019

Teaching Assistant, Introduction to Computer Science, UCSB Winter 2019

Teaching Assistant, Fundamentals of Database Systems, UCSB Fall 2018

Academic Services

Registration Chair: KDD'23

Program Committee: AAAI'23, KDD'22, SDM'22

Reviewer: NeurIPS'22, ICLR'22, KDD'20-21, WWW'20-21, TIST'22, TKDD'21-23

Representative: Graduate Affairs Committee, Computer Science Department, UCSB

Abstract

Learning Representations for Information-rich Graphs

by

Zexi Huang

Graphs are powerful data structures that are used to model some of the most complex systems in the world, such as interpersonal relationships in social networks, protein-protein interactions in biology, and user-item pairs in recommender systems. Generating representations that encode the rich information from graph data enables state-of-the-art solutions to many real-world applications across various domains.

This dissertation focuses on our recent work on representation learning for information-rich graphs. We first present a unified framework for random-walk based graph embedding approaches and analyze how different choices of model components affect downstream task performance. We then propose a novel embedding method for signed graphs, which incorporates social theory into random-walk dynamics to capture social polarization and enable effective signed link prediction. For attributed graphs, Graph Neural Networks have become the most popular representation learning paradigm. We address their limitations in combining structural and attribute information for link prediction and introduce the first global counterfactual explainer for their applications in graph classification. Finally, we investigate representation learning for multiscale graphs and discuss several related problems including semi-supervised node classification, community detection, and anomaly detection. Research in this dissertation demonstrates the importance of accounting for the interplay between the rich graph information and downstream task properties in successful graph representation learning models.

Permissions and Attributions

This dissertation contains material that has been published or is in the process of being published. The author of this dissertation claims principal contributions in the development of the published research works described below:

1. Part of the content of Chapter 2 has been previously published as: Zexi Huang, Arlei Silva, Ambuj Singh. *A Broader Picture of Random-walk Based Graph Embedding*. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD), 2021. DOI: 10.1145/3447548.3467300.

2. Part of the content of Chapter 3 has been previously published as: Zexi Huang, Arlei Silva, Ambuj Singh. *POLE: Polarized Embedding for Signed Networks*. Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM), 2022. DOI: 10.1145/3488560.3498454.

3. Part of the content of Chapter 5 has been previously published as: Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, Ambuj Singh. *Global Counterfactual Explainer for Graph Neural Networks*. Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM), 2023. DOI: 10.1145/3539597.3570376.

Based on the ACM Author Rights page (https://authors.acm.org/author-resources/author-rights), authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in an era when data is generated at an unprecedented speed. In 2020, 84 petabytes (10^{15} bytes) of data were created every minute, among which 42 million WhatsApp messages were exchanged, 350 thousand tweets were posted, 6,659 Amazon packages were shipped, and 300 hours of videos were uploaded to YouTube[1]. To take advantage of the increasing availability of data, a new research field, *data science*, has emerged at the intersection between computer science, statistics, and various application domains, such as social science, biology, physics, and health care.

One fundamental topic of data science is *representation learning*. The goal of representation learning is to extract key information from raw data to enable effective downstream applications. Compared to traditional feature engineering methods which rely on hand-designed features based on domain expertise, representation learning models leverage deep neural networks to directly learn high-quality features (embeddings) from data. Several representation learning paradigms have revolutionized our way of solving problems involving certain types of data, such as Convolutional Neural Network for images and videos, and Transformers for natural languages.

[1] `https://financesonline.com/how-much-data-is-created-every-day/`

The focus of this dissertation is representation learning on *graphs*. Graphs (or networks) are powerful data structures that are the natural choice for modeling many real-world complex systems including online social networks, protein interactions, and financial transactions, to name a few. In fact, other types of structured data (such as images and texts) can be modeled as (grid and line) graphs as well. The modeling power of graphs comes from both the graph structure (or topology) that encodes *relationship information*, but also node and edge attributes that capture other *descriptive information*. For example, the types of interpersonal relationships in a social network (e.g., friendly or adversarial) can be represented by the signs on edges, and the topics of paper in a citation network can be encoded as node attributes.

Owing to the modeling power of graphs, many real-world data science problems can be mapped to graph-related tasks. The applications investigated in this dissertation include those that fall under the general categories of node classification, link prediction, and community detection, and also more context-specific ones such as characterizing polarization in a social network or generating novel candidates via counterfactual explanation in drug discovery. The role of graph representation learning is to mobilize data modeled as information-rich graphs to solve those downstream applications, as shown in Figure 1.1. The main challenge of graph representation learning this dissertation attempts to address is how to account for the interplay between the rich graph information and downstream task properties to enable state-of-the-art performance.

The rest of the dissertation is organized by our contributions to the advancement of graph representation learning, summarized as follows:

1. *A Broader Picture of Random-walk Based Graph Embedding* [1] (Chapter 2): Graph embedding based on random-walks supports effective solutions for many graph-related downstream tasks. However, the abundance of embedding literature has

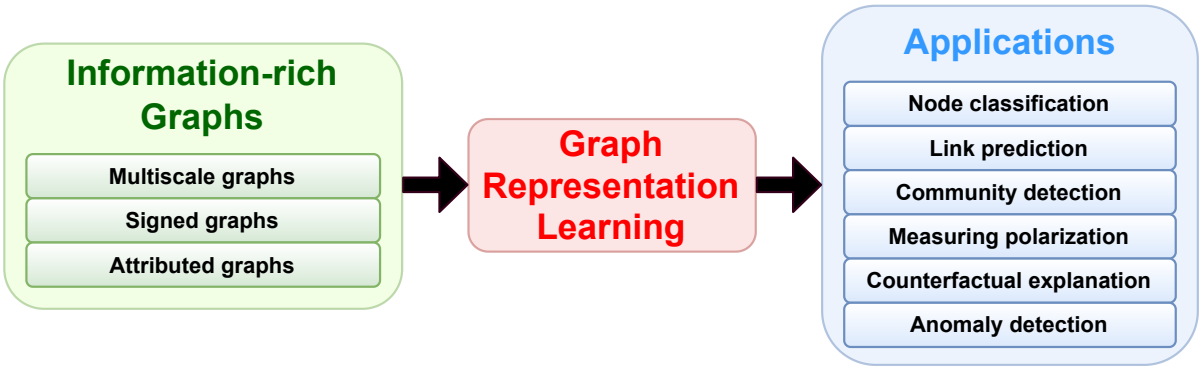


Figure 1.1: Graph representation learning as the bridge between various types of information-rich graphs and different downstream applications covered in this dissertation.

made it increasingly difficult to compare existing methods and to identify opportunities to advance the state-of-the-art. Meanwhile, existing work has left several fundamental questions—such as how embeddings capture different structural scales and how they should be applied for effective link prediction—unanswered. This work addresses these challenges with an analytical framework for random-walk based graph embedding that consists of three components: a random-walk process, a similarity function, and an embedding algorithm. Our framework not only categorizes many existing approaches but naturally motivates new ones. With it, we illustrate novel ways to incorporate embeddings at multiple scales to improve downstream task performance. We also show that embeddings based on autocovariance similarity, when paired with dot product ranking for link prediction, outperform state-of-the-art methods based on Pointwise Mutual Information similarity by up to 100%.

2. *POLE: Polarized Embedding for Signed Networks* [2] (Chapter 3): From the 2016 U.S. presidential election to the 2021 Capitol riots to the spread of misinformation related to COVID-19, many have blamed social media for today's deeply divided

society. Recent advances in machine learning for signed networks hold the promise to guide small interventions with the goal of reducing polarization in social media. However, existing models are especially ineffective in predicting conflicts (or negative links) among users. This is due to a strong correlation between link signs and the network structure, where negative links between polarized communities are too sparse to be predicted even by state-of-the-art approaches. To address this problem, we first design a partition-agnostic polarization measure for signed graphs based on the signed random-walk and show that many real-world graphs are highly polarized. Then, we propose POLE (POLarized Embedding for signed networks), a signed embedding method for polarized graphs that captures both topological and signed similarities jointly via signed autocovariance. Through extensive experiments, we show that POLE significantly outperforms state-of-the-art methods in signed link prediction, particularly for negative links with gains of up to one order of magnitude.

3. *Link Prediction without Graph Neural Networks* [3] (Chapter 4): Link prediction, which consists of predicting edges based on graph features, is a fundamental task in many graph applications. As for several related problems, Graph Neural Networks (GNNs), which are based on an attribute-centric message-passing paradigm, have become the predominant framework for link prediction. GNNs have consistently outperformed traditional topology-based heuristics, but what contributes to their performance? Are there simpler approaches that achieve comparable or better results? To answer these questions, we first identify important limitations in how GNN-based link prediction methods handle the intrinsic class imbalance of the problem—due to the graph sparsity—in their training and evaluation. Moreover, we propose *Gelato*, a novel topology-centric framework that applies a topological

heuristic to a graph enhanced by attribute information via graph learning. Our model is trained end-to-end with an N-pair loss on an unbiased training set to address class imbalance. Experiments show that Gelato is 145% more accurate, trains 11 times faster, infers 6,000 times faster, and has less than half of the trainable parameters compared to state-of-the-art GNNs for link prediction.

4. *Global Counterfactual Explainer for Graph Neural Networks* [4] (Chapter 5): Graph neural networks (GNNs) find applications in various domains such as computational biology, natural language processing, and computer security. Owing to their popularity, there is an increasing need to explain GNN predictions since GNNs are black-box machine learning models. One way to address this is *counterfactual* reasoning where the objective is to change the GNN prediction by minimal changes in the input graph. Existing methods for counterfactual explanation of GNNs are limited to instance-specific *local* reasoning. This approach has two major limitations of not being able to offer global recourse policies and overloading human cognitive ability with too much information. In this work, we study the *global* explainability of GNNs through global counterfactual reasoning. Specifically, we want to find a *small* set of representative counterfactual graphs that explains *all* input graphs. Towards this goal, we propose GCFExplainer, a novel algorithm powered by *vertex-reinforced random walks* on an *edit map* of graphs with a *greedy summary*. Extensive experiments on real graph datasets show that the global explanation from GCFExplainer provides important high-level insights of the model behavior and achieves a 46.9% gain in recourse coverage and a 9.5% reduction in recourse cost compared to the state-of-the-art local counterfactual explainers.

5. *Other Problems Related to Multiscale Graphs* [5, 6, 7] (Chapter 6): Here, we group ongoing projects related to representation learning for multiscale networks to which

the author of this dissertation has made significant contributions.

(a) *Graph Convolutional Networks Meet Neural Diffusions* [5] (Section 6.1): Graph Convolutional Networks (GCNs) are pioneering models for graph-based semi-supervised learning. However, they do not perform well on sparsely-labeled graphs. Its two-layer version cannot effectively propagate the label information to the entire graph (i.e., under-smoothing) while its deep version over-smoothens and is hard to train (i.e., the over-smoothing problem). To address these issues, we propose a new Graph Neural Network architecture called GND-Nets (for Graph Neural Diffusion Networks) that exploits the multi-scale structural neighborhood information of a node in a single layer. Using the shallow architecture mitigates the over-smoothing problem while leveraging both the local and global structural neighborhood information mitigates the under-smoothing problem. The extraction and utilization of the multiscale neighborhood information are achieved by a new graph diffusion method called neural diffusions, which integrate neural networks into the traditional linear and nonlinear graph diffusion dynamics. The adoption of neural networks enables our model to attend to different datasets. Extensive experiments on various sparsely-labeled graphs verify the effectiveness and efficiency of GND-Nets compared with state-of-the-art approaches.

(b) *Multiscale Community Detection with Pointwise Mutual Information* [6] (Section 6.2): Community detection consists of grouping nodes in a graph such that they are densely connected within each community (cluster) and sparsely across communities. Existing community detection algorithms tailored for capturing communities at multiple structural scales, such as Markov Stability based on the clustered Autocovariance (AC) similarity, assume a user-defined

and fixed scale for the entire graph. Instead, we propose to adopt a different node similarity metric for multiscale community detection, Pointwise Mutual Information (PMI), inspired by the network embedding literature. We demonstrate that community quality functions based on the PMI similarity reveal the optimal (natural) scale for each community automatically and independently, and can therefore lead to algorithms that detect heterogeneous community organization of real-world graphs.

(c) *Multiscale Anomaly Detection with Graph Autoencoders* [7] (Section 6.3): Graph anomaly detection, which consists of detecting suspicious or unexpected structural and feature patterns in attributed networks, has enabled many real-world applications, such as fraud detection in financial systems and intrusion detection in cyber security. Existing graph anomaly detection approaches focus on detecting individual anomalous nodes within a particular context or multiscale contexts. Here, we show that anomalous nodes in many real-world graphs form cohesive anomalous clusters of different structural scales and exhibit distinctive spectral energy distribution patterns. To effectively surface those multiscale anomalous subgraphs, we propose an unsupervised framework that incorporates spectral localized Beta Wavelet Graph Neural Networks into the graph autoencoder anomaly detection paradigm. Anomalies are then flagged as those with unstable embeddings and large autoencoder reconstruction errors.

Chapter 2

A Broader Picture of Random-walk Based Graph Embedding

2.1 Introduction

Random-walk based graph embedding enables the application of classical algorithms for high-dimensional data to graph-based downstream tasks (e.g., link prediction, node classification, and community detection). These embedding methods learn vector representations for nodes based on some notion of topological similarity (or proximity). Since DeepWalk [8], we have witnessed a great interest in graph embedding both by researchers and practitioners. Several regular papers [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] and a few surveys [23, 24, 25, 26, 27] have attempted to not only advance but also consolidate our understanding of these models. However, the abundance of literature also makes it increasingly difficult for practitioners and newcomers to the field to compare existing methods and to contribute with novel ones.

On the other hand, despite the rich literature, several fundamental questions about random-walk based graph embedding still remain unanswered. One such question is

(Q1) how do embeddings capture different structural scales? Random-walks of different lengths are naturally associated with varying scales [28]. However, downstream task performance of embeddings has been shown to be insensitive to random-walk lengths [8, 9]. Another relevant question is *(Q2) how should random-walk embeddings be used for link prediction?* Following node2vec [9], several works train classifiers to predict missing links based on a set of labeled pairs (edges and non-edges) [29, 30, 31]. This is counter-intuitive given that the embedding problem is often defined in terms of dot products. In fact, dot products are sometimes also applied for link prediction and for the related task of network reconstruction, where the entire graph is predicted based on the embeddings [32, 10, 26, 33].

With these questions in mind, we shall take a closer look at how embeddings are produced in random-walk based methods. It starts with selecting a *random-walk process*. DeepWalk [8] applies standard random-walks, while node2vec [9] considers biased random-walks and APP [15] adopts rooted PageRank, among others. Then, a *similarity function* maps realizations of the random-walk process into real values that represent some notion of node proximity. Most existing methods rely on the skip-gram language model [34], which has been shown to capture the Pointwise Mutual Information (PMI) similarity [35]. Finally, an *embedding algorithm* is used to generate vector representations that preserve the similarity function via optimization. This can be based on either sampled random-walks and gradient descent (or one of its variations) [8, 9], or (explicit) matrix factorization (e.g., Singular Value Decomposition) [11, 14].

This breakdown of random-walk based embedding methods allows us to build a simple, yet powerful analytical framework with three major components: *random-walk process*, *similarity function*, and *embedding algorithm*. As shown in Figure 2.1, our framework both categorizes existing approaches and facilitates the exploration of novel ones. For example, we will consider embeddings based on the autocovariance similarity, which

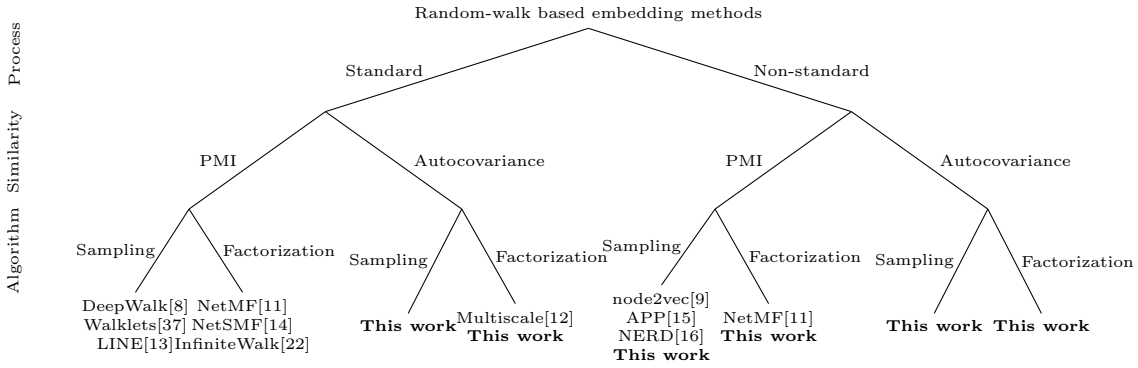


Figure 2.1: Different random-walk based embedding methods (old and new) classified according to our analytical framework—with process, similarity, and algorithm as main components. A key contribution of this work is to integrate autocovariance as a similarity metric and show that it outperforms Pointwise Mutual Information (PMI) in link prediction.

has been proposed for multiscale community detection [28, 36], and compare it against the more popular PMI similarity on multiple downstream tasks.

Our framework also provides tools to answer the aforementioned questions. For Q1, we will not only illustrate how past work implicitly combines multiple scales and *its implications to node-level tasks*, but also propose novel ways to incorporate scales to *improve edge-level task performance*. To answer Q2, we will show that to optimize performance, *embedding methods should be designed with task settings in mind*. Specifically, we find that embeddings based on autocovariance, when paired with dot products, lead to a two-fold improvement over existing methods. Our analysis shows the reason to be that the particular combination enables the embedding to capture heterogeneous degree distributions [38] in real graphs. One could argue that link prediction is the most relevant downstream task for (positional) node embeddings, as graph neural networks often outperform embedding approaches in node classification [39].

To summarize, the main contributions of our work are:

- We present a unified view of random-walk based graph embedding, incorporating different random-walk processes, similarity functions, and embedding algorithms.

- We show how autocovariance can be applied as a similarity function to create novel embeddings that outperform state-of-the-art methods using PMI by up to 100%.
- We illustrate ways to exploit the multiscale nature of random-walk similarity to further optimize embedding performance.
- We conduct an extensive experimental evaluation of our framework on various downstream tasks and provide theoretical insights behind the results.

2.2 Method

Consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots n\}$ denotes the set of n nodes and $\mathcal{E}$ denotes the set of m edges. The graph is represented by a weighted symmetric adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{uv} > 0$ if an edge of weight A_{uv} connects nodes u and v, and $A_{uv} = 0$, otherwise. The (weighted) degree of node u is defined as $\deg(u) = \sum_v A_{uv}$.

A node embedding is a function $\phi : \mathcal{V} \mapsto \mathbb{R}^d$ that maps each node v to a d-dimensional ($d \ll n$) vector $\mathbf{u}_v$. We refer to the embedding matrix of $\mathcal{V}$ as $U = (\mathbf{u}_1, \dots, \mathbf{u}_n)^\top \in \mathbb{R}^{n \times d}$. For some embedding algorithms, another embedding matrix $V \in \mathbb{R}^{n \times d}$ is also generated. Random-walk based embedding methods use a random-walk process to embed nodes u and v such that a similarity metric is preserved by dot products $\mathbf{u}_u^\top \mathbf{u}_v$ (or $\mathbf{u}_u^\top \mathbf{v}_v$). In the next section, we will formalize random-walks on graphs.

2.2.1 Random-walk Process

A random-walk is a Markov chain over the set of nodes $\mathcal{V}$. The transition probability of the walker jumping to node v is based solely on its previous location u and is characterized by the adjacency matrix A. For the *standard random-walk process*, the transition

probability is proportional to the edge weight A_{uv}:

$$p(x(t+1)=v|x(t)=u) = \frac{A_{uv}}{\deg(u)} \tag{2.1}$$

where $x(t) \in \mathcal{V}$ is the location of the walker at time t.

Transition probabilities between all pairs of nodes are represented by a transition matrix $M \in \mathbb{R}^{n \times n}$:

$$M = D^{-1}A \tag{2.2}$$

where $D = \text{diag}([\deg(1), \ldots, \deg(n)]) \in \mathbb{R}^n$ is the degree matrix.

For a connected non-bipartite graph, the standard random-walk is ergodic and has a unique stationary distribution $\pi \in \mathbb{R}^n$:

$$\pi_u = \frac{\deg(u)}{\sum_v \deg(v)} \tag{2.3}$$

Standard random-walks provide a natural way to capture node neighborhood in undirected connected graphs. One can also design *biased random-walks* to explore different notions of neighborhood [9]. For directed graphs, a *PageRank process* [40] is often applied in lieu of standard random-walks to guarantee ergodicity.

2.2.2 Similarity Function

A node similarity metric is a function $\varphi : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ that maps pairs of nodes to some notion of topological similarity. Large positive values mean that two nodes are similar to each other, while large negative values indicate they are dissimilar.

Random-walk based similarity functions are based on co-visiting probabilities of a walker. An important property of random-walks that has been mostly neglected in the embedding literature is their ability to capture similarity at different structural scales

(e.g., local vs. global). This is achieved via a Markov time parameter $\tau \in \mathbb{Z}_+$, which corresponds to the distance between a pair of nodes in a walk in terms of jumps. One of the contributions of this work is to show the effect of different Markov time scales on the embedding and ways to exploit them to optimize downstream task performance.

We will describe two random-walk based similarity functions in this section, *Pointwise Mutual Information (PMI)* and *autocovariance*. PMI has become quite popular in the graph embedding literature due to its (implicit) use by word2vec [34, 35] and DeepWalk [8, 11]. On the other hand, autocovariance is more popular in the context of multiscale community detection [28, 36]. As one of the contributions of this work, we will demonstrate the effectiveness of autocovariance based embeddings on edge-level tasks.

Pointwise Mutual Information (PMI)

Denote $X_v(t) \in \{0, 1\}$ as the event indicator of $x(t) = v$, which can be true (1) or false (0). The PMI between events $X_u(t) = 1$ and $X_v(t + \tau) = 1$ is defined as:

$$\begin{aligned} R_{uv}(\tau) &= \mathrm{PMI}(X_u(t) = 1, X_v(t + \tau) = 1) \\ &= \log \frac{p(X_u(t) = 1, X_v(t + \tau) = 1)}{p(X_u(t) = 1)p(X_v(t + \tau) = 1)} \end{aligned} \tag{2.4}$$

PMI provides a non-linear information theoretic view of random-walk based proximity. For an ergodic walk with the stationary distribution π as starting probabilities, PMI can be computed based on π and the τ-step transition probability:

$$R_{uv}(\tau) = \log(\pi_u p(x(t+\tau)=v|x(t)=u)) - \log(\pi_u \pi_v) \tag{2.5}$$

where $R_{uv}(\tau) \in [-\infty, -\log(\pi_v)]$. In matrix form:

$$R(\tau) = \log(\Pi M^\tau) - \log(\pi\pi^\top) \tag{2.6}$$

where $\Pi = \text{diag}(\pi) \in \mathbb{R}^{n\times n}$ and $\log(\cdot)$ is the element-wise logarithm. The PMI matrix is symmetric due to the time-reversibility of undirected random-walks.

It is noteworthy that LINE [13] and DeepWalk [8]—two random-walk embedding methods—implicitly factorize PMI. Specifically:

Theorem 2.1 *LINE and DeepWalk implicitly factorize the following matrices:*

$$R_{LINE} = R(1) - \log b \tag{2.7}$$

$$R_{DW} = \log\left(\frac{1}{T}\sum_{\tau=1}^{\top}\exp(R(\tau))\right) - \log b \tag{2.8}$$

where b and T are the number of negative samples and context window size, respectively, and $\exp(\cdot)$ is the elementwise exponential.

Proof: As shown in [11], LINE and DeepWalk factorize the following matrices:

$$R_{LINE} = \log(\text{vol}(\mathcal{G})D^{-1}AD^{-1}) - \log b \tag{2.9}$$

$$R_{DW} = \log(\text{vol}(G)(\frac{1}{T}\sum_{r=1}^{\top}(D^{-1}A)^r)D^{-1}) - \log b \tag{2.10}$$

where b and T are the number of negative samples and the context window size in skip-gram respectively, and $\text{vol}(\mathcal{G}) = \sum_u \deg(u)$. Now, consider the standard random-walk where $M = D^{-1}A$ and $\Pi = D/\text{vol}(\mathcal{G})$. We notice the following equivalence:

Lemma 2.1 *The PMI similarity matrix $R(\tau)$ in Equation 2.6 for the standard random-walk can be expressed as*

$$R(\tau) = \log(\text{vol}(\mathcal{G})(D^{-1}A)^{\tau}D^{-1}) \tag{2.11}$$

We use $\circ$ and $\oslash$ to represent elementwise multiplication and division between matrices.

We have:

$$\begin{aligned} R(\tau) &= \log(\Pi M^\tau) - \log(\pi\pi^\top) \\ &= \log((\pi,\ldots,\pi) \circ M^\tau) - \log((\pi,\ldots,\pi) \circ (\pi,\ldots,\pi)^\top) \\ &= \log(M^\tau) - \log((\pi,\ldots,\pi)^\top) \\ &= \log(M^\tau \oslash (\pi,\ldots,\pi)^\top) \\ &= \log(M^\tau \Pi^{-1}) \\ &= \log(\mathrm{vol}(\mathcal{G})(D^{-1}A)^\tau D^{-1}) \end{aligned} \tag{2.12}$$

Thus, R_{LINE} factorizes a shifted version of the similarity at $\tau=1$:

$$R_{LINE} = R(1) - \log b \tag{2.13}$$

And R_{DW} factorizes a smooth approximation of the average (*log-mean-exp*) for the shifted similarity with t from 1 to T:

$$R_{DW} = \log\left(\frac{1}{T}\sum_{\tau=1}^{T}\exp(R(\tau))\right) - \log b \tag{2.14}$$

■

As we see, LINE preserves the PMI similarity at Markov time 1, while DeepWalk factorizes the *log-mean-exp*—a smooth approximation of the average—of the PMI matrices from Markov time 1 to T.

Autocovariance

The autocovariance is defined as the covariance of $X_u(t)$ and $X_v(t+\tau)$ within a time interval τ:

$$\begin{aligned} R_{uv}(\tau) &= \operatorname{cov}(X_u(t), X_v(t+\tau)) \\ &= \mathrm{E}\left[\left(X_u(t) - \mathrm{E}[X_u(t)]\right)\left(X_v(t+\tau) - \mathrm{E}[X_v(t+\tau)]\right)\right] \end{aligned} \tag{2.15}$$

The value of $R_{uv}(\tau)$ is a linear measure of the joint variability of the walk visiting probabilities for nodes u and v within time τ. Similar to PMI, for an ergodic walk and starting probabilities π:

$$R_{uv}(\tau) = \pi_u p(x(t+\tau)=v|x(t)=u) - \pi_u\pi_v \tag{2.16}$$

where $R_{uv}(\tau) \in [-\pi_u\pi_v, \pi_u(1-\pi_v)]$. In the matrix form:

$$R(\tau) = \Pi M^\tau - \pi\pi^\top \tag{2.17}$$

The autocovariance (Equation 2.17) and PMI (Equation 2.6) matrices share a very similar form, differing only by the logarithm operation. However, this distinction has broad implications for graph embedding. PMI is a non-linear metric closely related to the *sigmoid function*, which is a quite popular activation in deep learning, while autocovariance is related to a *piecewise linear function*—see Section 2.2.3 for details. We will compare PMI and autocovariance in multiple tasks in Section 2.3 and provide theoretical and empirical insights on their different performance in Section 2.4.

2.2.3 Embedding Algorithm

The goal of an embedding algorithm is to generate vector representations that preserve a given similarity metric:

$$\begin{aligned} U^* &= \arg\min_U \sum_{u,v} (\mathbf{u}_u^\top \mathbf{u}_v - R_{uv})^2 \\ &= \arg\min_U \|UU^\top - R\|_F^2 \end{aligned} \tag{2.18}$$

where $\mathbf{u}_u^\top \mathbf{u}_v$ captures similarity in the embedding space, R is a similarity matrix, and $\|\cdot\|_F$ is the Frobenius norm.

In the following sections, we discuss two different techniques to optimize this embedding objective.

Matrix factorization

Matrix factorization is an explicit way to optimize the embedding. It generates a low-rank approximation of R as $UU^\top$ with $\text{rank}(UU^\top) = d$. Because R is symmetric, from the Eckart-Young-Mirsky theorem [41], the optimal $U^* = Q_d\sqrt{\Lambda_d}$, where $R = Q\Lambda Q^\top$ is the Singular Value Decomposition (SVD) of R. Notice that, different from classical spectral approaches [42], factorization-based embedding is not based on the graph Laplacian.

Direct SVD of R has a complexity of $O(n^3)$, which is infeasible for most applications. However, scalable factorization is possible for sparse graphs. For autocovariance, one can apply the Lanczos method [43], which only requires sparse matrix-vector multiplications. This approach reduces the complexity to $O(nd^2 + md\tau)$. For PMI, as discussed in [14], one can construct the spectral sparsifier of the similarity matrix and apply Randomized SVD [44]. The resulting complexity for this method is $O(\widetilde{m}\tau \log n + \widetilde{m}d + nd^2 + d^3)$, where $\widetilde{m} = O(m\tau)$ is the number of non-zeros in the sparsifier.

Sampling

Sampling-based algorithms produce embeddings that implicitly optimize Equation 2.18 by maximizing the likelihood of a corpus $\mathcal{D}$ generated based on samples from the process. A sample i is a random-walk sequence $\langle v_1^{(i)}, v_2^{(i)}, \dots v_L^{(i)} \rangle$ of length L with the initial node $v_1^{(i)}$ selected according to a distribution $p(v)$—we will assume that $p(v) = \pi_v$ for the remainder of this section. From each sample, we extract pairs $(v_t^{(i)}, v_{t+\tau}^{(i)})$, where τ is the Markov time scale. Thus, $\mathcal{D}$ is a multiset of node pairs (u, v).

Different from matrix factorization, sampling algorithms produce two embedding matrices, U and V, the source and target embeddings, respectively. The use of two embeddings was initially proposed by word2vec [34] and is considered a better alternative. We will focus our discussion on algorithms that exploit *negative sampling*, with b negative samples, to efficiently generate embeddings. Let z be a random variable associated with pairs (u, v) such that $z = 1$ if (u, v) appears in $\mathcal{D}$ and $z = 0$, otherwise. The log-likelihood ℓ of the corpus $\mathcal{D}$ can be expressed as:

$$\ell = \sum_{u,v} \#(u, v)(\log(p(z{=}1|u, v)) + b\mathbb{E}_{w\sim\pi} \log(p(z{=}0|u, w))) \tag{2.19}$$

where $\#(u, v)$ is the number of occurrences of the pair (u, v) in $\mathcal{D}$.

The form of the conditional probability function $p(z|u, v)$ is determined by the similarity function. For instance, in the case of PMI, $p(z|u, v)$ is known to take the form of the *sigmoid* function: $\sigma(x) = 1/(1 + \exp(x))$ [11]. More specifically, $p(z = 1|u, v) = \sigma(\mathbf{u}_u^\top \mathbf{v}_v)$ and $p(z = 0|u, v) = \sigma(-\mathbf{u}_u^\top \mathbf{v}_v)$. Here, we show how the objective in Equation 2.19 can be maximized for the autocovariance similarity. First, we define the following conditional

probability function:

$$p(z=1|u,v) = \rho\left(\frac{\mathbf{u}_u^\top \mathbf{v}_v + \pi_u \pi_v}{\mathbf{u}_u^\top \mathbf{v}_v + (b+1)\pi_u \pi_v}\right) \tag{2.20}$$

$$p(z=0|u,v) = \rho\left(\frac{\pi_u \pi_v}{\mathbf{u}_u^\top \mathbf{v}_v + (b+1)\pi_u \pi_v}\right) \tag{2.21}$$

where $\rho(x)$ is a *piecewise linear activation* with $\rho(x) = 0$, if $x < 0$, $\rho(x) = x$, if $0 \leq x \leq 1$, and $\rho(x) = 1$, otherwise.

The following theorem formalizes the connection between the above defined conditional probability and autocovariance:

Theorem 2.2 *For a large enough dimensionality d ($d = \Omega(n)$), the embedding that maximizes Equation 2.19 with a conditional probability given by Equation 2.20 and Equation 2.21 is such that:*

$$\mathbf{u}_u^\top \mathbf{v}_v = \frac{1}{b}\pi_u p(x(t{+}\tau){=}v|x(t){=}u)) - \pi_u \pi_v \tag{2.22}$$

Proof: We first show that, for $b = 1$, Equations 2.20 and 2.21 can be derived from the definition of autocovariance and the Bayes Theorem. The corpus $\mathcal{D}$ is composed of samples from the random-walk process, and thus, for autocovariance-based embeddings:

$$\begin{aligned} p(u,v|z=1) &= \pi_u p(x(t{+}\tau){=}v|x(t){=}u)) \\ &= \mathbf{u}_u^\top \mathbf{v}_v + \pi_u \pi_v \end{aligned} \tag{2.23}$$

For pairs that are not in the corpus (negative samples):

$$p(u,v|z=0) = \pi_u \pi_v \tag{2.24}$$

Because $b=1$, $p(z=0)=p(z=1)$, and $p(u,v)=\mathbf{u}_u^\top\mathbf{u}_v+2\pi_u\pi_v$. From the Bayes Theorem, we get Equation 2.20 and Equation 2.21. We will assume that Equation 2.22 can be computed exactly from samples:

$$\mathbf{u}_u^\top\mathbf{v}_v = \frac{\#(u,v)}{b|\mathcal{D}|} - \frac{\#(u)\#(v)}{|\mathcal{D}|^2} \tag{2.25}$$

Moreover, similar to [35], we will simplify Equation 2.19 as follows:

$$\begin{aligned}
\ell &= \sum_{u,v}\#(u,v)\log(p(z=1|u,v)) + b\sum_{u,v}\#(u,v)\mathbb{E}_{w\sim\pi}\log(p(z=0|u,w)) \\
&= \sum_{u,v}\#(u,v)\log(p(z=1|u,v)) + b\sum_{u}\#(u)\frac{\#(v)}{|\mathcal{D}|}\log(p(z=0|u,v)) \\
&\quad + b\sum_{u,w\neq v}\#(u)\frac{\#(w)}{|\mathcal{D}|}\log(p(z=0|u,w))
\end{aligned}$$

where $\#(v)=\sum_w\#(v,w)$.

For a large enough number of dimensions, we can minimize the above Equation for each pair (u,v):

$$\ell_{u,v} = \#(u,v)\log(p(z=1|u,v)) + b\#(u)\frac{\#(v)}{|\mathcal{D}|}\log(p(z=0|u,v)) \tag{2.26}$$

Now, we plug the conditional probabilities $p(z=1|u,v)$ and $p(z=0|u,v)$ from Equation 2.20 and Equation 2.21 into $\ell_{u,v}$, and compute its derivative with respect to $x=\mathbf{u}_u^\top\mathbf{v}_v$, and we have

$$\frac{\partial\ell_{u,v}}{\partial x} = \frac{-\#(u,v)\pi_u\pi_v}{(x+\pi_u\pi_v)(x+(b+1)\pi_u\pi_v)} + \frac{b\#(u)\#(v)}{|\mathcal{D}|(x+(b+1)\pi_u\pi_v)}$$

where we have conveniently dropped the function ρ.

Setting the derivative to zero, we get Equation 2.25. We emphasize that similar to

[35], our theorem only holds when d is large enough to allow embeddings to be optimal pairwise, which can only be guaranteed in the general case when $d = n$. ■

While here we only show the sampling-based algorithm for autocovariance, previous work has given a similar proof for PMI [11].

We minimize the likelihood from Equation 2.19 using gradient descent. The time complexity of the sampling algorithms (for PMI and autocovariance) is $O(|\mathcal{D}|b)$, where $|\mathcal{D}|$ is the size of the corpus and b is the number of negative samples—in practice, $|\mathcal{D}| = O(n)$.

2.3 Experiments

In this section, we evaluate several random-walk based graph embedding methods defined according to our analytical framework on various downstream tasks and datasets.

2.3.1 Dataset

We apply six datasets (see Table 2.1) in our experiments.

- BlogCatalog [45]: Undirected social network of bloggers with (multi) labels representing topics of interest.

- Airport: Flight network among global airports (from OpenFlights [46]). Edges are weighted by the number of flights through the corresponding air routes. Labels represent the countries and continents where airports are located. The largest undirected connected component of the network is used in our experiments.

- Wiki-words [47]. Undirected co-occurrence network of words in the first million bytes of the Wikipedia dump. Labels are Part-of-Speech (POS) tags inferred by the Stanford POS-Tagger.

- PoliticalBlogs [48]. Network of hyperlinks between weblogs on US politics, recorded in 2005. Labels represent political ideologies (conservative vs liberal). The largest undirected connected component of the network is used in our experiments.

- Cora [49]: Directed citation network with categories (e.g., AI) and subcategories (e.g, Knowledge Representation) as labels.

- Wiki-fields: Subset of the Wikipedia directed hyperlink network [50] covering Computer Science, Mathematics, Physics and Biology for the year 2019. Fields and subfields are used as labels.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	labels
BlogCatalog	10,312	333,983	interests
Airport	3,158	18,606	countries/continents
Wiki-words	4,777	92,157	tags
PoliticalBlogs	1,222	16,717	ideologies
Cora	23,166	91,500	categories/subcategories
Wiki-fields	10,675	137,606	fields/subfields

Table 2.1: An overview of the datasets.

2.3.2 Experiment Setting

We evaluate embedding methods on three downstream tasks:

- Node classification. We follow the same procedure as [8]. For each dataset, we randomly sample a proportion of node labels for training and the rest for testing. We use one-vs-rest logistic regression implemented by LIBLINEAR [51] for multi-class classification (Airport and Cora) and multi-label classification (BlogCatalog, Wiki-words, and Wiki-fields). To avoid the thresholding effect [52] in the multi-label setting, we assume that the number of true labels for each node is

known. Performance is reported as average *Micro-F1* and *Macro-F1* [53] for 10 repetitions.

- Link prediction. We randomly remove 20% of edges while ensuring that the residual graph is still connected and embed the residual graph. Edges are predicted as the top pairs of nodes ranked by either dot products of embeddings or classification scores from a logistic regression classifier with the concatenation of embeddings as input. We report *precision@k* [54] as the evaluation metric and also use *recall@k* in our analysis, where k is the number of top pairs, in terms of the ratio of removed edges.

- Community detection. We use k-means—with k-means++ initialization [55]—for community detection, with the number of clusters set to be the actual number of communities. Normalized Mutual Information (NMI) [56] between predicted and true communities is used as the evaluation metric.

For all experiments, the number of embedding dimensions is set to $d = 128$. When searching for the best Markov time, we sweep τ from 1 to 100. An implementation of our framework is available at `https://github.com/zexihuang/random-walk-embedding`.

2.3.3 Results

In this section, we evaluate how different components of the embedding methods affect their performance.

PMI vs autocovariance

We apply standard random-walks and the matrix factorization algorithm and analyze the difference between PMI and autocovariance similarity.

Figure 2.2 shows the node classification performance on undirected datasets (excluding PoliticalBlog as it is a simple binary classification task). We select the Markov time τ with the best performance for the 50% training ratio. Results show that PMI consistently outperforms autocovariance for both *Micro-F1* / *Macro-F1* scores. The average gain is 6.0%/11.3% for BlogCatalog, 14.2%/24.5% and 4.6%/5.2% for Airport with country and continent labels, and 12.9%/20.0% for Wiki-words. This is a piece of evidence that PMI, which is non-linear, is more effective at node-level tasks.

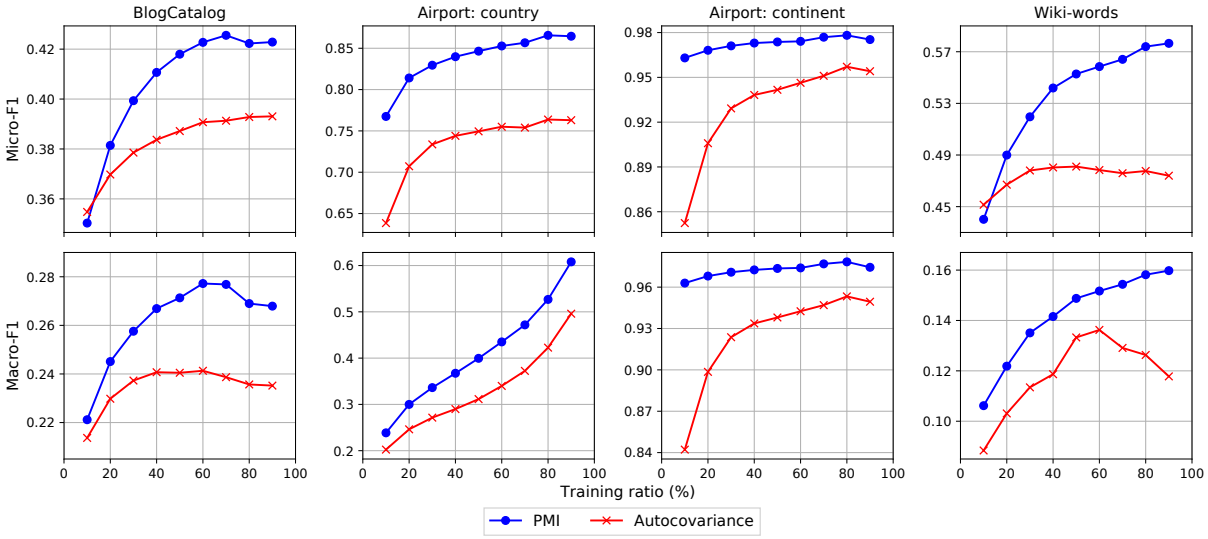


Figure 2.2: Node classification performance comparison between PMI and autocovariance on varying training ratios. PMI consistently outperforms autocovariance in all datasets.

Figure 2.3 shows the results for link prediction, where we select the best Markov time for $k = 100\%$. Autocovariance with dot product ranking consistently outperforms PMI with either ranking scheme in all datasets. The average gains over the best ones are 44.1% in BlogCatalog, 72.9% in Airport, 2.2% in Wiki-words, and 101.4% in PoliticalBlogs. To the best of our knowledge, we are the first to observe the effectiveness of autocovariance on edge-level tasks.

While both dot product-based and classification-based link prediction have been

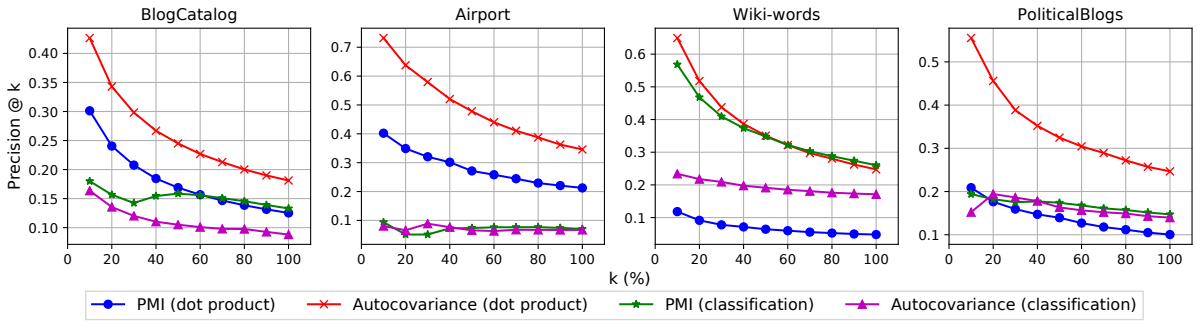


Figure 2.3: Link prediction performance comparison between PMI and autocovariance on varying percentages of top predictions (k). Autocovariance with dot product ranking consistently outperforms PMI (with either ranking scheme) in all datasets.

widely used in the embedding literature, our results show that dot product is a clearly superior choice for autocovariance embedding. It also leads to better or similar performance for PMI embedding for all but the WIKI-WORDS dataset. Thus, we will only show results based on dot products in later experiments.

Results in this section beg the deeper question of why specific similarities and ranking schemes lead to better performance. We will provide theoretical and empirical insights on this in Section 2.4.

Multiscale

In this section, we analyze the effect of different Markov time scales on the embedding performance using standard random-walks and matrix factorization.

Figure 2.4 compares the node classification performance for different similarity measures on varying Markov time scales from 1 to 50. The training ratio is fixed at 50% for all datasets. The metrics have a peak performance at a certain Markov time. However, notice that this aspect has not been given much relevance by previous work on embedding. This is a legacy of DeepWalk [8], which, as we have shown, implicitly applies the *log-mean-exp* of the PMI matrix within a range of Markov times. Thus, we also

show the performance for the moving *log-mean-exp* PMI and mean autocovariance (it is linear) in Figure 2.4. Interestingly, while both mean versions have smoother results, only *log-mean-exp* PMI has a higher peak performance—with gains of 2.2%/4.8% for BlogCatalog, 3.3%/17.4% and 0.4%/0.4% for Airport with country and continent labels, and 7.1%/6.7% for Wiki-words. This shows *log-mean-exp* is indeed an effective approach to combine PMI similarity at different scales. Conversely, we cannot apply a similar strategy for autocovariance.

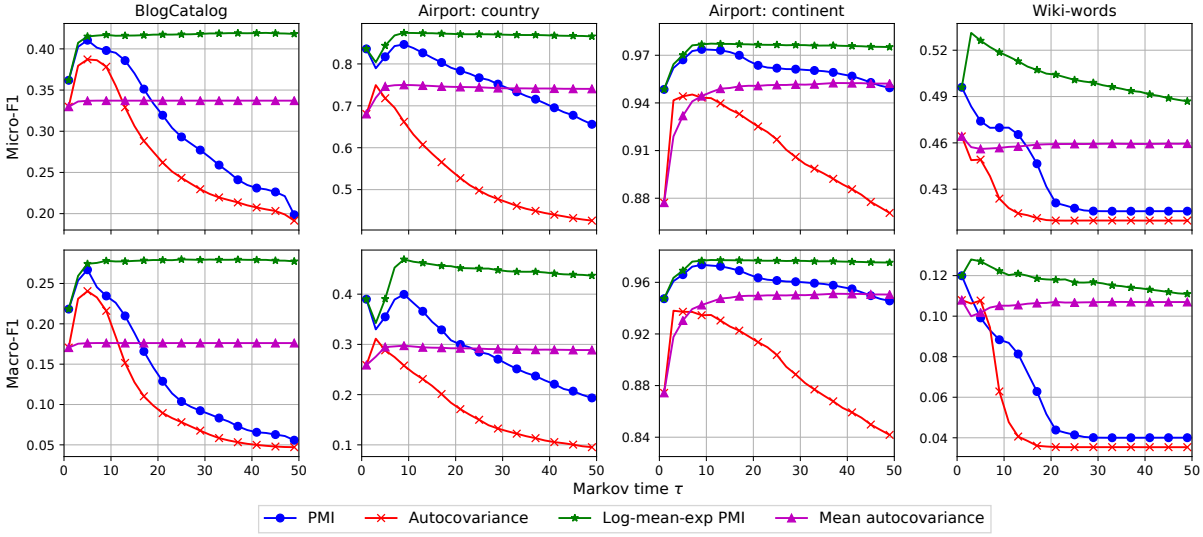


Figure 2.4: Node classification results for PMI, autocovariance, and their moving means on varying Markov times. While both means stabilize the performance for large Markov times, only *log-mean-exp* PMI consistently increases the peak performance.

We observe similar results for community detection (see Figure 2.5). Also included are results for Markov Stability [28], which applies clustered autocovariance for multiscale community detection. For both countries and continents in Airport, (*log-mean-exp*) PMI achieves the best performance. This is another piece of evidence for the effectiveness of PMI on node-level tasks.

We then evaluate the effect of different Markov time scales on link prediction using

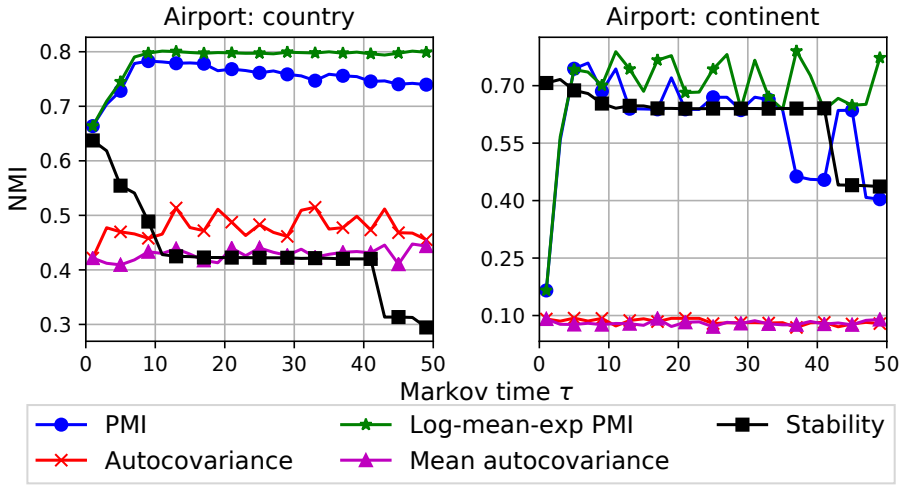


Figure 2.5: Community detection for PMI, autocovariance, their moving means, and Markov Stability on varying Markov times. *Log-mean-exp* PMI outperforms autocovariance and Markov Stability for country and continent levels.

BlogCatalog and Airport. We first note that a moving mean does not improve the performance for either PMI or autocovariance (see Figure 2.6). We hypothesize the reason to be that each edge plays a structural role that is specific to a few relevant scales (e.g., connecting two *mid-sized* communities). To validate it, we first find the best Markov time τ^* in terms of *precision@100%* for autocovariance. Then, for every Markov time τ' larger than τ^*, we compute its added *recall@100%*—i.e., the proportion of correctly predicted edges at τ' that are not predicted at τ^*. We show the relative gain of added recall compared to τ^* and the *precision*@100% for different values of τ' in Figure 2.7. For BlogCatalog, while precision drops as τ' increases, the relative added recall increases to up to 33.4% at $\tau' = 91$ (4,042 new edges added to the 12,104 edges correctly predicted at $\tau^* = 3$). To further understand the roles of those edges, we show the relative added recall for intra-continent edges and inter-continent edges separately for Airport. Most of the gain is for inter-community edges (up to 34.0% at $\tau' = 42$). This observation suggests that link prediction can be improved by accounting for the scale of edges.

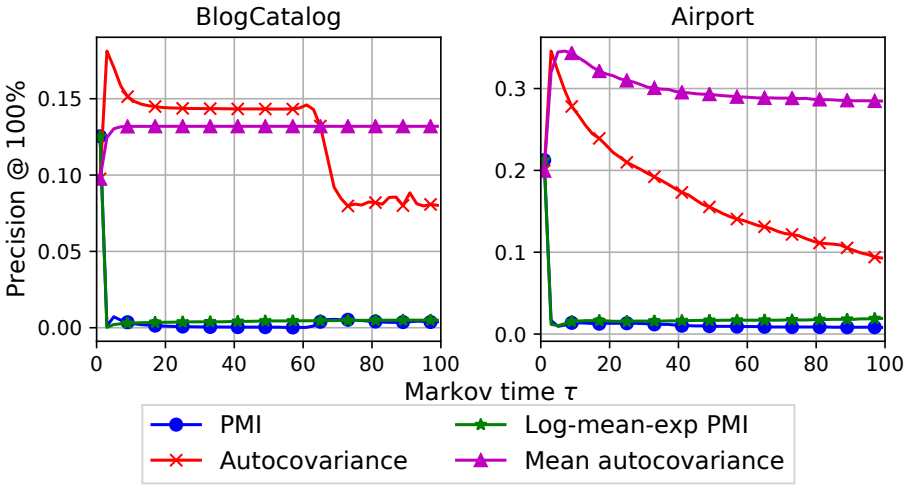


Figure 2.6: Link prediction for PMI, autocovariance, and their moving means on varying Markov times. Neither *log-mean-exp* PMI nor mean autocovariance increases the peak performance.

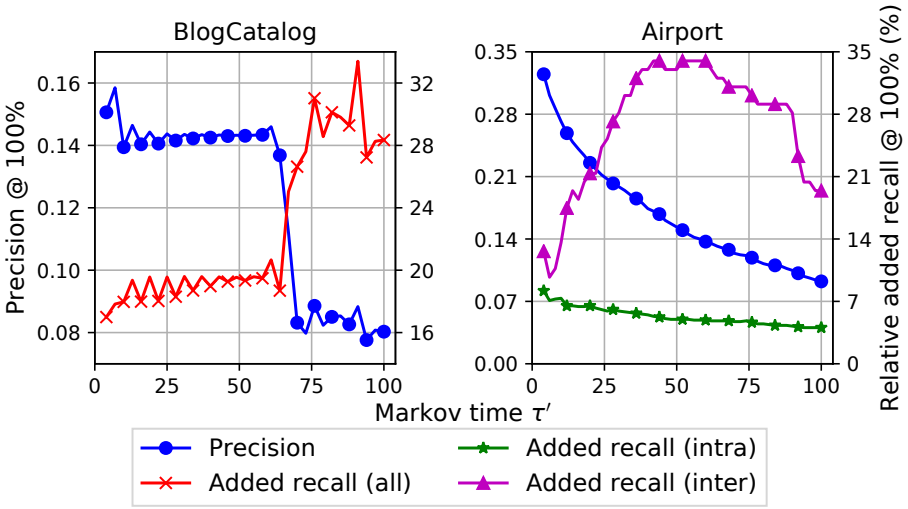


Figure 2.7: Relative added recall for link prediction after adding predictions from a larger Markov time τ' to the best Markov time τ^*. While the precision drops at larger Markov times, they predict a distinct set of true (inter-community) edges from those revealed at the best (and small) time.

Factorization vs sampling

We now switch our focus to evaluating the performance of sampling and matrix factorization algorithms. For PMI, we apply a publicly available node2vec implementation[1]

[1] `https://github.com/eliorc/node2vec/blob/master/node2vec/node2vec.py`

with parameters that make it equivalent to DeepWalk. For autocovariance, we implement our algorithm using the PyTorch framework with the Adam optimizer [57]. Parameters are set based on [9]: 10 walks per node with length 80, 400 epochs for convergence, 1000 walks per batch, and 5 negative samples.

Figure 2.8 shows the link prediction performance where the context window size for both PMI and autocovariance is set to the best Markov time for *precision@100%*. We focus on link prediction because [11] has already shown evidence that matrix factorization is superior to sampling for node classification. Factorization achieves better performance for both datasets and similarity functions. The average gain of *precision@k* for PMI is 277.0%/553.7% on BlogCatalog/Airport, and 240.7%/121.5% for autocovariance.

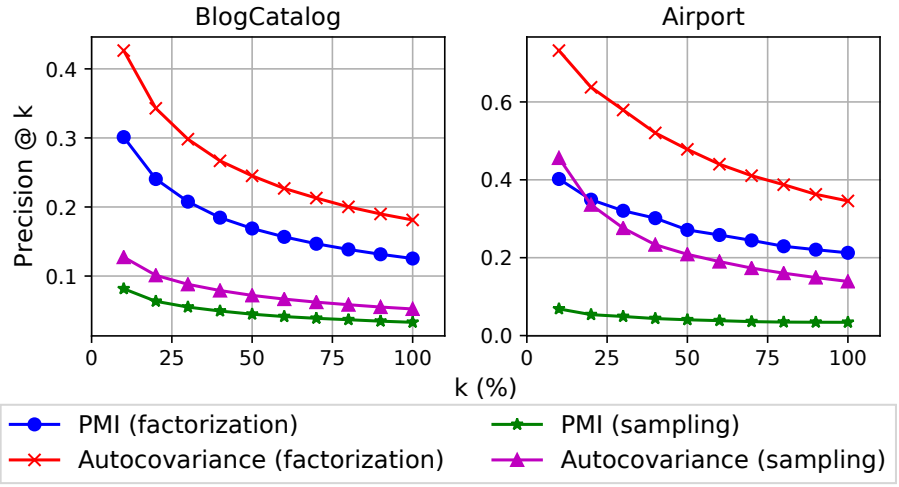


Figure 2.8: Link prediction performance for PMI and autocovariance using matrix factorization and sampling algorithms on varying percentages of top predictions (k). Factorization algorithms achieve the best performance.

Figure 2.9 shows how dot products of 16-D embeddings for Zachary's karate club generated using sampling and matrix factorization algorithms approximate the entries of similarity matrices. We notice that while embeddings produced by both sampling and factorization approaches are correlated with the similarities, matrix factorization achieves a higher correlation.

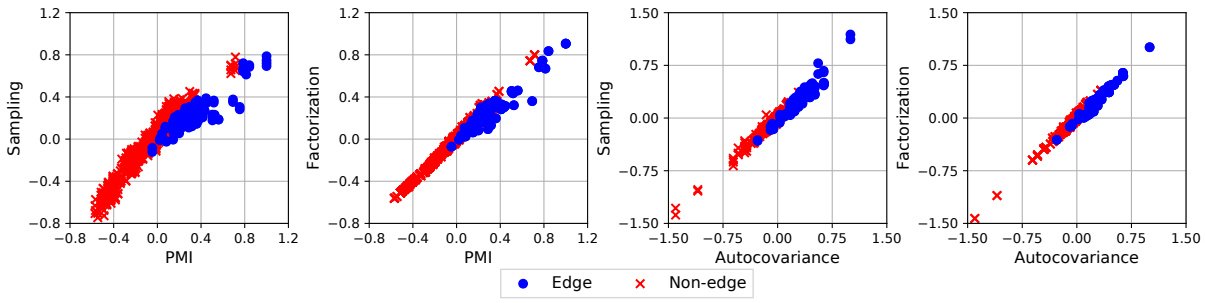


Figure 2.9: Correlation between entries of the similarity matrices (PMI and autocovariance) and the corresponding dot product reconstruction from the embeddings generated via sampling and matrix factorization algorithms. The results are generated using Zachary's karate club network. Both edge and non-edge pairs of nodes are shown. While a clear correlation can be noticed in all cases, matrix factorization methods provide a better approximation of the similarity metrics.

Directed vs undirected

The final part of our evaluation compares embeddings for undirected and directed graphs, which requires different random-walk processes, as discussed in Section 2.2.1. The results for CORA and WIKI-FIELDS are shown in Figure 2.10 and Figure 2.11. Overall, directed embeddings outperform undirected ones for the very top ranked pairs in link prediction, while undirected embeddings are better in node classification.

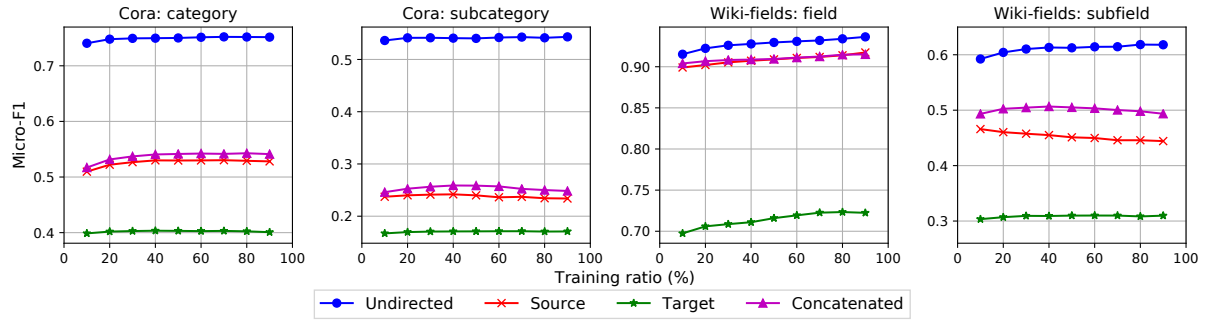


Figure 2.10: Node classification for directed and undirected embeddings with PageRank on varying training ratios. *Macro-F1* scores are not shown here as they follow similar patterns as *Micro-F1* scores. The undirected embedding consistently outperforms both source, target and concatenated directed embeddings in both datasets.

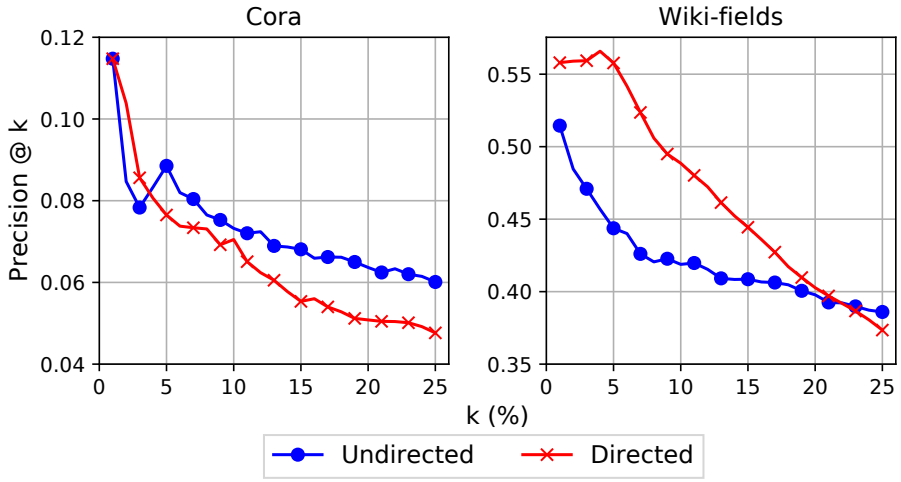


Figure 2.11: Link prediction for directed and undirected embeddings with PageRank on varying top pairs. Directed embedding outperforms undirected embedding for the very top ranked edges.

2.4 Insights on PMI vs Autocovariance

This section provides both theory and experiments supporting the differences in performance achieved by autocovariance and PMI. We will focus on link prediction, for which autocovariance was shown to achieve the best performance. Our analysis might benefit the design of novel embedding methods using our framework.

We will assume that the graph G has community structure [58] and heterogeneous degree distribution [38], which hold for many real graphs and for generalizations of Stochastic Block Models (SBM) [59]. Moreover, let the number of dimensions d be large enough to approximate the similarities well. For simplicity, we consider an SBM instance with two clusters and intra-cluster edge probability p significantly higher than the inter-cluster probability q.

We will use the dot product setting for link prediction [60]. Specifically, we estimate the probability of an edge as $P(e_{u,v}) \propto \max(0, \mathbf{u}^\top \mathbf{v})$, where we have conveniently dropped

the embedding subscripts. Further, for $p \gg q$, $\mathbf{u}^\top \mathbf{v} > 0$ iff $C(u) = C(v)$, where $C(v)$ is the cluster v belongs to. We then have the following observations.

Observation 2.1 *Link prediction based on dot products correlates predicted node degrees and the 2-norms of embedding vectors:*

$$\widetilde{\deg}(u) \approx \sum_{v \in C(u) - \{u\}} \mathbf{u}^\top \mathbf{v} = \|\mathbf{u}\|_2 \sum_{v \in C(u) - \{u\}} \|\mathbf{v}\|_2 \cos(\theta_{\mathbf{u},\mathbf{v}})$$

where $\cos(\theta_{u,v})$ *is the cosine of the angle between* $\mathbf{u}$ *and* $\mathbf{v}$.

The above property follows from the community structure, as the majority of the edges will be within communities. We now look at the norms of vectors generated by autocovariance and PMI.

Observation 2.2 *For autocovariance similarity:*

$$\|\mathbf{u}\|_2^2 = \pi(u)([M^\tau]_{u,u} - \pi(u)) = \frac{\deg(u)}{2m}\left([M^\tau]_{u,u} - \frac{\deg(u)}{2m}\right)$$

Norms for autocovariance depend on two factors. The first is proportional to the actual node degree, while the second expresses whether u belongs to a community at the scale τ. For SBM, there exists a τ such that $[M^\tau]_{u,u}$ is close to $\deg(u)/m$ in expectation. That is the case when the process is almost stationary within $C(u)$, with $m/2$ expected edges, but not in the entire graph. It implies that the embedding norms are proportional to actual node degrees. Combining this with Observation 1, we can conclude that autocovariance embedding predicts degrees related to the actual ones.

Observation 2.3 *For PMI similarity:*

$$\|\mathbf{u}\|_2^2 = \log\left(\pi(u)[M^\tau]_{u,u}\right) - \log\left(\pi^2(u)\right) = \log\left(\frac{2m[M^\tau]_{u,u}}{\deg(u)}\right)$$

Notice that as $[M^\tau]_{u,u}$ approaches to $\deg(u)/m$, $\|\mathbf{u}\|_2$ becomes constant. As a consequence, different from autocovariance, norms for PMI embeddings are not directly related with node degrees.

In Figure 2.12, we provide empirical evidence for Observations 2 and 3—the correlation between actual degrees and embedding norms for BlogCatalog, Airport, Wiki-words, and PoliticalBlogs. The correlation for autocovariance is significantly higher than that for PMI, showing the ability of autocovariance to capture heterogeneous degree distributions.

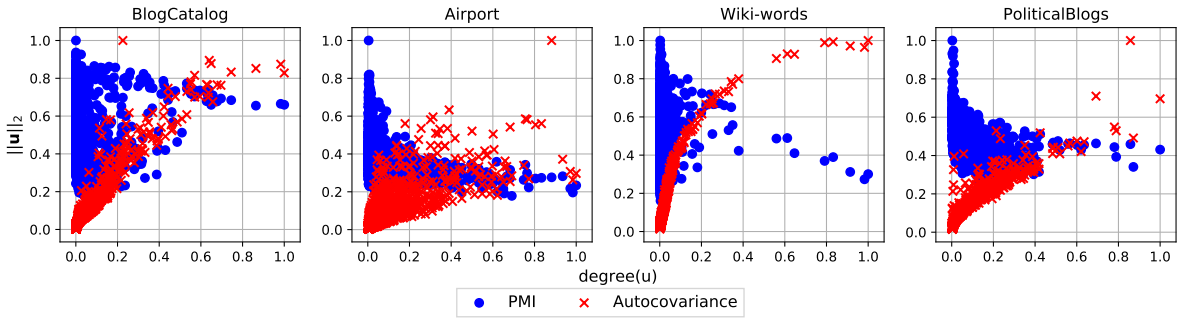
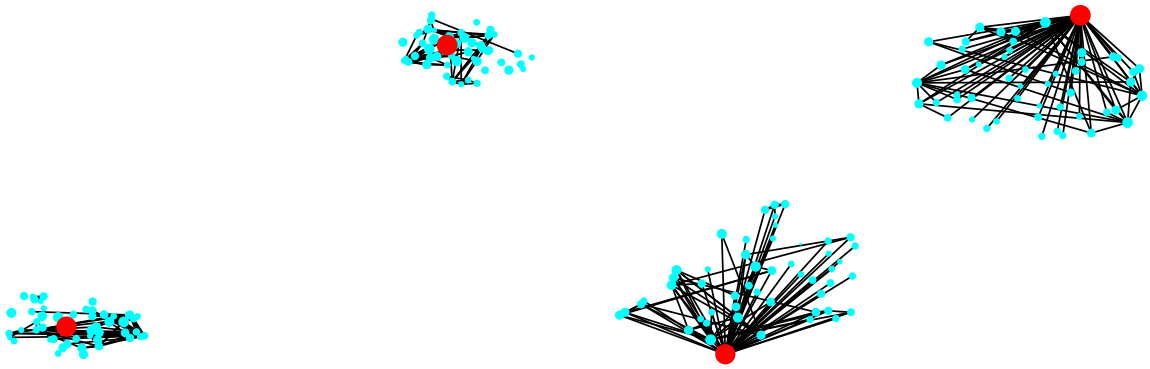


Figure 2.12: Correlation between (max normalized) degrees and 2-norms of embedding vectors based on PMI and autocovariance. Autocovariance produces embeddings with norms that are well correlated with node degrees in both datasets.

Figure 2.13 shows the predicted links for an instance of SBM ($p = 0.15$, $q = 0.02$) with two hubs—generated by merging 20 nodes inside each community. Visualization is based on t-SNE [61] projection from 16-D embeddings. Around half of the edges (97 out of top 200) are predicted to be connected to the hubs using autocovariance, but none using PMI. This example shows how autocovariance with dot product ranking enables state-of-the-art link prediction.

The analysis presented here does not apply to node-level tasks (node classification and clustering), where the degree distribution does not play a major role. In fact, Figure 2.13 shows that PMI produces a better separation between clusters with hubs for each cluster



(a) PMI (b) Autocovariance

Figure 2.13: Edges predicted using PMI and autocovariance embeddings for a synthetic graph with community structure and hubs (one per community, shown in red). Autocovariance is more effective at capturing the hubs in the graph.

positioned in the middle, which is desired for node-level tasks. This property might be explained by the non-linearity of PMI.

2.5 Related Work

We refer to [23, 24, 25, 26, 27] for an overview of graph embedding and representation learning. The general problem of embedding vertices of a graph into a vector space can be traced back to much earlier work on embedding metric spaces [62], spectral graph theory [42], nonlinear dimensionality reduction [63] and graph drawing [64]. However, the more recent appeal for graph embedding methods coincided with the renewed interest in deep learning, specially the skip-gram model for text, such as word2vec [34]. DeepWalk [8], which introduced the idea of using random-walks as graph counterparts of sentences in skip-gram, is considered the pioneering work on random-walk based graph embedding. Subsequent extensions have led to an extensive literature [9, 13, 10, 15]. Random-walk based embedding models have also been proposed for heterogeneous [65], dynamic [66] and multilayer networks [67].

Previous work has proposed embeddings based on specific processes. For instance, node2vec [9] applies a biased random-walk process that is flexible enough to capture homophily and structural similarity. In [15], the authors propose a variation of DeepWalk with PageRank [40] instead of the standard random-walks. These studies motivate generalizing embedding to other processes, such as the Ruelle-Bowen random-walk, which has maximum entropy [68], and even continuous-time random-walks [30, 69].

With the exception of [12], random-walk based embeddings cannot generalize—either explicitly or implicitly—to other similarities besides Pointwise Mutual Information (PMI) [11]. As an alternative, autocovariance similarity is the basis of Markov Stability [28, 36, 69], an effective multiscale community detection algorithm validated in many real networks. In [12], the authors introduced a general embedding scheme based on control theory, of which autocovariance is a special case. However, they neither contextualized it with other existing embedding approaches nor provided any theoretical or empirical evaluation on any particular task. A link prediction algorithm based on autocovariance was introduced in [70], but it does not produce embeddings. Integrating autocovariance into the broader framework of random-walk embedding—with different processes and embedding algorithms—and investigating its properties both theoretically and empirically is a contribution of our work.

For some time, graph embedding models were divided into those based on skip-gram [8, 9, 13], which are trained using sampled walks and stochastic gradient descent (or its variations) and those based on matrix factorization [10, 18]. It was later shown that, under certain assumptions, skip-gram based text embedding models perform matrix factorization implicitly [35]. More recently, Qiu et al. [11] showed that DeepWalk, node2vec and LINE [13] also perform implicit matrix factorization. Our framework incorporates both sampling-based and matrix factorization algorithms and we show how the former can generalize to similarities beyond PMI.

Multiscale graph embedding based on random-walks has attracted limited interest in the literature [19, 21], though many real-life networks are known to have a multiscale (or hierarchical) structure [71, 72]. Notice that we focus on graphs where scales are defined by clusters/communities. This is different from graphs where vertices belong to different levels of a hierarchy, such as trees, for which hyperbolic embedding [73] is a promising approach.

2.6 Conclusion

We have introduced a framework that provides a renewed bearing on random-walk based graph embedding. This area has capitalized on the close connection with skip-gram but has also been biased by some of its design choices. Our framework has enabled us to scrutinize them, which will benefit researchers and practitioners alike. In the future, we want to explore alternative choices of components in our framework, as well as extending it to graphs with richer information, such as signed, dynamic, and attributed graphs.

Chapter 3

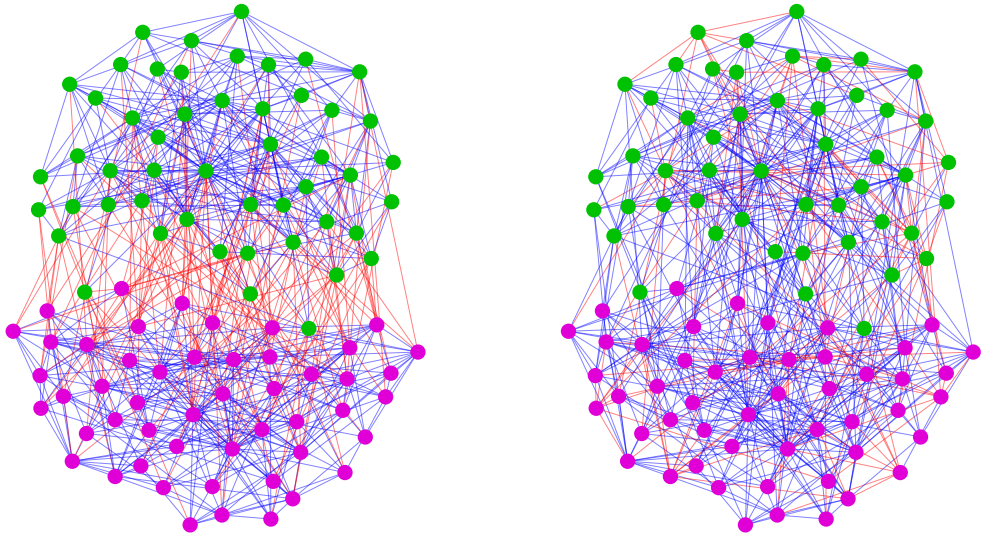
Polarized Embedding for Signed Networks

3.1 Introduction

Social media has made our world more polarized [74, 75, 76]. The events surrounding the 2016 U.S. election [77] and, more recently, the tragic U.S. Capitol riot [78] and spread of COVID-19 misinformation [79, 80], have illustrated the dangers of a deeply ideologically divided society. But if technology has led to the rise of polarization, can it also help us to solve it? More specifically, can recent advances in representation learning [8, 31] help us to address online polarization? One could argue these methods should be as effective for predicting conflicts as they are for recommending connections and content in social media platforms [26]. However, we will show that polarization leads to new challenges for representation learning.

Signed graphs are a powerful tool for analyzing social polarization, capturing both positive (friendly) and negative (hostile) connections between entities. They have been used to model relationships between politicians in the U.S. Congress [81] and interactions

between Twitter users on political matters [82], both of which are known to harbor polarization [83, 84]. In signed graphs, polarization is often related to the emergence of conflicting communities [85], where nodes within each community form dense positive connections and nodes across different communities are sparsely connected via negative links. Figure 3.1 shows two signed graphs with the same underlying topology (based on the LFR benchmark [86]) but different link signs. The left graph is more polarized as all of its negative links connect the two communities, while link signs in the right graph are unrelated to the community structure.



(a) LFR-polarized (b) LFR-unpolarized

Figure 3.1: Two synthetic graphs with the same underlying topology but different link signs. Node colors depict communities and link colors depict signs. (a) is more polarized than (b) as its link signs are related to the community structure—negative links connecting two polarized communities.

To fight polarization, we first need to measure it. Existing measures [85, 87] are defined as objective functions for community detection, requiring community memberships as input. This dependence makes those measures less useful for real-world graphs, where ground-truth memberships are often unknown. To solve this problem, we propose a novel polarization measure based on the correlation between unsigned and signed random-walk

dynamics. By varying the random-walk length, our measure naturally captures polarized community structure at different scales and does not rely on specific partitions. We will demonstrate its effectiveness in characterizing both node and graph-level polarization.

But how can we fight polarization in a network? For unsigned graphs, one approach is to bridge communities to reduce the effect of echo chambers [88]. Signed graphs open new possibilities in this endeavor. In particular, if one can identify potential hostile links that are likely to be formed in the future, they can take preventive measures to reduce further polarization. The success of this approach relies on the accuracy of signed link prediction [89, 90, 91], where both existence and signs of future links are inferred. However, while unsigned and signed graph embedding methods have shown success at the related link-level tasks of (unsigned) link prediction [9, 1] and sign prediction [29, 92], they cannot be easily combined for signed link prediction in polarized graphs. That is because negative links mostly connect antagonistic communities and are much sparser than positive links, making it nearly impossible to predict their existence with unsigned embedding methods. And without link existence information, even a perfect sign prediction oracle is unable to predict the negative links.

To address the challenges mentioned above, we propose POLE, a novel signed embedding method for polarized graphs. The key feature that distinguishes our method from existing ones is that it captures signed and topological similarities jointly. Specifically, it guarantees that positively related pairs are more similar than unrelated topologically distant pairs, which are in turn more similar than negatively related pairs. This is accomplished by leveraging the signed random-walk that incorporates social balance theory [93] and extending autocovariance similarity [28, 12] to signed graphs. In this way, negative links can be predicted as the most dissimilar node pairs in the graph, at the other end of the similarity spectrum.

To summarize, our main contributions are:

- We design a novel partition-agnostic polarization measure for signed graphs based on the signed random-walk.
- We analyze how existing signed embedding methods fail in signed link prediction for polarized graphs.
- We propose POLE, a novel signed embedding approach for polarized graphs that captures both topological and signed similarities via signed autocovariance.
- We conduct an extensive experimental evaluation of our method on six real-world signed graphs. Results show that POLE significantly outperforms state-of-the-art methods in signed link prediction, especially for negative links.

3.2 Random-walk on Signed Graphs

We introduce our formulation for random-walks on signed graphs, the basis for our polarization measure and embedding algorithm.

3.2.1 Notations

A signed undirected weighted graph is a tuple $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots n\}$ denotes the set of n nodes and $\mathcal{E} = \mathcal{E}_+ \cup \mathcal{E}_-$ denotes the set of m links, with positive and negative signs. The graph is represented by a signed weighted adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{uv} > 0$ (or < 0) if a positive (or negative) link of weight $|A_{uv}|$ connects nodes u and v, and $A_{uv} = 0$ otherwise. We also use the absolute adjacency matrix $|A|$ to construct the degree matrix/vector D/d with $D_{uu} = d_u = \sum_v |A|_{uv}$ as the degree of node u.

3.2.2 Signed Random-walk

Random-walks are widely used for unsigned graph embedding due to their ability to capture topological similarity at multiple structural scales. While there have been several attempts to extend random-walks to signed graphs [94, 95], we will define our own version to capture topological and signed similarity jointly and explicitly guarantee *polarized similarity consistency*—we will discuss this important property in more detail in the next subsection.

We start by recalling random-walks for unsigned graphs. A walk l is a sequence of nodes $\langle w_0, w_1, \dots w_t \rangle$ where $(w_\tau, w_{\tau+1}) \in \mathcal{E}$. The transition probability of the walk is the product of stepwise ones:

$$\begin{aligned} \text{Prob}(l) &= \prod_{(w_\tau, w_{\tau+1}) \in l} \text{Prob}(w_{\tau+1} | w_\tau) \\ &= \prod_{(w_\tau, w_{\tau+1}) \in l} |A|_{w_\tau w_{\tau+1}} / d_{w_\tau} \end{aligned} \tag{3.1}$$

Then, the t-step random-walk transition probability from node u to v can be expressed as the sum of the transition probabilities of all length-t walks between u and v, denoted as $\text{Walk}(u, v, t)$:

$$|M|_{uv}(t) = \sum_{l \in \text{Walk}(u,v;t)} \text{Prob}(l) \tag{3.2}$$

which serves as a measure of topological similarity between u and v. The Markov time t controls the scale of the walk.

For signed graphs, we keep transition probabilities of walks for topological similarity and add an inferred sign for each walk to capture signed similarity, leading to

$$M_{uv}(t) = \sum_{l \in \text{Walk}(u,v;t)} \text{Sign}(l) \, \text{Prob}(l) \tag{3.3}$$

where $\text{Sign}(l) = \text{Sign}(\prod_{(w,w')\in l} A_{ww'})$ determines the sign of the walk l between u and v. We leverage the well-established social balance theory [93] to infer the sign of the walk, which states the famous rule that "an enemy of my enemy is my friend" among other rules. Figure 3.2 shows examples of signed transitions.

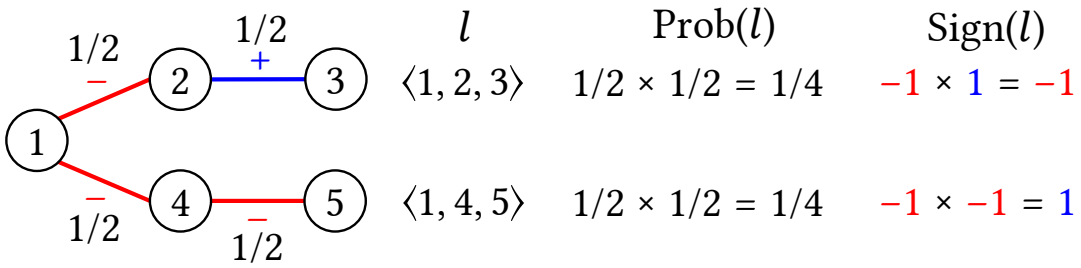


Figure 3.2: Examples of signed walks with corresponding transition probabilities (Prob) and inferred signs (Sign).

Note that after adding signs to walks, $M_{uv}(t) \in [-1, 1]$ is no longer a probability but rather captures a notion of signed similarity between u and v. In matrix form, we will still call $M(t) \in \mathbb{R}^{n\times n}$ the signed random-walk transition matrix even though it is not stochastic. A nice property of $M(t)$ is that it can be conveniently expressed in terms of the signed adjacency and degree matrices:

$$M(t) = \begin{cases} (D^{-1}A)^t & \text{for discrete random-walks} \\ \exp(-(I - D^{-1}A)t) & \text{for continuous random-walks} \end{cases} \tag{3.4}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix. We will use the continuous version of the transition matrix in the rest of the work due to its finer granularity of the Markov time ($t \in [0, +\infty)$ instead of $t \in \mathbb{N}_0$).

3.2.3 Similarity Consistency

The key advantage of our signed random-walk is that it guarantees *polarized similarity consistency*, a property critical to signed link prediction in polarized graphs. The *consistency* of similarity states which type of node pairs should be more similar than other types. For example, unsigned embedding methods satisfy:

Property 3.1 (Topological similarity consistency) *Topologically close nodes are more similar than topologically distant ones.*

And signed embedding methods guarantee:

Property 3.2 (Signed similarity consistency) *Positively related nodes are more similar than negatively related ones.*

These properties enable the corresponding link-level downstream tasks, namely (unsigned) link prediction and sign prediction.

The transitions of our signed random-walk as a similarity metric have a stronger consistency guarantee:

Property 3.3 (Polarized similarity consistency) *Positively related node pairs are more similar than unrelated topologically distant pairs, which are in turn more similar than negatively related pairs.*

Property 3.3 is a direct implication of the transition probability defined in Equation 3.3. Large positive/negative values of $M_{uv}(t)$ indicate that u and v are connected via mostly positive/negative paths while small values mean that they are unrelated and topologically distant. Property 3.3 also holds for a similarity defined by dot products between columns of M. More specifically, $\langle M_{:u}, M_{:v} \rangle$ is large and positive (or negative) if and only if u and v have transitions of same (or different) sign(s) to a common set of nodes. And the similarity is small if their transitions reach a distinct set of nodes.

3.3 A Measure of Polarization

In this section, we propose and evaluate a measure of polarization based on the signed random-walks we have just introduced.

3.3.1 Random-walk Based Polarization

A signed network is polarized if it comprises antagonistic communities with dense positive connections in each community and sparse negative ones across communities. Unlike previous works [85, 87] that define polarization as a quality measure of graph partitions, our aim is to measure polarization based solely on graph structure. To achieve this, we apply a soft characterization of the polarized community structure in the graph based on signed random-walks.

The key observation that supports our polarization measure is that polarization increases the correlation between unsigned and signed random-walk dynamics. Consider the two graphs shown in Figure 3.3. While they share the same underlying topology, (a) is more polarized than (b) as its negative link connects the two communities. We then analyze how their signed random-walk transitions differ from unsigned ones. For node 1 in graph (a), only signs of its inter-community transitions are changed, whose magnitudes are smaller, if not negligible, compared to the intra-community transitions. On the other hand, the negative link in the less polarized graph (b) significantly affects the signed intra-community transitions. As a consequence, the signed and unsigned transitions in the polarized graph are more correlated than in the less polarized one.

We define the *node-level polarization* as the (Pearson) correlation between a node's signed and unsigned random-walk transitions:

$$\mathrm{Pol}(u;t) = \mathrm{corr}(|M|_{:u}(t), M_{:u}(t)) \tag{3.5}$$

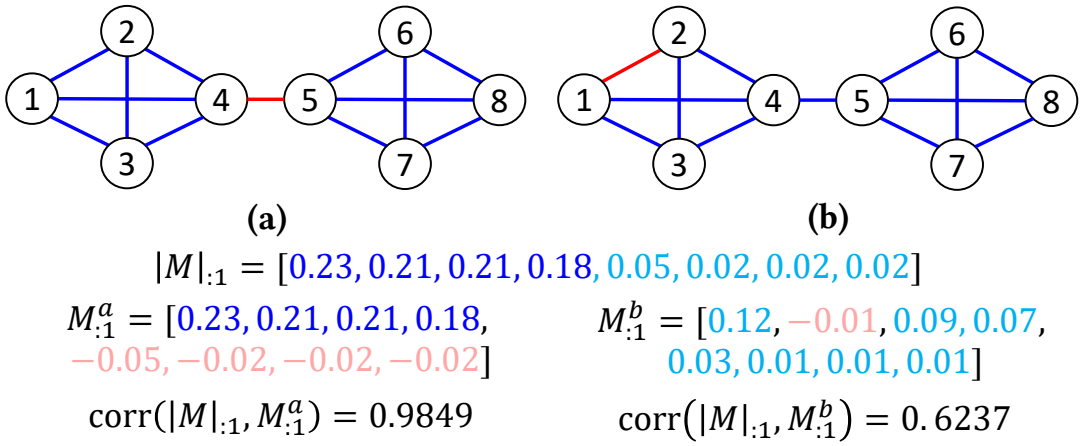


Figure 3.3: Illustration of our polarization measure based on signed random-walk dynamics ($t = 3$). The transitions for unsigned ($|M|$) and signed (M) random-walks are more correlated in the polarized network (a), where a negative link connects two antagonistic communities. On the other hand, the correlation is lower in the less polarized network (b).

We can then define the *graph-level polarization* as the mean node-level polarization for all nodes in the graph:

$$\text{Pol}(G; t) = \underset{u \in G}{\text{mean}}(\text{Pol}(u; t)) \tag{3.6}$$

The random-walk based polarization proposed here has two main advantages. First, it is partition-agnostic and does not depend on the availability of ground-truth communities or the outcome of community detection algorithms. Second, it can measure polarization at different structural scales by tuning the Markov time t—large t captures polarization between macro-level communities (e.g., hostility between political parties) while small t measures it at a micro scale (e.g., disagreement between factions within a party). In the next subsection, we will demonstrate that our measure is effective in characterizing both node and graph-level polarization.

3.3.2 Polarization of Real-world Graphs

Node-level polarization

We apply our polarization measure to characterize nodes in a political network, CONGRESS [81]. Signed links in this network represent (un/)favorable interactions between U.S. congresspeople on the House floor in 2005. The statistics of the network are shown in Table 3.3.

We rank all nodes by their polarization scores at a fixed $t = 10$. The top 20 least and most polarized nodes are shown in Table 3.1 and Table 3.2. The congressperson with the smallest score (-0.6542) is Henry Cuellar, a self-described moderate-centrist [96]. As a Democrat, he voted with President Trump (Republican) nearly 75% of the time, advocating his fellow Democrats to embrace a more conservative voting record [97]. The second least polarized person is Jane Harman (with -0.5376). She was described as a centrist, particularly on defense and intelligence issues [98]. She was also once called the best Republican in the Democratic Party [99]. In general, we found that our polarization measure captures the political views (centrists vs. extremists) of the congresspeople based on their signed interactions.

Graph-level polarization

We also apply our graph-level polarization measure to characterize real-world signed graphs. In addition to CONGRESS, we consider five other networks, with their statistics summarized in Table 3.3:

- WOW-EP8 [100]: Interaction network of editors in the eighth legislature of the European Parliament. Link signs indicate whether they collaborate or compete with each other.

Congressperson	State	Party	Score
Henry Cuellar	Texas	Democratic	-0.6542
Jane Harman	California	Democratic	-0.5376
Curt Weldon	Pennsylvania	Republican	-0.4381
Dutch Ruppersberger	Maryland	Democratic	-0.4318
Jim Moran	Virginia	Democratic	-0.3832
Dave Obey	Wisconsin	Democratic	-0.3588
Wayne Gilchrest	Maryland	Republican	-0.3503
Duke Cunningham	California	Republican	-0.3248
Al Edwards	Texas	Democratic	-0.3063
Lincoln Davis	Tennessee	Democratic	-0.2901
Joe Barton	Texas	Republican	-0.1492
Charles Bass	New Hampshire	Republican	-0.1381
Mary Bono	California	Republican	-0.1381
Charlie Dent	Pennsylvania	Republican	-0.1381
Nick Rahall	West Virginia	Democratic	-0.1357
Bill Young	Florida	Republican	-0.1250
Roger Wicker	Mississippi	Republican	-0.1189
Charles Rangel	New York	Democratic	-0.0926
Candice Miller	Michigan	Republican	-0.0743
Jim Kolbe	Arizona	Republican	-0.0716

Table 3.1: Top 20 least polarized members of the U.S. Congress in terms of our random-walk based polarization measure.

Congressperson	State	Party	Score
Steve Buyer	Indiana	Republican	0.9897
James T. Walsh	New York	Republican	0.9877
Charles H. Taylor	North Carolina	Republican	0.9851
Zach Wamp	Tennessee	Republican	0.9776
Tom Cole	Oklahoma	Republican	0.9747
Geoff Davis	Kentucky	Republican	0.9261
Steve Israel	New York	Democratic	0.9019
Marsha Blackburn	Tennessee	Republican	0.8875
Virginia Foxx	North Carolina	Republican	0.8814
Steve King	Iowa	Republican	0.8516
Joe Crowley	New York	Democratic	0.8444
Virgil Goode	Virginia	Republican	0.8175
Robert E. Cramer	Alabama	Democratic	0.8117
Rubén Hinojosa	Texas	Democratic	0.8030
Carolyn McCarthy	New York	Democratic	0.8030
David Dreier	California	Republican	0.7874
Christopher Cox	California	Republican	0.7868
Jo Ann Davis	Virginia	Republican	0.7867
John E. Peterson	Pennsylvania	Republican	0.7867
Ray LaHood	Illinois	Republican	0.7793

Table 3.2: Top 20 most polarized members of the U.S. Congress in terms of our random-walk based polarization measure.

- Bitcoin-Alpha and Bitcoin-OTC [101]: Trust networks of Bitcoin traders on the platforms *Bitcoin Alpha* and *Bitcoin OTC*. Link weights and signs are based on users' rating of each other.

- Referendum [82]: Interaction network of Twitter users on the 2016 constitutional referendum in Italy. Link signs are inferred from the stance of the users.

- Wiki-RfA [102]: Voting network of Wikipedia users on adminship. Link signs indicate whether users vote for/against each other.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{E}_-\rvert/\lvert\mathcal{E}\rvert$
Congress	219	523	20.46%
WoW-EP8	789	116,009	18.63%
Bitcoin-Alpha	3,772	14,077	9.31%
Bitcoin-OTC	5,872	21,431	14.71%
Referendum	10,864	251,396	5.09%
Wiki-RfA	11,275	169,925	22.04%

Table 3.3: An overview of the datasets.

As reference for graph-level polarization, we construct two synthetic graphs, one polarized and the other unpolarized, with the same underlying topology but different link signs, as shown in Figure 3.1. The topology is based on the LFR benchmark [86] with the following parameters: two structural communities with 50 nodes, an average node degree of 12, and an inter-community link ratio of 0.15. Then, for the polarized version (LFR-polarized), we assign link signs fully based on the structural communities—negative for inter-community links and positive for intra-community links. For the unpolarized version (LFR-unpolarized), we first assign nodes to two random communities with the same cut size as structural communities. We then assign link signs based on the random communities following the same rule. In this way, both graphs have the same proportion of negative links and the same social balance—proportion of closed triangles

in the graph that satisfy the social balance theory [103]—of 1.0.

The polarization and social balance for the real-world graphs along with the two synthetic ones are shown in Figure 3.4, with the Markov time t selected based on signed link prediction performance (details in Section 3.5.1). As we see, most real-world graphs are at the same level of polarization as LFR-POLARIZED. On the other hand, their social balance is not directly related to polarization, contrary to as one might think [76]. The only dataset that falls behind in polarization is WIKI-RFA, which might be due to the more collaborative nature of its inter-community interactions. As we will discuss in the next section, effective link prediction for those polarized graphs requires a novel embedding scheme that satisfies *polarized similarity consistency*.

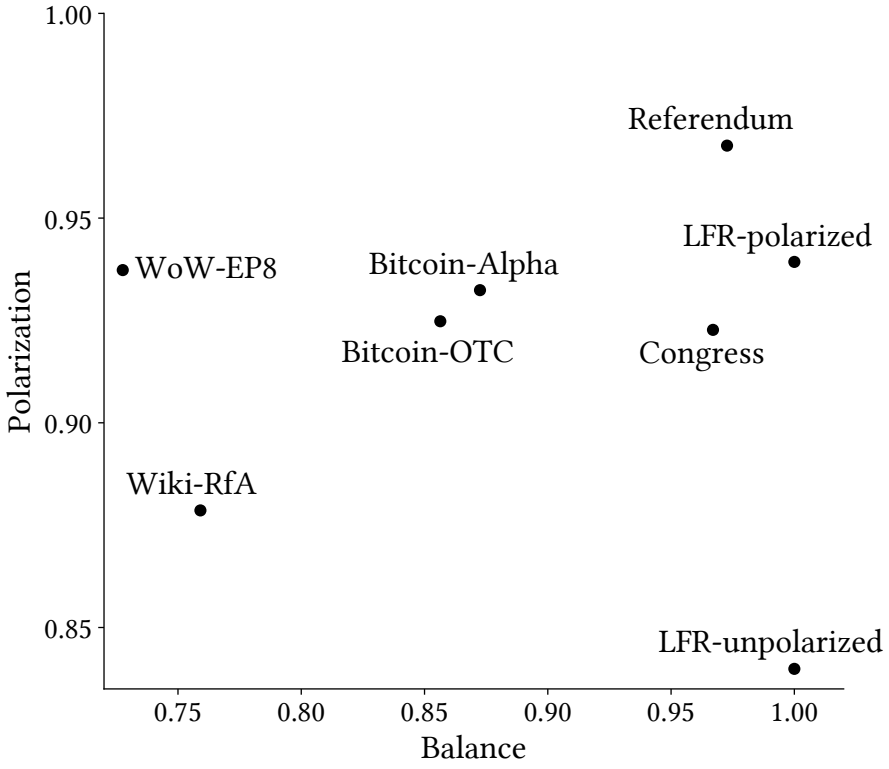


Figure 3.4: Polarization and social balance of real-world graphs, with reference to synthetic ones. Most real-world graphs are as polarized as the synthetic polarized one—LFR-POLARIZED.

3.4 Polarized Embedding for Networks

Now that we have shown that many real-world graphs are polarized, here, we propose a novel embedding method for effectively predicting signed links in polarized graphs. We first demonstrate that existing embedding methods are incapable of this task due to their weak similarity consistency (see Section 3.2.3). We then introduce polarized embedding based on autocovariance and matrix factorization that addresses the limitations of existing approaches.

3.4.1 Limitation of Existing Methods

The objective of signed link prediction [104, 105] is to predict both the existence and the signs of future links given the observed graph. Most of existing signed embedding methods focus on the second half of the task, sign prediction, which is specific to signed graphs. The first half, link prediction, is a common downstream task for unsigned embedding methods. Combining them seems like a viable solution for the task. However, this approach fails to predict negative links in polarized graphs because they are sparse and mostly appear as inter-community connections. They are hard, if not impossible, to be predicted by unsigned link prediction algorithms based on topology only. Without the existence of links known a priori, sign prediction itself is incapable of predicting negative links.

We identify the key weakness of existing embedding methods as that they only preserve weak similarity consistency. In particular, signed embedding methods [29, 106, 92, 107, 105, 31] just ensure *signed similarity consistency*—positively related pairs are separable from negatively related pairs—for sign prediction. And similarly, unsigned embedding methods [8, 9, 1] only guarantee the *topological similarity consistency*—topologically close node pairs are separable from distant pairs—for link prediction. While positive links are

detectable as the intersection of positive and topologically close pairs, negative links in polarized graphs would remain hidden with other topologically distant pairs. Figure 3.5 (a) and Figure 3.5 (b) illustrate this idea. They show the distributions of reconstructed similarity of positively connected pairs, negatively connected pairs, and disconnected pairs for LFR-POLARIZED via dot products of unsigned (RWE [1]) and signed embedding (ROSE [31]), respectively. While both embedding methods are able to capture their respective similarity consistency, negative links are always inseparable from disconnected pairs. This motivates us to design an embedding method with a stronger similarity consistency that enables the separation of negative pairs from the others. We describe such an embedding method next.

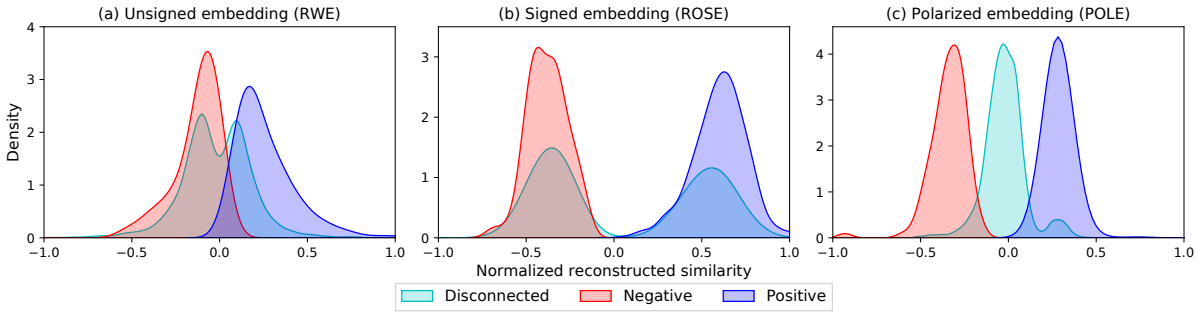


Figure 3.5: Distributions of the reconstructed similarity for different types of node pairs in LFR-POLARIZED. Polarized embedding (c) enables separation of negatively connected pairs from the others while both (a) unsigned embedding and (b) signed embedding fail to do so.

3.4.2 The Solution: Polarized Embedding

Our solution to the separability of negative pairs is polarized embedding, a novel embedding scheme that captures *polarized similarity consistency*, the strongest consistency among the three. This consistency guarantees that negatively related pairs are more dissimilar than unrelated topologically distant pairs (thus "polarized"), which are

more dissimilar than positively related pairs. In this way, the negatively related pairs stand out at the other end of the similarity spectrum, easily separable from others. In the following subsections, we introduce our polarized embedding method, POLE, that applies signed autocovariance similarity and matrix factorization.

Signed autocovariance similarity

POLE is based on the signed random-walk introduced in Section 3.2.2. As discussed in Section 3.2.3, both the entries and the dot products of columns of the signed random-walk transition matrix $M(t)$ can be viewed as similarity metrics that satisfy *polarized similarity consistency*. Therefore, one can directly use a low-rank representation of them—for example, the matrix factorization of $M(t)$ or $M(t)^\top M(t)$—as embeddings. However, those embeddings may not be effective for link-level inference. Instead, we propose to take advantage of the random-walk based similarity metrics that have been well understood in unsigned embedding. In particular, it has been shown that autocovariance similarity [28] enables state-of-the-art performance in unsigned link prediction by incorporating node degree information [1]. This coincides with our goal of predicting signed links in polarized graphs. Thus, we first extend autocovariance similarity to signed graphs.

The unsigned autocovariance similarity is built upon the co-visiting probability for node pairs in a walk. However, as defined in our signed random-walk, $M(t)$ is not a stochastic matrix and does not encode probabilities. Instead, we resort to another interpretation of autocovariance—a centered dynamic similarity metric [12]. The dynamic similarity based on $M(t)$ is

$$R(t) = M(t)^\top W M(t) \tag{3.7}$$

where $W \in \mathbb{R}^{n\times n}$ is a weight matrix. In the unsigned case, selecting $W = \Pi - \pi\pi^\top$ makes $R(t)$ equivalent to the autocovariance similarity, where $\pi \in \mathbb{R}^n$ is the stationary

distribution of the unsigned random-walk and $\Pi = \text{diag}(\pi)$. For the signed case, $M(t)$ is singular if and only if the graph is perfectly balanced [108]. This means that there is no "stationary distribution" for signed random-walks on most real-world graphs. However, as [69] points out, the role of stationary distribution in the similarity formulation is to provide a centrality measure—in the unsigned case, the degree centrality. Thus, we can also use node degrees to construct the weight matrix:

$$W = \frac{1}{\text{vol}(G)} D - \frac{1}{\text{vol}(G)^2} dd^\top \tag{3.8}$$

where $\text{vol}(G) = \sum_u d_u$. And substituting this weight matrix into Equation 3.7 leads to the signed autocovariance similarity matrix. To the best our knowledge, we are the first to extend random-walk based similarity metrics to signed graphs in a principled manner.

Matrix factorization

We are now ready to introduce how to generate embedding based on signed autocovariance. Let $\mathbf{u}_u \in \mathbb{R}^k$ be the embedding of node u and $U = (\mathbf{u}_1, \ldots, \mathbf{u}_n)^\top \in \mathbb{R}^{n \times k}$ be the embedding matrix. We use the dot product in the embedding space to preserve the signed autocovariance similarity R:

$$\begin{aligned} U^* &= \arg\min_U \sum_{u,v} (\mathbf{u}_u^\top \mathbf{u}_v - R_{uv})^2 \\ &= \arg\min_U \|UU^\top - R\|_F^2 \end{aligned} \tag{3.9}$$

This leads to a straightforward matrix factorization algorithm to find the optimal embedding. Specifically, $U^* = Q_k \sqrt{\Lambda_k}$—where $R = Q\Lambda Q^\top$ is the Singular Value Decomposition (SVD) of R—is the optimal solution of U under the constraint $\text{rank}(UU^\top) = k$ [41].

Figure 3.5 (c) shows distributions of reconstructed similarity of different types of node

pairs for polarized embedding. It is clear that negative pairs can be effectively separated from unrelated and positive pairs. In addition, with negative pairs staying at the negative end of the similarity spectrum, positive pairs are also more separable. This demonstrates the strength of our polarized embedding that captures *polarized similarity consistency*.

3.5 Experiments

In this section, we demonstrate the effectiveness of POLE in signed link prediction using the six datasets in Table 3.3. Our code is available at `https://github.com/zexihuang/POLE`.

3.5.1 Experimental Settings

Baselines

We consider the following baselines:

- SiNE [29]: Combines an objective function based on social balance theory with a learned pairwise similarity function.
- SIGNet [106]: Attempts to capture higher-order balance structure by computing embeddings for which dot product approximates the signed proximity for links in the graph.
- SIDE [92]: Extends random-walk based embedding to signed graphs. Embeddings are learned based on maximum likelihood using pairwise proximities that are sensitive to link signs.
- BESIDE [107]: Applies balance and status theory to model signed triangles and "bridge" links, which are not included in a triangle.

- SLF [105]: Decomposes node embeddings into two types of latent factors (positive and negative). These factors, which are learned via coordinate descent, are applied to generate four types of scores corresponding to positive, negative, neutral, and no link.

- ROSE [31]: Transforms the signed network into an unsigned bipartite one by representing each node multiple times. Embeddings for the unsigned network are generated using a random-walk based approach and combined into signed embeddings.

Downstream task

We focus on signed link prediction, which consists of predicting of both link existence and signs. We randomly remove 20% of links while ensuring that the residual graph is connected and embed the residual graph. Then, for POLE, we rank all disconnected node pairs based on reconstructed dot product similarity and predict the most similar/dissimilar pairs as positive/negative links. For baselines, we follow [105] and train two logistic regression classifiers on concatenated node embeddings for positive/negative pairs vs disconnected pairs, respectively. While ranking by classifier scores is adopted by most baselines, we also conduct experiments with baselines using the same dot product similarity ranking as for POLE. We report *precision@k* [54] for positive/negative links respectively, where k is the number of top pairs in terms of the ratio of removed positive/negative links, ranging from 10% to 100%.

Since the baselines are designed to only capture *signed similarity consistency* and thus may perform poorly without link existence information, we also consider supplementing them with an unsigned embedding method. This also allows us to analyze the interaction between unsigned similarity and the signed similarity captured by our polarized embedding. Specifically, we apply RWE [1] with unsigned autocovariance and continuous random-walk and compute the reconstructed similarity based on embeddings for the

unsigned residual graph. We then train logistic regression classifiers to combine the unsigned similarity (encoding link existence information) and either reconstructed similarity (for POLE) or classifier scores (for baselines) for final ranking and report *precision@k* scores.

Parameters

We set the number of embedding dimensions k to 40 for all methods (except ROSE which requires a multiple of 3, we set it to 42). The only other parameter in RWE and POLE is the Markov time t, which is selected from $\{10^{0.0}, 10^{0.1}, \ldots, 10^{1.0}\}$ based on signed/unsigned link prediction. Other parameters for baselines are set as recommended in the original papers.

3.5.2 Results

Signed link prediction

Figure 3.6 shows the signed link prediction performance for different methods without supplementing the unsigned link existence information. We first note that the proposed method, POLE, outperforms all baselines in almost all datasets for both positive and negative links. The average gains in *precision@k* over the best baseline for each dataset are 629.2%/58.8%/115.7%/108.3%/142.5%/468.8% on positive links and 220.0%/27.6%/261.8%/1539.4%/4.3%/-10.9% on negative links. This shows that POLE enables state-of-the-art signed link prediction performance.

While the main motivation for POLE is predicting negative links, its superior performance in positive link prediction is also consistent with our previous similarity analysis (see Figure 3.5 (b) and Figure 3.5 (c)). The only scenario where POLE underperforms the best baseline is the negative link prediction on Wiki-RfA. This is actually not surprising

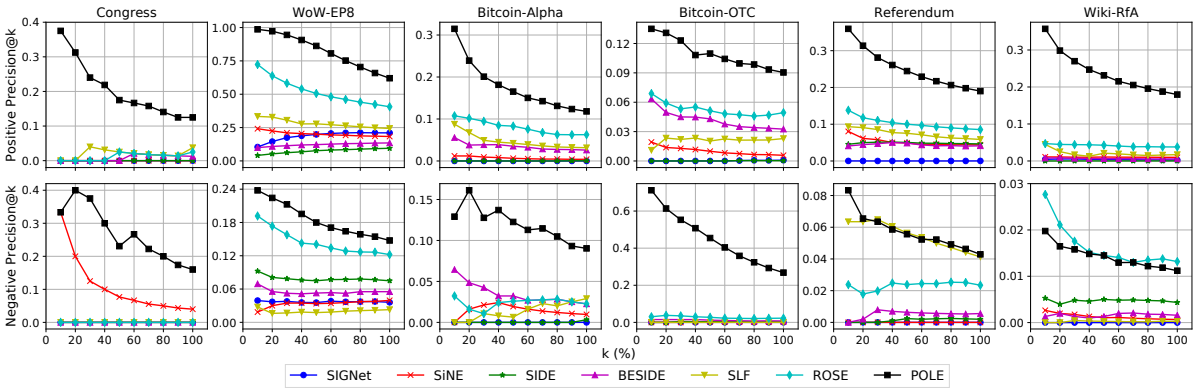


Figure 3.6: Comparison of performance of signed link prediction between POLE and baselines. POLE outperforms all baselines in all datasets on both positive and negative link prediction, except for negative links in Wiki-RfA, the least polarized network.

as Wiki-RfA is the least polarized network among the six, as shown in Figure 3.4. When the graph is less polarized (e.g., Figure 3.1 (b)), negative links are formed independently from community structure, and therefore the *polarized similarity consistency* may not be the best criterion to embed nodes.

We also conduct experiments with baselines using the same dot product similarity ranking as for POLE and the results are similar to those discussed above as shown in Figure 3.7.

Signed link prediction with link existence information

We now supplement each method with unsigned link existence information. Results are shown in Figure 3.8. Adding unsigned information narrows down the average performance gains of POLE over baselines on positive links to 18.0%/5.1%/4.4%/-6.6%/11.2%/3.5%. But our method still exhibits significant gains on negative links, outperforming the best baseline by 300.0%/93.9%/70.4%/243.8%/29.6%/79.7% for all datasets including the least polarized Wiki-RfA. Even with unsigned link existence information, our polarized embedding method has an edge over existing approaches, es-

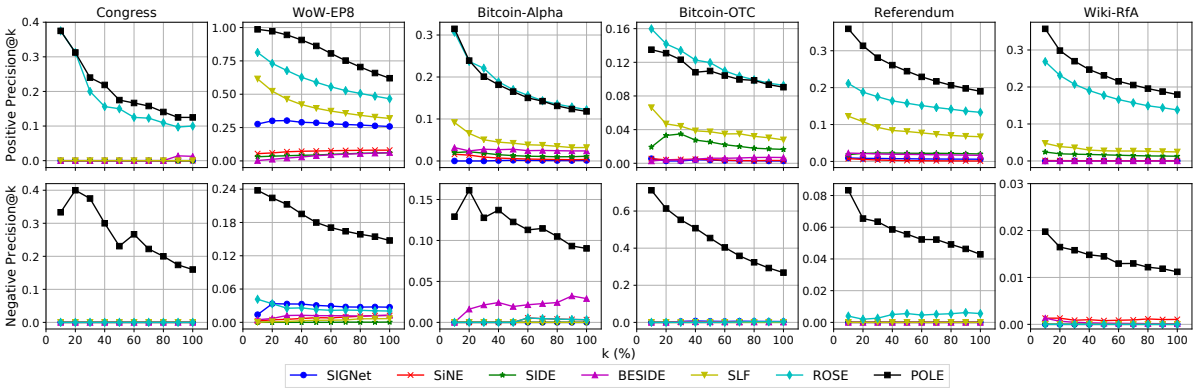


Figure 3.7: Comparison of performance of signed link prediction between POLE and baselines with reconstructed similarity ranking. POLE outperforms all baselines in almost all datasets on both positive and negative link prediction.

pecially for negative links.

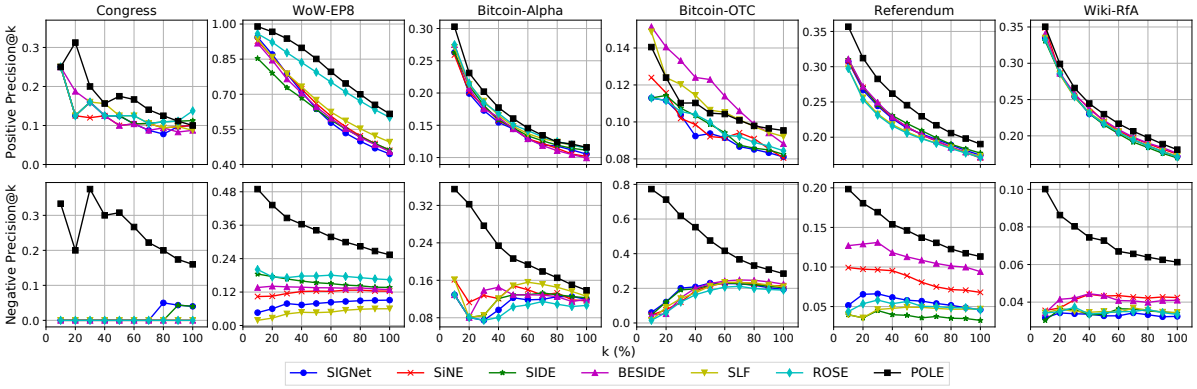


Figure 3.8: Comparison of performance of signed link prediction with link existence information between POLE and baselines. While adding unsigned similarity narrows the performance gap between POLE and baselines on positive link prediction, it significantly improves POLE on negative link prediction and keeps its edge.

The notable improvement of baselines after adding unsigned link existence information is consistent with our expectation. For positive links, combining the signed/unsigned embeddings enables the separation of positive pairs from negative/disconnected pairs, respectively, as illustrated in Figure 3.5 (a) and Figure 3.5 (b). This boosts baselines to a performance comparable with ours. For negative links, while all baselines still under-

perform POLE by a large margin, their predictions are improved significantly compared to the scenario without link existence information. The most prominent improvement is observed for BITCOIN-OTC, from negligible precision scores to notable ones of around 0.2 at $k = 100\%$. This is reasonable since none of the networks are fully polarized, and not all negative links in the networks are inter-community. Those that are intra-community can be correctly predicted by unsigned similarity first and then correct signs can be discovered using the baselines. The inter-community negative links, however, remain exclusively predictable via *polarized similarity consistency*, for which our polarized embedding method is designed.

We also conduct experiments with baselines using the same dot product similarity ranking as for POLE and supplemented with unsigned link existence information. The results are similar to those discussed above as shown in Figure 3.9.

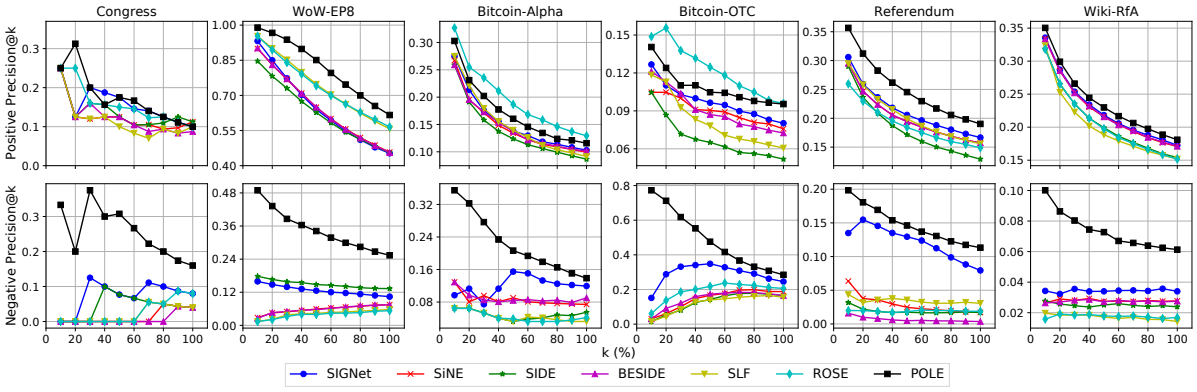


Figure 3.9: Comparison of performance of signed link prediction with link existence information between POLE and baselines with reconstructed similarity ranking. While adding unsigned similarity narrows the performance gap between POLE and baselines on positive link prediction, it significantly improves POLE on negative link prediction and keeps its edge.

At this point, it is important to analyze the interaction between unsigned similarity and our polarized embedding. While the performance for negative links more than doubles on average across the datasets, precision for positive links decreases by an average

of 2.9%. To understand this phenomenon, we draw scatter plots of the reconstructed signed and unsigned similarity for different node pairs in signed link prediction, along with the decision boundaries based on each similarity and a combination of both (via the classifier). The results for Referendum and Wiki-RfA are shown in Figure 3.10, with the rest in Figure 3.11. For positive pairs, signed and unsigned similarities are highly correlated, and signed similarity alone is predictive enough. This is consistent with the fact that regions above the signed and combined decision boundaries highly overlap. On the other hand, prediction for negative links is clearly improved by combining signed and unsigned similarities. The decision boundaries of the learned classifiers imply that negative pairs should have smaller signed similarity than unsigned similarity. This rule is especially useful as it filters out a large number of disconnected pairs with large negative signed similarity but even larger negative unsigned similarity. Finally, this plot is another illustration of how our method captures polarization. Referendum, the most polarized graph (see Figure 3.4) of all, shows a clearer separation between positive and negative pairs compared to Wiki-RfA, which is the least polarized.

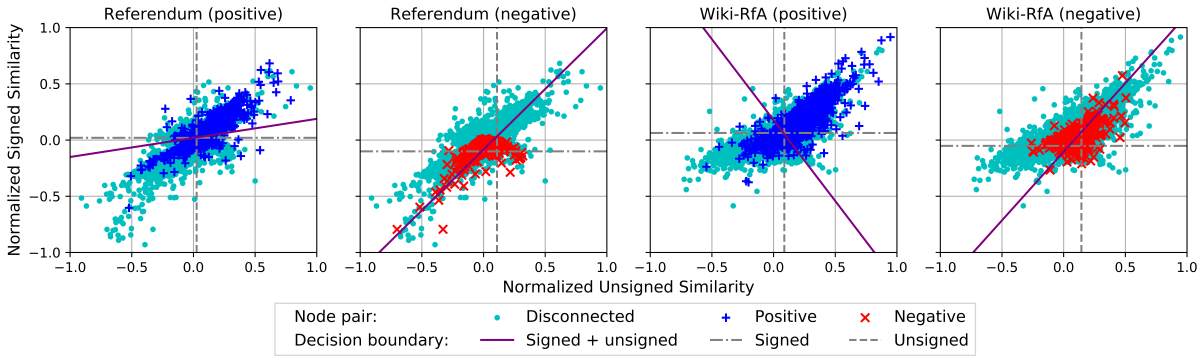


Figure 3.10: Scatter plot of the reconstructed signed and unsigned similarity for different node pairs in signed link prediction, along with the decision boundaries based on each similarity and a combination of both (via the classifier), for Referendum and Wiki-RfA. Combining signed and unsigned similarity improves prediction for negative links but has a negligible effect on predicting positive links.

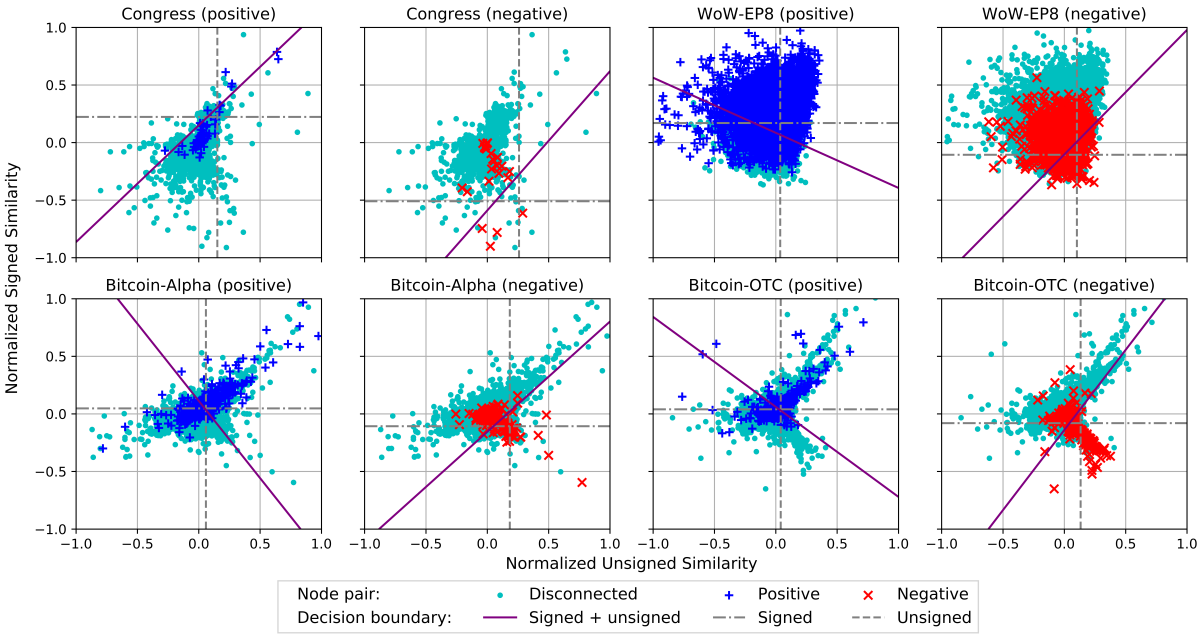


Figure 3.11: Scatter plot of the reconstructed signed and unsigned similarity for different node pairs in signed link prediction, along with the decision boundaries based on each similarity and a combination of both (via the classifier), for Congress, WoW-EP8, Bitcoin-Alpha and Bitcoin-OTC. Combining signed and unsigned similarity improves prediction for negative links but has a negligible effect on predicting positive links.

3.6 Related Work

Measuring polarization. Social polarization has received great attention with the increasing popularity in online social media [109]. In the context of networks/graphs, polarization is often defined as the cohesiveness of communities. Several community quality measures, such as conductance [110] and modularity [111], have been used for analyzing polarization in the U.S. congress [112] and online social media platforms [84]. Later works consider other measures based on the community structure. For example, [109] measures polarization based on the concentration of high-degree nodes in the community boundaries, and [113] extends it with betweenness and inter-community random-walk transitions. Likewise, polarization in signed graphs is often related to the objective

function of signed community detection. In [85], the authors define polarization as the size-normalized signed cut between two conflicting communities, which is later extended to multiple communities in [87]. The main distinction of our polarization measure compared with existing methods is that it leverages the signed random-walk to capture the community structure at multiple scales and does not require specific community assignments as input.

Signed link prediction. Link prediction is a common inference task for graphs [54, 114]. Link signs in signed graphs add another dimension to the problem, leading to the different tasks of (1) link sign prediction [115, 116, 117], (2) link existence prediction [118, 31], and (3) signed link prediction [89, 90, 91]. Signed link prediction is the most challenging one among them, requiring both existence and signs of links to be inferred. Existing signed link prediction methods are based on feature engineering [89], matrix factorization [90, 119], and graph embedding [105]. Due to sparsity of links, several methods also consider side information, including user-item ratings [104], user-user opinions [120, 104], and user-review ratings [121]. Our work focuses on signed link prediction in polarized graphs, which is an even harder problem as negative links mostly appear as sparse inter-community connections. Yet by incorporating polarization in the embedding, our method enables state-of-the-art signed link prediction without using additional information.

Signed graph embedding. Signed graph embedding enables the application of classical machine learning algorithms to graph-based downstream tasks, such as signed link prediction [89, 90], node classification [106], and polarized community detection [85, 87]. Existing methods are often based on extending unsigned embedding methods (such as those based on random-walks [8, 9]) and incorporating social theories (such as social balance theory [93] and social status theory [122]). Both SNE [123] and SIDE [92] extend random-walk based objective functions to signed embedding. SiNE [29] incorporates social balance theory and uses a learned pairwise similarity function modeled as a multi-

layer neural network. SIGNet [106] captures higher-order balance structure with efficient sampling algorithms. BESIDE [107] combines social balance theory and social status theory to model triangles and "bridge" links for embedding. SLF [105] models each node with four signed latent factors in order to capture positive, negative, neutral, and the absence of a relationship between them. ROSE [31] transforms the signed network into a bipartite unsigned network and leverages an existing random-walk based method [9] for embedding. While these methods capture the *signed similarity consistency* needed for sign prediction (see Figure 3.5), they are unable to predict negative links in polarized graphs even when combined with unsigned embedding [1]. By comparison, our polarized embedding preserves *polarized similarity consistency* and can effectively predict both positive and negative links in real-world signed graphs.

3.7 Conclusion

We have introduced several analytical tools for understanding and combating polarization in signed graphs. They include a partition-agnostic polarization measure for both nodes and graphs and an embedding algorithm that enables state-of-the-art signed link prediction, especially for the hostile links in polarized graphs. We hope that our work will be an important step towards making social media an environment for healthy and constructive communication among individuals with diverse viewpoints.

Chapter 4

Link Prediction without Graph Neural Networks

4.1 Introduction

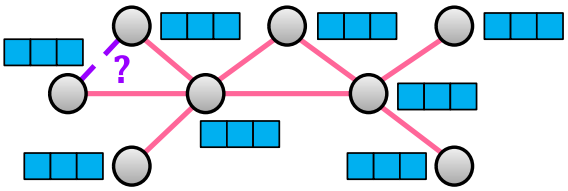
Machine learning on graphs supports various structured-data applications including social network analysis [124, 125, 126], recommender systems [127, 128, 129], natural language processing [130, 131, 132], and physics modeling [133, 134, 135]. Among the graph-related tasks, one could argue that link prediction [54, 114] is the most fundamental one. This is because link prediction not only has many concrete applications [136, 137, 138] but can also be considered an (implicit or explicit) step of the graph-based machine learning pipeline [139, 140, 141]—as the observed graph is usually noisy and/or incomplete.

In recent years, Graph Neural Networks (GNNs) [142, 143, 144] have emerged as the predominant paradigm for machine learning on graphs. Similar to their great success in node classification [145, 146, 147] and graph classification [148, 149, 150], GNNs have been shown to achieve state-of-the-art link prediction performance [151, 152, 153]. Compared

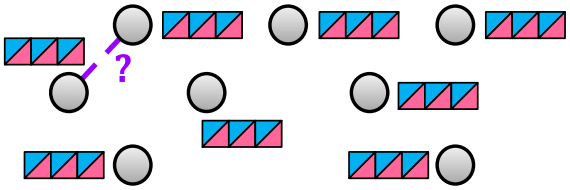
to classical approaches that rely on expert-designed heuristics to extract topological information (e.g., Common Neighbors [154], Adamic-Adar [155], Preferential Attachment [156]), GNNs have the potential to discover new heuristics via supervised learning and the natural advantage of incorporating node attributes.

However, there is little understanding of what factors contribute to the success of GNNs in link prediction, and whether simpler alternatives can achieve comparable performance, as recently found for node classification [157]. GNN-based methods approach link prediction as a binary classification problem. Yet different from other classification problems, link prediction deals with extremely class-imbalanced data due to the sparsity of real-world graphs. We argue that class imbalance should be accounted for in both training and evaluation of link prediction. In addition, GNNs combine topological and attribute information by learning topology-smoothened attributes (embeddings) via message-passing [158]. This attribute-centric mechanism has been proven effective for tasks *on* the topology such as node classification [159], but link prediction is a task *for* the topology, which naturally motivates topology-centric paradigms (see Figure 4.1).

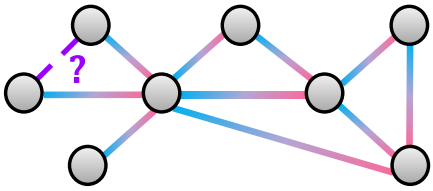
The goal of this work is to address the key issues raised above. We first show that the evaluation of GNN-based link prediction pictures an overly optimistic view of model performance compared to the (more realistic) imbalanced setting. Class imbalance also prevents the generalization of these models due to bias in their training. Instead, we propose the use of the N-pair loss with an unbiased set of training edges to account for class imbalance. Moreover, we present *Gelato*, a novel framework that combines topological and attribute information for link prediction. As a simpler alternative to GNNs, our model applies topology-centric graph learning to incorporate node attributes directly into the graph structure, which is given as input to a topological heuristic, Autocovariance, for link prediction. Extensive experiments demonstrate that our model significantly outperforms state-of-the-art GNN-based methods in both accuracy and scalability.



(a) Link prediction for attributed graphs



(b) GNN: topology → attributes



(c) Gelato: attributes → topology

Figure 4.1: GNN incorporates topology into attributes via message-passing, which is effective for tasks ***on*** the topology. Link prediction, however, is a task ***for*** the topology, which motivates the design of Gelato—a novel framework that leverages graph learning to incorporate attributes into topology.

To summarize, our contributions are:

- We scrutinize the training and evaluation of supervised link prediction methods and identify their limitations in handling class imbalance.
- We propose a simple, effective, and efficient framework to combine topological and attribute information for link prediction without using GNNs.
- We introduce an N-pair link prediction loss combined with an unbiased set of training edges that we show to be more effective at addressing class imbalance.

4.2 Limitations in Supervised Link Prediction Evaluation and Training

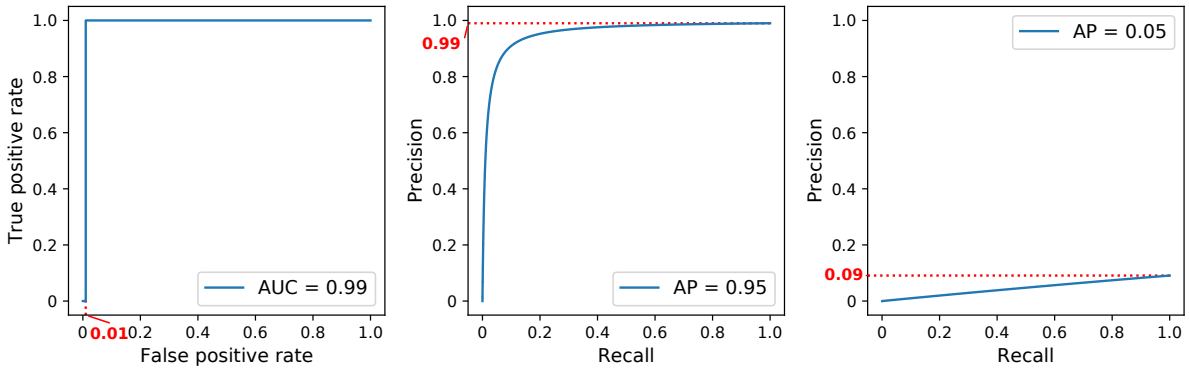
Supervised link prediction is often formulated as a binary classification problem, where the positive (or negative) class includes node pairs connected (or not connected)

by a link. A key difference between link prediction and typical classification problems (e.g., node classification) is that the two classes in link prediction are *extremely* imbalanced, since most real-world graphs of interest are sparse (see Table 4.1). However, we find that class imbalance is not properly addressed in both evaluation and training of existing supervised link prediction approaches, as discussed below.

Link prediction evaluation. Area Under the Receiver Operating Characteristic Curve (AUC) and Average Precision (AP) are the two most popular evaluation metrics for supervised link prediction [160, 151, 161, 162, 163, 164, 165, 166, 153]. We first argue that, as in other imbalanced classification problems [167, 168], AUC is not an effective evaluation metric for link prediction as it is biased towards the majority class (non-edges). On the other hand, AP and other rank-based metrics such as Hits@k—used in Open Graph Benchmark (OGB) [169]—are effective for imbalanced classification *if evaluated on a test set that follows the original class distribution*. Yet, existing link prediction methods [160, 151, 163, 165, 153] compute AP on a test set that contains all positive test pairs and only an equal number of random negative pairs. Similarly, OGB computes Hits@k against a very small subset of random negative pairs. We term these approaches *biased testing* as they highly overestimate the ratio of positive pairs in the graph. Evaluation metrics based on these biased test sets provide an overly optimistic measurement of the actual performance in *unbiased testing*, where every negative pair is included in the test set. In fact, in real applications where test positive edges are not known a priori, it is impossible to construct those biased test sets to begin with. Below, we also present an illustrative example of the misleading performance evaluation based on *biased testing*.

Example: Consider a graph with 10k nodes, 100k edges, and 99.9M disconnected (or negative) pairs. A (bad) model that ranks 1M false positives higher than the true edges achieves 0.99 AUC and 0.95 in AP under *biased testing* with equal negative samples.

Specifically, Figure 4.2a and Figure 4.2b show the receiver operating characteristic (ROC) and precision-recall (PR) curves for the model under *biased testing* with equal negative samples. Due to the downsampling, only 100k (out of 99.9M) negative pairs are included in the test set, among which only 100k/99.9M $\times$ 1M $\approx$ 1k pairs are ranked higher than the positive edges. In the ROC curve, this means that once the false positive rate reaches 1k/100k = 0.01, the true positive rate would reach 1.0, leading to an AUC score of 0.99. Similarly, in the PR curve, when the recall reaches 1.0, the precision is 100k/(1k + 100k) $\approx$ 0.99, leading to an overall AP score of ~0.95.



(a) ROC (b) PR under *biased testing* (c) PR under *unbiased testing*

Figure 4.2: Receiver operating characteristic and precision-recall curves for the bad link prediction model that ranks 1M false positives higher than the 100k true edges. The model achieves 0.99 in AUC and 0.95 AP under *biased testing*, while the more informative performance evaluation metric, Average Precision (AP) under *unbiased testing*, is only 0.05.

The above discussion motivates a more representative evaluation setting for supervised link prediction. Specifically, we argue for the use of rank-based evaluation metrics—AP, Precision@k [54], and Hits@k [170]—with *unbiased testing*, where positive edges are ranked against all negative pairs. These metrics have been widely applied in related problems, such as unsupervised link prediction [54, 10, 33, 1], knowledge graph completion [170, 171, 172], and information retrieval [173], where class imbalance is also significant. For the same example link prediction model mentioned above, with *unbiased testing*,

when the recall reaches 1.0, the precision under is only $100\text{k}/(1\text{M} + 100\text{k}) \approx 0.09$, as shown in Figure 4.2c, leading to an AP score of ~0.05, a more realistic evaluation of the model performance. Further, in our experiments, we will illustrate how these evaluation metrics combined with *unbiased testing* provide a drastically different and more informative performance evaluation in real-world datasets compared to existing approaches.

Link prediction training. Following the formulation of supervised link prediction as binary classification, most existing models adopt the binary cross entropy loss to optimize their parameters [160, 151, 161, 162, 164, 152, 165, 166]. To deal with class imbalance, these approaches downsample the negative pairs to match the number of positive pairs in the training set (*biased training*). We highlight two drawbacks of *biased training*: (1) it induces the model to overestimate the probability of positive pairs, and (2) it discards potentially useful evidence from most negative pairs. Notice that the first drawback is often hidden by *biased testing*. Instead, this work proposes the use of *unbiased training*, where the ratio of negative pairs in the training set is the same as in the input graph. To train our model in this highly imbalanced setting, we apply the N-pair loss for link prediction instead of the cross entropy loss (Section 4.3.3).

4.3 Method

Notation and problem. Consider an attributed graph $G = (V, E, X)$, where V is the set of n nodes, E is the set of m edges (links), and $X = (x_1, ..., x_n)^\top \in \mathbb{R}^{n \times r}$ collects r-dimensional node attributes. The topological (structural) information of the graph is represented by its adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{uv} > 0$ if an edge of weight A_{uv} connects nodes u and v and $A_{uv} = 0$, otherwise. The (weighted) degree of node u is given as $d_u = \sum_v A_{uv}$ and the corresponding degree vector (matrix) is denoted as $d \in \mathbb{R}^n$ ($D \in \mathbb{R}^{n \times n}$). The volume of the graph is $\text{vol}(G) = \sum_u d_u$. Our goal is to infer missing

links in G based on its topological and attribute information, A and X.

Model overview. Figure 4.3 provides an overview of our link prediction model. It starts with a topology-centric graph learning phase that incorporates node attribute information directly into the graph structure via a Multi-layer Perceptron (MLP). We then apply a topological heuristic, Autocovariance (AC), to the attribute-enhanced graph to obtain a pairwise score matrix. Node pairs with the highest scores are predicted as (positive) links. The scores for training pairs are collected to compute an N-pair loss. Finally, the loss is used to train the MLP parameters in an end-to-end manner. We named our model Gelato (Graph enhancement for link prediction with autocovariance). Gelato represents a paradigm shift in supervised link prediction by combining a graph encoding of attributes with a topological heuristic instead of relying on increasingly popular GNN-based embeddings.

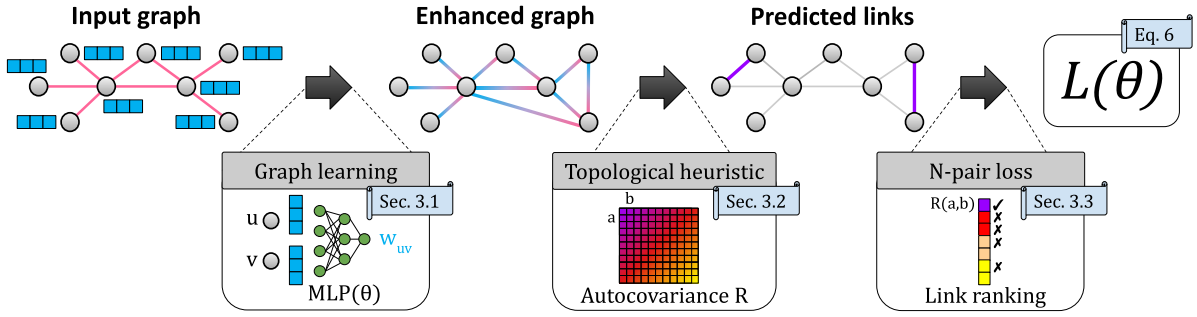


Figure 4.3: Gelato applies graph learning to incorporate attribute information into the topology via an MLP. The learned graph is given to a topological heuristic that predicts edges between node pairs with high Autocovariance similarity. The parameters of the MLP are optimized end-to-end using the N-pair loss. Experiments show that Gelato outperforms state-of-the-art GNN-based link prediction methods.

4.3.1 Graph Learning

The goal of graph learning is to generate an enhanced graph that incorporates node attribute information into the topology. This can be considered as the "dual" operation

of message-passing in GNNs, which incorporates topological information into attributes (embeddings). We argue that graph learning is the more suitable scheme to combine attributes and topology for link prediction, since link prediction is a task for the topology itself (as opposed to other applications such as node classification).

Specifically, our first step of graph learning is to augment the original edges with a set of node pairs based on their (untrained) attribute similarity (i.e., adding an ϵ-neighborhood graph):

$$\widetilde{E} = E + \{(u, v) \mid s(x_u, x_v) > \epsilon_\eta\} \tag{4.1}$$

where $s(\cdot)$ can be any similarity function (we use cosine in our experiments) and ϵ_η is a threshold that determines the number of added pairs as a ratio η of the original number of edges m.

A simple MLP then maps the pairwise node attributes into a trained edge weight for every edge in $\widetilde{E}$:

$$w_{uv} = \text{MLP}([x_u; x_v]; \theta) \tag{4.2}$$

where $[x_u; x_v]$ denotes the concatenation of x_u and x_v and θ contains the trainable parameters. For undirected graphs, we instead use the following permutation invariant operator [174]:

$$w_{uv} = \text{MLP}([x_u + x_v; |x_u - x_v|]; \theta) \tag{4.3}$$

The final edge weights of the enhanced graph are a weighted combination of the topological weights, the untrained weights, and the trained weights:

$$\widetilde{A}_{uv} = \alpha A_{uv} + (1 - \alpha)(\beta w_{uv} + (1 - \beta)s(x_u, x_v)) \tag{4.4}$$

where α and β are hyperparameters. The enhanced adjacency matrix $\widetilde{A}$ is then fed into a topological heuristic for link prediction introduced in the next section. Note that

the MLP is not trained directly to predict the links, but instead trained end-to-end to enhance the input graph given to the topological heuristic. Also note that the MLP can be easily replaced by a more powerful model such as a GNN, but the goal of this work is to demonstrate the general effectiveness of our framework and we will show that even a simple MLP leads to significant improvement over the base heuristic.

4.3.2 Topological Heuristic

Assuming that the learned adjacency matrix $\widetilde{A}$ incorporates structural and attribute information, Gelato applies a topological heuristic to $\widetilde{A}$. Specifically, we adopt Autocovariance, which has been shown to achieve state-of-the-art link prediction results for non-attributed graphs [1].

Autocovariance is a random-walk based similarity metric. Intuitively, it measures the difference between the co-visiting probabilities for a pair of nodes in a truncated walk and in an infinitely long walk. Given the enhanced graph $\widetilde{G}$, the Autocovariance similarity matrix $R \in \mathbb{R}^{n \times n}$ is given as

$$R = \frac{\widetilde{D}}{\mathrm{vol}(\widetilde{G})}(\widetilde{D}^{-1}\widetilde{A})^t - \frac{\tilde{d}\tilde{d}^\top}{\mathrm{vol}^2(\widetilde{G})} \tag{4.5}$$

where $t \in \mathbb{N}_0$ is the scaling parameter of the truncated walk. Each entry R_{uv} represents a similarity score for node pair (u, v) and top similarity pairs are predicted as links. Note that R_{uv} only depends on the t-hop enclosing subgraph of (u, v) and can be easily differentiated with respect to the edge weights in the subgraph. In fact, Gelato could be applied with any differentiable topological heuristic or even a combination of them. In our experiments (Section 4.4.2), we will show that Autocovariance alone enables state-of-the-art link prediction performance.

Next, we introduce how to train our model parameters with supervised information.

4.3.3 N-pair Loss and Unbiased Training

As we have mentioned in Section 4.2, current supervised link prediction methods rely on *biased training* and the cross entropy loss (CE) to optimize model parameters. Instead, Gelato applies the N-pair loss [175] that is inspired by the metric learning and learning-to-rank literature [176, 177, 178, 179] to train the parameters of our graph learning model (see Section 4.3.1) from highly imbalanced *unbiased training* data.

The N-pair loss (NP) contrasts each positive training edge (u, v) against a set of negative pairs $N(u, v)$. It is computed as follows:

$$L(\theta) = -\sum_{(u,v)\in E} \log \frac{\exp(R_{uv})}{\exp(R_{uv}) + \sum_{(p,q)\in N(u,v)} \exp(R_{pq})} \tag{4.6}$$

Intuitively, $L(\theta)$ is minimized when each positive edge (u, v) has a much higher similarity than its contrasted negative pairs: $R_{uv} \gg R_{pq}, \forall (p, q) \in N(u, v)$. Compared to CE, NP is more sensitive to negative pairs that have comparable similarities to those of positive pairs—they are more likely to be false positives. While NP achieves good performance in our experiments, alternative losses from the learning-to-rank literature [180, 181, 182] could also be applied.

Gelato generates negative samples $N(u, v)$ using *unbiased training*. This means that $N(u, v)$ is a random subset of all disconnected pairs in the training graph, and $|N(u, v)|$ is proportional to the ratio of negative pairs over positive ones. In this way, we leverage more information contained in negative pairs compared to *biased training*. Note that, similar to *unbiased training*, (unsupervised) topological heuristics implicitly use information from all edges and non-edges. Also, *unbiased training* can be combined with adversarial negative sampling methods [183, 184] from the knowledge graph embedding literature to increase the quality of contrasted negative pairs.

Complexity analysis. The only trainable component in our model is the graph learning MLP with $O(rh+lh^2)$ parameters—where r is the number of node features, l is the number of hidden layers, and h is the number of neurons per layer. Notice that the number of parameters is independent of the graph size. Constructing the ϵ-neighborhood graph based on cosine similarity can be done efficiently using hashing and pruning [185, 186]. Computing the enhanced adjacency matrix with the MLP takes $O((1+\eta)mr)$ time per epoch—where $m = |E|$ and η is the ratio of edges added to E from the ϵ-neighborhood graph. We apply sparse matrix multiplication to compute entries of the t-step AC in $O((1+\eta)mt)$ time. Note that unlike recent GNN-based approaches [151, 187, 153] that generate distinctive subgraphs for each link (e.g., via the labeling trick), enclosing subgraphs for links in Gelato share the same information (i.e., learned edge weights), which significantly reduces the computational cost. Our experiments will demonstrate Gelato's efficiency in training and inference.

4.4 Experiments

We provide empirical evidence for our claims regarding supervised link prediction and demonstrate the accuracy and efficiency of Gelato. Our implementation is available at `https://anonymous.4open.science/r/Gelato/`.

4.4.1 Experiment Settings

Datasets. Our method is evaluated on five attributed graphs commonly used as link prediction benchmark [161, 162, 164, 165, 166, 153], with statistics summarized in Table 4.1:

- Cora [188] and CiteSeer [189] are citation networks where nodes represent sci-

entific publications (classified into seven and six classes, respectively) and edges represent the citations between them. Attributes of each node is a binary word vector indicating the absence/presence of the corresponding word from a dictionary.

- PubMed [190] is a citation network where nodes represent scientific publications (classified into three classes) and edges represent the citations between them. Attributes of each node is a TF/IDF weighted word vector.

- Photo and Computers are subgraphs of the Amazon co-purchase graph [191] where nodes represent products (classified into eight and ten classes, respectively) and edges imply that two products are frequently bought together. Attributes of each node is a bag-of-word vector encoding the product review.

	#Nodes	#Edges	#Attrs	Avg. degree	Density
Cora	2,708	5,278	1,433	3.90	0.14%
CiteSeer	3,327	4,552	3,703	2.74	0.08%
PubMed	19,717	44,324	500	4.50	0.02%
Photo	7,650	119,081	745	31.13	0.41%
Computers	13,752	245,861	767	35.76	0.26%

Table 4.1: A summary of dataset statistics.

We use the publicly available version of the datasets from the `pytorch-geometric` library [192] (under the MIT licence) curated by [193] and [194].

Baselines. For GNN-based link prediction, we include six state-of-the-art methods published in the past two years: LGCN [162], TLC-GNN [164], Neo-GNN [152], NBFNet [165], BScNets [166], and WalkPool [153], as well as three pioneering works—GAE [160], SEAL [151], and HGCN [161]. For topological link prediction heuristics, we consider Common Neighbors (CN) [154], Adamic Adar (AA) [155], Resource Allocation (RA) [195], and Autocovariance (AC) [1]—the base heuristic in our model. To demonstrate

the superiority of the proposed end-to-end model, we also include an MLP trained directly for link prediction, the cosine similarity (Cos) between node attributes, and AC on top of the respective weighted/augmented graphs (i.e., two-stage approaches where the MLP is trained separately for link prediction rather than trained end-to-end) as baselines.

Hyperparameters. For Gelato, we tune the proportion of added edges η from {0.0, 0.25, 0.5, 0.75, 1.0}, the topological weight α from {0.0, 0.25, 0.5, 0.75}, and the trained weight β from {0.25, 0.5, 0.75, 1.0}, with a sensitivity analysis included in Section 4.4.6. All other settings are fixed across datasets: MLP with one hidden layer of 128 neurons, AC scaling parameter $t = 3$, Adam optimizer [57] with a learning rate of 0.001, a dropout rate of 0.5, and *unbiased training* without downsampling. For baselines, we use the same hyperparameters as in their papers.

Data splits for unbiased training and unbiased testing. Following [160, 151, 161, 162, 166, 153], we adopt 85%/5%/10% ratios for training, validation, and testing. Specifically, for *unbiased training* and *testing*, we first randomly (with seed 0) divide the (positive) edges E of the original graph into E^+_{train}, E^+_{valid}, and E^+_{test} for training, validation, and testing based on the selected ratios. Then, we set the negative pairs in these three sets as (1) $E^-_{train} = E^- + E^+_{valid} + E^+_{test}$, (2) $E^-_{valid} = E^- + E^+_{test}$, and (3) $E^-_{test} = E^-$, where E^- is the set of all negative pairs (excluding self-loops) in the original graph. Notice that the validation and testing *positive* edges are included in the *negative* training set, and the testing *positive* edges are included in the *negative* validation set. These inclusions simulate the real-world scenario where the testing edges (and the validation edges) are unobserved during validation (training).

Evaluation metrics. We adopt Precision@k ($prec@k$)—proportion of positive edges among the top k of all testing pairs, Hits@k ($hits@k$)—ratio of positive edges individually ranked above kth place against all negative pairs, and Average Precision (AP)—area under the precision-recall curve, as evaluation metrics. We report results from 10 runs

with random seeds ranging from 1 to 10.

Following are the more detailed experiment settings:

Positive masking. For *unbiased training*, a trick similar to *negative injection* [151] in *biased training* is needed to guarantee model generalizability. Specifically, we divide the training positive edges into batches and during the training with each batch E_b, we feed in only the residual edges $E - E_b$ as the structural information to the model. This setting simulates the testing phase, where the model is expected to predict edges without using their own connectivity information. We term this trick *positive masking*.

Other implementation details. We add self-loops to the enhanced adjacency matrix to ensure that each node has a valid transition probability distribution that is used in computing Autocovariance. The self-loops are added to all isolated nodes in the training graph for PubMed, Photo, and Computers, and to all nodes for Cora and CiteSeer. Following the postprocessing of the Autocovariance matrix for embedding in [1], we standardize Gelato similarity scores before computing the loss. For training with the cross entropy loss, we further add a linear layer with the sigmoid activation function to map our prediction score to a probability. We optimize our model with gradient descent via `autograd` in `pytorch` [196]. We find that the gradients are sometimes invalid when training our model (especially with the cross entropy loss), and we address this by skipping the parameter updates for batches leading to invalid gradients. Finally, we use *prec*@100% on the (unbiased) validation set as the criteria for selecting the best model from all training epochs. The maximum number of epochs for Cora/CiteSeer/PubMed and Photo/Computers are set to be 100 and 250, respectively.

Experiment environment. We run our experiments in an *a2-highgpu-1g* node of the Google Cloud Compute Engine. It has one NVIDIA A100 GPU with 40GB HBM2 GPU memory and 12 Intel Xeon Scalable Processor (Cascade Lake) 2nd Generation vCPUs

with 85GB memory.

Reference of baselines. We list link prediction baselines and their reference repositories we use in our experiments in Table 4.2. Note that we had to implement the batched training and testing for several baselines as their original implementations do not scale to *unbiased training* and *unbiased testing* without downsampling.

Baseline	Repository
GAE [142]	`https://github.com/zfjsail/gae-pytorch`
SEAL [151]	`https://github.com/facebookresearch/SEAL_OGB`
HGCN [161]	`https://github.com/HazyResearch/hgcn`
LGCN [162]	`https://github.com/ydzhang-stormstout/LGCN/`
TLC-GNN [164]	`https://github.com/pkuyzy/TLC-GNN/`
Neo-GNN [152]	`https://github.com/seongjunyun/Neo-GNNs`
NBFNet [165]	`https://github.com/DeepGraphLearning/NBFNet`
BScNets [166]	`https://github.com/BScNets/BScNets`
WalkPool [153]	`https://github.com/DaDaCheng/WalkPooling`
AC [1]	`https://github.com/zexihuang/random-walk-embedding`

Table 4.2: Reference of baseline code repositories.

Number of trainable parameters. The only trainable component in Gelato is the graph learning MLP, which for `Photo` has 208,130 parameters. By comparison, the best performing GNN-based method, Neo-GNN, has more than twice the number of parameters (455,200).

4.4.2 Link Prediction Performance

Table 4.3 summarizes the link prediction performance in terms of the mean and standard deviation of Average Precision (AP) for all methods. Figure 4.4 and Figure 4.5 show results based on $prec@k$ (k as a ratio of test edges) and $hits@k$ (k as the rank) for varying k.

First, we want to highlight the drastically different performance of GNN-based meth-

		Cora	CiteSeer	PubMed	Photo	Computers
GNN	GAE	0.27 ± 0.02	0.66 ± 0.11	0.26 ± 0.03	0.28 ± 0.02	0.30 ± 0.02
	SEAL	1.89 ± 0.74	0.91 ± 0.66	***	10.49 ± 0.86	6.84*
	HGCN	0.82 ± 0.03	0.74 ± 0.10	0.35 ± 0.01	2.11 ± 0.10	2.30 ± 0.14
	LGCN	1.14 ± 0.04	0.86 ± 0.09	0.44 ± 0.01	3.53 ± 0.05	1.96 ± 0.03
	TLC-GNN	0.29 ± 0.09	0.35 ± 0.18	OOM	1.77 ± 0.11	OOM
	Neo-GNN	2.05 ± 0.61	1.61 ± 0.36	1.21 ± 0.14	10.83 ± 1.53	6.75*
	NBFNet	1.36 ± 0.17	0.77 ± 0.22	***	11.99 ± 1.60	***
	BScNets	0.32 ± 0.08	0.20 ± 0.06	0.22 ± 0.08	2.47 ± 0.18	1.45 ± 0.10
	WalkPool	2.04 ± 0.07	1.39 ± 0.11	1.31*	OOM	OOM
Topological Heuristics	CN	1.10 ± 0.00	0.74 ± 0.00	0.36 ± 0.00	7.73 ± 0.00	5.09 ± 0.00
	AA	2.07 ± 0.00	1.24 ± 0.00	0.45 ± 0.00	9.67 ± 0.00	6.52 ± 0.00
	RA	2.02 ± 0.00	1.19 ± 0.00	0.33 ± 0.00	10.77 ± 0.00	7.71 ± 0.00
	AC	2.43 ± 0.00	2.65 ± 0.00	2.50 ± 0.00	16.63 ± 0.00	11.64 ± 0.00
Attributes + Topology	MLP	0.30 ± 0.05	0.44 ± 0.09	0.14 ± 0.06	1.01 ± 0.26	0.41 ± 0.23
	Cos	0.42 ± 0.00	1.89 ± 0.00	0.07 ± 0.00	0.11 ± 0.00	0.07 ± 0.00
	MLP+AC	3.24 ± 0.03	1.95 ± 0.05	2.61 ± 0.06	15.99 ± 0.21	11.25 ± 0.13
	Cos+AC	3.60 ± 0.00	4.46 ± 0.00	0.51 ± 0.00	10.01 ± 0.00	5.20 ± 0.00
	MLP+Cos+AC	3.39 ± 0.06	4.15 ± 0.14	0.55 ± 0.03	10.88 ± 0.09	5.75 ± 0.11
Gelato		**3.90 ± 0.03**	**4.55 ± 0.02**	**2.88 ± 0.09**	**25.68 ± 0.53**	**18.77 ± 0.19**

Table 4.3: Link prediction performance comparison (mean ± std AP). Gelato consistently outperforms GNN-based methods, topological heuristics, and two-stage approaches combining attributes and topology. (* Run only once as each run takes ~100 hrs; *** Each run takes >1000 hrs; OOM: Out Of Memory.)

ods compared to those found in the original papers [162, 164, 152, 165, 166, 153] with AUC performance reproduced in Table 4.4. While they achieve AUC/AP scores of often higher than 90% under *biased testing*, here we see most of them underperform even the simplest topological heuristics such as Common Neighbors under *unbiased testing*. These results support our arguments from Section 4.2 that evaluation metrics based on *biased testing* can produce misleading results compared to *unbiased testing*. The overall best

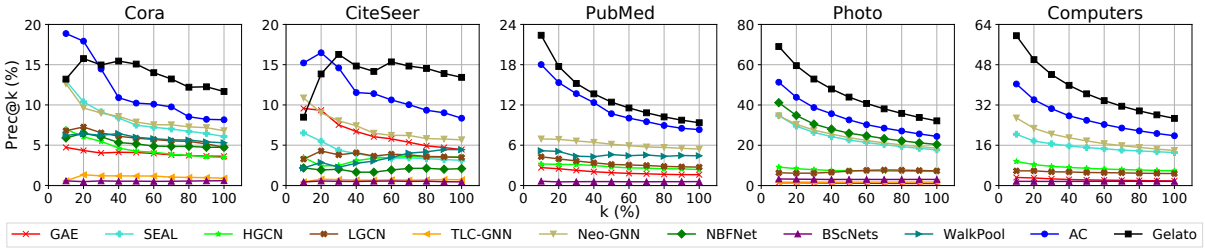


Figure 4.4: Link prediction performance in terms of *prec@k* for varying values of k (as percentages of test edges). With few exceptions, Gelato outperforms the baselines across different values of k.

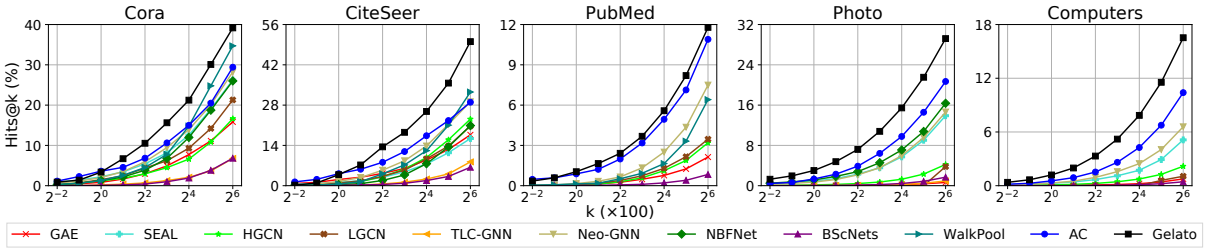


Figure 4.5: Link prediction performance in terms of *hits@k* for varying values of k. With few exceptions, Gelato outperforms the baselines across different values of k.

performing GNN model is Neo-GNN, which directly generalizes the pairwise topological heuristics. Yet still, it consistently underperforms AC, a random-walk based heuristic that needs neither node attributes nor supervision/training.

		Cora	CiteSeer	PubMed	Photo	Computers
GNN	GAE	87.30 ± 0.22	87.48 ± 0.39	94.10 ± 0.22	77.59 ± 0.73	79.36 ± 0.37
	SEAL	91.82 ± 1.08	90.37 ± 0.91	***	98.85 ± 0.04	98.7*
	HGCN	92.60 ± 0.29	92.39 ± 0.61	94.40 ± 0.14	96.08 ± 0.08	97.86 ± 0.10
	LGCN	91.60 ± 0.23	93.07 ± 0.77	95.80 ± 0.03	98.36 ± 0.01	97.81 ± 0.01
	TLC-GNN	91.57 ± 0.95	91.18 ± 0.78	OOM	98.20 ± 0.08	OOM
	Neo-GNN	91.77 ± 0.84	90.25 ± 0.80	90.43 ± 1.37	98.74 ± 0.55	98.34*
	NBFNet	86.06 ± 0.59	85.10 ± 0.32	***	98.29 ± 0.35	***
	BScNets	91.59 ± 0.47	89.62 ± 1.05	97.48 ± 0.07	98.68 ± 0.06	98.41 ± 0.05
	WalkPool	92.33 ± 0.30	90.73 ± 0.42	98.66*	OOM	OOM
Topological Heuristics	CN	71.15 ± 0.00	66.91 ± 0.00	64.09 ± 0.00	96.73 ± 0.00	96.15 ± 0.00
	AA	71.22 ± 0.00	66.92 ± 0.00	64.09 ± 0.00	97.02 ± 0.00	96.58 ± 0.00
	RA	71.22 ± 0.00	66.93 ± 0.00	64.09 ± 0.00	97.20 ± 0.00	96.82 ± 0.00
	AC	75.41 ± 0.00	72.41 ± 0.00	67.46 ± 0.00	98.24 ± 0.00	96.86 ± 0.00
Attributes + Topology	MLP	85.43 ± 0.36	87.89 ± 2.05	87.89 ± 2.05	91.24 ± 1.61	88.84 ± 2.58
	Cos	79.06 ± 0.00	89.86 ± 0.00	89.14 ± 0.00	71.46 ± 0.00	71.68 ± 0.00
	MLP+AC	79.77 ± 0.03	82.26 ± 0.29	66.31 ± 0.74	98.02 ± 0.05	96.69 ± 0.05
	Cos+AC	86.34 ± 0.00	89.29 ± 0.00	75.56 ± 0.00	97.28 ± 0.00	96.26 ± 0.00
	MLP+Cos+AC	86.90 ± 0.14	89.42 ± 0.09	75.78 ± 0.27	97.27 ± 0.01	96.24 ± 0.01
Gelato		83.12 ± 0.06	88.38 ± 0.02	64.93 ± 0.06	98.01 ± 0.03	96.72 ± 0.04

Table 4.4: Link prediction performance comparison (mean ± std AUC). AUC results conflict with other evaluation metrics, presenting a misleading view of the model performance for link prediction. (* Run only once as each run takes ~100 hrs; *** Each run takes >1000 hrs; OOM: Out Of Memory.)

We then look at two-stage combinations of AC and models for attribute information. We observe noticeable performance gains from combining attribute cosine similarity and AC in Cora and CiteSeer but not for other datasets. Other two-stage approaches

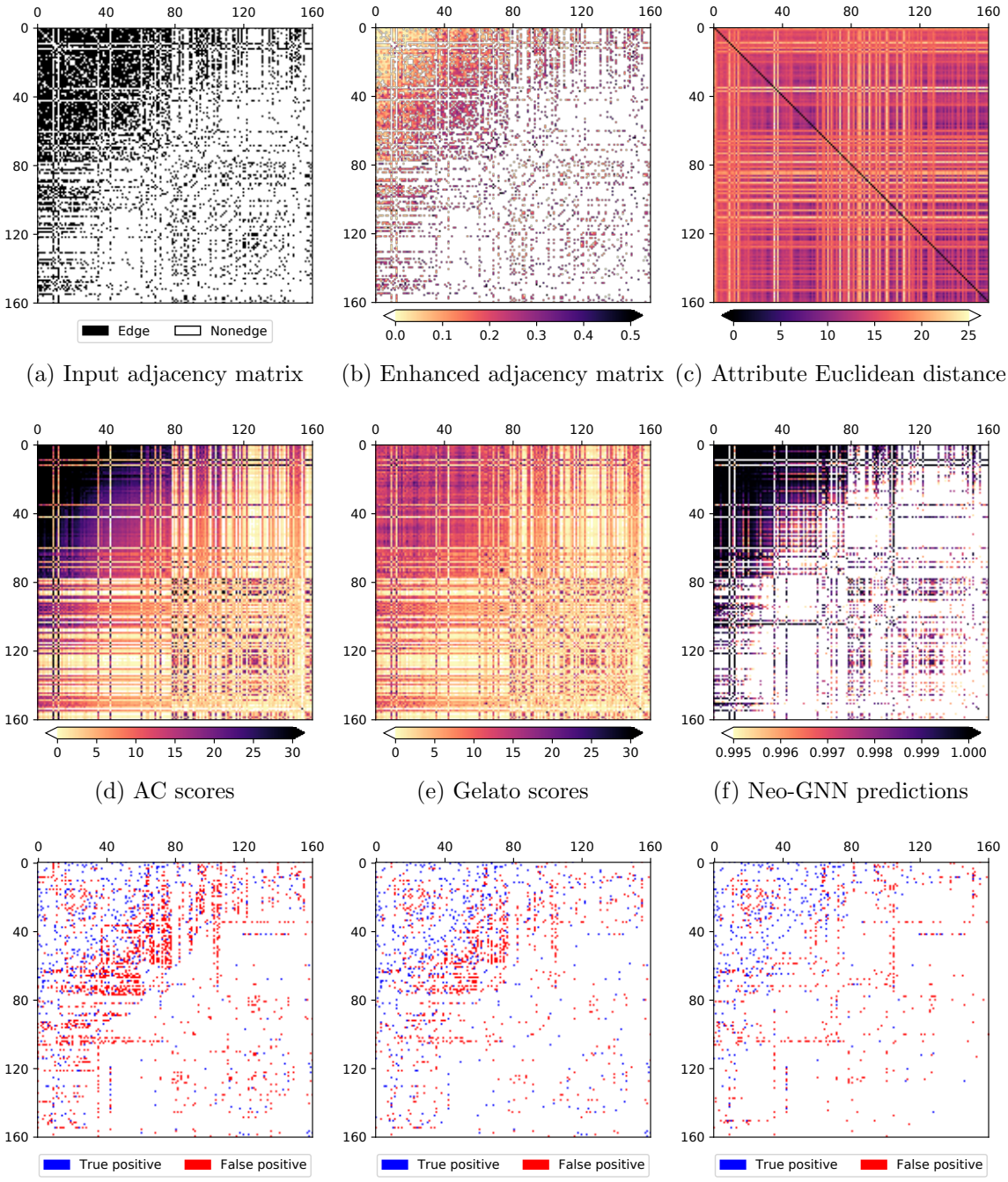
achieve similar or worse performance.

Finally, Gelato significantly outperforms the best GNN-based model with an average relative gain of **145.2%** and AC with a gain of **52.6%** in terms of AP—similar results were obtained for $prec@k$ and $hits@k$. This validates our hypothesis that a simple MLP can effectively incorporate node attribute information into the topology when our model is trained end-to-end. Next, we will provide insights behind these improvements and demonstrate the efficiency of Gelato on training and inference.

4.4.3 Visualizing Gelato Predictions

To better understand the performance of Gelato, we visualize its learned graph, prediction scores, and predicted edges in comparison with AC and the best GNN-based baseline (Neo-GNN) in Figure 4.6.

Figure 4.6a shows the input adjacency matrix for the subgraph of PHOTO containing the top 160 nodes belonging to the first class sorted in decreasing order of their (within-class) degree. Figure 4.6b illustrates the enhanced adjacency matrix learned by Gelato's MLP. Comparing it with the Euclidean distance between node attributes in Figure 4.6c, we see that our enhanced adjacency matrix effectively incorporates attribute information. More specifically, we notice the down-weighting of the edges connecting the high-degree nodes with larger attribute distances (matrix entries 0-40 and especially 0-10) and the up-weighting of those connecting medium-degree nodes with smaller attribute distances (40-80). In Figure 4.6d and Figure 4.6e, we see the corresponding AC scores based on the input and the enhanced adjacency matrix (Gelato). Since AC captures the degree distribution of nodes [1], the vanilla AC scores greatly favor high-degree nodes (0-40). By comparison, thanks to the down-weighting, Gelato assigns relatively lower scores to edges connecting them to low-degree nodes (60-160), while still capturing the edge



(a) Input adjacency matrix (b) Enhanced adjacency matrix (c) Attribute Euclidean distance

(d) AC scores (e) Gelato scores (f) Neo-GNN predictions

(g) AC predicted edges (h) Gelato predicted edges (i) Neo-GNN predicted edges

Figure 4.6: Illustration of the link prediction process of Gelato, AC, and the best GNN-based approach, Neo-GNN, on a subgraph of Photo. Gelato effectively incorporates node attributes into the graph structure and leverages topological heuristics to enable state-of-the-art link prediction.

density between high-degree nodes (0-40). The immediate result of this is the significantly improved precision as shown in Figure 4.6h compared to Figure 4.6g. Gelato covers as many positive edges in the high-degree region as AC while making far fewer wrong predictions for connections involving low-degree nodes.

The prediction probabilities and predicted edges for Neo-GNN are shown in Figure 4.6f and Figure 4.6i, respectively. Note that while it predicts edges connecting high-degree node pairs (0-40) with high probability, similar values are assigned to many low-degree pairs (80-160) as well. Most of those predictions are wrong, both in the low-degree region of Figure 4.6i and also in other low-degree parts of the graph that are not shown here. This analysis supports our claim that Gelato is more effective at combining node attributes and the graph topology, enabling state-of-the-art link prediction.

4.4.4 Loss and Training Setting

In this section, we demonstrate the advantages of the proposed N-pair loss and *unbiased training* for supervised link prediction. Figure 4.7 shows the training and validation losses and *prec*@100% (our validation metric) in training Gelato based on the cross entropy (CE) and N-pair (NP) losses under *biased* and *unbiased training* respectively. The final test AP scores are shown in the titles.

In the first column (CE with *biased training*), different from the training, both loss and precision for (unbiased) validation decrease. This leads to even worse test performance compared to the untrained base model (i.e., AC). In other words, albeit being the most popular loss function for supervised link prediction, CE with *biased training* does not generalize to *unbiased testing*. On the contrary, as shown in the second column, the proposed NP loss with *biased training*—equivalent to the pairwise logistic loss [197]—is a more effective proxy for *unbiased testing* metrics.

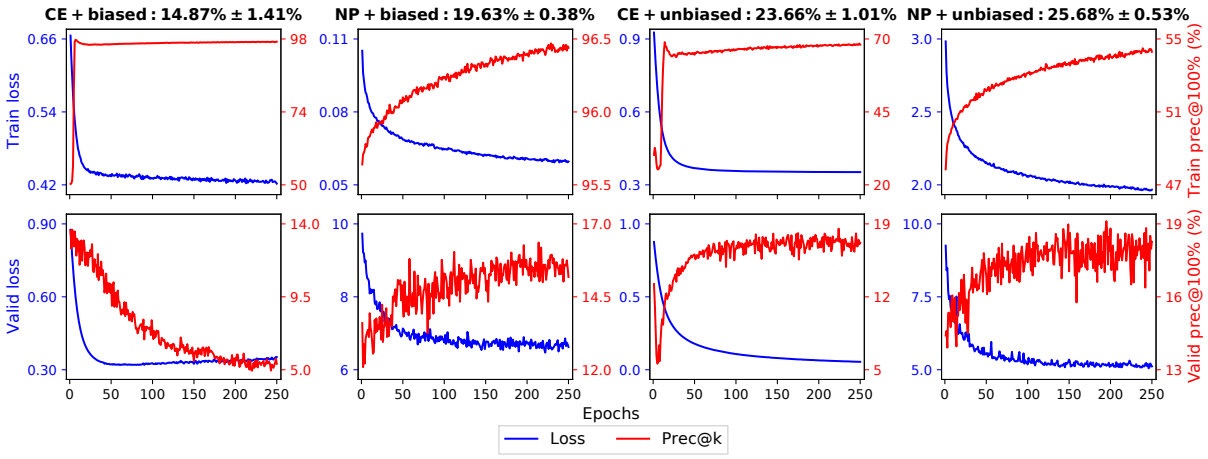


Figure 4.7: Training of Gelato based on different losses and training settings for Photo with test AP (mean ± std) shown in the titles. Compared with the cross entropy loss, the N-pair loss with *unbiased training* is a more consistent proxy for *unbiased testing* metrics and leads to better peak performance.

The right two columns show results with *unbiased training*, which is better for CE as more negative pairs are present in the training set (with the original class ratio). On the other hand, NP is more consistent with unbiased evaluation metrics, leading to 8.5% better performance. This is because, unlike CE, which optimizes positive and negative pairs independently, NP contrasts negative pairs against positive ones, giving higher importance to negative pairs that are more likely to be false positives.

4.4.5 Ablation Study

We have demonstrated the superiority of Gelato over its individual components and two-stage approaches in Table 4.3 and analyzed the effect of losses and training settings in Section 4.4.4. Here, we collect the results with the same hyperparameter setting as Gelato and present a comprehensive ablation study in Table 4.5. Specifically, *Gelato−MLP* (AC) represents Gelato without the MLP (Autocovariance) component, i.e., only using Autocovariance (MLP) for link prediction. *Gelato−NP* (UT) replaces the proposed N-

pair loss (*unbiased training*) with the cross entropy loss (*biased training*) applied by the baselines. Finally, *Gelato−NP+UT* replaces both the loss and the training setting.

	Cora	CiteSeer	PubMed	Photo	Computers
Gelato−MLP	2.43 ± 0.00	2.65 ± 0.00	2.50 ± 0.00	16.63 ± 0.00	11.64 ± 0.00
Gelato−AC	1.94 ± 0.18	3.91 ± 0.37	0.83 ± 0.05	7.45 ± 0.44	4.09 ± 0.16
Gelato−NP+UT	2.98 ± 0.20	1.96 ± 0.11	2.35 ± 0.24	14.87 ± 1.41	9.77 ± 2.67
Gelato−NP	1.96 ± 0.01	1.77 ± 0.20	2.32 ± 0.16	19.63 ± 0.38	9.84 ± 4.42
Gelato−UT	3.07 ± 0.01	1.95 ± 0.05	2.52 ± 0.09	23.66 ± 1.01	11.59 ± 0.35
Gelato	**3.90 ± 0.03**	**4.55 ± 0.02**	**2.88 ± 0.09**	**25.68 ± 0.53**	**18.77 ± 0.19**

Table 4.5: Results of the ablation study based on AP scores. Each component of Gelato plays an important role in enabling state-of-the-art link prediction performance.

We observe that removing either MLP or Autocovariance leads to inferior performance, as the corresponding attribute or topology information would be missing. Further, to address the class imbalance problem of link prediction, both the N-pair loss and *unbiased training* are crucial for effective training of Gelato.

While all supervised baselines originally adopt *biased training*, we also implement the same *unbiased training* (and N-pair loss) as Gelato for those that are scalable. Specifically, Table 4.6 summarizes the link prediction performance in terms of mean and standard deviation of AP for Gelato and the baselines using *unbiased training* without downsampling and the cross entropy loss, and Figure 4.8 and Figure 4.9 show results based on *prec*@k and *hits*@k for varying k values.

We first note that *unbiased training* without downsampling brings serious scalability challenges to most GNN-based approaches, making scaling up to larger datasets intractable. For the scalable baselines, *unbiased training* leads to marginal (e.g., Neo-GNN) to significant (e.g., BScNets) gains in performance. However, all of them still underperform the topological heuristic, Autocovariance, in most cases, and achieve much worse performance compared to Gelato. This supports our claim that the topology-centric graph learning mechanism in Gelato is more effective than the attribute-centric

		Cora	CiteSeer	PubMed	Photo	Computers
GNN	GAE	0.33 ± 0.21	0.69 ± 0.18	0.63*	1.36 ± 3.38	7.91*
	SEAL	2.24*	1.11*	***	***	***
	HGCN	0.54 ± 0.23	1.02 ± 0.05	0.41*	3.27 ± 2.97	2.60*
	LGCN	1.53 ± 0.08	1.45 ± 0.09	0.55*	2.90 ± 0.26	1.13*
	TLC-GNN	0.68 ± 0.16	0.61 ± 0.19	OOM	2.95 ± 1.50	OOM
	Neo-GNN	2.76 ± 0.36	1.80 ± 0.22	***	***	***
	NBFNet	***	***	***	***	***
	BScNets	1.13 ± 0.25	1.27 ± 0.20	0.47*	8.54 ± 1.55	4.40*
	WalkPool	***	***	***	OOM	OOM
Topological Heuristics	CN	1.10 ± 0.00	0.74 ± 0.00	0.36 ± 0.00	7.73 ± 0.00	5.09 ± 0.00
	AA	2.07 ± 0.00	1.24 ± 0.00	0.45 ± 0.00	9.67 ± 0.00	6.52 ± 0.00
	RA	2.02 ± 0.00	1.19 ± 0.00	0.33 ± 0.00	10.77 ± 0.00	7.71 ± 0.00
	AC	2.43 ± 0.00	2.65 ± 0.00	2.50 ± 0.00	16.63 ± 0.00	11.64 ± 0.00
Attributes + Topology	MLP	0.22 ± 0.27	1.17 ± 0.63	0.44 ± 0.12	1.15 ± 0.40	1.19 ± 0.46
	Cos	0.42 ± 0.00	1.89 ± 0.00	0.07 ± 0.00	0.11 ± 0.00	0.07 ± 0.00
	MLP+AC	0.63 ± 0.32	1.00 ± 0.43	1.17 ± 0.44	11.88 ± 0.83	8.73 ± 0.45
	Cos+AC	3.60 ± 0.00	4.46 ± 0.00	0.51 ± 0.00	10.01 ± 0.00	5.20 ± 0.00
	MLP+Cos+AC	3.80 ± 0.01	3.94 ± 0.03	0.77 ± 0.01	12.80 ± 0.03	7.57 ± 0.02
Gelato		**3.90 ± 0.03**	**4.55 ± 0.02**	**2.88 ± 0.09**	**25.68 ± 0.53**	**18.77 ± 0.19**

Table 4.6: Link prediction performance comparison (mean ± std AP) with supervised link prediction methods using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines. (* Run only once as each run takes ~100 hrs; *** Each run takes >1000 hrs; OOM: Out Of Memory.)

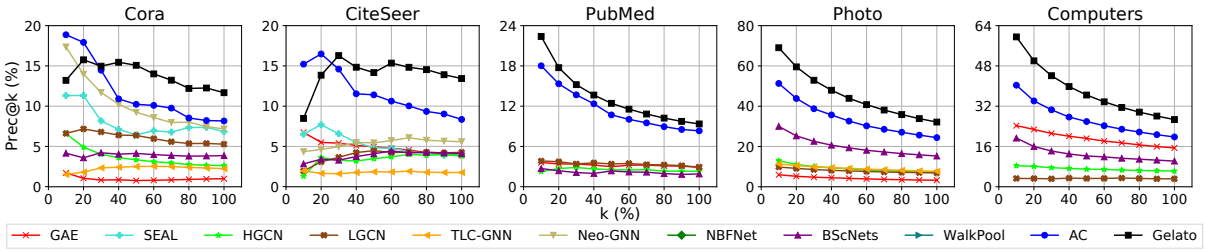


Figure 4.8: Link prediction performance in terms of *prec@k* (in percentage) for varying values of k with baselines using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines.

message-passing in GNNs for link prediction, leading to state-of-the-art performance.

As for the GNN-based baselines using both *unbiased training* and our proposed N-pair loss, we have attempted to modify the training functions of the reference repositories of the baselines and managed to train SEAL, LGCN, Neo-GNN, and BScNet. However, despite our best efforts, we are unable to obtain better link prediction performance using

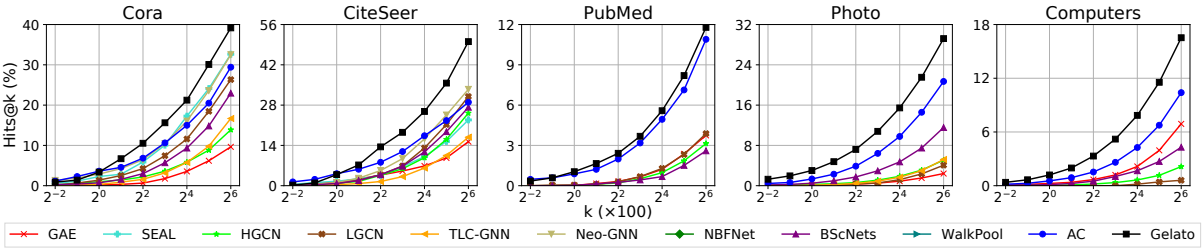


Figure 4.9: Link prediction performance in terms of *hits@k* (in percentage) for varying values of k with baselines using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines.

the N-pair loss. We defer the further investigation of the incompatibility of our loss and the baselines to future research. On the other hand, we have observed significantly better performance for MLP with the N-pair loss compared to the cross entropy loss under *unbiased training*. This can be seen by comparing the MLP performance in Table 4.6 here and the *Gelato*$-AC$ performance in Table 4.5. The improvement shows the potential benefit of applying our training setting and loss to other supervised link prediction methods.

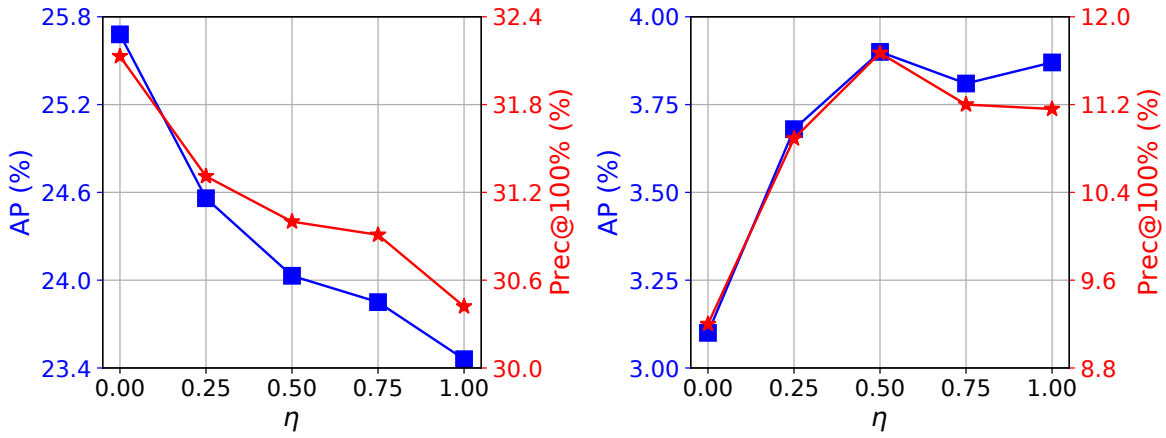
4.4.6 Sensitivity Analysis

The selected hyperparameters of Gelato for each dataset are recorded in Table 4.7, and a sensitivity analysis of η, α, and β are shown in Figure 4.10 and Figure 4.11 respectively for Photo and Cora.

	Cora	CiteSeer	PubMed	Photo	Computers
η	0.5	0.75	0.0	0.0	0.0
α	0.5	0.5	0.0	0.0	0.0
β	0.25	0.5	1.0	1.0	1.0

Table 4.7: Selected hyperparameters of Gelato.

For most datasets, we find that simply setting $\beta = 1.0$ and $\eta = \alpha = 0.0$ leads to



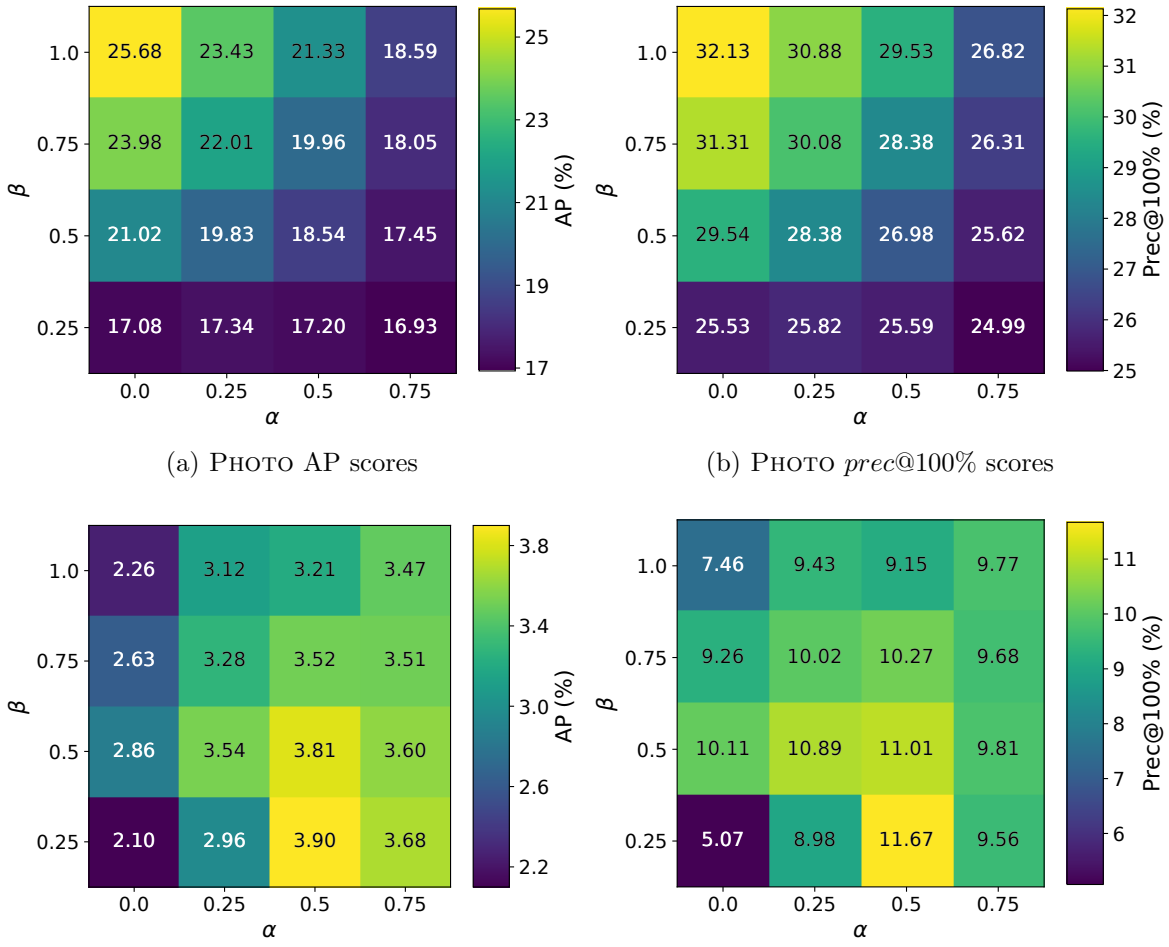
(a) PHOTO performance (b) CORA performance

Figure 4.10: Performance of Gelato with different values of η.

the best performance, corresponding to the scenario where no edges based on cosine similarity are added and the edge weights are completely learned by the MLP. For CORA and CITESEER, however, we first notice that adding edges based on untrained cosine similarity alone leads to improved performance (see Table 4.3), which motivates us to set $\eta = 0.5/0.75$. In addition, we find that a large trainable weight β leads to overfitting of the model as the number of node attributes is large while the number of (positive) edges is small for CORA and CITESEER (see Table 4.1). We address this by decreasing the relative importance of trained edge weights ($\beta = 0.25/0.5$) and increasing that of the topological edge weights ($\alpha = 0.5$), which leads to better generalization and improved performance. Based on our experiments, these hyperparameters can be easily tuned using simple hyperparameter search techniques, such as line search, using a small validation set.

4.4.7 Running Time

We compare Gelato with other supervised link prediction methods in terms of running time for PHOTO in Figure 4.12. As the only method that applies *unbiased training* by



(a) Photo AP scores

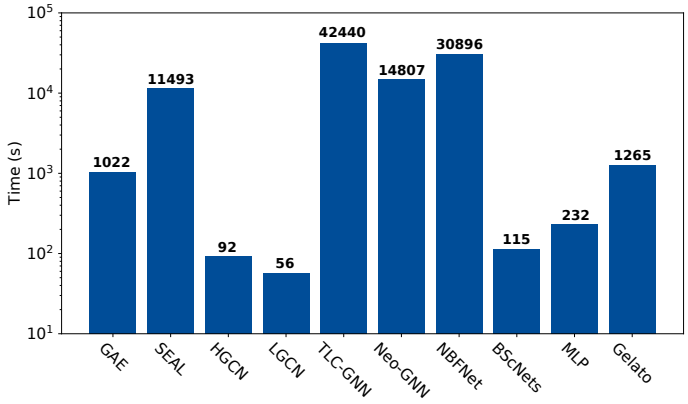
(b) Photo *prec*@100% scores

(c) Cora AP scores

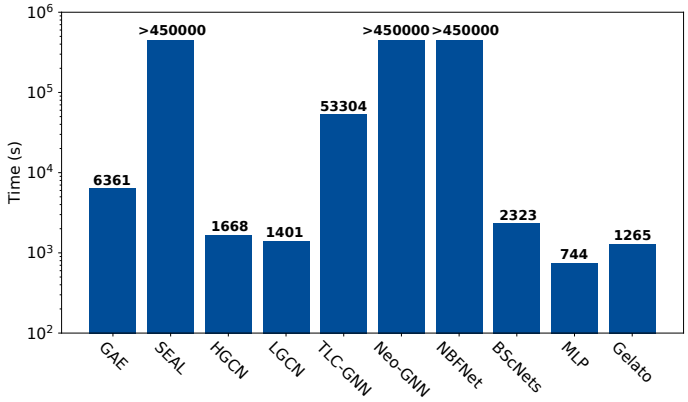
(d) Cora *prec*@100% scores

Figure 4.11: Performance of Gelato with different α and β.

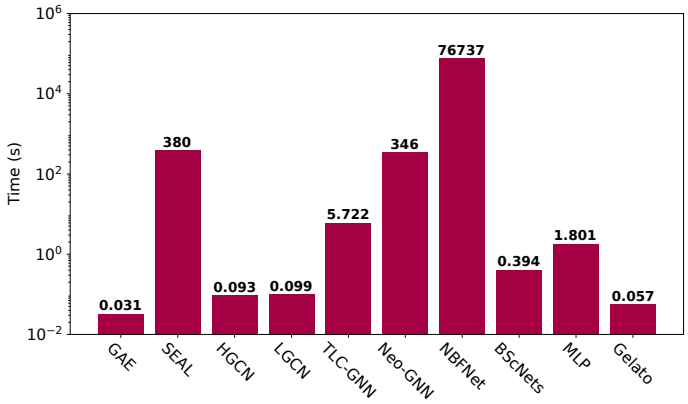
default—with more negative samples—Gelato shows a competitive training speed that is 11× faster than the best performing GNN-based method, Neo-GNN. And when baselines also adopt *unbiased training*, Gelato is the second fastest model, only slower than the vanilla MLP. In terms of inference time, Gelato is much faster than most baselines with a 6,000× speedup compared to Neo-GNN. We further observe more significant efficiency gains for Gelato over Neo-GNN for larger datasets—e.g., 14× (training) and 25,000× (testing) for Computers.



(a) Training time with baselines adopting *biased training*



(b) Training time with baselines adopting *unbiased training*



(c) Inference time per *unbiased testing*

Figure 4.12: Training and inference time comparison between supervised link prediction methods for Photo. Gelato has competitive training time (even when baselines adopt *biased training*) and is significantly faster than most baselines in terms of inference, especially compared to the best GNN model, Neo-GNN.

4.5 Related Work

Topological heuristics for link prediction. The early link prediction literature focuses on topology-based heuristics. This includes approaches based on local (e.g., Common Neighbors [154], Adamic Adar [155], and Resource Allocation [195]) and higher-order (e.g., Katz [198], PageRank [40], and SimRank [199]) information. More recently, random-walk based graph embedding methods, which learn vector representations for nodes [8, 9, 1], have achieved promising results in graph machine learning tasks. Popular embedding approaches, such as DeepWalk [8] and node2vec [9], have been shown to implicitly approximate the Pointwise Mutual Information similarity [11], which can also be used as a link prediction heuristic. This has motivated the investigation of other similarity metrics such as Autocovariance [28, 1, 2]. However, these heuristics are unsupervised and cannot take advantage of data beyond the topology.

Graph Neural Networks for link prediction. GNN-based link prediction addresses the limitations of topological heuristics by training a neural network to combine topological and attribute information and potentially learn new heuristics. GAE [160] combines a graph convolution network [142] and an inner product decoder based on node embeddings for link prediction. SEAL [151] models link prediction as a binary subgraph classification problem (edge/non-edge), and follow-up work (e.g., SHHF [187], WalkPool [153]) investigates different pooling strategies. Other recent approaches for GNN-based link prediction include learning representations in hyperbolic space (e.g., HGCN [161], LGCN [162]), generalizing topological heuristics (e.g., Neo-GNN [152], NBFNet [165]), and incorporating additional topological features (e.g., TLC-GNN [164], BScNets [166]). Motivated by the growing popularity of GNNs for link prediction, this work investigates key questions regarding their training, evaluation, and ability to learn effective topological heuristics directly from data. We propose Gelato, which is simpler, more accurate,

and faster than the state-of-the-art GNN-based link prediction methods.

Graph learning. Gelato learns a graph that combines topological and attribute information. Our goal differs from generative models [200, 201, 202], which learn to sample from a distribution over graphs. Graph learning also enables the application of GNNs when the graph is unavailable, noisy, or incomplete (for a recent survey, see [203]). LDS [204] and GAug [205] jointly learn a probability distribution over edges and GNN parameters. IDGL [206] and EGLN [207] alternate between optimizing the graph and embeddings for node/graph classification and collaborative filtering. [208] proposes two-stage link prediction by augmenting the graph as a preprocessing step. In comparison, Gelato effectively learns a graph in an end-to-end manner by minimizing the loss of a topological heuristic.

4.6 Conclusion

This work sheds light on key limitations in how GNN-based link prediction methods handle the intrinsic class imbalance of the problem and presents more effective and efficient ways to combine attributes and topology. Our findings might open new research directions on machine learning for graph data, including a better understanding of the advantages of increasingly popular and sophisticated deep learning models compared to more traditional and simpler graph-based solutions.

Chapter 5

Global Counterfactual Explainer for Graph Neural Networks

5.1 Introduction

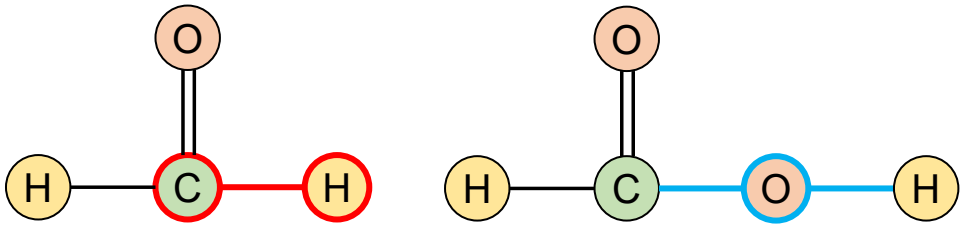
Graph Neural Networks (GNNs) [142, 143, 144, 209, 210, 211] are being used in many domains such as drug discovery [212], chip design [213], combinatorial optimization [214], physical simulations [215, 216] and event prediction [217, 218, 219]. Taking the graph(s) as input, GNNs are trained to perform various downstream tasks that form the core of many real-world applications. For example, graph classification has been applied to predict whether a drug would exhibit the desired chemical activity [212]. Similarly, node prediction is used to predict the functionality of proteins in protein-protein interaction networks [220] and categorize users into roles on social networks [221].

Despite the impressive success of GNNs on predictive tasks, GNNs are *black-box* machine learning models. It is non-trivial to explain or reason why a particular prediction is made by a GNN. Explainability of a prediction model is important to understand its shortcomings and identify areas for improvement. In addition, the ability to explain a

model is critical towards making it trustworthy. Owing to this limitation of GNNs, there has been significant efforts in recent times towards explanation approaches.

Existing work on explaining GNN predictions can be categorized mainly in two directions: 1) factual reasoning [222, 223, 224, 225], and 2) counterfactual reasoning [226, 227, 228, 229]. Generally speaking, the methods in the first category aim to find an important subgraph that correlates most with the underlying GNN prediction. In contrast, the methods with counterfactual reasoning attempt to identify the smallest amount of perturbation on the input graph that changes the GNN's prediction, for example, removal/addition of edges or nodes.

Compared to factual reasoning, counterfactual explainers have the additional advantage of providing the means for recourse [230]. For example, in the applications of drug discovery [212, 231], mutagenicity is an adverse property of a molecule that hampers its potential to become a marketable drug [232]. In Figure 5.1, formaldehyde is classified by a GNN to be mutagenic. Factual explainers can attribute the subgraph containing the carbon-hydrogen bond to the cause of mutagenicity, while counterfactual explainers provide an effective way (i.e., a recourse) to turn formaldehyde into formic acid, which is non-mutagenic, by replacing a hydrogen atom with a hydroxyl.



(a) Formaldehyde (b) Formic acid

Figure 5.1: Formaldehyde (a) is classified by a GNN to be an undesired mutagenic molecule with its important subgraph found by factual reasoning highlighted in red. Formic acid (b) is its non-mutagenic counterfactual example obtained by removing one edge and adding one node and two edges.

In this work, we focus on counterfactual explanations. Our work is based on the obser-

vation that existing counterfactual explainers [222, 223, 224, 225] for graphs take a *local* perspective, generating counterfactual examples for individual input graphs. However, this approach has two key limitations:

- **Lack of global insights:** It is desirable to offer insights that generalize across a multitude of data graphs. For example, instead of providing formic acid as a counterfactual example to formaldehyde, we can summarize global recourse rules such as "*Given any molecule with a carbonyl group (carbon-oxygen double bond), it needs a hydroxyl to be non-mutagenic*". This focus on global counterfactual explanation promises to provide higher-level insights that are complementary to those obtained from local counterfactual explanations.

- **Information overload:** The primary motivation behind counterfactual analysis is to provide human-intelligible explanations. With this objective, consider real-world graph datasets that routinely contain thousands to millions of graphs. Owing to instance-specific counterfactual explanations, the number of counterfactual graphs grows linearly with the graph dataset size. Consequently, the sheer volume of counterfactual graphs overloads the human cognitive ability to process this information. Hence, the initial motivation of providing human-intelligible insights is lost if one does not obtain a holistic view of the counterfactual graphs.

Contributions: In this work, we study the problem of model-agnostic, *global* counterfactual explanations of GNNs for graph classification. More specifically, given a graph dataset, our goal is to counterfactually explain the largest number of input graphs with a small number of counterfactuals. As we will demonstrate later in our experiments, this formulation naturally forces us to remove redundancy from instance-specific counterfactual explanations and hence has higher information density. Algorithmically, the proposed problem introduces new challenges. We theoretically establish that the pro-

posed problem is NP-hard. Furthermore, the space of all possible counterfactual graphs itself is exponential. Our work overcomes these challenges and makes the following contributions:

- **Novel formulation:** We formulate the novel problem of global counterfactual reasoning/explanation of GNNs for graph classification. In contrast to existing works on counterfactual reasoning that only generate instance-specific examples, we provide an explanation on the global behavior of the model.
- **Algorithm design:** While the problem is NP-hard, we propose GCFExplainer, which organizes the exponential search space as an *edit map*. We then perform *vertex-reinforced random walks* on it to generate diverse, representative counterfactual candidates, which are *greedily summarized* as the global explanation.
- **Experiments:** We conduct extensive experiments on real-world datasets to validate the effectiveness of the proposed method. Results show that GCFExplainer not only provides important high-level insights on the model behavior but also outperforms state-of-the-art baselines related to counterfactual reasoning in various recourse quality metrics.

5.2 Global Counterfactual Explanations

This section introduces the global counterfactual explanation (GCE) problem for graph classification. We start with the background on local counterfactual reasoning. Then, we propose a representation of the global recourse rule that provides a high-level counterfactual understanding of the classifier behavior. Finally, we introduce quality measures for recourse rules and formally define the GCE problem.

5.2.1 Local Counterfactual

Consider a graph $G = (V, E)$, where V and E are the sets of (labelled) nodes and edges respectively. A (binary) graph classifier (e.g., a GNN) ϕ classifies G into either the undesired class ($\phi(G) = 0$) or the desired one ($\phi(G) = 1$). An explanation of ϕ seeks to answer how these predictions are made. Those based on factual reasoning analyze what properties G possesses to be classified in the current class while those based on counterfactual reasoning find what properties G needs to be assigned to the opposite class.

Existing counterfactual explanation methods take a local perspective. Specifically, for each input graph G, they find a **counterfactual** (graph) C that is somewhat similar to G but is assigned to a different class. Without loss of generality, let G belong to the undesired class, i.e., $\phi(G) = 0$, then the counterfactual C satisfies $\phi(C) = 1$. The similarity between C and G is quantified by a predefined distance metric d, for example, the number of added/removed edges [226, 227].

In our work, we consider the graph edit distance (GED) [233], a more general distance measure, as the distance function to account for other types of changes. Specifically, $\text{GED}(G_1, G_2)$ counts the minimum number of "edits" to convert G_1 to G_2. An "edit" can be the addition or removal of edges and nodes, or change of node labels (see Figure 5.2). Moreover, to account for graphs of different sizes, we normalize the GED by the sizes of graphs: $\widehat{\text{GED}}(G_1, G_2) = \text{GED}(G_1, G_2)/(|V_1| + |V_2| + |E_1| + |E_2|)$. Nonetheless, our method can be applied with other graph distance metrics, such as those based on graph kernels (e.g., RW [234], NSPDG [235], WL [236]).

The distance function measures the quality of the counterfactual found by the explanation model. Ideally, the counterfactual C should be very close to the input graph G while belonging to a different class. Formally, we define the counterfactuals that are

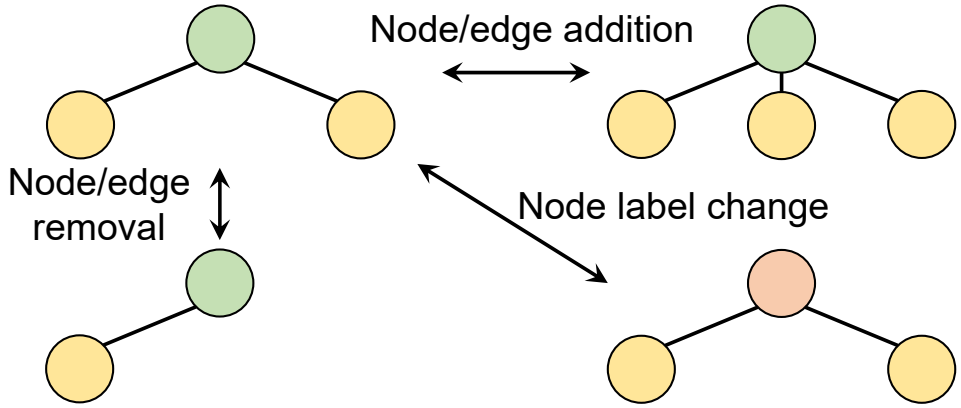


Figure 5.2: Edits between graphs.

within a certain distance θ from the input graph as **close counterfactuals**.

Definition 5.1 (Close Counterfactual) *Given the GNN classifier ϕ, distance parameter θ, and an input graph G with undesired outcome, i.e., $\phi(G) = 0$; a counterfactual graph, C, is a close counterfactual of G when $\phi(C) = 1$ and $d(G, C) \leq \theta$.*

While the (close) counterfactual C found by existing methods explains the classifier behavior for the corresponding input graph G, it is hard to generalize to understand the global pattern. Next, we introduce the global recourse rule that provides a high-level summary of the classifier behavior across different input graphs.

5.2.2 Global Recourse Representation

The global counterfactual explanation requires a global recourse rule r. Specifically, for any (undesired) input graph G with $\phi(G) = 0$, r provides a (close) counterfactual (i.e., a recourse) for G: $\phi(r(G)) = 1$. While both a recourse rule and a local counterfactual explainer find a counterfactual given an input graph, their goals are different. The goal of the local counterfactual explainer is to find the best (closest) counterfactual possible for each input graph, and therefore, r can be very complicated, e.g., in the form of an

optimization algorithm [227, 229]. On the other hand, a recourse rule aims to provide an explanation of the classifier's global behavior, which requires a simpler form that is understandable for domain experts without prior knowledge of deep learning on graphs.

Existing global recourse rules for classifiers with feature vectors as input take the form of short decision trees [237]. However, this is hard to be generalized to graph data with rich structure information. Instead, we propose the representation of a global recourse rule for a graph classifier to be a collection of counterfactual graphs $\mathbb{C}$ in the desired class that are *diverse* and *representative* enough to capture its global behavior. This representation does not require any additional knowledge for domain experts to understand and draw insights from, similar to the local counterfactual examples. It is also easy to find the local counterfactual for a given input graph G based on $\mathbb{C}$ by nominating the closest graph in $\mathbb{C}$: $r(G) = \arg\min_{C \in \mathbb{C}} d(G, C)$.

5.2.3 Quantifying Recourse Quality

Given a graph classifier ϕ and a set of n input graphs $\mathbb{G}$ in the undesired class, we want to compare the quality of different recourse representations $\mathbb{C}$. Similar to the quality metrics introduced for vector data [237], we aim to account for the following factors:

1. **Coverage**: Like local counterfactual explainers, we want to ensure that counterfactuals found for individual input graphs are of high quality. Specifically, we introduce recourse coverage—the proportion of input graphs that have close counterfactuals from $\mathbb{C}$ under a given distance threshold θ:

$$\textbf{coverage}(\mathbb{C}) = |\{G \in \mathbb{G} \mid \min_{C \in \mathbb{C}} \{d(G, C)\} \leq \theta\}|/|\mathbb{G}|$$

2. **Cost**: Another quality metric based on local counterfactual quality is the recourse

cost (i.e., the distance between the input graph and its counterfactual) across the input graphs:

$$\textbf{cost}(\mathbb{C}) = \underset{G \in \mathbb{G}}{\text{agg}}\{\min_{C \in \mathbb{C}}\{d(G, C)\}\}$$

where agg is an aggregation function, e.g., mean or median.

3. **Interpretability**: Finally, the recourse rule should be easy (small) enough for human cognition. We quantify the interpretability as the size of recourse representation:

$$\textbf{size}(\mathbb{C}) = |\mathbb{C}|$$

5.2.4 Problem Formulation and Characterization

An ideal recourse representation should maximize the coverage while minimizing the cost and the size. Formally, we define the global counterfactual explanation problem (GCE) as follows:

Definition 5.1 (Global Counterfactual Explanation for Graph Classification) *Given a GNN graph classifier ϕ that classifies n input graphs $\mathbb{G}$ to the undesired class 0 and a budget $k \ll n$, our goal is to find the best recourse representation $\mathbb{C}$ that maximizes the recourse coverage with size k:*

$$\max_{\mathbb{C}} \boldsymbol{coverage}(\mathbb{C}) \ s.t. \ \ \boldsymbol{size}(\mathbb{C}) = k$$

We note that in our problem formulation only coverage and size are explicitly accounted for, whereas cost is absent. We make this design choice since cost and coverage are intrinsically opposing forces. Specifically, if we are willing to allow a high cost, coverage increases since we allow for higher individual distances between an input graph

and its counterfactual. Therefore, we take the approach of binding the cost to the distance threshold θ in the coverage definition. Nonetheless, an explicit analysis of all these metrics including cost is performed to quantify recourse quality during our empirical evaluation in Section 5.4. Below we discuss the hardness of GCE.

Theorem 5.1 (NP-hardness) *The GCE problem is NP-hard.*

Proof: To establish NP-hardness of the proposed problem we reduce it from the classical *Maximum Coverage* problem.

Definition 5.2 (Maximum Coverage) *Given a budget k and a collection of subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ from a universe of items $U = \{u_1, \cdots, u_n\}$, find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of sets such that $|\mathcal{S}'| \leq k$ and the number of covered elements $|\bigcup_{\forall S_i \in \mathcal{S}'} S_i|$ is maximized.*

We show that given any instance of a maximum coverage problem $\langle \mathcal{S}, U \rangle$, it can be mapped to a GCE problem. For u_i, we construct a star graph with a center node with an empty label and n leaf nodes with $n-1$ empty labels and one label u_i. For S_i, we construct a similar star graph with a center node with a special label γ and n leaf nodes with $|S_i|$ labeled with the elements in S_i and $n - |S_i|$ with empty labels. The classifier ϕ classifies a graph as a desired one if and only if it is a star graph with a γ-labeled central node and n leaf nodes with a set of labels among $\mathcal{S} = \{S_1, \cdots, S_m\}$. The allowed edit operations are either adding or deleting a set of labels (as a single edit), but not both together. So, each S_i corresponds to a counterfactual candidate C_i and $d(G_j, C_i) \leq \theta = 1$ if and only if $u_j \in S_i$. With this construction, it is easy to see that an optimal solution for this instance of GCE is the optimal solution for the corresponding instance of the maximum coverage problem. ■

Owing to NP-hardness, it is not feasible to identify the optimal solution for the GCE problem in polynomial time unless NP = P. In the next section, we will introduce GCFExplainer, an effective and efficient heuristic that solves the GCE problem.

5.3 Proposed Method: GCFExplainer

In this section, we propose GCFExplainer, the first global counterfactual explainer for graph classification. The GCE problem requires us to find a collection of k counterfactual graphs that maximize the coverage of the input graphs. Intuitively, we want each individual counterfactual graph to be a close counterfactual to (i.e., "cover") as many input graphs as possible. Additionally, different counterfactual graphs should cover different sets of input graphs to maximize the overall coverage. These intuitions motivate the design of our algorithm GCFExplainer, which has three major components:

1. **Structuring the search space:** The search space of counterfactual graphs consists of *all* graphs that are in the same domain as the input graphs and within a distance of θ. In other words, any graph within a distance of θ from an input graph may be a potential counterfactual candidate and therefore needs to be analyzed. The number of potential graphs within θ increases exponentially with θ since the space of graph edits is combinatorial [238, 239]. GCFExplainer uses an *edit map* to organize these graphs as a meta-graph $\mathcal{G}$, where individual nodes are graphs that are created via a different number of edits from the input graphs and each edge represents a single edit.

2. **Vertex-reinforced random walk:** To search for good counterfactual candidates, GCFExplainer leverages vertex-reinforced random walks (VRRW) [240] on the edit map $\mathcal{G}$. VRRW has the nice property of converging to a set of nodes that are both important (i.e., cover many input graphs) and diverse (i.e., non-overlapping coverage), which will form a small set of counterfactual candidates for further processing.

3. **Iterative computation of the summary:** After obtaining good counterfactual

candidates from VRRW, GCFExplainer creates the final set of the counterfactual graphs (i.e., the summary) as the recourse representation by iteratively adding the best candidate based on the maximal gain of the coverage given the already added candidates.

5.3.1 Structuring the Search Space

The search space for counterfactual graphs in GCFExplainer is organized via an edit map $\mathcal{G}$. The edit map is a meta-graph whose nodes are graphs in the same domain as the input graphs and edges connect graphs that differ by a single graph edit. As an example, each graph in Figure 5.2 represents a node in the edit map, and the arrows denote edges between graphs (nodes) that are one edit away. In the edit map, we only include connected graphs since real graphs of interest are often connected (e.g., molecules, proteins, etc.).

While all potential counterfactual candidates are included as its nodes, the edit map has an exponential size and it is computationally prohibitive to fully explore it. However, a key observation is that a counterfactual candidate can only be a few hops away from some input graph. Otherwise, the graph distance between the counterfactual and the input graph would be too large for the counterfactual to cover it. This observation motivates our exploration of the edit map to be focused on the union of close neighborhoods of the input graphs (see Section 5.3.2). Additionally, while we cannot compute the entire edit map, it is easy to chart the close neighborhoods by iteratively performing all possible edits from the input graphs. Next, we introduce the vertex-reinforced random walk to efficiently explore the edit map to find counterfactual candidates.

5.3.2 Vertex-Reinforced Random Walk

Vertex-reinforced random walk (VRRW) [240] is a time-variant random walk. Different from other more widely applied random walk processes such as the simple random walk and the PageRank [8, 145, 241, 242], the transition probability $p(u, v)$ of VRRW from node u to node v depends not only on the edge weight $w(u, v)$ but also the *number of previous visits* in the walk to the target node v, which we denote using $N(v)$. Specifically,

$$p(u, v) \propto w(u, v)N(v) \tag{5.1}$$

GCFExplainer applies VRRW on the edit map and produces n most frequently visited nodes in the walk as the set of counterfactual candidates $\mathbb{S}$. Next, we formalize VRRW in our setting and explain how it surfaces good counterfactual candidates for GCE.

Vertex-reinforcement

Our main motivation for using VRRW to explore the edit map instead of other random walk processes is that VRRW converges to a diverse and representative set of nodes [243, 244] in different regions of the edit map. In this way, the frequently visited nodes in instances of VRRW have the potential to be good counterfactual candidates as they would cover a diverse set of input graphs in the edit map. The reason behind the diversity of the highly visited nodes is the previous visit count $N(v)$ in the transition probability. Specifically, nodes with larger visit counts tend to be visited more often later ("richer gets richer"), and thereby dominating all other nodes in their neighborhood. This leads to a bunch of highly visited nodes to "represent" each region of the edit map. We refer the readers to [243] for details on the mathematical basis and the theoretical correctness of this property. Moreover, as our goal is to find counterfactual candidates, we only

reinforce (i.e., increase the visit counts of) graphs in the counterfactual class.

Importance function

While the vertex-reinforcement mechanism ensures diversity of the highly visited nodes, we still need to guide the walker to visit graphs that are good counterfactual candidates. We achieve this by assigning large edge weight $w(u, v)$ to good counterfactual candidates via an importance function $I(v)$:

$$w(u, v) = I(v) \tag{5.2}$$

The importance function $I(v)$ should capture the quality of a graph v as a counterfactual candidate. It has the following components:

1. Counterfactual probability $p_\phi(v)$. The graph classifier ϕ predicts a probability for v to be in the counterfactual class ($\phi(v) = 1$). By using it as part of the importance function, the walker is encouraged to visit regions with rich counterfactual graphs.

2. Individual coverage $\textbf{coverage}(\{v\})$. The individual coverage of a graph v computes the proportion of input graphs that are close to v. This encourages the walker to visit graphs that cover a large number of input graphs.

3. Gain of coverage $\textbf{gain}(v; \mathbb{S})$. Given a graph v and the current set of counterfactual candidates $\mathbb{S}$ (i.e., the n most frequently visited nodes), we can compute the gain between the current coverage and the coverage after adding v to $\mathbb{S}$:

$$\textbf{gain}(v; \mathbb{S}) = \textbf{coverage}(\mathbb{S} \cup \{v\}) - \textbf{coverage}(\mathbb{S})$$

This guides the walker to find graphs that complement the current counterfactual

candidates to cover additional input graphs.

The importance function is a combination of these components:

$$I(v) = p_\phi(v)(\alpha\,\textbf{coverage}(\{v\}) + (1-\alpha)\,\textbf{gain}(v;\mathbb{S})) \tag{5.3}$$

where α is a hyperparameter between 0 and 1. With the above importance function, the VRRW in GCFExplainer converges to a set of diverse nodes that have high counterfactual probability and collectively cover a large number of input graphs.

Dynamic teleportation

The last component of VRRW, teleportation, is to help us manage the exponential search space of the edit map. Since our goal is to find close counterfactuals to the input graphs, the walker only needs to explore the nearby regions of the input graphs. Therefore, we start the walk from the input graphs, and also at each step, let the walker teleport back (i.e. transit) to a random input graph with probability τ.

To decide which input graph to teleport to, we adopt a dynamic probability distribution based on the current counterfactual candidate set $\mathbb{S}$. Specifically, let $g(G) = |\{v \in \mathbb{S} \mid d(v,G) \leq \theta \text{ and } \phi(v) = 1\}|$ be the number of close counterfactuals in $\mathbb{S}$ covering an input graph G. Then the probability to teleport to G is

$$p_\tau(G) = \frac{\exp(-g(G))}{\sum_{G' \in \mathbb{G}} \exp(-g(G'))} \tag{5.4}$$

This dynamic teleportation favors input graphs that are not well covered by the current solution set and encourages the walker to explore nearby counterfactuals to cover them after teleportation.

5.3.3 Iterative Computation of the Summary

We have applied VRRW to generate a good set of n counterfactual candidates $\mathbb{S}$. In the last step of GCFExplainer, we aim to further refine the candidate set and create the final recourse representation (i.e., the summary) with k counterfactual graphs. This summarization problem is also NP-hard and we propose to build $\mathbb{C}$ in an iterative and greedy manner from $\mathbb{S}$.

Specifically, we start with an empty solution set $\mathbb{C}_0$. Then, for each iteration t, we add the graph v to $\mathbb{C}_t$ with the maximal gain of coverage $\mathbf{gain}(v; \mathbb{C}_t)$. This is repeated k times to get the final recourse representation $\mathbb{C}$ with k graphs. It is easy to show that the summarization problem is submodular and therefore, our greedy algorithm provides $(1 - 1/e)$-approximation.

Notice that the greedy algorithm can also be applied to the local counterfactuals found by existing methods to generate a GCE solution. Here, we highlight three advantages of GCFExplainer:

1. Existing local counterfactual explainers [226, 227, 228, 229] are only able to generate counterfactuals based on one type of graph edits—edge removal, while GCFExplainer incorporates all types of edits to include a richer set of counterfactual candidates.

2. The set of counterfactual candidates from GCFExplainer is generated with the GCE objective in mind, while the local counterfactuals from existing methods are optimized for individual input graphs. Therefore, they may not be good candidates to capture the global behavior of the classifier.

3. It is easy to incorporate domain constraints (e.g., the valence of chemical bonds) into GCFExplainer by pruning the neighborhood of the edit map, while existing

methods based on optimization require non-trivial efforts to customize.

We will empirically demonstrate the superiority of GCFExplainer to this two-stage approach with state-of-the-art local counterfactual explanation methods in our experiments in Section 5.4.2.

Pseudocode and complexity: The pseudocode of GCFExplainer is presented in Algorithm 5.1. Line 1-16 summarizes the VRRW component of GCFExplainer. Specifically, Line 3-10 determines the next graph to visit based on VRRW transition probabilities and the dynamic teleportation, and Line 11-16 update the visit counts and the set of counterfactual candidates. The iterative computation of the counterfactual summary is described in Line 17-21. The overall complexity of GCFExplainer is $O(Mhn + kn)$, where M is the number of iterations for the VRRW, h is the average node degree in the meta-graph, n is the number of input graphs, and k is the size of the global counterfactual representation. In practice, we store the computed transition probabilities with a space-saving algorithm [245] to improve the running time of GCFExplainer.

5.4 Experiments

We provide empirical results for the proposed GCFExplanier along with baselines on commonly used graph classification datasets. Our code is available at `https://github.com/mertkosan/GCFExplainer`.

5.4.1 Experimental Settings

Datasets

We use four different real-world datasets for graph classification benchmark with their statistics in Table 5.1. Specifically, NCI1 [246], Mutagenicity [247, 232], and

Algorithm 5.1 GCFExplainer(ϕ, $\mathbb{G}$)

$G \leftarrow$ random input graph from $\mathbb{G}$, $N(G) \leftarrow 1$, $\mathbb{S} = \{G\}$
for $i \in 1 : M$ **do**
  Let $\epsilon \sim Bernoulli(\tau)$
  if $\epsilon = 0$ **then**
    for $v \in Neighbors(G)$ **do**
      Compute $I(v)$ based on Equation 5.3
      Compute $p(G, v)$ based on Equation 5.1
    $v \leftarrow$ random neighbor of G based on $p(G, v)$
  else
    $v \leftarrow$ random input graph from $\mathbb{G}$ based on Equation 5.4
  if $\phi(v) = 1$ **then**
    if $v \in \mathbb{S}$ **then**
      $N(v) \leftarrow N(v) + 1$
    else
      $\mathbb{S} \leftarrow \mathbb{S} + \{v\}$, $N(v) \leftarrow 1$
  $G \leftarrow v$
$\mathbb{S} \leftarrow$ top n frequently visited counterfactuals in $\mathbb{S}$
$\mathbb{C} \leftarrow \emptyset$
for $t \in 1 : k$ **do**
  $v \leftarrow \arg\max_{v \in \mathbb{S}}$ **gain**$(v; \mathbb{C})$
  $\mathbb{C} \leftarrow \mathbb{C} + \{v\}$
return $\mathbb{C}$

AIDS [247] are collections of molecules with nodes representing different atoms and edges representing chemical bonds between them. The molecules are classified by whether they are anticancer, mutagenic, and active against HIV, respectively. Proteins [220, 248] is a collection of proteins classified into enzymes and non-enzymes, with nodes representing secondary structure elements and edges representing structural proximity. For all datasets, we filter out graphs containing rare nodes with label frequencies smaller than 50.

	NCI1	Mutagenicity	AIDS	Proteins
#Graphs	3978	4308	1837	1113
#Nodes	118714	130719	28905	43471
#Edges	128663	132707	29985	81044
#Node Labels	10	10	9	3

Table 5.1: The statistics of the datasets.

Graph classifier

We follow [224] and train a GNN with 3 convolution layers [142] of embedding dimension 20, a max pooling layer, and a fully connected layer for classification. The model is trained with the Adam optimizer [57] and a learning rate of 0.001 for 1000 epochs. The datasets are split into 80%/10%/10% for training/validation/testing with the model accuracy shown in Table 5.2.

	NCI1	MUTAGENICITY	AIDS	PROTEINS
Training	0.8439	0.8825	0.9980	0.7800
Validation	0.8161	0.8302	0.9727	0.8198
Testing	0.7809	0.8000	0.9781	0.7297

Table 5.2: Accuracy of the GNN graph classifier.

Baselines

To the best of our knowledge, GCFEXPLAINER is the first global counterfactual explainer. To validate its effectiveness, we compare it against state-of-the-art local counterfactual explainers combined with the greedy summarization algorithm described in Section 5.3.3. The following local counterfactual generation methods are included in our experiments.

- GROUND-TRUTH: Using graphs belonging to the desired class from the original dataset as local counterfactuals.
- RCEXPLAINER [227]: Local counterfactual explainer based on the modeling of implicit decision regions of GNNs.
- CFF [229]: Local counterfactual explainer based on joint modeling of factual and counterfactual reasoning.

Explainer settings

We use a distance threshold θ of 0.05 for training all explainers. Since computing the exact graph edit distance is NP-hard, we apply a state-of-the-art neural approximation algorithm [239]. For GCFExplainer, we set the teleportation probability $\tau = 0.1$ and tune α, the weight between individual coverage and gain of coverage, from $\{0, 0.5, 1\}$. A sensitivity analysis is presented in Section 5.4.6. The number of VRRW iterations M is set to 50000, which is enough for convergence as shown in Section 5.4.5. For baselines, we tune their hyperparameters to achieve the best local counterfactual rates while maintaining an average distance to input graphs that is smaller than the distance threshold θ.

5.4.2 Recourse Quality

We start by comparing the recourse quality between GCFExplainer and baselines. Table 5.3 shows the recourse coverage with $\theta = 0.1$ and median recourse cost of the top 10 counterfactual graphs (i.e., $k = 10$). We first notice that the two state-of-the-art local counterfactual explainers have similar performance as Ground-Truth, consistent with our claim that local counterfactual examples from existing methods are not good candidates for a global explanation. The proposed GCFExplainer, on the other hand, achieves significantly better performance for global recourse quality. Compared to the best baseline, RCExplainer, GCFExplainer realizes a **46.9%** gain in recourse coverage and a **9.5%** reduction in recourse cost.

Next, we show the recourse coverage and cost for different sizes of counterfactual summary in Figure 5.3. As expected, adding more graphs to the recourse representation increases recourse coverage while decreasing recourse cost, at the cost of interpretability. And GCFExplainer maintains a constant edge over the baselines.

	NCI1		Mutagenicity		AIDS		Proteins	
	Coverage	Cost	Coverage	Cost	Coverage	Cost	Coverage	Cost
Ground-Truth	16.54%	0.1326	28.96%	0.1275	0.41%	0.2012	8.47%	0.2155
RCExplainer	15.22%	0.1370	31.99%	0.1290	8.96%	0.1531	8.74%	0.2283
CFF	17.61%	0.1331	30.43%	0.1327	3.39%	0.1669	3.83%	0.2557
GCFExplainer	**27.85%**	**0.1281**	**37.08%**	**0.1135**	**14.66%**	**0.1516**	**10.93%**	**0.1856**

Table 5.3: Recourse coverage ($\theta = 0.1$) and median recourse cost comparison between GCFExplainer and baselines for a 10-graph global explanation. GCFExplainer consistently and significantly outperforms all baselines across different datasets.

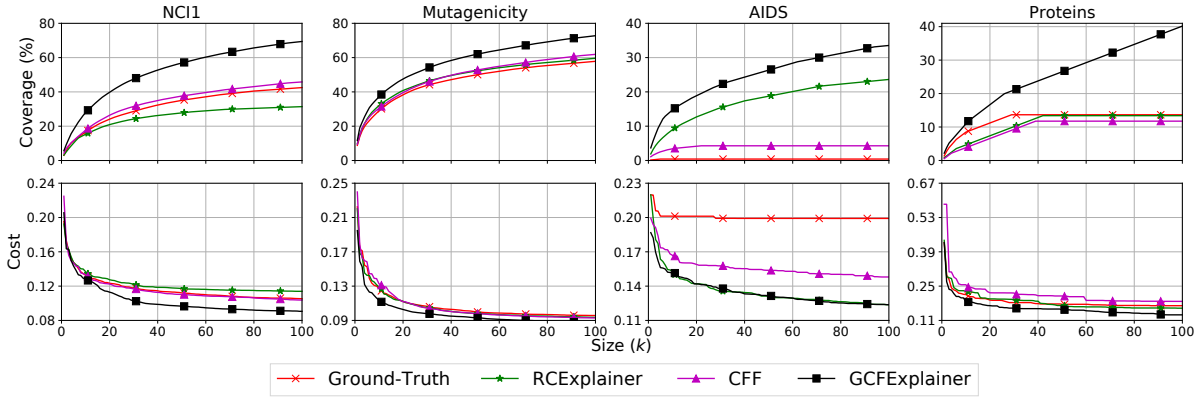


Figure 5.3: Coverage and cost performance comparison between GCFExplainer and baselines based on different counterfactual summary sizes. GCFExplainer consistently outperforms the baselines across different sizes.

We also compare the recourse coverage based on different distance thresholds θ, with results shown in Figure 5.4. While coverage increases for all methods as the threshold increases, GCFExplainer consistently outperforms the baselines across different thresholds.

5.4.3 Global Counterfactual Insight

We have demonstrated the superiority of GCFExplainer based on various quality metrics for global recourse. Here, we show how GCFExplainer provides global insights compared to local counterfactual examples. Figure 5.5 illustrates (a) four input undesired graphs with a similar structure from the AIDS dataset, (b) corresponding local

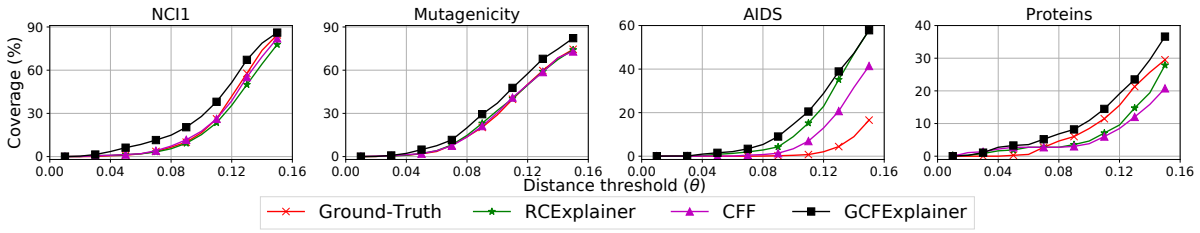
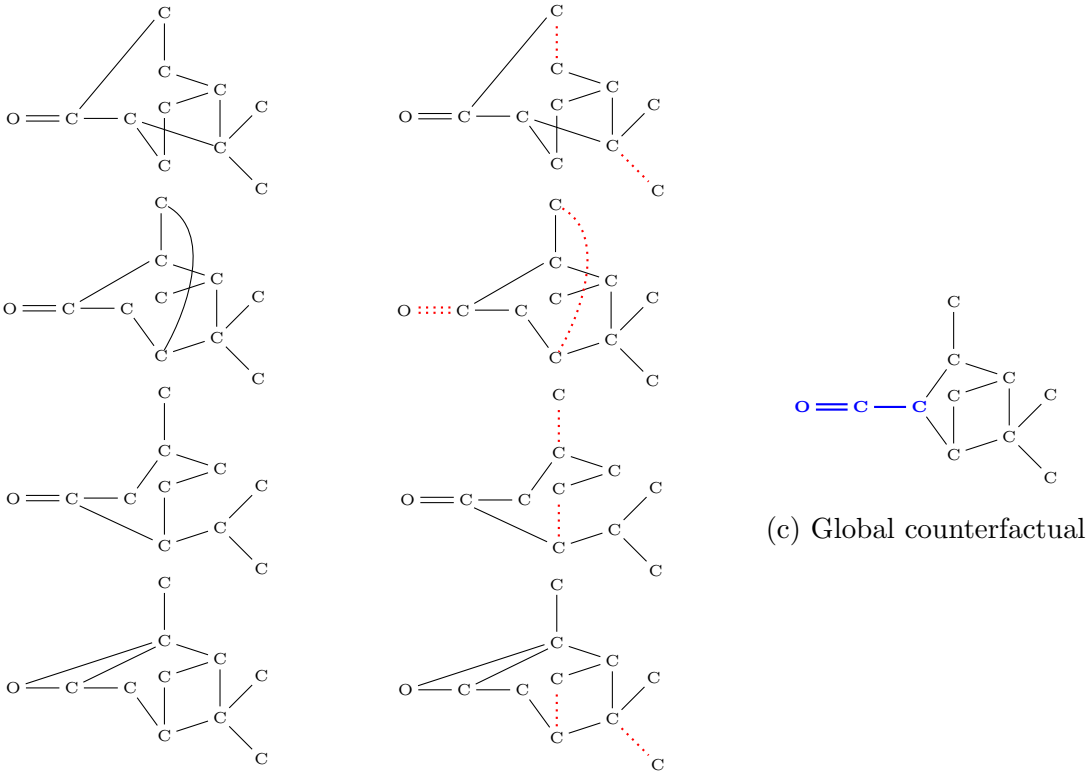


Figure 5.4: Recourse coverage comparison between GCFExplainer and baselines based on different distance threshold values (θ). GCFExplainer consistently outperforms the baselines across different θ.

counterfactual examples (based on RCExplainer and CFF), and (c) the representative global counterfactual graph from GCFExplainer covering the input graphs. Our goal is to understand why the input graphs are inactive against AIDS (undesired) and how to obtain the desired property with minimal changes.

The local counterfactuals in (b) attribute the classification results to different edges in individual graphs (shown as red dotted lines) and recommend their removal to make input graphs active against HIV. Note that while only two edits are proposed for each individual graph, they appear at different locations, which are hard to generalize for a global view of the model behavior. In contrast, the global counterfactual graph from GCFExplainer presents a high-level recourse rule. Specifically, the carbon atom with the carbon-oxygen bond is connected to *two* other carbon atoms in the input graphs, making them ketones (with a C=O bond) or ethers (with a C-O bond). On the other hand, the global counterfactual graph highlights a different functional group, aldehyde (shown in blue), to be the key for combating AIDS. In aldehydes, the carbon atom with a carbon-oxygen bond is only connected to *one* other carbon atom, leading to different chemical properties compared to ketones and ethers. Indeed, aldehydes have been shown to be effective HIV protease inhibitors [249].

Finally, this case study also demonstrates that counterfactual candidates found by



(a) Input graphs (b) Local counterfactuals (c) Global counterfactual

Figure 5.5: Illustration of global and local counterfactual explanations for the AIDS dataset. The global counterfactual graph (c) presents a high-level recourse rule—changing ketones and ethers into aldehydes (shown in blue)—to combat HIV, while the edge removals (shown in red) recommended by local counterfactual examples (b) are hard to generalize.

GCFExplainer are better for global explanation than local counterfactuals. We note that while the graph edit distance between the local counterfactuals and their corresponding input graphs is only 2, they do not cover other similarly structured input graphs (with distance > 5). Meanwhile, our global counterfactual graph covers all input graphs (with distance ≤ 4).

5.4.4 Ablation Study

We then conduct an ablation study to investigate the effectiveness of GCFExplainer components. We consider three alternatives:

- GCFExplainer-NVR: no vertex-reinforcement ($N(v) = 1$)
- GCFExplainer-NIF: no importance function ($I(v) = 1$)
- GCFExplainer-NDT: no dynamic teleportation ($p_\tau(G) = 1/|\mathbb{G}|$)

The coverage results are shown in Table 5.4. We observe decreased performance when any of GCFExplainer components is absent.

	NCI1	Mutagenicity	AIDS	Proteins
GCFExplainer-NVR	24.56%	35.44%	11.33%	8.56%
GCFExplainer-NIF	13.29%	29.16%	4.54%	6.83%
GCFExplainer-NDT	27.34%	36.35%	14.05%	9.28%
GCFExplainer	**27.85%**	**37.08%**	**14.66%**	**10.93%**

Table 5.4: Ablation study results based on recourse coverage.

5.4.5 Convergence Analysis

In this subsection, we show the empirical convergence of VRRW based on the Mutagenicity dataset in Figure 5.6. We observe that the coverage performance for different summary sizes starts to converge after 15000 iterations and fully converges after 50000 iterations, which is the number we applied in our experiments.

5.4.6 Sensitivity Analysis

The only hyperparameter of GCFExplainer we tune is α in Equation 5.3 that weights the individual coverage and gain of coverage for the importance function. Table 5.5 shows the results based on different α. While GCFExplainer outperforms

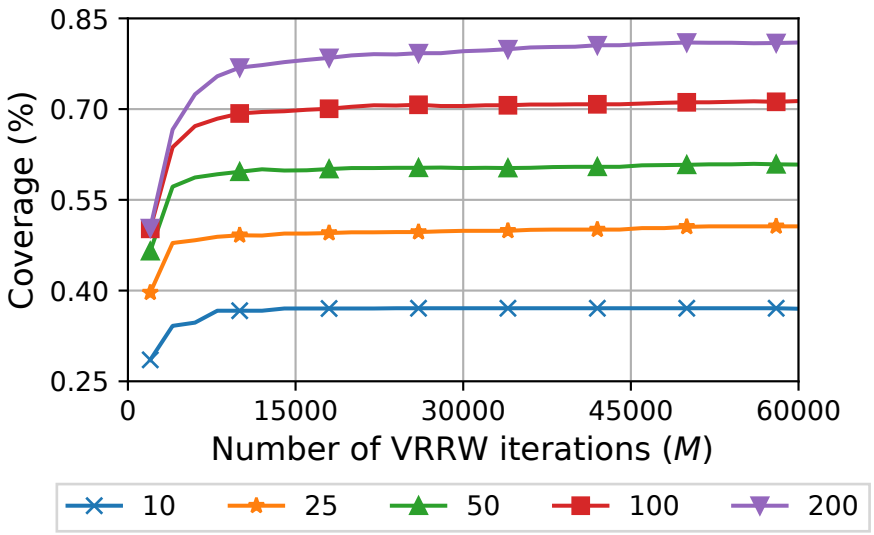


Figure 5.6: Convergence of VRRW for the MUTAGENICITY dataset based on recourse coverage with different summary sizes. VRRW fully converges after $M = 50000$ iterations.

baselines with all different α, we observe that individual coverage works better for NCI1 and gain of cumulative coverage works better for other datasets.

	NCI1	**Mutagenicity**	**AIDS**	**Proteins**
$\alpha = 0.0$	**27.85%**	36.87%	12.83%	10.11%
$\alpha = 0.5$	27.50%	36.59%	**14.66%**	10.38%
$\alpha = 1.0$	22.27%	**37.08%**	13.99%	**10.93%**

Table 5.5: Sensitivity analysis on α, the weight between individual coverage and gain of coverage in the importance function.

5.4.7 Running Time

Table 5.6 summarizes the running times of generating counterfactual candidates based on different methods. GCFEXPLAINER has a competitive running time albeit exploring more counterfactual graphs in the process. We also include results for GCFEXPLAINER-S which samples a maximum of 10000 neighbors for computing the importance at each step. It achieves better running time at a negligible cost of 3.3% performance loss on

average. Finally, summarizing the counterfactual candidates takes less than a second for all methods.

	NCI1	MUTAGENICITY	AIDS	PROTEINS
RCEXPLAINER	30454	52549	29047	8444
CFF	22794	31749	21296	**6412**
GCFEXPLAINER	19817	24006	2615	19246
GCFEXPLAINER-S	**19365**	**18798**	**2539**	7429

Table 5.6: Counterfactual candidates generation time comparison. GCFEXPLAINER (-S) has competitive running time albeit exploring more counterfactual graphs.

5.5 Related Work

Explanations for Graph Neural Networks. There is much research [222, 223, 224, 225] on explaining graph neural networks (GNNs). The first proposed method, GNNExplainer [222], finds the explanatory subgraph and sub-features by maximizing the mutual information between the original prediction and the prediction based on the subgraph and sub-features. Later, PGExplainer [223] provides an inductive framework that extracts GNN node embeddings and learns to map embedding pairs to the probability of edge existence in the explanatory weighted subgraph. PGMExplainer [224] builds a probabilistic explanation model that learns new predictions from perturbed node features, performs variable selection using Markov blanket of variables, and then produces a Bayesian network via structure learning. In XGNN [225], the authors find model-level explanations by a graph generation module that outputs a sequence of edges using reinforcement learning. These explanation methods focus on *factual* reasoning while the goal of our work is to provide a global *counterfactual* explanation for GNNs.

Counterfactual Explanations. Recently, there are several attempts to have explanations of graph neural networks (GNNs) via counterfactual reasoning [226, 227, 228, 229].

One of the earlier methods, CF-GNNExplainer [226], provides counterfactual explanations in terms of a learnable perturbed adjacency matrix that leads to the flipping of classifier prediction for a node. On the other hand, RCExplainer [227] aims to find a robust subset of edges whose removal changes the prediction of the remaining graph by modeling the implicit decision regions based on GNN graph embeddings. In [228], the authors investigate counterfactual explanations for a more specific class of graphs—the brain networks—that share the same set of nodes by greedily adding or removing edges using a heuristic. More recently, the authors of CFF [229] argue that a good explanation for GNNs should consider both factual and counterfactual reasoning and they explicitly incorporate those objective functions when searching for the best explanatory subgraphs and sub-features. Counterfactual reasoning has also been applied for link prediction [250]. All the above methods produce *local* counterfactual examples while our work aims to provide a *global* explanation in terms of a summary of representative counterfactual graphs.

5.6 Conclusion

We have proposed GCFExplainer, the first global counterfactual explainer for graph classification. Compared to local explainers, GCFExplainer provides a high-level picture of the model behavior and effective global recourse rules. We hope that our work will not only deepen our understanding of graph neural networks but also build a bridge for experts from other domains to leverage deep learning models for high-stakes decision-making.

Chapter 6

Other Problems Related to Multiscale Graphs

Many real-world complex systems can be modeled as multiscale networks, where nodes form dense clusters at different structural scales. For example, as shown in Figure 6.1, the global network of airports connected by domestic and international flights can be naturally organized into country and continent-level clusters [69]. And biological networks such as structural representation of proteins also exhibit a hierarchy of increasingly finer partitions from motifs such as α-helices and β-sheets (secondary structures) to sequences of amino acids (primary structures) [28].

In this chapter, we summarize our ongoing works that are related to representation learning for multiscale networks. Specifically, we first look at the problem of how to leverage multiscale neighborhood information for effective semi-supervised node classification with Graph Convolutional Networks. We then compare different quality functions for multiscale community detection and investigate how to find the natural scale for each community. Finally, we propose a novel method based on graph autoencoders for multiscale graph anomaly detection.

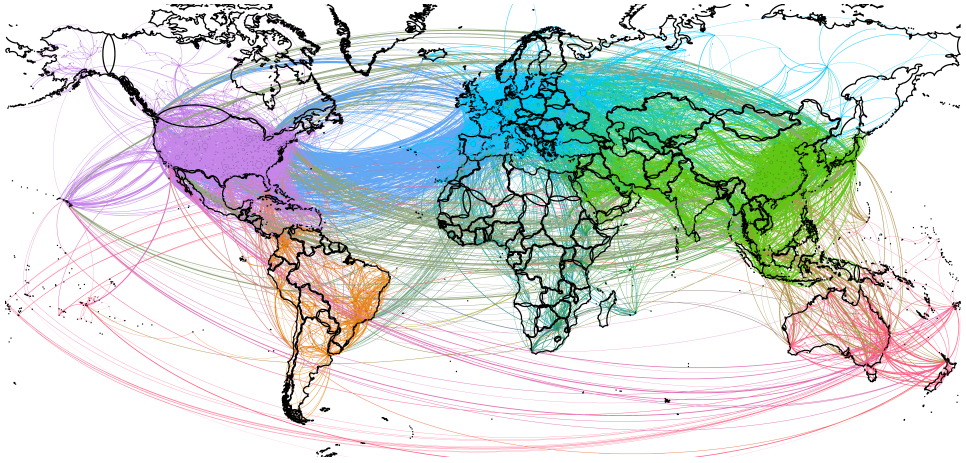


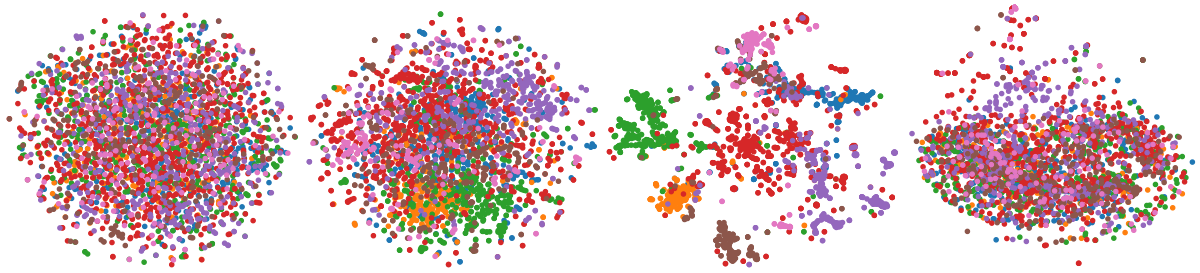
Figure 6.1: Global airport network where different airports are organized by countries and continents.

6.1 Graph Neural Diffusion Networks

6.1.1 Overview

Graph Neural Networks (GNNs) [142, 143, 144] have achieved great success in node classification by learning neighborhood-enriched node embeddings. Graph Convolutional Networks (GCN) [142] is a pioneering GNN model that introduces a simple but well-behaved layer-wise message passing rule, which is derived from the first-order approximation of spectral graph convolutions [251]. However, determining the scales of neighborhoods from which the information is aggregated is not a trivial task, especially when the labels are scarce. Specifically, stacking too many graph convolutional layers leads to over-smoothing, i.e., nodes from different classes become indistinguishable, while stacking too few layers leads to under-smoothing, i.e., node features may not be sufficiently propagated for effective embeddings. Figure 6.2 illustrates this idea by showing the t-SNE visualization [61] of the GCN node embeddings based on a randomized weight matrix and k layers of neighborhood aggregation for the Cora dataset [188]. By selecting the right scale of the neighborhood for aggregation (in this case, $k = 19$), the ground-truth

cluster structure can be revealed even with an untrained weight matrix.



(a) $k = 0$ (b) $k = 1$ (c) $k = 19$ (d) $k = 10,000$

Figure 6.2: The t-SNE visualization of the GCN node embeddings based on a randomized weight matrix and k layers of neighborhood aggregation for the CORA dataset. Selecting $k = 19$ as the scale of aggregation reveal the cluster structures, while neither small ($k = 0, 1$) nor large scales ($k = 10,000$) achieve comparable effects. Colors denote ground-truth node labels.

There are several attempts in the current literature to address the selection of the right scales for GCNs. JK-Nets [252] proposes to aggregate the output of each layer by skipping connections, which allows selective information aggregation from neighborhoods of different scales. However, their performance improvement over vanilla GCN is marginal, possibly due to their stacking of too many graph convolutional layers, which is hard to train. Simple Graph Convolution (SGC) [146] achieves similar results to GCN by removing all the nonlinear activation functions and raising the graph Laplacian matrix to the K-th power, but as a cost, the local neighborhood information is discarded. More recently, PPNP [253] and GDC [254] adopt aggregation weights from existing graph diffusions such as the personalized PageRank diffusion [255] and the heat kernel diffusion [256]. Their key weakness is that the diffusion weights for different scales are predefined and fixed, which limits their adaptivity to datasets with different characteristics. Besides, the closed-form solution of the personalized PageRank diffusion or the heat kernel diffusion is also computationally expensive.

To properly address the multiscale neighborhood aggregation problem of GCNs, we

propose to learn the aggregation weights for different structural scales in a single layer. The usage of a shallow architecture mitigates the hard-to-train problem of over-stacked GCNs while exploiting both local and global information in an integral manner. Since explicitly computing each scale of the node neighborhood information encoded in different powers of the graph Laplacian matrix is less efficient, we analyze the layer-wise propagation rule of GCN from the perspective of power iteration and derive a sequence of matrices. Then, we apply Single-Layer Perceptrons (SLPs) and Multi-Layer Perceptrons (MLPs) to learn the aggregation weights. Differing from linear graph diffusions adopted by existing methods, the aggregation weights in our model are learned by neural networks and can adapt to different datasets, which we call neural diffusions. In addition, linear diffusion dynamics may not capture the complex relationships between nodes. Thus, we also derive a variant of neural diffusions from nonlinear graph diffusion [257], by adopting the MLP to learn the nonlinear functions in the dynamical system. We call our GCN model equipped with neural diffusions GND-Nets (for Graph Neural Diffusion Networks). And as we will show in our experiments, GND-Nets outperforms other GNNs in semi-supervised node classification, especially for datasets with very sparse node labels.

Our main contributions can be summarized as follows:

- We interpret the layer-wise propagation rule of GCNs from the perspective of power iteration and propose to solve both the over-smoothing and under-smoothing of GCNs by aggregating multiscale neighborhood information in a single layer. The information is encoded in a sequence of matrices generated during the convergence process of power iteration.

- We propose a new class of graph diffusions called neural diffusions with three variants. The first two variants adopt the SLP and MLP to aggregate the multiscale

neighborhood information. The last variant adopts the MLP to learn the nonlinear function in the dynamical system. The utilization of neural networks enables neural diffusions to adapt to different datasets.

- We develop a new GNN model called GND-Nets that combines GCNs and graph neural diffusions and demonstrate its effectiveness and efficiency by carrying out extensive comparative studies with state-of-the-art methods on various sparsely-labeled graphs.

6.1.2 Method

Let $G = (V, E, X)$ be an attributed graph of interest where V is the set of n nodes, E is the set edges, $A \in \mathbb{R}^{n\times n}$ is the corresponding adjacency matrix and $X = (x_1, ..., x_n)^\top \in \mathbb{R}^{n\times d}$ is the d-dimensional node attribute (feature) matrix. Further, denote $\widetilde{A} = A + I$ as the adjacency matrix with self-loops and $\widetilde{D}$ as the corresponding degree matrix. Then the corresponding random-walk transition matrix $\widetilde{M}$ can be computed as $\widetilde{D}^{-1}\widetilde{A}$. We want to solve the semi-supervised node classification problem, defined as follows:

Definition 6.1 (Semi-Supervised Node Classification) *Given a graph $G = (V, E, X)$, nodes $\{v_1, \cdots, v_m\}$ are labeled as $Y_l = [y_1, \cdots, y_m]^\top, m \ll n$ (y_m denotes the one-hot encoding of the class label) and nodes $\{v_{m+1}, \cdots, v_n\}$ are unlabeled. The goal is to learn a classifier to infer the labels $\hat{Y}_u = [\hat{y}_{m+1}, \cdots, \hat{y}_n]^\top$ of the unlabeled nodes based on the given information.*

Graph Diffusions

Different graph diffusion dynamics can be generalized as follows:

$$u^{(K)} = \sum_{k=0}^{K-1} \alpha_k \widetilde{M}^k u^{(0)} \tag{6.1}$$

where $u^{(0)}$ is a vector of length n with its entries recording the initial material at each node, α_k is a sequence of nonnegative decaying weights satisfying $\sum_k \alpha_k = 1$, and $u^{(K)}$ captures how the material diffuses through the graph. If α_k takes the form of $\alpha_k = (1-\gamma)\cdot\gamma^k$ with teleport probability $\gamma \in (0,1)$, Equation 6.1 becomes the personalized PageRank diffusion [255]; if α_k takes the form of $\alpha_k = \exp(-t)\frac{t^k}{k!}$ with the diffusion time t, Equation 6.1 becomes the heat kernel diffusion [256].

We can also consider the continuous-time diffusion from the perspective of the dynamical system. Specifically, the heat kernel diffusion is a closed-form solution ($u^{(\infty)} = \exp(-t\widetilde{M})u^{(0)}$) of the following differential equation:

$$\frac{\partial u}{\partial t} = -\widetilde{M}g(u) \tag{6.2}$$

where $g(u) = u$.

Multiscale Neighborhood Information

By replacing the initial diffusion material vector $u^{(0)}$ with the projected node attribute matrix $Z = X\Theta$ in Equation 6.1 and Equation 6.2, where Θ is a learnable weight matrix, we can get the embedding matrix H encoding the multiscale neighborhood information as follows:

$$H = \sum_{k=0}^{K-1} \alpha_k \widetilde{M}^k Z \tag{6.3}$$

$$\frac{\partial H}{\partial t} = -\widetilde{M}g(H) \tag{6.4}$$

Neural Diffusions

Instead of using fixed diffusion weights (α_k and $g(\cdot)$) from existing diffusion dynamics, we propose to learn them directly from data with the three following parameterizations:

- GND-Nets-SLP: $H = \text{SLP}\left([Z, \widetilde{M}Z, \cdots, \widetilde{M}^{K-1}Z]\right) = \sigma\left((\sum_{k=0}^{K-1} \beta_k \widetilde{M}^k)Z\right)$, where β_k are the learnable weights, and σ is an non-linear activation function.
- GND-Nets-MLP: $H = \text{MLP}\left([Z, \widetilde{M}Z, \cdots, \widetilde{M}^{K-1}Z]\right)$, where the MLP is parameterized to take the stacked multiscale neighborhood information matrices as input.
- GND-Nets-DS: We parameterize the differential equation (Equation 6.4) as $\partial H/\partial t = -\widetilde{M}\text{MLP}(H)$, and then approximate the derivative with forward Euler integration to get the final embedding $H = H^{(K)}$ via iterative relationship $H^{(k+1)} = H^{(k)} - \widetilde{M}\text{MLP}(H^{(k)})$, where $H^{(0)} = Z$.

Finally, we add a fully connected layer parameterized by Θ' with softmax activation to predict the label probabilities:

$$Y = \text{softmax}(H\Theta') \tag{6.5}$$

6.1.3 Results

We compare GND-Nets with ten state-of-the-art GNN models for semi-supervised node classification: GCN [142], ChebyNet [258], JK-Nets [252], SGC [146], PPNP and its variant PPNP-HK [253], N-GCN [259], MixHop [260], LanczosNet [261], and DCNN [262]. The average classification accuracy for Cora [188], PubMed [190], and Computers [191] datasets are shown in Table 6.1, Table 6.2, and Table 6.3, respectively. For all datasets with different numbers of labeled nodes per class, our GND-Nets consistently outperforms existing approaches.

We also show the learned mean absolute weights (β_ks) from GND-Nets-SLP for Cora and PubMed in Figure 6.3. Compared to existing methods that adopt the fixed diffusion weights, GND-Nets learns the optimal weights for different datasets to enable

labeled nodes/class	1	2	3	4	5
GND-Nets-SLP	**56.04±11.54**	66.15±8.58	70.52±6.38	73.81±3.62	74.79±3.15
GND-Nets-MLP	55.85±10.54	**66.45±7.30**	**71.14±6.06**	**74.40±3.41**	**75.55±3.92**
GND-Nets-DS	55.98±8.54	63.22±9.59	70.02±6.54	71.57±5.22	73.82±4.77
GCN	36.33±14.90	52.03±16.63	60.71±10.90	66.24±4.57	68.61±3.89
ChebyNet	24.21±9.47	32.63±13.31	39.18±13.49	47.78±11.80	55.38±8.67
JK-Nets	43.67±8.66	55.65±7.15	60.61±6.19	64.58±4.47	66.99±4.29
SGC	38.97±11.81	55.59±8.31	61.31±6.73	65.50±4.49	67.44±3.75
PPNP	45.91±12.46	59.47±9.21	65.66±7.73	69.48±5.86	71.43±5.08
PPNP-HK	35.99±12.17	57.64±12.26	64.78±7.81	69.07±4.91	71.27±4.56
N-GCN	42.03±11.14	53.06±7.48	58.09±5.14	61.98±4.16	64.21±3.37
MixHop	31.09±13.46	44.50±11.61	52.41±8.99	58.38±8.51	62.07±8.03
LanczosNet	44.34±10.46	55.71±6.88	61.01±5.96	64.75±4.10	66.72±4.63
DCNN	21.52±9.04	26.55±10.54	36.90±9.34	44.72±5.97	49.19±4.13

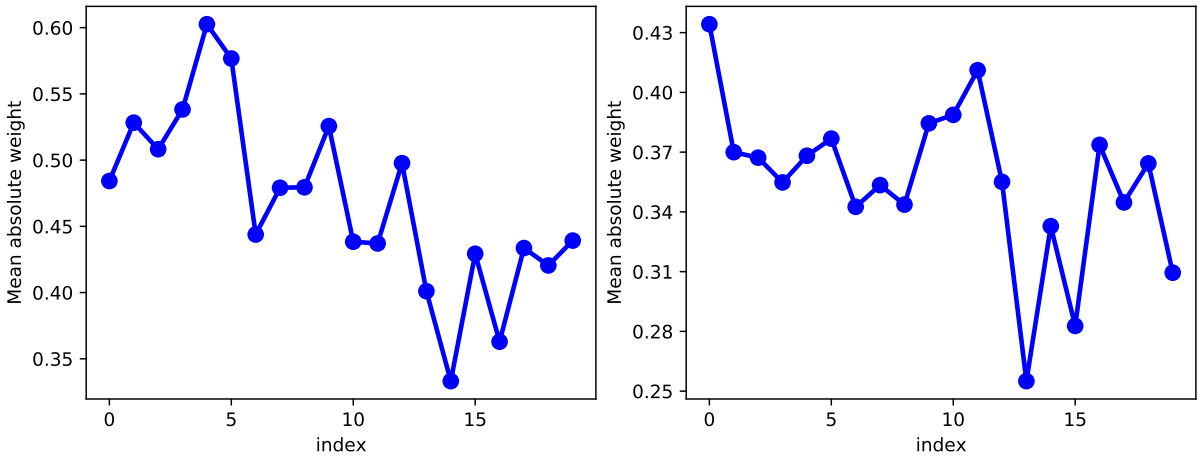
Table 6.1: Average classification accuracy (%) over 30 different data splits on Cora with varying numbers of labeled nodes. GND-Nets consistently outperforms existing approaches at all label sparsity levels.

labeled nodes/class	1	2	3	4	5
GND-Nets-SLP	58.63±9.83	65.52±9.57	68.12±5.94	**70.43±4.87**	71.10±4.48
GND-Nets-MLP	58.19±8.67	65.25±8.44	67.08±8.58	69.17±4.64	69.88±5.23
GND-Nets-DS	**59.44±10.21**	**65.75±8.96**	**69.08±5.44**	69.90±4.93	**71.51±4.53**
GCN	47.34±11.72	58.66±9.62	62.40±7.66	66.02±5.43	67.47±4.40
ChebyNet	45.95±8.56	52.81±10.94	58.26±8.02	60.57±8.19	62.31±6.92
JK-Nets	49.38±11.05	59.02±9.45	62.94±7.41	65.25±5.56	66.46±5.78
SGC	53.18±10.00	60.68±7.38	64.29±5.20	66.69±4.59	67.56±4.18
PPNP	OOM	OOM	OOM	OOM	OOM
PPNP-HK	OOM	OOM	OOM	OOM	OOM
N-GCN	51.37±10.02	58.89±8.27	62.81±4.42	64.81±4.73	66.17±4.30
MixHop	44.50±13.84	48.11±13.66	55.73±9.31	60.02±7.03	62.16±6.83
LanczosNet	52.21±10.12	60.55±10.51	65.00±5.59	67.51±4.72	68.27±4.56
DCNN	49.61±7.83	58.01±7.82	61.01±6.63	63.43±5.26	65.49±4.73

Table 6.2: Average classification accuracy (%) over 30 different data splits on Pubmed with varying numbers of labeled nodes (OOM: Out of Memory). GND-Nets consistently outperforms existing approaches at all label sparsity levels.

labeled nodes/class	1	2	3	4	5
GND-Nets-SLP	33.48±12.87	42.82±15.27	47.71±17.82	50.40±20.28	58.52±16.55
GND-Nets-MLP	**38.26±12.76**	**46.75±18.41**	**57.89±16.24**	**64.03±15.08**	**68.90±6.88**
GND-Nets-DS	23.14±12.32	29.78±10.05	34.06±11.01	43.39±9.14	50.30±10.61
GCN	10.54±8.95	12.64±11.12	13.41±12.75	12.13±10.89	13.30±14.29
ChebyNet	12.50±5.66	11.84±5.39	11.68±4.92	11.49±4.96	13.06±5.04
JK-Nets	8.31±8.81	9.29±7.96	12.36±9.42	10.71±9.14	10.91±8.10
SGC	10.66±10.66	13.49±10.76	12.38±11.23	13.43±11.82	13.25±12.84
PPNP	OOM	OOM	OOM	OOM	OOM
PPNP-HK	OOM	OOM	OOM	OOM	OOM
N-GCN	13.84±10.85	15.57±7.84	16.52±7.21	18.75±8.07	19.35±7.45
MixHop	10.27±7.82	10.91±8.16	10.34±7.17	11.57±9.12	10.95±6.71
LanczosNet	9.49±6.59	9.15±6.99	9.13±6.74	7.69±4.63	7.01±4.14
DCNN	9.35±9.73	12.25±11.88	13.84±14.36	13.47±14.92	17.28±18.71

Table 6.3: Average classification accuracy (%) over 30 different data splits on Computers with varying numbers of labeled nodes (OOM: Out of Memory). GND-Nets consistently outperforms existing approaches at all label sparsity levels.



(a) Cora (b) PubMed

Figure 6.3: Learned mean absolute weights (β_ks) from GND-Nets-SLP for different neighborhood scales. GND-Nets learns the optimal weights for different datasets to enable better performance for node classification.

better performance for node classification.

6.2 Multiscale Community Detection

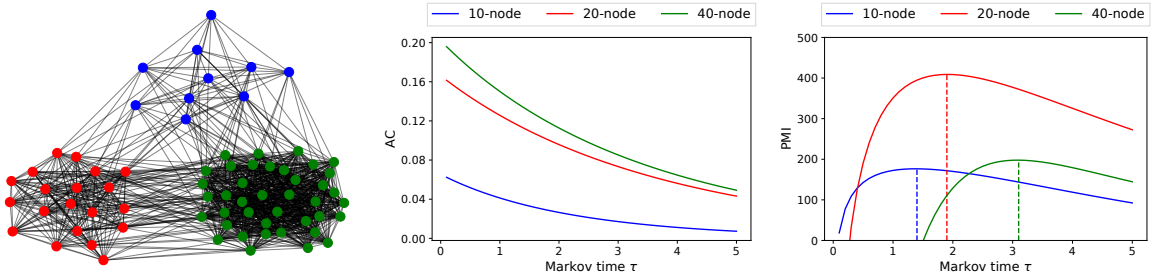
6.2.1 Overview

Community detection, which consists of grouping nodes in a graph such that they are connected densely within each community (cluster) and sparsely across communities, is one of the long-standing graph-based machine learning tasks [58]. Most existing community detection algorithms start with a predefined quality function for different community partitions, such as normalized cut [263] or modularity [111]. Since optimizing those quality functions exactly is usually NP-hard, many heuristics have been proposed to find the best community partition approximately [264, 265, 266].

However, extending the quality functions to detect communities at different structural scales is non-trivial. One of the recent approaches, Markov Stability [28, 36], designs a random-walk based multiscale node similarity metric, Autocovariance (AC), and constructs a clustered version of it as the community quality function. By tuning the length of the random-walk, clustered AC enables the detection of community structures at different structural scales. One important weakness of this method is that the scale has to be provided by the user and is fixed for the entire graph, which limits its application in practice when natural scales for different communities are unknown and heterogeneous. For example, in the global airport network shown in Figure 6.1, even communities at the same level, e.g., countries, can have different sizes.

In this work, we address this limitation of Markov Stability by considering a different random-walk based node similarity metric in the quality function, Pointwise Mutual Information (PMI) [35]. It is the similarity metric implicitly preserved by popular graph

embedding methods such as DeepWalk [8] and node2vec [9], and has been shown to lead to better node-level downstream task performance including community detection in our previous work [1]. However, the most interesting property of PMI for multiscale community detection is that it *reveals the natural scale of ground-truth communities*. Specifically, consider an instance of the Stochastic Block Model (SBM) [59] with three blocks (communities) of sizes $(10, 20, 40)$ connected by intra-block edges with probability $p = 0.9$ and inter-block edges with probability $q = 0.1$, as shown in Figure 6.4a. Figure 6.4b shows the AC scores for the three communities as a function of the scaling parameter, Markov time τ, and Figure 6.4c shows the corresponding PMI scores. Notice that AC scores monotonously decrease for all communities with increasing Markov time while PMI scores reach a unique peak at different Markov times for different communities—the Markov time at which the peak is reached corresponds to the natural scale of the community. The goal of this work is to formally establish this property for PMI based on theoretical analysis and design a multiscale community detection algorithm to take advantage of it. In the next subsection, we share our current progress in the theoretical analysis.



(a) SBM-$(10, 20, 40)$ (b) AC (c) PMI

Figure 6.4: Multiscale community detection with PMI. (a) Visualization of a Stochastic Block Model instance with three blocks (communities) of sizes $(10, 20, 40)$; (b) AC of each community monotonously decreases with increasing Markov time; (c) PMI of communities with different sizes reach unique peaks at different Markov times, revealing their natural scales.

6.2.2 Theoretical Analysis

Definitions

Let $G = (V, E)$ be an undirected weighted graph, where V denotes the set of n nodes and E denotes the set of edges, and $A \in \mathbb{R}^{n \times n}$ and $\mathbf{d}/D$ are the weighted adjacency matrix and degree vector/matrix, respectively. The Laplacian matrix L_{rw} and stationary distribution π/Π of the standard random-walk on the graph are

$$L_{rw} = I - D^{-1}A \tag{6.6}$$

$$\pi = \mathbf{d}/\operatorname{vol}(G), \Pi = D/\operatorname{vol}(G) \tag{6.7}$$

where $\operatorname{vol}(G) = \sum_{i=1}^{n} \mathbf{d}_i$ is the volume of the graph G.

AC [28] and PMI [35] are node similarity metrics that provide linear and non-linear views of random-walk based proximity, governed by a scaling parameter τ, which is also called Markov time. The AC and PMI similarity matrices are defined as

$$R_{AC}(\tau) = \Pi \exp(-\tau L_{rw}) - \pi\pi^{\top} \tag{6.8}$$

$$R_{PMI}(\tau) = \log(\Pi \exp(-\tau L_{rw})) - \log(\pi\pi^{\top}) \tag{6.9}$$

where exp() is the matrix exponential and log() is the element-wise logarithm.

To construct community quality functions based on these similarity matrices, consider a community indicator vector $\mathbf{h} \in \mathbb{R}^n$, where $\mathbf{h}_i = 1$ if node i belongs to community h and $\mathbf{h}_i = 0$, otherwise. Then the quality of community $\mathbf{h}$ is defined as the clustered node

similarity metric $R(\tau)$ (either $R_{AC}(\tau)$ or $R_{PMI}(\tau)$):

$$r(\mathbf{h};\tau) = \mathbf{h}^\top R(\tau)\mathbf{h} \tag{6.10}$$

The Markov stability algorithm [28] finds the best graph partition of c communities $H = [\mathbf{h_1}, \cdots, \mathbf{h_c}]$ that maximizes the quality of all communities in the partition at a given scale τ,

$$r(H;\tau) = \mathrm{tr}(H^\top R(\tau)H) = \sum_{j=1}^{c} \mathbf{h_j}^\top R(\tau)\mathbf{h_j} \tag{6.11}$$

Note that tuning the scale τ simultaneously changes the quality for each community in the partition. Instead, we argue that with the scale-revealing property of PMI, we can find the best partition that maximizes the quality of all communities computed at their natural scales τ_j. That is, we want to maximize

$$r(\mathbf{h_1}, \cdots, \mathbf{h_c}; \tau_1, \cdots, \tau_c) = \sum_{j=1}^{c} \mathbf{h_j}^\top R(\tau_j)\mathbf{h_j} \tag{6.12}$$

with respect to both $\mathbf{h_1}, \cdots, \mathbf{h_c}$ and $\tau_1, \cdots, \tau_c$.

Monotonicity analysis for AC

We first show that the community quality function based on AC cannot reveal the natural scale since it is monotonously decreasing with Markov time τ. Formally, we have the following theorem.

Theorem 6.1 (Monotonicity of AC) *For any community* $\mathbf{h}$*, the quality function based on AC,* $r_{AC}(\mathbf{h};\tau) = \mathbf{h}^\top R_{AC}(\tau)\mathbf{h}$*, is monotonously nonincreasing with* τ*.*

Proof: The derivative of $r_{AC}(\tau)$ with respect to τ is

$$\begin{aligned} r'_{AC}(\mathbf{h},\tau) &= \mathbf{h}^\top R'_{AC}(\tau)\mathbf{h} \\ &= -\mathbf{h}^\top \Pi L_{rw} \exp(-\tau L_{rw})\mathbf{h} \end{aligned} \tag{6.13}$$

To evaluate the matrix exponential $\exp(-\tau L_{rw})$, consider the symmetric Laplacian matrix $L_{sym} = D^{-1/2} L D^{-1/2}$, which is similar to L_{rw}:

$$L_{rw} = D^{-1/2} L_{sym} D^{1/2} \tag{6.14}$$

Let the eigendecomposition of $L_{sym} = U\Lambda U^\top$, we have:

$$\begin{aligned} \exp(-L_{rw}\tau) &= \exp(-D^{-1/2}U\Lambda U^\top D^{1/2}\tau) \\ &= \sum_{k=0}^{\infty} \frac{1}{k!}(-\tau)^k (D^{-1/2}U\Lambda U^\top D^{1/2})^k \\ &= \sum_{k=0}^{\infty} \frac{1}{k!}(-\tau)^k (D^{-1/2}U\Lambda^k U^\top D^{1/2}) \\ &= D^{-1/2}U(\sum_{k=0}^{\infty} \frac{1}{k!}(-\tau)^k \Lambda^k) U^\top D^{1/2} \\ &= D^{-1/2}U\exp(-\Lambda\tau)U^\top D^{1/2} \end{aligned} \tag{6.15}$$

Substituting this into Equation 6.13, we have

$$\begin{aligned} r'_{AC}(\mathbf{h},\tau) &= -\mathbf{h}^\top \Pi L_{rw} \exp(-\tau L_{rw})\mathbf{h} \\ &= -\mathbf{h}^\top D(D^{-1/2}U\Lambda U^\top D^{1/2})(D^{-1/2}U\exp(-\Lambda\tau)U^\top D^{1/2})\mathbf{h}/\operatorname{vol}(G) \\ &= -\mathbf{h}^\top D^{1/2}U(\Lambda\exp(-\tau\Lambda)U^\top D^{1/2}\mathbf{h}/\operatorname{vol}(G) \end{aligned} \tag{6.16}$$

Since L_{sym} is positive semidefinite with nonnegative eigenvalues, we can write $\Lambda = (\Lambda^{1/2})^2$ and let $S = D^{1/2}U\Lambda^{1/2}\exp(-\tau\Lambda/2)$, then

$$\begin{aligned} r'_{AC}(\mathbf{h},\tau) &= -\mathbf{h}^\top D^{1/2}U(\Lambda\exp(-\tau\Lambda)U^\top D^{1/2}\mathbf{h}/\operatorname{vol}(G) \\ &= -\mathbf{h}^\top SS^\top\mathbf{h}/\operatorname{vol}(G) = -\|S^\top\mathbf{h}\|^2/\operatorname{vol}(G) \leq 0 \end{aligned} \tag{6.17}$$

which completes the proof. ■

Monotonicity analysis for PMI

Analyzing the monotonicity of the community quality function based on PMI is more complicated than that of AC, due to the non-linearity introduced by the pointwise logarithm function. Our goal is to prove the following hypothesis that supports the scaling-revealing property of PMI:

Hypothesis 6.1 (Scaling-revealing property of PMI) *For any community* $\mathbf{h}$ *that is not a single node or an entire connected component, the quality function based on PMI,* $r_{PMI}(\mathbf{h};\tau) = \mathbf{h}^\top R_{PMI}(\tau)\mathbf{h}$*, has a single maximizer* $\tau^* \geq 0$*.*

The proof will start with computing the derivative of the quality function:

$$\begin{aligned} r'_{PMI}(\mathbf{h},\tau) &= \mathbf{h}^\top R'_{PMI}(\tau)\mathbf{h} \\ &= -\mathbf{h}^\top \frac{L_{rw}\exp(-\tau L_{rw})}{\exp(-\tau L_{rw})}\mathbf{h} \end{aligned} \tag{6.18}$$

Here, the division represents the elementwise division of the two matrices, which prohibits us from deriving a similar proof as the one for AC. Again, to evaluate the matrix exponential, we consider the symmetric Laplacian matrix and its eigendecomposition:

$L_{sym} = U\Lambda U^\top = \sum_{i=1}^n \lambda_i \mathbf{u_i}\mathbf{u_i}^\top$, then

$$\begin{aligned} r'_{PMI}(\mathbf{h},\tau) &= -\mathbf{h}^\top \frac{L_{rw}\exp(-\tau L_{rw})}{\exp(-\tau L_{rw})}\mathbf{h} \\ &= -\mathbf{h}^\top \frac{D^{-1/2}(\sum_i^n \lambda_i e^{-\lambda_i\tau}\mathbf{u_i}\mathbf{u_i}^\top)D^{1/2}}{D^{-1/2}(\sum_i^n e^{-\lambda_i\tau}\mathbf{u_i}\mathbf{u_i}^\top)D^{1/2}}\mathbf{h} \\ &= -\mathbf{h}^\top \frac{\sum_i^n \lambda_i e^{-\lambda_i\tau}\mathbf{u_i}\mathbf{u_i}^\top}{\sum_i^n e^{-\lambda_i\tau}\mathbf{u_i}\mathbf{u_i}^\top}\mathbf{h} \end{aligned} \tag{6.19}$$

Further reduction of the above equation requires us to fully characterize the spectrum of L_{sym}, which is non-trivial. So far, we have been able to analyze the following special cases, with the general proof of Hypothesis 6.1 left for future work:

- The complete graph.
  - $\mathbf{d} = (n-1)\mathbf{1} \triangleq d\mathbf{1}$, and

$$L_{sym} = \begin{cases} 1, i = j \\ -1/(n-1), i \neq j \end{cases}$$

  - $\lambda_1 = 0$, $\mathbf{u_1} = \mathbf{1}/\sqrt{n}$; $\lambda_2 = \cdots = \lambda_n = n/(n-1) \triangleq \lambda$, $\sum_{i=2}^n \mathbf{u_i}\mathbf{u_i}^\top = I - E_n/n$, where E_n is the all-one matrix of size n.
  - The PMI matrix and its derivative:

$$R_{PMI}(\tau) = \begin{cases} \log(1+(n-1)e^{-\lambda\tau}), i = j \\ \log(1-e^{-\lambda\tau}), i \neq j \end{cases}$$

$$R'_{PMI}(\tau) = \begin{cases} -\dfrac{ne^{-\lambda\tau}}{1+(n-1)e^{-\lambda\tau}}, i = j \\ \dfrac{ne^{-\lambda\tau}}{(n-1)(1-e^{-\lambda\tau})}, i \neq j \end{cases}$$

– For any size $1 < k < n$ community $\mathbf{h_{(k)}}$,

$$r'_{PMI}(\mathbf{h_{(k)}};\tau) = \frac{n^2 e^{-\lambda\tau}((n-1)ke^{-\lambda\tau} - n + k)}{(n-1)(1+(n-1)e^{-\lambda\tau})(1-e^{-\lambda\tau})}$$

which has a unique solution $\tau^* = \frac{n-1}{n}\log\frac{(n-1)k}{n-k}$. This means $r_{PMI}(\mathbf{h_{(k)}};\tau)$ has a single maximizer.

– The best partition assigns each node as a singleton community.

- Disconnected cliques.

  – The symmetric Laplacian matrix is a block diagonal matrix of individual symmetric Laplacian matrix for each clique, and the spectral analysis is similar to the complete graph case.

  – Any community that includes a node pair belonging to different cliques has a PMI value of $-\infty$. Any community that is a subset of a clique but not a single node has a single maximizer for its PMI value.

  – The best partition assigns each node to the clique it belongs to.

- Weakly connected cliques: c weakly connected cliques of size n each with inter-clique edge weight $\epsilon < 1$.

  – $\mathbf{d} = (n - 1 + (c-1)n\epsilon)\mathbf{1} \triangleq d\mathbf{1}$ and

$$(L_{sym})_{ij} = \begin{cases} 1, i = j \\ -\frac{1}{d}, i \neq j \text{ and } i,j \in C_s \\ -\frac{\epsilon}{d}, i \in C_s, j \in C_k \end{cases}$$

  – $\lambda_1 = 0$, $\mathbf{u_1} = \mathbf{1}/\sqrt{cn}$; $\lambda_2 = \cdots = \lambda_c = cn\epsilon/d \triangleq \lambda_\alpha$, $\sum_{i=2}^{c} \mathbf{u_i}\mathbf{u_i}^\top = (I_c -$

$E_c/c) \otimes E_n/n$, where $\otimes$ stands for the Kronecker product; $\lambda_{c+1} = \cdots = \lambda_{cn} = (n + (c-1)n\epsilon)/d \triangleq \lambda_\beta$, $\sum_{i=c+1}^{cn} \mathbf{u_i}\mathbf{u_i}^\top = I_{cn} - I_c \otimes E_n/n$.

- The PMI matrix and its derivative:

$$R_{PMI}(\tau) = \begin{cases} \log\left(1 + (c-1)e^{-\lambda_\alpha \tau} + c(n-1)e^{-\lambda_\beta \tau}\right), i = j \\ \log\left(1 + (c-1)e^{-\lambda_\alpha \tau} - ce^{-\lambda_\beta \tau}\right), i \neq j \text{ and } i, j \in C_s \\ \log\left(1 - e^{-\lambda_\alpha \tau}\right), i \in C_s, j \in C_k \end{cases}$$

$$R'_{PMI}(\tau) = \begin{cases} \dfrac{-(c-1)\lambda_\alpha e^{-\lambda_\alpha \tau} - c(n-1)\lambda_\beta e^{-\lambda_\beta \tau}}{1 + (c-1)e^{-\lambda_\alpha \tau} + c(n-1)e^{-\lambda_\beta \tau}}, i = j \\ \dfrac{-(c-1)\lambda_\alpha e^{-\lambda_\alpha \tau} + c\lambda_\beta e^{-\lambda_\beta \tau}}{1 + (c-1)e^{-\lambda_\alpha \tau} - ce^{-\lambda_\beta \tau}}, i \neq j \text{ and } i, j \in C_s \\ \dfrac{\lambda_\alpha e^{-\lambda_\alpha \tau}}{1 - e^{-\lambda_\alpha \tau}}, i \in C_s, j \in C_k \end{cases}$$

- However, based on the PMI and its derivative above, it is hard to analyze the optima and roots of the clustered PMI and its derivative.

- Two-level weakly connected cliques: Each micro community is a clique of size n. Every c_1 micro communities form a macro community with edges of weight ϵ_1 connecting inter-micro pairs. There are c_2 such macro communities in total in the graph, and inter-macro pairs are connected by edges weight ϵ_2.

  - $\mathbf{d} = (n - 1 + (c_1 - 1)n\epsilon_1 + (c_2 - 1)c_1 n\epsilon_2)\mathbf{1} \triangleq d\mathbf{1}$ and

$$(L_{sym})_{ij} = \begin{cases} 1, i = j \\ -\dfrac{1}{d}, i, j \text{ intra-micro} \\ -\dfrac{\epsilon_1}{d}, i, j \text{ intra-macro} \\ -\dfrac{\epsilon_2}{d}, i, j \text{ inter-macro} \end{cases}$$

– $\lambda_1 = 0$, $\mathbf{u_1} = \mathbf{1}/\sqrt{c_2 c_1 n}$; $\lambda_2 = \cdots = \lambda_{c_2} = c_2 c_1 n\epsilon_2/d \triangleq \lambda_\alpha$, $\sum_{i=2}^{c_2} \mathbf{u_i}\mathbf{u_i}^\top = I_{c_2} \otimes E_{c_1 n}/(c_1 n) - E_{c_2 c_1 n}/(c_2 c_1 n)$; $\lambda_{c_2+1} = \cdots = \lambda_{c_2 c_1} = (c_1 n\epsilon_1 + (c_2-1)c_1 n\epsilon_2)/d \triangleq \lambda_\beta$, $\sum_{i=c_2+1}^{c_1 c_2} \mathbf{u_i}\mathbf{u_i}^\top = I_{c_2} \otimes (I_{c_1}/n - E_{c_1}/(c_1 n)) \otimes E_n$; $\lambda_{c_2 c_1+1} = \cdots = \lambda_{c_2 c_1 n} = (n + (c_1-1)n\epsilon_1 + (c_2-1)c_1 n\epsilon_2)/d \triangleq \lambda_\gamma$, $\sum_{i=c_1 c_2+1}^{c_1 c_2 n} \mathbf{u_i}\mathbf{u_i}^\top = I_{c_2 c_1 n} - I_{c_2 c_1} \otimes E_n/n$.

– The PMI matrix and its derivative:

$$R_{PMI}(\tau) = \begin{cases} \log\left(1 + (c_2-1)e^{-\lambda_\alpha \tau} + c_2(c_1-1)e^{-\lambda_\beta \tau} + c_2 c_1 (n-1)e^{-\lambda_\gamma \tau}\right), i = j \\ \log\left(1 + (c_2-1)e^{-\lambda_\alpha \tau} + c_2(c_1-1)e^{-\lambda_\beta \tau} - c_2 c_1 e^{-\lambda_\gamma \tau}\right), i, j \text{ intra-micro} \\ \log\left(1 + (c_2-1)e^{-\lambda_\alpha \tau} - c_2 e^{-\lambda_\beta \tau}\right), i, j \text{ intra-macro} \\ \log\left(1 - e^{-\lambda_\alpha \tau}\right), i, j \text{ inter-macro} \end{cases}$$

$$R'_{PMI}(\tau) = \begin{cases} -\dfrac{(c_2-1)\lambda_\alpha e^{-\lambda_\alpha \tau} + c_2(c_1-1)\lambda_\beta e^{-\lambda_\beta \tau} + c_2 c_1 (n-1)\lambda_\gamma e^{-\lambda_\gamma \tau}}{1 + (c_2-1)e^{-\lambda_\alpha \tau} + c_2(c_1-1)e^{-\lambda_\beta \tau} + c_2 c_1 (n-1)e^{-\lambda_\gamma \tau}}, i = j \\ -\dfrac{(c_2-1)\lambda_\alpha e^{-\lambda_\alpha \tau} + c_2(c_1-1)\lambda_\beta e^{-\lambda_\beta \tau} - c_2 c_1 \lambda_\gamma e^{-\lambda_\gamma \tau}}{1 + (c_2-1)e^{-\lambda_\alpha \tau} + c_2(c_1-1)e^{-\lambda_\beta \tau} - c_2 c_1 e^{-\lambda_\gamma \tau}}, i, j \text{ intra-micro} \\ -\dfrac{(c_2-1)\lambda_\alpha e^{-\lambda_\alpha \tau} - c_2 \lambda_\beta e^{-\lambda_\beta \tau}}{1 + (c_2-1)e^{-\lambda_\alpha \tau} - c_2 e^{-\lambda_\beta \tau}}, i, j \text{ intra-macro} \\ \dfrac{\lambda_\alpha e^{-\lambda_\alpha \tau}}{1 - e^{-\lambda_\alpha \tau}}, i, j \text{ inter-macro} \end{cases}$$

– However, similar to the weakly connected cliques case, it is hard to analyze the optima and roots of the clustered PMI and its derivative based on the PMI and its derivative above.

6.3 Multiscale Anomaly Detection

6.3.1 Overview

Anomaly detection consists of uncovering noteworthy instances that significantly deviate from normal patterns or behaviors in datasets [267, 268]. It has a wide range of applications such as surfacing fraud in financial systems [269, 270, 271], detecting intru-

sion in computer networks [272, 273, 274], identifying fake news in online social media [275, 276, 277], to name a few.

In the context of graphs, anomalies are defined as nodes or subgraphs that exhibit distinctive attribute and/or topological patterns from a broader context (e.g., a local community or the entire graph). Most existing approaches focus on detecting node-level anomalies within a given [267, 278, 279, 280, 281, 282, 283, 284, 285] or across multiscale contexts [286, 287], while few recent works [288, 289] also seek to detect anomalies at a subgraph level. The goal of this work is to detect anomalies at different structural scales, including individual anomalous nodes and cohesive anomalous communities of different sizes.

Our work is built upon two lines of research for graph anomaly detection. First, we focus on anomaly detection in the unsupervised learning setting since anomaly labels in most real-world settings are unavailable. This motivates us to take advantage of the graph autoencoder paradigm [160], which attempts to reconstruct the graph topology with node embeddings generated by Graph Neural Networks (GNNs) and assumes that anomalies have poor reconstruction results (i.e., large reconstruction errors) [278, 280, 281, 279]. Second, to detect anomalies at multiple structural scales, we extend the spectral analysis of graph anomalies in [285] and find that anomalies at different structural scales demonstrate distinctive spectral energy distribution patterns. In particular, Figure 6.5 shows the spectral energy distribution of normal nodes, node-level anomalies, and three subgraph-level anomalies at different scales for the Cora dataset [188] with injected anomalies. As pointed out in [285], the spectral energy distributions of anomalous elements concentrate more on high frequencies and less on low frequencies (i.e. "right shift"). Our new observation is that the scales of anomalies affect the level of the right shift, with spectral energy distributions of smaller-scale anomalies concentrated on higher-frequency regions of the spectrum compared to larger-scale anomalies. Specifically in Figure 6.5,

node-level anomalies dominate the highest frequency band ($\lambda = 1.4$), followed by 1-scale anomalies ($\lambda = 1.2$), 2-scale anomalies ($\lambda = [0.8, 1.0]$), and 3-scale anomalies ($\lambda = 0.6$), while normal nodes dominate the low-frequency band ($\lambda = 0$). This motivates us to leverage spectral localized GNNs to learn to detect anomalies at multiple scales.

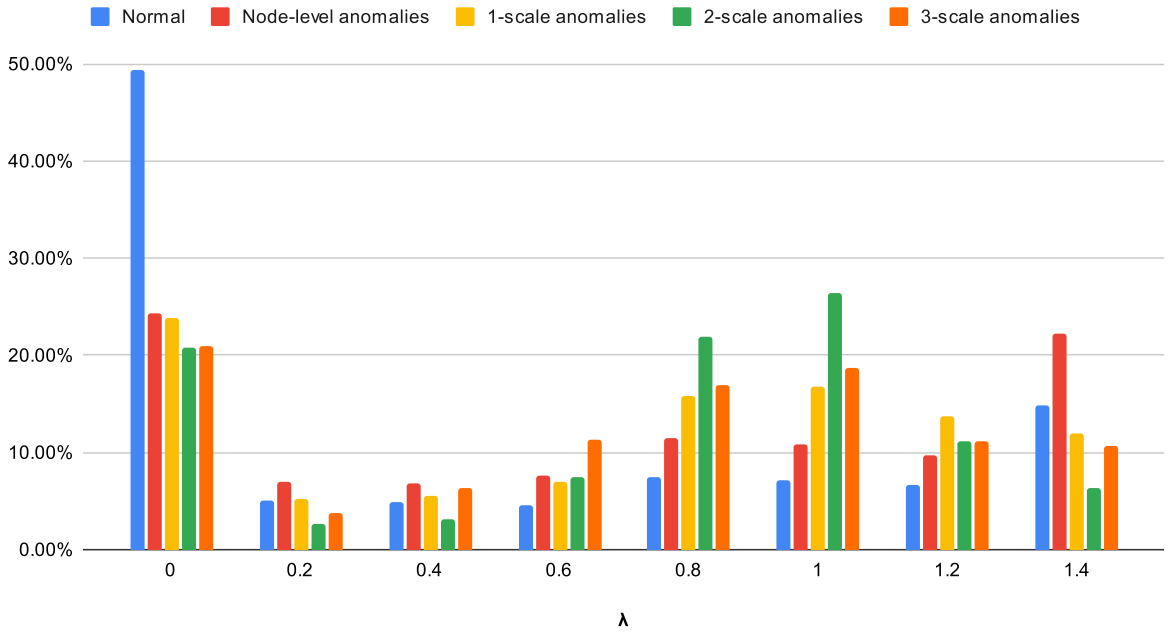


Figure 6.5: Spectral energy distributions of normal nodes, node-level anomalies, and three subgraph-level anomalies at different scales for CORA with injected anomalies. Compared to normal nodes, the spectral energy distributions of anomalous elements concentrate more on the high-frequency regions. Further, the smaller the scale of the anomalies, the higher the frequency bands they dominate.

In the next subsection, we briefly introduce our current model architecture based on the graph autoencoder paradigm and spectral localized Beta Wavelet GNNs.

6.3.2 Model Design

Let $G = (V, E, X)$ be an attributed graph, where V is the set of n nodes, E is the set of edges, and $X \in \mathbb{R}^{n \times d}$ is the node attribute matrix. Let $A \in \mathbb{R}^{n \times n}$ be the adjacency

matrix of G, D be the degree matrix of G, and $L = I - D^{-1/2}AD^{-1/2}$ be the corresponding normalized Laplacian matrix.

Graph Autoencoders

Given A and X, the encoder part of the graph autoencoder paradigm generates a node embedding matrix $Z \in \mathbb{R}^{n \times k}$ based on GNNs:

$$Z = \mathrm{ENC}(A, X) \tag{6.20}$$

Then, the node embedding matrix Z is mapped by the decoder part (e.g., inner product) to reconstruct the graph structure, represented by a predefined function of the adjacency matrix $f(A)$:

$$\hat{f}(A) = \mathrm{DEC}(Z) \tag{6.21}$$

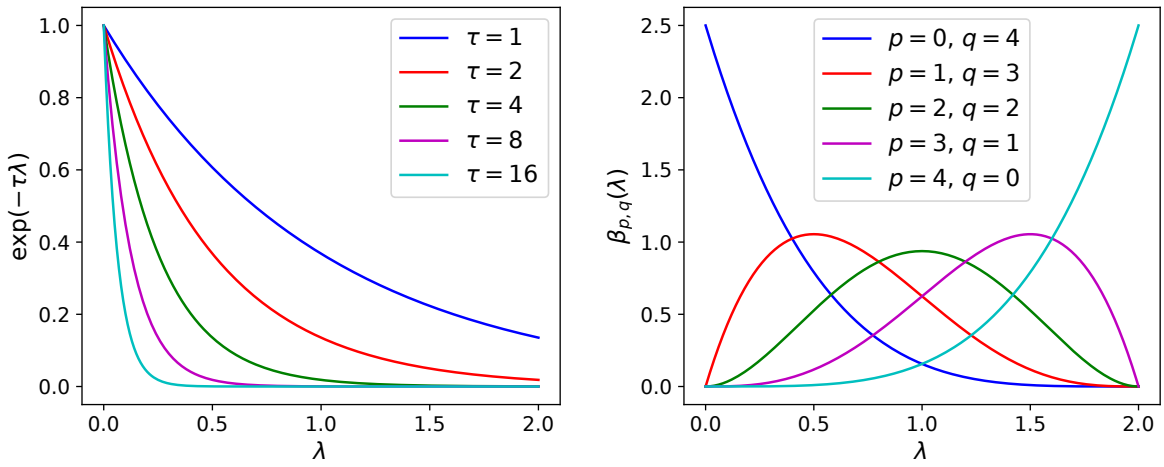
The reconstruction error $\ell(\hat{f}(A) \parallel f(A))$ is used for both training the encoder $\mathrm{ENC}(\cdot)$ and computing the anomaly scores for different graph elements.

Beta Wavelet GNNs

To attend to the distinctive spectral patterns of multiscale anomalies, we leverage Beta Wavelet GNNs [285] to construct our encoders. Specifically, the Beta wavelet kernel characterized by hyperparameters $p, q > 0$ is

$$W_{p,q} = \beta_{p,q}(L) = \frac{(L/2)^p (I - L/2)^q}{2B(p+1, q+1)} \tag{6.22}$$

where $B(p+1, q+1) = p!q!/(p+q+1)!$ is a constant. Compared to heat kernels that are more popular in GNNs, Beta kernels contain different band-pass filters that facilitate the detection of anomalies at different scales (see Figure 6.6).



(a) Heat kernels

(b) Beta kernels

Figure 6.6: Spectral property comparison between heat kernels and Beta kernels. Beta kernels contain different band-pass filters that facilitate multiscale anomaly detection.

For each scale s, we first apply a feature transformation Multi-Layer Perceptron (MLP) to the node attributes, then build $p + q = \tau$ number of Beta wavelet filters to generate τ number of embeddings:

$$Z_{p,q}^{(s)} = W_{p,q}\text{MLP}^{(s)}(X) \tag{6.23}$$

The output node embedding $Z^{(s)}$ is a weighted sum of the embeddings based on different filters, parameterized by $\{\alpha_{p,q}^{(s)}\}$:

$$Z^{(s)} = \sum_{p+q=\tau} \alpha_{p,q}^{(s)} Z_{p,q}^{(s)} \tag{6.24}$$

Finally, the parameters are learned with a scale-specific graph reconstruction loss, $\ell(\hat{f}^{(s)}(A) \parallel f^{(s)}(A))$. Choices of the graph structure function $f^{(s)}(A)$ can be, for example, $f^{(s)}(A) = A^s$, or any other multiscale node similarity metrics, such as Autocovariance

[28, 1] or Pointwise Mutual Information [35, 11]. Our next steps for this ongoing project are experimenting with these different loss functions and comparing our method with other node and subgraph-level anomaly detection approaches.

Chapter 7

Conclusions

This dissertation is devoted to bridging information-rich complex systems modeled as graphs and real-world data science applications of interest via graph representation learning. In particular, we have investigated multiscale graphs where communities appear at different structural scales (Chapters 2 and 6), signed graphs with edge signs representing positive or negative interactions (Chapter 3), and attributed graphs with descriptive node features (Chapters 4 and 5 and Sections 6.1 and 6.3), and covered applications spanning node classification (Chapter 2 and Section 6.1), link prediction (Chapters 2-4), community detection (Chapter 2 and Section 6.2), measuring polarization (Chapter 3), counterfactual explanation (Chapter 5), and anomaly detection (Section 6.3). Our approaches are built upon the two popular graph representation learning paradigms—graph embedding (Chapters 2-4 and Section 6.2) and Graph Neural Networks (Chapters 4 and 5 and Sections 6.1 and 6.3)—and also leverage other relevant tools including random-walk dynamics (Chapters 2-6), spectral graph theory (Chapters 2 and 3 and Section 6.2), graph structure learning (Chapters 4 and 5), combinatorial optimization (Chapter 5 and Section 6.2), and social theories (Chapter 3).

Through our research, we have demonstrated the importance of accounting for the

interplay between the rich graph information and downstream task properties for graph representation learning models to enable state-of-the-art performance. Specifically, we have shown that successful models:

- Capture *heterogeneous degree distributions* for link prediction in scale-free graphs (Chapter 2),
- Preserve *polarized similarity consistency* for signed link prediction in polarized graphs (Chapter 3),
- Adopt *topology-centric mechanism* to combine structural and attribute information for link prediction in attributed graphs (Chapter 4),
- Focus on *representativeness and diversity* for global counterfactual explanation in collections of molecule graphs (Chapter 5), and
- Leverage *adaptive neighborhood aggregation* for semi-supervised node classification (Section 6.1), *scale-revealing property* for community detection (Section 6.2), and *spectral energy distribution patterns* for anomaly detection (Section 6.3) in multi-scale graphs.

We hope that this dissertation will not only deepen our understanding of graph representation learning approaches but also benefit researchers and practitioners alike from other domains related to data science in their applications of machine learning on graphs.

Bibliography

[1] Z. Huang, A. Silva, and A. Singh, *A broader picture of random-walk based graph embedding*, in *SIGKDD*, 2021.

[2] Z. Huang, A. Silva, and A. Singh, *Pole: Polarized embedding for signed networks*, in *WSDM*, 2022.

[3] Z. Huang, M. Kosan, A. Silva, and A. Singh, *Link prediction without graph neural networks*, 2023.

[4] Z. Huang, M. Kosan, S. Medya, S. Ranu, and A. Singh, *Global counterfactual explainer for graph neural networks*, in *WSDM*, 2023.

[5] W. Ye, Z. Huang, Y. Hong, and A. Singh, *Graph neural diffusion networks for semi-supervised learning*, *arXiv preprint arXiv:2201.09698* (2022).

[6] Z. Huang, M. Kondapaneni, A. Silva, and A. Singh, *Multiscale community detection with pointwise mutual information*, 2023.

[7] M. Arriola, M. Kosan, Z. Huang, S. Sharma, and A. Singh, *Multiscale anomaly detection with graph autoencoders*, 2023.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, *Deepwalk: Online learning of social representations*, in *SIGKDD*, 2014.

[9] A. Grover and J. Leskovec, *node2vec: Scalable feature learning for networks*, in *SIGKDD*, 2016.

[10] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, *Asymmetric transitivity preserving graph embedding*, in *SIGKDD*, 2016.

[11] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, *Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec*, in *WSDM*, 2018.

[12] M. T. Schaub, J.-C. Delvenne, R. Lambiotte, and M. Barahona, *Multiscale dynamical embeddings of complex networks*, *Physical Review E* **99** (2019), no. 6 062308.

[13] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, *Line: Large-scale information network embedding*, in *WebConf*, pp. 1067–1077, 2015.

[14] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, *Netsmf: Large-scale network embedding as sparse matrix factorization*, in *WebConf*, 2019.

[15] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, *Scalable graph embedding for asymmetric proximity*, in *AAAI*, 2017.

[16] M. Khosla, J. Leonhardt, W. Nejdl, and A. Anand, *Node representation learning for directed graphs*, in *ECML-PKDD*, 2019.

[17] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, *Verse: Versatile graph embeddings from similarity measures*, in *WebConf*, 2018.

[18] S. Cao, W. Lu, and Q. Xu, *Grarep: Learning graph representations with global structural information*, in *CIKM*, 2015.

[19] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, *Harp: Hierarchical representation learning for networks*, *arXiv:1706.07845* (2017).

[20] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, *Learning structural node embeddings via diffusion wavelets*, in *SIGKDD*, 2018.

[21] Z. Xin, J. Chen, G. Chen, and S. Zhao, *Marc: Multi-granular representation learning for networks based on the 3-clique*, *IEEE Access* **7** (2019) 141715–141727.

[22] S. Chanpuriya and C. Musco, *Infinitewalk: Deep network embeddings as laplacian embeddings with a nonlinearity*, in *SIGKDD*, 2020.

[23] W. L. Hamilton, R. Ying, and J. Leskovec, *Representation learning on graphs: Methods and applications*, *arXiv preprint arXiv:1709.05584* (2017).

[24] P. Cui, X. Wang, J. Pei, and W. Zhu, *A survey on network embedding*, *IEEE TKDE* **31** (2018), no. 5 833–852.

[25] H. Cai, V. W. Zheng, and K. C.-C. Chang, *A comprehensive survey of graph embedding: Problems, techniques, and applications*, *IEEE TKDE* **30** (2018), no. 9 1616–1637.

[26] P. Goyal and E. Ferrara, *Graph embedding techniques, applications, and performance: A survey*, *Knowledge-Based Systems* **151** (2018) 78–94.

[27] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, *Machine learning on graphs: A model and comprehensive taxonomy*, *arXiv preprint arXiv:2005.03675* (2020).

[28] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, *Stability of graph communities across time scales*, *PNAS* **107** (2010), no. 29 12755–12760.

[29] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, *Signed network embedding in social media*, in *SDM*, 2017.

[30] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, *Continuous-time dynamic network embeddings*, in *WebConf*, 2018.

[31] A. Javari, T. Derr, P. Esmailian, J. Tang, and K. C.-C. Chang, *Rose: Role-based signed network embedding*, in *WebConf*, 2020.

[32] D. Wang, P. Cui, and W. Zhu, *Structural deep network embedding*, in *SIGKDD*, 2016.

[33] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, *Arbitrary-order proximity preserved network embedding*, in *SIGKDD*, 2018.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in *NeurIPS*, 2013.

[35] O. Levy and Y. Goldberg, *Neural word embedding as implicit matrix factorization*, in *NeurIPS*, 2014.

[36] J.-C. Delvenne, M. T. Schaub, S. N. Yaliraki, and M. Barahona, *The stability of a graph partition: A dynamics-based framework for community detection*, in *Dynamics On and Of Complex Networks, Volume 2*, pp. 221–242. Springer, 2013.

[37] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, *Don't walk, skip! online learning of multi-scale network embeddings*, in *ASONAM*, 2017.

[38] A.-L. Barabási and E. Bonabeau, *Scale-free networks*, *Scientific american* **288** (2003), no. 5 60–69.

[39] B. Srinivasan and B. Ribeiro, *On the equivalence between positional node embeddings and structural graph representations*, in *ICLR*, 2019.

[40] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: Bringing order to the web.*, tech. rep., Stanford InfoLab, 1999.

[41] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, *Psychometrika* **1** (1936), no. 3 211–218.

[42] F. R. Chung and F. C. Graham, *Spectral graph theory*. No. 92. American Mathematical Soc., 1997.

[43] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[44] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, *SIAM review* **53** (2011), no. 2 217–288.

[45] L. Tang and H. Liu, *Relational learning via latent social dimensions*, in *SIGKDD*, 2009.

[46] J. Patokallio, "Openflights.org: Flight logging, mapping, stats and sharing." `https://openflights.org/data.html`, 2020.

[47] M. Mahoney, "Large text compression benchmark." `https://www.mattmahoney.net/dc/textdata`, 2011.

[48] L. A. Adamic and N. Glance, *The political blogosphere and the 2004 us election: divided they blog*, in *Workshop on Link discovery*, 2005.

[49] L. Šubelj and M. Bajec, *Model of complex networks based on citation dynamics*, in *WebConf*, 2013.

[50] N. Aspert, V. Miz, B. Ricaud, and P. Vandergheynst, *A graph-structured dataset for wikipedia research*, in *WebConf*, 2019.

[51] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, *Liblinear: A library for large linear classification*, *JMLR* **9** (2008), no. Aug 1871–1874.

[52] L. Tang, S. Rajan, and V. K. Narayanan, *Large scale multi-label classification via metalabeler*, in *WebConf*, 2009.

[53] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Mining multi-label data*, in *Data mining and knowledge discovery handbook*, pp. 667–685. Springer, 2009.

[54] L. Lü and T. Zhou, *Link prediction in complex networks: A survey*, *Physica A: statistical mechanics and its applications* **390** (2011), no. 6 1150–1170.

[55] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, tech. rep., Stanford, 2006.

[56] A. Strehl and J. Ghosh, *Cluster ensembles—a knowledge reuse framework for combining multiple partitions*, *JMLR* **3** (2002), no. Dec 583–617.

[57] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *ICLR*, 2015.

[58] M. Girvan and M. E. Newman, *Community structure in social and biological networks*, *PNAS* **99** (2002), no. 12 7821–7826.

[59] B. Karrer and M. E. Newman, *Stochastic blockmodels and community structure in networks*, *Physical review E* **83** (2011), no. 1 016107.

[60] C. Seshadhri, A. Sharma, A. Stolman, and A. Goel, *The impossibility of low-rank representations for triangle-rich complex networks*, *PNAS* **117** (2020), no. 11 5631–5637.

[61] G. E. Hinton and S. T. Roweis, *Stochastic neighbor embedding*, in *NeurIPS*, 2003.

[62] J. Bourgain, *On lipschitz embedding of finite metric spaces in hilbert space*, *Israel Journal of Mathematics* **52** (1985), no. 1-2 46–52.

[63] M. Belkin and P. Niyogi, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, in *NeurIPS*, 2002.

[64] J. Díaz, J. Petit, and M. Serna, *A survey of graph layout problems*, *ACM Computing Surveys (CSUR)* **34** (2002), no. 3 313–356.

[65] Y. Dong, N. V. Chawla, and A. Swami, *metapath2vec: Scalable representation learning for heterogeneous networks*, in *SIGKDD*, 2017.

[66] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, *Dynamic network embedding: An extended approach for skip-gram based network embedding.*, in *IJCAI*, 2018.

[67] H. Zhang, L. Qiu, L. Yi, and Y. Song, *Scalable multiplex network embedding.*, in *IJCAI*, 2018.

[68] D. Ruelle, *Thermodynamic formalism: the mathematical structure of equilibrium statistical mechanics*. Cambridge University Press, 2004.

[69] R. Lambiotte, J.-C. Delvenne, and M. Barahona, *Random walks, markov processes and the multiscale modular organization of complex networks*, *IEEE TSNE* **1** (2014), no. 2 76–90.

[70] H. Gao, J. Huang, Q. Cheng, H. Sun, B. Wang, and H. Li, *Link prediction based on linear dynamical response*, *Physica A: Statistical Mechanics and its Applications* **527** (2019) 121397.

[71] E. Ravasz and A.-L. Barabási, *Hierarchical organization in complex networks*, *Physical review E* **67** (2003), no. 2 026112.

[72] A. Clauset, C. Moore, and M. E. Newman, *Hierarchical structure and the prediction of missing links in networks*, *Nature* **453** (2008), no. 7191 98–101.

[73] M. Nickel and D. Kiela, *Poincaré embeddings for learning hierarchical representations*, in *NeurIPS*, 2017.

[74] J. A. Tucker, A. Guess, P. Barberá, C. Vaccari, A. Siegel, S. Sanovich, D. Stukal, and B. Nyhan, *Social media, political polarization, and political disinformation: A review of the scientific literature*, tech. rep., The William and Flora Hewlett Foundation, 2018.

[75] N. Gillani, A. Yuan, M. Saveski, S. Vosoughi, and D. Roy, *Me, my echo chamber, and i: introspection on social media polarization*, in *WebConf*, 2018.

[76] V. R. K. Garimella and I. Weber, *A long-term analysis of polarization on twitter*, in *ICWSM*, 2017.

[77] A. Guess, B. Nyhan, and J. Reifler, *Selective exposure to misinformation: Evidence from the consumption of fake news during the 2016 us presidential campaign*, *European Research Council* **9** (2018), no. 3 4.

[78] A. Prabhu, D. Guhathakurta, M. Subramanian, M. Reddy, S. Sehgal, T. Karandikar, A. Gulati, U. Arora, R. R. Shah, P. Kumaraguru, *et. al.*, *Capitol (pat) riots: A comparative study of twitter and parler*, *arXiv preprint arXiv:2101.06914* (2021).

[79] S. B. Naeem, R. Bhatti, and A. Khan, *An exploration of how fake news is taking over social media and putting public health at risk*, *Health Information & Libraries Journal* **38** (2021), no. 2 143–149.

[80] J. M. Otala, G. Kurtic, I. Grasso, Y. Liu, J. Matthews, and G. Madraki, *Political polarization and platform migration: A study of parler and twitter usage by united states of america congress members*, in *Fairness, Accountability, Transparency, Ethics and Society on the Web*, pp. 224–231, 2021.

[81] M. Thomas, B. Pang, and L. Lee, *Get out the vote: determining support or opposition from congressional floor-debate transcripts*, in *EMNLP*, 2006.

[82] M. Lai, V. Patti, G. Ruffo, and P. Rosso, *Stance evolution and twitter interactions in an italian political debate*, in *NLDB*, 2018.

[83] S. M. Theriault, *Party polarization in the us congress: Member replacement and member adaptation*, *Party Politics* **12** (2006), no. 4 483–503.

[84] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, F. Menczer, and A. Flammini, *Political polarization on twitter*, in *ICWSM*, 2011.

[85] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo, *Discovering polarized communities in signed networks*, in *CIKM*, 2019.

[86] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, *Physical review E* **78** (2008), no. 4 046110.

[87] R.-C. Tzeng, B. Ordozgoiti, and A. Gionis, *Discovering conflicting groups in signed networks*, in *NeurIPS*, 2020.

[88] K. Garimella, G. De Francisci Morales, A. Gionis, and M. Mathioudakis, *Reducing controversy by connecting opposing views*, in *WSDM*, 2017.

[89] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon, *Exploiting longer cycles for link prediction in signed networks*, in *CIKM*, 2011.

[90] C.-J. Hsieh, K.-Y. Chiang, and I. S. Dhillon, *Low rank modeling of signed networks*, in *SIGKDD*, 2012.

[91] J. Wang, J. Shen, P. Li, and H. Xu, *Online matrix completion for signed link prediction*, in *WSDM*, 2017.

[92] J. Kim, H. Park, J.-E. Lee, and U. Kang, *Side: representation learning in signed directed networks*, in *WebConf*, 2018.

[93] F. Heider, *Attitudes and cognitive organization*, *The Journal of psychology* **21** (1946), no. 1 107–112.

[94] J. Jung, W. Jin, L. Sael, and U. Kang, *Personalized ranking in signed networks using signed random walk with restart*, in *ICDM*, 2016.

[95] X. Yin, X. Hu, Y. Chen, X. Yuan, and B. Li, *Signed-pagerank: An efficient influence maximization framework for signed social networks*, *IEEE TKDE* **33** (2021), no. 5 2208–2222.

[96] D. McCumber, *From the house on the hill: congressman looks back at his life*, *Laredo Morning Times* (2014).

[97] C. Malone, *A q&a with the house democrat who's voted with trump 75 percent of the time*, *FiveThirtyEight* (2017).

[98] B. Pershing, *Pelosi, harman have long history*, *The Washington Post* (2009).

[99] G. Skelton, *California and the west: in the ring, with contenders for governor*, *Los Angeles Times* (1998).

[100] V. Kristof, M. Grossglauser, and P. Thiran, *War of words: The competitive dynamics of legislative processes*, in *WebConf*, 2020.

[101] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, *Edge weight prediction in weighted signed networks*, in *ICDM*, 2016.

[102] R. West, H. S. Paskov, J. Leskovec, and C. Potts, *Exploiting social network structure for person-to-person sentiment analysis*, *TACL* **2** (2014) 297–310.

[103] J. Leskovec, D. Huttenlocher, and J. Kleinberg, *Signed networks in social media*, in *SIGCHI*, 2010.

[104] G. Beigi, S. Ranganath, and H. Liu, *Signed link prediction with sparse data: The role of personality information*, in *WebConf*, 2019.

[105] P. Xu, W. Hu, J. Wu, and B. Du, *Link prediction with signed latent factors in signed social networks*, in *SIGKDD*, 2019.

[106] M. R. Islam, B. A. Prakash, and N. Ramakrishnan, *Signet: Scalable embeddings for signed networks*, in *PAKDD*, 2018.

[107] Y. Chen, T. Qian, H. Liu, and K. Sun, *"bridge" enhanced signed directed network embedding*, in *CIKM*, 2018.

[108] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak, *Spectral analysis of signed graphs for clustering, prediction and visualization*, in *SDM*, 2010.

[109] P. C. Guerra, W. Meira Jr, C. Cardie, and R. Kleinberg, *A measure of polarization on social media networks based on community boundaries*, in *ICWSM*, 2013.

[110] R. Kannan, S. Vempala, and A. Vetta, *On clusterings: Good, bad and spectral*, *JACM* **51** (2004), no. 3 497–515.

[111] M. E. Newman, *Modularity and community structure in networks*, *PNAS* **103** (2006), no. 23 8577–8582.

[112] A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter, *Party polarization in congress: a network science approach*, *arXiv preprint arXiv:0907.3509* (2011).

[113] K. Garimella, G. D. F. Morales, A. Gionis, and M. Mathioudakis, *Quantifying controversy on social media*, *TSC* **1** (2018), no. 1 1–27.

[114] V. Martínez, F. Berzal, and J.-C. Cubero, *A survey of link prediction in complex networks*, *ACM computing surveys (CSUR)* **49** (2016), no. 4 1–33.

[115] J. Leskovec, D. Huttenlocher, and J. Kleinberg, *Predicting positive and negative links in online social networks*, in *WebConf*, 2010.

[116] A. Javari and M. Jalili, *Cluster-based collaborative filtering for sign prediction in social networks with positive and negative links*, *TIST* **5** (2014), no. 2 1–19.

[117] D. Song and D. A. Meyer, *Link sign prediction and ranking in signed directed social networks*, *Social network analysis and mining* **5** (2015), no. 1 1–14.

[118] W. Yuan, K. He, D. Guan, L. Zhou, and C. Li, *Graph kernel based link prediction for signed social networks*, *Information Fusion* **46** (2019) 1–10.

[119] J. Ye, H. Cheng, Z. Zhu, and M. Chen, *Predicting positive and negative links in signed social networks by transfer learning*, in *WebConf*, 2013.

[120] G. Beigi, J. Tang, S. Wang, and H. Liu, *Exploiting emotional information for trust/distrust prediction*, in *SDM*, 2016.

[121] J. Tang, X. Hu, Y. Chang, and H. Liu, *Predictability of distrust with interaction data*, in *CIKM*, 2014.

[122] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, *Propagation of trust and distrust*, in *WebConf*, 2004.

[123] S. Yuan, X. Wu, and Y. Xiang, *Sne: signed network embedding*, in *PAKDD*, 2017.

[124] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, *Arnetminer: extraction and mining of academic social networks*, in *SIGKDD*, 2008.

[125] C. Li, J. Ma, X. Guo, and Q. Mei, *Deepcas: An end-to-end predictor of information cascades*, in *WebConf*, 2017.

[126] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, *Deepinf: Social influence prediction with deep learning*, in *SIGKDD*, 2018.

[127] M. Jamali and M. Ester, *Trustwalker: a random walk model for combining trust-based and item-based recommendation*, in *SIGKDD*, 2009.

[128] F. Monti, M. Bronstein, and X. Bresson, *Geometric matrix completion with recurrent multi-graph neural networks*, in *NeurIPS*, 2017.

[129] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, *Kgat: Knowledge graph attention network for recommendation*, in *SIGKDD*, 2019.

[130] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen, *Open domain question answering using early fusion of knowledge bases and text*, in *EMNLP*, 2018.

[131] S. K. Sahu, F. Christopoulou, M. Miwa, and S. Ananiadou, *Inter-sentence relation extraction with document-level graph convolutional neural network*, in *ACL*, 2019.

[132] L. Yao, C. Mao, and Y. Luo, *Graph convolutional networks for text classification*, in *AAAI*, 2019.

[133] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, *Graph networks as learnable physics engines for inference and control*, in *ICML*, 2018.

[134] B. Ivanovic and M. Pavone, *The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs*, in *ICCV*, 2019.

[135] A. L. da Silva, F. Kocayusufoglu, S. Jafarpour, F. Bullo, A. Swami, and A. Singh, *Combining physics and machine learning for network flow estimation*, in *ICLR*, 2020.

[136] Y. Qi, Z. Bar-Joseph, and J. Klein-Seetharaman, *Evaluation of different biological data and computational classification methods for use in protein interaction prediction*, *Proteins: Structure, Function, and Bioinformatics* **63** (2006), no. 3 490–500.

[137] D. Liben-Nowell and J. Kleinberg, *The link-prediction problem for social networks*, *Journal of the American society for information science and technology* **58** (2007), no. 7 1019–1031.

[138] Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems*, *Computer* **42** (2009), no. 8 30–37.

[139] T. Martin, B. Ball, and M. E. Newman, *Structural inference for uncertain networks*, *Physical Review E* **93** (2016), no. 1 012306.

[140] A. Bahulkar, B. K. Szymanski, N. O. Baycik, and T. C. Sharkey, *Community detection with edge augmentation in criminal networks*, in *ASONAM*, 2018.

[141] B. Wilder, E. Ewing, B. Dilkina, and M. Tambe, *End to end learning and optimization on graphs*, in *NeurIPS*, 2019.

[142] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, in *ICLR*, 2017.

[143] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in *NeurIPS*, 2017.

[144] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, in *ICLR*, 2018.

[145] J. Klicpera, A. Bojchevski, and S. Günnemann, *Predict then propagate: Graph neural networks meet personalized pagerank*, in *ICLR*, 2018.

[146] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, *Simplifying graph convolutional networks*, in *ICML*, 2019.

[147] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, *Robust graph representation learning via neural sparsification*, in *ICML*, 2020.

[148] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, *Hierarchical graph representation learning with differentiable pooling*, in *NeurIPS*, 2018.

[149] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, *An end-to-end deep learning architecture for graph classification*, in *AAAI*, 2018.

[150] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, *Weisfeiler and leman go neural: Higher-order graph neural networks*, in *AAAI*, 2019.

[151] M. Zhang and Y. Chen, *Link prediction based on graph neural networks*, in *NeurIPS*, 2018.

[152] S. Yun, S. Kim, J. Lee, J. Kang, and H. J. Kim, *Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction*, in *NeurIPS*, 2021.

[153] L. Pan, C. Shi, and I. Dokmanić, *Neural link prediction with walk pooling*, in *ICLR*, 2022.

[154] M. E. Newman, *Clustering and preferential attachment in growing networks*, *Physical review E* **64** (2001), no. 2 025102.

[155] L. A. Adamic and E. Adar, *Friends and neighbors on the web*, *Social networks* **25** (2003), no. 3 211–230.

[156] A.-L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, *Evolution of the social network of scientific collaborations*, *Physica A: Statistical mechanics and its applications* **311** (2002), no. 3-4 590–614.

[157] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, *Combining label propagation and simple models out-performs graph neural networks*, in *ICLR*, 2021.

[158] Q. Li, Z. Han, and X.-M. Wu, *Deeper insights into graph convolutional networks for semi-supervised learning*, in *AAAI*, 2018.

[159] J. Ma, B. Chang, X. Zhang, and Q. Mei, *Copulagnn: towards integrating representational and correlational roles of graphs in graph neural networks*, in *ICLR*, 2020.

[160] T. N. Kipf and M. Welling, *Variational graph auto-encoders*, *arXiv preprint arXiv:1611.07308* (2016).

[161] I. Chami, Z. Ying, C. Ré, and J. Leskovec, *Hyperbolic graph convolutional neural networks*, in *NeurIPS*, 2019.

[162] Y. Zhang, X. Wang, C. Shi, N. Liu, and G. Song, *Lorentzian graph convolutional networks*, in *WebConf*, 2021.

[163] L. Cai, J. Li, J. Wang, and S. Ji, *Line graph neural networks for link prediction*, *IEEE TPAMI* (2021).

[164] Z. Yan, T. Ma, L. Gao, Z. Tang, and C. Chen, *Link prediction with persistent homology: An interactive view*, in *ICML*, 2021.

[165] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, *Neural bellman-ford networks: A general graph neural network framework for link prediction*, in *NeurIPS*, 2021.

[166] Y. Chen, Y. R. Gel, and H. V. Poor, *Bscnets: Block simplicial complex neural networks*, in *AAAI*, 2022.

[167] J. Davis and M. Goadrich, *The relationship between precision-recall and roc curves*, in *ICML*, 2006.

[168] T. Saito and M. Rehmsmeier, *The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets*, *PloS one* **10** (2015), no. 3 e0118432.

[169] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, *Open graph benchmark: Datasets for machine learning on graphs*, in *NeurIPS*, 2020.

[170] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, *Translating embeddings for modeling multi-relational data*, in *NeurIPS*, 2013.

[171] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, *Embedding entities and relations for learning and inference in knowledge bases*, in *ICLR*, 2015.

[172] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, *Rotate: Knowledge graph embedding by relational rotation in complex space*, in *ICLR*, 2018.

[173] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.

[174] X. Chen, X. Cheng, and S. Mallat, *Unsupervised deep haar scattering on graphs*, in *NeurIPS*, 2014.

[175] K. Sohn, *Improved deep metric learning with multi-class n-pair loss objective*, in *NeurIPS*, 2016.

[176] B. McFee and G. Lanckriet, *Metric learning to rank*, in *ICML*, 2010.

[177] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, *Deep metric learning to rank*, in *CVPR*, 2019.

[178] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, *Learning with average precision: Training image retrieval with a listwise loss*, in *ICCV*, 2019.

[179] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, *Ranked list loss for deep metric learning*, in *ICCV*, 2019.

[180] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, *An efficient boosting algorithm for combining preferences*, *JMLR* **4** (2003), no. Nov 933–969.

[181] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, *Listwise approach to learning to rank: theory and algorithm*, in *ICML*, 2008.

[182] S. Bruch, *An alternative cross entropy loss for learning-to-rank*, in *WebConf*, 2021.

[183] L. Cai and W. Y. Wang, *Kbgan: Adversarial learning for knowledge graph embeddings*, in *NAACL*, 2018.

[184] P. Wang, S. Li, and R. Pan, *Incorporating gan for negative sampling in knowledge representation learning*, in *AAAI*, 2018.

[185] V. Satuluri and S. Parthasarathy, *Bayesian locality sensitive hashing for fast similarity search*, in *VLDB*, 2012.

[186] D. C. Anastasiu and G. Karypis, *L2ap: Fast cosine similarity search with prefix l-2 norm bounds*, in *ICDE*, 2014.

[187] Z. Liu, D. Lai, C. Li, and M. Wang, *Feature fusion based subgraph classification for link prediction*, in *CIKM*, 2020.

[188] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, *Automating the construction of internet portals with machine learning*, *Information Retrieval* **3** (2000), no. 2 127–163.

[189] C. L. Giles, K. D. Bollacker, and S. Lawrence, *Citeseer: An automatic citation indexing system*, in *ACM conference on Digital libraries*, 1998.

[190] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, *Collective classification in network data*, *AI magazine* **29** (2008), no. 3 93–93.

[191] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, *Image-based recommendations on styles and substitutes*, in *SIGIR*, 2015.

[192] M. Fey and J. E. Lenssen, *Fast graph representation learning with PyTorch Geometric*, in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[193] Z. Yang, W. Cohen, and R. Salakhudinov, *Revisiting semi-supervised learning with graph embeddings*, in *ICML*, 2016.

[194] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, *Pitfalls of graph neural network evaluation*, *arXiv preprint arXiv:1811.05868* (2018).

[195] T. Zhou, L. Lü, and Y.-C. Zhang, *Predicting missing links via local information*, *The European Physical Journal B* **71** (2009), no. 4 623–630.

[196] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et. al.*, *Pytorch: An imperative style, high-performance deep learning library*, in *NeurIPS*, 2019.

[197] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, *Learning to rank using gradient descent*, in *ICML*, 2005.

[198] L. Katz, *A new status index derived from sociometric analysis*, *Psychometrika* **18** (1953), no. 1 39–43.

[199] G. Jeh and J. Widom, *Simrank: a measure of structural-context similarity*, in *SIGKDD*, 2002.

[200] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, *Graphrnn: Generating realistic graphs with deep auto-regressive models*, in *ICML*, 2018.

[201] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, *Learning deep generative models of graphs*, in *ICML*, 2018.

[202] A. Grover, A. Zweig, and S. Ermon, *Graphite: Iterative generative modeling of graphs*, in *ICML*, 2019.

[203] T. Zhao, G. Liu, S. Günnemann, and M. Jiang, *Graph data augmentation for graph machine learning: A survey*, *arXiv preprint arXiv:2202.08871* (2022).

[204] L. Franceschi, M. Niepert, M. Pontil, and X. He, *Learning discrete structures for graph neural networks*, in *ICML*, 2019.

[205] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, *Data augmentation for graph neural networks*, in *AAAI*, 2021.

[206] Y. Chen, L. Wu, and M. Zaki, *Iterative deep graph learning for graph neural networks: Better and robust node embeddings*, in *NeurIPS*, 2020.

[207] Y. Yang, L. Wu, R. Hong, K. Zhang, and M. Wang, *Enhanced graph learning for collaborative filtering via mutual information maximization*, in *SIGIR*, 2021.

[208] A. Singh, Q. Huang, S. L. Huang, O. Bhalerao, H. He, S.-N. Lim, and A. R. Benson, *Edge proposal sets for link prediction*, *arXiv preprint arXiv:2106.15810* (2021).

[209] Y. Wang, B. Feng, G. Li, S. Li, L. Deng, Y. Xie, and Y. Ding, *Gnnadvisor: An efficient runtime system for gnn acceleration on gpus*, in *OSDI*, 2021.

[210] Y. Wang, B. Feng, and Y. Ding, *Qgtc: accelerating quantized graph neural networks via gpu tensor core*, in *PPoPP*, 2022.

[211] S. Nishad, S. Agarwal, A. Bhattacharya, and S. Ranu, *Graphreach: Position-aware graph neural network using reachability estimations*, in *IJCAI*, 2021.

[212] M. Jiang, Z. Li, S. Zhang, S. Wang, X. Wang, Q. Yuan, and Z. Wei, *Drug–target affinity prediction using graph neural network and contact maps*, *RSC advances* **10** (2020), no. 35 20701–20712.

[213] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. J. Jiang, E. M. Songhori, S. Wang, Y. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, *Chip placement with deep reinforcement learning*, *CoRR* **abs/2004.10746** (2020).

[214] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, *Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs*, in *NeurIPS*, 2020.

[215] R. Bhattoo, S. Ranu, and N. Krishnan, *Learning articulated rigid body dynamics with lagrangian graph neural network*, in *NeurIPS*, 2022.

[216] A. Thangamuthu, G. Kumar, S. Bishnoi, R. Bhattoo, N. M. A. Krishnan, and S. Ranu, *Unravelling the performance of physics-informed graph neural networks for dynamical systems*, in *NeurIPS*, 2022.

[217] M. Kosan, A. Silva, S. Medya, B. Uzzi, and A. Singh, *Event detection on dynamic graphs*, *arXiv preprint arXiv:2110.12148* (2021).

[218] S. Medya, M. Rasoolinejad, Y. Yang, and B. Uzzi, *An exploratory study of stock price movements from earnings calls*, in *WebConf*, 2022.

[219] S. Gupta, S. Manchanda, S. Bedathur, and S. Ranu, *TIGGER: scalable generative modelling for temporal interaction graphs*, in *AAAI*, 2022.

[220] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, *Protein function prediction via graph kernels*, *Bioinformatics* **21** (2005), no. suppl_1 i47–i56.

[221] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, *Like like alike: joint friendship and interest propagation in social networks*, in *WebConf*, 2011.

[222] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, *Gnnexplainer: Generating explanations for graph neural networks*, in *NeurIPS*, 2019.

[223] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, *Parameterized explainer for graph neural network*, in *NeurIPS*, 2020.

[224] M. Vu and M. T. Thai, *Pgm-explainer: Probabilistic graphical model explanations for graph neural networks*, in *NeurIPS*, 2020.

[225] H. Yuan, J. Tang, X. Hu, and S. Ji, *Xgnn: Towards model-level explanations of graph neural networks*, in *SIGKDD*, 2020.

[226] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, *Cf-gnnexplainer: Counterfactual explanations for graph neural networks*, in *AISTATS*, 2022.

[227] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, and Y. Zhang, *Robust counterfactual explanations on graph neural networks*, in *NeurIPS*, 2021.

[228] C. Abrate and F. Bonchi, *Counterfactual graphs for explainable classification of brain networks*, in *SIGKDD*, 2021.

[229] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, *Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning*, in *WebConf*, 2022.

[230] P. Voigt and A. Von dem Bussche, *The eu general data protection regulation (gdpr)*, *A Practical Guide, 1st Ed., Cham: Springer International Publishing* **10** (2017), no. 3152676 10–5555.

[231] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng, *Graph neural networks for automated de novo drug design*, *Drug Discovery Today* **26** (2021), no. 6 1382–1393.

[232] J. Kazius, R. McGuire, and R. Bursi, *Derivation and validation of toxicophores for mutagenicity prediction*, *Journal of medicinal chemistry* **48** (2005), no. 1 312–320.

[233] A. Sanfeliu and K.-S. Fu, *A distance measure between attributed relational graphs for pattern recognition*, *IEEE transactions on systems, man, and cybernetics* (1983), no. 3 353–362.

[234] K. Borgwardt, N. Schraudolph, and S. Vishwanathan, *Fast computation of graph kernels*, in *NeurIPS*, 2006.

[235] F. Costa and K. De Grave, *Fast neighborhood subgraph pairwise distance kernel*, in *ICML*, 2010.

[236] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, *Weisfeiler-lehman graph kernels.*, *JMLR* **12** (2011), no. 9.

[237] K. Rawal and H. Lakkaraju, *Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses*, in *NeurIPS*, 2020.

[238] Y. Liang and P. Zhao, *Similarity search in graph databases: A multi-layered indexing approach*, in *ICDE*, 2017.

[239] R. Ranjan, S. Grover, S. Medya, V. Chakravarthy, Y. Sabharwal, and S. Ranu, *Greed: A neural framework for learning graph distance functions*, in *NeurIPS*, 2022.

[240] R. Pemantle, *Vertex-reinforced random walk*, *Probability Theory and Related Fields* **92** (1992), no. 1 117–136.

[241] Z. Huang, A. Silva, and A. Singh, *A broader picture of random-walk based graph embedding*, in *SIGKDD*, 2021.

[242] Z. Huang, A. Silva, and A. Singh, *Pole: Polarized embedding for signed networks*, in *WSDM*, 2022.

[243] Q. Mei, J. Guo, and D. Radev, *Divrank: the interplay of prestige and diversity in information networks*, in *SIGKDD*, 2010.

[244] D. Natarajan and S. Ranu, *A scalable and generic framework to mine top-k representative subgraph patterns*, in *ICDM*, 2016.

[245] A. Metwally, D. Agrawal, and A. E. Abbadi, *Efficient computation of frequent and top-k elements in data streams*, in *ICDT*, 2005.

[246] N. Wale and G. Karypis, *Comparison of descriptor spaces for chemical compound retrieval and classification*, in *ICDM*, 2006.

[247] K. Riesen and H. Bunke, *Iam graph database repository for graph based pattern recognition and machine learning*, in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297, Springer, 2008.

[248] P. D. Dobson and A. J. Doig, *Distinguishing enzyme structures from non-enzymes without alignments*, *Journal of molecular biology* **330** (2003), no. 4 771–783.

[249] E. Sarubbi, P. F. Seneci, M. R. Angelastro, N. P. Peet, M. Denaro, and K. Islam, *Peptide aldehydes as inhibitors of hiv protease*, *FEBS letters* **319** (1993), no. 3 253–256.

[250] T. Zhao, G. Liu, D. Wang, W. Yu, and M. Jiang, *Learning from counterfactual links for link prediction*, in *ICML*, 2022.

[251] D. K. Hammond, P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory*, *Applied and Computational Harmonic Analysis* **30** (2011), no. 2 129–150.

[252] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, *Representation learning on graphs with jumping knowledge networks*, in *ICML*, 2018.

[253] J. Klicpera, A. Bojchevski, and S. Günnemann, *Predict then propagate: Graph neural networks meet personalized pagerank*, in *ICLR*, 2019.

[254] J. Klicpera, S. Weißenberger, and S. Günnemann, *Diffusion improves graph learning*, in *NeurIPS*, 2019.

[255] R. Andersen, F. Chung, and K. Lang, *Local graph partitioning using pagerank vectors*, in *IEEE Symposium on Foundations of Computer Science*, pp. 475–486, IEEE, 2006.

[256] F. Chung, *The heat kernel as the pagerank of a graph*, *PNAS* **104** (2007), no. 50 19735–19740.

[257] J. L. Vázquez, *The mathematical theories of diffusion: nonlinear and fractional diffusion*, in *Nonlocal and nonlinear diffusions and interactions: new methods and directions*, pp. 205–278. Springer, 2017.

[258] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, in *NeurIPS*, 2016.

[259] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, *N-gcn: Multi-scale graph convolution for semi-supervised node classification*, in *UAI*, 2020.

[260] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, *Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing*, in *ICML*, 2019.

[261] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, *Lanczosnet: Multi-scale deep graph convolutional networks*, in *ICLR*, 2019.

[262] J. Atwood and D. Towsley, *Diffusion-convolutional neural networks*, in *NeurIPS*, 2016.

[263] J. Shi and J. Malik, *Normalized cuts and image segmentation*, *IEEE TPAMI* **22** (2000), no. 8 888–905.

[264] A. Clauset, M. E. Newman, and C. Moore, *Finding community structure in very large networks*, *Physical review E* **70** (2004), no. 6 066111.

[265] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, *Journal of statistical mechanics: theory and experiment* **2008** (2008), no. 10 P10008.

[266] V. A. Traag, L. Waltman, and N. J. Van Eck, *From louvain to leiden: guaranteeing well-connected communities*, *Scientific reports* **9** (2019), no. 1 5233.

[267] B. Perozzi and L. Akoglu, *Anomalous: A joint modeling approach for anomaly detection on attributed networks*, in *IJCAI*, 2018.

[268] L. Gutiérrez-Gómez, A. Bovet, and J.-C. Delvenne, *Multi-scale anomaly detection on attributed networks*, in *AAAI*, 2020.

[269] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, *A semi-supervised graph attentive network for financial fraud detection*, in *ICDM*, 2019.

[270] B. Branco, P. Abreu, A. S. Gomes, M. S. Almeida, J. T. Ascensão, and P. Bizarro, *Interleaved sequence rnns for fraud detection*, in *SIGKDD*, 2020.

[271] C. Liu, Q. Zhong, X. Ao, L. Sun, W. Lin, J. Feng, Q. He, and J. Tang, *Fraud transactions detection via behavior tree with local intention calibration*, in *SIGKDD*, 2020.

[272] B. Perozzi and L. Akoglu, *Scalable anomaly ranking of attributed neighborhoods*, in *SDM*, 2016.

[273] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, *A deep learning approach to network intrusion detection*, *IEEE transactions on emerging topics in computational intelligence* **2** (2018), no. 1 41–50.

[274] D. Chou and M. Jiang, *A survey on data-driven network intrusion detection*, *ACM Computing Surveys (CSUR)* **54** (2021), no. 9 1–36.

[275] S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu, *Unsupervised fake news detection on social media: A generative approach*, in *AAAI*, 2019.

[276] V.-H. Nguyen, K. Sugiyama, P. Nakov, and M.-Y. Kan, *Fang: Leveraging social context for fake news detection using graph representation*, in *CIKM*, 2020.

[277] Y. Liu and Y.-F. B. Wu, *Fned: a deep network for fake news early detection on social media*, *ACM Transactions on Information Systems (TOIS)* **38** (2020), no. 3 1–33.

[278] K. Ding, J. Li, R. Bhanushali, and H. Liu, *Deep anomaly detection on attributed networks*, in *SIM*, 2019.

[279] S. Bandyopadhyay, S. V. Vivek, and M. Murty, *Outlier resistant unsupervised deep architectures for attributed network embedding*, in *WSDM*, 2020.

[280] H. Fan, F. Zhang, and Z. Li, *Anomalydae: Dual autoencoder for anomaly detection on attributed networks*, in *ICASSP*, 2020.

[281] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng, *A deep multi-view framework for anomaly detection on attributed networks*, *IEEE TKDE* **34** (2020), no. 6 2539–2552.

[282] T. Zhao, T. Jiang, N. Shah, and M. Jiang, *A synergistic approach for graph anomaly detection with pattern mining and feature learning*, *IEEE Transactions on Neural Networks and Learning Systems* **33** (2021), no. 6 2393–2405.

[283] K. Ding, Q. Zhou, H. Tong, and H. Liu, *Few-shot network anomaly detection via cross-network meta-learning*, in *WebConf*, 2021.

[284] Z. Xu, X. Huang, Y. Zhao, Y. Dong, and J. Li, *Contrastive attributed network anomaly detection with data augmentation*, in *PAKDD*, 2022.

[285] J. Tang, J. Li, Z.-C. Gao, and J. Li, *Rethinking graph neural networks for anomaly detection*, in *ICLR*, 2022.

[286] G.-G. Leonardo, A. Bovet, and J.-C. Delvenne, *Multi-scale anomaly detection on attributed networks*, in *AAAI*, 2020.

[287] X. Luo, J. Wu, A. Beheshti, J. Yang, X. Zhang, Y. Wang, and S. Xue, *Comga: Community-aware attributed graph anomaly detection*, in *WSDM*, 2022.

[288] H. Wang, C. Zhou, J. Wu, W. Dang, X. Zhu, and J. Wang, *Deep structure learning for fraud detection*, in *ICDM*, 2018.

[289] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, *Error-bounded graph anomaly loss for gnns*, in *CIKM*, 2020.