UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Learning on Complex Graphs

by Xu Chen

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Ivor W. Tsang

University of Technology Sydney
Faculty of Engineering and Information Technology

Aug. 2021

Certificate of Original Authorship

Graduate research students are required to make a declaration of original authorship when they submit the thesis for examination and in the final bound copies. Please note, the Research Training Program (RTP) statement is for all students. The Certificate of Original Authorship must be placed within the thesis, immediately after the thesis title page.

Required wording for the certificate of original authorship

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Xu Chen declare that this thesis, is submitted in fulfilment of the requirements for the award of doctor of philosophy, in the faculty of engineering and information technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.
**If applicable, the above statement must be replaced with the collaborative doctoral degree statement (see below).*

**If applicable, the Indigenous Cultural and Intellectual Property (ICIP) statement must be added (see below).*

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 2022–5-03

Collaborative doctoral research degree statement

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with Shanghai Jiao Tong University.

Indigenous Cultural and Intellectual Property (ICIP) statement

This thesis includes Indigenous Cultural and Intellectual Property (ICIP) belonging to custodians or traditional owners. Where I have used ICIP, I have followed the relevant protocols and consulted with appropriate Indigenous people/communities about its inclusion in my thesis. ICIP rights are Indigenous heritage and will always remain with these groups. To use, adapt or reference the ICIP contained in this work, you will need to consult with the relevant Indigenous groups and follow cultural protocols.

Acknowledgements

I would like to take this good opportunity to show my gratitude to my advisors, several professors, my colleagues, my friends and my family for their significant help during my doctoral study in University of Technology Sydney and Shanghai Jiao Tong University.

First, I would like to express my foremost and deepest gratitude to my supervisor Prof. Ivor W. Tsang, who plays an important role in my doctoral life. During my research life in University of Technology Sydney, I am deeply impressed by his broad knowledge in machine learning and his pure altitude toward research. In each discussion with him, I can learn something new about how to understand different works, how to bridge them and how to generalize them to other tasks. These cases improve my research taste to a new level and tell me what is research philosophy. He teaches me how to think, how to summarize and then how to create, which helps me to touch top conferences and journals. Besides, he also gives me a lot of courage to face difficulties, especially when I was nearly on the brink of collapse in my doctoral life. With his help, I really find the happiness of doing pure research.

Second, I also would like to appreciate my co-supervisor Prof. Ya Zhang, who gives me sufficient freedom to explore the research area I like and helps me build my research basis. She patiently corrects my bad research habits, showing me how to formalize a good research style in the doctoral life. In particular, when I was new to the doctoral research, I often made some stupid mistakes and got stuck in various problems. Prof. Ya Zhang always encourages me and pulls me back from dilemmas. Her vision on the proposal of how to judge a good idea and endless help on teaching me the writing of research papers really help me a lot in my later research life. Besides, through the discussion with her, I learned how to think and understand a research work from the view of reviewers. I cannot finish this thesis if without her high standards, unlimited patience, generous support and guidance.

I feel very lucky to attend the joint degree program between University of Technology Sydney and Shanghai Jiao Tong University. Wherein I would like to express my sincere appreciation to Prof. Ivor W. Tsang, Prof. Ya Zhang and Prof. Siheng Chen, who provide their selfless help to me and offer me a great opportunity to work in such great teams.

Within my doctoral life, I have met and made friends with many famous researchers in different areas and learned many useful experiences from them. Specifically, with the help of Jiangchao Yao, I have the chance to meet Prof. Ivor W. Tsang. Thanks to my friend Yuangang Pan, I can quickly adapt to the life in Australia. With the help of them, I have broaden my eyes on the machine learning community, and also realize what I lack for the top research level in detail.

Then, it comes to my dear colleagues and friends in my doctoral study. I would like to thank Dr.Yexun Zhang, Dr.Zhiyi Tan and Fei Ye for their selfless help in my research, Huangjie Zheng, Yujun Gu, Jie Chang, Peisen Zhao, Haoyan Guan, Jiajie Wang, Hanqing Zhao, Yu Li, and Bo Huang for being the "The ten brothers of gourd" in our team, the lovely juniors Maosen Li, Kenan Cui, Hao Wu, Zhikang Li, Jingchao Su, Chengyang Li, Yingying Xue, Tengfei Hou, Yifei Hu, Luochen Lyu and many others. I am happy to study with you all and wish you all have a bright future. I am also grateful to all the other friends in Sydney: Yuangang Pan, Yueming Lyu, Yinghua Yao, Muming Zhao, Jing Li, Xiaowei Zhou, Yan Zhang, Yaxin Shi, Xingrui Yu, Xiaofeng Xu and Xiaofeng Cao for their support and company.

Finally, I would like to express my deep gratitude to my parents, my sister and my girl friend. You all always get my back and stay around. It is you that give me great courage to solve so many difficulties and then I could better purse my research goal. The love from you will be remembered by me forever. In the unforgettable five years, I experienced so many things including both happiness and sadness. I can see that I grew up a lot, with my own cognition towards life. The life is usually of regrets, but cherishing the moment now is the key to a happy life.

DEEP LEARNING ON COMPLEX GRAPHS

ABSTRACT

Deep learning on graphs has recently become a hot research topic in machine learning and shown great promise in a wide variety of applications such as recommendation and social network analysis. This dissertation considers a set of more practical and challenging cases of learning on complex graphs, *i.e.*, edges with attribute information, nodes with missing attributes and multiple graphs with overlapped nodes.

Our first work aims to model the topological information in signed directed networks where the edges have additional attribute information, *i.e.* signs and directions. In signed directed networks, different signs and directions have different effects in information propagation, which raises challenges to model the structural information. We propose to decouple the modeling of signs and directions with different network parameters, and meanwhile maximize the log-likelihood of observed edges by a variational *evidence lower bound* to learn the node representations. The experimental results show the effectiveness of the proposed model on both link sign prediction and node recommendation task.

Our second work considers learning on attribute-missing graphs where attributes of partial nodes are entirely missing. Previous graph learning algorithms have limitations in dealing with this kind of graphs. The random walk based methods suffer from the sampling bias issue of structures. The popular graph neural networks feed the structures and attributes to a shared network, and thus become incompatible for attribute-missing nodes. To better learn on attribute-missing graphs, we consider the structures and

attributes as two correlated views of the node information and make a shared-latent space assumption of these two views. Based on the assumption, we propose to model the two views by two different encoders and meanwhile maintain their joint distribution by a novel distribution matching scheme. Extensive experiments on seven real-world datasets show the superiority of the proposed model on both the link prediction task and the newly introduced node attribute completion task.

Moreover, single graphs may have overlapped nodes and become one complex graph. A popular case is the user-item bipartite graphs in cross domain recommendation where users are shared while items are from different domains. Previous methods usually emphasize the overlapped features of user preferences while compromise the domain-specific features or learn the domain-specific features by heuristic human knowledge. Our third work proposes to learn both features in a more practical way by an equivalent transformation assumption. The assumption hypothesizes the user preference in each domain can be mutually converted to each other by equivalent transformation. Then, a novel equivalent transformation based distribution matching scheme is developed to model the joint distribution of user behaviors across domains and conduct the recommendation task. The results on three real-world benchmarks confirm the superiority of the proposed model.
KEY WORDS: deep learning, graph learning, complex graphs

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

$\mathcal{G}$	a graph
$\mathcal{V}$	the set of nodes
v_i	the ith node
$\mathcal{E}$	the observed edges
X	the attribute matrix of nodes
A	the adjacent matrix
N	the number of nodes
F	the attribute dimension
L	the symmetric normalized Laplacian matrix
I_N	an identity matrix of size N
d	the latent dimension
E	an encoder
D	an decoder
$\mathcal{D}$	a discriminator in adversarial learning
D_{KL}	Kullback-Leibler divergence
$\mathcal{L}$	loss function
$\mathcal{N}(\mu, \sigma)$	a Gaussian distribution
GNN	graph neural networks
ELBO	*evidence lower bound*
CDR	cross domain recommendation

Chapter 1 Introduction

1.1 Motivation

Graphs, as shown in Figure 1–1, are ubiquitous in real-world scenarios such as social networks and citation networks. Deep learning on graphs (DLG) has become an important machine learning topic which not only benefits many learning paradigms such as semi-supervised learning [1-3] and relation inference [4-6], but also facilitates enormous applications such as recommendation [7-9] and community detection [10]. Despite DLG has achieved remarkable achievements, it is confronted with many real-world cases where the graphs are more complex than that shown in Figure 1–1 and require specifically designed learning algorithms. For example, a complex graph may have heterogeneous nodes or edges (*e.g.* user-movie-director-actor network) or unconventional graph types (*e.g.* attributes of partial nodes are completely missing), where popular graph learning algorithms working on homogeneous graphs or conventional graphs cannot handle it. Deep learning on complex graphs is a broad topic due to the various types of complex graphs. In this dissertation, we mainly concentrate on three kinds of complex graphs, which also correspond to Chapter 3, Chapter 4 and Chapter 5 in the following part. As reviewed in Figure 1–2, In the edge level, edges may have additional attribute information such as signs and directions in signed directed networks (*e.g.* Epinions and Facebook) [11]. Different signs and directions have different effects in information propagation and can provide additional value for modeling the complex graph structures. In the attribute level, attributes of some nodes could be entirely missing. For example, in citation networks, abstracts or detailed contents of some papers may be inaccessible due to the copyright protection. In social networks, profiles of several users may be entirely missing due to the privacy protection. In the graph level, different single graphs may have overlapped nodes and become one more complex graph. For example, in cross domain recommendation, different user-item bipartite graphs may have overlapped users while non-overlapped items from different domains. These graphs work together to be a more complex graph which is one kind of heterogeneous information networks [12]. The above raises new challenges and difficulties of graph learning and motivates us the corresponding research to apply graph learning algorithms in more practical scenarios.

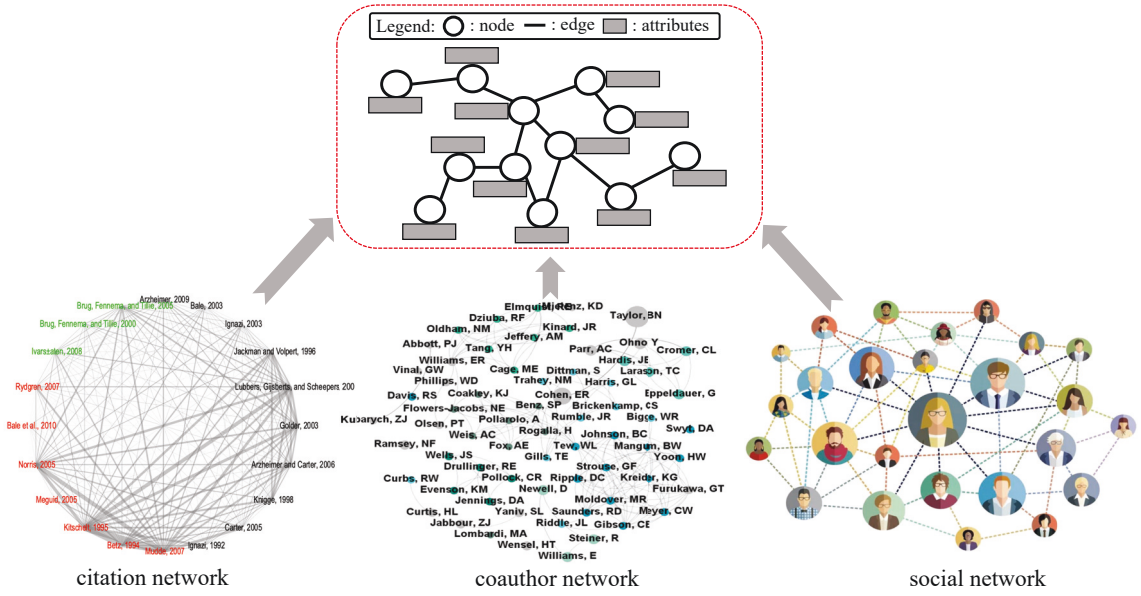


Figure 1–1 An example to show the graph structured data in different scenarios. The bottom row lists three graph examples (*i.e.* citation network, coauthor network, social network). The top row indicates an abstract graph example that summarizes the instances in different scenarios.

The popular proposal to overcome the difficulties is to consider the specific information of the complex graphs when designing the learning algorithms. For example, in signed directed networks, it is crucial to model the signs and directions since they provide additional information on the graph structures. If there is an algorithm that learns representative embeddings of nodes in signed directed networks, the algorithm could greatly boost many practical problems such as friend recommendation. Formally, deep learning on complex graphs aims to learn a reliable deep model on real-world graphs. In this dissertation, our target is to solve three types of complex graphs, and make a contribution on the progress of graph deep learning algorithms into practice.

1.2 Deep Learning on Graphs

Deep learning on graphs has been investigated from a variety of perspectives, including the methodologies [1, 11, 13-14] and the applications [7, 15]. Since we care more about the learning problems, we mainly introduce the different methodologies here. Recent deep learning on graphs can be classified into two categories: the network embedding based methods and the graph neural network based methods, which is structured in Figure 1–3.

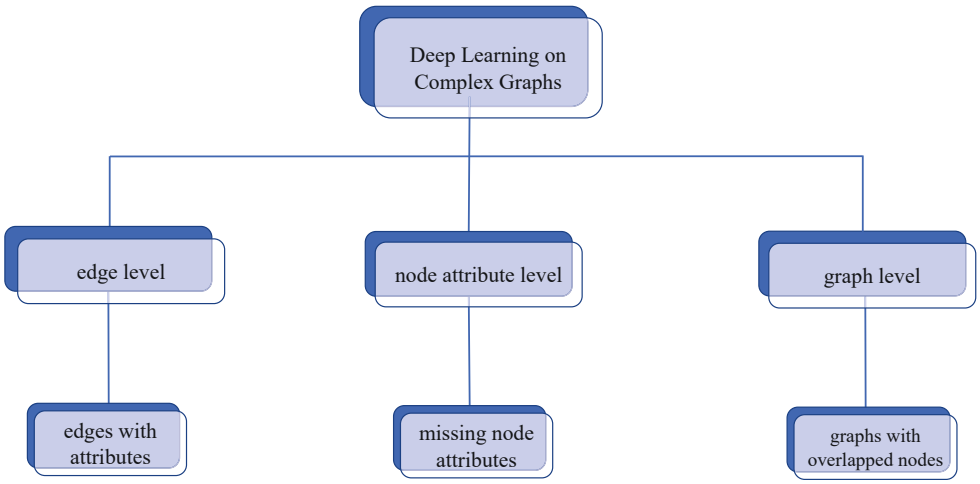


Figure 1–2 The tree to depict the studied three different scenarios of deep learning on complex graphs. In the edge level, the edges may have attribute information such as signs and directions. In the node attribute level, attributes of partial nodes may be entirely missing. In the graph level, different graphs may have overlapped nodes and become one more complex graph. This figure also indicates the studied three kinds of complex graphs in Chapter 3, Chapter 4 and Chapter 4, respectively.

Network Embedding: Network embedding can be partitioned into three popular families: the random walk based, the matrix factorization based and the deep neural network based. In the first family, random walk based methods resort to sample graph structures into sequences and then language models (*e.g.* skip-gram [16]) are applied to learn the node embeddings. They are motivated by the fact that the distribution of nodes appearing in short random walks follows a similar power-law distribution of words in language sentences [11, 17]. The random walk based method is one kind of statistical model that statistically represents the graph structures with sequences. Approved by the efficiency of random walk sampling, it supports distributed training well and can work on large-scale graphs with millions of nodes. Like a double-edged blade, the random walk sampling of graph structures is a biased approach [18] and usually requires fined-tuned hyper-parameters to guarantee the statistical quality of walked sequences.

In the second family, matrix factorization based methods take adjacent matrices of graphs as input, and then a variety of matrix factorization approaches can be utilized to learn low-dimensional node embeddings [13, 19]. Matrix factorization in network embedding is proposed to learn a low-rank space to encode the graph structures, in contrast with the original high-dimensional space. NetMF [13], as one typical matrix

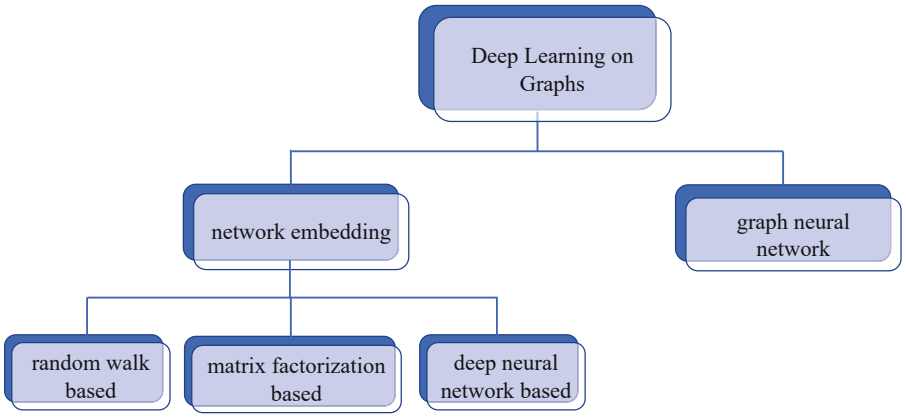


Figure 1–3 The tree to depict different methodologies of deep learning on graphs.

factorization method in network embedding, has been proved to be a more generalized case of some typical random walk based methods [11, 17, 20]. However, nowadays, matrix factorization in network embedding is less popular since it requires the whole adjacency matrix as input and thus consumes expensive computational resources on large-scale graphs. Moreover, most matrix factorization based models are linear and cannot capture the complicated and highly non-linear patterns in graphs.

In the third family, the main idea is to use deep neural networks to learn non-linear functions that can fit the graph data well. The key challenge lies in how to fully leverage the graph structures to constrain the learning of deep neural networks. Several typical models are SDNE [21] and SDAE [22], which are based on auto-encoding to address the challenge. Apart from the capacity of modeling the non-linear patterns, deep neural network based methods have the advantage of incorporating task-specific information in an end-to-end manner.

Graph Neural Network: Graph neural network (GNN) has become a well-known graph learning methodology in recent five years. It starts from ChebNet [14] which reformulates the inefficient spectral graph convolution with a Chebshev polynomial approximation. There are two key characteristics of GNN that prompt its success. One key characteristic is low-passing filtering, which emphasizes the low-frequency graph signals (*i.e.* useful information) and suppresses the high-frequency graph signals (*i.e.* noise) [3, 23]. The other one is smoothing that aggregates node features from its neighbours [24]. The smoothing characteristic encourages a node resembles its neighbours in an embedding space. Although it has the advantage of providing complementary information when the information of a single node is not sufficient, it could have the over-smoothing

problem where all node embeddings converged to the same subspace so that they are hard to be distinguished [24]. In general, GNN is a promising direction for deep learning on graphs and emerging works are polishing the graph convolutional theory.

1.3 Deep Learning on Complex Graphs

In last five years, the advances of DLG including different convolution filters [2-3, 25-26], a variety of pooling techniques [27-29] and advanced learning approaches [30-33] have put forward the research progress in many areas. Most graph algorithms work on the standard graphs shown in Figure 1–1. However, the graphs in real-world systems are usually more complex, and deep learning on complex on graphs is worth more research attention. In summary, the following challenges raise when considering learning on complex graphs:

- *Especially on graphs, there are various types of complex graphs such as edges with attribute information, missing node attributes and single graphs with overlapped nodes. The difficulties in different types vary a lot and need type-specific solutions.*
- *Different graphs may have different statistics that match different graph convolutional filters. The designed algorithms should be flexible to adapt to different convolutional filters.*

Most aforementioned graph algorithms cannot well tackle the learning difficulties of complex graphs. For the studied scenarios (*i.e.* edges with attribute information, missing node attributes and graphs with overlapped nodes), we demonstrate what the limitations of existing algorithms are and how we resolve the corresponding problems.

Considering the learning in signed directed networks, most current works [1, 11, 17] concentrate on modeling the structural information for unsigned and undirected graphs. However, in many social systems, users may build both positive directed edges (*e.g.* trust) and negative directed edges (*e.g.* distrust) with others. The negative sign contain additional information [34-36] that helps many tasks, *e.g.* link sign prediction and node classification. Besides, the edge direction indicates the asymmetric relationship between two users, which is important for friend recommendation in social networks [37-38]. As shown in Figure 1–4, the patterns in signed directed networks are coupled with signs and directions. Most aforementioned models [1, 11, 17] cannot well capture the patterns since they fail to recognize the value of signs and directions. How to better model the structures considering the signs and directions is a key challenge. In particular,

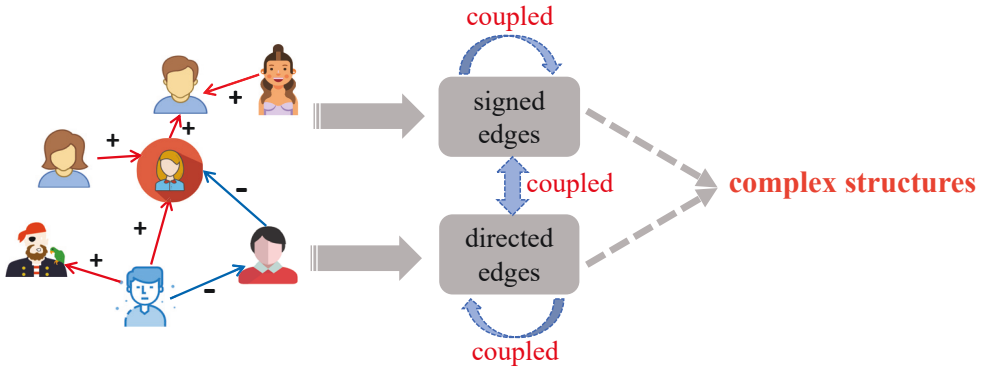


Figure 1–4 A social network example shows the characteristics (i.e. signed edges and directed edges) in signed directed networks. The structures in signed directed networks are coupled coupling signals inner signed edges, inner directed edges, between signed and directed edges.

we propose to decouple the modeling of signs and directions with different network parameters, and learn the node representations in a variational auto-encoding framework. This framework ensures the decoupling formulation is an evidence-bounded objective on maximizing the log-likelihoods of the observed data. By using this framework, we are able to learn more representative node embeddings for downstream tasks.

When considering the attribute-missing graph where attributes of some nodes are entirely missing, there are few methods specifically studying on how to learn on these graphs. Although recently emerged attributed random walk methods [39-40] can potentially deal with it, they rely on high-quality random walks and require carefully designed sampling strategies [41]. Moreover, current GNN framework is incompatible with these graphs since it feeds structures and attributes into a shared encoder. In our analysis, we consider structures and attributes as two correlated views that follow an implicit and intractable joint distribution to reflect the information of nodes. To tackle the learning on attribute-missing graphs, we make a shared-latent space assumption of the two views and model the joint distribution by deep generative techniques. Specifically, we encode the structures and attributes with two different encoders, and model the joint distribution by a variational *evidence lower bound* (*ELBO*) of the log-likelihood on the observed data. Then, we take the *ELBO* as a surrogate objective for learning node representations. The proposed method is a general framework that can incorporate a variety of GNN backbones to adapt to different data statistics.

When studying the learning on graphs with overlapped nodes, we explore to collaborate the learning of graphs from different domains. Specifically, in this dissertation,

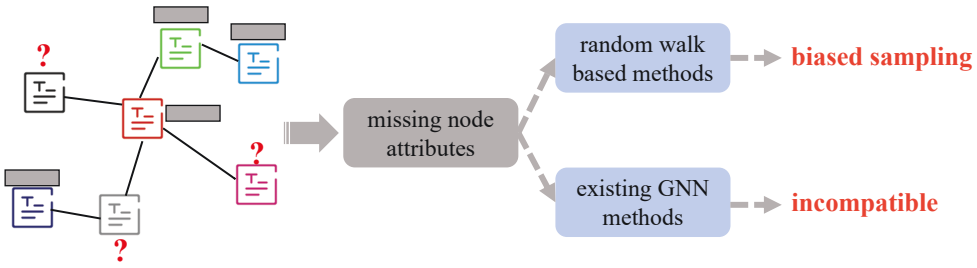


Figure 1–5 A citation network example shows the characteristics of attribute-missing graphs, where attributes of only partial nodes are available and those of other nodes are entirely missing. Current popular methods including the random walk based and the GNN based cannot well handle this case.

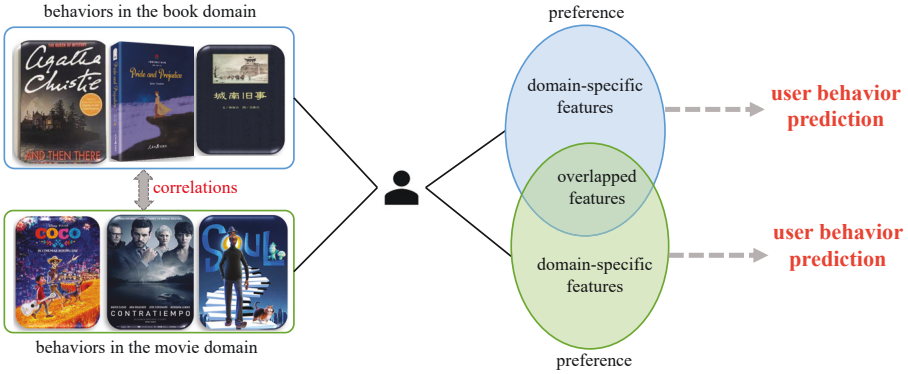


Figure 1–6 An example to show that the user preference in CDR has both overlapped features and domain-specific features. With the correlations of user behaviors in different domains, knowledge transfer techniques can be employed to improve the learning of user preference and help the user behavior prediction in each domain.

we study how to incorporate the learning of different user-item bipartite graphs in cross domain recommendation (CDR). In CDR, most previous works are based on the idea of shared-user representations, which resort to learn the overlapped features for feature alignment, but usually compromise the domain-specific features that help better user behavior prediction [42-44]. Although several works [42, 45-46] employ Multi-Layer Perception (MLP) as the mapping function to allow the learning flexibility for user representations in each domain, they usually require heuristic human knowledge of choosing training samples to avoid over-fitting for the mapping function [42]. We propose to learn both features in a more practical way by an equivalent transformation assumption. To be specific, the assumption indicates each user's preference in each domain can be mutually converted to each other with equivalent transformation. Further, we develop an equivalent transformation based distribution matching scheme to model the joint distribution

of user behaviors across domains and facilitate better user behavior prediction. Extensive experiments on three CDR benchmarks demonstrate the superiority of the proposed model.

1.4 Contributions and Organizations

This thesis explores the problems of deep learning on complex graphs. In correspondence, we propose different methods to solve a series of issues in each case and achieve consistent improved performance compared to existing methods. The main contributions of this thesis are summarized in the following two perspectives:

- **Theoretical contributions**: When studying specific problems of deep learning on complex graphs in Chapter 4 and Chapter 5, we present theoretical methods to model the joint distribution of two different views. These methods are generic and could be applied to other kinds of applications.
- **Algorithmic contributions**: By analyzing various scenarios of complex graphs, we provide a deep analysis on the necessity of deep learning on complex graphs. Further, we propose the corresponding algorithms to work in different scenarios.
- **Application contributions**: The proposed algorithms have been evaluated on a variety of graph benchmarks, and a series of practical tasks in real-world cases. Extensive experiments have confirmed the superiority of the proposed algorithms both quantitatively and qualitatively.

The structures of the remaining Chapters in this dissertation are organized as follows:

- Chapter 2 briefly reviews related works of deep learning on graphs, deep learning on complex graphs and deep generative modeling that involves in the design of our algorithms.
- Chapter 3 proposes to decouple the modeling of signs and directions in signed directed networks to better capture the topological information. In particular, the decoupling idea is formulated from a variational auto-encoding perspective, which makes the proposed model is evidence-bounded on the log-likelihoods of the observed data.
- Chapter 4 targets to solve the problems of learning on attribute-missing graphs. In this chapter, we reformulate GNN from a distribution matching perspective and propose a novel distribution matching based GNN framework that maximizes the joint log-likelihoods on the observed data. Further, we derive the *ELBO* as a

surrogate objective for efficient optimization.

- Chapter 5 studies the problem of collaborating different user-item bipartite graphs in CDR. Specifically, an equivalent transformation based model is proposed to better transfer knowledge and predict user behaviors in each domain.
- Chapter 6 concludes this thesis and discusses the future directions.

Chapter 2 Background

As discussed in Chapter 1, the main goal of this thesis is to design deep learning algorithms on complex graphs. This chapter will simply review the related graph learning algorithms. Section 2.1 reviews deep learning works on graphs in recent years. Section 2.2 introduces related works on signed directed networks. As there are few works targeting on the attribute-missing graph, we do not split a section for the introduction. Instead, graph algorithms that can potentially deal with it are mentioned in Section 2.1. Section 2.3 reviews the related works of incorporating different user-item bipartite graphs in CDR. Further, since the designed algorithms involve the techniques of deep generative modeling, we use Section 2.4 to show recent advances in deep generative models. It is also worthwhile to mention that this thesis concentrates on the node focused learning whose target is related to individual nodes on graphs [47]. The graph focused learning whose target is related to the whole graph is beyond the scope of this research.

2.1 Deep Learning on Graphs

Recent studies of DLG are mainly from two perspectives: network embedding and GNN. The main distinction between them is network embedding contains various kinds of methods for the link reconstruction task while GNN is designed for various tasks [48]. Therefore, GNN is able to tackle the network embedding problem through a auto-encoder framework. In the following parts, we respectively introduce the related network embedding based and GNN based methods.

2.1.1 Network Embedding

Network embedding [11, 17, 20] is proposed to learn representative node embeddings on graphs. There are various techniques including the random walk based, the matrix factorization based and the deep neural network based.

Random walk based: The random walk based is the most popular methodology in network embedding. At the beginning, Perozzi et al. found that the distribution of nodes appearing in short random walks follows a power-law distribution that is similar to the distribution of words in sentences. Based on this, they proposed DeepWalk [11] which

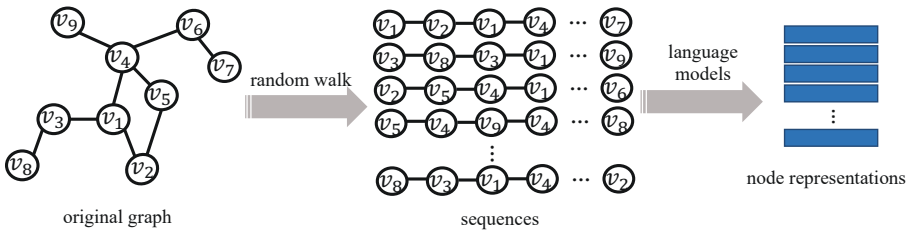


Figure 2–1 An example to illustrate the typical learning process of random walk based network embedding.

is the first random walk based method on graphs. In DeepWalk, the authors treat random walks as sentences and apply advanced models (*e.g.* skip-gram [49]) in natural language processing (NLP) to learn node representations. DeepWalk is a statistical model that uses the truncated random walks to represent the structures of graphs. In LINE [20], the authors designed an objective function that maintains both the local and global topological information on graphs. In the objective function, a specific "context" of nodes is defined and nodes in a similar "context" are expected to be similar in the embedding space. LINE is also a special case of DeepWalk when the size of "context" equals one [13]. Node2Vec [17] extends DeepWalk by defining a flexible notion of a node' s neighbours and designs a biased random walk to control the Bread First Search (BFS) and Deep First Search (DFS) on graphs. Then Node2Vec learns a feature extractor that maximizes the likelihood of preserving the neighbours of nodes on graphs. In practice, Node2Vec needs fine-tuned hyper-parameters of the BFS and DFS to achieve satisfied performance.

After DeepWalk and Node2Vec, various techniques are proposed to improve them. For example, Dai et al. [50] incorporated adversarial learning [51-52] to impose prior distribution on the latent embeddings. As the above random walk based methods do not support end-to-end training with auxiliary information (*e.g.* node attributes, node labels), researchers have investigated different ways to benefit from richer information. For instance, in [53-54], node attributes are taken as another kind of nodes and attributed random walking is proposed to exploit the information of attributes for graph learning. The attributed random walk based algorithms can also potential deal with the attribute-missing graphs, while they usually suffer from a biased sampling of structures and require fine-tuned hyper-parameters to guarantee the high-quality of walked sequences. A typical learning process of random walk based network embedding is shown in Figure 2–1.

Matrix factorization based: The matrix factorization based is another perspective to conduct network embedding. As a graph can be represented by an adjacency

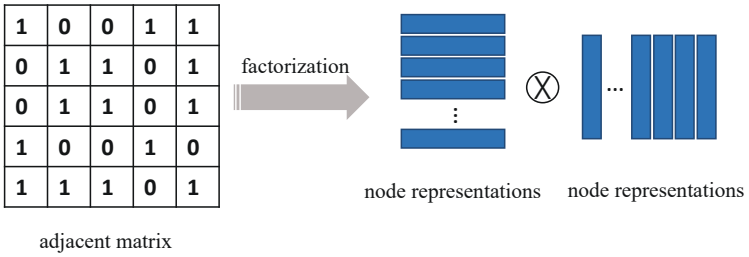


Figure 2–2 An example to illustrate the typical learning process of matrix factorization based network embedding. In this figure, the adjacency matrix can be substituted with other graph matrices such as graph Laplacian matrix. Various factorization approaches can be employed such as SVD. ⊗ indicates a calculation function which is specific to the factorization method.

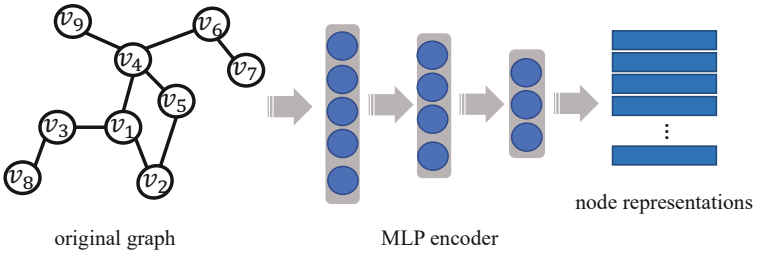


Figure 2–3 An example to illustrate the general process of DNN based graph learning algorithms.

matrix where each row and column is a node and the values indicate the relationships among nodes, some typical matrix factorization based methods can be used to learn low-dimensional node embeddings. For example, Singular Value Decomposition (SVD) is widely used in network embedding because of its optimality for low-rank approximation [37]. Non-negative matrix factorization is another common technique that can be used due to its interpretability and advantages as an additive model [55]. In addition, Qiu et al. [13] provided the theoretical connections between graph Laplacian matrix and the random walk based algorithms. They pointed out DeepWalk [11] and Node2Vec [17] are also matrix factorization based methods and developed NetMF [13] to more efficiently compute network embeddings. Although the matrix factorization based has its advantages such as interpretability, they usually are linear functions and hard to capture the non-linear topological information on graphs. Besides, matrix factorization on large-scale graphs is computational-inefficient. A typical learning process of matrix factorization based methods is shown in Figure 2–2.

Deep neural networks based: The goal of network embedding is to transform the original network space into an embedding space, which means to learn a mapping

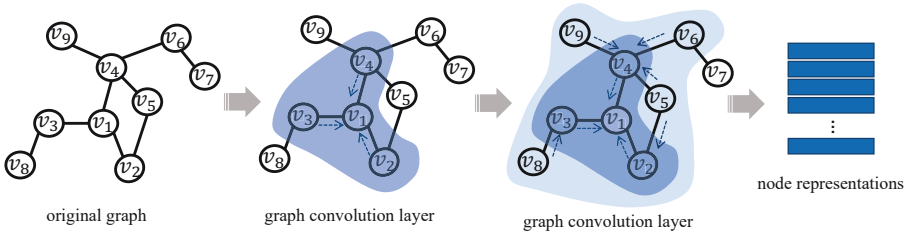


Figure 2–4 An example to illustrate the general process of GNN based graph learning algorithms. In this figure, we take node v_1 as an anchor node and show two convolutional layers that involves different orders of the information propagation for v_1. In the convolutional layers, the orders are visualized by different shades.

function that conducts the transformation. The aforementioned matrix factorization based methods assume the mapping function is linear and thus are not adequate to capture the complicated and highly non-linear patterns in the transformation [56].

In seeking for a non-linear mapping function, deep neural networks (DNN) is a quite suitable choice since it has shown impressive success in many areas [57-58]. Researchers have studied how to use DNN to encode the non-Euclidean graph structures into the embedding space. For example, SDNE [21] proposes to encode the structural information by multiple non-linear layers and an objective function that preserves the first order and second proximity on graphs. Different from previous works [11, 17] that employ random walking to sample graph structures, DNGR [22] introduces a random surfing model to preserve the structures directly. The authors theoretically demonstrated that the random surfing model has better capability to gather graph structures than the widely used sampling strategies. However, as the results shown in [22], DNGR requires a lot of efforts and expertise of tuning the hyper-parameters to achieve satisfied performance. Besides, the idea of using DNN on graphs is also extended in some specific graph types such as signed graphs [59] and heterogeneous information networks [60]. A general learning process of DNN based methods is shown in Figure 2–3.

2.1.2 Graph Neural Networks

Inspired by recent success of convolution on images in Euclidean space, enormous researchers have attempted to define the convolution on graphs in non-Euclidean space. GNN has received great attention and become the most popular methodology in graph representation learning. Since the algorithm design is based on GNN in this thesis, we

provide a more detailed introduction about it.

The emerging of GNN: To better understand GNN, we start from the definition of spectral convolution on graphs. Spectral convolution on graphs is defined as the multiplication of a signal $x \in \mathbb{R}^N$ by a parameterized filter g_θ in the Fourier domain. Let $\star$ be the convolution operation, the convolution on graphs is written as:

$$g_\theta \star x = U g_\theta(\Lambda) U^T x \tag{2–1}$$

where U is the eigenvector matrix and Λ is the eigenvalue matrix of the graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$. $U^T x$ indicates the graph Fourier transform of x. According to [14], a polynomial filter is usually taken as $g_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k \Lambda^k$.

However, the convolution filter defined in Eq. 2–1 involves the eigen-decomposition of L and might be computationally expensive for large graphs. To overcome this difficulty, according to [14], $g_\theta(\Lambda)$ with the polynomial filter can be well-approximated by a truncated expansion in terms of Chebshev polynomials $T_k(x)$ up to K^{th} order:

$$g_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k \Lambda^k \approx \sum_{k=0}^{K} \theta_k T_k(\widetilde{\Lambda}) \tag{2–2}$$

where $\widetilde{\Lambda} = \frac{2}{\lambda_{max}} \Lambda - I_N$ is a rescaled version of Λ and λ_{max} indicates the largest eigenvalue of L. The Chebshev polynomials are recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$. Then taking Eq. 2–2 into consideration, Eq. 2–1 can be written as:

$$g_\theta \star x \approx \sum_{k=0}^{K} \theta_k T_k(\widetilde{L}) x \tag{2–3}$$

where $\widetilde{L} = \frac{2}{\lambda_{max}} L - I_N$. Eq. 2–3 is also called as the K-localized convolution on graphs since it is a K-th order polynomial in the Laplacian.

Inspired by the idea that high-order convolutions can be built by stacking multiple convolutional layers [61], graph convolutional networks (GCN) [1] achieves K^{th} convolution by stacking multiple convolutional layers of Eq. 2–3, and each layer is followed by a point-wise non-linear function. In particular, the layer-wise convolution in Eq. 2–3 is defined as $K = 1$, which indicates a linear function on the graph Laplacian spectrum. Additionally, GCN approximates $\lambda_{max} \approx 2$ by assuming the neural network can adapt to this change in the training process, which simplifies Eq. 2–3 as:

$$g_\theta \star x \approx \theta_0 x + \theta_1 (L - I_N) x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \tag{2–4}$$

where θ_0 and θ_1 are two free parameters. In practice, GCN constrains $\theta = \theta_0 = -\theta_1$ to avoid over-fitting, leading to the following expression:

$$g_\theta \star x \approx \theta(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x \tag{2–5}$$

Note that $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ now has eigenvalues that range in $[0, 2]$. Repeating the calculation in Eq. 2–5 will lead to numerical instabilities and even exploding or vanishing gradients when stacking multiple layers. To solve this problem, a renormalization trick is introduced as $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \widehat{D}^{-\frac{1}{2}}(A + I_N)\widehat{D}^{-\frac{1}{2}}$, where $\widehat{D}$ is the degree matrix of $A + I_N$. Then when giving a signal matrix $X \in \mathbb{R}^{N\times F}$ where N denotes the number of samples and F denotes the feature dimension, the layer-wise graph convolution in [1] is defined as follows:

$$Z^{l+1} = \widetilde{A}H^l\Theta^l,\ H^l = h(Z^l),\ H^0 = X \tag{2–6}$$

where the propagation matrix $\widetilde{A} = \widehat{D}^{-\frac{1}{2}}(A+I_N)\widehat{D}^{-\frac{1}{2}}$. H^l is the activation matrix in the l-th layer, whose each row is the vector representation of a node. Θ^l is now a matrix of filter parameters in the l-th layer and $h(\cdot)$ is the non-linear $Relu$ function. $Z^{l+1} \in \mathbb{R}^{N\times d}$ is the node representation of $(l + 1)$-th layer. This layer-wise convolution as well connects the graph convolution operation in spectral domain to that in the spatial domain. From above we can see that GCN learns a node's representation by aggregating its neighbors which are also called as the *receptive field*. The *receptive field* is enlarged through stacking layers like the L-hop in a graph. When $\widetilde{A} = I_N$, GCN degrades to a multi-layer perceptron (MLP) model, which indicates the graph structures are not considered and the *receptive field* of a node is just itself.

The development of GNN: After ChebNet [14] and GCN [1], enormous researchers have explored the idea of convolutions on graphs. For instance, GraphSage [8] formulates the convolutional kernel as different aggregation functions such as "mean-pooling", "max-pooling" and "Long-short Term Memory (LSTM)" [62]. Different aggregation functions indicate different information propagation schemes and are able to recognize different structures [63]. Due to the success of attention mechanism in various topics [58, 64], Petar et al. [26] stated that the neighbours contribute differently to a node in the information propagation process and proposed graph attention networks (GAT). GAT has more powerful representation learning ability since it has a dynamic and learnable graph convolution filter formulated by the attention module. Other works [65-66] focus on improving the inefficient recursive sampling strategy of neighbours in GraphSage.

Importance sampling and variance reduction techniques are introduced in [65] and [66], respectively.

Deep insights and improvement of GNN: Recently, several works [3, 23] reveal that the above graph convolution is a low-pass filter which emphasizes low-frequency signals (*i.e.* useful information) and suppresses high-frequency signals (*i.e.* noise). Following this, GraphHeat [25] is proposed to enhance the low-pass filtering characteristic by a heat kernel [67]. Meanwhile, Li et al. [24] pointed out that the graph convolution of GCN is a special form of Laplacian smoothing and stacking too many layers would lead to the over-smoothing problem. The over-smoothing indicates that the node representations are over-smoothed to the same subspace and thus indistinguishable from each other. Besides, GCN also has a severe over-fitting problem when the training labels are few [24, 31, 48, 68]. Over-smoothing and over-fitting are two important problems of existing GNN algorithms, and have been investigated from different perspectives. For example, dropout [69] is a widely used technique to alleviate over-fitting by randomly setting certain feature dimensions to zeros. Specially for GNN, Rong et al. [68] proposed DropEdge, where the edges in graphs are dropped with a certain ratio before graph convolutions. DropEdge generates different perturbations of the graph connections and acts as one data augmentation technique to alleviate the over-fitting and over-smoothing problem during training. Inspired by the residual connections in Convolutional Neural Networks (CNN) [70], Li et al. [30] introduced residual connections in GCN and proposed ResGCN. Similarly, JKNet [31] designs a jumping knowledge network to fuse the features from different layers. CGNN [71] proposes two characteristics of node representations, *i.e.* the stability which means a node's representation is stable to the perturbations and identifiability which indicates nodes with different structures have different representations, to prevent the over-fitting and over-smoothing issue.

GNN shows promising performance not only because of its integrity in combining node attributes but also due to its idea of graph signal processing. It is a fast-developing research field that consists of exciting practical values as well as great challenges.

2.2 Learning on Signed Directed Networks

The graph learning algorithms discussed in Section 2.1 are designed for unsigned or undirected networks. In reality, both the existence of directed and signed (positive and negative) links in social media are ubiquitous. The negative links have been proven

to have distinct properties and added value over positive links [72-73]. Several works have studied how to distinctly model the positive and negative links in signed directed networks. Degree based features like the number of positive-incoming and negative-incoming links are explored in [72]. While these hand-crafted features are limited and not capable in many situations. Instead, in [74], spectral analysis is extended for signed networks. Matrix Factorization (MF) [75] is also adopted to learn low-dimensional embeddings for signed directed networks. To reduce the computation burden of matrix decomposition, a specific aggregation manner for learning node embeddings in signed networks is proposed in [76]. It follows the principle that the enemy of a friend is an enemy and the enemy of an enemy is a friend, which extends the positive and negative neighbors for each node.

Due to the superior representation learning ability of deep learning, researchers attempt to use deep learning techniques to learn more representative node embeddings. A deep learning framework for signed network named SiNE is proposed in [59], where the objective function is guided by social theory. Although the framework leverages non-linearity to learn node representations, it does not model link direction which is an important factor for some asymmetric tasks. SNE is proposed in [77] and log-bilinear model is extended to support sign and direction modeling. SNE trains node embeddings based on a uniform random walk and node context rather than social theory. However, random walk in SNE applies homophily effects on different signs and fail to capture the local structures in signed directed networks, as well as does not support end-to-end training. SIDE [78] is another random walk based method based on social balance theory [79]. SNEA [80] exploits both the network structures and node attributes simultaneously for network embedding on attributed signed networks. Specifically, a margin ranking loss is proposed in SNEA. However, the margin ranking loss is non-smooth and difficult to be optimized by gradient based algorithms.

From the above, we see that most existing works focus on capturing the first-order topology, namely modeling the directly linked nodes. Although some methods have introduced random walk [77] to capture the *high-order* topology, they cannot recognize the value of different signs.

2.3 Learning to Collaborate Different Bipartite Graphs

When single graphs have overlapped nodes and become one complex graph, researchers propose to incorporate different graphs together to boost the representation learning performance. In this dissertation, we mainly focus on the user-item bipartite graphs in CDR. The research in CDR has been investigated from both the non-deep-learning aspect and deep-learning aspect.

Non-deep-learning based CDR: Early CDR methods mainly concentrate on neighborhood-based solutions [81]. However, the neighborhood-based CDR methods tend to detect localized relationships and fail to capture the totality of weak signals implied by the user's whole interactions [82]. Consequently, these methods are dominated by MF-based [82-84] and clustering-based [85-86] methods. For example, collective matrix factorization (CMF) [83] factorizes multiple user-item interaction matrices from different domains by sharing the user latent factor. CCCFNet [84] combines collaborative filtering and content-based filtering into one unified matrix factorization framework. Considering clustering is one practical technique to alleviate the sparsity problem in single domain recommendation [87-88], cluster-level matrix factorization [85] leverages K-means to capture the shared patterns between the cluster of users and the cluster of items from different domains. Although the neighborhood-based, MF-based and clustering-based methods have achieved promising results, they are limited for CDR due to the following reasons [89]. First, most models are linear, which fail to extract the complex patterns in user-item interactions. Second, the relatively dense information of other domains is required to augment the target domains.

Deep-learning based CDR: Owing to the superior representation learning ability of deep learning, many deep-learning based CDR methods [84, 89-92] have been proposed. Deep learning in CDR not only explores how to capture the complex patterns in user-item interactions, but also pursues a more effective way to transfer knowledge across domains [93]. Inspired by the concept of modeling data in semantic space of Deep Structured Semantic models (DSSM) [94], Elkahky et al. [90] proposed a deep learning approach to project users and items in a shared semantic space, and recommend items that have maximum similarity with users in that space. CoNet [91] employs deep cross connection network to transfer knowledge between user-item interactions from different domains. PPGN [95] propagates user preferences with graph neural networks in CDR. Motivated by dual learning [96], Li et al. [97] introduced a deep dual transfer

network named DDTCDR to enhance the bidirectional knowledge in CDR. Moreover, an orthogonal mapping is used in DDTCDR to extract user preferences across domains while preserving relations between users in the latent space. Inspired by domain adaptation [98], [92] and [89] employ domain adaptation techniques to perform knowledge transfer by learning the overlapped features of user preferences. Further, some researchers point out that the domain-specific features of user preferences are also important in CDR. They employ MLP as the mapping function across domains to offer learning flexibility for user representations in each domain [42, 45-46]. However, although MLP increases the learning flexibility, it is easier to have the over-fitting problem for the MLP due to the data sparisity in recommendation [42]. Correspondingly, EMCDR [42] proposes to choose users with sufficient behaviors for learning the MLP. DCDCSR [45] defines a metric to measure the sparse degree and incorporates the sparse degree when learning the representation mapping across domains. ATLRec [99] employs different MLP functions to learn the domain-shareable and domain-specific features by an adversarial transfer learning based scheme. DCDIR [46] employs additional knowledge graphs to learn sparsity-insensitive representations for the mapping.

In summary, previous methods [83-84, 89, 98] are mainly based on the idea of shared-user representation, and focus on learning the overlapped features. In order to further capture the domain-specific features, recent works mainly employ MLP as the mapping function, and train the MLP with selected user behaviors to avoid the over-fitting problem [42]. This manner usually requires heuristic knowledge and may introduce human-bias when selecting samples.

2.4 Advances in Deep Generative Models

Deep generative modeling targets to capture the inner probabilistic distribution that generates the data. Variational auto-encoding (VAE) [100] and generative adversarial networks (GAN) [101] are two popular and representative deep generative methods.

VAE first emerged in [100] where the authors aim to perform efficient inference and learning in directed probabilistic graphic models even with the intractable posteriors. In [100], the authors first derived the variational *ELBO* of the marginal log-likelihood of observed datapoints. Then a reparameterization trick is applied to approximate the intractable posteriors, which also enables VAE to be straightforwardly optimized using standard stochastic gradient based methods. After [100], an enormous amounts of

researchers have studied VAE from different perspectives, which advances the whole community of variational auto-encoding. Recent advances of VAE theory could be categorized into two aspects. First, from more expressive likelihood aspect, the standard VAE [100] makes an assumption that the likelihoods factorizes over dimensions, which may cause poor approximation for tasks involving images. Thereby, Gulrajani et al. [102] proposed to model the dependencies within an image and further developed an auto-regressive decoder in VAE for fine-grained image generation. Besides, there are some works trying to derive more expressive likelihoods from information theory such as [103]. Second, from more expressive posterior aspect, the main idea is that the standard VAE uses mean field approach, which lacks expressiveness for modeling complex posteriors. Thus, IWAE [104] weights the samples in the posterior approximation process, which increases the model's flexibility to capture complex posteriors.

GAN [101] is another representative methodology in deep generative modeling. GAN contains a generator and a discriminator, where the discriminator tries to distinguish the real samples with the fake samples and the generator tries to confuse the discriminator. Adversarial learning in GAN has the advantage of measuring the distribution distance in a more elegant way by a binary classifier and frees researchers from the painful practice of defining a tricky objective function. Several works [101, 105-106] have pointed out that the adversarial loss in GAN actually minimizes the Jensen-Shannon divergence (JSD) between the data distribution and the generator distribution. Original GAN has risk facing the vanishing gradient and mode collapse problem. To handle this, a lot of works have been developed to improve the objective function such as f-GAN [105], LSGAN [107] and WGAN [108].

There are also some works [109-111] trying to combine VAE and GAN theory together. For example, in VAE, the generation quality crucially relies on the expressiveness of the inference model. Thus, Adversarial Variational Bayes (AVB) is proposed in [109] to train VAE with arbitrary inference models. VEEGAN in [110] changes the matching manner from data space to latent space. Meanwhile, Rosca et al. [111] pointed out that VAE based methods fail to match marginal distributions in both latent and visible space while GAN has the potential to overcome these limitations. Thus they develop a VAE-GAN hyprid model to improve and generation quality. Combining VAE and GAN together can employ the advantages and avoid the drawbacks of these two methodologies for better data distribution modeling. The proposed algorithms in this thesis are also

based on these methodologies and will be detailed later.

Chapter 3 Learning on Signed Directed Networks

Many graphs have both directed and signed (*i.e.* positive and negative) links, such as Epinions[1] and Slashdot[2], which are called signed directed networks. Negative links in social networks hold opposite semantic meaning and contain additional information [34-36] that helps many tasks, e.g. link sign prediction and node classification. In addition, link direction indicates the asymmetric relationship between two nodes, which is important for node (*i.e.* user) recommendation in social media [37-38]. For example, stars may not follow common people while common people tend to follow stars. As discussed in Chapter 1, learning on signed directed networks is not easy since the structures with coupled signs and directions are rather difficult to model.

To overcome this difficulty, in this Chapter, we propose to decouple the modeling of signs and directions in signed directed networks. Moreover, in order to bound the optimization of this decoupling idea, we reformulate the representation learning problem from a variational auto-encoding perspective and derive an *ELBO* as a surrogate objective function for learning. Thanks to this surrogate objective function, we are able to capture the complex patterns of structures and learn more representative node embeddings for downstream tasks. Extensive experiments on three widely used real-world datasets have confirmed the effectiveness of the proposed model.

3.1 Introduction

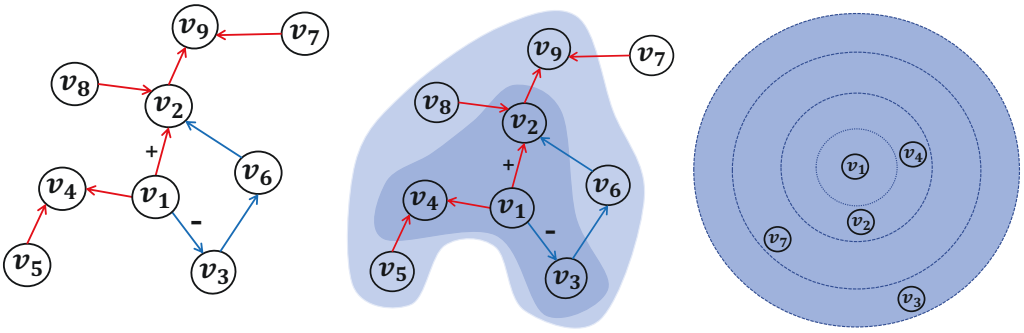
In signed directed networks, it is challenging to encode the topological information into low-dimensional node embeddings. The topological information is composed of both the *high-order* and the *first-order* topology. The *high-order* topology indicates the local structures that are formed by information propagation of a node's neighbours and the *first-order* topology indicates the closeness relationships between a node and its directly linked neighbours. To make it more explicitly, we give an example in Figure 3–1.

However, existing embedding methods usually fail to well capture the *first-order* and *high-order* structures. Firstly, the majority of them concentrate on how to mine the *first-*

[1] http://www.epinions.com/?sb=1

[2] https://slashdot.org/

order topology, namely preserving the closeness relationships of nodes. For example, MF [75] performs matrix factorization on the signed directed adjacency matrix to learn low-dimensional node embeddings. SNE [77] exploits random walk and log-bilinear model to learn node embeddings with signed links. SiNE [59] learns node embeddings through a deep neural network model based on social theory. They model the closeness relationships in restrictive distance metrics or usually ignore the additional value of non-existent links. Secondly, the *high-order* topology, indicating the local structures of nodes, is difficult to be extracted because of the coupled signs and directions. Different signs and directions have distinctive information propagation influence. SNE [77] with random walk applies homophily effects on different signs and fail to capture the *high-order* patterns. How to encode both the *high-order* and *first-order* topological information is an important problem in signed directed networks.



(a) An example of signed directed networks $\mathcal{G}_e$ (b) The *high-order* topology (c) The *first-order* topology

Figure 3–1 An example to illustrate the *high-order* topology and *first-order* topology in signed directed networks. Red arrows mean positive directed edges and blue arrows indicate negative directed edges. (a) is an example of signed directed networks $\mathcal{G}_e$. (b) indicates the *high-order* topology of node v_1 in $\mathcal{G}_e$, namely the local structures of v_1. Different depth of shades represent different orders of structures for v_1. (c) shows the *first-order* topology of v_1, namely the closeness relationships between v_1 and its directly linked neighbours. The concentric circles with v_1 as the center indicate the closeness between v_1 and its positively linked nodes v_2, v_4, the non-linked node v_7 and the negatively linked node v_3.

In this Chapter, we propose to simultaneously capture the *first-order* and *high-order* topology by decoupling the effects of signs and directions in signed directed networks. In particular, we reformulate the representation learning on signed directed

networks from a variational auto-encoding perspective and further propose a decoupled variational embedding (DVE) method. DVE is a specially designed variational auto-encoding model that contains a decoupled variational encoder and a structure decoder. In the decoupled variational encoder, the representation of a node is decoupled into source node embeddings and target node embeddings according to the link direction. Both the source node embeddings and the target node embeddings contain the local structures that are extracted by graph convolutions on the decoupled positive and negative graph according to the link sign. The structure decoder is formulated as a novel Balance Pair-wise Ranking (BPWR) loss that is developed from the Extended Structural Balance Theory [112-113]. BPWR extracts the closeness relationships among positive links, negative links and non-existent links in a Bayesian personalized ranking manner, as well as refines embeddings learned from the former encoder. The auto-encoding formulation encourages DVE to preserve the network topology in an end-to-end manner. In brief, the contributions of this Chapter are summarized as follows:

- We propose a variational auto-encoding based method named DVE to learn more representative node embeddings for signed directed networks. To the best of our knowledge, DVE is the first model that simultaneously models both the *first-order* and the *high-order* topology in signed directed networks;
- Based on Extended Structural Balance Theory, we develop a novel Balance Pair-wise Ranking (BPWR) loss that also works as the decoder in DVE. BPWR mines the mediator value of non-existent links between positive and negative links;
- We conducted extensive experiments on three real-world datasets. The comparison results illustrate the superiority of DVE both quantitatively and qualitatively.

3.2 DVE: Decoupled Variational Embedding

In this section, we first give the problem definition and then demonstrate how the decoupling idea works and what are the specific components of the proposed algorithm.

3.2.1 Problem Definition

A signed directed network is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}^p, \mathcal{E}^n)$, where $\mathcal{V}$ is the set of all nodes and $\mathcal{E}^p$ (*resp.* $\mathcal{E}^n$) denotes positive (*resp.* negative) links. Let $\mathcal{E} = \mathcal{E}^p \bigcup \mathcal{E}^n$ be the observed links in $\mathcal{G}$. For each link $e \in \mathcal{E}$, it is denoted as $e_{u\to v} = (u, v, \epsilon_{u\to v})$, where $u \to v$ denotes the direction from source node u to target node v. And $\epsilon_{u\to v}$ indicates the

sign value of link $e_{u\to v}$, *i.e.* $\epsilon_{u\to v} = 1$ if $e_{u\to v} \in \mathcal{E}^p$ or $\epsilon_{u\to v} = -1$ if $e_{u\to v} \in \mathcal{E}^n$. When the nodes $\mathcal{V}$ have raw features, the feature matrix is denoted as $X \in \mathbb{R}^{N\times F}$, where F indicates the feature dimension. Given $\mathcal{G}$, the objective of node representation learning on signed directed networks is to embed nodes into low-dimensional embeddings $Z \in \mathbb{R}^{N\times d}$ that benefit downstream tasks such as node recommendation and link prediction.

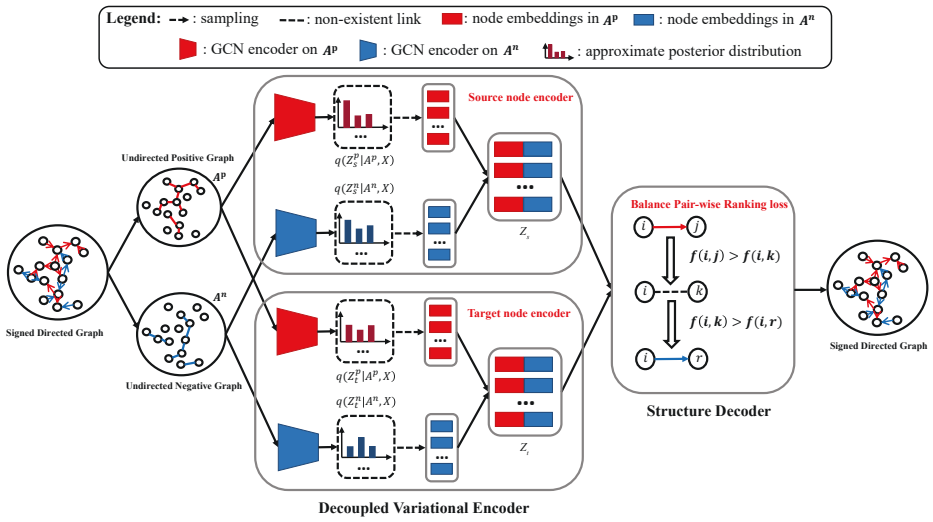


Figure 3–2 The model architecture of DVE. We first decouple the signed directed graph into an undirected positive graph whose adjacency matrix is A^p and an undirected negative graph whose adjacency matrix is A^n. Then our decoupled variational encoder encodes A^p and A^n as the source node representation Z_s and target node representation Z_t, respectively. Finally, Z_s and Z_t are used to perform the balance pair-wise ranking loss which is also the structure decoder in DVE. Node i is the source node and from Z_s, node. Nodes j, k, r are the target nodes and from Z_t. $f(\cdot,\cdot)$ indicates the score of two nodes connecting with positive links.

3.2.2 Variational Auto-Encoding Formulation

In this part, we start to illustrate the proposed decoupled variational embedding (DVE) model whose architecture is shown in Figure 3–2. Link direction and sign are two key elements when describing signed directed networks. Link direction between two nodes indicates the asymmetric relationship that implies the different roles of two nodes in an interaction. This asymmetric information is an essential factor that facilitates information propagation in signed directed networks. However, it is inappropriate to apply some GNN methods such as [1, 14, 25] on directed graphs since they require a symmetric Laplacian matrix for graph convolutions. As a node in a directed relationship

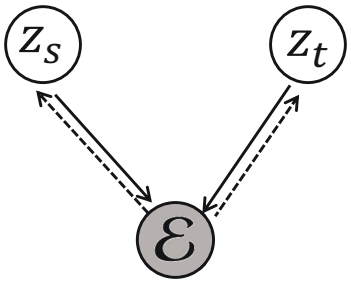


Figure 3–3 The graphical model of our the decoupling idea on link direction. In this figure, Z_s and Z_t are the latent variables for the source node representation and target node representation, respectively. $\mathcal{E}$ indicates the observed signed directed edges. Solid arrows denote the generative process and dashed arrows denote the inference process.

may both be the source node and target node, we thus try to leverage the asymmetric information by decoupling node embeddings into source node embeddings Z_s and the target node embeddings Z_t that are independent when given the observed edges $\mathcal{E}$. The graphical model of this decoupling idea is shown in Figure 3–3. Further, to bound the optimization of this decoupling idea, we formulate the node representation learning from a variational auto-encoding perspective. To be specific, we assume the edges are drawn from some underlying distributions. To clarify, we denote θ as the parameter symbol for all non-specified models. The probability distribution function of all observed signed directed links $\mathcal{E}$ is denoted as $P(\mathcal{E})$ and it can be written as Eq. 3–1:

$$P(\mathcal{E}) = \int_{Z_s, Z_t} p_\psi(\mathcal{E}|Z_s, Z_t) p_\theta(Z_s, Z_t) dZ_s dZ_t \tag{3–1}$$

where Z_s and Z_t also indicate the latent variables of source nodes and target nodes respectively. By modeling node embeddings through two different latent variables, the asymmetric relationship can be well captured. The true posterior distribution of Z_s, Z_t can be written as:

$$p_\theta(Z_s, Z_t|\mathcal{E}) = \frac{p_\psi(\mathcal{E}|Z_s, Z_t) p_\theta(Z_s, Z_t)}{p_\theta(\mathcal{E})} \tag{3–2}$$

where $p_\theta(\mathcal{E})$ denotes the probability density function of $\mathcal{E}$. The true posterior $p_\theta(Z_s, Z_t|\mathcal{E})$ in Eq. 3–2 is intractable because of the moderately complicated likelihood function of $p_\psi(\mathcal{E}|Z_s, Z_t)$ such as a neural network with non-linear layer [100, 114-115]. We thus introduce a tractable posterior $q_\phi(Z_s, Z_t|\mathcal{E})$ to approximate $p_\theta(Z_s, Z_t|\mathcal{E})$. In this case, the marginal log-likelihood $\log P(\mathcal{E})$ can be rewritten as:

$$\log P(\mathcal{E}) = D_{KL}[q_\phi(Z_s, Z_t|\mathcal{E})||p_\theta(Z_s, Z_t|\mathcal{E})] + \mathcal{L} \tag{3–3}$$

where D_{KL} means the Kullback-Liebler (KL) divergence and $\mathcal{L}$ is the variational *ELBO* of $\log P(\mathcal{E})$. Since the KL divergence term is non-negative, we can maximize the log-likelihood $\log P(\mathcal{E})$ by maximizing $\mathcal{L}$. Denoting the joint prior for Z_s and Z_t as $p_\theta(Z_s, Z_t)$, $\mathcal{L}$ is derived as:

$$\mathcal{L} = -D_{KL}[q_\phi(Z_s, Z_t|\mathcal{E})||p_\theta(Z_s, Z_t)] + \mathop{\mathbb{E}}_{q_\phi(Z_s,Z_t|\mathcal{E})}[p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{3–4}$$

where $p_\psi(\mathcal{E}|Z_s, Z_t)$ indicates the probabilistic decoder parameterized by ψ. The derivation of this *ELBO* is written as: In our method, we simplify the joint prior $p_\theta(Z_s, Z_t)$ by assuming $p_\theta(Z_s, Z_t) = p_\theta(Z_s)p_\theta(Z_s)$. Complex prior is a specific research topic in variational inference [116-118]. We do not explore more here since we mainly focus on the general variational auto-encoding idea for modeling the topological information in signed directed networks. According to the graphical model in Figure3–3, the latent variable Z_s and Z_t are conditional independent when given the observed links $\mathcal{E}$, with which we rewrite the approximate posterior $q_\phi(Z_s, Z_t|\mathcal{E})$ as:

$$q_\phi(Z_s, Z_t|\mathcal{E}) = q_{\phi_s}(Z_s|\mathcal{E})q_{\phi_t}(Z_t|\mathcal{E}) \tag{3–5}$$

where $q_{\phi_s}(Z_s|\mathcal{E})$ and $q_{\phi_t}(Z_t|\mathcal{E})$ are the approximate posteriors parameterized by ϕ_s and ϕ_t respectively. If we denote $p_\theta(Z_s)$ and $p_\theta(Z_t)$ are the prior for Z_s and Z_t respectively, we rewrite the *ELBO* in Eq. 3–4 as:

$$\mathcal{L} = -D_{KL}[q_{\phi_s}(Z_s|\mathcal{E})||p_\theta(Z_s)] - D_{KL}[q_{\phi_t}(Z_t|\mathcal{E})||p_\theta(Z_t)] + \mathop{\mathbb{E}}_{\substack{q_{\phi_s}(Z_s|\mathcal{E}) \\ q_{\phi_t}(Z_t|\mathcal{E})}}[\log p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{3–6}$$

More detailed derivation is provided is Appendix A.1. DVE tries to learn Z_s and Z_t via maximizing the above *ELBO*. To better understand DVE, we firstly introduce the two variational approximate posteriors $q_{\phi_s}(Z_s|\mathcal{E})$ and $q_{\phi_t}(Z_t|\mathcal{E})$. Modeling these two distributions also indicates the decoupled variational encoder in Figure 3–2. The conditional distribution $p_\psi(\mathcal{E}|Z_s, Z_t)$ which indicates the structure decoder, will be discussed later.

3.2.3 Decoupled Variational Encoder

In this part, how the decoupled variational encoder works is introduced. In our expectation, Z_s and Z_t are the representation for the source node and target node respectively. These two representations should capture the local structures indicated by the positive links and negative links. Take the source node representation Z_s as an example,

directly learning Z_s through existing GCN methods is not appropriate, because this makes GCN do homophily effects on different signs. Instead, we decouple the signed directed graph into an undirected positive graph and an undirected negative graph, and consider that Z_s could be generated by the node representation Z_s^p involved in the undirected positive graph and the node representation Z_s^n involved in the undirected negative graph. In other words, Z_s is represented as $Z_s = f_s(Z_s^p, Z_s^n)$, where f_s is the generative function. A proper choice of f_s can capture the interactions between positive and negative links.

In the learning process of Z_s^p and Z_s^n, if we denote A^p and A^n as the adjacency matrix of the undirected positive graph and the undirected negative graph, respectively, variational GCN is applied on A^p and A^n. In this context, $Z_s \sim q_{\phi_s}(Z_s|\mathcal{E})$ can be denoted by the combination of $Z_s^p \sim q_{\phi_s^p}(Z_s^p|A^p, X)$, $Z_s^n \sim q_{\phi_s^n}(Z_s^n|A^n, X)$ and f_s, where $q_{\phi_s^p}(Z_s^p|A^p, X)$ and $q_{\phi_s^n}(Z_s^n|A^n, X)$ indicate the approximate posterior distribution for the source node involved in the undirected positive graph and undirected negative graph, respectively. We set f_s as concatenation operation here for simplicity. Note that the adjacent matrices of both the undirected positive graph and the undirected negative graph are composed of 0 and 1, where 1 means linked and 0 otherwise. The variational inference procedure for $q_{\phi_s}(Z_s|\mathcal{E})$ indicates the source node encoder shown in Figure 3–2 and is introduced in the following part.

Let the node feature matrix be $X \in \mathbb{R}^{N\times F}$ where N is the number of nodes and F is the feature dimension. Let $Z_{s,i}^p \in \mathbb{R}^{1\times d}$ and $Z_{s,i}^n \in \mathbb{R}^{1\times d}$ be the source node embeddings of i-th node involved in the undirected positive graph and the undirected negative graph, respectively. Then, we can have:

$$q_{\phi_s^p}(Z_s^p|A^p, X) = \prod_{i=1}^{N} q_{\phi_s^p}(Z_{s,i}^p|A^p, X) \tag{3–7}$$

$$q_{\phi_s^n}(Z_s^n|A^n, X) = \prod_{i=1}^{N} q_{\phi_s^n}(Z_{s,i}^n|A^n, X) \tag{3–8}$$

Inspired by the idea that different semantics can come from the same family of functions (*e.g.* Gaussian) since these semantics are modeled by different parameters and are in different spaces [100, 119-120]. We assume that both $q_{\phi_s^p}(Z_s^p|A^p, X)$ and $q_{\phi_s^n}(Z_s^n|A^n, X)$ follow Gaussian distribution, then the reparametrization Gaussian parameters $\mu_s^{p,l} \in$

$\mathbb{R}^{N\times d}, \sigma_s^{p,l} \in \mathbb{R}^{N\times d}, \mu_s^{n,l} \in \mathbb{R}^{N\times d}, \sigma_s^{n,l} \in \mathbb{R}^{N\times d}$ in l-th layer are defined as[1]:

$$\begin{cases} \mu_s^{p,l+1} = \widetilde{A}^p H_{s,\mu}^{p,l} W_{s,\mu}^{p,l},\ H_{s,\mu}^{p,l} = h(\mu_s^{p,l}),\ H_{s,\mu}^{p,0} = X \\ \log \sigma_s^{p,l+1} = \widetilde{A}^p H_{s,\sigma}^{p,l} W_{s,\sigma}^{p,l},\ H_{s,\sigma}^{p,l} = h(\log \sigma_s^{p,l}),\ H_{s,\sigma}^{p,0} = X \end{cases} \tag{3–9}$$

$$\begin{cases} \mu_s^{n,l+1} = \widetilde{A}^n H_{s,\mu}^{n,l} W_{s,\mu}^{n,l},\ H_{s,\mu}^{n,l} = h(\mu_s^{n,l}),\ H_{s,\mu}^{n,0} = X \\ \log \sigma_s^{n,l+1} = \widetilde{A}^n H_{s,\sigma}^{n,l} W_{s,\sigma}^{n,l},\ H_{s,\sigma}^{n,l} = h(\log \sigma_s^{n,l}),\ H_{s,\sigma}^{n,0} = X \end{cases} \tag{3–10}$$

where $\widetilde{A}^p = [\widehat{D}^p]^{-\frac{1}{2}}(A^p + I_N)[\widehat{D}^p]^{-\frac{1}{2}}$ and $\widetilde{A}^n = [\widehat{D}^n]^{-\frac{1}{2}}(A^n + I_N)[\widehat{D}^n]^{-\frac{1}{2}}$ are the propagation matrices. $\widehat{D}^p$ and $\widehat{D}^n$ are the degree matrices of $A^p + I_N$ and $A^n + I_N$, respectively. $h(\cdot)$ denotes the non-linear $Relu$ function. $W_{s,\mu}^{p,l} \in \mathbb{R}^{F\times d}$ and $W_{s,\sigma}^{p,l} \in \mathbb{R}^{F\times d}$ denote the l-layer reparametrization parameters for Z_s^p. Similarly, $W_{s,\mu}^{n,l} \in \mathbb{R}^{F\times d}$ and $W_{s,\sigma}^{n,l} \in \mathbb{R}^{F\times d}$ are the l-layer reparametrization parameters for Z_s^n. If we denote $p_\theta(Z_s^p)$ and $p_\theta(Z_s^n)$ are prior distributions for Z_s^p and Z_s^n respectively, the prior regularization loss on $q_{\phi_s^p}(Z_s^p|A^p, X)$ and $q_{\phi_s^n}(Z_s^n|A^n, X)$ are written as:

$$\min_{\phi_s} L_{KL}^s = D_{KL}[q_{\phi_s^p}(Z_s^p|A^p, X)||p_\theta(Z_s^p)] + D_{KL}[q_{\phi_s^n}(Z_s^n|A^n, X)||p_\theta(Z_s^n)] \tag{3–11}$$

where $\phi_s = \{\phi_s^p, \phi_s^n\} = \{W_{s,\mu\ or\ \sigma}^{p,l_0\ or\ l_1}, W_{s,\mu\ or\ \sigma}^{n,l_0\ or\ l_1}\}$ is the parameter of the source node encoder. Therefore, the source node representation Z_s is $Z_s = Z_s^p \oplus Z_s^n$ ($\oplus$ means concatenation), where $Z_s^p \sim q_{\phi_s^p}(Z_s^p|A^p, X)$ and $Z_s^n \sim q_{\phi_s^n}(Z_s^n|A^n, X)$. The target node representation Z_t can be computed in similar procedure.

It is worthwhile to highlight that GCN working on both A^p and A^n with different parameters models the distinctive effects of different link signs. Conducting GCN on A^p and A^n is reasonable since GCN does not specify positive or negative meaning of links in graphs. Instead, GCN emphasizes the correlation that links two nodes. How to leverage the information from A^p and A^n in the subsequent modules determines the positive or negative semantics. In our case, we use GCN to summarize the correlation patterns among nodes, and then ask the following loss to determine the positive semantics in A^p and negative semantics in A^n. By this way, the local structures in signed directed networks can be modeled in a decoupled manner. In addition, splitting the graph into a positive and a negative graph is one way to decouple the effects of signed directed edges but may not the best way. We will explore the how to model the signed directed edges in the future.

[1] A quick note: s means the source node, μ, σ denote the mean value and standard deviation parameter of Gaussian distribution, p means undirected positive graph and n indicates undirected negative graph.

3.2.4 Structure Decoder

In auto-encoding theory, decoder is an essential module and we introduce our structure decoder here. The structure decoder is expected to reconstruct the signed directed links and guide the encoder learning. This requires that the structure decoder should preserve the structural characteristics in signed directed networks. Note that the Extended Structural Balance Theory [112] states the closeness of users in signed networks. The essential insight of this theory is that for four users i, j, k, r, if the link signs are $\epsilon_{ij} = 1, \epsilon_{ik} = 0, \epsilon_{ir} = -1$, the closeness among them follows Eq. 3–12.

$$g(i, j) < g(i, k) < g(i, r) \tag{3–12}$$

where $g(i, j)$ denotes distance between user i and j. For example, if a positive link means trust and a negative link means distrust in social networks, user i prefers to trust j than k and trusts k more than r. Actually, this theory states the *first-order* topology that indicates the closeness relationships among nodes. By combining this theory with Bayesian Personalized Ranking [121], we naturally develop a novel Balance Pair-wise Ranking (BPWR) loss to guide the whole model learning. To clarify, we denote the distance in Eq. 3–12 as the score of two nodes building positive links. The higher score is, the more probably the positive link exists. Thus, the Extended Structural Balance Theory can be interpreted as Eq. 3–13:

$$f(i \to j) > f(i \to k) > f(i \to r) \tag{3–13}$$

where $f(i \to j)$ indicates the score of building positive links from source node i to target node j. Given node i as the reference object, we use $j >_i k$ to indicate the score of $i \to j$ is larger than that of $i \to k$ for samples (i, j, k, r) where $\epsilon_{i\to j} = 1, \epsilon_{i\to k} = 0, \epsilon_{i\to r} = -1$. The maximum posteriors satisfy:

$$\begin{cases} \max\limits_{\phi_s, \phi_t} \prod\limits_{(i,j,k)} p(\phi_s, \phi_t | j >_i k) \propto \prod\limits_{(i,j,k)} p(j >_i k | \phi_s, \phi_t) p(\phi_s, \phi_t) \\ \max\limits_{\phi_s, \phi_t} \prod\limits_{(i,k,r)} p(\phi_s, \phi_t | k >_i r) \propto \prod\limits_{(i,k,r)} p(k >_i r | \phi_s, \phi_t) p(\phi_s, \phi_t) \end{cases} \tag{3–14}$$

where ϕ_s and ϕ_t are the parameters of the decoupled variational encoder to obtain Z_s and Z_t. $p(j >_i k | \phi_s, \phi_t)$ and $p(k >_i r | \phi_s, \phi_t)$ indicate the likelihood functions which are written as:

$$\begin{cases} p(j >_i k | \phi_s, \phi_t) = \sigma(f(i \to j) - f(i \to k)) \\ p(k >_i r | \phi_s, \phi_t) = \sigma(f(i \to k) - f(i \to r)) \end{cases} \tag{3–15}$$

where $f(i \rightarrow j)$ is calculated by the inner product of the source node embedding $Z_{s,i}$ of node i and the target node embedding $Z_{t,j}$ of node j. $f(i \rightarrow k)$ and $f(i \rightarrow r)$ can be obtained in similar way. σ is the sigmoid function. Following [121], the Balance Pair-wise Ranking (BPWR) loss of our structure decoder can be written as:

$$\begin{aligned} \min_{\phi_s, \phi_t} \quad L_{BPWR} = & - \mathbb{E}_{(i,j,k)\sim P(\mathcal{E})} \ln \sigma(Z_{s,i}^T Z_{t,j} - Z_{s,i}^T Z_{t,k}) \\ & - \mathbb{E}_{(i,k,r)\sim P(\mathcal{E})} \ln \sigma(Z_{s,i}^T Z_{t,k} - Z_{s,i}^T Z_{t,r}) \end{aligned} \tag{3–16}$$

When Z_s and Z_t are not learned from the decoupled variational encoder, Z_s and Z_t can be initialized trainable embedding matrices. In other words, BPWR can be an independent model to learn node embeddings in signed directed networks.

3.2.5 Model Learning

Putting the encoder and decoder together, we can write the objective function of DVE as follows[1]:

$$\begin{aligned} \min_{\phi_s, \phi_t} \quad L_{DVE} = & - \mathbb{E}_{(i,j,k)\sim P(\mathcal{E})} \ln \sigma(Z_{s,i}^T Z_{t,j} - Z_{s,i}^T Z_{t,k}) \\ & - \mathbb{E}_{(i,k,r)\sim P(\mathcal{E})} \ln \sigma(Z_{s,i}^T Z_{t,k} - Z_{s,i}^T Z_{t,r}) \\ & + \frac{1}{N} \sum_{i=1}^{N} \{D_{KL}[q_{\phi_s^p}(Z_{s,i}^p | A^p, X) || p_\theta(Z_s^p)] + D_{KL}[q_{\phi_s^n}(Z_{s,i}^n | A^n, X) || p_\theta(Z_s^n)]\} \\ & + \frac{1}{N} \sum_{i=1}^{N} \{D_{KL}[q_{\phi_t^p}(Z_{t,i}^p | A^p, X) || p_\theta(Z_t^p)] + D_{KL}[q_{\phi_t^n}(Z_{t,i}^n | A^n, X) || p_\theta(Z_t^n)]\} \end{aligned} \tag{3–17}$$

where $\phi_s = \{\phi_s^p, \phi_s^n\} = \{W_{s,\mu \text{ or } \sigma}^{p,l_0 \text{ or } l_1}, W_{s,\mu \text{ or } \sigma}^{n,l_0 \text{ or } l_1}\}$ is the parameter of the source node encoder and $\phi_t = \{\phi_t^p, \phi_t^n\} = \{W_{t,\mu \text{ or } \sigma}^{p,l_0 \text{ or } l_1}, W_{t,\mu \text{ or } \sigma}^{n,l_0 \text{ or } l_1}\}$ denotes the parameter of the target node encoder. The source node embeddings and target node embeddings are respectively denoted as $Z_s = Z_s^p \oplus Z_s^n$, $Z_t = Z_t^p \oplus Z_t^n$. All priors $p_\theta(Z_s^p)$,$p_\theta(Z_s^n)$,$p_\theta(Z_t^p)$ and $p_\theta(Z_t^n)$ are standard Gaussian distributions. In DVE, the matrix multiplication is conducted between a sparse adjacency matrix and a dense matrix, *e.g.* Eq. 3–9,3–10, which can be implemented with high efficiency in recent deep learning programming frameworks. For each positive link $e_{i\rightarrow j}$ and negative link $e_{i\rightarrow r}$, we randomly sample n_{noise} non-linked nodes to play as k and construct the training quadruples (i, j, k, r). We adopt

[1]Note that we take the expectation formula here for scaled loss values instead of summarization.

Dropout technique for regularization. Many widely used optimization algorithms such as RMSProp can be applied for model learning. A concise description of DVE is depicted in Algorithm 3–1.

3.2.6 Comparison Between DVE and Existing Methods

There are differences and connections between DVE and existing methods. Most existing methods [59, 75, 80] focus on modeling the *first-order* topology. Some works [77-78] are trying to learn the *high-order* topology but usually apply homophily effects of different signs or directions. By contrast, DVE integrally models both the *first-order* and *high-order* topology and is aware of the different effects of signs and directions by a decoupling idea.

Meanwhile, there are connections between DVE and existing methods in terms of the *first-order* topology modeling. Both DVE and existing methods perform the *first-order* topology modeling regarding signed directed links. Existing methods usually ignore the mediator value of non-existent links. Instead, BPWR models the non-existent links and has more potential to capture the closeness relationships of nodes. It is also worthwhile to point out that both the margin ranking (MR) loss in SNEA [80] and BPWR loss in DVE have similar target that are developed from Extended Structural Balance Theory. However, MR is a deterministic non-smooth metric and BPWR maximizing the posterior of the observations is smooth and easy to be optimized by gradient based techniques [121]. The superior performance of BPWR over SNEA-MR is also verified in Section 3.3.4.

3.2.7 Time Complexity Analysis

Stochastic training of DNN methods involves two steps, the forward and backward computations. DVE supports the mini-batch training and the time cost lies in the decoupled variational encoder and structure decoder. We thus decompose the time complexity of DVE into two parts, namely the time complexity of decoupled variational encoder and structure decoder. In each batch of DVE, the decoupled variational encoder learns node embeddings for all nodes. Thus, following the analysis of GCN in [48], the time complexity of decoupled variational encoder is $O(2|\mathcal{E}^p|+2|\mathcal{E}^n|) = O(2|\mathcal{E}|)$, where $|\mathcal{E}^p|$, $|\mathcal{E}^p|$ and $|\mathcal{E}|$ denote the number of edges of undirected positive graph, undirected negative graph and signed directed graph, respectively. Note that for this decoupled variational encoder, the source node encoder and target node encoder can be parallelly conducted.

Algorithm 3–1 Decoupled Variational Embedding (DVE)

Input: A signed directed adjacency matrix $A \in \mathbb{R}^{N\times N}$; A node feature matrix $X \in \mathbb{R}^{N\times F}$, X could be the identity matrix I_N if there is no feature; the training quadruples (i, j, k, r) where $e_{i\to j}$ is positive, $e_{i\to r}$ is negative and $e_{i\to k}$ is non-existent.

Output: Source node embeddings $Z_s \in \mathbb{R}^{N\times d}$ and target node embeddings $Z_t \in \mathbb{R}^{N\times d}$.

1: Decouple signed directed graph A into an undirected positive graph A^p and an undirected negative graph A^n, where both A^p and A^n are composed of 0,1.
2: Calculate the symmetric propagation matrices $\widetilde{A^p}$ and $\widetilde{A^n}$.
3: **while** not converged **do**
4: # source node encoder.
5: Calculate th parameters μ_s^p, σ_s^p, μ_s^n, σ_s^n defined in Eq 3–9,3–10.
6: # target node encoder.
7: Calculate reparametrization parameters μ_t^p, σ_t^p, μ_t^n, σ_t^n that share similar definition in Eq 3–9,3–10 but with different parameters.
8: Calculate target node embeddings $Z_t = Z_t^p \oplus Z_t^n$, $Z_t^p \sim \mathcal{N}(\mu_t^p, \sigma_t^p)$, $Z_t^n \sim \mathcal{N}(\mu_t^n, \sigma_t^n)$.
9: For training quadruples (i, j, k, r), lookup source node embeddings $Z_{s,i}$ for node i, target node embeddings $Z_{t,j}$, $Z_{t,k}$ and $Z_{t,r}$ for node j, k, r respectively;
10: Apply an optimization method to update model parameters based on the loss defined in Eq. 3–17.
11: **end while**

In this case, the time complexity for decoupled variational encoder can be reduced to $O(|\mathcal{E}^p| + |\mathcal{E}^n|) = O(|\mathcal{E}|)$. Furthermore, the graph convolution on A^p and A^n to learn Z_s and Z_t could also be parallel, which leads to the time complexity as $O(\max\{|\mathcal{E}^p|, |\mathcal{E}^n|\})$. When considering the structure decoder in each batch, we compute the BPWR loss by sampling non-existent links. If we denote the sampling size as n_{noise} and the batch size of training positive or negative links as B, the time complexity is $O(n_{noise}B)$.

In summary, the time complexity of the non-parallel DVE in each batch is $O(2|\mathcal{E}|+n_{noise}B)$, the half-parallel counterpart is $O(|\mathcal{E}|+n_{noise}B)$ and the quarter-parallel counterpart is $O(\max\{|\mathcal{E}^p|, |\mathcal{E}^n|\} + n_{noise}B)$. Generally, the main time cost lies in the decoupled variational encoder, since we usually have $2|\mathcal{E}| > |\mathcal{E}| > \max\{|\mathcal{E}^p|, |\mathcal{E}^n|\} >> n_{noise}B$. Actually, the time complexity of decoupled variational encoder is related to the specific graph convolutional network. For datasets with too large $\max\{|\mathcal{E}^p|, |\mathcal{E}^n|\}$, the

$O(\max\{|\mathcal{E}^p|, |\mathcal{E}^n|\})$ in each batch may still be time-consuming. This can be solved by using other kinds of GCN [2, 65-66] that reduce $O(\max\{|\mathcal{E}^p|, |\mathcal{E}^n|\})$ to the scale of the training batch size B. This makes DVE be scalable to much larger datasets. We do not explore more here since we mainly focus on the general idea of variational auto-encoding to learn the *first-order* and *high-order* structures in signed directed networks.

3.3 Experiments and Analysis

In this section, we conduct experiments on three widely used datasets. Performance on both link sign prediction task and node recommendation task are implemented to verify the effectiveness of DVE. Further ablation study and qualitative analysis are investigated to provide a deep understanding about DVE.

3.3.1 Dataset Description

We conduct the experiments on three widely used real-world datasets. Epinions[1]: Epinions is one popular product review site in which users can create both trust (positive) and distrust (negative) links to others. Slashdot[2] is a technology news platform where users can create friend (*i.e.* positive) and foe (*i.e.* negative) links to others. Wiki[3] is a dataset collected from the Wikipedia site, where users vote for or against other users in order to determine administration promotion. For each dataset, we randomly sample a subset links as our experimental dataset. We also filter out users who have no link with others. The statistics of processed data are shown in Table 3–1. From the table, it is obvious that both the undirected positive graph and the undirected negative graph are quite sparse.

3.3.2 Baselines

We compare DVE with nine competitive baselines.

- LINE [20]: LINE defines loss functions to preserve the first-order or second-order proximity between nodes in a graph. Here, we conduct LINE on the positive links since it does not work on signed graphs. As LINE's first order proximity usually

[1] https://snap.stanford.edu/data/soc-sign-epinions.html

[2] https://snap.stanford.edu/data/soc-sign-Slashdot090216.html

[3] https://snap.stanford.edu/data/wiki-RfA.html

Table 3–1 The statistics of Epinions, Slashdot and Wiki utilized in our experiments.

Dataset	Epinions	Slashdot	Wiki
#nodes	22,503	17,496	6,836
#edges	84,102	54,920	89,365
#positive edges	60,044	46,189	70,075
#negative edges	24,058	8,731	19,290
undirected positive graph density	0.0119%	0.0151%	0.1499%
undirected negative graph density	4.75e-3%	2.85e-3%	4.12e-4%

presents better performance than the second order one, the performance of LINE's first-order proximity is reported here.

- MF [75]: Matrix factorization is one popular technique for network embedding. We employ MF with the same noise sampling approach of DVE to learn low-dimensional node embeddings for signed directed networks.
- SNE [77]: This method develops the log-bilinear model with random walks to learn low-dimensional node embeddings for signed networks. On signed directed networks, we use directed random walk for SNE.
- SiNE [59]: SiNE is a deep neural network method that makes a distinction between positively linked nodes and negatively linked nodes. It is capable of capturing the non-linear pattern in signed directed networks.
- SIDE [78]: SIDE is a random walk based model which formulates the social balance theory into a likelihood for signed directed networks.
- SNEA-MR [80]: SNEA is a method for attributed signed social networks. Considering that the margin ranking (MR) loss [1] in SNEA is also based on Extended Structural Balance Theory, we thus extend it here as a baseline to make a comparison with BPWR.
- BPWR (Ours): As the structure decoder of our DVE model, this Balance Pairwise Ranking loss could be an independent model and be a comparison to the loss in SiNE and SNEA-MR.
- SLVE (Ours): SLVE substitutes the decoupled variational encoder in DVE with a non-decoupled one by leveraging the signed Laplacian matrix [122].
- DE (Ours): DE is the non-variational variant of DVE method.

[1] It refers to Eq.5 in the original paper.

Table 3–2 Link sign prediction performance. Names with * refer to our methods. Compared to SiNE, the absolute improvement percentage of DE and DVE are given.

Dataset	Epinions		Slashdot		Wiki	
Method	AUC	F1	AUC	F1	AUC	F1
LINE	0.906	0.902	0.855	0.920	0.782	0.889
MF	0.934	0.927	0.801	0.918	0.595	0.884
SNE	0.952	0.933	0.869	0.928	0.848	0.902
SiNE	0.929	0.919	0.870	0.916	0.864	0.904
SIDE	0.807	0.861	0.798	0.917	0.647	0.884
SNEA-MR	0.864	0.891	0.753	0.913	0.732	0.889
BPWR*	0.933	0.926	0.891	0.929	0.881	0.911
SLVE*	0.924	0.918	0.871	0.922	0.874	0.905
DE*	0.958(+2.9%)	0.939(2.0%)	0.899(2.9%)	0.930(1.4%)	0.885(2.1%)	0.909(0.5%)
DVE*	**0.960(3.1%)**	**0.940(2.1%)**	**0.905(3.5%)**	**0.934(1.8%)**	**0.889(2.5%)**	**0.911(0.7%)**

3.3.3 Experimental Setups

For each baseline, we follow the parameter settings in their papers or codes. Batch training size is 1,000 for all methods. For our model, we set the training epoch size as 200 and the number of GCN layers as $l = 2$. Dropout probability is 0.2. Learning rate is taken as 0.01. RMSProp optimizer [123] is adopted to optimize our objective function. According to the training loss, the size of randomly sampled noise ($e_{i\to k}$=0) is set as 5 for Epinions and 20 for Slashdot and Wiki. Embedding size is $d_1 = 128$ and $d = 64$ on all three datasets. We randomly split each dataset into 80% train data and 20% test data. For every model, we conduct 10 times and report the averaged best performance on test set as the model performance.

3.3.4 Performance Comparison

3.3.4.1 Link Sign Prediction

We first compare the model performance on link sign prediction task. Link sign prediction aims to predict the unobserved signs of existing links. Following the evaluation protocols in existing works [59, 77], we train a binary classifier which is a two-layer MLP with $Relu$ as the non-linear function. Then, we use signed links in the model training stage as the train data for the binary classifier and predict signs for the test links. More specifically, we concatenate two node embeddings as the link representation and take the link representation as input for the binary classifier. Due to the imbalanced signs in test links, AUC and F1 are adopted to make the evaluation. The results are shown in

Table 3–2. From this table, we summarize that:

- The proposed DVE outperforms recent competitive models and achieves the best performance. For example, regarding AUC on Slashdot, DVE obtains a 3.5% improvement compared to SiNE and a 3.6% improvement compared to SNE. This verifies that DVE learns more representative node embeddings in signed directed networks.
- Comparing BPWR with other baselines (SiNE, SNE, MF), BPWR outperforms them on AUC and F1 on all three datasets. This exposes the deficiencies of the baselines in mining the *first-order* topology. Developed from Extended Structural Balance Theory, BPWR working in a personalized pair-wise ranking manner is more capable of mining the closeness relationships among nodes. It is worthwhile to point out that although SNEA-MR and BPWR are both based on the Extended Structural Balance Theory and have a similar training goal. However, the objective of SNEA-MR is not smooth while BPWR based on maximizing the posterior of signed directed links is smooth and easy to be optimized by gradient based optimization methods. The comparison results as well confirm the superiority of BPWR.
- Modeling the *high-order* topology facilitates to learn more representative node embeddings in signed directed networks. Compared to BPWR, DVE and DE with graph convolution are able to model both the *first-order* and the *high-order* topology. However, BPWR works as an independent model can only model the *first-order* topology. The gap is more obvious in the following node recommendation task.
- It is obvious that DVE always outperforms SLVE, which confirms the importance of our decoupling idea. Particularly, SLVE is the non-decoupled variant of DVE by applying signed Laplacian matrix [122] in GCN. Thus the only difference between SLVE and DVE is the encoder part. From the comparison between SLVE and BPWR, we can see that SLVE even damages its own decoder's (BPWR) performance. This highlights the necessity of applying distinctive effects on different types of links in signed directed networks.
- Compared to DE, DVE models the uncertainty of node embeddings in signed directed networks and thus shows better performance than the non-variational DE on Slashdot and Wiki. If provided with a better prior distribution, the performance

Table 3–3 Node recommendation performance on Epinions. Names with $*$ refer to our methods. The metrics for this task are Recall@k and Precision@k. We pick k=10,20,50 here. Compared to SiNE, the absolute improvement of DVE is given in the table.

Dataset	Epinions					
Methods	R@10	R@20	R@50	P@10	P@20	P@50
LINE	0.004	0.011	0.025	0.004	0.004	0.005
MF	0.022	0.036	0.069	0.029	0.025	0.021
SNE	0.002	0.003	0.008	0.001	9.5e-4	9.0e-4
SiNE	0.027	0.039	0.074	0.031	0.026	0.021
SIDE	5.5e-4	8.2e-4	0.002	8.3e-4	7.1e-4	5.9e-4
SNEA-MR	0.024	0.037	0.065	0.024	0.020	0.016
BPWR*	0.031	0.053	0.088	0.026	0.024	0.021
SLVE*	0.0181	0.029	0.070	0.017	0.015	0.012
DE*	0.030	0.045	0.082	0.022	0.020	0.017
DVE*	**0.035(0.8%)**	**0.053(1.4%)**	**0.100(2.6%)**	**0.035(0.4%)**	**0.030(0.4%)**	**0.024(0.3%)**

Table 3–4 Node recommendation performance on Slashdot. Names with $*$ refer to our methods. The metrics for this task are Recall@k and Precision@k. We pick k=10,20,50 here. Compared to SiNE, the absolute improvement of DVE is given in the table.

Dataset	Slashdot					
Methods	R@10	R@20	R@50	P@10	P@20	P@50
LINE	0.011	0.017	0.033	0.006	0.006	0.006
MF	0.015	0.026	0.053	0.025	0.022	0.018
SNE	0.002	0.006	0.011	0.002	0.002	0.002
SiNE	0.052	0.068	0.107	0.027	0.025	0.021
SIDE	8.2e-4	0.001	0.005	6.5e-4	6.5e-4	7.6e-4
SNEA-MR	0.005	0.007	0.014	0.005	0.004	0.003
BPWR*	0.058	0.073	0.111	0.028	0.023	0.020
SLVE*	0.037	0.049	0.089	0.018	0.017	0.016
DE*	0.039	0.057	0.101	0.025	0.022	0.019
DVE*	**0.060(0.8%)**	**0.086(1.8%)**	**0.134(2.7%)**	**0.036(0.9%)**	**0.031(0.6%)**	**0.024(0.3%)**

gap could be more obvious.

3.3.4.2 Node Recommendation

Another practical application in signed directed networks is node recommendation. It matches a fact that friend recommendation in social media. We conduct the node recommendation task here to investigate the quality of learned node embeddings.

In particular, for a specific node, we recommend nodes that have high-probabilities to build positive directed links. For example, denote i as the source node, we want to recommend a target node list $\mathcal{J}_i = [j_1, j_2, ..., j_k]$ which is ranked according to the prediction scores to build positive links. k means the cut off number. Specifically, we

Table 3–5 Node recommendation performance on Wiki. Names with $*$ refer to our methods. The metrics for this task are Recall@k and Precision@k. We pick k=10,20,50 here. Compared to SiNE, the absolute improvement of DVE is given in the table.

Dataset	Wiki					
Methods	R@10	R@20	R@50	P@10	P@20	P@50
LINE	0.037	0.054	0.112	0.014	0.010	0.009
MF	0.011	0.024	0.048	0.005	0.005	0.004
SNE	0.002	0.005	0.011	8.4e-4	9.9e-4	9.5e-4
SiNE	0.033	0.055	0.111	0.012	0.010	0.009
SIDE	0.001	0.002	0.009	5.8e-4	5.2e-4	7.6e-4
SNEA-MR	0.002	0.004	0.014	9.4e-4	7.6e-4	0.011
BPWR*	0.049	0.087	0.174	0.018	0.016	0.014
SLVE*	0.013	0.037	0.101	0.004	0.006	0.007
DE*	0.043	0.073	0.152	0.018	0.016	0.014
DVE*	**0.050(1.7%)**	**0.092(3.7%)**	**0.179(6.8%)**	**0.020(0.8%)**	**0.018(0.8%)**	**0.016(0.7%)**

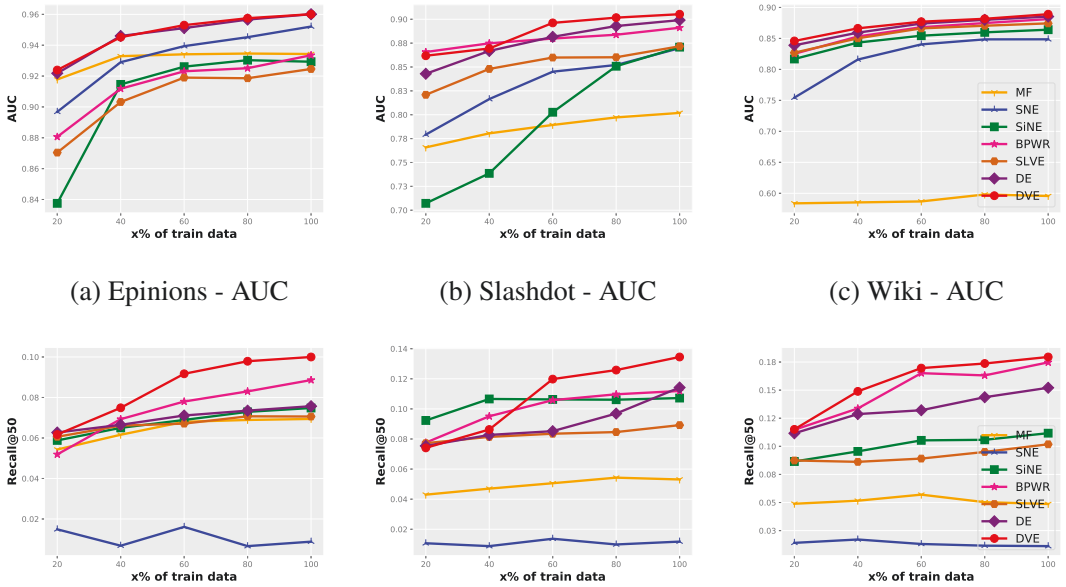
use the learned embeddings and calculate the prediction scores by the trained model on the test nodes. The results are shown in Table 3–3,3–4,3–5, from which we see that:

- DVE outperforms other baselines on all datasets. Compared to SiNE on Recall@50, DVE reaches a 2.6%, a 2.7% and a 6.8% improvement on Epinions, Slashdot and Wiki, respectively. Compared to the baselines that ignore the *high-order* topology, DVE integrally extracts both the *first-order* and *high-order* topology, and learns more representative node embeddings for node recommendation.
- Compared with other baselines (SiNE, SNE, MF), BPWR models the mediator value of non-existent links. In this case, it has better ability to learn the relative difference of node embeddings for better node recommendation performance.

3.3.4.3 Effects of Sparse Training Data

We investigate the effects of sparse training data on model performance. In particular, we vary the ratio of the 80% training data as new train data and keep the 20% test data fixed. Results are shown in Figure 3–4. From this figure, we see that:

- Generally, the performance of each model decreases with the decline of training data, which indicates that sparse data causes deterioration to the model performance. Meanwhile, DVE consistently shows the best performance in most sparse cases, which indicates DVE has better adaptability with more sparse edges.
- For node recommendation task in Figure 3–4 (d)(e)(f), it is obvious that both



(a) Epinions - AUC (b) Slashdot - AUC (c) Wiki - AUC

(d) Epinions - Recall@50 (e) Slashdot - Recall@50 (f) Wiki - Recall@50

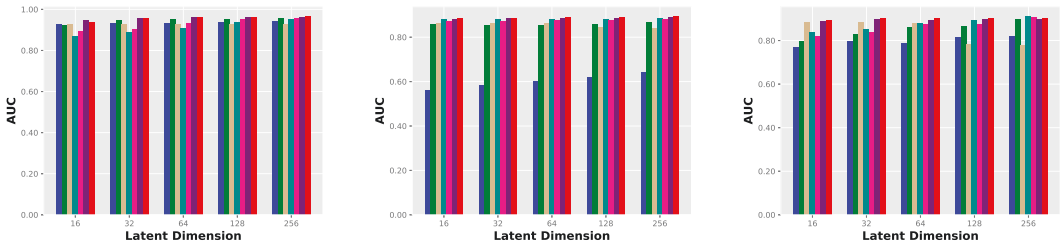
Figure 3–4 Comparison of methods with different training data on Epinions, Slashdot and Wiki for two tasks. AUC in (a)(b)(c) is the metric for link sign prediction task and Recall@50 in (d)(e)(f) is the metric for node recommendation task.

MF and SNE perform worse than other methods (*e.g.* SiNE, BPWR) in most cases, because MF and SNE are not ranking based loss. By contrast, SiNE and BPWR are both based on ranking loss which is more advantageous in node recommendation task.

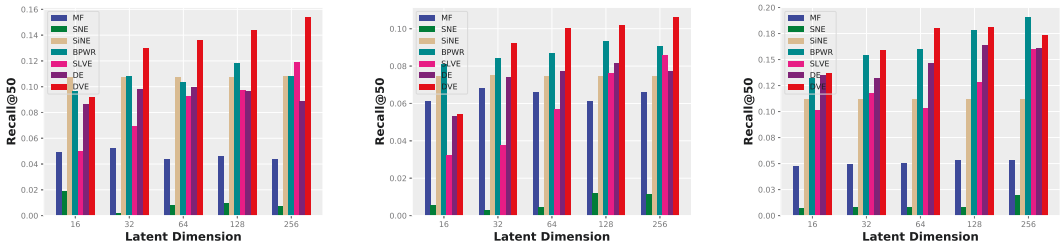
3.3.4.4 Effects of Different Latent Dimensions

The latent dimension of embeddings is an important factor that accounts for the model performance in network embedding. We thus conduct an experiment to investigate the effects of different latent dimensions. The results are shown in Figure 3–5. From this figure, we have the following observations:

- The proposed methods (BPWR, DE and DVE) consistently outperform other baselines with different latent dimensions. DVE achieves the best performance in most cases, because DVE is the only method that simultaneously captures both the *first-order* and *high-order* topological information in signed directed networks.
- It is worthwhile to notice that DVE tends to reach a better performance at a



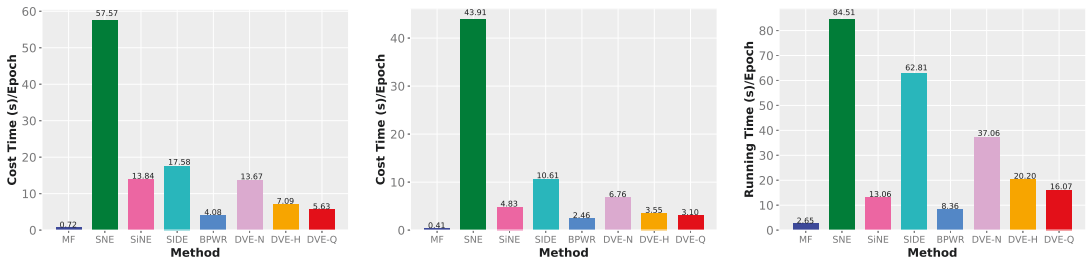
(a) Epinions - AUC (b) Slashdot - AUC (c) Wiki - AUC



(d) Epinions - Recall@50 (e) Slashdot - Recal@50 (f) Wiki - Recall@50

Figure 3–5 Comparison of methods with different latent dimension on Epinions, Slashdot and Wiki for two tasks. AUC in (a)(b)(c) is the metric for link sign prediction task and Recall@50 in (d)(e)(f) is the metric for node recommendation task.

higher dimension in comparison with SiNE. DVE considering both the *high-order* and the *first-order* topology requires a high dimension to encode the additional information. As for the baselines that only consider the *first-order* topology, when the dimension is high, the information in learned embeddings tends to be redundant and yields unsatisfied performance on the test set.



(a) Epinions (b) Slashdot (c) Wiki

Figure 3–6 The empirical running time in each epoch of different methods. In this figure, DVE-N indicates the non-parallel DVE, DVE-H means the half-parallel one and the DVE-Q denotes the quarter-parallel one.

3.3.4.5 Empirical Running Time Analysis

To investigate the time complexity, we conduct an experiment to compare the empirical running time in each epoch of different methods. In particular, we set the training batch number as 1,000 for all methods. All these methods are implemented with deep learning programming frameworks such as Pytorch, Tensorflow or Theano. For DVE, we implement it with Tensorflow and the sampling size n_{noise} of non-existent links is 5,5,20 on Epinions, Slashdot and Wiki, respectively. Since DVE has parallel versions, we thus denote DVE-N as the non-parallel one, DVE-H as the half-parallel one and DVE-Q as the quarter-parallel one. We conduct the experiments 10 times on the same machine with one Nvidia 1080-Ti GPU. The mean value of running time per epoch is reported in Figure 5–10. From the table, we can summarize that:

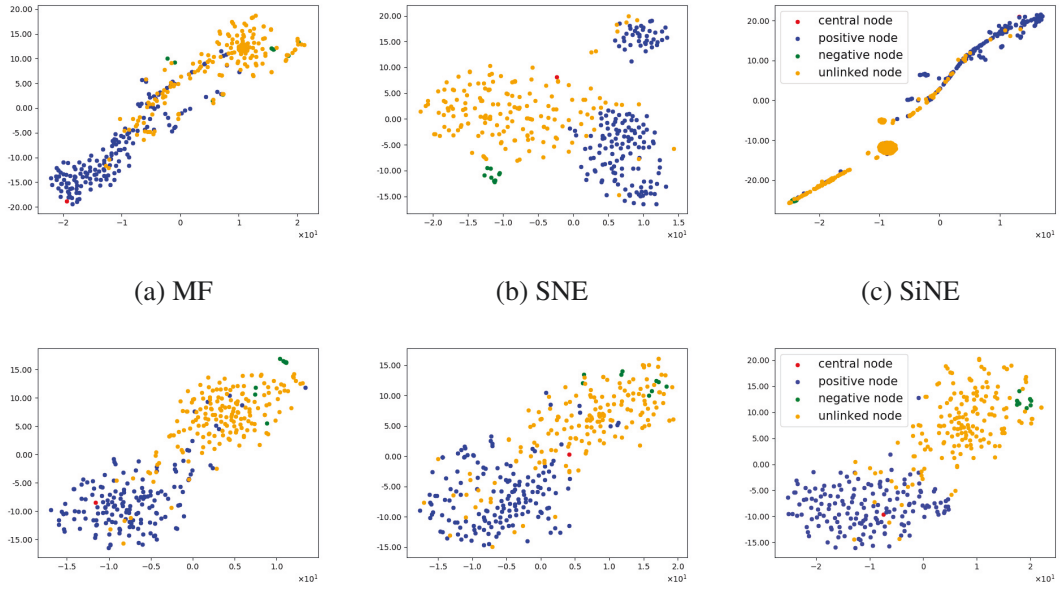
- MF costs the least time because of its simple scheme. SNE and SIDE involving the softmax operation consume much more time than other methods. The proposed DVE in non-parallel version (DVE-N) generally costs more time than MF and SiNE, because DVE is more complex in order to capture both the *high-order* and *first-order* topological information.
- DVE-H and DVE-Q take much less time than DVE-N. They are even faster than SiNE in some cases. Meanwhile, DVE provides better performance in comparison with SiNE. In summary, the decoupling idea has advantages to accelerate the training process as well as learns more representative node embeddings.

3.3.5 Qualitative Visualization

3.3.5.1 Topology Preservation

Signed directed networks have complex topology pattern and there are some obvious topology characteristics. If we denote i as a source node, $\mathcal{N}_p(i)$ as the positively linked target neighbours, $\mathcal{N}_n(i)$ as negatively linked target neighbours and $\mathcal{N}_{un}(i)$ as the non-linked neighbours, there are several characteristics of topology in signed directed networks:

- $\mathcal{N}_p(i)$, $\mathcal{N}_n(i)$ and $\mathcal{N}_{un}(i)$ tend to be three clusters since they play different roles for node i;
- Closeness between $\mathcal{N}_p(i)$ and node i tends to be larger than that between $\mathcal{N}_{un}(i)$ and node i;



(a) MF (b) SNE (c) SiNE

(d) BPWR (e) SLVE (f) DVE

Figure 3–7 t-SNE visualization of topology preservation in Epinions. The node in red color means the sampled central node i as source node. The nodes in blue color represents the positively linked neighbours $\mathcal{N}_p(i)$ and nodes in green color are the negatively linked neighbours $\mathcal{N}_n(i)$. While the yellow ones are randomly sampled non-linked nodes $\mathcal{N}_{un}(i)$ for center node i. Both $\mathcal{N}_p(i)$ and $\mathcal{N}_n(i)$ are target nodes.

- Closeness between $\mathcal{N}_{un}(i)$ and node i is larger than that between $\mathcal{N}_n(i)$ and node i.

In order to study whether the learned node embeddings preserve the above characteristics, we conduct an experiment to visualize the node embeddings by t-SNE [124]. In particular, we randomly sample a source node i whose number of directly linked neighbours is larger than 100 from Epinions. The positively linked neighbours $\mathcal{N}_p(i)$ and negatively linked neighbours $\mathcal{N}_n(i)$ are both from target nodes. Next, we also randomly sample some non-linked nodes $\mathcal{N}_{un}(i)$. Finally, we visualize the corresponding embeddings for 6 methods. The results are shown in Figure 3–7. From the figure, we can summarize that:

- DVE has the best visualization performance in terms of the well clustered nodes and clear closeness pattern among different types of nodes. For SNE in Figure 3–7 (b), we can see that the closest neighbours for the central node are non-linked nodes and the positively linked nodes are not well clustered. For SiNE in Figure 3–7 (c), the nodes are not well distributed and it is even impossible to recognize

some nodes. MF in Figure 3–7 (a) clusters the positively linked nodes and non-linked nodes well but fails in negatively linked nodes. Compare to these three competitive baselines, our proposed methods BPWR and DVE in Figure 3–7 (d) and (f) are capable of learning the distributed and well clustered node embeddings. The central node in red color are surrounded by the positively linked nodes in blue color. The clear closeness pattern among different types of nodes as well matches the fact that we have illustrated before. These advantages are benefited from modeling both the *first-order* and *high-order* topology in signed directed networks.

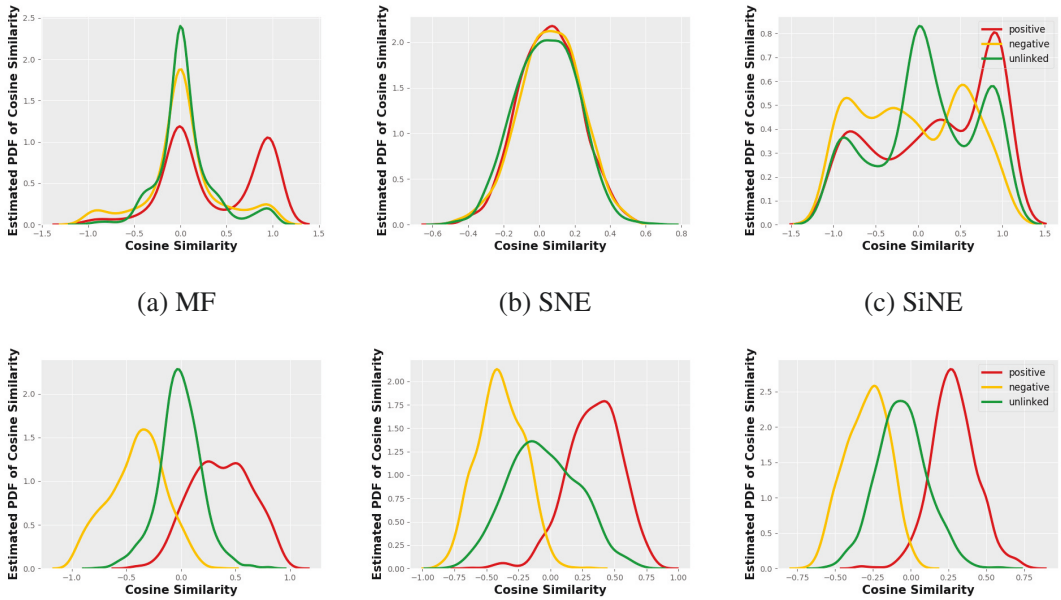
- DVE models the distinctive influence of messaging propagation in signed directed networks, and yields better topology preservation. Compared to DVE in Figure 3–7 (f), SLVE in Figure 3–7 (e) tends to mix positively linked nodes and non-linked nodes. Moreover, the central node in red color is false positioned in the middle part of positively linked nodes and non-linked nodes. This is because SLVE applies homophily effects on different signs, which cannot model the distinctive influence of message propagation. In contrary, DVE with decoupled variational encoder can learn distinctive effects for different signs.

3.3.5.2 Closeness Distribution

In signed directed networks, positive edges mean trust or friend while negative edges represent distrust or enemy, and the non-existent edges may both have the probability to be positive ones or negative ones. According to Extended Structural Balance theory, different types of node pairs pose different closeness distributions and we may have the following rules:

- The similarity between positively linked node pairs is expected to be large because of the semantics of positive edges;
- The similarity between negatively linked node pairs should be small due to the meaning of negative edges;
- For the node pairs with non-existent edges, they have potential to be either positive or negative relation, and should be in the middle position between positively linked node pairs and negatively linked ones.

Thereby, we conduct an experiment to investigate whether DVE has better ability of preserving the closeness distribution pattern. In particular, we visualize the estimated



(a) MF (b) SNE (c) SiNE

(d) BPWR (e) SLVE (f) DVE

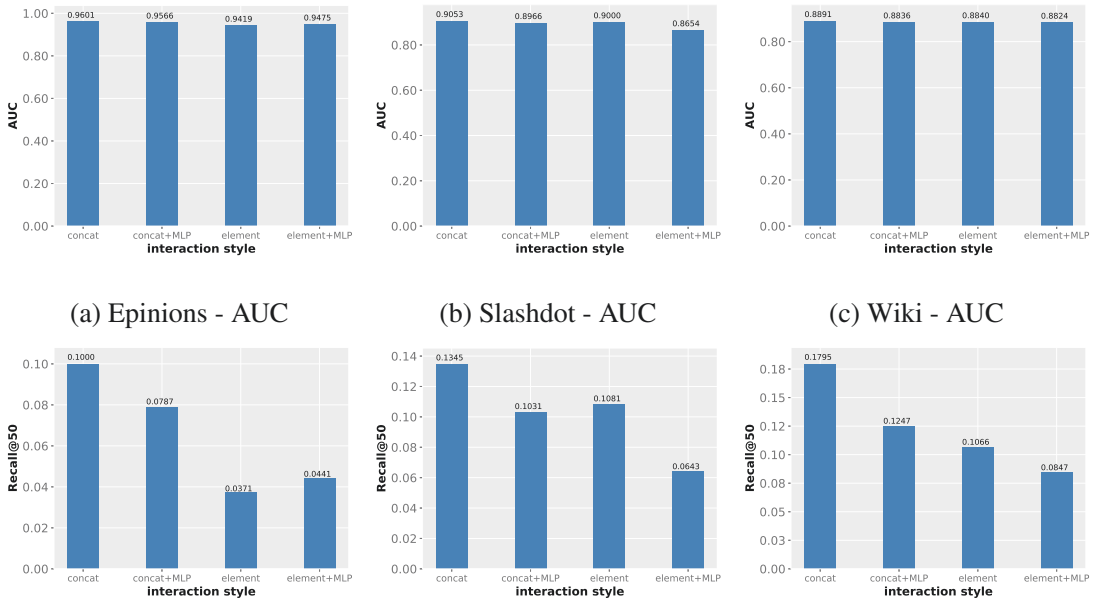
Figure 3–8 Estimated probability density function of different types of node pairs on Slashdot for 6 methods. The red curve means the estimated PDF (Probability Density Function) of cosine similarity among positively linked node pairs. Similarly, the yellow curve and green curve denote the estimated PDF among the negatively linked node pairs and non-linked node pairs, respectively.

Probability Density Function (PDF) of different node pairs on Slashdot for 6 methods. In particular, we calculate the cosine similarity of all positively linked node pairs, negatively linked node pairs and randomly sampled non-linked node pairs by leveraging the learned embeddings from 6 methods. The estimated PDF curves of cosine similarity are shown in Figure 3–8. The red curve, yellow curve and green curve indicate the estimated PDF curve for positively linked node pairs, negatively linked node pairs and non-linked node pairs, respectively. From this figure, we have the following observations:

- From Figure 3–8 (a)(b)(c), we can see that the baseline methods MF, SNE and SiNE all exhibit high overlap of different curves, especially for SNE in Figure 3–8. This indicates these methods are not capable of capturing the different closeness distribution patterns of different node pairs. By contrast, considering the results of BPWR and DVE in Figure 3–8 (d)(e)(f), it is obvious that these three curves show different distributions. BPWR and DVE follow the closeness rules where positively linked node pairs have highest cosine similarity, non-linked node pairs have the second and negatively linked ones have the last.

- In Figure 3–8 (d)(e), the estimated PDF of non-linked node pairs in green color tends to have more overlap with the other curves, which may lead to indistinguishable node embeddings. Instead, DVE in Figure 3–8 (f) presents both distinguishable estimated PDF curves with smaller overlap and obvious cosine similarity gap among different kinds of node pairs. This confirms that DVE can better preserve the closeness distribution pattern in signed directed networks.

3.3.6 Ablation Study



(a) Epinions - AUC (b) Slashdot - AUC (c) Wiki - AUC

(d) Epinions - Recall@50 (e) Slashdot - Recall@50 (f) Wiki - Recall@50

Figure 3–9 Performance of DVE with different generative functions. AUC in (a)(b)(c) refers to the metric for link sign prediction task. Recall@50 in (d)(e)(f) is the metric for node recommendation task.

3.3.6.1 Effects of Different Generative Functions

Remind that we assume the source node embeddings Z_s could be generated through $Z_s = f_s(Z_s^p, Z_s^n)$, where f_s is the generative function. In order to explore the influence of different generative functions, we conduct an experiment with various functions that are defined in Table 3–6. Note that we only use f_s as an example to illustrate the experiment

Table 3–6 Different generative functions for f_s. In this table, $[\cdot,\cdot]$ means the concatenation operation and $W_C \in \mathbb{R}^{2d\times 2d}$ is the weight of MLP for concatenation. $\odot$ indicates the element-wise product operation and $W_E \in \mathbb{R}^{d\times d}$ is the weight of MLP for element-wise product.

type	formula
concat	$Z_s = [Z_s^p, Z_s^n]$
concat+MLP	$Z_s = ([Z_s^p, Z_s^n])W_C$
element-wise product	$Z_s = Z_s^p \odot Z_s^n$
element-wise product+MLP	$Z_s = (Z_s^p \odot Z_s^n)W_E$

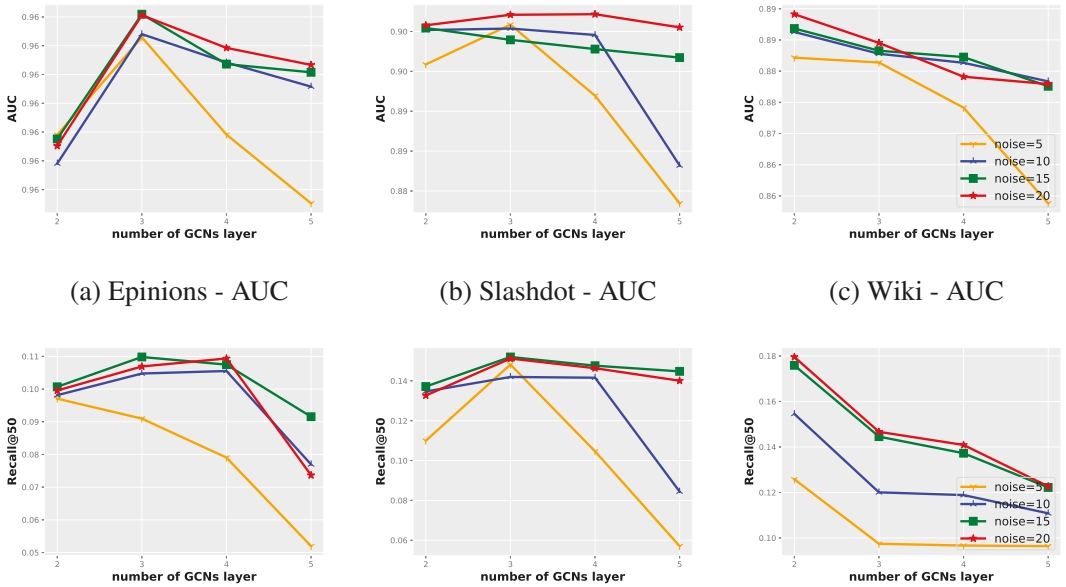
setting here. The target node representation Z_t has similar formulation but the notations are Z_t^p, Z_t^n and f_t. The results are shown in Figure 3–9.

From Figure 3–9, we observe two interesting phenomenons: 1) concatenation does better than concatenation+MLP and element-wise product does better than element-wise product+MLP; 2) concatenation performs better than element-wise product and concatenation+MLP performs better than element-wise product+MLP. The main reason for the first phenomenon may be that additional trainable parameters lead to over-fitting on the sparse graph data. For the second phenomenon, it is because both Z_s^p and Z_s^n are learned with distinctive deep neural networks, which indicates they are in different latent spaces. The aligned element-wise product may lead to information loss to represent source node embeddings Z_s. Therefore, concatenation operation tends to be the most suitable choice among them in terms of both efficiency and easy implementation.

3.3.6.2 Hyper-Parameter Sensitivity

In DVE, the two hyper-parameters are the number of GCN layer n_{GCN} and the noise sampling size n_{noise}. n_{GCN} controls the order of a node's local structures and n_{noise} influences the sampling size of non-existent links. We here investigate the model sensitivity on these two hyper-parameters. The results are show in Figure 3–10, from which we have the following observations:

- DVE achieves its best performance on different datasets when $n_{GCN} = 2, 3$ and $n_{noise} = 5, 20$. The slight change of n_{GCN} indicates that most useful topological information is within low-order neighborhoods. While the noise sampling size n_{noise} varies a lot on different datasets, which means n_{noise} is better chosen according to the datasets.



(a) Epinions - AUC (b) Slashdot - AUC (c) Wiki - AUC

(d) Epinions - Recall@50 (e) Slashdot - Recall@50 (f) Wiki - Recall@50

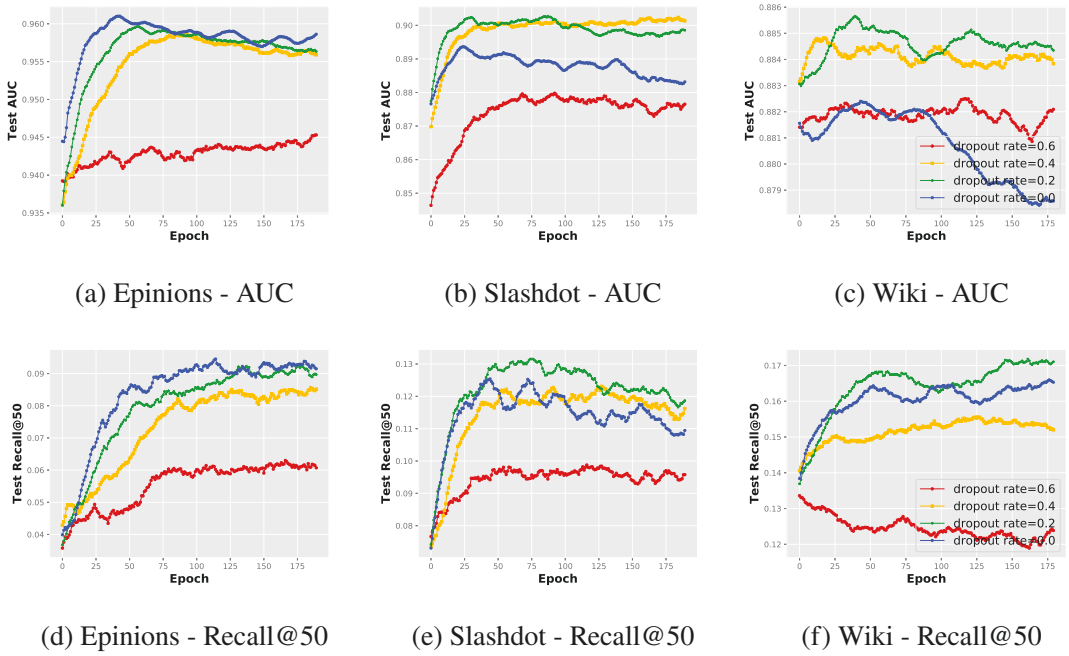
Figure 3–10 Performance of DVE with different parameter settings. AUC in (a)(b)(c) is the metric for link sign prediction task and Recall@50 in (d)(e)(f) is the metric for node recommendation task.

- From Figure 3–10 (a)(b)(c) for link sign prediction task, we see that the performance of link sign prediction does not change a lot (0.958~ 0.964 on Epinions and 0.88~ 0.90 on Slashdot and 0.86~ 0.88 on Wiki). This slight change indicates that the link sign prediction performance is robust to n_{GCN} and n_{noise}. However, as shown in Figure 3–10 (d)(e)(f), the node recommendation performance changes obviously (0.10~ 0.06 on Epinions and 0.14~ 0.06 on Slashdot and 0.09~ 0.17 on Wiki). This is because the node recommendation is instinctively measured from model training, while a binary classifier is additionally trained for link sign prediction. The binary classifier reduces model sensitivity on n_{GCN} and n_{noise}.

3.3.6.3 Dropout Regularization

In our method, we apply Dropout for regularization. In order to study the influence of Dropout, we investigate the model performance with different Dropout rates along the training process. The results are shown in Figure 3–11.

From Figure 3–11, we can see that different Dropout rates may have different influence on different datasets. In Figure 3–11 (a)(d) for Epinions, it is obvious that



(a) Epinions - AUC (b) Slashdot - AUC (c) Wiki - AUC

(d) Epinions - Recall@50 (e) Slashdot - Recall@50 (f) Wiki - Recall@50

Figure 3–11 Performance of DVE with different dropout rates. Dropout rate=0.0 means dropout keep probability is 1.0 during training. AUC in (a)(b)(c) is the metric for link sign prediction task and Recall@50 in (d)(e)(f) is the metric for node recommendation task.

DVE reaches its best performance when dropout=0.0, which means the Dropout keep probability equals 1.0 is better for Epinions. The reason for this may be that the training data is enough to model the data distribution and no Dropout encourages the model to fit data better. In contrary, in Figure 3–11 (b)(e) for Slashdot and Figure 3–11 (c)(f) for Wiki, Dropout rate equals 0.2 facilitates better performance, which indicates DVE needs necessary regularization on these two datasets to avoid over-fitting.

3.4 Summary

In this Chapter, we study the representation learning problem on signed directed networks. In particular, we reformulate the problem from a variational auto-encoding perspective and further propose a decoupled variational embedding (DVE) method to learn representative node embeddings. DVE is capable of preserving both the *first-order* and *high-order* topology for signed directed networks. Extensive experiments on three real-world datasets of two tasks proves the superiority of DVE compared to recent competitive baselines.

Remind that DVE constructs source node embeddings Z_s just by the limited concatenation operation of two latent embeddings Z_s^p and Z_s^n. We will explore how to better model the interaction between Z_s^p and Z_s^n for better performance in future.

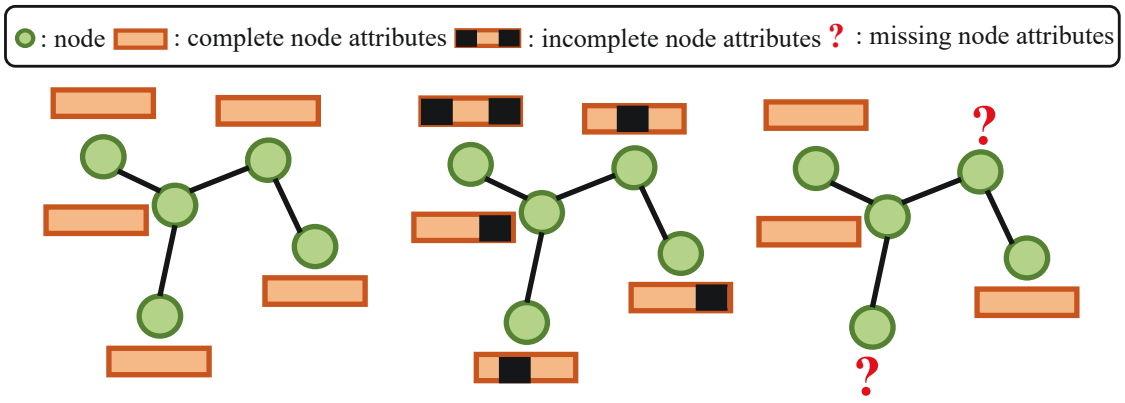
Chapter 4 Learning on Attribute-Missing Graphs

Graph-structured data are heterogeneous in nature, with correlated information about graph structures and node attributes. Given a graph with attributes, based on the completeness of the node attributes, we may classify the graph into the three types shown in Figure 4–1. 1) Attribute-complete graph: every node is with complete set of attributes; 2) Attribute-incomplete graph : every node is with a non-empty set of attributes; 3) Attribute-missing graph : attributes of partial nodes are entirely missing.

The attribute-missing one is related to real-world applications. For example, in citation networks, raw attributes or detailed descriptions of some papers may be inaccessible due to the copyright protection. In social networks, user profiles may be unavailable because of the privacy protection. Existing graph learning algorithms on this graph type either face the sampling bias issue of graph structures or are not compatible for learning. In this Chapter, we study the learning problems on attribute-missing graphs and propose a novel distribution matching based graph learning framework. We perform extensive experiments on seven benchmark datasets. The experimental results have demonstrated that the proposed framework achieves superior learning performance.

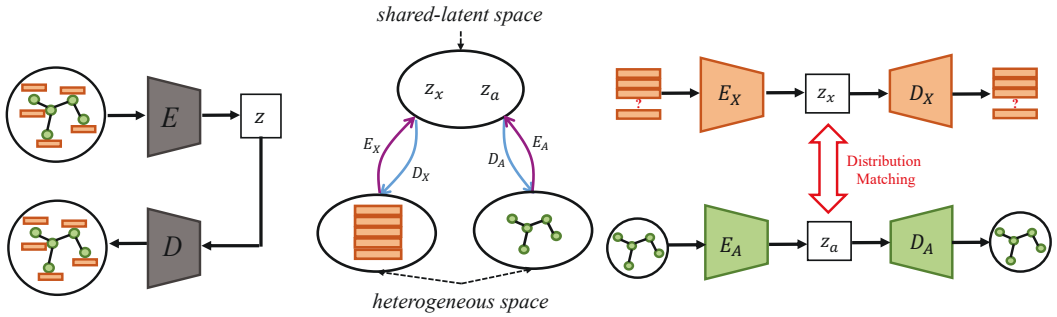
4.1 Introduction

In current stage of DLG, there are limited studies investigating learning on attribute-missing graphs. Existing graph learning methods such as the random walk based [11, 17], the attributed random walk based [39-40] and GNN [1-2, 26, 125-126] are not specified for attribute-missing graphs and are limited in solving the corresponding learning problems. Random walk based methods emerged as large-scale and effective graph embedding approaches which only take structures into consideration. They cannot take the advantage of rich information from node attributes [1]. Although the attributed random walk based methods [39-40] can potentially deal with the attribute-missing graph, these methods rely on high-quality random walks and require carefully designed sampling strategies and fine-tuned hyper-parameters [41]. GNN leverages structures and attributes as a whole in a unified framework which is shown in Figure 4–2 (a) [1, 8, 27, 125]. On attribute-missing graphs with only partial nodes associated with attributes, GNN can work with some



(a) Attribute-complete (b) Attribute-incomplete (c) Attribute-missing

Figure 4–1 Given a graph with attributes, based on the completeness of the node attributes, we may classify the graph into three types.



(a) Recent GNN (b) *shared-latent space* assumption (c) Our SAT

Figure 4–2 The comparison between recent GNN and our SAT, together with the illustration of our *shared-latent space* assumption. (a)means recent GNN usually requires structures and attributes as a whole input. E is the encoder and D is the decoder. (b) shows our *shared-latent space* assumption where E_X and E_A are two encoders and D_X and D_A are two decoders. (c) shows the general architecture of the proposed SAT.

attribute-filling tricks such as zero-filling. However, these tricks would introduce noise to the learning process. Therefore, designing a learning algorithm for attribute-missing graphs is a burning issue to the graph learning community.

One possible way to solve the issue is to input structures and attributes in a decoupled scheme and simultaneously allow the joint distribution modeling of structures and attributes. Structures and attributes are two resources that come from two marginal distributions. The coupling theory [127] states that there is an infinite set of joint distributions

that can reach the given marginal distributions in general. Therefore, it is impossible for us to perform the joint distribution modeling without any assumption. Besides, the graph structured data has two characteristics: 1) the structures and attributes come from two heterogeneous spaces (*heterogeneity*); 2) the node attributes can either be real-valued or categorical (*discontinuity*). These two characteristics hinder us from achieving joint distribution modeling in data space by existing techniques such as adversarial training in data space [52, 128-129]. To this end, we make a *share-latent-space assumption* on graphs which assumes that the heterogeneous structures and attributes are related to each other and come from the same latent space (illustrated in Figure 4–2 (b)).

In this Chapter, we achieve the *shared-latent space* assumption by distribution matching techniques and further develop a novel distribution matching based GNN for learning on attribute-missing graphs, called structure-attribute transformer (SAT). The general architecture of SAT is shown in Figure 4–2 (c). SAT leverages structures and attributes in a decoupled scheme and achieves the joint distribution modeling by matching the latent codes of structures and attributes. It can not only perform the link prediction task but also retrieve the missing attributes. We also call the latter as *node attribute completion* which benefits several subsequent tasks such as node classification and profiling. The *node attribute completion* is a new problem on graphs which distinguishes from other existing problems and requires specific evaluation measures. Main contributions of this paper can be summarized as follows:

- We investigate the learning problems on attribute-missing graphs. In this scenario, we make a *shared-latent space* assumption on graphs and develop a novel distribution matching based GNN called SAT. SAT is a generic framework that can be equipped with other popular GNN backbones such as GCN and GAT;
- We introduce a new task called *node attribute completion* on graphs which aims to complete the missing node attributes and benefit other subsequent tasks. To quantify the performance of *node attribute completion*, practical evaluation measures are introduced including both node classification in the *node level* and profiling in the *attribute level*;
- Extensive experiments on seven real-world datasets show that our method can not only perform well on link prediction task but also restore high-quality attributes that benefit subsequent tasks such as node classification and profiling.

4.2 SAT: Structure-Attribute Transformer

4.2.1 Problem Definition

Let $\mathcal{G} = (\mathcal{V}, A, X)$ be a graph with node set $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, $A \in R^{N \times N}$ be the graph adjacency matrix and $X \in \mathbb{R}^{N \times F}$ be the node attribute matrix. Note that the element in X could be either categorical values or real values. Let $\mathcal{V}^o = \{v_1^o, v_2^o, ..., v_{N_o}^o\}$ be the set of attribute-observed nodes. Let $x_i^o \in \mathbb{R}^F$ and a_i^o be the attribute vector and structural information (the neighbors of a node) of node v_i^o, respectively. Then, for $\mathcal{V}^o$, the attribute information is denoted as $X^o = \{x_0^o, x_1^o, ..., x_{N_o}^o\}$ and structural information is denoted as $A^o = \{a_0^o, a_1^o, ..., a_{N_o}^o\}$. Let $\mathcal{V}^u = \{v_1^u, v_2^u, ..., v_{N_u}^u\}$ be the set of attribute-missing nodes. Let $x_i^u \in \mathbb{R}^F$ and a_i^u be the attribute vector and structural information of node v_i^u, respectively. Then, for $\mathcal{V}^u$, the attribute information is denoted as $X^u = \{x_0^u, x_1^u, ..., x_{N_u}^u\}$ and structural information is denoted as $A^u = \{a_0^u, a_1^u, ..., a_{N_u}^u\}$. According to our definition, A is another expression of A^o and A^u. X is another expression of X^o and X^u. To clarify more clearly, we have $\mathcal{V} = \mathcal{V}^u \cup \mathcal{V}^o$, $\mathcal{V}^u \cap \mathcal{V}^o = \emptyset$ and $N = N_o + N_u$. Learning on attribute-missing graphs has several applications such as link prediction and the newly introduced *node attribute completion*. In link prediction, we expect to predict the missing links in A. In *node attribute completion*, we aim to restore the missing node attributes X^u based on the observed node attributes X^o and graph structures A.

4.2.2 Overview

Graph-structured data include two perspectives of representations: structures and attributes. On attribute-missing graphs, attributes of some nodes might be entirely missing. One possible way to learn on attribute-missing graphs is to input structures and attributes in a decoupled scheme and simultaneously allow the joint distribution modeling of structures and attributes. In particular, if we denote x_i and a_i as the attribute vector and structural information of node v_i, we can see (x_i, a_i) is a paired sample to describe node v_i. The joint log-likelihood of the attributes and structures is composed of a sum over the likelihoods of individual datapoints $\sum_i p_\theta(x_i, a_i)$, where $p_\theta(x_i, a_i)$ is the joint probability density function. Inspired by the idea of maximizing the marginal likelihood in VAE [100], if z_x and z_a are the latent factors of x_i and a_i, respectively, $\log p_\theta(x_i, a_i)$

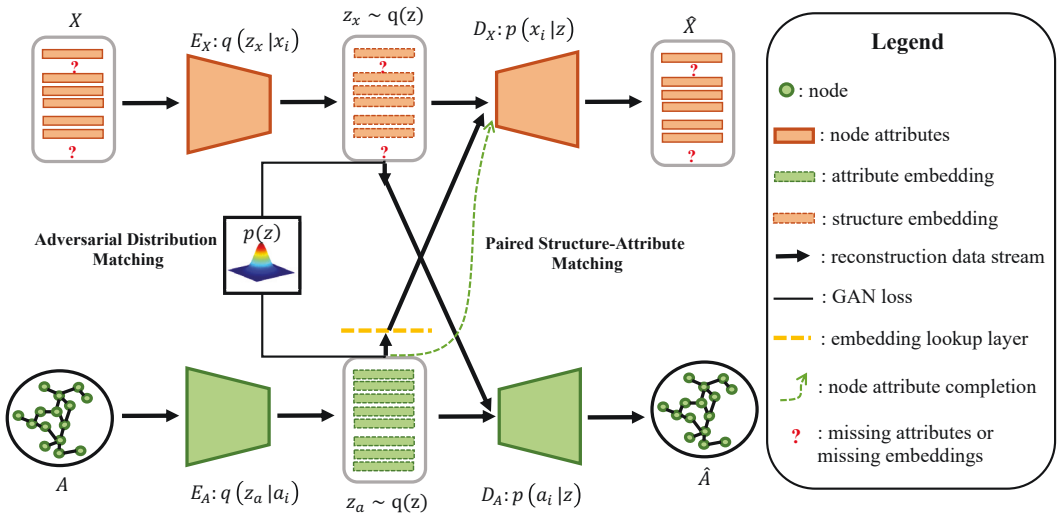


Figure 4–3 Architecture of SAT. SAT first transforms attributes and structures into the latent space, then aligns the paired latent representations via adversarial distribution matching, and finally decodes to the original attributes and structures, namely the paired structure-attribute matching.

can be formulated as:

$$\begin{aligned}\log p_\theta(x_i, a_i) =& D_{KL}[q_\phi(z_x, z_a|x_i, a_i)||p(z_x, z_a|x_i, a_i)] \\ &+ \mathcal{L}(\theta, \phi; x_i, a_i)\end{aligned} \tag{4–1}$$

where the first term is the KL divergence of the approximate posterior $q_\phi(z_x, z_a|x_i, a_i)$ from the true posterior $p(z_x, z_a|x_i, a_i)$. Since this KL term is non-negative, the second term is the *ELBO* on the log-likelihood $\log p_\theta(x_i, a_i)$. And $\mathcal{L}(\theta, \phi; x_i, a_i)$ is written as:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x_i, a_i) =& \mathbb{E}_{q_\phi(z_x, z_a|x_i, a_i)}[\log p_\theta(x_i, a_i|z_x, z_a)] \\ &- D_{KL}[q_\phi(z_x, z_a|x_i, a_i)||p(z_x, z_a)]\end{aligned} \tag{4–2}$$

where $p_\theta(x_i, a_i|z_x, z_a)$ denotes the conditional distribution parameterized by θ. The first term in Eq. 4–2 indicates the joint reconstruction loss where z_x, z_a encoded from x_i, a_i are used to reconstruct x_i, a_i. The second term in Eq. 4–2 indicates the prior regularization loss where $q_\phi(z_x, z_a|x_i, a_i)$ is expected to match the prior distribution $p(z_x, z_a)$. The proposed SAT implements the the joint reconstruction loss via a paired structure-attribute matching strategy and approximates and the prior regularization loss via an adversarial distribution matching strategy. The architecture of SAT is shown in Figure 4–3. In the following sections, we provide details on the two strategies, followed by the objective function and implementation.

4.2.3 Paired Structure-Attribute Matching

The first term $\mathbb{E}_{q_\phi(z_x,z_a|x_i,a_i)}[\log p_\theta(x_i,a_i|z_x,z_a)]$ in Eq. 4–2 consists of an approximate posterior $q_\phi(z_x,z_a|x_i,a_i)$ parameterized by ϕ and a conditional distribution $p_\theta(x_i,a_i|z_x,z_a)$ parameterized by θ. The posterior $q_\phi(z_x,z_a|x_i,a_i)$ is rather complicated to infer since the two latent variables z_x and z_a are coupled together. In order to solve this complicated posterior, recent image-to-image translation works [129-131] employ the mean field theory to approximate the posterior at a low computational cost. Similarly, we make an analogy of the nodes with two different views (*i.e.* the attribute view and the structure view) to objects with two different image views and define the posterior as:

$$q_\phi(z_x,z_a|x_i,a_i) \overset{\text{def}}{=\!=} q_{\phi_x}(z_x|x_i,a_i)q_{\phi_a}(z_a|x_i,a_i) = q_{\phi_x}(z_x|x_i)q_{\phi_a}(z_a|a_i) \tag{4–3}$$

where $q_{\phi_x}(z_x|x_i)$ indicates the encoder E_X that encodes x_i to z_x. Similarly, $q_{\phi_a}(z_a|a_i)$ indicates the encoder E_A that encodes a_i to z_a. In order to solve the conditional distribution, we make the following *shared-latent space* assumption on graphs, which allows us to perform the joint distribution modeling in latent space.

Shared-latent Space Assumption: *In graph structured data, each node's attributes and structures are correlated together and can be represented with the same distribution in a shared-latent space*

This assumption indicates the latent variables z_x and z_a are equivalent to each other. It is also illustrated in Figure 4–2 (b), with which we have:

Proposition 4.1. *Given latent variables z_x and z_a, the observations x_i and a_i are conditional independent.*

With Proposition 4.1 and the assumption, we have:

$$\begin{aligned} p_\theta(x_i,a_i|z_x,z_a) &\overset{\text{Proposition 4.1}}{=\!=\!=} p_{\theta_x}(x_i|z_x,z_a)p_{\theta_a}(a_i|z_x,z_a) \\ &\overset{\textit{Shared-latent space assumption}}{=\!=\!=} p_{\theta_x}(x_i|z_x)p_{\theta_a}(a_i|z_a) \\ &= \sqrt{p_{\theta_x}(x_i|z_x)p_{\theta_x}(x_i|z_a)}\sqrt{p_{\theta_a}(a_i|z_a)p_{\theta_a}(a_i|z_x)} \end{aligned} \tag{4–4}$$

where θ_x is the parameter of the shared decoder D_X and θ_a is the parameter of the shared decoder D_A. Eq. 4–4 indicates a paired structure-attribute matching strategy where z_x is used to reconstruct x_i and a_i, and z_a is used to reconstruct a_i and x_i. It is worthwhile to mention that the shared-latent space assumption is not the best one since node attributes and structures might not be fully overlapped. That's why we use distribution matching

scheme to formulate this assumption rather than discriminative approaches, to provide certain flexibility and uncertainty in learning. The experiments in Section 4.3 can also confirm the effectiveness of our method.

Taking the above into consideration, we write $\mathbb{E}_{q_\phi(z_x,z_a|x_i,a_i)}[\log p_\theta(x_i,a_i|z_x,z_a)]$ in Eq. 4–2 as:

$$\begin{aligned}\mathbb{E}_{q_\phi(z_x,z_a|x_i,a_i)}[\log p_\theta(x_i,a_i|z_x,z_a)] = \frac{1}{2}\big\{&\mathbb{E}_{x_i\sim p_X}[\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_x}(x_i|z_x)]]+\\ &\mathbb{E}_{a_i\sim p_A}[\mathbb{E}_{q_{\phi_a}(z_a|a_i)}[\log p_{\theta_a}(a_i|z_a)]]+\\ &\mathbb{E}_{a_i\sim p_A}[\mathbb{E}_{q_{\phi_a}(z_a|a_i)}[\log p_{\theta_x}(x_i|z_a)]]+\\ &\mathbb{E}_{x_i\sim p_X}[\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_a}(a_i|z_x)]]\big\}\end{aligned} \tag{4–5}$$

The optima of of Eq. 4–5 is the same as Eq. 4–5 multiplied by a constant. Thereby, for simplified expression, we write the joint reconstruction loss as:

$$\begin{aligned}\min_{\theta_x,\theta_a,\phi_x,\phi_a}\mathcal{L}_r = &-\mathbb{E}_{x_i\sim p_X}[\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_x}(x_i|z_x)]]\\ &-\mathbb{E}_{a_i\sim p_A}[\mathbb{E}_{q_{\phi a}(z_a|a_i)}[\log p_{\theta_a}(a_i|z_a)]]\\ &-\lambda_c\cdot\mathbb{E}_{a_i\sim p_A}[\mathbb{E}_{q_{\phi a}(z_a|a_i)}[\log p_{\theta_x}(x_i|z_a)]]\\ &-\lambda_c\cdot\mathbb{E}_{x_i\sim p_X}[\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_a}(a_i|z_x)]].\end{aligned} \tag{4–6}$$

where the first two terms indicate the self-reconstruction stream. It means the information from attributes (*resp.* structures) is decoded as attributes (*resp.* structures). The last two terms indicate the cross-reconstruction stream. It means the information from attributes (*resp.* structures) is decoded as attributes (*resp.* structures). λ_c is a hyper-parameter to weight the cross reconstruction stream.

4.2.4 Adversarial Distribution Matching

The second term $D_{KL}[q_\phi(z_x,z_a|x_i,a_i)||p(z_x,z_a)]$ in Eq. 4–2 indicates a prior distribution matching for the latent factors z_x and z_a. Learning joint prior is a specific topic that could be explored in the future with the guidance of recent works [116-118], here we set $p(z_x,z_y)=p(z)p(z)$ for simplicity, where $p(z)$ is the prior distribution for the *shared-latent space*. Taking Eq. 5–4 into consideration, the prior regularization term is formulated as:

$$\begin{aligned}D_{KL}[q_\phi(z_x,z_a|x_i,a_i)||p(z_x,z_a)] = &D_{KL}[q_{\phi_x}(z_x|x_i)||p(z)]\\ &+D_{KL}[q_{\phi_a}(z_a|a_i)||p(z)]\end{aligned} \tag{4–7}$$

Algorithm 4–1 Structure-attribute Transformer (SAT)

Input: A graph $\mathcal{G}$, the adjacency matrix $A \in R^{N\times N}$, the attributes of attribute-observed nodes $X^o \in \mathbb{R}^{N\times F}$, the learning hyper-parameters such as λ_c.

Output: The restored node attributes $X^u \in R^{N_u\times F}$ or missing links in A.

while not converged **do**
    # encoding for attributes and structures
    Calculate the latent codes Z_{X^o} by encoder E_X with X^o as input.
    Calculate the latent codes Z_A by encoder E_A with A as input.
    # distribution matching
    Make embedding lookup operation to obtain Z_{A^o} from Z_A.
    Get samples for adversarial distribution matching from Gaussian prior $z_p \sim \mathcal{N}(0, 1)$.
    Conduct the paired structure-attribute matching by Eq.4–6 and adversarial distribution matching by Eq.4–8.
    # objective function and optimization
    Calculate loss by Eq.4–9 and optimize the network parameters.
end while

Eq. 5–8 states the regularization that matches $q_{\phi_x}(z_x|x_i)$ and $q_{\phi_a}(z_a|a_i)$ to prior $p(z)$.

Since it is not easy to derive explicit formulations for complicated priors in KL divergence, SAT employs the adversarial distribution matching that can impose an arbitrary prior distribution for the latent codes without hard derivation [52]. Following [52], the prior regularization loss in adversarial distribution matching is written as:

$$\begin{aligned}\min_{\psi}\max_{\phi_x,\phi_a}\mathcal{L}_{adv} = &-\mathbb{E}_{z_p\sim p(z)}[\log\mathcal{D}(z_p)]\\ &-\mathbb{E}_{z_x\sim q_{\phi_x}(z_x|x_i)}[\log(1-\mathcal{D}(z_x))]\\ &-\mathbb{E}_{z_p\sim p(z)}[\log\mathcal{D}(z_p)]\\ &-\mathbb{E}_{z_a\sim q_{\phi_a}(z_a|a_i)}[\log(1-\mathcal{D}(z_a))]\end{aligned} \tag{4–8}$$

where ψ is the parameters of the shared discriminator $\mathcal{D}$. z_p indicates true samples sampled from the prior $p(z)$ for adversarial learning.

The employed adversarial distribution matching in Eq. 4–8 has several advantages compared to the KL divergence in Eq. 4–7. The KL divergence tries to match $q_{\phi_x}(z_x|x_i)$ to prior $p(z)$, which will have risk to lose the information from input x_i. By contrast, the

adversarial distribution matching in latent space makes the posterior $q_{\phi_x}(z_x|x_i)$ to be the aggregated posterior $q_{\phi_x}(z_x)$, which encourages z_x to match the whole distribution of $p(z)$ [51-52]. Accordingly, z_a can match the whole distribution of $p(z)$ in similar way. Meanwhile, the mode collapse problem in adversarial learning could be avoided since our method involves a reconstruction operation which encourages the latent embeddings to match both the prior and the entire true data distribution [110].

4.2.5 Objective Function and Implementation

Taking the above into summary, the objective function of SAT is:

$$\min_{\Theta} \max_{\Phi} \mathcal{L} = \mathcal{L}_r + \mathcal{L}_{adv} \tag{4–9}$$

where $\Theta = \{\theta_x, \theta_a, \phi_x, \phi_a, \psi\}$ and $\Phi = \{\phi_x, \phi_a\}$. By leveraging the *shared-latent space* assumption, SAT leverages the correlation between observed node attributes and structures and thus facilitates learning on attribute-missing graphs. It is worthwhile to point out that SAT cannot handle attribute-incomplete graphs. On attribute-incomplete graphs, the incomplete node attributes would cause too much information gap for structures and attributes, and SAT based on the *shared-latent space* assumption is not appropriate.

For the implementation, SAT consists of three components: (1) self-reconstruction stream, (2) cross-reconstruction stream, and (3) adversarial distribution matching. In the self-reconstruction stream, both the observed attributes X^o and structures A are encoded as latent codes Z_{X^o} and Z_A, respectively. Then Z_{X^o} and Z_A are decoded as X^o and A, respectively. In the cross-reconstruction stream, the structure embedding Z_{A^o} of attribute-observed nodes are obtained through an embedding lookup layer from Z_A. Then, the latent codes Z_{X^o} and Z_{A^o} are decoded as A^o and X^o, respectively. These appear in Figure 5–2, in which E_X is a two-layer MLP, E_A is a two-layer GNN, D_X is a two-layer MLP and D_A is a two-layer MLP followed by a *sigmoid* function. In the adversarial distribution matching module, we apply adversarial learning between Z_{X^o}, Z_A and samples from a standard normal distribution $\mathcal{N}(0, 1)$, sharing the same two-layer MLP discriminator. And *Relu* is used as the non-linear activation function for all three modules. The concise description of SAT is shown in Algorithm 4–1.

4.2.6 Time Complexity Analysis

Stochastic training of DNN methods involves two steps, the forward and backward computations. SAT includes a paired structure-attribute matching strategy and an adver-

Table 4–1 The statistics of seven datasets. In this table, "attribute form" means the attribute style. "#avg hot num" means the average hot number for multi-hot attributes of nodes. #class indicates the number of categories.

	Cora	Citeseer	Steam	Pubmed	Coauthor -CS	Amazon -Computer	Amazon -Photo
#nodes	2,708	3,327	9,944	19,717	18,333	13,752	7,650
#edges	5,278	4,228	266,981	44,324	81,894	245,861	119,081
#graph density	0.07%	0.04%	0.26%	0.01%	0.02%	0.13%	0.20%
#attribute dim	1,433	3,703	352	500	6,805	767	745
#avg hot num	18.17	31.6	8.45	-	-	267.23	258.81
#class	7	6	-	3	15	10	8
attribute form	categorical	categorical	categorical	real-valued	real-valued	categorical	categorical

sarial distribution matching strategy. We thus decompose the time complexity of SAT into two parts, namely the time complexity of the paired structure-attribute matching and the time complexity of the adversarial distribution matching.

In the paired structure-attribute matching, a GNN backbone encodes the structures into latent codes and a MLP module encodes the attributes into latent embeddings. Following the analysis of GNN models in [48], the time complexity of the GNN backbone is $O(|\mathcal{E}|)$, where $|\mathcal{E}|$ denote the number of edges. And for the MLP module that encodes node attributes, the time complexity is $O(N_o F)$, where N_o is the number of attribute-observed nodes and F is the attribute dimension. Since the GNN backbone and the MLP module can be calculated parallelly, then the time complexity for the paired structure-attribute matching is $O(\max\{|\mathcal{E}|, N_o F\})$. After the calculation of the paired structure-attribute matching, the adversarial distribution matching can be conducted to impose a prior distribution on the latent codes. The time complexity for the adversarial distribution matching is $O(N_o d)$, where d is the latent dimension.

In summary, the time complexity of the non-parallel SAT is $O(|\mathcal{E}| + N_o F + N_o d)$, and the parallel counterpart is $O(\max\{|\mathcal{E}|, N_o F\} + N_o d)$. Note that SAT has a flexible GNN backbone such as GCN [1] and GraphSage [2]. This indicates SAT can be scalable to much larger datasets when a scalable GNN backbone is employed.

4.3 Experiments and Analysis

In this section, we will first introduce the datasets and baselines as well as the experimental settings. Then, the experiments for *node attribute completion* and link prediction are introduced. We further provide some visualization results for better comprehension of SAT.

4.3.1 Dataset Description

We evaluate the proposed SAT on seven real-world datasets to quantify the performance.

- Cora. Cora [132] is a citation graph with 2,708 papers as nodes and 10,556 citation links as edges. The attribute vector of each node indicates whether the corresponding paper contains specific word tokens, and it is represented as a multi-hot vector with dimension 1,433.
- Citeseer. Citeseer [133] is also a citation graph which contains 9,228 edges and 3,327 papers. After processing of the content, 3,703 distinct words compose the attribute corpus. Each attribute vector is formed from the corpus and expressed by a multi-hot vector with dimension 3,703.
- Steam. Steam is a dataset collected from a game website with user-bought behaviors of 9,944 items and 352 tags. We count the co-purchase frequency between every two games and make a sparse item co-purchase graph through binarization operation with the threshold as 10. After that, we obtain 533,962 edges for this graph. The tag corpus constructs the multi-hot attribute vector for each item with dimension 352.
- Pubmed. Pubmed [134] is a citation graph with 19,717 nodes and 88,651 edges. Each attribute vector is described by a Term Frequency-Inverse Document Frequency (TF-IDF) vector from 500 distinct terms.
- Coauthor-CS. Coauthor-CS [135] is a coauthor dataset based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. In this dataset, we have 18,333 nodes that are authors and 81,894 edges that mean two authors are connected if they coauthored a paper. Node attributes are frequencies of the 6,805 keywords from each author's papers.
- Amazon-Computer and Amazon-Photo. Amazon-Computer [135] and Amazon-Photo [135] are from segments of the Amazon co-purchase graph [136]. In

Amazon-computer, there are 13,752 items and 245,861 edges. In Amazon-Photo, there are 7,650 items and 119,081 edges. For both datasets, multi-hot bag-of-words encoded from the product reviews construct the multi-hot node attributes.

Among these datasets, attributes of Cora, Citeseer, Steam, Amazon-Computer and Amazon-Photo are categorical and represented as multi-hot vectors. For Pubmed and Coauthor-CS, attributes are real-valued and represented as scalars. More details are shown in Table 4–1.

4.3.2 Baselines

Node Attribute Completion. Since there is no specialized method for the *node attribute completion* task, we compare our SAT with the following representative baselines selected from five aspects: NeighAggre is from the *classical aggregation* aspect, VAE is from the *auto-encoding* aspect, GNN methods are from the *GNN-encoding* aspect, GraphRNA and ARWMF are from the *attributed random walking* aspect and Hers is from the *cold-start recommendation* aspect.

- NeighAggre. NeighAggre [137] aggregates the neighbors' attributes for nodes without attributes through mean pooling. NeighAggre is a classical profiling algorithm, which is simple, but usually provides strong empirical performance. When a neighbor' attributes are missing, we do not regard it as the aggregation node. We use the one-hop neighbors as a node's neighbors here.
- VAE. VAE [100] is one well-known generative method. In our setting, we make normal VAE for the attribute-observed nodes, encoding attributes of these nodes as latent codes. For test nodes without any attribute, we use neighbor aggregation like NeighAggre in the latent space to obtain the aggregated latent codes for test nodes. Then, the decoder in VAE can be used to restore attributes for test nodes.
- GCN, GraphSage and GAT. GCN [1], GraphSage [2] and GAT [26] are three representative GNN methods in recent years. In our problem, only graph structures are used as input and encoded as latent codes. Then the latent codes are decoded as node attributes. In the test stage, the latent codes of test nodes are used to restore attributes by the decoder of GNN. Note that we do not use the zero-filling trick here since this task require reconstruction of attributes and the zero-filling trick largely deteriorates the performance.
- GraphRNA [39] and ARWMF [40] are two representative attributed random

walk based methods. In attributed random walk based methods, graphs with node attributes are taken as bipartite graphs and then random walks based approaches are applied to learn node embeddings. Thereby, attributed random walk based methods can potentially work on attribute-missing graphs. Note that there are three variants of ARWMF in [40], we introduce the third variant as the baseline since it generally shows better performance than the other two variants [40].

- Hers. Since the attribute-completion on attribute-missing graphs is similar to the cold-start recommendation problem. We thus introduce one representative cold-start recommendation method called Hers [138] to be a comparison method.

Link Prediction. Even on attribute-missing graphs, link prediction is still an important task. We choose baselines for link prediction comparison from two aspects: the *structure-only* aspect and the *fused-structure-attribute* aspect. SPM, DeepWalk, Node2Vec and Hers are only suitable for the *structure-only* aspect. GraphRNA and ARWMF are only feasible for the *fused-structure-attribute* aspect. GAE and VGAE are feasible for both aspects.

- SPM. Structure perturbation method (SPM) [139] is one classic link prediction model on graphs. It assumes that the regularity of a graph is reflected in the consistency of structures before and after structural perturbations. Based on this assumption, it proposes a universal structural consistency index for link prediction on graphs.
- DeepWalk. DeepWalk [11] learns node representation from random walk sequences on graphs. It only considers graph structures in the learning process.
- Node2vec. Node2vec [17] extends DeepWalk by designing a biased random walk to control the Bread First Search (BFS) and Deep First Search (DFS). It also learns node embeddings with only graph structures.
- GAE. Graph Auto-encoder (GAE) [125] combines GCN and the auto-encoder theory. Specifically, GCN plays as a graph convolution encoder to replace the MLP encoder in vanilla auto-encoder [100]. When GAE works for the *fused-structure-attribute* aspect, it requires structures and attributes as a whole input. Thus we use zero-filling, a common trick in data mining community, for the attribute-missing nodes. Furthermore, we also use GAT as the graph convolution encoder called GAE(GAT) as another baseline method.
- VGAE. Variational Graph Auto-encoder (VGAE) [125] performs variational in-

ference for GAE with reparameterization tricks. We use it in the same way as GAE and also combine it with GAT called VGAE(GAT) as another baseline.

- Hers, GraphRNA and ARWMF. Since Hers [138], GraphRNA [39] and ARWMF [40] are able to learn node embeddings, we thus also introduce them here for the link prediction task.

4.3.3 Experimental Setups

For *node attribute completion*, we randomly sample 40% nodes with attributes as training data and 10% as validation data and the rest 50% as test data. For link prediction, we randomly sample 60%, 20% and 20% links with equivalent non-links as train, validation and test data, respectively.

For the cold-start recommendation method Hers [138], we use the graph data as the social context and the attribute matrix as the target to complete. For random walk based methods (DeepWalk[1], Node2Vec[2]) and attributed random walk based methods (GraphRNA and ARWMF), their hyper-parameters such as number of walks, walk length and window size are set as the default according to the codes online. All GNN methods use a two-layer graph convolution. For GraphSage, we set the neighborhood sampling size is (5,5) for Cora, Citeseer and Steam, while (10,25) for Pubmed according to the default settings of the online codes[3]. For GraphSage on Coauthor-CS, Amazon-Computer and Amazon-Photo, we found it performs well when neighborhood sampling size is (10,25). For GAT[4], in order to reduce training time and storage, we use one-head attention. By following recent works [1, 26], we set the latent dimension as 64 for all learning-based methods. Learning rate is 0.005. Dropout rate equals 0.5, and the maximum iteration number is 1,000. Adam optimizer is applied for them to learn the model parameters.

For our distribution matching based SAT, we set the generative step as 2 and discriminative step as 1. For balanced comparison, the hyper-parameters of different GNN backbones are the same as the GNN baselines. The hyper-parameter λ_c ranges in $[0.1, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 100.0]$. According to the performance on the validation set, for the *node attribute completion* task, we have λ_c = 10.0, 10.0, 50.0, 10.0, 100.0, 100.0, 100.0 for Cora, Citeseer, Pubmed, Steam, Coauthor-CS, Amazon-Computer

[1] https://github.com/phanein/deepwalk

[2] https://github.com/aditya-grover/node2vec

[3] https://github.com/williamleif/graphsage-simple/

[4] https://github.com/Diego999/pyGAT

and Amazon-Photo, respectively. For the link prediction task, we set λ_c = 10.0, 10.0, 1.0, 10.0, 0.1, 0.1, 0.1 for Cora, Citeseer, Pubmed, Steam, Coauthor-CS, Amazon-Computer and Amazon-Photo, respectively. For all methods, The best trained model is chosen for testing according to the performance on the validation set.

For datasets with categorical attributes, weighted Binary Cross Entropy loss (BCE) is applied. The weight put on non-zero values equals $\frac{\#zero\ count}{\#non-zero\ count}$ which is calculated from the training attribute matrix. For datasets with real-valued attributes, Mean Square Error (MSE) loss is used. We do not use GraphSage as the baseline for link prediction since it takes much time for random neighborhood sampling in each iteration. The experiments are conducted 5 times, and then the mean value is adopted as the performance. The method is implemented by Pytorch on a machine with one Nvidia TitanX GPU.

4.3.4 Node Attribute Completion

4.3.4.1 Necessity of *Node Attribute Completion*

Node attribute completion is a new problem on graphs and distinguishes from other existing problems including label prediction and graph embedding.

Although node labels could be one kind of attributes, *node attribute completion* is quite different from the widely explored label prediction problem on graphs [1] due to their *characteristic*, *functionality* and *methodology*. From the aspect of *characteristic*, labels are designed to be clean and only limited to categorical values, while attributes are more general, noisy and either numeric or categorical. From the aspect of *functionality*, label prediction on graphs is one subsequent task of *node attribute completion*. Many label prediction algorithms require both structures and attributes as inputs to achieve superior performance [47]. *Node attribute completion* enables recovering unknown attributes of nodes, which is useful for many subsequent tasks, including label prediction. From the aspect of *methodology*, when typical label-prediction methods are applied to *node attribute completion*, such as GCN [1] used to predict labels (categorical attributes) by minimizing classication loss, they ignore the joint relationship between node structures and attributes. Furthermore, compare to the graph embedding problem [11, 17, 125], *node attribute completion* restores high-dimensional and human-understandable attributes while graph embedding learns to represent nodes or graphs in low-dimensional and hard-perceptive vectors.

4.3.4.2 Evaluation Measures

In the *node attribute completion* task, whether the restored attributes can benefit real-world applications should be considerable. Consequently, we propose to measure the quality of restored attributes from both the *node level* and the *attribute level* with two real-world applications.

- Node classification. This task aims to evaluate whether the restored attributes can serve as data augmentation and benefit the classification model. In this task, we use the restored attributes of test nodes to make comparison of node classification among different methods. In other words, this task evaluates the overall quality of restored attributes by classifiers, which is also termed as the evaluation in the *node level*. We implement this task on datasets with class labels.
- Profiling. Profile provides a cognitive description for objects such as key terms of papers on Cora and tags of items on Steam. Profiling aims to predict the possible profile for test nodes, and we use Recall@k and NDCG@k as the evaluation metrics. In other words, this task evaluates the recall and ranking quality of restored attributes in the *attribute level*. For this task, we compare different methods on datasets with categorical attributes.

4.3.4.3 Node Classification

In this task, the restored node attributes are split into 80% train data and 20% test data with five-fold validation in 10 times. We consider two classifiers, including MLP and GCN, both with the class information as supervision. Three settings are designed to conduct the comparisons: the *node-attribute-only* approach, the *graph-structure-only* approach, and the *fused* approach. In the *node-attribute-only* approach, we directly use the restored attributes and a two-layer MLP as the classifier to do the classification task. In the *graph-structure-only* approach, only graph structures are applied and this has been studied by many methods such as DeepWalk [11], Node2Vec [17] and GCN [1]. DeepWalk and Node2Vec both aim to learn node embeddings, and then an MLP classifier is used. GCN is an end-to-end method which learns the node embeddings supervised by the classification loss. In the fused approach, we combine restored node attributes with structures by a GCN classifier.

Table 4–2 shows the classification performance, where "X" indicates the *node-attribute-only* approach, "A" indicates the *structure-only* approach and "A+X" is the

Table 4–2 Node classification of the *node-level* evaluation for *node attribute completion*. The first column with "X", "A" and "A+X" indicates three settings to do node classification with only attributes, only structures and the fused one. Note that SAT has an extendable GNN backbone, so we combine it with different models as SAT(GCN), SAT(GraphSage) and SAT(GAT). SAT(GCN)-no self, SAT(GCN)-no cross and SAT(GCN)-no adver respectively denotes SAT without self-reconstruction terms, cross-reconstruction terms and adversarial learning terms. The term "True attributes" indicates we use the ground truth attributes to do node classification.

	attribute completion method	classification method	Cora	Citeseer	Pubmed	Coauthor -CS	Amazon -Computer	Amazon -Photo
X	NeighAggre	MLP	0.6248	0.5539	0.5150	0.7562	0.8365	0.8846
	VAE	MLP	0.2826	0.2551	0.4008	0.2317	0.3747	0.2598
	GCN	MLP	0.3943	0.3768	0.3992	0.2180	0.3660	0.2683
	GraphSage	MLP	0.4852	0.3933	0.4013	0.2317	0.3747	0.2598
	GAT	MLP	0.4143	0.2129	0.3996	0.2317	0.3747	0.2598
	Hers	MLP	0.3046	0.2585	0.4004	0.2317	0.3747	0.2598
	GraphRNA	MLP	0.7581	0.6320	0.6035	**0.7710**	0.6968	0.8407
	ARWMF	MLP	0.7769	0.2267	**0.6180**	0.2320	0.5608	0.4675
	SAT(GCN)-no self	MLP	0.7074	0.4976	0.4000	0.7504	0.7410	0.8585
	SAT(GCN)-no cross	MLP	0.3036	0.2289	0.4023	0.2317	0.3748	0.2613
	SAT(GCN)-no adver	MLP	0.7587	0.6051	0.4680	0.6879	0.7356	0.8629
	SAT(GCN)	MLP	0.7644	0.6010	0.4652	0.7592	0.7410	0.8762
	SAT(GraphSage)	MLP	0.7032	0.5936	0.4585	0.6637	**0.8396**	**0.9035**
	SAT(GAT)	MLP	**0.7937**	**0.6475**	0.4618	0.7672	0.8201	0.8976
	True attributes	MLP	0.7618	0.7174	0.656	0.9396	0.8423	0.9151
A	-	DeepWalk+MLP	0.7149	0.4802	0.6917	0.7561	0.8444	0.8955
	-	Node2Vec+MLP	0.6830	0.4422	0.6721	0.7554	0.8415	0.8908
	-	GCN	**0.7631**	**0.5651**	**0.7125**	**0.8370**	**0.8785**	**0.9117**
A+X	NeighAggre	GCN	0.6494	0.5413	0.6564	0.8031	0.8715	0.901
	VAE	GCN	0.3011	0.2663	0.4007	0.2335	0.4023	0.3781
	GCN	GCN	0.4387	0.4079	0.4203	0.2180	0.3974	0.3656
	GraphSage	GCN	0.5779	0.4278	0.4200	0.2335	0.4019	0.3784
	GAT	GCN	0.4525	0.2688	0.4196	0.2334	0.4034	0.3789
	Hers	GCN	0.3405	0.3229	0.4205	0.2334	0.4025	0.3794
	GraphRNA	GCN	0.8198	0.6394	**0.8172**	**0.8851**	0.8650	0.9207
	ARWMF	GCN	0.8025	0.2764	0.8089	0.8347	0.7400	0.6146
	SAT(GCN)-no self	GCN	0.7727	0.5358	0.4197	0.8575	0.8455	0.9127
	SAT(GCN)-no cross	GCN	0.3402	0.2698	0.4204	0.3499	0.4394	0.3846
	SAT(GCN)-no adver	GCN	0.8231	0.6609	0.7371	0.8161	0.8439	0.9123
	SAT(GCN)	GCN	0.8327	0.6599	0.7537	0.8576	0.8519	0.9163
	SAT(GraphSage)	GCN	0.8255	0.6547	0.7360	0.8001	**0.8834**	0.9234
	SAT(GAT)	GCN	**0.8579**	**0.6767**	0.7439	0.8402	0.8766	**0.9260**
	True attributes	GCN	0.8493	0.7348	0.8723	0.9186	0.9097	0.9412

fused one. Considering results of the *node-attribute-only* approach, we have the following observations:

- SAT generally presents better performance than baseline methods. For example, compared to NeighAggre, SAT(GCN) reaches nearly 14% and 5% gain on Cora and Citeseer, respectively. On Pubmed, it seems that NeighAggre suits this dataset well, but it deteriorates quickly when attribute-observed nodes are less, which is

shown in Section 4.3.4.5.

- The attributed random walk based methods (GraphRNA and ARWMF) show the best performance on Pubmed and Coauthor-CS. It is mainly because the attributed random walk based methods apply random walks on the node-attribute bipartite graphs, which could potentially capture the correlation between attribute dimensions and facilitate the *node attribute completion* task. On Pubmed and Coauthor that have real-valued node attributes, the correlation between attributes could be more obvious because attributes have relative sizes for different attribute dimensions, and thus GraphRNA and ARWMF show better performance. Instead, the proposed SAT cannot capture the correlation between attributes and thus perform worse than GraphRNA and ARWMF. On other datasets, SAT perform better than GraphRNA and ARWMF mainly because SAT has the advantage of extracting features by graph convolution techniques.
- From the results of Hers, we can see that simply taking the graph structures as one kind of side information to augment the recommendation model cannot well exploit the structures and node attributes. By contrast, the proposed SAT explicitly models the joint distribution of node structures and attributes and thus shows superior performance compared to Hers.
- The performance of SAT(GCN) gets closer to that of true attributes. SAT(GAT) even gets better than the true attributes on Cora mainly because the structural information may take more important role for classification and the restored attributes contain transferred structural information.
- Both SAT(GCN)-no self, SAT(GCN)-no cross and SAT(GCN)-no adver perform worse than SAT(GCN) because the incompleteness of SAT cannot guarantee the joint distribution modeling.
- SAT with different GNN backbones generally show better performance other baselines, which also states SAT is extendable and robust with different choices of graph convolution filters.

Comparing the results between the *structure-only* approach and the *fused* one, we have the following observations.

- The restored attributes from SAT(GCN) can augment the GCN classifier with 6.96%, 9.48% and 4.12% gain on Cora, Citeseer and Pubmed, respectively. While NeighAggre fails and harms the GCN performance with the figures as 11.37%,

2.38% and 5.61% on Cora, Citeseer and Pubmed, respectively.

- The *classical aggregation* method NeighAggre, the *auto-encoding* method VAE, and the *GNN-encoding* methods restore inferior attributes. They hurt the GCN classifier since they are not specifically designed for attribute-missing graphs.

4.3.4.4 Profiling

For the profiling task, restored attributes are probabilities that the node may have in each attribute dimension. High-quality restored attributes should have high probabilities in specific dimensions as the true attributes. The results of Recall@k and NDCG@k are shown in Table 4–3. From Table 4–3, we can summarize as follows.

- NeighAggre performs the worst among all methods on the three datasets. It is not a learning algorithm and cannot perform robustly in this fine-grained *attribute-level* evaluation.
- The attributed random walk based methods have the potential to capture the correlation between attribute dimensions, GraphRNA and ARWMF show the most competitive performance on this *attribute-level* evaluation. However, the applied random walks may introduce noise to statistically represent the original graph data. Thereby, GraphRNA and ARWMF cannot outperform SAT on this fine-grained *attribute-level* evaluation.
- SAT achieves superior performance over other methods because it restores attributes based on the transformation knowledge.

4.3.4.5 Less Attribute-Observed Nodes

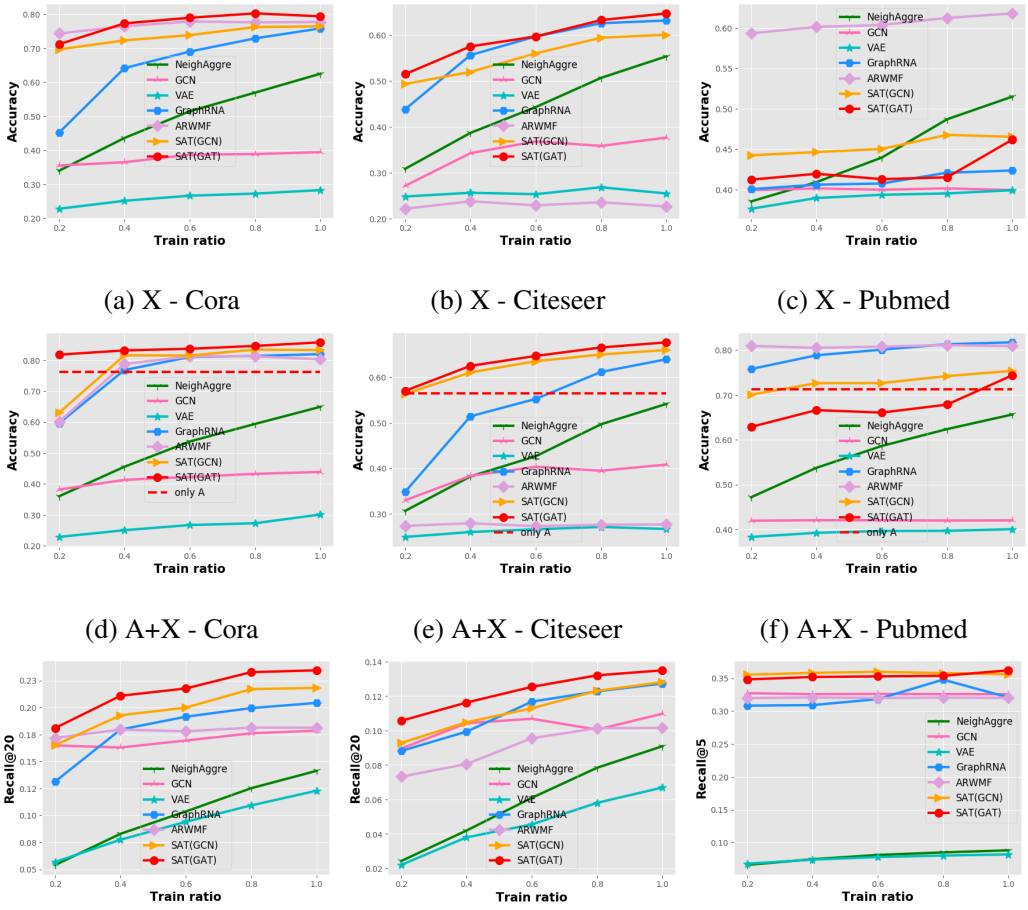
The attribute-observed nodes are necessary and supervise the learning on attribute-missing graphs. In some scenarios, this supervision could be less, so it is curious to see whether SAT can still restore reliable and high-quality node attributes when less attribute-observed nodes are available. We experiment to see the node classification and profiling performance in this scenario.

Figure 4–4 (a)(b)(c) show the node classification results when only node attributes are used. From these figures, we can see that SAT(GCN) generally performs much better than other methods. NeighAggre deteriorates quickly and the gap is more obvious when attribute-observed nodes are less. Figure 4–4 (d)(e)(f) show the node classification performance when "A+X" is used. In these figures, the dotted line indicates only "A"

Table 4–3 Profiling of the *attribute-level* evaluation for *node attribute completion*. Note that we use top 3, 5, 10 to evaluate the performance on Steam since the average non-zero hot number of node attributes on Steam is quite small.

	Method	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
Cora	NeighAggre	0.0906	0.1413	0.1961	0.1217	0.1548	0.1850
	VAE	0.0887	0.1228	0.2116	0.1224	0.1452	0.1924
	GCN	0.1271	0.1772	0.2962	0.1736	0.2076	0.2702
	GraphSage	0.1284	0.1784	0.2972	0.1768	0.2102	0.2728
	GAT	0.1350	0.1812	0.2972	0.1791	0.2099	0.2711
	Hers	0.1226	0.1723	0.2799	0.1694	0.2031	0.2596
	GraphRNA	0.1395	0.2043	0.3142	0.1934	0.2362	0.2938
	ARWMF	0.1291	0.1813	0.296	0.1824	0.2182	0.2776
	SAT(GCN)-no self	0.1378	0.2018	0.3339	0.1931	0.2360	0.3052
	SAT(GCN)-no cross	0.1224	0.1724	0.2823	0.1686	0.2023	0.2599
	SAT(GCN)-no adver	0.1356	0.1966	0.3179	0.1924	0.2331	0.2971
	SAT(GCN)	0.1508	0.2182	0.3429	0.2112	0.2546	0.3212
	SAT(GraphSage)	0.1356	0.1981	0.3165	0.1905	0.2320	0.2947
	SAT(GAT)	**0.1653**	**0.2345**	**0.3612**	**0.2250**	**0.2723**	**0.3394**
Citeseer	NeighAggre	0.0511	0.0908	0.1501	0.0823	0.1155	0.1560
	VAE	0.0382	0.0668	0.1296	0.0601	0.0839	0.1251
	GCN	0.0620	0.1097	0.2052	0.1026	0.1423	0.2049
	GraphSage	0.0612	0.1097	0.2058	0.1003	0.1393	0.2034
	GAT	0.0561	0.1012	0.1957	0.0878	0.1253	0.1872
	Hers	0.0576	0.1025	0.1973	0.0904	0.1279	0.1900
	GraphRNA	0.0777	0.1272	0.2271	0.1291	0.1703	0.2358
	ARWMF	0.0552	0.1015	0.1952	0.0859	0.1245	0.1858
	SAT(GCN)-no self	0.0679	0.1163	0.2140	0.1167	0.1570	0.2209
	SAT(GCN)-no cross	0.0564	0.1013	0.1963	0.0863	0.1238	0.1860
	SAT(GCN)-no adver	0.0705	0.1168	0.2145	0.1196	0.1582	0.2223
	SAT(GCN)	0.0764	0.1280	0.2377	0.1298	0.1729	0.2447
	SAT(GraphSage)	0.0704	0.1163	0.2174	0.1179	0.1563	0.2227
	SAT(GAT)	**0.0811**	**0.1349**	**0.2431**	**0.1385**	**0.1834**	**0.2545**
Amazon-Computer	NeighAggre	0.0321	0.0593	0.1306	0.0788	0.1156	0.1923
	VAE	0.0255	0.0502	0.1196	0.0632	0.0970	0.1721
	GCN	0.0273	0.0533	0.1275	0.0671	0.1027	0.1824
	GraphSage	0.0269	0.0528	0.1278	0.0664	0.1020	0.1822
	GAT	0.0271	0.0530	0.1278	0.0673	0.1028	0.1830
	Hers	0.0273	0.0525	0.1273	0.0676	0.1025	0.1825
	GraphRNA	0.0386	0.0690	0.1465	0.0931	0.1333	0.2155
	ARWMF	0.0280	0.0544	0.1289	0.0694	0.1053	0.1851
	SAT(GCN)-no self	0.039	0.0698	0.1506	0.0956	0.1368	0.2229
	SAT(GCN)-no cross	0.0287	0.0554	0.1295	0.0691	0.1054	0.1852
	SAT(GCN)-no adver	0.0322	0.0603	0.1364	0.0790	0.1170	0.1984
	SAT(GCN)	0.0391	0.0703	0.1514	0.0963	0.1379	0.2243
	SAT(GraphSage)	0.0419	0.0738	0.1562	0.1030	0.1457	0.2333
	SAT(GAT)	**0.0421**	**0.0746**	**0.1577**	**0.1030**	**0.1463**	**0.2346**
Amazon-Photo	NeighAggre	0.0329	0.0616	0.1361	0.0813	0.1196	0.1998
	VAE	0.0276	0.0538	0.1279	0.0675	0.1031	0.1830
	GCN	0.0294	0.0573	0.1324	0.0705	0.1082	0.1893
	GraphSage	0.0295	0.0562	0.1322	0.0712	0.1079	0.1896
	GAT	0.0294	0.0573	0.1324	0.0705	0.1083	0.1892
	Hers	0.0292	0.0574	0.1328	0.0714	0.1094	0.1906
	GraphRNA	0.0390	0.0703	0.1508	0.0959	0.1377	0.2232
	ARWMF	0.0294	0.0568	0.1327	0.0727	0.1098	0.1915
	SAT(GCN)-no self	0.0399	0.0732	0.1583	0.0982	0.1425	0.2330
	SAT(GCN)-no cross	0.0302	0.0578	0.1334	0.0738	0.1112	0.1923
	SAT(GCN)-no adver	0.0354	0.0656	0.1463	0.0880	0.1287	0.2149
	SAT(GCN)	0.0410	0.0743	0.1597	0.1006	0.145	0.2359
	SAT(GraphSage)	**0.0483**	**0.0766**	0.1601	**0.1082**	0.1475	0.2402
	SAT(GAT)	0.0427	0.0765	**0.1635**	0.1047	**0.1498**	**0.2421**
	Method	Recall@3	Recall@5	Recall@10	NDCG@3	NDCG@5	NDCG@10
Steam	NeighAggre	0.0603	0.0881	0.1446	0.0955	0.1204	0.1620
	VAE	0.0564	0.0820	0.1251	0.0902	0.1133	0.1437
	GCN	0.2392	0.3258	0.4575	0.3366	0.4025	0.4848
	GraphSage	0.2356	0.3068	0.4568	0.3311	0.3892	0.4817
	GAT	0.2395	0.3431	0.4649	0.3364	0.4138	0.4912
	Hers	0.2387	0.3346	0.4586	0.3305	0.4059	0.4842
	GraphRNA	0.2490	0.3208	0.4372	0.3437	0.4023	0.4755
	ARWMF	0.2104	0.3201	0.4512	0.3066	0.3877	0.4704
	SAT(GCN)-no self	0.2429	0.3116	0.4614	0.3414	0.3969	0.4889
	SAT(GCN)-no cross	0.2382	0.3381	0.4611	0.3282	0.4057	0.4835
	SAT(GCN)-no adver	0.2371	0.3382	0.4707	0.3353	0.4114	0.4966
	SAT(GCN)	0.2527	0.3560	0.4933	0.3544	0.4332	0.5215
	SAT(GraphSage)	0.2518	0.3470	0.4845	0.3529	0.4271	0.5133
	SAT(GAT)	**0.2536**	**0.3620**	**0.4965**	**0.3585**	**0.4400**	**0.5272**

is used by a GCN classifier, which serves as a criterion to evaluate whether the restored attributes can enhance the GCN classifier. It is clear that SAT(GCN) reaches superior



(a) X - Cora (b) X - Citeseer (c) X - Pubmed

(d) A+X - Cora (e) A+X - Citeseer (f) A+X - Pubmed

(g) Recall - Cora (h) Recall - Citeseer (i) Recall - Steam

Figure 4–4 Node classification and profiling performance with less attribute-observed nodes. (a)-(c) illustrates the results of node classification with "X" setting. (d)-(f) shows the results of node classification with "A+X" setting. The dashed line is a criterion to criticize whether the restored attributes can enhance the GCN classifier. (g)-(i) shows the results of profiling task. Train ratio means the ratio of samples from the original train data.

performance than baselines on Cora and Citeseer. On Pubmed with real-valued node attributes, GraphRNA and ARWMF show better performance because they have the potential to capture the correlation between attribute dimensions. Figure 4–4 (g)(h)(i) indicate the fine-grained profiling performance. These figures also demonstrate the advantage of SAT in restoring fine-grained node attributes.

Table 4–4 Area under curve (AUC) and average precision (AP) of Link prediction task. "A" indicates the *structure-only* aspect and only structures are used. While "A+X" means the *fused structure-attribute* aspect and both structures and attributes are used.

	Method	Cora		Citeseer		Pubmed		Steam		Coauthor -CS		Amazon -Computer		Amazon -Photo	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
A	SPM	0.8627	0.9001	0.8293	0.8646	0.8339	0.8544	0.8848	0.9355	0.9169	0.9385	0.9345	0.9309	0.9372	**0.9416**
	DeepWalk	0.8735	0.9018	0.8082	0.8467	0.8361	0.8560	0.9591	0.9217	0.9174	0.9119	0.8735	0.8657	0.9148	0.9079
	Node2Vec	0.8496	0.8695	0.7644	0.8246	0.8346	0.8884	0.9591	0.9233	0.9359	0.9348	0.8551	0.8355	0.8770	0.8608
	GAE(GCN)	0.8281	0.8309	0.8254	0.8344	0.7072	0.6548	0.9804	0.9772	0.7674	0.7307	0.8436	0.7755	0.9135	0.8960
	GAE(GAT)	0.8340	0.8642	0.8563	0.8777	0.7028	0.6559	0.9730	0.9637	0.7702	0.7359	0.8408	0.8419	0.8802	0.8493
	VGAE(GCN)	0.8433	0.8616	0.8396	0.8456	0.8011	0.8171	0.9738	0.9693	0.8591	0.8440	0.9345	0.9281	0.9359	0.9329
	VGAE(GAT)	0.7267	0.7329	0.7046	0.7156	0.7011	0.6264	0.8227	0.8094	0.7370	0.6935	0.7492	0.6564	0.8313	0.7500
	Hers	0.6908	0.7169	0.6779	0.7197	0.5001	0.5182	0.9869	0.9818	0.5433	0.5338	0.5314	0.5122	0.5809	0.5589
A+X	GAE(GCN)	0.8375	0.8374	0.8227	0.8245	0.7947	0.7855	0.9735	0.9643	0.9409	**0.9386**	0.9361	0.9336	0.9336	0.9296
	GAE(GAT)	0.8575	0.8493	0.8387	0.8661	0.7839	0.7821	0.9717	0.9706	**0.9452**	0.9374	0.7945	0.7836	0.9441	0.9282
	VGAE(GCN)	0.7910	0.8018	0.7962	0.8011	0.8229	0.8156	0.9581	0.9454	0.8960	0.8936	0.9315	0.924	0.9205	0.9202
	VGAE(GAT)	0.7062	0.7172	0.7069	0.7061	0.7861	0.7578	0.9728	0.9195	0.7678	0.7382	0.7609	0.6909	0.7125	0.6699
	GraphRNA	0.8579	0.8695	0.8298	0.8580	0.8406	0.8445	0.9157	0.8807	0.8688	0.8579	0.7115	0.6859	0.8359	0.8010
	ARWMF	0.7601	0.7770	0.7260	0.7646	0.7022	0.7320	0.7832	0.6580	0.8582	0.8503	0.8325	0.8251	0.9102	0.8949
	SAT(GCN)-no self	0.8549	0.8432	0.8542	0.8550	0.7661	0.7497	0.9702	0.9599	0.8166	0.8029	0.8593	0.8253	0.8888	0.8470
	SAT(GCN)-no cross	0.8048	0.7894	0.8042	0.8097	0.7817	0.7998	0.9617	0.9560	0.7990	0.7592	0.8877	0.8918	0.8950	0.8556
	SAT(GCN)-no adver	0.8438	0.8405	0.8438	0.8459	0.7851	0.8018	0.9703	0.9605	0.9005	0.8920	0.9363	0.9303	0.9466	0.9390
	SAT(GCN)	0.8554	0.8500	0.8569	0.8566	0.8258	0.8039	0.9713	0.9636	0.9077	0.8977	**0.9433**	**0.9368**	**0.9467**	0.9393
	SAT(GAT)	**0.8929**	**0.9018**	**0.8916**	**0.9139**	**0.8510**	**0.8554**	**0.9896**	**0.9881**	0.9138	0.9035	0.9100	0.8940	0.9283	0.9112

4.3.5 Link Prediction

4.3.5.1 Overall Performance

Link prediction is a key problem on graphs which aims to predict the missing links among nodes. It has various applications such as friend recommendation [140]. We conduct the link prediction task on attribute-missing graphs from two aspects: the *structure-only* aspect and the *fused structure-attribute* aspect. The results are shown in Table 4–4.

In Table 4–4, "A" indicates the *structure-only* aspect where only structures are used. While "A+X" means the *fused structure-attribute* aspect where both structures and attributes are used. From this table, we can summarize that:

- SPM with the idea of perturbations on graphs show competitive performance because of its denoising characteristic. However, it is limited in learning deep representations for nodes and cannot beat SAT. Meanwhile, introducing the perturbation idea from SPM to SAT is also an interesting problem that can be explored in future.
- In the "A+X" setting, SAT generally outperforms recent GNN based methods (*e.g.* GAE and VGAE) and attributed random walk based methods (*e.g.* GraphRNA

and ARWMF). Compared to GAE(GAT), SAT(GAT) obtains a 3.73% gain, 6.2% gain and 6.71% gain on Cora, Citeseer and Pubmed respectively. This also indicates that SAT has the advantage of using attribute-observed nodes while recent GNN methods cannot handle the link prediction task in this scenario properly. Compared to GraphRNA and ARWMF, SAT has the advantage of encoding structures with graph convolution scheme and thus show better performance on link prediction task.

- Breaking the unified loss in Eq. 4–9 causes deterioration to the performance since it cannot guarantee the joint distribution modeling depicted in Section 4.2.2.

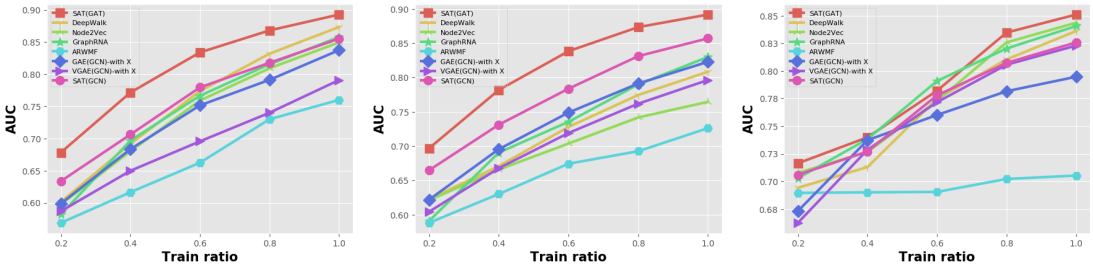
4.3.5.2 Less Observed Links

In link prediction task, the robustness of methods with less observed links is usually explored. We also conduct the link prediction experiment of link sparsity here. The results are shown in Figure 4–5.

From Figure 4–5, we can summarize that: (1) SAT(GAT) outperforms other competitive baselines even with less observed links. Considering the results of SAT(GCN) and SAT(GAT), we see that GAT is a strong graph convolution module and benefits SAT a lot. (2) Applying zero-filling trick for attribute-missing nodes sometimes augments the models (*e.g.* GAE), but it is not consistent on all datasets. (3) On Pubmed, DeepWalk and Node2Vec are the most two competitive baselines, they even beat SAT(GCN) in some sparse cases. Introducing the advantageous ranking loss from DeepWalk and Node2Vec to GNN might augment SAT(GCN) for link prediction task, but we do not explore it here since we mainly focus on the general method for learning on attribute-missing graphs.

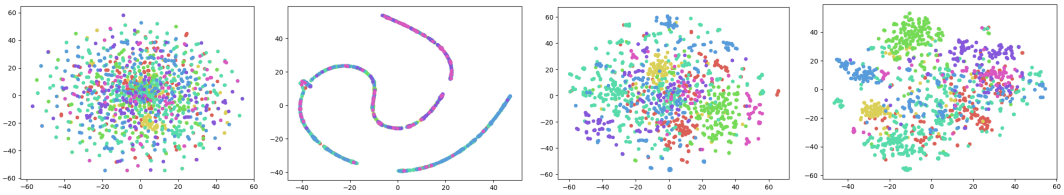
4.3.6 Analysis of Learned Node Representations

In *node attribute completion*, VAE, GCN, GraphRNA and SAT(GCN) infer node attributes by learning the latent embeddings of nodes. Good representation ability means that a method can learn representative embeddings where nearby nodes correspond to similar objects. Therefore, we experiment to visualize the learned node embeddings by t-SNE. Specifically, the latent embeddings for all test nodes are sampled. Then we use t-SNE to make dimension reduction and visualize them in 2-D space on Cora dataset. Nodes in the same class are expected to be clustered together. Note that for all methods, they do not use any label information in the training process.



(a) Cora (b) Citeseer (c) Pubmed

Figure 4–5 Link prediction with less observed links on three datasets. We use AUC here to evaluate the performance. Train ratio means the ratio of random samples from original train data. The term "with X" means attributes are taken into consideration and "no X" means only structures are considered.



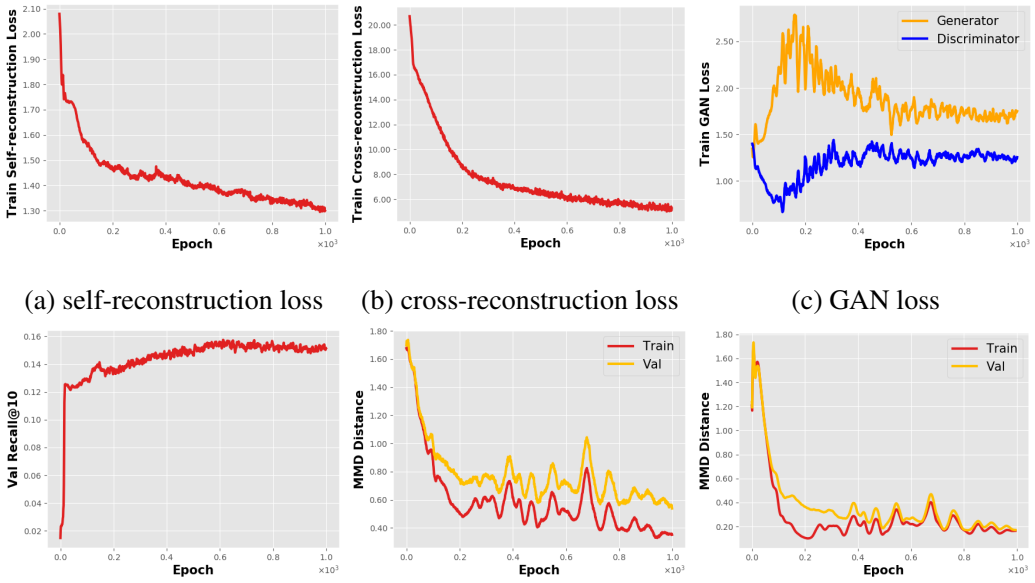
(a) VAE (b) GCN (c) GraphRNA (d) SAT(GCN)

Figure 4–6 The t-SNE visualization of test node embeddings on Cora. Each color represents one class. Note that all methods learn node embeddings without class supervision.

From Figure 4–6 (a) of VAE, we can see that nodes of different classes are mixed, which means VAE cannot distinguish the nodes belonging to different categories. For GCN in Figure 4–6 (b), the nodes are encoded into a narrow and stream-like space, where different nodes are mixed and overlapped. Compared to VAE, GCN has no prior assumption, which makes it lose distributed constraint and lead to the narrow and stream space. Compared to GraphRNA, SAT(GCN) shows more distinguished node representations and different nodes are clustered well. Although Gaussian prior is imposed on the latent space of both VAE and SAT, our SAT can capture the joint relationship between attributes and structures, and learn better node embeddings.

4.3.7 Learning Process Visualization

In order to better understand the learning process of SAT, we plot learning curves including the train reconstruction loss, train GAN loss, validation metric, and MMD



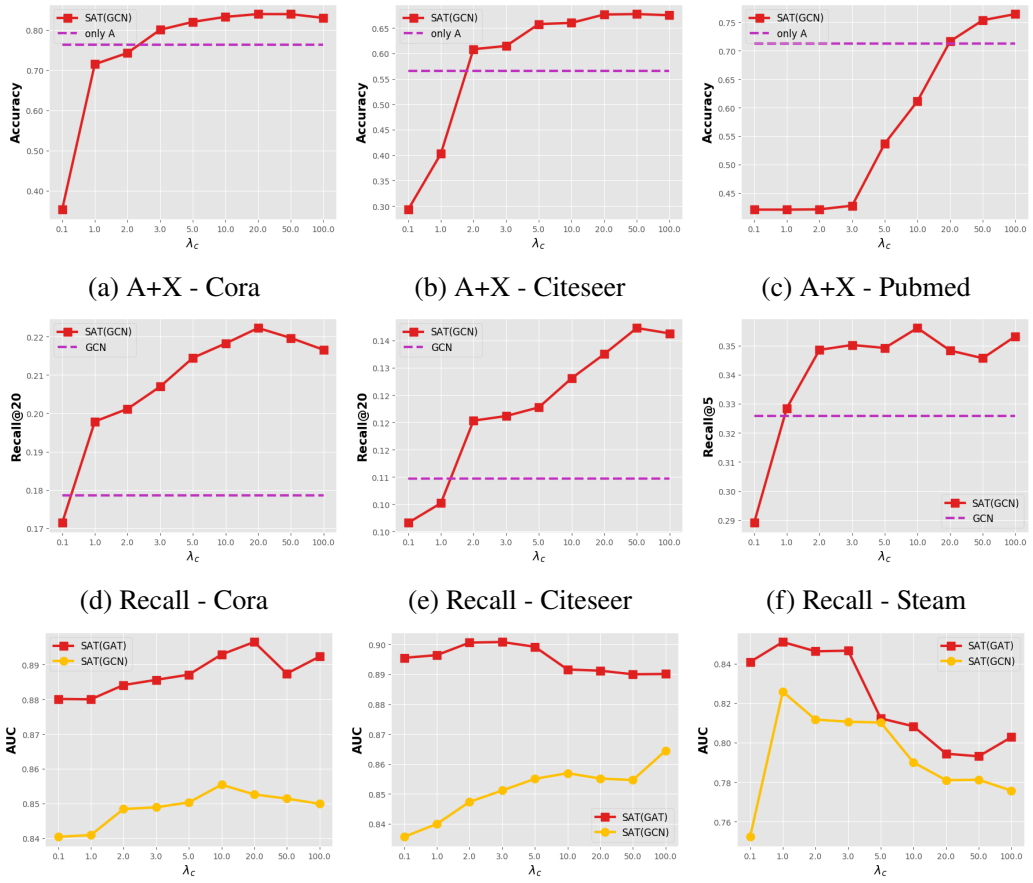
(a) self-reconstruction loss (b) cross-reconstruction loss (c) GAN loss

(d) Recall@10 (e) MMD distance (f) MMD distance

Figure 4–7 Visualization of the training process for SAT(GCN) on Cora. (a) The self-reconstruction loss. (b) The cross-reconstruction loss. (c) The GAN loss in adversarial distribution matching. (d) Validation Recall@10 along the training steps. (e) The train and validation MMD distance between the aggregated distribution $q(z)$ and Gaussian prior $p(z)$. (f) The train and validation MMD distance between distributions of z_x and z_a.

distance along the training epochs. The results are shown in Figure 4–7.

From Figure 4–7, we can summarize that that both the train reconstruction loss in (a)(b) and train GAN loss in (c) converge in the training process. And the validation Recall@10 in (d) increases step by step and finally converges at around $800th$ epoch. The MMD distance between learned aggregated distribution $q(z)$ and Gaussian prior $p(z)$ in (e) decreases step by step, which shows $q(z)$ matches the whole distribution of $p(z)$ successfully. Furthermore, since SAT involves distribution matching between latent codes z_x encoded from attributes and z_a encoded from structures, thus it is necessary to see whether our method matches them as what we expected. We show the train and validation MMD distance between z_x and z_a in Figure 4–7 (f). In this figure, the MMD distance between z_x and z_a is minimized gradually. This also keeps consistency with our *shared-latent space* assumption and further benefits the joint distribution modeling on attribute-missing graphs.



(a) A+X - Cora (b) A+X - Citeseer (c) A+X - Pubmed

(d) Recall - Cora (e) Recall - Citeseer (f) Recall - Steam

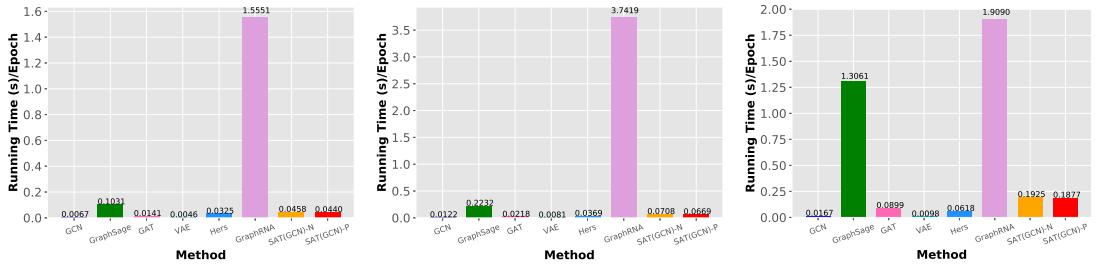
(g) AUC - Cora (h) AUC - Citeseer (i) AUC - Pubmed

Figure 4–8 The effects of λ_c on both the node classification and profiling task. (a-c) means the result for node classification with “A+X” setting on Cora, Citeseer and Pubmed. The dotted line with "only A" indicates that only the structural information is used, where GCN is the classifier. (d-f) indicates the result for profiling on Cora, Citeseer and Steam. The dotted line with "GCN" means we use the GCN as the attribute completion model. (g-i) shows the link prediction result with different λ_c.

4.3.8 Hyper-parameter λ_c

In SAT, we introduce λ_c to weight the cross-reconstruction stream in the objective function. It is desirable to see how our method responds to this hyper-parameter. Intuitively, we conduct an experiment about the *node attribute completion* and link prediction performance with different λ_c and the results are shown in Figure 4–8.

From Figure 4–8 (a-f), we can see that we need a large λ_c to restore high-quality node attributes. λ_c is vital for *node attribute completion* since we rely on the cross-reconstruction stream to restore node attributes. For link prediction in Figure 4–8 (g-i), it seems that link prediction performance is more robust with λ_c compared to the *node attribute completion* task. This is mainly because *node attribute completion* is a more difficult task since it generates high-dimensional data, which requires more fine-grained restoration.



(a) Cora (b) Citeseer (c) Steam

Figure 4–9 The empirical running time in each epoch of different methods. In this figure, SAT(GCN)-N indicates the non-parallel SAT(GCN), and SAT(GCN)-P means the parallel one.

4.3.9 Empirical Running Time Analysis

To investigate the time complexity, we conduct an experiment to compare the empirical running time of each epoch for different methods. Since SAT has parallel versions, we thus denote SAT-N as the non-parallel one, SAT-P as the parallel one. We conduct the experiments 10 times on the same machine with one Nvidia-TitanX GPU. The mean value of running time per epoch is reported in Figure 5–10.

From Figure 5–10, we can see that: (1) GraphRNA costs the most time because it involves the LSTM for feature encoding. Although SAT(GCN) involves a two-level distribution matching, it takes less time than GraphRNA. Taking the previous experimental results into consideration, we see that the proposed SAT generally achieves better

performance with less time complexity compared to GraphRNA. (2) SAT(GCN)-N and SAT(GCN)-P have similar time consumption. This indicates that the time cost of MLP encoding is trivial compared to the GNN encoding. In practice, we can use the non-parallel SAT instead of the parallel one.

4.4 Summary

In this Chapter, we explore the learning problems on attribute-missing graphs and make a *shared-latent space* assumption. Based on the assumption, we develop a novel distribution matching based GNN framework called structure-attribute transformer (SAT). SAT can not only handle the link prediction task but also the newly introduced *node attribute completion* task on graphs. Furthermore, for the *node attribute completion* task, we introduce practical measures including both node classification in the *node level* and profiling in the *attribute level* to evaluate the quality of restored node attributes. Empirical results validate the superiority of our method on both *node attribute completion* and link prediction task.

Learning on attribute-missing graphs is still an open problem and many topics could be studied from both the methodology and application aspect. For example, under our SAT framework, more efficient distribution matching methods for this problem could be investigated. There are some related real-world applications could be explored such as fraud detection in social networks, author description generation in co-author networks and image caption generation. Furthermore, more complex graph data such as heterogeneous attribute-missing graphs could also be an interesting problem. These will be studied in future works.

Chapter 5 Learning to Collaborate Different Bipartite Graphs

In recommendation, the user-item behaviors can be considered as a bipartite graph. When single user-item bipartite graphs have overlapped nodes, researchers propose to incorporate the bipartite graphs from other domains for boosted performance. It is also popularly mentioned as cross domain recommendation (CDR) [141]. According to real-world application conditions, CDR can be categorized into four different scenarios [142]: 1. User–No Item overlap (U-NI); 2. No User–Item overlap (NU-I); and 3. User–Item overlap (U-I). Among different scenarios, the U-NI scenario is a common case where items from different domains have no overlap and the users are shared. Recommendation in the U-NI scenario has been widely explored for real-world recommender system such as Netflix and Amazon [91, 143]. How to employ the correlations of user behaviors across domains and design the knowledge transfer scheme is an important topic in CDR. As we discussed in Chapter 5, the user preference encoded from the user behaviors includes both the overlapped and domain-specific features that are important for the recommendation task. To transfer knowledge, previous works usually resort to learn the overlapped features for feature alignment and compromise the domain-specific features or learn the domain-specific features based on heuristic human knowledge. How to better capture both features of user preferences is a key challenge in CDR.

To tackle this challenge, we make an equivalent transformation assumption that hypothesizes the user preference encoded from the behaviors in each domain can be mutually converted to each other with equivalent transformation. With this assumption, we further develop a novel distribution matching scheme to model the joint distribution of user behaviors across domain, yielding a variational *evidence lower bound* for optimizing the recommendation objective. By projecting the user preference from different domains into equivalently transformed space, the user representations could have the capacity of learning both features, which further facilitates better knowledge transfer. Extensive experiments on three real-world benchmarks confirm the superiority of the proposed model both quantitatively and qualitatively.

5.1 Introduction

In the U-NI scenario, researchers have studied various perspectives to transfer knowledge across domains and better predict user behaviors. For example, Li et al. [144] introduced a shared cluster-level rating model which defines a rating function for the latent user- and item-cluster variables to transfer knowledge. Others [82-83, 85] utilized different variants of matrix factorization (MF) approaches. However, most clustering and MF-based methods cannot capture the complex pattern in user-item interactions. Thus, some deep learning based methods [84, 89-91] emerged to improve the knowledge transfer and mine the complex patterns indicated by user-item interactions. For example, a deep cross connection network is designed in [91] to learn and transfer the shared interaction knowledge among domains. DARec [89] employs the domain adaptation technique [98] to learn domain-invariant user representations for CDR and it has achieved remarkable performance. Further, some works [42, 45-46] propose to model domain-specific features of user representations by employing a multi-layer perceptron (MLP) as the mapping function across domains.

Recent knowledge transfer works [145-147] indicate that modeling the joint distribution of different domain samples facilitates better knowledge transfer since joint distribution inherently captures the correlation of different domain samples. Similarly, modeling the joint distribution of user behaviors across domains is crucial in CDR, because the user behaviors in different domains are correlated together. Recent works [83, 89-90] based on the idea of shared-user representation attempt to model the above joint distribution. With the idea of shared-user representation, the CDR model resorts to learn the overlapped features for feature alignment, which usually leads to compromise the domain-specific features that help to better predict user behaviors [42-43]. In this context, the CDR model may not well do the feature alignment because of the user behavior prediction loss, as well as be hard to learn the domain-specific features for improved recommendation performance [44]. Several works [42, 45-46] have proposed to model the domain-specific features in addition to the overlapped features by employing MLP as the mapping function of user representations in each domain. However, the sparse data in user behaviors easily causes over-fitting for the mapping function [42]. EMCDR [42] proposes to choose users with dense behaviors to learn the MLP, while it requires heuristic human knowledge for choosing users. ATLRec [99] improves them with intuitively designed network architecture, but also requires heuristic ways for network design.

In this paper, we attempt to learn both the overlapped and domain-specific features for CDR in a more principled way. In particular, we assume that each user's preferences of different domains can be mutually converted to each other with equivalent transformation. Then, we propose an equivalent transformation learner (ETL) which models the joint distribution of user behaviors across domains. The equivalent transformation in ETL relaxes the idea of shared-user representation and enables a user's preferences across domains to have the capacity of preserving the domain-specific features as well as learning the overlapped features. Moreover, the proposed ETL has no requirement of carefully selecting training samples like [42, 45]. Figure 5–1 shows our idea of equivalent transformation based recommendation method. We show that when using ETL, the recommendation accuracy is largely improved compared to state-of-the-art methods on several public benchmarks. The contributions are summarized as follows:

- We highlight the importance of modeling the joint distribution of user behaviors across domains for CDR;
- We make an equivalent transformation assumption and further propose a novel method named ETL that models both the overlapped and domain-specific features in a joint distribution matching scheme. The proposed model works in a more principled way and does not require choosing training users or intuitively designing the networks;
- Extensive experiments on three public benchmarks demonstrate the effectiveness of the proposed ETL. Empirically, the results of designed experiments show that ETL has better ability to capture the overlapped and domain-specific features of user preferences in CDR.

5.2 ETL: Equivalent Transformation Learner

In this section, we first give the problem definition. Then, details about the proposed method is introduced. The model architecture of the proposed method is shown in Figure 5–2.

5.2.1 Problem Definition

In this Chapter, we focus on the U-NI scenario [142] in CDR. In this setting, different domains $\mathcal{X}$ and $\mathcal{Y}$ have the same set of users $\mathcal{U} = \{U_1, U_2, ..., U_N\}$ where N denotes the number of users. The item set of domain $\mathcal{X}$ and $\mathcal{Y}$ respectively is $I^{\mathcal{X}} = \{I_1^{\mathcal{X}}, I_2^{\mathcal{X}}, ..., I_M^{\mathcal{X}}\}$

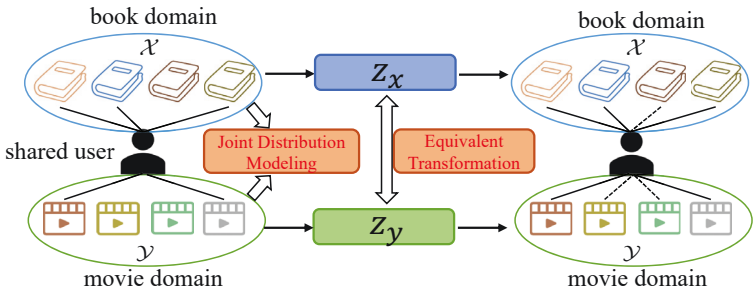


Figure 5–1 Our equivalent transformation based model for recommendation in the U-NI scenario. Solid lines mean observed user-item interactions and dashed lines are the interactions we aim to predict with probabilities. Given a user, z_x and z_y are the preference representations encoded the user-item interactions in domain $\mathcal{X}$ and domain $\mathcal{Y}$, respectively. The proposed ETL has an equivalent transformation between z_x and z_y, and models the joint distribution of the user behaviors across domains.

and $I^{\mathcal{Y}} = \{I_1^{\mathcal{Y}}, I_2^{\mathcal{Y}}, ..., I_T^{\mathcal{Y}}\}$, where M and T indicate the number of items in domain $\mathcal{X}$ and $\mathcal{Y}$, respectively. The user-item interactions of domain $\mathcal{X}$ could be represented by a matrix $R^{\mathcal{X}} \in \mathbb{R}^{N \times M}$ where the values are explicit feedback, such as ratings, or implicit feedback, such as clicks. Similarly, the user-item interactions of domain $\mathcal{Y}$ is indicated by $R^{\mathcal{Y}} \in \mathbb{R}^{N \times T}$. Usually, $R^{\mathcal{X}}$ and $R^{\mathcal{Y}}$ are very sparse since a user only interacts with a small subset of items in each domain. The goal of CDR is to improve the recommendation accuracy for users in domain $\mathcal{X}$ and $\mathcal{Y}$. Unlike [89], we do not distinguish a source domain or a target domain since the recommendation task for each domain is performed in an unified method here.

5.2.2 Joint Distribution Modeling

Modeling the joint distribution of user behaviors across domains is essential since the behaviors exhibit correlations in CDR. By modeling the joint distribution, we learn more representative user preferences which can help to predict the missing interactions. The proposed ETL is based on modeling the above joint distribution. Thus, we start with the joint distribution to introduce ETL. We here model the joint distribution via maximizing the joint log-likelihood of the observations.

Assume x_i and y_i are the behaviors for user U_i in domain $\mathcal{X}$ and $\mathcal{Y}$ respectively. In other words, x_i and y_i are the row vectors of $R^{\mathcal{X}}$ and $R^{\mathcal{Y}}$. Let (x_i, y_i) be one paired sample for user U_i. The joint log-likelihood of observations is composed of a sum over the likelihoods of individual data points $\sum_{i=1}^{N} \log p_\theta(x_i, y_i)$, where $p_\theta(x_i, y_i)$ denotes the

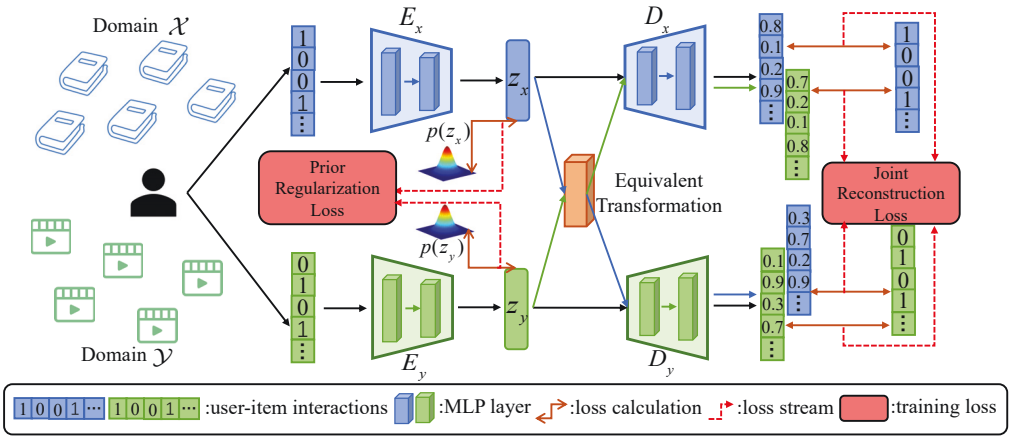


Figure 5–2 The architecture of ETL. ETL encodes user behaviors in two domains with different encoders and then decodes the latent codes to user behaviors in each domain. The joint reconstruction loss and a prior regularization loss facilitates knowledge transfer between two domains and benefits the user behavior prediction.

probability density function.

$$\log p_\theta((x_1, y_1), (x_2, y_2), ..., (x_N, y_N)) = \sum_{i=1}^{N} \log p_\theta(x_i, y_i) \tag{5–1}$$

Borrowing the idea of maximizing the marginal log-likelihood in VAE [100], if z_x and z_y are the latent factors of x_i and y_i respectively, $\log p_\theta(x_i, y_i)$ can be formulated as:

$$\begin{aligned} \log p_\theta(x_i, y_i) =& D_{KL}[q_\phi(z_x, z_y|x_i, y_i)||p(z_x, z_y|x_i, y_i)] \\ &+ \mathcal{L}(\theta, \phi; x_i, y_i) \end{aligned} \tag{5–2}$$

where the first term is the KL divergence of the approximate posterior $q_\phi(z_x, z_y|x_i, y_i)$ from the true posterior $p(z_x, z_y|x_i, y_i)$. Since this KL term is non-negative, the second term is the *ELBO* on the log-likelihood $\log p_\theta(x_i, y_i)$. Following the derivation in VAE [100], $\mathcal{L}(\theta, \phi; x_i, y_i)$ can be written as:

$$\begin{aligned} \mathcal{L}(\theta, \phi; x_i, y_i) =& \mathbb{E}_{q_\phi(z_x, z_y|x_i, y_i)}[\log p_\theta(x_i, y_i|z_x, z_y)] \\ &- D_{KL}[q_\phi(z_x, z_y|x_i, y_i)||p(z_x, z_y)] \end{aligned} \tag{5–3}$$

where $p_\theta(x_i, y_i|z_x, z_y)$ denotes the conditional distribution parameterized by θ. The first term in Eq. 5–3 indicates the joint reconstruction loss where z_x, z_y encoded from x_i, y_i are used to reconstruct x_i, y_i. The second term in Eq. 5–3 indicates the prior regularization loss where $q_\phi(z_x, z_y|x_i, y_i)$ is expected to match the prior distribution $p(z_x, z_y)$. ETL

implements the the joint reconstruction loss and the prior regularization loss via a dual auto-encoder structure and an adversarial learning scheme. The architecture of ETL is shown in Figure 5–2. In the following sections, we provide details on the two losses, followed by the objective function and implementation.

5.2.3 Joint Reconstruction Loss

The first term $\mathbb{E}_{q_\phi(z_x,z_y|x_i,y_i)}[\log p_\theta(x_i,y_i|z_x,z_y)]$ in Eq. 5–3 consists of an approximate posterior $q_\phi(z_x,z_y|x_i,y_i)$ parameterized by ϕ and a conditional distribution $p_\theta(x_i,y_i|z_x,z_y)$ parameterized by θ. To ease the solution of the posterior $q_\phi(z_x,z_y|x_i,y_i)$, we employ the mean filed theory [148] to define the approximate posterior by following recent works [131, 149-150]:

$$q_\phi(z_x,z_y|x_i,y_i) \overset{\text{def}}{=\!=} q_{\phi_x}(z_x|x_i)q_{\phi_y}(z_y|y_i) \tag{5–4}$$

which means the latent codes z_z and z_y are encoded from x_i and y_i, respectively. In order to further solve the conditional distribution $p_\theta(x_i,y_i|z_x,z_y)$, we make an equivalent transformation (ET) assumption.

Equivalent Transformation Assumption: *In CDR, each user's preferences of different domains are correlated and can be mutually converted to each other with equivalent transformation.*

According to the definition in mathematics [151], the equivalent transformation between z_x and z_y is defined as $z_x = Q^{-1}z_yP$, where Q, P are two invertible matrices. Note that we set Q as an identity matrix I here for simplicity. Thus, if we denote $z_{y\to x}$ as the equivalently transformed z_y and $z_{x\to y}$ as the equivalently transformed z_x, we have $z_{y\to x} = z_yW_x$, $z_{x\to y} = z_xW_y$ and $W_x = P, W_xW_y = I$. Further, according to the graphical model of ETL in Figure 5–3, we have the following proposition:

Proposition 5.1. ①*: Given latent variables z_x, z_y, the observations x_i and y_i are conditional independent.* ②*: Given the latent variable z_x, the observation x_i and the latent variable z_y are conditional independent.* ③*: Given the latent variable z_y, the observation y_i and the latent variable z_x are conditional independent.*

By this proposition and the ET assumption, we rewrite the conditional distribution

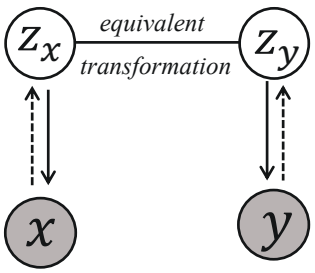


Figure 5–3 The graphical model of our the equivalent transformation based model. In this figure, z_x and z_y are the latent variables of user representations encoded the behaviors x of domain $\mathcal{X}$ and y of domain $\mathcal{Y}$, respectively. Solid arrows denote the generative process and dashed arrows denote the inference process.

as:

$$
\begin{aligned}
p_\theta(x_i, y_i|z_x, z_y) &\overset{①}{=} p_{\theta_x}(x_i|z_x, z_y)p_{\theta_y}(y_i|z_x, z_y) \\
&\overset{②\text{ and }③}{=\!=\!=} p_{\theta_x}(x_i|z_x)p_{\theta_y}(y_i|z_y) \\
&= \sqrt{p_{\theta_x}(x_i|z_x)p_{\theta_x}(x_i|z_x)}\sqrt{p_{\theta_y}(y_i|z_y)p_{\theta_y}(y_i|z_y)} \\
&\overset{\text{ET assumption}}{=\!=\!=\!=} \sqrt{p_{\theta_x}(x_i|z_x)p_{\theta_x}(x_i|z_yW_x)}\sqrt{p_{\theta_y}(y_i|z_y)p_{\theta_y}(y_i|z_xW_y)}
\end{aligned} \tag{5–5}
$$

where θ_x is the parameter of a decoder D_x and θ_y is the parameter of a decoder D_y. Eq. 5–5 indicates a dual auto-encoder structure where z_x and z_y are used to reconstruct x_i and y_i, respectively. Meanwhile, with the equivalent transformation, z_x and z_y are able to generate y_i and x_i. $p_{\theta_x}(x_i|z_yW_x)$ and $p_{\theta_y}(y_i|z_xW_y)$ indicate the cross domain generation that can help to learn the equivalent transformation. In the end, the equivalent transformation allows z_x and z_y to have the capacity of maintaining the domain-specific features as well as learning the overlapped features by this dual auto-encoder structure.

Specification of Equivalent Transformation: Different transformations may have different impacts on knowledge transfer, and they are discussed with experiments in Section 5.3.6.1. In ETL, inspired by [86], we consider that the equivalent transformation in CDR should avoid false correlations between users. Thus, the orthogonal transformation is employed here since it preserves the inner product of vectors, namely it keeps the user similarities across domains. According to the definition of orthogonal transformation, W_x and W_y satisfies $W_x = W$ and $W_y = W^T$, where $W \in \mathbb{R}^{d\times d}$ is the trainable orthogonal mapping matrix. Taking the above into summary, we rewrite

$\mathbb{E}_{q_\phi(z_x,z_y|x_i,y_i)}[\log p_\theta(x_i,y_i|z_x,z_y)]$ in Eq. 5–3 as:

$$\begin{aligned}\mathbb{E}_{q_\phi(z_x,z_y|x_i,y_i)}[\log p_\theta(x_i,y_i|z_x,z_y)] = \frac{1}{2}\big\{&\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_x}(x_i|z_x)]\\ &+\mathbb{E}_{q_{\phi_y}(z_y|y_i)}[\log p_{\theta_x}(x_i|z_yW)]\\ &+\mathbb{E}_{q_{\phi_y}(z_y|y_i)}[\log p_{\theta_y}(y_i|z_y)]\\ &+\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_y}(y_i|z_xW^T)]\big\}\end{aligned} \tag{5–6}$$

The optima of Eq. 5–6 is the same as that of Eq. 5–6 multiplied by a constant. Thereby, for simplified expression, we write the joint reconstruction loss as:

$$\begin{aligned}\min_{\phi_x,\phi_y,\theta_x,\theta_y,W}\mathcal{L}_{JRL} = &-\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_x}(x_i|z_x)]\\ &-\mathbb{E}_{q_{\phi_y}(z_y|y_i)}[\log p_{\theta_x}(x_i|z_yW)]\\ &-\mathbb{E}_{q_{\phi_y}(z_y|y_i)}[\log p_{\theta_y}(y_i|z_y)]\\ &-\mathbb{E}_{q_{\phi_x}(z_x|x_i)}[\log p_{\theta_y}(y_i|z_xW^T)]\\ &+\lambda(||z_x - z_xW^TW||_F^1 + ||z_y - z_yWW^T||_F^1)\end{aligned} \tag{5–7}$$

where the last term indicates the regularization loss for equivalent transformation when the transformation is specified as orthogonal mapping. λ is the hyper-parameter to weight the importance of the regularization. $\{\phi_x,\phi_y\}$ and $\{\theta_x,\theta_y\}$ are the parameters of encoders and decoders, respectively.

5.2.4 Prior Regularization Loss

The second term $D_{KL}[q_\phi(z_x,z_y|x_i,y_i)||p(z_x,z_y)]$ involves joint prior $p(z_x,z_y)$ which indicates a complex prior for z_x,z_y. In this Chapter, we set $p(z_x,z_y) = p(z_x)p(z_y)$ for simplicity. Taking Eq. 5–4 into consideration, the prior regularization term is:

$$\begin{aligned}D_{KL}[q_\phi(z_x,z_y|x_i,y_i)||p(z_x,z_y)] = &D_{KL}[q_{\phi_x}(z_x|x_i)||p(z_x)]\\ &+D_{KL}[q_{\phi_y}(z_y|y_i)||p(z_y)]\end{aligned} \tag{5–8}$$

where $p(z_x)$ and $p(z_y)$ are the prior distributions. Eq. 5–8 states the regularization that matches $q_{\phi_x}(z_x|x_i)$ to prior $p(z_x)$ and matches $q_{\phi_y}(z_y|y_i)$ to prior $p(z_y)$.

Since it is not easy to derive explicit formulations for some complex priors in KL divergence, ETL employs the adversarial distribution matching that can impose an arbitrary prior distribution for the latent codes without hard derivation [52]. Inspired

by [52, 109-111], we propose to use adversarial learning to perform the distribution matching between $q_{\phi_x}(z_x|x_i)$ (*resp.* $q_{\phi_y}(z_y|y_i)$) and $p(z_x)$ (*resp.* $p(z_y)$). Following [52], the prior regularization loss can be formulated with the following adversarial learning objective:

$$\begin{aligned}\min_{\psi}\max_{\phi}\mathcal{L}_{PRL} = &-\mathbb{E}_{z_x\sim p(z_x)}[\log \mathcal{D}_x(z_x)] \\ &-\mathbb{E}_{z_x\sim q_{\phi x}(z_x|x_i)}[\log(1-\mathcal{D}_x(z_x))] \\ &-\mathbb{E}_{z_y\sim p(z_y)}[\log \mathcal{D}_y(z_y)] \\ &-\mathbb{E}_{z_y\sim q_{\phi y}(z_y|y_i)}[\log(1-\mathcal{D}_y(z_y))] \end{aligned} \tag{5–9}$$

where $\phi = \{\phi_x, \phi_y\}$ shares the same definition in Eq. 5–7 and $\psi = \{\psi_x, \psi_y\}$ are the parameters of the discriminators $\mathcal{D}_x, \mathcal{D}_y$. $p(z_x)$ and $p(z_y)$ are the prior distributions of z_x and z_y respectively.

The employed adversarial distribution matching in Eq. 5–9 has several advantages compared to the KL divergence in Eq. 5–8. The KL divergence tries to match $q_{\phi_x}(z_x|x_i)$ to prior $p(z)$, which will have risk to lose the information from input x_i. By contrast, the adversarial distribution matching in latent space makes the posterior $q_{\phi_x}(z_x|x_i)$ to be the aggregated posterior $q_{\phi_x}(z_x)$, which encourages z_x to match the whole distribution of $p(z_x)$ [51-52]. Meanwhile, the mode collapse problem in adversarial learning could be avoided since ETL involves the reconstruction loss which encourages the latent embeddings to match both the prior and the entire true data distribution [110].

5.2.5 Objective Function and Implementation

Taking Eq. 5–7 and Eq. 5–9 into account, maximizing the *ELBO* in Eq. 5–3 can be performed via optimizing the following objective function:

$$\min_{\Theta}\max_{\Phi}\mathcal{L}_{ETL} = \mathcal{L}_{JRL} + \eta\mathcal{L}_{PRL} \tag{5–10}$$

where $\Theta = \{\phi_x, \phi_y, \theta_x, \theta_y, \psi_x, \psi_y, W\}$ and $\Phi = \{\phi_x, \phi_y\}$ are the network parameters. η is the hyper-parameter to weight the importance of the prior regularization term.

The architecture of ETL is shown in Fig. 5–2, where E_x is the encoder for $q_{\phi_x}(z_x|x_i)$ and E_y is the encoder for $q_{\phi_y}(z_y|y_i)$. Similarly, D_x and D_y are the decoders for $p_{\theta_x}(x_i|\cdot)$ and $p_{\theta_y}(y_i|\cdot)$, respectively. The discriminators $\mathcal{D}_x$ and $\mathcal{D}_y$ are designed for adversarial learning. In our implementation, all $E_x, E_y, D_x, D_y, \mathcal{D}_x, \mathcal{D}_y$ are two-layer MLP with

Algorithm 5–1 Equivalent Transformation Learner (ETL)

Input: User behavior data R^X and R^Y from different domains, the hyper-parameters of the model such as λ and η, the hyper-parameters for learning such as the number of epochs nb_epochs, batch size B and learning rate lr.

Output: Predict the scores of user-item interactions and rank them for recommendation.

for epoch in range(nb_epochs) **do**
    # sample mini-batch data
    Shuffle and split the training data to have the dataloader.
    for $(i, (x_i, y_i))$ in enumerate(dataloader) **do**
        # encoding for mini-batch data
        Encode x_i, y_i into latent codes z_x, z_y by E_x, E_y.
        # joint reconstruction loss
        Calculate the joint reconstruction loss according Eq. 5–7.
        # prior regularization loss
        Calculate the prior regularization loss according Eq. 5–9.
        Calculate the loss in Eq. 5–10 and update model parameters.
    end for
end for

Relu as the non-linear activation function. We use standard Gaussian distributions such that $p(z_x) \sim \mathcal{N}(0, 1)$ and $p(z_y) \sim \mathcal{N}(0, 1)$, which is a common scenario in recent adversarial learning based methods [52, 101, 110]. It is also worthwhile to point out that although both $p(z_x)$ and $p(z_y)$ follow the standard Gaussian distribution, it does not mean z_x and z_y are in the same latent space and it will not break our motivation of modeling both features of user preferences in CDR. Moreover, the reconstruction loss between the predictions and true data could be *MSE* if the user-item interactions are explicit feedback and *binary cross entropy* if the user-item interactions are implicit feedback. A concise description of ETL is provided in Algorithm 5–1.

5.2.6 Discussion of Equivalent Transformation

In this section, we have some discussion about the equivalent transformation to provide deep insight on how ETL connects to and differs from previous methods. Although recent methods [83-84, 89-90, 92] with the idea of shared-user representation do

not mention this, they attempt to model the joint distribution of user behaviors across domains. According to the coupling theory [127], if we want to model a joint distribution of two marginal observations $\mathcal{X}, \mathcal{Y}$, an assumption is indispensable to describe the relationship of the corresponding latent factors z_x and z_y. The assumption can be expressed as $z_x = z_y W_x, z_y = z_x W_y$ (or as non-linear formulation). There are two possible formulations which are *unconstrained* and *constrained*. The *unconstrained* formulation means there is no constraint between W_x and W_y, and it is similar to the idea of using MLP as the mapping function across domains [42, 46]. While the *constrained* one imposes constraints on W_x and W_y.

In CDR, z_x and z_y in different domains have correlations. Thus, in ETL, we adopt the *constrained* formulation which specifically is our ET assumption. This assumption involves the equivalent transformation with $W_x W_y = I$, which degenerates to the idea of shared-user representation in [83, 89-90] when $W_x = W_y = I$. However, $W_x = W_y = I$ would make the domain-specific features be suppressed. In contrary, ETL with ET assumption allows a user's preferences in different domains to have the capacity of learning the domain-specific features as well as the overlapped features. It is worthwhile to point out that the ET assumption ensures cross domain generation that facilitates better knowledge transfer. Different variants of the *unconstrained* and *constrained* formulations are discussed with experiments in Section 5.3.6.1.

It is also worthwhile to mention that not all data points will follow this equivalent transformation assumption. A simple way to judge whether the assumption holds for two domains is whether these domains have obvious overlapped and domain-specific attributes. If they have, then the assumption holds, otherwise, it does not. For example, Movie and Book domain have obvious overlapped attributes such as theme (*e.g.* love, war) and obvious domain-specific attributes such as "director" in the movie domain and "writing style" in the book domain.

5.2.7 Time Complexity Analysis

Stochastic training of DNN methods involves two steps, the forward and backward computations. ETL supports the mini-batch training and the time cost lies in the joint reconstruction term and the prior regularization term. We thus decompose the time complexity of ETL into two parts, namely the time complexity of the joint reconstruction term and the prior regularization term. In each batch of ETL, the joint reconstruction

term encodes user behaviors into latent codes and then decodes the latent codes into user behaviors. If we denote B as the batch size, M and T as the number of items in each domain, then the complexity of the joint reconstruction term is $O(B(M+T))$. The prior regularization term imposes prior distribution on the latent codes. If we denote the latent dimension as d, the complexity of the prior regularization term is $O(Bd)$. In summary, the time complexity of ETL is $O(B(M+T)+Bd)$. In addition, user behaviors in each domain usually are extremely sparse vectors. This indicates most values in the $M-$dimensional and $T-$dimensional user behavior vectors are zeros. Then the sparse vectors can be fast calculated by the sparse matrix multiplication in Pytorch or Tensorflow. If we denote the average number of non-zero values of the $M-$dimensional and $T-$dimensional user behavior vectors in a batch as c, we have $c \ll \frac{1}{2}(M+T)$. And the complexity of the joint reconstruction term can be largely reduced to $O(2Bc)$. In this case, the complexity of ETL is $O(2Bc + Bd)$, which ensures that ETL can work on much larger datasets.

Moreover, when applying ETL on multiple domains, a two combination of each two domains can be used, which is the same for other CDR methods (*e.g.* DARec). If we denote the domain number as N_d, the complexity of ETL is $O(\frac{N_d(N_d-1)}{2}(B(M+T)+Bd))$. Since two domains are the most common setting in CDR and this paper mainly focuses on the idea of modeling both the overlapped and domain-specific features, we thus do not explore more in the multiple-domain case by following recent works [89, 91, 133].

5.3 Experiments and Analysis

In this section, we first give the details about the datasets and experimental settings. Then we systemically evaluate ETL via the comparison with recent state-of-the-art methods on multiple public benchmarks. Next, we design two experiments to demonstrate that ETL simultaneously learns the overlapped and domain-specific features in CDR. Finally, we conduct the ablation study on ETL.

5.3.1 Dataset Description

To evaluate the effectiveness of ETL, we utilize three largest benchmarks from Amazon[1]. The three datasets are Movies and TV (**Movie**), Books (**Book**), CDs and Vinyl (**Music**). Note that these three dataset are benchmarks for corss domain recommendation and have been used in recent works [89, 91]. Following [89, 91], we make a two

[1] http://jmcauley.ucsd.edu/data/amazon/

Table 5–1 The statistics of datasets.

Datasets	Movie & Book		Movie & Music		Music & Book	
#Users	29,476		15,914		16,267	
Domain	Movie	Book	Movie	Music	Music	Book
#Items	24,091	41,884	17,794	20,058	18,467	23,988
#Interactions	591,258	579,131	416,228	280,398	233,251	291,325
Density	0.08%	0.05%	0.14%	0.09%	0.08%	0.07%

combinations amongst the three datasets and find the shared users in each of the two domains for the U-NI CDR scenario [142]. Next, we obtain **Movie & Book**, **Movie & Music** and **Music & Book** as our experimental datasets. Compared to the explicit feedback (*e.g.* the user ratings on items), the implicit feedback (*e.g.* the user clicks or does not click an item) are more common in real-world recommender systems [152-153], we thus focus on the implicit user-item interactions in this paper. In other words, the user-item interaction matrices $R^{\mathcal{X}}, R^{\mathcal{Y}}$ are binary matrices where the value is 1 (observed or clicked) if the user interacted with the item and 0 (unobserved or not clicked) otherwise. Since the user-item interactions in these benchmarks are ratings ranging from 0 to 5, we convert the ratings of 3,4,5 as positive samples by following [91]. Finally, we filter users and items whose number of interactions is less than 5. The dataset statistics are shown in Table 5–1. As shown in Table 5–1, both domains in each dataset are extremely sparse with at least 99.86% interactions are unobserved. It presents a great challenge on most clustering-based and MF-based CDR methods since these methods normally require dense interactions in at least one domain [89]. In addition, the number of items in **Movie & Book** is unbalanced and the density in **Movie & Music** is imbalanced, which provides more comprehensive evaluation conditions for different CDR methods.

5.3.2 Baselines

To illustrate the effectiveness, we compare ETL with single domain methods (PMF, CDAE, CFVAE and AAE) and recent cross domain methods (CMF, AAE++, CoNet, sCoNet, ATLRec, DDTCDR and DARec) as follows:

- PMF [154]: Probabilistic matrix factorization is a classic factorization-based

method for single domain recommendation, which has been successfully applied in real systems [155]. Since we focus on the implicit feedback here, we replace the original mean square error (MSE) loss in [155] with binary cross entropy loss for fair comparison.

- CDAE [156]: Collaborative denoising auto-encoder is a generalization of several auto-encoder based recommendation methods but with more flexible components.
- CFVAE [157]: Collaborative variational auto-encoder is a variational auto-encoder model for collaborative filtering.
- AAE [52]: Adversarial auto-encoder combines the recent generative adversarial networks (GAN) and the auto-encoding variational inference. We follow CDAE's setting here for AAE to perform the recommendation task.
- CMF [83]: Collective matrix factorization is a multi-relational learning method that jointly factorizes the user-item interaction matrices of different domains.
- AAE++ [52]: We extend AAE as AAE++ here for CDR. To be specific, AAE++ performs the adversarial auto-encoder for different domains with the same standard Gaussian distribution as prior and different discriminators. It also serves as a variant of our ETL model with no cross generation stream.
- CoNet and sCoNet [91]: CoNet transfers knowledge of different domains through a modified cross-stitch neural network. Specifically, it constructs deep cross-connections for the predicted user-item interactions from different domains. sCoNet is a sparse version of CoNet with L1-regularization on the user and item representations.
- ATLRec [99]: ATLRec is an adversarial transfer learning based model that captures domain-shareable and domain-specific features by different neural networks. In particular, it use two domain-specific neural networks to learn the domain-specific features and one shared neural network to capture the domain-shareable features.
- DDTCDR [97]: DDTCDR introduces the mechanism of dual learning in CDR and proposes a deep dual transfer network. DDTCDR requires user features and item features with the same dimension as input, and these features are obtained by employing the matrix factorization technique on user-item interaction matrix.
- DARec [89]: DARec introduces domain adaptation techniques [158] for CDR and has achieved remarkable recommendation performance.

- ETL-JRL: ETL-JRL is a variant of our ETL model, which only contains the joint reconstruction loss $\mathcal{L}_{JRL}$.

5.3.3 Experimental Setups

Evaluation Protocols: In item recommendation, the leave-one-out (LOO) evaluation is widely used [91, 159-162] and we also use LOO here. In other words, we randomly reserve two items for each user, one as the validation item and the other one as the test item. Following [91, 159], we randomly sample 99 items that are not interacted by the user as negative items, and then evaluate how the recommender can rank the validation and test item against the negative items. Since we focus on the implicit feedback in recommendation, we adopt three widely used evaluation metrics: hit ratio (HR), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR). The predicted rank list is cut off at $topK = 5, 10$. A higher value means a better recommendation performance for all three metrics. Also, during training, we save the best trained model according to the performance on the validation set and perform testing with the saved model. Moreover, all models are run 5 times and the mean value on the test are reported as the model performance. Empirically, we also show the t-test results to illustrate that ETL has statistically significant performance over other methods.

Parameter Settings: We implement our ETL with Pytorch on a machine with one 1080Ti GPU. The embedding size is fixed to 200 for all methods. We optimize all models with Adam optimizer [163] and the batch size is set as 256. The default Xavier initializer [164] is used to initialize all model parameters. For all methods, the dropout ratio is set as 0.5 and the learning rate is 0.001. The number of training epochs is set to 300 which could ensure the convergence for all models. In ETL, we do not tune hyper-parameter η and fix it as 1.0 on the three benchmarks for simplicity. For hyper-parameter λ, we tune it among [0.1,0.5,1.0,2.0,5.0,10.0] according to the performance on the validation set. Then we obtain $\lambda = 5.0$ for Movie & Book, $\lambda = 0.5$ for Movie & Music and $\lambda = 1.0$ for Music & Book. The codes of PMF, CDAE, CFVAE, AAE and CMF are easily obtained online. For CoNet, sCoNet, ATLRec and DDTCDR, we directly use the codes provided by the authors from emails and keep the default settings. Since we do not acquire the codes of DARec from the authors, we implemented them with Pytorch according to details in [89].

Table 5–2 The overall comparison on Movie & Book. The underlined results are the best performance of baselines.

Movie & Book												
topK	topK=5						topK=10					
Domain	Movie			Book			Movie			Book		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
PMF	0.4364	0.3147	0.2745	0.4003	0.2961	0.2621	0.5737	0.3591	0.2928	0.5121	0.3327	0.2772
CDAE	0.4660	0.3471	0.3056	0.4483	0.3492	0.3157	0.5991	0.3901	0.3263	0.5640	0.3851	0.3315
CFVAE	0.4587	0.3396	0.3006	0.4277	0.3258	0.2918	0.5928	0.3852	0.3206	0.5508	0.3646	0.3091
AAE	0.4661	0.3471	0.3080	0.4457	0.3509	0.3128	0.5989	0.3900	0.3269	0.5559	0.3871	0.3291
CMF	0.4433	0.3224	0.2815	0.4373	0.3225	0.2848	0.5848	0.3674	0.3000	0.5583	0.3616	0.3009
AAE++	0.4803	0.3590	0.3189	0.4537	0.3592	0.3280	0.6098	0.4009	0.3362	0.5656	0.3954	0.3429
CoNet	0.3886	0.2702	0.2279	0.3451	0.2316	0.2033	0.5244	0.3145	0.2464	0.4690	0.2716	0.2195
sCoNet	0.3914	0.2709	0.2277	0.3408	0.2322	0.2043	0.5308	0.3167	0.2463	0.4711	0.2724	0.2209
ATLRec	0.3851	0.2836	0.2435	0.3452	0.2587	0.2020	0.5349	0.3354	0.2607	0.4763	0.3131	0.2362
DDTCDR	0.4090	0.2942	0.2576	0.4008	0.3153	0.2893	0.5394	0.3382	0.2732	0.5073	0.3492	0.3013
DARec	0.4914	0.3641	0.3224	0.4690	0.3591	0.3227	0.6202	0.4069	0.3401	0.5919	0.3989	0.3392
ETL-JRL	0.5109	0.3805	0.3427	0.5020	0.3940	0.3663	0.6412	0.4157	0.3600	0.6266	0.4221	0.3819
ETL	**0.5115**	**0.3812**	**0.3431**	**0.5111**	**0.3989**	**0.3705**	**0.6419**	**0.4244**	**0.3608**	**0.6329**	**0.4383**	**0.3861**
%Improv.	4.09%	4.69%	6.42%	8.97%	11.05%	12.95%	3.49%	4.30%	6.08%	6.92%	9.87%	12.59%

Table 5–3 The overall comparison on Movie & Music. The underlined results are the best performance of baselines.

Movie & Music												
topK	topK=5						topK=10					
Domain	Movie			Music			Movie			Music		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
PMF	0.4081	0.2872	0.2474	0.4505	0.3350	0.2969	0.5490	0.3326	0.2261	0.5769	0.3759	0.3137
CDAE	0.4191	0.3093	0.2723	0.4433	0.3396	0.3053	0.5544	0.3528	0.2898	0.5662	0.3792	0.3225
CFVAE	0.4318	0.3110	0.2750	0.4362	0.3281	0.2884	0.5699	0.3605	0.2945	0.5646	0.3663	0.3082
AAE	0.4357	0.3226	0.2860	0.4557	0.3445	0.3086	0.5689	0.3658	0.3023	0.5772	0.3863	0.3248
CMF	0.4309	0.3025	0.2603	0.4794	0.3568	0.3166	0.5736	0.3487	0.2793	0.6124	0.4011	0.3349
AAE++	0.4281	0.3142	0.2754	0.4538	0.3501	0.3142	0.5628	0.3564	0.2928	0.5789	0.3887	0.3301
CoNet	0.3729	0.2556	0.2176	0.3887	0.2658	0.2183	0.5146	0.3013	0.2369	0.5380	0.3140	0.2229
sCoNet	0.3773	0.2594	0.2197	0.3953	0.2670	0.2080	0.5209	0.3060	0.2390	0.5411	0.3148	0.2283
ATLRec	0.3771	0.2758	0.2440	0.3820	0.2782	0.2435	0.5189	0.3215	0.2627	0.5198	0.3225	0.2514
DDTCDR	0.3880	0.2748	0.2366	0.4204	0.3169	0.2804	0.5220	0.3177	0.2542	0.5421	0.3563	0.2962
DARec	0.4589	0.3349	0.2950	0.4822	0.3636	0.3241	0.5973	0.3790	0.3134	0.6125	0.4051	0.3413
ETL-JRL	0.4869	0.3629	0.3210	0.5260	0.4027	0.3631	0.6222	0.4057	0.3387	0.6548	0.4422	0.3766
ETL	**0.4891**	**0.3632**	**0.3224**	**0.5314**	**0.4037**	**0.3653**	**0.6241**	**0.4076**	**0.3404**	**0.6550**	**0.4442**	**0.3819**
%Improv.	6.58%	8.45%	9.28%	10.20%	11.02%	12.71%	4.48%	7.54%	8.61%	6.93%	9.65%	11.89%

Table 5–4 The overall performance on Music & Book. The underlined results are the best performance of baselines. Compared to ETL, the t-test results of other baselines are shown in this table.

Music & Book												
topK	topK=5						topK=10					
Domain	Music			Book			Music			Book		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
PMF	0.4213	0.3138	0.2783	0.4015	0.3182	0.2889	0.5360	0.3508	0.2936	0.4992	0.3480	0.3009
CDAE	0.4266	0.3259	0.2839	0.4046	0.3129	0.2868	0.5471	0.3615	0.3031	0.5139	0.3478	0.2985
CFVAE	0.4101	0.3104	0.2718	0.3763	0.2891	0.2573	0.5342	0.3488	0.2860	0.5077	0.3275	0.2747
AAE	0.4302	0.3326	0.3007	0.3983	0.3159	0.2852	0.5498	0.3712	0.3152	0.5121	0.3491	0.2992
CMF	0.4113	0.3084	0.2748	0.4017	0.3126	0.2920	0.5280	0.3468	0.2906	0.5132	0.3468	0.3055
AAE++	0.4270	0.3287	0.2956	0.3996	0.3200	0.2917	0.5450	0.3661	0.3110	0.5084	0.3535	0.3055
CoNet	0.3380	0.2235	0.2186	0.3265	0.2032	0.2061	0.4699	0.2663	0.2365	0.4505	0.2452	0.2419
sCoNet	0.3508	0.2370	0.2261	0.3263	0.2185	0.2297	0.4846	0.2780	0.2440	0.4490	0.2590	0.2460
ATLRec	0.3540	0.2512	0.2372	0.3292	0.2576	0.2376	0.4935	0.2942	0.2549	0.4522	0.2773	0.2529
DDTCDR	0.3965	0.3061	0.2749	0.3689	0.2992	0.2734	0.5110	0.3412	0.2879	0.4700	0.3300	0.2872
DARec	0.4535	0.3422	0.3060	0.4368	0.3350	0.3013	0.5796	0.3832	0.3229	0.5494	0.3710	0.3161
ETL-JRL	0.4646	0.3586	0.3228	0.4458	0.3389	0.3139	0.5855	0.3968	0.3385	0.5650	0.3828	0.3288
ETL	**0.4686**	**0.3683**	**0.3282**	**0.4496**	**0.3493**	**0.3155**	**0.5942**	**0.4034**	**0.3444**	**0.5669**	**0.3865**	**0.3369**
%Improv.	3.32%	7.62%	7.25%	2.93%	4.26%	4.71%	2.51%	5.27%	6.65%	3.18%	4.17%	6.58%

5.3.4 Performance Comparison

5.3.4.1 Overall Comparison

In this evaluation, after we learned the user representations z_x and z_y from the user behaviors in each domain, we use z_x and z_y to predict the user behaviors by the decoder D_x and D_y, and thus make the recommendation. The performance comparison results are reported in Table 5–2,5–3,5–4, together with the t-test results. Compared to the most competitive baseline, the percentage of relative improvement (%Improv.) of ETL is calculated through $100 * (v_{ETL} - v_{base})/v_{base}$. From these tables, we have the following observations:

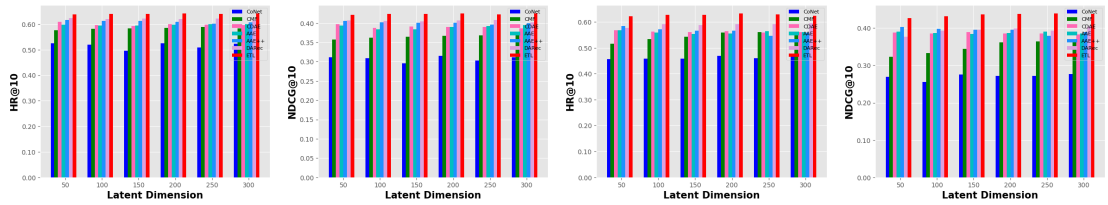
- ETL consistently yields the best performance on the three datasets. In particular, ETL improves over the most competitive baseline with a 7.54%, 9.65% relative gain of NDCG@10 on movie and music domain of Movie & Music. Compared to DDTCDR, ETL explicitly models the joint distribution of user behaviors across domains by the equivalent transformation assumption and achieves better performance. The superior performance of ETL over AAE++ indicates the importance

of cross domain generation in ETL. Compared to ATLRec, the proposed ETL contains the cross-generation stream that helps to better learn the user representations in each domain, and shows better performance.

- Cross domain based methods generally outperform the single domain based methods, indicating the importance of transferring knowledge across domains in recommendation [83, 89, 97]. In particular, in order to transfer knowledge from other domains, CMF utilizes the linear collective matrix factorization technique while AAE++ and DARec employ various deep learning techniques. Compare to CMF, the proposed ETL consistently shows better performance on different datasets in different sparsity levels. For CoNet, it presents an unsatisfactory performance, because the learning mechanism in CoNet breaks the joint behavior pattern in CDR.
- ETL achieves better performance than ETL-JRL which only has the joint reconstruction loss. The reason for this is that the prior regularization in *ELBO* of Eq. 5–3 encourages the preferences to be learned in a specific space with prior knowledge, which benefits the learning process. Moreover, although the original intention of the prior regularization is the joint prior $p(z_x, z_y)$, the results show the standard Gaussian distribution in Section 5.2.4 also works. This verifies the effectiveness of ETL even with a simple prior.
- The improvements of ETL are higher when using Movie as one of the domains. A possible explanation is that the movie domain tends to have more information (*e.g.* background music, prototype book, actor, director, etc.) than other two domains. When more useful information is captured, the recommendation performance can be improved. Previous works based on the idea of shared-user representation focus on modeling the overlapped features and are not able to learn sufficient information from the movie domain, while ETL has the flexibility of capturing more information in the movie domain and thus helps to learn more informative user representations.

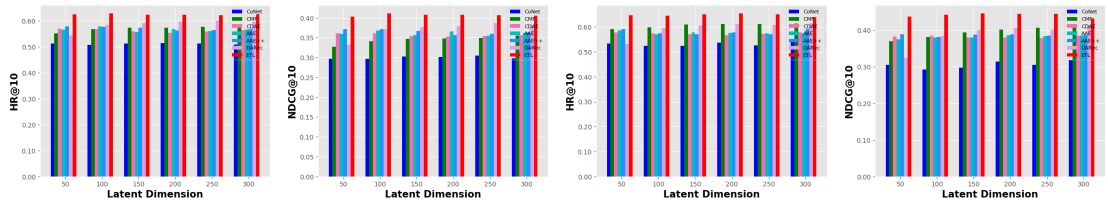
5.3.4.2 Different Latent Dimensions

The latent dimension is an important factor that accounts for the recommendation performance of different methods. We thus investigate the impact of different latent dimensions for different methods. To be specific, we fix the other factors of all methods



(a) Movie & Book-Movie (b) Movie & Book-Movie (c) Movie & Book-Book (d) Movie & Book-Book

Figure 5–4 The effects of different latent dimensions on Movie & Book.



(a) Movie & Music-Movie (b) Movie & Music-Movie (c) Movie & Music-Music (d) Movie & Music-Music

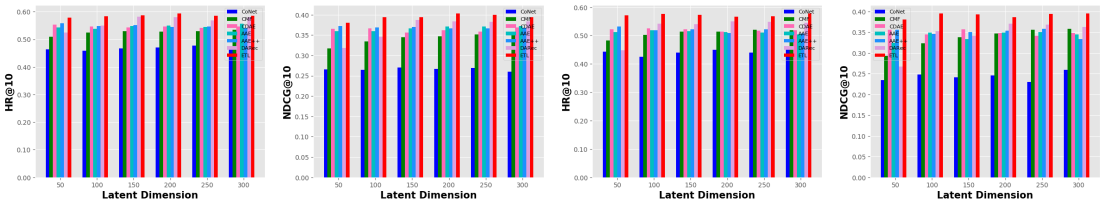
Figure 5–5 The effects of different latent dimensions on Movie & Music.

and allow the latent dimension d to range in [50,100,150,200,250,300]. The results on three datasets are shown in Figure 5–4, Figure 5–5 and Figure 5–6. From these figures, we summarize that:

- Compared with other methods, ETL consistently achieves the best performance on almost every latent dimension. ETL is robust to the change of latent dimension according to the slight change of performance. This verifies the significance of modeling both features, which helps to robustly transfer knowledge with different latent dimensions.
- It is worthwhile to point out that DARec does not perform robustly with the change of latent dimensions. This main reason is that DARec involves the pretraining of AutoRec [165] as the base model, which could accumulate noise with a bad latent dimension because of the two-step scheme.

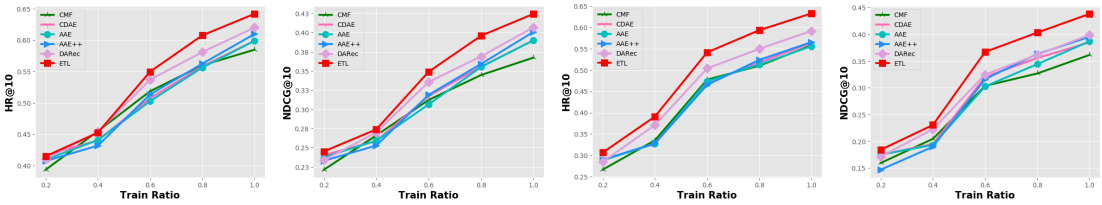
5.3.4.3 Different Sparsity Levels

Since sparsity is an important problem in recommendation systems, it is necessary to investigate whether ETL can still perform better than other methods under more sparse conditions. To this end, we vary sparsity levels of the training data to investigate the method's corresponding performance. In particular, fixing the validation and test set,



(a) Music & Book-Music (b) Music & Book-Music (c) Music & Book-Book (d) Music & Book-Book

Figure 5–6 The effects of different latent dimensions on Music & Book.

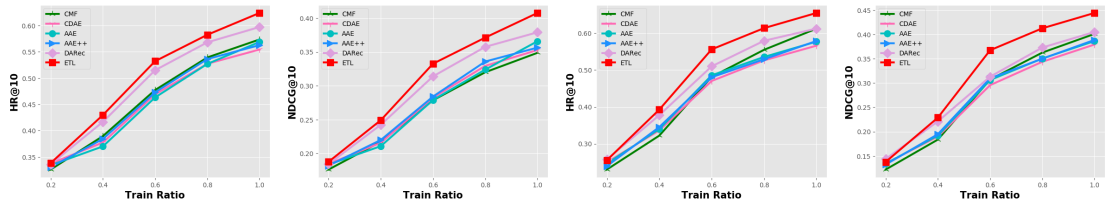


(a) Movie & Book-Movie (b) Movie & Book-Movie (c) Movie & Book-Book (d) Movie & Book-Book

Figure 5–7 The effects of different sparsity levels on Movie & Book. Train ratio means the ratio of the original train data.

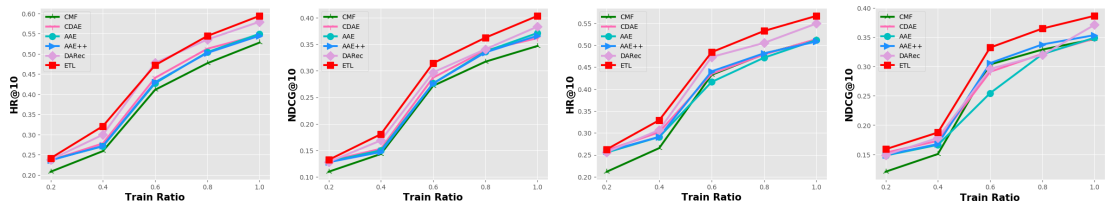
we randomly sample a ratio ranging in [20%,40%,60%,80%,100%] of the original train data as new train data for different sparsity levels. The results are shown in Figure 5–7, Figure 5–8 and Figure 5–9. According to these figures, we can see that:

- It is obvious that ETL performs better than other methods on almost all sparsity levels. We also observe that ETL has less improvement over other counterparts when the data are extremely sparse. One possible reason is that ETL involves the parameterized equivalent transformation that needs necessary data to exert its performance. Too sparse data may limit the training of the transformation, which may be a limitation of ETL and could be explored in the future.
- The extreme sparse cases would cause deterioration to the recommendation performance. As shown in Figure 5–7, Figure 5–8 and Figure 5–9, all methods would have a decrease when training data are less. In the extreme sparse case 0.2, the gap among most CDR methods is not obvious, because the observed data are too less to train a reliable model with 0.2 train ratio. For example, with 0.2 train ratio, we only have 0.01% observed interactions on Movie & Book-Book.



(a) Movie & Music-Movie (b) Movie & Music-Movie (c) Movie & Music-Music (d) Movie & Music-Music

Figure 5–8 The effects of different sparsity levels on Movie & Music. Train ratio means the ratio of the original train data.



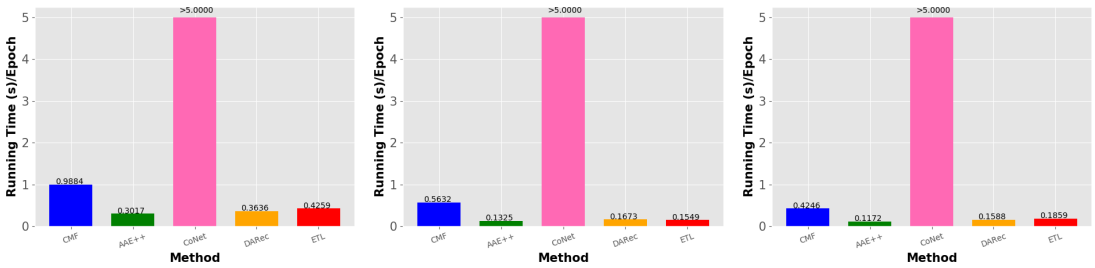
(a) Music & Book-Music (b) Music & Book-Music (c) Music & Book-Book (d) Music & Book-Book

Figure 5–9 The effects of different sparsity levels on Music & Book. Train ratio means the ratio of the original train data.

5.3.4.4 Empirical Running Time Analysis

To investigate the time complexity, we conduct an experiment to compare the empirical running time of each epoch for different models. We conduct the experiments 10 times on the same machine with one 1080Ti GPU. The mean value of running time per epoch is reported in Figure 5–10.

From Figure 5–10, we can see that: (1) CoNet costs the most time because it involves the pretraining of a single domain and stacks multiple cross connection units to model the interactions of different domains. (2) Compared to AAE++ and DARec, ETL takes the running time in the same level and achieves better recommendation performance. This also verifies the time efficiency of the proposed model. (3) The matrix factorization based method CMF costs more time than auto-encoding based methods (*e.g.* AAE++, DARec and ETL). This is because with the same batch size, CMF takes user-item interaction pairs for training, while auto-encoding based methods can take all behaviors of batch users.



(a) Movie & Book (b) Movie & Music (c) Music & Book

Figure 5–10 The empirical running time in each epoch of different methods.

5.3.5 Analysis of Learned User Preferences

As we mentioned before, the equivalent transformation enables ETL to better capture both the domain-specific and overlapped features for CDR. In order to verify this, we design two experiments in this section. Note that our model mainly concentrates on the user-item interactions, specific instances about the overlapped and domain-specific features is beyond the interest of this study.

5.3.5.1 Overlapped Features of User Preferences

In this experiment, we make a hypothesis that the preferences (representations) across domains belonging to the same user usually would have more overlaps than those belonging to different users in CDR. If the overlaps are well captured, there will be an obvious difference between the embedding pair of the same user and that of different users, which means it can be formulated as a binary classification problem. This intuition is similar to that in some multi-view translation works [129-130, 166-168] that use a binary classifier to measure the distribution-level distance between the joint distribution $q(z_x, z_y)$ and $q(z_x, z_{y'})$ where z_x, z_y belong to the same instance and $z_x, z_{y'}$ belong to different instances, and judge whether the two embeddings have consistent features. Thereby, we employ a binary classifier here to verify whether ETL can better capture the overlapped features of user preferences across domains.

In particular, we denote $Z_x \in \mathbb{R}^{N\times d}$ and $Z_y \in \mathbb{R}^{N\times d}$ as the user embedding matrix for $\mathcal{X}$ and $\mathcal{Y}$ domain, respectively. In this experiment, we hold a rule where representations (*i.e.* Z_{xi} and Z_{yi}) belonging to the same user are the paired sample and representations (*i.e.* Z_{xi} and Z_{yj} with $j \neq i$) belonging to different users are the unpaired sample. We obtain the classification data for this experiment by constructing one paired sample and

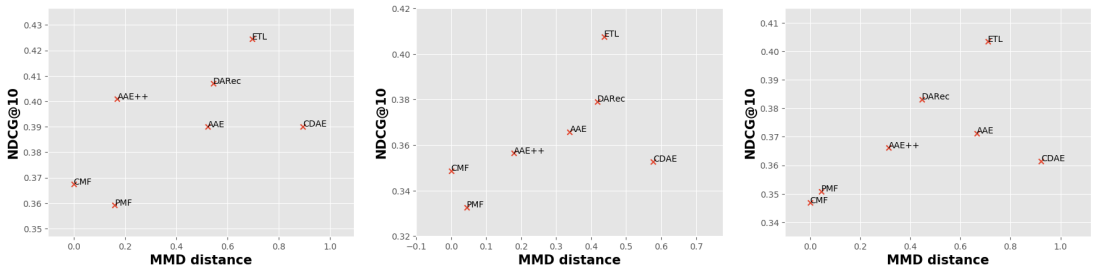
Table 5–5 The binary classification results with AUC. This experiment is designed to verify that the proposed ETL can better learn the overlapped features of user preferences in CDR.

	Movie & Book	Movie & Music	Music & Book
PMF	0.6571	0.7322	0.5709
CDAE	0.6796	0.7267	0.6195
AAE	0.7128	0.7734	0.6516
AAE++	0.7253	0.7777	0.6565
DARec	0.7460	0.8203	0.6934
ETL	**0.9160**	**0.9574**	**0.8826**

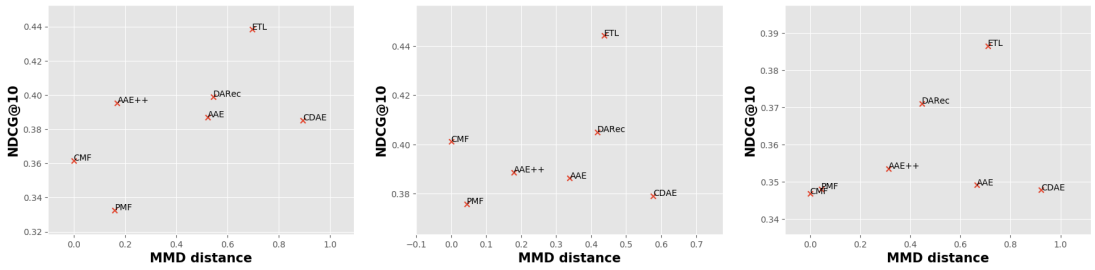
one random unpaired sample for each user. Then, we train a two-layer MLP classifier to perform the binary classification task with (Z_{xi}, Z_{yi}) as label 1 and (Z_{xi}, Z_{yj}) as label 0. In this experiment, the concatenation operation is employed between Z_x and Z_y and we follow the common 6(train)-2(validation)-2(test) setting for classification. The experiment is conducted 10 times and we report the mean value as the results[1]. The results are shown in Table 5–5. There are two key observations:

- The proposed ETL yields the best classification performance. This indicates that ETL has better ability to discriminate embeddings that belongs to the same user. In other words, ETL can better capture the overlapped features of the same user and distinguish the user from others. It is also worthwhile to mention that the result accuracy is consistently lower for Music & Book combo since the correlations between Music and Book domain are not as high as other two domains. As we analyzed before, with the idea of shared-user representation, the CDR model may converge to a comprised solution where both the overlapped and domain-specific features are not well-captured. In contrast, the equivalent transformation in ETL allows the flexibility for user representations in each domain, and can also better learn the overlapped features by the transformed user representations. Learning the overlapped features and domain-specific features is not a conflict for better recommendation performance as long as we find an effective way to do the feature alignment. Through the results, we can see that ETL indeed better captures the

[1]CMF and CoNet both learn one low-dimensional embedding for each user across domains, which makes them not suitable for this experiment here.



(a) Movie & Book-Movie (b) Movie & Music-Movie (c) Music & Book-Music



(d) Movie & Book-Book (e) Movie & Music-Music (f) Music & Book-Book

Figure 5–11 The results of model performance and MMD distance on three benchmarks. This experiment is designed to show that ETL has the capacity to learn the domain-specific features of user preferences.

correlations across domains.

- Two interesting phenomenons are observed in Table 5–5. First, the single domain based methods (PMF, CDAE, AAE) have worse AUC compared with the cross domain based methods (AAE++, DARec and ETL). Second, considering the recommendation performance in Table 5–2,5–3,5–4, it seems that the recommendation accuracy has a positive correlation with the classification AUC, which implies that a model with better classification performance tends to have better recommendation accuracy. These two phenomenons emphasize our motivation of learning z_x and z_y in CDR with equivalent transformation that helps to learn the overlapped features. It is quite different from the idea of other works that learns those features by fully aligning z_x and z_y, where the alignment can be affected by the user behavior prediction loss in each domain.

5.3.5.2 Domain-Specific Features of User Preferences

Assuming the overlapped features are well captured, a CDR model can further boost the recommendation by learning the domain-specific features. The idea of shared-user representation expects to align z_x and z_y, and thus leads to low distance between $q(z_x|x_i)$ and $q(z_y|y_i)$ in two domains. In contrast, when the domain-specific features are captured, $q(z_x|x_i)$ and $q(z_y|y_i)$ are expected to have a larger distribution distance.

We thus can verify whether ETL is able to learn the domain-specific features by measuring the distance between $q(z_x|x)$ and $q(z_y|y)$. In particular, for different methods, we use the z_x and z_y learned by different methods to calculate the Maximum Mean Discrepancy (MMD) distance [169]. Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \rightarrow \mathbb{R}$, then the MMD distance is defined as:

$$MMD[\mathcal{F}, q(z_x|x), q(z_y|y)] = \sup_{f \in \mathcal{F}} (\mathbb{E}_{z_x}[f(z_x)] - \mathbb{E}_{z_y}[f(z_y)])$$

When given n samples from $q(z_x|x)$ and m samples from $q(z_y|y)$, we can empirically estimate the MMD distance with the following equation[1]:

$$\widehat{MMD} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(z_x^i, z_x^j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(z_y^i, z_y^j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(z_x^i, z_y^j)$$

where $z_x^{i\ or\ j}$ and $z_y^{i\ or\ j}$ are embedding vectors and $k(\cdot,\cdot)$ denotes the kernel function. In our case, we use the widely used Gaussian radial basis function (RBF) kernel that is defined as $k(x, x') = exp(-\frac{1}{2\sigma^2\|x-x'\|^2})$. Following recent works [170], we use multiple kernels of different σ and then sum them as the final kernel function[2]. The results of MMD distance and recommendation performance on three benchmarks are shown in Figure 5–11. From this figure, we have the following observations:

- Based on the well-captured overlapped features, ETL can further learn the domain-specific features for better recommendation performance and thus has larger MMD distance than other CDR methods[3]. Unlike the idea of shared-user representation that compresses domain-specific features, the idea of equivalent transformation enables ETL to have the capacity of capturing the domain-specific variations as well as the correlations across domains.

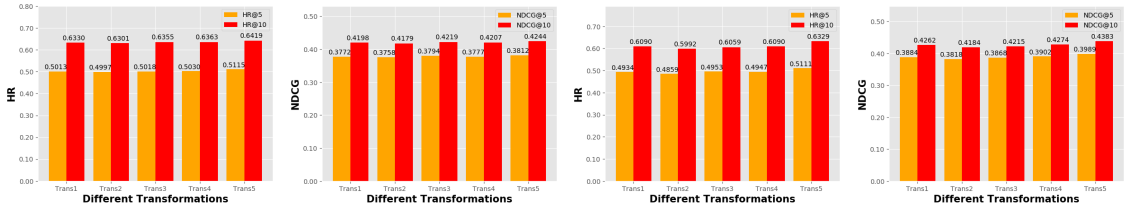
[1] https://en.wikipedia.org/wiki/Kernel_embedding_of_distributions

[2] https://github.com/OctoberChang/MMD-GAN/blob/master/mmd.py

[3] CMF and CoNet both share exactly the same user representation and have zero MMD distance.

Table 5–6 The definitions of different transformations. Trans1 and Trans2 are both not equivalent transformation. Trans3 is one simple equivalent transformation. Trans4 is our extended non-linear equivalent transformation from Trans3.

Trans	Formulation	Type
Trans1	$z_{y\to x} = z_y W_x,\ z_{x\to y} = z_x W_y$	*unconstrained* (linear)
Trans2	$z_{y\to x} = \sigma(z_y W_x^1) W_x^2,\ z_{x\to y} = \sigma(z_x W_y^1) W_y^2$	*unconstrained* (non-linear)
Trans3	$z_{y\to x} = z_y W_x,\ z_{x\to y} = z_x W_y$ $s.t.\ \min \|\|z_x - z_{x\to y} W_x\|\|_F^1$ $s.t.\ \min \|\|z_y - z_{y\to x} W_y\|\|_F^1$	*constrained* (linear, equivalent)
Trans4	$z_{y\to x} = \sigma(z_y W_x^1) W_x^2,\ z_{x\to y} = \sigma(z_x W_y^1) W_y^2$ $s.t.\ \min \|\|z_x - \sigma(z_{x\to y} W_x^1) W_x^2\|\|_F^1$ $s.t.\ \min \|\|z_y - \sigma(z_{y\to x} W_y^1) W_y^2\|\|_F^1$	*constrained* (non-linear, equivalent)



(a) Movie-HR (b) Movie-NDCG (c) Book-HR (d) Book-NDCG

Figure 5–12 The effect of different transformations on Movie & Book. Note that we show the results on Movie & Book here as an example to illustrate our idea. Results on other two benchmarks follow similar pattern.

- It is also worthwhile to point out that large distance between z_x and z_y does not mean better recommendation performance. The recommendation performance in CDR relies on both the overlapped features and the domain-specific features. In this context, we can understand why some single domain methods (*e.g.* CDAE and AAE) have larger MMD distance but achieve poor recommendation performance.

5.3.6 Ablation Study

5.3.6.1 The Effects of Different Transformations

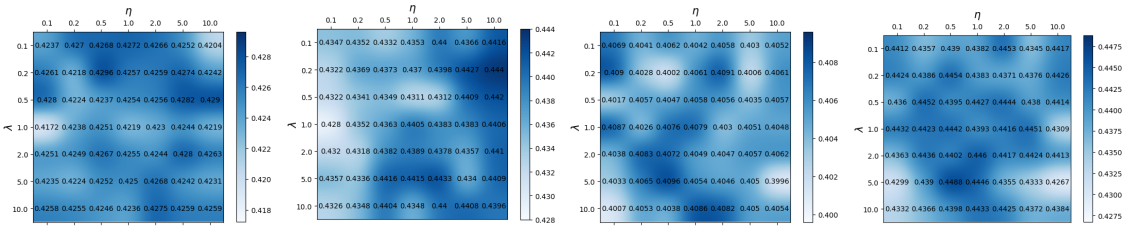
In case that we adopt the orthogonal transformation for the equivalent transformation, it is curious to explore what are the effects of other transformations. We here conduct an experiment with other 4 different transformations for ETL. The results on Movie & Book are shown in Figure 5–12, where Trans5 is the used orthogonal transformation and Trans1~4 indicate other 4 different transformations that are defined in Table 5–6. From

Table 5–7 The effects of different priors on three benchmarks. Uniform indicates samples from $U(0,1)$, Laplace indicates samples from $L(0,1)$ and Gaussian means samples from $\mathcal{N}(0,1)$. For multi-variate Gaussian (MVGaussian), we set it as $MVG = \mathcal{N}(0,1) + \mathcal{N}(3,1)$ to form multiple peaks.

Movie and Book												
topK	topK=5						topK=10					
Domain	Movie			Book			Movie			Book		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
Uniform	0.5030	0.3761	0.3348	0.4976	0.3761	0.3368	0.6341	0.4185	0.3527	0.6211	0.4167	0.3536
Laplace	0.5095	**0.3826**	0.3406	**0.5149**	**0.4052**	0.3637	0.6418	**0.4254**	0.3583	0.6320	**0.4437**	0.3792
MVGaussian	0.5044	0.3770	0.3349	0.4960	0.3788	0.3399	0.6398	0.4208	0.3529	0.6191	0.4186	0.3564
Gaussian	**0.5115**	0.3812	**0.3431**	0.5111	0.3989	**0.3705**	**0.6419**	0.4244	**0.3608**	**0.6329**	0.4383	**0.3861**
Movie and Music												
topK	topK=5						topK=10					
Domain	Movie			Music			Movie			Music		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
Uniform	0.4831	0.3561	0.3168	0.5123	0.3875	0.3470	0.6214	0.4005	0.3350	0.6382	0.4295	0.3644
Laplace	**0.4931**	**0.3684**	**0.3291**	0.5297	**0.4063**	**0.3660**	**0.6274**	**0.4117**	**0.3464**	**0.6578**	**0.4477**	**0.3823**
MVGaussian	0.4734	0.3517	0.3103	0.4992	0.3824	0.3433	0.6113	0.3949	0.3281	0.6294	0.4242	0.3603
Gaussian	0.4891	0.3632	0.3224	**0.5314**	0.4037	0.3653	0.6241	0.4076	0.3404	0.6550	0.4442	0.3819
Music and Book												
topK	topK=5						topK=10					
Domain	Music			Book			Music			Book		
Metrics	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
Uniform	0.4403	0.3305	0.3054	0.4394	0.3338	0.2989	0.5629	0.3700	0.3212	0.5576	0.372	0.3147
Laplace	0.4614	0.3568	0.3230	**0.4623**	**0.3572**	**0.3217**	0.5766	0.3954	0.3388	**0.5768**	**0.3928**	0.3364
MVGaussian	0.4392	0.3431	0.3056	0.4325	0.3303	0.2952	0.5675	0.3818	0.3219	0.5590	0.3700	0.3116
Gaussian	**0.4686**	**0.3683**	**0.3282**	0.4496	0.3493	0.3155	**0.5942**	**0.4034**	**0.3444**	0.5669	0.3865	**0.3369**

Table 5–6 and Figure 5–12, we can see:

- The orthogonal transformation (Trans5) outperforms other 4 transformations. This is reasonable since only Trans5 is the equivalent one as well as does not introduce spurious correlations between users after knowledge transfer.
- It is worthwhile to notice that the equivalent transformations (Trans3∼5) present better performance compared to the non-equivalent ones (Trans1∼2). This verifies the correctness of our ET assumption that encourages ETL to learn better coverage of user preferences. The *unconstrained* ones deteriorate the recommendation performance since they are too flexible and may easily be over-fitting on the sparse data in recommendation. Moreover, even with other equivalent transformations (Trans3∼4), ETL still performs better than DARec according to Table 5–2∼5–4.



(a) Movie & Book Movie-NDCG@10 (b) Movie & Book Book-NDCG@10 (c) Movie & Music Movie-NDCG@10 (d) Movie & Music Music-NDCG@10

Figure 5–13 The effects of hyper-parameters λ, η on two datasets. We show the results of two datasets here to illustrate the observations and the results of another dataset follow similar pattern.

5.3.6.2 The Effects of Different Priors

In Section 5.2.5, we take standard Gaussian distributions as priors. Different priors have different prior knowledge for the learned user preferences. Thus it is interesting to see the effects of different priors. In this part, we explore the effects of different priors on the model performance. In particular, we apply four common priors in generative modeling and show the results in Table 5–7. From Table 5–7, we have the following key observations:

- Compared to other priors, uniform prior consistently leads to poor recommendation performance. It is because users usually have non-uniform interests on different topics and the uniform distribution does not match the implicit distribution of user preferences.
- Gaussian and Laplace prior have comparable performance. On Movie & Music, Laplace prior gain better performance than Gaussian prior, while on other two benchmarks, they have slight gaps and Gaussian prior has better performance in some cases. Thus the choice of prior should be data-dependent.
- MVGaussian prior generally does not have better performance than Gaussian prior. This is mainly because it is hard to define the parameters of MVGaussian and it is easy to involve human bias. Instead, the most common prior (*i.e.* Gaussian) can provide satisfied performance.

5.3.6.3 Hyper-parameter Sensitivity

In ETL, λ controls the weight of the equivalent transformation and η weights the prior regularization. To investigate how the hyper-parameters λ, η influence the performance of ETL, we conduct an experiment to study the sensitivity of these two hyper-parameters.

The corresponding results are shown in Figure 5–13.

From this figure, we can see that the best hyper-parameter setting is different on different datasets. For example, in Figure 5–13 (b), ETL performs better when η is around 10.0 and λ is around 0.2 on Movie & Book-Book. While for Movie & Music-Music in Figure 5–13 (d), the hyper-parameters are around $\eta = 0.5, \lambda = 5.0$. This is because different datasets have different distributions and require different weights of the equivalent transformation and the prior regularization.

5.4 Summary

In this Chapter, based on the equivalent transformation assumption, we propose a novel distribution matching based CDR model that advocates to capture both the overlapped and domain-specific features. Extensive experiments on three public benchmarks demonstrate the effectiveness of the proposed model for boosting the recommendation accuracy.

Although ETL has shown better performance than previous methods, there are still some inadequacies that limit its potential. Firstly, we make a simplified Gaussian assumption for the joint prior $p(z_x, z_y)$ in Section 5.2.4. A complex prior that provides more informative prior knowledge can be explored later. Secondly, in common CDR, usually popular related domains are considered for recommendation. In the future, it is interesting to study how to do the recommendation among unpopular domains. Thirdly, how to incorporate ETL with auxiliary information to boost the recommendation performance and even solve the cold-start problem is also interesting.

Chapter 6 Conclusion

In this thesis, we focus on deep learning on complex graphs to put forward the graph learning algorithms into practice. Specifically, we study the issues of complex graphs in different levels, *i.e.* signed directed networks in the edge level, attribute-missing graphs in the attribute level and user-item bipartite graphs of CDR in the graph level, and propose the corresponding solutions. The summarization of the proposed methods will be given in this Chapter and meanwhile we will discuss some future directions on algorithms and applications.

6.1 Summary of Contributions

Chapter 3 studies the representation learning problem on signed directed networks. Specifically, the structures in signed directed networks are coupled by signs and directions, which raises challenges to model the structural patterns. We propose to decouple the modeling of signs and directions in a principled way by maximizing the variational *ELBO* on the log-likelihoods of the observed data. Further, based on the *ELBO*, we develop a decoupled variational embedding (DVE) model to learn node embeddings for downstream tasks. The empirical results on three real-world datasets confirm the promising performance of DVE on learning representative node embeddings.

In Chapter 4, we study the learning problem of attribute-missing graphs. In this Chapter, we analyze that recent graph algorithms either suffer from the sampling bias of structures or are incompatible to learn on attribute-missing graphs. We consider attributes and structures as two heterogeneous views to describe the information of nodes, and model the two views by decoupled encoder while preserving their joint distribution by deep generative techniques. Following this, we make a *shared-latent space* assumption of attributes and structures, and further propose a novel distribution matching based GNN framework for learning on attribute-missing graphs. Extensive experiments on seven benchmarks demonstrate that our method outperforms the current popular graph learning algorithms on both link prediction task and the newly introduced *node attribute completion* task.

Chapter 5 studies how to collaborate the learning of different user-item bipartite

graphs in CDR. We analyze that most models in CDR are based the idea of shared user representation, and thus resort to learn the overlapped features and compromise the domain-specific features. Although several works employ MLP as the mapping function to allow the learning flexibility for user preference in each domain, they usually require heuristic human knowledge of choosing training samples to avoid over-fitting for the mapping function. We propose to learn the both features by an equivalent transformation assumption which means each user's preference in each domain can be mutually converted to each other with equivalent transformation. Based on this assumption, we derive the *ELBO* of the joint log-likelihood of the user behaviors across domains as a surrogate objective function to optimize the recommendation target. Quantitative and qualitative results on three widely-used benchmarks confirm the superiority of our model.

6.2 Future Works

Although we have present some useful methods to handle the learning problems of deep learning on complex graphs, there is still a long way to go in this area. The potential works on algorithm designs and application developments deserve further exploration in the future, which are summarized as follows.

- **More effective posterior solutions**. One common drawback of the methods in Chapter 4 and Chapter 5 is the imperfect posterior approximation by mean field theory. In real data, the latent codes are usually dependent even when given the observed data, which causes limitations of the approximate posterior. Employing more effective posterior approximation techniques will be useful to improve the representation learning and could be studied in future.
- **Dynamic complex graphs with temporal information**. In many scenarios, the complex graphs are usually not static and change the structures or attributes along the time. For instance, the users in social media may change his or her social relationships or profiles when facing different things offline. The traffic networks in a city may be extended or cut according to the policy from the government. Exploring the involving patterns in these dynamic graphs encourages us to better understand how the edges build and what the mutual influence of node structures and attributes, and facilitates better knowledge of several graph prediction tasks, *e.g.* link prediction and label prediction. Research on this topic may involve the techniques in sequential modeling, online learning and Bayesian reasoning,

which can be investigated in future works.

- **Potential applications**. Deep learning on complex graphs is related to many practical applications. For example, the edges of signs and directions are very useful for friend recommendation and enemy detection. Meanwhile, learning on attribute-missing graphs in Chapter 4 relates to many practical tasks, e.g. auto-tagging in e-commence systems, and fraud detection in financial systems. Employing the designed algorithms into these practical applications might provide more value for the society.

Appendix A Deduction in Chapter 3

A.1 Variational Evidence Lower Bound

The detailed derivation of the *ELBO* in Eq. 3–6 is shown as follows.

$$\log P(\mathcal{E}) = \int q_\phi(Z_s, Z_t|\mathcal{E}) \log p_\theta(\mathcal{E}) dZ_s dZ_t \tag{A–1}$$

$$= \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(\mathcal{E}, Z_s, Z_t)}{p_\theta(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \tag{A–2}$$

$$= \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(\mathcal{E}, Z_s, Z_t)}{q_\phi(Z_s, Z_t|\mathcal{E})} \cdot \frac{q_\phi(Z_s, Z_t|\mathcal{E})}{p_\theta(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \tag{A–3}$$

$$\begin{aligned} &= \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(\mathcal{E}, Z_s, Z_t)}{q_\phi(Z_s, Z_t)|\mathcal{E}} dZ_s dZ_t \\ &\quad + \int q_\phi(Z_s, Z_t) \log \frac{q_\phi(Z_s, Z_t|\mathcal{E})}{p_\theta(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \end{aligned} \tag{A–4}$$

$$\begin{aligned} &= \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(\mathcal{E}, Z_s, Z_t)}{q_\phi(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \\ &\quad + D_{KL}[q_\phi(Z_s, Z_t)|\mathcal{E}||p_\theta(Z_s, Z_t|\mathcal{E})] \end{aligned} \tag{A–5}$$

We then have the formulation of the *ELBO* in Eq. 3–4 as:

$$\mathcal{L} = \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(\mathcal{E}, Z_s, Z_t)}{q_\phi(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \tag{A–6}$$

$$\begin{aligned} &= \int q_\phi(Z_s, Z_t|\mathcal{E}) \log \frac{p_\theta(Z_s, Z_t)}{q_\phi(Z_s, Z_t|\mathcal{E})} dZ_s dZ_t \\ &\quad + \int q_\phi(Z_s, Z_t|\mathcal{E}) \log p_\psi(\mathcal{E}|Z_s, Z_t) dZ_s dZ_t \end{aligned} \tag{A–7}$$

$$= - D_{KL}[q_\phi(Z_s, Z_t|\mathcal{E})|p_\theta(Z_s, Z_t)] + \mathop{\mathbb{E}}_{q_\phi(Z_s, Z_t|\mathcal{E})}[p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{A–8}$$

Following the proposition and prior assumption, we have the *ELBO* in Eq. 3–6 as follows:

$$\mathcal{L} = - D_{KL}[q_\phi(Z_s, Z_t|\mathcal{E})|p_\theta(Z_s, Z_t)] + \mathop{\mathbb{E}}_{q_\phi(Z_s, Z_t|\mathcal{E})}[p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{A–9}$$

$$= \int q_{\phi_s}(Z_s|\mathcal{E}) q_{\phi_t}(Z_s|\mathcal{E}) \log \frac{p_\theta(Z_s) p_\theta(Z_t)}{q_{\phi_s}(Z_s|\mathcal{E}) q_{\phi_t}(Z_s|\mathcal{E})} dZ_s dZ_t + \mathop{\mathbb{E}}_{q_\phi(Z_s, Z_t|\mathcal{E})}[p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{A–10}$$

$$= - D_{KL}[q_{\phi_s}(Z_s|\mathcal{E})||p_\theta(Z_s)] - D_{KL}[q_{\phi_t}(Z_t|\mathcal{E})||p_\theta(Z_t)] + \mathop{\mathbb{E}}_{q_\phi(Z_s, Z_t|\mathcal{E})}[p_\psi(\mathcal{E}|Z_s, Z_t)] \tag{A–11}$$

Bibliography

[1] KIPF T N, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[C]//International Conference on Learning Representations (ICLR). 2017.

[2] HAMILTON W, YING Z, LESKOVEC J. Inductive representation learning on large graphs[C]//Advances in neural information processing systems. 2017: 1024-1034.

[3] LI Q, WU X M, LIU H, et al. Label efficient semi-supervised learning via graph filtering[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 9582-9591.

[4] BATTAGLIA P W, HAMRICK J B, BAPST V, et al. Relational inductive biases, deep learning, and graph networks[J]. ArXiv preprint arXiv:1806.01261, 2018.

[5] CHEN W, XIONG W, YAN X, et al. Variational Knowledge Graph Reasoning[J]. ArXiv preprint arXiv:1803.06581, 2018.

[6] KIPF T, FETAYA E, WANG K C, et al. Neural relational inference for interacting systems[J]. ArXiv preprint arXiv:1802.04687, 2018.

[7] WANG H, ZHANG F, WANG J, et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems[C]//Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018: 417-426.

[8] YING R, HE R, CHEN K, et al. Graph convolutional neural networks for web-scale recommender systems[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 974-983.

[9] WANG X, HE X, CAO Y, et al. Kgat: Knowledge graph attention network for recommendation[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 950-958.

[10] PAPADOPOULOS S, KOMPATSIARIS Y, VAKALI A, et al. Community detection in social media[J]. Data Mining and Knowledge Discovery, 2012, 24(3): 515-554.

[11] PEROZZI B, AL-RFOU R, SKIENA S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710.

[12] SHI C, LI Y, ZHANG J, et al. A survey of heterogeneous information network analysis[J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 29(1): 17-37.

[13] QIU J, DONG Y, MA H, et al. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec[C]//Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. 2018: 459-467.

[14] DEFFERRARD M, BRESSON X, VANDERGHEYNST P. Convolutional neural networks on graphs with fast localized spectral filtering[C]//Advances in neural information processing systems. 2016: 3844-3852.

[15] LI M, CHEN S, CHEN X, et al. Actional-structural graph convolutional networks for skeleton-based action recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 3595-3603.

[16] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient Estimation of Word Representations in Vector Space[C]//International Conference on Learning Representation. 2013.

[17] GROVER A, LESKOVEC J. Node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[18] BOJCHEVSKI A, SHCHUR O, ZÜGNER D, et al. Netgan: Generating graphs via random walks[C/OL]//Proceedings of Machine Learning Research: Proceedings of the 35th International Conference on Machine Learning: vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 2018: 610-619. http://proceedings.mlr.press/v80/bojchevski18a.html.

[19] QIU J, DONG Y, MA H, et al. Netsmf: Large-scale network embedding as sparse matrix factorization[C]//The World Wide Web Conference. 2019: 1509-1520.

[20] TANG J, QU M, WANG M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th International Conference on World Wide Web. 2015: 1067-1077.

[21] WANG D, CUI P, ZHU W. Structural deep network embedding[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 1225-1234.

[22] CAO S, LU W, XU Q. Deep neural networks for learning graph representations[C]//Thirtieth AAAI conference on artificial intelligence. 2016.

[23] NT H, MAEHARA T. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters[J]. ArXiv preprint arXiv:1905.09550, 2019.

[24] LI Q, HAN Z, WU X M. Deeper insights into graph convolutional networks for semi-supervised learning[C]//Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[25] XU B, SHEN H, CAO Q, et al. Graph convolutional networks using heat kernel for semi-supervised learning[C]//Proceedings of the 28th International Joint Conference on Artificial Intelligence. 2019: 1928-1934.

[26] VELIKOVI P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks[J/OL]. International Conference on Learning Representations, 2018. https://openreview.net/forum?id=rJXMpikCZ.

[27] YING R, YOU J, MORRIS C, et al. Hierarchical Graph Representation Learning with Differentiable Pooling[C]//Neural Information Processing Systems. 2018: 4800-4810.

[28] GAO H, JI S. Graph u-nets[J]. ArXiv preprint arXiv:1905.05178, 2019.

[29] LEE J, LEE I, KANG J. Self-Attention Graph Pooling[J]., 2019.

[30] LI G, MULLER M, THABET A, et al. Deepgcns: Can gcns go as deep as cnns?[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 9267-9276.

[31] XU K, LI C, TIAN Y, et al. Representation learning on graphs with jumping knowledge networks[J]. International Conference on Machine Learning (ICML), 2018.

[32] VELIKOVI P, FEDUS W, HAMILTON W L, et al. Deep Graph Infomax[C/OL]//International Conference on Learning Representations. 2019. https://openreview.net/forum?id=rklz9iAcKQ.

[33] HU Z, DONG Y, WANG K, et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks[C]//Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2020.

[34] KUNEGIS J, PREUSSE J, SCHWAGEREIT F. What is the added value of negative links in online social networks?[C]//Proceedings of the 22nd international conference on World Wide Web. 2013: 727-736.

[35] CACHEDA F, BLANCO R, BARBIERI N. Characterizing and Predicting Users’ Behavior on Local Search Queries[J/OL]. ACM Trans. Web, 2018, 12(2): 11:1-11:32. http://doi.acm.org/10.1145/3157059. DOI: 10.1145/3157059.

[36] VICTOR P, VERBIEST N, CORNELIS C, et al. Enhancing the Trust-based Recommendation Process with Explicit Distrust[J/OL]. ACM Trans. Web, 2013, 7(2): 6:1-6:19. http://doi.acm.org/10.1145/2460383.2460385. DOI: 10.1145/2460383.2460385.

[37] OU M, CUI P, PEI J, et al. Asymmetric transitivity preserving graph embedding[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 1105-1114.

[38] AIELLO L M, BARRAT A, SCHIFANELLA R, et al. Friendship Prediction and Homophily in Social Media[J/OL]. ACM Trans. Web, 2012, 6(2): 9:1-9:33. http://doi.acm.org/10.1145/2180861.2180866. DOI: 10.1145/2180861.2180866.

[39] HUANG X, SONG Q, LI Y, et al. Graph recurrent networks with attributed random walks[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019: 732-740.

[40] LEI CHEN J B, Shunwang Gong, BRONSTEIN M M. Attributed Random Walk as Matrix Factorization[J]. Graph Representation Learning Workshop NeurIPS, 2019.

[41] YANG Q, ZHOU Z H, GONG Z, et al. Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings[M]. Springer, 2019.

[42] MAN T, SHEN H, JIN X, et al. Cross-Domain Recommendation: An Embedding and Mapping Approach.[C]//International joint Conferences on artificial intelligence. 2017.

[43] KAZEMI H, SOLEYMANI S, TAHERKHANI F, et al. Unsupervised Image-to-Image Translation Using Domain-Specific Variational Information Bound[C]//BENGIO S, WALLACH H, LAROCHELLE H, et al. Advances in neural information processing systems. Curran Associates, Inc., 2018.

[44] CHANG H Y, WANG Z, CHUANG Y Y. Domain-Specific Mappings for Generative Adversarial Style Transfer[C]//European conference on computer vision. 2020.

[45] ZHU F, WANG Y, CHEN C, et al. A Deep Framework for Cross-Domain and Cross-System Recommendations[C]//International joint conference on artificial intelligence. International joint Conferences on artificial intelligence Organization, 2018.

[46] BI Y, SONG L, YAO M, et al. DCDIR: A deep cross-domain recommendation system for cold start users in insurance domain[C]//ACM SIGIR conference on research and development in information retrieval. 2020.

[47] ZHANG Z, CUI P, ZHU W. Deep learning on graphs: A survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2020.

[48] WU Z, PAN S, CHEN F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[49] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[50] DAI Q, LI Q, TANG J, et al. Adversarial network embedding[C]//Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[51] MAKHZANI A. Implicit autoencoders[J]. ArXiv preprint arXiv:1805.09804, 2018.

[52] MAKHZANI A, SHLENS J, JAITLY N, et al. Adversarial Autoencoders[C/OL]//International Conference on Learning Representation. 2016. http://arxiv.org/abs/1511.05644.

[53] GAO H, HUANG H. Deep Attributed Network Embedding.[C]//IJCAI: vol. 18. 2018: 3364-3370.

[54] LIAO L, HE X, ZHANG H, et al. Attributed social network embedding[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(12): 2257-2270.

[55] WANG X, CUI P, WANG J, et al. Community preserving network embedding[C]//Thirty-first AAAI conference on artificial intelligence. 2017.

[56] CUI P, WANG X, PEI J, et al. A survey on network embedding[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 31(5): 833-852.

[57] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[58] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]// Advances in neural information processing systems. 2017: 5998-6008.

[59] WANG S, TANG J, AGGARWAL C, et al. Signed network embedding in social media[C]//Proceedings of the 2017 SIAM international conference on data mining. 2017: 327-335.

[60] CHANG S, HAN W, TANG J, et al. Heterogeneous network embedding via deep architectures[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2015: 119-128.

[61] KARPATHY A, et al. Cs231n convolutional neural networks for visual recognition[J]. Neural networks, 2016, 1.

[62] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[63] XU K, HU W, LESKOVEC J, et al. How Powerful are Graph Neural Networks?[C/OL]//International Conference on Learning Representations. 2019. https://openreview.net/forum?id=ryGs6iA5Km.

[64] GEHRING J, AULI M, GRANGIER D, et al. Convolutional sequence to sequence learning[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. 2017: 1243-1252.

[65] CHEN J, MA T, XIAO C. Fastgcn: fast learning with graph convolutional networks via importance sampling[J]. ArXiv preprint arXiv:1801.10247, 2018.

[66] CHEN J, ZHU J, SONG L. Stochastic Training of Graph Convolutional Networks with Variance Reduction[C]//International Conference on Machine Learning. 2018: 941-949.

[67] CHUNG F R, GRAHAM F C. Spectral graph theory[M]. American Mathematical Soc., 1997.

[68] RONG Y, HUANG W, XU T, et al. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification[C]//International Conference on Learning Representations. 2019.

[69] HINTON G E, SRIVASTAVA N, KRIZHEVSKY A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. ArXiv preprint arXiv:1207.0580, 2012.

[70] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[71] CHEN X, ZHANG Y, TSANG I, et al. Learning Robust Node Representation on Graphs[J]. ArXiv preprint arXiv:2008.11416, 2020.

[72] LESKOVEC J, HUTTENLOCHER D, KLEINBERG J. Predicting positive and negative links in online social networks[C]//Proceedings of the 19th international conference on World wide web. 2010: 641-650.

[73] TANG J, CHANG S, AGGARWAL C, et al. Negative link prediction in social media[C]//Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. 2015: 87-96.

[74] KUNEGIS J, SCHMIDT S, LOMMATZSCH A, et al. Spectral analysis of signed graphs for clustering, prediction and visualization[C]//Proceedings of the 2010 SIAM International Conference on Data Mining. 2010: 559-570.

[75] HSIEH C J, CHIANG K Y, DHILLON I S. Low rank modeling of signed networks[C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. 2012: 507-515.

[76] DERR T, MA Y, TANG J. Signed Graph Convolutional Network[J]. ArXiv preprint arXiv:1808.06354, 2018.

[77] YUAN S, WU X, XIANG Y. SNE: signed network embedding[C]//Pacific-Asia conference on knowledge discovery and data mining. 2017: 183-195.

[78] KIM J, PARK H, LEE J E, et al. Side: representation learning in signed directed networks[C]//Proceedings of the 2018 World Wide Web Conference on World Wide Web. 2018: 509-518.

[79] CARTWRIGHT D, HARARY F. Structural balance: a generalization of Heider's theory.[J]. Psychological review, 1956, 63(5): 277.

[80] WANG S, AGGARWAL C, TANG J, et al. Attributed signed network embedding[C]//Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017: 137-146.

[81] BERKOVSKY S, KUFLIK T, RICCI F. Cross-domain mediation in collaborative filtering[C]//International Conference on User Modeling. 2007: 355-359.

[82] HU L, CAO J, XU G, et al. Personalized recommendation via cross-domain triadic factorization[C]//Proceedings of the 22nd international conference on World Wide Web. 2013: 595-606.

[83] SINGH A P, GORDON G J. Relational learning via collective matrix factorization[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008: 650-658.

[84] LIAN J, ZHANG F, XIE X, et al. CCCFNet: a content-boosted collaborative filtering neural network for cross domain recommender systems[C]//Proceedings of the 26th international conference on World Wide Web companion. 2017: 817-818.

[85] MIRBAKHSH N, LING C X. Improving top-n recommendation for cold-start users via cross-domain information[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2015, 9(4): 33.

[86] GAO S, LUO H, CHEN D, et al. Cross-domain recommendation via cluster-level latent factor model[C]//Joint European conference on machine learning and knowledge discovery in databases. 2013: 161-176.

[87] LINDEN G, SMITH B, YORK J. Amazon. com recommendations: Item-to-item collaborative filtering[J]. IEEE Internet computing, 2003(1): 76-80.

[88] MIRBAKHSH N, LING C X. Clustering-based factorized collaborative filtering[C]//Proceedings of the 7th ACM conference on Recommender systems. 2013: 315-318.

[89] YUAN F, YAO L, BENATALLAH B. DARec: Deep Domain Adaptation for Cross-Domain Recommendation via Transferring Rating Patterns[J]. IJCAI 2019, 2019.

[90] ELKAHKY A M, SONG Y, HE X. A multi-view deep learning approach for cross domain user modeling in recommendation systems[C]//Proceedings of the 24th International Conference on World Wide Web. 2015: 278-288.

[91] HU G, ZHANG Y, YANG Q. Conet: Collaborative cross networks for cross-domain recommendation[C]//Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018: 667-676.

[92] KANAGAWA H, KOBAYASHI H, SHIMIZU N, et al. Cross-domain recommendation via deep domain adaptation[C]//European Conference on Information Retrieval. 2019: 20-29.

[93] PAN S J, YANG Q. A survey on transfer learning[J]. IEEE Transactions on knowledge and data engineering, 2009, 22(10): 1345-1359.

[94] HUANG P S, HE X, GAO J, et al. Learning deep structured semantic models for web search using clickthrough data[C]//Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013: 2333-2338.

[95] ZHAO C, LI C, FU C. Cross-Domain Recommendation via Preference Propagation GraphNet[C]//International conference on information and knowledge management. 2019.

[96] YINGCE X, DI H, TAO Q, et al. Dual Learning for Machine Translation[C]// Advances in neural information processing systems. 2016.

[97] LI P, TUZHILIN A. DDTCDR: Deep Dual Transfer Cross Domain Recommendation[J]. ACM International Conference on Web Search and Data Mining, 2020.

[98] GANIN Y, USTINOVA E, AJAKAN H, et al. Domain-adversarial training of neural networks[J]. The Journal of Machine Learning Research, 2016, 17(1): 2096-2030.

[99] LI Y, XU J J, ZHAO P P, et al. ATLRec: An Attentional Adversarial Transfer Learning Network for Cross-Domain Recommendation[J]. Journal of Computer Science and Technology, 2020.

[100] KINGMA D P, WELLING M. Auto-encoding variational bayes[J]. ArXiv preprint arXiv:1312.6114, 2013.

[101] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]//Neural Information Processing Systems. 2014: 2672-2680.

[102] GULRAJANI I, KUMAR K, AHMED F, et al. Pixelvae: A latent variable model for natural images[J]. International Conference on Learning Representations, 2017.

[103] ZHAO S, SONG J, ERMON S. Infovae: Information maximizing variational autoencoders[J]. ArXiv preprint arXiv:1706.02262, 2017.

[104] BURDA Y, GROSSE R, SALAKHUTDINOV R. Importance weighted autoencoders[J]. ArXiv preprint arXiv:1509.00519, 2015.

[105] NOWOZIN S, CSEKE B, TOMIOKA R. F-gan: Training generative neural samplers using variational divergence minimization[C]//Neural Information Processing Systems. 2016: 271-279.

[106] CHEN X, DUAN Y, HOUTHOOFT R, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets[C]//Neural Information Processing Systems. 2016: 2172-2180.

[107] MAO X, LI Q, XIE H, et al. Least squares generative adversarial networks[C]// Proceedings of the IEEE International Conference on Computer Vision. 2017: 2794-2802.

[108] ARJOVSKY M, CHINTALA S, BOTTOU L. Wasserstein generative adversarial networks[C]//International Conference on Machine Learning. 2017: 214-223.

[109] MESCHEDER L, NOWOZIN S, GEIGER A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks[C]// Proceedings of the 34th International Conference on Machine Learning-Volume 70. 2017: 2391-2400.

[110] SRIVASTAVA A, VALKOV L, RUSSELL C, et al. Veegan: Reducing mode collapse in gans using implicit variational learning[C]//Advances in Neural Information Processing Systems. 2017: 3308-3318.

[111] ROSCA M, LAKSHMINARAYANAN B, MOHAMED S. Distribution matching in variational inference[J]. ArXiv preprint arXiv:1802.06847, 2018.

[112] QIAN Y, ADALI S. Extended structural balance theory for modeling trust in social networks[C]//Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on. 2013: 283-290.

[113] QIAN Y, ADALI S. Foundations of Trust and Distrust in Networks: Extended Structural Balance Theory[J/OL]. ACM Trans. Web, 2014, 8(3): 13:1-13:33. http://doi.acm.org/10.1145/2628438. DOI: 10.1145/2628438.

[114] ZHENG H, YAO J, ZHANG Y, et al. Degeneration in VAE: in the Light of Fisher Information Loss[J]. ArXiv, 2018, abs/1802.06677.

[115] ZHENG H, YAO J, ZHANG Y, et al. Understanding VAEs in Fisher-Shannon Plane[C]//Proceedings of the 33rd Association for the Advancement of Artificial Intelligence. 2019.

[116] TOMCZAK J M, WELLING M. VAE with a VampPrior[J]. ArXiv preprint arXiv:1705.07120, 2017.

[117] REZENDE D J, MOHAMED S. Variational inference with normalizing flows[J]. ArXiv preprint arXiv:1505.05770, 2015.

[118] YIN M, ZHOU M. Semi-implicit variational inference[J]. International Conference on Machine Learning, 2018.

[119] PU Y, GAN Z, HENAO R, et al. Variational autoencoder for deep learning of images, labels and captions[C]//Advances in neural information processing systems. 2016: 2352-2360.

[120] KUSNER M J, PAIGE B, HERNÁNDEZ-LOBATO J M. Grammar variational autoencoder[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. 2017: 1945-1954.

[121] RENDLE S, FREUDENTHALER C, GANTNER Z, et al. BPR: Bayesian personalized ranking from implicit feedback[C]//Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. 2009: 452-461.

[122] GALLIER J. Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey[J]. ArXiv preprint arXiv:1601.04692, 2016.

[123] TIELEMAN T, HINTON G. RMSprop gradient optimization[J]. URL http://www. cs. toronto. edu/tijmen/csc321/slides/lecture_slides_lec6. pdf, 2014.

[124] MAATEN L V D, HINTON G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(Nov): 2579-2605.

[125] KIPF T N, WELLING M. Variational graph auto-encoders[J]. ArXiv preprint arXiv:1611.07308, 2016.

[126] WU F, ZHANG T, SOUZA JR A H D, et al. Simplifying graph convolutional networks[J]., 2019.

[127] LINDVALL T. Lectures on the coupling method[M]. Courier Corporation, 2002.

[128] TZENG E, HOFFMAN J, SAENKO K, et al. Adversarial discriminative domain adaptation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 7167-7176.

[129] ISOLA P, ZHU J Y, ZHOU T, et al. Image-to-image translation with conditional adversarial networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1125-1134.

[130] ZHU J Y, ZHANG R, PATHAK D, et al. Toward multimodal image-to-image translation[C]//Advances in neural information processing systems. 2017.

[131] LIU M Y, BREUEL T, KAUTZ J. Unsupervised image-to-image translation networks[C]//Neural Information Processing Systems. 2017: 700-708.

[132] MCCALLUM A K, NIGAM K, RENNIE J, et al. Automating the construction of internet portals with machine learning[J]. Information Retrieval, 2000, 3(2): 127-163.

[133] SEN P, NAMATA G, BILGIC M, et al. Collective classification in network data[J]. AI magazine, 2008, 29(3): 93-93.

[134] NAMATA G, LONDON B, GETOOR L, et al. Query-driven active surveying for collective classification[C]//10th International Workshop on Mining and Learning with Graphs. 2012.

[135] SHCHUR O, MUMME M, BOJCHEVSKI A, et al. Pitfalls of graph neural network evaluation[J]. ArXiv preprint arXiv:1811.05868, 2018.

[136] MCAULEY J, TARGETT C, SHI Q, et al. Image-based recommendations on styles and substitutes[C]//ACM SIGIR conference on research and development in information retrieval. 2015.

[137] IMEK Ö, Et.al. Navigating networks by using homophily and degree[J]. National Academy of Sciences, 2008,

[138] HU L, JIAN S, CAO L, et al. Hers: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation[C]//Association for the advancement of artificial intelligence: vol. 33. 2019: 3830-3837.

[139] LÜ L, PAN L, ZHOU T, et al. Toward link predictability of complex networks[J]. National academy of sciences, 2015, 112(8): 2325-2330.

[140] ADAMIC L A, ADAR E. Friends and neighbors on the web[J]. Social networks, 2003, 25(3): 211-230.

[141] CANTADOR I, FERNÁNDEZ-TOBIAS I, BERKOVSKY S, et al. Cross-domain recommender systems[G]//Recommender systems handbook. Springer, 2015: 919-959.

[142] KHAN M M, IBRAHIM R, GHANI I. Cross domain recommender systems: A systematic literature review[J]. ACM Computing Surveys (CSUR), 2017, 50(3): 36.

[143] CREMONESI P, TRIPODI A, TURRIN R. Cross-domain recommender systems[C]//International conference on data mining workshops. 2011.

[144] LI B, YANG Q, XUE X. Transfer learning for collaborative filtering via a rating-matrix generative model[C]//Proceedings of the 26th annual international conference on machine learning. 2009: 617-624.

[145] DUMOULIN V, BELGHAZI I, POOLE B, et al. Adversarially learned inference[J]. ArXiv preprint arXiv:1606.00704, 2016.

[146] DU C, DU C, XIE X, et al. Multi-view Adversarially Learned Inference for Cross-domain Joint Distribution Matching[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 1348-1357.

[147] CHEN Z, YANG Z, WANG X, et al. Multivariate-Information Adversarial Ensemble for Scalable Joint Distribution Matching[C]//International Conference on Machine Learning. 2019: 1112-1121.

[148] ZHANG C, BÜTEPAGE J, KJELLSTRÖM H, et al. Advances in variational inference[J]. IEEE transactions on pattern analysis and machine intelligence, 2018.

[149] HUANG X, LIU M Y, BELONGIE S, et al. Multimodal Unsupervised Image-to-image Translation[C]//European Conference on Computer Vision. 2018.

[150] ZHU J Y, PARK T, ISOLA P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2223-2232.

[151] GUSTAFSON W H, ZELMANOWITZ J M. On matrix equivalence and matrix equations[J]. Linear Algebra and Its Applications, 1979, 27: 219-224.

[152] HU Y, KOREN Y, VOLINSKY C. Collaborative filtering for implicit feedback datasets[C]//2008 Eighth IEEE International Conference on Data Mining. 2008: 263-272.

[153] OARD D W, KIM J, et al. Implicit feedback for recommender systems[C]// Proceedings of the AAAI workshop on recommender systems: vol. 83. 1998.

[154] MNIH A, SALAKHUTDINOV R R. Probabilistic matrix factorization[C]// Advances in neural information processing systems. 2008: 1257-1264.

[155] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer, 2009(8): 30-37.

[156] WU Y, DUBOIS C, ZHENG A X, et al. Collaborative denoising auto-encoders for top-n recommender systems[C]//Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. 2016: 153-162.

[157] DAWEN L, RAHUL K, MATTHEW H, et al. Variational autoencoders for collaborative filtering[C]//International conference on world wide web. 2018.

[158] GANIN Y, LEMPITSKY V. Unsupervised Domain Adaptation by Backpropagation[C/OL]//ICML'15: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. Lille, France: JMLR.org, 2015: 1180-1189. http://dl.acm.org/citation.cfm?id=3045118.3045244.

[159] HE X, LIAO L, ZHANG H, et al. Neural collaborative filtering[C]//Proceedings of the 26th international conference on world wide web. 2017: 173-182.

[160] ZAMANI H, CROFT W B. Learning a joint search and recommendation model from user-item interactions[C]//International conference on web search and data mining. 2020.

[161] DU Y, LIU H, QU Y, et al. Online personalized next-item recommendation via long short term preference learning[C]//Pacific Rim international conference on artificial intelligence. 2018.

[162] YUAN F, YAO L, BENATALLAH B. Adversarial collaborative neural network for robust recommendation[C]//ACM SIGIR conference on research and development in information retrieval. 2019.

[163] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. ArXiv preprint arXiv:1412.6980, 2014.

[164] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks[C]//Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010: 249-256.

[165] SEDHAIN S, MENON A K, SANNER S, et al. Autorec: Autoencoders meet collaborative filtering[C]//Proceedings of the 24th International Conference on World Wide Web. 2015: 111-112.

[166] GUO C, WU D. Canonical Correlation Analysis (CCA) Based Multi-View Learning: An Overview[J]. ArXiv preprint arXiv:1907.01693, 2019.

[167] CHEN X, CHEN S, YAO J, et al. Learning on Attribute-Missing Graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020: 1-1. DOI: 10.1109/TPAMI.2020.3032189.

[168] SHI Y, PAN Y, XU D, et al. Multiview Alignment and Generation in CCA via Consistent Latent Encoding[J]. Neural Computation, 2020, 32(10): 1936-1979.

[169] GRETTON A, BORGWARDT K M, RASCH M J, et al. A kernel two-sample test[J]. The Journal of Machine Learning Research, 2012, 13(1): 723-773.

[170] LI C L, CHANG W C, CHENG Y, et al. MMD GAN: Towards Deeper Understanding of Moment Matching Network[C]//Advances in neural information processing systems. Curran Associates, Inc., 2017.

Publications

[1] Xu Chen, Jiangchao Yao, Maosen Li, Ya Zhang and Yanfeng Wang. Decoupled Variational Embedding for Signed Directed Networks, ACM Transactions on the Web (TWEB), doi: 10.1145/3408298, 2020. (CCF B)

[2] Xu Chen, Siheng Chen, Jiangchao Yao, Huangjie Zheng, Ya Zhang and Ivor W. Tsang. Learning on Attribute-Missing Graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), doi: 10.1109/TPAMI.2020.3032189, 2020. (CCF A)

[3] Xu Chen, Ya Zhang, Ivor Tsang, Yuangang Pan and Jingchao Su. Towards Equivalent Transformation of User Preferences in Cross Domain Recommendation, ACM Transactions on Information Systems (TOIS), 2021. (accept, CCF A)

[4] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang and Qi Tian. Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. (CCF A)

[5] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang and Qi Tian. Symbiotic Graph Neural Networks for 3D Skeleton-based Human Action Recognition and Motion Prediction, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), doi: 10.1109/TPAMI.2021.3053765, 2020. (CCF A)

[6] Jingchao Su, Xu Chen, Ya Zhang, Siheng Chen, Dan Lv and Chenyang Li. Collaborative Adversarial Learning for Relational Learning on Multiple Bipartite Graphs, IEEE International Conference on Knowledge Graphs (ICKG), 2020.

[7] Chenyang Li, Xu Chen, Ya Zhang, Siheng Chen, Dan Lv and Yanfeng Wang,. Dual Graph Embedding for Object-Tag Link Prediction on the Knowledge Graph, IEEE International Conference on Knowledge Graphs (ICKG), 2020.

[8] Kenan Cui, Xu Chen, Jiangchao Yao, Ya Zhang. Variational Collaborative Learning for User Probabilistic Representation, Proceedings of the Association for

the Advancement of Artificial Intelligence Conference on Artificial Intelligence workshops (AAAI workshops), 2019.

[9] Hanqing Zhao, Xu Chen, Jiangchao Yao, Ya Zhang and Yanfeng Wang. Recommendation with Hybrid Interest Model, Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence workshops (ICDM workshops), 2018.