FEATURE REPRESENTATION LEARNING IN COMPLEX NETWORKS

SHEN XIAO

PhD

The Hong Kong Polytechnic University

2019

The Hong Kong Polytechnic University

Department of Computing

Feature Representation Learning in Complex Networks

Shen Xiao

A thesis submitted in partial fulfilment of

the requirements for the degree of

Doctor of Philosophy

July 2018

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

__________________________________(Signature)

SHEN Xiao (Name of student)

Abstract

Complex networks are ubiquitous in the real world. Learning appropriate feature representations for complex networks is important for a wide variety of graph mining tasks. Motivated by this, in this thesis, we propose four models to learn informative feature vector representations for nodes or edges in the networks, which can effectively and efficiently address several canonical graph mining tasks. In the first work, we utilize a feature-engineering approach to define explicit topological features for nodes and edges in the influence maximization (IM) scenario. Next, we propose three deep network embedding models to learn the low-dimensional latent node vector representations which can well preserve the original network structures and properties. Preserving various network properties is important for learning informative feature representations for different graph mining tasks. Thus, the first two proposed deep network embedding models focus on preserving the asymmetric network transitivity and the signed network property to effectively address the typical graph mining tasks within a single network, including node classification, node clustering and link prediction. In addition, the third proposed deep network embedding model incorporates domain adaptation technique into deep network embedding to learn generalized and comparable feature representations which can effectively address the cross-network prediction task.

In the first work, we propose a cross-network learning (CNL) framework to leverage the greedy seed selection and influence propagation knowledge pre-learned from a smaller source network to select seed nodes and remove inactive edges for multiple larger target networks. To address domain discrepancy, we assign lower weights to the explicit topological features which perform less similarly between the source network and the target network. In addition, we utilize a fuzzy self-training algorithm to iteratively retrain the prediction model based on not only the fully labeled instances from the source network, but also the most confident predicted instances in the target network with their predicted fuzzy labels. Extensive experiments demonstrate that the proposed CNL model can achieve a good trade-off between the efficiency and effectiveness of the IM task in the target networks.

In addition, the three proposed deep network embedding models focus on addressing several open issues in current network embedding research, i.e., asymmetric network embedding, signed network embedding and cross-network embedding. Firstly, an asymmetry-aware deep network embedding (AsDNE) model is proposed, which is composed of two semi-supervised stacked auto-encoders (SAEs) to preserve the asymmetric outward and inward network proximities. To well capture the asymmetric relationships, we design pairwise constraints to map node pairs with bi-directionally strong connections much closer than those with strong connection in only one direction. Extensive experiments demonstrate that AsDNE can learn task-independent network representations outperforming the state-of-the-art network embedding algorithms, in both directed and undirected networks. Secondly, we propose a deep network embedding model with structural balance preservation (DNE-SBP) for signed networks. A semi-supervised SAE is employed to reconstruct the signed adjacency matrix, where larger penalty is added to make the SAE focus more on reconstructing the scarce negative links than the abundant positive links. To well preserve the structural balance property, we design pairwise constraints to map positively connected nodes much closer than negatively connected nodes. Extensive experiments demonstrate the superiority of DNE-SBP over the state-of-the-art network embedding algorithms for graph representation learning in signed networks. Finally, we propose a cross-network deep network embedding (CDNE) model, which innovatively integrates deep network embedding and domain adaptation techniques to learn label-discriminative and network-invariant node vector representations. Two semi-supervised SAEs are employed to embed nodes from the source network and the target network into a unified low-dimensional latent space. In addition, similar nodes within a network and across networks would be mapped closer to each other, based on their network structures, attributes and labels. Extensive experiments demonstrate that CDNE significantly outperforms the state-of-the-art network embedding algorithms for node classification in the target network.

Publications

The following papers, published, in press or submitted, are partial outputs of my PhD study in PolyU.

Conference Papers

1. X. Shen, Q. Dai, S. Mao and F.-l. Chung, "Network together: Node classification via cross-network deep network embedding," *submitted to ACM SIGIR Conference on Research and Development in Information Retrieval,* 2019.
2. Q. Dai, X. Shen, L. Zhang, Q. Li and D. Wang, "Adversarial training methods for network embedding," *accepted by International Conference on World Wide Web,* 2019.
3. S. Mao, X. Shen, and F.-l. Chung, "Deep domain adaptation based on multi-layer joint kernelized distance," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.
4. X. Shen, F.-l. Chung, and S. Mao, "Leveraging cross-network information for graph sparsification in influence maximization," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.
5. X. Shen and F.-l. Chung, "Deep network embedding with aggregated proximity preserving," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, 2017.
6. X. Yi, X. Shen, W. Lu, T. S. Chan, and F.-l. Chung, "Persuasion driven influence analysis in online social networks," in *Proceedings of the International Joint Conference on Neural Networks*, 2016.

Journal Papers

7. X. Shen, S. Mao, and F.-l. Chung, "Asymmetry-aware deep network embedding," *under 2nd review by IEEE Transactions on Knowledge and Data Engineering,* 2018.
8. X. Shen and F.-l. Chung, "Deep network embedding for graph representation learning in signed networks," *IEEE Transactions on Cybernetics,* 2018.
9. X. Shen, S. Mao, and F.-l. Chung, "Cross-network learning with fuzzy labels for seed selection and graph sparsification in influence maximization," *under 3rd round review by IEEE Transactions on Fuzzy Systems,* 2018.

Acknowledgements

First and foremost, I would like to express my sincere appreciation to my supervisor, Prof. Korris Fu-lai Chung, who always gives me enlightening guidance, insightful suggestions, great support, continuous encouragement, and enough freedom to let me explore my interested research.

I would also like to thank Prof. Maggie Wenjie Li, Prof. Lei Zhang, Prof. Jiannong Cao, Prof. Xiao-Ming Wu and Prof. Qin Lu in the Department of Computing, the Hong Kong Polytechnic University, who gave me insightful comments and suggestions during my presentations for the guided study and confirmation. Their great encouragement makes me have more confidence and passion for my research. I also would like to appreciate Prof. Wai Hung TSANG and Prof. Kwok Wai CHEUNG's constructive comments to help me further improve the work in my thesis.

I also want to thank my colleagues and friends in PolyU, Yumeng Guo, Wei Lu, Chengyao Chen, Quanyu Dai, Jiaxin Chen and Sitong Mao, for their stimulating research discussions and sharing happiness with me in my daily life.

In addition, I would like to thank the Research Grants Council of Hong Kong and the Department of Computing, the Hong Kong Polytechnic University, for awarding me the Hong Kong PhD Fellowship and the COMP Scholarship, which provides me an excellent research environment.

Finally and most importantly, I would like to express my deepest thanks to my beloved parents, my sister, and my husband. Without their great support, continuous encouragement, and sincere love, I cannot complete this journey.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Nowadays, networks are ubiquitous in the real world, such as social networks, citation networks, collaboration networks, brain networks, protein-protein interaction networks, transportation networks, computer networks, communication networks and so on. Network science, “the study of network representations of physical, biological, and social phenomena leading to predictive models of these phenomena [1]”, is a young and active discipline of mathematics, statistics, physics, biology, social sciences and computer sciences.

A network can be represented as a graph, which consists of a set of vertices (also called nodes) and a set of edges (also called links) capturing the relationships between nodes. A network can be directed or undirected, weighted or unweighted, according to the relationships between nodes. For example, in the Facebook social network, an undirected edge connecting two social network users indicates the bi-directional friendship between them. While in a paper citation network, a directed edge from paper A to paper B indicates that paper A unidirectionally cites paper B. In addition, in a collaboration network, if the relationship between two nodes is defined as “author A collaborated with author B”, then we say this collaboration network is unweighted. If the relationship is defined as “author A collaborated with author B 4 times”, then the collaboration network is weighted. Moreover, a network can be homogeneous or heterogeneous. If all the nodes in a network belong to a single type and all the edges in a network also belong to a single type, then we say this network is homogeneous. In contrast, if there are more than one type of nodes or more than one type of edges in a network, then we say this network is heterogeneous. For example, a multimedia network containing different types of nodes, i.e., image, video and text, is heterogeneous.

1.1 Graph Mining Tasks

Real-world networks are with very complex network structures. Mining useful information from complex networks can benefit a wide variety of graph analytics tasks, such as node classification, node clustering, node/link retrieval and ranking, link prediction, link classification and network visualization. Node classification aims to predict node labels in a network [2]. For example, in a protein-protein interaction network, one might be interested in identifying the functional labels of proteins [3]. Node clustering aims to cluster nodes in a network into several disjoint partitions, where nodes belonging to the same partition should be more tightly connected than those across different partitions [4]. It can be applied to detect communities in social networks [5], [6]. Node/link retrieval and ranking tasks aim to prioritize nodes or links in a network according to specific rules [7]. For example, in viral marketing, one aims to select top-*k* seed nodes which are the most influential to spread the information in a network [8]. Also, in large-scale influence maximization (IM) problem, one aims to retrieve a fraction of edges to be removed which are the most useless for influence propagation [9]. Link prediction aims to predict whether a link exists between two nodes in a network and it can be utilized to recommend new friends to users in a social network [10]. Link classification aims to predict edge labels in a network. For example, one might be interested in predicting the signed labels of links in a signed social network [11], [12], or inferring the type of social relationships among family, colleagues, classmates and friends in a heterogeneous social network [13]. In addition, motivated by the recent success of transfer learning [14], some innovative cross-network graph mining tasks [8], [9], [13], [15], [16], [17], [18],

[19], [20] have been proposed to leverage the knowledge pre-learned from a mature source network to predict the labels of nodes or edges in a newly formed target network.

1.2 Graph Representation Learning

All the aforementioned canonical graph mining tasks require a set of informative and discriminative feature vector representations pre-defined for nodes or edges in the networks. A former solution is to utilize a feature engineering approach to manually define explicit features for nodes or edges based on their topological structures. Some widely used hand-crafted topological features [20], [21], [22] include Degree, Weighted Degree, Eigenvector Centrality, Closeness Centrality, Betweenness Centrality, HITS Hub/Authority, PageRank Score, Clustering Coefficient, Modularity Class Size, Number of Triangles, Common Neighbors, Jaccard Coefficient and Adamic Adar Score.

In addition, network embedding has recently drawn significant attentions and it can automatically learn the low-dimensional latent vector representations with the preservation of original network properties. Existing network embedding techniques can be categorized into three families, namely random walk based [23], [24], [25], [26], [27], matrix factorization based [6], [28], [29], [30], [31], [32], [33], and deep learning based [4], [5], [10], [11], [12], [17], [34], [35], [36]. A comprehensive survey of network embedding research can be found in [37], [38], [39], [40]. Random walk based embedding algorithms employ the truncated random walks sampled from a given network to exploit network structures. Then, borrowing the idea of word embedding [41] in natural language processing (NLP), the low-dimensional continuous node vector representations with neighborhood preservation would be learned. In addition, the network structures can be represented via a matrix form, such as adjacency matrix [10], k-step transition probability matrix [29], and high-order proximity matrix [30]. On one hand, the matrix factorization based network embedding models linearly project the associated matrix into a low-rank space, via Singular Value Decomposition (SVD) [30] or Nonnegative Matrix Factorization (NMF) [6], which requires a high time complexity at least super-quadratic to the number of nodes in the network. On the other hand, to better capture the non-linear complex underlying network structure, a family of deep network embedding algorithms [4], [5], [10], [11], [12], [34], [42] adopts stacked auto-encoder (SAE) to reconstruct the associated matrix so as to learn the non-linear node vector representations. In practice, SAE is more efficient than the matrix factorization techniques, just with the time complexity being linear to the number of nodes in the network [4]. Based on the low-dimensional node vector representations learned by the network embedding algorithms, one can simply apply the vector-based machine learning techniques to efficiently and effectively address a wide variety of graph mining tasks, such as node classification [10], [11], [42], network clustering [4], [5], link prediction [10], [11], [12], node recommendation [10], [30] and network visualization [43].

1.3 Contributions

To exploit the latest advances of machine learning, deep learning and data mining techniques in canonical graph mining tasks, a set of informative feature vector representations for nodes or edges in the networks should be pre-defined. To this end, we propose four models in this thesis to learn appropriate feature vector representations to efficiently and effectively address several graph mining tasks. Figure 1.1 shows the connections between the four proposed models.

On one hand, from the perspective of feature representation learning technique, the first model utilizes a feature engineering approach to manually define the explicit topological features which can reflect the influence of nodes in a network

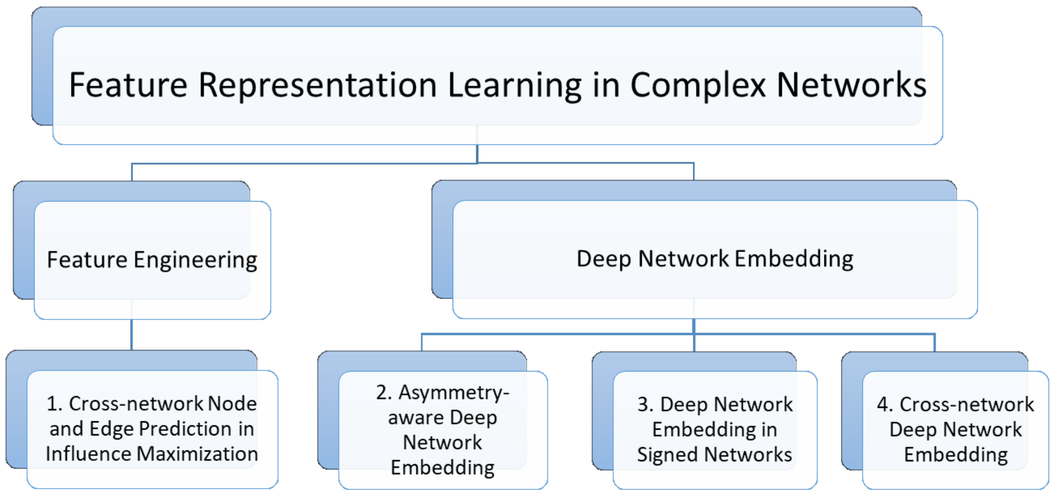

Feature Representation Learning in Complex Networks
Feature Engineering
Deep Network Embedding
1. Cross-network Node and Edge Prediction in Influence Maximization
2. Asymmetry-aware Deep Network Embedding
3. Deep Network Embedding in Signed Networks
4. Cross-network Deep Network Embedding

Figure 1.1: Connections between four proposed models in this thesis.

in the IM scenario. While the other three proposed models correspond to the state-of-the-art deep network embedding techniques which can automatically learn the low-dimensional latent node vector representations. Preserving various network properties by the network embedding algorithms is important for learning informative feature representations for different graph mining tasks. For example, preserving high-order network proximities can benefit node classification and node clustering [34], [29], and preserving asymmetric network transitivity is advantageous for link prediction [30], [26]. In addition, preserving structural balance property is important for learning informative feature representations for the graph analytics tasks in the signed networks [44], [45]. Motivated by this, we proposed two deep network embedding models to learn the latent vector representations which can well preserve the asymmetric network property and the signed network property, respectively. In addition, it has been shown that incorporating domain adaptation techniques into feature representation learning process can effectively address the prediction tasks across different domains, in the areas of computer vision (CV) [46], [47] and NLP [48], [49]. Motivated by this, we proposed a cross-network deep network embedding model to learn generalized and comparable feature vector representations to effectively address the graph mining tasks across different networks. The three deep network embedding models can successfully address several important issues which have not been well investigated by existing network embedding research, i.e., asymmetric network embedding, signed network embedding and cross-network embedding.

On the other hand, from the perspective of graph mining applications, the second model for asymmetric network embedding and the third model for signed network embedding target at the graph prediction tasks involving one network. While the first and the fourth models making use of feature engineering and feature learning respectively further employ domain adaptation techniques to address the cross-network node/link prediction tasks. In the following sub-sections, we would further elaborate on the contributions of each proposed model.

1.3.1 Cross-network Node and Edge Prediction in Influence Maximization

The conventional IM problem [50] has been extensively studied, aiming at selecting a limited number of seed nodes to maximize the influence spread in a given network. However, very little work exists for the cross-network IM problem. To fill this gap, we propose a cross-network learning (CNL) model to leverage the knowledge pre-learned from a smaller source network with the IM task being successfully completed, to help maximize the influence in the new larger target networks for two important tasks, i.e., seed selection and graph sparsification. On one hand, we consider cross-network seed selection as a cross-network node prediction task, with the goal of selecting the nodes most likely to act as seeds for each target network, by leveraging the greedy seed selection knowledge pre-learned from a source network. On the other hand, we consider graph sparsification as a cross-network edge prediction task, with the goal of removing the edges least likely to contribute to influence propagation in the target network, by leveraging the influence propagation knowledge pre-learned from the source network. In the cross-network node and edge prediction tasks in the IM scenario, we define some explicit features for node and edge based on their topological structures, which can reflect the influence of a node in a network. To address the domain discrepancy issue, we assign lower absolute weights to the features which perform more differently between the source network and the target network. In addition, a fuzzy self-training algorithm is proposed to iteratively retrain the prediction model by leveraging not only the fully labeled instances from the source network, but also the most confident predicted instances in the target network with their predicted fuzzy labels. Experimental results in the real-world datasets demonstrate that the proposed CNL model can achieve a good trade-off between the efficiency and effectiveness of the IM task in the target networks.

1.3.2 Asymmetry-Aware Deep Network Embedding

Network transitivity and proximities should be asymmetric in both directed and undirected networks [30], [26]. However, most existing network embedding algorithms fail to capture such important asymmetric properties. In this regard, we propose an asymmetry-aware deep network embedding (AsDNE) model to preserve the asymmetric outward and inward network proximities. Existing deep network embedding models [4], [10], [11], [34] employing one SAE would only consider a specific node as a source role in its input raw vector, when learning the corresponding hidden vector representation. In contrast, the proposed AsDNE model employs two semi-supervised SAEs to learn the outward and inward latent vector representations for each node, by considering the node with a source role and a target role, respectively, within its *K*-step network connections. In addition, to better capture the asymmetric relationships, we devise pairwise constraints to map node pairs with bi-directionally strong connections much closer than those node pairs with strong connection in only one direction. We evaluated the network representation learning ability of AsDNE for two graph mining tasks, namely multi-label node classification and link sign prediction. Extensive experimental results in both directed and undirected real-world networks demonstrate that the proposed AsDNE model can learn task-independent network representations outperforming the state-of-the-art network embedding algorithms.

1.3.3 Deep Network Embedding in Signed Networks

The signed networks containing both positive and negative links have pretty distinct properties from the unsigned counterparts [51], [52]. However, the vast majority of the state-of-the-art network embedding algorithms have only been designed for unsigned networks. To fill this gap, we propose a deep network embedding with structural balance preservation (DNE-SBP) model to learn network representations for the signed networks. The DNE-SBP model employs a semi-supervised SAE to reconstruct the adjacency connections of a signed network. As the connections are overwhelmingly positive in the real-world signed networks, we impose a larger penalty to make the SAE focus more on reconstructing the scarce negative links than the abundant positive links. In addition, to preserve the structural balance property of the signed networks, we design pairwise constraints to map the positively connected nodes much closer than the negatively connected nodes in the low-dimensional embedding space. Based on the network representations learned by DNE-SBP, we conduct link sign prediction and community detection in signed networks. Extensive experimental results in the real-world signed networks demonstrate the superiority of the proposed DNE-SBP model over the state-of-the-art network embedding algorithms for graph representation learning in signed networks.

1.3.4 Cross-Network Deep Network Embedding

Existing network embedding algorithms only target for a single network, which aim to preserve the proximities between nodes within one network. However, many important graph mining tasks involve more than one network, such as cross-network node classification or cross-network link prediction. It has been shown that the single-network embedding algorithms fail to learn generalized and comparable feature representations across different networks [53], [54]. To address this, we propose an innovative cross-network deep network embedding (CDNE) model to capture the proximities between nodes within a network and across different networks. The CDNE model employs two semi-supervised SAEs to embed nodes from the source network and the target network into a unified low-dimensional latent space. It should be the first work to integrate deep network embedding and domain adaptation techniques to learn label-discriminative and network-invariant feature vector representations for cross-network node classification. Extensive experimental results in the real-world networks demonstrate that CDNE can achieve significantly better node classification performance in the target network, as compared to the state-of-the-art network embedding algorithms.

The rest of this thesis is organized as follows. Chapter 2 introduces the proposed CNL model for cross-network node and edge prediction in IM. Chapter 3 introduces the proposed AsDNE model which can learn low-dimensional node vector representations with the preservation of the asymmetric outward and inward network proximities in both directed and undirected networks. Chapter 4 introduces the proposed DNE-SBP for signed network embedding which can well preserve the structural balance property of the signed networks. Chapter 5 introduces the proposed CDNE model for learning label-discriminative and network-invariant feature vector representations for cross-network node classification. Chapter 6 gives the conclusions of this thesis and our future research plans.

Chapter 2
Cross-network Node and Edge Prediction in Influence Maximization

2.1 Introduction

Nowadays, people tend to trust the information from their friends, relatives and families more than that from general advertising media [55]. Thus, one promising marketing strategy for product promotion is to select a few most influential initial users to give them free samples and let them influence their friends through the word-of-mouth effect. Such an approach is referred to viral marketing [56]. Motivated by the idea of viral marketing, the influence maximization (IM) problem can be formulated as a discrete optimization problem [50], i.e., to select *k* seed nodes (i.e. initial users) in a given network such that the expected number of nodes influenced by the *k* seeds (i.e. influence spread) is as large as possible, under a certain influence cascade model. Existing IM algorithms can be grouped into two families, namely, greedy algorithms and heuristic algorithms. Generally, greedy algorithms [50], [57], [58], [59] are highly effective (i.e. achieving large influence spread) but with low efficiency (i.e. long running time). In contrast, heuristic algorithms [59], [60], [61], [62] are highly efficient but generally fail to guarantee large influence spread.

Although the IM problem has been extensively studied, very little work addresses this problem in a cross-network scenario. Suppose that a company intends to promote a new product through multiple media (e.g. online social networks, email communication networks, and telephone communication networks) by viral marketing. Then, if the company greedily selects the initial users for each target network independently, it should be extremely time consuming and expensive. On the other hand, if the company heuristically selects initial users, it might be difficult to guarantee a high influence spread in a new target network, where the company lacks enough preliminary knowledge about which types of users should be most influential. To address such a cross-network IM problem more effectively and efficiently, we propose a cross-network learning (CNL) model to leverage the knowledge pre-learned from a smaller source network to help maximize the influence in the new larger target networks. Specifically, we address the cross-network IM problem from two perspectives, i.e., seed selection and graph sparsification.

On one hand, we consider seed selection for IM across multiple networks as a cross-network node prediction task. Here, the goal is to select the nodes most likely to act as seeds for each target network, by leveraging the greedy seed

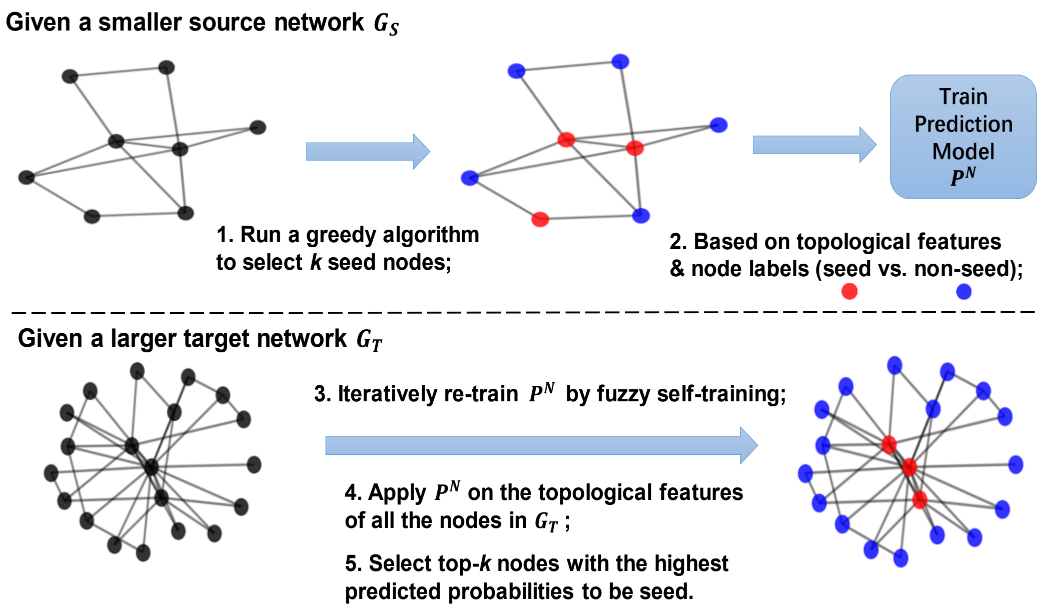
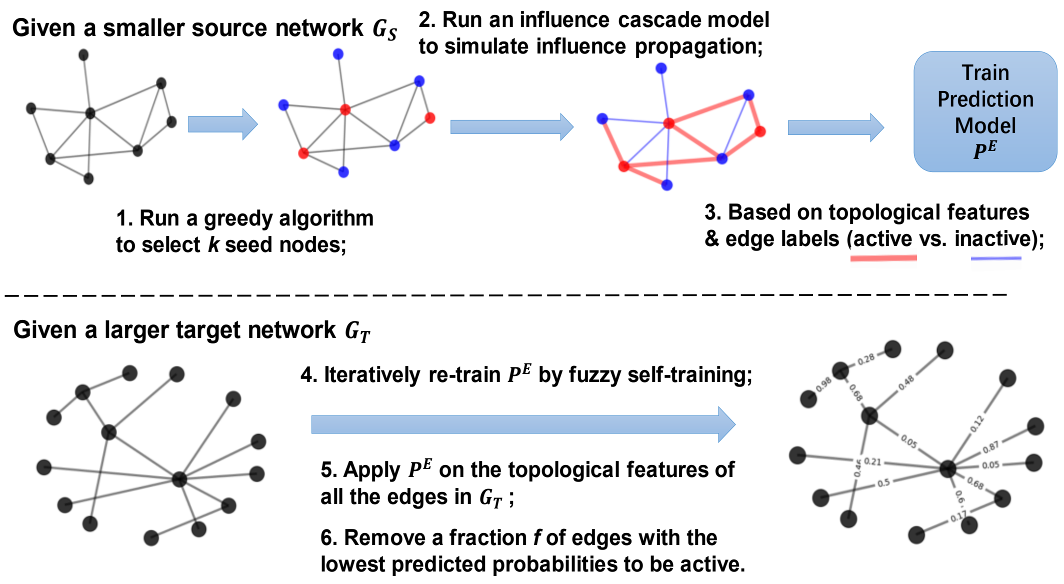

Given a smaller source network G_S
1. Run a greedy algorithm to select k seed nodes;
Train Prediction Model P^N
2. Based on topological features & node labels (seed vs. non-seed);
Given a larger target network G_T
3. Iteratively re-train P^N by fuzzy self-training;
4. Apply P^N on the topological features of all the nodes in G_T ;
5. Select top-k nodes with the highest predicted probabilities to be seed.

Figure 2.1: Illustration of cross-network seed selection in IM. The red nodes indicate seed nodes and the blue nodes represent non-seed nodes.

selection knowledge pre-learned from a source network. Figure 2.1 illustrates the main idea of the cross-network seed selection problem. Firstly, in a smaller source network G_S, one can run a standard IM greedy algorithm to select k seed nodes. Then, based on the topological structures and labels (i.e. seed or non-seed) of all the nodes in G_S, we can train a node prediction model P^N to learn what characterized nodes would be selected as seeds by the greedy algorithm. Then, the prediction model P^N is iteratively adapted to the larger target network G_T to heuristically select the nodes most likely to act as seeds for IM. Since the proposed CNL model learns the knowledge from a highly effective greedy algorithm in the source network, it is more reliable to achieve a high influence spread in the target network, as compared to the conventional IM heuristic algorithms without the help of greedy algorithms. In addition, since CNL heuristically selects seed nodes, it runs much faster than the greedy algorithms in the target network. Thus, the CNL model can achieve a good trade-off between efficiency and effectiveness of seed selection in IM.

On the other hand, to tackle large-scale IM problem, some studies proposed to employ graph sparsification as a pre-processing step, by removing a fraction of edges to make the original network become more concise and tractable for IM. Previous IM-based graph sparsification algorithms only leverage the information in a single network, either using an unsupervised approach [21], [63], or requiring a log of past influence propagation traces in the given network [64]. To the best of our knowledge, we are the first to leverage cross-network information to conduct graph sparsification for IM. Here, we consider graph sparsification as a cross-network edge prediction task, with the goal of removing the edges least likely to contribute to influence propagation in the target network, by leveraging the influence propagation knowledge pre-learned from the source network. As illustrated in Figure 2.2, we firstly simulate the influence propagation traces in a



Given a smaller source network G_S
2. Run an influence cascade model to simulate influence propagation;
Train Prediction Model P^E
1. Run a greedy algorithm to select k seed nodes;
3. Based on topological features & edge labels (active vs. inactive);
Given a larger target network G_T
4. Iteratively re-train P^E by fuzzy self-training;
5. Apply P^E on the topological features of all the edges in G_T ;
6. Remove a fraction f of edges with the lowest predicted probabilities to be active.

Figure 2.2: Illustration of cross-network graph sparsification in IM. The red nodes indicate seed nodes and the blue nodes represent non-seed nodes. The red lines indicate active edges during influence propagation simulations, and the blue lines represent inactive edges.

smaller source network G_S, by running an influence cascade model. After that, we label all the edges in G_S as either active or inactive, where active edges indicate that the influence has actually been propagated through them to maximize the influence during simulations. Then, based on the explicit topological features and labels of all the edges in G_S, we can train an edge prediction model P^E to learn what characterized edges would be helpful for influence propagation in IM. While those unsupervised graph sparsification algorithms [21], [63] without any labeled information would fail to do so. Next, we iteratively adapt the prediction model P^E to a larger target network G_T to predict the probability of each edge to be active for influence propagation. By removing the edges least likely to be active, we can make 1) existing IM greedy algorithm runs more efficiently; and 2) the loss of influence spread of the greedy algorithm as small as possible, in the sparse target networks.

The aforementioned cross-network seed selection and graph sparsification problem can be regarded as a domain adaptation task, which aims to transfer the knowledge pre-learned from a source domain to assist in solving the same task in a target domain, in the condition that the source domain and the target domain share an identical feature space but have different data distributions [14]. To address the domain discrepancy between different networks, we employ a self-training for domain adaptation (SEDA) algorithm [65] to iteratively retrain the prediction model by leveraging not only the fully labeled data in the source network, but also the most confident predictions in the target network. It is worth noting that the predictions generated by different self-training iterations are with different levels of confidence. Thus, directly utilizing the predicted binary labels to retrain the prediction model might cause negative effect on the prediction performance when the target network predictions become not confident enough. Fuzzy techniques are advantageous in capturing the imprecise, uncertain and vague information during knowledge transfer [66], [67], [68], [69]. Motivated by this, we propose a fuzzy self-training algorithm to assign fuzzy labels to the most confident predicted target network instances, when they are iteratively added to retrain the prediction model. With the provision of fuzzy labels, we can easily differentiate the confidence levels of the predictions generated by different self-training iterations during retraining, thus, the negative effects caused by such not confident enough predictions can be alleviated. The contributions of this work can be summarized as follows:

1) We propose a CNL model to study two issues of the cross-network IM problem, namely seed selection and graph sparsification, by viewing them as a cross-network node prediction task and a cross-network edge prediction task, respectively;
2) For seed selection, CNL leverages the greedy seed selection knowledge pre-learned from a smaller source network, to heuristically select top-*k* nodes most likely to act as seeds for IM in multiple larger target networks;

3) For graph sparsification, CNL leverages the influence propagation knowledge previously acquired in a smaller source network to remove a fraction of edges least useful for influence propagation in multiple larger target networks;
4) To address domain discrepancy, a fuzzy self-training approach is proposed to iteratively adapt the prediction model to the target network, by utilizing fuzzy labels to capture prediction uncertainty;
5) Extensive experiments in the real-world public datasets demonstrate that CNL can achieve a good trade-off between efficiency and effectiveness of IM.

The rest of this chapter is organized as follows. Section 2.2 introduces the related work about the IM algorithms and IM-based graph sparsification approaches. Section 2.3 formulates the cross-network seed selection and cross-network graph sparsification problem, respectively. Section 2.4 presents the proposed CNL model. Section 2.5 discusses the experimental results in the real-world datasets. Section 2.6 summaries this work.

2.2 Related Work

In this section, we review the standard IM algorithms and the graph sparsification algorithms developed for IM.

2.2.1 Influence Maximization

Domingos *et al.* [70] are the first to study influence propagation in a social network using a probabilistic algorithm. Then, Kempe *et al.* [50] proposed two pioneering influence cascade models, namely Independent Cascade (IC) model and Linear Threshold (LT) model. In an influence cascade model, each node is associated with a status at a certain time, either active or inactive. In IC model, each edge is associated with an influence probability p, which is set to be a constant. Then, an inactive node becomes active if it is successfully influenced by any of its active neighbors independently. While in LT model, an inactive node would become active if the sum of the influence probabilities from all of its active neighbors exceeds a given threshold. Given a certain influence cascade model, the influence is propagated from seed nodes to all the other nodes in the network. The goal of the IM problem is to maximize the influence spread in the given network with the constraint that the seed set size is fixed to be a small value, say k.

Existing algorithms to address the IM problem can be grouped into two families, namely greedy and heuristic methods. Originally, Kempe *et al.* [50] proposed a greedy hill-climbing approach to iteratively add a new node to the seed set which provides the largest marginal gain on the influence spread. This greedy algorithm can guarantee a $(1 - 1/e)$ approximation of the optimal solution but requires a rather high computation cost. To improve the efficiency of the general greedy algorithm, the CELF [57] and CELF++ [58] greedy algorithms have been proposed to reduce the number of evaluations on the influence spread estimation, by exploiting the sub-modularity property of the influence spread function. In addition, Chen *et al.* [59] proposed a NewGreedyIC algorithm to employ a breadth first search (BFS) on a deterministic graph which is converted from the influence probabilistic graph, to calculate the influence spread. To efficiently estimate influence spread, a pruned BFS method [71] and a static greedy algorithm [72] were proposed to reduce the number of Monte-Carlo simulations. Also, Wang *et al.* [73] developed a community based greedy algorithm to select the most influential nodes within each community rather than in the whole network. Instead of utilizing Monte-Carlo simulations to estimate the influence spread, some recent work [74], [75], [76] utilize the reverse influence sampling (RIS) method to select seed nodes based on reverse reachability tests. The idea of RIS [74] is to randomly sample a collection of reserve reachable (RR) sets from the given network and then select the set of nodes which can cover the maximum number of RR sets as seeds. Although such sampling can guarantee up to a $(1 - 1/e - \varepsilon)$ approximation, [77] has shown that these sampling methods would incur a high memory overhead in practice. Besides greedy algorithms, some heuristic algorithms are also developed to tackle the IM problem. Degree and centrality-based heuristics are the common metrics to estimate the influence of nodes in a network. To improve the pure degree heuristic, Chen *et al.* [59] proposed a DegreeDiscountIC algorithm to discount the degree contributed by the nodes already in the seed set. Luo *et al.* [62] proposed a PageRank-based heuristic which greedily selects seed nodes only from the nodes with high PageRank scores. Chen *et al.* [60] developed a PMIA algorithm to approximate the influence spread based on the maximum influence paths. Then, an IRIE algorithm [61] and an IPA algorithm [78] were proposed to reduce the high memory overhead incurred by PMIA. Besides, Tang *et al.* [77] designed a hop-based influence estimation algorithm to compute the influence spread up to two hops, the idea is similar to the time-constrained IM problem [79].

Although the IM problem has been extensively studied in the literature, very little work focuses on the cross-network IM problem. In [80], [81], the influence propagation across multiple aligned social networks has been studied, where some common users must be shared by different networks. Besides, Hu *et al.* [20] studies the cross-network IM problem in a more generalized scenario where the source network and the target network do not share any common users. A Transfer Influence Learning (TIL) method was proposed to transfer the influence across multiple networks, by viewing seed selection for IM as a node classification task. However, this TIL method does not consider domain discrepancy between different networks.

2.2.2 Graph Sparsification

Many real-world networks are with massive number of nodes and edges, hampering some promising IM greedy algorithms to work in practice. To tackle the large-scale IM problem, one can construct a more succinct representation of the original network by retaining fewer nodes or edges. Graph sparsification is one technique to construct a sparse network by retaining all the nodes in the original network, while removing a fraction of edges. Several graph sparsification algorithms have been developed as a pre-processing step for IM. For example, Wilder *et al.* [63] developed a Random Walk algorithm to preserve a subset of edges by minimizing the Kullback-Leibler divergence [82] between a random walk on the original network and the sparse network. Mathioudakis *et al.* [64] proposed a SPINE algorithm to detect the "backbone" of an influence network, by preserving the edges most important for influence propagation. However, the SPINE algorithm requires the input of not only the topology structure, but also a log of past influence propagation traces in the given network, which are impracticable to obtain in most real-world applications. Purohit *et al.* [83] proposed a COARSENET algorithm to merge a fraction of adjacent node pairs, by minimizing the difference of the first eigenvalue of the adjacency matrix between the original network and the coarsened network. In addition, Lamba *et al.* [21] proposed a model independent approach to remove the least informative edges, according to an overall ranking weighted by several topological features. To aggregate multiple feature rankings, they measure the Kendall Tau distances [84] between different rankings, and then assign higher weight to more unique feature ranking.

2.3 Problem Formulation

In order to achieve a good trade-off between efficiency and effectiveness of IM, we propose to leverage the cross-network information to address the seed selection and graph sparsification tasks in IM. Let $G_S = (V_S, E_S)$ be a smaller source network with a set of nodes V_S and a set of edges E_S, and $G_T = (V_T, E_T)$ be a larger target network with a set of nodes V_T and a set of edges E_T. Next, we formulate the cross-network seed selection and cross-network graph sparsification problem, respectively.

2.3.1 Cross-network Seed Selection in IM

We consider cross-network seed selection as a cross-network node prediction task, with the goal of selecting the nodes most likely to act as seed in G_T by leveraging the greedy seed selection knowledge pre-learned in G_S.

Firstly, in G_S, we run an IM greedy algorithm to select k seed nodes multiple times. Then, a node $v_S^i \in V_S$ is labeled as seed if it is selected by the greedy algorithm at least one time; otherwise, v_S^i is labeled as non-seed. Next, based on $D_S = \{(x_S^i, y_S^i)\}_{i=1}^{|V_S|}$, where x_S^i and y_S^i represent the set of explicit topological features and the label of node v_S^i, we can train a node prediction model P^N.

In G_T, we compute the same set of topological features (as in G_S) for all the nodes, i.e., $D_T = \{x_T^i\}_{i=1}^{|V_T|}$. Then, we iteratively retrain P^N by the proposed CNL algorithm. After t iterations, we apply P^N on D_T to predict $\{\hat{y}_T^i\}_{i=1}^{|V_T|}$, where $\hat{y}_T^i$ denotes the predicted probability of v_T^i to be labeled as seed. Then, we rank $\{\hat{y}_T^i\}_{i=1}^{|V_T|}$ and heuristically select top-k nodes with the highest predicted probabilities as seeds, denoted as $\hat{A}$. In addition, we run the same greedy algorithm (as in G_S) to select a set of k seed nodes in G_T, denoted as A, which is treated as the ground-truth seed selection result. The aim of cross-network seed selection is to make $\hat{A}$ as similar as possible to A such that the influence spread achieved by $\hat{A}$ can be approximately maximized in G_T.

2.3.2 Cross-network Graph Sparsification in IM

We consider cross-network graph sparsification as a cross-network edge prediction task. Here, the goal is to remove the edges least likely to contribute to influence propagation in G_T, by leveraging the influence propagation knowledge pre-learned in G_S.

In G_S, we firstly run an IM greedy algorithm to select k seed nodes. Then, we run an influence cascade model multiple times to simulate the influence propagation traces induced by the k seed nodes. In an influence cascade model [50], [85], the influence is firstly propagated from the seed nodes to their inactive neighbors. Then, if a neighbor has been successfully influenced to become active, it can further influence its inactive neighbors with a specific probability. After simulations, we can label all the edges in G_S as either active or inactive as follows:

In an undirected network, an edge e^{ij} is labeled as active, if during at least one time of influence propagation simulation, node v^i successfully influences node v^j or node v^j successfully influences node v^i; otherwise, e^{ij} is labeled as inactive. In a directed network, an edge e^{ij} is denoted as active, iff node v^i successfully influences node v^j, during at least one time of influence propagation simulation; otherwise, e^{ij} is labeled as inactive.

Next, based on $D_S = \{(x_S^{ij}, y_S^{ij})\}_{ij=1}^{|E_S|}$, where x_S^{ij} and y_S^{ij} represent the set of explicit topological features and the label of edge e_S^{ij}, we can train an edge prediction model P^E. In G_T, we define the same set of topological features for all the edges, i.e., $D_T = \{x_T^{ij}\}_{ij=1}^{|E_T|}$. Then, we iteratively retrain P^E by the CNL algorithm for t iterations. Next, we apply P^E on D_T to predict $\{\hat{y}_T^{ij}\}_{ij=1}^{|E_T|}$, where $\hat{y}_T^{ij}$ denotes the predicted probability of e_T^{ij} to be labeled as active. Finally, we rank $\{\hat{y}_T^{ij}\}_{ij=1}^{|E_T|}$ and remove a fraction f of the edges with the least predicted probabilities to be active for influence propagation, denoted as $\hat{I}$. In addition, we define the ground-truth labels for all the edges in G_T, via the same approach as in G_S, and denote the set of ground-truth inactive edges in G_T as I. The goal of cross-network graph sparsification in IM is to make all the edges in $\hat{I}$ are indeed inactive, i.e., belonging to I. Thus, we would only remove the edges inactive for influence propagation in the target network, which makes the loss of influence spread as small as possible in the sparse target network.

It is worth noting that in the proposed CNL model, it is flexible to learn the greedy seed selection and influence propagation knowledge from any IM greedy algorithms and any influence cascade models in G_S. But the greedy algorithm and the influence cascade model employed to define the ground-truth labels in G_T should be the same as in G_S.

2.4 The Proposed Algorithm

In this section, we briefly introduce several topological features adopted in the prediction model and then present the detailed framework of the CNL model.

2.4.1 Explicit Topological Features

As shown in the literature [21], [20], [86], the following topological features can reflect the influence of a node in a network.

1) **Degree**. It calculates the number of edges adjacent to a node.

2) **Weighted Degree**. Different from degree, it calculates the number of edges

adjacent to a node by taking the weight of each edge into consideration.

3) **Eigenvector Centrality**. It evaluates a node's influence in the scenario of information diffusion. A node with high eigenvector centrality indicates it is highly influential to spread the influence in the network [87].
4) **HITS Hub**. HITS algorithm [88] computes two values for each node, namely HITS authority and HITS hub. The authority and hub values of a node are estimated based on the incoming links and the outgoing links from the node, respectively.
5) **PageRank Score**. PageRank algorithm [89] was originally designed to rank page authority. By viewing each node as a page, it can be used to compute the ranking of nodes based on the structure of the incoming links to the nodes.
6) **Clustering Coefficient**. It reflects the fraction of a node's friends who are also friends with each other. A node with high clustering coefficient indicates that the node's neighbors are likely to be connected with each other.

On one hand, for seed node prediction, we employ the aforementioned topological features as node features. On the other hand, for inactive edge prediction, we assume that the likelihood of an edge to be active for influence propagation depends on the influence of two nodes on the edge. Thus, we construct edge features based on the average topological feature values of the two nodes on each edge. To make the feature values network independent, we rank the absolute values of each feature ascendingly and map them into [0, 1]. All these selected features can be efficiently measured by NetworkX[1], thus making the proposed CNL model more efficient for the IM task, as compared to the standard greedy algorithms. In addition, it is flexible to employ other informative topological features in the CNL model as long as they can effectively reflect a node's influence and also be efficiently computed in large-scale networks.

[1] http://networkx.github.io/

2.4.2 Cross-Network Learning (CNL) Model

We apply the CNL model to address both cross-network seed selection and cross-network graph sparsification problems. For seed selection, we treat seed nodes as positive class and non-seed nodes as negative class. For graph sparsification, we treat active edges as positive class while inactive edges as negative class. Then, with positive instances labeled as "1" and negative instances labeled as "0", a supervised learning method can be devised to train a node or edge prediction model via the logistic regression (LR) algorithm as follows:

$$J(\theta) = -\frac{1}{m_S}\sum_{u=1}^{m_S}\left(y_S^u\, log\left(h_\theta(x_S^u)\right) + (1-y_S^u)\, log\left(1-h_\theta(x_S^u)\right)\right) \qquad (2.1)$$

where m_S denotes the number of training examples in D_S; $x_S^u = \{x_{S,j}^u\}_{j=1}^n$ is a feature vector representing the topological feature values of instance u in D_S, and n is the number of features; y_S^u is the label of instance u in D_S; $\theta = \{\theta_j\}_{j=1}^n$ is a weight vector denoting the importance degree of different typological features for the prediction task; $h_\theta(x_S^u) = P(y_S^u = 1|x_S^u;\theta)$ refers to the predicted probability of instance u to be labeled as positive. After $\theta^* = arg\min_\theta(2.1)$ has been learned in D_S, one can firstly leverage it to estimate the probability of an instance u to be positive in D_T, using a sigmoid function as:

$$\hat{y}_T^u = (1 + e^{-(\theta^*)^T x_T^u})^{-1} \qquad (2.2)$$

Due to its simplicity and high efficiency, LR was adopted as the prediction algorithm in this work. However, the proposed CNL model is flexible to work with other prediction algorithm, as long as it can work efficiently and also provide the probabilities for its predictions.

2.4.2.1 Feature Incompatibility

Due to domain discrepancy, an identical feature might have different importance degree for the same prediction task in different networks. To address this, we measure the incompatibility of each feature X_j between D_S and D_T, following the approach in [65] as below:

$$IC(X_j) = 1 - Pcc(X_{S,j}, Y_S)Pcc(X_{T,j}, \hat{Y}_T) \tag{2.3}$$

where $Pcc(X_{S,j}, Y_S)$ represents the Pearson correlation coefficient (PCC) between the value of the j-th feature and the label of all the instances in D_S; $Pcc(X_{T,j}, \hat{Y}_T)$ indicates the PCC between the value of the j-th feature and the predicted probability of all the instances to be positive in D_T. The smaller the incompatibility, the more similarly the feature performs between the source network and the target network. Based on the feature incompatibility measure, a regularization term is defined as:

$$R_1(\theta) = \sum_{j=1}^{n} IC(X_j)\,|\theta_j| \tag{2.4}$$

By minimizing $R_1(\theta)$, lower absolute weights would be assigned to the features with larger incompatibility (i.e. perform more differently between D_S and D_T). In addition, a L_2 regularization is defined to prevent overfitting as below:

$$R_2(\theta) = \frac{1}{2}\sum_{j=1}^{n} {\theta_j}^2 \tag{2.5}$$

By integrating the regularization terms (2.4), (2.5) and the cost function (2.1), an overall loss function is developed as:

$$L(\theta) = J(\theta) + \frac{\lambda_1}{m_S}R_1(\theta) + \frac{\lambda_2}{m_S}R_2(\theta) \tag{2.6}$$

where $\lambda_1, \lambda_2 \geq 0$ are the trade-off parameters to balance the effects of the regularizations (2.4) and (2.5). Next, gradient descent algorithm is employed to find the parameters minimizing the overall loss function (2.6), as follows:

$$\frac{\partial L(\theta)}{\partial \theta_j} = \frac{1}{m_S}\begin{cases}\sum_{u=1}^{m_S}[(h_\theta(x_S^u) - y_S^u)x_{S,j}^u + \lambda_1 IC(X_j) + \lambda_2\theta_j], if\,\theta_j \geq 0\\ \sum_{u=1}^{m_S}[(h_\theta(x_S^u) - y_S^u)x_{S,j}^u - \lambda_1 IC(X_j) + \lambda_2\theta_j], if\,\theta_j < 0\end{cases} \tag{2.7}$$

$$\theta_j = \theta_j - \alpha \frac{\partial L(\theta)}{\partial \theta_j} \tag{2.8}$$

where $\alpha > 0$ denotes the learning rate.

2.4.2.2 Iterative Self-Training

So far, we have considered feature incompatibility between D_S and D_T. However, the training data are merely obtained from D_S. To make the prediction model also consider the training data from D_T, one can employ a SEDA algorithm [65] to leverage not only the fully labeled data from the source network but also the unlabeled data from the target network to iteratively retrain the prediction model. Specifically, at each self-training iteration, the top-c most confident predicted positive samples with the pseudo binary label "1" and the top-c most confident predicted negative samples with the pseudo binary label "0", are moved from D_T to D_S to retrain the prediction model in the next self-training iteration. However, in the IM application, the number of seed nodes should be much smaller than that of non-seed nodes, and the number of active edges is generally smaller than that of inactive edges. To tackle such imbalanced data, in the CNL model, we propose to move the top-c most confident predicted positive instances, while the top-c' most confident predicted negative instances from D_T to D_S at each self-training iteration, where $c' = c * l$ and $l > 1$ denotes the ratio of the number of most confident predicted negative instances over that of positive instances moved from D_T to D_S.

2.4.2.3 Fuzzy Labels

In the SEDA algorithm [65], the most confident predicted positive (or negative) instances generated by different self-training iterations are actually not with equal likelihood to be positive (or negative). Thus, assigning predicted binary labels to them would fail to differentiate the degree of their membership to be positive (or negative). Moreover, when the self-training iteration becomes large or too large, the most confident predictions in the target network might become not confident enough. In such a case, if some wrongly predicted labeled instances in the target network are employed to retrain the prediction model, the prediction performance will be degraded. Fuzzy techniques have demonstrated high effectiveness to handle imprecision, uncertainty and vagueness during knowledge transfer [66], [90]. Motivated by this, we exploit the use of fuzzy labels to alleviate the weakness of the SEDA algorithm, by differentiating the confidence levels of the predictions generated by different self-training iterations. Here, the fuzzy labels represent the degree of the membership of the instances predicted as positive. Note that in the SEDA algorithm [65], when the most confident predictions in the target network are iteratively added to D_S, they are also simultaneously removed from D_T. Intuitively, as the self-training iteration increases, the most confident predictions in D_T will become less confident. Thus, we would assign higher membership of positive to the most confident predicted positive instances, while lower membership of positive to the most confident predicted negative instances, generated by the earlier self-training iterations. Specifically, at the i-th iteration, the fuzzy labels assigned to the top-c most confident predicted positive instances are defined as:

$$1-\frac{1-\alpha}{t-1}(i-1) \tag{2.9}$$

where t indicates the total number of self-training iterations and $1 \leq i \leq t, t > 1$; $0.5 < \alpha < 1$ denotes the fuzzy label (i.e. predicted membership to be positive) assigned to the top-c most confident predicted positive instances, generated by the last (i.e. t-th) self-training iteration. Note that if $\alpha = 1$, the fuzzy labels are equivalent to binary labels; and if $t = 1$, we do not assign fuzzy labels. On the other hand, the fuzzy labels assigned to the top-c' most confident predicted negative instances, generated at the i-th self-training iteration are defined as:

$$\frac{1-\alpha}{t-1}(i-1) \tag{2.10}$$

For simplicity, we only differentiate the confidence levels of the predictions generated by different self-training iterations. While we treat all the top-c most confident positive instances (and all the top-c' most confident negative instances) generated by the same self-training iteration as equally confident.

Algorithm 2.1: Cross-network Learning (CNL)

Input: Source network $D_S = \{(x_S^u, y_S^u)\}_{u=1}^{m_S}$ with m_S labeled instances; Target network $D_T = \{x_T^u\}_{u=1}^{m_T}$ with m_T unlabeled instances; Number of self-training iterations: t; Number of most confident predicted positive instances moved from D_T to D_S at each iteration: c; Ratio of the number of most confident predicted negative instances over that of positive instances moved from D_T to D_S: l.

```
1. $D'_T = D_T$;
2. On $D_S$, train a model to obtain $\theta^* = arg \min_{\theta}(2.1)$;
3. For $i$=1:$t$ iterations, do:
    3.1 Based on $\theta^*$, apply (2.2) on $D_T$ to predict $\{\hat{y}_T^u\}_{u=1}^{m_T}$;
    3.2 Based on $\{(x_S^u, y_S^u)\}_{u=1}^{m_S}$ and $\{(x_T^u, \hat{y}_T^u)\}_{u=1}^{m_T}$, measure feature incompatibility
        between $D_S$ and $D_T$, via (2.3);
    3.3 Rank $\{\hat{y}_T^u\}_{u=1}^{m_T}$ and move the top-$c$ most confident predicted positive
        instances with their fuzzy labels from $D_T$ to $D_S$:
        $D_S := D_S + \left\{\left(x_T^u, 1 - \frac{1-\alpha}{t-1}(i-1)\right) | \hat{y}_T^u \in Top_{\ c}\ highest\ \{\hat{y}_T^u\}_{u=1}^{m_T}\right\}$;
        $D_T := D_T - \left\{\left(x_T^u, 1 - \frac{1-\alpha}{t-1}(i-1)\right) | \hat{y}_T^u \in Top_{\ c}\ highest\ \{\hat{y}_T^u\}_{u=1}^{m_T}\right\}$;
    3.4 Rank $\{\hat{y}_T^u\}_{u=1}^{m_T}$ and move the top-$(c * l)$ most confident predicted negative
        instances with their fuzzy labels from $D_T$ to $D_S$:
        $D_S := D_S + \left\{\left(x_T^u, \frac{1-\alpha}{t-1}(i-1)\right) | \hat{y}_T^u \in Top_{\ (c*l)}\ lowest\ \{\hat{y}_T^u\}_{u=1}^{m_T}\right\}$;
        $D_T := D_T - \left\{\left(x_T^u, \frac{1-\alpha}{t-1}(i-1)\right) | \hat{y}_T^u \in Top_{\ (c*l)}\ lowest\ \{\hat{y}_T^u\}_{u=1}^{m_T}\right\}$;
    3.5 On new $D_S$, retrain the model to obtain updated $\theta^* = arg \min_{\theta}(2.6)$;
   End for
4. Based on $\theta^*$, apply (2.2) on $D'_T$ to predict $\{\hat{y}_T^u\}_{u=1}^{m_T}$.
```

Output: Predicted probabilities of all the instances on D_T to be positive: $\{\hat{y}_T^u\}_{u=1}^{m_T}$.

Next, by iteratively moving the most confident predicted instances (i.e. both feature vectors and predicted fuzzy labels) from D_T to D_S, the prediction model can be updated based on not only all the fully labeled data in the source network, but also the newly added most confident predicted labeled data in the target network. In addition, with the devised fuzzy labels, lower degree of membership would be assigned to less confident predicted instances. Thus, we can lower the negative effects caused by adding the less confident predicted target network instances into the training set. Finally, after *t* self-training iterations, for cross-network seed selection, we employ the latest trained node prediction model to select the top-*k* nodes with the highest predicted probability to be positive (i.e. act as seed for IM) in the target network. On the other hand, for cross-network graph sparsification, we leverage the latest trained edge prediction model to remove a fraction *f* of the edges predicted as least likely to be positive (i.e. active for influence propagation in IM) in the target network.

2.5 Experiments

2.5.1 Datasets

The performance of the proposed CNL model was tested for both cross-network seed selection and cross-network graph sparsification tasks on four public real-world datasets, namely, NetHEPT[2], Email-Enron[3], Epinions[4] and DBLP[5]. These datasets were frequently employed to evaluate the IM performance in the literature [21], [58], [59], [63]. Both NetHEPT [59] and DBLP [91] datasets are collaboration networks, where each node represents an author and each link connecting two nodes indicates the co-author relationship. The Email-Enron

[2] https://www.microsoft.com/en-us/research/people/weic/
[3] https://snap.stanford.edu/data/email-Enron.html
[4] https://snap.stanford.edu/data/soc-Epinions1.html
[5] https://snap.stanford.edu/data/com-DBLP.html

dataset [92] is an email communication network, where each node represents an email address and each link connecting two nodes indicates the existence of communication between them. The Epinions dataset [93] is a "who trust whom" online social network generated from the Epinions site. Table 2.1 gives some statistics of these datasets. To demonstrate the efficiency of the CNL model, we employed the smallest NetHEPT network as the source network, while the other three larger networks as the target networks.

Table 2.1: Statistics of real-world datasets.

Dataset	Type	Source/Target	# Nodes	# Edges
NetHEPT	Undirected	Source	15233	31398
Email-Enron	Undirected	Target	36692	183831
Epinions	Directed		75879	508837
DBLP	Undirected		317080	1049866

2.5.2 Implementation Details

In the experiments, the IC model [50] with the influence probability p=0.01 was employed as the influence propagation model in the IM task. On one hand, for cross-network seed selection, the NewGreedyIC algorithm [59] was employed to select 100 seed nodes in the source network over 10 times to learn node labels. Next, in the CNL model, we set the weight of the feature incompatibility regularization as $\lambda_1 = 10$ and divided it by a factor of 1.1 after each self-training iteration, following the practice in [65]; and set the L_2-regularization weight as $\lambda_2 = 0.5$. For self-training process, we set t=5, c=10, l=30 for all the datasets. It means that at each of the 5 self-training iterations, the top-10 most confident predicted seed nodes and top-300 most confident predicted non-seed nodes in the target network would be iteratively moved to the training set to retrain the node prediction model in the next self-training iteration. In addition, for fuzzy label design, we set $\alpha = 0.8$, which indicates that at the last (i.e. 5-th) self-training iteration, the top-10 most confident predicted seed nodes are with 80% of membership to be seed, while the top-300 most confident predicted non-seed nodes are with 80% of membership to be non-seed.

On the other hand, for cross-network graph sparsification, in the source network, we run the IC model [50] 1000 times to simulate the influence propagation traces induced by 50 seed nodes selected by NewGreedyIC [59] so as to learn the edge labels. Then, in the CNL model, λ_1 was set with the same value as that for cross-network seed selection. While $\lambda_2 = 1$, 0.1 and 1 were experimentally adopted for the Email-Enron, Epinions and DBLP target network, respectively. For self-training process, we set t=3, c=200, l=30, meaning that during each of 3 self-training iterations, the top-200 most confident predicted active edges and top-6000 most confident predicted inactive edges in the target network would be moved to the training set to iteratively update the edge prediction model. For fuzzy label design, we conducted a grid search on $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ and consequently set α=0.7, 0.7 and 0.8 for the Email-Enron, Epinions and DBLP target networks, respectively.

2.5.3 Performance of CNL for Seed Selection

In this subsection, we report the performance of the proposed CNL model for seed selection in three target networks.

2.5.3.1 Baselines

The following IM algorithms are benchmarked against the proposed CNL model.

1) **NewGreedyIC** [59]: It is a greedy IM algorithm which firstly converts the influence probabilistic graph into a deterministic graph and then employs a BFS on the deterministic graph to calculate the influence spread;
2) **CELF** [57]: It is a greedy IM algorithm which greatly reduces the number of evaluations on influence spread by exploiting the sub-modularity property;

3) **CELF++** [58]: It further exploits the sub-modularity property to avoid unnecessary re-computations of marginal gains incurred by CELF;
4) **EigenCen**: It is a heuristic method based on Eigenvector Centrality which reflects a node's influence in information diffusion [87]. It heuristically selects nodes with the highest eigenvector centrality to be seeds;
5) **TIL** [20]: It is most related to the proposed CNL model, which views seed selection for IM as a cross-network node classification task. However, it ignores domain discrepancy between the source network and the target network.

2.5.3.2 Evaluation Metrics

For each of the compared IM algorithms, the same number of seed nodes from [10, 100] was assigned for each target network. We evaluate the cross-network seed selection performance from two perspectives, i.e., the performance w.r.t. IM in the target network and the performance w.r.t. cross-network node retrieval. Firstly, to evaluate the IM performance in the target networks, we adopt two metrics as in the IM literature [50], [57], [59], namely, influence spread and running time. The higher the influence spread, the better the performance; while the shorter the running time, the better the performance. Secondly, by considering seed selection as a node prediction task, we let both TIL [20] and the proposed CNL model learn the greedy seed selection knowledge from the same greedy algorithm (i.e. NewGreedyIC [59] in the experiments) in the source network. Thus, in the target networks, we should check whether TIL and CNL could obtain similar seed selection results w.r.t. the greedy algorithm which they learned the knowledge from. To evaluate it, we adopted the precision@k metric to measure the accuracy of their top-k seed node retrieval results. Here, the ground-truth seed nodes in the target networks should be the k seed nodes selected by the same greedy algorithm (i.e. NewGreedyIC) as in the source network. The higher the seed node precision, the better the performance.

2.5.3.3 Performance Analysis

Next, we report the performance of CNL when it is applied to three target networks for seed selection. Firstly, as shown in Figure 2.3, CNL can always achieve a good influence spread almost matched with NewGreedyIC in the three target networks. In addition, as shown in Figure 2.3(a), in the Email-Enron target network, both CNL and NewGreedyIC achieved lower influence spread than CELF and CELF++ when selecting no more than 40 seed nodes; while achieving higher influence spread than CELF and CELF++ when selecting at least 60 seed nodes. In the Epinions target network, as shown in Figure 2.3(b), both CNL and NewGreedyIC achieved lower influence spread than CELF and CELF++ when the seed set size was smaller than 50, while achieving higher influence spread than CELF and CELF++ if at least 50 seed nodes were selected. While in the DBLP target network, as shown in Figure 2.3(c), both CNL and NewGreedyIC achieved higher influence spread than CELF and CELF++ for any seed set sizes within [10, 100]. These results could be explained by the fact that in the experiments, CNL learned the greedy seed selection knowledge from NewGreedyIC in the source network, thus CNL would tend to match with the influence spread of NewGreedyIC, rather than other greedy algorithms, such as CELF and CELF++. On the other hand, as shown in Figure 2.3, the running time of CNL was much shorter than all the greedy algorithms in all the three target networks. For example, when selecting 100 seed nodes in the largest DBLP target network containing millions of edges, the running time of NewGreedyIC, CELF and CELF++ was about 36, 11, and 9 hours, respectively. While in the CNL model, it only took 4 minutes to measure all the topological features for all the nodes in the network and another 4 minutes to train the prediction model via 5 self-training iterations. Since

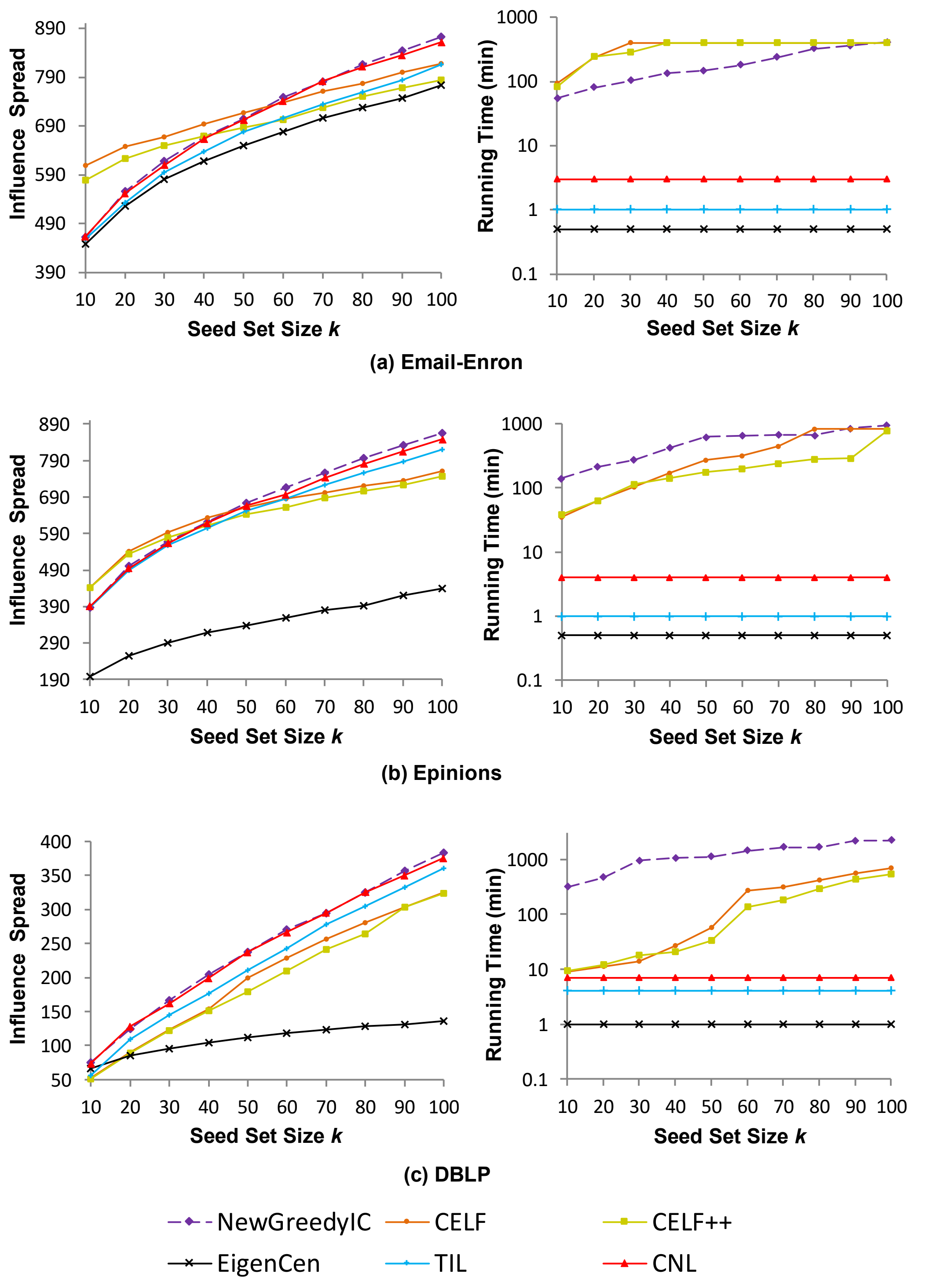


Figure 2.3: Performance of CNL for seed selection for IM in three target networks. The higher the influence spread, the better the performance; while the shorter the running time, the better the performance.

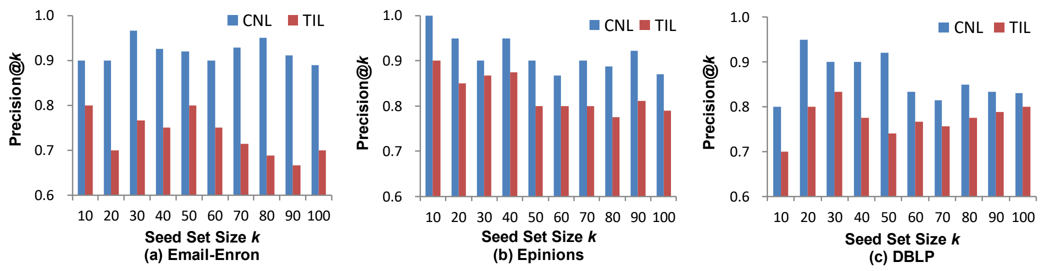


Figure 2.4: Performance of CNL for seed node prediction in three target networks, in terms of the seed node precision at top-*k* retrieved most likely seed nodes. The higher the seed node precision, the better the performance.

the selected topological features can be measured efficiently and the time taken to train the prediction model via self-training was quite short, CNL can run much more efficiently than the greedy algorithms. In addition, as shown in Figure 2.3, although the EigenCen heuristic was fastest among all the comparing algorithms, it failed to achieve a high influence spread in all the three target networks. Also, we can see that although the greedy algorithms can achieve a high influence spread, the running time was extremely long. In contrast, the proposed CNL model can obtain a good trade-off between the efficiency and effectiveness of the IM problem, i.e., greatly saving the running time while still achieving a good influence spread almost matched with the greedy algorithm.

Next, we compare the performance of TIL and the proposed CNL model. As shown in Figures 2.3(a) and 2.3(c), in the Email-Enron and DBLP target networks, CNL always achieved higher influence spread than TIL for different seed set sizes between [10, 100]. In the Epinions target network, as shown in Figure 2.3(b), CNL achieved higher influence spread than TIL when selecting more than 30 seed nodes. In addition, as shown in Figure 2.4, CNL always achieved much higher seed node precisions than TIL, when retrieving any number of *k* seed nodes between [10, 100] in all the three target networks. Both the higher influence spread and the higher precisions of CNL over TIL demonstrates the importance and necessity of addressing domain discrepancy between the source network and the target network, for cross-network seed selection in IM.

Moreover, we observe an interesting phenomenon of IM in different types of networks. As shown in Figure 2.3, when the same number of seed nodes, say 10, was selected, the influence spread achieved by CNL in the DBLP collaboration network was only 74; in the Epinion trust social network, it was 390; while in the email communication network, it was 463. These reveal that the email communication and online social networks are much easier to spread the information effectively than the collaboration network.

2.5.4 Performance of CNL for Graph Sparsification

Next, we investigate the performance of the proposed CNL model for graph sparsification in three target networks.

2.5.4.1 Baselines

We compete with the following graph sparsification algorithms.

1) **Random Heuristic**: It randomly selects a fraction of edges to remove;
2) **RandomWalk** [63]: It removes a fraction of edges such that the Kullback-Leibler divergence between a random walk on the original network and the sparse network can be minimized;
3) **AggRanks** [21]: It removes a fraction of edges according to an overall ranking aggregated by multiple topological feature rankings. It assigns higher weights to more unique feature rankings during aggregation.

2.5.4.2 Evaluation Metrics

For each of the compared graph sparsification algorithms, a fraction *f* of edges chosen from [10%, 90%] were removed in the original network to extract the sparse networks. Then, in both the original network and the sparse networks, the NewGreedyIC algorithm [59] was employed to select the same number of (i.e. 50

in the experiments) seed nodes to maximize the influence. Next, we evaluate the performance of the graph sparsification algorithms from two perspectives, namely the performance w.r.t. IM and w.r.t. cross-network edge retrieval. To evaluate the performance of graph sparsification as a pre-processing step for IM, we adopt two metrics as in the related literature [9], [21], [63]. Firstly, we measure how much influence spread will be lost in the sparse network, as compared to that in the original network. The less the loss of influence spread, the better the performance. In addition, we check how much running time of the greedy algorithm (i.e. NewGreedyIC) could be saved in the sparse network, as compared to that taken in the original network. Note that the running time of NewGreedyIC [59] depends on the number of edges in the target network, rather than the topological structures of the edges. Thus, after removing an equal fraction of edges by different graph sparsification algorithms, the save of running time of NewGreedyIC would be the same for different graph sparsification algorithms.

On the other hand, by viewing graph sparsification as a cross-network edge retrieval task, both AggRanks [21] and the proposed CNL model aim to remove a fraction *f* of edges most likely to be inactive (i.e. useless) for influence propagation. Thus, we employ the precision@*f* metric to examine the accuracy of the top-*f* (%) most likely inactive edges retrieved by the graph sparsification algorithms. To learn the ground-truth edge labels in the target networks, we run the same influence cascade model as in the source network (i.e. IC model) to simulate the influence propagation traces. The detailed approach for edge label learning has been introduced in section 2.3.2. The higher the inactive edge precision, the better the performance.

2.5.4.3 Performance Analysis

Next, we report the performance of CNL for graph sparsification in the three target networks. Firstly, as shown in Figure 2.5, we can see that among all the compared graph sparsification algorithms, CNL always performed the best (i.e. achieved the lowest influence spread loss) in all the sparse target networks with different edge removal fractions between [10%, 90%]. In addition, if 40% of edges were

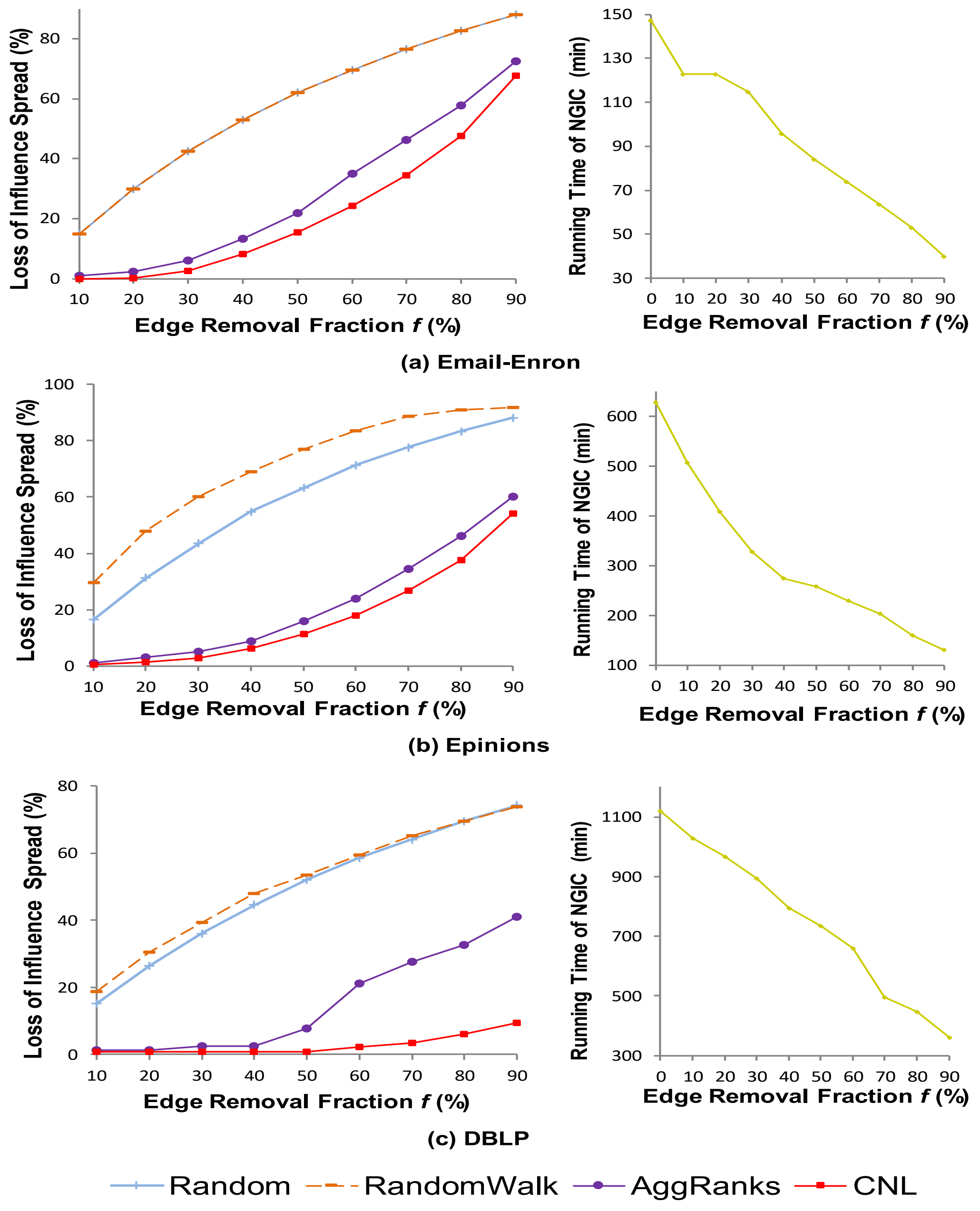


Figure 2.5: Performance of CNL for graph sparsification as a pre-processing step for IM in three target networks. The lower the loss of influence spread, the better the performance. (NGIC is short for NewGreedyIC)

removed by CNL in the Email-Enron, Epinions and DBLP target networks, the influence spread were just reduced by 8.07%, 6.37% and 0.84%, respectively, while the running time of NewGreedyIC can be greatly saved by 34.7%, 56.2% and 29.2% in return, respectively. In addition, as a graph sparsification algorithm for IM, the proposed CNL model is with high efficiency and scalability. For example, even in the largest DBLP target network, the time taken to measure all the topological features and train the prediction model via 3 self-training iterations was only about 15 minutes. These reveal that the proposed CNL model indeed acts as an effective graph sparsification algorithm for IM, since it can obtain a good trade-off between efficiency and effectiveness of IM, i.e., greatly speeding up the greedy algorithm without causing a notable loss of influence spread in the sparse networks.

Secondly, we look at the performance of the random heuristic and RandomWalk algorithms. As shown in Figure 2.5, in all the three target networks, both AggRanks and CNL can all achieve much lower influence spread loss than the random heuristic and RandomWalk algorithms. This could be explained by the fact that for the IM task, the influence tends to be propagated through those active edges which are adjacent to the highly influential nodes, rather than the randomly selected edges. Since the highly influential nodes are with discriminative

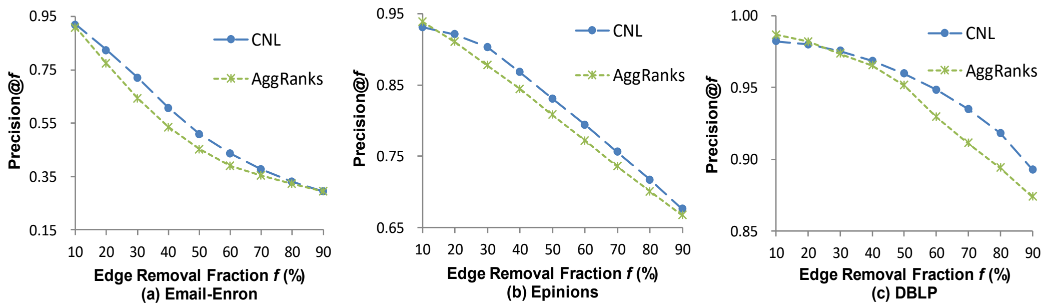


Figure 2.6: Performance of CNL for inactive edge prediction in three target networks, in terms of the inactive edge precision at top-*f(%)* retrieved most likely inactive edges. The higher the inactive edge precision, the better the performance.

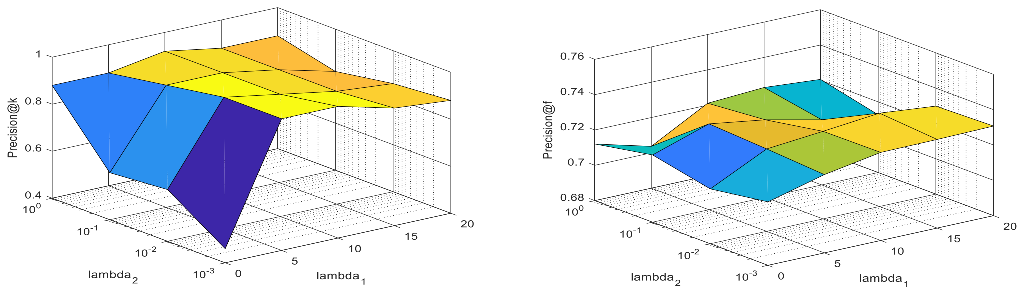
topological features w.r.t. randomly selected nodes. The active edges should also be with discriminative topological structures w.r.t inactive edges. Thus, the AggRanks and CNL algorithms based on the discriminative topological features can significantly outperform the random-based approaches.

Next, we compare the performance of AggRanks and the proposed CNL algorithms in the sparse target networks, in terms of the influence spread loss and inactive edge precision. As shown in Figure 2.5, when removing less than 30% edges in the target networks, these two algorithms would perform quite similarly, i.e., only lead to a little loss of influence spread in the sparse target networks. However, CNL can achieve lower influence spread loss than AggRanks, when the edge removal fraction is more than 30% in the three target networks. On the other hand, as shown in Figure 2.6(a), CNL achieved higher inactive edge precision than AggRanks in the sparse Email-Enron target networks with the edge removal fraction between [10%, 70%]. In addition, as shown in Figures 2.6(b) and 2.6(c), CNL achieved higher precision than AggRanks when the edge removal fraction is more than 10% and 20% in the Epinions and DBLP target networks, respectively. It is worth noting that for fair comparison, we let both AggRanks and CNL aggregate the same set of topological features to remove the edges least useful for influence propagation. But when learning the feature weightings for aggregation, AggRanks only leverages the topological information in a single network and assigns higher weight to more unique feature in an unsupervised manner. However, the more unique feature might not necessarily be more important for inactive edge prediction. In contrast to AggRanks, the proposed CNL model employ a fuzzy self-training approach to iteratively leverage the influence propagation knowledge pre-learned in a source network to learn the feature weightings for the target network. Thus, the better overall performance of CNL over AggRanks demonstrates the advantage of leveraging the cross-network information w.r.t. the single network information for inactive edge prediction in IM.

Finally, let us look at some interesting differences when graph sparsification is applied to different types of networks. As shown in Figure 2.5, when removing equal fraction of edges in the three target networks, the loss of influence spread is lowest in the DBLP target network. On the other hand, as shown in Figure 2.6, when giving the same edge removal fraction, the inactive edge precisions are highest in the DBLP target network. These results reveal that the email communication (i.e. Email-Enron) and trust social networks (i.e. Epinions) show greater challenge to graph sparsification than the collaboration network (i.e. DBLP). This could be explained by our previous observation in seed selection that the email communication and trust social networks are much easier to spread the information than the collaboration network. In other words, the fraction of active edges should be smallest (i.e., most of the edges are inactive for influence propagation) in the DBLP collaboration network. Thus, even though removing quite a large fraction (e.g. 70%) of edges in the DBLP network, the influence spread in the sparse network would not be substantially affected.

2.5.5 Parameter Sensitivity

In this subsection, we analyze the sensitivity of the parameters, i.e. $\lambda_1, \lambda_2, t, c, l, \alpha$ on the performance of CNL for seed node selection and inactive edge prediction. Specifically, for seed selection, the sensitivity tests were conducted in the DBLP target network when the seed set size is 50. For inactive edge prediction, the sensitivity analyses were performed in the Email-Enron target network when the edge removal fraction is 30%.



(a) Seed node prediction (b) Inactive edge prediction

Figure 2.7: Impact of λ_1 and λ_2 on the performance of CNL for seed node prediction in the DBLP target network when the seed set size is 50 and for inactive edge prediction in the Email-Enron target network when the edge removal fraction is 30%.

Parameter $\boldsymbol{\lambda_1}$ denotes the weight of the feature incompatibility regularization (2.4) and parameter $\boldsymbol{\lambda_2}$ indicates the weight of the L_2-norm regularization (2.5). As shown in Figure 2.7, $\lambda_1 > 0$ generally yields higher precisions for both seed node prediction and inactive edge prediction, as compared to $\lambda_1 = 0$. This demonstrates the effectiveness of incorporating (2.4) in CNL to assign lower absolute weights to the features which perform less similarly between the source network and the target network. In addition, as shown in Figure 2.7(a), when $\lambda_1 \in \{5, 10,15,20\}$, the seed node prediction performance of CNL is not sensitive to the value of λ_2. On the other hand, for inactive edge prediction, as shown in Figure 2.7(b), when $\lambda_1 = 10$, the performance of CNL is also not sensitive to the value of λ_2. However, when $\lambda_1 \in \{15, 20\}$, relatively smaller values of λ_2, i.e., 0.01 or 0.001, would lead to good performance of CNL for inactive edge prediction.

Parameter $\boldsymbol{t}$ denotes the total number of self-training iterations. As shown Figure 2.8(a), for seed node prediction, setting $t = 5$ leads to a significant improvement over $t = 1$ and 3. However, when $t > 5$, the seed node precision can only be slightly improved. In addition, note that the running time of CNL will increase as *t* increase. Thus, in order to achieve a good prediction performance and also save the running time, we fix *t*=5 for seed node prediction on all the datasets. On the other hand, as shown in Figure 2.9(a), *t*=3 can significantly increase the inactive edge precision w.r.t. t=1, however, when t>3, the inactive edge prediction

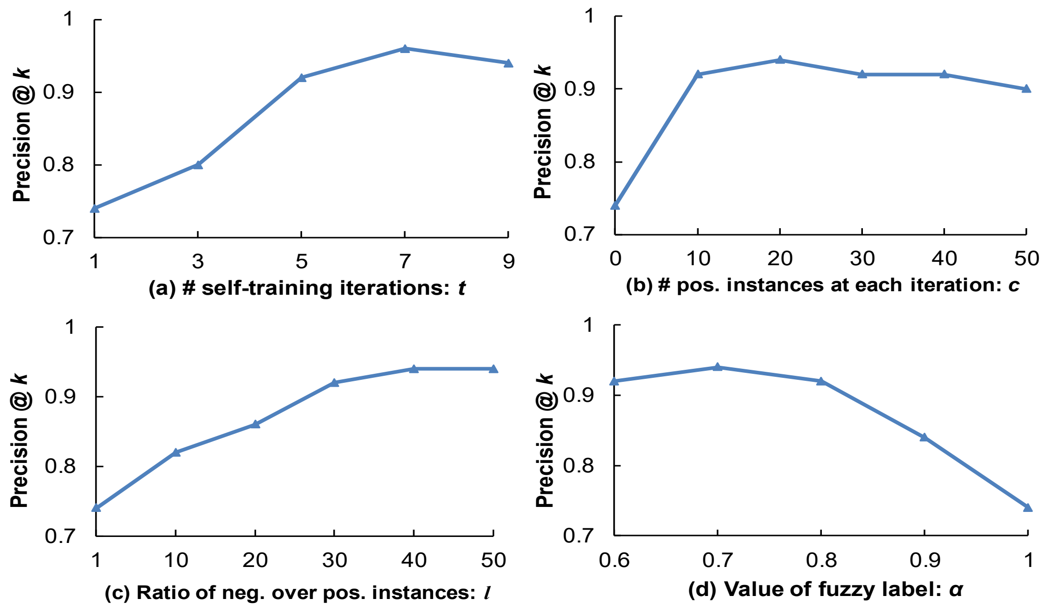


Figure 2.8: Parameter sensitivity of CNL over t, c, l, α for seed node prediction in the DBLP target network when the seed set size is 50. The default settings are $\lambda_1 = 10, \lambda_2 = 0.5, t = 5, c = 10, l = 30, \alpha = 0.8$.

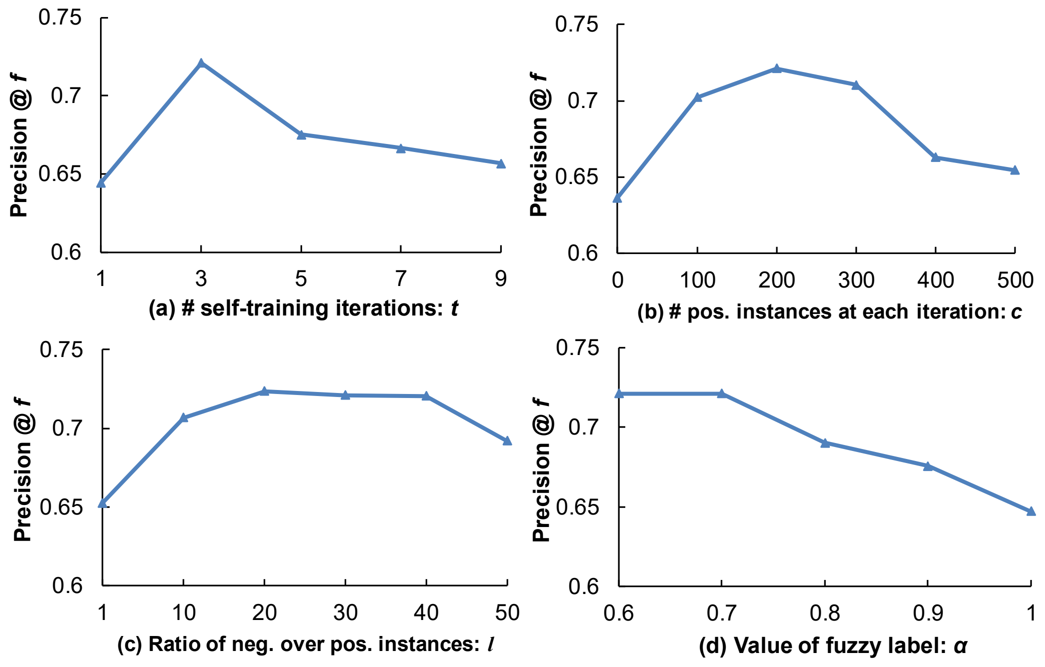


Figure 2.9: Parameter sensitivity of CNL over t, c, l, α for inactive edge prediction in the Email-Enron target network when the edge removal fraction is 30%. The default settings are $\lambda_1 = 10, \lambda_2 = 1, t = 3, c = 200, l = 30, \alpha = 0.7$.

performance will be degraded. This might be caused by the shortcoming of the self-training approach [65], i.e., when t becomes too large, the most confident predicted instances in the target network might become not confident enough. Then, if some wrongly predicted labeled target network instances are utilized as the training data to retrain the prediction model, the prediction performance will be declined. Thus, we fix t=3 in the CNL model for cross-network graph sparsification in all the target networks.

Parameter $\boldsymbol{c}$ indicates the number of most confident predicted positive instances moved from the target network to the training set, at each iteration. As shown in Figure 2.8(b), $c \in \{10,20,30,40,50\}$ all achieve much higher seed node precision than c=0. Also, as shown in Figure 2.9(b), $c \in \{100,200,300\}$ yields much higher inactive edge precision than c=0. These demonstrate the effectiveness of the self-training approach to leverage the target network data to train the prediction model. In addition, we assigned a larger value of c for inactive edge prediction than that for seed node prediction. This is because the number of edges is generally much larger than that of nodes in a target network. Intuitively, when training an edge prediction model, we would require larger number of training samples (i.e. larger c) than that for node prediction.

Parameter $\boldsymbol{l}$ represents the ratio of the number of most confident predicted negative instances over that of positive instances moved from the target network to the training set. As shown in Figure 2.8(c), $l \geq 1$ contributes to much higher seed node precision than $l = 1$. This is because the number of non-seed nodes are much larger than that of seed nodes. If we just add the equal number of most confident predicted seed nodes and non-seed nodes (i.e. $l = 1$) in the target network to the training set, the prediction model might fail to consider enough negative training samples from the target network. Thus, it is effective to incorporate the parameter l to address the imbalanced data condition for seed node prediction. In addition, as shown in Figure 2.9(c), $l \geq 1$ also leads to much higher inactive edge precisions than $l = 1$. In addition, as shown in Figures 2.9(b) and 2.9(c), when c or l become too large, i.e., $c \geq 400, l = 50$, the inactive edge precision will significantly decrease. This is also caused by the weakness of the self-training approach as we explained above, i.e., utilizing not confident enough target network predictions to retrain the model would lead to negative effect on the prediction performance.

Finally, we evaluate the effectiveness of incorporating fuzzy labels in the iterative self-training algorithm for cross-network seed selection and graph sparsification. Parameter $\boldsymbol{\alpha}$ denotes the degree of membership to be positive assigned to the most confident predicted positive instances generated at the last self-training iteration. Note that $0.5 < \alpha < 1$ indicates that we incorporate fuzzy labels, while $\alpha = 1$ indicates that we utilize binary labels instead of fuzzy labels. As shown in Figures 2.8(d) and 2.9(d), $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ all lead to both higher seed node precisions and higher inactive edge precisions than $\alpha = 1$. This confirms the effectiveness of incorporating fuzzy labels in the CNL model for both cross-network seed selection and graph sparsification. In addition, as shown in Figure 2.8(d), the performance of CNL w.r.t. seed node prediction is not sensitive to the value of α, when $\alpha \in \{0.6, 0.7, 0.8\}$. While for inactive edge prediction, as shown in Figure 2.9(d), $\alpha \in \{0.6, 0.7\}$ can achieve much higher precisions than $\alpha = 0.8$. This could be explained by the fact that we utilize a larger number of target network instances (i.e. larger value of c) to train the inactive edge prediction model, as compared to that for seed node prediction. Such larger number of target network instances might be more likely to include the predictions which are not confident enough. Thus, for inactive edge prediction, we would assign lower confident level (smaller value of α) to those most confident target network predictions generated by the last self-training iteration.

2.6 Summary

Although the IM problem has been extensively studied, very little work addresses this problem in a cross-network scenario. In this work, we propose an innovative cross-network learning approach to study two issues of the cross-network IM problem, i.e., cross-network seed selection and cross-network graph sparsification. On one hand, we consider cross-network seed selection for IM as a cross-network node prediction task, with the goal of selecting the nodes most likely to act as seed for IM in the target network. On the other hand, we view cross-network graph sparsification as a cross-network edge prediction task, aiming to remove the edges least likely to contribute to influence propagation for IM in the target network. To achieve such goals, a CNL model is proposed to leverage the knowledge pre-learned from a smaller source network to help predict seed nodes and inactive edges for multiple larger target networks. To address domain discrepancy, lower weights would be assigned to the features which perform less similarly between the source network and the target network. In addition, a fuzzy self-training approach is employed to iteratively retrain the prediction model based on not only the fully labeled data in the source network, but also the most confident predicted labeled data in the target network with their predicted fuzzy labels. With the help of fuzzy labels, we can differentiate the levels of prediction confidence at different self-training iterations so as to reduce the negative effects of the less confident target network predictions on iterative retraining. Experiments on the real-world datasets demonstrate that the proposed CNL model can achieve a good trade-off between the efficiency and effectiveness of the IM task in the target networks. On one hand, by leveraging the cross-network seed selection knowledge, CNL can achieve a satisfactory influence spread comparable to the greedy algorithm in the target network while greatly saving the required running time. On the other hand, by leveraging the cross-network influence propagation knowledge for graph sparsification, CNL just causes a little loss of influence spread in the sparse target networks, while significantly speeding up the IM greedy algorithms. A limitation of the CNL model is that it requires the greedy algorithm and the influence propagation model to be the same in the source network and the target network. Otherwise, the topological features and node/edge labels might be incomparable across networks and might cause negative transfer.

Some preliminary results for cross-network graph sparsification in IM without the provision of fuzzy labels has been published in [9]. In addition, employing the CNL model empowered by fuzzy labeling for both cross-network seed selection and graph sparsification in IM are currently under review in [8]. In the future, we plan to leverage the knowledge pre-learned from multiple source networks instead of a single source network, to make predictions over seed nodes and inactive edges for the target networks. In addition, we can apply the proposed fuzzy self-training approach to address other domain adaptation tasks, i.e., not just limited to the IM application.

Chapter 3
Asymmetry-Aware Deep Network Embedding

3.1 Introduction

Networks are ubiquitous in many real-world applications, such as social networks, co-author networks, biological networks, word co-occurrence networks, and communication networks. Mining the information behind complex networks is important for a variety of graph analytics tasks, such as node classification, network clustering, link prediction, node recommendation, and network visualization. To address such graph mining tasks successfully, a set of informative and discriminative feature vector representations should be predefined for nodes or links. Network embedding aims to learn a low-dimensional feature vector representation for each node in a given network such that the original network structures can be well preserved by the embedding vectors. Then, one can simply apply the vector-based machine learning techniques on the low-dimensional latent vector representations to solve diverse graph mining applications efficiently and effectively.

Network transitivity should be asymmetric in both directed and undirected networks. For example, as shown in Figure 3.1, given two nodes v_i and v_j, due to different local neighborhood structures, the transition probability from v_i to v_j can be rather different from that towards v_i from v_j, in both directed and undirected networks. In addition, considering the asymmetric proximities is important for various graph mining tasks. For example, in node classification, the

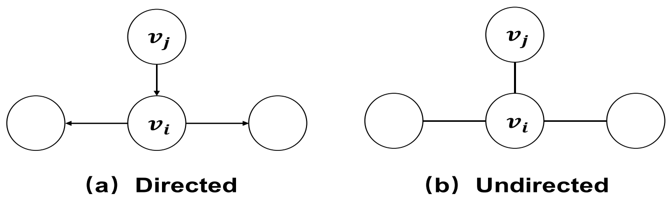


Figure 3.1: Illustration of asymmetric network transitivity in both directed and undirected networks. In a directed network (a), the 1-step transition probability from v_i to v_j is 0, while the 1-step transition probability towards v_i from v_j is 1. In an undirected network (b), the 1-step transition probability from v_i to v_j is 0.33, while the 1-step transition probability towards v_i from v_j is 1.

node pairs with bi-directional strong connections would be more likely to share the same labels as compared to those with unidirectional strong connection. Also, for link prediction in signed networks, the positive links would more tend to be bi-directional than the negative links [51]. However, most existing network embedding algorithms fail to capture such asymmetric relationships. The only few asymmetric network embedding algorithms, namely HOPE [30] and APP [26], employ the matrix factorization and random walk approaches, respectively, while taking advantage of deep network embedding to capture asymmetric proximities has not yet been studied. In the state-of-the-art deep network embedding models [34], [11], [10], [4], when learning the latent vector representation of a specific node, say v_i, the corresponding input raw vector would only consider v_i as a source role, thus, only the network transitivity outward from v_i towards all the other nodes would be captured, while the network transitivity inward towards v_i from all the other nodes have been ignored.

The outward and inward network transitivity capture different neighborhood structure of a specific node. If one network embedding model can well capture both the associated outward and inward transitivity in each node's input raw vector space, then the learned embedding node vector representation should be more comprehensive to preserve the original neighborhood structure. To this end, we propose an <u>as</u>ymmetry-aware <u>d</u>eep <u>n</u>etwork <u>e</u>mbedding (AsDNE) model with outward and inward proximity preservation. Unlike existing deep network embedding models which employ one single stacked auto-encoder (SAE) as the main building block, the proposed AsDNE model consists of two SAEs, i.e., SAE-Out and SAE-In. For each node, SAE-Out and SAE-In will learn an outward and an inward vector representations, respectively, by considering the node as a source role and a target role within its *K*-step network connections. Then, by concatenating the outward and inward vector representations of each node, the final node vector representation can well preserve the asymmetric network proximities. In addition, we incorporate pairwise constraints into SAE-Out and SAE-In to embed node pairs that are more strongly connected in the original network closer to each other in the embedding space. To better capture the asymmetric proximities, we impose stronger constraint on the node pairs with bi-directionally strong connections, as compared to those with only unidirectionally strong connection. Thus, the node pairs which can easily reach each other bi-directionally would have more similar latent vector representations. The network representation learning ability of AsDNE was evaluated on two graph mining tasks, i.e., multi-label node classification and link sign prediction. The distinctive features of AsDNE can be summarized as follows:

1) Two semi-supervised SAEs are employed to learn non-linear network representations with asymmetric proximities preservation;
2) The designed pairwise constraints can distinguish bi-directionally strong connections from unidirectionally strong connections so as to better capture asymmetric relationships;
3) Extensive experiments in the real-world undirected and directed, unweighted and weighted networks demonstrates that AsDNE can learn task-independent network representations outperforming the state-of-the-art network embedding algorithms.

The rest of this chapter is organized as follows. Section 3.2 reviews the state-of-the-art network embedding algorithms. Section 3.3 introduces the detailed framework of AsDNE. Section 3.4 reports the experimental results of AsDNE on public real-world datasets. Section 3.5 summaries this work.

3.2 Related Work

In this section, we review the state-of-the-art network embedding algorithms, which are categorized as random walk based, matrix factorization based, and deep learning based.

3.2.1 Random Walk based Network Embedding Algorithms

The random walk based network embedding algorithms were inspired by word embedding in NLP. DeepWalk [24] is a pioneer work in this family of algorithms which utilizes a Depth-first Sampling (DFS) approach to generate a collection of truncated random walks in the given network. Then, by viewing a network as a document, a node as a word, and the sampled random walks as short sentences, the Skip-Gram language model [41] was extended to learn the low-dimensional node vector representations. Tang *et al.* [43] proposed a LINE algorithm to generate the first-order and second-order neighborhood via a Breath-first Sampling (BFS) strategy. Then, an objective function was carefully designed to learn the node vector representations which can preserve the first-order and second-order proximities between nodes in the network. Instead of defining rigid notions of neighborhoods like DeepWalk [24] and LINE [43], Grover and Leskovec [23] introduced a biased random walk sampling strategy interpolating between DFS and BFS to generate the flexible neighborhood. Then, a node2vec algorithm which employs Skip-Gram language model [41] and negative sampling [94] was developed to learn the low-dimensional node vector representations, with the goal of maximizing the likelihood of neighborhood preservation.

3.2.2 Matrix Factorization based Network Embedding Algorithms

A family of low rank embedding algorithms has been proposed to linearly project the representation space of the original network into a low rank space. For example, Wang *et al.* [6] proposed a Modularized Nonnegative Matrix Factorization (M-NMF) model to preserve both microscopic structure and mesoscopic community structure in the network. To learn the low-dimensional node representations, the NMF technique was adopted to factorize the adjacency matrix of the given network. Cao *et al.* [29] proposed a GraRep algorithm to factorize each *k*-step positive pointwise mutual information (PPMI) matrix via SVD and then concatenate multiple *k*-step low-rank representations as the final representation. The matrix factorization based network embedding models can be seen as performing linear dimensionality reduction, thus, they might fail to capture the highly non-linear properties of the complex network structures.

3.2.3 Deep Learning based Network Embedding Algorithms

Due to the recent success of deep learning in representation learning [95], [96], several deep network embedding algorithms have been proposed. For example, Tian *et al.* [4] utilized a sparse SAE to learn deep network representations for network clustering. Instead of utilizing an unsupervised SAE, Yang *et al.* [5] employed a semi-supervised SAE to reconstruct the modularity matrix of a given network so as to learn the deep network representations for community detection. A pairwise constraint was incorporated into the SAE to make the nodes belonging to the same community have similar embedding vector representations. In SDNE, Wang *et al.* [10] proposed a semi-supervised SAE to reconstruct the adjacency matrix and map the directly connected node pairs closer to each other in the embedding space. SDNE only captures the first-order and second-order proximities between nodes in a network. To capture high-order proximities, Cao *et al.* [34] developed a DNGR model which utilizes an unsupervised de-noising SAE to reconstruct the PPMI matrix. Furthermore, to take advantage of the semi-supervised approach and preserving high-order proximities, we proposed a DNE-APP model [11] which employs a semi-supervised SAE to learn deep network representations by reconstructing the aggregated *K*-th order proximity matrix and mapping node pairs with higher aggregated proximities closer to each other in the embedding space. Besides, some deep network embedding models have been proposed to employ convolutional neural networks to learn deep network representations [35], [36], [97].

Network transitivity and proximity should be asymmetric in both directed and undirected networks. However, most existing network embedding algorithms fail to capture such asymmetric properties. Only few studies focus on asymmetric network embedding. Ou *et al.* [30] proposed a HOPE algorithm to factorize the high-order Katz proximity matrix via SVD. Then, a source and a target vector representations would be learned for each node so as to capture the asymmetric network transitivity. Zhou *et al.* [26] designed an APP model based on random walk sampling and Skip-Gram language model. By considering each node as a source role and a target role in the sampled paths, a source and a target vector representations would be learned for each node to capture the asymmetric network proximities. However, taking advantage of deep network embedding to preserve the asymmetric network proximities has not been exploited. To fill this gap, instead of utilizing one SAE as existing deep network embedding models [10], [34], [11], [4], [5], we employed two SAEs in the proposed AsDNE model to simultaneously capture the asymmetric outward and inward network proximities.

In addition, the pairwise constraints incorporated in DNE-APP [11] and SDNE [10] do not specifically consider the asymmetric proximities between connected nodes. To better capture the asymmetric pairwise proximities, we devised the asymmetry-aware pairwise constraints to judiciously make the bi-directionally strongly connected node pairs possess more similar latent vector representations than the unidirectionally strongly connected node pairs.

3.3 AsDNE Model

In this section, we elaborate the framework of the AsDNE model. Firstly, we define the asymmetric outward and inward proximity matrices, then, we explain how to employ two SAEs to preserve the asymmetric proximities, how to design the asymmetry-aware pairwise constraints, and how to optimize the AsDNE model. Table 3.1 summaries the frequently used notations and the corresponding descriptions in this chapter.

Table 3.1: Frequently used notations and descriptions in Chapter 3.

Notations	Descriptions
$\mathcal{A}, \mathcal{A}^T$	Aggregated outward and inward transition probability matrices
O, I	Outward and inward proximity matrices
l	Number of layers in SAE-Out and SAE-In
$d(k)$	Dimensionality of the *k*-th hidden layer of SAE-Out and SAE-In
$W_1^{O(k)}, W_2^{O(k)}$	Encoding and decoding weight matrices of *k*-th layer of SAE-Out
$B_1^{O(k)}, B_2^{O(k)}$	Encoding and decoding bias matrices of *k*-th layer of SAE-Out
$W_1^{I(k)}, W_2^{I(k)}$	Encoding and decoding weight matrices of *k*-th layer of SAE-In
$B_1^{I(k)}, B_2^{I(k)}$	Encoding and decoding bias matrices of *k*-th layer of SAE-In
$H^{O(k)}$	Outward matrix representation learned by *k*-th layer of SAE-Out
$H^{I(k)}$	Inward matrix representation learned by *k*-th layer of SAE-In
r	Ratio of pairwise constraint weight on bi-directionally strongly connected node pairs over that of unidirectionally strongly connected node pairs

3.3.1 Asymmetric Outward and Inward Proximities

Given a network $G = (V, E)$ with a set of nodes $V = \{v_i\}_{i=1}^{n}$ and a set of edges

$E = \{e_{ij}\}$, the adjacency matrix S is defined as: $S_{ij} > 0$ if v_i can reach v_j after exactly one-step; otherwise, $S_{ij} = 0$. Based on S, the one-step transition probability matrix is computed as $A^1 = D^{-1}S$, where D denotes the degree matrix of network G. Then, any *k*-step $(k \geq 2)$ transition probability matrix can be computed as $A^k = A^{k-1}A^1$, where A_{ij}^k represents the transition probability from the source node v_i to the target node v_j after exactly *k*-steps. Alternatively, in very large-scale networks, we can approximate the *k*-step transition probability matrix based on the random walk sampling approach [25] or the random surfing strategy [34]. After generating a series of *k*-step transition probability matrices up to the maximum *K*-step, we can aggregate an overall transition probability matrix by assigning higher weights to closer neighbors as: $\mathcal{A} = \sum_{k=1}^{K} A^k/k$, where $\mathcal{A}_{ij} > 0$ if v_i can reach v_j within K steps; otherwise, $\mathcal{A}_{ij} = 0$. Due to different local neighborhood structures, the overall transition probability matrix $\mathcal{A}$ is asymmetric in both directed and undirected networks. In this work, we refer to $\mathcal{A}$ and $\mathcal{A}^T$ as the outward and inward transition probability matrix, respectively, where the *i*-th row of $\mathcal{A}$ and $\mathcal{A}^T$ capture the outward transition probabilities from v_i towards all the nodes, and the inward transition probabilities towards v_i from all the nodes, in network G.

The PPMI metric [98] has been typically employed by the state-of-the-art network embedding algorithms [29], [11], [34] to measure node proximities based on the outward transition probability matrix. However, the outward and inward network transitivity play a rather different role in capturing the local neighborhood structure of a specific node. Thus, in this work, we employ PPMI in both outward and inward transition probability matrices to measure the asymmetric proximities between node pairs. Specifically, if v_i can reach v_j within *K* steps, i.e., $\mathcal{A}_{ij} > 0$, the outward proximity from v_i to v_j is measured as:

$$O_{ij} = max\left(log \frac{OP(v_i,v_j)}{\sum_{k=1}^{n} OP(v_k,v_j)OP(v_k)}, 0\right) \tag{3.1}$$

where $OP(v_i, v_j) = \mathcal{A}_{ij}/\sum_{k=1}^{n} \mathcal{A}_{ik}$ indicates the normalized outward transition probability from v_i to v_j within *K* steps; $OP(v_k)$ denotes the probability of a path randomly starting at v_k, by assuming a uniform distribution, we set $OP(v_k) = 1/n$, where n is the number of nodes in network *G*. In addition, if there is no connection from v_i to v_j within *K* steps, i.e., $\mathcal{A}_{ij} = 0$, then we set $O_{ij} = 0$. To reduce noises, PPMI [98] replaces all the negative PMI [99] values by 0. Thus, even though $\mathcal{A}_{ij}$ is positive but not large enough, i.e., v_i has a very weak outward connection towards v_j, then O_{ij} will still be 0. Therefore, $O_{ij} > 0$ iff there exists a strong outward connection from v_i to v_j within *K* steps.

Similarly, given an inward connection towards v_i from v_j within *K* steps, i.e., $(\mathcal{A}^T)_{ij} > 0$, the inward proximity towards v_i from v_j is computed as:

$$I_{ij} = max(log \frac{IP(v_i,v_j)}{\sum_{k=1}^{n} IP(v_k,v_j)IP(v_k)}, 0) \tag{3.2}$$

where $IP(v_i, v_j) = (\mathcal{A}^T)_{ij}/\sum_{k=1}^{n}(\mathcal{A}^T)_{ik}$ represents the normalized inward transition probability towards v_i from v_j; $IP(v_k) = 1/n$ denotes the probability of a path randomly ending at v_k; and if $(\mathcal{A}^T)_{ij} = 0$, we set $I_{ij} = 0$. Note that $I_{ij} > 0$ iff the inward connection towards v_i from v_j is strong enough.

In this work, we use O and I to denote the outward and inward proximity matrices, respectively. The i-th row of O and I represent the proximities associated with v_i, by viewing v_i as a source role and a target role within its associated *K*-step network connections, respectively.

3.3.2 SAE-Out and SAE-In

Next, we employ two SAEs, i.e., SAE-Out and SAE-In, to learn the low-dimensional node vector representations with asymmetric proximities preservation. Firstly, given the outward proximity matrix O as the input to SAE-Out (i.e. $H^{O(0)} = O$), the hidden representations can be learned layer-by-layer as:

$$H^{O(k)} = f\left(H^{O(k-1)}\left(W_1^{O(k)}\right)^T + B_1^{O(k)}\right), \ k = 1, \dots, l \tag{3.3}$$

where f is a non-linear encoding function; l indicates the number of layers of SAE-Out; $W_1^{O(k)} \in R^{d(k)\times d(k-1)}$ and $B_1^{O(k)} \in R^{n\times d(k)}$ denote the encoding weight and bias matrices at the k-th layer of SAE-Out, and $d(k)$ indicates the dimensionality of the k-th hidden layer of SAE-Out. The i-th row of the input matrix, i.e., $O(i) \in R^{1\times n}$, captures the outward proximities from v_i towards all the nodes in a network. $H^{O(k)} \in R^{n\times d(k)}$ represents the hidden matrix representation learned by the k-th layer of SAE-Out. The i-th row of $H^{O(k)}$, denoted as $H^{O(k)}(i) \in R^{1\times d(k)}$, represents the latent vector representation of v_i learned by the k-th layer of SAE-Out, which captures the outward proximities associated with v_i.

Then, at the decoding step, giving $\widehat{H}^{O(l)} = H^{O(l)}$, the input matrix can be reconstructed in a reverse order as below:

$$\widehat{H}^{O(k-1)} = f\left(\widehat{H}^{O(k)}\left(W_2^{O(k)}\right)^T + B_2^{O(k)}\right), \ k = l, \dots, 1 \tag{3.4}$$

where f is a non-linear decoding function; $W_2^{O(k)} \in R^{d(k-1)\times d(k)}$ and $B_2^{O(k)} \in R^{n\times d(k-1)}$ refer to the decoding weight and bias matrices associated with the k-th layer of SAE-Out; and $\widehat{H}^{O(0)} = \widehat{O}$ is the reconstructed input matrix learned by SAE-Out. In this work, we employed the sigmoid function $f(x) = 1/(1 + e^{-x})$ as both the encoding and decoding function.

In general, the number of strongly connected node pairs are smaller than that of weakly connected and disconnected node pairs, thus yielding a sparse outward proximity matrix O. To address the sparsity issue, we modify the reconstruction errors of SAE-Out by incorporating a penalty matrix $P^{O(1)} \in R^{n \times n}$, as in [10]:

$$\mathcal{J}_1^{O(1)} = \frac{1}{2n} \| P^{O(1)} \odot (\hat{O} - O) \|_F^2 \tag{3.5}$$

where $\odot$ denotes an element-wise Hadamard product; $P_{ij}^{O(1)} = \beta$, if $O_{ij} > 0$ and $P_{ij}^{O(1)} = 1$, if $O_{ij} = 0$. $\beta > 1$ denotes the ratio of penalty on the reconstruction errors of the positive outward proximities over that of the zero outward proximities. Incorporating β makes SAE-Out focus more on reconstructing the strong outward connections than the weak or unobserved outward connections. Similarly, we define the reconstruction errors for any k-th $(1 \leq k \leq l)$ layer of SAE-Out as:

$$\mathcal{J}_1^{O(k)} = \frac{1}{2n} \| P^{O(k)} \odot (\widehat{H}^{O(k-1)} - H^{O(k-1)}) \|_F^2 \tag{3.6}$$

where $P^{O(k)} \in R^{n \times d(k-1)}, P_{ij}^{O(k)} = \beta$, if $H^{O(k-1)} > 0$ and $P_{ij}^{O(k)} = 1$, if $H^{O(k-1)} = 0$. Even though when $k \geq 2$, the input matrix $H^{O(k-1)}$ has become dense, we still keep $P^{O(k)}$ in $\mathcal{J}_1^{O(k)}$ by regarding β^2 as the weight of the reconstruction errors in the overall loss function introduced latter.

On the other hand, giving the inward proximity matrix I as the input to SAE-In, i.e., $H^{I(0)} = I$, the reconstruction errors at any k-th layer of SAE-In are similarly defined as:

$$\mathcal{J}_1^{I(k)} = \frac{1}{2n} \| P^{I(k)} \odot (\widehat{H}^{I(k-1)} - H^{I(k-1)}) \|_F^2 \tag{3.7}$$

where $P^{I(k)} \in R^{n \times d(k-1)}$, $P_{ij}^{I(k)} = 1$, if $H_{ij}^{I(k-1)} = 0$ and $P_{ij}^{I} = \beta > 1$, if $H_{ij}^{I(k-1)} > 0$. In addition, the i-th row of $H^{I(k-1)}$, i.e., $H^{I(k-1)}(i) \in R^{1 \times d(k-1)}$ learned by the (k-1)-th layer of SAE-In, represents the hidden vector representation of v_i capturing its associated inward proximities.

3.3.3 Asymmetry-Aware Pairwise Constraints

Next, we incorporate pairwise constraints into SAE-Out and SAE-In to map node pairs with higher proximities closer to each other in the low-dimensional embedding space. Given two node pairs, (v_i, v_j) and (v_i, v_k), assume that $O_{ij} + O_{ji} = O_{ik} + O_{ki}$. In addition, both v_i and v_j can easily reach each other within K steps, while only v_i can easily reach v_k but v_k fails to easily reach v_i within K steps. Then, according to the pairwise constraints designed in [10], [11], v_i would be mapped closely to v_j and v_k to the same extent. However, v_i should be considered as more similar to v_j than v_k, due to the bi-directionally strong connections between v_i and v_j. To capture such asymmetric relationships, we devise the following pairwise constraint for SAE_Out:

$$\mathcal{J}_2^{O(k)} = \frac{1}{2n} \sum_{i,j=1}^{n} R_{ij}^{O} (O_{ij} + O_{ji}) \, ||H^{O(k)}(i) - H^{O(k)}(j)||_2^2 \tag{3.8}$$

where $R_{ij}^O = r$ iff $O_{ij} > 0$ and $O_{ji} > 0$; otherwise, $R_{ij}^O = 1$. $r > 1$ denotes the ratio of the weight of the pairwise constraint on the bi-directionally strongly connected node pairs over that of the unidirectionally strongly connected node pairs. Minimizing $\mathcal{J}_2^{O(k)}$ yields an embedding space where the node pairs having bi-directionally positive proximities are much closer, w.r.t. the node pairs having unidirectionally positive proximity. Equation (3.8) can be written as $\mathcal{J}_2^{O(k)} = Tr((H^{O(k)})^T L^O H^{O(k)})/n$, where $Tr(.)$ denotes the trace of a matrix; L^O refers to the Laplacian matrix of $U^O = R^O \odot (O + O^T)$, i.e., $L^O = D_{U^O} - U^O$ and D_{U^O} is a diagonal matrix with the diagonal entries as the row summation of U^O, i.e., $\left(D_{U^O}\right)_{ii} = \sum_{j=1}^{n} U_{ij}^O$. Similarly, for SAE-In, the pairwise constraint is devised as:

$$\mathcal{J}_2^{I(k)} = \frac{1}{2n} \sum_{i,j=1}^{n} R_{ij}^{I} (I_{ij} + I_{ji}) \, ||H^{I(k)}(i) - H^{I(k)}(j)||_2^2 \tag{3.9}$$

where $R_{ij}^I = r > 1$ iff $I_{ij} > 0$ and $I_{ji} > 0$; otherwise, $R_{ij}^I = 1$.

Then, by combining the reconstruction errors $\mathcal{J}_1^{O(k)}$ and the pairwise constraint $\mathcal{J}_2^{O(k)}$, the overall loss function of SAE-Out is defined as follows:

$$\mathcal{J}^O = \sum_{k=1}^{l} \mathcal{J}^{O(k)} = \sum_{k=1}^{l} \mathcal{J}_1^{O(k)} + \alpha_k \mathcal{J}_2^{O(k)} + \lambda_k \mathcal{J}_3^{O(k)} \tag{3.10}$$

where $\mathcal{J}_3^{O(k)} = ||W_1^{O(k)}||_F^2 + ||W_2^{O(k)}||_F^2$ refers to a *L-2* norm regularization to prevent over-fitting; $\alpha_k, \lambda_k > 0$ are the weights of $\mathcal{J}_2^{O(k)}$ and $\mathcal{J}_3^{O(k)}$ at the *k*-th layer of SAE-Out. Similarly, the overall loss function of SAE-In is defined as:

$$\mathcal{J}^I = \sum_{k=1}^{l} \mathcal{J}^{I(k)} = \sum_{k=1}^{l} \mathcal{J}_1^{I(k)} + \alpha_k \mathcal{J}_2^{I(k)} + \lambda_k \mathcal{J}_3^{I(k)} \tag{3.11}$$

where $\mathcal{J}_3^{I(k)} = ||W_1^{I(k)}||_F^2 + ||W_2^{I(k)}||_F^2$.

3.3.4 Optimization of AsDNE

To optimize AsDNE, we can optimize SAE-Out and SAE-In in parallel. Firstly, to optimize each *k*-th layer of SAE-Out, one can employ back-propagation algorithm to compute the "error" terms of its output layer $\delta_3^{O(k)}$ and hidden layer $\delta_2^{O(k)}$ as follows:

$$\delta_3^{O(k)} = \left(\hat{H}^{O(k-1)} - H^{O(k-1)}\right) \odot P^{O(k)} \odot P^{O(k)} \odot f'\left(Z_3^{O(k)}\right) \tag{3.12}$$

$$\delta_2^{O(k)} = \left(\delta_3^{O(k)} W_2^{O(k)} + \alpha_k (L^O + (L^O)^T) H^{O(k)}\right) \odot f'\left(Z_2^{O(k)}\right) \tag{3.13}$$

where $Z_3^{O(k)} = H^{O(k)}\left(W_2^{O(k)}\right)^T + B_2^{O(k)}$, $Z_2^{O(k)} = H^{O(k-1)}\left(W_1^{O(k)}\right)^T + B_1^{O(k)}$ and f' denotes the derivative of the activation function. Based on $\delta_3^{O(k)}$ and $\delta_2^{O(k)}$, we can compute the partial derivatives of the overall loss function of the *k*-th layer of SAE-Out w.r.t. the encoding weight $W_1^{O(k)}$, decoding weight $W_2^{O(k)}$, encoding bias $B_1^{O(k)}$, and decoding bias $B_2^{O(k)}$ as follows:

$$\frac{\partial \mathcal{J}^{O(k)}}{\partial W_1^{O(k)}} = \frac{1}{n}\left(\delta_2^{O(k)}\right)^T H^{O(k-1)} + \lambda_k W_1^{O(k)} \tag{3.14}$$

$$\frac{\partial \mathcal{J}^{O(k)}}{\partial W_2^{O(k)}} = \frac{1}{n}\left(\delta_3^{O(k)}\right)^T H^{O(k)} + \lambda_k W_2^{O(k)} \tag{3.15}$$

$$\frac{\partial \mathcal{J}^{O(k)}}{\partial B_1^{(k)}} = \delta_2^{O(k)}/n \tag{3.16}$$

$$\frac{\partial \mathcal{J}^{O(k)}}{\partial B_2^{O(k)}} = \delta_3^{O(k)}/n \tag{3.17}$$

To minimize the loss function $\mathcal{J}^{O(k)}$, one can use stochastic gradient descent (SGD) to iteratively update the parameters as follows:

$$W_1^{O(k)} = W_1^{O(k)} - \eta_k \frac{\partial \mathcal{J}^{O(k)}}{\partial W_1^{O(k)}} \tag{3.18}$$

$$W_2^{O(k)} = W_2^{O(k)} - \eta_k \frac{\partial \mathcal{J}^{O(k)}}{\partial W_2^{O(k)}} \tag{3.19}$$

$$B_1^{O(k)} = B_1^{O(k)} - \eta_k \frac{\partial \mathcal{J}^{O(k)}}{\partial B_1^{O(k)}} \tag{3.20}$$

$$B_2^{O(k)} = B_2^{O(k)} - \eta_k \frac{\partial \mathcal{J}^{O(k)}}{\partial B_2^{O(k)}} \tag{3.21}$$

where η_k indicates the learning rate of the *k*-th layer of SAE-Out. Next, we greedily layer-wise optimize SAE-Out until reaching the deepest (i.e. *l*-th) layer and learn the deepest hidden matrix representation with outward proximity preservation, i.e., $H^{O(l)}$. Similarly, with greedily layer-wise optimization of SAE-In, we can learn the deepest hidden matrix representation with inward proximity preservation, i.e., $H^{I(l)}$. Finally, we concatenate $H^{O(l)}$ and $H^{I(l)}$ to get the final matrix representation $H^{(l)}$. Note that the *i*-th row of $H^{(l)}$, i.e., $H^{(l)}(i) \in R^{1\times d}, d = 2 \times d(l)$, corresponds to the feature vector representation of v_i, where the first half (i.e. $H^{O(l)}(i) \in R^{1\times d/2}$) captures the outward network transitivity from v_i while the latter half (i.e. $H^{I(l)}(i) \in R^{1\times d/2}$) preserves the inward transitivity towards v_i.

The time complexity of AsDNE is O(*nchi*), where n denotes the number of nodes in a network, c indicates the average number of strongly connected neighbors (within *K* steps) per node, $h = d(1)$ represents the maximum hidden dimensionality in SAE-Out and SAE-In, and *i* refers to the number of training

Algorithm 3.1: AsDNE

Input: Outward and inward proximity matrices within K steps, i.e., O and I; Parameters $l, \beta, \alpha, r, \lambda$.

```
1. Greedy layer-wised training for SAE-Out:
Set $H^{O(0)} = O$
For k=1: $l$
    1.1 Leverage $H^{O(k-1)}$ as input to k-th layer of SAE-Out;
    1.2 Given $H^{O(k-1)}$ and $O$, optimize k-th layer of SAE-Out by finding $\theta^{O(k)*} = \left\{W_1^{O(k)*}, W_2^{O(k)*}, B_1^{O(k)*}, B_2^{O(k)*}\right\} = arg\min_{\theta^{O(k)}} \mathcal{J}^{O(k)}$ via SGD;
    1.3 Leverage $\theta^{O(k)*}$ to learn $H^{O(k)}$;
End for
2. Greedy layer-wised training for SAE-In:
Set $H^{I(0)} = I$
For k=1: $l$
    2.1 Leverage $H^{I(k-1)}$ as input to k-th layer of SAE-In;
    2.2 Given $H^{I(k-1)}$ and $I$, optimize k-th layer of SAE-In by finding $\theta^{I(k)*} = \left\{W_1^{I(k)*}, W_2^{I(k)*}, B_1^{I(k)*}, B_2^{I(k)*}\right\} = arg\min_{\theta^{I(k)}} \mathcal{J}^{I(k)}$ via SGD;
    2.3 Leverage $\theta^{I(k)*}$ to learn $H^{I(k)}$;
End for
```

Output: Concatenate $H^{O(l)}$ and $H^{I(l)}$ to get $H^{(l)}$.

iterations. Since *chi* is independent of n, the overall time complexity of AsDNE is linear to the number of nodes.

3.4 Experiments

3.4.1 Datasets

We evaluate the proposed AsDNE model in several real-world datasets, including weighted and unweighted, directed and undirected networks. The statistics of the datasets are shown in Table 3.2. Blogcatalog [100] is an online social network, where each user can be associated with multiple interested groups. Cora [101] is a paper citation network, where one paper can be labeled with multiple topics. Since a citation reflects that the two associated papers tend to have similar topic, we modeled citation as undirected relationship in the experiments. IMDb [102] is a weighted movie co-stars network, where each movie can be associated with multiple genres, and the weight of each edge represents the number of common stars between two movies. We evaluated multi-label node classification on the Blogcatalog, Cora and IMDb datasets.

In addition, Wiki [103], Slashdot [104] and Epinions [105] are directed signed networks, where each directed edge is associated with a signed label, either positive or negative. 78.43%, 76.18% and 89.55% of edges are positive in the Wiki, Slashdot and Epinions datasets, respectively. We employed these three datasets for link sign prediction.

Table 3.2: Statistics of the datasets.

Dataset	# Nodes	# Edges	# Labels
Blogcatalog	10312	333983	39
Cora	11471	33416	10
IMDb	19359	362079	21
Wiki	7118	103675	2
Slashdot	7000	238029	
Epinions	7000	451149	

3.4.2 Baselines

AsDNE was benchmarked against the following state-of-the-art network embedding algorithms, including random walks based, matrix factorization based, and deep learning based.

1) **DeepWalk** [24]: It generates a collection of random walks via DFS, and then

employs Skip-Gram language model [41] to learn node vector representations.

2) **LINE** [43]: It utilizes a BFS strategy to generate the first-order and second-order neighborhood and preserve the first-order and second-order proximities.
3) **GraRep** [29]: It uses SVD to factorize each k-step PPMI matrix and then concatenates all the k-step representations as the final representation.
4) **SDNE** [10]: It employs a semi-supervised SAE to reconstruct the adjacency matrix and map the directly connected node pairs closer to each other. It preserves the first-order and second-order proximities.
5) **DNGR** [34]: It adopts an unsupervised denoising SAE to reconstruct the PPMI matrix. It can preserve high-order proximities.
6) **DNE-APP** [11]: It employs a semi-supervised SAE to reconstruct the aggregated k-th order PPMI matrix and map nodes pairs with higher aggregated proximities closer to each other.
7) **APP** [26]: It is a random walk based embedding algorithm based on Skip-gram language model. Unlike DeepWalk, it can preserve asymmetric proximities by learning a source and a target vector representations for each node.
8) **HOPE** [30]: It employs SVD to factorize the high-order Katz proximity matrix and learns a source and a target vector representations for each node so as to capture the asymmetric network transitivity.

3.4.3 Implementation Details

In the proposed AsDNE model, both SAE-Out and SAE-In were constructed as a 2-layer SAE, where the number of dimensions at the first hidden layer and the second hidden layer were set as $d(1) = 256$ and $d(2) = 128$, respectively. By concatenating the deepest hidden representations of SAE-Out and SAE-In, the number of dimensions of our final node vector representation was $d = 2 \times d(2) =$

256. For all datasets, we set the ratio of the weight of pairwise constraint on the bi-directionally strong connections over that of the unidirectionally strong connections as *r*=4; and set the weight of pairwise constraint for different layers of SAE-Out and SAE-In as $\alpha_1 = \alpha = 0.5$ and $\alpha_k = \alpha/2, \forall\, k \geq 2$. In addition, we set the weight of regularization for different layers of SAE-Out and SAE-In as $\lambda_1 = 0.05, \lambda_k = 0.01, \forall\, k \geq 2$. Also, we set the ratio of reconstruction penalty on the strong connections over that of weak or unobserved connections as β =5 on the IMDb, Slashdot and Epinions datasets and β =6 on the other datasets. In addition, we set the maximum step *K*=6 for node classification in all datasets, while set *K* as 1, 1 and 2 for link sign prediction in the Wiki, Slashdot and Epinions datasets, respectively.

For fair comparisons, we fixed the same dimensionality of node vector representations, i.e., *d*=256 for all the baselines. We also built a 2-layer SAE for all the deep network embedding baselines (i.e. SDNE, DNGR and DNE-APP), where the number of hidden dimensions at each layer of SAE was equal to that of the concatenated hidden representations in our AsDNE model. The maximum *K*-step in GraRep, DNGR and DNE-APP were set with the same values as in our AsDNE model. For GraRep, the dimension of each *k*-step representation was set as $round(d/K)$, i.e., d/K is rounded to the nearest integer. For APP and HOPE, the dimensionality of the source and the target vector representation was set as *d*/2, as in AsDNE. For DeepWalk, the number of walks started from each node was set as 40, and the walk length was set as 80. Note that DeepWalk only works in undirected and unweighted networks, while node2vec is flexible to work in weighted and directed networks. In addition, when setting the parameters as *p*=1, *q*=1 in node2vec, DeepWalk can be a special case of node2vec [23]. Thus, in the directed or weighted networks, we report the results of DeepWalk by running node2vec with the specific parameter setting.

3.4.4 Multi-label Node Classification

For multi-label node classification, we randomly split all the nodes as a training set and a testing set, where the training fraction ranges from 10% to 90%. Then, we train a one-vs-rest LR classifier based on the labeled nodes in the training set and then leverage the classifier to predict multiple labels for each testing node. We repeated the random split 10 times for each network embedding algorithm and report their average Macro-F1 and Micro-F1 scores [106] over the same 10 random splits. The Macro-F1 metric computes F1 score by giving equal weight to each class while the Micro-F1 metric gives equal weight to each sample.

Firstly, as shown in Figures. 3.2(a) and 3.2(b), AsDNE can achieve the highest Micro-F1 and Macro-F1 scores among all the comparing algorithms, no matter what percentage of labeled nodes were used for training in the Blogcatalog and Cora networks. In addition, as shown in Figure 3.2(c), although DNE-APP achieved higher Micro-F1 scores than AsDNE when using less than 40% of labeled nodes for training in the IMDb network, AsDNE can achieve significantly higher Macro-F1 scores than DNE-APP under all training percentages. It is worth noting that both DNGR and DNE-APP adopt a single SAE to reconstruct the outward proximity matrix. While in the AsDNE model, two SAEs are employed to reconstruct the asymmetric outward and inward proximity matrices. The better overall performance of AsDNE over DNGR and DNE-APP demonstrates that even in undirected networks, considering both outward and inward proximities would yield more informative node feature representations.

In addition, we can see that DNE-APP, DNGR, and DeepWalk achieved much higher Micro-F1 and Macro-F1 scores than SDNE and LINE. Note that DNE-APP, DNGR and DeepWalk can capture high-order proximities between different nodes in the network, while SDNE and LINE are only able to capture the first-order and

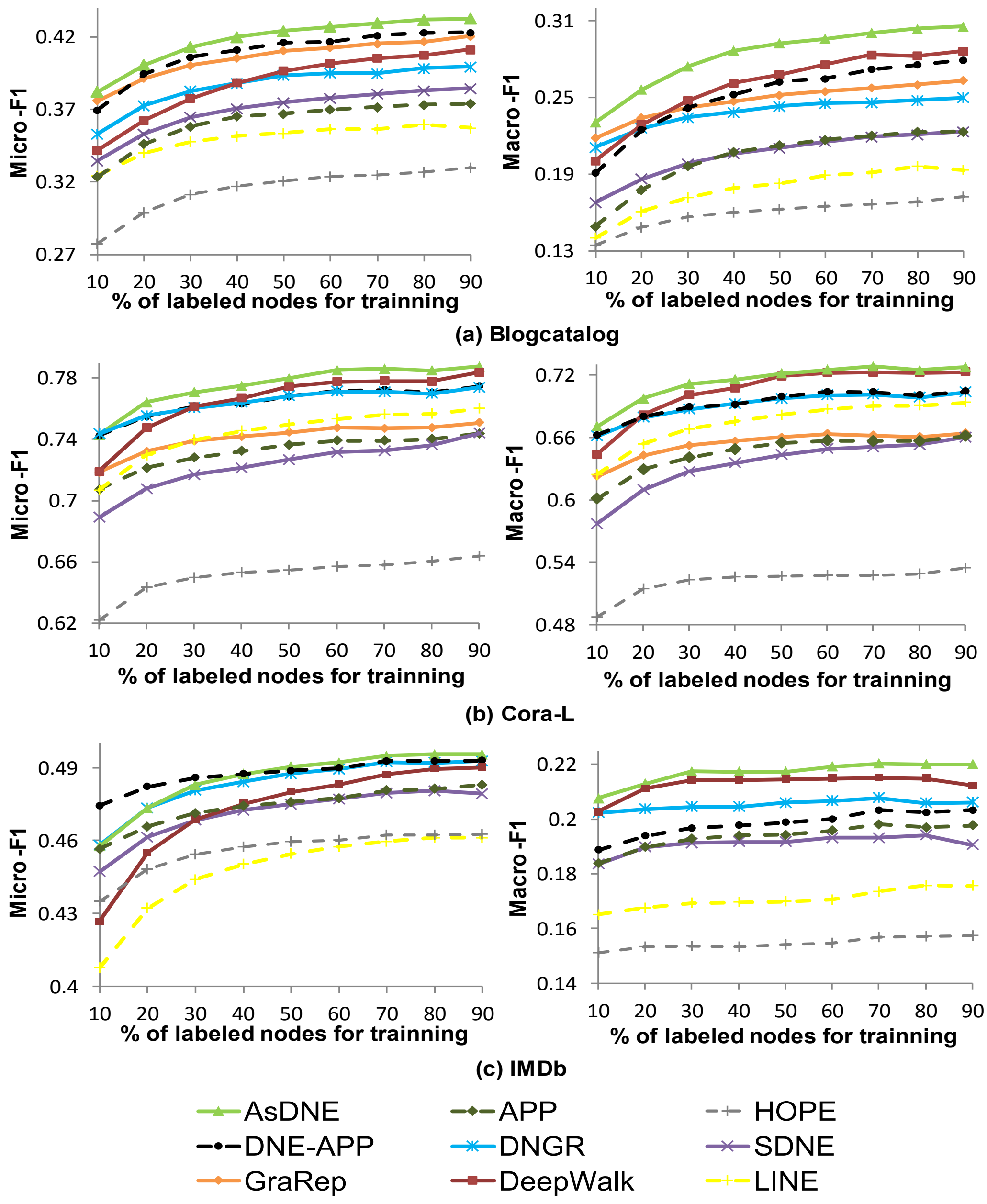


Figure 3.2: Micro-F1 and Macro-F1 scores of multi-label node classification on Blogcatalog, Cora and IMDB networks. GraRep failed to work in the largest IMDb dataset in the experiments, due to its high complexity.

second-order proximities. Thus, it demonstrates that capturing high-order proximities is rather important and necessary for learning the informative feature vector representations for node classification.

3.4.5 Link Sign Prediction

For link sign prediction, we firstly utilize the unsigned version of the signed networks to learn the low-dimensional node vector representations. Then, given an edge e_{ij}, its vector representation is constructed as $H(e_{ij}) = H(i) \odot H(j)$, where $H(i)$ and $H(j)$ denote the deepest latent vector representations of node v_i and v_j, respectively. Next, we randomly sample a fraction of labeled edges to train a LR classifier and employ the classifier to predict the signed labels of remaining edges. Since the real-world signed networks are rather imbalanced, i.e., containing overwhelmingly positive links, directly evaluating the accuracy on such datasets will be misleading. Thus, we followed [107], [108], [109] to adopt the Area Under ROC Curve (AUC) metric to evaluate the link sign prediction performance, which is insensitive to the imbalanced data. The higher the AUC score, the better the performance. For all the network embedding algorithms, we report the average AUC scores over the same 5 random splits.

Firstly, as shown in Tables 3.3, 3.4, and 3.5, the proposed AsDNE model always significantly outperforms all the baselines, i.e., achieves the highest AUC score in all the three networks. For example, when using 20% of labeled edges for training in the sparest Wiki dataset (as shown in Table 3.3), AsDNE can achieve a 6.4% higher AUC score than the best baseline, i.e., DNE-APP.

Secondly, we can see that in all the three networks, both DNE-APP and SDNE achieved much higher AUC scores than DNGR. Note that both DNE-APP and SDNE employ a semi-supervised SAE for network embedding, while DNGR adopts an unsupervised SAE. Thus, the better performance of DNE-APP and SDNE over DNGR demonstrates the effectiveness of the semi-supervised approach for learning network representations for link sign prediction. The same findings were also observed in our previous work [11].

Table 3.3: The AUC Scores of link sign prediction on the Wiki dataset.

Algorithms	**% of labeled edges for training**			
	20	40	60	80
AsDNE	**0.7485**	**0.7535**	**0.7542**	**0.7529**
DNE-APP	0.7036	0.7097	0.7108	0.7133
SDNE	0.7029	0.7084	0.7090	0.7105
DNGR	0.6773	0.6830	0.6851	0.6878
GraRep	0.5971	0.6065	0.6101	0.6120
LINE	0.6561	0.6634	0.6658	0.6690
DeepWalk	0.6222	0.6302	0.6305	0.6289
APP	0.6467	0.6555	0.6581	0.6610
HOPE	0.6043	0.6100	0.6137	0.6119

Table 3.4: The AUC Scores of link sign prediction on the Slashdot dataset.

Algorithms	**% of labeled edges for training**			
	20	40	60	80
AsDNE	**0.8261**	**0.8274**	**0.8282**	**0.8283**
DNE-APP	0.8079	0.8094	0.8104	0.8103
SDNE	0.8033	0.8045	0.8052	0.8052
DNGR	0.7376	0.7397	0.7405	0.7402
GraRep	0.6102	0.6139	0.6161	0.6162
LINE	0.7692	0.7714	0.7725	0.7726
DeepWalk	0.6728	0.6758	0.6764	0.6761
APP	0.7051	0.7075	0.7083	0.7082
HOPE	0.6401	0.6448	0.6482	0.6506

Table 3.5: The AUC Scores of link sign prediction on the Epinions dataset.

Algorithms	**% of labeled edges for training**			
	20	40	60	80
AsDNE	**0.9121**	**0.9128**	**0.9129**	**0.9137**
DNE-APP	0.8884	0.8895	0.8894	0.8897
SDNE	0.8843	0.8855	0.8856	0.8862
DNGR	0.8553	0.8566	0.8567	0.8571
GraRep	0.7809	0.7826	0.7830	0.7846
LINE	0.8818	0.8827	0.8827	0.8837
DeepWalk	0.7485	0.7503	0.7513	0.7524
APP	0.7904	0.7923	0.7927	0.7932
HOPE	0.7580	0.7605	0.7616	0.7623

Thirdly, DeepWalk can preserve high-order proximities, while LINE only preserves the first-order and second-order proximities. As shown in Figure 3.2, for node classification, DeepWalk performs much better (i.e. achieves much higher Macro-F1 and Micro-F1 scores) than LINE. However, as shown in Tables 3.3, 3.4, and 3.5, in all the three networks, LINE would achieve much higher AUC score than DeepWalk for link sign prediction. In addition, in the experiments, we found that when the maximum step K becomes larger than 2, the prediction performance will be all degraded for DNGR, GraRep, DNE-APP, and AsDNE. This is because the high-order proximities are measured based on the transitive assumption which suggests that "the friend of my friend is likely to be my friend" [110], [111]. However, the negative links are not transitive in the signed networks, since both "the enemy of my enemy is my friend" and "the enemy of my enemy is my enemy" can be observed in the signed networks [112], [51]. Thus, the high-order proximities measured based on such a transitive assumption might be inaccurate in the signed networks and would lead to noises for negative link prediction. Conformably, we can see that SDNE which only captures the low-order proximity performs much better in link sign prediction, as compared to its performance in node classification. Thus, it reflects that preserving high-order proximities is rather important for learning network representations for node classification, while preserving low-order proximities yields better link sign prediction performance.

Next, we discuss the performance of the asymmetric network embedding algorithms. Both APP and DeepWalk are random-walk based embedding algorithms, however, APP always achieved much higher AUC score than DeepWalk in all the three directed networks. In addition, both GraRep and HOPE are matrix factorization based embedding algorithms. While HOPE can achieve much higher AUC score than GraRep in the Wiki and Slashdot networks (as shown in Tables 3.3 and 3.4). Moreover, SDNE, DNGR, and our DNE-APP and AsDNE models are all deep network embedding models, while the AsDNE model always performed significantly better than other deep network embedding baselines, in all the three networks. The better performance of the asymmetry-aware network embedding algorithms (i.e. APP, HOPE and AsDNE) w.r.t. their corresponding category of asymmetry-unaware benchmarks demonstrates that it is indeed necessary to capture the asymmetric proximities for learning network representations for link sign prediction in directed networks.

3.4.6 Parameter Sensitivity

In this subsection, we examine how different values of the parameters $\beta, \alpha, r, K, l, d$ would affect the performance of AsDNE. Figure 3.3 shows the parameter sensitivity of AsDNE w.r.t. multi-label node classification when 50% of labeled nodes were used for training on the Blogcatalog dataset. Figure 3.4 reports the sensitivity of AsDNE w.r.t. link sign prediction when 20% of labeled links were utilized for training on the Epinions dataset.

Parameter $\boldsymbol{\beta}$ denotes the ratio of penalty on the reconstruction errors of positive proximities (i.e. strong connections) over that of zero proximities (i.e. weak or unobserved connections). As shown in Figures. 3.3(a) and 3.4(a), $\beta > 1$ always leads to much higher Micro-F1 score for node classification and also much higher AUC score for link sign prediction than $\beta = 1$. This demonstrates that imposing larger penalty to make the SAE-Out and SAE-In focus more on reconstructing the strong connections than the weak or unobserved connections is highly effective for learning informative feature representations for both node classification and link sign prediction.

Parameter $\boldsymbol{\alpha}$ indicates the weight of pairwise constraint in the overall loss function. As shown in Figures 3.3(b) and 3.4(b), $\alpha > 0$ always yields better node classification and link sign prediction performance than $\alpha = 0$. This reflects that incorporating the pairwise constraints into SAE-Out and SAE-In can learn more

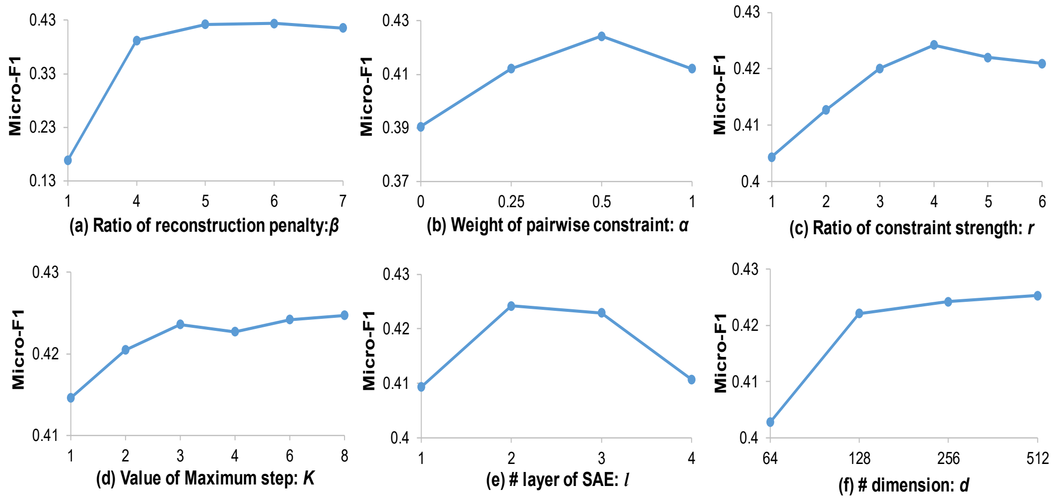


Figure 3.3: Sensitivity of the parameters $\beta, \alpha, r, K, l, d$ on the Micro-F1 score of AsDNE for multi-label node classification, when 50% of labeled nodes were used for training on the Blogcatalog dataset. The higher the Micro-F1 score, the better the performance.

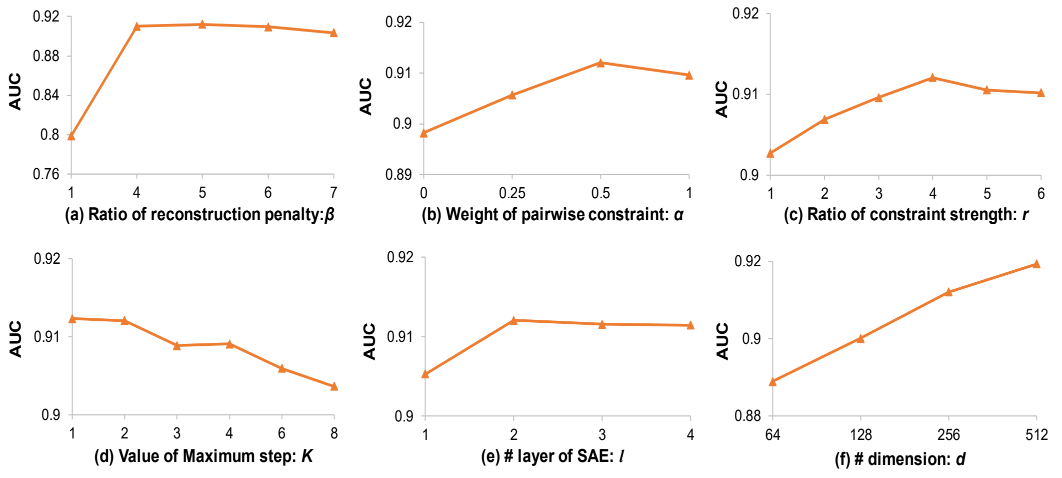


Figure 3.4: Sensitivity of the parameters $\beta, \alpha, r, K, l, d$ on the AUC score of AsDNE for link sign prediction, when 20% of labeled links were used for training on the Epinions dataset. The higher the AUC score, the better the performance.

informative network representations, as compared to the unsupervised SAEs. In addition, in AsDNE, we incorporate the parameter $\boldsymbol{r}$ to impose stronger constraints on the bi-directionally strongly connected node pairs so as to map them much closer than the unidirectionally strongly connected node pairs. As shown in Figures 3.3(c) and 3.4(c), $r > 1$ leads to higher Micro-F1 score for node classification and higher AUC score for link sign prediction than $r = 1$. This demonstrates the effectiveness of our designed asymmetry-aware pairwise constraints for different prediction tasks in the directed (i.e. Epinions) and even undirected (i.e. Blogcatalog) networks.

Parameter $\boldsymbol{K}$ refers to the maximum step of neighbor which is considered as similar to a target node. As shown in Figure 3.3(d), a larger value of K would lead to better node classification result. This indicates that capturing high-order proximities is rather effective for learning the feature representations for node classification. However, as shown in Figure 3.4(d), conversely, for link sign prediction, the AUC score would decrease as K increases, especially when K is larger than 2. This finding is consistent with the experimental results we have discussed previously. The high-order proximities are recursively measured based on the transitive assumption [110], [111], however, in the signed networks, the positive links are transitive while the negative links are not [112], [51]. Thus, if a K-step connection between two nodes contains any negative links, then the K-step pairwise proximities measured based on the transitive assumption would be inaccurate. Then, predicting the link signed labels based on such inaccurate high-order proximities would degrade the performance. However, if one aims to predict the existence of links in the unsigned networks without negative links, preserving the high-order proximities might be useful.

Parameter $\boldsymbol{l}$ indicates the number of layers in SAE-Out and SAE-In. We constructed 4 SAEs for both SAE-Out and SAE-In, with the number of layer differently set as 1, 2, 3, and 4, and fixing their deepest hidden dimensionality equally as 128. As shown in Figures 3.3(e) and 3.4(e), as compared with a shallow architecture (i.e. 1-layer SAE), a 2-layer SAE yields better node classification and also better link sign prediction performance. However, a deeper architecture more than 3 layers would yield even worse node classification results.

Parameter $\boldsymbol{d}$ is the dimensionality of the concatenated hidden vector representations. As shown in Figure 3.3(f), for node classification, the Micro-F1 score significantly increases when *d* is increased from 64 to 128. After that, the Micro-F1 score just slightly increases as *d* further increases. As shown in Figure 3.4(f), for link sign prediction, the AUC score keeps increasing as *d* increases.

3.5 Summary

In this chapter, we propose a deep network embedding model, AsDNE with asymmetric proximities preservation. AsDNE consists of two semi-supervised SAEs, i.e., SAE-Out and SAE-In, which are employed to learn the low-dimensional outward and inward vector representations, by considering each node as a source role and a target role, respectively, within its *K*-step network connections. To better capture the asymmetric relationships, we devise pairwise constraint to map the bi-directionally strongly connected nodes much closer than the unidirectionally strongly connected nodes. Extensive experimental results demonstrate that capturing asymmetric proximities can significantly improve prediction over both nodes and links, in both undirected and directed networks. Moreover, we found that capturing high-order proximities leads to better node classification results, while introducing noises to degrade link sign prediction. As the proposed AsDNE model can capture any *K*-th order proximities, it is flexible to learn informative network representations for both node classification and link sign prediction.

Some preliminary results of the proposed DNE-APP model were published in [11]. In addition, the comprehensive results of the proposed AsDNE model are currently under review in [42]. In the future, we can extend AsDNE for heterogenous network embedding, which not only considers the network structure but also utilizes the abundant content information associated with the nodes or edges for learning network representations.

Chapter 4
Deep Network Embedding in Signed Networks

4.1 Introduction

The vast majority of existing network embedding algorithms are only designed for unsigned networks, without considering the polarities of edges in the signed networks. A signed network contains both positive and negative links, where positive links indicate proximity or similarity, while negative links reflect dissimilarity or distance [113]. Recent studies [51], [52] have shown that the signed networks have properties substantially distinct from the unsigned networks. For example, in unsigned networks, the homophily effect [111] and social influence [110] theories suggest that the connected users tend to have similar preferences. However, such theories are not applicable to the signed networks due to the existence of negative links. For instance, in a signed network like Epinions[6], two negatively connected users reveal that they have rather opposite opinions instead of similar preferences. In addition, the transitivity property of unsigned networks which suggests that “the friend of my friend is likely to be my friend” is also not true for the negative links in the signed networks as both “the enemy of my enemy is my friend” and “the enemy of my enemy is my enemy” can be observed in the signed networks [112]. Due to the substantially distinct properties between signed networks and unsigned networks, existing network embedding algorithms designed for unsigned networks cannot be directly applied to the signed

[6] http://www.epinions.com/

networks. Thus, it is indeed necessary to design signed network embedding algorithms to capture the specific properties of the signed networks. “Structural balance” is one prevailing social property of the signed networks [114]. The balance theory states that “a network is balanced if and only if all the edges are positive; or all the nodes in the network can be grouped into two clusters where the edges within the same cluster are all positive while across different clusters are all negative” [115]. A weak balance theory [116] was proposed to generalize the original balance theory from two-way clustering to *k*-way clustering. Recently, Cygan *et al.* [117] further extended the structural balance theory as “the nodes connected with positive links should sit closer than those connected with negative links”.

Existing signed network embedding algorithms mostly employ the spectral techniques [113], [44], [118], [119] to embed the representation space of the original network into a low-dimensional space spanned by the top-*k* eigenvectors of the characteristic matrix associated with the given network. It has been shown that such spectral methods based on matrix decomposition techniques are with limited representation learning ability to capture the highly nonlinear properties of the complex network structure [34]. In addition, the spectral methods based on Eigen Value Decomposition (EVD) are computationally highly expensive, i.e., even the fastest implementation of EVD requires a super-quadratic computational complexity [4]. On the other hand, deep learning techniques have demonstrated powerful ability to learn more complex and non-linear feature representations in CV [120], [121], speech recognition [122] and NLP [48]. Thus, most recently, several promising deep network embedding algorithms [4], [5], [10], [11], [34] have been proposed to learn deep graph representations for unsigned networks. However, very little deep network embedding work exists for the signed networks.

In this work, we propose a deep network embedding with structural balance preservation (DNE-SBP) model to learn deep graph representations for the signed networks. A stacked auto-encoder (SAE) is employed to learn the nonlinear hidden node vector representations, by reconstructing the adjacency matrix of a given signed network. As the real-world signed networks are generally overwhelmingly positive [112], we impose larger penalty on the reconstruction errors of negative links so as to make the SAE focus more on reconstructing the scarce negative links as compared to the abundant positive links. In addition, we design the pairwise constraints to map each positively connected node pair closer to each other (i.e. having similar hidden vector representations), and to map each negatively connected node pair more far apart from each other (i.e. having rather different hidden vector representations), in the low-dimensional embedding space. Thus, the important structural balance property of the signed networks can be well captured by the embedding vector representations. Then, we apply vector-based machine learning algorithms on the node vector representations learned by DNE-SBP to carry out two important signed network mining tasks, namely, link prediction and community detection. The contributions of this work can be summarized as follows:

1) We propose a novel DNE-SBP model for signed network embedding, which leverages a semi-supervised SAE to learn the low-dimensional nonlinear graph representations;
2) By reconstructing the signed adjacency matrix, the learned hidden representations can capture positive, negative and unobserved network connections in the original network;
3) By designing the pairwise constraints to map the positively connected nodes nearer than the negatively connected nodes, the structural balance property of the signed networks can be well preserved by the embedding vector representations;

4) To deal with the highly imbalanced data in the real-world signed networks, we impose larger penalty and stronger pairwise constraint on negative links to make them have very distinctive embedding vector representations w.r.t. the positive links;
5) Extensive experiments on real-world datasets demonstrate the superiority of the proposed DNE-SBP model over the state-of-the-art network embedding algorithms for graph representation learning in the signed networks.

The rest of this chapter is organized as follows. Section 4.2 reviews the state-of-the-art signed network embedding algorithms. Section 4.3 introduces the detailed framework of DNE-SBP. Section 4.4 reports the experimental results of DNE-SBP for link sign prediction and community detection in three public real-world signed networks. Section 4.5 summaries this work.

4.2 Related Work

The state-of-the-art unsigned network embedding algorithms have been comprehensively reviewed in section 3.2. Here, we focus on reviewing the signed network embedding algorithms and the semi-supervised learning techniques.

4.2.1 Network Embedding for Signed Networks

Firstly, we review the spectral embedding algorithms designed for the signed networks. Kunegis *et al.* [113] introduced a signed Laplacian matrix by extending the conventional Laplacian matrix [123] designed for unsigned networks. Then, a signed network can be embedded into a d-dimensional space spanned by the top-d eigenvectors corresponding to the smallest eigenvalues of the signed Laplacian matrix. Chiang *et al.* [114] proposed a multi-level clustering framework based on the balanced normalized cut objective, which is proved to be mathematically equivalent to the weighed kernel k-means clustering objective. However, this framework only outputs the partitions of a network rather than an embedded map. Zheng *et al.* [44] further proposed a spectral embedding algorithm for the signed networks, by defining the simple normalized signed (SNS) graph Laplacian matrix and the balanced normalized signed (BNS) graph Laplacian matrix. Then, the top-*d* eigenvectors corresponding to the smallest non-zero eigenvalues of the SNS and BNS Laplacian matrices, respectively, were employed to construct a *d*-dimensional embedding space. Hsieh *et al.* [119] proposed to utilize Singular Value Projection to complete the adjacency matrix of a given signed network. Then, the top-*d* eigenvectors of the completed adjacency matrix were employed as the low rank embeddings. However, a recent study [118] has shown that the eigenvector encoding of the cluster structure does not necessarily correspond to the smallest eigenvalues. Thus, the standard spectral clustering techniques based on the top-*k* eigenvectors associated with the smallest eigenvalues might fail to guarantee the recovery of the ground truth cluster structures. To address this issue, Mercado *et al.* [118] proposed to use the geometric mean of the Laplacian matrices, instead of the arithmetic mean used by the standard spectral clustering methods. However, measuring the geometric mean of the Laplacian matrix is computationally expensive, thus limiting this method to be scaled to the large sparse networks.

Recently, Wang *et al.* [45] proposed a SiNE algorithm to utilize a deep learning framework to learn embedding representations for the signed networks, based on the extended structural balance theory [117]. Firstly, for each node v_i, a set of triplets $\{(v_i, v_j, v_k)|e_{ij} = 1, e_{ik} = -1\}$ were randomly sampled from a given signed network, where v_j and v_k denote a positive neighbor and a negative neighbor of v_i, respectively. Then, based on the sampled triplets, the goal of SiNE is to make the similarity between the hidden vector representations of a node and its positive neighbor larger than that between the node and its negative neighbor. Since SiNE learns the network representations based on the sampled triplets rather than the whole network connections, some information in the original network might be unavoidably missing. For example, for the nodes with a very large degree, sampling a limited number of triplets might fail to get enough information to learn informative feature representations. In addition, such sampled triplets only capture the observed connections, while ignoring all the unobserved connections. Thus, the network representations learned by SiNE would fail to easily distinguish the disconnected nodes from the connected ones. In contrast to SiNE, our proposed DNE-SBP model learns the network representation from the adjacency matrix, which captures not only the positive and negative connections, but also the unobserved connections. Thus, the network representations learned by DNE-SBP can not only distinguish the positively connected nodes from the negatively connected nodes, but also differentiate the connected nodes from the disconnected ones.

4.2.2 Semi-Supervised Learning

In the real-world applications, acquiring the fully labeled data is generally very expensive and time-consuming, while it is much easier to obtain the unlabeled data. Semi-supervised learning is an effective technique to leverage both the limited labeled data and the abundant unlabeled data to improve learning performance. The semi-supervised learning techniques can be grouped into two categories, i.e., semi-supervised classification [124], [125] and semi-supervised clustering [126], [127], [128]. On one hand, semi-supervised classification explores how to utilize a large amount of unlabeled samples as the extra training data to improve the classification performance, such as self-training and co-training [8], [9], [65]. On the other hand, semi-supervised clustering studies how to incorporate the prior information, such as pairwise constraints, to boost the clustering performance. For example, Klein *et al.* [127] proposed a semi-supervised clustering algorithm by incorporating the must-link (ML) and cannot-link (CL) pairwise constraints into the clustering process. The ML pairwise constraint indicates that the two instances are similar and should belong to the same cluster. While the CL pairwise constraint reflects that the two instances are dissimilar and cannot be assigned to the same cluster. Yu *et al.* [128] proposed a transitive closure based constraint propagation approach, which fully utilizes the ML and CL pairwise constraints in an ensemble framework for semi-supervised clustering. He *et al.* [126] developed a semi-supervised clustering algorithm to propagate the ML and CL pairwise constraints through multi-level random walks. In addition, in the recently proposed deep network embedding models [10], [11], [42], [5], pairwise constraints have been incorporated into SAEs to capture the proximities between different nodes, which are similar to the ML constraints in semi-supervised clustering. However, none of them have utilized the CL pairwise constraints to capture the dissimilarity between the nodes. To well capture the structural balance property of the signed networks, in the proposed DNE-SBP model, the ML and CL pairwise constraints have been incorporated to target for the positively and negatively connected node pairs, respectively.

4.3 Deep Network Embedding Model with Structural Balance Preservation

In this section, we introduce how a SAE is employed to reconstruct the signed adjacency matrix, how the pairwise constraints are designed and how the DNE-SBP model can be optimized. Algorithm 4.1 represents the framework of the DNE-SBP model. For clarity, we summary the frequently used notations and the corresponding descriptions in Table 4.1.

Table 4.1: Frequently used notations and descriptions in Chapter 4.

Notations	Descriptions
A	Signed adjacency matrix of the network
l	Number of layers in SAE
$d(k)$	Dimensionality of *k*-th hidden layer of SAE
$X^{(k)},\ \hat{X}^{(k)}$	Input and reconstructed matrices of *k*-th layer of SAE
$W_1^{(k)}$, $W_2^{(k)}$	Encoding and decoding weight matrices of *k*-th layer of SAE
$B_1^{(k)},\ B_2^{(k)}$	Encoding and decoding bias matrices of *k*-th layer of SAE
$H^{(k)}$	Hidden matrix representation learned by *k*-th layer of SAE
γ	Ratio of reconstruction penalty and pairwise constraint on negative links over that of positive links

Given a signed network $G = (V, E)$ with a set of nodes $V = \{v_i\}_{i=1}^{n}$ and a set of edges $E = \{e_{ij}\}$, the associated signed adjacency matrix $A \in R^{n\times n}$ is defined as below:

$$A_{ij} = \begin{cases} = 1, & if\ relation\ of\ (v_i, v_j)\ is\ positive \\ = -1, & if\ relation\ of\ (v_i, v_j)\ is\ negative \\ = 0, & if\ relation\ of\ (v_i, v_j)\ is\ unknown \end{cases}$$

Then, the signed adjacency matrix *A* can be broken into a positive part $A^{+} \in R^{n\times n}$ and a negative part $A^{-} \in R^{n\times n}$ as: $A_{ij}^{+} = max(A_{ij}, 0)$, $A_{ij}^{-} = -min(A_{ij}, 0)$, where $A_{ij}^{+},\ A_{ij}^{-} \geq 0$ represent the absolute weight of positive link and negative link, respectively.

4.3.1 Stacked Auto-Encoder

Next, we employ a SAE to reconstruct the signed adjacency matrix A to learn the nonlinear hidden vector representations for all the nodes in the signed network *G*. A SAE consists of *l* layers of basic auto-encoder is constructed as follows:

$$H^{(k)} = f\left(X^{(k)}\left(W_1^{(k)}\right)^T + B_1^{(k)}\right),\ k = 1, \dots, l \tag{4.1}$$

$$\hat{X}^{(k)} = f\left(\hat{H}^{(k)}\left(W_2^{(k)}\right)^T + B_2^{(k)}\right),\ k = l, \dots, 1 \tag{4.2}$$

where (1) and (2) represent the encoding and decoding process at the *k*-th layer of SAE, respectively. $H^{(k)} \in R^{n\times d(k)}$ denotes the hidden matrix representation learned by the *k*-th layer of SAE, *n* is the number of nodes in network *G*, and $d(k)$ represents the dimensionality of the *k*-th hidden layer of SAE. Specifically, the *i*-th row of $H^{(k)}$, i.e., $H_i^{(k)} \in R^{1\times d(k)}$ represents the hidden vector representation of node v_i, learned by the *k*-th layer of SAE. $X^{(k)} \in R^{n\times d(k-1)}$ denotes the input matrix of the *k*-th layer of SAE, $X^{(1)} = A$ and $X^{(k)} = H^{(k-1)}, \forall\, k = 2, \dots, l$ indicating that the hidden matrix representation learned by the (*k*-1)-th layer of SAE are utilized as the input matrix to the *k*-th layer of SAE. $W_1^{(k)} \in R^{d(k)\times d(k-1)}$ and $B_1^{(k)} \in R^{n\times d(k)}$ refer to the encoding weight and bias matrices associated with the *k*-th layer of SAE, respectively. In addition, $\hat{X}^{(k)} \in R^{n\times d(k-1)}$ and $\widehat{H}^{(k)} \in R^{n\times d(k)}$ indicate the reconstructed matrices of $X^{(k)}$ and $H^{(k)}$, respectively, where $\widehat{H}^{(l)} = H^{(l)}$ and $\widehat{H}^{(k)} = \hat{X}^{(k+1)}, \forall\, k = l-1, \dots, 1$. $W_2^{(k)} \in R^{d(k-1)\times d(k)}$ and $B_2^{(k)} \in R^{n\times d(k-1)}$ denote the decoding weight and bias matrices associated with the *k*-th layer of SAE, respectively. f is a non-linear activation function, in the proposed DNE-SBP model, the *tanh* function $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is employed as the activation function for each layer of SAE.

Then, by minimizing the reconstruction errors $\left\|\hat{A} - A\right\|_F^2$, we can learn the low-dimensional hidden vector representations which can best preserve the original network connections between different nodes. However, the connections in the real-world networks are generally rather sparse, yielding much more zero elements than non-zero elements in the adjacency matrix *A*. Then, directly reconstructing matrix *A* would make the SAE more tend to reconstruct the zero elements (i.e. unknown connections) than non-zero elements (i.e. observed connections). However, reconstructing observed connections should be more meaningful than reconstructing unobserved ones. To address the sparsity issue, we follow the approach in [10] to add a larger penalty on the reconstruction errors of non-zero input elements. Moreover, unlike SDNE [10] which lacks the consideration about negative links, the proposed DNE-SBP model targets for the signed networks. A recent study has shown that forming negative links requires higher cost than forming positive links [112], which leads to the overwhelmingly positive links in the real-world signed networks. To handle such highly imbalanced data in the signed networks, we design the following penalty matrix $P \in R^{n \times n}$ to make the SAE focus more on reconstructing the scarce negative links than the abundant positive links:

$$P_{ij} = \begin{cases} 1, & A_{ij} = 0 \\ \beta, & A_{ij} > 0 \\ \gamma * \beta, & A_{ij} < 0 \end{cases}$$

where $\beta \geq 1$ denotes the ratio of penalty on the reconstruction errors of observed connections (i.e. non-zero input elements) over that of unobserved connections (i.e. zero input elements); $\gamma \geq 1$ indicates the ratio of penalty on the reconstruction errors of negative links over that of positive links. By incorporating the penalty matrix P, we have the modified reconstruction error as:

$$\mathcal{J}_1^{(1)} = \frac{1}{2n} \left\| \left(\hat{A} - A \right) \odot P \right\|_F^2 \tag{4.3}$$

4.3.2 Pairwise Constraints

Next, we design a semi-supervised SAE by incorporating the ML and CL pairwise constraints to capture the extended structural balance property of the signed networks. For each k-th layer of SAE, the ML pairwise constraint $\mathcal{J}_2^{(k)}$ and the CL pairwise constraint $\mathcal{J}_3^{(k)}$ are devised as follows:

$$\mathcal{J}_2^{(k)} = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{+} \left\| H_i^{(k)} - H_j^{(k)} \right\|_2^2 \tag{4.4}$$

$$\mathcal{J}_3^{(k)} = -\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{-} \left\| H_i^{(k)} - H_j^{(k)} \right\|_2^2 \tag{4.5}$$

where $H_i^{(k)}, H_j^{(k)} \in R^{1\times d(k)}$ indicate the hidden vector representation of node v_i and v_j, respectively, learned by the k-th layer of SAE. On one hand, minimizing the ML pairwise constraint $\mathcal{J}_2^{(k)}$ which is equivalent to minimizing the positive ratio cut objective in signed network spectral clustering [114], we can push the positively connected nodes close to each other in the embedding space. On the other hand, via minimizing the CL pairwise constraint $\mathcal{J}_3^{(k)}$ which is equivalent to maximizing the negative ratio cut objective [114], we can pull the negatively connected nodes far away from each other in the embedding space. Moreover, to handle the highly imbalanced data (i.e. overwhelming positive links) in the signed networks, we also utilize the parameter γ to integrate $\mathcal{J}_2^{(k)}$ and $\mathcal{J}_3^{(k)}$, as follows:

$$\mathcal{J}_2^{(k)} + \gamma\mathcal{J}_3^{(k)} = \frac{1}{n}\left(Tr\left(\left(H^{(k)}\right)^T L^{+} H^{(k)}\right) - \gamma\, Tr\left(\left(H^{(k)}\right)^T L^{-} H^{(k)}\right)\right)$$

$$= \frac{1}{n} Tr\left(\left(H^{(k)}\right)^T L H^{(k)}\right) \tag{4.6}$$

where $L^{+} \in R^{n\times n}$ is the Laplacian matrix of A^{+}, i.e., $L^{+} = D^{+} - A^{+}$ and $D^{+} \in R^{n\times n}$ denotes the diagonal degree matrix of A^{+}, with the diagonal entries $D_{ii}^{+} = \sum_{j=1}^{n} A_{ij}^{+}$ representing the positive degree of node v_i. Similarly, $L^{-} = D^{-} - A^{-}$ is the Laplacian matrix of A^{-}, and D^{-} is the diagonal degree matrix of A^{-}, where $D_{ii}^{-} = \sum_{j=1}^{n} A_{ij}^{-}$ denoting the negative degree of node v_i. In addition, $L = L^{+} - \gamma L^{-}$, where $\gamma \geq 1$ indicates the ratio of weight of the CL pairwise constraint targeting for the negative links over that of the ML pairwise constraint targeting for the positive links. A larger value of γ would make DNE-SBP more tend to enlarge the distance between the negatively connected nodes, w.r.t.

narrowing the distance between the positively connected nodes. Note that when $\gamma = 1$ minimizing $\left(\mathcal{J}_2^{(k)} + \mathcal{J}_3^{(k)}\right)$ is analogous to minimizing the Rayleigh quotient of SNS graph Laplacian in signed network spectral embedding [44]. This reflects that our designed pairwise constraints are indeed able to capture the extended structural balance property of the signed networks. In addition, in both the pairwise constraint (4.6) and the penalty-modified reconstruction errors (4.3), we set the same value to the parameter γ to make the DNE-SBP model easily distinguish the scarce negative links from the abundant positive links.

Algorithm 4.1: DNE-SBP

Input: Signed adjacency matrix A; Parameters $l, \beta, \alpha, \gamma, \lambda$.

```
Set $X^{(1)} = A$
For k=1: $l$
    1. Leverage $X^{(k)}$ as input to $k$-th layer of SAE;
    2. Given $X^{(k)}$ and $A$, optimize $k$-th layer of SAE by finding $\theta^{(k)*} = \left\{W_1^{(k)*}, W_2^{(k)*}, B_1^{(k)*}, B_2^{(k)*}\right\} = arg\min_{\theta^{(k)}} \mathcal{J}^{(k)}$ via SGD;
    3. Leverage $\theta^{(k)*}$ to learn $H^{(k)}$;
    4. Set $X^{(k+1)} = H^{(k)}$;
End for
```

Output: The deepest hidden matrix representation, $H^{(l)}$.

4.3.3 Overall Loss Function

By integrating the reconstruction errors (4.3), the pairwise constraint (4.6), and a *L2*-norm regularization term $\mathcal{J}_4^{(k)} = \frac{1}{2}\left(\left\|W_1^{(k)}\right\|_F^2 + \left\|W_2^{(k)}\right\|_F^2\right)$, the overall loss function is defined as:

$$\mathcal{J} = \sum_{k=1}^{l} \mathcal{J}^{(k)} = \sum_{k=1}^{l}\left(\mathcal{J}_1^{(k)} + \alpha_k\left(\mathcal{J}_2^{(k)} + \gamma_k \mathcal{J}_3^{(k)}\right) + \lambda_k \mathcal{J}_4^{(k)}\right) \tag{4.7}$$

where l represents the number of layers in the SAE; $\mathcal{J}^{(k)}$ denotes the loss function of the k-th layer of SAE; $\alpha_k, \lambda_k > 0$ refer to the weights of the pairwise constraint and the regularization term at the k-th layer of SAE, respectively. By minimizing the overall loss function (4.7), we can learn the hidden node vector representations which not only preserve the original network connections, but also capture the structural balance property of the signed networks.

4.3.4 Optimization of DNE-SBP

Here, we explain how to obtain the optimized parameters minimizing the overall loss function (4.7). Firstly, for each k-th layer basic auto-encoder in the SAE, with the help of back-propagation algorithm, the "error" terms of its output layer $\delta_3^{(k)} \in R^{n\times d(k-1)}$ and hidden layer $\delta_2^{(k)} \in R^{n\times d(k)}$ can be computed, respectively, as follows:

$$\delta_3^{(k)} = \begin{cases} (\hat{A} - A) \odot P \odot P \odot f'\left(Z_3^{(k)}\right), k = 1 \\ \left(\hat{X}^{(k)} - X^{(k)}\right) \odot f'\left(Z_3^{(k)}\right), k \geq 2 \end{cases} \tag{4.8}$$

$$\delta_2^{(k)} = \left(\delta_3^{(k)} W_2^{(k)} + \alpha_k (L + L^T) H^{(k)}\right) \odot f'\left(Z_2^{(k)}\right) \tag{4.9}$$

where $Z_3^{(k)} = H^{(k)}\left(W_2^{(k)}\right)^T + b_2^{(k)}$, $Z_2^{(k)} = X^{(k)}\left(W_1^{(k)}\right)^T + b_1^{(k)}$ and f' denotes the derivative of the activation function.

Next, the partial derivatives w.r.t. the encoding weight $W_1^{(k)}$, decoding weight $W_2^{(k)}$, encoding bias $B_1^{(k)}$, and decoding bias $B_2^{(k)}$, can be computed, respectively, as follows:

$$\frac{\partial \mathcal{J}^{(k)}}{\partial W_1^{(k)}} = \frac{1}{n}\left(\delta_2^{(k)}\right)^T X^{(k)} + \lambda_k W_1^{(k)} \tag{4.10}$$

$$\frac{\partial \mathcal{J}^{(k)}}{\partial W_2^{(k)}} = \frac{1}{n}\left(\delta_3^{(k)}\right)^T H^{(k)} + \lambda_k W_2^{(k)} \tag{4.11}$$

$$\frac{\partial \mathcal{J}^{(k)}}{\partial B_1^{(k)}} = \delta_2^{(k)} / n \tag{4.12}$$

$$\frac{\partial \mathcal{J}^{(k)}}{\partial B_2^{(k)}} = \delta_3^{(k)}/n \tag{4.13}$$

To minimize the loss function $\mathcal{J}^{(k)}$, we use stochastic gradient descent (SGD) to update the parameters as: $W_1^{(k)} = W_1^{(k)} - \eta_k \frac{\partial \mathcal{J}^{(k)}}{\partial W_1^{(k)}}$; $W_2^{(k)} = W_2^{(k)} - \eta_k \frac{\partial \mathcal{J}^{(k)}}{\partial W_2^{(k)}}$; $B_1^{(k)} = B_1^{(k)} - \eta_k \frac{\partial \mathcal{J}^{(k)}}{\partial B_1^{(k)}}$; $B_2^{(k)} = B_2^{(k)} - \eta_k \frac{\partial \mathcal{J}^{(k)}}{\partial B_2^{(k)}}$, where η_k denotes the learning rate of the k-th layer basic auto-encoder in the SAE. Next, to optimize a SAE consists of multiple layers of basic auto-encoder, we adopt a greedily layer-wised training approach, as in [34], [129].

4.4 Experiments

4.4.1 Datasets

We evaluated the graph representation learning performance of the proposed DNE-SBP model for link sign prediction and community detection in three real-world signed networks, namely Epinions, Slashdot and Wiki. The Epinions dataset [105] is a "who trust whom" online social network generated from the Epinions site, where one user can "trust" (positive) or "distrust" (negative) another. The Slashdot dataset [104] is a signed social network extracted from the technology news site Slashdot, where users can form the relationships as friends (positive) or foes (negative). The Wiki dataset [103] is extracted from the Wikipedia site, which describes the votes "for" (positive) and "against" (negative) the other in elections. In the experiments, we used the full Wiki dataset, and extracting 7000 nodes with the largest degree and retaining all the edges between the selected nodes, from the original Epinions and Slashdot datasets. Table 4.2 shows some statistics of the three datasets. Among the three networks, Wiki is the sparsest one, Slashdot is second sparsest, while Epinions is the densest.

Table 4.2: Statistics of the signed networked datasets.

Datasets	**Epinions**	**Slashdot**	**Wiki**
# Users	7000	7000	7118
# Links	451149	238029	103675
# Positive links	404006	181354	81318
# Negative links	47143	56675	22357

Table 4.3: Layer configuration of the SAE in three datasets for link sign prediction and community detection.

Datasets	**Dimensionality of each layer of SAE**	
	Link Sign Prediction	**Community Detection**
Wiki	7118-256-64	7118-512-256-128-64
Slashdot	7000-256-64	7000-512-256-128-64
Epinions	7000-256-64	7000-512-256-128-64

4.4.2 Implementation Details

For link sign prediction, we built a two-layer SAE in the DNE-SBP model. The layer configurations of the SAEs for the three datasets are shown in Table 4.3. For example, for Wiki dataset, the layer configuration "7118-256-64" indicates that the 1-st layer basic auto-encoder and the 2-nd layer basic auto-encoder in the SAE were configured with the dimensionality of each layer as 7118-256-7118 and 256-64-256, respectively. In addition, the batch sizes of the 1-st layer and the deeper layers of SAE were set as 500 and 100, respectively, and the learning rates were set as $\eta_1 = 0.025$ and $\eta_k = 0.015, \forall k \geq 2$. Also, we set the weight of *L2*-norm regularization as $\lambda_1 = 0.05$ and $\lambda_k = 0.1, \forall k \geq 2$ in both Epinions and Slashdot datasets; while set $\lambda_1 = 0.05$ and $\lambda_k = 0.25, \forall k \geq 2$ in the Wiki dataset. In addition, we set the weight of pairwise constraint at the 1-st layer and the deeper layers of SAE as: $\alpha_1 = 16, 14, 10$ and $\alpha_k = 0.4, 0.2, 0.2, \forall k \geq 2$ for the Wiki, Slashdot and Epinions datasets, respectively. The ratio of penalty on the reconstructing errors of non-zero input elements over that of zero input elements were set as β= 25, 25 and 10 in the Wiki, Slashdot and Epinions datasets, respectively. Besides, we set the ratio of penalty on the reconstructing errors of negative links over that of positive links, and the ratio of the weight of the CL constraint over that of the ML constraint as $\gamma_1 = floor\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{+}}{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{-}}\right)$, where $floor(x)$ rounds x to the nearest integer less than or equal to x. While we set $\gamma_k = 1, \forall\, k \geq 2$, because after the first layer embedding, the larger penalty and stronger pairwise constraint for negative links have already been imposed to learn the hidden representations.

On the other hand, for community detection, a four-layer SAE was built for the three datasets, respectively, as shown in Table 4.3. The parameter settings for community detection are similar to those introduced for link sign prediction. Here, we only report the differences. Firstly, in contrast to link sign prediction, we set a larger batch size of 1000 for each layer of SAE. Secondly, we assigned a larger weight to the pairwise constraint at the deeper layers of SAE, i.e., $\alpha_k = 1.5, \forall\, k \geq 2$. This is because the structural balance theory [115], [116] was originally proposed for network clustering, thus it should be more important and necessary to preserve the structural balance property for the community detection task, by assigning larger weight to the pairwise constraint. In addition, for both link sign prediction and community detection, we employed the deepest hidden vector representations learned by the last layer of SAE as the node vector representations, with the dimensionality of d=64.

4.4.3 Analysis of Embedding Learned by DNE-SBP

Here, we analyze whether the network representations learned by the proposed DNE-SBP model can preserve the extended structural balance property of the signed networks, i.e., whether the positively connected nodes are sitting closer than the negatively connected nodes in the embedding space. In this regard, we adopt three distance measures introduced in [44], namely average edge ratio

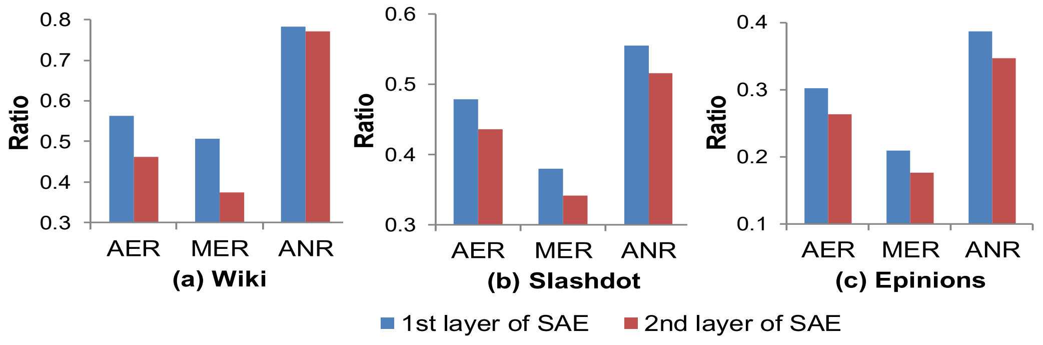


Figure 4.1: The AER, MER and ANR ratios of the distances between the positively connected nodes over that between the negatively connected nodes in the embedding spaces learned by the 1-st layer and the 2-nd layer of SAE in the DNE-SBP model.

(AER), median edge ratio (MER) and average node ratio (ANR). AER is defined as the ratio of the average embedded distance between the positively connected nodes over that between the negatively connected nodes: $\frac{(\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{+}d_{ij})/\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{+}}{(\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{-}d_{ij})/\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{-}}$, where d_{ij} indicates the Euclidean distance between the hidden vector representations of node v_i and v_j. MER is computed as the ratio of the median of the embedded distance between the positively connected nodes over that between the negatively connected nodes: $\frac{median\left(d_{ij}|A_{ij}^{+}>0\right)}{median\left(d_{ij}|A_{ij}^{-}>0\right)}$. ANR is computed as the ratio of the average embedded length of positive links over that of negative links, from the perspective of nodes: $\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}A_{ij}^{+}d_{ij}/D_{ii}^{+}\right)/n_p}{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}A_{ij}^{-}d_{ij}/D_{ii}^{-}\right)/n_n}$, where n_p and n_n indicate the number of nodes having at least one positive link and having at least one negative link, respectively. As shown in Figure 4.1, all the three ratios are smaller than 1 in the three datasets, indicating that the embeddings learned by DNE-SBP can actually preserve the extended structural balance property. Moreover, we can observe that the ratios measured in the embedding space learned by the 2nd layer of SAE were smaller than that learned by the 1st layer of SAE. This indicates that the hidden representations learned by the deeper layer of SAE can better satisfy the extended structural balance condition.

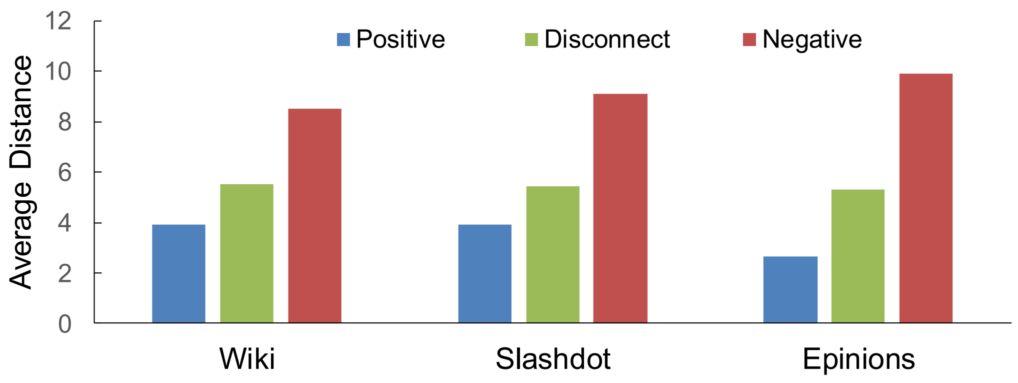


Figure 4.2: The average distances between positively connected nodes, negatively connected nodes and disconnected nodes in the embedding space learned by the 2-nd layer of SAE in the DNE-SBP model.

Moreover, we further compare the distances between the node pairs with positive connections, negative connections, and unobserved connections in the low-dimensional embedding space learned by the proposed DNE-SBP model. As shown in Figure 4.2, the positively connected nodes would be mapped much closer than the negatively connected nodes. In addition, the disconnected nodes would be mapped closer than negatively connected nodes while more far apart than positively connected nodes.

4.4.4 Baselines

The following state-of-the-art network embedding algorithms were employed to benchmark against the proposed DNE-SBP model.

1) **SL** [113]: It is a spectral clustering algorithm, with a signed Laplacian matrix defined as $\bar{L} = \bar{D} - A$, where $\bar{D}_{ii} = \sum_{j=1}^{n} |A_{ij}|$ indicates the sum of positive and negative degree of node v_i. The top-d eigenvectors of $\bar{L}$ are selected as the node vector representations.
2) **SNS** [44]: It is a spectral embedding algorithm defining the SNS Laplacian matrix as $L_{SNS} = \bar{D}^{-1}(D^{+} - D^{-} - A)$. The top-$d$ eigenvectors of L_{SNS} are selected as the node vector representations.

3) **BNS** [44]: It is a spectral embedding algorithm with the BNS Laplacian matrix defined as $L_{BNS} = \overline{D}^{-1}(D^{+} - A)$. The top-$d$ eigenvectors of L_{BNS} are selected as the node vector representations.
4) **SiNE** [45]: It is a deep network embedding model designed to preserve the extended structural balance property of the signed networks. It randomly samples a set of triplets containing the positive and negative neighbors of each node in the given network. Then, it employs a deep learning framework to learn the node vector representations based on the sampled triplets.
5) **SDNE** [10]: It is a deep network embedding model designed for unsigned networks. It employs a semi-supervised SAE to map the connected node pairs close to each other, without differentiating the positive and negative links.

Since the structural balance theory is naturally defined for undirected networks [51], we evaluated all the comparing algorithms in undirected signed networks. To guarantee the best performance of SiNE, we used the default settings in [45], i.e., building a 3-layer neural network with the dimensionality of each layer and the node vector representations as *d*=20. For other baselines in link sign prediction, we used the same dimensionality of node vector representation as in our DNE-SBP model, i.e., *d*=64. In addition, note that the spectral embedding algorithms, i.e., SL, SNS and BNS, assume that the top-k eigenvectors encode the corresponding k-way clustering structure [113], [44]. Thus, for the community detection task, the dimensionality of the node vector representations learned by these spectral embedding algorithms were set as equal to the number of clusters in the given network, i.e. *d*=k.

4.4.5 Experimental Results

In this subsection, we report the experimental results of the network embedding algorithms for link sign prediction and community detection in three real-world signed networks.

4.4.5.1 Link Sign Prediction

For a given signed network, we firstly randomly sampled a fraction $f\%$ of the observed (positive and negative) connections from its signed adjacency matrix. Then, we employed such sampled edges as the training set and the remaining $(1-f)\%$ of edges as the testing set. Based on all the edges in the training set, one can construct a training signed adjacency matrix A, where $A_{ij}=1$ if there is one positive edge between v_i and v_j in the training set; $A_{ij}=-1$ if one negative edge connecting v_i and v_j is in the training set; and $A_{ij}=0$ if there is no edge between v_i and v_j in the training set. Next, giving the training signed adjacency matrix as the input, the network embedding algorithms can learn the low-dimensional node vector representations. Based on the node vector representations, four types of edge feature representations were built as follows:

L1: $H(e_{ij}) = |H_i - H_j|$

L2: $H(e_{ij}) = |H_i - H_j|^2$

Had: $H(e_{ij}) = H_i \odot H_j$

Avg: $H(e_{ij}) = (H_i + H_j)/2$

where $H(e_{ij})$ indicates the feature vector of edge e_{ij}; H_i and H_j denote the low-dimensional feature vector representation of node v_i and v_j, respectively. A LR classifier was trained based on the edge vector representations and the (positive and negative) edge labels in the training set. Next, the classifier was employed to predict the signed labels of edges in the testing set. As the signed network datasets are overwhelmingly positive, directly evaluating the accuracy on the highly imbalanced dataset will be misleading. Thus, following [107], [108],

Table 4.4: AUC and AP of link sign prediction on the Wiki dataset. The highest AUC and AP scores among all the comparing methods are shown in Boldface. * and ** indicate statistically superior performance to SiNE (with its best suitable edge feature) at level of (0.05, 0.01) using a paired t-test.

Algorithm	**Edge Feature**	**AUC**				**AP**			
		% of observed links for training				**% of observed links for training**			
		20	40	60	80	20	40	60	80
DNE-SBP	**L1**	0.7337	0.8184	0.8431	0.8517	0.4662	0.5987	0.6346	0.6389
	L2	0.7161	0.7991	0.8319	0.8431	0.4604	0.5805	0.6238	0.6321
	Had	0.7937	**0.8459**	**0.8626** **	**0.8681** *	0.5210	**0.6268** *	**0.6595** **	**0.6642** **
	Avg	0.8038	0.8454	0.8562	0.8591	0.5311	0.6057	0.6267	0.6305
SiNE	**L1**	0.6143	0.6700	0.6818	0.7021	0.3005	0.3390	0.3601	0.3671
	L2	0.6775	0.7092	0.7155	0.7001	0.3677	0.3952	0.3965	0.3682
	Had	0.7961	0.8215	0.8053	0.8093	0.5212	0.5396	0.5047	0.5233
	Avg	**0.8080**	0.8399	0.8537	0.8644	**0.5492**	0.5863	0.6130	0.6305
SL	**L1**	0.5726	0.5858	0.5900	0.6024	0.2771	0.2777	0.2795	0.2857
	L2	0.5118	0.5046	0.5026	0.5012	0.2319	0.2205	0.2177	0.2164
	Had	0.5000	0.5000	0.5000	0.5000	0.2156	0.2133	0.2142	0.2141
	Avg	0.5001	0.5003	0.5038	0.5040	0.2232	0.2172	0.2162	0.2157
SNS	**L1**	0.6747	0.6928	0.7011	0.6951	0.3500	0.3665	0.3768	0.3734
	L2	0.6497	0.6595	0.6571	0.6703	0.3367	0.3508	0.3558	0.3597
	Had	0.5106	0.5028	0.4998	0.5018	0.2663	0.2529	0.2487	0.2635
	Avg	0.5461	0.5252	0.5156	0.5141	0.2784	0.2741	0.2543	0.2513
BNS	**L1**	0.6541	0.6440	0.6505	0.6536	0.4073	0.3918	0.4044	0.4000
	L2	0.5149	0.5078	0.5042	0.5025	0.2491	0.2323	0.2252	0.2199
	Had	0.5000	0.5000	0.5000	0.5000	0.2146	0.2128	0.2137	0.2135
	Avg	0.5122	0.5108	0.5037	0.5051	0.2504	0.2442	0.2328	0.2227
SDNE	**L1**	0.6032	0.6232	0.6353	0.6374	0.3027	0.3475	0.3899	0.4072
	L2	0.6030	0.6223	0.6307	0.6380	0.3121	0.3594	0.3893	0.4090
	Had	0.6494	0.6746	0.6825	0.6804	0.3456	0.3935	0.4223	0.4332
	Avg	0.6507	0.6744	0.6940	0.6977	0.3365	0.3756	0.4228	0.4464

[45], we adopt the AUC metric to evaluate the link sign prediction performance, which is insensitive to the imbalanced data. The higher the AUC score, the better the link sign prediction performance. In addition, we employ AP [130] as another metric. To better reflect the link sign prediction performance in such highly

Table 4.5: AUC and AP of link sign prediction on the Slashdot dataset.

Algorithm	Best Edge Feature	AUC				AP			
		% of observed links for training				% of observed links for training			
		20	40	60	80	20	40	60	80
DNE-SBP	**Had**	**0.8473** **	**0.8855** **	**0.8979** **	**0.9058** **	**0.6821** **	**0.7571** **	**0.7808** **	**0.7957** **
SiNE	**Avg**	0.8192	0.8467	0.8572	0.8623	0.5986	0.6427	0.6645	0.6713
SL	**Had**	0.7824	0.8350	0.8500	0.8528	0.5117	0.6065	0.6432	0.6363
SNS	**L1**	0.7654	0.7726	0.7215	0.7712	0.5213	0.5484	0.5191	0.5745
BNS	**L1**	0.7761	0.8023	0.8167	0.8292	0.5250	0.5864	0.6278	0.6544
SDNE	**Avg**	0.6672	0.7213	0.7429	0.7503	0.3813	0.4339	0.4627	0.4726

Table 4.6: AUC and AP of link sign prediction on the Epinions dataset.

Algorithm	Best Edge Feature	AUC				AP			
		% of observed links for training				% of observed links for training			
		20	40	60	80	20	40	60	80
DNE-SBP	**Had**	**0.9137** **	**0.9288** **	**0.9336** **	**0.9373** **	**0.7280** **	**0.7686** **	**0.7846** **	**0.7925** **
SiNE	**Avg**	0.9009	0.9127	0.9135	0.9114	0.6201	0.6432	0.6481	0.6491
SL	**Had**	0.8811	0.8991	0.9076	0.9031	0.5911	0.6618	0.6932	0.6842
SNS	**L1**	0.8381	0.7092	0.6792	0.8074	0.5417	0.3524	0.3905	0.5221
BNS	**L2**	0.8494	0.8725	0.8817	0.8908	0.5170	0.5889	0.6311	0.6617
SDNE	**Avg**	0.7248	0.7515	0.7635	0.7620	0.2622	0.2917	0.3101	0.3033

imbalanced datasets, we calculated the AP of the scarce class (i.e. negative links). The higher the AP, the better the link sign prediction performance. Table 4.4 reports the AUC and AP scores of all the comparing algorithms with four types of edge representation in the Wiki dataset. Tables 4.5 and 4.6 report the highest AUC and AP scores each algorithm can achieve with its best suitable edge representation, in the Slashdot and Epinions datasets. The reported AUC and AP scores for each comparing algorithm were averaged over the same 5 random splits.

As shown in Table 4.4, when the given network is rather sparse, e.g., just giving 20% of observed links in the sparest Wiki network, the SiNE algorithm with the Avg edge representation can achieve the highest AUC and AP scores among all the comparing algorithms. Except that, the proposed DNE-SBP model with the Had edge representation always achieves the highest AUC and AP scores in the three signed networks, as shown in Tables 4.4, 4.5 and 4.6. This could be explained by the fact that SiNE learns the network representations based on the sampled triplets [45]. Then, if the given network is very dense (i.e. having a large number of possible triplets), a limited number of samples would unavoidably lose some information. Thus, SiNE would perform worse as the given network is denser. In contrast to SiNE, our DNE-SBP model learns the network representations based on the adjacency matrix of a given network. Thus, we can take advantage of the whole network connections to learn more informative network representations. In addition, we observe that both DNE-SBP and SiNE outperform the spectral embedding algorithms in the three signed networks. This demonstrates the higher effectiveness of deep learning techniques for graph representation learning, as compared to the linear matrix decomposition methods.

Secondly, we discuss the performance of the spectral embedding algorithms, i.e., SL, SNS and BNS. We can see that in the densest Epinions dataset (shown in Table 4.6), SL with the Had edge representation can achieve the highest AUC scores among all the spectral embedding baselines. However, conversely, on the sparsest Wiki dataset (shown in Table 4.4), SL always achieved the lowest AUC and AP scores, no matter what percentage of observed links were used for training. This reflects that SL is rather unsuitable for the sparse networks, despite of the fact that it can perform much better in the dense networks. In addition, as shown in Table 4.4, SNS with the L1 edge representation can achieve the highest AUC scores among all the spectral embedding methods on the Wiki dataset. While BNS can achieve both higher AUC and AP scores than SNS on both Slashdot and Epinions datasets, as shown in Tables 4.5 and 4.6. This reflects that BNS performs better in the dense networks, while showing greater challenge when dealing with the sparse networks.

Next, let us continue to evaluate the performance of the unsigned network embedding algorithm, i.e., SDNE. As shown in Tables 4.5 and 4.6, SDNE always achieved much lower AUC and AP scores than all the signed network embedding algorithms on Slashdot and Epinions datasets. This is because that SDNE aims to map the connected node pairs closer to each other based on the social theories which suggest that the connected nodes tend to have similar preferences [111], [110]. However, such theories are not applicable for the signed networks where negative links indicate dissimilarity while positive links indicate similarity. Thus, directly applying the unsigned network embedding algorithm to the signed networks would fail to capture the important structural balance property [115], [116], [117] which requires the positively connected nodes to sit closer than the negatively connected ones.

Moreover, we can observe that all the network embedding algorithms achieve the highest AUC scores on the Epinions dataset, while the lowest AUC scores on the Wiki dataset. These could be explained by the previous findings in [107], [131] that the structural balance condition is most satisfied in the Epinions network, while least satisfied in the Wiki network. Thus, it should be easier to predict the link signed labels based on the structural balance property for the Epinions dataset.

4.4.5.2 Community Detection

For community detection, a *k*-means algorithm was run on the node vector representations learned by each network embedding algorithm to get the clustering results. Unlike community detection in unsigned networks, the objective of signed network clustering is to group the nodes into *k* clusters, where the connections between the nodes within the same cluster should be mostly positive, while the connections between the nodes belonging to different clusters should be mostly negative [51], [114]. To evaluate the performance of signed network community detection, we adopt the **"error rate"** metric, which is widely utilized in the related literature [114], [119], [132]. The error rate E is defined as the sum of the number of negative edges within the same cluster and the number of positive edges between different clusters, normalized by the total number of edges in the network:

$$E = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{-}\delta(c_i, c_j) + A_{ij}^{+}\left(1 - \delta(c_i, c_j)\right)}{\sum_{i=1}^{n}\sum_{j=1}^{n}|A_{ij}|}$$

where c_i , c_j indicate the community node v_i and v_j belonging to, respectively. If v_i and v_j are assigned to the same community, then $\delta(c_i, c_j) = 1$; otherwise, $\delta(c_i, c_j) = 0$. The lower the error rate, the better the signed network clustering performance.

Firstly, as shown in Table 4.7, the proposed DNE-SBP model always outperformed all the baselines (i.e. achieved the lowest error rates) in all the three datasets. This again proves that the network representations learned by our DNE-SBP model can well capture and preserve the structural balance property of signed networks. In addition, among all the spectral embedding algorithms, BNS achieved the lowest error rates in all the three datasets. SNS achieved slightly lower error rates than SL in the Slashdot and Epinions networks. Note that the objective function of SNS is analogue to the pairwise constraint designed in our DNE-SBP model. However, when learning the network representations, SNS employs EVD to linearly project the original network into a low-dimensional embedding space. In contrast, we take advantage of deep learning technique to learn non-linear network representations. The significant outperformance of our DNE-SBP model over SNS reflects that the deep learning techniques possess more powerful feature representation learning ability to capture the complex underlying network structure.

Table 4.7: The error rates (%) of *k*-way Clustering in three signed networks. The lowest error rates among all the comparing algorithms are shown in Boldface.

Dataset	**Algorithms**	**Number of cluster *k***								
		2	3	4	5	6	7	8	9	10
Wiki	**DNE-SBP**	**15.64**	**15.66**	**15.74**	**15.71**	**15.71**	**15.71**	**15.71**	**15.70**	**15.69**
	SL	21.94	21.94	21.94	21.94	21.94	21.93	21.93	21.93	21.93
	SNS	21.94	21.94	21.94	21.94	21.94	21.94	21.94	21.94	21.94
	BNS	21.86	21.80	21.77	21.76	21.74	21.72	21.72	21.71	21.70
	SiNE	33.48	53.57	58.62	61.14	62.01	63.30	66.85	67.74	69.12
	SDNE	26.01	33.67	42.16	42.69	44.97	55.81	56.03	55.70	55.84
Slashdot	**DNE-SBP**	**18.54**	**18.19**	**18.15**	**18.17**	**18.15**	**18.23**	**18.22**	**18.23**	**18.23**
	SL	25.38	25.39	25.39	25.39	25.39	25.39	25.39	25.40	25.40
	SNS	25.36	25.24	25.21	25.20	25.20	25.06	25.05	25.05	24.90
	BNS	25.19	25.15	25.00	24.81	24.74	24.84	24.81	24.44	23.49
	SiNE	40.38	47.65	51.18	57.00	55.89	56.16	58.37	62.35	63.72
	SDNE	48.91	55.75	57.78	61.29	62.34	63.65	64.00	65.35	66.09
Epinions	**DNE-SBP**	**7.95**	**8.25**	**8.20**	**8.29**	**8.30**	**8.33**	**8.31**	**8.31**	**8.32**
	SL	12.20	12.20	12.20	12.20	12.20	12.20	12.21	12.21	12.21
	SNS	11.73	12.08	12.10	11.93	11.97	11.78	11.93	12.05	11.90
	BNS	9.01	9.01	9.77	9.02	9.57	9.54	9.50	9.54	9.55
	SiNE	40.70	48.03	56.43	61.87	66.58	68.21	69.64	70.82	71.61
	SDNE	36.76	38.75	43.19	48.35	49.26	50.02	52.89	56.34	57.01

Secondly, we can see that SiNE achieved the highest error rates among all the comparing algorithms in the Wiki and Epinions datasets. As the number of cluster *k* increases, SiNE would perform even worse. It might be explained by the fact that SiNE learns network representations based on the triplets only sampled from the observed connections, while all the unobserved connections in the original network have been ignored. Thus, the network representations learned by SiNE would fail to distinguish the disconnected nodes from the connected nodes, which is not desirable for community detection. In addition, the sampled triplets only capture the local neighborhood structure, however, community detection also requires the global structural information. Hence, SiNE performs rather badly when learning the network representations for signed network community detection.

Moreover, we can see that SDNE performed significantly worse (i.e. achieved much higher error rates) than all the spectral embedding algorithms developed for the signed networks. This again confirms that the unsigned network embedding algorithm fails to learn informative network representations for the signed networks. Thus, it is indeed necessary to design the network embedding algorithms targeting for the signed networks, which can well capture the important structural balance property so as to easily distinguish the negative links from the positive links.

4.4.6 Parameter Sensitivity

In this subsection, the sensitivities of the parameters $\beta, \gamma, \alpha, l, d$ on the performance of DNE-SBP are reported. Figures 4.3 and 4.4 show the parameter sensitivity of DNE-SBP for link sign prediction and community detection, respectively.

Parameter $\boldsymbol{\beta}$ denotes the ratio of the penalty on the reconstruction errors of non-zero input elements over that of zero input elements. As shown in Figures 4.3(a) and 4.4(a), $\beta > 1$ leads to much better link sign prediction (i.e. higher AUC score) and also much better community detection (i.e. lower error rate) than β=1. This demonstrates that it is highly effective to assign larger penalty to make the SAE more prone to reconstruct the observed connections than unknown connections.

Parameter $\boldsymbol{\gamma}$ specifies the ratio of penalty on the reconstruction errors of negative links over that of positive links, and the ratio of the weight of the pairwise constraint targeting for the negative links over that of the positive links. As shown in Figure 4.3(b), for link sign prediction, $\gamma > 1$ can achieve much higher AUC scores than $\gamma = 1$. This demonstrates the significant effectiveness

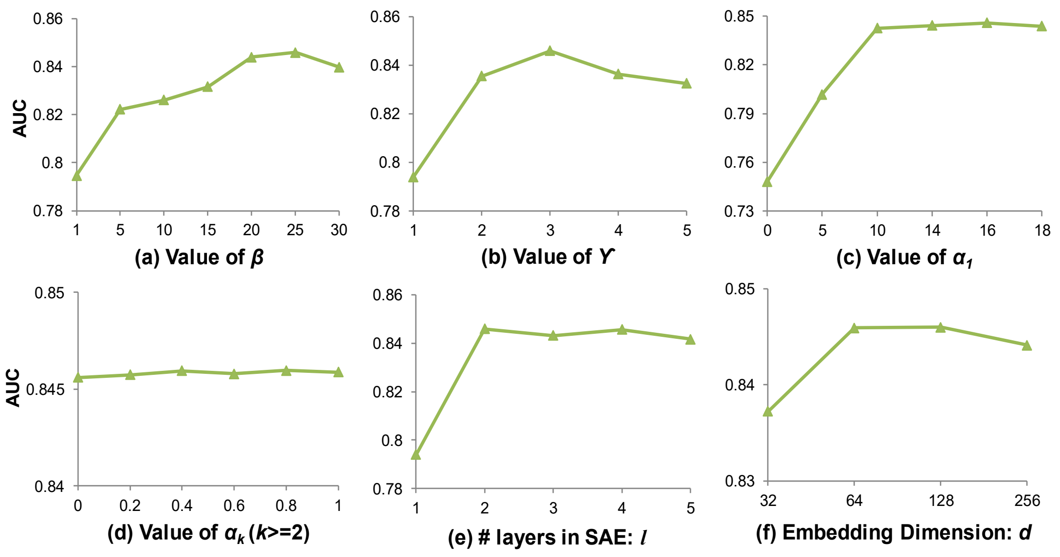


Figure 4.3: Sensitivity of the parameters $\beta, \gamma, \alpha, l, d$ on the AUC score of DNE-SBP (with the Had edge representation) for link sign prediction, when 40% of observed links were used for training on the Wiki dataset. The higher the AUC score, the better the performance.

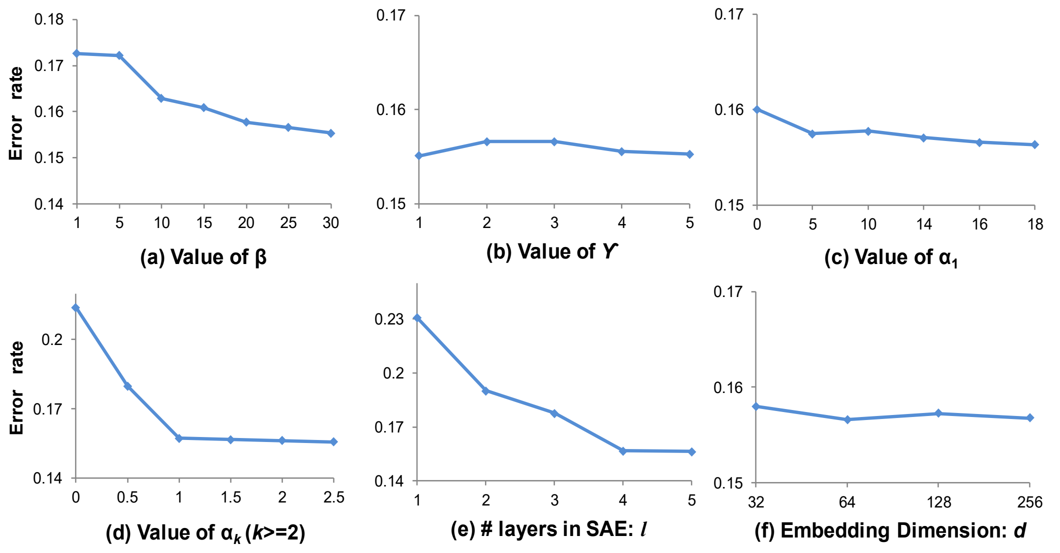


Figure 4.4: Sensitivity of the parameters $\beta, \gamma, \alpha, l, d$ on the error rate of DNE-SBP for 3-way signed network community detection on the Wiki dataset. The lower the error rate, the better the performance.

and necessity of imposing larger penalty and stronger pairwise constraint on the scarce negative links to handle the highly imbalanced data (i.e. overwhelming positive links) in the real-world signed networks. Also, when $\gamma \leq 3$, a higher value of γ would lead to higher AUC score, while after that, the AUC scores will slightly decrease. Note that on the Wiki dataset, the ratio of the number of positive links over that of negative links is 3.63. Thus, it indicates that setting $\gamma = floor\left(\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{+}/\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}^{-}\right)$ is reasonable for DNE-SBP to achieve a good performance for link sign prediction. On the other hand, as shown in Figure 4.4(b), all different values of γ can achieve a satisfactory low error rate for signed network clustering. It indicates that the performance of DNE-SBP for signed network clustering is insensitive to the value of γ.

Parameter $\boldsymbol{\alpha_k}$ denotes the weight assigned to the pairwise constraints at the k-th layer of SAE. As shown in Figures 4.3(c) and 4.4(c), $\alpha_1 > 0$ would contribute to better link sign prediction and also better signed network clustering performance, as compared to α_1=0. This demonstrates the effectiveness of designing a semi-supervised SAE to capture the structural balance property for signed network embedding. In addition, as shown in Figure 4.3(d), when $k \geq 2$, the AUC scores of DNE-SBP are insensitive to the value of α_k. This indicates that for link sign prediction, incorporating pairwise constraints to the first layer of SAE is much more effective than doing that for the deeper layers of SAE. However, as shown in Figure 4.4(d), when $k \geq 2$, $\alpha_k > 0$ can significantly reduce the error rate achieved by $\alpha_k = 0$. Thus, in contrast to link sign prediction, incorporating pairwise constraints at the deeper layers of SAE is more effective for community detection than doing that at the first layer of SAE.

Parameter $\boldsymbol{l}$ denotes the number of layers in the SAE. We constructed five SAEs, with the layer configuration setting as 7118-64, 7118-256-64, 7118-512-256-64, 7118-512-256-128-64 and 7118-1024-512-256-128-64, respectively. The number of layers in these five SAEs are 1, 2, 3, 4 and 5, respectively. Then, for all the five SAEs, we employ the deepest hidden representations as node vector representations, which are with the same dimensionality as d=64. As shown in Figure 4.3(e), $l > 1$ contributes to much higher AUC score than $l = 1$. While when $l > 2$, the AUC score does not further increase as l increases. This reveals that building a 2-layer SAE can contribute to much better link sign prediction performance, as compared to a basic auto-encoder. However, the link sign prediction performance cannot be further improved, even though a deeper SAE is built. In contrast, as shown in Figure 4.4(e), for signed network clustering, the error rate would keep decreasing as l increases. This indicates that the deeper SAE framework contributes to better signed network community detection performance. Such interesting differences could be explained by the fact that link prediction generally focuses on the concrete local neighborhood structure, thus the more abstract feature representations might not be more informative. However, community detection requires to capture the global network structure which is more abstract, thus the deeper SAE framework would be more powerful for learning the meaningful feature representations.

Parameter $\boldsymbol{d}$ indicates the dimensionality of the node vector representation learned by the deepest layer of SAE. As shown in Figures 4.3(f) and 4.4(f), when $d \in \{64, 128, 256\}$, DNE-SBP always achieves good performance for both link sign prediction and signed network clustering.

4.5 Summary

Although several promising network embedding algorithms have been proposed recently, the vast majority of them are only designed for unsigned networks, without considering the polarities of the links in the signed networks. In this work, we propose a DNE-SBP model to learn the nonlinear hidden vector representations for nodes of a given signed network, which employs a semi-supervised SAE to reconstruct the signed adjacency matrix. To handle the overwhelmingly positive links in the real-world signed networks, we impose larger penalty on the reconstruction errors of negative links to make the SAE focus more on reconstructing the scarce negative links than the abundant positive links. In addition, to capture the structural balance property of signed networks, we incorporate the ML and CL pairwise constraints into the SAE to map the positively connected nodes closer to each other and map the negatively connected nodes more far apart from each other in the embedding space. Based on the low-dimensional node vector representations learned by DNE-SBP, we apply vector-based machine learning techniques to conduct link sign prediction and signed network community detection. Comprehensive experimental results demonstrate that the proposed DNE-SBP model significantly outperforms the state-of-the-art network embedding algorithms for graph representation learning in the signed networks.

This work is accepted in [12]. In the future, we can extend the signed network embedding algorithm to capture other properties of the signed networks, such as status theory [52]. In addition, we can apply DNE-SBP to recommender systems, by modeling users and items as nodes, and the "like" and "dislike" relations as the "positive" and "negative" links. Moreover, by integrating network structures and rich content information (e.g. users/items features), we can extend DNE-SBP to heterogeneous network embedding. Then, we can make recommendations for users based on the link sign prediction results of DNE-SBP and conduct customer segmentation according to the network clustering results of DNE-SBP.

Chapter 5
Cross-network Deep Network Embedding

5.1 Introduction

Domain adaptation, aiming to transfer the knowledge pre-learned from a source domain to assist in solving the same task in a target domain, has received significant attentions in recent years [14]. Domain adaptation has been widely applied to computer vision (CV) [133], [134], [135] and natural language processing (NLP) [49], [136]. However, applying domain adaptation to classify nodes across networks has not been extensively investigated. In this chapter, we address a cross-network node classification problem, where the source network has fully labeled nodes while the target network has a very small fraction of labeled nodes together with a large number of unlabeled nodes. Our aim is to leverage the rich labeled information from the source network to help build an accurate node classifier for the target network in a collaborative manner. Here, following the assumption in domain adaption [14], the source network and the target network should share the same set of node labels. In order to achieve good performance in cross-network node classification, it is required to learn appropriate feature vector representations which are not only discriminative for different categories of nodes, but also invariant across different networks.

Network embedding is an effective method to learn the low-dimensional node vector representations which can preserve the original network structures. The homophily hypothesis [111] and social influence theory [110] suggest that the connected nodes tend to have similar labels. Based on such theories, recent state-of-the-art network embedding algorithms [29], [34], [24], [11], [10], [23], [43] aim to preserve network proximities by learning similar latent feature vector representations for nodes having connections in a given network. The proximity-based network embedding algorithms have demonstrated promising results for graph mining tasks involving one network. However, such proximities are defined based on the similarity between the neighborhood of the nodes in a given network, while the nodes from different networks do not have direct network connections, i.e., do not share common neighborhood. Thus, the node vector representations learned by the network embedding algorithms with single-network proximity preservation would lack consistency across different network and would be unsuitable for prediction tasks involving multiple networks [53], [54]. To learn appropriate feature vector representations for cross-network node classification, we need to preserve the proximities between nodes not only within a single network but also across different networks. It is indeed challenging to measure cross-network node proximities solely based on network topological structures. At the same time, nodes in the real-world networks are often associated with rich attributes, e.g., users in the social networks have profile information and papers in the citation networks have titles and abstracts. Node attributes have been shown to be generalized and comparable across different networks [54], [137]. For example, the papers belonging to the same research area, say "Information Security", from two different citation networks, might be likely to include some common keywords (textual attributes) in their titles, e.g., "Privacy, Verification, Encipher, Encryption, Decryption, Cryptography", while they might not have similar network topological structures across different networks. Thus, it would be beneficial to take advantage of network structures, node attributes and node labels together to learn appropriate feature vector representations for cross-network node classification.

In this regard, we propose an innovative cross-network deep network embedding (CDNE) model to learn label-discriminative and network-invariant node feature vector representations. Figure 5.1 illustrates the main ideas of the proposed CDNE model. Two semi-supervised SAEs, namely SAE_s and SAE_t,

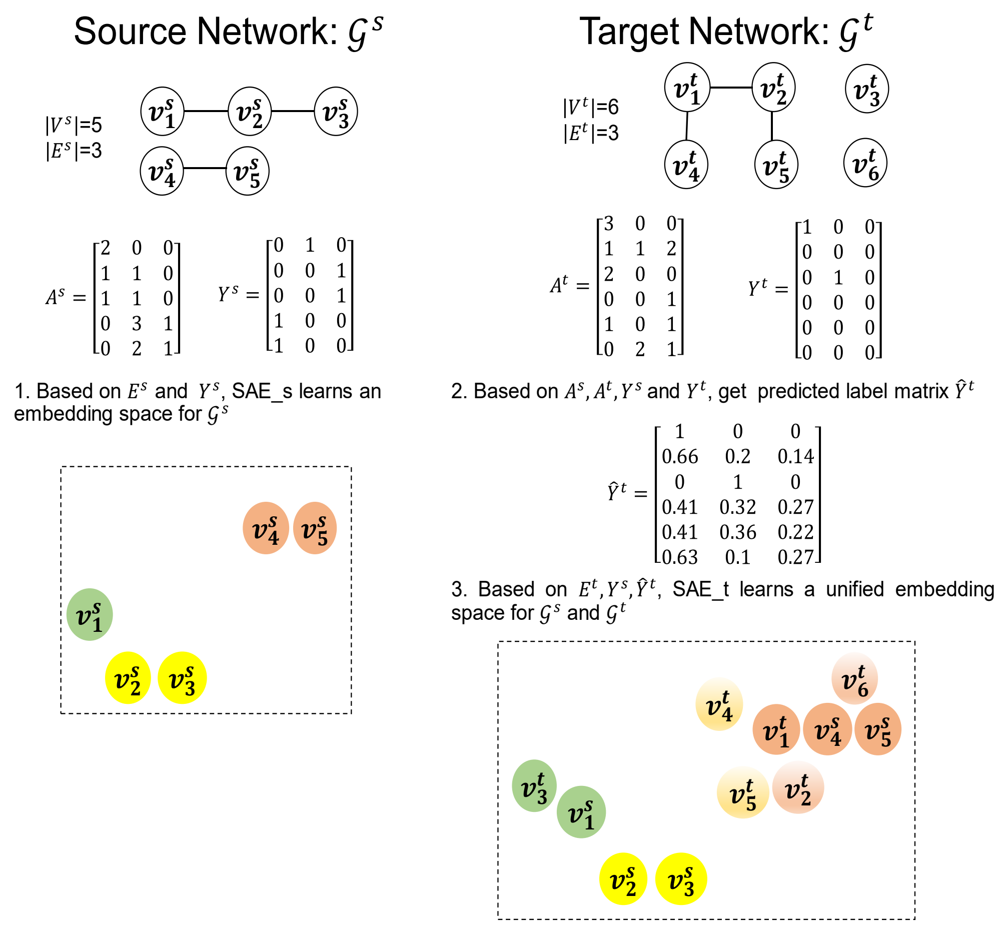

Source Network: $\mathcal{G}^s$
Target Network: $\mathcal{G}^t$
$|V^s|$=5
$|E^s|$=3
$|V^t|$=6
$|E^t|$=3
v_1^s v_2^s v_3^s v_4^s v_5^s
v_1^t v_2^t v_3^t v_4^t v_5^t v_6^t
$A^s = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 3 & 1 \\ 0 & 2 & 1 \end{bmatrix}$
$Y^s = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
$A^t = \begin{bmatrix} 3 & 0 & 0 \\ 1 & 1 & 2 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{bmatrix}$
$Y^t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
1. Based on E^s and Y^s, SAE_s learns an embedding space for $\mathcal{G}^s$
2. Based on A^s, A^t, Y^s and Y^t, get predicted label matrix $\hat{Y}^t$
$\hat{Y}^t = \begin{bmatrix} 1 & 0 & 0 \\ 0.66 & 0.2 & 0.14 \\ 0 & 1 & 0 \\ 0.41 & 0.32 & 0.27 \\ 0.41 & 0.36 & 0.22 \\ 0.63 & 0.1 & 0.27 \end{bmatrix}$
3. Based on $E^t, Y^s, \hat{Y}^t$, SAE_t learns a unified embedding space for $\mathcal{G}^s$ and $\mathcal{G}^t$

Figure 5.1: Illustration of the ideas of the CDNE model. v_i^s and v_i^t represent the i-th node in $\mathcal{G}^s$ and $\mathcal{G}^t$; A^s, A^t denote node attribute matrices in $\mathcal{G}^s$ and $\mathcal{G}^t$; Y^s, Y^t indicate node label matrices in $\mathcal{G}^s$ and $\mathcal{G}^t$. Different colors correspond to different labels. Full colors indicate observed labels while gradient colors represent the predicted fuzzy labels of unlabeled target network nodes. Firstly, in $\mathcal{G}^s$, SAE_s maps the connected nodes closer to each other and maps the same labeled nodes closer while different labeled nodes far apart from each other. Then, in $\mathcal{G}^t$, SAE_t maps the connected nodes closer and also maps the labeled nodes in $\mathcal{G}^t$ closer to the nodes associated with the same labels in $\mathcal{G}^s$.

are employed to learn the low-dimensional feature vector representations for the nodes in the source network and the target network, respectively. Firstly, in the source network, we employ SAE_s to map more strongly connected nodes closer to each other, and also map the nodes belonging to the same class closer while belonging to different classes far apart from each other, in the embedding space. Secondly, in order to successfully transfer the knowledge from the source network to the target network, we need to minimize the difference of the marginal and class-conditional distributions of the hidden node vector representations between the source network and the target network. However, the target network just has very limited labeled nodes, making it rather difficult to approximate its class-conditional distribution. To address this, we leverage the available node attributes and the labeled information from both the source network and the target network to predict pseudo-labels for the unlabeled target network nodes. Since such predictions might not be accurate, instead of utilizing binary labels, we employ fuzzy labels to capture the prediction uncertainty. Then, with the limited observed labels and the predicted fuzzy labels, we can easily estimate the class-conditional distributions of the target network. Thirdly, we employ SAE_t to learn more similar hidden vector representations for more strongly connected target network nodes. In addition, by minimizing the cross-network class-conditional distributions in SAE_t, we can make the labeled target network nodes have similar hidden vector representations w.r.t. the source network nodes associated with the same labels. Note that in SAE_s, different categories of source network nodes have already been mapped separately in the embedding space. Thus, the cross-network label-guided alignment in SAE_t would make the nodes associated with same labels no matter within the target network or across the source network and the target network have similar hidden vector representations, while making the nodes associated with completely different labels within the target network or across networks have rather different hidden vector representations. Such properties yield label-discriminative and network-invariant node vector representations, which can significantly benefit cross-network node classification. The contributions of this work are summarized as follows:

1) We should be the first to take advantage of both deep network embedding and domain adaptation to address the cross-network node classification problem;
2) We leverage network structures, node attributes and node labels to map similar nodes no matter within a network or across different networks closer to each other in the unified low-dimensional embedding space so as to learn label-discriminative and network-invariant node vector representations;
3) Extensive experimental results in the real-world datasets demonstrate that the proposed CDNE model significantly outperforms the state-of-the-art related algorithms for cross-network node classification.

The rest of this chapter is organized as follows. Section 5.2 reviews the network embedding algorithms and network transfer learning algorithms. Section 5.3 formulates the cross-network node classification problem. Section 5.4 introduces the detailed framework of CDNE. Section 5.5 reports the experimental results of CDNE for cross-network node classification. Section 5.6 summaries this work.

5.2 Related Work

In this section, we review the state-of-the-art single-network embedding algorithms, cross-network embedding algorithms and network transfer learning algorithms.

5.2.1 Single-network Embedding Algorithms

Recent network embedding algorithms aim to embed a given network into a low-dimensional space where the original network properties can be well preserved. The vast majority of network embedding algorithms are proximity-based [29], [34], [24], [11], [10], [23], [43], with the goal of mapping nodes with higher proximities closer to each other. Such proximities are defined in terms of the neighborhoods between nodes, thus, these network embedding algorithms can only preserve the proximities between connected nodes. If two nodes are disconnected from each other in a network, then they would not have similar hidden vector representations. In contrast to proximity-based network embedding algorithms, Ribeiro *et al.* [138] proposed a struc2vec algorithm to measure structural similarity between nodes independently of their positions in the network. Thus, two disconnected nodes that are far apart from each other in the network but are structurally similar would have similar latent feature representations. struc2vec can achieve better node classification performance than the proximity-based embedding algorithms when the node labels depend more on structural identity while less on homophily. Both proximity-based and structural identity-based network embedding algorithms just employ the pure network topological structures to learn the latent node vector representations. While in the real-world networks, nodes are often associated with rich attributes. In addition, it has been shown that capturing the correlations between network structures and node attributes can benefit graph mining applications [139]. Thus, recently, some attributed network embedding algorithms have been proposed to incorporate node attributes into network representation learning so as to preserve both network topological proximity and node attribute affinity. For example, Chang *et al.* [35] proposed a heterogenous network embedding framework to leverage deep learning technique to learn node vector representations based on both node contents and linkage structures in the network. Huang *et al.* [139] developed a LANE algorithm to incorporate the label information into attributed network embedding and jointly project node labels, network structures and node attributes into a unified embedding space via EVD. Pan *et al.* [140] designed a TriDNR model which utilizes a coupled neural network architecture to learn deep network representations from node structures, node content and node labels.

5.2.2 Cross-network Embedding

Although the single-network embedding algorithms can well capture the properties necessary for the graph mining tasks involving one network, they fail to learn generalized and comparable network representations across different networks [53], [54]. Recently, some cross-network embedding approaches have been proposed to address the network alignment problem, which aims to find the corresponding nodes across different networks. For example, in [141], [142], the observed anchor links are utilized as the supervised information to learn a cross-network embedding space based on the network structural information. Then, the unknown anchor links across networks would be predicted in the unified embedding space. In [54], Heimann *et al.* proposed to measure cross-network node similarities based on the nodes' structural and attribute identities, and then employ SVD to factorize the cross-network similarity matrix so as to learn the low-dimensional node vector representations. Next, they infer network alignment based on the cross-network node vector representations, by finding the top-*c* most similar nodes in one network given a query node in the other network.

However, to the best of our knowledge, leveraging cross-network embedding to address the cross-network node classification task has not been investigated. Cross-network node classification is different from network alignment in two aspects. Firstly, network alignment assumes that some users should be simultaneously involved in the two aligned social networks [137]. In contrast, in cross-network node classification, the source network and the target network do not share any common nodes. Secondly, the goal of network alignment is to infer a node mapping between two networks, while cross-network node classification aims to predict node labels in the target network by leveraging the abundant label information from the source network. Thus, the existing cross-network embedding algorithms developed for network alignment cannot be applied to the cross-network node classification task.

5.2.3 Transfer Learning Across Networks

Network transfer learning aims to transfer the useful knowledge from a source network to help predict node or edge labels in the target network. For example, Ye *et al.* [15] proposed a transfer learning approach to predict the signed labels of edges in a target network by leveraging the edge label information in the source network. They construct the generalized edge features from two aspects, namely 1) the explicit topological features, such as node degree, betweenness centrality, triad count and edge embeddedness, and 2) the latent topological features learned by projecting the adjacency matrices of the source network and the target network into a common latent space via Nonnegative Matrix Tri-Factorization (NMTF) [143]. Then, based on the constructed edge feature vector representations, they adopt an AdaBoost [144] scheme for transfer learning by assigning higher weights to the source edge instances which are more useful for edge label prediction in the target network. In [13], Tang *et al.* aim to classify the type of social relationships in a target network by borrowing the knowledge from a source network. They manually defined the edge features based on the social theories, such as social balance, structural hole and social status. Fang *et al.* [16] developed a network transfer learning algorithm to predict node labels in the target network by transferring the useful knowledge from the source network. To learn domain-independent latent structural features, the NMTF [143] technique is employed to project the label propagation matrices of the two networks into a common latent space. Then, they predict node labels in the target network based on the latent structural features, node content features and relational features. In our previous work [9], [8], we proposed a CNL model to predict most likely seed nodes and inactive edges in the target network, by leveraging the greedy seed selection and influence propagation knowledge pre-learned from a source network. The model selects a set of discriminative topological features as the node and edge features and assigns higher weights to the features which perform more similarly between the source network and the target network.

To effectively transfer the knowledge from a source network to a target network for cross-network prediction, it is required to learn the feature vector representations which are generalized and comparable across networks. Existing literatures either manually define explicit topological features or learn the common latent features by factorizing the associated characteristic matrices of the source network and the target network into a common latent space. To the best of our knowledge, we are the first to take advantage of deep network embedding to learn generalized and comparable feature vector representations for cross-network prediction.

5.3 Problem Statement

Given a target network with very limited labeled nodes and a source network with fully labeled nodes, the goal of cross-network node classification is to leverage the abundant labeled information from the source network to help classify unlabeled nodes in the target network. In this chapter, we use superscripts s and t to denote the source network and the target network, respectively. Table 5.1 lists the used notations and corresponding descriptions.

Table 5.1: Frequently used notations and descriptions in Chapter 5.

Notations	Descriptions
$\mathcal{G}^s, \mathcal{G}^t$	Source network and target network
n^s, n^t	Number of nodes in source network and in target network
A^s, A^t	Node attribute matrices of source network and target network
c	Number of node labels
Y^s, Y^t	Observable node label matrices of source network and target network
$\hat{Y}^t$	Predicted node label matrix of target network
X^s, X^t	Aggregated PPMI matrices of source network and target network
l	Number of layers in SAE_s and SAE_t
$d(k)$	The hidden dimensionality at the *k*-th layer of SAE_s and SAE_t
$W_1^{s(k)}, W_2^{s(k)}$	Encoding and decoding weight matrices of *k*-th layer of SAE_s
$B_1^{s(k)}, B_2^{s(k)}$	Encoding and decoding bias matrices of *k*-th layer of SAE_s
$H^{s(k)}$	Hidden matrix representation of source network learned by *k*-th layer of SAE_s
$Q^{s(k)}$	*i*-th row represents average hidden vector representation of the source network nodes associated with label *i*
$W_1^{t(k)}, W_2^{t(k)}$	Encoding and decoding weight matrices of *k*-th layer of SAE_t
$B_1^{t(k)}, B_2^{t(k)}$	Encoding and decoding bias matrices of *k*-th layer of SAE_t
$H^{t(k)}$	Hidden matrix representation of target network learned by *k*-th layer of SAE_t
$Q^{t(k)}$	*i*-th row represents average hidden vector representation of the target network nodes associated with label *i*

Let $\mathcal{G}^s = (V^s, E^s, Y^s, A^s)$ be a fully labeled source network, where V^s denotes the set of all labeled nodes and E^s indicates the set of edges. $Y^s \in R^{n^s \times c}$ is a label matrix encoding the label information of all nodes in $\mathcal{G}^s$, where $n^s = |V^s|$ is the number of nodes in $\mathcal{G}^s$ and *c* is the number of labels. The entry in the *i*-th row and *k*-th column of matrix Y^s, i.e., $Y_{ik}^s = 1$ if node $v_i^s \in V^s$ is associated with label *k*; otherwise, $Y_{ik}^s = 0$. Here, a node can have multiple labels. $A^s \in R^{n^s \times w}$ is the associated attribute matrix representing the attribute values of all nodes in $\mathcal{G}^s$, where w indicates the number of attributes and $A_{ik}^s \geq 0$ indicates the *k*-th attribute value of node v_i^s.

Let $\mathcal{G}^t = (V^t, E^t, Y^t, A^t)$ be an insufficiently labeled target network with a set of nodes $V^t = \{V_l^t, V_u^t\}$ and a set of edges E^t, where $n^t = |V^t|$ denotes the number of nodes in $\mathcal{G}^t$, V_l^t indicates the set of a very small fraction of labeled nodes and V_u^t represents the set of a large amount of unlabeled nodes in $\mathcal{G}^t$. $Y^t \in R^{n^t \times c}$ is the observable node label matrix associated with $\mathcal{G}^t$, where $Y_{ik}^t = 1$ if node $v_i^t \in V^t$ has an observable label k; otherwise, $Y_{ik}^t = 0$. $A^t \in R^{n^t \times w}$ refers to the associated node attribute matrix, where $A_{ik}^t \geq 0$ indicates the k-th attribute value of node v_i^t.

It should be noted that in the defined cross-network node classification problem, the network dimensionality (i.e. number of nodes) and the distribution of network connections are varied between the source network and the target network. However, the two networks should share the same set of node labels and node attributes, while the data distributions across networks are varied.

5.4 Cross-network Deep Network Embedding Model

In this section, we introduce the proposed CDNE model which is composed of two semi-supervised SAEs, i.e., SAE_s and SAE_t. Algorithm 5.1 shows the framework of the CDNE model. Firstly, we employ SAE_s to learn the label-discriminative feature vector representations for nodes in the source network. Next, we employ SAE_t to learn node vector representations for the target network, which can match with the marginal and class-conditional distributions of the source network. Before introducing SAE_s and SAE_t, we firstly compute the aggregated k-step PPMI matrices for the source network and the target network, denoted as X^s and X^t, respectively. The approach of computing the PPMI matrix can be refereed to Section 3.3.1.

5.4.1 SAE_s for Deep Network Embedding in Source Network

In SAE_s, we aim to map nodes with higher network proximities in the source network closer to each other, and map nodes associated with more common labels closer while mapping nodes not sharing any common labels far apart from each other.

5.4.1.1 Penalty-modified Reconstruction Errors

Given the aggregated k-step PPMI matrix of the source network as the input, i.e., $H^{s(0)} = X^s$, a l-layer SAE_s is constructed as follow:

$$H^{s(k)} = f\left(H^{s(k-1)}\left(W_1^{s(k)}\right)^T + B_1^{s(k)}\right),\ k = 1, \dots, l \tag{5.1}$$

$$\widehat{H}^{s(k-1)} = f\left(\widehat{H}^{s(k)}\left(W_2^{s(k)}\right)^T + B_2^{s(k)}\right),\ k = l, \dots, 1 \tag{5.2}$$

where (5.1) and (5.2) represent the encoding and decoding process of SAE_s, respectively. $H^{s(k)} \in R^{n^s \times d(k)}$ denotes the hidden matrix representation learned by the k-th layer of SAE_s, with the i-th row (i.e. $H_i^{s(k)} \in R^{1\times d(k)}$) representing the feature vector representation of v_i^s, and $d(k)$ indicates the hidden dimensionality at the k-th layer of SAE_s. $\widehat{H}^{s(k)}$ indicates the reconstructed matrix of $H^{s(k)}$ and $\widehat{H}^{s(l)} = H^{s(l)}$. In addition, $W_1^{s(k)} \in R^{d(k)\times d(k-1)}, B_1^{s(k)} \in R^{n^s \times d(k)}, W_2^{s(k)} \in R^{d(k-1)\times d(k)}, B_2^{s(k)} \in R^{n^s\times d(k-1)}$ refer to the encoding weight, encoding bias, decoding weight and decoding bias matrices associated with the k-th layer of SAE_s, respectively; f is a non-linear activation function, in this work, the sigmoid function $f(x) = 1/(1 + e^{-x})$ is utilized as the activation function for each layer of SAE_s. The i-th row of the input matrix X^s, i.e., $X_i^s \in R^{1\times n^s}$ captures the neighborhood of v_i^s with the associated proximities, where $X_{ij}^s > 0$ if v_j^s is a strong neighbor of v_i^s within K steps; otherwise, $X_{ij}^s = 0$.

The nodes having similar neighborhood structure would have similar input raw vectors. Then, by minimizing th3 reconstruction errors of SAE_s, we can learn more similar hidden vector representations for nodes with more similar neighborhood structure. In addition, to address the network sparsity issue, we follow [10] to incorporate a penalty matrix $P^{s(1)} \in R^{n^s \times n^s}$ into the reconstruction errors as:

$$\mathcal{R}^{s(1)} = \frac{1}{2n^s} \left\| P^{s(1)} \odot \left(\hat{X}^s - X^s\right) \right\|_F^2 \tag{5.3}$$

where $P_{ij}^{s(1)} = \beta > 1$, if $X_{ij}^s > 0$ and $P_{ij}^{s(1)} = 1$, if $X_{ij}^s = 0$. Incorporating $\beta > 1$ makes SAE_s focus more on reconstructing the strong connections (positive proximities) than the weak connections or disconnections (zero proximities). Although the input matrix (i.e. $H^{s(k-1)}$) of the deep k-th $(k \geq 2)$ layer of SAE_s is not sparse any more, we still incorporate the penalty matrix by regarding β^2 as the weight of the reconstruction errors in the overall loss function. Thus, the reconstruction errors at any k-th $(1 \leq k \leq l)$ layer of SAE-s is defined as:

$$\mathcal{R}^{s(k)} = \frac{1}{2n^s} \left\| P^{s(k)} \odot \left(\hat{H}^{s(k-1)} - H^{s(k-1)}\right) \right\|_F^2 \tag{5.4}$$

where $P^{s(k)} \in R^{n^s \times d(k-1)}$, if $H_{ij}^{s(k-1)} > 0$, $P_{ij}^{s(k)} = \beta > 1$; if $H_{ij}^{s(k-1)} = 0$, $P_{ij}^{s(k)} = 1$.

5.4.1.2 Pairwise Constraint on Connected Nodes

Next, we design the following pairwise constraint on strongly connected node pairs:

$$\begin{aligned} \mathcal{C}^{s(k)} &= \frac{1}{2n^s} \sum_{i=1}^{n^s} \sum_{j=1}^{n^s} \left(X_{ij}^s + X_{ji}^s\right) \left\| H_i^{s(k)} - H_j^{s(k)} \right\|_2^2 \\ &= \frac{1}{n^s} Tr\left(\left(H^{s(k)}\right)^T L_{X^s} H^{s(k)} \right) \end{aligned} \tag{5.5}$$

where $L_{X^s} = D_{X^s} - (X^s + (X^s)^T)$ is the Laplacian matrix of $X^s + (X^s)^T$, and D_{X^s} is a diagonal matrix with the diagonal entry computed as $(D_{X^s})_{ii} =$

$\sum_{j=1}^{n^s} X_{ij}^s + \sum_{j=1}^{n^s} X_{ji}^s$. Minimizing $\mathcal{C}^{s(k)}$ would map the more strongly connected node pairs which are with higher aggregated proximities to have more similar hidden vector representations.

5.4.1.3 Pairwise Constraint on Labeled Nodes

In addition, we devise the pairwise constraint on the labeled nodes in the source network. Firstly, we define the following matrix $O^s \in R^{n^s \times n^s}$ to represent whether two nodes in $\mathcal{G}^s$ share common labels or not:

$$O_{ij}^s = \begin{cases} (Y^s(Y^s)^T)_{ij}, if\ i \neq j\ and\ (Y^s(Y^s)^T)_{ij} > 0 \\ -1, if\ i \neq j\ and\ (Y^s(Y^s)^T)_{ij} = 0 \\ 0,\ if\ i = j \end{cases}$$

where $O_{ij}^s = -1$ if v_i^s and v_j^s do not share any common labels; $O_{ij}^s \geq 1$ indicates the number of common labels shared by v_i^s and v_j^s. Then, the pairwise constrain is devised as:

$$\mathcal{L}^{s(k)} = \frac{1}{2n^s} \sum_{i=1}^{n^s} \sum_{j=1}^{n^s} O_{ij}^s \left\| H_i^{s(k)} - H_j^{s(k)} \right\|_2^2 = \frac{1}{n^s} Tr\left(\left(H^{s(k)}\right)^T L_{O^s} H^{s(k)} \right) \quad (5.6)$$

where O^s can be divided into a positive part O^{s+} and a negative part O^{s-} as: $O^{s+} = max(O^s, 0)$ and $O^{s-} = -min(O^s, 0)$. In addition, $L_{O^s} = L_{O^{s+}} - L_{O^{s-}}$, where $L_{O^{s+}} = D_{O^{s+}} - O^{s+}$ is the associated Laplacian matrix of O^{s+} and $D_{O^{s+}}$ is a diagonal matrix of O^{s+} with the diagonal entries as the row summation of O^{s+}, i.e., $(D_{O^{s+}})_{ii} = \sum_{j=1}^{n^s} O_{ij}^{s+}$. Similarly, $L_{O^{s-}} = D_{O^{s-}} - O^{s-}$ is the associated Laplacian matrix of O^{s-} . Minimizing $\mathcal{L}^{s(k)}$ would make the node pairs sharing more common labels have more similar hidden vector representations, while making the node pairs not sharing any common labels (i.e. belonging to completely different classes) have rather different hidden vector representations.

5.4.1.4 Overall Loss Function of SAE_s:

By integrating the penalty-modified reconstruction errors (5.4), the pairwise constraints on strongly connected nodes (5.5) and on labeled nodes (5.6), and a *L2*-norm regularization term to prevent overfitting $\Omega^{s(k)} = \frac{1}{2}\left(\left\|W_1^{s(k)}\right\|_F^2 + \left\|W_2^{s(k)}\right\|_F^2\right)$, the overall loss function of SAE_s is defined as:

$$\mathcal{J}^s = \sum_{k=1}^{l} \mathcal{J}^{s(k)} = \sum_{k=1}^{l} \mathcal{R}^{s(k)} + \alpha^{s(k)}\, \mathcal{C}^{s(k)} + \varphi^{s(k)} \mathcal{L}^{s(k)} + \lambda^{s(k)} \Omega^{s(k)} \quad (5.7)$$

where $\alpha^{s(k)}$, $\varphi^{s(k)}$ and $\lambda^{s(k)}$ are the trade-off parameters to balance the effect of different terms in the overall loss function.

5.4.2 SAE_t for Deep Network Embedding in Target Network

Next, in SAE_t, we aim to learn node vector representations for the target network which can match with the distributions of nodes in the source network. It is worth noting that the number of layers of SAE_t and the hidden dimensionality at each k-th $(\forall\, 1 \leq k \leq l)$ of SAE_t are set as the same as in SAE_s. However, the raw input dimensionality between SAE_s and SAE_t are varied, since the number of nodes in the source network and the target network can be rather different.

Given the aggregated k-step PPMI matrix of the target network as the input of SAE_t, i.e., $H^{t(0)} = X^t \in R^{n^t \times n^t}$, the penalty-modified reconstruction errors of SAE_t are similarly defined as:

$$\mathcal{R}^{t(k)} = \frac{1}{2n^t} \left\| P^{t(k)} \odot \left(\hat{H}^{t(k-1)} - H^{t(k-1)}\right) \right\|_F^2 \quad (5.8)$$

where $P^{t(k)} \in R^{n^t \times d(k-1)}$, and $P_{ij}^{t(k)} = \beta > 1$, if $H_{ij}^{t(k-1)} > 0$ and $P_{ij}^{t(k)} = 1$, if $H_{ij}^{t(k-1)} = 0$. In addition, similar to SAE_s, we define the pairwise constraint to map more strongly connected nodes in the target network closer to each other as:

$$\mathcal{C}^{t(k)} = \frac{1}{n^t} Tr\left(\left(H^{t(k)}\right)^T L_{X^t} H^{t(k)}\right) \quad (5.9)$$

where $L_{X^t} = D_{X^t} - (X^t + (X^t)^T)$ is the Laplacian matrix of $X^t + (X^t)^T$, and

D_{X^t} is a diagonal matrix with the diagonal entry computed as $(D_{X^t})_{ii} = \sum_{j=1}^{n^t} X_{ij}^t + \sum_{j=1}^{n^t} X_{ji}^t$.

5.4.2.1 Matching Marginal and Conditional Cross-network Distributions

Next, we need to align the node vector representations learned for the target network to that of the source network. In domain adaptation algorithms [134], [145], [46] the nonparametric maximum mean discrepancy (MMD) [146] metric have been widely employed to measure the difference of data distributions across domains. Motivated by this, we utilize the MMD metric to measure the marginal and conditional distributions of the embedding node vector representations between the source network and the target network.

Firstly, the empirical marginal MMD between the source network and the target network is defined as:

$$\mathcal{M}_M^{t(k)} = \frac{1}{2}\left\|\frac{1}{n^s}\mathbf{1}^s H^{s(k)} - \frac{1}{n^t}\mathbf{1}^t H^{t(k)}\right\|_2^2 \tag{5.10}$$

where $\mathbf{1}^s \in R^{1\times n^s}$ and $\mathbf{1}^t \in R^{1\times n^t}$ denote two ones-vectors. By minimizing (5.10), the marginal distributions between the source network and the target network can be matched.

Secondly, the class-conditional MMD [46] between the source network and the target network is defined as:

$$\mathcal{M}_C^{t(k)} = \frac{1}{2}\left\|Q^{t(k)} - Q^{s(k)}\right\|_F^2 \tag{5.11}$$

where $Q^{s(k)} = \left(Y^s (D_{Y^s})^{-1}\right)^T H^{s(k)}$, $D_{Y^s} \in R^{c\times c}$ is a diagonal matrix with the diagonal entries as the column summation of Y^s, $(D_{Y^s})_{ii} = \sum_{j=1}^{n^s} Y_{ji}^s$. The i-th row of $Q^{s(k)}$, i.e., $Q_i^{s(k)} \in R^{1\times d(k)}$ represents the average feature vector representation of all the nodes associated with label i in the source network.

In addition, we need to compute $Q^{t(k)}$, i.e., the label-specific average feature vector representations of the nodes in the target network. Unlike the nodes in the source network which are with completely observable labels, the target network nodes are just with very limited observable labels. Then, directly utilizing the sparse observable label matrix Y^t to compute $Q^{t(k)}$ would fail to obtain sufficient statistics to update the weights of SAE_t, and thus cannot learn a good cross-network embedding alignment. To address this, we take advantage of the available node attributes in the source network and the target network to predict labels for the unlabeled nodes in the target network. Since the original node attributes might contain noises, we firstly employ Principal Components Analysis (PCA) [147] to extract the low-dimensional (i.e. 128-D in the experiments) node attribute vector representations. Then, we employ a one-vs-rest logistic regression (LR) classifier to predict the labels of the unlabeled target network nodes based on the low-dimensional attribute vector representations and all the observable labels in the source network and the target network. Fuzzy labels have been shown to be effective in capturing imprecise, uncertain and vague information during knowledge transfer [67], [66]. Thus, instead of using binary labels, we utilize fuzzy labels to capture the degree of the membership of each node belonging to a specific class. Let $\hat{Y}^t$ denotes the predicted label matrix of the target network, where if $v_i^t \in V_l^t$, $\hat{Y}_{ij}^t = Y_{ij}^t \in \{0,1\}$; and if $v_i^t \in V_u^t$, $0 < \hat{Y}_{ij}^t < 1$ represents the predicted probability of v_i^t to be labeled as *j*. Next, we utilize $\hat{Y}^t$ which includes both limited observable binary labels and predicted fuzzy labels to compute the average node feature vector representation of each class in the target network as $Q^{t(k)} = \left(\hat{Y}^t (D_{\hat{Y}^t})^{-1}\right)^T H^{t(k)}$, where $D_{\hat{Y}^t}$ is a diagonal matrix with the diagonal entries as the column summation of $\hat{Y}^t$, i.e., $(D_{\hat{Y}^t})_{ii} = \sum_{j=1}^{n^t} \hat{Y}_{ji}^t$; and the *i*-th row of $Q^{t(k)}$ represents the average feature vector representations of all the target network nodes associated with (observable or predicted) label *i*, learned by the *k*-th layer of SAE_t.

By minimizing (5.11), we can align the true and pseudo labeled target network nodes to the source network nodes which are associated with the same labels. Thus, the same category of nodes across different networks would have similar hidden vector representations. In addition, it should be noted that minimizing (5.6) in SAE_s has already pulled the nodes belonging to different classes far apart from each other. Therefore, minimizing (5.11) would simultaneously make different categories of target network nodes have rather different feature vector representations. In addition, by minimizing (5.10) and (5.11) at each layer of SAE_t, we can match the cross-network marginal and class-conditional distributions of the embedding node vector representations at each corresponding layer of SAE_s and SAE_t. Then, with the narrowed cross-network distributions, it is beneficial to leverage the abundant label information from the source network to help classify unlabeled nodes in the target network.

5.4.2.2 Overall Loss Function of SAE_t

By integrating the penalty-modified reconstruction errors (5.8), the pairwise constraints on strongly connected nodes (5.9), the marginal MMD (5.10), the conditional MMD (5.11), and a *L2*-norm regularization $\Omega^{t(k)} = \frac{1}{2}\left(\left\|W_1^{t(k)}\right\|_F^2 + \left\|W_2^{t(k)}\right\|_F^2\right)$, the overall loss function of SAE_t is defined as:

$$\mathcal{J}^t = \sum\nolimits_{k=1}^{l} \mathcal{J}^{t(k)}$$

$$= \sum\nolimits_{k=1}^{l} \mathcal{R}^{t(k)} + \alpha^{t(k)}\, \mathcal{C}^{t(k)} + \mu^{t(k)} \mathcal{M}_M^{t(k)} + \gamma^{t(k)} \mathcal{M}_C^{t(k)} + \lambda^{t(k)} \Omega^{t(k)} \quad (5.12)$$

where $\alpha^{t(k)}$, $\mu^{t(k)}$, $\gamma^{t(k)}$ and $\lambda^{t(k)}$ are the trade-off parameters to balance the effect of different terms in the overall loss function.

On one hand, for a labeled node $v_i^t \in V_l^t$ in the target network, minimizing (5.11) would make v_i^t have a feature vector representation similar to the same categories of nodes in the source network. On the other hand, for an unlabeled node $v_i^t \in V_u^t$ in the target network, if v_i^t has some network connection with any labeled nodes in V_l^t within *K* steps, then minimizing (5.9) would make v_i^t have similar hidden vector representation w.r.t. its labeled neighbors in the target network. In addition, if an unlabeled target network node $v_i^t \in V_u^t$ is disconnected from all the labeled nodes in V_l^t within *K* steps, then minimizing (5.11) will make v_i^t have similar hidden vector representations w.r.t. the same labeled source network nodes, according to the predicted fuzzy labels of v_i^t based on the available node attribute information.

By simultaneously taking advantage of network connections, node attributes and node labels, the proposed CDNE model can 1) make more strongly connected nodes within a network have more similar hidden vector representations, and 2) make nodes associated with same labels within a network or across networks have similar hidden vector representations while making nodes associated with different labels within a network or across networks have rather different hidden vector representations. Thus, the proposed CDNE model can learn label-discriminative and network-invariant feature vector representations for cross-network node classification.

To optimize CDNE, SAE_s and SAE_t should be trained in sequence, as shown in Algorithm 5.1. To optimize SAE_s or SAE_t, one can use stochastic gradient descent (SGD) to optimize each *k*-th layer of a SAE and employ a greedy layer-wise training approach [34], [129] until reaching the deepest *l*-th layer so as to learn the deepest hidden matrix representation.

Algorithm 5.1: CDNE

Input: Source network $\mathcal{G}^s = (V^s, E^s, X^s, Y^s, A^s)$ and target network $\mathcal{G}^t = (V^t, E^t, X^t, Y^t, A^t)$.

```
1. Greedy layer-wised training for SAE-s:
Set $H^{s(0)} = X^s$
For $k$=1: $l$
    1.1 Leverage $H^{s(k-1)}$ as input to $k$-th layer of SAE-s;
    1.2 Given $H^{s(k-1)}, X^s, Y^s$, optimize $k$-th layer of SAE-s by finding $\theta^{s(k)*} = \left\{W_1^{s(k)*}, W_2^{s(k)*}, B_1^{s(k)*}, B_2^{s(k)*}\right\} = arg \min_{\theta^{s(k)}} \mathcal{J}^{s(k)}$ via SGD;
    1.3 Leverage $\theta^{s(k)*}$ to learn $H^{s(k)}$;
End for
2. Get predicted label matrix for $\mathcal{G}^t$:
    2.1 Convert $A^s$ and $A^t$ into low-dimensional attribute node vector representations via PCA;
    2.2 Predict fuzzy labels for nodes in $V_u^t$ based on attribute vector representations and obtain $\hat{Y}^t$;
3. Greedy layer-wised training for SAE-t:
Set $H^{t(0)} = X^t$
For $k$=1: $l$
    3.1 Leverage $H^{t(k-1)}$ as input to $k$-th layer of SAE-t;
    3.2 Given $H^{t(k-1)}, X^t, \hat{Y}^t, H^{s(k-1)}, Y^s$, optimize $k$-th layer of SAE-t by finding $\theta^{t(k)*} = \left\{W_1^{t(k)*}, W_2^{t(k)*}, B_1^{t(k)*}, B_2^{t(k)*}\right\} = arg \min_{\theta^{t(k)}} \mathcal{J}^{t(k)}$ via SGD;
    3.3 Leverage $\theta^{t(k)*}$ to learn $H^{t(k)}$;
End for
```

Output: Discriminative and generalized hidden node vector representations for $\mathcal{G}^s$ and $\mathcal{G}^t$, i.e., $H^{s(l)}$ and $H^{t(l)}$.

5.5 Experiments

5.5.1 Datasets

We evaluated the proposed CDNE model in five real-world networked datasets. The statistics of these datasets are shown in Table 5.2. BlogCatalog1 and BlogCatalog2 are two disjoint subnetworks we extracted from the BlogCatalog dataset [148]. In these two networks, a node represents a blogger and an edge indicates friendship between two bloggers. In addition, each node is associated with some attributes, which are the keywords extracted from the blogger's self-description. In the original BlogCatalog dataset [148], each node has only one label, indicating the blogger's interested group. Since the two networks were extracted from the same original network, the attribute distributions between the two networks are not varied enough. To enlarge the cross-network distribution discrepancy, we randomly altered 30% of non-zero attribute values to be zeroes and randomly altered 30% of zero attribute values to be "1" in each network so as to simulate missing and inconsistent attribute values across different networks.

In addition, Citationv1, DBLPv7 and ACMv9 are the citation networks, where each node denotes a paper and each edge indicates that one paper cites another. We extracted Citationv1, DBLPv7 and ACMv9 from the datasets provided by ArnetMiner[7] [149], which were from different sources, i.e., Microsoft Academic Graph, DBLP and ACM, respectively. The papers were labeled according to their associated research areas and one paper can have multiple labels. The possible labels include "Database, Artificial Intelligence, Computer Vision, Information Security, and Networking". For each paper, we adopted the bag-of-words features extracted from its title as the attributes. Besides, we removed the attributes that are

[7] https://www.aminer.cn/citation

not shared by the three networks so as to ensure a common set of attributes across networks. In addition, the extracted papers we keep in Citationv1, DBLPv7 and ACMv9 were published before year 2008, between year 2004 and 2008 and after year 2010, respectively. Since the three citation networks were extracted from different original sources and also formed in different time periods, there have already existed varied cross-network distributions among them.

Table 5.2: Statistics of the networked datasets.

Dataset	# Nodes	# Edges	# Attributes	# Labels
BlogCatalog1	2300	33471	8189	6
BlogCatalog2	2896	53836		
Citationv1	8935	15113	3092	5
DBLPv7	5484	8130		
ACMv9	9360	15602		

5.5.2 Implementation Details

In the proposed CDNE model, we built a 2-layer SAE for both SAE_s and SAE_t, with the number of hidden dimensions as $d(1) = 256$ and $d(2) = 128$ for the 1-st layer and the 2-nd layer of SAE, respectively. We employed the deepest hidden vector representations learned by SAE_s and SAE_t as the node vector representations for the source network and the target network. For all the datasets, we set the maximum step as *K*=6. In both SAE_s and SAE_t, we set the ratio of penalty on the reconstruction errors of strong connections over that of weak connections and disconnections as $\beta = 6$; set the weight of L2-norm regularization as $\lambda^{s(k)} = \lambda^{t(k)} = 0.05$; and set the weight of pairwise constraints on strongly connected nodes as $\alpha^{s(1)} = \alpha^{t(1)} = \alpha = 4$ and $\alpha^{s(k)} = \alpha^{t(k)} = \alpha/2, \forall\, k \geq 2$. In addition, in SAE_s, we set the weight of pairwise constraints on the labeled nodes as $\varphi^{s(1)} = \varphi = 2$ and $\varphi^{s(k)} = \varphi/2, \forall\, k \geq 2$. In SAE_t, we set the weight of marginal MMD as $\mu^{t(1)} = \mu = 2$ and $\mu^{t(k)} = \mu/2, \forall\, k \geq 2$ for all the datasets; and set the weight of conditional MMD as $\gamma^{t(1)} = \gamma = 60, \gamma^{t(k)} = \gamma/2, \forall\, k \geq 2$ for the cross-network node classification task across BlogCatalog1 and BlogCatalog2, and set $\gamma = 50$ for cross-network node classification among Citationv1, DBLPv7 and ACMv9.

5.5.3 Baselines

The proposed CDNE was benchmarked against two transfer learning algorithms. In addition, since there is no existing cross-network embedding algorithm developed for cross-network node classification, we compare CDNE with the state-of-the-art single-network embedding algorithms.

1) **NetworkTr** [16]: It is a transfer learning algorithm specifically developed for the networked data. It employs NMTF to project the label propagation matrices of the source network and the target network into a common latent space so as to learn the latent structural features shared by the two networks. Then, it employs node attributes and latent structural features as the node features for cross-network node classification.
2) **TrAdaBoost** [144]: It is an instance weighting transfer learning algorithm which assigns higher weights to the source domain instances that are more useful for prediction in the target domain. It requires the source domain and the target domain to share the same set of pre-defined feature vector representations. To tailor TrAdaBoost for cross-network node classification, we employed the low-dimensional attribute vector representations extracted by PCA as the node feature vector representations.
3) **DeepWalk** [24]: It is a random-walk based network embedding algorithm, which exploits the network structure by generating a collection of truncated random walks via DFS. Then, by regarding a node in a network as a word in a

document, it extends the Skip-Gram language model [41] to learn low-dimensional node vector representations.

4) **GraRep** [29]: It is a matrix-factorization based network embedding algorithm, which employs SVD to factorize each *k*-step PPMI matrix so as to learn the low-dimensional node vector representations. Then, it concatenates all the *k*-step representations as the final representations.
5) **DNE-APP** [11]: It is a deep network embedding algorithm which employs a semi-supervised SAE to learn the low-dimensional node vector representations from the aggregated *k*-step PPMI matrix. It incorporates pairwise constraint to make the node pairs with higher aggregated proximities have more similar hidden vector representations.
6) **LANE** [139]: It is an attributed network embedding algorithm, which aims to preserve both network topological proximity and node attribute affinity by the embedding vector representations. It incorporates label information into attributed network embedding so as to capture the correlations among network topological structures, node attributes and node labels.

Note that DeepWalk, GraRep, DNE-APP and LANE were originally developed for a single-network scenario. To tailor them to cross-network node classification, we construct a unified network containing all the nodes in the source network and the target network. Then, by utilizing the unified network for training, the cross-network proximities between nodes from the source network and the target network can be directly captured in the unified network.

5.5.4 Cross-network Node Classification

For cross-network node classification, all the nodes in the source network have observable labels, while in the target network, we randomly sample a very small fraction of nodes to give them accessible labels. Then, we run each baseline to learn the low-dimensional node vector representations with the same

Table 5.3: Cross-network node classification between the BlogCatalog1 and BlogCatalog2 networks when only 1% of labeled nodes are available in the target network. The training set as "T" indicates that only leveraging the labeled nodes in the target network for training, while "S+T" indicates that leveraging the labeled nodes from both the source network and the target network for training. The highest Micro-F1 and Macro-F1 scores among all the comparing algorithms are shown in Boldface.

$\mathcal{G}^s \rightarrow \mathcal{G}^t$		BlogCatalog1→BlogCatalog2		BlogCatalog2→BlogCatalog1	
Algorithms	**Training Set**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
CDNE	S+T	**0.6466**	**0.6452**	**0.6218**	**0.6192**
NetworkTr	S+T	0.5041	0.4965	0.5234	0.5162
TrAdaBoost	S+T	0.4742	0.4532	0.4817	0.4630
DeepWalk	T	0.3272	0.2717	0.3434	0.2979
	S+T	0.2856	0.2544	0.2483	0.2323
GraRep	T	0.4234	0.3850	0.4542	0.4303
	S+T	0.3049	0.2650	0.2615	0.2475
DNE-APP	T	0.3531	0.2980	0.3676	0.3251
	S+T	0.2721	0.2317	0.2879	0.2360
LANE	T	0.2221	0.1333	0.2202	0.1209
	S+T	0.5278	0.5190	0.5318	0.5270

Table 5.4: Cross-network node classification between the Citationv1 and DBLPv7 networks when only 1% of labeled nodes are available in the target network.

$\mathcal{G}^s \rightarrow \mathcal{G}^t$		Citationv1→DBLPv7		DBLPv7→Citationv1	
Algorithms	**Training Set**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
CDNE	S+T	**0.7642**	**0.7420**	**0.8099**	**0.7904**
NetworkTr	S+T	0.6600	0.6207	0.6722	0.6443
TrAdaBoost	S+T	0.6114	0.5673	0.6074	0.5761
DeepWalk	T	0.5787	0.4996	0.6773	0.6125
	S+T	0.3130	0.2724	0.4847	0.4297
GraRep	T	0.6207	0.5541	0.6982	0.6439
	S+T	0.4021	0.3495	0.5427	0.4796
DNE-APP	T	0.6165	0.5429	0.7128	0.6509
	S+T	0.5517	0.4945	0.6107	0.5609
LANE	T	0.4280	0.2718	0.4674	0.3759
	S+T	0.5961	0.5549	0.5728	0.5344

Table 5.5: Cross-network node classification between the Citationv1 and ACMv9 networks when only 1% of labeled nodes are available in the target network.

$\mathcal{G}^s \rightarrow \mathcal{G}^t$		**Citationv1→ACMv9**		**ACMv9→Citationv1**	
Algorithms	**Training Set**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
CDNE	S+T	**0.7899**	**0.7890**	**0.8263**	**0.8087**
NetworkTr	S+T	0.6406	0.6107	0.6643	0.6402
TrAdaBoost	S+T	0.5637	0.5387	0.6199	0.5966
DeepWalk	T	0.5983	0.5903	0.6773	0.6125
	S+T	0.3974	0.3721	0.4260	0.3635
GraRep	T	0.6308	0.6260	0.6982	0.6439
	S+T	0.5384	0.5307	0.4833	0.4243
DNE-APP	T	0.6658	0.6625	0.7128	0.6509
	S+T	0.5672	0.5566	0.6262	0.5644
LANE	T	0.4298	0.3222	0.4674	0.3759
	S+T	0.5688	0.5337	0.6020	0.5656

Table 5.6: Cross-network node classification between the DBLPv7 and ACMv9 networks when only 1% of labeled nodes are available in the target network.

$\mathcal{G}^s \rightarrow \mathcal{G}^t$		**DBLPv7→ACMv9**		**ACMv9→DBLPv7**	
Algorithms	**Training Set**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**
CDNE	S+T	**0.7829**	**0.7847**	**0.7543**	**0.7325**
NetworkTr	S+T	0.6171	0.5818	0.6364	0.6008
TrAdaBoost	S+T	0.5396	0.5099	0.5957	0.5493
DeepWalk	T	0.5983	0.5903	0.5787	0.4996
	S+T	0.3880	0.3407	0.4078	0.3502
GraRep	T	0.6308	0.6260	0.6207	0.5541
	S+T	0.4355	0.4350	0.3754	0.2803
DNE-APP	T	0.6658	0.6625	0.6165	0.5429
	S+T	0.5400	0.5332	0.4801	0.3776
LANE	T	0.4298	0.3222	0.4280	0.2718
	S+T	0.5416	0.4832	0.6004	0.5435

dimensionality as 128-D. Next, we train a one-vs-rest LR classifier based on the fully labeled nodes in the source network and the scarce labeled nodes in the target network, and leverage the classifier to predict the labels of the unlabeled nodes in the target network. To evaluate the cross-network node classification performance, we measure the Micro-F1 and Marco-F1 scores of the predictions for the testing nodes in the target network. The reported Micro-F1 and Macro-F1 scores are the average over the same 5 random splits, for each comparing algorithm.

Firstly, as shown in Tables 5.3, 5.4, 5.5 and 5.6, the proposed CDNE model always achieved the highest Micro-F1 and Macro-F1 scores for node classification in all the target networks, among all the comparing algorithms. In addition, the improvement of CDNE over the best baseline is rather significant. For example, as shown in Table 5.3, for cross-network node classification from BlogCatalog1 to BlogCatalog2, CDNE can achieve a 22.51% higher Micro-F1 score and also a 24.32% higher Macro-F1 score than the best baseline, i.e., LANE (S+T).

Secondly, we can see that among all the cross-network node classification tasks, both CDNE and NetworkTr always achieve much higher Micro-F1 and Macro-F1 scores than TrAdaBoost. Note that both CDNE and NetworkTr consider network connections and node attributes for cross-network node classification, however, TrAdaBoost only leverages the attribute information while ignoring network connections. Thus, it reflects that it is indeed necessary to design the network-specific transfer learning algorithms to consider the important network structural properties during knowledge transfer across networks. In addition, the proposed CDNE model always significantly outperforms NetworkTr for all the cross-network node classification tasks. This could be explained by the fact that we employ deep network embedding to learn a unified embedding space shared by the source network and the target network, which is more powerful to capture the non-linear properties of the complex underlying network structures than the matrix-factorization approach utilized by NetworkTr. Moreover, NetworkTr does not minimize the cross-network distribution discrepancy when learning the unified latent space. In contrast, we incorporate conditional MMD constraint into the overall loss function of CDNE to make the same labeled nodes across networks closer to each other in the unified embedding space, which is rather beneficial for cross-network node classification.

Thirdly, we found that the three state-of-the-art network embedding algorithms, i.e., DeepWalk, GraRep and DNE-APP, are rather unsuitable for cross-network node classification. As shown in Tables 5.3, 5.4, 5.5 and 5.6 when utilizing the fully labeled source network nodes and the limited labeled target network nodes together to train the classifier, DeepWalk (S+T), GraRep (S+T) and DNE-APP (S+T) perform even much worse than DeepWalk (T), GraRep (T) and DNE-APP (T) which only utilize 1% of the target network labeled nodes for training. This is because these network embedding algorithms only preserve the topological proximities between nodes within a single network, which makes the learned node vector representations incomparable across different networks [53] and leads to negative transfer. However, we can see that when leveraging the labeled nodes from both the source network and the target network for training, LANE (S+T) can always achieve much better performance than LANE (T) which just utilizes the scarce target network labeled nodes for training. This might be because LANE can capture the correlations among node attributes, network proximities and node labels, and such correlations might be generalized across different networks.

In addition, as shown in Table 5.3, for node classification across BlogCatalog1 and BlogCatalog2, NetworkTr, TrAdaBoost and LANE (S+T) can achieve significantly better performance than DeepWalk, GraRep and DNE-APP. However, as shown in Tables 5.4, 5.5 and 5.6, when the target network is Citationv1 or ACMv9, GraRep (T) and DNE-APP (T) can achieve even better performance than NetworkTr, TrAdaBoost and LANE (S+T). Note that NetworkTr, TrAdaBoost and LANE consider node attributes and network connections when learning the hidden node feature vector representations, while all of DeepWalk, GraRep and DNE-APP ignore the attribute information. The different results in the BlogCatalog and the citation networks reflect that the attribute information are more important and useful for node classification in BlogCatalog. This might be explained by the fact that the node labels in the citation networks are more perfectly satisfied with the homophily theory, i.e., the cited papers are more likely to belong to the same research area. However, in the dense BlogCatalog networks with abundant network connections, two connected bloggers might be interested in different groups. In such a case, only leveraging network connections to predict node labels might be misleading sometimes, while leveraging node attributes as the complementary information would significantly improve node classification performance.

Next, we vary the target network training fraction in $\{0.5\%, 1\%, 3\%, 5\%, 10\%\}$ and test the performance of the algorithms. Since DeepWalk, GraRep and DNE-APP are rather unsuitable for cross-network node classification, we report their results for single-network node classification by only utilizing the target network labeled nodes for training. As shown in Figure 5.2, as the target network training fraction increases, the DeepWalk (T), GraRep (T) and DNE-APP (T) algorithms will achieve significantly higher Micro-F1 and Macro-F1 scores. In addition, as shown in Figure 5.2(b), when the training fraction is more than 1% in the Citationv1 target network, GraRep (T) and DNE-APP (T) can achieve even much better performance than NetworkTr and TrAdaBoost. This is because the state-of-the-art single-network embedding algorithms can well preserve the proximities between connected nodes within a single network, which is necessary for learning informative node vector representations for node classification in a single network. On the other hand, although NetworkTr can learn the shared latent structural features across networks, it cannot guarantee such latent features to be label-discriminative for node classification. In addition, we can see that although LANE (S+T) achieved promising performance for cross-network node classification from BlogCatalog1 to BlogCatalog2, it

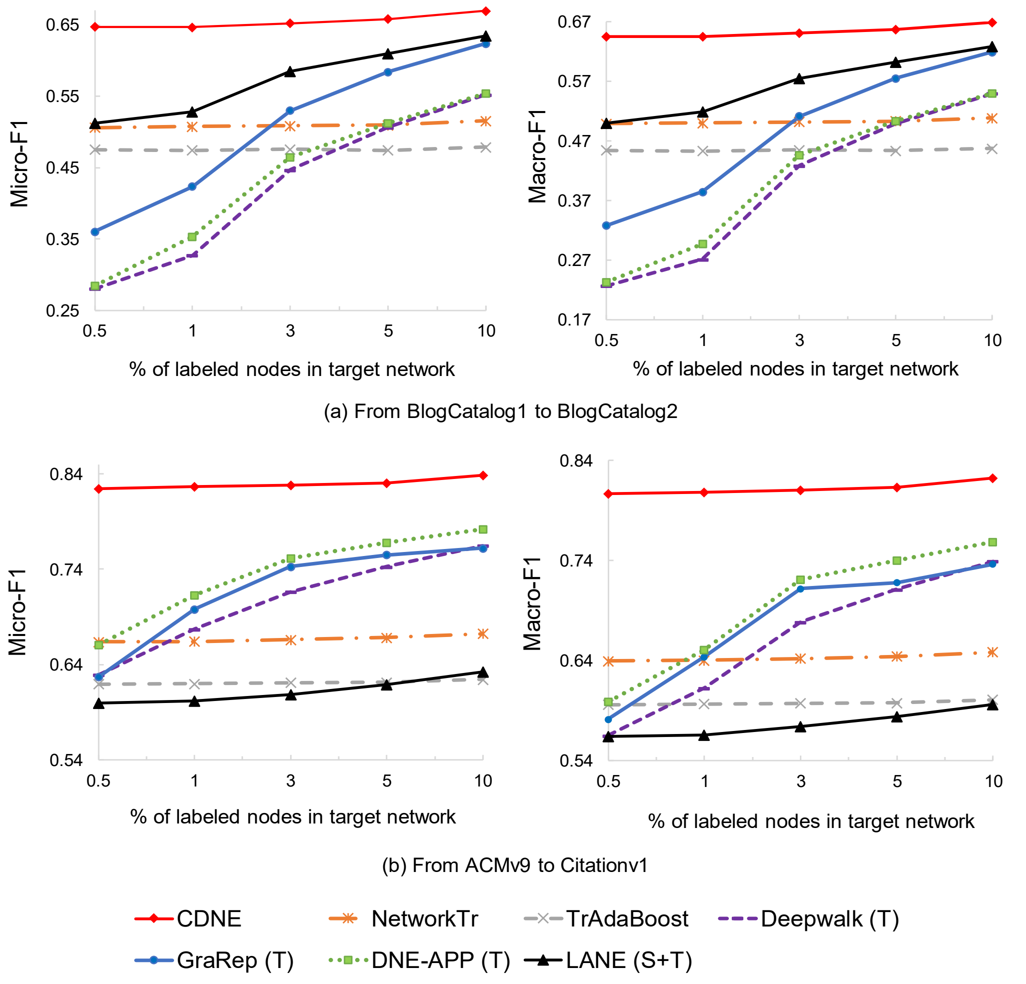


Figure 5.2: Micro-F1 and Macro-F1 scores of predicting labels for unlabeled nodes in the target network, with varied fractions of labeled nodes observed in the target network. (a) and (b) show the results of the cross-network node classification task from BlogCatalog1 to BlogCatalog2, and from ACMv9 to Citationv1, respectively.

performed the worst among all the comparing algorithms for the task from ACMv9 to Citationv1. It might because the citation networks are rather sparse, which makes the matrix factorization technique employed in LANE more challenging in capturing the correlations among node attributes, network proximities and node labels. The proposed CDNE model not only takes advantage of deep network embedding to learn the informative and discriminative node vector representations, but also incorporates MMD constraints to match the cross-network distributions. Thus, even with varied target network training fractions, the proposed CDNE model always achieves significantly better cross-network node classification results than the single-network embedding algorithms and the transfer learning baselines without deep network embedding.

5.5.5 Parameter Sensitivity

In this subsection, we analyze the sensitivity of the parameters $K, \beta, \alpha, \varphi, \mu, \gamma$ on the performance of CDNE.

Parameter $\boldsymbol{K}$ denotes the maximum step of neighbors utilized to measure the aggregated PPMI matrix. As shown in Figure 5.3(a), as K increases, both the Micro-F1 and Macro-F1 scores will increase. For example, setting K=6 yields a 6.51% higher Macro-F1 score and a 5.28% higher Micro-F1 score than K=1. This indicates that preserving high-order proximities is indeed useful for learning informative feature vector representations for node classification.

Parameter $\boldsymbol{\beta}$ indicates the ratio of penalty on the reconstruction errors of strong connections over that of weak connections and disconnections. As shown in Figure 5.3(b), $\beta > 1$ always yields much higher Micro-F1 and Macro-F1 scores than $\beta = 1$. Thus, adding larger penalty to make SAE_s and SAE_t focus more on reconstructing strong connections is indeed helpful for learning informative feature vector representations for node classification.

Parameter $\boldsymbol{\alpha}$ refers to the weight of pairwise constraint on strongly connected nodes. As shown in Figure 5.3(c), $\alpha > 0$ always achieves better cross-network node classification results than $\alpha = 0$. This demonstrates the effectiveness of incorporating the pairwise constraint into SAE_s and SAE_t to map more strongly connected nodes within a network closer to each other so as to well capture the homophily effect.

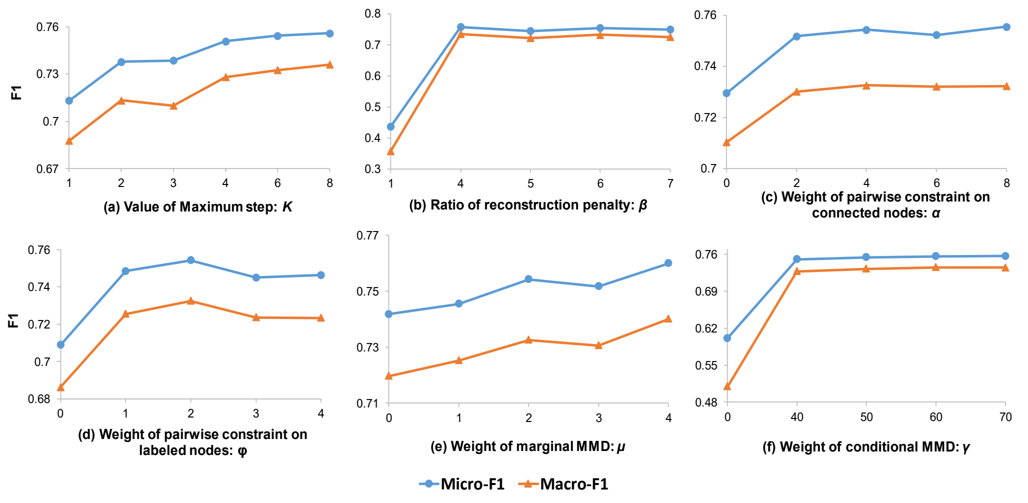


Figure 5.3: Sensitivity of the parameters, i.e., $K, \beta, \alpha, \varphi, \mu, \gamma$ on the cross-network node classification performance of CDNE, when the source network is ACMv9 and the target network is dblpv7, and 1% of nodes in the target networks are with observed labels.

Parameter $\boldsymbol{\varphi}$ denotes the weight of pairwise constraint on labeled nodes in the source network. As shown in Figure 5.3(d), $\varphi > 0$ always leads to much higher F1 scores than $\varphi = 0$. Thus, incorporating pairwise constraint to map nodes sharing more common labels closer to each other, while mapping nodes belonging to completely different categories far apart from each other is indeed effective for learning discriminative feature vector representations for node classification.

Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ represent the weight of marginal MMD and conditional MMD in the overall loss function of SAE_t, respectively. As shown in Figures 5.3(e) and 5.3(f), μ and γ with positive values would always lead to much better performance than with zero values. This reflects that incorporating the MMD constraints is indeed effective for learning network-invariant node vector representations. Moreover, we can see that $\gamma = 50$ leads to a 25.5% higher Micro-F1 score and a 43.7% higher Macro-F1 score than $\gamma = 0$. Thus, it demonstrates that incorporating the conditional MMD to map the same labeled nodes across networks to have similar latent vector representations should play a rather important role in achieving good performance in cross-network node classification.

5.6 Summary

In this work, we address a cross-network node classification problem. Given a target network with scarce labeled nodes and a source network with fully labeled nodes, we aim to leverage the abundant labeled information from the source network to help classify unlabeled nodes in the target network. The state-of-the-art single-network embedding algorithms can achieve promising results for node classification involving one network, however, they fail to learn generalized and comparable network representations across networks. To address this, we propose a novel cross-network embedding model, i.e., CDNE, which is composed of two semi-supervised SAEs, to embed the nodes from the source network and from the target network into a unified low-dimensional latent space. To well capture the homophily effect for each single network, we devise pairwise constraints to make more strongly connected nodes within a network have more similar hidden vector representations. In addition, to learn discriminative feature vector representations for node classification, we devise pairwise constraints to make the source network nodes sharing more common labels closer while those belonging to completely different categories far apart from each other in the embedding space. Moreover, to make the hidden vector representations comparable across networks, we incorporate the marginal and class-conditional MMD constraints to minimize the cross-network distribution discrepancy. Specifically, we employ the associated node attributes to predict fuzzy labels for the unlabeled nodes in the target network. Then, we leverage both the observable binary labels and the predicted fuzzy labels to guide the cross-network label-aligned embedding so as to learn similar hidden vector representations for the same category of nodes across networks. Extensive experimental results demonstrate that the proposed CDNE model achieves significant gains over the related algorithms for node classification in the target network.

This work is now under review in [17]. There are several directions to extend this work. Firstly, we can further exploit the abundant node attribute information as one of the input components to learn the node vector representations. In addition, we can address a more challenging task where nodes from the source network and the target network do not share the same categories of node attributes, and consequently how the CDNE model can be extended to select the commonly useful node attributes across networks. Moreover, instead of leveraging all the labeled nodes from the source network to help classify the unlabeled target network nodes, we can design a scheme to filter the useless source network nodes so as to achieve better cross-network node classification performance. Moreover, it is interesting to adapt CDNE to a more generalized cross-network node classification task, where the node labels in the source network and the target network can be not fully identical.

Chapter 6
Conclusions and Future Work

6.1 Conclusions

Complex networks are ubiquitous in many real-world scenarios, such as social networks, publication networks, biological networks and transportation networks. Mining information from complex networks is important for a wide variety of applications. To effectively and efficiently address the canonical graph mining tasks, such as node classification, node clustering, link prediction, node/link retrieval, one should pre-define a set of informative and discriminative feature vector representations for nodes or edges in the networks. Motivated by this, in this thesis, we proposed four models to learn the informative node or edge feature vector representations in the complex networks.

In Chapter 2, we proposed a cross-network learning (CNL) framework to address the cross-network prediction tasks over nodes and edges in the IM problem. Specifically, the proposed CNL model aims to leverage the greedy seed selection and influence propagation knowledge pre-learned from a smaller source network to help predict seed nodes and inactive edges in multiple larger target networks. In the CNL model, we employed a feature engineering approach to manually select explicit topological features which can reflect the influence of nodes in the IM problem. In addition, to address domain discrepancy, we assign lower weights to the features which perform less similarly between the source network and the target network. Moreover, we proposed an innovative fuzzy self-training for domain adaptation algorithm to iteratively retrain the prediction model based on not only the fully labeled instances in the source network, but also the most confident predicted instances in the target network with their predicted fuzzy labels. On one hand, for seed selection, since the CNL model learns the greedy seed selection knowledge from the source network, it can achieve a high influence spread comparable to the greedy algorithms in the target networks. In addition, since the CNL model heuristically selects seed nodes in the target network, it can greatly save the running time. On the other hand, for graph sparsification, by leveraging the influence propagation knowledge pre-learned from the source network, the CNL model would only remove the edges least useful for influence propagation in the target network. Thus, it would just cause a little loss of influence spread in the sparse target networks, while significantly speeding up the greedy algorithms.

In addition, motivated by the recent success of deep learning, we proposed three state-of-the-art deep network embedding models to automatically learn the low-dimensional node vector representations, which can tackle a wide variety of graph mining tasks. These three deep network embedding models aim to address the open issues in current network embedding research, namely asymmetric network embedding, signed network embedding and cross-network embedding.

Most existing network embedding algorithms ignore the important asymmetric relationships between nodes in a network. To address this, in Chapter 3, we proposed an asymmetry-aware deep network embedding (AsDNE) model to preserve the asymmetric outward and inward proximities between nodes, in both directed and undirected networks. We employed two semi-supervised SAEs, i.e., SAE-Out and SAE-In, to learn the non-linear outward and inward node vector representations, respectively. To well capture the asymmetric relationships, we incorporated pairwise constraints into SAE-Out and SAE-In to map node pairs with bi-directionally strong connections much closer than those with strong connection in only one direction. Extensive experimental results in both directed and undirected real-world networks demonstrate that the proposed AsDNE model can learn task-independent network representations outperforming the state-of-the-art network embedding algorithms.

In addition, the vast majority of existing network embedding algorithms are developed for unsigned networks, while the signed networks have pretty distinct properties from the unsigned networks. Thus, it is necessary to design the signed network embedding algorithms to capture the distinct properties of the signed networks. In this regard, in Chapter 4, we proposed a deep network embedding with structural balance preservation (DNE-SBP) model to learn deep graph representations for the signed networks. We employed a semi-supervised SAE to reconstruct the adjacency connections of a signed network. To preserve the structural balance property of the signed networks, we incorporated pairwise constraints into the SAE to map positively connected nodes much closer than negatively connected nodes. Extensive experimental results in the real-world signed networks demonstrate the superiority of the proposed DNE-SBP model over the state-of-the-art network embedding algorithms for graph representation learning in signed networks.

Existing network embeddings are generally developed for a single network while failing to capture the proximities between nodes across different networks. In Chapter 5, we proposed a cross-network deep network embedding (CDNE) model to learn label-discriminative and network-invariant node vector representations. We should be the first to integrate deep network embedding and domain adaptation to address the cross-network node classification task. We employed two semi-supervised SAEs, i.e., SAE_s and SAE_t, to embed the nodes from the source network and from the target network into a unified low-dimensional latent space. In addition, we leveraged network structures, node attributes and node labels to capture the proximities between nodes not only within a network but also across different networks. Extensive experimental results in the real-world networks demonstrate that the proposed CDNE model significantly outperforms the state-of-the-art single-network embedding algorithms and the traditional transfer learning algorithms (without deep network embedding), for node classification in the target network.

In summary, in this thesis, we not only utilize a feature-engineering approach to manually define explicit topological features for nodes or edges in the networks, but also employ the state-of-the-art deep network embedding models to automatically learn the low-dimensional latent feature vector representations. In addition, we leverage the learned feature vector representations to address various graph mining tasks within a single network, such as node classification, node clustering, link prediction, and node/edge retrieval. Moreover, we are among the first to incorporate the domain adaptation techniques in the graph mining tasks across different networks, such as cross-network node classification and cross-network link prediction. We believe that the innovative ideas and proposed techniques in this thesis would inspire the research in complex network analysis, graph mining, and graph representation learning within a single network and across different networks.

6.2 Future Work

There are several directions to extend the existing network embedding work.

6.2.1 Signed Network Embedding

Existing signed network embedding algorithms [114], [113], [45], [44], [12] only focus on preserving the structural balance properties of the signed networks. However, the signed networks also have other distinct properties, as described in

[51]. For example, the nodes connected with positive links are with higher clustering coefficients than those connected with negative links. In addition, the positively connected nodes would have stronger tendency to form bi-directional connections than those negatively connected ones. Moreover, the structural balance theory is naturally defined for undirected networks [51]. Thus, existing signed network embedding algorithms with structural balance preservation would be problematic when applied to the directed signed networks. For example, in a directed signed network, if node v_i has a positive link towards v_j, while v_j has a negative link towards v_i, then according to $e_{ij} = 1$, v_i and v_j should be mapped close to each other, while according to $e_{ji} = -1$, v_j and v_i should be mapped far apart from each other. To address this issue, we can consider another important property of the signed networks, i.e., status theory, which is relevant to the directed signed networks [150], [52]. The status theory suggests that v_i has a higher status than v_j, if there is a positive link from v_j to v_i or a negative link from v_i to v_j. Thus, in the future work, we can design the signed network embedding algorithms to capture other important properties of the signed networks.

6.2.2 Cross-network Embedding

The proposed CDNE model should be the first cross-network deep network model for learning label-discriminative and network-invariant feature vector representations for cross-network node classification. Several issues can be further investigated. Firstly, in the CDNE model, we leverage all the labeled nodes from the source network to help classify the unlabeled target network nodes. In the future research, it is better to design a scheme to filter the useless source network nodes which might cause negative transfer. Also, we can design a scheme to re-weight the latent features according to their generalization across networks. For example, we can employ the proposed CNL model to address the feature incompatibility between the source network and the target network and assign lower weights to the features which perform less similarly between the two networks. In addition, in the CDNE model, we assume that the nodes from the source network and the target network share the same categories of node attributes. A more challenging task can be addressed where the source network and the target network do not share the identical node attributes, and by then, we need to design algorithms to automatically select the commonly useful node attributes across networks. Besides, most recently, several attributed network embedding algorithms [151], [152], [97], [153] have been proposed to employ node attributes as the main input for learning network representations, with the goal of preserving both attributed affinity and network topological proximities. Thus, instead of utilizing node attributes as the side information, we can exploit them as the main input to learn the cross-network node vector representations. Moreover, instead of utilizing SAEs as the main building block in the deep network model, we can also try some other deep neural network architectures, such as convolutional neural network, variational auto-encoder and generative adversarial networks.

6.2.3 Heterogeneous Network Embedding

In this thesis, we focus on network embedding in homogeneous networks. However, in a heterogeneous network, there are more than one type of nodes or more than one type of edges. For example, in a recommender system, nodes can be users or items, and edges can be the relationships between two users, between one user and one item, or between two items. Similarly, in a question answering system, a node can be a question, an answer, or a user, and the edges can be any relationships among different types or the same type of nodes. In the future work, we can extend the ideas and the proposed models in this thesis to heterogeneous network embedding. For example, we can extend our signed network embedding model, i.e., DNE-SBP, to a recommender system, by modeling users and items as two types of nodes. Then, we can model the “like” and “dislike” relationships between users and items as the positive and negative links between users and items, and model the “similar” and “dissimilar” relationships between two items as the positive and negative links between items. Next, we can make recommendations for users based on the link sign prediction results between users and items and conduct customer segmentation according to the network clustering results of users.

Bibliography

[1] N. R. Council, *Network Science*. Washington, DC: The National Academies Press, p. 124, 2005.

[2] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM),* vol. 3, no. 3, pp. 1-13, 2007.

[3] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, and A. Ben-Hur, "A large-scale evaluation of computational protein function prediction," *Nature Methods,* vol. 10, no. 3, p. 221, 2013.

[4] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1293-1299, 2014.

[5] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2252-2258, 2016.

[6] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 203-209, 2017.

[7] Z. Wang, L. Dueñas-Osorio, and J. E. Padgett, "A new mutually reinforcing network node and link ranking algorithm," *Scientific Reports,* vol. 5, p. 15141, 2015.

[8] X. Shen, S. Mao, and F.-l. Chung, "Cross-network learning with fuzzy labels for seed selection and graph sparsification in influence maximization," *under 3rd round review by IEEE Transactions on Fuzzy Systems,* 2018.

[9] X. Shen, F.-l. Chung, and S. Mao, "Leveraging cross-network information for graph sparsification in influence maximization," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 801-804, 2017.

[10] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225-1234, 2016.

[11] X. Shen and F.-l. Chung, "Deep network embedding with aggregated proximity preserving," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pp. 40-44, 2017.

[12] X. Shen and F.-l. Chung, "Deep network embedding for graph representation learning in signed networks," *IEEE Transactions on Cybernetics,* 2018.

[13] J. Tang, T. Lou, and J. Kleinberg, "Inferring social ties across heterogenous networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 743-752, 2012.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345-1359, 2010.

[15] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *Proceedings of the International Conference on World Wide Web*, pp. 1477-1488, 2013.

[16] M. Fang, J. Yin, and X. Zhu, "Transfer learning across networks for collective classification," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 161-170, 2013.

[17] X. Shen, Q. Dai, S. Mao, and F.-l. Chung, "Network together: Node classification via cross-network deep network embedding," *submitted to ACM SIGIR Conference on Research and Development in Information Retrieval,* 2019.

[18] M. Fang, J. Yin, X. Zhu, and C. Zhang, "TrGraph: Cross-network transfer learning via common signature subgraphs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 27, no. 9, pp. 2536-2549, 2015.

[19] R. Niu, S. Moreno, and J. Neville, "Analyzing the transferability of collective inference models across networks," in *Proceedings of IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 908-916, 2015.

[20] Q. Hu, G. Wang, and S. Y. Philip, "Transferring influence: Supervised learning for efficient influence maximization across networks," in *Proceedings of International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pp. 45-54, 2014.

[21] H. Lamba and R. Narayanam, "A novel and model independent approach for efficient influence maximization in social networks," in *Proceedings of International Conference on Web Information Systems Engineering*, pp. 73-87, 2013.

[22] P. J. Carrington, J. Scott, and S. Wasserman, *Models and Methods in Social Network Analysis*. Cambridge University Press, 2005.

[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855-864, 2016.

[24] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701-710, 2014.

[25] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip! Online learning of multi-scale network embeddings," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pp. 258-265, 2017.

[26] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2942-2948, 2017.

[27] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135-144, 2017.

[28] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear

embedding," *Science,* vol. 290, no. 5500, pp. 2323-2326, 2000.

[29] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pp. 891-900, 2015.

[30] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1105-1114, 2016.

[31] J. T. Pang, F. Nie, and J. Han, "Flexible orthogonal neighborhood preserving embedding," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2592-2598, 2017.

[32] F. Nie, W. Zhu, and X. Li, "Unsupervised large graph embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2422-2428, 2017.

[33] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, "High-order proximity preserved embedding for dynamic networks," *IEEE Transactions on Knowledge and Data Engineering,* 2018.

[34] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1145-1152, 2016.

[35] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119-128, 2015.

[36] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of International Conference on Machine Learning*, pp. 2014-2023, 2016.

[37] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *arXiv preprint arXiv:1705.02801,* 2017.

[38] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *arXiv preprint arXiv:1711.08752,* 2017.

[39] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *arXiv preprint arXiv:1801.05852,* 2017.

[40] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering,* 2018.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of International Conference on Learning Representations,* 2013.

[42] X. Shen, S. Mao, and F.-l. Chung, "Asymmetry-aware deep network embedding," *under 2nd round review by IEEE Transactions on Knowledge and Data Engineering,* 2018.

[43] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale

information network embedding," in *Proceedings of the International Conference on World Wide Web*, pp. 1067-1077, 2015.

[44] Q. Zheng and D. B. Skillicorn, "Spectral embedding of signed networks," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 55-63, 2015.

[45] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proceedings of the SIAM International Conference on Data Mining*, 2017.

[46] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200-2207, 2013.

[47] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4119-4125, 2015.

[48] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of International Conference on Machine Learning*, pp. 513-520, 2011.

[49] F. Wu and Y. Huang, "Sentiment domain adaptation with multiple sources," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 301-310, 2016.

[50] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137-146, 2003.

[51] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Computing Surveys (CSUR),* vol. 49, no. 3, 2016.

[52] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1361-1370, 2010.

[53] M. Heimann and D. Koutra, "On generalizing neural node embedding methods to multi-network problems," in *Proceedings of Mining and Learning with Graphs Workshop*, 2017.

[54] M. Heimann, H. Shen, and D. Koutra, "Node representation learning for multiple networks: The case of graph alignment," *arXiv preprint arXiv:1802.06257,* 2018.

[55] J. Nail, "The consumer advertising backlash," *Forrester Research and Intelliseek Market Research Report,* 2004.

[56] I. R. Misner, *The World's Best Known Marketing Secret: Building Your Business With Word-Of-Mouth Marketing*. Bard Press, 1994.

[57] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, N. Glance, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 420-429, 2007.

[58] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "CELF++:optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the International Conference Companion on World Wide Web*, pp. 47-48, 2011.

[59] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 199-208, 2009.

[60] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1029-1038, 2010.

[61] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in *Proceedings of the International Conference on Data Mining (ICDM)*, pp. 918-923, 2011.

[62] Z. L. Luo, W. D. Cai, Y. J. Li, and D. Peng, *A PageRank-based heuristic algorithm for influence maximization in the social network*. Springer Berlin Heidelberg, pp. 485-490, 2012.

[63] B. Wilder and G. Sukthankar, "Sparsification of social networks using random walks," in *Proceedings of International Conference on Social Computation*, 2015.

[64] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 529-537, 2011.

[65] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 2456-2464, 2011.

[66] S. Kumar, A. K. Shukla, P. K. Muhuri, and Q. D. Lohani, "Atanassov intuitionistic fuzzy domain adaptation to contain negative transfer learning," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 2295-2301, 2016.

[67] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Granular fuzzy regression domain adaptation in takagi-sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems,* vol. 26, no. 2, pp. 847-858, 2017.

[68] J. Shell and S. Coupland, "Fuzzy transfer learning: methodology and application," *Information Sciences,* vol. 293, pp. 59-79, 2015.

[69] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in Takagi-Sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems,* vol. 25, no. 6, pp. 1795-1807, 2017.

[70] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 57-66, 2001.

[71] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi, "Fast and accurate influence maximization on large networks with pruned Monte-Carlo simulations," in *Proceedings*

of the AAAI Conference on Artificial Intelligence, pp. 138-144, 2014.

[72] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng, "Staticgreedy: solving the scalability-accuracy dilemma in influence maximization," in *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)* pp. 509-518, 2013.

[73] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-K influential nodes in mobile social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1039-1048, 2010.

[74] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pp. 946-957, 2014.

[75] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 75-86, 2014.

[76] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 695-710, 2016.

[77] J. Tang, X. Tang, and J. Yuan, "Influence maximization meets efficiency and effectiveness: A hop-based approach," in *Proceedings of IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pp. 64-71, 2017.

[78] J. Kim, S.-K. Kim, and H. Yu, "Scalable and parallelizable processing of influence maximization for large-scale social networks," in *Proceedings of IEEE International Conference on Data Engineering*, pp. 266-277, 2013.

[79] T. N. Dinh, H. Zhang, D. T. Nguyen, and M. T. Thai, "Cost-effective viral marketing for time-critical campaigns in large-scale social networks," *IEEE/ACM Transactions on Networking,* vol. 22, no. 6, pp. 2001-2011, 2014.

[80] Y. Shen, T. N. Dinh, H. Zhang, and M. T. Thai, "Interest-matching information propagation in multiple online social networks," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, pp. 1824-1828, 2012.

[81] Q. Zhan, J. Zhang, S. Wang, S. Y. Philip, and J. Xie, "Influence maximization across partially aligned heterogenous social networks," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pp. 58-69, 2015.

[82] S. Kullback, "Letter to the Editor: The Kullback-Leibler Distance," *American Statistician,* vol. 41, 1987.

[83] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian, "Fast influence-based coarsening for large networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1296-1305, 2014.

[84] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the

web," in *Proceedings of the International Conference on World Wide Web*, pp. 613-622, 2001.

[85] X. Yi, X. Shen, W. Lu, T. S. Chan, and F.-l. Chung, "Persuasion driven influence analysis in online social networks," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 4451-4456, 2016.

[86] M. Newmann, *Networks*. Oxford University Press, 2010.

[87] M. E. Newman, "The mathematics of networks," *The New Palgrave Encyclopedia of Economics,* vol. 2, no. 2008, pp. 1-12, 2008.

[88] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM,* vol. 46, no. 5, pp. 604-632, 1999.

[89] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," Stanford Info Lab, Technical Report,1999.

[90] V. Behbood, J. Lu, G. Zhang, and W. Pedrycz, "Multistep fuzzy bridged refinement domain adaptation algorithm and its application to bank failure prediction," *IEEE Transactions on Fuzzy Systems,* vol. 23, no. 6, pp. 1917-1935, 2015.

[91] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems,* vol. 42, no. 1, pp. 181-213, 2015.

[92] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics,* vol. 6, no. 1, pp. 29-123, 2009.

[93] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Proceedings of International Semantic Web Conference*, pp. 351-368, 2003.

[94] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 3111-3119, 2013.

[95] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Transactions on Fuzzy Systems,* vol. 25, no. 4, pp. 1006-1012, 2017.

[96] S. Mao, X. Shen, and F.-l. Chung, "Deep domain adaptation based on multi-layer joint kernelized distance," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1049-1052, 2018.

[97] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of International Conference on Learning Representations*, 2017.

[98] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 2177-2185, 2014.

[99] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational Linguistics,* vol. 16, no. 1, pp. 22-29, 1990.

[100] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings*

of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 817-826, 2009.

[101] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval,* vol. 3, no. 2, pp. 127-163, 2000.

[102] S. Nandanwar and M. N. Murty, "Structural neighborhood based classification of nodes in a network," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1085-1094, 2016.

[103] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Governance in social media: A case study of the Wikipedia promotion process," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pp. 98-105, 2010.

[104] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges," in *Proceedings of the International Conference on World Wide Web*, pp. 741-750, 2009.

[105] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions. com community," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 121-126, 2005.

[106] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[107] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of International Conference on World Wide Web*, pp. 641-650, 2010.

[108] G. Beigi, J. Tang, S. Wang, and H. Liu, "Exploiting emotional information for trust/distrust prediction," in *Proceedings of SDM International Conference on Data Mining*, pp. 81-89, 2016.

[109] J. Wang, J. Shen, P. Li, and H. Xu, "Online matrix completion for signed link prediction," in *Proceedings of the ACM International Conference on Web Search Data Mining*, pp. 475-484, 2017.

[110] P. V. Marsden and N. E. Friedkin, "Network studies of social influence," *Sociological Methods & Research,* vol. 22, no. 1, pp. 127-151, 1993.

[111] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology,* vol. 27, no. 1, pp. 415-444, 2001.

[112] J. Tang, X. Hu, and H. Liu, "Is distrust the negation of trust?: the value of distrust in social media," in *Proceedings of the ACM Conference on Hypertext and Social Media*, pp. 148-157, 2014.

[113] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak, "Spectral analysis of signed graphs for clustering, prediction and visualization," in *Proceedings of SIAM International Conference on Data Mining*, pp. 559-570, 2010.

[114] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, "Scalable clustering of signed networks using balance normalized cut," in *Proceedings of ACM Conference on Information and*

Knowledge Management (CIKM), pp. 615-624, 2012.

[115] F. Harary, "On the notion of balance of a signed graph," *The Michigan Mathematical Journal,* vol. 2, no. 2, pp. 143-146, 1953.

[116] J. A. Davis, "Clustering and structural balance in graphs," *Human Relations,* vol. 20, no. 2, pp. 181-187, 1967.

[117] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, "Sitting closer to friends than enemies, revisited," in *Proceedings of International Symposium on Mathematical Foundations of Computer Science*, pp. 296-307, 2012.

[118] P. Mercado, F. Tudisco, and M. Hein, "Clustering signed networks with the geometric mean of Laplacians," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 4421-4429, 2016.

[119] C.-J. Hsieh, K.-Y. Chiang, and I. S. Dhillon, "Low rank modeling of signed networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 507-515, 2012.

[120] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE Transactions on Cybernetics,* vol. 47, no. 1, pp. 27-37, 2017.

[121] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Transactions on Cybernetics,* vol. 47, no. 4, pp. 1017-1027, 2017.

[122] S. M. Siniscalchi and V. M. Salerno, "Adaptation to new microphones using artificial neural networks with trainable activation functions," *IEEE Transactions on Neural Network and Learning System,* vol. 28, no. 8, pp. 1959-1965, 2017.

[123] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 585-591, 2002.

[124] Z. Yu, Y. Lu, J. Zhang, J. You, H.-S. Wong, Y. Wang, and G. Han, "Progressive semisupervised learning of multiple classifiers," *IEEE Transactions on Cybernetics,* 2017.

[125] Z. Yu, Y. Zhang, J. You, C. P. Chen, H.-S. Wong, G. Han, and J. Zhang, "Adaptive semi-supervised classifier ensemble for high dimensional data classification," *IEEE Transactions on Cybernetics,* 2017.

[126] P. He, X. Xu, K. Hu, and L. Chen, "Semi-supervised clustering via multi-level random walk," *Pattern Recognition,* vol. 47, no. 2, pp. 820-832, 2014.

[127] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering," in *Proceedings of the International Conference on Machine Learning*, pp. 307-314, 2002.

[128] Z. Yu, Z. Kuang, J. Liu, H. Chen, J. Zhang, J. You, H.-S. Wong, and G. Han, "Adaptive ensembling of semi-supervised clustering solutions," *IEEE Transactions on Knowledge and Data Engineering,* vol. 29, no. 8, pp. 1577-1590, 2017.

[129] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 153-160, 2007.

[130] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision,* vol. 88, no. 2, pp. 303-338, 2010.

[131] G. Facchetti, G. Iacono, and C. Altafini, "Computing global structural balance in large-scale signed social networks," *Proceedings of the National Academy of Sciences,* vol. 108, no. 52, pp. 20953-20958, 2011.

[132] A. Amelio and C. Pizzuti, "Community mining in signed networks: a multiobjective approach," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pp. 95-99, 2013.

[133] Y. Chen, S. Song, S. Li, L. Yang, and C. Wu, "Domain space transfer extreme learning machine for domain adaptation," *IEEE Transactions on Cybernetics,* 2018.

[134] Y.-H. Hubert Tsai, Y.-R. Yeh, and Y.-C. Frank Wang, "Learning cross-domain landmarks for heterogeneous domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5081-5090, 2016.

[135] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: A generalized framework for domain adaptation," *IEEE Transactions on Cybernetics,* 2018.

[136] F. Wu, Z. Yuan, and Y. Huang, "Collaboratively training sentiment classifiers for multiple domains," *IEEE Transactions on Knowledge and Data Engineering,* vol. 29, no. 7, pp. 1370-1383, 2017.

[137] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1345-1354, 2016.

[138] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385-394, 2017.

[139] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 731-739, 2017.

[140] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1895-1901, 2016.

[141] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1823-1829, 2016.

[142] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using

network embedding," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1774-1780, 2016.

[143] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 126-135, 2006.

[144] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the International Conference on Machine Learning*, pp. 193-200, 2007.

[145] X. Wang, Y. Ma, Y. Cheng, L. Zou, and J. J. Rodrigues, "Heterogeneous domain adaptation network based on autoencoder," *Journal of Parallel and Distributed Computing,* 2017.

[146] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 513-520, 2007.

[147] A. Mackiewicz and W. Ratajczak, "Principal components analysis (PCA)," *Computers and Geosciences,* vol. 19, pp. 303-342, 1993.

[148] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proceedings of the ACM International on Conference on Information and Knowledge Management*, pp. 1041-1050, 2015.

[149] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 990-998, 2008.

[150] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the International Conference on World Wide Web*, pp. 403-412, 2004.

[151] H. Gao and H. Huang, "Deep attributed network embedding," in *IJCAI*, pp. 3364-3370, 2018.

[152] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International Conference on Machine Learning*, pp. 40-48, 2016.

[153] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, "ANRL: Attributed network representation learning via deep neural networks," in *IJCAI*, pp. 3155-3161, 2018.