EFFECTIVE REPRESENTATION LEARNING FOR GRAPH-STRUCTURED DATA WITH ADVERSARIAL LEARNING

QUANYU DAI
PhD
The Hong Kong Polytechnic University
2020

The Hong Kong Polytechnic University
Department of Computing

Effective Representation Learning for Graph-Structured Data with Adversarial Learning

Quanyu DAI

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

March 2020

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.

_____ (Signed)

Quanyu DAI (Name of student)

Abstract

Graph-structured data is widely existed in real-world applications such as social networks, paper citation networks and protein-protein interaction networks. It encodes very rich information about data entities in graph structures through the complicated connections among them. How to extract such abundant information is an important and challenging problem which has attracted a great amount of attention from both academia and industry. Traditional methods rely on hand-engineered features which are both ineffective and inefficient. In recent years, representation learning emerges as the most promising way for modeling graph-structured data, which aims to learn low-dimensional vectors for nodes in the graph. The learned node representations can further be utilized to facilitate downstream learning tasks such as network analysis (e.g., node classification and link prediction), recommendation, and fraud detection. This technique is called graph representation learning or network embedding in the literature.

In this thesis, we aim to learn effective node representations for both plain networks and attributed networks with the assistance of adversarial learning including adversarial learning principle based on generative adversarial networks (GANs) and adversarial training methods from adversarial machine learning.

For plain network embedding, we design regularization methods and sampling method to enhance embedding learning. Specifically, we first propose a global regularization method for deep embedding models via GANs, of which a prior distribution is imposed on embedding vectors to help alleviate overfitting. Then, we introduce a succinct and effective local regularization method, namely adversarial training, for negative sampling based embedding models such as DeepWalk, LINE and node2vec,

which can improve both model robustness and generalization performance. Furthermore, we propose an adversarial ranking network embedding model to preserve node similarity rankings in representations, which unifies a triplet sampling phase and an embedding learning phase with the framework of GANs. It can encourage the generation of more difficult and relevant negative nodes for given positive target-context node pairs to improve representation learning. For attributed network embedding, we focus on a challenging cross-network learning problem that aims to transfer the label information from an attributed source network to an attributed target network. Specifically, we propose a novel network transfer learning framework AdaGCN via adversarial domain adaptation and graph convolution, which enables the learning of both class discriminative and domain invariant node representations and thus facilitates cross-network node classification. Extensive empirical evaluations on benchmark datasets demonstrate the effectiveness of the proposed methods.

Publications Arising from the Thesis

1. **Quanyu Dai**, Qiang Li, Jian Tang, and Dan Wang, "Adversarial Network Embedding", *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, Louisiana, USA, February 2-7, 2018.

2. **Quanyu Dai**, Xiao Shen, Liang Zhang, Qiang Li and Dan Wang, "Adversarial Training Methods for Network Embedding", *The World Wide Web Conference (WWW)*, San Francisco, CA, USA, May 13-17, 2019.

3. **Quanyu Dai**, Qiang Li, Liang Zhang and Dan Wang, "Ranking Network Embedding via Adversarial Learning", *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference (PAKDD)*, Macau, China, April 14-17, 2019.

4. **Quanyu Dai**, Xiao Shen, Xiao-Ming Wu and Dan Wang, "Network Transfer Learning via Adversarial Domain Adaptation with Graph Convolution", *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Under Review.

5. Xiao Shen, **Quanyu Dai**, Sitong Mao, Fu-lai Chung, and Kup-Sze Choi, "Network Together: Node Classification via Cross-network Deep Network Embedding", *IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2019.* To appear.

6. Zimu Zheng, Yuqi Wang, **Quanyu Dai**, Huadi Zheng and Dan Wang, "Metadata-driven Task Relation Discovery for Multi-task Learning", *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, China, August 10-16, 2019.

7. Yikai Wang, Liang Zhang, **Quanyu Dai**, Fuchun Sun, Bo Zhang, Yang He, Weipeng Yan and Yongjun Bao, "A Regularized Adversarial Sampling Strategy for CTR Prediction with Time-aware Attention Embedding", *The 28th ACM International Conference on Information and Knowledge Management (CIKM)*, Beijing, China, November 03-07, 2019.

8. Xiao Shen, **Quanyu Dai**, Fu-lai Chung, Wei Lu, and Kup-Sze Choi, "Adversarial Deep Network Embedding for Cross-network Node Classification", *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, New York, USA, February 7-12, 2020.

9. Junyang Chen, Zhiguo Gong, **Quanyu Dai**, Chunyuan Yuan, and Weiwen Liu, "Adversarial Learning for Overlapping Community Detection and Network Embedding", *The 24th European Conference on Artificial Intelligence (ECAI)*, 2020.

10. Yumin Su, Liang Zhang, **Quanyu Dai**, Bo Zhang, Jinyao Yan, Sulong Xu, Dan Wang, Yang He, Yongjun Bao and Weipeng Yan., "An attention-based model for conversion rate prediction with delayed feedback via post-click calibration", *The 29th International Joint Conference on Artificial Intelligence (IJCAI)*, Yokohama, Japan, July 11-17, 2020.

Acknowledgements

This might be the most difficult part to write in the whole thesis, since I have so much to say such as my expectations, my disappointments, my sadnesses, bewilderments and joys all through these years but I cannot think of any perfect words to express them. I decide to retrospect and taste all those feelings again myself in some rainy nights. Here, there is only one word for myself: be patient, positive, and dedicated.

It is not easy to purse a Ph.D. degree. I cannot achieve what I have without the selfless encouragements and supports from my colleagues, my friends and my family.

First, I would like to express my sincere gratitude to my colleagues who helped me so much in my research. Dr. Dan Wang, my supervisor, has taught me a lot during the past few years. I can purse my own research interests because of his kind tolerance, patience and wise guidance. Dr. Liang Zhang shared with me a lot about his experiences in research and gave me many valuable advice during my postgraduate years. Dr. Qiang Li not only inspired me a lot with his solid and profound professional knowledge, but also helped improve my research writings by his own examples during our collaborations. Dr. Xiao-Ming Wu gave me many encouragements at critical times, impressed me with her rigorous attitude of scholarship, and taught me a lot with her precious experiences selflessly. The collaborations with Dr. Xiao Shen, Dr. Jian Tang, Dr. Zimu Zheng and Mr. Yikai Wang also benefited me a lot.

Second, I would like to express my heartfelt thanks to my friends. Dr. Wenshu Zeng shared with me a lot about his feelings, understandings, ideas and imaginations of research and life. The times spent with Mr. Tianxian Yang and Mr. Runjie Tan are filled with laughters. The times spent in the fitness room or baseketball court

with my friends such as Dr. Yu Lei, Dr. Lei Han, Dr. Chuang Hu, Dr. Shuhang Gu, Dr. Bo Tang, Dr. Wenjian Xu, Dr. Shang Gao and Mr. Yu An Huang are so relaxed. In addition, Dr. Kunfeng Lai, Dr. Yi Yuan, Dr. Abraham Hang-Yat Lam, Dr. Wengen Li, Dr. Lei Xue, Dr. Tsz Nam Chan, Dr. Linchuan Xu, Dr. Ziqiang Cao, Dr. Zhitao Wang, Dr. Jiaxing Shen, Dr. Hui Li, Mr. Kang Xu, Mr. Jianliang Gao, Dr. Bo Sun and many others also gave me many kind helps.

Finally and most importantly, I would like to express my deepest gratitude to my dear families including my mother, my sisiters, my brother, my little nephew and my brother-in-law for their unconditional love and support.

I respectfully dedicate this thesis to memory of my father.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graph-structured data is ubiquitous in real-world applications, which can help organize complicated relationships among data entities. Examples include paper citation networks, social networks and protein-protein interaction networks, where all these real networks can be abstracted as graph consisting of nodes and edges between nodes. Substantial efforts have been devoted to analyzing graph-structured data with various research goals, such as classifying nodes in the graph (node classification) [110, 62], predicting edges between nodes (link prediction) [2, 72], and visualizing graph in low-dimensional space (network visualization) [126, 125]. Besides, modeling graph structures also plays vital role in a broad range of applications, such as recommendations [119, 151], fraud detection [28, 142], natural language processing [44, 133] and computer vision [70, 58]. Thus, how to extract the abundant information from graph to facilitate these learning tasks and applications is a continuous research problem, which has attracted great interests from both academia and industry.

However, modeling graph-structured data is a challenging problem because of its special characteristics. We summarize these characteristics as follows:

- **Discreteness and non-linearity.** Graph-structured data is discrete in nature. Many existing techniques, such as machine learning algorithms with

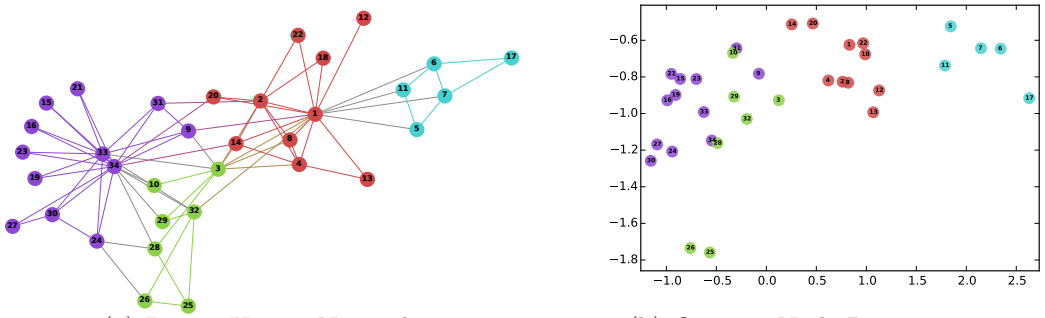
vector-based inputs and gradient descent technique for optimization, are inapplicable for modeling graph directly. Nodes in the network are correlated with each other in a complicated way. There can be first-order proximity, second-order proximity, and high-order proximities among nodes. It is highly difficult to model such non-linear structures.

- **Sparsity and high-dimensionality.** Network data is usually very high-dimensional and sparse in real-world applications. For example, there are billions of users in Facebook social network, but each user only connects to a few hundred others on average. How to process such huge amounts of data with reasonable time and space consumption and how to capture user dependencies by leveraging such sparse connectivity patterns pose great challenges to network analysis methods.

- **Noisiness and incompleteness.** In real-world applications, there could be many noisy connections among nodes. For example, influencer marketing with fake followers is widely existed in Twitter [21], which tries to improve the importance of certain user in the social network by adding artificial connections. Such manipulation causes great challenge for learning tasks. Besides, it is common to have many missing links in a network, which further exacerbates the sparsity issue and makes it difficult to capture the inherent node dependencies.

Traditional methods extract structural information from graphs with user-defined heuristics such as summary of graph statistics (e.g., node degrees or clustering coefficients) [8], kernel functions [141] and carefully engineered features to measure node neighborhood structure [72], which are both ineffective and inefficient in modeling graph structures [47].

In recent years, learning node representations for graph-structured data automatically has become the most effective way to extract useful structural informa-



(a) Input: Karate Network. (b) Output: Node Representations.

Figure 1.1: Visualization of Zachary's Karate network [159] (left) and the two dimensional node representations of Zachary's Karate network from DeepWalk (right). Different colors represent different communities detected by modularity-based clustering. As shown by the figure, node representations learned by DeepWalk can well preserve community structure information. Figures 1.1(a) and 1.1(b) are directly extracted from DeepWalk [95].

tion [95, 126, 62]. In the literature, it is termed network embedding or graph representation learning, i.e., learning low-dimensional vector for each node in the network with the complicated relations between nodes encoded in the embedding vectors. The fundamental purpose is to learn generalizable node representations which can be used for the widely existed machine learning algorithms to facilitate learning tasks. The whole process could be conducted in a phased or end-to-end manner. The former is to learn node representations first and then utilize the learned features as input for the downstream learning tasks in a seperate phase, while the latter conducts these two phases in a mutually enhanced manner. We use a toy example from [95] to illustrate this technique intuitively as shown in Figure 1.1. It is based on an unsupervised network embedding method DeepWalk [95] which aims to preserve the structural information in low-dimensional vectors.

Graph representation learning methods have been widely applied in industry. There are two major application scenarios:

- **Enriching user and item profiles.** In social networks, such as WeChat

and Weibo, users are explicitly organized in a graph through the friendship or follower/followee relations. In recommender systems, such as Amazon and Taobao, user-user, user-item, and item-item relations can be easily constructed based on the purchasing behaviors of users and other knowledge. There are many other examples. To model these graph-structured data, one of the most important approaches is to leverage graph representation learning technique to extract feature vectors for nodes automatically [95, 126, 41]. Such learned feature vectors are then utilized as additional user/item features to facilite various applications.

- **End-to-end learning for a specific application.** In many real-world scenarios, the main purpose is to improve a specific learning task, such as predicting user-item interactions in recommender systems [136, 156] and identifying supicious users in financial applications [28]. To improve these applications, graph representation leanrning techniques are directly integrated into the model framework as the feature extractor.

In these applications, we need to handle different types of network data, such as pure network and network with side information (e.g., node attributes and node labels). To handle these complex data, a series of network embedding methods have been proposed, including plain network embedding [95, 126, 41], attributed network embedding [162, 91, 150], semi-supervised network embedding [154, 62, 46], and multi-network embedding [149, 86, 114].

This thesis is dedicated to designing effective representation learning algorithms for both plain networks and attributed networks.

1.1 Motivation

For plain network embedding, existing works mainly focus on preserving network structures and properties in node representations based on network topology. The connection information is actually considered as pseudo labels for capturing node similarities. For example, LINE [126] and SDNE [143] aim to learn first-order and second-order proximities, of which the former refers to observed links and the latter depends on the similarity of node neighborhoods. In the learning process, the embedding vectors are optimized to fit to such information based on different models, such as negative sampling technique in LINE and autoencoder in SDNE. However, the sparsity, noiseness, and incompleteness of many real-world networks are largely neglected by the existing methods, which can easily result in overfitting issue. Thus, designing regularization method for network embedding models is highly important. In the thesis, we aim to tackle this problem by leveraging adversarial learning principle and adversarial training technique with the former based on generative adversarial networks (GANs) [40] and the latter based on adversarial machine learning [39]. The first idea is to add some prior knowledge on embedding vectors via adversarial learning to avoid overfitting by leveraging the distribution fitting capability of GANs, which is presented in Chapter 3. The second idea is to design adversarial training regularization method for network embedding to help improve model robustness and generalization ability, which is demonstrated in Chapter 4.

Sampling negative nodes for given positive target-context node pairs is an indispensable process for many network embedding methods such as those based on negative sampling technique [82] (e.g. DeepWalk, LINE and node2vec). The negative sampling technique is a simplified variant of negative contrastive estimation [45], which can help speed up the training process of the model. However, this strategy may generate false negative samples that violate pairwise relationships presented in

the network structure and thus may generate less desirable node representations, since the negative samples are constructed according to a simple modified unigram noise generation process. While such drawback exists, these models are widely used in both academic and industrial applications. In the thesis, we manage to design better sampling method for generating negative nodes for given positive target-context pairs via adversarial learning technique [39]. The basic idea is to unify the sampling process and embedding learning process with generative adversarial networks, of which the generator is leveraged for sampling more difficult and relevant negative nodes. This is presented in Chapter 5.

The majority of existing network embedding methods are designed for single network learning, which inevitably suffer from the sparsity, incompleteness, or label insufficiency issues of a single network in many application scenarios. Meanwhile, massive amount of information networks are available, and many of them are similar or related to each other because of common nodes, cross-network connections or common node attributes across networks. Leveraging such valuable and abundant information across multiple networks is a promising direction for tackling the problems in a single network as mentioned above. However, existing embedding methods [95, 126, 41] cannot easily generalize to multi-network learning. For example, plain network embedding methods are inapplicable for multi-network learning due to lack of a similarity preserving component to push similar nodes from different networks close in the embedding space [50]. In the thesis, we aim to boost representation learning in a single network by leveraging information of multiple similar or related networks. Specifically, graph convolution [62, 69] is leveraged to capture the information of both network structures and node attributes for learning useful node representations, while adversarial domain adaptation [35, 111] is exploited to mitigate the discrepancy between different network domains as presented in Chapter 6.

1.2 Thesis Overview and Contributions

This thesis builds upon my works on plain network embedding [23, 26, 24] and multi-network embedding [25]. First, Chapter 2 summarizes representative and related network embedding methods. Second, in Chapter 3, we present a global regularization method for deep embedding models based on generative adversarial networks. It imposes a prior distribution such as Gaussian on embedding vectors through adversarial learning to help alleviate overfitting. Third, in Chapter 4, we introduce a succinct and effective local regularization method, namely adversarial training, to network embedding so as to improve model robustness and generalization performance. This method can be applied to a series of negative sampling based models such as DeepWalk, LINE and node2vec. Next, an adversarial sampling method is described in Chapter 5, which can be utilized to sample high-quality negative nodes for given positive target-context node pairs to facilitate graph representation learning. Then, in Chapter 6, we study a multi-network learning problem that aims to learn both label-discriminative and domain-invariant node representations for two different but related networks to assist in node classification in the target network. A novel learning framework namely AdaGCN is presented, which builds upon graph convolution and adversarial domain adaptation techniques. Finally, we draw a conclusion of the thesis and describe possible directions for future work in Chapter 7. We further summarize the main contributions of the thesis as follows:

Chapter 3: Network Embedding with Prior Regularization via Adversarial Learning [23]. Chapter 3 presents an Adversarial Network Embedding (ANE) framework for robust representation learning. It consists of two components, including a structure-preserving component for capturing structural information and an adversarial learning component to impose a prior distribution on embedding vectors. Specifically, we design an inductive DeepWalk for representation learn-

ing, which exploits random walk for exploring node neighborhoods and optimizes negative sampling loss, but employs multi-layer perceptron to generate embedding vectors. Besides, the adversarial learning component is acting as a regularizer to alleviate overfitting, which consists of a generator and a discriminator. The generator is the representation learner shared with the structure-preserving component, while the discriminator is a multi-layer perceptron for distinguishing representation sources. Different from GANs [28], in our framework, a prior distribution is selected as the data distribution for generating real data, while the embedding vectors are regarded as fake samples. Through adversarial learning, the embedding vectors outputted by the generator will be matched to the prior distribution.

Experiments on benchmark datasets including Cora, Citeseer and Wiki demonstrate the effectiveness of the ANE framework. To the best of my knowledge, this is the first work to design network embedding model with the adversarial learning principle for regularization.

Chapter 4: Adversarial Training Methods for Network Embedding [26]. Chapter 4 presents a local regularization method, namely adversarial training (AdvT), for network embedding. AdvT forces the model to be robust to adversarial examples generated from the clean ones with small crafted perturbation, which can help achieve local smoothness of model parameters and thus improve model robustness and generalization ability. The original AdvT method is designed for vector-based data such as image. To adapt it for graph-structured data, we define adversarial examples in the embedding space instead of the discrete graph domain to circumvent the difficulties of generating adversarial examples for discrete nodes and connections. We also design adaptive L_2 norm constraints on adversarial perturbations by leveraging the connectivity patterns of nodes to enable more reasonable regularization. To improve its interpretability, we further design an adversarial training method by restricting the perturbation directions towards the embedding vectors of other nodes,

which might help reconstruct adversarial examples in the discrete graph domain.

We conduct node classification and link prediction on benchmark datasets including Cora, Citeseer and Wiki to evaluate the proposed AdvT regularization method. It demonstrates that AdvT can help greatly improve the performance of the base network embedding model.

Chapter 5: Ranking Network Embedding via Adversarial Learning [24]. Chapter 5 presents a Ranking Network Embedding (RNE) framework with two ranking strategies, i.e., a vanilla strategy based on uniform sampling method and an adversarial strategy based on generative adversarial networks. The latter improves over the former by generating high-quality negative nodes for given positive target-context node pairs via adversarial learning. Specifically, in addition to a discriminator for capturing node similarity rankings via a triplet ranking loss, a generator is exploited for sampling negative nodes given target node. In the learning process, the discriminator tries to pull similar nodes closer in the embedding space, while pushing dissimilar nodes apart. The generator aims to generate difficult negative nodes for the given target from a set of candidates. The model is trained with the reinforced algorithm due to the discrete sampling process.

We empirically evaluate the proposed vanilla and adversarial RNE models through network visualization, link prediction and node classification, on several benchmark datasets. Experimental results show that both models achieve competitive performance with state-of-the-art baselines and negative sampling via adversarial learning outperforms the uniform sampling strategy.

Chapter 6: Network Transfer Learning via Adversarial Domain Adaptation with Graph Convolution [25]. Chapter 6 presents a network transfer learning framework AdaGCN based on adversarial domain adaptation with graph convolution. It tackles a challenging cross-network node classification problem where a partially labeled attributed source network is leveraged to assist in node classifi-

cation in a different completely unlabeled or partially labeled attributed target network. On one hand, graph convolution naturally combines network structures and node attributes for learning useful node representations, which helps to ease knowledge transfer across networks and node classification in the target networks. On the other hand, adversarial domain adaptation component helps mitigate the domain divergence between the source and target networks to enable label information of the source network transferring to the target network.

Extensive empirical evaluations on real-world datasets show that AdaGCN can successfully transfer class information with a low label rate on the source network and a substantial divergence between the source and target domains for both unsupervised and semi-supervised settings. The label efficiency and model robustness against domain discrepancy enable its application for solving a wide range of real-world problems.

Chapter 2

Literature Review

In this section, we will review existing network embedding methods according to the types of input including network topolgy only, network with side information, and multiple networks.

2.1 Plain Network Embedding

Plain network embedding models take only network topology as input. They are categorized in Table 2.1 according to their utilized techniques and goals. The categories of models are not mutually exclusive. From the perspective of techniques, network embedding methods belong to skip-gram based methods [95, 126, 41, 96, 100, 78], matrix factorization based methods [12, 163, 98, 90, 147, 97, 160], deep learning models [13, 143, 144, 101, 63, 132, 23, 158] , and others such as adversarial learning enhanced models [36, 24] and adversarial training regularized method [26]. From the perspective of model goals, network embedding models include structure-preserving methods [95, 126, 41, 96, 100, 12, 163], property-perserving methods [90, 147, 79, 42], regularization-enhanced methods [23, 158, 26] and efficiency-pursuing methods [97, 160, 15]. In the rest of this section, we will present a review of existing plain network embedding models according to model goals. The main purpose is to summarize and analyze some of the representative models in each category

Table 2.1: Plain network embedding methods.

	Structure-preserving	Property-preserving	Regularization	Efficiency
Skip-gram	DeepWalk [95], LINE [126], node2vec [41], WALKLETS [96], struc2vec [100], SNS [78]	-	-	-
Matrix factorization	GraRep [12], AROPE [163], NetMF [98], M-NMF [147]	HOPE [90]	-	NetSMF [97], ProNE [160]
Graphical model	NetHiex [79], vGraph [121]	RaRE [42], CNE [56]	-	-
Deep learning	DNGR [13], SDNE [143], GraphGAN [144], DeepGL [101], PRUNE [63]	DRNE [132], DVNE [165]	**ANE*** [23], NETRA [158]	DeepGL [101]
Others	HARP [14], Poincaré [87], R-NS [5], NEU [153], ASeedNE [36], **A-RNE*** [24]	-	**Dwns_AdvT*** [26]	FastRP [15]

* My works are highlighted in bold.

and the most relevant models to our works instead of reviewing all existing methods exhaustively.

2.1.1 Structure Preserving Network Embedding

Structure-preserving methods mainly aim to preserve network structural information such as first-order and second-order proximities, latent community structures and hierarchical structures in the embedding vectors. We review these methods according to their utilized techniques.

The Skip-gram model or the negative sampling technique is widely leveraged for capturing node neighborhood structures in embedding models such as DeepWalk [95], LINE [126], node2vec [41] and WALKLETS [96]. DeepWalk first explores neighborhood structures through node sequence sampling with truncated random walk, and then learns the latent representations using Skip-gram model [82] by regarding node sequences as sentences. Both node2vec and WALKLETS are extensions of DeepWalk. node2vec improves random-walk based sampling by introducing two hyper-parameters for more flexible sampling so as to balance local and global structures.

WALKLETS is designed for learning multiscale node representations from multiscale relationships generated by subsampling short random walks from different powers of the adjacency matrix. Besides, LINE tries to preserve first-order and second-order proximities in two separate objective functions, and then directly concatenates the representations.

Matrix factorization technique is also widely exploited for learning structure-preserving node representations. Representative models include GraRep [12], AROPE [163], NetMF [98] and M-NMF [90]. Both GraRep and AROPE aim to preserve high-order proximities, NetMF unifies several negative sampling based models including DeepWalk, LINE, PTE and node2vec with a matrix factorixation framework, while M-NMF [147] learns community structure preserving embedding vectors by building upon the modularity based community detection model [**?**].

Both NetHiex [79] and vGraph [121] model network structures through graphical models. NetHiex tries to capture the latent hierarchical taxonomy of nodes by employing the nested Chinese restaurant process to guide the search of the most plausible hierarchical taxonomy. vGraph models the generation of node neighbors through node to community assignment and community to node generation process.

Meanwhile, deep learning embedding models [13, 143, 144] have also been proposed to capture highly non-linear network structures. DNGR [13] takes advantages of deep denoising autoencoder for learning compact node embeddings, which can capture the non-linear structural information with the high model capacity and improve model robustness with the denoising criterion. SDNE [143] modifies the framework of stacked autoencoder to learn both first-order and second-order proximities simultaneously. GraphGAN [144] leverages generative adversarial networks to facilitate node representation learning, which unifies the generative models and discriminative models of network embedding to boost the performance.

2.1.2 Property Preserving Network Embedding

Aside from the above mentioned structure-preserving methods, many research works investigate the learning of property-aware network embeddings.

Network transitivity, as the driving force of link formation, is considered in [90] by approximating high-order proximities which are based on asymmetric transitivity such as Katz Index, Rooted PageRank and Adamic-Adar. Node popularity, as another important factor affecting link generation, is incorporated into RaRE [42] to learn social-rank aware and proximity-preserving embedding vectors via a probabilistic link formation model. CNE [56] models the posterior distribution of embedding vectors conditioned on given network and applies Bayes rule to allow the consideration of prior knowledge for property-preserving. Some important prior knowedge includes knowledge about overall network density, knowledge about the individual node degrees, and knowledge about the edge density within or between particular subsets of nodes. DRNE [132] preserves regular equivalence in the embedding vectors by designing a deep recursive embedding model via a layer normalized LSTM which takes a sequence of node neighbors as input.

To model the uncertainty of nodes, many existing works use Gaussian distribution as node representation, such as HCGE [48] for heterogeneous graph, KG2E [106] for knowledge graph, Graph2Gauss for attributed network [9] and DVNE [165] for plain network. DVNE learns a Gaussian distribution in the Wasserstein space as the latent representation of each node with a deep variational model, which can preserve network structures and transitivity and model node uncertainty.

2.1.3 Efficient Network Embedding

Some structure-preserving and property-preserving models are also scalable to large networks, such as DeepWalk, LINE and node2vec. However, they may still suffer

from inefficiency issue due to the necessity of learning node representations for extremely large-scale networks in real applications. For example, it takes months for DeepWalk to learn embeddings for a network with tens of millions of nodes and edges as shown in [97]. To overcome this challenge and fulfill real applications, some works are dedicated to designing efficient algorithms for learning node representations [97, 160, 15].

Both NetSMF [97] and ProNE [160] tackles this problem with sparse matrix factorization. NetSMF first sparsifies the dense NetMF matrix [98] by leveraging the theories from spectral sparsification which guarantees the sparsified matrix is spectrally close to the original one with a theoretically bounded approximation error, and then learns node representations with truncated singular value decomposition (tSVD). Differently, ProNE directly factorizes NetMF matrix with the randomized tSVD to achieve a speedup over tSVD for embedding learning first, and then improves node representations via spectral propagation. FastRP [15] first constructs a normalized node similarity matrix that captures high-order proximitie, and then obtains node representations with very sparse random projection to achieve high efficiency. They all can achieve much better model efficiency than some existing scalable methods such as DeepWalk, LINE and node2vec.

Three of my works including ANE [23], DWNS_AdvT and A-RNE are proposed for effective representation learning in plain networks. Specifically, ANE and DWNS_AdvT is designed to handle the noisy and incomplete network data in real applications so as to alleviate overfitting and improve generalization performance. A-RNE leverages the framework of generative adversarial networks to help sample high-quality negative nodes to facilitate structure-preserving embedding learning. More detailed analysis of the relations and differences between these works and existing works will be presented in the corresponding chapters in the rest of the thesis.

2.2 Attributed Network Embedding

Aside from topology-only methods, many models are proposed to incorporate other information associated with networks such as node attributes [162, 91, 150], edge attributes [16, 53] and node labels [154, 54].

2.2.1 Unsupervised Attributed Network Embedding

Unsupervised attributed network embedding methods leverage node attributes or (and) edge attributes to improve node representation learning. TADW [152] enhances DeepWalk with text information associcated with nodes. It formulates DeepWalk with a matrix factorization framework and incorporates text information into the framework for better representation learning. IIRL [150] explores the semantic meanings of links by jointly modeling network structures and node contents. It captures structure-close links and content-close links through two types of node representations, i.e. identity representations and interest representations, respectively. PGE [53] uses attributes of nodes and edges to improve the sampling strategy of random walk.

Deep learning methods are also designed for modeling attributed networks [162, 91, 91]. Autoencoder based methods include ANRL [162] and DANE [91]. ANRL optimizes both network structure preserving loss and attributes reconstruction loss with stacked autoencoders. DANE differs from ANRL by utilizing Skip-gram model to learn network structures. VGAE [61] is a variational graph autoencoder framework which consists of a graph convolutional network encoder and a simple inner product decoder for attributed network embedding. ARVGA [91] extends our work ANE [23] to attributed networks by leveraging VGAE for node representation learning.

2.2.2 Semi-Supervised Attributed Network Embedding

Semi-supervised network embedding methods can learn more discriminative node representations by leveraging available node labels compared with unsupervised embedding methods. In [131], the authors adapted DeepWalk to semi-supervised setting by introducing an additional loss of max-margin SVM to regularize representaion learning. TriDNE [92] is specially designed for networks with text information associated with nodes, which jointly captures the inter-node, node-word, and label-word relationships with the Skip-gram based model framework. LANE [54] jointly models network topology and partially available node labels via matrix decomposition. These models all benefit node classification implicitly since they are not directly optimized for it.

Another branch of related works are proposed for semi-supervised node classification [154, 62]. Planetoid [154] jointly optimizes the supervised loss and context-preserving loss to achieve better performance in semi-supervised learning. GCN [62] is a deep convolutional learning paradigm for graph-structured data which nicely integrates local node attributes and network structures in the convolutional layers. GraphSAGE [46] is a variant of GCN which designs different methods such as mean aggregator and LSTM aggregator for aggregating features from node neighbors. GAT [138], as another variant of GCN, leverages attention mechanism to aggregate features from the neighbors of a node with discrimination. These graph convolution based methods achieve state-of-the-art performance in semi-supervised learning.

2.3 Multi-Network Embedding

Some methods aim to leverage the relationship between multiple networks to facilitate graph representation learning, including those relying on inter-network edges [149,

86] and those managing to transfer knowledge from the source network(s) to the target network(s) [33, 114].

Both EOE [149] and DMNE [86] learn embeddings for multiple networks simultaneously. Specifically, EOE introduces a harmonious embedding matrix to model inter-network node similarities, while DMNE adapts autoencoder for multi-network embedding with a co-regularized loss to manipulate cross-network relationships. These methods heavily rely on the existence of cross-network connections.

There is also some literature focusing on transferring knowledge from source network(s) to target network(s) [33, 64]. In [33], non-negative matrix factorization technique is jointly conducted on label propagation matrices of both the source and target networks so as to learn transferable structural node representations. However, it suffers from expensive computation in the matrix decomposition process, and it cannot jointly model the relationships among structural information, node attributes and node labels, which might cause negative transfer. CDNE [114] learns node embeddings for multiple networks with different stacked autoencoders and mitigates the distribution shift of node representations between networks by minimizing the MMD loss. However, it heavily relies on the preprocessing of the adjacency matrix with positive pointwise mutual information (PPMI) matrix, which will densify the adjacency matrix and thus aggravate the computation complexity due to the autoencoder based model architecture.

My work AdaGCN [25] can learn both class discriminative and domain invariant node representations for two networks via graph convolution and adversarial domain adaptation. It achieves state-of-the-art performance in cross-network node classification. In Chapter 6, more detailed explanations of the relations and differences between AdaGCN and existing works will be provided.

Chapter 3

Network Embedding with Prior Regularization via Adversarial Learning

Learning low-dimensional representations of networks has proved effective in a variety of tasks such as node classification, link prediction and network visualization. Existing methods can effectively encode different structural properties into the representations, such as neighborhood connectivity patterns, global structural role similarities and other high-order proximities. However, except for objectives to capture network structural properties, most of them suffer from lack of additional constraints for enhancing the robustness of representations. In this chapter, we aim to exploit the strengths of generative adversarial networks in capturing latent features, and investigate its contribution in learning stable and robust graph representations. Specifically, we propose an Adversarial Network Embedding (ANE) framework, which leverages the adversarial learning principle to regularize the representation learning. It consists of two components, i.e., a structure preserving component and an adversarial learning component. The former component aims to capture network structural properties, while the latter contributes to learning robust representations

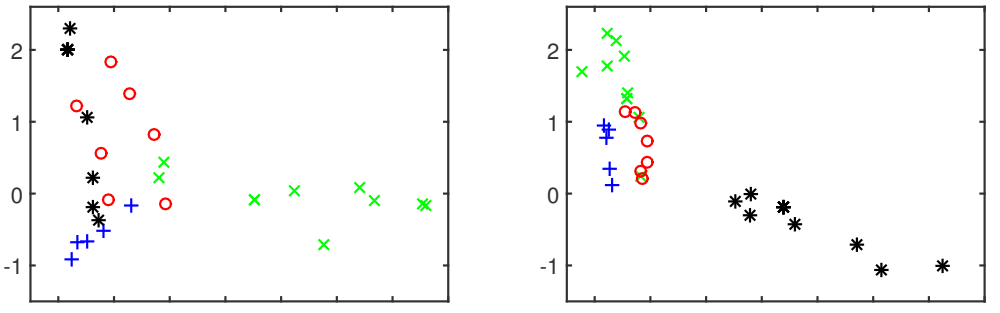
by matching the posterior distribution of the latent representations to given priors. As shown by the empirical results, our method is competitive with or superior to state-of-the-art approaches on benchmark network embedding tasks.

3.1 Introduction

Graph is a natural way of organizing data objects with complicated relationships, and encodes rich information of nodes in the graph. For example, paper citation networks capture the information of innovation flow, and can reflect topic relatedness between papers. To analyze graphs, an efficient and effective way is to learn low-dimensional representations for nodes in the graph, i.e., node embedding [95, 126, 12]. The learned representations should encode meaningful semantic, relational and structural information, so that they can be used as features for downstream tasks such as network visualization, link prediction and node classification. Network embedding is a challenging research problem because of the high-dimensionality, sparsity and non-linearity of the graph data.

In recent years, many methods for network embedding have been proposed, such as DeepWalk [95], LINE [126] and node2vec [41]. They aim to capture various connectivity patterns in network during representation learning. These patterns include relations of local neighborhood connectivity, first and second order proximities, global structural role similarities (i.e. structural equivalence), and other high-order proximities. As demonstrated in the literature, network embedding methods were shown to be more effective in many network analysis tasks than some classical approaches, such as Common Neighbors [72] and Spectral Clustering [129].

Though existing methods are effective in structure preserving with different carefully designed objectives, they suffer from lack of additional constraints for enhancing the robustness of the learned representations. When processing noisy network data,



(a) IDW. (b) AIDW.

Figure 3.1: Visualization of two dimensional representations of Zachary's Karate network [159] from Inductive DeepWalk (IDW) and Adversarial Inductive DeepWalk (AIDW). Different colors represent different communities detected by modularity-based clustering. As shown by the figure, AIDW can better capture community structure information, demonstrating that adversarial learning contributes to learning more meaningful and robust representations.

which is very common in real-world applications, these unsupervised network embedding techniques probably result in poor representations [77, 89]. Thus, it is critical to consider some amount of uncertainty in the process of representation learning. One famous technique for robust representation learning in unsupervised manner is denoising autoencoder [140]. It obtains stable and robust representations by recovering clean input from the corrupted one, that is denoising criterion. In [13], the authors have applied this criterion for network embedding. Recently, many generative adversarial models [99, 80, 29, 30] have also been proposed for learning robust and reusable representations. They have been shown to be effective in learning representations for image [99] and text data [38]. However, none of such models have been specially designed for dealing with graph data.

In this chapter, we propose a novel approach called Adversarial Network Embedding (ANE) for learning robust network representations by leveraging the principle of adversarial learning [39]. In addition to optimize the objective for preserving network structures, a process of adversarial learning is incorporated for modeling the data un-

certainty. Figure 3.1 presents an illustrative example with the well-known Zachary's Karate network on the effect of adversarial learning. By comparing representations of two schemes without/with adversarial learning regularization, it can be easily found that the latter scheme obtains more meaningful and robust representations.

More specifically, ANE naturally combines a structure preserving component and an adversarial learning component in a unified framework. The former component can help capture network structural properties, while the latter contributes to the learning of more robust representations through adversarial training with samples from some prior distribution. For structure preserving, we propose an inductive variant of DeepWalk that is suitable for our ANE framework. It maintains random walk for exploring neighborhoods of nodes and optimizes similar objective function, but employs parameterized function to generate embedding vectors. Besides, the adversarial learning component consists of two parts, i.e., a generator and a discriminator. It is acting as a regularizer for learning stable and robust feature extractor, which is achieved by imposing a prior distribution on the embedding vectors through adversarial training. To the best of our knowledge, this is the first work to design network embedding model with the adversarial learning principle. We empirically evaluate the proposed ANE approach through network visualization and node classification on benchmark datasets. The qualitative and quantitative results prove the effectiveness of our method.

3.2 Adversarial Network Embedding

In this section, we will first introduce the problem definition and notations to be used. Then, we will present an overview of the proposed adversarial network embedding framework, followed by detailed descriptions of each component.

3.2.1 Problem Definition and Notations

Network embedding is aimed at learning meaningful representations for nodes in information network. An information network can be denoted as $\mathcal{G} = (V, E, A)$, where V is the node set, E is a set of edges with each representing the relationship between a pair of nodes, and A is a weighted adjacency matrix with its entries quantifying the strength of the corresponding relations. Particularly, the value of each entry in A is either 0 or 1 in an unweighted graph specifying whether an edge exists between two nodes. Given an information network $\mathcal{G}$, network embedding is doing a mapping from nodes $v_i \in V$ to low-dimensional vectors $\boldsymbol{u_i} \in R^d$ with the formal format as follows: $f : V \mapsto U$, where $\boldsymbol{u_i^T}$ is the ith row of U ($U \in R^{N \times d}$, $N = |V|$) and d is the dimension of representations. We call U representation matrix. These representations should encode structural information of networks.

3.2.2 An Overview of the Framework

In this work, we leverage adversarial learning principle to help learn stable and robust representations. Figure 3.2 shows the proposed framework of *Adversarial Network Embedding* (ANE), which mainly consists of two components, i.e., a structure preserving component and an adversarial learning component. Specifically, the structure preserving component is dedicated to encoding network structural information into the representations. These information include the local neighborhood connectivity patterns, global structural role similarities, and other high-order proximities. There are many possible alternatives for the implementation of this component. Actually, existing methods [95, 126, 12] can be considered as structure preserving models, but without any constraints to help enhance the robustness of the representations. In this chapter, we propose an inductive DeepWalk for structure preserving. It maintains random walk for exploring neighborhoods of nodes and optimizes similar

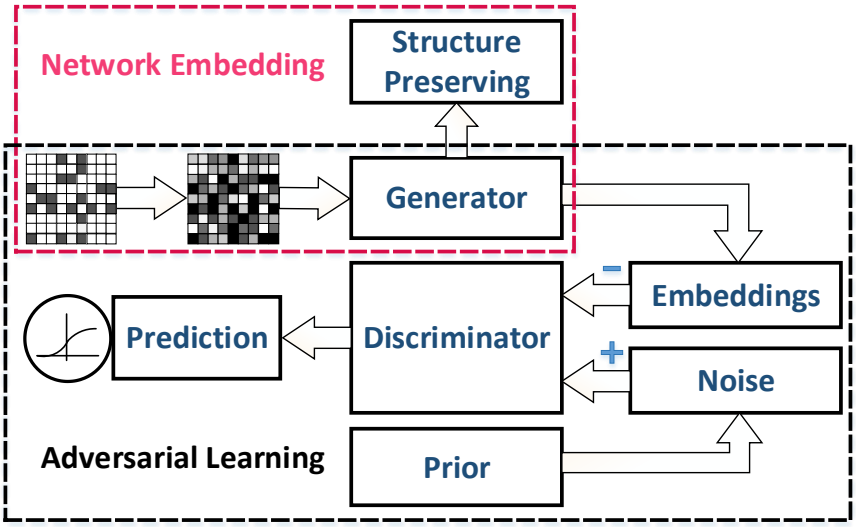


Figure 3.2: Adversarial Network Embedding Framework

objective function, but employs parameterized function $G(\cdot)$ to generate embedding vectors. In the training process, parameters of $G(\cdot)$ are directly updated instead of the embedding vectors. Besides, the adversarial learning component consists of two parts, i.e., a generator $G(\cdot)$ and a discriminator $D(\cdot)$. It is acting as a regularizer for learning stable and robust feature extractor, which is achieved by imposing a prior distribution on the embedding vectors through adversarial training. It needs to emphasize that the parameterized function $G(\cdot)$ is shared by both the structure preserving component and the adversarial learning component. These two components will update the parameters of $G(\cdot)$ alternatively in the training process.

3.2.3 Graph Preprocessing

In real-world applications, information network is usually extremely sparse, which may result in serious over-fitting problem when training deep models. To help alleviate the sparsity problem, one commonly used method is to preprocess the adjacency matrix with high-order proximities [126, 12]. In this chapter, we employ the shifted positive pointwise mutual information (PPMI) matrix X [66] as input features for

the generator*, which is defined as

$$X_{ij} = \max\{\log(\frac{M_{ij}}{\sum_k M_{kj}}) - \log(\beta), 0\}, \tag{3.1}$$

where $M = \hat{A} + \hat{A}^2 + \cdots + \hat{A}^t$ can capture different high-order proximities, $\hat{A}$ is the 1-step probability transition matrix obtained from the weighted adjacency matrix A after a row-wise normalization, and β is set to $\frac{1}{N}$ for experiments. Row vector $\boldsymbol{x}_i^T$ in X is the feature vector characterizing the context information of node v_i in the graph $\mathcal{G}$, but with high-dimension.

3.2.4 Structure Preserving Model

Ideally, existing unsupervised network embedding methods can be utilized as structure preserving component in our framework for encoding node dependencies into representations. However, many of them are transductive methods with an embedding lookup as embedding generator such as DeepWalk and LINE, which are not directly suitable for the generator of the adversarial learning component since we utilize parameterized generator as standard GANs. With parameterized generator, our framework can well deal with networks with node attributes and explore non-linear properties of network with deep learning models. In this work, we design an inductive variant of DeepWalk that is applicable for both weighted and unweighted graphs. Theoretically, it can also generalize to unseen nodes for networks with node attributes as some inductive methods [154, 46], but we do not explore it in this work. Besides, we also investigate to use denoising autoencoder [140] as the structure preserving component.

*Note that we can use other ways to preprocess raw graph data to obtain the input feature X with lower dimension for large graphs. One simple way is to directly use existing scalable methods, e.g. DeepWalk and LINE, to obtain initial embeddings X as input.

Inductive DeepWalk (IDW)

The IDW model uses random walk to sample node sequences as that in DeepWalk. Starting from each node v_i, η sequences are randomly sampled with the length as l. In every step, a new node is randomly selected from the neighbors of the current node with the probability proportional to the corresponding weight in matrix A. To improve efficiency, the alias table method [67] is employed to sample node from the candidate node set in every sampling step. It only takes $O(1)$ time in a single sampling step. Then, positive node pairs can be constructed from node sequences. For every node sequence $\mathcal{W}$, we determine the positive target-context pairs as the set $\{(w_i, w_j) : |i-j| \leqslant s\}$, where w_i is the ith node in sequence $\mathcal{W}$ and s denotes the context size.

Similar to Skip-gram [82], a node v_i has two different representations, i.e., a target representation $\boldsymbol{u_i}$ and a context representation $\boldsymbol{u'_i}$, which are generated by the target generator $G(\cdot;\boldsymbol{\theta_1})$ and context generator $F(\cdot;\boldsymbol{\theta'_1})$, respectively. The generators are parameterized functions which are implemented with multi-layer perceptron in this work. Given row vector $\boldsymbol{x_i^T}$ in X corresponding to node v_i, we have $\boldsymbol{u_i} = G(\boldsymbol{x_i};\boldsymbol{\theta_1})$ and $\boldsymbol{u'_i} = F(\boldsymbol{x_i};\boldsymbol{\theta'_1})$. To capture network structural properties, we define the following objective function for each positive target-context pair (v_i, v_j) with negative sampling approach:

$$\begin{aligned}&\mathcal{O}_{IDW}(\boldsymbol{\theta_1};\boldsymbol{\theta'_1}) = \log\sigma(F(\boldsymbol{x_j};\boldsymbol{\theta'_1})^T G(\boldsymbol{x_i};\boldsymbol{\theta_1}))\\&+\sum_{n=1}^{K}\mathbb{E}_{v_n\sim P_n(v)}[\log\sigma(-F(\boldsymbol{x_n};\boldsymbol{\theta'_1})^T G(\boldsymbol{x_i};\boldsymbol{\theta_1}))],\end{aligned} \tag{3.2}$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the sigmoid function, K is the number of negative samples for each positive pair, $P_n(v)$ is the noise distribution for sampling negative context nodes, $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta'_1}$ are parameters to be learned. As suggested in [82], $P_n(v) = d_v^{3/4}/\sum_{v_i\in V} d_{v_i}^{3/4}$ can achieve quite good performance in practice, where d_v is the degree of node v.

3.2.5 Adversarial Learning

The adversarial learning component is employed to regularize the representations. It consists of a generator $G(\cdot;\boldsymbol{\theta_1})$ and a discriminator $D(\cdot;\boldsymbol{\theta_2})$. Specifically, $G(\cdot;\boldsymbol{\theta_1})$ represents a non-linear transformation of input high-dimensional features to embedding vectors. $D(\cdot;\boldsymbol{\theta_2})$ represents the probability of a sample coming from real data. The generator function is shared with the structure preserving component. Different from GANs [39], in our framework, a prior distribution $p(\boldsymbol{z})$ is selected as the data distribution for generating real data, while the embedding vectors are regarded as fake samples. In the training process, the discriminator is trained to tell apart the prior samples from the embedding vectors, while the generator is aimed to fit embedding vectors to the prior distribution. This process can be considered as a two-player minimax game with the generator and discriminator playing against each other. The utility function of the discriminator is:

$$\mathcal{O}_D(\boldsymbol{\theta_2}) = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log D(\boldsymbol{z};\boldsymbol{\theta_2})] + \mathbb{E}_{\boldsymbol{x}}[\log(1 - D(G(\boldsymbol{x};\boldsymbol{\theta_1});\boldsymbol{\theta_2}))]. \tag{3.3}$$

In order to camouflage its output as prior samples, the generator is trained to improve the following payoff:

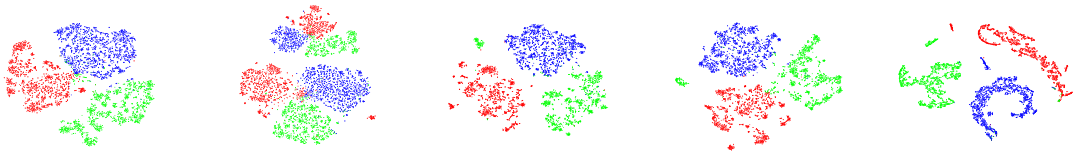
$$\mathcal{O}_G(\boldsymbol{\theta_1}) = \mathbb{E}_{\boldsymbol{x}}[\log(D(G(\boldsymbol{x};\boldsymbol{\theta_1});\boldsymbol{\theta_2}))]. \tag{3.4}$$

We argue that the adversarial learning component can help improve the learned representations in terms of robustness and structural meanings. We instantiate our framework with two structure preserving models, i.e., inductive DeepWalk (IDW) and denoising autoencoder (DAE). We call the ANE framework with IDW as Adversarial Inductive DeepWalk (AIDW) for easy illustration. Actually, with DAE as the structure preserving component, the ANE framework will become an adversarial autoencoder [80], and we represent it as ADAE to highlight the importance of the denoising criterion in learning representations.

During adversarial learning, it is also important to choose a proper prior distribution. Like many practices in GANs research [99, 80, 29], the prior distribution is usually defined as Uniform or Gaussian noise which enables GANs to learn meaningful and robust representations against uncertainty. In our experiments, we also considered the ANE framework with both kinds of prior distributions but find no significant difference. One possible reason is both kinds of uncertainty can help the ANE framework to achieve a certain level of robustness against noise. It is likely that a careful choice of prior distribution, possibly guided by prior domain knowledge, may further improve application-specific performance.

3.2.6 Algorithm

To implement the ANE approach, we consider a joint training procedure with two phases, including a structure preserving phase and an adversarial learning phase. In the structure preserving phase, we optimize objective function (3.2) for AIDW. In the adversarial learning phase, a prior distribution is imposed on representations through a minimax optimization problem. Firstly, the discriminator is trained to distinguish between prior samples and embedding vectors by optmizing the objective in Eq. (3.3). Then, the parameters of the generator are updated to fit the embedding vectors to prior space to fool the discriminator by optimizing the objective in Eq. (3.4). Besides, the Wassterstein-1 distance can be employed as the loss for the discriminator to help improve the stability of learning and avoid the mode collapse problem in traditional GANs training [4].



(a) DeepWalk. (b) LINE. (c) node2vec. (d) IDW. (e) AIDW.

Figure 3.3: Visualization of Cit-DBLP dataset. Each point represents one paper. Different colors correspond to different publication divisions. Red: "Information Science", blue: "ACM Transactions on Graphics", green: "Human-Computer Interaction".

Table 3.1: Statistics of datasets

Dataset	$\mid V \mid$	$\mid E \mid$	$\mid \mathcal{Y} \mid$
Cora	2,708	5,278	7
Citeseer	3,264	4,551	6
Wiki	2,363	11,596	17
Cit-DBLP	5,318	28,085	3

3.3 Experiments

3.3.1 Experiment Setup

Datasets

We conduct experiments on four real-world datasets with the statistics presented in Table 3.1, where $\mathcal{Y}$ denotes the label set. Cora and Citeseer are paper citation networks constructed by [81] with labels representing the research topics of papers. Wiki [110] is a network with nodes as web pages and edges as the hyperlinks between web pages, where the label represents the topic of the webpage. We regard these three networks as undirected networks, and do some preprocessing on the original datasets by deleting self-loops and nodes with zero degree. Cit-DBLP is a paper citation network extracted from DBLP dataset [128].

Baselines

We compare our model with several baseline methods, including DeepWalk, LINE, GraRep and node2vec. There are many other network embedding methods, but we do not consider them here, because their performances are inferior to these baseline models as shown in corresponding papers. The descriptions of the baselines are as follows.

- **DeepWalk** [95]: DeepWalk first transforms the network into node sequences by truncated random walk, and then uses it as input to the Skip-gram model to learn representations.
- **LINE** [126]: LINE can preserve both first-order and second-order proximities for undirected graph through modeling node co-occurrence probability and node conditional probability.
- **GraRep** [12]: GraRep preserves node proximities by constructing different k-step probability transition matrix.
- **node2vec** [41]: node2vec develops a biased random walk procedure to explore neighborhood of a node, which can strike a balance between local properties and global properties of a network.

Besides, we consider inductive DeepWalk and denoising autoencoder as another two baseline methods. Note that both of them employ shifted PPMI matrix as preprocessing for fair comparison.

Parameter Settings

For LINE, we follow the settings of parameters in [126]. The embedding vectors are normalized by L2-norm. Besides, we specially preprocess the original sparse networks

by adding two-hop neighbors to low degree nodes. For both DeepWalk and node2vec, the window size s, the walk length l and the number of walks η per node are set to 10, 80 and 10, respectively, for fair comparison. For GraRep, the maximum matrix transition step is set to 4, and the settings of other parameters follow those in [12]. Note that the dimension of representations for all methods are set to 128 for fair comparison.

For our methods, we only use the most simple structure for the generator. Specifically, the generator is a single-layer network with leaky ReLU activations (with a leak of 0.2) and batch normalization [55] on the output. The shifted PPMI matrix X is obtained by setting t as 4 for Cora and Citeseer, and 3 for Wiki. For inductive DeepWalk, the number of negative samples K is set to 5, and other parameters are set the same as DeepWalk. For denoising autoencoder, it has only one hidden layer with dimension as 128. For the discriminator of the framework, it is a three-layer neural networks, with the layer structure as 512-512-1. For the first two layers, we use leaky ReLU activations (with leak of 0.2) and batch normalization. For the output layer, we use sigmoid activation. For AIDW and ADAE, the settings of the structure preserving component are the same as those of IDW and DAE, respectively. The prior distribution of the adversarial learning component is set to $z_i \sim U[-1, 1]$. We use RMSProp optimizer with learning rate as 0.001.

3.3.2 Network Visualization

Network visualization is an indispensable way to analyze high-dimensional graph data, which can help reveal intrinsic structure of the data intuitively [125]. In this section, we visualize the representations of nodes generated by several different models using *t-SNE* [137]. We construct a paper citation network, namely Cit-DBLP, from DBLP with papers from three different publication divisions, including Information Sciences, ACM Transactions on Graphics and Human-Computer Interaction.

Table 3.2: Accuracy (%) of multi-class classification on Cora

%Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	71.43	73.83	75.61	76.92	77.79	77.78	78.47	79.17	79.04
LINE	71.26	74.50	76.04	76.81	77.68	77.99	78.46	79.00	79.11
GraRep	74.78	76.78	78.56	78.99	79.39	79.85	79.96	80.94	81.29
node2vec	75.06	78.49	80.06	80.94	81.52	82.07	82.39	83.28	83.17
DAE	75.21	78.07	79.39	80.51	80.91	81.41	82.36	83.23	83.32
ADAE	75.01	77.45	79.65	80.96	81.64	82.11	82.62	83.52	84.10
IDW	66.32	72.21	75.23	76.65	77.66	78.32	79.20	79.93	80.63
AIDW	**76.93**	**79.50**	**81.31**	**82.01**	**82.28**	**83.03**	**83.23**	**84.46**	**84.21**

Some statistics of this dataset have been presented in Table 6.2. These papers are naturally classified into three categories based on the research fields they belong to.

Figure 5.4 shows the visualization of embedding vectors obtained from different models using *t-SNE* tookit under the same parameter configuration. For both DeepWalk and LINE, papers from different categories are mixed with each other in the center of the figure. For LINE, there are 6 clusters with each category corresponding to two separate clusters, which is in conflict with the true structure of the network. Besides, the boundaries between different clusters are not clear. The visualizations of node2vec and IDW form three main clusters, which are better than those of DeepWalk and LINE. However, the boundary between blue cluster and green cluster for node2vec is not clear, while that of red cluster and green cluster is a little messy for IDW. AIDW performs better compared with baseline methods. We can observe that the visualization of AIDW has three clusters with quite large margin between each other. Furthermore, each cluster is linearly separable with another cluster in the figure, which can not be achieved by other baselines as showed by the figures. Intuitively, this experiment demonstrates that adversarial learning regularization can help learn more meaningful and robust representations.

Table 3.3: Accuracy (%) of multi-class classification on Citeseer

%Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	49.45	52.68	54.60	55.71	56.44	57.04	57.42	58.04	59.11
LINE	47.70	51.04	52.95	54.19	55.00	55.82	56.02	57.08	57.52
GraRep	51.62	53.29	53.59	53.83	54.55	54.62	54.97	54.90	56.27
node2vec	52.50	55.47	56.66	57.70	58.81	59.26	60.10	60.34	60.58
DAE	51.08	54.54	55.84	56.50	57.47	58.30	58.50	59.19	60.15
ADAE	52.40	55.58	56.64	57.32	58.34	59.60	60.27	60.63	61.31
IDW	45.45	50.47	52.32	53.38	54.75	55.18	55.98	56.23	57.19
AIDW	**53.25**	**56.76**	**57.95**	**59.06**	**59.45**	**59.95**	**60.28**	**60.87**	**62.26**

Table 3.4: Accuracy (%) of multi-class classification on Wiki

%Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	57.23	61.22	63.53	64.68	65.96	66.24	67.17	68.63	68.69
LINE	56.29	61.63	63.98	65.43	66.25	67.04	67.94	68.86	68.61
GraRep	**57.68**	61.14	62.73	63.89	64.86	65.55	66.01	67.55	68.02
node2vec	57.61	61.52	63.47	64.83	65.54	66.16	67.17	68.44	68.69
DAE	57.08	61.63	63.71	65.20	66.84	67.41	67.91	69.03	69.45
ADAE	57.24	61.67	63.85	65.34	66.67	67.11	67.79	69.68	70.59
IDW	56.01	60.77	63.08	64.37	65.66	66.47	67.15	67.86	68.52
AIDW	57.43	**62.14**	**64.18**	**65.53**	**67.07**	**68.00**	**69.44**	**71.63**	**72.03**

3.3.3 Node Classification

The label information can indicate interests, beliefs or other characteristics of nodes, which can help facilitate many applications, such as friend recommendation in online social networks and targeted advertising. However, in many real-world contexts, only a subset of nodes are labeled. Thus, node classification can be conducted to dig out information of unlabeled nodes. In this section, we conduct multi-class classification on three benchmark datasets, i.e., Cora, Citeseer and Wiki. We range the training ratio from 10% to 90% for comprehensive evaluation. All experiments are carried out with support vector classifier in Liblinear package[†][31]. Specifically, the learned node embeddings from network embedding methods are utilized as node features, and the classifier is trained with a portion labeled nodes and then used to predict

[†]https://www.csie.ntu.edu.tw/∼cjlin/liblinear/

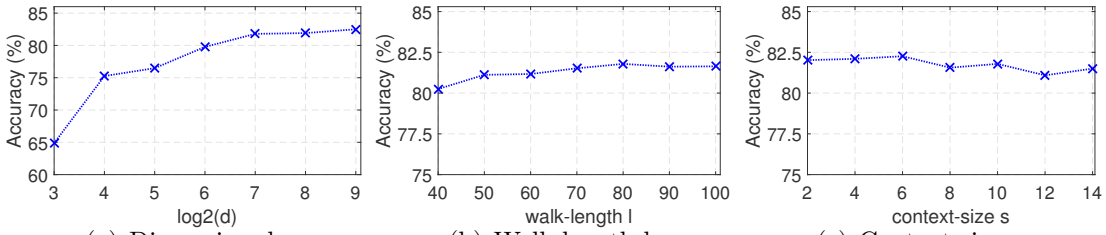
the labels of the testing nodes.

Results and discussion. To ensure the reliability, we obtain the experimental results by taking an average of that of 10 runs, which are shown in Tables 3.2, 3.3 and 3.4. We have the following observations:

- IDW produces similar results on Cora and Wiki with DeepWalk, and slightly inferior performance on Citeseer. The proposed model AIDW is built upon IDW with additional adversarial learning component. It consistently outperforms both IDW and DeepWalk on three datasets across all training ratios. For example, on Cora, AIDW gives more than 4% gain in accuracy over DeepWalk under all training ratio settings. It demonstrates that adversarial learning regularization can significantly improve the robustness and discrimination of the learned representations. The quantitative results also verify our previous qualitative findings in network visualization analysis.

- ADAE achieves about 1% gain in accuracy over DAE on Citeseer when varying the training ratio from 10% to 90%, slightly better results on Cora, and comparable performance on Wiki. It shows that ANE framework can also guide the learning of more robust embeddings when building upon DAE. However, we notice that ADAE does not achieve obvious improvements over the corresponding structure preserving model as AIDW does. One reason is that denoising criterion already contributes to learning stable and robust representations [140].

- Overall, the proposed method AIDW consistently outperforms all the baselines. As shown in Tables 3.2, 3.3 and 3.4, node2vec produces better results than DeepWalk, LINE and GraRep on average. Our method can further achieve improvements over node2vec. More specifically, AIDW achieves the best classification accuracy on all three benchmark datasets across different training ratios, with only one exception on Wiki with training ratio as 10%.



(a) Dimension d. (b) Walk-length l. (c) Context-size s.

Figure 3.4: Parameter sensitivity analysis of AIDW using multi-class classification on Cora with train ratio as 50%.

3.3.4 Model Sensitivity

In this section, we investigate the performance of AIDW w.r.t parameters and the type of prior on Cora dataset. Specifically, for parameter sensitivity analysis, we examine how the representation dimension d, walk-length l and context-size s affect the performance of node classification with the training ratio as 50%. Note that except for the parameter being tested, all other parameters are set to default values. We also compare the performance of AIDW with two different priors, i.e., a Gaussian distribution ($\mathcal{N}(0,1)$) and a Uniform distribution ($U[-1,1]$).

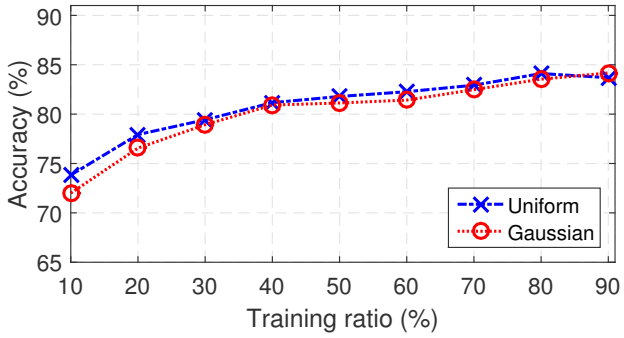


Figure 3.5: Multi-class classification on Cora with two different priors, i.e., Uniform and Gaussian, on AIDW model.

Figure 3.4(a) displays the results on the test of dimension d. When the dimension increases from 8 to 512, the accuracy shows apparent increase at first, and then tends to saturate once the dimension reaches around 128. Besides, the performance of AID-

W is not sensitive on walk-length and context-size. As shown in Figure 3.4(b), the accuracy slightly increases first, and then becomes stable when the walk-length varies from 40 to 100. With the increase of context-size, the performance keeps stable first, and then slightly degrades after the context-size is over 6, as shown in Figure 3.4(c). The degradation might be caused by the noisy neighborhood information brought in by the large context-size, since the average node degree of Cora is just about 1.95.

Figure 3.5 shows the results of multi-class classification on Cora with training ratio ranging from 10% to 90%. The accuracy curve of AIDW with uniform prior is almost coincided with that of AIDW with gaussian prior. It demonstrates that both types of prior can contribute to learning robust representations with no significant difference.

3.4 Related Work

3.4.1 Network Embedding

In recent years, many unsupervised network embedding methods have been proposed, which can be divided into three groups according to the techniques they use, i.e., probabilistic methods, matrix factorization based methods and autoencoder based methods. The probabilistic methods include DeepWalk [95], LINE [126], node2vec [41] and so on. DeepWalk firstly obtains node sequences from the original graph through random walk, and then learns the latent representations using Skip-gram model [82] by regarding node sequences as word sentences. LINE tries to preserve first-order and second-order proximities in two separate objective functions, and then directly concatenates the representations. In [41], the authors proposed to use biased random walk to determine neighboring structure, which can strike a balance between homophily and structural equivalence. It is actually a variant of DeepWalk.

Matrix factorization based methods first preprocess the adjacency matrix to capture different kinds of high-order proximities and then decompose the processed matrix to obtain graph embeddings. For example, GraRep [12] employs positive pointwise mutual information (PPMI) matrix as the preprocessing based on a proof of the equivalence between a k-step random walk in DeepWalk and a k-step probability transition matrix. HOPE [90] preprocesses the adjacency matrix of the directed graph with high-order proximity measurements, such as Katz Index [57], which can help capture asymmetric transitivity property. M-NMF [147] learns embeddings that can well capture community structure by building upon the modularity based community detection model [85].

Autoencoder is a widely used model for learning compact representations of high-dimensional data, which aims to preserve as much information in the latent space as possible for the reconstruction of the original data [52]. DNGR [13] firstly calculates the PPMI matrix, and then learns the representations through stacked denosing autoencoder. SDNE [143] is a variant of stacked autoencoder which adds a constraint in the loss function to force the connected nodes to have similar embedding vectors. In [61], the authors proposed a variational graph autoencoder (VGAE) by using a graph convolutional network [62] encoder for capturing network structural properties. Compared to variational autoencoder (VAE) [60], our ANE approach explicitly regularizes the posterior distribution of the latent space while VAE only assumes a prior distribution.

3.4.2 Generative Adversarial Networks

Generative adversarial networks (GANs) [39] are deep generative models, of which the framework consists of two components, i.e., a generator and a discriminator. GANs can be formulated as a minimax adversarial game, where the generator aims to map data samples from some prior distribution to data space, while the discriminator

tries to tell fake samples from real data. This framework is not directly suitable for unsupervised representation learning, due to the lack of explicit structure for inference.

There are three possible solutions for this problem as demonstrated by existing works. Firstly, some works managed to integrate some structures into the framework to do inference, i.e., projecting sample in data space back into the space of latent features, such as BiGAN [29], ALI [30] and EBGAN [164]. These methods can learn robust representations in many applications, such as image classification [29] and document retrieval [38]. The second approach is to generate representations from the hidden layer of the discriminator, like DCGANs [99]. By employing fractionally-strided convolutional layers, DCGANs can learn expressive image representations from both the generator and discriminator networks for supervised tasks. The third idea is to use adversarial learning process to regularize the representations. One successful practice is the Adversarial Autoencoders [80], which can learn powerful representations from unlabeled data without any supervision.

3.5 Conclusion

An adversarial network embedding framework has been proposed for learning robust graph representations. This framework consists of a structure preserving component and an adversarial learning component. For structure preserving, we proposed inductive DeepWalk to capture network structural properties. For adversarial learning, we formulated a minimax optimization problem to impose a prior distribution on representations to enhance the robustness. Empirical evaluations in network visualization and node classification confirmed the effectiveness of the proposed method.

Chapter 4

Adversarial Training Methods for Network Embedding

Network Embedding is the task of learning continuous node representations for networks, which has been shown effective in a variety of tasks such as link prediction and node classification. Most of existing works aim to preserve different network structures and properties in low-dimensional embedding vectors, while neglecting the existence of noisy information in many real-world networks and the overfitting issue in the embedding learning process. Most recently, generative adversarial networks (GANs) based regularization methods are exploited to regularize embedding learning process, which can encourage a global smoothness of embedding vectors. These methods have very complicated architecture and suffer from the well-recognized non-convergence problem of GANs. In this chapter, we aim to introduce a more succinct and effective local regularization method, namely adversarial training, to network embedding so as to achieve model robustness and better generalization performance. Firstly, the adversarial training method is applied by defining adversarial perturbations in the embedding space with an adaptive L_2 norm constraint that depends on the connectivity pattern of node pairs. Though effective as a regularizer, it suffers from the interpretability issue which may hinder its application in certain real-world

scenarios. To improve this strategy, we further propose an interpretable adversarial training method by enforcing the reconstruction of the adversarial examples in the discrete graph domain. These two regularization methods can be applied to many existing embedding models, and we take DeepWalk as the base model for illustration in this work. Empirical evaluations in both link prediction and node classification demonstrate the effectiveness of the proposed methods.

4.1 Introduction

Network embedding strategies, as an effective way for extracting features from graph structured data automatically, have gained increasing attention in both academia and industry in recent years. The learned node representations from embedding methods can be utilized to facilitate a wide range of downstream learning tasks, including some traditional network analysis tasks such as link prediction and node classification, and many important applications in industry such as product recommendation in e-commerce website and advertisement distribution in social networks. Therefore, under such great application interest, substantial efforts have been devoted to designing effective and scalable network embedding models.

Most of the existing works focus on preserving network structures and properties in low-dimensional embedding vectors [95, 12, 143]. Firstly, DeepWalk [95] defines random walk based neighborhood for capturing node dependencies, and node2vec [41] extends it with more flexibility in balancing local and global structural properties. LINE [126] preserves both first-order and second-order proximities through considering existing connection information. Further, GraRep [12] manages to learn different high-order proximities based on different k-step transition probability matrix. Aside from the above mentioned structure-preserving methods, several research works investigate the learning of property-aware network embeddings. For example, network

transitivity, as the driving force of link formation, is considered in [90], and node popularity, as another important factor affecting link generation, is incorporated into RaRE [42] to learn social-rank aware and proximity-preserving embedding vectors. However, the existence of nosiy information in real-world networks and the overfitting issue in the embedding learning process are neglected in most of these methods, which leaves the necessity and potential improvement space for further exploration.

Most recently, adversarial learning regularization method is exploited for improving model robustness and generalization performance in network embedding [23, 158]. ANE [23] is the first try in this direction, which imposes a prior distribution on embedding vectors through adversarial learning. Then, the adversarially regularized autoencoder is adopted in NetRA [158] to overcome the mode-collapse problem in ANE method. These two methods both encourage the global smoothness of the embedding distribution based on generative adversarial networks (GANs) [39]. Thus, they have very complicated frameworks and suffer from the well-recognized hard training problems of GANs [104, 4].

In this work, we aim to leverage the adversarial training (AdvT) method [123, 40] for network embedding to achieve model robustness and better generalization ability. AdvT is a local smoothness regularization method with more succinct architecture. Specifically, it forces the learned classifier to be robust to adversarial examples generated from clean ones with small crafted perturbation [123]. Such designed noise with respect to each input example is dynamically obtained through finding the direction to maximize model loss based on current model parameters, and can be approximately computed with fast gradient method [40]. It has been demonstrated to be extremely useful for some classification problems [40, 83].

However, how to adapt AdvT for graph representation learning remains an open problem. It is not clear how to generate adversarial examples in the discrete graph domain since the original method is designed for continuous inputs. In this chapter,

we propose an adversarial training DeepWalk model, which defines the adversarial examples in the embedding space instead of the original discrete relations and obtains adversarial perturbation with fast gradient method. We also leverage the dependencies among nodes based on connectivity patterns in the graph to design perturbations with different L_2 norm constraints, which enables more reasonable adversarial regularization. The training process can be formulated as a two-player game, where the adversarial perturbations are generated to maximize the model loss while the embedding vectors are optimized against such designed noises with stochastic gradient descent method. Although effective as a regularization technique, directly generating adversarial perturbation in embedding space with fast gradient method suffers from interpretability issue, which may restrict its application areas. Further, we manage to restore the interpretability of adversarial examples by constraining the perturbation directions to embedding vectors of other nodes, such that the adversarial examples can be considered as the substitution of nodes in the original discrete graph domain.

Empirical evaluations show the effectiveness of both adversarial and interpretable adversarial training regularization methods by building network embedding method upon DeepWalk. It is worth mentioning that the proposed regularization methods, as a principle, can also be applied to other embedding models with embedding vectors as model parameters such as node2vec and LINE. The main contributions of this chapter can be summarized as follows:

- We introduce a novel, succinct and effective regularization technique, namely adversarial training method, for network embedding models which can improve both model robustness and generalization ability.

- We leverage the dependencies among node pairs based on network topology to design perturbations with different L_2 norm constraints for different positive target-context pairs, which enables more flexible and effective adversarial training regu-

larization.

- We also equip the adversarial training method with interpretability for discrete graph data by restricting the perturbation directions to embedding vectors of other nodes, while maintaining its usefulness in link prediction and only slightly sacrificing its regularization ability in node classification.

- We conduct extensive experiments to evaluate the effectiveness of the proposed methods.

The organization of this chapter is as follows. In Section 4.2, the background and motivation are introduced. In Section 4.3, a detailed explanation of the proposed methods are described. In Section 4.4, the experimental results and analysis are presented. Then, the related work is reviewed in Section 4.5 and finally a short summary with the contributions and possible future work are provided in Section 4.6.

4.2 Background

4.2.1 Framework of Network Embedding

The purpose of network embedding is to transform discrete network structure information into compact embedding vectors, which can be further used to facilitate downstream learning tasks, such as node classification and link prediction. The research problem can be formally formulated as follows: Given a weighted (unweighted) directed (undirected) graph $\mathcal{G}(V, E, A)$ $(N = |V|)$, with $V = \{v_i\}_{i=1}^{N}$ as the node set, $E = \{e_{ij}\}_{i,j=1}^{N}$ as the edge set, and A as the weighted adjacency matrix with A_{ij} quantifying the strength of the relationship between node v_i and v_j, network embedding is aimed at learning a mapping function $f : V \mapsto U$, where $U \in R^{N \times d}$ $(d \ll N)$ is the embedding matrix with the ith row $\boldsymbol{u}_i{}^T$ as the embedding vector of node v_i. Note that for many network embedding models, a context embedding matrix U' will also

be learned. For these methods, embedding matrix U is also called target embedding matrix.

The learning framework of many famous network embedding methods, such as DeepWalk [95], LINE [126] and node2vec [41], can be summarized into two phases: a sampling phase that determines node pairs with strong relationships, and an optimization phase that tries to preserve pairwise relationships in the embedding vectors through the negative sampling approach [82]. In particular, in the first phase, these three methods capture structural information by defining different neighborhood structures, such as random walk explored neighborhood in [95, 41], first-order and second-order proximities in [126]. We denote the generalized neighbors (not restricted to directly connected nodes) of node v_i as $\mathcal{N}(v_i)$, i.e., nodes in this set are closely related with v_i and should be close with v_i in the embedding space. The loss function of this framework can be abstracted as follows:

$$\begin{aligned}\mathcal{L}(\mathcal{G}|\Theta) = -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}(v_i)} \{\log \sigma(s(v_i, v_j|\Theta)) \\ + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_k(v)}[\log \sigma(-s(v_i, v_k|\Theta))]\},\end{aligned} \tag{4.1}$$

where Θ represents model parameters such as target and context embedding matrices, $s(v_i, v_j|\Theta)$ represents the similarity score of node v_i and v_j based on model parameters Θ, and $\sigma(\cdot)$ is the sigmoid function. $P_k(v)$ denotes the distribution for sampling negative nodes, and a simple variant of unigram distribution is usually utilized, i.e., $P_k(v) \propto d_v^{3/4}$, where d_v is the out-degree of node v. Eq. (4.1) is actually a cross entropy loss with closely related node pair (v_i, v_j) as positive samples and (v_i, v_k) as negative samples, and thus network embedding can be considered as a classification problem.

4.2.2 Adversarial Training

Adversarial training [123, 40] is a newly proposed effective regularization method for classifiers which can not only improve the robustness of the model against adversarial attacks, but also achieve better generalization performance on learning tasks. It augments the original clean data with the dynamically generated adversarial examples, and then trains the model with the newly mixed examples. Denote the input as $\boldsymbol{x}$ and model parameters as $\boldsymbol{\theta}$. The loss on adversarial examples can be considered as a regularization term in the trained classifier $p(y|\cdot)$, which is as follows:

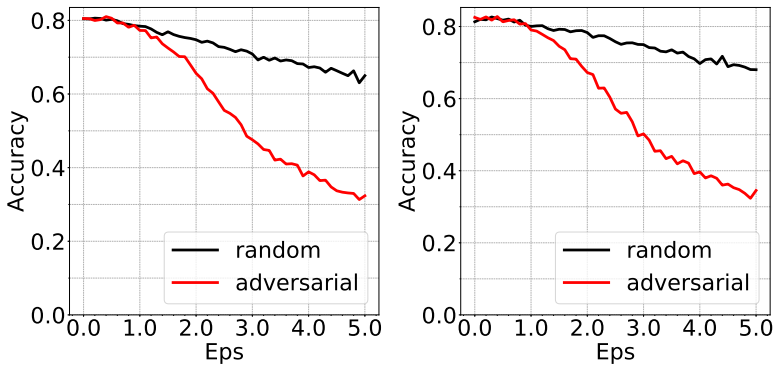
$$-\log p(y|\boldsymbol{x}+\boldsymbol{n}_{adv};\boldsymbol{\theta}), \tag{4.2}$$

$$where\ \boldsymbol{n}_{adv} = \underset{\boldsymbol{n},\,\|\boldsymbol{n}\|\leqslant\epsilon}{\arg\min} \log p(y|\boldsymbol{x}+\boldsymbol{n};\hat{\boldsymbol{\theta}}), \tag{4.3}$$

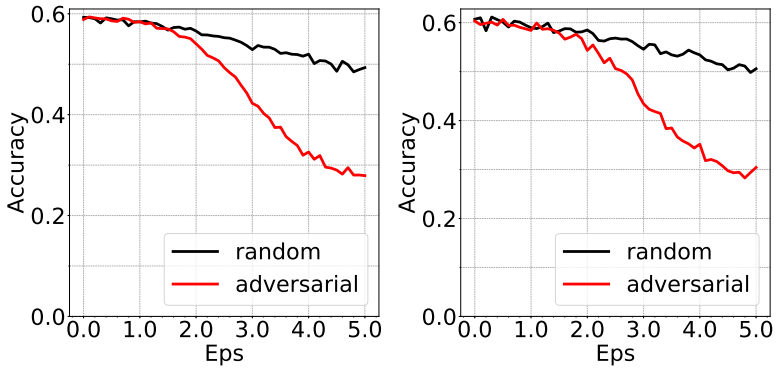
where $\boldsymbol{n}$ is the perturbation on the input, ϵ represents the norm constraint of $\boldsymbol{n}$, and $\hat{\boldsymbol{\theta}}$ are current model parameters but fixed as constants. We employ L_2 norm, while L_1 norm has also been used in the literature [40]. Eq. (4.2) means that the model should be robust on the adversarial perturbed examples. Before each batch training, the adversarial noise $\boldsymbol{n}$ with respect to the input $\boldsymbol{x}$ is firstly generated by solving optimization problem (4.3) to make it resistant to current model. Since it is difficult to calculate Eq.(4.3) exactly in general, fast gradient descent method [40] is widely used to obtain the adversarial noise approximately by linearizing $\log p(y|\boldsymbol{x};\boldsymbol{\theta})$ around $\boldsymbol{x}$. Specifically, the adversarial perturbation with L_2 norm constraint can be obtained as follows:

$$\boldsymbol{n}_{adv} = -\epsilon\cdot\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}\ where\ \boldsymbol{g} = \nabla_{\boldsymbol{x}}\log p(y|\boldsymbol{x};\hat{\boldsymbol{\theta}}). \tag{4.4}$$

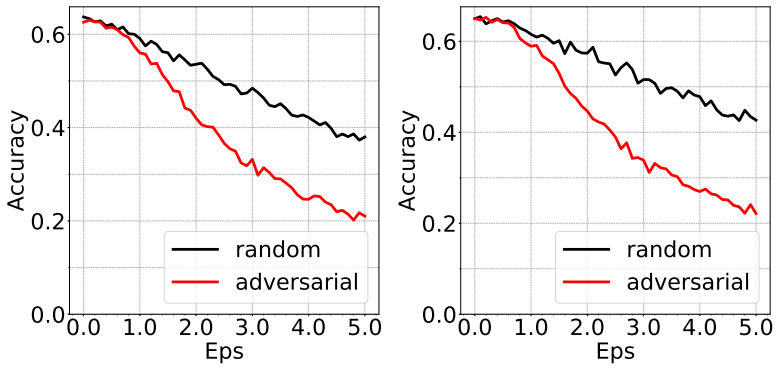
It can be easily calculated with backpropagation method.



(a) Cora, training ratio=50%, 80%.



(b) Citeseer, training ratio=50%, 80%.



(c) Wiki, training ratio=50%, 80%.

Figure 4.1: Impact of applying adversarial and random perturbations to the embedding vectors learned by DeepWalk on Cora, Citeseer and Wiki on multi-class classification with training ratio as 50% and 80%. Note that "random" represents random perturbations (noises generated from a normal distribution), while "adversarial" represents adversarial perturbations.

4.2.3 Motivation

To improve the generalization ability of network embedding models, two ways have been used: firstly, some denoising autoencoder based methods [13, 23] improve model robustness by adding random perturbation to input data or hidden layers of deep models; secondly, some existing methods [23, 158] regularize embedding vectors from a global perspective through GAN-based method, i.e., encouraging the global smoothness of the distribution of embeddings. In this chapter, we aim to introduce a novel, more succinct and effective regularization method for network embedding models, i.e., adversarial training (AdvT) [40]. AdvT generates crafted adversarial perturbations to model inputs and encourages local smoothness for improving model robustness and generalization performance, which can be expected to be more effective than the random perturbation methods [13] and global regularization methods [23, 158]. In the following, we would like to compare the impact of adversarial and random perturbation on embedding vectors to better motivate this new regularization method.

However, it is not clear how to integrate adversarial training into existing network embedding methods. Graph data is discrete, and the continuous adversarial noise can not be directly imposed on the discrete connected information. To bypass this difficulty, we seek to define the adversarial perturbation on embedding vectors instead of the discrete graph domain as inspired by [83]. We define the adversarial perturbation on node embeddings as follows:

$$\boldsymbol{n}_{adv} = \underset{\boldsymbol{n},\, \|\boldsymbol{n}\| \leqslant \epsilon}{\arg\max}\, \mathcal{L}(\mathcal{G}|\hat{\Theta} + \boldsymbol{n}), \tag{4.5}$$

which can be further approximated with fast gradient method as presented in Eq. (4.4).

Take DeepWalk [95] with negative sampling loss as an illustrative example. We

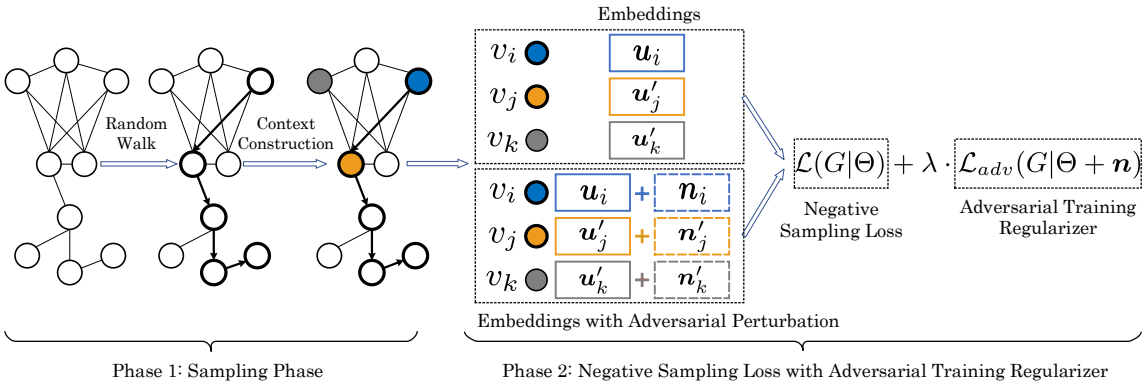


Figure 4.2: DeepWalk with Adversarial Training Regularization

explore the effect of adversarial perturbations on embedding vectors by adding them to the learned embedding vectors from DeepWalk, and then perform multi-class classification with the perturbed embeddings on several datasets. Besides, we choose random perturbations as the compared baseline, i.e., noises generated from a normal distribution. Figure 4.1 displays node classification results with varying L_2 norm constraints on the perturbations. We can find that embedding vectors are much more vulnerable to adversarial perturbations than random ones. For example, when ε is set to 2.0, the performance of node classification with training ratio as 80% on Cora drops 3.35% under random perturbation, while that decreases 16.25% under adversarial perturbation which is around 4 times more serious. If the embedding vectors can be trained to be more robust on adversarial noises, we can expect more significant improvements in generalization performance.

4.3 Proposed Methods

In this section, we first describe the adapted adversarial training method for network embedding models, and present the algorithm based on DeepWalk. Then, we will tackle its interpretability issue by designing a new adversarial perturbation generation method.

4.3.1 Adversarial Training DeepWalk

Figure 4.2 shows the framework of DeepWalk with adversarial training regularization. It consists of two phases: a sampling phase that determines node pairs with strong relationships, and an optimization phase that tries to preserve pairwise relationships in the embedding vectors based on negative sampling approach. Note that in this work we take DeepWalk as the illustrative example, and the proposed framework can be applied to the network embedding methods, such as LINE and node2vec, with the main difference in the sampling phase only.

In the first phase, DeepWalk transforms the network into node sequences by truncated random walk. For each node $v_i \in V$, η sequences each with l nodes will be randomly sampled based on network structure with v_i as the starting point. In every walking step, the next node v_j will be sampled from the neighbors of current node v_k with the probability proportional to the edge strength A_{kj} between v_k and v_j. In practice, the alias table method [67] is usually leveraged for node sampling given the weight distribution of neighbors of current node, which only takes $O(1)$ time in a single sampling step. Then in the context construction process, closely related node pairs will be determined based on the sampled node sequences. Denote a node sequence as S with the ith node as s_i. The positive target-context pairs from S is defined as $\mathcal{P} = \{(s_i, s_j) : |i - j| \leqslant c\}$, where c represents the window size. With the constructed node pairs, the negative sampling loss will be optimized, which is defined as follows:

$$\begin{aligned}\mathcal{L}(\mathcal{G}|\Theta) = -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}(v_i)} \{\log \sigma({\boldsymbol{u}'_j}^T \cdot \boldsymbol{u}_i) \\ + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_k(v)} [\log \sigma(-{\boldsymbol{u}'_k}^T \cdot \boldsymbol{u}_i)]\},\end{aligned} \tag{4.6}$$

where (v_i, v_j) is from the constructed positive target-context pairs, and $\boldsymbol{u}_i$ and $\boldsymbol{u}'_j$ are the target embedding of node v_i and context embedding of node v_j respectively.

For the adversarial version of DeepWalk, an adversarial training regularization term is added to the original loss to help learn robust node representations against adversarial perturbations. The regularization term shares the same set of model parameters with the original model, but with the perturbed target and context embeddings as input. Existing methods consider the input examples independently, and impose the unique L_2 norm constraint on all adversarial perturbations [40, 83]. For graph structured data, entities often correlate with each other in a very complicated way, so it is inappropriate to treat all positive target-context relations equally without discrimination. Adversarial regularization helps alleviate overfitting issue, but it may also bring in some noises that can hinder the preservation of structural proximities, i.e., adding noises to those inherently closely-related node pairs will prevent them from having similar embeddings. Thus, we take advantages of the denpendencies among nodes to assign different L_2 norm constraints to different positive target-context relations adaptively. Specifically, the more closely two nodes are connected, the smaller the constraint should be. The intuition is that less noises should be added to those node pairs which are inherently strongly-connected in the original network, thus they can be pushed closer in the embedding space with high flexibility, while for those weakly-connected pairs larger constraint can help alleviate the overfitting issue.

We obtain the similarity score of two nodes through computing the shifted positive pointwise mutual information matrix [66]:

$$M_{ij} = \max\{\log(\frac{\hat{M}_{ij}}{\sum_k \hat{M}_{kj}}) - \log(\beta), 0\}, \tag{4.7}$$

where $\hat{M} = \hat{A} + \hat{A}^2 + \cdots + \hat{A}^t$ captures different high-order proximities, $\hat{A}$ is the 1-step probability transition matrix obtained from A after the row-wise normalization, and β is a shift factor. We set t to 2 and β to $\frac{1}{N}$ in the experiments. Then, the

adaptive scale factor for the L_2 norm constraint of the target-context pair v_i and v_j is calculated as below:

$$\Phi_{ij} = 1 - M_{ij}/\max\{M\}, \tag{4.8}$$

where $\max\{M\}$ represents the maximum entry of matrix M. Since $M_{ij} > 0$ $(\forall i, j)$, $\Phi_{ij} \in [0, 1]$. For those strongly-connected target-context pairs, the adaptive scale factor can help scale down the L_2 norm constraint of the adversarial perturbation, and thus alleviate the negative effect from the noises.

Then, the adversarial training regularizer with scale factor for L_2 norm constraint is defined as follows:

$$\begin{aligned}
&\mathcal{L}_{adv}(\mathcal{G}|\Theta + \boldsymbol{n}_{adv}) \\
&= -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}(v_i)} \{\log \sigma((\boldsymbol{u}'_j + \Phi_{ij} \cdot (\boldsymbol{n}'_j)_{adv})^T \cdot (\boldsymbol{u}_i + \Phi_{ij} \cdot (\boldsymbol{n}_i)_{adv})) \\
&\quad + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_k(v)} [\log \sigma(-(\boldsymbol{u}'_k + (\boldsymbol{n}'_k)_{adv})^T \cdot (\boldsymbol{u}_i + (\boldsymbol{n}_i)_{adv}))]\},
\end{aligned} \tag{4.9}$$

where $(\boldsymbol{n}_i)_{adv}$ and $(\boldsymbol{n}'_j)_{adv}$ represent the original adversarial perturbation for target embedding of node v_i and context embedding of node v_j respectively.

Finally, one key problem is how to compute the adversarial perturbation for the given embedding vector of a node v. Here, we follow the famous adversarial training method directly [123, 40], and generate the perturbation noises to maximize model loss under current model parameters. The adversarial perturbation for node v is defined as follows:

$$\boldsymbol{n}_{adv} = \arg\max_{\boldsymbol{n}, \|\boldsymbol{n}\| \leqslant \epsilon} \mathcal{L}(\mathcal{G}|\hat{\Theta} + \boldsymbol{n}). \tag{4.10}$$

It can be further approximated with fast gradient method as follows:

$$\boldsymbol{n}_{adv} = \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2} \; where \; \boldsymbol{g} = \nabla_{\boldsymbol{u}} \mathcal{L}(\mathcal{G}|\hat{\Theta}). \tag{4.11}$$

Therefore, the overall loss for the proposed adversarial training DeepWalk is defined as follows:

$$\mathcal{L}(\mathcal{G}|\Theta) + \lambda \cdot \mathcal{L}_{adv}(\mathcal{G}|\Theta + \boldsymbol{n}_{adv}), \tag{4.12}$$

where λ is a hyper-parameter to control the importance of the regularization term.

In this work, we utilize DeepWalk with negative sampling loss as the base model for building the adversarial version of network embedding methods. Since the original implementation is based on the well encapsulated library, which lacks flexibility for further adaption, we re-implement the model with Tensorflow [1] and utilize a slightly different training strategy. Specifically, in each training epoch, we independently construct positive target-context pairs with random walk based method, and then optimize model parameters with mini-batch stochastic gradient descent technique. Algorithm 4.1 summarizes the training procedure for the adversarial training DeepWalk. The model parameters are firstly initialized by training DeepWalk with the method introduced above. For each batch training, adversarial perturbations are generated with fast gradient method for each node in the batch as presented in Line 7. Then, the target and context embeddings will be updated by optimizing the negative sampling loss with adversarial training regularization as shown in Line 9. Asynchronous version of stochastic gradient descent [88] can be utilized to accelerate the training as DeepWalk. Note that we ignore the derivative of $\boldsymbol{n}_{adv}$ with respect to model parameters. The adversarial perturbations can be computed with simple back-propagation method, which enjoys low computational cost. Thus, the adversarial training DeepWalk is scalable as the base model.

4.3.2 Interpretable Adversarial Training DeepWalk

Adversarial examples refer to examples that are generated by adding viciously designed perturbations with norm constraint to the clean input, which can significantly increase model loss and probably induce prediction error [123]. Take an example from [40] for illustration, a "panda" image with imperceptibly small adversarial perturbation is assigned to be "gibbon" by the well-trained classification model with high confidence, while the original image can be correctly classified. Such adversari-

Algorithm 4.1: The adversarial training DeepWalk

Input : graph $\mathcal{G}(V, E, A)$, window size c, embedding size d, walks per node η, negative size K, walk length l, adversarial noise level ϵ, adversarial regularization strength λ, batch size b

Output: Embedding matrix U

Initialize target and context embeddings with DeepWalk;

while *not converge* **do**

  Generate a set of positive target-context pairs $\mathcal{P}$ with random walk based method;

  repeat

    Sample a batch $\mathcal{B}$ of target-context pairs from $\mathcal{P}$;

    // Generate adversarial perturbations

    $\boldsymbol{n}_{adv} = \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$ where $\boldsymbol{g} = \nabla_{\boldsymbol{u}} \mathcal{L}(\mathcal{G}|\hat{\Theta})$ for each node v in the batch;

    // Optimize model parameters

    Update target and context embeddings by applying gradient descent technique to Eq. (4.12) ;

  until $[|\mathcal{P}|/b]_+$ *times*;

al examples can be well interpreted since the perturbations are imposed on the input space. For the adversarial training DeepWalk, adversarial perturbations are added to node embeddings instead of the discrete nodes and connected relations, and thus can not be easily reconstructed in the discrete graph domain. Though effective as a regularizer for improving model generalization performance, it suffers from lack of interpretability, which may create a barrier for its adoption in some real-world applications.

In this section, we propose an interpretable adversarial DeepWalk model by restoring the interpretability of adversarial perturbations. Instead of pursuing the worst perturbation direction only, we restrict the direction of perturbations toward a subset of nodes in the graph in the embedding space, such as the neighbors of the considered node. In this way, the adversarial perturbations in the node embedding space might be interpreted as the substitution of nodes in the original input space, i.e., the discrete target-context relations. However, there might be a certain level of sacrifice on the regularization performance because of the restriction on perturbation

directions.

The *direction vector* from node v_t to v_k in the embedding space is defined as follows:

$$\boldsymbol{v}_k^{(t)} = \frac{\tilde{\boldsymbol{v}}_k^{(t)}}{\|\tilde{\boldsymbol{v}}_k^{(t)}\|_2},\ where\ \tilde{\boldsymbol{v}}_k^{(t)} = \boldsymbol{u}_k - \boldsymbol{u}_t. \tag{4.13}$$

Denote $V^{(t)} \subseteq V$ ($|V^{(t)}| = T$, $|V^{(t)}| \ll |V|$) as a set of nodes for generating adversarial perturbation for node v_t. We define $V^{(t)}$ as the top T nearest neighbors of node v_t in the embedding space based on current model parameters. To improve model efficiency, we can also obtain $V^{(t)}$ based on the pretrained model parameters, and fix it for all training epochs. We use the latter strategy for experiments in this chapter. Denote $\boldsymbol{w}^{(t)} \in \mathcal{R}^T$ as the weight vector for node v_t with $w_k^{(t)}$ representing the weight associated with direction vector $\boldsymbol{v}_k^{(t)}$, where v_k is the kth node in $V^{(t)}$. The interpretable perturbation for v_t is defined as the weighted sum of the direction vectors starting from v_t and ending with nodes in $V^{(t)}$:

$$\boldsymbol{n}(\boldsymbol{w}^{(t)}) = \sum_{k=1}^{T} w_k^{(t)} \boldsymbol{v}_k^{(t)},\ v_k \in V^{(t)},\ \forall k = 1, \cdots, T. \tag{4.14}$$

The adversarial perturbation is obtained by finding the weights that can maximize the model loss:

$$\boldsymbol{w}_{iAdv}^{(t)} = \underset{\boldsymbol{w}^{(t)}, \|\boldsymbol{w}^{(t)}\| \leqslant \epsilon}{\arg\max}\ \mathcal{L}_{iAdv}(\mathcal{G}|\Theta + \boldsymbol{n}(\boldsymbol{w}^{(t)})), \tag{4.15}$$

where $\boldsymbol{\mathcal{L}}_{iAdv}$ is obtained by replacing $\boldsymbol{n}_{adv}$ in Eq. (4.9) with $\boldsymbol{n}(\boldsymbol{w}^{(t)})$. In consideration of model efficiency, the above regularization term is approximated with first-order Taylor series for easy computation as in [40]. Thus, the weights for constructing interpretable adversarial perturbation for node v_t can be computed as follows:

$$\boldsymbol{w}_{iAdv}^{(t)} = \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}\ where\ \boldsymbol{g} = \nabla_{\boldsymbol{w}^{(t)}} \mathcal{L}_{iAdv}(\mathcal{G}|\Theta + \boldsymbol{n}(\boldsymbol{w}^{(t)})). \tag{4.16}$$

Table 4.1: Statistics of benchmark datasets

Name	Cora	Citeseer	Wiki	CA-GrQc	CA-HepTh
$\|V\|$	2,708	3,264	2,363	5,242	9,877
$\|E\|$	5,278	4,551	11,596	14,484	25,973
Avg. degree	1.95	1.39	4.91	2.76	2.63
#Labels	7	6	17	-	-

Substituting $\boldsymbol{w}_{iAdv}^{(t)}$ into Eq. (4.14), we can get the adversarial perturbations $\boldsymbol{n}(\boldsymbol{w}_{iAdv}^{(t)})$. Further, by replacing $\boldsymbol{n}_{adv}$ with $\boldsymbol{n}(\boldsymbol{w}_{iAdv}^{(t)})$ in Eq. (4.9), we can have the interpretable adversarial training regularizer for DeepWalk. The algorithm for interpretable adversarial training DeepWalk is different from Algorithm 4.1 in the way of generating adversarial perturbations, and thus we do not present it to avoid repetition. Since $|V^{(t)}| \ll |V|$, the computation of adversarial perturbation for one node takes constant time. Therefore, the time complexity of this model is also linear to the number of nodes in the graph as DeepWalk.

4.4 Experiments

In this section, we empirically evaluate the proposed methods through performing link prediction and node classification on several benchmark datasets.

4.4.1 Experiment Setup

Datasets

We conduct experiments on several benchmark datasets from various real-world applications. Table 4.1 shows some statistics of them. Note that we do some preprocessing on the original datasets by deleting self-loops and nodes with zero degree. Some descriptions of these datasets are summarized as follows:

- **Cora, Citeseer** [81]: Paper citation networks. Cora consists of 2708 papers with 7 categories, and Citeseer consists 3264 papers including 6 categories.

- **Wiki** [110]: Wiki is a network with nodes as web pages and edges as the hyperlinks between web pages.

- **CA-GrQc, CA-HepTh** [65]: Author collaboration networks. They describe scientific collaborations between authors with papers submitted to General Relativity and Quantum Cosmology category, and High Energy Physics, respectively.

Baseline Models

The descriptions of the baseline models are as follows:

- **Graph Factorization (GF)** [3]: GF directly factorizes the adjacency matrix with stochastic gradient descent technique to obtain the embeddings, which enables it scale to large networks.

- **DeepWalk** [95]: DeepWalk regards node sequence obtained from truncated random walk as word sequence, and then uses skip-gram model to learn node representations. We directly use the publicly available source code with hierarchical softmax approximation for experiments.

- **LINE** [126]: LINE preserves network structural proximities through modeling node co-occurrence probability and node conditional probability, and leverages the negative sampling approach to alleviate the expensive computation.

- **node2vec** [41]: node2vec differs from DeepWalk by proposing more flexible method for sampling node sequences to strike a balance between local and global structural properties.

- **GraRep** [12]: GraRep applies SVD technique to different k-step probability transition matrix to learn node embeddings, and finally obtains global representations through concatenating all k-step representations.

- **AIDW** [23]: AIDW is an inductive version of DeepWalk with GAN-based regularization method. A prior distribution is imposed on node representations through adversarial learning to achieve a global smoothness in the distribution.

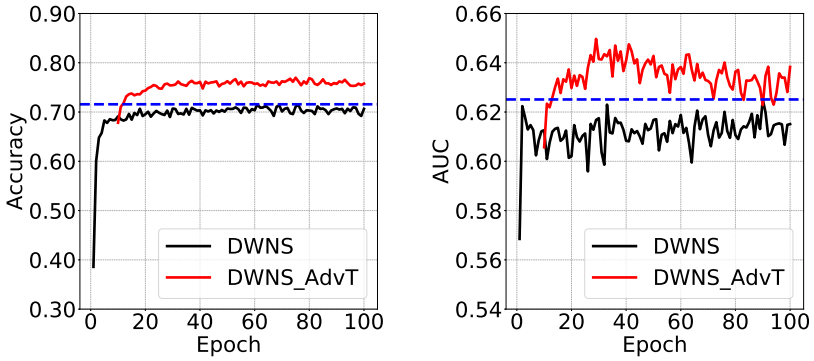
Our implemented version of DeepWalk is based on negative sampling approach, thus we denote it as DWNS to avoid confusion. We also include a baseline, namely DWNS_rand, with noises from a normal distribution as perturbations in the regularization term. Following existing work [107], we denote the adversarial training DeepWalk as DWNS_AdvT, and the interpretable adversarial training DeepWalk as DWNS_iAdvT in the rest of this chapter.

Parameter Settings

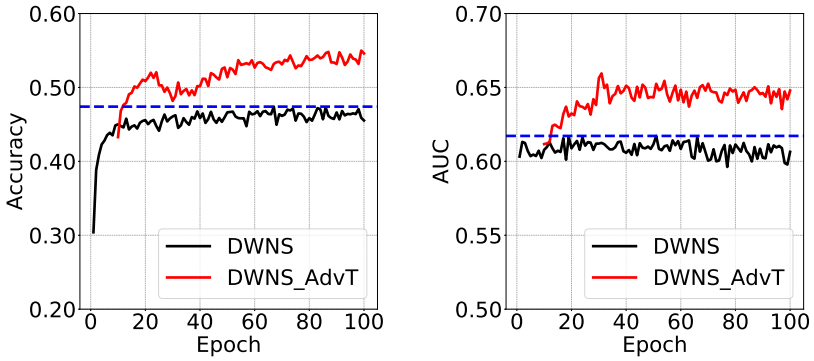
For DWNS and its variants including DWNS_rand, DWNS_AdvT and DWNS_iAdvT, the walk length, walks per node, window size, negative size, regularization strength, batch size and learning rate are set to 40, 1, 5, 5, 1, 1024 and 0.001, respectively. The adversarial noise level ϵ has different settings in DWNS_AdvT and DWNS_iAdvT, while DWNS_rand follows the settings of DWNS_AdvT. For DWNS_AdvT, ϵ is set to different value for different datasets. Specifically, ϵ is set to 0.9 for Cora, 1.1 for Citeseer in both link prediction and node classification, and 0.6 and 0.5 for Wiki in node classification and link prediction respectively, while it is set to 0.5 for all other datasets in these two learning tasks. For DWNS_iAdvT, ϵ is set to 5 for all datasets in both node classification and link prediction tasks, and the size of the nearest neighbor set T is set to 5. Besides, the dimension of embedding vectors are set to 128 for all methods.

4.4.2 Impact of Adversarial Training Regularization

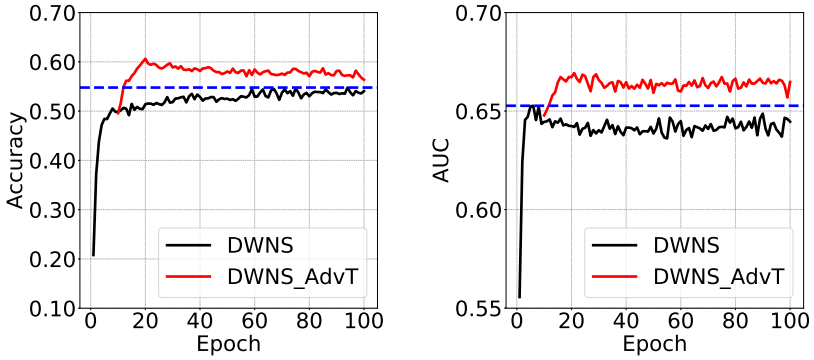
In this section, we conduct link prediction and multi-class classification on adversarial training DeepWalk, i.e., DWNS_AdvT, to study the impact of adversarial training



(a) Cora.

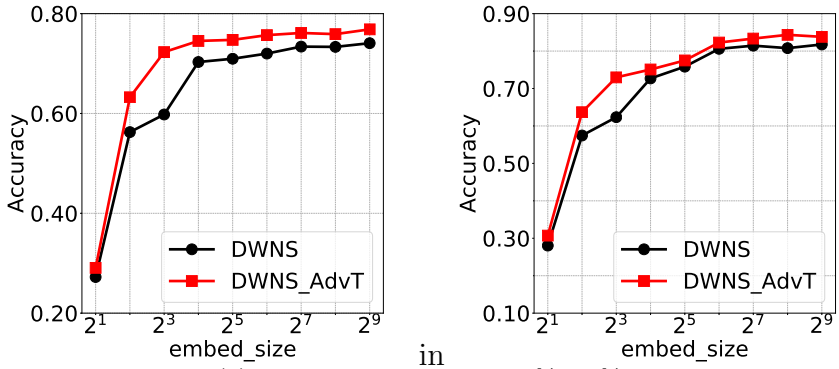


(b) Citeseer.



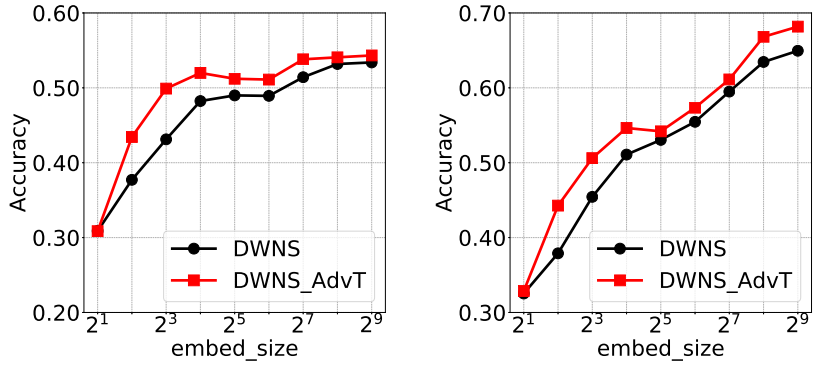
(c) Wiki.

Figure 4.3: Training curves of node classification (left, training ratio 10%) and link prediction (right).

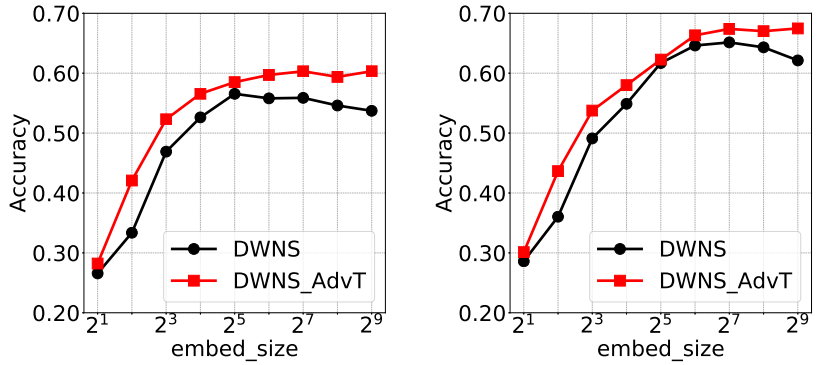


in

(a) Cora, training ratio=10%, 50%.



(b) Citeseer, training ratio=10%, 50%.



(c) Wiki, training ratio=10%, 50%.

Figure 4.4: Performance comparison between DWNS and DWNS_AdvT on multi-class classification with training ratio as 10% (left) and 50% (right) respectively under varying embedding size.

regularization on network representation learning from two aspects: model performance on different training epochs and model performance under different model size.

Node classification is conducted with support vector classifier in Liblinear package*[31] in default settings with the learned embedding vectors as node features. In link prediction, network embedding is firstly performed on a sub-network, which contains 80% of edges in the original network, to learn node representations. Note that the degree of each node is ensured to be greater than or equal to 1 during subsampling process to avoid meaningless embedding vectors. We use AUC score as the performance measure, and treat link prediction as a classification problem. Specifically, a L_2-SVM classifier is trained with edge feature inputs obtained from the Hadamard product of embedding vectors of two endpoints as many other works [41, 148], positive training samples as the observed 80% edges, and the same number of negative training samples randomly sampled from the network, i.e., node pairs without direct edge connection. The testing set consists of the hidden 20% edges and two times of randomly sampled negative edges. All experimental results are obtained by making an average of 10 different runs.

Training Process

We train DWNS model for 100 epochs, and evaluate the generalization performance of the learned embedding vectors in each epoch with node classification and link prediction on Cora, Citeseer and Wiki. We also conduct similar experiments on DWNS_AdvT for 90 epochs with the model parameters initialized from those of DWNS after $10th$ training epochs. Figures 4.3 shows the experimental results.

In general, adversarial training regularization can bring a significant improvement in generalization ability to DWNS through the observation of training curves in both

*https://www.csie.ntu.edu.tw/∼cjlin/liblinear/

node classification and link prediction. Specifically, after 10 training epochs, the evaluation performance has little improvements for all datasets in two learning tasks with further training for DWNS, while adversarial training regularization leads to an obvious performance increase. In Figure 4.3, the blue line is drew by setting its vertical coordinates as the maximum value of the metrics of DWNS in the corresponding experiments. We can find that the training curve of DWNS_AdvT is continuously above the blue line in different training epochs. Particularly, there is an impressive 7.2% and 9.2% relative performance improvement in link prediction for Cora and Citeseer respectively. We notice that the performance of DWNS_AdvT drops slightly after about 40 training epochs for Cora in link prediction, and about 20 training epochs for Wiki in node classification. The reason might be that some networks are more vulnerable to overfitting, and deeper understanding of this phenomenon needs further exploration.

Performance vs. Embedding Size

We explore the effect of adversarial regularization under different model size with multi-class classification. Figure 4.4 demonstrates the classification results on Cora, Citeseer and Wiki with training ratio as 10% and 50%. In general, adversarial training regularization is essential for improving model generalization ability. Across all tested embedding size, our proposed adversarial training DeepWalk can consistently outperform the base model. For two models, when varying embedding size from 2 to 512, the classification accuracy firstly increases in a relatively fast speed, then grows slowly, and finally becomes stable or even drops slightly. The reason is that model generalization ability is improved with the increase of model capacity firstly until some threshold, since more network structural information can be captured with larger model capacity. However, when the model capacity becomes too large, it can easily result in overfitting, and thus cause performance degradation. We notice that

Table 4.2: AUC score for link prediction

Dataset	Cora	Citeseer	Wiki	CA-GrQc	CA-HepTh
GF	0.550 ± 0.005	0.550 ± 0.002	0.584 ± 0.007	0.593 ± 0.003	0.554 ± 0.001
DeepWalk	0.620 ± 0.003	0.621 ± 0.002	0.658 ± 0.002	0.694 ± 0.001	0.683 ± 0.000
LINE	0.626 ± 0.011	0.625 ± 0.004	0.647 ± 0.010	0.641 ± 0.002	0.629 ± 0.005
node2vec	0.626 ± 0.023	0.627 ± 0.022	0.639 ± 0.010	0.695 ± 0.006	0.667 ± 0.009
GraRep	0.609 ± 0.035	0.589 ± 0.025	0.642 ± 0.045	0.500 ± 0.000	0.500 ± 0.000
AIDW	0.552 ± 0.034	0.606 ± 0.035	0.511 ± 0.019	0.615 ± 0.023	0.592 ± 0.019
DWNS	0.609 ± 0.018	0.609 ± 0.011	0.648 ± 0.007	0.690 ± 0.004	0.662 ± 0.006
DWNS_rand	0.606 ± 0.012	0.608 ± 0.005	0.645 ± 0.010	0.696 ± 0.006	0.662 ± 0.003
DWNS_AdvT	0.644 ± 0.009	**0.656 ± 0.007**	**0.665 ± 0.005**	**0.707 ± 0.004**	**0.692 ± 0.003**
DWNS_iAdvT	**0.655 ± 0.015**	0.653 ± 0.006	0.660 ± 0.002	**0.707 ± 0.004**	0.688 ± 0.004

the performance improvement of DWNS_AdvT over DWNS is quite small when the embedding size is 2. It is probably because model capacity is the main reason limiting model performance and model robustness is not a serious issue when embedding size is too small.

4.4.3 Link Prediction

Link prediction is essential for many applications such as extracting missing information and identifying spurious interaction [76]. In this section, we conduct link prediction on five real-world networks, and compare our proposed methods with the state-of-the-art methods. The experimental settings have been illustrated in Section 4.4.2. Table 4.2 summarizes the experimental results.

It can be easily observed that both our proposed methods, including DWNS_AdvT and DWNS_iAdvT, perform better than DWNS in all five datasets, which demonstrates that two types of adversarial regularization methods can help improve model generalization ability. Specifically, there is a 4.62% performance improvement for DWNS_AdvT over DWNS on average across all datasets, and that for DWNS_iAdvT is 4.60%, which are very impressive.

We noticed that AIDW has a poor performance in link prediction. The reasons can be two-folds: firstly, AIDW encourages the smoothness of embedding distri-

Table 4.3: Accuracy (%) of multi-class classification on Cora with training ratios ranging from 1% to 9%

%Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%
GF	24.55	28.87	32.07	33.11	34.45	35.83	38.25	39.05	39.84
DeepWalk	44.63	49.30	52.25	53.05	55.21	59.10	59.26	62.20	63.07
LINE	38.78	49.62	54.51	56.49	58.99	61.30	63.05	64.19	66.59
node2vec	58.02	63.98	66.33	68.07	69.91	69.87	71.41	72.60	73.63
GraRep	54.24	63.58	65.36	68.78	70.67	72.69	72.37	72.70	73.53
AIDW	54.55	63.30	65.86	66.20	67.62	68.61	69.52	71.07	71.44
DWNS	57.72	64.82	67.93	68.50	68.27	70.81	70.72	72.30	72.00
DWNS_rand	56.46	64.87	67.44	68.24	70.38	71.16	71.34	72.67	73.51
DWNS_AdvT	**62.66**	**68.46**	69.91	**73.62**	**74.71**	**75.55**	**76.18**	**76.77**	**77.72**
DWNS_iAdvT	58.67	66.65	**70.17**	70.52	71.42	72.47	74.26	75.32	74.52

bution from a global perspective by imposing a prior distribution on them, which can result in over-regularization and thus cause performance degradation; secondly, AIDW suffers from mode-collapse problem because of its generative adversarial network component, which can also result in model corruption. Besides, DWNS_rand has similar performance with DWNS, which means that the regularization term with random perturbation contributes little to model generalization ability. By comparison, our proposed novel adversarial training regularization method is more stable and effective.

It can be observed that the performance of DWNS_AdvT and DWNS_iAdvT are comparable. Either DWNS_AdvT or DWNS_iAdvT achieves the best results across the five datasets, which shows the remarkable usefulness of the proposed regularization methods. For Cora and CA-GrQc, DWNS_iAdvT has better performance, although we restrict the perturbation directions toward the nearest neighbors of the considered node. It suggests that such restriction of perturbation directions might provide useful information for representation learning.

Table 4.4: Accuracy (%) of multi-class classification on Cora with training ratios ranging from 10% to 90%

%Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
GF	39.42	46.14	48.57	50.09	50.85	51.88	52.89	52.34	51.51
DeepWalk	64.60	69.85	74.21	76.68	77.59	77.68	78.63	79.35	79.23
LINE	66.06	70.86	72.25	73.94	74.03	74.65	75.12	75.30	75.76
node2vec	73.96	78.04	80.07	81.62	82.16	82.25	82.85	84.02	84.91
GraRep	74.98	77.48	78.57	79.38	79.53	79.68	79.75	80.89	80.74
AIDW	73.83	77.93	79.43	81.16	81.79	82.27	82.93	84.11	83.69
DWNS	73.20	76.98	79.83	80.56	82.27	82.52	82.92	82.97	84.54
DWNS_rand	73.45	78.04	79.76	81.66	81.72	82.53	83.57	83.51	83.69
DWNS_AdvT	**77.73**	**80.50**	**82.33**	**83.54**	**83.63**	**84.41**	**84.99**	**85.66**	**85.65**
DWNS_iAdvT	76.12	78.88	80.31	81.61	82.80	83.03	83.63	83.75	85.02

Table 4.5: Accuracy (%) of multi-class classification on Citeseer with training ratios ranging from 1% to 9%

%Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%
GF	22.63	24.49	25.76	28.21	28.07	29.02	30.20	30.70	31.20
DeepWalk	27.82	32.44	35.47	36.85	39.10	41.01	41.56	42.81	45.35
LINE	29.98	34.91	37.02	40.51	41.63	42.48	43.65	44.25	45.65
node2vec	36.56	40.21	44.14	45.71	46.32	47.47	49.56	49.78	50.73
GraRep	37.98	40.72	43.33	45.56	47.48	47.93	49.54	49.87	50.65
AIDW	38.77	42.84	44.04	44.27	46.29	47.89	47.73	49.61	49.55
DWNS	38.13	42.88	46.60	46.14	46.38	48.18	48.58	48.35	50.16
DWNS_rand	39.29	43.42	42.73	46.00	46.13	48.69	48.15	49.92	50.08
DWNS_AdvT	**41.33**	45.00	**46.73**	**48.57**	**50.37**	**51.06**	**52.07**	**53.09**	**53.73**
DWNS_iAdvT	40.88	**45.53**	46.01	47.10	50.02	50.79	49.59	52.78	51.95

Table 4.6: Accuracy (%) of multi-class classification on Citeseer with training ratios ranging from 10% to 90%

%Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
GF	31.48	34.05	35.69	36.26	37.18	37.87	38.85	39.16	39.54
DeepWalk	45.53	50.98	53.79	55.25	56.05	56.84	57.36	58.15	59.11
LINE	47.03	50.09	52.71	53.52	54.20	55.42	55.87	55.93	57.22
node2vec	50.78	55.89	57.93	58.60	59.44	59.97	60.32	60.75	61.04
GraRep	50.60	53.56	54.63	55.44	55.20	55.07	56.04	55.48	56.39
AIDW	50.77	54.82	56.96	58.04	59.65	60.03	60.99	61.18	62.84
DWNS	50.00	53.74	57.37	58.59	59.00	59.53	59.62	59.51	60.18
DWNS_rand	50.84	55.26	58.51	59.59	59.12	60.22	60.62	61.59	60.55
DWNS_AdvT	**54.79**	**59.21**	**61.06**	**61.26**	**62.56**	**62.63**	**62.40**	**63.05**	**63.73**
DWNS_iAdvT	52.26	56.65	59.07	60.27	61.96	62.04	62.20	62.21	63.15

Table 4.7: Accuracy (%) of multi-class classification on Wiki with training ratios ranging from 1% to 9%

%Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%
GF	19.76	22.70	27.00	28.41	30.28	31.49	31.87	32.18	34.16
DeepWalk	28.65	32.84	36.66	37.98	40.73	42.94	45.57	45.47	46.06
LINE	32.46	40.84	44.56	49.59	51.11	52.37	54.32	55.72	56.51
node2vec	32.41	41.96	47.32	48.15	50.65	51.08	52.71	54.66	54.81
GraRep	33.38	45.61	49.10	50.92	53.01	54.43	54.84	57.50	57.01
AIDW	35.17	43.05	46.63	51.29	52.40	52.72	55.92	56.78	55.92
DWNS	35.76	42.71	48.08	50.01	50.21	52.26	53.26	53.80	55.27
DWNS_rand	36.12	44.57	46.71	49.15	51.74	53.37	53.22	53.27	54.21
DWNS_AdvT	**38.42**	**45.80**	**50.21**	51.12	**54.29**	**56.43**	**57.12**	**57.82**	**58.60**
DWNS_iAdvT	37.46	45.11	49.14	**51.57**	51.88	54.43	55.42	56.05	55.93

Table 4.8: Accuracy (%) of multi-class classification on Wiki with training ratios ranging from 10% to 90%

%Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
GF	34.25	36.13	37.66	37.43	39.48	40.17	39.83	40.25	41.01
DeepWalk	46.60	54.48	59.05	62.70	64.66	65.95	66.98	68.37	68.78
LINE	57.88	61.08	63.50	64.68	66.29	66.91	67.43	67.46	68.61
node2vec	55.94	59.67	61.11	64.21	65.08	65.58	66.76	67.19	68.73
GraRep	58.57	61.91	63.58	63.77	64.68	65.39	65.92	65.18	67.05
AIDW	57.32	61.84	63.54	64.90	65.58	66.54	65.59	66.58	68.02
DWNS	55.77	59.63	61.98	64.01	64.59	66.99	66.45	67.55	67.51
DWNS_rand	56.33	59.41	61.94	64.07	65.17	66.18	65.64	68.20	67.34
DWNS_AdvT	**59.97**	**63.33**	**65.32**	**66.53**	**67.06**	**67.69**	**68.94**	**68.35**	**69.32**
DWNS_iAdvT	57.81	61.40	63.37	65.71	65.56	67.09	66.81	67.70	68.02

4.4.4 Node Classification

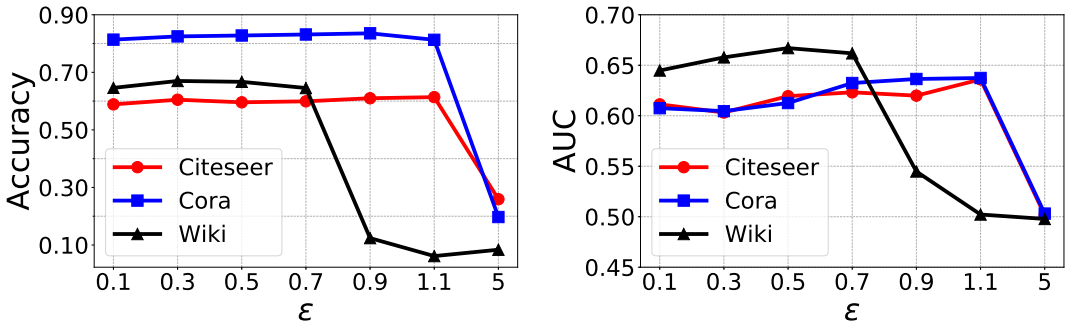
Node classification can be conducted to dig out missing information in a network. In this section, we conduct multi-class classification on three benchmark datasets, including Cora, Citeseer and Wiki, with the training ratio ranging from 1% to 90%. Tables 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8 summarize the experimental results.

Firstly, DWNS_rand and DWNS have similar performance in all three datasets. For example, the average improvement of DWNS_rand over DWNS is 0.16% across all training ratios in Wiki, which can be negligible. It validates that random per-

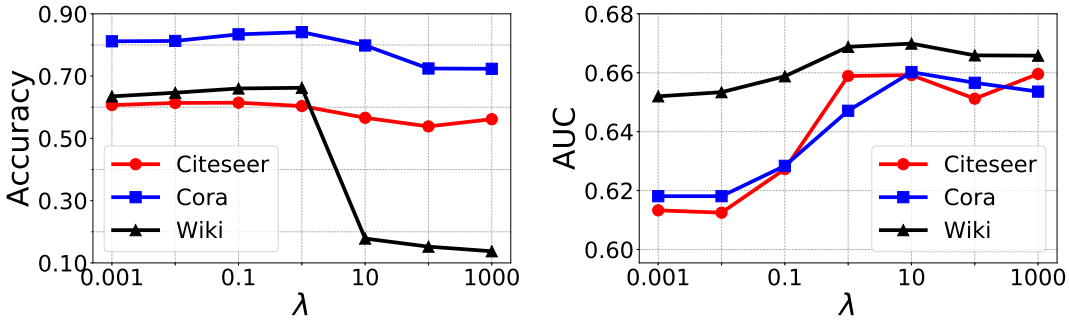
turbation for the regularization term contributes little to the model generalization performance again. It is understandable, since the expected dot product between any reference vector and the random perturbation from a zero mean Gaussian distribution is zero, and thus the regularization term will barely affect the embedding learning.

Secondly, DWNS_AdvT and DWNS_iAdvT consistently outperform DWNS across all different training ratios in the three datasets, with the only exception of DWNS_iAdvT in Citeseer when the training ratio is 3%. Specifically, DWNS_AdvT achieves 5.06%, 6.45% and 5.21% performance gain over DWNS on average across all training ratios in Cora, Citeseer and Wiki respectively, while the improvement over DWNS for DWNS_iAdvT are 2.35%, 4.50% and 2.62% respectively. It validates that adversarial perturbation can provide useful direction for generating adversarial examples, and thus brings significant improvements to model generalization ability after the adversarial training process. For DWNS_iAdvT, it brings less performance gain compared with DWNS_AdvT, which might because the restriction on perturbation direction limit its regularization ability for classification tasks. In this case, there is a tradeoff between interpretability and regularization effect.

Thirdly, AIDW achieves better results than DeepWalk, LINE and GraRep, which shows that global regularization on embedding vectors through adversarial learning can help improve model generalization performance. Our proposed methods, especially DWNS_AdvT, demonstrate superiority over all the state-of-the-art baselines, including AIDW and node2vec, based on experimental results comparison. We can summarize that the adversarial training regularization method has advantages over the GAN-based global regularization methods in three aspects, including more succinct architecture, better computational efficiency and more effective performance contribution.



(a) Noise level ϵ.



(b) Adversarial regularization strength λ.

Figure 4.5: Impact of hyper-parameters on node classification (left, training ratio 50%) and link prediction (right).

4.4.5 Parameter Sensitivity

We conduct parameter sensitivity analysis with link prediction and multi-class classification on Cora, Citeseer and Wiki in this section. Here we only present the results for DWNS_AdvT due to space limitation. Adversarial training regularization method is very succinct. DWNS_AdvT only has two more hyper-parameters compared with DWNS, which are noise level ϵ and adversarial regularization strength λ. Note that when studying one hyper-parameter, we follow default settings for other hyper-parameters. The experimental settings of link prediction and node classification have been explained in Section 4.4.2.

Fig. 4.5(a) presents the experimental results when varying ϵ from 0.1 to 5.0. For both learning tasks, we can find that the performance in these three datasets first

improves with the increase of ϵ, and then drops dramatically after ϵ passing some threshold. It suggests that appropriate setting of ϵ improves the model robustness and generalization ability, while adversarial perturbation with too large norm constraint can destroy the learning process of embedding vectors. Besides, it can be easily noticed that the best settings of ϵ are different for different datasets in general. Specifically, Citeseer has the best results in both link prediction and node classification when $\epsilon = 1.1$, Cora achieves the best results when $\epsilon = 0.9$, while the best setting of ϵ for Wiki is around 0.5. Based on the experimental results on these three datasets only, it seems that the denser the network is, the smaller the best noise level parameter ϵ should be.

We conduct link prediction and node classification on three datasets with the adversarial regularization strength λ from the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. Fig. 4.5(b) displays the experimental results. For node classification, the best result is obtained when λ is set to around 1, larger values can result in performance degradation. For example, the classification accuracy on Wiki drops dramatically when λ reaches 10, and larger setting produces worse results. For link prediction, the performance is quite consistent among the three datasets. Specifically, when λ increases from 0.001 to 10, the AUC score shows apparent increase for all datasets, and then tends to saturate or decrease slightly. Empirically, 1 is an appropriate value for the adversarial regularization strength λ.

4.5 Related Work

Network Embedding. Some early methods, such as IsoMap [130] and LLE [102], assume the existence of a manifold structure on input vectors to compute low-dimensional embeddings, but suffer from the expensive computation and their inability in capturing highly non-linear structural information of networks. More re-

cently, some negative sampling approach based models have been proposed, including DeepWalk [95], LINE [126] and node2vec [41], which enjoys two attractive strengths: firstly, they can effectively capture high-order proximities of networks; secondly, they can scale to the widely existed large networks. DeepWalk obtains node sequences with truncated random walk, and learns node embeddings with Skip-gram model [82] by regarding node sequences as sentences. node2vec differs from DeepWalk by proposing more flexible random walk method for sampling node sequences. LINE defines first-order and second-order proximities in network, and resorts to negative sampling for capturing them.

Further, some works [12, 90, 147] tried to preserve various network structural properties in embedding vectors based on matrix factorization technique. GraRep [12] can preserve different k-step proximities between nodes independently, HOPE [90] aims to capture asymmetric transitivity property in node embeddings, while N-NMF [147] learns community structure preserving embedding vectors by building upon the modularity based community detection model [85]. Meanwhile, deep learning embedding models [13, 143, 112, 113] have also been proposed to capture highly non-linear structure. DNGR [13] takes advantages of deep denoising autoencoder for learning compact node embeddings, which can also improve model robustness. SDNE [143] modifies the framework of stacked autoencoder to learn both first-order and second-order proximities simultaneously. DNE-SBP [113] utilizes a semi-supervised SAE to preserve the structural balance property of the signed networks. Both GraphGAN [144] and A-RNE [24] leverage generative adversarial networks to facilitate network embedding, with the former unifies the generative models and discriminative models of network embedding to boost the performance while the latter focuses on sampling high-quality negative nodes to achieve better similariy ranking among node pairs.

However, the above mentioned models mainly focus on learning different net-

work structures and properties, while neglecting the existence of noisy information in real-world networks and the overfitting issue in embedding learning process. Most recently, some methods, including ANE [23] and NetRA [158], try to regularize the embedding learning process for improving model robustness and generalization ability based on generative adversarial networks (GANs). They have very complicated frameworks and suffer from the well-recognized hard training problems of GANs. Furthermore, these two methods both encourage the global smoothness of the embedding distribution, while in this chapter we utilize a more succinct and effective local regularization method.

Adversarial Machine Learning. It was found that several machine learning models, including both deep neural network and shallow classifiers such as logistic regression, are vulnerable to examples with imperceptibly small designed perturbations, called adversarial examples [123, 40]. This phenomenon was firstly observed in areas like computer vision with continuous input vectors. To improve model robustness and generalization ability, adversarial training method [40] is shown to be effective. It generates adversarial perturbations for original clean input with the aim of maximizing current model loss, and further approximates the difficult optimization objective with first-order Taylor Series. Such method has also been applied to text classification problem in [83, 107] by defining the perturbation on continuous word embeddings, and recommendation in [49] by generating adversarial perturbations on model parameters. However, to the best of our knowledge, there is no practice of adversarial training regularization for graph representation learning.

For graph structured data, they are fundamentally different from images because of their discrete and indifferentiable characteristics. Some existing works [22, 167, 19] aimed to explore how to generate the adversarial examples in the discrete, binary graph domain, and whether similar vulnerability exists in graph analysis applications. In [22], adversarial attacks are generated by modifying combinatorial structure of

graph with a reinforcement learning based method, which is shown to be effective in Graph Neural Network models. Both [167] and [19] designed attack methods to Graph Convolutional Network [62]. Particularly, NETTACK [167] focuses on attributed graph classification problem and FGA [19] tackles network representation learning. However, all of them studied adversarial attack methods without providing any defense algorithms for improving the robustness of existing methods against these attacks. Differently, in this chapter, we aim to propose adversarial regularization method for network embedding algorithms to improve both model robustness and generalization ability.

4.6 Conclusion

In this chapter, we proposed two adversarial training regularization methods for network embedding models to improve the robustness and generalization ability. Specifically, the first method is adapted from the classic adversarial training method by defining the perturbation in the embedding space with adaptive L_2 norm constraint. Though it is effective as a regularizer, the lack of interpretability may hinder its adoption in some real-world applications. To tackle this problem, we further proposed an interpretable adversarial training method by restricting the perturbation directions to embedding vectors of other nodes, such that the crafted adversarial examples can be reconstructed in the discrete graph domain. Both methods can be applied to the existing embedding models with node embeddings as model parameters, and DeepWalk is used as the base model in the chapter for illustration. Extensive experiments prove the effectiveness of the proposed adversarial regularization methods for improving model robustness and generalization ability. Future works would include applying adversarial training method to the parameterized network embedding methods such as deep learning embedding models.

Chapter 5

Ranking Network Embedding via Adversarial Learning

Network Embedding is an effective and widely used method for extracting graph features automatically in recent years. To handle the widely existed large-scale networks, most of the existing scalable methods, e.g., DeepWalk, LINE and node2vec, resort to the negative sampling objective so as to alleviate the expensive computation. Though effective at large, this strategy can easily generate false, thus low-quality, negative samples due to the trivial noise generation process which is usually a simple variant of the unigram distribution. In this chapter, we propose a Ranking Network Embedding (RNE) framework to leverage the ranking strategy to achieve scalability and quality simultaneously. RNE can explicitly encode node similarity ranking information into the embedding vectors, of which we provide two ranking strategies, vanilla and adversarial, respectively. The vanilla strategy modifies the uniform negative sampling method with a consideration of edge existance. The adversarial strategy unifies the triplet sampling phase and the learning phase of the model with the framework of generative adversarial networks. Through adversarial training, the triplet sampling quality can be improved thanks to a softmax generator which constructs hard negatives for a given target. The effectiveness of our RNE framework

is empirically evaluated on a variety of real-world networks with multiple network analysis tasks.

5.1 Introduction

Network embedding, i.e., learning low-dimensional representations for nodes in graph-structured data, can help encode meaningful semantic, relational and structural information of a graph into embedding vectors. Typically, such learning process is conducted in an unsupervised manner [95, 126, 12] due to the lack of labeled data, and thus the learned representations can be used to facilitate different kinds of downstream tasks such as network visualization, link prediction and node classification. In real-world applications, data entities with complicated relationships can be well organized with graphs. For example, paper citation networks characterize the information of innovation flow, social networks entail complicated relationships among people, groups and organizations, and protein-protein interaction networks capture information between different proteins. Therefore, it is of great application interest to develop effective and scalable methods for unsupervised network embedding.

Network data are usually high-dimensional, very sparse and non-linear, which makes network embedding a challenging problem. Some classical methods, such as MDS [20], IsoMap [130] and LLE [102], can be used for network representation learning. However, they can neither effectively capture highly nonlinear structure of networks, nor scale to large networks. When handling large-scale networks, DeepWalk [95], LINE [126] and node2vec [41] are shown to be quite effective and efficient. These three methods preserve network structural properties in the embedding vectors through the negative sampling technique [82]. The negative sampling method is a simplified variant of negative contrastive estimation [45], which can help speed up the training process of the model. However, since the negative samples are con-

structed according to a unigram noise generation process, this strategy may generate false negative samples that violate pairwise relationships presented in the network structure. Here, we aim to answer two questions: 1) can we find some other ways for encoding pairwise relationships into node representations instead of the negative sampling approach? 2) how to sample better negative nodes for target-positive pairs (i.e., closely related node pairs) for training?

In this chapter, we propose a Ranking Network Embedding (**RNE**) framework based on triplet ranking loss for preserving pairwise relationships of nodes in embedding vectors. Specifically, we firstly construct triplets based on network structure where each triplet consists of a target, a positive and a negative node. In the training process, the distance between embedding vectors of the target and positive node will be minimized while the distance between that of target and negative node will be maximized until they are separated by a predefined margin. Different from the negative sampling technique used in [95, 126, 41], the ranking strategy enforces a non-trivial margin between similar node pairs and dissimilar ones, thus explicitly encodes similarity ranking information among node pairs into the embedding vectors.

With the RNE framework, we propose two network embedding models by using a vanilla ranking strategy and an adversarial ranking technique respectively. In the vanilla RNE model, we utilize a simple negative node sampling method to construct triplets, which uniformly samples nodes from the node set without direct link to the target. This vanilla approach can perfectly avoid false negative samples while maintain the efficiency. Though works well to some extend, this vanilla strategy may also generate totally unrelated negative nodes for the target node, which will be of little help for the training. This phenomenon is even more common in very high-dimensional and sparse networks. To improve the vanilla RNE, we propose a generative adversarial model to unify the triplet sampling process and the learning process with the framework of generative adversarial networks (GANs) [39], which

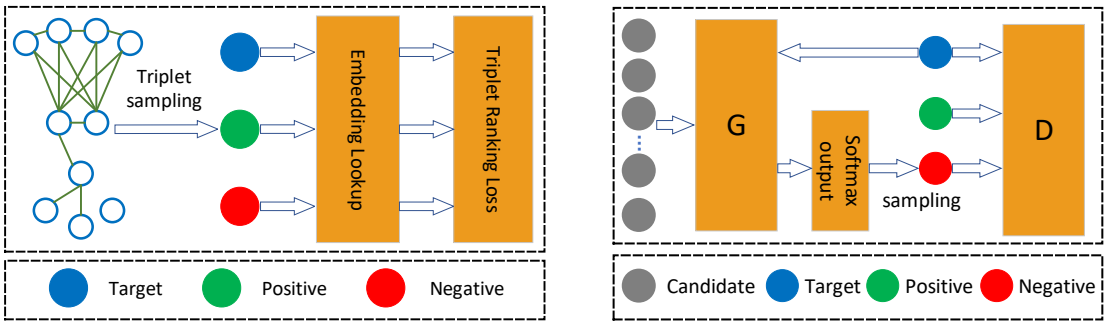
leads to an adversarial RNE model. It leverages a generator for generating hard negative nodes with respect to a given target to help construct high-quality triplets, and thus achieves better node similarity rankings in the embedding space. We empirically evaluate the proposed vanilla and adversarial RNE models through several network analysis tasks, including network visualization, link prediction and node classification, on benchmark datasets. Experimental results show that both models achieve competitive performance with state-of-the-art methods.

5.2 RNE: Ranking Network Embedding

5.2.1 Framework

The framework of Ranking Network Embedding method is shown in Figure 5.1(a). It consists of two phases, i.e., the triplet construction phase and the learning phase. Firstly, we leverage some sampling methods to construct triplets based on network structure, which can help specify the similarity ranking of some pairwise relationships. Then, in the learning process, triplet ranking loss is minimized by directly updating embeddings to pull similar nodes closer in the embedding space, while pushing dissimilar nodes apart.

To help better understand our model, we first introduce some notations and describe the research problem. A network is denoted as $\mathcal{G} = (V, E)$, with a set of nodes V representing data entities and a set of edges E each representing the relationship between two nodes. We mainly consider undirected graph in this chapter. Given a graph $\mathcal{G}$, we aim to learn low-dimensional representations $\boldsymbol{u_i} \in R^d$ for each node $v_i \in V$, which can capture network structural properties. We denote U as embedding matrix with $\boldsymbol{u_i}$ as its ith row.



(a) RNE. (b) Adversarial RNE.

Figure 5.1: Model Architecture.

5.2.2 Vanilla Ranking Network Embedding

The vanilla RNE model is a simple instantiation of the proposed RNE framework with uniform negative sampling method. Some detailed descriptions of its triplet sampling method and loss function are provided below.

Vanilla Triplet Sampling

Triplet sampling method plays an important role in learning good embedding vectors for downstream learning tasks. The constructed triplets directly specify pairwise relationships from network structure which will be regarded as ground-truth in learning process to be encoded into embedding vectors. We only explicitly consider first-order proximity when constructing positive pairs. The triplet set $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \{(v_t, v_p, v_n) | (v_t, v_p) \in E, (v_t, v_n) \notin E\}, \tag{5.1}$$

where (v_t, v_p, v_n) is a triplet with v_t, v_p and v_n as the target, positive and negative node, respectively. Since network is usually very sparse, for each positive pair, there can be a large number of negative nodes. To improve model efficiency, we only uniformly sample K negative nodes from the negative space for each positive pair.

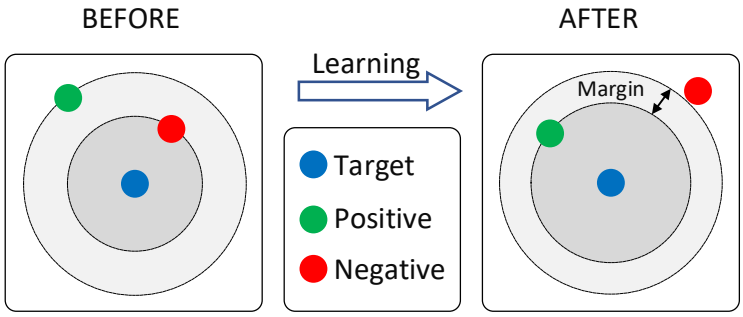


Figure 5.2: The triplet ranking loss minimizes the distance between a target node and a positive node while maximizing that of the target and a negative node until they are separated by at least a margin distance. The pairwise relationships can be well preserved in embedding vectors after the learning process.

Triplet Ranking Loss

For vanilla RNE model, we seek to minimize the following loss function:

$$\mathcal{L} = \sum_{(v_t, v_p, v_n) \in \mathcal{T}} [m + D(v_t, v_p; \theta_D) - D(v_t, v_n; \theta_D)]_+, \tag{5.2}$$

where $[x]_+$ denotes the positive part of x, $D(v_1, v_2; \theta_D)$ is a distance function of two nodes, θ_D is the union of all node embeddings, and $m > 0$ is a margin hyper-parameter separating the positive pair and the corresponding negative one. We use the squared L2 distances in the embedding space, i.e., $D(v_1, v_2; \theta_D) = \|\boldsymbol{u_1} - \boldsymbol{u_2}\|^2$. The triplet ranking loss explicitly encodes similarity ranking among node pairs into the embedding vectors, and the visualization explanation can be found in Figure 5.2 [108].

5.2.3 Adversarial Ranking Network Embedding

For the vanilla RNE model, we only use uniform negative sampling method for constructing triplets. It can easily generate totally unrelated negative nodes for the target node due to the sparsity and high-dimensionality of the network, which will be of little help for the training process. To help alleviate this problem, we propose an Adversarial Ranking Network Embedding model, which unifies the triplet sampling

phase and the learning phase of the RNE method with the framework of GANs. The model architecture is presented in Figure 5.1(b). It consists of a generator G and a discriminator D (we abuse the notation and directly use the distance function D to represent the discriminator). In the learning process, the discriminator tries to pull similar nodes closer in the embedding space, while pushing dissimilar nodes apart. The generator aims to generate difficult negative nodes for a given target from a set of negative candidates by optimizing its own parameters.

Discriminator

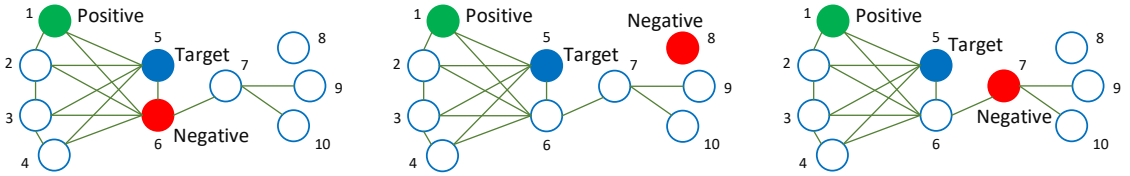
The discriminator is aimed at optimizing the following triplet ranking loss function similar to the vanilla RNE model:

$$\mathcal{L}_D = \sum_{(v_t,v_p)\in\mathcal{P}} \mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[m + D(v_t, v_p; \theta_D) - D(v_t, v_n; \theta_D)]_+, \tag{5.3}$$

where $\mathcal{P} = \{(v_1, v_2), (v_2, v_1)|(v_1, v_2) \in E\}$ is the positive pair set in graph $\mathcal{G}$, and $G(\cdot|v_t; \theta_G)$ is the generator. Only first-order proximity is directly considered, and each edge $(v_i, v_j) \in E$ corresponds to two positive pairs (v_i, v_j) and (v_j, v_i). Particularly, a softmax generator is employed to construct high-quality triplets instead of simple uniform sampling method. More detailed illustrations of this sampling method will be introduced below. Note that Eq. (5.3) can be directly optimized with gradient descent technique.

Generator

Softmax function is widely used in network embedding literature [95, 154] to model node conditional probability. In this work, we also employ softmax function as the generator to sample negative nodes given a target, but it is defined over the negative node space with respect to the given positive pair according to network structure. Specifically, the generator $G(v_n|v_t; \theta_G)$ is defined as a softmax function over a set of



(a) Negative sampling. (b) Uniform sampling. (c) Adversarial sampling.

Figure 5.3: For the negative sampling approach, each node is sampled according to its unigram distribution (regard each node as a word) raised to the 3/4 power, which can violate pairwise relationships reflected by network structure. For example, node 6 is very likely to be sampled as negative node for target-positive pair (5, 1), even though node 5 and 6 have strong relationship. For our triplet sampling method, such problem can be well avoided. However, simple uniform sampling method can easily generate totally unrelated nodes (node 8 in the example graph), which can be improved with adversarial sampling method.

negative candidates:

$$G(v_n|v_t;\theta_G) = \frac{\exp(\boldsymbol{u_n^T u_t})}{\sum_{v_{n_i}\in Neg(v_t,v_p)} \exp(\boldsymbol{u_{n_i}^T u_t})}, \tag{5.4}$$

where $Neg(v_t, v_p) = \{v_{n_1}, v_{n_2}, \cdots, v_{n_{N_c}}\}$ is a set of negative candidates with size as N_c. In implementation, $Neg(v_t, v_p)$ is a subsample of the original negative space of the positive pair to reduce the computation complexity, which is actually a common practice in network embedding literature [95, 126]. For each positive pair, N_c negative nodes will be first uniformly randomly sampled from the negative space, and used as input for the generator. Then, a hard negative node will be sampled from $Neg(v_t, v_p)$ according to the probability distribution $G(v_n|v_t;\theta_G)$. Besides, in the training process, K hard negatives will be sampled for each positive pair.

The loss function of the generator is defined as follows:

$$\mathcal{L}_G = \sum_{(v_t,v_p)\in\mathcal{P}} \mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t, v_n;\theta_D)]. \tag{5.5}$$

It can encourage the softmax generator to generate useful negative nodes for a given positive pair instead of totally unrelated ones. The sampling process of hard negatives

is discrete, which hinders the objective from directly being optimized by gradient descent method as that of the discriminator. According to [109, 157], this loss can be optimized with the following policy gradient:

$$\begin{aligned}\nabla_{\theta_G}\mathcal{L}_G &= \nabla_{\theta_G}\sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)] \\ &= \sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)\nabla_{\theta_G}\log G(v_n|v_t;\theta_G)].\end{aligned} \tag{5.6}$$

The gradient of $\mathcal{L}_G$ is an expected summation of the gradient $\nabla_{\theta_G}\log G(v_n|v_t;\theta_G)$ weighted by the distance of node pair (v_t, v_n). When training the generator, the parameters will be shifted to involve high-quality negatives with high probability from softmax generator, i.e., node pairs (v_t, v_n) with small distance from discriminator will be encouraged to be generated. In practice, the expectation can be approximated with sampling in the negative space. Besides, the REINFORCE algorithm suffers from the notorious high variance, which can be alleviated by subtracting a baseline function from the reward term of the objective, i.e., adding a baseline function to the reward term in the loss [122]. Specifically, we replace $D(v_t, v_n;\theta_D)$ in the loss by its advantage function as follows:

$$D(v_t,v_n;\theta_D) + \sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)], \tag{5.7}$$

where $\sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)]$ is the average reward of the whole training set, and acts as the baseline function in policy gradient.

A comparison of the sampling methods is presented in Figure 5.3 with toy examples. Our proposed adversarial sampling method can help select difficult negative nodes with respect to given target. With high-quality triplets, the tricky pairwise relationship rankings can be encoded into node representations through the training of the discriminator as illustrated in Figure 5.2. Note that false negative nodes can still be generated by the generator due to the incompleteness and non-linearity of

Algorithm 5.1: The adversarial RNE algorithm

Input : $\mathcal{G}(V, E)$, Dimension d, Margin m, Negative size K, Candidate size N_c
Output: The parameters of Discriminator θ_D

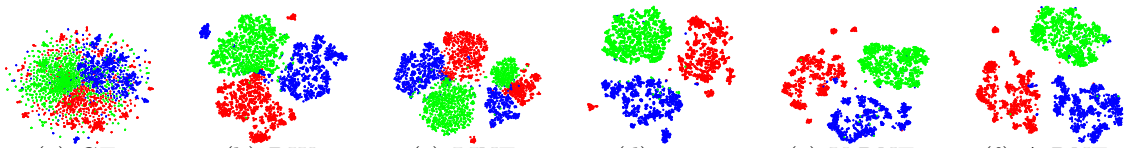
Initialize the Generator $G(v_n|v_t; \theta_G)$ and Discriminator $D(v_1, v_2; \theta_D)$ with pretrained embedding vectors;
while *not converge* **do**
    Sample a batch of positive pairs $\mathcal{B}$ from positive set $\mathcal{P}$;
    $\mathcal{T} = \{\}$; $\mathcal{N} = \{\}$;
    // Adversarial negative sampling with softmax generator
    for *each* $(v_t, v_p) \in \mathcal{B}$ **do**
        repeat
            Sample N_c negative candidates $Neg(v_t, v_p)$ uniformly from the negative space of (v_t, v_p);
            Sample a hard negative v_n from $Neg(v_t, v_p)$ according to $G(v_n|v_t, \theta_G)$;
            $\mathcal{T} = \mathcal{T} \cup \{v_n\}$; $\mathcal{N} = \mathcal{N} \cup \{Neg(v_t, v_p)\}$;
        until K *times*;
    // Parameters updating
    update θ_D according to Eq. (5.3) with $\mathcal{T}$ as training batch;
    update θ_G according to Eq. (5.5) and (5.4) with $\mathcal{T}$ and $\mathcal{N}$ as training batch;

Table 5.1: Statistics of benchmark datasets from real-world applications

Name	Citeseer [81]	Cit-DBLP [128]	PubMed [84]	CA-GrQc[65]	CA-HepTh[65]	Wiki [110]	USA-AIR [100]
$\|V\|$	3,264	5,318	19,717	5,242	9,877	2,363	1,190
$\|E\|$	4,551	28,065	44,335	14,484	25,973	11,596	13,599
Avg. degree	1.39	5.28	2.25	2.76	2.63	4.91	11.43
#Labels	6	3	3	-	-	17	4

the real-world networks, but in a very low probability since the subsampling trick is employed for generating negative candidates among a very large negative space. So, the embedding vectors can be improved in general. This is also validated by our experiments.

Algorithm 5.1 presents the pseudocode of the adversarial RNE model, which employs a joint training procedure. The overall time complexity of the algorithm is linear to the number of edges, i.e., $\mathcal{O}(dKN_c|E|)$ (d, K and N_c are some constants independent of the network size), which enables it scale to large networks.



(a) GF. (b) DW. (c) LINE. (d) n2c. (e) V-RNE. (f) A-RNE.

Figure 5.4: Visualization of Cit-DBLP network.

5.3 Experiments

5.3.1 Experiment Setup

Datasets

We conduct experiments on benchmark datasets from various real-world applications. Table 5.1 shows some statistics of them. Note that we regard all paper citation networks as undirected networks, and do some preprocessing on the original datasets by deleting self-loops and nodes with zero degree.

Baseline Models

We only consider scalable baselines in this chapter. Some matrix factorization based methods such as M-NMF [12, 147] are excluded from the baselines due to the $O(|V|^2)$ time complexity. The descriptions of the baseline models are as follows: Graph Factorization(GF) [3] directly factorizes the adjacency matrix to obtain the embeddings. DeepWalk(DW) [95] regards node sequence obtained from truncated random walk as word sequence, and then uses skip-gram model to learn node representations. LINE [126] preserves proximities through modeling node co-occurrence probability and node conditional probability. node2vec(n2v) [41] develops a biased random walk procedure to explore neighborhood of a node, which can strike a balance between local and global properties. We denote the vanilla RNE model as V-RNE, and the adversarial RNE model as A-RNE in the rest of the chapter.

Parameter Settings

The window size, walk length and the number of walks per node of both DeepWalk and node2vec are set to 10, 80 and 10, respectively. We use node2vec in an unsupervised manner by setting both in-out and return hyperparameters to 1.0 for fair comparison. For LINE, we follow the original paper [126] to set the parameters. For our method, the parameter settings are the margin $m = 2.5$, the negative size per edge $K = 5$,and the negative candidate size $N_c = 5$. The learning rate of V-RNE is set to 0.01, while A-RNE to 0.0001. L2-normalization is conducted on node embeddings for both the V-RNE and A-RNE model after each training epoch. Besides, the dimension of embedding vectors are set to 128 for all methods.

5.3.2 Network Visualization

We leverage a commonly used toolkit *t-SNE* [137] to visualize node embeddings of Cit-DBLP generated by different models. Cit-DBLP is a citation network constructed from the DBLP datsest [128], which consists of papers from publication venues including Information Sciences, ACM Transactions on Graphics and Human-Computer Interaction. These papers are naturally classified into three categories according to their publication venues, and represented with different colored nodes in the visualization.

Experimental results. Figure 5.4 displays the visualization results. Papers from different publication venues are mixed together terribly for GF as shown in Figure 5.4(a). In the center part of both DeepWalk and LINE, papers from different categories are mixed with each other. Visualizations from node2vec, V-RNE and A-RNE are much better as three clusters are formed with quite clear margin. Compared with V-RNE, A-RNE model has better visualization result, since the margin between different clusters are larger. The reason is that adversarial sampling method aims to

Table 5.2: AUC score for link prediction

Training ratio	80%			50%		
Dataset	Wiki	CA-GrQc	CA-HepTh	Wiki	CA-GrQc	CA-HepTh
GF	0.583 ± 0.008	0.593 ± 0.003	0.554 ± 0.001	0.566 ± 0.002	0.572 ± 0.003	0.531 ± 0.001
DeepWalk	0.656 ± 0.001	0.694 ± 0.001	0.683 ± 0.001	0.639 ± 0.001	0.657 ± 0.002	0.630 ± 0.001
LINE	0.649 ± 0.007	0.638 ± 0.005	0.630 ± 0.001	0.627 ± 0.014	0.600 ± 0.003	0.561 ± 0.002
node2vec	0.634 ± 0.016	0.690 ± 0.007	0.668 ± 0.003	0.621 ± 0.010	0.667 ± 0.010	0.624 ± 0.007
V-RNE	0.647 ± 0.008	0.691 ± 0.005	0.657 ± 0.005	0.627 ± 0.007	0.655 ± 0.004	0.606 ± 0.004
A-RNE	**0.670 ± 0.005**	**0.708 ± 0.004**	**0.688 ± 0.004**	**0.655 ± 0.006**	**0.673 ± 0.004**	**0.639 ± 0.004**

generate hard negative nodes, i.e., negative nodes near the boundary, which directly contributes to producing more clear margin between different clusters. On the whole, this experiment demonstrates that ranking network embedding method can help capture intrinsic structure of original network in embedding vectors.

5.3.3 Link Prediction

We conduct link prediction on three benchmark datasets. For each network, we randomly and uniformly sample 20% and 50% of the edges as test labels and use the remaining network as input to the models, i.e., training ratio as 80% and 50%. When sampling edges, we ensure the degree of each node is greater than or equal to 1 to avoid meaningless embedding vectors. The prediction performance is measured by AUC score. To calculate AUC score, we first obtain the edge features from the learned node embeddings through Hadamard product of embeddings of two endpoints as many other works [41], and then train a L2-SVM classifier with under-sampling to get prediction results.

Experimental results. The link prediction results are the average of 10 different runs, which are shown in Table 5.2. The AUC scores of A-RNE model consistently outperform those of the V-RNE model. It validates that A-RNE can help achieve better node similarity rankings in embedding space, since link prediction task can be considered as similarity ranking among node pairs. The performance of the proposed RNE method is competitive with the baselines, which shows that using ranking

Table 5.3: Accuracy (%) of multi-class classification on USA-AIR and PubMed

Dataset	USA-AIR					Pubmed				
Ratio	10%	30%	50%	70%	90%	10%	30%	50%	70%	90%
GF	41.10	42.21	42.27	41.12	41.60	35.63	36.69	37.56	37.74	38.08
DeepWalk	43.43	51.79	53.41	55.74	56.05	69.43	71.33	71.74	71.82	72.37
LINE	48.80	53.95	56.35	56.72	58.91	67.23	69.20	69.84	69.97	70.48
node2vec	42.76	47.07	48.62	49.86	50.76	79.66	80.89	81.09	81.07	81.27
V-RNE	55.20	58.96	60.05	61.29	61.09	77.56	79.08	79.39	79.46	79.73
A-RNE	**56.94**	**61.96**	**62.79**	**65.71**	**64.12**	**80.48**	**81.20**	**81.58**	**81.56**	**81.64**

strategy for learning node representations is a good practice. In particular, the AUC scores of A-RNE model are superior to all the baselines in all test datasets when the training ratios are 80% and 50%.

5.3.4 Node Classification

Node classification can be conducted to dig out missing information. In this section, we carry out experiments on the air-traffic network USA-AIR and paper citation network PubMed. The learned embedding vectors are used as feature input for the classification model. We randomly sample a portion of nodes as training data ranging from 10% to 90%, and the rest for testing. For both datasets, multi-class classification is conducted, and accuracy score is used for performance comparison. All experiments are conducted with support vector classifier in Liblinear package*[31] with default settings.

Experimental results. The experimental results are presented in Tables 5.3. Both V-RNE and A-RNE perform competitively with baseline models for these two datasets while varying the train-test split from 10% to 90%. It shows the effectiveness of the proposed Ranking Network Embedding models for learning discriminative embedding vectors for classification. Specifically, both V-RNE and A-RNE achieve better performance in USA-AIR, and A-RNE obtains the best results in these two datasets across all training ratios. In particular, A-RNE gives us 13.32% gain

*https://www.csie.ntu.edu.tw/∼cjlin/liblinear/

on average over the best baseline, i.e., LINE on USA-AIR. Besides, A-RNE consistently achieves more excellent performance than V-RNE as shown in the tables, which demonstrates that adversarial sampling method contributes to learning more discriminative node representations.

5.4 Related Work

Many scalable network embedding methods, such as DeepWalk [95], LINE [126] and node2vec [41], have been proposed to learn node representations to facilitate downstream tasks. They model node conditional probability with softmax function over the whole network, which is computationally expensive. Further, the negative sampling approach [82] is usually leveraged to replace the log likelihood objective, and thus enabling a scalability to large networks. However, it can generate some negative samples violating pairwise relationships reflected by network structure because of the simple unigram noise generation process. To handle this issue, we propose to use the triplet ranking loss to learn embedding vectors and leverage an adversarial sampling method to sample negative nodes. We noticed that the triplet ranking loss is also employed by [37] in learning embeddings, but for networks with node attributes.

Recently, some methods are proposed to learn node representations through adversarial training [23, 144]. In ANE [23], a prior distribution is imposed on node representations through adversarial training to achieve robustness. In [144], the authors proposed to unify the generative models and discriminative models of network embedding into the framework of GANs to help learn a stronger generator. Different from these two methods, our method aim to learn a stronger discriminator to obtain node representations.

Some knowledge graph embedding methods are also related [10, 146, 11]. TransE [10] is a translation-based knowledge graph embedding model, which learns embeddings

for both data entities and relations with triplet ranking loss. KBGAN [11] is an adversarial learning framework for knowledge graph embedding. Our method is by part inspired by these works. However, this line of research has notable differences with our work. Firstly, knowledge graph is fundamentally different from the networks we study. The assumption, that two connected nodes should be similar and close in embedding space, of network embedding methods does not hold in knowledge graph. Secondly, knowledge graph embedding learns representations for both data entities (nodes) and relations (edges) simultaneously, while network embedding is designed to learn node representations only.

5.5 Conclusion

This chapter presented a novel scalable Ranking Network Embedding method, which can explicitly encode node similarity ranking information into the embedding vectors. Firstly, a vanilla RNE model was proposed with uniform negative sampling method. Then, we improved the vanilla RNE model by unifying the triplet sampling phase and the learning phase with the framework of GANs which leads to an adversarial RNE model. The adversarial RNE model utilizes a softmax generator to generate hard negatives for a given target, which can help strengthen the discriminator. Empirical evaluations prove the effectiveness of the proposed method on several real-world networks with a variety of network analysis tasks.

Chapter 6

Network Transfer Learning via Adversarial Domain Adaptation with Graph Convolution

This chapter studies the problem of cross-network node classification to overcome the insufficiency of labeled data in a single network. It aims to leverage the label information in a partially labeled source network to assist node classification in a completely unlabeled or partially labeled target network. Existing methods for single network learning cannot solve this problem due to the domain shift across networks. Some multi-network learning methods heavily rely on the existence of cross-network connections, thus are inapplicable for this problem. To tackle this problem, we propose a novel network transfer learning framework AdaGCN by leveraging the techniques of adversarial domain adaptation and graph convolution. It consists of two components: a semi-supervised learning component and an adversarial domain adaptation component. The former aims to learn class discriminative node representations with given label information of the source and target networks, while the latter contributes to mitigating the distribution divergence between the source and target domains to facilitate knowledge transfer. Extensive empirical e-

valuations on real-world datasets show that AdaGCN can successfully transfer class information with a low label rate on the source network and a substantial divergence between the source and target domains.

6.1 Introduction

Node classification [8] is a central task in network analysis. It is an important building block of numerous real-world applications, such as product recommendation in e-commerce websites, advertisement distribution in social networks, and protein function identification for disease diagnosis. Many research efforts have been made to develop reliable and efficient methods for node classification in network data.

In the era of big data, massive amount of raw data in information networks is produced everyday. However, labeled data is significantly expensive and slow to acquire due to the high cost and long time of human annotations, making it difficult to train a well-generalized classifier [117]. Moreover, in some newly-formed networks such as a protein-protein interaction network constructed by some researchers, there may be no labels at all. Hence, it would be impossible to classify the nodes with only the information of this network. To tackle these issues, a promising approach is to utilize class information from other similar or related networks to assist in classification, i.e., transfer learning on network data [33, 114].

In this chapter, we consider a cross-network node classification problem that aims to leverage a partially labeled source attributed network to facilitate node classification in another completely unlabeled or partially labeled target attributed network (Figure 6.1). The challenges lie in several aspects. First, there may be a significant domain divergence between the source and target networks and they may not have many attributes in common. Second, there are no cross-network edges to propagate knowledge from the source network to the target network. Third, only a

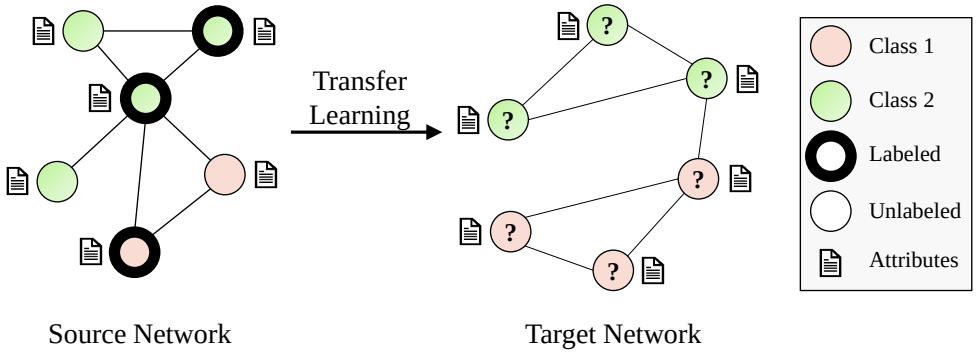


Figure 6.1: Cross-network node classification. We aim to transfer knowledge from a partially labeled source attributed network to assist the classification task in a completely unlabeled or partially labeled target attributed network. Here we use an unlabeled target network for illustration.

small portion of nodes in the source network are labeled.

Existing network embedding methods [95, 126, 41] are insufficient to address these challenges. They first learn compact node representations to preserve network structural information, and then train a classifier with the learned representations for node classification. Most of these methods learn node representations in an unsupervised manner, and are often less effective than graph-based semi-supervised learning methods for node classification [154, 62]. Moreover, topology-only embedding methods cannot be easily generalized to cross-network problems due to lack of a similarity preserving component to push nodes of the same category from two networks close in the embedding space [50].

Graph-based semi-supervised learning methods [154, 62] have been demonstrated highly effective for node classification in a single network with only a few labeled nodes. The recently proposed graph convolutional networks (GCN) [62] and follow-up works such as GraphSAGE [46] and GAT [138], naturally integrate network topology, node attributes and observed node labels into an end-to-end learning framework, and achieve superior performance on node classification. However, these methods are designed for learning tasks in a single network domain and will inherently have

difficulties in generalizing to another network domain that may have a substantially different attribute set.

There are some methods [86, 33] proposed to leverage the relationship between multiple networks to improve learning performance. Both EOE [149] and DMNE [86] learn embeddings for multiple networks simultaneously, but they heavily rely on the existence of cross-network connections, making them inapplicable for our problem. Currently there is little exploration of knowledge transfer across different networks to assist in learning tasks such as node classification.

Domain adaptation utilizes the knowledge of relevant source domain(s) to assist the same learning task in the target domain [93, 145]. Although there are many existing domain adaptation methods for vector-based data such as images and texts (bag-of-words) [35, 111], they are not applicable for graph-structured data, as entities in a graph are highly correlated with each other which violates the assumption of independent and identically distributed (IID) data samples in each individual domain. Little research has been conducted on domain adaptation for graph-structured data. CDNE [114] is the only attempt to our best knowledge, which learns transferable node embeddings for cross network learning tasks by minimizing the maximum mean discrepancy (MMD) loss. However, it cannot jointly model network structures and node attributes, which might limit its modeling capacity. Besides, it heavily relies on the preprocessing of the adjacency matrix with the positive pointwise mutual information (PPMI) matrix, which makes the sparse adjacency matrix denser and thus aggravates computational complexity due to the autoencoder-based model architecture.

To address the challenges for cross-network node classification, we propose a novel network transfer learning framework AdaGCN that is based on **a**dversarial **d**omain **a**daptation with **g**raph **c**onvolutional **n**etworks. The idea is two-fold: to learn class discriminative node representations via graph convolutional networks, and to learn

domain invariant node representations via adversarial learning. Hence, AdaGCN consists of a semi-supervised learning component and an adversarial domain adaptation component.

On one hand, the semi-supervised component is dedicated to learning discriminative node representations for classification with the available labeled data from both the source and target networks. GCN enables training a well-behaved classifier with even only a small set of labeled nodes in the source network (as shown in Section 6.4.2). However, the original GCN layer only conducts Laplacian smoothing on nearby nodes' features within one hop, and it requires stacking many layers to increase the smoothing level, which will greatly increase the number of trainable parameters and result in overfitting [68, 69]. To alleviate this issue, we propose to use an improved GCN layer designed with a smoothing strength hyper-parameter [69], which makes the model more efficient.

On the other hand, the adversarial domain adaptation component is aimed at mitigating the distribution shift between the source and target domains to encourage knowledge transfer by learning domain invariant representations via adversarial learning. Specifically, we model the domain adaptation process as a two-player game similar to GANs [39], where the representation learner GCN acts as the generator for learning domain invariant node representations while a domain critic as the discriminator is optimized to distinguish node representations from the source and target networks. By combining the two components, AdaGCN can learn both class discriminative and domain invariant node representations for transferring class information across networks.

Extensive experiments on real attributed networks show that AdaGCN can work in both unsupervised setting (i.e., completely unlabeled target network) and semi-supervised setting (i.e., scarcely labeled target network). Besides, it has low dependence on the common attributes shared by the source and target networks. The

main contributions of this chapter can be summarized as follows:

- We pioneer in studying a challenging network transfer learning problem under a realistic setting, where a partially labeled source network is utilized to assist node classification in a completely unlabeled or partially labeled target network.
- We develop a novel and principled framework for network transfer learning by efficiently integrating techniques of adversarial domain adaptation and graph convolution.
- We conduct extensive experiments on real-world information networks to verify the effectiveness of our model, which demonstrates its superior performance compared with state-of-the-art baselines, impressive label efficiency, and good model robustness against distribution discrepancy.

The organization of this chapter is as follows. We formulate the research problem in Section 6.2. We present a detailed description of the proposed methods in Section 6.3. In Section 6.4, the experimental results and analysis are provided. Then, we review the literature in Section 6.5. Finally, a short summary with the contributions and possible directions of future work are included in Section 6.6.

6.2 Problem Definition

In this chapter, we study domain adaptation for network data, i.e., leveraging the information of a source network to assist node classification in a completely unlabeled or partially labeled target network. The source network can be either partially labeled or fully labeled. In this section, we formally define the research problem and introduce notations used throughout the chapter as summarized in Table 6.1.

Denote by $G^s = (V^s, A^s, X^s)$ the source network, where V^s is the node set ($n^s = |V^s|$), $A^s \in \mathbb{R}^{n^s \times n^s}$ is the weighted adjacency matrix with A^s_{ij} quantifying the strength

Table 6.1: Notations

Notation	Description
$G = (V, A, X)$	A weighted attributed network
V	Node set of G
A	Weighted adjacency matrix of G
X	Feature matrix of G
$G^s = (V^s, A^s, X^s)$	Source network
$G^t = (V^t, A^t, X^t)$	Target network
V^{sl}	Set of all labeled nodes in G^s
Y^{sl}	Label matrix of V^{sl}
$\mathcal{X}^s$, $\mathcal{X}^t$	Sets of node attributes of G^s and G^t
$\mathcal{X}$	Set of node attributes of both G^s and G^t
$\mathcal{Y}$	Label set of both G^s and G^t
n^s, n^t	# of nodes in G^s and G^t
n^{sl}	# of labeled nodes in G^s
c^s, c^t	# of node attributes in G^s and G^t
c	# of node attributes in $\mathcal{X}$
L	# of categories in label space $\mathcal{Y}$
f_g, f_c, f_d	Representation learner, label classifier, and domain critic
$\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_c$, $\boldsymbol{\theta}_d$	Sets of model parameters in f_g, f_c, and f_d
H_g^s, H_g^t	Source and target node representations
λ	Domain adaptation coefficient
γ	Gradient penalty coefficient
n_d	Domain critic training step per iteration
n_I	Smoothing parameter of the IGCN layer
α_1, α_2	Learning rates of domain critic, and representation learner and label classifier

of connection between nodes i and j, and $X^s \in \mathbb{R}^{n^s \times c^s}$ is the feature matrix with c^s as the number of node attributes in G^s and the i-th row of X^s as the feature vector associated with node i. Denote by V^{sl} the set of labeled nodes in G^s and $Y^{sl} \in \mathbb{R}^{n^{sl} \times L}$ the label matrix of V^{sl}, where $Y^{sl}_{ik} = 1$ if node $i \in V^{sl}$ is associated with label k and $Y^{sl}_{ik} = 0$ otherwise.

Similarly, the target network is represented as $G^t = (V^t, A^t, X^t)$, where V^t is the node set ($n^t = |V^t|$), $A^t \in \mathbb{R}^{n^t \times n^t}$ is the weighted adjacency matrix, and $X^t \in \mathbb{R}^{n^t \times c^t}$ is the feature matrix with c^t as the number of node attributes in G^t. The target network G^t can be either completely unlabeled or partially labeled. Here, we assume that it is completely unlabeled for simplicity, but our method can be straightforwardly

extended to the partially labeled setting and we have conducted experiments for both scenarios in Sections 6.4.2 and 6.4.3.

The source network and the target network may contain different attributes. Denote by $\mathcal{X}^s$ and $\mathcal{X}^t$ the set of node attributes in G^s and G^t respectively. We construct a new attribute set $\mathcal{X} = \mathcal{X}^s \cup \mathcal{X}^t$, where $c = |\mathcal{X}|$ represents the total number of attributes. We then reformulate the feature matrix of both G^s and G^t to make them include all the attributes in $\mathcal{X}$. With a slight abuse of notation, we still use $X^s \in \mathbb{R}^{n^s \times c}$ and $X^t \in \mathbb{R}^{n^t \times c}$ to represent the newly formed feature matrices of G^s and G^t. In particular, X^r_{ik} $(r \in \{s, t\})$ represents the value of the k-th attribute associated with node i in G^r and $X^r_{ik} = 0$ indicates that it is not associated with node i.

Define a network domain as $D = \{G, f(G)\}$, which includes an attributed network G and a function $f(G)$ for the node classification task. Then, the source network domain and the target network domain can be represented by $D^s = \{G^s, f(G^s)\}$ and $D^t = \{G^t, f(G^t)\}$, respectively. The problem considered in this chapter is similar to the conventional domain adaptation problem as in [93, 145]. Specifically, there exists a domain divergence between the source and target networks, i.e., $D^s \neq D^t$, but the label space $\mathcal{Y} = \{1, \cdots, L\}$ is the same, and our goal is to learn a classifier f to accurately classify the nodes in the target network with the assistance of the partially labeled source network.

6.3 Proposed Method

6.3.1 An Overview of Model Architecture

To solve cross-network node classification problem, two major challenges need to be addressed. Firstly, how to fully exploit the available data information including graph structures, node attributes and observed node labels to learn useful node

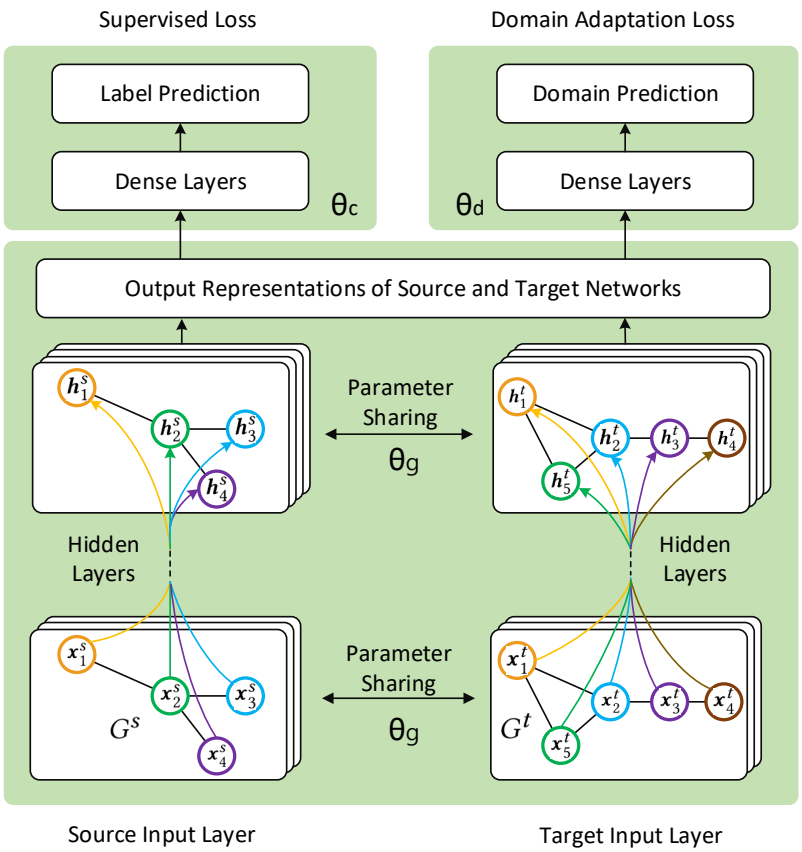


Figure 6.2: Model architecture of AdaGCN.

representations for the two networks? Secondly, how to overcome the serious domain divergence between two networks to facilitate knowledge transfer with the absence of cross network edges and only a few common node attributes across networks?

To address the first challenge, we leverage graph convolution to integrate network topology and node attributes in a semi-supervised learning model, which is capable of learning discriminative node representations with available node labels. To tackle the second challenge, we manage to mitigate distribution discrepancy between two networks with the technique of adversarial domain adaptation. In particular, we propose a network transfer learning framework AdaGCN by naturally combining the techniques of adversarial domain adaptation and graph convolution. The model architecture is shown in Figure 6.2. It consists of two components: a semi-supervised

learning component and an adversarial domain adaptation component. With the cooperation of them, AdaGCN can learn both class discriminative and domain invariant node representations, thus enabling classifying nodes in the target network with only a few labeled nodes in the source network. Note that our model is also applicable for the semi-supervised scenario where the target network is partially labeled.

6.3.2 Network Representation Learning

We propose to use graph convolution to jointly model network structures and node attributes for learning network representations, which has recently been demonstrated highly effective in various learning tasks such as node classification [62], graph clustering [161] and social recommendation [32].

Graph convolution is an operation that applies a linear graph convolutional filter [118, 105] $\hat{A} \in \mathbb{R}^{n \times n}$ on a graph signal $\boldsymbol{h} \in \mathbb{R}^n$ and outputs a new signal $\bar{\boldsymbol{h}} \in \mathbb{R}^n$ (n is the number of nodes in the underlying graph):

$$\bar{\boldsymbol{h}} = \hat{A}\boldsymbol{h}. \tag{6.1}$$

The graph filter $\hat{A}$ is a matrix designed by manipulating the spectrum of the underlying graph. The graph signal $\boldsymbol{h}$ is a real-valued function defined on the nodes of the graph, i.e., each node is associated with a real number. For example, a column of the node feature matrix X can be considered as a graph signal.

Graph convolution provides a principled way to combine graph structures and node features for learning useful node representations. For the graph convolutional networks (GCN) proposed in [62], the graph filter is a renormalized adjacency matrix, which actually performs Laplacian smoothing that updates the features of each node with a weighted average of its own and neighbors' to obtain smooth embeddings [68]. Further, it was shown in [69] that to produce smooth embeddings for

nodes in the same cluster, the graph filter $\hat{A}$ needs to be low-pass. With a proper low-pass graph filter, graph convolution will generate useful representations that help to ease knowledge transfer across networks and node classification in the target network.

In this chapter, we propose two methods for learning network representation with graph convolution. The first one is based on the layer-wise propagation rule of GCN. Specifically, the hidden representations of the k-th convolutional layer in the feature extractor are learned by:

$$H_g^{(k)} = \sigma(\hat{A}H_g^{(k-1)}W_g^{(k)}), \tag{6.2}$$

where $\hat{A}$ is a renormalized adjacency matrix with a self-loop at each node, $H_g^{(k-1)}$ is the output of the previous layer ($H_g^0 = X$), $W_g^{(k)}$ is a projection matrix with trainable parameters, and $\sigma(\cdot)$ is the activation function. As illustrated in Figure 6.2, we use two GCNs for learning node representations for the source and target networks respectively, but they share a common set of trainable parameters ($W_g^{(k)}$) so as to help transfer knowledge across networks. For simplicity of notation, we denote a GCN as $f_g(A, X; \boldsymbol{\theta}_g)$, which takes the graph adjacency matrix A and the feature matrix X as input, and $\boldsymbol{\theta}_g$ represents the trainable parameters. Then, we can obtain the output node representations of the source and target networks as:

$$H_g^r = f_g(A^r, X^r; \boldsymbol{\theta}_g),\ r \in \{s, t\}, \tag{6.3}$$

where $[H_g^r]_i$ as the ith row of H_g^r is the representation of node i.

However, with the GCN layer defined as in Eq. (6.2), one has to stack multiple layers to increase the strength of feature smoothing, which will also increase model complexity because of the accompanied trainable parameters in each layer, and thus can easily cause overfitting, especially for learning tasks with low source training rates. To address this issue, we propose to use an improved GCN (IGCN) layer proposed in [69] to improve the strength of the graph convolutional filter for learning

better representations. Then, for our second method, the hidden representations of the k-th convolutional layer are obtained with:

$$H_g^{(k)} = \sigma(\hat{A}^{n_I} H_g^{(k-1)} W_g^{(k)}), \tag{6.4}$$

where n_I is the exponent of $\hat{A}$, i.e., the smoothing parameter. By setting an appropriate n_I, we can easily control the smoothing strength of graph convolution to facilitate knowledge transfer and classification while avoiding overfitting. As suggested in [69], normally, larger n_I should be used with lower source training rates.

6.3.3 Semi-Supervised Learning

In AdaGCN, the node representations of the source and target networks learned by GCNs will be fed to a classifier for label prediction, and together they form the semi-supervised learning component. The classifier could be a single layer logistic regression classifier or a multi-layer perceptron. We denote the classifier as $f_c(H; \boldsymbol{\theta}_c)$, where H represents the node representations as input and $\boldsymbol{\theta}_c$ represents the trainable parameters. We then denote the prediction scores of nodes in the source and target networks as:

$$\hat{Y}^r = f_c(H_g^r; \boldsymbol{\theta}_c),\ r \in \{s, t\}, \tag{6.5}$$

where H_g^r are the node representations generated by GCNs and $\hat{Y}_{ik}^r$ ($r \in \{s, t\}$) represents the prediction score for node i in class k. One can conduct multi-class or multi-label classification by changing the activation function of the output layer in the classifier $f_c(\cdot)$. For multi-class classification, the activation function can be the softmax function. For multi-label classification, the activation function is the sigmoid function. We use the cross-entropy error over all the labeled nodes in the source network as the classification loss:

$$\mathcal{L}_c = -\frac{1}{n^{sl}} \sum_{i=1}^{n^{sl}} \sum_{k=1}^{L} Y_{ik}^{sl} \log(\hat{Y}_{ik}^{sl}), \tag{6.6}$$

where $\hat{Y}^{sl}$ is the prediction score matrix of the labeled nodes V^{sl} in the source network. Note that our method can be easily extended to the semi-supervised setting by incorporating available target labels into the above cross entropy loss.

6.3.4 Adversarial Domain Adaptation

Domain adaptation theory [7, 6] suggests that when an algorithm cannot learn to identify the domain of given hidden representations, they are good for knowledge transfer across domains. In AdaGCN, we leverage the adversarial domain adaptation method [35, 111] to achieve this goal. Specifically, we model the domain adaptation process as a two-player game similar to GANs [39], where the representation learning networks $f_g(A, X; \boldsymbol{\theta}_g)$ is acting as the generator for learning network invariant node representations, while a domain critic acting as the discriminator is optimized to distinguish node representations from the source and target networks. After the adversarial learning, network invariant representations can be obtained, and class information can be transferred from the source network to the target network.

In the original GANs [39], the domain critic is a binary classifier, and the generator and the discriminator fight against each other over a log likelihood objective. However, directly formulating the problem as a binary classification problem and leveraging cross-entropy loss for model optimization may suffer from training instability such as mode collapse [4, 43]. To improve learning stability, we instead minimize the Wasserstein-1 distance between the source and target distributions of node representation as suggested in [4, 43, 111].

We set the domain critic as a fully-connected neural network that takes a node representation as input and returns a real number. Denote by $f_d(\boldsymbol{h}; \boldsymbol{\theta}_d)$ the domain critic, where $\boldsymbol{h} = [f_g(A, X; \boldsymbol{\theta}_g)]_v$ is the representation of node v generated by a GCN with X as the input node feature matrix, and $\boldsymbol{\theta}_d$ represents the trainable parameters. The first Wasserstein distance between the source and target distributions of

node representation $\mathbb{P}_{\boldsymbol{h}^s}$ and $\mathbb{P}_{\boldsymbol{h}^t}$ can be computed using the Kantorovich-Rubinstein duality [139]:

$$W_1(\mathbb{P}_{\boldsymbol{h}^s}, \mathbb{P}_{\boldsymbol{h}^t}) = \sup_{\|f_d\|_{L_c} \leqslant 1} \mathbb{E}_{\mathbb{P}_{\boldsymbol{h}^s}}[f_d(\boldsymbol{h}; \boldsymbol{\theta}_d)] - \mathbb{E}_{\mathbb{P}_{\boldsymbol{h}^t}}[f_d(\boldsymbol{h}; \boldsymbol{\theta}_d)], \tag{6.7}$$

where $\| f_d \|_{L_c} \leqslant 1$ is the Lipschitz continuity constraint. It can be interpreted as the minimum cost of transporting mass for transforming one distribution into another with the cost defined as the mass times the transport distance [4]. We can further approximate the empirical Wasserstein distance under the 1-Lipschitz assumption by maximizing the following domain critic loss with respect to $\boldsymbol{\theta}_d$:

$$\begin{aligned} \mathcal{L}_d = & \frac{1}{n^s} \sum_{i=1}^{n^s} f_d([f_g(A^s, X^s; \boldsymbol{\theta}_g)]_i; \boldsymbol{\theta}_d) \\ & - \frac{1}{n^t} \sum_{i=1}^{n^t} f_d([f_g(A^t, X^t, \boldsymbol{\theta}_g)]_i; \boldsymbol{\theta}_d). \end{aligned} \tag{6.8}$$

To enforce the Lipschitz constraint, we add a gradient penalty $\mathcal{L}_{grad}$ for the parameters $\boldsymbol{\theta}_d$ of the domain critic as suggested in [43]:

$$\mathcal{L}_{grad}(\hat{\boldsymbol{h}}) = (\| \nabla_{\hat{\boldsymbol{h}}} f_d(\hat{\boldsymbol{h}}; \boldsymbol{\theta}_d) \|_2 - 1)^2, \tag{6.9}$$

where the representation $\hat{\boldsymbol{h}}$ can be the source representations, the target representations, and the random points along the straight line between the source and target representation pairs. It can help avoid the capacity underuse and gradient vanishing/exploding problems of weight clipping methods [4] for 1-Lipschitz enforcement.

Hence, we solve the following minimax problem for learning network invariant node representations:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \{\mathcal{L}_d - \gamma \mathcal{L}_{grad}\}, \tag{6.10}$$

where γ is the gradient penalty coefficient, which should be set to 0 when optimizing the generator. The optimization problem suggests that the domain critic $f_d(\cdot)$

Algorithm 6.1: Training Algorithm of AdaGCN

Input : source data $\{G^s = (V^s, A^s, X^s), Y^{sl}\}$, target data $\{G^t = (V^t, A^t, X^t)\}$, domain critic training step n_d, coefficients γ, λ, learning rates α_1, α_2

1 Initialize parameters $\boldsymbol{\theta}_g$ for representation learner f_g, $\boldsymbol{\theta}_c$ for label classifier f_c, and $\boldsymbol{\theta}_d$ for domain critic f_d;

2 **while** *not converge* **do**

3 // Optimize domain critic

4 **for** $t = 1, \ldots, n_d$ **do**

5 $H_g^s \leftarrow f_g(A^s, X^s; \boldsymbol{\theta_g})$, $H_g^t \leftarrow f_g(A^t, X^t; \boldsymbol{\theta_g})$;

6 $N \leftarrow \min\{N^s, N^t\}$;

7 Construct $H = \{\boldsymbol{h}_i\}_{i=1}^N$ with $\boldsymbol{h}_i \leftarrow \varepsilon \boldsymbol{h}^s + (1-\varepsilon)\boldsymbol{h}^t$, where ϵ is a random number sampled from $U[0,1]$, $\boldsymbol{h}^s$ and $\boldsymbol{h}^t$ are sampled from H_g^s and H_g^t, respectively;

8 $\hat{H} \leftarrow \{H_g^s, H_g^t, H\}$;

9 $\boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d + \alpha_1 \cdot \nabla_{\boldsymbol{\theta}_d}\{\mathcal{L}_d - \gamma \mathcal{L}_{grad}(\hat{H})\}$;

10 // Optimize representation learner and label classifier

11 $\boldsymbol{\theta} \leftarrow \{\boldsymbol{\theta}_g, \boldsymbol{\theta}_c\}$;

12 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_2 \cdot \nabla_{\boldsymbol{\theta}}\{\mathcal{L}_c + \lambda \mathcal{L}_d\}$;

should be first trained to be optimal and then parameters in the generator $f_g(\cdot)$ are updated to minimize the Wasserstein distance between the source and target node representations.

Note that our proposed AdaGCN is very flexible, and some other adversarial-based domain adaptation methods [94, 75] can also be integrated into our framework.

6.3.5 Overall Loss and Model Training

The overall loss of the proposed model AdaGCN is as follows:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_c}\{\mathcal{L}_c + \lambda \max_{\boldsymbol{\theta}_d}[\mathcal{L}_d - \gamma \mathcal{L}_{grad}]\}, \tag{6.11}$$

where λ is the coefficient for balancing semi-supervised learning and domain adaptation. We summarize the training procedure for AdaGCN in Algorithm 6.1. Note that here we do a full-batch training with gradient descent, but some existing methods can be applied to train the model in a mini-batch manner [17, 18]. First, as presented in line 4-10, we optimize the parameters $\boldsymbol{\theta}_d$ of the domain critic $f_d(\cdot)$ via

gradient descent with other model parameters fixed. Then, as shown in lines 12 and 13, we fix $\boldsymbol{\theta}_d$, and update the parameters $\boldsymbol{\theta}_g$ of the generator $f_g(\cdot)$ and $\boldsymbol{\theta}_c$ of the classifier $f_c(\cdot)$ by minimizing the classification loss $\mathcal{L}_c$ and the domain adaptation loss $\mathcal{L}_d$ simultaneously. When the model converges, we can obtain class discriminative and domain invariant node representations. To classify nodes in the target network, one can simply feed the learned node representations to the trained classifier $f_c(\cdot)$.

The computational complexity of the model mainly consists of three parts, including the GCN layers (Eq. (6.2)), the label classifier (Eq. (6.4)) and the domain critic (Eq. (6.7)). It takes $\mathcal{O}((|E^s| + |E^t|)w_1w_2)$ (suppose that $W_g^{(k)} \in \mathbb{R}^{w_1 \times w_2}$, and E^s and E^t are the edge sets of the source and target networks) to compute hidden representations with single GCN layer for both the source and target networks through Eq. (6.2), which is linear to the number of edges. Note that the IGCN layer can ensure linearity with only an additional constant scale factor n_I added to the complexity through left multiplying $H_g^{(k)}$ by $\hat{A}$ repeatedly for n_I times in Eq. (6.4). Obviously, the time complexity of label classifier and domain critic is linear to the number of nodes. Thus, the overall complexity of the proposed methods are linear to the size of the networks.

6.4 Experiments

In this section, we aim to answer the following research questions (RQ) via experiments:

RQ1 How do the proposed methods perform compared with state-of-the-art methods?

RQ2 How do the training rates of the source and target networks, i.e., the ratio of labeled nodes in G^s and G^t, affect the transfer learning performance?

Table 6.2: Statistics of the real-world network datasets

Dataset	#Nodes	#Edges	#Attributes	#Union Attributes	#Labels
DBLPv7	5,484	8,130	4,412		
Citationv1	8,935	15,113	5,379	6,775	5
ACMv9	9,360	15,602	5,571		

RQ3 How does the distribution discrepancy between source and target networks affect the transfer learning results?

RQ4 How does the strength of graph convolution affect the domain adaptation performance?

RQ5 How do the hyper-parameters affect the performance of the proposed methods?

We also visualize the learned node embeddings from representation learner to provide an intuitive understanding of our proposed methods.

6.4.1 Experiment Setup

Datasets

We conduct experiments on three real-world attributed networks constructed by [114] based on datasets provided by ArnetMiner [128]. Some statistics of the experimental datasets are displayed in Table 6.2. DBLPv7, Citationv1 and ACMv9 are three paper citation networks from different original sources, i.e., DBLP, Microsoft Academic Graph and ACM respectively, and contain papers published in different periods, i.e., between years 2004 and 2008, before year 2008, and after year 2010, respectively. Here we consider them as undirected networks with each edge representing a citation relation between two papers. Each paper belongs to some of the following five categories according to its research topics, including "Databases", "Artificial Intelligence", "Computer Vision", "Information Security", and "Networking". Besides, the keywords extracted from the title of each paper were utilized as its attributes in

the form of bag-of-words vector. We evaluate our proposed methods by conducting multi-label classification on these three network domains through six transfer learning tasks including C→D, A→D, D→C, A→C, D→A, and C→A, where D, C, A denote DBLPv7, Citationv1 and ACMv9, respectively.

Baselines

We select baselines from several related research lines including single network embedding methods, graph-based semi-supervised learning methods, deep domain adaptation methods, and transfer learning methods for network data. The descriptions of them are listed as follows:

- **DeepWalk** [95], **node2vec** [41], **ANRL** [162]: They are single network embedding methods. Both DeepWalk and node2vec first transform network topology into node sequences, and then use skip-gram model to learn node representations. ANRL is a deep attributed network embedding model adapted from autoencoder, and we use its best variant ANRL-WAN.

- **GCN** [62], **GraphSAGE** [46]: They can be used for semi-supervised learning and representation learning. GCN is a deep convolutional network for graph-structured data, which integrates network topology, node attributes and observed labels into an end-to-end learning framework. GraphSAGE is a variant of GCN with different aggregation methods.

- **DNNs**, **WDGRL** [111]: These two deep models only utilize node attributes. DNNs is a multi-layer perceptron. WDGRL is a state-of-the-art adversarial domain adaptation method with the assumption of IID vector-based inputs in each domain.

- **NetTr** [33], **CDNE** [114]: These are transfer learning methods for network data. NetTr learns transferable representations based on network topology

only. CDNE is adapted from autoencoder by adding MMD loss across networks for domain adaptation.

We denote our proposed methods with regular GCN layers (Eq. (6.2)) as AdaGCN and improved GCN layers (Eq. (6.4)) as AdaIGCN.

Implementation Details

We implement our proposed methods using Tensorflow with Adam optimizer. For all transfer learning tasks, we use the same set of parameter configurations unless otherwise specified. We first describe the settings of AdaGCN. The GCNs of both the source and target networks contain three hidden layers with structure as 1000-100-16. The dropout rate for each GCN layer is set to 0.3. The classifier $f_c(\cdot)$ is a logistic regression model with sigmoid output layer for multi-label classification. The domain critic $f_d(\cdot)$ contains only one hidden layer with 16 units. A l_2-norm regularization term is imposed on model parameters except those of $f_d(\cdot)$ with the regularization coefficient as 5×10^{-5}. The domain adaptation coefficient λ, gradient penalty coefficient γ, and domain critic training step n_d are set to 1, 10 and 10, respectively. The learning rates for both components of our method are set to 1.5×10^{-3}. We train the model for 1000 epochs, and perform a learning rate decaying by multiplying a decaying factor 0.8 per 100 epochs after the first 500 epochs to stabilize training. For AdaIGCN, it has similar configurations as AdaGCN with the only difference in the representation learner, which consists of only one IGCN layer and two additional fully connected layers. n_I is set to 10 for all tasks. GCN and AdaGCN have the same settings for common hyper-parameters and model structure.

For single network embedding methods, including DeepWalk, node2vec and AN-RL, node representations are first learned and then a one-vs-rest logistic regression classifier is trained with labeled nodes of both networks. For fair comparison, the dimension of node representations for these methods are all set to 128. For Graph-

Table 6.3: Multi-label classification with source training rate as 10%

	S	T	DeepWalk	node2vec	ANRL	GraphSAGE	DNNs	WDGRL	NetTr	CDNE	GCN	AdaGCN	AdaIGCN
Mi-F1 (%)	C	D	24.67	21.73	49.37	67.42	28.71	29.05	48.32	70.80	67.00	**70.89**	**75.14**
	A	D	25.96	27.70	48.41	65.95	33.88	32.86	48.24	62.94	65.92	**69.32**	**74.85**
	D	C	31.05	21.78	40.47	59.23	16.36	21.63	44.47	71.34	63.47	**77.77**	**79.34**
	A	C	25.34	23.48	46.55	66.49	27.33	28.82	47.60	72.10	69.02	**78.83**	**78.97**
	D	A	27.85	26.24	39.99	53.08	9.64	16.90	42.42	66.79	53.32	**67.92**	**71.43**
	C	A	28.98	19.39	44.22	61.05	24.63	28.02	44.73	**71.02**	64.33	68.30	**74.48**
Ma-F1 (%)	C	D	22.02	15.32	43.19	62.03	28.14	29.27	42.63	68.62	64.28	**69.75**	**72.53**
	A	D	21.90	23.53	40.11	61.66	31.95	31.79	41.78	60.87	62.24	**68.46**	**72.29**
	D	C	25.14	16.50	34.13	52.13	16.61	21.33	39.66	70.47	59.76	**76.42**	**77.95**
	A	C	20.94	19.70	40.58	61.54	26.83	28.36	42.88	70.29	65.10	**77.45**	**77.53**
	D	A	25.86	20.90	33.32	45.85	10.11	17.17	36.17	65.72	50.12	**69.31**	**72.26**
	C	A	23.85	14.38	38.94	52.98	24.43	27.39	40.78	**70.16**	61.35	67.83	**75.04**

* S: Source, T: Target, Mi-F1: Micro-F1, Ma-F1: Macro-F1, D: DBLPv7, C: Citationv1, A: ACMv9. The top 2 classification f1-scores are highlighted in bold for each task.

SAGE, we adapt it to the transductive setting for better utilization of linkage information of the two networks, and use its best variant GraphSAGE-LSTM for comparison. Since these methods are designed for single network, we simply combine two networks into one and then conduct experiments as single network learning. DNNs have similar parameter settings with GCN, and WDGRL have similar parameter settings with AdaGCN. We have also tried to improve the input features of DNNs and WDGRL by augmenting the feature matrix of graph with the learned embedding vectors from DeepWalk, but found it deteriorates performance, which is explainable since the learned embeddings of the source and target networks from DeepWalk are not comparable. Experiments for NetTr and CDNE are conducted as suggested by the corresponding papers.

6.4.2 Performance Comparison (RQ1)

The training rate of a network is defined as $R_l = \frac{|V^l|}{|V|}$, where V^l represents the set of labeled nodes in the network. Different settings of R_l are constructed by randomly sampling V^l from V while ensuring nodes in V^l covering all labels. In this section, we

Table 6.4: Multi-label classification with source training rate as 10% and target training rate as 5%

	S	T	DeepWalk	node2vec	ANRL	GraphSAGE	DNNs	WDGRL	NetTr	CDNE	GCN	AdaGCN	AdaIGCN
Mi-F1 (%)	C	D	53.10	59.17	55.97	70.42	32.41	33.59	50.74	**73.69**	71.25	71.90	**75.25**
	A	D	48.02	57.67	52.55	69.13	40.18	30.73	48.31	69.61	70.29	**75.18**	**75.95**
	D	C	66.57	69.13	53.91	68.99	28.03	23.33	50.28	78.93	73.09	**79.66**	**80.33**
	A	C	61.56	66.91	54.42	71.92	34.60	32.73	49.98	77.86	75.28	**81.45**	**82.00**
	D	A	58.85	64.23	49.37	64.64	27.57	21.76	45.24	**77.38**	71.51	75.15	**78.18**
	C	A	57.58	62.60	51.39	69.20	35.17	33.43	46.26	**77.10**	72.84	74.51	**77.14**
Ma-F1 (%)	C	D	47.48	53.11	48.54	66.24	31.71	33.63	44.12	**71.96**	70.02	71.71	**73.66**
	A	D	42.60	51.25	44.01	65.13	38.14	30.41	42.03	66.73	68.29	**73.57**	**74.87**
	D	C	62.63	64.49	47.27	63.78	28.21	23.66	45.41	77.24	71.44	**77.92**	**78.18**
	A	C	56.44	62.20	48.34	67.77	34.22	32.63	45.37	75.90	73.16	**79.44**	**80.09**
	D	A	58.92	64.43	43.91	64.00	27.90	21.65	41.09	**77.22**	71.69	75.66	**78.65**
	C	A	56.85	63.10	46.91	67.60	35.07	33.11	42.83	**77.44**	73.13	74.70	**76.90**

* S: Source, T: Target, Mi-F1: Micro-F1, Ma-F1: Macro-F1, D: DBLPv7, C: Citationv1, A: ACMv9. The top 2 classification f1-scores are highlighted in bold for each task.

conduct multi-label classification on three datasets with six transfer learning tasks. We consider two settings, including an unsupervised setting where only the source network is partially labeled, and a semi-supervised setting where both the source and target networks are partially labeled.

Unsupervised Setting: Partially Labeled Source Network and Completely Unlabeled Target Network

In the unsupervised setting, we conduct experiments with the source training rate as 10% while the target network is completely unlabeled. The experimental results are shown in Table 6.3. It can be easily observed that our proposed method AdaGCN outperforms all the baselines in five out of six tasks, and has comparable results with the best baseline CDNE on the sixth task Citationv1→ACMv9. It demonstrates the effectiveness of our proposed AdaGCN model for cross-network node classification. Specifically, there is a 4.41% relative performance improvement in Micro-F1 score and a 5.81% in Macro-F1 score over the best baseline CDNE on average across all transfer tasks. AdaIGCN can further improve AdaGCN, and outperforms all the baselines consistently in all learning tasks.

GCN and GraphSAGE have comparable performance. The proposed AdaGCN method adapts GCN for cross-network learning by combining it with domain adaptation technique. It achieves significant 13.54% and 19.03% relative gains in Micro-F1 and Macro-F1 scores respectively over GCN, which suggests that the adversarial domain adaptation component can effectively mitigate the distribution divergence of two domains and enables a successful knowledge transfer. The proposed AdaIGCN model further achieves significant 4.83% and 4.28% relative improvements in Micro-F1 and Macro-F1 scores respectively over AdaGCN on average, which shows that IGCN can learn better node representations to facilitate knowledge transfer.

We noticed that both DeepWalk and node2vec have poor performance in all transfer learning tasks as shown in Table 6.3. The reason is that node representations used for multi-label classification are trained independently for the source and target networks since no connections between them exist. This makes the learned representations incomparable across networks, and thus the learned classifier based on source labeled data can not generalize to the target domain. Similar observations have also been made in [50]. Therefore, single network embedding methods with only network topology as input are not directly suitable for multi-network learning. ANRL, as an attributed network embedding method, has much better performance compared with DeepWalk and node2vec, which benefits from the shared node attributes between the source and target networks. However, it is inferior to GCN by a large margin, not to mention the proposed AdaGCN method. The reasons lie in two aspects: firstly, ANRL is an unsupervised embedding method, so node classification can only be conducted after node representations have been learned, while GCN can perform semi-supervised learning in an end-to-end manner; secondly, ANRL suffers from the distribution shift between the source and target domains, while AdaGCN addresses this issue by introducing an adversarial domain adaptation component.

Both DNNs and WDGRL cannot leverage network topology information. It can

be observed that the performances of DNNs and WDGRL are poor, although more available labeled nodes can help improve their performances. Besides, we noticed that WDGRL performs worse than DNNs in some tasks, which means that the domain adaptation component of WDGRL results in negative transfer. The reason might be that the distribution divergence between node attributes of two domains are too large for the adversarial domain adaptation method to work. Overall, it suggests that existing domain adaptation methods can not handle cross-network node classification problem due to their inability in leveraging network structure information. In contrast, our proposed AdaGCN method jointly models network structures and node attributes with graph convolution. The Laplacian smoothing on node features with graph convolution in the representation learner enables an easy knowledge transfer across networks.

NetTr and CDNE are two transfer learning methods for cross-network node classification. Our methods outperform NetTr by a large margin. Specifically, AdaGCN achieves remarkable 57.28% and 76.46% relative improvements over NetTr in Micro-F1 and Macro-F1 scores, respectively. One important reason is that NetTr learns transferable representations based on network topology only. Our proposed methods also produce a significant improvement over CDNE on average as mentioned before. Particularly, the relative performance gain of AdaGCN over CDNE reaches the desirable 12.47% and 10.14% in Micro-F1 and Macro-F1 scores respectively on ACMv9→DBLPv7. The advantages of our methods over CDNE can be summarized into two aspects: firstly, graph convolution enables a natural combination of node attributes and network structures for representation learning, while CDNE only leverages network structures to extract features; secondly, the adversarial domain adaptation method is shown to be more effective compared with MMD in the literature [111].

Semi-Supervised Setting: Partially Labeled Source and Target Networks

In the semi-supervised setting, both the source and target networks are partially labeled with the training rates as 10% and 5%, respectively. The experimental results are shown in Table 6.4.

Due to the additional available labeled data in the target network, all models achieve better classification performance compared with the unsupervised setting as shown in Tables 6.3 and 6.4. There are many similar findings in both the unsupervised and semi-supervised scenarios, and we only highlight some new insights. Firstly, both DeepWalk and node2vec perform significantly better even though only 5% additional labeled nodes in the target network are available. It shows the effectiveness of the learned node embeddings in the target network. Both GraphSAGE and GCN have better results compared with DeepWalk and node2vec because of the proper utilization of both node attributes and network topology in learning tasks and a certain level of knowledge transfer due to the shared weights in the representation learner. AdaGCN consistently outperforms GCN across all learning tasks by a large margin, which can be attributed to the successful knowledge transfer from the source to the target network thanks to the domain adaptation component. Similarly, AdaIGCN further improves over AdaGCN with 2.40% and 2.04% relative gains in Micro-F1 and Macro-F1 scores respectively because of the improved GCN layer for alleviating overfitting. It also produces 3.14% and 3.55% relative improvements in Micro-F1 and Macro-F1 scores respectively over the best baseline CDNE.

Overall, the empirical results demonstrate that our proposed methods achieve state-of-the-art cross-network node classification performance in both the unsupervised and semi-supervised settings, thus can be potentially applied to a wide range of scenarios.

6.4.3 Effect of Training Rate (RQ2)

In this section, we study the effect of training rate R_l of the source and target networks on model performance.

Effect of Source Training Rate

We conduct experiments with the training rate of source network ranging from 5% to 90% while the target network is completely unlabeled. The experimental results are displayed in Figure 6.3. Note that only some of the baselines are selected for comparison to ensure clear presentations, and only the results on tasks with DBLPv7 and Citationv1 as targets are presented here to avioid repetition. We have the following observations:

- Our proposed methods, including AdaGCN and AdaIGCN, consistently outperform all the baselines on these four tasks for all training rates, which demonstrates their effectiveness for knowledge transfer across networks. AdaIGCN performs better than AdaGCN, especially when the source training rate is low. It validates that the utilization of IGCN layer can help alleviate the overfitting issue and facilitate knowledge transfer.

- For almost all baselines except DeepWalk, the performance first improves, and then becomes stable with the increase of source training rate. For our proposed AdaIGCN, it shows remarkably good performance even with only 5% labeled nodes in the source network, which suggests its high label efficiency.

- We noticed that the performance of DeepWalk decreases as the source training rate increases. It actually further confirms our finding that single network embedding methods based on topology only are not applicable for cross-network learning due to the incomparable node representations for two networks. Similar results can also be observed for node2vec which are not shown here.

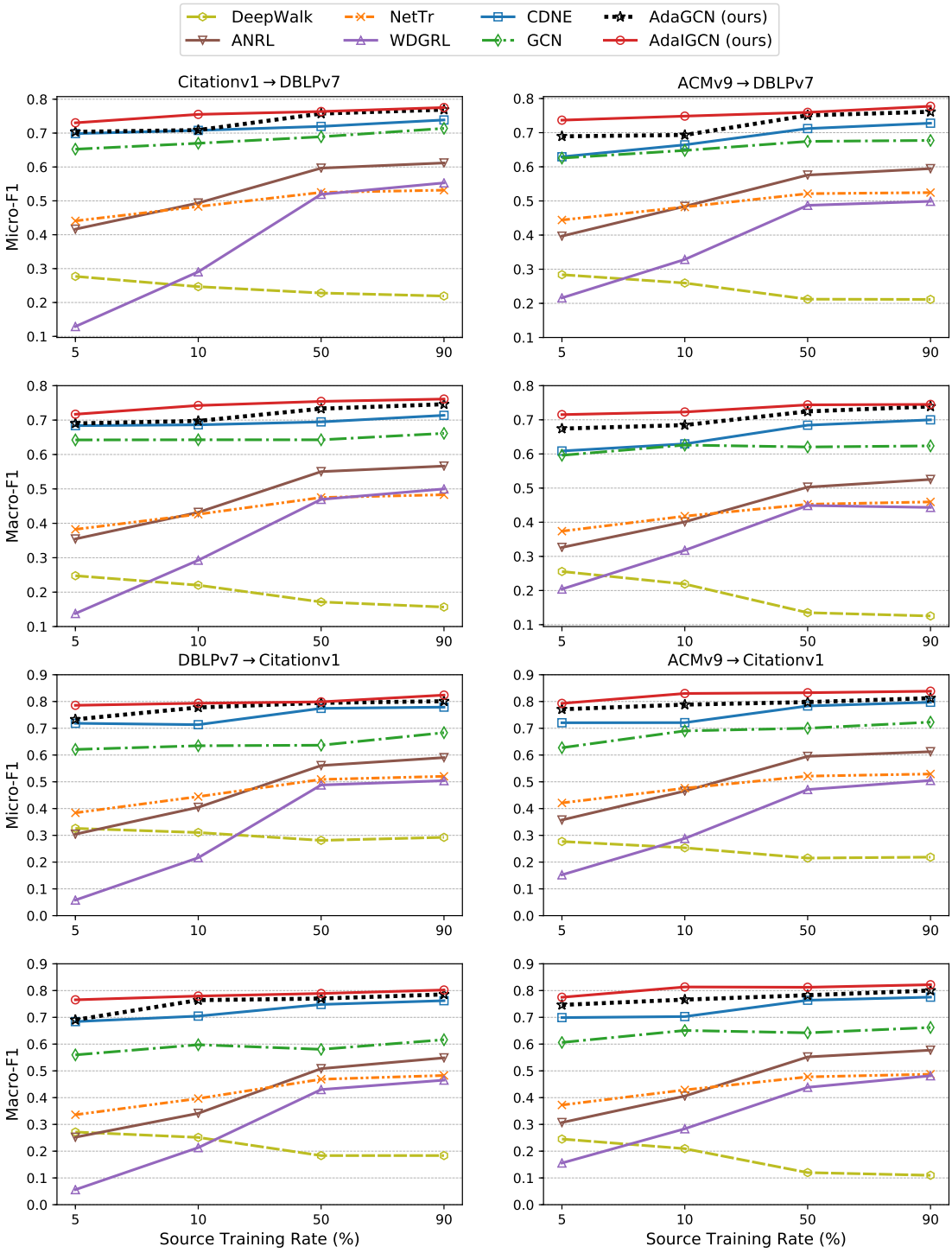


Figure 6.3: Multi-label classification with varying source training rates.

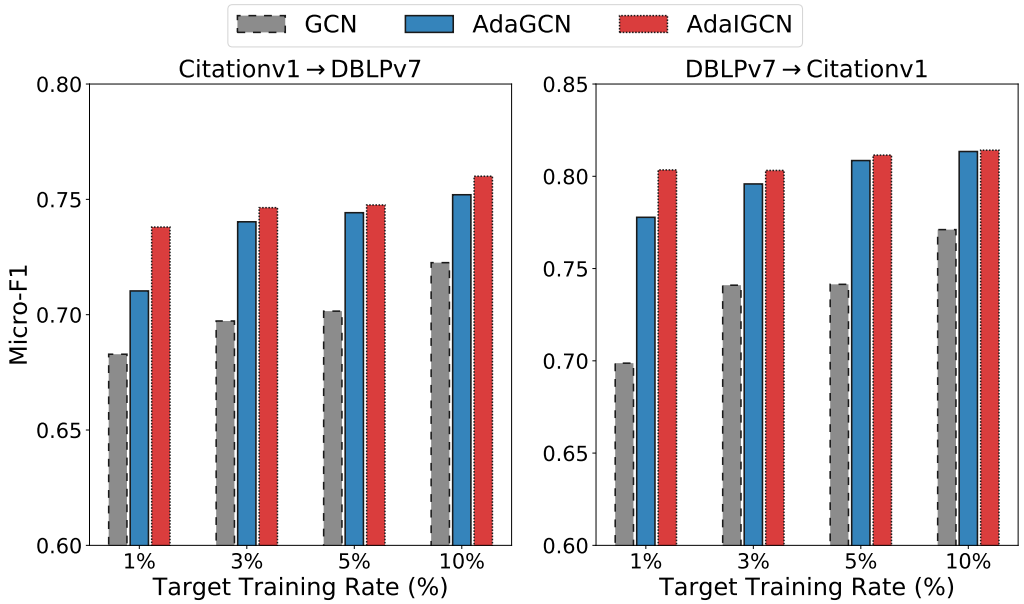


Figure 6.4: Multi-label classification with varying target training rates.

Effect of Target Training Rate

We investigate the effect of target training rate by varying it from 1% to 10% while fixing source training rate as 10%. We only show the Micro-F1 scores in Figure 6.4 on learning tasks Citationv1→DBLPv7 and DBLPv7→Citationv1 for succinct presentation. We have the following observations. Firstly, AdaGCN significantly and consistently outperforms GCN on both learning tasks for all target training rates, which means that the adversarial domain adaptation component can successfully mitigate the distribution discrepancy between two domains and help knowledge transfer across networks. Specifically, AdaGCN exhibits an impressive 5.08% relative improvement over GCN on average. Secondly, AdaIGCN further achieves improvements upon AdaGCN consistently, and the gap is more significant with low target training rate. In particular, it produces a 3.90% relative gain over AdaGCN on Citationv1→DBLPv7 when the target training rate is 1%. It proves that the improved GCN layer can make a good balance between the strength of Laplacian smoothing and model complexity. Overall, it demonstrates the effectiveness of our

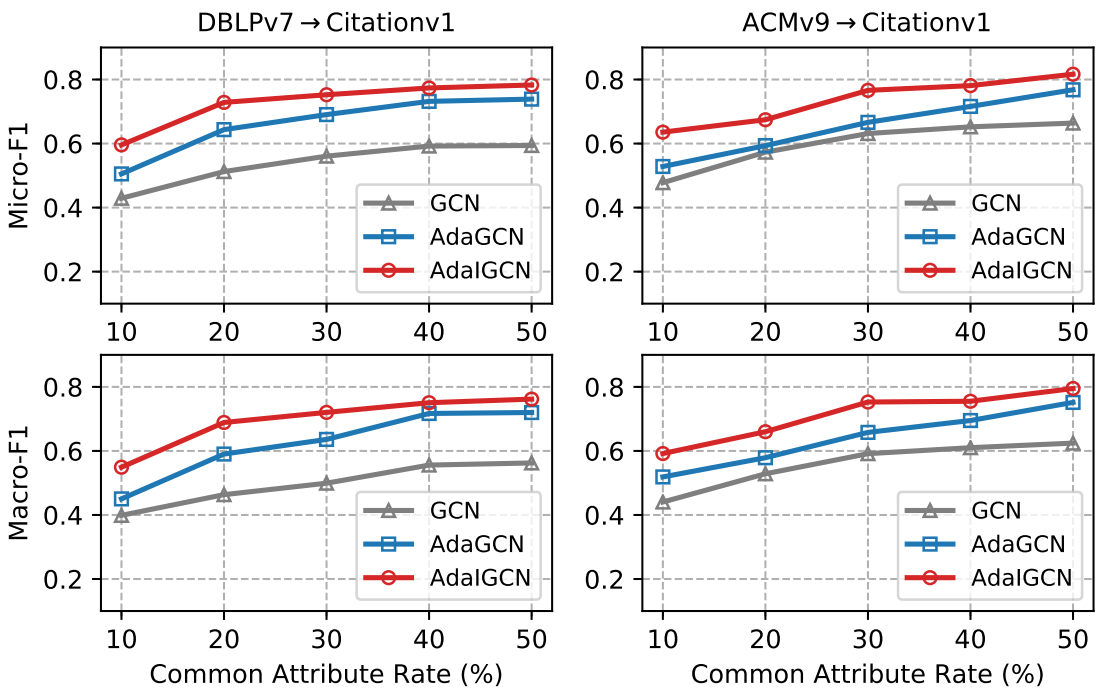


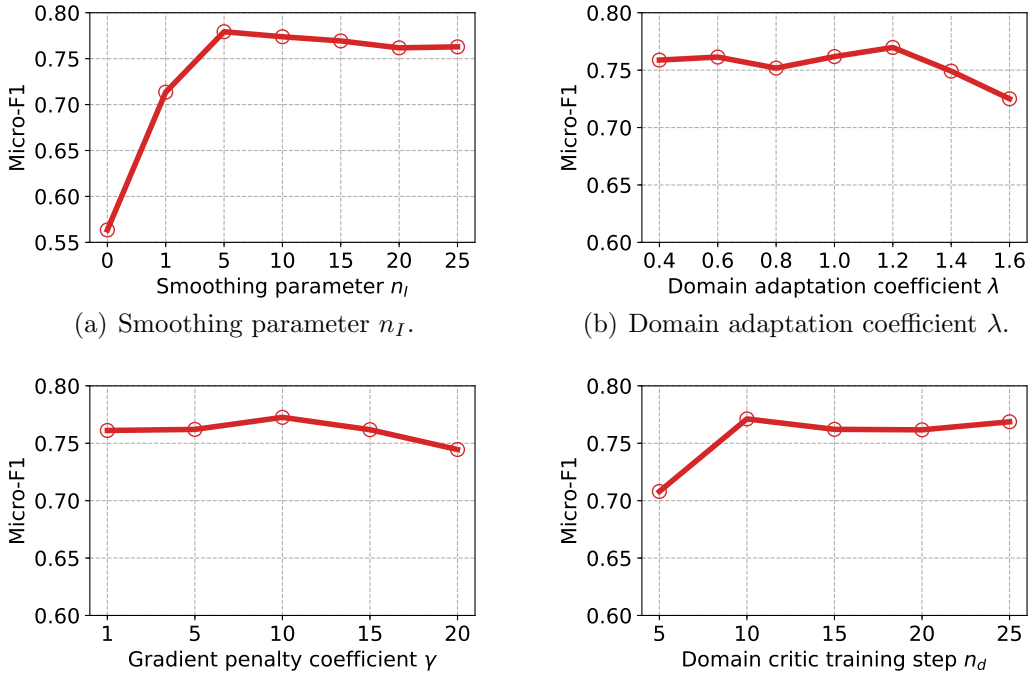
Figure 6.5: Multi-label classification on Citationv1 with varying common attribute rates of the source and target networks.

proposed methods for network transfer learning in the semi-supervised setting.

6.4.4 Effect of Distribution Discrepancy (RQ3)

In this section, we explore the effect of distribution discrepancy between the source and target networks on domain adaptation. We define common attribute rate between the source and target networks as $R_a = \frac{|\mathcal{X}^s \cap \mathcal{X}^t|}{|\mathcal{X}^s \cup \mathcal{X}^t|}$, and we randomly delete some of the common attributes of two networks to change R_a in the experiments. Lower R_a means larger distribution discrepancy. We conduct multi-label classification on tasks with Citationv1 as target with varying R_a in the unsupervised setting where the source training rate is 10%, and compare the performance of GCN, AdaGCN and AdaIGCN.

Figure 6.5 displays the experimental results when R_a ranges from 10% to 50%. Both AdaGCN and AdaIGCN consistently outperform GCN across all common at-



(a) Smoothing parameter n_I.

(b) Domain adaptation coefficient λ.

(c) Gradient penalty coefficient γ.

(d) Domain critic training step n_d.

Figure 6.6: Impact of hyper-parameters.

tribute rates for both transfer tasks. More specifically, AdaGCN achieves 22.88% and 24.93% relative gains on Micro-F1 and Macro-F1 scores respectively over GCN for DBLPv7→Citationv1, and 9.02% and 14.56% for ACMv9→Citationv1. It demonstrates that the adversarial domain adaptation component contributes to the classification performance even when the source and target networks only share a very small proportion of attributes. Besides, AdaIGCN performs better than AdaGCN consistently, which further confirms that the IGCN layer can learn better node representations for domain adaptation. In summary, the proposed methods are very robust and can work well under large distribution discrepancy between the source and target networks, which enables their applications for solving a wide range of real-world problems.

6.4.5 Effect of Graph Convolution (RQ4)

In this section, we vary the smoothing parameter n_I of the IGCN layer in AdaIGCN from 1 to 25 to study the effect of graph convolution on domain adaptation. Note that AdaIGCN can be reduced to WDGRL when $n_I = 0$, i.e., no smoothing on node features. The experiments are conducted in the unsupervised setting with source training rate as 10%. We present the experimental results on DBLPv7→Citationv1 in Figure 6.6(a). We can observe that graph convolution on node features brings extraordinary improvements to node classification performance on the target network, since there is a remarkable 26.62% relative improvement when increasing n_I from 0 to 1. When varying n_I from 1 to 25, the classification accuracy first increases and then slightly drops. It shows that appropriate setting of n_I can help further facilitate knowledge transfer, but too large n_I can result in over-smoothing of node features and thus harm the transfer performance. Specifically, features of neighborhood nodes become similar with Laplacian smoothing in the graph convolutional layer, and a large smoothing parameter can make them converge to very similar value and blur the class boundaries. On the whole, graph convolution plays a crucial role for the successful knowledge transfer across networks in our proposed framework.

6.4.6 Parameter Sensitivity (RQ5)

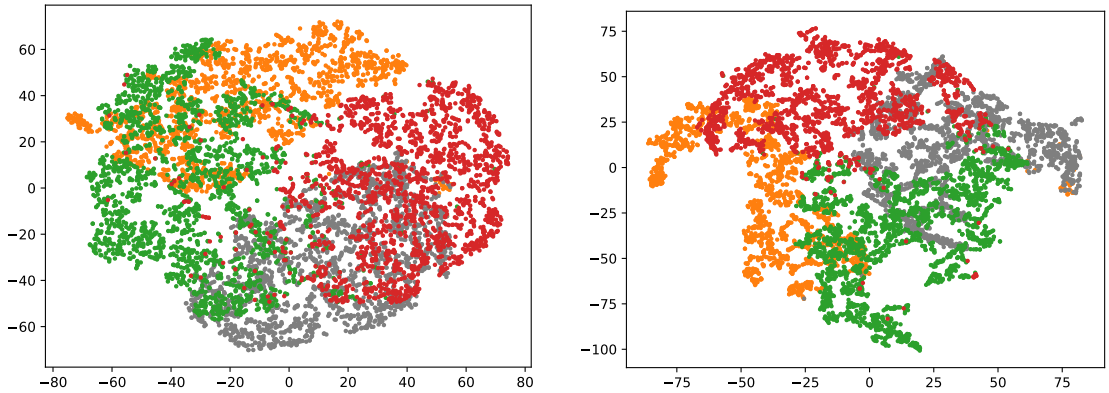
In this section, we perform sensitivity analysis of AdaGCN on domain adaptation coefficient λ, gradient penalty coefficient γ, and domain critic training step n_d. The experiments are conducted in the unsupervised setting with source training rate as 10%. It is expected to shed some lights on how to configure these hyper-parameters. Here we only present the Micro-F1 score for Citationv1→DBLPv7 to avoid repetition, and similar tendency can be observed in other tasks. Note that when studying one hyper-parameter, we fix all others with default settings mentioned in Section 6.4.1.

λ is a coefficient for balancing the semi-supervised loss and domain adaptation loss. We can find that the performance slightly improves with the increase of λ from 0.4 to 1.2, and then drops quickly afterwards as shown in Figure 6.6(b). It suggests that it is important to maintain the balance between the two parts so as to learn both class discriminative and domain invariant representations. γ is a hyper-parameter for controlling the weight of gradient penalty when training the discriminator of the adversarial domain adaptation component. From Figure 6.6(c), it can be observed that the best result is obtained when γ is set to 10, and smaller or larger configurations might result in performance degradation, which is consistent with the finding in [43], and thus 10 would be a recommended setting. Theoretically, the domain critic network $f_d(\cdot)$ should be trained to optimality by optimizing its own parameters while fixing those of other components, and thus the training step n_d should be set to a large enough number for this purpose. From Figure 6.6(d), it can be noticed that the Micro-F1 score shows apparent increase when increasing n_d from 5 to 10, and then becomes stable, which is consistent with our theoretic analysis.

6.4.7 Visualization of Node Representations

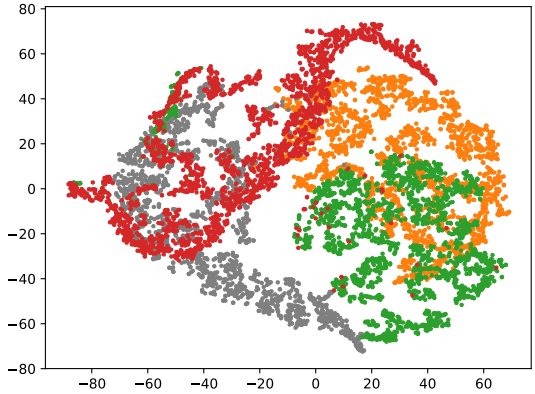
Figure 6.7 visualizes the node representations generated by GCN, IGCN, AdaGCN and AdaIGCN in the unsupervised setting for ACMv9→Citationv1 using t-SNE [137] where the source network is fully labeled. We only visualize nodes from "Databases" and "Computer Vision" for clear presentation. The gray and orange points represent papers of "Databases" and "Computer Vision" respectively from ACMv9, while red and green points represent papers of "Databases" and "Computer Vision" from Citationv1.

On one hand, the domain adaptation component helps mitigate domain divergence and benefits knowledge transfer. Specifically, from Figures 6.7(a) and 6.7(b),

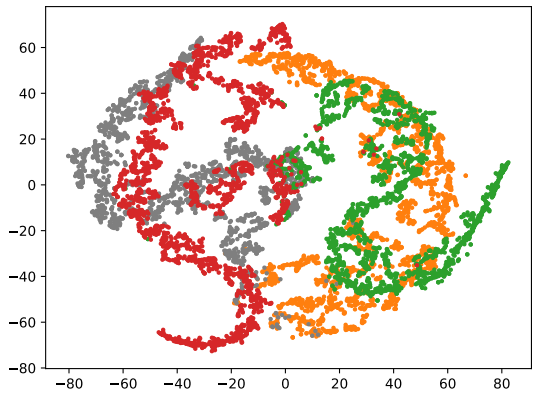


(a) GCN.

(b) IGCN.



(c) AdaGCN.



(d) AdaIGCN.

Figure 6.7: Visualization of the learned node representations from ACMv9→Citationv1. Each point represents one paper. Gray and orange points are from the source network, and red and green points are from the target network. Gray and red: "Databases". Orange and green: "Computer Vision". These plots are best viewed in color.

it can be observed that both the GCN and IGCN models suffer from distribution shift between different networks, since nodes from different categories, e.g., green and gray points, are mixed together. In contrast, from Figures 6.7(c) and 6.7(d), we can find that gray and red points are clustered together, while orange and green nodes are clustered together. It demonstrates that the adversarial domain adaptation successfully mitigates the distribution divergence between the source and target networks, since papers from the same categories of both domains are well clustered

together. Besides, the boundary between these two clusters are quite clear, which means that the learned node representations are discriminative. On the other hand, the IGCN layer also brings two significant advantages. Firstly, the IGCN layer allows adjusting the smoothing strength on node features without increasing model complexity, and an appropriate smoothing of node features helps to learn more compact node representations within the same category as shown in Figures 6.7(b) and 6.7(d), thus contributes to the classification task. Furthermore, it makes the domain adaptation process easier, which is confirmed by the visualization results that AdaIGCN aligns the source and target node representations better than AdaGCN as shown in Figures 6.7(c) and 6.7(d).

6.5 Related Work

6.5.1 Single Network Learning

Network embedding is aimed at learning compact node representations based on network topology only or with side information in an unsupervised manner to facilitate a range of learning tasks, such as node classification and network visualization. For topology-only embedding methods, most of existing works focused on preserving network structures and properties in embedding vectors through various techniques such as negative sampling approach [95, 126, 41], matrix factorization technique [12, 147] and deep learning models [13, 143, 112, 113]. Most recently, regularization methods based on generative adversarial networks or adversarial training are exploited to handle noisy and incomplete network data to improve generalization ability [23, 91, **?**]. Aside from topology-only methods, many models are proposed to incorporate side information such as node attributes [162, 91, 150]. For example, ANRL [162] optimizes both network structure preserving loss and feature reconstruction loss based on stacked autoencoder.

The unsupervised learning methods don't specially tailor the latent vectors for node classification, which makes them inferior to some customized models. Semi-supervised learning methods, including those using network topology and observed labels [131] and those combining network structures with available labels and node attributes [92, 154, 62, 54, 46, 138, 71], achieve state-of-the-art performance. Planetoid [154] optimizes a supervised loss and a context-preserving loss. GCN [62] is a deep convolutional learning paradigm for graph-structured data which nicely integrates local node attributes and graph topology in convolutional layers. GraphSAGE [46] is a variant of GCN which designs different aggregation methods for feature extraction. GAT [138] improves GCN by leveraging attention mechanism to aggregate features from the neighbors of a node with discrimination.

While these methods can be modified to cross-network learning, the distribution drift between different network domains severely hampers knowledge transfer, especially for the topology-only methods [50].

6.5.2 Multi-Network Learning

A branch of work aims to leverage the relationship between multiple networks to facilitate learning, including those relying on inter-network edges [149, 86], those focusing on identifying common nodes across networks [73, 51], and those managing to transfer knowledge from the source network(s) to the target network(s) [127, 33, 115, 116, 114].

Both EOE [149] and DMNE [86] learn embeddings for multiple networks simultaneously. Specifically, EOE introduces a harmonious embedding matrix to model inter-network node similarities, while DMNE adapts autoencoder for multi-network embedding with a co-regularized loss to model cross-network relationships. However, these methods heavily rely on the existence of cross-network connections, which makes them inapplicable for our problem. Another line of related research is network

alignment [73, 51], which aims to identify the node correspondence across networks with/without cross-network edges. It differs from our problem in the assumption of common nodes across networks and the research goal of finding common nodes across networks.

There is also some literature focusing on transferring knowledge from the source network(s) to the target network(s) for various tasks, such as social ties inference [127], positive/negative link prediction [155], and node classification [33, 64]. In this work, we aim to utilize knowledge in the source network to assist classification in the target network as [33, 64, 114]. In [33], non-negative matrix factorization is jointly applied on the label propagation matrices of both the source and target networks so as to learn transferable structure features. However, it suffers from expensive computation in the matrix decomposition process, and it cannot jointly model the relationships among structural information, node attributes and node labels, which might cause negative transfer. CDNE [114] is closely related to our work. It first learns node embeddings for multiple networks with different stacked autoencoders and mitigates the distribution shift of node representations between networks by minimizing the MMD loss, and then trains a node classifier with the learned node representations.

6.5.3 Domain Adaptation

Domain adaptation is a subtopic of transfer learning, which aims to mitigate the harmful effect of domain drift when transferring knowledge from source to target [93, 145]. Approaches for domain adaptation can be classified into three groups, including the instance-based methods [124], parameter-based methods [103], and feature-based methods [74, 120]. Among them, deep feature-based domain adaptation methods have attracted a lot of attention in recent years due to its effectiveness. They can be categoried into three branches, i.e., discrepancy-based methods [135, 74], reconstruction-based methods [166, 59], and adversarial-based method-

s [35, 134, 111, 94, 75].

In this chapter, we are interested in adversarial-based methods. They are motivated by the theory in [7, 6], which suggests that when an algorithm cannot learn to identify the domain of given representations, such representations are good for knowledge transfer. DANN [35] was proposed to learn domain invariant features by formulating the problem as a minimax game similar to GANs [39] with a feature extractor acting as the generator and a domain classifier acting as the discriminator. Further, WDGRL [111] exploits the Wasserstein distance to improve the loss of DANN, which equips the model with better gradient property and more promising generalization bound. Meanwhile, MADA [94] and CDAN [75] manage to leverage discriminative information from label classifier to facilitate the alignment of multimodal distributions from different domains. In this chapter, we leverage these adversarial-based techniques for domain adaptation on graph-structured data. The main difference is that the majority of previous methods are proposed for vector-based data such as image and text with the assumption of IID samples within each domain, while here we aim to explore domain adaptation for graph-structured data that has complicated correlations among data entities.

6.6 Conclusion

In this chapter, we successfully address the cross-network node classification problem by proposing a novel network transfer learning framework AdaGCN, which leverages the techniques of adversarial domain adaptation and graph convolution. It can learn both class discriminative and network invariant node representations with the help of a semi-supervised learning (SSL) component and an adversarial domain adaptation (ADA) component. The SSL component is capable of learning a well-generalized node classifier with graph convolutional layers for representation learning, while the

ADA component ensures successful knowledge transfer from the source network to the target network through adversarial learning. Together they enable AdaGCN to work well in real-world attributed networks under a realistic setting.

The research of transfer learning on network data is still in an early stage, and much more efforts are needed. This work serves as a step further in this direction. Future work will include investigating knowledge transfer from multiple source networks to a target network and exploring conditional adversarial domain adaptation for better alignment of multimodal data distribution.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we present effective representation learning methods for both plain networks and attributed networks with the assistance of techniques of adversarial learning principle [39] and adversarial training methods [40]. In particular, several frameworks and algorithms are developed with the descriptions as follows:

- Firstly, an Adversarial Network Embedding (ANE) framework, as the first network embedding method based on GANs, is introduced for robust representation learning. It learns structure-preserving node embeddings with an inductive DeepWalk and imposes a prior distribution on the embeddings through adversarial learning to enhance model robustness.

- Secondly, Adversarial Training (AdvT) methods with adaptive L_2 norm constraints on adversarial perturbations are designed for network embedding, which can improve both model robustness and generalization performance. It can be directly applied to a series of network embedding methods based on negative sampling technique, such as DeepWalk, LINE and node2vec.

- Third, a Ranking Network Embedding (RNE) framework is presented for p-reserving node similarity rankings in the embedding vectors. Two sampling

strategies are explored, including a vanilla strategy based on uniform sampling method and an adversarial strategy based on adversarial learning. The latter can generate more relevant and difficult negative nodes for given positive target-context node pairs instead of totally unrelated ones, which can help better capture structural information of networks.

- Finally, a network transfer learning framework AdaGCN is proposed to tackle a challenging cross-network node classification problem that a partially labeled attributed source network is leveraged to assist in node classification in a completely unlabeled or partially labeled target network. It combines the techniques of adversarial domain adaptation and graph convolution naturally in a framework, which enables the learning of both class discriminative and network invariant node representations. Empirical experiments demonstrate that AdaGCN has obvious superiority over state-of-the-art baseline models on node classification in the target network.

7.2 Future Work

The ANE framework leverages adversarial learning for imposing a prior distribution on embedding vectors to improve model robustness for plain network embedding. Many possible extensions can be explored:

- First, ANE designs a regularization method based on generative adversarial networks (GANs). However, GANs suffer from the non-convergence problem such as mode-collapse [104, 4], which makes ANE difficult for training too. It would be a promising direction to tackle this difficulty. Actually, there exists follow-up work focusing on this problem [158] by leveraging improved GANs with Wasserstein distance for quantifying distribution distance. More efforts might be in need.

- Second, ANE is designed for plain network embedding. How to apply similar regularization method for attributed network embedding is worth studying. One follow-up work [91] combines adversarial learning regularization and variational graph autoencoder to achieve this purpose. Further research in this direction might be needed.

- Third, in ANE framework, a preprocessing of adjacency matrix with PPMI matrix is needed to boost the performance, which suffers from the expensive computation in matrix multiplication. It would be a good idea to design more efficient method with adversarial learning regularization. Directly utilizing the original DeepWalk as the base model instead of the inductive DeepWalk might be a feasible way.

The AdvT regularization method is also introduced for plain network embedding. Possible directions for future work are as follows:

- The AdvT method is designed for network embedding methods with embedding vectors as model parameters. How to apply it to the parameterized network embedding methods such as deep learning embedding models would be a promising direction.

- Graph neural networks (GNN) such as GCN [62] and GAT [138] is widely used for modeling attributed networks for various important applications. Designing adversarial training method for GNNs would be a promising and challenging direction, which has actually attracted some research efforts [34, 27, 27]. This problem is still under-explored.

The research of transfer learning on network data is still in an early stage, and much more efforts are needed. The proposed AdaGCN serves as a step further in this direction. Future work will include the following aspects:

- In AdaGCN, the adversarial domain adaptation component only aligns the source and target node representations with a single domain discriminator while neglecting the complex multimode structures. Thus, it may produce inferior results which mixes nodes from different categories together. To solve this problem, it is necessary to leverage the discriminative information from label classifier for distribution alignment. Existing methods in domain adaptation for vector-based inputs [75, 94] are inapplicable for graph-structured data directly. More research efforts are needed.

- AdaGCN is designed to tackle cross-network learning setting with only a source network and a target network. However, there can be far more than two similar or related networks available for utilization in real-world applications. Investigating knowledge transfer from multiple source networks to a target network would be an interesting and promising research problem.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283. USENIX Association, 2016.

[2] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[3] Amr Ahmed, Nino Shervashidze, Shravan M. Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *WWW*, pages 37–48, 2013.

[4] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[5] Mohammadreza Armandpour, Patrick Ding, Jianhua Huang, and Xia Hu. Robust negative sampling for network embedding. In *AAAI*, pages 3191–3198, 2019.

[6] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[7] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144. MIT Press, 2006.

[8] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. 2011.

[9] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[10] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[11] Liwei Cai and William Yang Wang. KBGAN: adversarial learning for knowledge graph embeddings. *CoRR*, abs/1711.04071, 2017.

[12] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.

[13] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016.

[14] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. HARP: hierarchical representation learning for networks. In *AAAI*, pages 2127–2134, 2018.

[15] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. Fast and accurate network embeddings via very sparse random projection. In *CIKM*, 2019.

[16] Haochen Chen, Xiaofei Sun, Yingtao Tian, Bryan Perozzi, Muhao Chen, and Steven Skiena. Enhanced network embeddings via exploiting edge labels. In *CIKM*, pages 1579–1582, 2018.

[17] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, volume 80, pages 941–949, 2018.

[18] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

[19] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *CoRR*, abs/1809.02797, 2018.

[20] T. F. Cox and M. A. Cox, editors. *Multidimensional Scaling*. CRC Press, 2000.

[21] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems*, 80:56–71, 2015.

[22] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 1123–1132, 2018.

[23] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. Adversarial network embedding. In *AAAI*, 2018.

[24] Quanyu Dai, Qiang Li, Liang Zhang, and Dan Wang. Ranking network embedding via adversarial learning. In *PAKDD*, 2019.

[25] Quanyu Dai, Xiao Shen, Xiao-Ming Wu, and Dan Wang. Network transfer learning via adversarial domain adaptation with graph convolution. *arXiv:1909.01541*, 2019.

[26] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. Adversarial training methods for network embedding. In *WWW*, pages 329–339, 2019.

[27] Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. 2019.

[28] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *SDM*, pages 594–602, 2019.

[29] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017.

[30] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *ICLR*, 2017.

[31] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[32] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *WWW*, pages 417–426, 2019.

[33] Meng Fang, Jie Yin, and Xingquan Zhu. Transfer learning across networks for collective classification. In *ICDM*, pages 161–170, 2013.

[34] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *arXiv:1902.08226*, 2019.

[35] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17:59:1–59:35, 2016.

[36] Hongchang Gao and Heng Huang. Self-paced network embedding. In *KDD*, pages 1406–1415, 2018.

[37] Alberto Garcia Duran and Mathias Niepert. Learning graph representations with embedding propagation. In *NIPS*, pages 5125–5136, 2017.

[38] John Glover. Modeling documents with generative adversarial networks. In *Workshop on Adversarial Training, NIPS*, 2016.

[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[40] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[41] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[42] Yupeng Gu, Yizhou Sun, Yanen Li, and Yang Yang. Rare: Social rank regulated large-scale network embedding. In *WWW*, pages 359–368. ACM, 2018.

[43] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[44] Zhijiang Guo, Yan Zhang, and Wei Lu. Attention guided graph convolutional networks for relation extraction. In *ACL*, pages 241–251, 2019.

[45] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *JMLR*, 13:307–361, 2012.

[46] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[47] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

[48] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *CIKM*, pages 623–632, 2015.

[49] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for recommendation. In *SIGIR*, pages 355–364. ACM, 2018.

[50] Mark Heimann and Danai Koutra. On generalizing neural node embedding methods to multi-network problems. In *KDD MLG Workshop*, 2017.

[51] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. REGAL: representation learning-based graph alignment. In *CIKM*, pages 117–126, 2018.

[52] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313:504–507, July 2006.

[53] Yifan Hou, Hongzhi Chen, Changji Li, James Cheng, and Ming-Chang Yang. A representation learning framework for property graphs. In *KDD*, pages 65–73, 2019.

[54] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *WSDM*, pages 731–739, 2017.

[55] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[56] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Conditional network embeddings. In *ICLR*, 2019.

[57] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

[58] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, 2019.

[59] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, pages 1857–1865, 2017.

[60] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[61] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

[62] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[63] Yi-An Lai, Chin-Chi Hsu, Wen-Hao Chen, Mi-Yen Yeh, and Shou-De Lin. PRUNE: preserving proximity and global ranking for network embedding. In *NIPS*, pages 5257–5266, 2017.

[64] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. Transfer learning for deep learning on graph-structured data. In *AAAI*, pages 2154–2160, 2017.

[65] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1(1):2, 2007.

[66] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[67] Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. Reducing the sampling complexity of topic models. In *KDD*, pages 891–900, 2014.

[68] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545, 2018.

[69] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label efficient semi-supervised learning via graph filtering. In *CVPR*, 2019.

[70] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation recognition with graph neural networks. In *ICCV*, pages 4183–4192, 2017.

[71] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy. Semi-supervised embedding in attributed networks with outliers. In *SDM*, 2018.

[72] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[73] Li Liu, William K. Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *IJCAI*, pages 1774–1780, 2016.

[74] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015.

[75] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1647–1657, 2018.

[76] Linyuan Lv and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[77] Tianshu Lyu, Fei Sun, Peng Jiang, Wenwu Ou, and Yan Zhang. Compositional network embedding. *CoRR*, abs/1904.08157, 2019.

[78] Tianshu Lyu, Yuan Zhang, and Yan Zhang. Enhancing the network embedding quality with structural similarity. In *CIKM*, pages 147–156, 2017.

[79] Jianxin Ma, Peng Cui, Xiao Wang, and Wenwu Zhu. Hierarchical taxonomy aware network embedding. In *KDD*, pages 1920–1929, 2018.

[80] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *ICLR Workshop*, 2016.

[81] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163, 2000.

[82] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[83] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.

[84] Sharad Nandanwar and M. Narasimha Murty. Structural neighborhood based classification of nodes in a network. In *KDD*, pages 1085–1094, 2016.

[85] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006.

[86] Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, and Xiang Zhang. Co-regularized deep multi-network embedding. In *WWW*, pages 469–478, 2018.

[87] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, pages 6338–6347, 2017.

[88] Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[89] Akifumi Okuno and Hidetoshi Shimodaira. Robust graph embedding with noisy link weights. In *AISTATS*, pages 664–673, 2019.

[90] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114, 2016.

[91] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, 2018.

[92] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *IJCAI*, pages 1895–1901, 2016.

[93] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.

[94] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI*, pages 3934–3941, 2018.

[95] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, pages 701–710, 2014.

[96] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip!: Online learning of multi-scale network embeddings. In *ASONAM*, pages 258–265, 2017.

[97] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WWW*, pages 1509–1520, 2019.

[98] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467, 2018.

[99] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[100] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. *struc2vec*: Learning node representations from structural identity. In *KDD*, pages 385–394, 2017.

[101] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. Deep inductive network representation learning. In *WWW*, pages 953–960, 2018.

[102] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[103] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *CoRR*, abs/1603.06432, 2016.

[104] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pages 2226–2234, 2016.

[105] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Processing*, 61(7):1644–1656, 2013.

[106] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *ECML PKDD*, pages 606–622, 2016.

[107] Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. Interpretable adversarial perturbation in input embedding space for text. In *IJCAI*, pages 4323–4330, 2018.

[108] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[109] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, pages 3528–3536, 2015.

[110] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazing*, 29(3):93–106, 2008.

[111] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *AAAI*, pages 4058–4065, 2018.

[112] Xiao Shen and Fu-Lai Chung. Deep network embedding with aggregated proximity preserving. In *ASONAM*, pages 40–43, 2017.

[113] Xiao Shen and Fu-Lai Chung. Deep network embedding for graph representation learning in signed networks. *IEEE Transactions on Cybernetics*, 2018.

[114] Xiao Shen and Fulai Chung. Network embedding for cross-network node classification. In *arXiv:1901.07264*, 2019.

[115] Xiao Shen, Fulai Chung, and Sitong Mao. Leveraging cross-network information for graph sparsification in influence maximization. In *SIGIR*, pages 801–804, 2017.

[116] Xiao Shen, Fulai Chung, and Sitong Mao. Cross-network learning with fuzzy labels for seed selection and graph sparsification in influence maximization. *IEEE Transactions on Fuzzy Systems*, 2019.

[117] Jun Shu, Zongben Xu, and Deyu Meng. Small sample learning in big data era. *CoRR*, abs/1808.04572, 2018.

[118] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[119] Rashmi R. Sinha and Kirsten Swearingen. Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[120] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016.

[121] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. vgraph: A generative model for joint community detection and node representation learning. In *NIPS*, 2019.

[122] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[123] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[124] Ben Tan, Yu Zhang, Sinno Jialin Pan, and Qiang Yang. Distant domain transfer learning. In *AAAI*, pages 2604–2610, 2017.

[125] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *WWW*, pages 287–297, 2016.

[126] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[127] Jie Tang, Tiancheng Lou, and Jon M. Kleinberg. Inferring social ties across heterogenous networks. In *WSDM*, pages 743–752, 2012.

[128] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, pages 990–998, 2008.

[129] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Min. Knowl. Discov.*, 23(3):447–478, 2011.

[130] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[131] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, 2016.

[132] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. Deep recursive network embedding with regular equivalence. In *KDD*, pages 2357–2366, 2018.

[133] Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *ACL*, pages 2704–2713, 2019.

[134] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 2962–2971, 2017.

[135] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[136] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017.

[137] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9:2579–2605, 2008.

[138] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[139] Cdric Villani. *Optimal transport: old and new*, volume 338. Springer-Verlag Berlin Heidelberg, 2008.

[140] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

[141] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, and K.M. Borgwardt. Graph kernels. *JMLR*, 11:1201–1242, 2010.

[142] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi. A semi-supervised graph attentive network for financial fraud detection. In *ICDM*, pages 598–607, 2019.

[143] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, pages 1225–1234, 2016.

[144] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, 2018.

[145] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[146] PeiFeng Wang, Shuangyin Li, and Rong Pan. Incorporating GAN for negative sampling in knowledge representation learning. In *AAAI*, 2018.

[147] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, pages 203–209, 2017.

[148] Zhitao Wang, Chengyao Chen, and Wenjie Li. Predictive network representation learning for link prediction. In *SIGIR*, pages 969–972, 2017.

[149] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S. Yu. Embedding of embedding (EOE): joint embedding for coupled heterogeneous networks. In *WSDM*, pages 741–749, 2017.

[150] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S. Yu. On exploring semantic meanings of links for embedding social networks. In *WWW*, pages 479–488, 2018.

[151] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(8):1633–1647, 2017.

[152] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. Network representation learning with rich text information. In *IJCAI*, 2015.

[153] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. Fast network embedding enhancement via high order proximity approximation. In *IJCAI*, pages 3894–3900, 2017.

[154] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

[155] Jihang Ye, Hong Cheng, Zhe Zhu, and Minghua Chen. Predicting positive and negative links in signed social networks by transfer learning. In *WWW*, 2013.

[156] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.

[157] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[158] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. In *KDD*, pages 2663–2671, 2018.

[159] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[160] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. Prone: Fast and scalable network representation learning. In *IJCAI*, pages 4278–4284, 2019.

[161] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. In *IJCAI*, pages 4327–4333, 2019.

[162] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. ANRL: attributed network representation learning via deep neural networks. In *IJCAI*, pages 3155–3161, 2018.

[163] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *KDD*, pages 2778–2786, 2018.

[164] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *ICLR*, 2016.

[165] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. Deep variational network embedding in wasserstein space. In *KDD*, pages 2827–2836, 2018.

[166] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *IJCAI*, pages 4119–4125, 2015.

[167] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, pages 2847–2856, 2018.