MACHINE LEARNING WITH GRAPHS

by

JIANJIN DENG

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2021

Copyright © by Jianjin Deng 2021

All Rights Reserved

To my families, for their endless trust, support and love.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervising professor, Dr. Chris H.Q. Ding who inspired me to do this thesis without whom this thesis would not have been possible. His irreplaceable encouragement and supervision are the main reasons of the successful outcomes of my research. I am always impressed by his sharp thinking and myriad mind. Besides valuable advice in academics, he also helped me adapt to American culture. Moreover, my sincere thanks also go to the rest of my committee: Dr. Jean Gao, Dr. Junzhou Huang and Dr. Jeff (Yu) Lei for providing me great insights and feedbacks on my thesis.

I indebted to Dr. Jin-gang Yu, who is an associate professor in South China University of Technology. He has been a great mentor and collaborator. He helped me by improving my scientific writing and giving me thoughtful comments on the thesis. I really enjoyed our work on graph-based semi-supervised learning.

I also owe a great debt of gratitude to my colleagues and friends in UTA, who made my Ph.D. journey a very memorable experience: Guodong Liu, Di Ming, Qicheng Wang, Yun Liu, Teng Long, Shuai Zheng, Hongchang Gao, De Wang, Zhouyuan Huo, Jie Xu, Jiawen Yao, Xinliang Zhu, Zhifei Deng, Weizhi An, Yong Zhao, Xin Miao. Hope our friendship will last forever. In addition, I am very grateful to everyone in Christian Campus Center (Tri-C) at UTA. In particular, I would like to thank Jerry Sandoval, who has been teaching me American culture and great Bible stories. Our friendship makes my Ph.D. life much easier and more enjoyable.

Finally I would like to thank my family. My parents Xiongjie Deng and Yanyu Zhou inspired my curiosity about the natural world. My dear wife Yan Luo has

been always believing in me and encouraging me to pursue my dream, making thesis writing an enjoyable endeavor. Finally, my 1-year-old son Siqi brings to my life so much love and happiness.

April, 2021

ABSTRACT

MACHINE LEARNING WITH GRAPHS

Jianjin Deng, Ph.D.

The University of Texas at Arlington, 2021

Supervising Professor: Dr. Chris H.Q. Ding

In recent years, graph-based machine learning methods have attracted great attention because of their effectiveness and efficiency. Inspired by this trend, this thesis summarizes my research topics on machine learning techniques for the purpose of handling various kinds of problems on large graph data.

Generally, this thesis contains two parts. The first part is devoted to graph embedding, which aims to encode graph structure into dense vectors (or embeddings). In particular, we will consider a low rank-matrix factorization based approach to learn embeddings of attributed graphs. By jointly preserving graph structure and attribute-level similarity, our approach can generate embeddings, whose quality is higher than that of embeddings generated by state-of-the-art methods.

The second part of the thesis is devoted to graph-based semi-supervised learning, which attempts to predict labels for unlabeled nodes given a small set of labeled nodes and a large set of unlabeled nodes. In this part, we consider two different approaches: graph-regularization based semi-supervised learning and graph convolutional network, which deal with non-attributed and attributed graphs respectively. For graph-regularization based semi-supervised learning, we develop a simple ap-

proach for imbalanced classification, which can not only learn a smooth label function on the graph but also take into account the class imbalance of datasets. For graph convolutional network, we first introduce an attention mechanism induced by sub-maximal entropy random walks. Given this, we propose an attention-based graph convolutional network, which can jointly learn node attributes and graph structures at multiply scales. Both approaches can achieve promising performance on several benchmark datasets.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Graphs are typically used to represent complex systems of interacting objects, such as social networks [1], biological networks [2], citation networks [3], and so forth. Generally, a graph is a collection of objects (i.e., nodes) and interactions (i.e., edges) between them. For instance, in a citation network, nodes can represent academic publications and edges can represent citation links between these publications. This graph formalism provides us both mathematical elegance and effectiveness to analyze real-world complex systems. For instance, we can obtain the similarity between two publications by only analyzing topological structure of the citation network without the access to the content of these publications.

However, the inputs of most dominant machine learning algorithms are real number vectors, such as k-means [4], support vector machine (SVM) [5], neural networks [6], and so forth. Hence, it is difficult to directly apply them to graph data. Moreover, in the last 20 years, there has been a dramatic increase in the large-scale graph data which are available to researchers. Thus, the main challenge is to develop effective and efficient way to incorporate information of graph data into machine learning frameworks.

The aim of this thesis is to develop graph-based machine learning methods to solve various real-world problems. In the first part of this thesis, we propose effective graph embedding method to represent nodes as dense vectors, which can be easily exploited by traditional machine learning algorithms. In the second part,

Table 1.1: Notations and definitions in this thesis.

Notation	Description
G	A graph.
V	The set of nodes in a graph.
v_i	A node $v_i \in V$.
E	The set of edges in a graph.
e_{ij}	An edge $e_{ij} \in E$ between v_i and v_j.
$\mathcal{N}_i$	The index set of immediate neighbors of node v_i
A	The adjacency matrix.
D	The degree matrix, $D_{ii} = \sum_j A_{ij}$.
n	The number of nodes.
X	The attribute matrix of a graph.
y_i	The label of a node or data point.
Y	The indicator matrix corresponding to labels.

we develop two different graph-based semi-supervised learning methods to deal with node classification task.

1.1 Notations and Definitions

Before we discuss graph-based machine learning, it is necessary to introduce several notations and definitions required to understand this thesis. Formally, a graph is denoted by G = (V, E) where V is the set of nodes and E is the set of edges. Let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ denote an edge going from v_i to v_j. The index set of immediate neighbors of node v_i is defined as $\mathcal{N}_i = \{j \in V | e_{ij} \in E\}$. The adjacency matrix A is a n-by-n matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. D is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. A graph may have node attributes $X \in \mathbb{R}^{n \times p}$, where X is an attribute matrix with $x_i \in \mathbb{R}^p$ representing the attribute vector attached to node v_i. y_i is the label of node v_i or data point x_i, and Y is the indicator matrix consistent to labels. In this thesis, the non-attributed graph

refers to a graph without node attributes and the attributed graph represents a graph attached with node attributes. Table 1.1 summarizes the notations used in this thesis.

1.2 Tasks of Graph-based Machine Learning

Generally, main tasks of graph-based machine learning can fall into three categories as follows.

Node-level prediction. For node-level prediction, there are two main graph-based machine learning tasks: node classification and clustering. In recent years, *node classification* is the most popular supervised machine learning task. Given a graph with a portion of nodes labeled, the goal of node classification is to predict labels for unlabeled nodes, which can be very important for many reasons. For instance, a recommendation systems can suggest products, such as music, books or movies, to users with similar interests or experiences (i.e., labels). Another important node-level task is *node clustering*, which is also known as community detection or graph partition. Without labels for nodes, it aims to find groups of nodes, where nodes in the same group are more similar to each other than to the other nodes.

Link-level prediction. Another class of graph-based machine learning applications is *link prediction*. Give a graph with some edges (i.e., links between nodes) which are missing, the goal of link prediction is to infer missing edges between nodes in the graph. Link prediction is really important for many real-world applications. For instance, in an online social network, we can recommend new friends to users by using link prediction.

Graph-level prediction. The graph-level prediction involves classification and clustering on entire graphs. For graph-level classification problem, each graph is regarded as a data point and associated with a label. Instead of making prediction for components (i.e., nodes) of a single graph, *graph classification* attempts to predicts la-

Table 1.2: Overview of graph-based machine learning.

Category	Approach	Inputs	End-to-end	Application
Unsupervised	Non-attributed graph embedding	A	No	node classification [7], clustering [8] and link prediction [9]
	Attributed graph embedding*	X, A	No	node classification [10], clustering [11] and link prediction [12]
Semi-supervised	Graph-regularization based semi-supervised learning*	A, Y	Yes	node classification [13]
	Graph convolutional network*	X, A, Y	Yes	node classification [14], link prediction [15] and graph classification [16]

*The topic will be included in this thesis.

bels of multiple graphs. Graph classification is an important problem with application to many fields, such as social network analysis, bioinformatics and chemoinformatics. For instance, in chemoinformatics molecules can be viewed as graphs where nodes and edges correspond to atoms and chemical bonds between atoms respectively. Graph classification could be used to predict solubility or toxicity of a molecule. Similarly, the goal of *graph clustering* is to generate groups of graphs which are more similar to each other than to the other graphs.

1.3 Overview of Graph-based Machine Learning

Generally, graph-based machine learning methods can be categorized to two classes (see Table 1.1): unsupervised and semi-supervised methods[1]. Graph-based unsupervised learning is typically termed as graph embedding, which acts as the preprocessing method to learn informative vector representations (embeddings) for nodes while preserving graph structure. These learned embeddings can be used as input of down-stream machine learning tasks, such as node classification, clustering, link prediction, and so forth. According to different types of inputs, graph embedding

[1]For graph-based machine learning, we usually have access to the structure of the full graph, which includes both labeled and unlabeled nodes. To take advantage of unlabeled nodes, semi-supervised methods are always preferred instead of supervised ones.

falls into two categories: non-attributed and attributed graph embedding, which take non-attributed and attributed graphs as input respectively. On the other hand, graph-based semi-supervised learning is an end-to-end approach, which attempts to directly predict unlabeled data by learning graph structure and labeled data jointly. Similarly, it also contains two different approaches: graph-regularization based semi-supervised learning and graph convolutional network, which take partially labeled non-attributed and attributed graphs as input respectively.

1.4 Main Contributions of This Thesis

My Ph.D. research primarily focuses on both themes: graph embedding (part I) and graph-based semi-supervised learning (part II). Specifically, the main contributions of my thesis are as follows:

1. We proposed a low-rank matrix factorization based method for attributed graph embedding (Chapter 2).
2. We introduced a simple graph-regularization based semi-supervised approach for imbalanced classification (Chapter 3). We also proposed multi-entropy-rate graph convolutional network for semi-supervised node classification (Chapter 4).

In summary, all proposed approaches provide insights towards better understanding of graph-based machine learning and enjoy solid theoretical foundations. In practice, all proposed approaches can outperform strong basedlines on popular benchmark datasets. We also took much effort to develop software tools for these algorithms and make them publicly available.

1.5 Overview of Part I: Graph Embedding Approach

As we mentioned above, many complex data can be represented as graphs in the real world. However, it is difficult to directly apply most popular machine learning methods to graph-structured data, since they typically take attribute vector as input. Hence, it is natural to transform graphs into data with an underlying Euclidean structure, where each node is represented by a dense vector. In this part of the thesis, we mainly focus on learning embeddings for nodes in the attributed graph in an unsupervised manner.

1.5.1 Low-Rank Embedding of Attributed Graph

In the real world, graphs are typically attached with node attributes. In other words, each node is associated with an attribute vector, which can enhance the quality of embeddings. The goal of attributed graph embedding is to generate embeddings by preserving the node-level similarity and attribute-level similarity simultaneously. Since real-world graphs are typically sparse and their adjacency matrices are low-rank, in Chapter 2 we propose an attributed graph embedding approach which is based on low-rank matrix factorization.

1.6 Overview of Part II: Graph-based Semi-supervised Learning (GSSL) Approaches

For node classification tasks, supervised learning can only take use of labeled nodes on the graph while missing the information of graph structure. To alleviate this issue, GSSL approaches were proposed to learn labeled nodes and graph structure jointly. They have achieved great success on several machine learning tasks, such as node classification, link prediction and graph classification [12, 15, 16].

1.6.1 A Simple Graph-Regularization based Semi-supervised Learning Approach for Imbalanced Classification

Graph-Regularization based Semi-Supervised Learning (GRSSL) methods aim to label unlabeled data by learning graph structure and labeled data jointly. However, classification for imbalanced datasets is still an open problem. Especially, as we will discuss in Chapter 3, most existing GRSSL approaches can not deal with a wide range of class imbalance since the graph-regularization term encourages the balanced classification result. Hence, we propose a simple GRSSL approach, which can deal with various class imbalance of given datasets. The key idea of our approach is to introduce a novel term in our regularization framework to control the balance of the classification result, which can enhance the discriminative power of learned smooth classification function. Moreover, it has interesting connections to the Markov stability of graph partition and the group inverse of normalized Laplacain matrix. For classification problems, experimental results demonstrate our approach can achieve promising performance on several datasets with different class imbalance.

1.6.2 Multi-Entropy-Rate Graph Convolutional Network

Graph convolutional network attempts to generalize convolutional operation on graph-structured data. Its main innovation is to combine graph convolutional operation with multi-layer perceptron (MLP). Hence, it can generate a vector representation for a node by aggregating feature information from its local neighborhood. In Chapter 4, we propose a novel attention-based neural network architecture for semi-supervised learning for node classification. Inspired by the fact that random walks with different entropy rates can extract graph topology at multiple scales, we construct the attention mechanism via random walks with different entropy rates. Using this simple graph attention, we introduce Multi-Entropy-Rate Graph Convolutional

Network (MER-GCN), which can efficiently learn node features and graph topology at multiple scales. Experimental results demonstrate that, for node classification tasks, our approach outperforms several strong baselines under both transductive and inductive learning settings.

1.7 Structure of the Thesis

The outline of this thesis is as follows. Chapter 2 introduces an attributed graph embedding approach, which is framed as a matrix factorization framework. Chapter 3 introduces a graph-regularization based semi-supervised learning method to handle the image and text classification problems. Chapter 4 presents multi-entropy-rate convolutional network, which can learn different topological structures and node features jointly for node classification problem. Chapter 5 draws the conclusion of this thesis and presents our future research plans.

Part I

Graph Embedding Approach

CHAPTER 2

LOW-RANK EMBEDDING OF ATTRIBUTED GRAPH

2.1 Introduction

In the last few years, a surge of approaches [7, 9, 17, 18] have been proposed to incorporate the information of graph structure into machine learning algorithms. Their key idea is to encode a node into a dense vector (embedding), which can capture graph structure. Then, these node embeddings can be used for the downstream vector-based machine learning tasks, such as node classification, clustering, link prediction, and so forth [7, 9, 19].

More Recently, matrix-factorization (or factorization) based graph embedding has attracted great attention, since most dominate graph embedding approaches can be unified as matrix factorization [10], such as DeepWalk [7], Node2vec[9], Line [17]. Following this research line, attributed graph embedding approaches [10, 20] were proposed to enhance node representations by incorporating node attributes.

However, the sparsity of real-world graphs is one of the main challenges of existing factorization-based approaches. Given a sparse graph, the corresponding similarity matrix can be defined by computing the local or global pairwise node similarities. For instance, one might simply use the corresponding adjacent matrix as the similarity matrix. Typically, this similarity matrix to be factorized is low-rank and very sparse, when some approaches [21, 10, 20] only consider the local similarity between nodes in the graph. On the other hand, for the methods [7, 8, 22, 23] exploring the higher-order similarity, it is very expensive to construct and factorize a large dense matrix in terms of both time and space.

To addressing the graph sparseness problem, in this chapter we propose a simple but effective attributed graph embedding approach, which approximates the similarity matrix using a low-rank matrix. Moreover, we prove the objective of our approach is convex and proposed algorithm can achieve the globally optimal solution. Empirically, we evaluate our approach through extensive experiments on four wildly used benchmarks.

In summary, our main contributions are as follows:

- We figure out a deeper insight of existing factorization based graph embedding methods. For real-word networks, since the matrix to be factorized is typically low-rank and very sparse, it is natural to use a low-rank matrix to approximate the original similarity matrix.
- Inspired by this insight, we propose an attributed graph embedding algorithm, which can efficiently obtain the globally optimal solution of proposed framework.
- Extensive experiments on four benchmarks suggest our method can outperform several strong baselines (including deep learning approaches) on node classification and clustering tasks.

2.2 Problem Definition and Preliminaries

In this section, we first define the problem of attributed graph embedding. Then we briefly introduce preliminaries of matrix factorization based approaches.

2.2.1 Problem Definition

The primary input to our algorithm is an undirected [1] graph $G = (V, E, A)$. In addition, we suppose the graph is associated with an attribute matrix $X = [x_1, \cdots, x_n]^T \in \mathbb{R}^{n \times p}$, where each attribute vector x_i corresponds to $v_i \in V$. Given

[1] If G is directed, we will convert it to the undirected.

these, the attributed graph embedding learning aims to map nodes into a low-dimensional vector space, where embeddings simultaneously preserve the node similarity in the graph and encode the raw attributes. Formally, our goal is to find a mapping $\Phi : \mathbb{R}^p \mapsto \mathbb{R}^d (d << p)$, which satisfies two criteria: (1) $similarity(v_i, v_j) \approx \Phi(v_i)^T \Phi(v_j)$ with $v_i, v_j \in V$, and (2) each $\Phi(v_i)$ should extract the information of node attributes x_i.

2.2.2 Non-attributed Graph Embedding as Matrix Factorization

Given a non-attributed graph, most dominate graph embedding approaches can be unified as matrix factorization [23]. In this subsection, we use factorization based DeepWalk to illustrate their main idea. Inspired by word embedding algorithm Skip-gram [24], DeepWalk [25] first applies the random walk on the graph to generate node sequences, which can be analogized as sentences in a special language. Then it applies the neural language model (Skip-gram algorithm) to transform these node sequences into node embeddings.

Given a node sequence $\{v_0, \cdots, v_n\}$ generated by the random walk, they formulate graph embedding as an optimization problem to maximize the likelihood of observing the neighbors given a node.

$$\begin{aligned} &\max_{\Phi} \sum_{i}^{S} \log Pr(\{v_{i-w}, \cdots, v_{i+w}\} \backslash v_i | v_i), \\ &= \max_{\Phi} \sum_{i} \sum_{j \in N(i)} \log Pr(v_j | v_i), \end{aligned} \tag{2.1}$$

where w is the window size, $\mathcal{N}_i$ is the index set of the neighbors of node v_i.

In Skip-gram model, the conditional probability $Pr(v_j|v_i)$ in Eq. 2.1 can be defined by the softmax function,

$$Pr(v_j|v_i) = \frac{exp(\Phi(v_i)^T \Phi'(v_j))}{\sum_{j'} exp(\Phi(v_i)^T \Phi'(v_{j'}))},$$

where $\Phi(v_i)$ and $\Phi'(v_{j'}) \in \mathbb{R}^d$ are the two embeddings of the nodes v_i and $v_{j'}$, respectively. The objective function in Eq. 2.1 can be optimized using stochastic gradient decent with hierarchical softmax [7].

It has been proven in [10, 23] that DeepWalk is equivalent to the matrix factorization,

$$M \approx W \times H^T, \tag{2.2}$$

where $W \in \mathbb{R}^{n \times d}$, $H \in \mathbb{R}^{n \times d}$ are the representations of nodes. And given the row-normalized transition matrix $P = D^{-1}A$ with $D_{ii} = \sum_j A_{ij}$, M can be defined as $log(\frac{P+P^2+\cdots+P^l}{l})$, which defines the similarity between nodes v_i and v_j in the graph by the logarithm of the average probability that a random walk starts from node v_i to node v_j in fixed t steps. An alternative matrix factorization algorithm is Singular Value Decomposition (SVD), whose top-d singular values and vectors can be used to approximate $M = U_d \Sigma_d V_d^T$. Usually, $W = U_d \sqrt{\Sigma_d}$ is used as the embedding matrix, whose row vectors correspond to node embeddings.

2.2.3 Attributed Graph Embedding as Matrix Factorization

Besides graph structures, many real-world graphs are often attached with node attributes. Many attributed graph embedding methods [10, 20, 26] can be framed as matrix factorization. In this subsection, we use text-associated DeepWalk (TADW) to show the main idea. The key innovation of TADW is to factorize the similarity matrix into three matrices as follows

$$\min_{W,H} \|M - XWH^T\|_F^2 + \frac{\alpha}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{2.3}$$

where $W \in \mathbb{R}^{p \times d}$and $H \in \mathbb{R}^{n \times d}$ are the parameter matrices to optimize, and $X \in \mathbb{R}^{n \times p}$ is the attribute matrix. α is the parameter to control the trade-off between matrix approximation term and regularization term. In practice, factorizing $\frac{P+P^2}{2}$

instead of $log(\frac{P+P^2+\cdots+P^l}{l})$ is a good trade-off between efficiency and accuracy. Node representations are obtained by concatenating XW and H. Similarly, HSCA [20] is another approach which adds an extra graph Laplacian regularization term into Eq. 2.3 to preserve the homophily property of the graph.

2.3 Approach

We propose a novel algorithm to solve the attributed graph embedding problem. Its goal is to minimizes the following objective function,

$$J(Z) = \|M - XZX^T\|_F^2 + \lambda\|Z\|_*, \tag{2.4}$$

where $\|Z\|_* = Tr(ZZ^T)^{1/2}$ is the nuclear norm and λ is a trade-off parameter. Since $J(Z)$ is convex, the global optimal solution can be computed.

This formalism can efficiently deal with large graphs, because the variable Z is a p-by-p matrix, where p is the number of attributes on a node. In most real applications, the number of attributes on a node is typically several thousands or less, while the number of nodes in the graph, n could be much larger, up to several millions. Thus typically $p \ll n$. We will see that in the computational algorithm, we only deal with p-by-p matrices, never deal with n-by-n matrices. Although the graph adjacency matrix M is n-by-n, we only use repeatedly the p-by-p matrix X^TMX which is assembled once and stored for later usages. The computation of X^TMX does not require completely form the entire matrix M in memory.

One model closely related to the target model in Eq. 2.4 is

$$J_2(Z) = \|M - XZX^T\|_F^2 + \lambda\|Z\|_F^2, \tag{2.5}$$

Here, the regularization term is the widely used L_2 norm on Z, instead of the rank-suppression nuclear norm in Eq. 2.4. One big advantage of this model is that the model solution can be expressed in closed form as the following Sylvester equation

$$\tilde{A}Z + Z\tilde{B} = \tilde{C}. \tag{2.6}$$

where

$$\tilde{A} = X^T X,\ \tilde{B} = \lambda(X^T X)^{-1},\ \tilde{C} = X^T M X (X^T X)^{-1}.$$

The Sylvester equation is a linear equation: using vectorization notation, it can be expressed as $(I \otimes \tilde{A} + \tilde{B} \otimes I)vec(Z) = vec(\tilde{C})$, where $\otimes$ denotes Kronecker product and I is identity matrix of size $p \times p$ same as $\tilde{A}, \tilde{B}, \tilde{C}$.

A well-known theorem [27] says that if $\tilde{A}$ and $-\tilde{B}$ share no identical eigenvalues, the linear equation has a unique solution. In our problem, $\tilde{A}, \tilde{B}$ are, by definition, semi-positive-definite matrices; thus $\tilde{A}$ and $-\tilde{B}$ do not have same nonzero eigenvalues. A subtle issue is the zero eigenvalue. In typical attributed graph applications, $p \ll n$, thus $\tilde{A} = X^T X$ is full rank. In the case when $\tilde{A} = X^T X$ is not full rank, we usually compute $(X^T X)^{-1}$ by $(X^T X + \epsilon I)^{-1}$. In summary, in our problem, $\tilde{A}$ and $-\tilde{B}$ do not have same eigenvalues. Thus the Sylvester equation has a unique solution, which gives the optimal solution Z^* for minimizing $J_2(Z)$.

The model Eq. 2.4 is motivated by the work of Xu et al. [28] who attempt to solve matrix completion problems, which is extended to do many other problems unrelated the attributed graph embedding problem that we are dealing with in this chapter.

In the following, we first present an efficient algorithm to solve the model Eq. 2.4. Later we prove the convergence and correctness of the algorithm.

2.3.1 Algorithm

We use an iterative algorithm to compute the globally optimal solution. We first consider the initialization of Z. At $t = 0$, the initialization of $Z_{t=0}$ can be set to either a random matrix or the solutions to the closely related models which can be solved in closed form. In our approach, we simply set $\lambda = 0$ in model Eq. 2.4 and use the solution at $\lambda = 0$ as the initialization of the model. The $\lambda = 0$ solution of the model is given by

$$Z^*_{\lambda=0} = (X^T X)^{-1} X^T M X (X^T X)^{-1}. \tag{2.7}$$

Note that $X^T X$ is a p-by-p matrix, whose inverse can be easily computed in typically applications where $p \ll n$. When the semi-positive-definite matrix $X^T X$ has zero (or very small) eigenvalues, we typically replace $(X^T X)^{-1}$ by $(X^T X + \epsilon I)^{-1}$ with $\epsilon \approx 10^{-10}$.

Given Z_t, the key algorithm step is to compute Z_{t+1}. Setting

$$\begin{aligned} \tilde{A} &= (Z_t Z_t^T)^{\frac{1}{2}} X^T X, \\ \tilde{B} &= \frac{\lambda}{2} (X^T X)^{-1}, \\ \tilde{C} &= (Z_t Z_t^T)^{\frac{1}{2}} X^T M X (X^T X)^{-1}, \end{aligned} \tag{2.8}$$

we solve the Sylvester equation as follows,

$$\tilde{A} Z + Z \tilde{B} = \tilde{C}. \tag{2.9}$$

The solution is for Z_{t+1}. Note that all key quantities involved in this step, $X^T X$, $X^T M X$, $Z_t Z_t^T$ are p-by-p matrices. $X^T X, (X^T X)^{-1}, X^T M X$ are pre-computed and stored for efficient execution. This completes the description of the computational algorithm, which is summarized in Algorithm 1. In step 7, $\tilde{X}$ is the enhanced attribute matrix, which will be introduced in Section 2.4.1.

Algorithm 1 Our Algorithm

Input: Adjacency matrix A; attribute matrix X.

Output: A matrix of node embeddings, where each row corresponds to a node embedding.

1: Initialization: $Z^* = (X^TX)^{-1}X^TM^TX(X^TX)^{-1}$.

2: **for** $t = 1$ to T **do**

3: Compute $\tilde{A}, \tilde{B}, \tilde{C}$ from Eq. 2.8 for $Z = Z_t$,

4: Solve Sylvester equation (Eq. 2.9) to obtain Z_{t+1},

5: **end for**

6: Perform spectral decomposition: $Z_{T+1} = USU^T$.

7: Embedding vectors are obtained by concatenating XU and $\tilde{X}U$.

2.3.2 Convergence of Our Algorithm

Here we prove that the algorithm described above converges by introducing the theorem below. In next subsection, we show that the converged Z_∞ satisfies the KKT condition of the convex optimization problem. Thus the converged solution is a correct globally optimal solution.

Theorem 2.3.1. *In the iterative algorithm, the successive updated solutions Z_t, Z_{t+1} monotonically decrease the objective function $J(Z)$:*

$$J(Z_{t+1}) \leq J(Z_t).$$

Because $J(Z)$ is bounded from below, i.e., $J(Z) \geq 0$, Z_t converges to a fixed point.

Proof. We first introduce an auxiliary function

$$F(Z, Z_t) = \|M - XZX^T\|_F^2 + \Gamma(Z, Z_t),$$

where

$$\Gamma(Z, Z_t) = Tr(Z_t Z_t^T)^{\frac{1}{2}} + \frac{1}{2}Tr(Z^T(Z_t Z_t^T)^{-\frac{1}{2}}Z) - \frac{1}{2}Tr(Z_t^T(Z_t Z_t^T)^{-\frac{1}{2}}Z_t),$$

for any Z and Z_t. We define Z_{t+1} as the optimal solution

$$Z_{t+1} = \arg\min_Z F(Z, Z_t). \tag{2.10}$$

Now we wish to prove the following facts

$$J(Z) \leq F(Z, Z_t) \ \ \forall Z, Z_t, \tag{2.11}$$

and

$$J(Z) = F(Z, Z). \tag{2.12}$$

Eq. 2.12 is obvious. To prove the inequality Eq. 2.11, we write

$$\begin{aligned} J(Z) - F(Z, Z_t) = Tr(ZZ^T)^{\frac{1}{2}} - \big[Tr(Z_t Z_t^T)^{\frac{1}{2}} \\ + \frac{1}{2}Tr(Z^T(Z_t Z_t^T)^{-\frac{1}{2}}Z) - \frac{1}{2}Tr(Z_t^T(Z_t Z_t^T)^{-\frac{1}{2}}Z_t)\big]. \end{aligned} \tag{2.13}$$

Now, we invoke a useful lemma from (Luo et al. 2011 [29]).

Lemma 2.3.1. *Given any Z and $Z_t \in \mathbb{R}^{p\times p}$, we have*

$$\begin{aligned} Tr(ZZ^T + \epsilon I)^{\frac{1}{2}} \leq & Tr(Z_t Z_t^T + \epsilon I)^{\frac{1}{2}} \\ & + \frac{1}{2}Tr(Z^T(Z_t Z_t^T + \epsilon I)^{-\frac{1}{2}}Z) \\ & - \frac{1}{2}Tr(Z_t^T(Z_t Z_t^T + \epsilon I)^{-\frac{1}{2}}Z_t), \end{aligned}$$

where ϵ is a positive constant.

See [29] for the proof. Using Lemma 2.3.1, with $\epsilon \to 0$, Eq. 2.13 becomes $J(Z) - F(Z, Z_t) \leq 0$, which gives Eq. 2.11.

From Eqs. 2.11 and 2.12, we obtain the following property of the iteration solution,

$$J(Z_t) = F(Z_t, Z_t) \geq F(Z_{t+1}, Z_t) \geq J(Z_{t+1}). \tag{2.14}$$

The first equality is from Eq. 2.12. The second inequality is from the fact that Z_{t+1} is the global optimal solution of Eq. 2.10. Thus $F(Z_{t+1}, Z_t) \leq F(Z, Z_t)$ for any Z. In particular, this holds for $Z = Z_t$, thus leads to the second inequality. The third inequality is from Eq. 2.13.

The inequality Eq. 2.14 establishes the monotonic decreasing (not increasing) property of the iterative algorithm.

Now we need to prove the solution for Z_{t+1} computed through Eqs. 2.8 and 2.9 is indeed the locally optimal solution to Eq. 2.10. For this purpose, we take

$$\begin{aligned} 0 &= \frac{\partial F(Z, Z_t)}{\partial Z} \\ &= -2X^T M X + 2X^T X Z X^T X + \lambda (Z_t Z_t^T)^{-\frac{1}{2}} Z, \end{aligned} \tag{2.15}$$

which can be written as

$$\begin{aligned} \big[(Z_t Z_t^T)^{\frac{1}{2}} X^T X\big] Z + Z\big[\frac{\lambda}{2}(X^T X)^{-1}\big] \\ = (Z_t Z_t^T)^{\frac{1}{2}} X^T M X (X^T X)^{-1}. \end{aligned} \tag{2.16}$$

One can see that this is exactly the Sylvester equation defined in Eqs. 2.8 and 2.9. Thus the computed Z_{t+1} is the locally optimal solution to the optimization problem of Eq. 2.10. □

2.3.3 Optimality of Our Algorithm

Here we show that the converged Z_∞ satisfies the KKT condition of the convex optimization problem. Thus the converged solution is a correct globally optimal solution.

Setting $\frac{\partial J(Z)}{\partial Z}\big|_{Z=Z^*} = 0$, we obtain the KKT optimality condition

$$-2X^T M X + 2X^T X Z^* X^T X + \lambda [Z^* (Z^*)^T]^{-\frac{1}{2}} Z^* = 0. \tag{2.17}$$

Now we check through the solution for Z_{t+1} in Eqs. 2.17 and 2.18, when the algorithm converged, $Z_t = Z_\infty$, and $Z_{t+1} = Z_\infty$, thus the Sylvester equation in Eq. 2.16 becomes

$$\begin{aligned}\big[(Z_\infty Z_\infty^T)^{\frac{1}{2}} X^T X\big] Z_\infty + Z_\infty \big[\frac{\lambda}{2}(X^T X)^{-1}\big] \\ = (Z_\infty Z_\infty^T)^{\frac{1}{2}} X^T M X (X^T X)^{-1}.\end{aligned} \tag{2.18}$$

One can see that Z_∞ satisfies the KKT condition Eq. 2.17. Thus the computed solution is the optimal solution Z^* to the optimization problem.

2.4 Data Transformations

In above algorithm, the graph similarity matrix M and attribute vectors X are assumed to be the original input data for the attributed graph. The output is the embedding data Z.

However, like many other methods [30, 31] in machine learning, we can transform the original input data to achieve better results. Below we discuss several data transformation techniques.

First, we discuss data transformation of the graph similarity matrix. For clarity, we assume the input graph adjacency matrix is $A \in \mathbb{R}^{n \times n}$.

We first consider the node pairwise similarity introduced by DeepWalk [25], which defines the similarity between nodes v_i and v_j in the graph by $M_{ij} = log(\frac{1}{t}(P + P^2 + \cdots + P^t))_{ij}$. In this work, to take advantage of graph sparsity, following [10] we omit the log operation and choose

$$M = (P + P^2)_{sym}, \tag{2.19}$$

where for a matrix G, $G_{sym} = (G + G^T)/2$, which is an ideal trade-off between effectiveness and efficiency in practice.

2.4.1 Attribute Transformations

In our model Eq. 2.4, $X = [x_1, \cdots, x_n]^T$ are original attribute vectors on the nodes of the graph. However, we may equally well use transformed attributes here. For example, we may use principal component analysis (PCA) to reduce the dimension of attributes; This will reduce the computation of the final solution of the model; it will also reduce the noise in the attributes data.

Another transformation to enhance (smooth) the initial attributes is to update each node by aggregating the attribute vectors in its local neighborhood,

$$\tilde{x}_i \leftarrow \sum_{j \in \mathcal{N}_i} x_j P_{j \to i}, \tag{2.20}$$

where $P_{j \to i}$ is the transition probability from node v_j to node v_i. The widely used graph random walk transition probability is

$$P_{j \to i} = A_{ji} / \sum_k A_{jk} = A_{ji}/d_j = (D^{-1}A)_{ji}, \tag{2.21}$$

where A is the graph adjacency matrix and d_j is the degree for node v_j. With these analysis, the transformed attributes are

$$\tilde{X} \triangleq [\tilde{x}_1, \cdots, \tilde{x}_n]^T = \hat{P}X. \tag{2.22}$$

Note that this smoothing transformation can be repeated several times. Furthermore, the standard random walk transition probability in Eq. 2.21 can be replaced by

$$\hat{P} = (D + I)^{-1}(A + I) \tag{2.23}$$

which is found to perform better in graph convolutional networks [14, 30]. Summarizing all above discussions, the final smooth transformation on attribute vectors is

$$\tilde{X} = \hat{P}^K X. \tag{2.24}$$

Let us briefly explain why the smooth transformation in Eq. 2.24 can enhance the node attributes. Theoretically, $\hat{P}^K$ can be viewed as a filter from the perspective of graph signal processing [32]. It will prefer the large eigenvalues of $\tilde{P}$ corresponding to large-scale structure in the graph, and suppress the noise corresponding to the small eigenvalues. Intuitively, the propagated attribute matrix $\tilde{X} = \hat{P}^K X$ is smoother than initial one X. That is, the nodes nearby in the graph are more likely to share the similar attributes. Hence, the similarity between node attribute vectors of $\tilde{X}$ can reflect the initial attribute similarity and the node similarity in the graph, which makes propagated attributes $\tilde{X}$ more informative.

2.5 Experiments

We compare our approach against several strong baselines on four benchmark datasets on two tasks, i.e., node classification and clustering.

The statistics of four datasets are summarized in Table 2.1.

Table 2.1: Dataset statistics

Dataset	# Nodes	# Edges	# Classes	#Attributes
Citeseer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
Pubmed	19,717	44,338	3	500
Wiki	2,405	17,981	17	4,973

2.5.1 Datasets

We conduct experiments on four widely-used datasets [2]: Citeseer, Cora, Pubmed and Wiki. Citeseer, Cora and Pubmed are three citation networks, where nodes rep-

[2]http://linqs.cs.umd.edu/projects//projects/lbc/index.html

resent scientific publications and edges represent citation links. Labels indicate the research fields of papers. Moreover, each paper is represented as a TF-IDF vector. Wiki dataset is a web page network, where nodes represent web pages and edges represent link relations. Similar to other three datasets, each page is represented by a TF-IDF vector.

2.5.2 Baselines

We compare our approach against the following baselines on both node classification and clustering tasks.

Singular Value Decomposition (SVD): It applies Singular Value Decomposition [33] on the raw attribute matrix to extract node embeddings. The dimensionality of embeddings is set to 200, following the setting in [10, 20].

DeepWalk: DeepWalk [7] is a graph embedding method, which is presented in Section 2.2.2. We set the number of walks, walk length and window size to 10, 80, 10, respectively. And the embedding dimension is set to 128.

Node2vec: Inspired by DeepWalk, Node2vec [9] designs a second order random walk on the graph to explore different types of network structures. And it also uses Skip-gram algorithm to generate node embeddings. We choose the optimal parameters p and q with grid search over $p, q \in \{0.25, 0.5, 1, 2, 4\}$. In addition, for other parameters, it follows the settings of DeepWalk.

SVD + DeepWalk: To learn the information of graph structure and node attributes, we simply concatenate the embeddings learned by SVD and DeepWalk .

TADW: TADW [10] is a matrix factorization based approach inspired by inductive matrix completion model [34]. Parameters for this algorithm are set as default values.

HSCA: HSCA [20] is another matrix factorization based approach which aims to integrate graph structure, node attribute and homophily property to learn an informative graph embedding. Parameters for this algorithm are set as default values.

STNE: STNE [35] casts graph embedding problem as a sequence-to-sequence task, where the node sequences generated by random walks are mapped to node embeddings. We apply the default parameters for Cora, Citeseer and Wiki as mentioned in the original paper. For Pubmed, we choose the same neural network configuration as that for Citeseer.

DANE: DANE [11] utilizes two autoencoders to learn the hidden embeddings by jointly capturing the information of topological structure and node attributes. This model is trained with default parameters.

DGI: DGI [36] is an unsupervised graph convolutional neural network to maximize mutual information between patch embeddings and corresponding high-level summaries of graphs.

GMI: GMI [37] aims to learn embeddings that maximize the mutual information of both features and edges between the input (i.e., an input graph) and output (i.e., an output graph) of a graph convolutional neural network.

GAE/VGAV: To learn the graph structure and node attributes jointly, GAE and VGAV [12] apply graph convolutional neural network to reconstruct the graph. We train these models with default parameters.

RWR-GAE/RWR-VGAV: RWR-GAE and RWR-VGAV [38] are random walk based approaches to regularize the hidden embeddings learned by graph autoencoders. We train these models with default parameters.

All baselines can be categorized into three groups. SVD is an algorithm only using the node attribute information. DeepWalk and Node2vec are baselines only considering the information of graph structure. The rest approaches jointly learn

Table 2.2: Node classification results in terms of Micro-F1 (%) on Citeseer dataset.

Method	1%	3%	5%	10%	30%	50%
SVD [33]	40.58±4.25	54.25±2.29	59.03±1.96	63.42±1.13	69.02±0.88	71.05±0.97
DeepWalk [7]	37.96±4.28	48.18±2.11	51.41±1.55	53.69±1.43	56.98±0.82	57.80±0.84
Node2vec [9]	38.29±3.72	47.62±2.10	50.81±1.58	53.73±1.15	57.51±0.88	58.42±0.88
SVD + DeepWalk	36.40±3.76	49.61±2.57	54.91±1.73	60.85±1.26	68.71±0.74	71.85±0.83
TADW [10]	41.94±6.33	61.91±1.80	66.54±1.59	70.58±0.79	73.34±0.66	74.09±0.83
HSCA [20]	41.80±4.83	58.51±2.25	64.24±1.64	69.15±0.97	73.43±0.76	74.75±0.91
STNE [35]	35.08±6.21	56.73±3.17	63.64±2.03	69.02±0.79	72.97±0.71	74.22±0.82
DANE [11]	40.58±4.93	57.63±2.32	63.05±1.40	67.97±0.98	71.97±0.62	73.31±0.91
DGI [36]	48.89±6.90	67.21±3.36	68.92±1.50	71.67±0.61	74.38±0.61	74.94±0.79
GMI [37]	50.61±4.36	67.34±2.03	69.04±1.21	71.80±0.69	74.90±0.77	74.87±0.90
GAE [12]	45.51±3.34	57.85±2.10	61.58±1.40	64.88±1.02	67.75±0.63	68.32±0.80
VGAE [12]	35.31±3.63	44.87±2.57	49.45±2.16	56.48±1.15	64.46±0.88	66.83±0.94
RWR-GAE [38]	42.80±4.93	50.56±2.62	55.00±2.03	62.74±1.26	68.22±0.79	69.24±0.83
RWR-VGAE [38]	36.92±3.87	46.32±2.82	51.15±1.70	56.50±1.17	63.58±0.70	65.69±0.81
Ours	**59.37**±3.98	**68.03**±1.80	**69.91**±1.18	**71.77**±1.01	**74.57**±0.64	**75.97**±0.93

node embeddings that encode the graph structure and node attributes. Among them, TADW and HSCA are shallow models.

For all baselines, we execute their officially released code and report the results for a fair comparison. All deep learning models are executed on a NVIDIA GeForce GTX 1080 TI GPU. And the shallow models including our approach are executed on an Intel i7-6700k @ 4.00GHz x8 CPU and 32G RAM.

2.5.3 Experimental Settings for Our Approach

Following [10, 20], as the pre-processing step, we perform SVD decomposition of the TF-IDF matrix to reduce the dimensionality of raw attribute vectors to d. Note that, the smooth transformation in Eq. 2.24 is performed before SVD decomposition. We choose $d = 50$ for Cora and Citeseer, and $d = 200$ for Pubmed and Wiki. In addition, we choose regularization weight $\lambda = 1$ and number of iterations $T = 20$ across all datasets. As described in Algorithm 1, embeddings are obtained by concatenating XU and $\hat{P}^K XU$. In our experiments, we choose $K = 3$ for all datasets. Note that, the dimensionality of final embeddings is $2d$.

Table 2.3: Node classification results in terms of Micro-F1 (%) on Cora dataset.

Method	1%	3%	5%	10%	30%	50%
SVD [33]	39.66±4.31	50.89±2.67	55.10±2.18	60.58±1.73	67.36±0.97	70.97±1.13
DeepWalk [7]	51.58±5.06	68.17±2.70	73.13±1.81	76.81±1.01	80.54±0.74	81.35±0.86
Node2vec [9]	53.66±4.83	70.69±2.61	73.70±1.61	76.22±0.90	79.48±0.85	80.56±0.92
SVD + DeepWalk	47.31±6.56	66.88±3.23	72.78±1.57	77.62±1.02	82.49±0.78	84.12±0.90
TADW [10]	46.98±7.22	68.52±3.79	75.95±2.36	81.75±1.07	85.69±0.82	86.57±0.68
HSCA [20]	54.56±6.09	74.48±2.40	78.97±1.84	83.24±0.91	**86.56**±0.54	**87.89**±0.81
STNE [35]	53.44±4.23	67.12±2.21	71.90±1.58	76.89±1.17	83.14±0.82	85.52±0.77
DANE [11]	42.98±6.13	63.58±3.44	71.02±1.77	77.58±1.18	83.02±0.83	84.33±0.81
DGI [36]	60.58±7.09	76.45±3.79	80.41±1.59	83.13±0.87	86.12±0.59	86.64±0.84
GMI [37]	61.58±4.65	76.85±3.72	80.94±1.32	83.11±0.81	85.43±0.66	86.09±0.78
GAE [12]	61.33±4.11	72.75±2.23	76.37±1.51	78.82±1.06	81.03±0.73	81.57±0.76
VGAE [12]	51.50±4.71	65.24±2.78	71.23±1.63	76.57±1.18	82.49±0.68	84.33±0.87
RWR-GAE [38]	53.69±4.81	66.56±3.32	70.40±2.10	75.29±1.48	81.44±0.74	83.14±0.77
RWR-VGAE [38]	53.78±4.02	68.36±2.49	73.31±1.72	77.50±1.02	82.34±0.73	83.76±0.92
Ours	**65.45**±4.66	**78.59**±1.78	**80.95**±1.25	**83.62**±0.78	86.48±0.61	87.22±0.68

2.5.4 Node Classification

To evaluate different graph embedding approaches, we use learned embeddings to train a SVM classifier implemented by Liblinear [39]. Since the value of L_2 regularization penalty C can affect the result significantly, we apply the grid search to choose C over $\{0.1, 1, 10, 100\}$ for all approches. For each dataset, we randomly sample 1% to 50% of labeled data as training data and leave the rest for testing. In Tables 2.2 to 2.5, we report the classification metric Micro-F1 scores (with standard deviation), which are averaged over 50 different runs. For each training ratio, the best result is boldfaced.

Our approach outperforms all baselines on Citeseer, Pubmed, Wiki (see Tables 2.2-2.4). On Cora (see Table 2.5), our approach is the best when the training ratio is less than 30%. Moreover, our approach is still competitive with the state-of-the-art method (HSCA) when the training ratio becomes larger. It is interesting to note that our approach can significantly outperform matrix factorization based baselines, such

Table 2.4: Node classification results in terms of Micro-F1 (%) on Pubmed dataset.

Method	1%	3%	5%	10%	30%	50%
SVD [33]	53.74±5.48	70.80±3.17	75.69±2.00	79.64±0.97	82.90±0.32	83.61±0.31
DeepWalk [7]	73.76±2.07	78.49±0.51	79.36±0.34	79.99±0.25	80.36±0.22	80.45±0.33
Node2vec [9]	75.15±1.04	78.30±0.51	79.10±0.39	80.10±0.26	80.88±0.22	81.00±0.28
SVD + DeepWalk	74.39±1.37	77.62±0.68	79.30±0.47	81.22±0.26	83.80±0.23	84.86±0.24
TADW [10]	74.27±3.92	82.60±0.70	83.57±0.50	84.56±0.30	85.46±0.21	85.82±0.30
HSCA [20]	78.74±1.75	83.35±0.45	84.32±0.34	85.14±0.28	86.19±0.24	86.40±0.23
STNE [35]	74.54±1.43	77.75±0.61	79.16±0.50	80.99±0.28	83.30±0.28	84.30±0.28
DANE [11]	72.70±1.53	76.90±0.64	79.02±0.46	81.43±0.36	84.38±0.27	85.36±0.33
DGI [36]	82.24±0.90	84.18±0.35	84.74±0.24	85.39±0.16	86.36±0.21	86.81±0.22
GMI [37]	81.92±1.08	83.59±0.46	84.05±0.44	84.92±0.32	85.79±0.28	86.40±0.24
GAE [12]	70.30±2.65	78.23±0.96	79.31±0.30	79.70±0.17	79.98±0.19	80.13±0.27
VGAE [12]	80.90±0.74	82.84±0.36	83.45±0.24	83.98±0.21	84.66±0.24	84.86±0.23
RWR-GAE [38]	78.86±0.90	80.40±0.43	80.72±0.26	80.99±0.22	81.27±0.23	81.29±0.30
RWR-VGAE [38]	80.60±0.91	83.12±0.38	83.72±0.23	84.24±0.17	84.74±0.24	84.79±0.28
Ours	**82.18**±0.99	**84.55**±0.50	**85.65**±0.40	**86.79**±0.26	**88.21**±0.22	**88.72**±0.25

Table 2.5: Node classification results in terms of Micro-F1 (%) on Wiki dataset.

Method	1%	3%	5%	10%	30%	50%
SVD [33]	40.01±7.21	53.86±3.62	58.64±3.16	66.24±2.24	72.60±1.11	74.63±1.09
DeepWalk [7]	32.55±3.95	47.90±2.87	53.47±1.68	59.51±1.07	65.40±1.09	66.79±1.18
Node2vec [9]	34.86±4.51	49.04±2.41	53.43±2.06	58.21±1.37	63.48±1.00	65.23±1.14
SVD + DeepWalk	39.20±5.93	55.47±3.73	60.62±2.94	67.63±1.59	74.07±1.19	76.35±1.07
TADW [10]	37.52±5.78	57.46±4.04	65.38±3.03	71.89±1.49	78.51±0.84	80.29±0.93
HSCA [20]	38.46±5.74	56.46±2.83	63.16±2.05	68.88±1.59	74.97±1.00	77.24±1.03
STNE [35]	39.31±5.21	60.02±2.69	65.61±1.99	72.20±1.26	78.90±0.67	81.27±0.92
DANE [11]	41.61±5.08	60.20±3.45	66.10±2.29	71.51±1.24	76.35±0.76	77.75±1.00
DGI [36]	45.47±5.76	60.81±3.19	65.61±1.58	68.83±1.18	72.32±1.10	73.84±1.11
GMI [37]	44.70±4.64	60.43±3.65	65.87±1.91	69.16±1.41	73.29±0.99	74.59±1.24
GAE [12]	41.36±5.89	56.47±2.45	59.71±1.98	62.94±0.99	65.55±1.06	66.20±1.06
VGAE [12]	44.38±6.71	58.85±3.01	63.24±1.76	67.62±1.00	71.24±0.89	72.33±1.12
RWR-GAE [38]	40.95±4.65	51.87±3.54	57.62±2.06	62.39±1.40	66.26±1.05	67.21±1.11
RWR-VGAE [38]	39.75±4.84	54.73±3.05	60.01±2.40	66.23±1.63	72.78±0.97	74.52±1.09
Ours	**48.05**±5.43	**64.90**±3.49	**69.58**±1.99	**74.74**±1.27	**80.10**±0.80	**81.56**±0.85

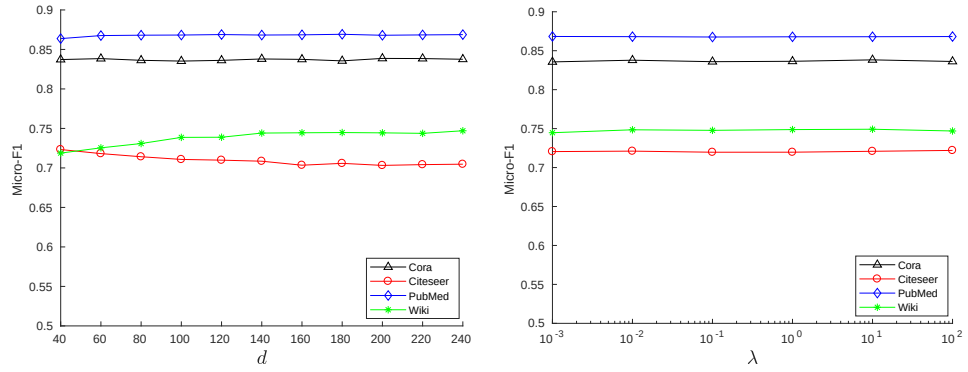
as TADW and HSCA, when training ratio is small, which indicates our embeddings are more consistent and robust.

2.5.5 Node Clustering

In Table 2.6, we report node clustering experiments on all datasets by applying k-means algorithm (with k-means++ initialization) implemented in Python's Scikit-

Table 2.6: Node clustering results in terms of ACC (%) and NMI (%).

Method	Citeseer		Cora		PubMed		Wiki	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
SVD [33]	65.18±0.41	38.78±0.46	55.04±2.64	35.80±1.19	60.52±0.01	31.07±0.01	47.72±1.94	48.24±1.16
DeepWalk [7]	44.75±2.62	20.27±0.88	66.82±2.55	46.77±1.42	66.68±0.02	29.96±0.01	42.74±2.63	36.68±1.17
Node2vec [9]	45.66±0.77	25.31±0.84	65.33±1.74	46.33±0.67	68.16±0.03	28.50±0.02	40.05±1.84	37.76±1.09
SVD + DeepWalk	62.95±1.65	37.82±1.39	67.21±1.89	50.38±1.90	66.69±0.02	29.97±0.02	46.15±1.66	48.80±1.42
TADW [10]	64.93±0.31	38.77±0.29	61.66±1.56	43.29±0.44	62.00±0.05	32.19±0.06	45.19±1.91	48.31±0.96
HSCA [20]	57.65±3.39	39.06±2.07	60.62±1.64	49.71±0.97	61.47±0.04	25.55±0.05	41.16±1.72	43.89±0.76
STNE [35]	62.90±1.07	37.83±0.96	64.21±3.08	46.62±1.82	64.64±0.01	25.94±0.02	48.82±1.75	48.08±1.05
DANE [11]	64.74±0.56	38.25±0.54	65.28±2.26	44.83±1.16	66.29±0.01	29.68±0.02	48.61±1.83	**49.13**±1.59
DGI [36]	67.83±0.58	42.99±0.29	70.32±1.95	55.48±0.91	66.12±0.01	29.11±0.01	47.33±1.98	48.70±0.41
GMI [37]	65.37±0.45	41.21±0.36	66.88±0.97	53.42±0.75	65.81±0.01	28.23±0.01	47.98±1.45	48.24±0.38
GAE [12]	40.79±3.13	18.71±1.41	52.56±2.46	38.22±1.51	62.36±0.07	21.44±0.05	44.65±0.91	43.44±0.60
VGAE [12]	41.83±1.65	18.98±0.77	53.74±0.20	42.06±0.46	64.93±0.02	24.45±0.02	45.31±1.13	42.59±0.05
RWR-GAE [38]	47.58±2.98	24.78±2.12	58.74±2.15	43.21±1.45	**69.66**±0.01	33.07±0.01	45.19±0.59	42.31±0.57
RWR-VGAE [38]	39.56±2.94	18.66±2.76	57.36±1.79	44.22±0.97	65.02±0.04	21.23±0.02	41.74±1.14	39.48±0.67
Ours	**69.23**±0.40	**44.25**±0.34	**71.39**±2.45	**57.81**±1.36	67.62±0.02	**34.32**±0.01	**49.63**±1.46	49.06±0.83



(a) d (b) λ

Figure 2.1: The Micro-F1 score versus d and λ on different datasets.

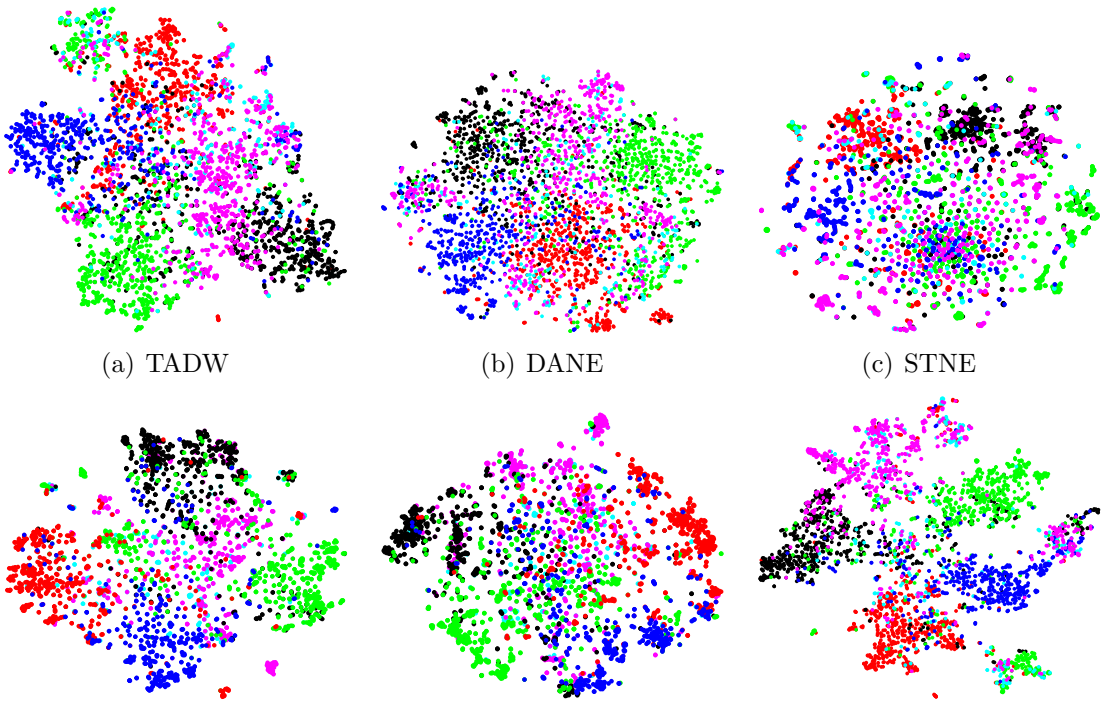
learn [40] to the learned embeddings. We compared output clusters to ground truth by computing averaged accuracy (ACC) and normalized mutual information (NMI). Numbers show mean results and standard deviation over 20 runs. For each dataset, the best results are boldfaced.

Our approach outperforms all baselines on Citeseer and Cora. On Pubmed and Wiki, our approach is competitive with few state-of-the-art algorithms and outperforms other baselines.

2.5.6 Parameter Sensitivity

Our approach has two parameters: embedding dimension d and regularization weight λ. We fix the training ratio to 10%, and evaluate the performance of SVM classifier on the embeddings for remaining nodes. In Fig. 2.1, we plot the Micro-F1 score (averaged over 50 runs) versus d and λ respectively. Except for the parameter being tested, the other parameters are set as default values.

Fig. 2.1(a) shows our approach is relatively robust to the representation dimension d on Cora and Pubmed. On Citeseer and Wiki, our approach exhibits very stable performance when d is large enough. Moreover, when d is small, it has relatively low compact on the performance of our approach. In addition, Fig. 2.1(b) indicates our approach is insensitive with respect to λ across all datasets.



(a) TADW (b) DANE (c) STNE

(d) DGI (e) GAE (f) Ours

Figure 2.2: Visualizing node embeddings of Citeseer using t-SNE. Different colors mark different classes.

2.5.7 Visualization of Node Embeddings

Visualizing a graph in a two-dimensional space is an important application of graph embedding. Hence, we use t-SNE algorithm [41] to visualize embeddings of Citeseer learned by our approach (see Fig. 2.2(f)). For a fair comparison, we also plot corresponding embeddings of four strong baselines (see Fig. 2.2(a-e)). In Fig. 2.2 each document is mapped to a point and different colors mark different classes. For all baselines, their mapped points of different classes are still mixed with each other. On the other hand, our result shows a reasonably good separation of classes, which indicates the high-quality of our embeddings.

2.6 Related Works

2.6.1 Non-attributed Graph Embedding as Matrix Factorization

A large number of graph embedding methods can be viewed as matrix factorization. They aims to learn embeddings in the low embedding space by preserving the node proximity in the original graph. Many classical graph embedding approaches aim to preserve the local pairwise (first-order) proximity, such as Locally Linear Embedding [42], Laplacian eigenmaps [43], Hessian eigenmaps [44]. Besides the first-order proximity, to learn the richer graph structure, some approaches [17, 45] consider the second-order proximity, which is defined by the number of neighbors shared by two nodes. Moreover, a number of recent methods [7, 8, 22] explore the global graph structure, which is determined by the k-step ($k \geq 3$) relations between nodes.

2.6.2 Attributed Graph Embedding

A variety of recent advances focus on attributed graph embedding, which attempts to enhance embeddings by incorporating node attributes. These approaches fall into two categories: matrix factorization based approaches and neural network

based approaches. The former approaches view the attributed graph embedding as a matrix factorization problem, such as TADW [10], HSCA [20]. On the other hand, feeding adjacency matrix and attribute matrix, neural network based approaches aim to use different types of neural networks to learn more informative embedding, such as GraphSage [18], STNE [35], GDI [36], which will be introduced in the next chapter.

2.6.3 Inductive Matrix Completion

Our approach is inspired by the inductive matrix completion (IMC) model [34]. whose goal is to approximately recover a low-rank matrix based on the observed entries. Typically, it can be applied to the problem of the recommendation system, which aims to learn the relevance between queries and items, for example user-movie rating. IMC is a standard approach for this problem when attribute vectors of queries and items are available. Furthermore, IMC has been applied to multi-label classification [46], gene-disease prediction [47], link prediction [48], semi-supervised clustering [49], and attributed graph embedding [10, 20].

2.7 Conclusion

In this chapter, we presented a novel attributed graph embedding approach which can be viewed as a low-rank matrix factorization framework. Moreover, we propose a novel algorithm to optimize the corresponding objective. Theoretically, we prove that the objective is convex and our algorithm can achieve the globally optimal solution efficiently. The experiments suggest that our approach can learn more informative node embeddings than several strong baselines on four benchmark datasets.

Part II

Graph-based Semi-supervised Learning Approaches

CHAPTER 3

A SIMPLE GRAPH-REGULARIZATION BASED SEMI-SUPERVISED LEARNING APPROACH FOR IMBALANCED CLASSIFICATION

3.1 Introduction

In part I, we have considered the graph embedding problem, which does not take use of the information of labels. In this part, we will introduce a graph-regularization based semi-supervised learning (GRSSL) method which aims to learn graph structure and node labels jointly.

Over the last 20 years, GRSSL approaches have attracted great attention due to their effectiveness for classification and elegance in mathematics [50, 51, 52]. Hence, they have been widely applied to various real-world tasks, including compute vision [53, 54], natural language processing [55] and network analysis [56, 57]. Their key assumption is nearby nodes in the graph are likely to share the same label, which usually can be framed as an optimization problem or a label propagation process. The former defines a loss function to be the tradeoff between the graph Laplacian constraint and the initial label fitting constraint on a smooth classification function [58, 59, 60], which can be further learned by propagating label information from labeled nodes to unlabeled ones in an iterative fashion [61, 62].

However, aforementioned GRSSL approaches can be sensitive to class imbalance, since graph Laplacian constraints of them do not provide sufficient flexibility in dealing with a wide range of class imbalance [63, 64]. For instance, from the perspective of graph partition, normalized graph Laplacian constraint encourages balanced partition results [65]. To mitigate this problem, various GRSSL methods have been

proposed to utilize the class proportion information. In [13], the GRSSL method based on Gaussian Fields and Harmonic Functions (GFHF) can incorporate the class proportion knowledge to adjust the classification result. And class proportion can be directly used as a regularizer to improve the discriminative power in [66]. Additionally, in [67], the proposed method can incorporate the prior knowledge of class proportion from a graph max-cut perspective. However, these approaches suffer from the drawback that they need prior knowledge about class proportion.

In this chapter, we present a simple GRSSL approach to jointly take into account graph structures and class imbalance. Our approach provides a strategy to estimate class proportion from a dynamical perspective based on random walks on the graph. It introduces an imbalance parameter for classification such that the classification result will become more balanced with respect to the class size as the parameter increases. Moreover, as we will show in Section 3.5.1, our approach can also adjust the affinities among local neighbors, which makes it insensitive with respect to the bandwidth of the Gaussian kernel defined in Section 3.2. The experiments show that our approach can deal with various class imbalance for real-world classification tasks.

Our approach also has interesting connections to other areas: (1) it can be derived as an optimization problem, which provides a novel insight of our approach; (2) it is striking that the loss function of our approach introduces the continuous-time Markov stability for the graph naturally; (3) it also can be viewed as the group inverse of the normalized Laplacian matrix.

In summary, our main contributions are as follows:

- We introduce a unified framework of many existing GRSSL approaches.
- We propose simple GRSSL approach, which can be used to handle datasets with different class imbalance.

- Our proposed approach has interesting connections to several areas, which verifies our motivation.
- Our experiments indicates that our approach can outperform several strong baselines on four real-world datasets with different class imbalance.

3.2 Preliminaries

For a C-class classification problem, given a small set of labeled data $\mathbf{X}_l = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ with $y_i \in \{1, \ldots, C\}$ and a large set of unlabeled data $\mathbf{X}_u = \{x_{l+1}, \ldots, x_n\}$, the goal of GRSSL is to learn labels for unlabeled data from both of them. It starts with constructing a k-Nearest-Neighbor (kNN) graph whose corresponding affinity matrix is given by the Gaussian kernel with bandwidth σ,

$$A_{ij} = \begin{cases} exp(-\frac{d(x_i,x_j)^2}{2\sigma^2}) & \text{if } i \in \mathcal{N}_j \text{ or } j \in \mathcal{N}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

where $d(x_i, x_j)$ is the distance between nodes i and j, $\mathcal{N}_i$ is the index set of the kNN of node i.

We start by reviewing a popular standard method LGC [58], which can be interpreted as label propagation. It assumes that similar data points are very likely to share the same label. Thus, the key idea is to define a good similarity measure between data points. Interestingly, this similarity can be learned by a propagation process, which propagates labels from labeled nodes to unlabeled nodes through the entire graph. This procedure is also adopted by many GRSSL methods [13, 61, 68].

To describe this propagation process formally, we first define a propagation matrix $S = D^{-1/2}AD^{-1/2}$, where $D = diag(d_1, d_2, \ldots, d_n)$ and $d_i = \sum_j A_{ij}$. Then, we create a n-by-C indicator matrix Y, such that $Y_{ij} = 1$ if $y_i = j$ otherwise $Y_{ij} = 0$.

That is, Y is consistent with initial labels. The label propagation of LGC is the iteration algorithm to calculate a n-by-C soft label matrix F as follows,

$$F^{(m+1)} = \alpha S F^{(m)} + (1-\alpha)Y. \tag{3.2}$$

The algorithm repeats Eq. 3.2 until $\{F^{(m)}\}$ converges. In each iteration, every point receives the label information from its neighbors and ensures the initial label information. The parameter α controls the balance between these two kinds of information. To make sure this iteration process converge, α should be in $(0, 1)$.

Like many other GRSSL methods [59, 69, 60], LGC can also be interpreted as an approach to estimate a soft (smooth) label function F on the graph. F is supposed to satisfy two constraints: 1) it should be smooth on the whole graph, and 2) it should retain given labels on the labeled points. A regularization framework was proposed to learn F under these two constraints,

$$\min_F \frac{1}{2}\sum_{c=1}^{C}\sum_{i,j}^{n} A_{ij}(\frac{F_{ic}}{\sqrt{d_i}} - \frac{F_{jc}}{\sqrt{d_j}})^2 + \rho\sum_{c=1}^{C}\sum_{i}^{n}(F_{ic} - Y_{ic})^2, \tag{3.3}$$

where the first term encourages the smoothness of the function, and the second term penalizes the difference between initial labels and learned labels. The hyper-parameter ρ makes a tradeoff between them. Hence the optimal soft label function is $F^* = \arg\min_F J(F)$. Fortunately, $J(F)$ is convex, we can obtain the closed form solution easily by taking the derivative of $J(F)$ with respect to F.

3.3 A Unified Framework of GRSSL

To provide more in-depth understanding to the difference and connection among GRSSL approaches, in this section we will provide a unified framework of many main-

Table 3.1: Unifying various GRSSL algorithms as generalized solutions to a linear system $LF = \Sigma Y$. These solutions can be expressed as $F^* = (L + Q)^{-1}\Sigma Y$ with different choices of L, Σ and Q. Notation is as follows: ρ, η, μ_1, μ_2 are all hyper-parameters. Λ is a diagonal matrix whose diagonal entries are hyper-parameters. In the last column, $L^{\{i,j,\ldots,k\}}$ denotes the generalized inverse of L, which is defined in Appendix A.1.

Methods	L	Σ	Q	Generalized solutions: F^*	Generalized inverses
GFHF [13]	$I - \widetilde{P}$	I	J	$(I - \widetilde{P} + J)^{-1}Y$	$(I - \widetilde{P})^{\{1,3,4,5\}}$
LGC [58]	$I - S$	ρI	ρI	$\rho(I - S + \rho I)^{-1}Y$	$(I - S)^{\{3,4,5\}}$
MR [59]	$D - A$	ηI	ηJ	$\eta(D - A + \eta J)^{-1}Y$	$(D - A)^{\{3,4,5\}}$
LPQC [70]	$D - A$	$\mu_1 I$	$\mu_2 I$	$\mu_1(D - A + \mu_2 I)^{-1}Y$	$(D - A)^{\{3,4,5\}}$
PARW [61]	$D - A$	Λ	Λ	$(D - A + \Lambda)^{-1}\Lambda Y$	$(D - A)^{\{3,4,5\}}$
Our approach	$I - P$	D^{-1}	G	$(D - A + DG)^{-1}Y$	$(I - P)^{\{1,2,5\}}$

stream GRSSL baselines [13, 58, 59, 70, 61], which can be viewed as generalized solutions to a linear system,

$$LF = \Sigma Y, \tag{3.4}$$

where Σ is a diagonal matrix with nonnegative entries and L can be any following Laplacian matrices: (1) $L_{un} = D - A$, (2) $L_{rw} = I - P$ and (3) $L_{sym} = I - S$ [65], where $S = D^{-1/2}AD^{-1/2}$. However, it is easy to verify that L is singular for all methods. For instance, since 1 is the largest eigenvalue of S with non-zero eigenvector, $I - S$ is singular. Hence, to solve Eq. 3.4, we need to find the generalized solution. Considering some specific matrix Q, we modify L to $L + Q$ which is nonsingular. Naturally, $(L + Q)^{-1}$ can be regarded as the generalized inverse of L. Then the generalized solution to Eq. 3.4 is

$$F^* = (L + Q)^{-1}\Sigma Y. \tag{3.5}$$

Table 3.1 summarizes that many existing GRSSL baselines including our approach can be viewed as generalized solutions to Eq. 3.4. That is, all methods in

Algorithm 2 Iteration algorithm of proposed approach

Input: Affinity matrix A defined in Eq. 3.1 with $A_{ii} = 0$; initial label matrix Y; hyper-parameter γ.

Output: Label y_i for each point x_i.

1: Calculate the following matrices: diagonal matrix $D = diag(d_1, d_2, \ldots, d_n)$ with $d_i = \sum_j A_{ij}$; transition matrix $P = D^{-1}A$; rank-one matrix $G = \mathbb{1}\pi^T$,where $\mathbb{1}$ is an all-ones vector and $\pi = (\frac{d_1}{d}, \frac{d_2}{d}, \ldots, \frac{d_n}{d})$ with $d = \sum_i d_i$.

2: Initialize $F^{(0)}$ with $D^{-1}Y$.

3: **while** not converge **do**

4: $\quad F^{(m+1)} = PF^{(m)} - \gamma GF^{(m)} + \gamma D^{-1}Y$.

5: **end while**

6: Assign each point x_i to class $y_i = \arg\max_j F^*_{ij}$, where $F^* = \lim_{m\to\infty} F^{(m)} = \gamma(D - A + \gamma DG)^{-1}Y$.

Table 3.1 can be viewed as special cases of Eq. 3.4 with different choices of L, Σ and Q. In Appendix A.2, we use GFHF as an example to verify the corresponding expressions in Table 3.1.

3.4 Algorithm

Our approach for GRSSL is presented in Algorithm 2, whose intuition is as follows: in each iteration, every node obtains three types of label information. The first term on the right side in step 4 is the information from its neighbors. The second term means it loses information to all nodes proportional to their degrees, which will make it less similar to nodes in large-size classes. To illustrate this point, we consider the label information each node loses to every class. For simplicity, we consider the extreme case, in which F is an indicator matrix with $F_{ic} \in \{0, 1\}$ where 1 denotes

node i belongs to class c. According to the second term, node i loses information $\gamma \sum_i \frac{d_i}{d} F_{ic}$ to the nodes in class c, which can be viewed as an estimated quantity proportional to the size of class c. Note that, the estimation will be more accurate when F is closer to Y. This means each node will become less similar to nodes in large-size classes. In other words, if γ is large enough, the size of large class will decrease gradually as the label propagates. Note that, in the case that F is a soft label matrix, $\gamma \sum_i \frac{d_i}{d} F_{ic}$ not only contains the knowledge of the class size but also the graph structure. The third term retains the initial label scaled by D^{-1}. In summary, Algorithm 2 can not only propagate label information through local affinity but also take into account the class imbalance. The hyper-parameter γ of second term trades off the information gained from neighbors against that lost to nodes in each class. Specifically, γ acts as an imbalance degree parameter: the larger the parameter γ is, the more balanced the classification result is.

Before proving the convergence of Algorithm 2, we introduce the following lemma.

Lemma 3.4.1. *Let P be the transition matrix for a regular Markov chain* [1]*, and let G be the limiting (rank one) matrix $G = \lim_{m\to\infty} P^m$. Then*

$$(P - \gamma G)^m = P^m - G + (1-\gamma)^m G, \tag{3.6}$$

where γ is a non-zero constant.

[1]Here, we assume W is a primitive matrix. We will show later that our approach is still valid when W is not primitive.

Proof. Since P is the transition matrix for the regular Markov chain, it has two properties: (1) $PG = P\mathbb{1}\pi^T = G$, and (2) $G^m = (\mathbb{1}\pi^T)^m = G$ for all positive integers m. Applying these two facts, we obtain

$$\begin{aligned}
(P-\gamma G)^m &= \sum_{i=0}^{m}(-\gamma)^i\binom{n}{i}P^{m-i}G^i \\
&= P^m + \sum_{i=1}^{m}(-\gamma)^i\binom{m}{i}G^i \\
&= P^m + \sum_{i=1}^{m}(-\gamma)^i\binom{m}{i}G \\
&= P^m + (\sum_{i=0}^{m}(-\gamma)^i\binom{m}{i} - 1)G \\
&= P^m - G + (1-\gamma)^m G.
\end{aligned}$$

□

Specifically, setting $\gamma = 1$, we have $(P-G)^m = P^m - G$ [71]. We now turn to the proof of the convergence of Algorithm 2.

Theorem 3.4.1. *The sequence* $\{F^{(m)}\}$ *converges to* $F^* = \gamma(D - A + \gamma DG)^{-1}Y$*, for any* $\gamma \in (0, 2)$.

Proof. We set $F^{(0)} = Y$, then

$$F^{(m)} = \sum_{i=0}^{m}(P-\gamma G)^i D^{-1}Y. \tag{3.7}$$

According to Lemma 3.4.1 with $\gamma \in (0,2)$, $\lim_{m\to\infty}(P-\gamma G)^m = \lim_{m\to\infty}[P^m - G + (1-\gamma)^m G] = \mathbf{0}$, where $\mathbf{0}$ is a zero matrix. Thus $(I - P + \gamma G)^{-1}$ exists. In addition, we have

$$F^* = \lim_{m\to\infty} F^{(m)} = \gamma(D - A + \gamma DG)^{-1}Y. \tag{3.8}$$

□

Note that, in the case that the graph is connected, but W is not primitive, the Markov chain does not have property that $\lim_{m\to\infty} P^m = G$. However, as we will show

later, $I - P + \gamma G$ with $\gamma \neq 0$ is still nonsingular [71], which means Eq. 3.8 is always valid when the graph is connected.

3.5 Interpretation and Connections

One of striking properties of our approach is that it can be derived from different perspectives. In this section, we briefly present these different viewpoints which provide us novel insights into our approach.

3.5.1 Regularization Framework

Our approach can be derived from the following regularization framework,

$$\begin{aligned} \min_F \frac{1}{2} \sum_{c=1}^{C} \sum_{i,j}^{n} A_{ij}(F_{ic} - F_{jc})^2 - \frac{\gamma}{2} \sum_{c=1}^{C} \sum_{i,j}^{n} \frac{d_i d_j}{d}(F_{ic} - F_{jc})^2 \\ +\gamma \sum_{c=1}^{C} \sum_{i}^{n} (d_i^{\frac{1}{2}} F_{ic} - d_i^{-\frac{1}{2}} Y_{ic})^2, \end{aligned} \tag{3.9}$$

where the first term encourages smoothness of the function on the graph. And the second term pushes labels of high-degree nodes apart from each other, since it emphasizes the terms with the large value of $\frac{d_i d_j}{d}$. This means high-degree nodes tend to be divided evenly among all classes to balance the class size. Moreover, the quantity $\sum_{c=1}^{C} \sum_{i,j}^{n} \pi_i \pi_j (F_{ic} - F_{jc})^2$ is also known as the Gimi-Simpsin diversity index [72, 73], which is large when the classification result has many classes of close size, and is low when the classification result has few and uneven classes. In our case, since the number of classes is fixed, the diversity index is maximum for the classes with equal size. This also means the second term can control the imbalance of classification results by tuning the value of γ: when γ is very large, and $\frac{\gamma}{2} \sum_{c=1}^{C} \sum_{i,j}^{n} \frac{d_i d_j}{d}(F_{ic} - F_{jc})^2$ is the dominant term, $\sum_{c=1}^{C} \sum_{i,j}^{n} \frac{d_i d_j}{d}(F_{ic} - F_{jc})^2$ will become large to minimize the objective. As a result, the classification result will be balanced. On the other hand, when

γ is small, and the term $\frac{\gamma}{2}\sum_{c=1}^{C}\sum_{i,j}^{n}\frac{d_i d_j}{d}(F_{ic}-F_{jc})^2$ is not emphasized, minimizing the objective will allow imbalanced classification result. The third term penalizes the difference between learned labels and given labels, which are scaled by $D^{\frac{1}{2}}$ and $D^{-\frac{1}{2}}$ respectively.

Moreover, our approach is insensitive with respect to the Gaussian kernel width σ. Let us consider the local and global relationships between nodes, respectively. The first two terms of our loss function in Eq. 3.9 can be written as

$$\frac{1}{2}\sum_{c=1}^{C}\sum_{i\sim j}(A_{ij}-\gamma\frac{d_i d_j}{d})(F_{ic}-F_{jc})^2-\frac{\gamma}{2}\sum_{c=1}^{C}\sum_{i\nsim j}\frac{d_i d_j}{d}(F_{ic}-F_{jc})^2, \tag{3.10}$$

where $i\sim j$ means nodes i and j are mutual k-nearest neighbors, $i\nsim j$ otherwise. The first term in Eq. 3.10 means this framework can adjust the local affinity between each node and its mutual kNN automatically. The higher the degree of its neighbor is, the more affinity between them will be reduced. When σ is small, the local affinities between each node and its neighbors are highly skewed since they are calculated by the Gaussian kernel. Therefore, the high-affinity edge has high probability to attach high-degree nodes, because if A_{ij} is large, d_i and d_j are more likely to be large. Then, high affinities will be reduced more than small ones to balance the local affinity distribution among neighbors. This means, as we will show in our experiments, our approach can be insensitive with respect to σ.

Since the loss function in Eq. 3.9 is convex, we can obtain the closed form solution by setting its first derivative to zero. We now differentiate our loss function with respect to F,

$$(D-A)F^*-\gamma(D-DG)F^*+\gamma DF^*-\gamma Y=0,$$

which can be written as

$$(D-A+\gamma DG)F^*=\gamma Y.$$

Finally, we have

$$F^* = \gamma(D - A + \gamma DG)^{-1}Y,$$

which is the closed form expression of our iteration algorithm.

3.5.2 Linearisation of Markov Stability at Short Time

Our approach can be explained as the linearisation of Markov stability at short time. We now ignore the label information and consider a graph partition problem. First we introduce a finite continuous-time Markov process on the kNN graph. According to [74], the corresponding transition matrix is defined as

$$p(t) = e^{-t(I-P)}.$$

The entry P_{ij} is the probability that the process moves from state i to state j. From this Markov viewpoint, a good partition should be: a random walker should stay in a component for a given time scale before escaping it. This observation leads to the definition of continuous-time Markov stability [73],

$$r(t, F) = Tr[F^T(\frac{D}{d}e^{-t(I-P)} - \pi\pi^T)F], \tag{3.11}$$

where $\pi_i(e^{-t(I-P)})_{ij}$ is the joint probability that the walker visits node i at time 0 and j at time t at stationarity, and $\pi_i\pi_j$ is the joint probability of finding a first walker on node i at time 0 and a second independent walker on node j at time t. $\pi\pi^T$ is known as the null model, which serves as the benchmark to display some non-trivial features, such as the component structure in the graph.

However, in practice, calculating the exponential of the Laplacian $L = D - W$ for a large graph can be computationally expensive. Therefore, it is natural to study the first-order approximation [75] in the limit of small times, $t \to 0$,

$$\begin{aligned} r(t,F) &= r(0,F) + t\frac{dr(t,F)}{dt}\Big|_{t=0} \\ &= \frac{1}{d}(Tr[F^T(D - d\pi\pi^T)F] - t\,Tr[F^T L F]). \end{aligned} \tag{3.12}$$

Given this, the optimal graph partition can be obtained by maximizing the the linearized Markov stability $r(t,H)$, for which we further have

$$\max_F r(t,F) \Leftrightarrow \min_F \; t \text{ tr}\left(F^T L F\right) - \text{tr}\left[F^T(D - d\pi\pi^T)F\right]. \tag{3.13}$$

After setting $\gamma = \frac{1}{t}$, Eq. 3.13 becomes first two terms in our regularization framework. Therefore, from the perspective of Markov stability, our approach can be interpreted as follows: the random walker starting from unlabeled node is more likely to stay in the same community given time scale t, and the walker starting from labeled node has higher probability staying at the initial node. If this probability is equal to one, this node becomes the absorbing state. Over short time, the process is dominated by the null model so, as we explained above, it encourages equal-size classes. And as t increases, the Markov process explores larger regions with various sizes of the graph. Thus, from the perspective of classification, the Markov time scale acts as an intrinsic parameter to uncover the graph structure at different class imbalance.

3.5.3 Group Inverse of $I - P$

Our approach can be derived from the group inverse of $I - P$. It is known that GRSSL can be viewed as the solution to a linear system [61]. Now we consider a linear system as follows,

$$I - P = D^{-1}F. \tag{3.14}$$

Since 1 is the largest eigenvalue of P with nonzero eigenvector, $I - P$ is singular. Our approach can be derived from the group inverse of $I - P$, which is a generalized solution of Eq. 3.14.

Let us first define the group inverse of $I - P$. We use $\{\lambda_i\}$ to denote the eigenvalues of P, and order them decreasingly $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n$. Then the spectral decomposition of $I - P$ is

$$I - P = \sum_{i=2}^{n} (1 - \lambda_i) u_i v_i^T, \tag{3.15}$$

where u_i and v_i^T are the right and left eigenvectors of $I - P$ corresponding to 1-λ_i respectively. According to [76], the group inverse of $I - P$ is defined as

$$(I - P)^{\#} = \sum_{i=2}^{n} \frac{1}{(1 - \lambda_i)} u_i v_i^T. \tag{3.16}$$

According to this definition, it is easy to prove $(I - P + \gamma G)$ with $\gamma \neq 0$ is nonsingular. Furthermore, we have the following theorem.

Theorem 3.5.1. *$(I - P)^{\#} = (I - P + \gamma G)^{-1} - \frac{1}{\gamma} G$, for any $\gamma \neq 0$.*

Proof. Since

$$G = \lim_{m \to \infty} P^m \quad = U \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} U^{-1}, \tag{3.17}$$

where $U = [u_1, \cdots, u_n]$ and $U^{-1} = [v_1, \cdots, v_n]$, we have

$$I - P + \gamma G \quad = U \begin{bmatrix} \gamma & 0 & \ldots & 0 \\ 0 & 1 - \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 - \lambda_n \end{bmatrix} U^{-1}. \tag{3.18}$$

Thus,

$$
\begin{aligned}
&(I - P + \gamma G)^{-1} \\
&= U \begin{bmatrix} \frac{1}{\gamma} & 0 & \dots & 0 \\ 0 & \frac{1}{1-\lambda_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{1-\lambda_n} \end{bmatrix} U^{-1} \\
&= U \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \frac{1}{1-\lambda_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{1-\lambda_n} \end{bmatrix} U^{-1} + U \begin{bmatrix} \frac{1}{\gamma} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} U^{-1} \\
&= (I - P)^{\#} + \frac{1}{\gamma} G.
\end{aligned}
\tag{3.19}
$$

□

It has already been proven in Theorem 3.5.1 that, the closed-form solution to our approach is given by $F^* = \gamma(D - A + \gamma DG)^{-1}Y$. By using Theorem 3.5.1 above, we further have

$$F^* = \gamma \tilde{L}^{\#} F^{(0)} + \frac{1}{d} \mathbb{1}\mathbb{1}^T Y, \tag{3.20}$$

with $F^{(0)} = D^{-1}Y$ being the initialization of our approach (see Algorithm 2), and

$$\mathbb{1}\mathbb{1}^T Y = \begin{bmatrix} \sum_i Y_{i1} & \sum_i Y_{i2} & \dots & \sum_i Y_{in} \\ \sum_i Y_{i1} & \sum_i Y_{i2} & \dots & \sum_i Y_{in} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_i Y_{i1} & \sum_i Y_{i2} & \dots & \sum_i Y_{in} \end{bmatrix}.$$

As implied by Eq. 3.20, the significance of Theorem 3.5.1 is to enable us to decouple our closed-form solution into two terms, allowing for an intuitive interpretation of our approach. Specifically, the first term is usually called graph filtering [77, 57], which encodes the basic label propagation procedure on the graph, similar to general

GRSSL [13, 58]. And the second term actually adds a constant bias $\bar{y}_c = \frac{1}{d}\sum_i Y_{ic}$, which is proportional to the number of labeled samples for the c-th class, to the c-th column of the soft label matrix obtained by the first term. It can therefore play a role of retaining all the classes in the label propagation results, or otherwise the c-th class will vanish if the c-th column of the obtained label matrix all has very low values (small classes usually tend to vanish in the case of imbalanced data).

To sum up, by resorting to the concept of group inverse, the solution given by our approach can be decoupled into two terms with intuitive interpretation, i.e., a graph filtering term that encodes the label propagation procedure and a bias term that prevents all the classes from vanishing during label propagation.

3.6 Experiments

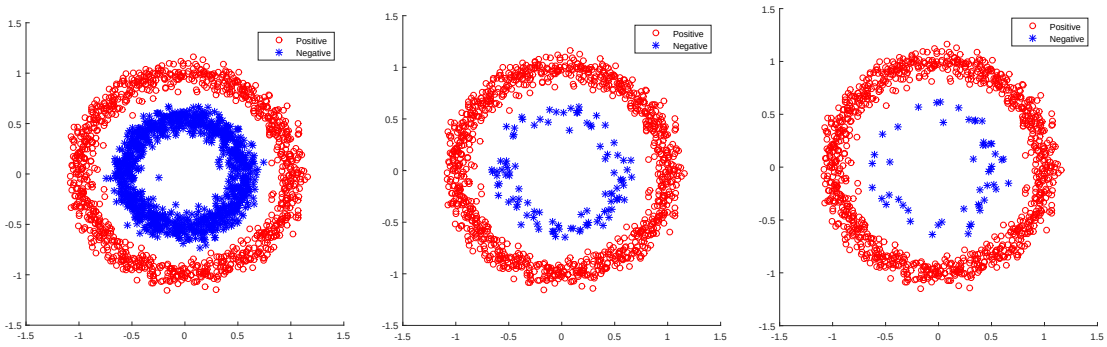
In this section, experiments are conducted on semi-supervised classification tasks on both synthetic and real-world datasets.

3.6.1 Experiments on Synthetic Datasets

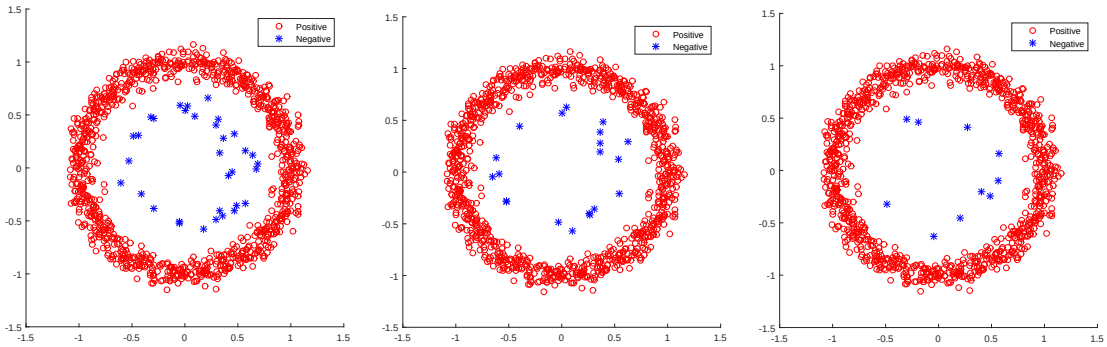
Table 3.2: Classification errors of individual classes on two-circles patterns with different imbalance ratios. Pos. and Neg. represent positive and negative classes respectively.

Imbalance ratio	1		20		40		60		80		100	
Class name	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
LGC [58]	**0.267**	**0.149**	0.085	0.884	0.097	0.850	0.109	0.826	0.138	0.878	0.153	0.784
Ours	0.301	0.243	**0.021**	**0.635**	**0.017**	**0.607**	**0.007**	**0.605**	**0.019**	**0.731**	**0.027**	**0.706**

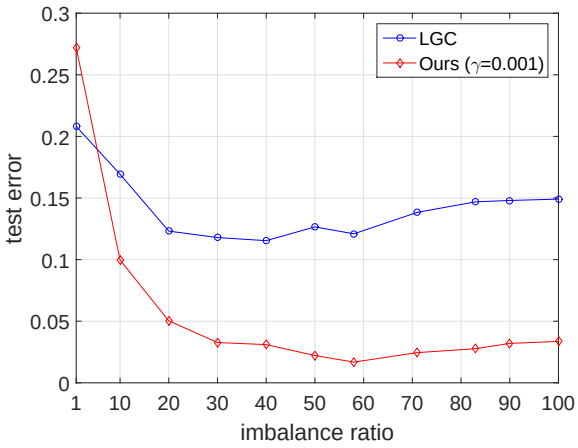
To intuitively illustrate the effectiveness of our approach on imbalanced patterns, synthetic datasets (two-circles patterns) with different imbalance ratios were generated (see Figs. 3.1(a)- 3.1(f)). The number of positive data was fixed to 1000,



(a) Imbalance ratio=1 (b) Imbalance ratio=10 (c) Imbalance ratio=20



(d) Imbalance ratio=30 (e) Imbalance ratio=50 (f) Imbalance ratio=100



(g) The classification error versus the imbalance ratio on two-circles patterns.

Figure 3.1: (a-f) Ideally classified two-circles patterns with imbalance ratio ({size of positive class}/{size of negative class}) ranging from 1 to 100; (g) classification results of LGC and our approach on two-circles patterns with different imbalance ratios.

and the number of negative data was set from 1000 to 10. As a result, the imbalance ratio varies from 1 to 100. For each synthetic dataset, a kNN graph with $k = 20$ was constructed. Given 6 labeled data points, we applied LGC and our approach ($\gamma = 0.001$) to classify each pattern. For both methods, every test error is averaged over 50 random trials. In each trial every class includes at least a labeled point, otherwise we redo the random sampling.

Generally, our approach consistently outperforms LGC on all datasets except balanced one. Fig. 3.1(g) shows LGC can outperform our approach on balanced dataset (imbalance ratio = 1), since $\gamma = 0.001$ is too small for balanced datasets. When the imbalance ratio increases, our approach will become better with imbalance ratios smaller than 60. Note that, LGC also becomes better, because the dataset size becomes smaller and the number of labeled data is fixed. In addition, on datasets with higher imbalance ratios (lager than 60), our approach becomes worse but still outperforms LGC. Moreover, since some synthetic datasets are highly imbalanced, we also report classification errors of individual classes on selected datasets (see Table 3.2). For each class, the best result is boldfaced. In Table 3.2, except the balanced dataset, our approach outperforms LGC for each class.

3.6.2 Experiments on Real-world Datasets

Our approach is also evaluated on four real-world tasks with different class imbalance: natural image classification, hand-written digit recognition, text classification, cross-media learning. Dataset statistics are shown in Table 3.3. The datasets and hyper-parameter settings for them are described as follows.

- **Stanford Dogs**: The Stanford Dogs dataset [78] contains natural images of 120 categories of dogs, where 10 categories were selected (*Maltese dog, Afghan hound, Irish wolfhound, Airedale, Bernese mountain dog, basenji, pug, Leonberg,*

Table 3.3: Dataset statistics

Name	# examples	dimension	# classes	k	$d(x_i, x_j)^2$
Stanford Dogs	2180	9216	10	15	$\|\|x_i - x_j\|\|^2$
USPS	9298	256	10	20	$\|\|x_i - x_j\|\|^2$
20 Newsgroups	1738	8014	4	15	$1 - \langle x_i, x_j \rangle / \|\|x_i\|\| \|\|x_j\|\|$
Wikipedia	2866	138	10	15	$\|\|x_i - x_j\|\|^2$

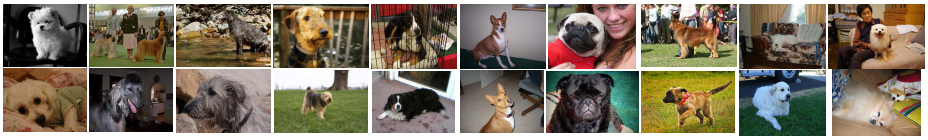


Figure 3.2: Sample images from our subset of Stanford Dogs dataset.

Great Pyrenees, Pomeranian). The class size varies from 200 to 252 (see Fig. 3.2). After applying pre-trained AlexNet [79] on ImageNet [80] to extract features, each image was transformed to a 9216-dimensional vector. Then, AlexNet features were scaled by l_1 norm. No further preprocessing was done.

- **USPS**: The USPS dataset [81] contains 9298 gray handwritten 16x16 digits in 10 classes (see Fig. 3.3). The number of digits per class varies from 708 to 1553, which makes the dataset imbalanced.
- **20 Newsgroups**: The 20 Newsgroups dataset [82] consists of approximately 20,000 articles divided into 20 different groups. Each article is represented by a 61188 dimensional sparse vector. A subset of the topic *rec* was selected. It contains *autos, motorcycles, baseball*, and *hockey*. The articles were preprocessed following [58]. Then feature vectors were normalized into TF-IDF representations. To show the effectiveness of our approach to deal with the class imbalance, a highly imbalanced subset of the preprocessed dataset was selected. The class size of each class is as follows: 988 (*autos*), 500 (*motorcycles*), 200 (*baseball*), 50 (*hockey*).

Figure 3.3: Sample images from USPS dataset.

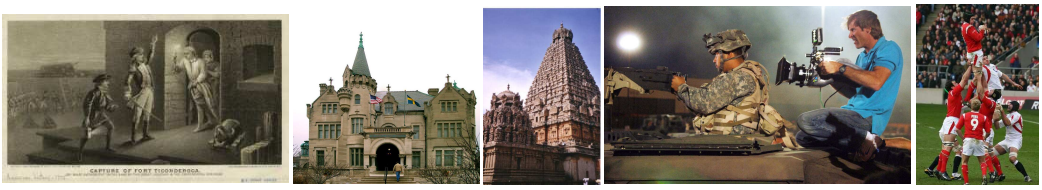


Figure 3.4: Sample documents (image+text) from Wikipedia dataset.

- **Wikipedia**: The Wikipedia dataset [83] is a widely-used dataset for cross-media retrieval (see Fig. 3.4). It consists of 2866 featured documents of 10 classes, each including a single image and at least 70 words. Following [84], the features of each document were extracted by the bag-of-words encoder for text and the SIFT descriptor for the image. The class size of each class is as follows: 172 (*art*), 360 (*biology*), 340 (*geography*), 333 (*history*), 267 (*literature*), 236 (*media*), 237 (*music*), 185 (*royalty*), 285 (*sport*), 451 (*warfare*).

In all experiments, when we talk about 20 Newsgroups and Stanford Dogs datatsets, we always refer to their subsets reported in Table 3.3. Note that, besides two imbalanced datasets, the relatively balanced one (Stanford Dogs) was also adopted, since we believe that an ideal GRSSL approach should also work well on balanced data. Otherwise, if a method can only perform well in highly imbalanced data but not on balance data, it will be very limited in practice since the level of class imbalance is

Table 3.4: Hyper-parameter settings with respect to different datasets

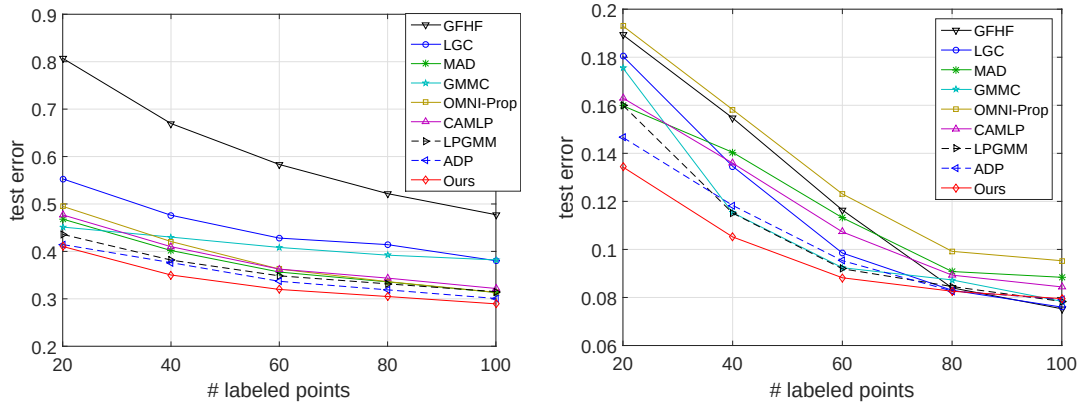
Method	GFHF [13]	LGC [58]		MAD [69]				GMMC [67]		OMNI [85]		CAMLP [68]		LPGMM [86]		ADP [84]			Ours
	σ	α	σ	μ_1	μ_2	μ_3	σ	μ	σ	λ	σ	β	σ	α	U	α	β	θ	γ
Stanford Dogs	0.005	0.99	0.005	1	10	10	0.015	0.01	0.015	10	0.02	0.1	0.035	1	30	0.99	0.99	0.01	1
USPS	1.25	0.99	1.25	1	10	1	2.75	0.01	1.25	1	4.75	1	3.25	1	30	0.99	0.99	0.01	-0.01
20 Newsgroups	0.15	0.99	0.15	1	1	10	0.75	0.01	0.5	0.1	0.3	10	0.3	1	30	0.99	0.99	0.01	0.3
Wikipedia	0.1	0.99	0.05	10	10	10	4.5	0.01	0.5	1	3.25	1	3.75	1	30	0.99	0.99	0.01	0.5

typically unknown. According to our analysis in Sections 3.4 and 3.5, our approach with large γ can deal with balanced datasets.

3.6.2.1 Baselines

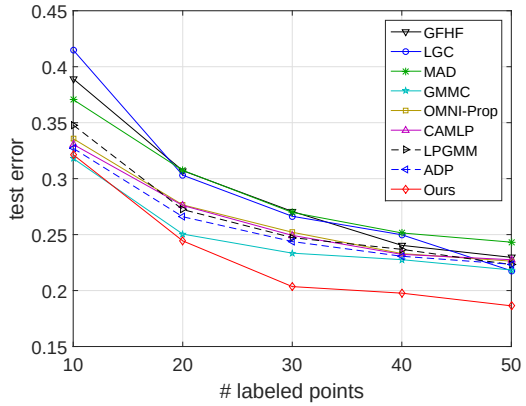
On each dataset, our approach is compared against two standard baselines of GRSSL: Gaussian Fields and Harmonic Functions (GFHF) coupled with the Class Mass Normalization (CMN) [13] and Local and Global Consistency (LGC) [58]. In addition it is further compared against the state-of-the-art methods: Modified Adsorption (MAD) [69], greedy gradient Max-Cut (GMMC) [67], OMNI-Prop [85], CAMLP [68], LPGMM [86] and ADP [84]. Moreover, from the perspective of imbalanced classification, aforementioned baselines can be categorized into two groups. LGC, MAD, CAMLP, LPGMM and ADP are standard baselines of GRSSL, which do not take into account the class imbalance. And, GFHF, GMMC and OMNI-Prop can adapt to imbalanced datasets, since they can incorporate class proportion estimated from labeled data.

For fair comparison, all methods use their optimal parameters, which are summarised in Table 3.4. Note that, LPGMM [86] adopts geodestic distance for graph construction and ADP [84] takes an adaptive scheme to set the Gaussian kernel bandwidth, so they do not have the parameter σ. And for the other parameters in these two methods, we use the default settings suggested by the authors in the original papers, which we experimentally observed are insensitive across different datasets,

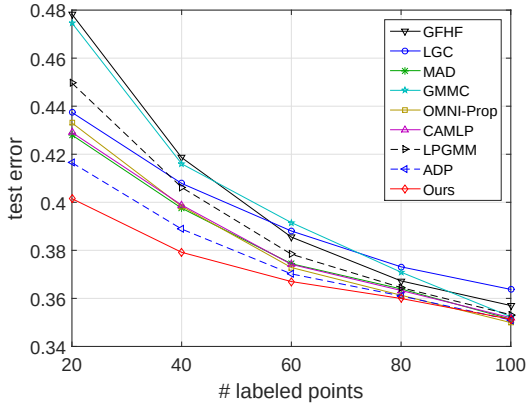


(a) Stanford Dogs

(b) USPS



(c) 20 Newsgroups



(d) Wikipedia

Figure 3.5: The classification error versus the number of labeled points on different datasets.

consistent with the claims of the authors [86, 84]. For each setting of the number of labeled points, we repeat for 50 trials by using the same random sampling strategy as the experiments on synthetic datasets above, and report the average test error.

3.6.2.2 Classification Results

Fig. 3.5 shows the performance in terms of test error versus the number of labeled points obtained by various approaches on the four datasets. Notice that, Stanford Dogs is a balanced dataset, USPS and 20 Newsgroups are more imbalanced,

Table 3.5: Classification errors of individual classes on USPS with 20 labeled points.

Digit	0	1	2	3	4	5	6	7	8	9
GFHF [13]	0.049	**0.006**	0.184	0.109	0.454	**0.279**	0.055	0.202	0.264	0.517
LGC [58]	**0.037**	0.011	**0.106**	0.159	0.488	0.345	0.067	0.116	0.263	0.286
MAD [69]	0.039	0.019	0.198	0.132	0.387	0.318	0.052	0.086	0.199	0.367
GMMC [67]	**0.037**	0.032	0.331	0.227	0.336	0.358	0.157	0.203	0.244	0.247
ONMI [85]	0.070	0.034	0.243	0.176	0.418	0.404	0.076	0.098	0.246	0.308
CAMLP [68]	0.078	0.045	0.233	0.139	0.356	0.303	0.082	0.088	0.183	0.292
LPGMM [86]	0.038	0.018	0.202	0.161	0.353	0.333	0.091	0.125	0.221	0.274
ADP [84]	0.054	0.021	0.167	0.119	0.323	0.306	0.063	0.097	0.185	**0.236**
Ours	0.040	0.007	0.120	**0.095**	**0.302**	0.302	**0.036**	**0.051**	**0.177**	0.294

Table 3.6: Classification errors of individual classes on 20 Newsgroups with different number of labeled points.

Class name	*autos*			*motorcycles*			*baseball*			*hockey*		
# labeled points	10	30	50	10	30	50	10	30	50	10	30	50
GFHF [13]	**0.130**	**0.048**	**0.038**	0.664	0.526	0.441	0.646	0.581	0.518	0.589	0.711	0.729
LGC [58]	0.388	0.233	0.169	0.358	0.268	0.238	0.427	0.370	0.316	0.507	0.622	0.733
MAD [69]	0.247	0.143	0.101	0.508	0.421	0.402	0.508	0.421	0.402	0.437	0.590	0.655
GMMC [67]	0.295	0.226	0.172	0.313	0.238	0.245	0.398	0.378	0.369	0.373	0.581	**0.495**
ONMI [85]	0.172	0.075	0.048	0.495	0.433	0.415	0.565	0.546	0.504	0.500	0.716	0.796
CAMLP [68]	0.186	0.102	0.081	0.476	0.412	0.386	0.502	0.481	0.430	0.438	0.628	0.688
LPGMM [86]	0.304	0.211	0.157	0.361	0.281	0.275	0.426	0.385	0.365	0.413	0.591	0.575
ADP [84]	0.225	0.144	0.108	0.417	0.349	0.338	0.479	0.459	0.428	0.431	0.636	0.643
Ours	0.317	0.183	0.130	**0.304**	**0.226**	**0.225**	**0.356**	**0.241**	**0.192**	**0.369**	**0.573**	0.677

and Wikipedia is a cross-media dataset including most balanced classes with a few imbalanced ones. As can be observed, our approach can outperform the other compared approaches on all the datasets consistently, which demonstrates its effectiveness. Among the other methods compared, ADP [84] and LPGMM[86, 84] that are published more recently, show relatively better overall performance. Further, among the four datasets, our approach can generally outperform by a larger margin on 20 Newsgroups than on the other three datasets. This should not come as a surprise since our approach takes the class imbalance issue into particular consideration. Similarly, GMMC [67] also performs well on 20 Newsgroups, although it performs rather poorly on the other datasets, since it explicitly considers imbalanced data as well like our approach.

Table 3.7: Classification errors of individual classes on Wikipedia with 20 labeled points.

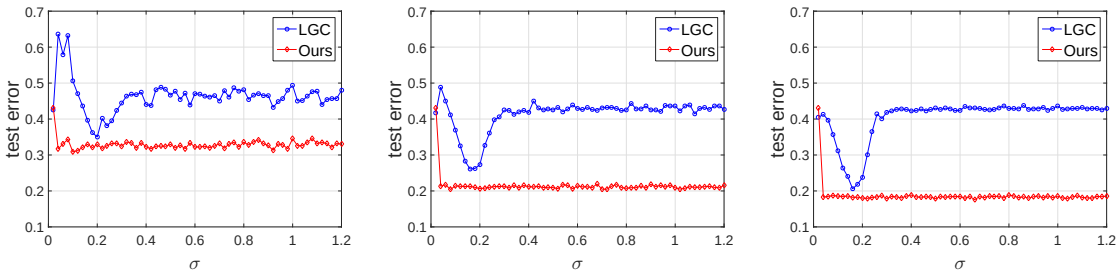
Class name	*art*	*biology*	*geography*	*history*	*literature*	*media*	*music*	*royalty*	*sport*	*warfare*
GFHF [13]	0.860	0.188	0.547	0.780	0.531	**0.553**	0.580	0.465	0.096	0.263
LGC [58]	0.865	0.184	0.560	0.777	0.430	0.576	0.561	0.442	0.128	0.249
MAD [69]	**0.784**	0.201	0.531	**0.720**	0.425	0.576	0.575	0.357	0.155	0.286
GMMC [67]	0.818	0.211	0.584	0.775	0.446	0.599	**0.531**	0.529	0.126	0.309
ONMI [85]	0.809	0.198	0.553	0.741	0.433	0.555	0.589	0.387	0.142	0.281
CAMLP [68]	0.812	0.200	0.526	0.736	0.444	0.556	0.587	0.384	0.143	0.272
LPGMM [86]	0.822	0.195	0.544	0.750	0.470	0.561	0.581	0.409	0.128	0.275
ADP [84]	0.819	0.178	0.528	0.758	0.411	0.565	0.570	0.352	0.116	0.222
Ours	0.828	**0.158**	**0.513**	0.778	**0.385**	0.573	0.551	**0.318**	**0.090**	**0.167**

To further show the merits of our approach, we comparatively list in Tables 3.5-3.7 the performance in terms of test error for each class on the three imbalanced datasets, i.e., USPS, 20 Newsgroups and Wikipedia. One can observe that, our approach can retain relatively balanced accuracy across different classes in each dataset, leading to superior overall performance. By contrast, other methods (most typically LGC [58] and MAD [69]) can be able to perform well only on a part of classes, but not on the others, particularly for the imbalanced 20 Newsgroups dataset. We believe this is due to that we carefully designed the algorithm to take into account imbalanced data.

3.6.2.3 Parameter Study

Our approach involves three parameters: the Gaussian kernel width σ, the number of nearest neighbors k and the parameter γ. We now carry out experiments to investigate the impact of these parameters on performance. Here we report the results and make discussion on the most imbalanced dataset 20 Newsgroups, although similar conclusions can be achieved on the other datasets.

Sensitivity to σ: According to the theoretical analysis in Section 3.5, our approach should be less insensitive to the Gaussian kernel width σ than conventional



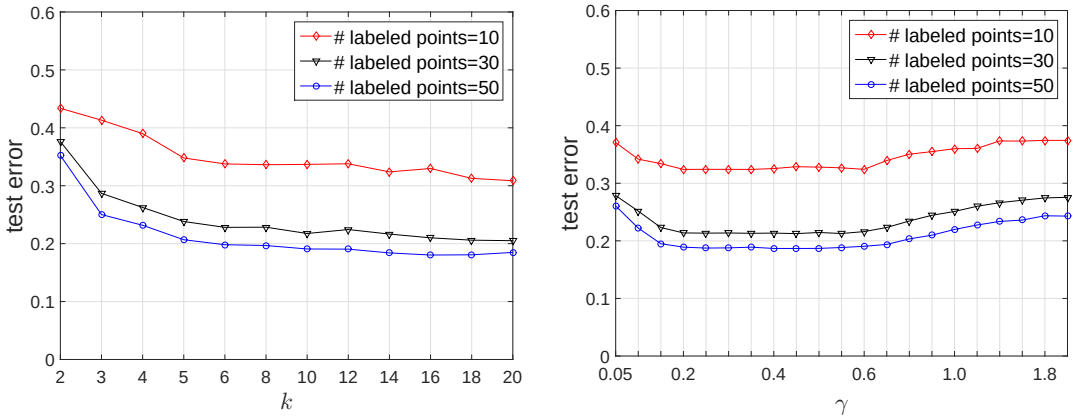
(a) #labeled points = 10 (b) #labeled points = 30 (c) #labeled points = 50

Figure 3.6: Hyper-parameter sensitivity tests on 20 Newsgroups. Each sub-figure shows the test error versus the value of σ.

GRSSL methods (e.g., most typically LGC [58]). To experimentally verify this point, we vary the values of σ to compare performance between our approach and LGC [58] under three different numbers of labeled points, as shown in Fig. 3.6. One can observe that, the curve of LGC exhibits a sharp peak, which means favorable performance can only be achieved within a very narrow interval. This suggests LGC is highly sensitive to σ, making the parameter selection very difficult. By contrast, the performance of our approach remains very stable as σ, indicating its insensitivity to this parameter.

Sensitivity to k: Fig. 3.7(a) plots the performance of our approach with varying k. As can be observed, our approach exhibits very stable performance as long as k is not too small ($k \geq 5$ is favorable for this dataset). Generally our approach is not sensitive to this parameter, which is also the case for most conventional GSSL methods [13, 58].

Sensitivity to γ: The parameter γ controls the trade-off between the terms in Algorithm 2 and finally influences the strength of our approach in dealing with imbalanced data. As suggested in Section 3.5, the general principle for tuning this parameter is that, smaller γ is preferred for more imbalanced data, and vice versa. We experimentally observed that, there usually exists a relatively wide interval for various datasets where the performance remains stably good, making the parameter



(a) Sensitivity to k (b) Sensitivity to γ

Figure 3.7: Hyper-parameter sensitivity tests on 20 Newsgroups. Each sub-figure shows the test error versus the value of k and γ.

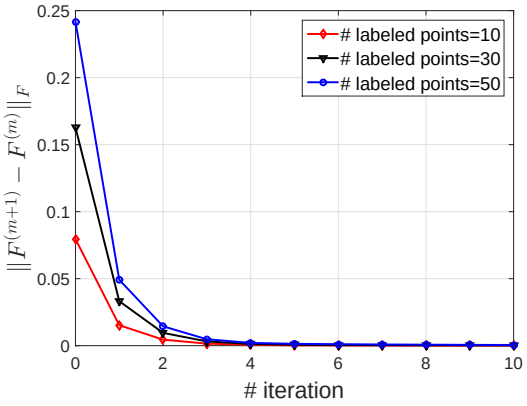


Figure 3.8: Convergence analysis of proposed approach on 20 Newsgroups.

tuning feasible and easy. Fig. 3.7(b) illustrates results on the 20 Newsgroups dataset specifically as an example. As can be seen, the performance is favorable if γ lies within the interval $[0.2, 0.6]$, which becomes worse to some extent if γ is inappropriately set to be too small or too large.

3.6.2.4 Convergence Analysis

Since our approach works in an iterative fashion, one may be naturally concerned with its convergence. In Section 3.4, we have already theoretically proven the convergence of Algorithm 2. Here we further perform another experiment to demonstrate the convergence intuitively, by taking the 20 Newsgroups as an example. The results are shown in Fig. 3.8, which indicates that our approach can converge fast within only a couple of iterations.

3.7 Related Works

3.7.1 Imbalanced Classification

Learning from imbalanced data is challenging for classification problems, since many classifiers are designed without considering the underlying data distribution. To overcome the class-imbalance problem, many approaches were proposed at the data or algorithmic levels. The data-level approaches aim to generate a balanced class distribution by over-sample minor classes or under-sampling major classes [87]. On the other hand, many algorithmic-level methods have been proposed to deal with class imbalance, such as cost-sensitive learning [88], feature selection [63], and hybrid/ensemble techniques [89].

Besides aforementioned methods, the main strategy of GRSSL to deal with class imbalance is incorporating the (estimated) class proportion, since GRSSL takes into account labeled and unlabeled data. As we have explained in Section 3.1, several methods [66, 67] directly incorporate class proportion knowledge, which is not available for most applications. In this case, it is nature to use labeled data to estimate class proportion [13]. However, this is not always feasible since for real-world

datasets, the class proportion of labeled data can be different from that of unlabeled data, especially in the case when very few labeled data are available.

3.7.2 Graph-Regularization based Semi-Supervised Learning

The most well-known standard GRSSL baselines are GFHF [13] and LGC [58], which can be derived from the perspective of both label propagation and optimization problem. Inspired by them, a large number of GRSSL methods have been proposed to incorporate richer graph structures or prior knowledge [50, 52]. For instance, recently several confidence-aware GRSSL approaches [90, 68] were proposed. Their key idea is that it will be more confident to predict a node with more neighbors, since it has enough evidence to make prediction. Similar to our approach, considering the node degree during the propagation process is crucial to these methods.

As we explained in Section 3.5.2, GRSSL can also be viewed as the problem of semi-supervised graph partition, which aims to divide nodes of a graph into a number of groups by incorporating the label information. For real-world graphs, e.g. social networks, the most well-known algorithm for graph partition is modularity optimization [91], which has been applied to semi-supervised learning [92]. However, the modularity optimization suffers from a drawback that it implicitly favours clusters with a certain size, depending on the size of the entire graph, not only on its internal structure. To address this issue, as we explained in Section 3.5.2, Markov stability [93, 94] was introduced based on the fact that a continuous-time random walk on a graph is trapped for long time in good clusters before being able to escape.

3.8 Conclusion

We have introduced a novel GRSSL approach, which could control the imbalance degree of classification result to improve classification accuracy. Theoretically,

our approach can be derived from the perspective of different areas. Especially, we use the linearisation of Markov stability to construct our regularization framework. From a practical standpoint, our approach is extremely easy to implement and has fewer hyper-parameters than most of our counterparts. Finally, our experimental results validate the effectiveness of our approach, as well as the robustness to the Gaussian kernel width. As a result, it enables us to simply choose the unweighted kNN graph for real-world applications.

Besides these advantages, our approach also suffers from a drawback. It is well known that, some standard methods [13, 58] are efficient since their algorithms are based on the sparse matrix induced by kNN graphs. On the other hand, for our approach, it is easy to notice that G in Algorithm 2 is a dense matrix, which makes our algorithm more computationally expensive than these standard ones. In the future, we would like to develop more efficient algorithm for our approach.

CHAPTER 4

MULTI-ENTROPY-RATE GRAPH CONVOLUTIONAL NETWORK

4.1 Introduction

In Chapter 3, we have discussed graph-regularization based semi-supervised learning, which can jointly learn information of graph structure and node labels. However, it can not incorporate node attributes into its framework when it deals with attributed graphs. It can be problematic since node attributes typically contains rich content information, which can be crucial for node classification tasks.

In recent years, encouraged by the great success of convolutional neural networks, an increasing number of works applied neural network models to learn from graph-structured data, which are generated from non-Eulidean domains and represented as graphs with node attributes. And these models are typically termed as graph convolutional networks (GCN). By learning graph structure and node attributes jointly, these GCN-like models can dramatically improve the performance of Multi-Layer Perceptron (MLP) in practice. Hence, they have been widely applied to various real-world tasks, including 3D point clouds classification [95, 96], molecular structure studying [97, 98], text classification [99, 100, 101], and so forth.

In this chapter, we focus on attention-based GCN, which deploys a certain attention mechanism to guide graph nodes to aggregate attribute information from their neighborhoods more effectively. These approaches are effective mainly because that, graphs in real-world applications are usually complexly structured and noisy, and the attention mechanism enables us to suppress the noise and reveal the most task-relevant information. For this purpose, existing attention-based approaches [102, 103]

usually learn graph attention from feature representations at each layer, which assigns larger weights to more important neighbors for each node. Despite their effectiveness to some extent, previous attention-based GCN models share one major limitation, that is, they only take into account immediate neighborhood in the calculation of attention coefficients while ignoring higher-order topological structures in the graph. Moreover, their attention parameters have to be learned along with neural network weights by gradient decent, which makes them computationally inefficient.

To address these issues, we propose a novel method called Multi-Entropy-Rate Graph Convolutional Network (MER-GCN) for semi-supervised node classification. One major contribution of MER-GCN is that, it establishes a simple but effective graph attention mechanism upon the so-called Sub-maximal Entropy-rate Random Walk (SERW), which can capture the topological structures underlying the given graph at multiple scales. With this attention mechanism, MER-GCN can encode graph structures at multiple scales so that richer structural information can be exploited to guide the feature aggregation on nodes, leading to performance improvement. Since our proposed graph attention only relies on the graph structure, it does not need to train the attention parameters by gradient descent. Hence, MER-GCN can be more efficient than existing approaches relying on sophisticated attention mechanisms.

Our main contributions in this chapter summarized as follows:

- We introduce an efficient attention mechanism induced by random walks with different entropy rates.
- We propose the Multi-Entropy-Rate Graph Convolutional Network (MER-GCN), a simple but effective attention-based GCN model which can capture richer topological structures in graphs to derive the graph attention.

- We apply MER-GCN to semi-supervised node classification tasks. Extensive experiments on four benchmarks under both transductive and inductive learning settings suggest that, our method can outperform several strong baselines.

4.2 Preliminaries

In this section, we first formally define the problem we are trying to solve. Then we briefly present a standard baseline [14] of GCN-like approaches. The notations in this chapter are consistent to these defined in Chapter 1.1.

4.2.1 Graph Convolutional Network (GCN)

Here we use a standard baseline GCN [14] to illustrate the main idea of GCN-like approaches, which attempt to incorporate feature aggregation into MLP. At each layer, nodes aggregate feature information from their neighborhoods via graph convolution. For multi-layer GCN , at the l^{th} layer, each node obtains feature information from its neighbors uniformly as follows,

$$\tilde{h}_i^{(l)} = P_{ii} * h_i^{(l)} + \sum_{j \in \mathcal{N}_i} P_{ij} * h_j^{(l)}, \tag{4.1}$$

where $\mathcal{N}_i$ is the index set of v_i's neighbors, $h_i^{(l)}$ and $\tilde{h}_i^{(l)}$ are the hidden representations of v_i before and after the graph convolution respectively, and $P = (D + I)^{-1}(A + I)$ is a normalized adjacency matrix [1].

Combing convolution operation with the fully connected layer, we can write the building block layer of GCN in matrix form as

$$H^{(l+1)} = \sigma(PH^{(l)}W^{(l)}), \tag{4.2}$$

[1] In the original paper, P is a symmetric normalized matrix. In practice, these two matrices are very close with respect to classification problems.

where $H^{(l+1)}$ is the output, $W^{(l)}$ is a trainable weight matrix and $\sigma()$ is a non-linear activation. To relieve the over-smoothing problem [104], typically a two-layer GCN architecture is applied to capture high-order between nodes. Moreover, at the output (L^{th}) layer, GCN evaluates loss on labeled nodes only,

$$L = -\sum_{i \in \mathcal{Y}_{\mathcal{L}}} \sum_{j} Y_{ij} \ln H_{ij}^{(L)}, \tag{4.3}$$

where $\mathcal{Y}_{\mathcal{L}}$ is the index set of labeled nodes, Y is the given label matrix and $H^{(L)}$ is the output of GCN.

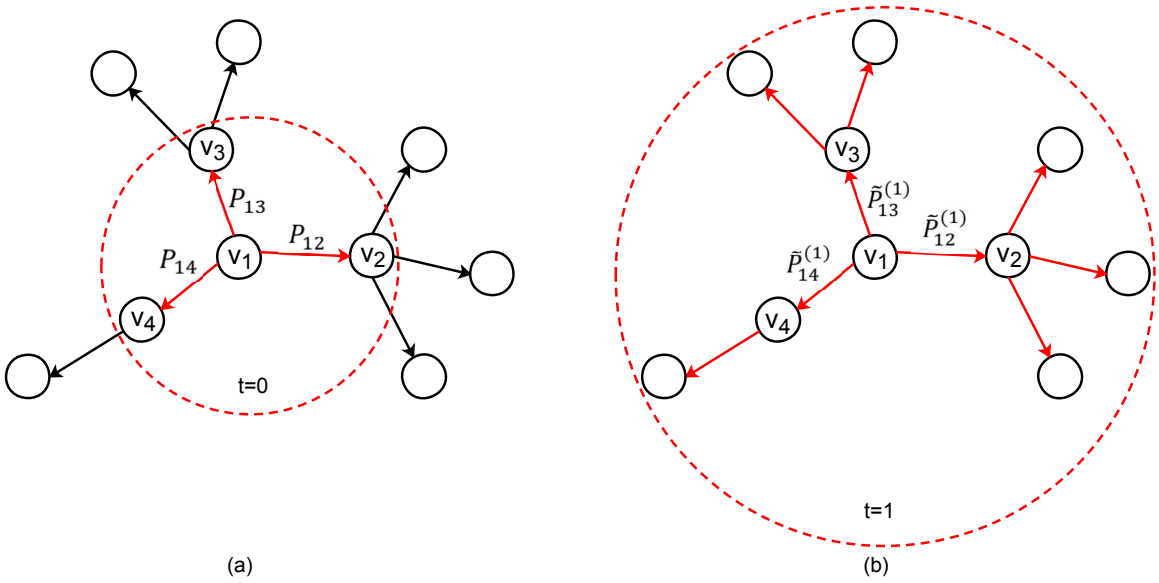


Figure 4.1: Illustration of different random walks on an unweighted directed graph. (a) The generic random walk (GRW) uniformly chooses among paths of length-1 starting from node v_1. The corresponding probability distribution is $(P_{12}, P_{13}, P_{14}) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. (b) Considering the local topological structure beyond the first-order, SERW uniformly chooses among paths of length-2 starting from node v_1. The corresponding probability distribution is $(\tilde{P}_{12}^{(1)}, \tilde{P}_{13}^{(1)}, \tilde{P}_{14}^{(1)}) = (\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$.

4.3 Sub-maximal Entropy-rate Random Walk (SERW)

Before we present our proposed attention mechanism, we first introduce a series of random walks, which can extract different topological structures of the graph.

Given an unweighted graph G with its adjacency matrix A, we can define a Generic Random Walk (GRW) on the graph. Its corresponding transition matrix is commonly defined by $P = D^{-1}A$, where D is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. This means a random walker at node v_i always uniformly chooses among immediate neighbors to decide where to jump.

However, GRW cannot fully extract the topological structure of the graph, since it only consider the immediate neighbors. When the graph structure is complicated, it is helpful to consider the topological structure with higher-order neighbors. To capture richer graph topology, we introduce the Sub-maximal Entropy-rate Random Walk (SERW) [105], which is the local approximation to Maximal Entropy-rate Random Walk (MERW) [106]. The transition probability of SERW is defined as

$$\tilde{P}_{ij}^{(t)} = \frac{A_{ij} \sum_k A_{jk}^{(t)}}{\sum_{j'} A_{ij'} \sum_{k'} A_{j'k'}^{(t)}}, \tag{4.4}$$

where $A_{jk}^{(t)}$ is the number of paths of length-t from v_j to v_k. Thus the transition probability $\tilde{P}_{ij}^{(t)}$ is defined by the proportion of all paths of length-$(t+1)$ starting from v_i and routing through v_j compared to all paths of length-$(t+1)$ starting from v_i. In matrix form, the corresponding transition matrix $\tilde{P}$ can be efficiently calculated by

$$\tilde{A} = A \odot (A^t E)^T, \tag{4.5a}$$

$$\tilde{P} = \tilde{D}^{-1}\tilde{A}, \tag{4.5b}$$

where $\odot$ is the Hadamard product, E is a square matrix with all 1s and $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. According to Eq. 4.4, a random walker of SERW

at node v_i always uniformly chooses among paths of length-$(t+1)$ starting from v_i. Intuitively, the paths generated by SERW with $t > 0$ are more random than those generated by GRW, especially when $t \to \infty$, This randomness can be quantitatively measured by the entropy rate (see Appendix B.1). Note that, GRW is a special case of SERW with $t = 0$. That is, $P = \tilde{P}^{(0)}$.

We now use a simple example to demonstrate SERW can capture various topological structures of the graph by considering paths of different length. Without loss of generality, we consider a directed graph (see Fig. 4.1) since the undirected graph can be regarded as its special case. Moreover, we assume this graph is unweighted. In Fig. 4.1 (a), with the knowledge about the first-order (immediate) neighbors, the walker of GRW at node v_1 applies the uniform probability distribution $(P_{12}, P_{13}, P_{14}) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ to choose its next position. On the other hand, in Fig. 4.1 (b), knowing the first-order and the second-order neighbors, the walker at node v_1 uniformly chooses among paths of length-2 starting from v_1. The corresponding distribution is $(\tilde{P}^{(1)}_{12}, \tilde{P}^{(1)}_{13}, \tilde{P}^{(1)}_{14}) = (\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$. This distribution is much different from the one of GRW since the degrees of v_1's neighbors are different, which also reveals the different topological structure around v_1. Especially, these two distributions can be the same if and only if every node has the same degree. That is, the graph is regular. However, for real world graphs, the local topological structure around each node can be various. Thus, on these graphs, SERW considering paths of different length can naturally reveals graph topology at different scales. Note that, without introducing any additional assumptions, SERW can be applied to the weighted graph, because it be can viewed as the multi-edge graph.

On the other hand, SERW can be viewed as the local approximation to the Maximal Entropy-rate Random Walk (MERW) (see Appendix B.2), whose transition probability is defined as

$$\tilde{P}_{ij}^{(\infty)} = \lim_{t\to\infty} \frac{A_{ij}\sum_k A_{jk}^{(t)}}{\sum_{j'} A_{ij'}\sum_{k'} A_{j'k'}^{(t)}} = \frac{A_{ij}\pi_j}{\lambda\pi_i}, \tag{4.6}$$

where λ is the largest eigenvalue of A, and π is the corresponding eigenvector. SERW is equal to MERW when $t = \infty$.

However, in practice, the walker does not need the global information of the graph for most applications [105]. Moreover, especially for large graphs, the calculation of the eigenvector is computationally expensive.

It has been shown [18, 107, 108] that, the way to aggregate features is crucial to GCN-like approaches. To capture more task-relative information, several attention-based GCNs [102, 103] were proposed. Moreover, from the perspective of random walk, graph convolution can be viewed as applying transition matrix (normalized adjacency matrix) P of GRW to do feature aggregation. Inspired by this viewpoint, we apply the transition matrix of SERW in Eq. 4.4 to capture richer graph structures, which can be viewed as SERW-based attention mechanism. In the following section, we will explain how to combine this attention mechanism with graph neural network to compose a novel GCN-like model.

4.4 MER-GCN Architecture

In this section, we first introduce the attention mechanism based on SERW. Then, with this mechanism, we present the building block layers of MER-GCN. In addition, by adopting different transition matrices to do feature aggregation, we propose three variants of MER-GCN. Finally, we analyze the computational complexity of our approach.

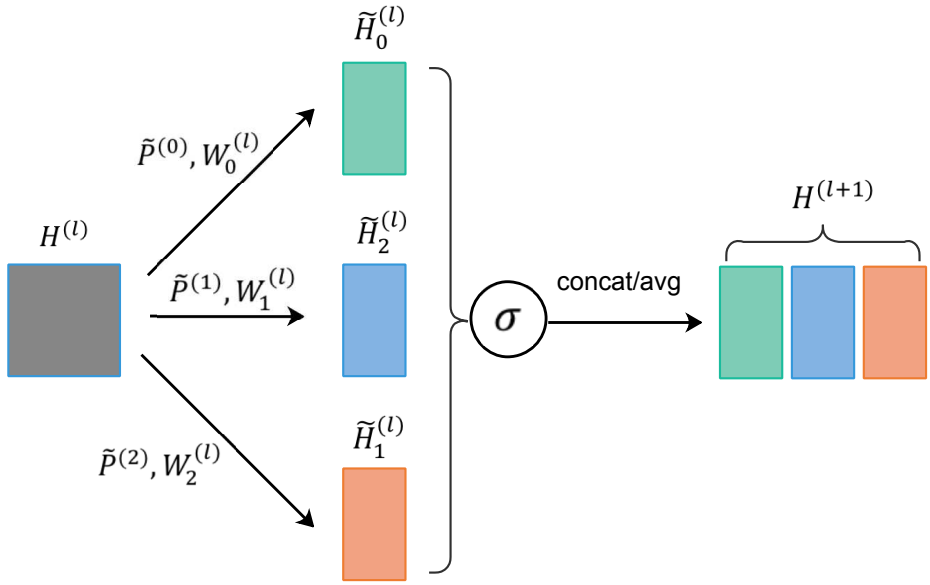


Figure 4.2: The l^{th} layer with three-head attention, where $\tilde{H}_t^{(l)} = \tilde{P}^{(t)} H^{(l)} W_t^{(l)}$. $\tilde{P}^{(t)}$ and $W_t^{(l)}$ corresponds to t^{th} attention head. We concatenate or average all attention heads as the output.

4.4.1 Attention Mechanism via SERW

Considering the local topological structure of the graph beyond the first-order, we can construct the attention mechanism via SERW to aggregate features as follows

$$\tilde{h}_i^{(l)} = \tilde{P}_{ii}^{(t)} * h_i^{(l)} + \sum_{j \in N(i)} \tilde{P}_{ij}^{(t)} * h_j^{(l)}, \tag{4.7}$$

where $\tilde{h}_i^{(l)}$ is the hidden representation of v_i at l^{th} layer after feature aggregation, and $\tilde{P}^{(t)}$ is transition matrix of SERW defined in Eq. 4.4. Intuitively, considering all paths of length-$(t+1)$ starting from v_i, if there are more paths starting from v_i routing through its immediate neighbor v_j, v_j is more important than other neighbors of v_i. Thus, our attention mechanism encourages v_i to aggregate more feature information from v_j. Note that, according to Eq. 4.4, our attention mechanism only re-weights the existing edges without changing the graph structure, which means it does not add or delete any edges.

4.4.2 Graph Attentional Layer

To capture the graph topology at multiple scales, at each layer we apply T-head attention to aggregate feature representations (see Fig. 4.2). Each attention head is independent and corresponds to $\tilde{P}^{(t)}$ with specific t. Then, we concatenate all attention heads, resulting in the following output representations,

$$H^{(l+1)} = \sigma(\tilde{P}^{(0)} H^{(l)} W_0^{(l)} \| \cdots \| \tilde{P}^{(T-1)} H^{(l)} W_{T-1}^{(l)}), \tag{4.8}$$

where $\|$ represents concatenation operation, $H^{(l)} \in \mathbb{R}^{N \times R}$is the output of the previous layer, and specifically $H^{(0)} = X$. $W_t^{(l)}$ is the trainable weight matrix at l^{th} layer corresponding to t^{th} attention head.

4.4.3 Output Layer

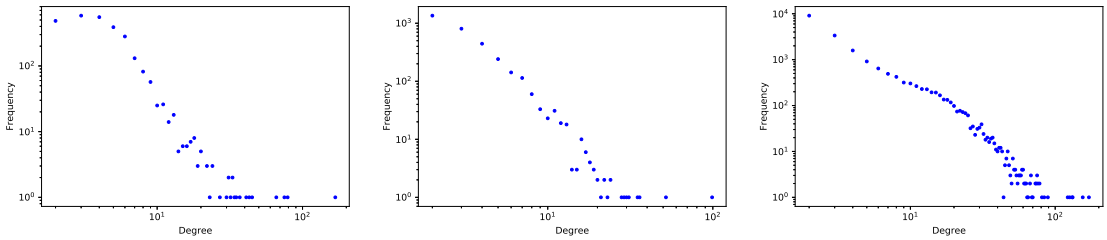
As showed in Fig. 4.2, we also perform multi-head attention at the output layer. However, following [102], instead of concatenating different attention heads, we apply the averaging as follows:

$$H^{(L)} = \sigma(\frac{1}{T} \sum_{t=0}^{T-1} \tilde{P}^{(t)} H^{(L-1)} W_t^{(L-1)}). \tag{4.9}$$

For classification tasks, we feed $H^{(L)}$ to a softmax or logistic sigmoid function, and evaluate the loss on labeled nodes only as in Eq. 4.3.

4.4.4 Transition Matrix

Obviously, our attention mechanism is defined by the transition matrix of SERW, which determines how each node aggregates feature information from its neighbors. However, SERW suffers from a drawback that it can be over degree-biased, which means the random walker inherently biased visits nodes with higher degree. In the real world, many graphs have been reported to be scale-free, which means the degree distribution of them asymptotically follows a power law [109, 110].



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

Figure 4.3: The degree distributions of three citation graphs, which are scale-free. For all graphs, there are some nodes whose degree greatly exceeds the average.

For instance, in Fig. 4.3, there are some nodes in three citation graphs whose degrees greatly exceed the average. Even though the number of high-degree nodes is small, they can affect SERW dramatically, especially when SERW takes into account the long paths.

To alleviate this over degree-biased issue, without introducing any hyper-parameters, we can define SERW via normalized adjacency matrices, i.e. $A_{sym} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}$ and $A_{rw} = (A+I)(D+I)^{-1}$.

Given A_{sym}, we first calculate the re-weighted adjacency matrix $\hat{A}_{sym} = A_{sym} \odot (A_{sym}^t E)^T$ following Eq. 4.5a. Then, we transform $\hat{A}_{sym}$ to a symmetric matrix $\tilde{A}_{sym} = \max(\hat{A}_{sym}, \hat{A}_{sym}^T)$. Finally, we obtain the symmetric transition matrix [2]

$$\tilde{P}_{sym} = \tilde{D}_{sym}^{-1/2} \tilde{A}_{sym} \tilde{D}_{sym}^{-1/2}, \tag{4.10}$$

where $\tilde{D}_{sym}$ is a diagonal matrix with $(\tilde{D}_{sym})_{ii} = \sum_j (\tilde{A}_{sym})_{ij}$. Note that, the way we construct the transition matrix here is slightly different from Eq. 4.4, because here we calculate the symmetric transition matrix.

Given A_{rw}, since it is asymmetric, we consider the out-degree and in-degree of nodes in the corresponding directed graph, respectively. By counting the directed paths on this graph, we can calculate $\hat{A}'_{rw} = A_{rw} \odot (A_{rw}^t E)^T$ and $\hat{A}_{rw} = \hat{A}_{rw}^T \odot$

[2] In this work, we use the term "transition matrix" loosely.

$((\hat{A}^t_{rw})^T E)^T$ following Eq. 4.5a. Then, we obtain $\tilde{A}_{rw} = \max(\hat{A}'_{rw}, \hat{A}^T_{rw})$. Finally, we have the transition matrix

$$\tilde{P}_{rw} = \tilde{D}^{-1/2}_{rw} \tilde{A}_{rw} \tilde{D}^{-1/2}_{rw}, \tag{4.11}$$

where $\tilde{D}_{rw}$ is a diagonal matrix with $(\tilde{D}_{rw})_{ii} = \sum_j (\tilde{A}_{rw})_{ij}$.

4.4.5 Computational Complexity

MER-GCN is computationally efficient. In the pre-training procedure, since the set of transition matrices $\{\tilde{P}^{(t)}\}$ can be calculated efficiently according to Eq. 4.5a and 4.5b before training the neural network, our model is more efficient than sophisticated attention-based approaches. During the network training procedure, for a single layer with T-head attention, the complexity of computing Td_{out} features is $\mathcal{O}(nTd_{in}d_{out} + |E|Td_{out})$, where d_{in} and d_{out} are the number of input and output features corresponding to a single attention head respectively, and $|E|$ is the number of edges in the graph. Hence, when $T = 1$, the time complexity of MER-GCN is exactly equal to that of GCN [14]. Furthermore, since each attention head is independent, the computation of all heads can be parallelized.

4.5 Experiments

We compare our approach with several baselines on node classification tasks under both transductive and inductive learning settings.

4.5.1 Datasets

Transductive learning. We employ three citation graph datasets: Cora, Citeseer and Pubmed, whose node represents a document and edge represents a citation link between documents. The feature vector corresponding to each node is the bag-

Table 4.1: Dataset statistics

Dataset	# Nodes	# Edges	# Classes	# Features	Label rate	Task
Cora	2,708	5,429	7	1,433	0.052	Transductive
Citeseer	3,327	4,732	6	3,703	0.036	Transductive
Pubmed	19,717	44,338	3	500	0.003	Transductive
PPI	56,944 (24 graphs)	818,716	121	50	0.789	Inductive

of-words representation of each document. Following the experimental setup in [111], we only use 20 labeled nodes per class for training, 1000 nodes for testing and 500 nodes for validation.

Inductive learning. We use a protein-protein interaction (PPI) dataset [18], which contains 24 graphs. Each graph has 2372 nodes on average. Every node has 50 features and can be assigned to multiple labels falling into 121 categories. Following the same data preprocessing in [18], we use 20 graphs for training, 2 for validation and 2 for testing. During training, the testing graphs and their corresponding node features are unobserved.

The characteristics and statistics of datasets are described in Table 4.1.

4.5.2 Baseline Methods

Transductive learning. For transductive learning tasks, we compare against several strong baseline approaches as follows:

- **Label Propagation**: Label Propagation (LP) [13] is a graph-based semi-supervised learning baseline, which aims to label unlabeled nodes with the information of graph structure and labeled nodes.
- **DeepWalk**: DeepWalk [25] is a graph embedding method, which is presented in Section 2.2.2.

Table 4.2: The grid search space for the hyper-parameters.

Hyper-parameter	Range
Learning rate	1e-3, 5e-3, 1e-2, 5e-2
Dropout rate	0.4, 0.5, 0.6, 0.7
Weight decay	1e-6, 1e-5, 1e-4

- **Planetoid**: Planetoid [111] is a neural network model to jointly learn the label information and the neighborhood context in the graph.
- **GCN**: GCN [14] is a standard baseline which combines graph convolutional operation with MLP to learn the graph-structured data efficiently. It is the special case of our approach with one-head attention.
- **MoNet**: MoNet [112] is a graph neural network approach adopting peudo-coordinates of nodes to determine the relative position between them, then applying the shared graph filter to do feature aggregation.
- **GAT**: GAT [102] is a GCN-like approach applying trainable attention mechanism to improve the model's expressive capability.
- **AGNN**: AGNN [103] is an attention-based approach, which aims to learn the attention coeffient based on the cosine similarity in the latent space.

Inductive learning. For inductive learning task, we compare against four variants of GraphSAGE [18]: GraphSAGE-GCN, GraphSAGE-mean, GraphSAGE-LSTM and GraphSAGE-pool. GraphSAGE is an inductive framework to generate node representations by sampling and aggregating features from node's neighbors. In addition, we report the performance of GAT.

4.5.3 Experimental Setup

By applying different transition matrices, we compare three variants of MER-GCN: MER-GCN (unnorm), MER-GCN (sym) and MER-GCN (rw). Specifically, MER-GCN (unnorm) corresponds to transition matrix P defined in Eq. 4.4, and MER-GCN (sym) and MER-GCN (rw) correspond to P_{sym}, P_{rw} defined in Eq. 4.10 and 4.11, respectively.

Transductive learning. For transductive learning tasks, we apply a two-layer MER-GCN model, whose first layer is composed of T = 8 attention heads with 8 units each (for a total of 64 units). Following the settings in [102], for Cora and Citerseer datasets, the output layer contains one attention head, and for Pubmed dataset, the output layer averages $T = 8$ attention heads. Finally, we feed the output to a softmax function.

Inductive learning. For inductive learning task, we use a three-layer model. The first two layers contains $T = 4$ attention heads with 256 units each (for a total of 1024 units). The output layer averages $T = 6$ attention heads with 121 units each, then followed by a softmax function. The batch size for training is one graph. For a fair comparison, following [102], our model also applies the skip connection [113] to all attention layers.

The network weights are initialized using Glorot initialization [114]. Furthermore, we train our model for maximum of 10,000 epochs using Adam [115], and stop training if validation loss does not decrease for 100 epochs. The LeakyReLU nonlinearity [116] and dropout are applied to each layer. The hyper-parameters (learning rate, dropout rate and weight decay (L_2 regularization) of our model are optimized on different datasets by grid search. The ranges of grid search for all dataset are summarized in Table 4.2.

Table 4.3: Results (%) of transductive learning experiments in terms of classification accuracy on Citeseer, Cora and Pubmend datasets.

Method	Cora	Citeseer	Pubmed
LP [13]	68.3	45.3	63.0
DeepWalk [25]	67.2	43.2	65.3
Planetoid [111]	75.7	64.7	77.2
GCN [14]	81.5	70.3	79.0
MoNet [112]	81.7	–	78.8
GAT [102]	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
AGNN [103]	82.6 ± 0.08	71.7 ± 0.07	**79.9** ± 0.07
MER-GCN(unnorm)	83.0 ± 0.7	72.8 ± 0.7	79.0 ± 0.3
MER-GCN(sym)	83.5 ± 0.6	73.1 ± 0.6	79.5 ± 0.3
MER-GCN(rw)	**83.8** ± 0.6	**73.4** ± 0.7	**79.9** ± 0.3

4.5.4 Results

Experimental results of transductive and inductive learning are summarized in Tables 4.3 and 4.4, respectively.

Transductive learning. Every classification accuracy in percent is averaged over 100 random runs. Besides reporting the results of three variants of our approach, we also reuse the metrics reported in [102] as the baselines. On Cora and Citeseer datasets, all variants of MER-GCN can outperform not only GCN, but also sophisticated attention-based approaches. On Pubmed dataset, MER-GCN (sym) and MER-GCN (rw) consistently perform better than all baselines except AGNN [103]. Moreover, MER-GCN (rw) outperforms MER-GCN (unnorm) and MER-GCN (sym) on all citation datasets, because we believe MER-GCN (rw) can capture richer topological structures than other variants by capturing the asymmetric node similarity.

Inductive learning: In Table 4.4, we report the micro-averaged F_1 score averaged over 10 random runs. In addition, we reuse the metrics reported in [18] and

Table 4.4: Results of inductive learning experiments in terms of micro-averaged F_1 scores on PPI dataset.

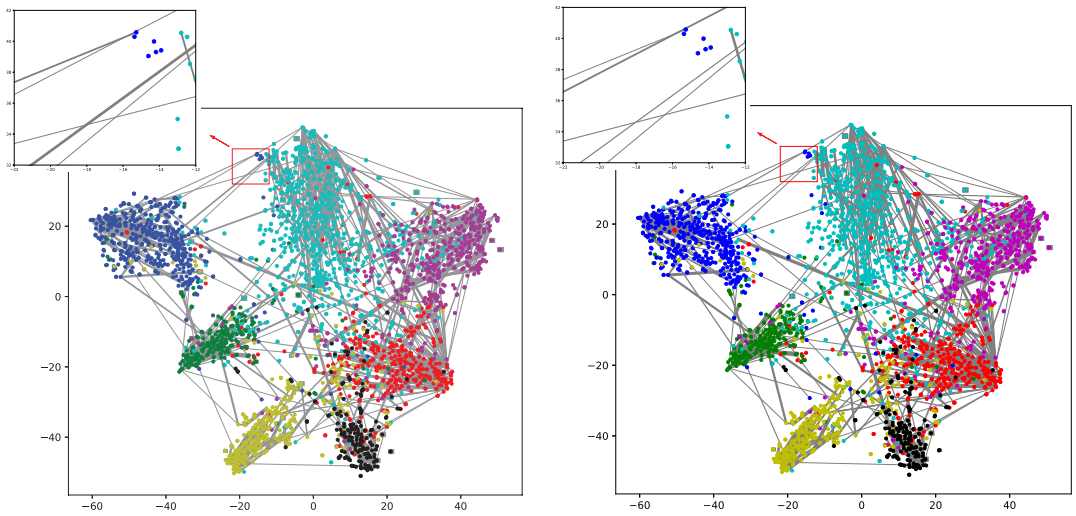
Method	PPI
GraphSAGE-GCN [18]	0.500
GraphSAGE-mean [18]	0.598
GraphSAGE-LSTM [18]	0.612
GraphSAGE-pool [18]	0.600
Const-GAT [102]	0.934 ± 0.006
GAT [102]	0.973 ± 0.002
MER-GCN(unnorm)	0.986 ± 0.001
MER-GCN(sym)	**0.987** ± 0.001
MER-GCN(rw)	**0.987** ± 0.001

Table 4.5: Average training time (seconds) for each dataset over 50 runs. The number in parentheses is the time for calculating transition matrices $\{\tilde{P}^{(t)}\}$.

Dataset	GAT	MER-GCN(rw)
Cora	445.4	105.5 (0.5)
Citeseer	663.5	139.8 (0.3)
Pubmed	489.5	308.4 (31.1)
PPI	4320.7	1086.1 (17.2)

[102] as the baselines. The Const-GAT [102] in Table 4.4 shares the same architecture with GAT, but has the constant graph attention. MER-GCN (sym) and MER-GCN (rw) outperform all baselines and improve upon Const-GAT and GAT by 5.3% and 1.4%, respectively.

For all datasets, MER-GCN (sym) and MER-GCN (rw) perform better than MER-GCN (unnorm), which empirically verifies normalizing the adjacency matrix can overcome the over degree-biased problem.



(a) $\tilde{P}_{rw}^{(0)}$ (b) $\tilde{P}_{rw}^{(1)}$

Figure 4.4: Visualizing the feature representations of the first layer of pre-trained MER-GCN (unnorm) on Cora datasets. (a) Visualization of the feature representations and the attention coefficients of $\tilde{P}_{rw}^{(0)}$. (b) Visualization of the same feature representations and the attention coefficients of $\tilde{P}_{rw}^{(1)}$. Node colors represent classes. Edge thickness indicates 10 times attention coefficients. The upper-left figures in Fig. (a) and Fig. (a) are the zoom-in views corresponding to the red rectangles.

4.5.5 Visualization of Feature Representations and Graph Attention

To demonstrate the effectiveness of our model intuitively, we visualize the learned feature representations of the first layer on Cora dataset in Fig. 4.4 with the t-SNE [41] package. Each node in Fig. 4.4 corresponds to a feature representation. The nodes sharing the same color belong to the same class. Fig. 4.4 indicates most nodes having the same color group together, which reveals our approach has strong discriminative power for Cora dataset. In addition, we visualize the relative strength of the attention coefficients of two attention heads ($\tilde{P}_{rw}^{(0)}$ and $\tilde{P}_{rw}^{(1)}$) in Fig. 4.4 (a) and (b), respectively. For clarity, we only plot one-tenth edges, and edge thickness indicates 10 times attention coefficients. The difference between corresponding attention coefficients in Fig. 4.4 (a) and (b) shows different attention heads can actually capture different topological structures.

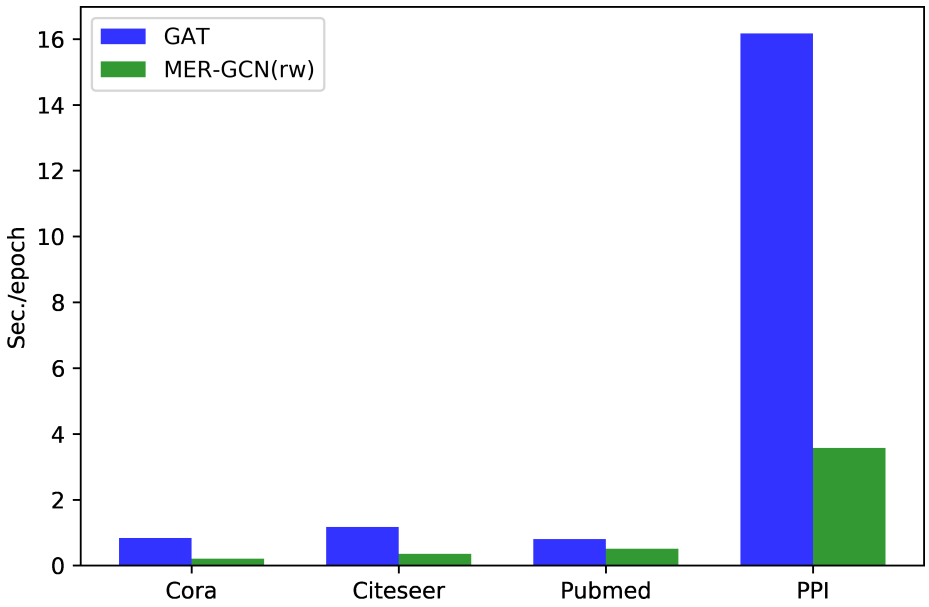


Figure 4.5: Training time per epoch of GAT and MER-GCN (rw) on different datasets. For Pubmed and PPI datset, due to the limitation of our GPU memory, we applied the sparse matrix multiplication to train two models.

4.5.6 Training Time

Fig. 4.5 reports the average training time (seconds) per epoch over 100 epochs on all datasets. We decided to only compare MER-GCN (rw) with GAT since they share the same architecture. Both models were implemented with NVIDIA CUDA ® Deep Neural Network library (cuDNN) to be run on a Nvidia GTX 1080 Ti. For Pubmed and PPI datasets, due to the limitation of our GPU memory, we applied the sparse matrix multiplication to train two models. Obviously, MER-GCN (rw) is significantly faster than GAT on all datasets with respect to each epoch. In Table 4.5, we also provide the average training time (including the time for calculating transition matrices) of GAT and MER-GCN (rw) for each dataset over 50 runs. The number in parentheses is the time (seconds) for calculating transition matrices $\{\tilde{P}^{(t)}\}$. For MER-GCN (rw), calculating transition matrices is roughly 10 to 400 times faster

than training the neural network. As a result, MER-GCN (rw) is more efficient than GAT on all datasets.

4.6 Related Works

4.6.1 Biased Random Walks on the Graph

Random walks have been playing a crucial role in analyzing graph structures, since they are ideal tools to uncover various structural properties of graphs. For instance, diffusion maps [117] utilize random walks to define diffusion distance between data points to do non-linear dimensionality reduction. On the other hand, since nodes are not identical in the real-world graph, algorithms based on biased random walks were proposed to capture richer topological graph structures. PageRank [118] is the most well konwn biased-random-walk based algorithm to estimate how important the a web page is. Recently, [9] proposed an algorithmic framework to learn the node representations through biased random walks.

4.6.2 Graph Convolutional Networks

Inspired by the convolutional neural network, [119] generalized the convolutional operation to the graph-structured data based on the spectral graph theory. Following this work, [120] proposed an efficient localized spectral filtering approximation. Furthermore, [14] introduced an efficient approach by utilizing a localized first-order approximation of the spectral graph convolution. Since this time, there have been an increasing number of GCN-like approaches were proposed. As presented in Section 4.1, generally they fall into two categories: the spectral-based approaches and the spatial-based approaches.

4.6.3 Graph Attention-based Approaches

Recently, attention mechanism was applied to the graph-based approaches. For graph embedding tasks, [121] proposed attention model on the power series of the transition matrix to guide the random walk on the graph. On the other hand, [102] trained a self-attention mechanism to encourage each node to aggregate features from its more relevant neighbors. Following this, [103] proposed a similarity-based attention where a function is applied to calculate the similarity of representations of two nodes. In addition, [122] proposed a motif-based attention mechanism, where each node selects the most motif-induced neighborhood to aggregate features from.

4.7 Conclusion

In this chapter, we first demonstrate SERW can extract topological structures at multiple scales with different local knowledge of the graph. With SERW, we construct a novel attention mechanism which can capture various local topological structures of the graph. Furthermore, we utilize this attention mechanism to build an attention-based graph convolutional network (MER-GCN) for semi-supervised learning. With SERW-based attention mechanism, MER-GCN can jointly learn the node features and graph topology at multiple scales effectively. Experimental results on four benchmarks illusrate the effectiveness and efficiency of MER-GCN.

Besides above advantages, our approach also has several limitations. As we mentioned in Section 4.4.4, our approach suffers from over degree-biased problem. Besides the re-normalized trick, in the future we would like to develop new strategies to allivate this issue. Moreover, since MER-GCN applies full-batch gradient descent, it is not scalable to large graphs which can not fit in GPU memory. Fortunately, to alleviate this issue, several recent GCN approaches [18, 123, 124] using mini-batch

stochastic gradient descent (SGD) have been proposed. Their key ideal is to down-sample neighbors of nodes in the original graph as the pre-processing step. Hence, our approach can incorporate these neighborhood sampling strategies to deal with large datasets.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In the previous chapters, we have introduced three graph-based machine learning approaches, which belong to main themes, i.e., graph embedding and graph-based semi-supervised learning. All proposed approaches have the solid theoretical justifications, which can verify our motivations. In practice, they can achieve the state-of-the-art performance on standard benchmark datasets.

Besides proposed graph-based methods, this thesis also provides several promising directions and open problems, which are discussed below.

In Chapter 2, we introduced a low-rank embedding approach for attributed graphs. It is easy to notice that the goal of our low-rank embedding approach is to learn a linear projection, which maps initial feature vectors into a low-dimensional embedding space. Hence, to learn non-linear projection, it is natural to extend proposed approach to a deep learning model by replace X by $f(X)$, where $f(\cdot)$ can be any non-linear neural network encoder, whose weights can be learned via gradient decent. Consequently, we may obtain more informative node embedding by extract non-linear information from initial attributes.

In Chapter 3, we discussed graph-regularization based semi-supervised learning for node classification. Since our approach is not sensitive to edge weights, we are interested in applying it to unweighted real-world graphs, such as social networks, biological networks and information networks. However, real-world graphs can be very sparse, hence some of them may not be connected. In this case, like other label-

propagation based baselines, if every connected component in the network has at least one labeled node in it, our approach can be valid for node classification.

In addition, as we presented in Section 3.3, our approach can be viewed as a special case of generalized inverses of Laplacian matrices. Hence, it is interesting to apply other generalized inverses to GRSSL or network analysis, for instance, Moore-Penrose inverse. To our best knowledge, there is no work which has fully studied the difference and connections between these generalized inverses from the perspective of GRSSL. In the future, we plan to give a literature survey to fill this gap and develop new efficient GRSSL algorithms based on other generalized inverses.

In Chapter 4, we discussed an attention-based GCN method on attributed graphs to solve node classification problem. Besides attributed graphs, we are interested in applying our approach to non-attributed graphs. In the real world, there are numerous non-attributed graphs. Since our approach focuses on topological structures of the graph, it may produce node representations with high quality on non-attributed graphs. In addition, our approach can be applied to other important tasks, such as graph classification, link prediction and graph embedding.

An intriguing open question is how to suppress the noise in the graph. Graph convolution is the key operation of most GCN models. It is typically represented by the message passing between immediate neighbors. However, real-world graphs are typically noisy because some edges in them may attach two dissimilar nodes. In this case, a node attached to arbitrarily defined edges will obtain noisy feature information via graph convolution, which will reduce the quality of hidden representation. Generally, there are two approaches to overcome this problem. The first is attention-based method, which can suppress the noise and reveal the most task-relevant information. The second method is suppressing the noise via graph diffusion, such as the

heat kernel and personalized PageRank, which will encourage the node to aggregate information from a larger neighborhood.

APPENDICES

A.1 Definition of Generalized Inverses

Definition 1. *For every real square matrix A, $A^{\{i,j,\ldots,k\}}$ denotes the set of matrices X that satisfy equations $(i), (j), \ldots, (k)$ of the following conditions. $X \in A^{\{i,j,\ldots,k\}}$ is called an $\{i, j, \ldots, k\}$-inverse of A.*

$$
\begin{aligned}
&(Condition\ 1) \quad AXA = A,\\
&(Condition\ 2) \quad XAX = X,\\
&(Condition\ 3) \quad (AX)^T = AX,\\
&(Condition\ 4) \quad (XA)^T = XA,\\
&(Condition\ 5) \quad AX = XA.
\end{aligned}
$$

Specifically, the unique matrix $A^{\{1,2,5\}}$ is called the group inverse of A, which is studied in Section 3.5.3. In this thesis, we focus on the generalized inverse of Laplacian matrix, which has intimate connection to GRSSL.

A.2 GFHF as a Generalized Inverse of $I - P$

We now show GFHF can be explained as a generalized solution to Eq. 3.4. Let us first consider an absorbing Markov chain on the graph, where labeled points correspond to absorbing states and unlabeled points correspond to transient states. In addition, we let l denote the number of labeled data, and let u denote the number of unlabeled data. The corresponding transition matrix will have the canonical form

$$
\widetilde{P} = \begin{bmatrix} I_{ll} & O_{lu} \\ P_{ul} & P_{uu} \end{bmatrix},
$$

where I_{ll} is a l-by-l identity matrix, O_{lu} is a l-by-u zero matrix, P_{ul} and P_{uu} are the submatrices of P at the corresponding positions. Note that, $I - \widetilde{P}$ can be viewed a generalized Laplacian matrix. We also divide Y and F^* into $\left[\begin{smallmatrix} Y_l \\ Y_u \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} F_l^* \\ F_u^* \end{smallmatrix}\right]$ respectively. In addition, we set $\Sigma = I$, and $Q = J$, where J is a diagonal matrix with the first l diagonal entries as 1 and the rest as 0. Then, $I - \widetilde{P} + J$ can be written as

$$\begin{bmatrix} I_{ll} & O_{lu} \\ -P_{ul} & I_{uu} - P_{uu} \end{bmatrix}.$$

Since $I_{uu} - P_{uu}$ is nonsingular [71], $I - \widetilde{P} + J$ is nonsingular as well. Hence, we can write the solution to Eq. 3.4 in matrix form,

$$\begin{bmatrix} F_l \\ F_u \end{bmatrix} = \begin{bmatrix} I_{ll} & O_{lu} \\ -P_{ul} & I_{uu} - P_{uu} \end{bmatrix}^{-1} \begin{bmatrix} I_{ll} & O_{lu} \\ O_{ul} & O_{uu} \end{bmatrix} \begin{bmatrix} Y_l \\ Y_u \end{bmatrix}.$$

Finally, we obtain the closed form expression of GFHF,

$$F_u = (I_{uu} - P_{uu})^{-1} P_{ul} Y_L.$$

Moreover, with the definition in Appendix A.1, we have $(I - \widetilde{P} + J)^{-1} \in (I - \widetilde{P})^{\{1,3,4,5\}}$.

Similarly, it is easy to verify that other methods in Table 3.1 correspond to generalized inverses $L^{\{i,j,...,k\}}$. Consequently, they can be viewed as the special cases of Eq. 3.5 (see Table 3.1).

B.1 Entropy Rate of Random Walks on the Graph

Let $v_1, v_2, ..., v_t$ denote a path of length-t generated by the random walk on the graph. And the corresponding joint probability of this path is denoted by $p(v_1, v_2, ..., v_t)$. Given these, we can define the Shannon entropy in the set of all paths of length-t as

$$S_t = -\sum_{v_1,...,v_t} p(v_1, ..., v_t) \ln p(v_1, ..., v_t). \tag{B.1.1}$$

Then, the entropy rate of this random walk is the fixed rate of the entropy S_t increasing with time,

$$\begin{aligned}
s &= \lim_{t\to\infty} \frac{S_t}{t} \\
&= -\lim_{t\to\infty} \sum_{v_1,...,v_t} \frac{p(v_1,...,v_t)\ln p(v_1,...,v_t)}{t} \\
&= -\lim_{t\to\infty} \sum_{v_1,...,v_t} \frac{p(v_1,...,v_t)\ln(p(v_1)p(v_2|v_1)...p(v_t|v_{t-1}))}{t} \\
&= -\lim_{t\to\infty} \sum_{v_1,...,v_t} \frac{p(v_1,...,v_t)(\ln p(v_1)+\ln p(v_2|v_1)+...+\ln p(v_t|v_{t-1}))}{t} \\
&= -\lim_{t\to\infty} \sum_{v_1,...,v_t} \frac{p(v_1,...,v_t)(\ln p(v_1)+(t-1)\ln p(v_2|v_1))}{t} \\
&= -\sum_{v_1,v_2} p(v_1,v_2)\ln p(v_2|v_1) \\
&= -\sum_{i,j} \pi_i P_{ij} \ln P_{ij},
\end{aligned} \tag{B.1.2}$$

where P is the transition matrix of this random walk, and π is the stationary distribution. Note that, since random walks in this thesis are time-homogeneous, we have $p(v_2|v_1) = p(v_{k+1}|v_k)\ \forall k$. On the other hand, s can be regarded as the average entropy per step in ensemble of paths of infinite length generated by the random walk. In the long run, s is maximized when every path has equal probability.

To extract the local topological structure of the graph, MERW considers the paths of finite length instead of infinite length. In practice, for each node v_i, its goal is to determine the appropriate local distribution in the neighborhood so that all possible paths of equal length starting from v_i are equally likely. The corresponding matrix is defied in Eq. 4.6.

B.2 Maximal Entropy-rate Random Walks

We now prove that $\tilde{P}^{(t)}$ is the local approximation to $\tilde{P}^{(\infty)}$.

Theorem B.2.1.

$$\tilde{P}_{ij}^{(\infty)} = \lim_{t\to\infty} \frac{A_{ij}\sum_k A_{jk}^{(t)}}{\sum_{j'} A_{ij'}\sum_{k'} A_{j'k'}^{(t)}} = \frac{A_{ij}\pi_j}{\lambda\pi_i}, \tag{B.2.1}$$

where λ is the largest eigenvalue of A, and π is the corresponding eigenvector.

Proof. By introducing the all-ones vector $\mathbf{1}$, we can rewrite the term $\sum_k A_{jk}^{(t)}$ as $[A^t\mathbf{1}]_j$. Then, we have

$$\lim_{t\to\infty} \frac{\sum_k A_{jk}^{(t)}}{\sum_{k'} A_{j'k'}^{(t)}} = \lim_{t\to\infty} \frac{[A^t\mathbf{1}]_j}{[A^t\mathbf{1}]_{j'}}. \tag{B.2.2}$$

On the other hand, we use $\{\lambda_i\}$ to denote the eigenvalues of A, and order them decreasingly $\lambda = \lambda_1 > \lambda_2 = ... = \lambda_n$. Then the spectral decomposition of A is $A = \sum_{i=1}^n \lambda_i\psi_i\phi_i^T$, where ψ_i and ϕ_i^T are the right and left eigenvectors of A corresponding to λ_i. Given this, we have

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\sum_k A_{jk}^{(t)}}{\sum_{k'} A_{j'k'}^{(t)}} = \lim_{t\to\infty} \frac{[A^t\mathbf{1}]_j}{[A^t\mathbf{1}]_{j'}} \\
&= \lim_{t\to\infty} \frac{[(\lambda_1^t\psi_1\phi_1^T + \sum_{i=2}^n \lambda_i\psi_i\phi_i^T)\mathbf{1}]_j}{[(\lambda_1^t\psi_1\phi_1^T + \sum_{i=2}^n \lambda_i\psi_i\phi_i^T)\mathbf{1}]_{j'}} \\
&= \lim_{t\to\infty} \frac{[(\psi_1\phi_1^T + \sum_{i=2}^n (\frac{\lambda_n}{\lambda_1})^t\psi_n\phi_n^T)\mathbf{1}]_j}{[(\psi_1\phi_1^T + \sum_{i=2}^n (\frac{\lambda_n}{\lambda_1})^t\psi_n\phi_n^T)\mathbf{1}]_{j'}} \\
&= \frac{[\psi_1]_j}{[\psi_1]_{j'}} \\
&= \frac{\pi_j}{\pi_{j'}}.
\end{aligned} \tag{B.2.3}$$

where ψ_1 is denoted as π. Since $\sum A_{ij'}\pi_{j'} = \lambda\pi_i$, we obtain

$$\lim_{t\to\infty} \frac{A_{ij}\sum_k A_{jk}^{(t)}}{\sum_{j'} A_{ij'}\sum_{k'} A_{j'k'}^{(t)}} = \frac{A_{ij}\pi_j}{\sum_{j'} A_{ij'}\pi_{j'}} = \frac{A_{ij}\pi_j}{\lambda\pi_i}. \tag{B.2.4}$$

□

According to Eq. B.2.4, the entropy rate of MERW is $\ln\lambda$. We now show that MERW indeed maximize the entropy rate. The entropy production rate of the

random walk is maximized when every path shares the same probability in the long run. That is, the maximal entropy rate is

$$s_{max} = \lim_{t\to\infty} \frac{\ln \sum_{i,j} A_{ij}^{(t)}}{t} = \ln \lambda, \tag{B.2.5}$$

where $\sum_{i,j} A_{ij}^{(t)}$ is the number paths of length-t. Hence, we see that MERW obtains the maximal entropy rate.

REFERENCES

[1] D. Easley, J. Kleinberg, *et al.*, *Networks, crowds, and markets*. Cambridge university press Cambridge, 2010, vol. 8.

[2] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.

[3] J. Scott, "Social network analysis," *Sociology*, vol. 22, no. 1, pp. 109–127, 1988.

[4] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[7] Perozzi et al., "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[8] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 891–900.

[9] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[10] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *IJCAI*, vol. 2015, 2015, pp. 2111–2117.

[11] H. Gao and H. Huang, "Deep attributed network embedding." in *IJCAI*, vol. 18. New York, NY, 2018, pp. 3364–3370.

[12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[13] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *arXiv preprint arXiv:1802.09691*, 2018.

[16] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[17] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[19] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[20] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 609–618.

[21] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 37–48.

[22] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[23] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 459–467.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[26] L. Chen, S. Gong, J. Bruna, and M. Bronstein, "Attributed random walk as matrix factorization," in *neural information processing systems, graph representation learning workshop*, 2019.

[27] R. Bhatia and P. Rosenthal, "How and why to solve the operator equation ax-xb= y," *Bulletin of the London Mathematical Society*, vol. 29, no. 1, pp. 1–21, 1997.

[28] M. Xu, R. Jin, and Z.-H. Zhou, "Speedup matrix completion with side information: Application to multi-label learning," in *Advances in neural information processing systems*, 2013, pp. 2301–2309.

[29] D. Luo, F. Nie, C. Ding, and H. Huang, "Multi-subspace representation and discovery," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 405–420.

[30] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.

[31] G. Salha, R. Hennequin, and M. Vazirgiannis, "Keep it simple: Graph autoencoders without graph convolutional networks," *arXiv preprint arXiv:1910.00942*, 2019.

[32] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[33] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*. Springer, 1971, pp. 134–151.

[34] P. Jain and I. S. Dhillon, "Provable inductive matrix completion," *arXiv preprint arXiv:1306.0626*, 2013.

[35] J. Liu, Z. He, L. Wei, and Y. Huang, "Content to node: Self-translation network embedding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1794–1802.

[36] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." in *ICLR (Poster)*, 2019.

[37] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259–270.

[38] Huang et al., "Rwr-gae: Random walk regularization for graph auto encoders," *arXiv preprint arXiv:1908.04003*, 2019.

[39] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[41] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[42] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[43] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[44] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[45] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding." in *AAAI*, vol. 17, 2017, pp. 203–209.

[46] H.-F. Yu, P. Jain, P. Kar, and I. Dhillon, "Large-scale multi-label learning with missing labels," in *International conference on machine learning*, 2014, pp. 593–601.

[47] N. Natarajan and I. S. Dhillon, "Inductive matrix completion for predicting gene–disease associations," *Bioinformatics*, vol. 30, no. 12, pp. i60–i68, 2014.

[48] K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon, "Matrix completion with noisy side information," in *Advances in Neural Information Processing Systems*, 2015, pp. 3447–3455.

[49] S. Si, K.-Y. Chiang, C.-J. Hsieh, N. Rao, and I. S. Dhillon, "Goal-directed inductive matrix completion," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1165–1174.

[50] X. J. Zhu, "Semi-supervised learning literature survey," *Computer Sciences Technical Report 1530, University of Wisconsin-Madison Department of Computer Sciences, 2005.*

[51] A. Subramanya and P. P. Talukdar, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, 2014.

[52] Y. Chong, Y. Ding, Q. Yan, and S. Pan, "Graph-based semi-supervised learning: A review," *Neurocomputing (2020) 216-230*, 2020.

[53] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3166–3173.

[54] L. Ma, A. Ma, C. Ju, and X. Li, "Graph-based semi-supervised learning for spectral-spatial hyperspectral image classification," *Pattern Recognition Letters*, vol. 83, pp. 133–142, 2016.

[55] Z. Qiu, E. Cho, X. Ma, and W. Campbell, "Graph-based semi-supervised learning for natural language understanding," in *Proceedings of the 13th Workshop on Graph-Based Methods for Natural Language Processing*, 2019, pp. 151–158.

[56] L. Peel, "Graph-based semi-supervised learning for relational networks," in *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2017, pp. 435–443.

[57] Q. Li, X.-M. Wu, H. Liu, X. Zhang, and Z. Guan, "Label efficient semi-supervised learning via graph filtering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9582–9591.

[58] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, 2004, pp. 321–328.

[59] M. Belkin and P. Niyogi, "Semi-supervised learning on riemannian manifolds," *Machine Learning*, vol. 56, no. 1-3, pp. 209–239, 2004.

[60] D. Slepcev and M. Thorpe, "Analysis of p-laplacian regularization in semisupervised learning," *SIAM Journal on Mathematical Analysis*, vol. 51, no. 3, pp. 2085–2120, 2019.

[61] X.-M. Wu, Z. Li, A. M. So, J. Wright, and S.-F. Chang, "Learning with partially absorbing random walks," in *Advances in Neural Information Processing Systems*, 2012, pp. 3077–3085.

[62] Q. Li, W. Liu, and L. Li, "Self-reinforced diffusion for graph-based semi-supervised learning," *Pattern Recognition Letters*, vol. 125, pp. 439–445, 2019.

[63] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, p. 42, 2018.

[64] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[65] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[66] G. S. Mann and A. McCallum, "Simple, robust, scalable semi-supervised learning via expectation regularization," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 593–600.

[67] J. Wang, T. Jebara, and S.-F. Chang, "Semi-supervised learning using greedy max-cut," *Journal of Machine Learning Research*, vol. 14, no. Mar, pp. 771–800, 2013.

[68] Y. Yamaguchi, C. Faloutsos, and H. Kitagawa, "Camlp: Confidence-aware modulated label propagation," in *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2016, pp. 513–521.

[69] P. P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 442–457.

[70] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning*. MIT Press, 2006, p. 193â216.

[71] J. G. Kemeny and J. L. Snell, *Finite markov chains*. D. Van Nostrand, 1960.

[72] E. H. Simpson, "Measurement of diversity," *Nature*, vol. 163, no. 4148, p. 688, 1949.

[73] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12 755–12 760, 2010.

[74] J. G. Kemeny and J. L. Snell, "Finite continuous time markov chains," *Theory of Probability & Its Applications*, vol. 6, no. 1, pp. 101–105, 1961.

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

[76] C. D. Meyer, *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, 2000.

[77] B. Girault, P. Gonçalves, E. Fleury, and A. S. Mor, "Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2014, pp. 1115–1119.

[78] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[80] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009.* Ieee, 2009, pp. 248–255.

[81] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[82] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 331–339.

[83] J. C. Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. R. Lanckriet, R. Levy, and N. Vasconcelos, "On the role of correlation and abstraction in cross-modal multimedia retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 521–535, 2013.

[84] Q. Li, S. An, W. Liu, and L. Li, "Semisupervised learning on graphs with an alternating diffusion process," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2020.

[85] Y. Yamaguchi, C. Faloutsos, and H. Kitagawa, "Omni-prop: Seamless node classification on arbitrary label correlation," in *29th AAAI Conference on Artificial Intelligence*, 2015.

[86] M. Fan, X. Zhang, L. Du, L. Chen, and D. Tao, "Semi-supervised learning through label propagation on geodesics," *IEEE Transactions on Cybernetics*, vol. 48, no. 5, pp. 1486–1499, 2017.

[87] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and R. A. Bauder, "Severely imbalanced big data challenges: investigating data sampling approaches," *Journal of Big Data*, vol. 6, no. 1, p. 107, 2019.

[88] V. López, S. Del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data," *Fuzzy Sets and Systems*, vol. 258, pp. 5–38, 2015.

[89] A. Maurya, "Bayesian optimization for predicting rare internal failures in manufacturing processes," in *IEEE International Conference on Big Data*. IEEE, 2016, pp. 2036–2045.

[90] M. Orbach and K. Crammer, "Graph-based transduction with confidence," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 323–338.

[91] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.

[92] R. Devooght, A. Mantrach, I. Kivimäki, H. Bersini, A. Jaimes, and M. Saerens, "Random walks based modularity: application to semi-supervised learning," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 213–224.

[93] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit," *PloS ONE*, vol. 7, no. 2, p. e32210, 2012.

[94] Z. Liu and M. Barahona, "Geometric multiscale community detection: Markov stability and vector partitioning," *Journal of Complex Networks*, vol. 6, no. 2, pp. 157–172, 2018.

[95] G. Te, W. Hu, A. Zheng, and Z. Guo, "Rgcnn: Regularized graph cnn for point cloud segmentation," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 746–754.

[96] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[97] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems*, 2015, pp. 2224–2232.

[98] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[99] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," *Proceedings of EMNLP*, pp. 1506–1515, 2017.

[100] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *Proceedings of EMNLP*, pp. 1957–1967, 2017.

[101] D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," *Proceedings of the 2018 Conference of the North American*, pp. 486–492, 2018.

[102] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations (ICLR)*, 2018.

[103] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[104] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *International Conference on Learning Representations*, 2019.

[105] R. Sinatra, J. Gómez-Gardenes, R. Lambiotte, V. Nicosia, and V. Latora, "Maximal-entropy random walks in complex networks with limited information," *Physical Review E*, vol. 83, no. 3, p. 030103, 2011.

[106] Z. Burda, J. Duda, J.-M. Luck, and B. Waclaw, "Localization of the maximal entropy random walk," *Physical Review Letters*, vol. 102, no. 16, p. 160602, 2009.

[107] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[108] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," *n Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018.

[109] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[110] G. Lima-Mendez and J. van Helden, "The powerful law of the power law and other myths in network biology," *Molecular BioSystems*, vol. 5, no. 12, pp. 1482–1493, 2009.

[111] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *International Conference on Machine Learning (ICML)*, 2016.

[112] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[113] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[114] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[115] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[116] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *International Conference on Learning Representations (ICLR) Workshop*, 2015.

[117] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences*, vol. 102, no. 21, pp. 7426–7431, 2005.

[118] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[119] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *Proceedings of the 3rd International Conference on Learning Representations*, 2013.

[120] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *In Advances in Neural Information Processing Systems, 2016*.

[121] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Advances in Neural Information Processing Systems*, 2018, pp. 9180–9190.

[122] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Higher-order graph convolutional networks," *arXiv preprint arXiv:1809.07697*, 2018.

[123] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," *International Conference on Machine Learning*, 2017.

[124] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 4558–4567.

BIOGRAPHICAL STATEMENT

Jianjin Deng received his Ph.D. in Computer Science and Engineering from the University of Texas at Arlington (UTA) at 2021. Prior to the Ph.D. program in UTA, he received his M.Eng. degree in Pattern Recognition and Intelligent System and B.Eng. degree in Software Engineering from Huazhong University of Science and Technology, Wuhan, China in 2015 and 2012, respectively. His main research interests are deep learning and machine learning on graph structured data.