1

Deep Learning on Graphs: An Introduction

1.1 Introduction

We start this chapter by answering a few questions about the book. First, we discuss why we should pay attention to deep learning on graphs. In particular, why do we represent real-world data as graphs, why do we want to bridge deep learning with graphs, and what are challenges for deep learning on graphs? Second, we introduce what content this book will cover. Specifically, which topics we will discuss and how we organize these topics? Third, we provide guidance about who should read this book. Especially what is our target audience and how to read this book with different backgrounds and purposes of reading? To better understand deep learning on graphs, we briefly review the history under the more general context of feature learning on graphs.

1.2 Why Deep Learning on Graphs?

Since data from real-world applications have very diverse forms, from matrix and tensor to sequence and time series, a natural question that arises is why we attempt to represent data as graphs. There are two primary motivations. First, graphs provide a universal representation of data, as shown in Figure 1.1. Data from many systems across various areas can be explicitly denoted as graphs such as social networks, transportation networks, protein-protein interaction networks, knowledge graphs, and brain networks. Meanwhile, as indicated by Figure 1.1, numerous other types of data can be transformed into the form of graphs. Second, a vast number of real-world problems can be addressed as a small set of computational tasks on graphs. Inferring nodes attributes, detecting anomalous nodes (e.g., spammers or terrorists), identifying relevant genes to diseases, and suggesting medicines to patients can be summarized

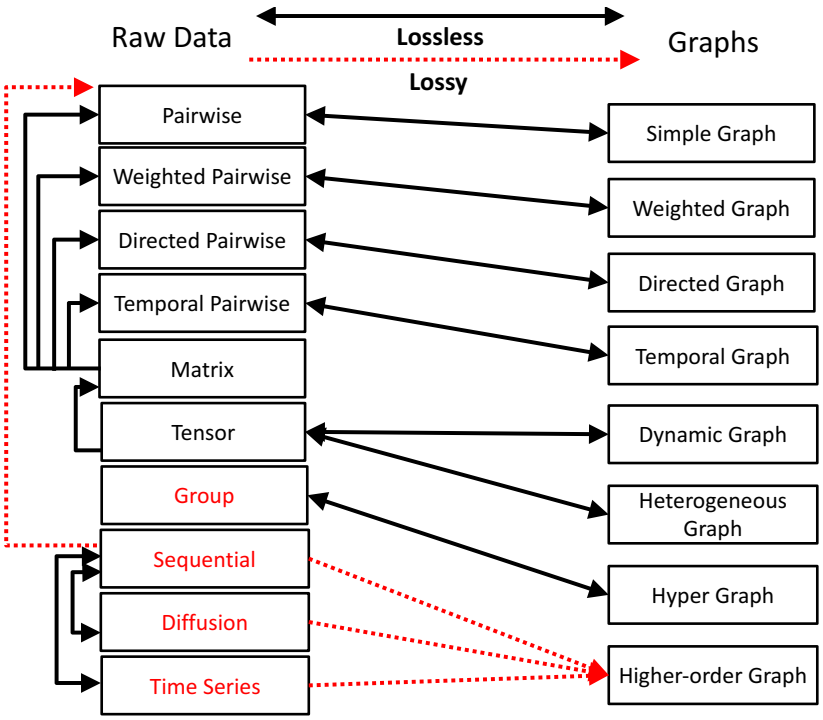


Figure 1.1 Representing real-world data as graphs. The figure is reproduced from (Xu, 2017). Solid lines are utilized to denote lossless representations, while dotted lines are used to indicate lossy representations. Note that we replace "network" in the original figure with "graph" .

as the problem of node classification. Recommendations, polypharmacy side effect prediction, drug-target interaction identification, and knowledge graph completion are essentially the problem of link prediction.

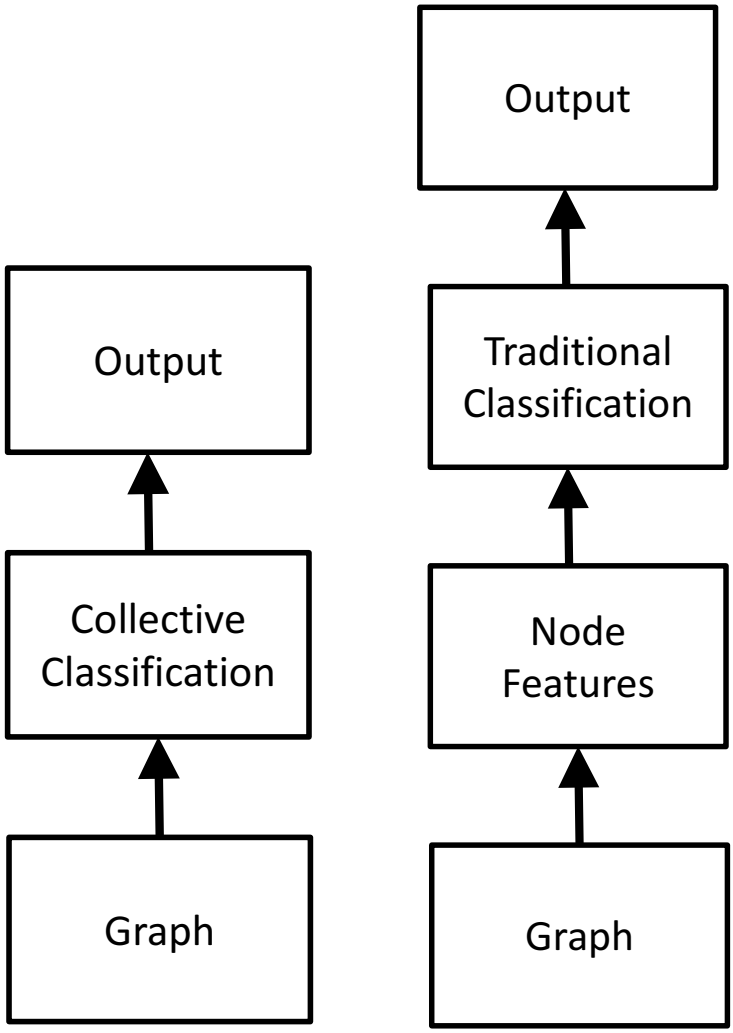
Nodes on graphs are inherently connected that suggests nodes not independent and identically distributed. Thus, traditional machine learning techniques cannot be directly applied for the computational tasks on graphs. There are two main directions to develop solutions. As shown in Figure 1.2, we will use node classification as an illustrative example to discuss these two directions. One direction is to build a new mechanism specific to graphs. The classification problem designed for graphs is known as collective classification (Sen et al., 2008) as demonstrated in Figure 1.2a. Different from traditional classification, for a node, collective classification considers not only the mapping between its features and its label but also the mapping for its neighborhood. The other di-

rection is to flatten a graph by constructing a set of features to denote its nodes where traditional classification techniques can be applied, as illustrated in Figure 1.2b. This direction can take advantage of traditional machine learning techniques; thus, it has become increasingly popular and dominant. The key to the success of this direction is how to construct a set of features for nodes (or node representations). Deep learning has been proven to be powerful in representation learning that has greatly advanced various domains such as computer vision, speech recognition, and natural language processing. Therefore, bridging deep learning with graphs present unprecedented opportunities. However, deep learning on graphs also faces immense challenges. First, traditional deep learning has been designed for regular structured data such as images and sequences, while graphs are irregular where nodes in a graph are unordered and can have distinct neighborhoods. Second, the structural information for regular data is simple; while that for graphs is complicated especially given that there are various types of complex graphs (as shown in Figure 1.1) and nodes and edges can associate with rich side information; thus traditional deep learning is not sufficient to capture such rich information. A new research field has been cultivated – deep learning on graphs, embracing unprecedented opportunities and immense challenges.

1.3 What Content is Covered?

The high-level organization of the book is demonstrated in Figure 1.3. The book consists of four parts to best accommodate our readers with diverse backgrounds and purposes of reading. Part ONE introduces basic concepts; Part TWO discusses the most established methods; Part THREE presents the most typical applications, and Part FOUR describes advances of methods and applications that tend to be important and promising for future research. For each chapter, we first motivate the content that will be covered, then present the material with compelling examples and technical details; and finally, provide more relevant content as further reading. Next, we briefly elaborate on each chapter.

- Part ONE: Foundations. These chapters focus on the basics of graphs and deep learning that will lay the foundations for deep learning on graphs. In Chapter 2, we introduce the key concepts and properties of graphs, Graph Fourier Transform, graph signal processing, and formally define various types of complex graphs and computational tasks on graphs. In Chapter 3,



(a) Collective classification

(b) Traditional classification with node feature extraction

Figure 1.2 Two major directions to develop solutions for node classification on graphs

we discuss a variety of basic neural network models, key approaches to train deep models, and practical techniques to prevent overfitting during training.

- Part TWO: Methods. These chapters cover the most established methods of deep learning on graphs from the basic to advanced settings. In Chap-

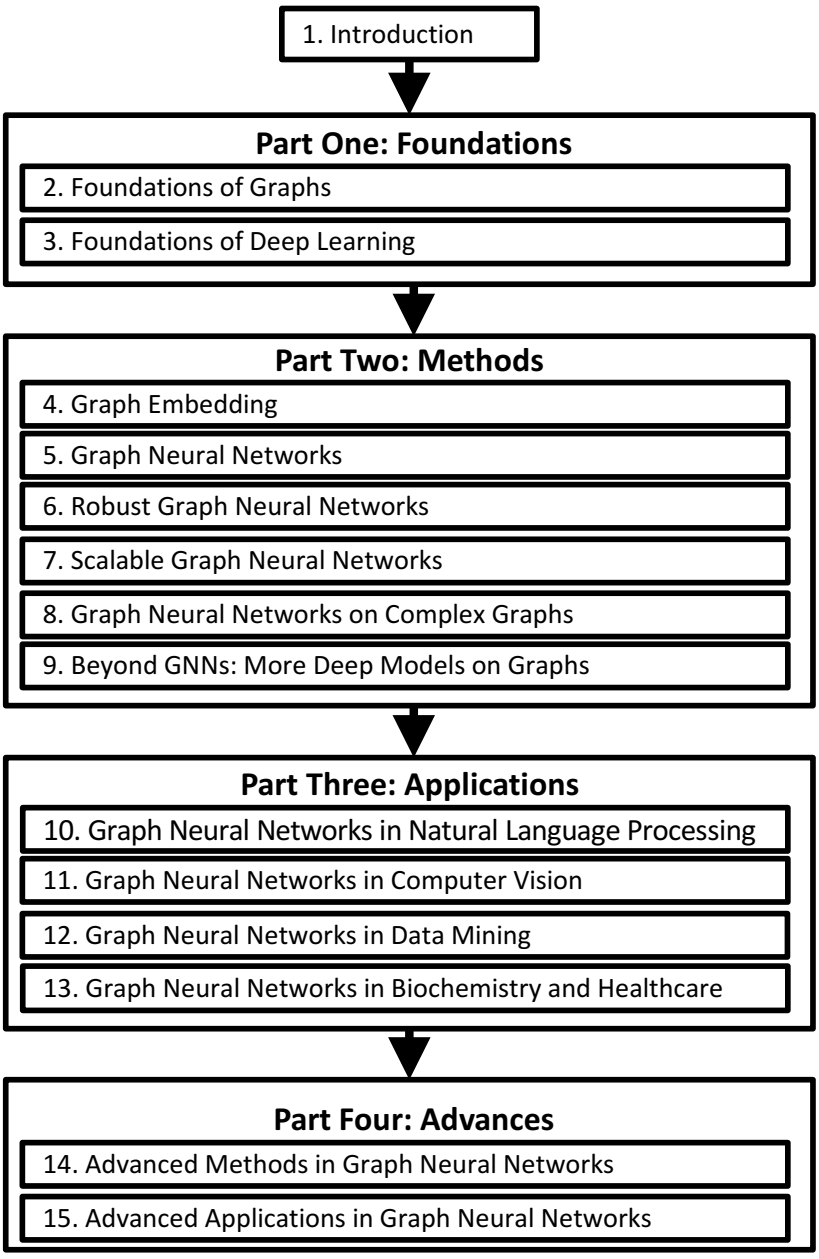


Figure 1.3 The high-level organization of the book

ter 4, we introduce a general graph embedding framework from the information preserving perspective, provide technical details on representative algorithms to preserve numerous types of information on graphs and present embedding approaches specifically designed for complex graphs. A typical graph neural network (GNN) model consists of two important operations, i.e., the graph filtering operation and the graph pooling operation. In Chapter 5, we review the state of the art graph filtering and pooling operations and discuss how to learn the parameters of GNNs for a given downstream task. As the generalizations of traditional deep models to graphs, GNNs inherit the drawbacks of traditional deep models and are vulnerable to adversarial attacks. In Chapter 6, we focus on concepts and definitions of graph adversarial attacks and detail representative adversarial attack and defense techniques. GNNs perform the recursive expansion of neighborhoods across layers. The expansion of the neighborhood for a single node can rapidly involve a large portion of the graph or even the whole graph. Hence, scalability is a pressing issue for GNNs. We detail representative techniques to scale GNNs in Chapter 7. In Chapter 8, we discuss GNN models that have been designed for more complicated graphs. To enable deep learning techniques to advance more tasks on graphs under wider settings, we introduce numerous deep graph models beyond GNNs in Chapter 9.

- Part THREE: Applications. Graphs provide a universal representation for real-world data; thus, methods of deep learning on graphs have been applied to various fields. These chapters present the most typical applications of GNNs, including Natural Language Processing in Chapter 10, Computer Vision in Chapter 11, Data Mining in Chapter 12 and Biochemistry and Healthcare in Chapter 13.
- Part FOUR: Advances. These chapters focus on recent advances in both methods and applications. In Chapter 14, we introduce advanced methods in GNNs such as expressiveness, depth, fairness, interpretability, and self-supervised learning. We discuss more areas that GNNs have been applied to, including Combinatorial Optimization, Physics, and Program Representation in Chapter 15.

1.4 Who Should Read the Book?

This book is easily accessible to readers with a computer science background. Basic knowledge of calculus, linear algebra, probability, and statistics can help understand its technical details in a better manner. This book has a wide range of target audiences. One target audience is senior undergraduate and gradu-

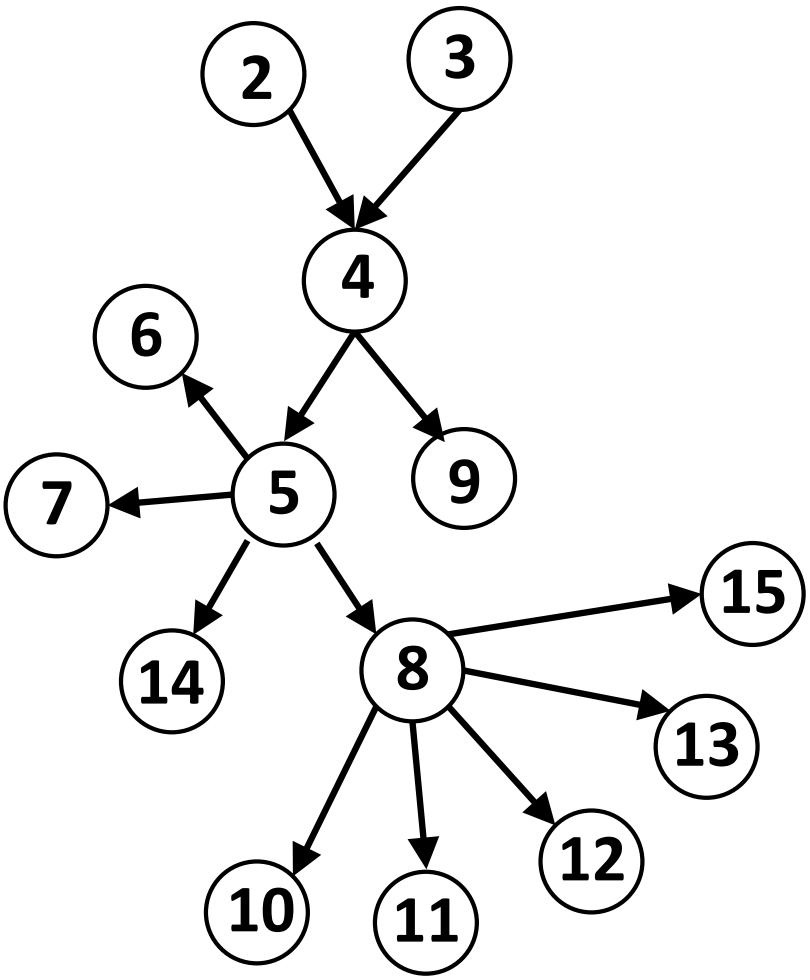


Figure 1.4 The guidance on how to read this book. Note that the number in the circle indicates the corresponding chapter as shown in Figure 1.3.

ate students interested in data mining, machine learning, and social network analysis. This book can be independently used for a graduate seminar course on deep learning on graphs. It also can be utilized as a part of a course. For example, Part TWO and Part FOUR can be considered as advanced topics in courses of data mining, machine learning, and social network analysis, and Part THREE can be used as advanced methods in solving traditional tasks in computer vision, natural language processing, and healthcare. Practition-

ers and project managers, who want to learn the basics and tangible examples of deep learning on graphs and adopt graph neural networks into their products and platforms, are also our target audience. Graph neural networks have been applied to benefit numerous disciplines beyond computer science. Thus, another target audience is researchers who do not have a computer science background but want to use graph neural networks to advance their disciplines.

Readers with different backgrounds and purposes of reading should go through the book differently. The suggested guidance on how to read this book is demonstrated in Figure 1.4. If readers aim to understand graph neural network methods on simple graphs (or Chapter 5), knowledge about foundations of graphs and deep learning and graph embedding is necessary (or Chapters 2, 3 and 4). Suppose readers want to apply graph neural networks to advance healthcare (or Chapter 13). In that case, they should first read prerequisite materials in foundations of graphs and deep learning, graph embedding and graph neural networks on simple and complex graphs. (or Chapters 2, 3, 4, 5, and 8). For Part THREE, we assume that the readers should have the necessary background in the corresponding application field. Besides, readers should feel free to skip some chapters if they have already been equipped with the corresponding background. Suppose readers know the foundations of graphs and deep learning. In that case, they should skip Chapters 2 and 3 and only read Chapters 4 and 5 to understand GNNs on simple graphs.

1.5 Feature Learning on Graphs: A Brief History

As aforementioned, to take advantage of traditional machine learning for computational tasks on graphs, it is desired to find vector node representations. As shown in Figure 1.5, there are mainly two ways to achieve this goal: feature engineering and feature learning. Feature engineering relies on hand-designed features such as node degree statistics, while feature learning is to learn node features automatically. On the one hand, we often do not have prior knowledge about what features are essential, especially for a given downstream task; thus, features from feature engineering could be suboptimal for the downstream task. The process requires an immense amount of human efforts. On the other hand, feature learning is to learn features automatically, and the downstream task can guide the process. Consequently, the learned features are likely to be suitable for the downstream task that often obtain better performance than those via feature engineering. Meanwhile, the process requires minimal human intervention and can be easily adapted to new tasks. Thus, feature learning on graphs has been extensively studied, and various types of feature learning tech-

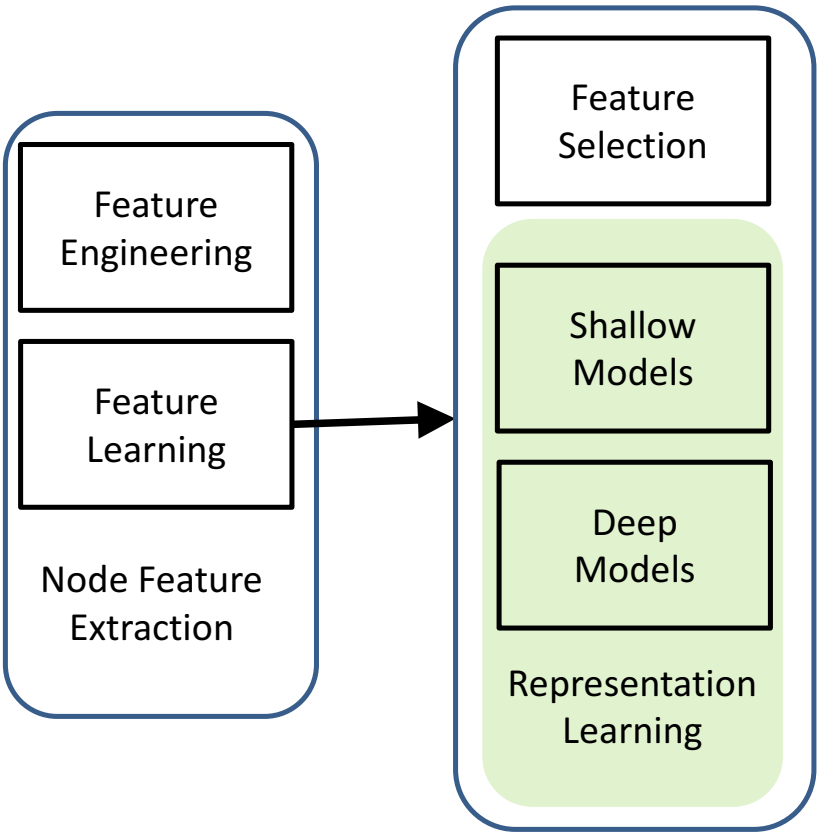


Figure 1.5 Node feature extraction

niques have been proposed to meet different requirements and scenarios. We roughly divide these techniques into feature selection on graphs that aims to remove irrelevant and redundant node features and representation learning on graphs that targets on generating a set of new node features. In this section, we briefly review these two groups of techniques that provide a general and historical context for readers to understand deep learning on graphs.

1.5.1 Feature Selection on Graphs

Real-world data is often high-dimensional, and there exist noisy, irrelevant, and redundant features (or dimensions), particularly when considering a given task. Feature selection aims to automatically select a small subset of features

that have minimal redundancy but maximal relevance to the target, such as the class labels under the supervised setting. In many applications, the original features are crucial for knowledge extraction and model interpretation. For example, in genetic analysis for cancer study, in addition to differentiating cancerous tissues, it is of greater importance to identify the genes (i.e., original features) that induce cancerogenesis. In these demanding applications, feature selection is particularly preferred since it maintains the original features, and their semantics usually provide critical insights to the learning problem. Traditional feature selection assumes that data instances are independent and identically distributed (i.i.d.). However, data samples in many applications are embedded in graphs that are inherently not i.i.d.. It has motivated the research area of feature selection on graphs. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set, we assume that each node is originally associated with a set of d features $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$. Feature selection on graphs aims to select K features from $\mathcal{F}$ to denote each node where $K \ll d$. The problem was first investigated under the supervised setting in (Tang and Liu, 2012a; Gu and Han, 2011). A linear classifier is employed to map from the selected features to the class labels, and a graph regularization term is introduced to capture structural information for feature selection. In particular, the term aims to ensure that connected nodes with the selected features can be mapped into similar labels. Then, the problem was further studied under the unsupervised setting (Wei et al., 2016, 2015; Tang and Liu, 2012b). In (Tang and Liu, 2012b), pseudo labels are extracted from the structural information that serve as the supervision to guide the feature selection process. In (Wei et al., 2016), both the node content and structural information are assumed to be generated from a set of high-quality features that can be obtained by maximizing the likelihood of the generation process. Later on, the problem has been extended from simple graphs to complex graphs such as dynamic graphs (Li et al., 2016), multi-dimensional graphs (Tang et al., 2013b), signed graphs (Cheng et al., 2017; Huang et al., 2020) and attributed graphs (Li et al., 2019b).

1.5.2 Representation Learning on Graphs

Different from feature selection on graphs, representation learning on graphs is to learn a set of new node features. It has been extensively studied for decades and has been dramatically accelerated by deep learning. In this subsection, we will give it a brief historical review from shallow models to deep models.

At the early stage, representation learning on graphs has been studied under the context of spectral clustering (Shi and Malik, 2000; Ng et al., 2002), graph-based dimension reduction (Belkin and Niyogi, 2003; Tenenbaum et al.,

2000; Roweis and Saul, 2000), and matrix factorization (Zhu et al., 2007; Tang et al., 2013a; Koren et al., 2009). In spectral clustering (Shi and Malik, 2000; Ng et al., 2002), data points are considered as nodes of a graph, and then clustering is to partition the graph into communities of nodes. One key step for spectral clustering is spectral embedding. It aims to embed nodes into a low-dimensional space where traditional clustering algorithms such as k-means can be applied to identify clusters. Techniques of graph-based dimension reduction can be directly applied to learn node representations. These approaches typically build an affinity graph using a predefined distance (or similarity) function based on the raw features of data samples. They aim to learn node representations to preserve structural information of this affinity graph. For example, IsoMap (Tenenbaum et al., 2000) is to preserve the global geometry via geodesics, while LLE (Roweis and Saul, 2000) and eigenmap (Belkin and Niyogi, 2003) are to preserve local neighborhoods in the affinity graph. The methods mentioned above often need to perform eigendecomposition on the affinity matrix (or adjacency matrix or Laplacian matrix). Thus, they are often computationally expensive. Matrix is one of the most popular approaches to denote graphs such as adjacency matrix, incidence matrix, and Laplacian matrix. As a result, matrix factorization can be naturally applied to learn node representations. Suppose we use the adjacency matrix to denote a graph. In that case, it aims to embed nodes into a low-dimensional space where the new node representations can be utilized to reconstruct the adjacency matrix. A document corpus can be denoted as a bipartite graph where documents and words are nodes, and an edge exists between a word and a document if the word appears in the document. LSI has employed truncated SVD to learn representations of documents and words (Deerwester et al., 1990). In recommender systems, interactions between users and items can be captured as a bipartite graph, and matrix factorization has been employed to learn representations of users and items for recommendations (Koren et al., 2009). Matrix factorization is also leveraged to learn node representations for node classification (Zhu et al., 2007; Tang et al., 2016a), link prediction (Menon and Elkan, 2011; Tang et al., 2013a) and community detection (Wang et al., 2011). A family of modern graph embedding algorithms we will introduce later can also be unified as matrix factorization (Qiu et al., 2018b).

Word2vec is a technique to generate word embeddings (Mikolov et al., 2013) . It takes a large corpus of text as input and produces a vector representation for each unique word in the corpus. The huge success of Word2vec in various natural language processing tasks has motivated increasing efforts to apply Word2vec, especially the Skip-gram model to learn node representations in the graph domain. DeepWalk (Perozzi et al., 2014) takes the first step

to achieve this goal. Specifically, nodes in a given graph are treated as words of an artificial language, and sentences in this language are generated by random walks. Then, it uses the Skip-gram model to learn node representations, which preserves the node co-occurrence in these random walks. After that, a large body of works have been developed in four major directions – (1) Developing advanced methods to preserve node co-occurrence (Tang et al., 2015; Grover and Leskovec, 2016; Cao et al., 2015); (2) Preserving other types of information such as node's structural role (Ribeiro et al., 2017), community information (Wang et al., 2017c) and node status (Ma et al., 2017; Lai et al., 2017; Gu et al., 2018); and (3) Designing frameworks for complex graphs such as directed graphs (Ou et al., 2016), heterogeneous graphs (Chang et al., 2015; Dong et al., 2017), bipartite graphs (Gao et al., 2018b), multi-dimensional graphs (Ma et al., 2018d), signed graphs (Wang et al., 2017b), hyper graphs (Tu et al., 2018), and dynamic graphs (Nguyen et al., 2018; Li et al., 2017a).

Given the power and the success of DNNs in representation learning, increasing efforts have been made to generalize DNNs to graphs. These methods are known as graph neural networks (GNNs) that can be roughly divided into spatial approaches and spectral approaches. Spatial approaches explicitly leverage the graph structure, such as spatially close neighbors, and the first spatial approach was introduced by (Scarselli et al., 2005). Spectral approaches utilize the spectral view of graphs by taking advantage of Graph Fourier Transform and the Inverse Graph Fourier Transform (Bruna et al., 2013). In the era of deep learning, GNNs have been rapidly developed in the following aspects.

- A huge amount of of new GNN models have been introduced including spectral approaches (Defferrard et al., 2016; Kipf and Welling, 2016a) and spatial approaches (Atwood and Towsley, 2016; Niepert et al., 2016; Gilmer et al., 2017; Monti et al., 2017; Veličković et al., 2017; Hamilton et al., 2017a).
- For graph-focused tasks such as graph classification, the representation of the whole graph is desired. Thus, numerous pooling methods have been introduced to obtain the graph representation from node representations (Li et al., 2015; Ying et al., 2018c; Gao and Ji, 2019; Ma et al., 2019b).
- Traditional DNNs are vulnerable to adversarial attacks. GNNs inherit this drawback. A variety of graph adversarial attacks have been studied (Zügner et al., 2018; Zügner and Günnemann, 2019; Dai et al., 2018; Ma et al., 2020a) and various defense techniques have been developed (Dai et al., 2018; Zhu et al., 2019a; Tang et al., 2019; Jin et al., 2020b).
- As aforementioned, scalability is a pressing issue for GNNs. Many strategies

have been studied to allow GNNs scale to large graphs (Chen et al., 2018a,b; Huang et al., 2018).
- GNN models have been designed to handle complex graphs such as heterogeneous graphs (Zhang et al., 2018b; Wang et al., 2019i; Chen et al., 2019b), bipartite graphs (He et al., 2019), multi-dimensional graphs (Ma et al., 2019c), signed graphs (Derr et al., 2018), hypergraphs (Feng et al., 2019b; Yadati et al., 2019), and dynamic graphs (Pareja et al., 2019).
- Diverse deep architectures have been generalized to graphs such as autoencoder (Wang et al., 2016; Cao et al., 2016), variational eutoencoder (Kipf and Welling, 2016b), recurrent neural networks (Tai et al., 2015; Liang et al., 2016) and generative adversarial networks (Wang et al., 2018a).
- As graphs are a universal data representation, GNNs have been applied to advance many fields such as natural language processing, computer vision, data mining, and healthcare.

1.6 Conclusion

In this chapter, we discussed the opportunities and challenges when we bridge deep learning with graphs that have motivated the focus of this book – deep learning on graphs. The book will cover the essential topics of deep learning on graphs that are organized into four parts to accommodate readers with diverse backgrounds and purposes of reading, including foundations, methods, applications, and advances. This book can benefit a broader range of readers, including senior undergraduate students, graduate students, practitioners, project managers, and researchers from various disciplines. To provide more context for readers, we give a brief historical review on the area of feature learning on graphs.

1.7 Further Reading

In this chapter, we have briefly reviewed the history of feature selection on graphs. If readers want to know more about feature selection, there are several important books (Liu and Motoda, 2012, 2007) and comprehensive surveys (Tang et al., 2014a). An open-source feature selection repository named *scikit-feature* has been developed, consisting of most of the popular feature selection algorithms (Li et al., 2017b). This is the first comprehensive book on the topic of deep learning on graphs. There are books on the general topics on deep learning (Goodfellow et al., 2016; Aggarwal, 2018), deep learning

on speech recognition (Yu and Deng, 2016; Kamath et al., 2019), and deep learning in natural language processing (Deng and Liu, 2018; Kamath et al., 2019).