Network Representation Learning: A Survey

Daokun Zhang, Jie Yin, Xingquan Zhu, *Senior Member, IEEE*, and Chengqi Zhang, *Senior Member, IEEE*

Abstract—With the widespread use of information technologies, information networks are becoming increasingly popular to capture complex relationships across various disciplines, such as social networks, citation networks, telecommunication networks, and biological networks. Analyzing these networks sheds light on different aspects of social life such as the structure of societies, information diffusion, and communication patterns. In reality, however, the large scale of information networks often makes network analytic tasks computationally expensive or intractable. Network representation learning has been recently proposed as a new learning paradigm to embed network vertices into a low-dimensional vector space, by preserving network topology structure, vertex content, and other side information. This facilitates the original network to be easily handled in the new vector space for further analysis. In this survey, we perform a comprehensive review of the current literature on network representation learning in the data mining and machine learning field. We propose new taxonomies to categorize and summarize the state-of-the-art network representation learning techniques according to the underlying learning mechanisms, the network information intended to preserve, as well as the algorithmic designs and methodologies. We summarize evaluation protocols used for validating network representation learning including published benchmark datasets, evaluation methods, and open source algorithms. We also perform empirical studies to compare the performance of representative algorithms on common datasets, and analyze their computational complexity. Finally, we suggest promising research directions to facilitate future study.

Index Terms—Information networks, graph mining, network representation learning, network embedding

1 Introduction

Information networks are becoming ubiquitous across a large spectrum of real-world applications in forms of social networks, citation networks, telecommunication networks and biological networks, *etc*. The scale of these networks ranges from hundreds to millions or even billions of vertices [1]. Analyzing information networks plays a crucial role in a variety of emerging applications across many disciplines. For example, in social networks, classifying users into meaningful social groups is useful for many important tasks, such as user search, targeted advertising and recommendations; in communication networks, detecting community structures can help better understand the rumor spreading process; in biological networks, inferring interactions between proteins can facilitate new treatments for diseases. Nevertheless, efficient analysis of these networks heavily relies on the ways how networks are represented. Often, a discrete adjacency matrix is used to represent a network, which only captures neighboring relationships between vertices. Indeed, this simple representation cannot embody more complex, higher-order structure relationships, such as paths, frequent substructure, *etc*. As a result, such a traditional routine often makes many network analytic tasks computationally expensive and intractable over large-scale networks. Taking community detection as an example, most existing algorithms involve calculating the spectral decomposition of a matrix [2] with at least quadratic time complexity with respect to the number of vertices. This computational overhead makes algorithms hard to scale to large-scale networks with millions of vertices.

Recently, network representation learning (NRL) has aroused a lot of research interest. NRL aims to learn latent, low-dimensional representations of network vertices, while preserving network topology structure, vertex content, and other side information. After new vertex representations are learned, network analytic tasks can be easily and efficiently carried out by applying conventional vector-based machine learning algorithms to the new representation space. This obviates the necessity for deriving complex algorithms that are applied directly on the original network.

Earlier work related to network representation learning dates back to the early 2000s, when researchers proposed graph embedding algorithms as part of dimensionality reduction techniques. Given a set of i.i.d. (independent and identically distributed) data points as input, graph embedding algorithms first calculate the similarity between pairwise data points to construct an affinity graph, e.g., the k-nearest neighbor graph, and then embed the affinity graph into a new space having much lower dimensionality. The idea is to find a low-dimensional manifold structure hidden in the high-dimensional data geometry reflected by the constructed graph, so that connected vertices are kept closer to

- *D. Zhang and C. Zhang are with the Centre for Artificial Intelligence, FEIT, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: Daokun.Zhang@student.uts.edu.au, Chengqi.Zhang@uts.edu.au.*
- *J. Yin is with the Discipline of Business Analytics, The University of Sydney, Sydney, NSW 2006, Australia. E-mail: jie.yin@sydney.edu.au.*
- *X. Zhu is with the Department of CEECS, Florida Atlantic University, Boca Raton, FL 33431. E-mail: xqzhu@cse.fau.edu.*

Manuscript received 3 Dec. 2017; revised 26 Apr. 2018; accepted 12 June 2018. Date of publication 25 June 2018; date of current version 28 Feb. 2020. (Corresponding author: Jie Yin.)
Recommended for acceptance by Y. Zheng.
Digital Object Identifier no. 10.1109/TBDATA.2018.2850013

each other in the new embedding space. Isomap [3], Locally Linear Embedding (LLE) [4] and Laplacian Eigenmap [5] are examples of algorithms based on this rationale. However, graph embedding algorithms are designed on i.i.d. data mainly for dimensionality reduction purpose. Most of these algorithms usually have at least quadratic time complexity with respect to the number of vertices, so the scalability is a major issue when they are applied to large-scale networks.

Since 2008, significant research efforts have shifted to the development of effective and scalable representation learning techniques that are directly designed for complex information networks. Many NRL algorithms, e.g., [6], [7], [8], [9], have been proposed to embed existing networks, showing promising performance for various applications. These algorithms embed a network into a latent, low-dimensional space that preserves structure proximity and attribute affinity, such that the original vertices of the network can be represented as low-dimensional vectors. The resulting compact, low-dimensional vector representations can be then taken as features to any vector-based machine learning algorithms. This paves the way for a wide range of network analytic tasks to be easily and efficiently tackled in the new vector space, such as node classification [10], [11], link prediction [12], [13], clustering [2], recommendation [14], [15], similarity search [16], and visualization [17]. Using vector representation to represent complex networks has now been gradually advanced to many other domains, such as point-of-interest recommendation in urban computing [15], and knowledge graph search [18] in knowledge engineering and database systems.

1.1 Challenges

Despite its great potential, network representation learning is inherently difficult and is confronted with several key challenges that we summarize as follows.

Structure-Preserving. To learn informative vertex representations, network representation learning should preserve network structure, such that vertices similar/close to each other in the original structure space should also be represented similarly in the learned vector space. However, as stated in [19], [20], the structure-level similarity between vertices is reflected not only at the local neighborhood structure but also at the more global community structure. Therefore, the local and global structure should be simultaneously preserved in network representation learning.

Content-Preserving. Besides structure information, vertices of many networks are attached with rich content on attributes. Vertex attributes not only exert huge impacts on the forming of networks, but also provide direct evidence to measure attribute-level similarity between vertices. Therefore, if properly imported, attribute content can compensate network structure to render more informative vertex representations. However, due to heterogeneity of the two information sources, how to effectively leverage vertex attributes and make them compensate rather than deteriorate network structure is an open research problem.

Data Sparsity. For many real-world information networks, due to the privacy or legal restrictions, the problem of data sparsity exists in both network structure and vertex content. At the structure level, only very limited links are sometimes observed, making it difficult to discover the structure-level relatedness between vertices that are not explicitly connected. At the vertex content level, many values of vertex attributes are usually missing, which increases the difficulty of measuring content-level vertex similarity. Thus, it is challenging for network representation learning to overcome the data sparsity problem.

Scalability. Real-world networks, social networks in particular, consist of millions or billions of vertices. The large scale of the networks challenges not only the traditional network analytic tasks but also the newborn network representation learning task. Without special concern, learning vertex representations for large-scale networks with limited computing resources may cost months of time, which is practically infeasible, especially for the case involving a large number of trails for tuning parameters. Therefore, it is necessary to design NRL algorithms that can learn vertex representations efficiently and meanwhile guarantee the effectiveness for large-scale networks.

1.2 Our Contribution

This survey provides a comprehensive up-to-date review of the state-of-the-art network representation learning techniques, with a focus on the learning of vertex representations. It covers not only early work on preserving network structure, but also a new surge of recent studies that incorporate vertex content and/or vertex labels as auxiliary information into the learning process of network embedding. By doing so, we hope to provide a useful guideline for the research community to better understand (1) new taxonomies of network representation learning methods, (2) the characteristics, uniqueness, and the niche of different types of network embedding methods, and (3) the resources and future challenges to stimulate research in the area. In particular, this survey has four major contributions:

- We propose new taxonomies to categorize existing network representation learning techniques according to the underlying learning mechanisms, the network information intended to preserve, as well as the algorithmic designs and methodologies. As a result, this survey provides new angles to better understand the existing work.
- We provide a detailed and thorough study of the state-of-the-art network representation learning algorithms. Compared to the existing graph embedding surveys, we not only review a more comprehensive set of research work on network representation learning, but also provide multifaceted algorithmic perspectives to understand the advantages and disadvantages of different algorithms.
- We summarize evaluation protocols used for validating network representation learning techniques, including published benchmark datasets, evaluation methods, and open source algorithms. We also perform empirical studies to compare the performance of representative algorithms, along with a detailed analysis of computational complexity.
- To foster future research, we suggest six promising future research directions for network representation learning, and summarize the limitations of current research work and propose new research ideas for each direction.

TABLE 1
A Summary of Common Notations

G	The given information network
V	Set of vertices in the given information network
E	Set of edges in the given information network
$\|V\|$	Number of vertices
$\|E\|$	Number of edges
m	Number of vertex attributes
d	Dimension of learned vertex representations
$X \in \mathbb{R}^{\|V\| \times m}$	The vertex attribute matrix
$\mathcal{Y}$	Set of vertex labels
$\|\mathcal{Y}\|$	Number of vertex labels
$Y \in \mathbb{R}^{\|V\| \times \|\mathcal{Y}\|}$	The vertex label matrix

1.3 Related Surveys and Differences

A few graph embedding and representation learning related surveys exist in the recent literature. The first is [21], which reviews a few representative methods for network representation learning and visits some key concepts around the idea of representation learning and its connections to other related field such as dimensionality reduction, deep learning, and network science. [22] categorizes representative network embedding algorithms from a methodology perspective. [23] reviews a few representation learning methods for embedding individual vertices as well as subgraphs, especially those inspired by deep learning, within an encoder-decoder framework. Yet, the majority of embedding algorithms reviewed by these surveys primarily preserve network structure. Recently, [24], [25] extend to cover work leveraging other side information, such as vertex attributes and/or vertex labels, to harness representation learning.

In summary, existing surveys have the following limitations. First, they typically focus on one single taxonomy to categorize the existing work. None of them provides a multi-faceted view to analyze the state-of-the-art network representation learning techniques and to compare their advantages and disadvantages. Second, existing surveys do not have in-depth analysis of algorithm complexity and optimization methods, or they do not provide empirical results to compare the performance of different algorithms. Third, there is a lack of summary on available resources, such as publicly available datasets and open source algorithms, to facilitate future research. In this work, we provide the most comprehensive survey to bridge the gap. We believe that this survey will benefit both researchers and practitioners to gain a deep understanding of different approaches, and provide rich resources to foster future research in the field.

1.4 Organization of the Survey

The rest of this survey is organized as follows. In Section 2, we provide preliminaries and definitions required to understand the problem and the models discussed next. Section 3 proposes new taxonomies to categorize the existing network representation learning techniques. Sections 4 and 5 review representative algorithms in two categories, respectively. A list of successful applications of network representation learning are discussed in Section 6. In Section 7, we summarize the evaluation protocols used to validate network representation learning, along with a comparison of algorithm performance and complexity. We discuss potential research directions in Section 8, and conclude the survey in Section 9.

2 Notations and Definitions

In this section, as preliminaries, we first define important terminologies that are used to discuss the models next, followed by a formal definition of the network representation learning problem. For ease of presentation, we first define a list of common notations that will be used throughout the survey, as shown in Table 1.

Definition 1 (Information Network). *An information network is defined as $G = (V, E, X, Y)$, where V denotes a set of vertices, and $|V|$ denotes the number of vertices in network G. $E \subseteq (V \times V)$ denotes a set of edges connecting the vertices. $X \in \mathbb{R}^{|V| \times m}$ is the vertex attribute matrix, where m is the number of attributes, and the element X_{ij} is the value of the ith vertex on the jth attribute. $Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$ is the vertex label matrix with $\mathcal{Y}$ being a set of labels. If the ith vertex has the kth label, the element $Y_{ik} = 1$; otherwise, $Y_{ik} = -1$. Due to privacy concern or information access difficulty, vertex attribute matrix X is often sparse and vertex label matrix Y is usually unobserved or partially observed. For each $(v_i, v_j) \in E$, if information network G is undirected, we have $(v_j, v_i) \in E$; if G is directed, (v_j, v_i) unnecessarily belongs to E.[1] Each edge $(v_i, v_j) \in E$ is also associated to a weight w_{ij}, which is equal to 1, if the information network is binary (unweighted).*

Intuitively, the generation of information networks is not groundless, but guided or dominated by certain latent mechanisms. Although the latent mechanisms are hardly known, they can be reflected by some network properties that widely exist in information networks. Hence, the common network properties are essential for the learning of vertex representations that are informative to accurately interpret information networks. Below, we introduce several common network properties.

Definition 2 (First-order Proximity). *The first-order proximity is the local pairwise proximity between two connected vertices [1]. For each vertex pair (v_i, v_j), if $(v_i, v_j) \in E$, the first-order proximity between v_i and v_j is w_{ij}; otherwise, the first-order proximity between v_i and v_j is 0. The first-order proximity captures the direct neighbor relationships between vertices.*

Definition 3 (Second-order Proximity and High-order Proximity). *The second-order proximity captures the 2-step relations between each pair of vertices [1]. For each vertex pair (v_i, v_j), the second order proximity is determined by the number of common neighbors shared by the two vertices, which can also be measured by the 2-step transition probability from v_i to v_j equivalently. Compared with the second-order proximity, the high-order proximity [26] captures more global structure, which explores k-step $(k \geq 3)$ relations between each pair of vertices. For each vertex pair (v_i, v_j), the higher-order proximity is measured by the k-step $(k \geq 3)$ transition probability from vertex v_i to vertex v_j, which can also be reflected by the number of k-step $(k \geq 3)$ paths from v_i to v_j. The second-order and high-order proximity capture the similarity between a pair of, indirectly connected, vertices with similar structural contexts.*

Definition 4 (Structural Role Proximity). *The structural role proximity depicts similarity between vertices serving as the*

1. Without any specific declaration, the networks discussed in this survey are assumed to be undirected.

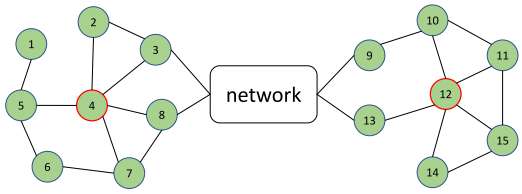


Fig. 1. An illustrative example of structural role proximity. Vertex 4 and vertex 12 have similar structural roles, but are located far away from each other.

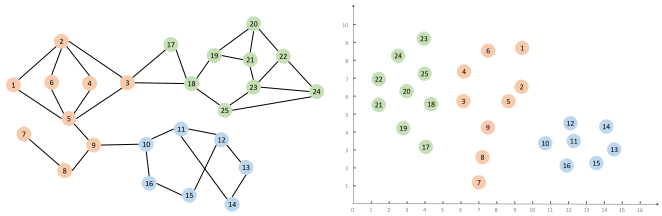
similar roles in their neighborhood, such as edge of a chain, center of a star, and a bridge between two communities. In communication and traffic networks, vertices' structural roles are important to characterize their properties. Different from the first-order, second-order and high-order proximity, which capture the similarity between vertices close to each other in the network, the structural role proximity tries to discover the similarity between distant vertices while sharing the equivalent structural roles. As is shown in Fig. 1, vertex 4 and vertex 12 are located far away from each other, while they serve as the same structural role, center of a star. Thus, they have high structural role proximity.

Definition 5 (Intra-community Proximity). *The intra-community proximity is the pairwise proximity between vertices in a same community. Many networks have community structure, where vertex-vertex connections within the same community are dense, but connections to vertices outside the community are sparse [27]. As cluster structure, a community preserves certain kinds of common properties of vertices within it. For example, in social networks, communities might represent social groups by interest or background; in citation networks, communities might represent related papers on a same topic. The intra-community proximity captures such cluster structure by preserving the common property shared by vertices within a same community [28].*

Vertex Attribute. In addition to network structure, vertex attributes can provide direct evidence to measure content-level similarity between vertices. As shown in [7], [20], [29], vertex attributes and network structure can help each other filter out noisy information and compensate each other to jointly learn informative vertex representations.

Vertex Label. Vertex labels provide direct information about the semantic categorization of each network vertex to certain classes or groups. Vertex labels are strongly influenced by and inherently correlated to both network structure and vertex attributes [30]. Though vertex labels are usually partially observed, when coupled with network structure and vertex attributes, they encourage a network structure and vertex attribute consistent labeling, and help learn informative and discriminative vertex representations.

Definition 6 (Network Representation Learning). *Given an information network $G = (V, E, X, Y)$, by integrating network structure in E, vertex attributes in X and vertex labels in Y (if available), the task of network representation learning is to learn a mapping function $f : v \longmapsto r_v \in \mathbb{R}^d$, where r_v is the learned vector representation of vertex v, and d is the dimension of the learned representation. The transformation f preserves the original network information, such that two vertices similar in the original network should also be represented similarly in the learned vector space.*



(a) Input: Information Network (b) Output: Vertex Representations

Fig. 2. A conceptual view of network representation learning. Vertices in (a) are indexed using their ID and color coded based on their community information. The network representation learning in (b) transforms all vertices into a two-dimensional vector space, such that vertices with structural proximity are close to each other in the new embedding space.

The learned vertex representations should satisfy the following conditions: (1) *low-dimensional*, i.e., $d \ll |V|$, in other words, the dimension of learned vertex representations should be much smaller than the dimension of the original adjacency matrix representation for memory efficiency and the scalability of subsequent network analytic tasks; (2) *informative*, i.e., the learned vertex representations should preserve vertex proximity reflected by network structure, vertex attributes, and vertex labels (if available); (3) *continuous*, i.e., the learned vertex representations should have continuous real values to support subsequent network analytic tasks, like vertex classification, vertex clustering, or anomaly detection, and have smooth decision boundaries to ensure the robustness of these tasks.

Fig. 2 demonstrates a conceptual view of network representation learning, using a toy network. In this case, only network structure is considered to learn vertex representations. Given an information network shown in Fig. 2a, the objective of NRL is to embed all network vertices into a low-dimensional space, as depicted in Fig. 2b. In the embedding space, vertices with structural proximity are represented closely to each other. For example, as vertex 7 and vertex 8 are directly connected, the first-order proximity enforces them close to each other in the embedding space. Though vertex 2 and vertex 5 are not directly connected, they are also embedded closely to each other because they have high second-order proximity, which is reflected by 4 common neighbors shared by these two vertices. Vertex 20 and vertex 25 are not directly connected, nor do they share common direct neighbors. However, they are connected by many k-step paths ($k \geq 3$), which proves that they have high-order proximity. Thus, vertex 20 and vertex 25 also have close embeddings. Different from other vertices, vertex 10–16 clearly belong to the same community in the original network. This intra-community proximity guarantees the images of these vertices also exhibit a clear cluster structure in the embedding space.

3 Categorization

In this section, we propose a new taxonomy to categorize existing network representation learning techniques in the literature, as shown in Fig. 3. The first layer of the taxonomy is based on whether vertex labels are provided for learning. According to this, we categorize network representation learning into two groups: *unsupervised network representation learning* and *semi-supervised network representation learning*.

Unsupervised Network Representation Learning. In this setting, there are no labeled vertices provided for learning

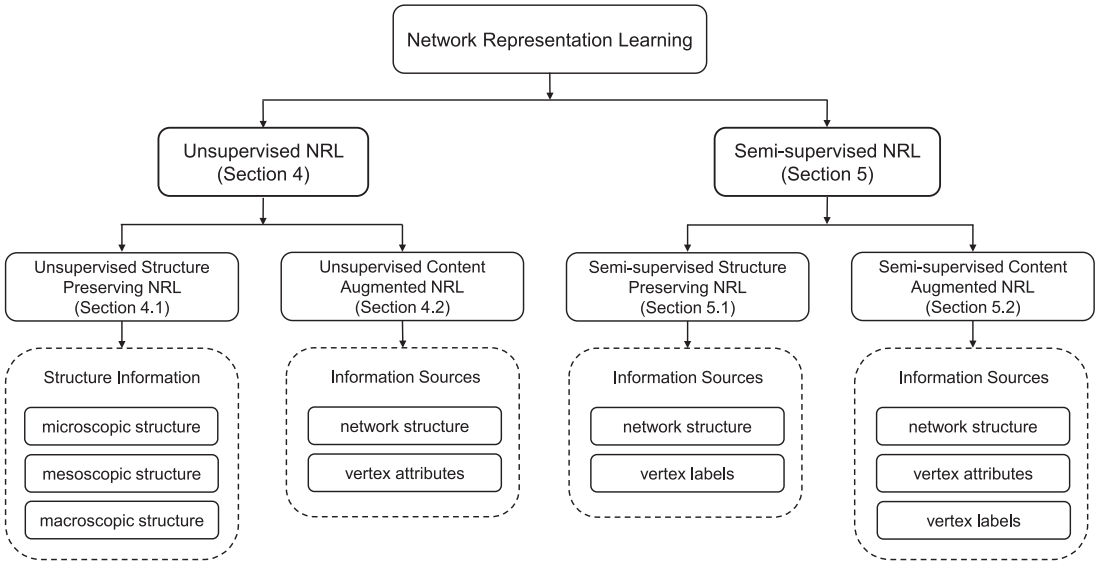


Fig. 3. The proposed taxonomy to summarize network representation learning techniques. We categorize network representation learning into two groups, *unsupervised network representation learning* and *semi-supervised network representation learning*, depending on whether vertex labels are available for learning. For each group, we further categorize methods into two subgroups, depending on whether the representation learning is based on network topology structure only, or augmented with information from node content.

vertex representations. Network representation learning is therefore considered as a generic task independent of subsequent learning, and vertex representations are learned in an unsupervised manner.

Most of the existing NRL algorithms fall into this category. After vertex representations are learned in a new embedding space, they are taken as features to any vector-based algorithms for various learning tasks. Unsupervised NRL algorithms can be further divided into two subgroups based on the type of network information available for learning: unsupervised structure preserving methods that preserve only network structure, and unsupervised content augmented methods that incorporate vertex attributes and network structure to learn joint vertex embeddings.

Semi-Supervised Network Representation Learning. In this case, there exist some labeled vertices for representation learning. Because vertex labels play an essential role in determining the categorization of each vertex with strong correlations to network structure and vertex attributes, semi-supervised network representation learning is proposed to take advantage of vertex labels available in the network for seeking more effective joint vector representations.

In this setting, network representation learning is coupled with supervised learning tasks such as vertex classification. A unified objective function is often formulated to simultaneously optimize the learning of vertex representations and the classification of network vertices. Therefore, the learned vertex representations can be both informative and discriminative with respect to different categories. Semi-supervised NRL algorithms can also be categorized into two subgroups, semi-supervised structure preserving methods and semi-supervised content augmented methods.

Table 2 summarizes all NRL algorithms, according to the information sources that they use for representation learning. In general, there are three main types of information sources: network structure, vertex attributes, and vertex labels. Most of the unsupervised NRL algorithms focus on preserving network structure for learning vertex representations, and only a few algorithms (e.g., TADW [7], HSCA [8]) attempt to leverage vertex attributes. By contrast, under the semi-supervised learning setting, half of the algorithms intend to couple vertex attributes with network structure and vertex labels to learn vertex representations. On both settings, most of the algorithms focus on preserving microscopic structure, while very few algorithms (e.g., M-NMF [28], DP [41], HARP [42]) attempt to take advantage of the mesoscopic and macroscopic structure.

Approaches to network representation learning in the above two different settings can be summarized into five categories from algorithmic perspectives.

(1) *Matrix factorization based methods*. Matrix factorization based methods represent the connections between network vertices in the form of a matrix and use matrix factorization to obtain the embeddings. Different types of matrices are constructed to preserve network structure, such as the k-step transition probability matrix, the modularity matrix, or the vertex-context matrix [7]. By assuming that such high-dimensional vertex representations are only affected by a small quantity of latent factors, matrix factorization is used to embed the high-dimensional vertex representations into a latent, low-dimensional structure preserving space.

Factorization strategies vary across different algorithms according to their objectives. For example, in the Modularity Maximization method [31], eigen decomposition is performed on the modularity matrix to learn community indicative vertex representations [53]; in the TADW algorithm [7], inductive matrix factorization [54] is carried out on the vertex-context matrix to simultaneously preserve vertex textual features and network structure in the learning of vertex representations. Although matrix factorization based methods have been proved effective in learning informative vertex representations, the scalability is a major bottleneck because carrying out

TABLE 2
A Summary of NRL Algorithms According to the Information Sources They Use for Learning

Category	Algorithms	Network Structure				Vertex Attributes	Vertex Labels
		Microscopic	Mesoscopic		Macroscopic		
			Structural Role Proximity	Intra-community Proximity			
Unsupervised	Social Dim. [31], [32], [33]			✓			
	DeepWalk [6]	✓					
	LINE [1]	✓					
	GraRep [26]	✓					
	DNGR [9]	✓					
	SDNE [19]	✓					
	node2vec [34]	✓					
	HOPE [35]	✓					
	APP [36]	✓					
	M-NMF [28]	✓		✓			
	GraphGAN [37]	✓					
	struct2vec [38]		✓				
	GraphWave [39]		✓				
	SNS [40]	✓	✓				
	DP [41]	✓			✓		
	HARP [42]	✓			✓		
	TADW [7]	✓				✓	
	HSCA [8]	✓				✓	
	pRBM [29]	✓				✓	
	UPP-SNE [43]	✓				✓	
	PPNE [44]	✓				✓	
Semi-supervised	DDRW [45]	✓					✓
	MMDW [46]	✓					✓
	TLINE [47]	✓					✓
	GENE [48]	✓					✓
	SemiNE [49]	✓					✓
	TriDNR [50]	✓				✓	✓
	LDE [51]	✓				✓	✓
	DMF [8]	✓				✓	✓
	Planetoid [52]	✓				✓	✓
	LANE [30]	✓				✓	✓

factorization on a matrix with millions of rows and columns is memory intensive and computationally expensive or, sometime, even infeasible.

(2) *Random walk based methods*. For scalable vertex representation learning, random walk is exploited to capture structural relationships between vertices. By performing truncated random walks, an information network is transformed into a collection of vertex sequences, in which, the occurrence frequency of a vertex-context pair measures the structural distance between them. Borrowing the idea of word representation learning [55], [56], vertex representations are then learned by using each vertex to predict its contexts. DeepWalk [6] is the pioneer work in using random walks to learn vertex representations. node2vec [34] further exploits a biased random walk strategy to capture more flexible contextual structure.

As the extensions of the structure only preserving version, algorithms like DDRW [45], GENE [48] and SemiNE [49] incorporate vertex labels with network structure to harness representation learning, PPNE [44] imports vertex attributes, and Tri-DNR [50] enforces the model with both vertex labels and attributes. As these models can be trained in an online manner, they have great potential to scale up.

(3) *Edge modeling based methods*. Different from approaches that use matrix or random walk to capture network structure, the edge modeling based methods directly learn vertex representations from vertex-vertex connections. For capturing the first-order and second-order proximity, LINE [1] models a joint probability distribution and a conditional probability distribution, respectively, on connected vertices. To learn the representations of linked documents, LDE [51] models the document-document relationships by maximizing the conditional probability between connected documents. pRBM [29] adapts the RBM [57] model to linked data by making the hidden RBM representations of connected vertices similar to each other. GraphGAN [37] adopts Generative Adversarial Nets (GAN) [58] to accurately model the vertex connectivity probability. Edge modeling based methods are more

TABLE 3
A Categorization of NRL Algorithms from Methodology Perspectives

Methodology	Algorithms	Advantage	Disadvantage
Matrix Factorization	Social Dim. [31], [32], GraRep [26], HOPE [35], GraphWave [39], M-NMF [28], TADW [7], HSCA [20], MMDW [46], DMF [8], LANE [30]	capture global structure	high time and memory cost
Random Walk	DeepWalk [6], node2vec [34], APP [36], DDRW [45], GENE [48], TriDNR [50], UPP-SNE [43], struct2vec [38], SNS [40], PPNE [44], SemiNE [49]	relatively efficient	only capture local structure
Edge Modeling	LINE [1], TLINE [47], LDE [51], pRBM [29], GraphGAN [37]	efficient	only capture local structure
Deep Learning	DNGR [9], SDNE [19]	capture non-linearity	high time cost
Hybrid	DP [41], HARP [42], Planetoid [52]	capture global structure	

efficient compared to matrix factorization and random walk based methods. However, these methods cannot capture global network structure as they only consider observable vertex connectivity information.

(4) *Deep learning based methods*. To extract complex structure features and learn deep, highly non-linear vertex representations, deep learning techniques [59], [60] are also applied to network representation learning. For example, DNGR [9] applies the *stacked denoising autoencoders* (SDAE) [60] on the high-dimensional matrix representations to learn deep low-dimensional vertex representations. SDNE [19] uses a semi-supervised deep autoencoder model [59] to model non-linearity in network structure. Deep learning based methods have the ability to capture non-linearity in networks, but their computational time cost is usually high. Traditional deep learning architectures are designed for 1D, 2D, or 3D euclidean structured data, but efficient solutions need to be developed on non-euclidean structured data like graphs.

(5) *Hybrid methods*. Some other methods make use of a mixture of above methods to learn vertex representations. For example, DP [41] enhances spectral embedding [5] and DeepWalk [6] with the degree penalty principle to preserve the macroscopic scale-free property. HARP [42] takes advantage of random walk based methods (DeepWalk [6] and node2vec [34]) and edge modeling based method (LINE [1]) to learn vertex representations from small sampled networks to the original network.

We summarize all five categories of network representation learning techniques and compare their advantages and disadvantages in Table 3.

4 Unsupervised Network Representation Learning

In this section, we review unsupervised network representation learning methods by separating them into two sections, as outlined in Fig. 3. After that, we summarize key characteristics of the methods and compare their differences across the two categories.

4.1 Unsupervised Structure Preserving Network Representation Learning

Structure preserving network representation learning refers to methods that intend to preserve network structure, in the sense that vertices close to each other in the original network space should be represented similarly in the new embedding space. In this category, research efforts have been focused on designing various models to capture structure information conveyed by the original network as much as possible.

We summarize network structure considered for learning vertex representations into three types: (i) *microscopic structure*, which includes local closeness proximity, i.e., the first-order, second-order, and high-order proximity, (ii) *mesoscopic structure*, which captures structural role proximity and the intra-community proximity, and (iii) *macroscopic structure*, which captures global network properties, such as the scale-free property or small world property. The following sections are organized according to our categorization of network structure, as depicted in Fig. 4 .

4.1.1 Microscopic Structure Preserving NRL

This category of NRL algorithms aim to preserve local structure information among directly or indirectly connected vertices in their neighborhood, including first-order, second-order, and high-order proximity. The first-order proximity captures the homophily, i.e., directly connected vertices tend to be similar to each other, while the second-order and high-order proximity captures the similarity between vertices sharing common neighbors. Most of structure preserving NRL algorithms fall into this category.

DeepWalk. DeepWalk [6] generalizes the idea of the Skip-Gram model [55], [56] that utilizes word context in sentences to learn latent representations of words, to the learning of latent vertex representations in networks, by making an analogy between natural language sentence and short random walk sequence. The workflow of DeepWalk is given in Fig. 5. Given a random walk sequence with length L, $\{v_1, v_2, \ldots, v_L\}$, following Skip-Gram, DeepWalk learns the representation of vertex v_i by using it to predict its context vertices, which is achieved by the optimization problem

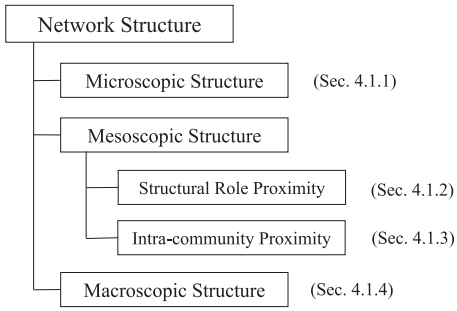


Fig. 4. Categorization of network structure.

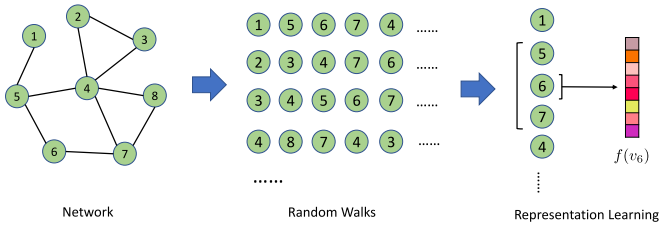


Fig. 5. The workflow of DeepWalk. It first generates random walk sequences from a given network, and then applies the Skip-Gram model to learn vertex representations.

$$\min_{f} \ -\log \Pr(\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i | f(v_i)), \tag{1}$$

where $\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i$ are the context vertices of vertex v_i within t window size. Making conditional independence assumption, the probability $\Pr(\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i | f(v_i))$ is approximated as

$$\Pr(\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i | f(v_i)) = \prod_{j=i-t, j \neq i}^{i+t} \Pr(v_j | f(v_i)). \tag{2}$$

Following the DeepWalk's learning architecture, vertices that share similar context vertices in random walk sequences should be represented closely in the new embedding space. Considering the fact that context vertices in random walk sequences describe neighborhood structure, DeepWalk actually represents vertices sharing similar neighbors (direct or indirect) closely in the embedding space, so the second-order and high-order proximity is preserved.

Large-Scale Information Network Embedding (LINE). Instead of exploiting random walks to capture network structure, LINE [1] learns vertex representations by explicitly modeling the first-order and second-order proximity. To preserve the first-order proximity, LINE minimizes the following objective:

$$O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot)). \tag{3}$$

For each vertex pair v_i and v_j with $(v_i, v_j) \in E$, $p_1(\cdot, \cdot)$ is the joint distribution modeled by their latent embeddings r_{v_i} and r_{v_j}. $\hat{p}_1(v_i, v_j)$ is the empirical distribution between them. $d(\cdot, \cdot)$ is the distance between two distributions.

To preserve the second-order proximity, LINE minimizes the following objective:

$$O_2 = \sum_{v_i \in V} \lambda_i d(\hat{p}_2(\cdot | v_i), p_2(\cdot | v_i)), \tag{4}$$

where $p_2(\cdot | v_i)$ is the context conditional distribution for each $v_i \in V$ modeled by vertex embeddings, $\hat{p}_2(\cdot | v_i)$ is the empirical conditional distribution and λ_i is the prestige of vertex v_i. Here, vertex context is determined by its neighbors, i.e., for each v_j, v_j is v_i's context, if and only if $(v_i, v_j) \in E$.

By minimizing these two objectives, LINE learns two kinds of vertex representations that preserve the first-order and second-order proximity, and takes their concatenation as the final vertex representation.

GraRep. Following the idea of DeepWalk [6], GraRep [26] extends the skip-gram model to capture the high-order proximity, i.e., vertices sharing common k-step neighbors $(k \geq 1)$ should have similar latent representations. Specifically, for each vertex, GraRep defines its k-step neighbors $(k \geq 1)$ as context vertices, and for each $1 \leq k \leq K$, to learn k-step vertex representations, GraRep employs the matrix factorization version of skip-gram

$$\left[U^k, \Sigma^k, V^k\right] = SVD(X^k), \tag{5}$$

where X^k is the log k-step transition probability matrix. The k-step representation for vertex v_i is constructed as the ith row of matrix $U_d^k (\Sigma_d^k)^{\frac{1}{2}}$, where U_d^k is the first-d columns of U^k and Σ_d^k is the diagonal matrix composed of the top d singular values. After k-step vertex representations are learned, GraRep concatenates them together as the final vertex representations.

Deep Neural Networks for Graph Representations (DNGR). To overcome the weakness of truncated random walks in exploiting vertex contextual information, i.e., the difficulty in capturing correct contextual information for vertices at the boundary of sequences and the difficulty in determining the walk length and the number of walks, DNGR [9] utilizes the random surfing model to capture contextual relatedness between each pair of vertices and preserves them into $|V|$-dimensional vertex representations X. To extract complex features and model non-linearities, DNGR applies the *stacked denoising autoencoders* [60] to the high-dimensional vertex representations X to learn deep low-dimensional vertex representations.

Structural Deep Network Embedding (SDNE). SDNE [19] is a deep learning based approach that uses a semi-supervised deep autoencoder model to capture non-linearity in network structure. In the unsupervised component, SDNE learns the second-order proximity preserving vertex representations via reconstructing the $|V|$-dimensional vertex adjacent matrix representations, which tries to minimize

$$\mathcal{L}_{2nd} = \sum_{i=1}^{|V|} \| (r_{v_i}^{(0)} - \hat{r}_{v_i}^{(0)}) \odot \boldsymbol{b}_i \|_2^2, \tag{6}$$

where $r_{v_i}^{(0)} = S_{i:}$ is the input representation and $\hat{r}_{v_i}^{(0)}$ is the reconstructed representation. $\boldsymbol{b}_i$ is a weight vector used to penalize construction error more on non-zero elements of S.

In the supervised component, SDNE imports the first-order proximity by penalizing the distance between connected vertices in the embedding space. The loss function for this objective is defined as

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{|V|} S_{ij} \| r_{v_i}^{(K)} - r_{v_j}^{(K)} \|_2^2, \tag{7}$$

where $r_{v_i}^{(K)}$ is the Kth layer representation of vertex v_i, with K being the number of hidden layers.

In all, SDNE minimizes the joint objective function

$$\mathcal{L} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg}, \tag{8}$$

where $\mathcal{L}_{reg}$ is a regularization term to prevent overfitting. After solving the minimization of (8), for vertex v_i, the Kth layer representation $r_{v_i}^{(K)}$ is taken as its representation r_{v_i}.

node2vec. In contrast to the rigid strategy of defining neighborhood (context) for each vertex, node2vec [34] designs a flexible neighborhood sampling strategy, i.e., biased random walk, which smoothly interpolates between two extreme sampling strategies, i.e., Breadth-first Sampling (BFS) and

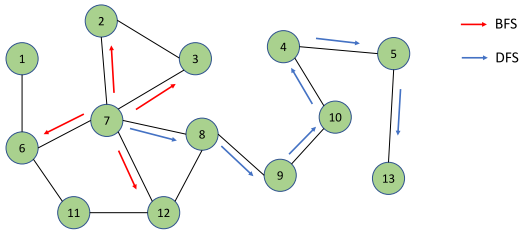


Fig. 6. Two different neighborhood sampling strategies considered by node2vec: BFS and DFS.

Depth-first Sampling (DFS), as illustrated in Fig. 6. The biased random walk exploited in node2vec can better preserve both the second-order and high-order proximity.

Following the skip-gram architecture, given the set of neighbor vertices $N(v_i)$ generated by biased random walk, node2vec learns the vertex representation $f(v_i)$ by optimizing the occurrence probability of neighbor vertices $N(v_i)$ conditioned on the representation of vertex v_i, $f(v_i)$

$$\max_f \sum_{v_i \in V} \log \Pr(N(v_i)|f(v_i)). \tag{9}$$

High-Order Proximity Preserved Embedding (HOPE). HOPE [35] learns vertex representations that capture the asymmetric high-order proximity in directed networks. In undirected networks, the transitivity is symmetric, but it is asymmetric in directed networks. For example, in an directed network, if there is a directed link from vertex v_i to vertex v_j and from vertex v_j to vertex v_k, it is more likely to have a directed link from v_i to v_k, but not from v_k to v_i.

To preserve the asymmetric transitivity, HOPE learns two vertex embedding vectors $U^s, U^t \in \mathbb{R}^{|V| \times d}$, which is called source and target embedding vectors, respectively. After constructing the high-order proximity matrix S from four proximity measures, i.e., Katz Index [61], Rooted PageRank [62], Common Neighbors and Adamic-Adar. HOPE learns vertex embeddings by solving the following matrix factorization problem:

$$\min_{U_s, U_t} \|S - U^s \cdot U^{t\mathrm{T}}\|_F^2. \tag{10}$$

Asymmetric Proximity Preserving Graph Embedding (APP). APP [36] is another NRL algorithm designed to capture asymmetric proximity, by using a Monte Carlo approach to approximate the asymmetric Rooted PageRank proximity [62]. Similar to HOPE, APP has two representations for each vertex v_i, the one as a source role $r^s_{v_i}$ and the other as a target role $r^t_{v_i}$. For each sampled path starting from v_i and ending with v_j, the representations are learned by maximizing the target vertex v_j's occurrence probability conditioned on the source vertex v_i

$$\Pr(v_j|v_i) = \frac{\exp(r^s_{v_i} \cdot r^t_{v_j})}{\sum_{v \in V} \exp(r^s_{v_i} \cdot r^t_v)}. \tag{11}$$

GraphGAN. GraphGAN [37] learns vertex representations by modeling the connectivity behavior through an adversarial learning framework. Inspired by Generative Adversarial Nets [58], GraphGAN works through two components: (i) Generator $G(v|v_c)$, which fits the distribution of the vertices connected to v_c across V and generates the likely connected vertices, and (ii) Discriminator $D(v, v_c)$, which outputs a connecting probability for the vertex pair (v, v_c), to differentiate the vertex pairs generated by $G(v|v_c)$ from the ground truth. $G(v|v_c)$ and $D(v, v_c)$ compete in a way that $G(v|v_c)$ tries to fit the true connecting distribution as much as possible and generates fake connected vertex pairs to fool $D(v, v_c)$, while $D(v, v_c)$ tries to increase its discriminative power to distinguish the vertex pairs generated by $G(v|v_c)$ from the ground truth. The competition is achieved by the following $minimax$ game:

$$\begin{aligned}\min_{\theta_G} \max_{\theta_D} \sum_{v_c \in V} (&\mathbb{E}_{v \sim \Pr_{true}(\cdot|v_c)}[\log D(v, v_c; \theta_D)] \\ &+ \mathbb{E}_{v \sim G(\cdot|v_c;\theta_G)}[\log(1 - D(v, v_c; \theta_D))]).\end{aligned} \tag{12}$$

Here, $G(v|v_c; \theta_G)$ and $D(v, v_c; \theta_D)$ are defined as following:

$$\begin{aligned} G(v|v_c; \theta_G) &= \frac{\exp(\mathbf{g}_v \cdot \mathbf{g}_{v_c})}{\sum_{v \neq v_c} \exp(\mathbf{g}_v \cdot \mathbf{g}_{v_c})}, \\ D(v, v_c; \theta_D) &= \frac{1}{1 + \exp(\mathbf{d}_v \cdot \mathbf{d}_{v_c})}, \end{aligned} \tag{13}$$

where $\mathbf{g}_v \in \mathbb{R}^k$ and $\mathbf{d}_v \in \mathbb{R}^k$ is the representation vector for generator and discriminator, respectively, and $\theta_D = \{\mathbf{d}_v\}$, $\theta_G = \{\mathbf{g}_v\}$. After the $minimax$ game in Eq. (12) is solved, $\mathbf{g}_v$ serves as the final vertex representations.

Summary. The proximity preserved by microscopic structure preserving NRL algorithms is summarized in Table 4. Most algorithms in this category preserve the second-order and high-order proximity, whereas only LINE [1], SDNE [19] and GraphGAN [37] consider the first-order proximity. From the methodology perspective, DeepWalk [6], node2vec [34] and APP [36] employ random walks to capture vertex neighborhood structure. GraRep [26] and HOPE [35] are realized by performing factorization on a $|V| \times |V|$ scale matrix, making them hard to scale up. LINE [1] and GraphGAN [37] directly model the connectivity behavior, while deep learning based methods (DNGR [9] and SDNE [19]) learn non-linear vertex representations.

TABLE 4
A Summary of Microscopic Structure Preserving NRL Algorithms

Algorithms	First-order Proximity	Second-order Proximity	High-order Proximity
DeepWalk [6]		✓	✓
LINE [1]	✓	✓	
GraRep [26]		✓	✓
DNGR [9]		✓	✓
SDNE [19]	✓	✓	
node2vec [34]		✓	✓
HOPE [35]		✓	✓
APP [36]		✓	✓
GraphGAN [37]	✓		

4.1.2 Structural Role Proximity Preserving NRL

Besides local connectivity patterns, vertices often share similar structural roles at a mesoscopic level, such as centers of stars or members of cliques. Structural role proximity

preserving NRL aims to embed vertices that are far away from each other but share similar structural roles close to each other. This not only facilitates the downstream structural role dependent tasks but also enhances microscopic structure preserving NRL.

struct2vec. struct2vec [38] first encodes the vertex structural role similarity into a multilayer graph, where the weights of edges at each layer are determined by the structural role difference at the corresponding scale. DeepWalk [6] is then performed on the multilayer graph to learn vertex representations, such that vertices close to each other in the multilayer graph (with high structural role similarity) are embedded closely in the new representation space.

For each vertex pair (v_i, v_j), considering their k-hop neighborhood formed by their neighbors within k steps, their structural distance at scale k, $D_k(v_i, v_j)$, is defined as

$$D_k(v_i, v_j) = D_{k-1}(v_i, v_j) + g(s(R_k(v_i)), s(R_k(v_j))), \tag{14}$$

where $R_k(v_i)$ is the set of vertices in v_j's k-hop neighborhood, $s(R_k(v_i))$ is the ordered degree sequence of the vertices in $R_k(v_i)$, and $g(s(R_k(v_i)), s(R_k(v_j)))$ is the distance between the ordered degree sequences $s(R_k(v_i))$ and $s(R_k(v_j))$. When $k = 0$, $D_0(v_i, v_j)$ is the degree difference between vertex v_i and v_j.

GraphWave. By making use of the spectral graph wavelet diffusion patterns, GraphWave [39] embeds vertex neighborhood structure into a low-dimensional space and preserves the structural role proximity. The assumption is that, if two vertices residing distantly in the network share similar structural roles, the graph wavelets starting at them will diffuse similarly across their neighbors.

For vertex v_k, its spectral graph wavelet coefficients Ψ_k is defined as

$$\Psi_k = U\mathrm{Diag}(g_s(\lambda_1), \ldots, g_s(\lambda_{|V|}))U^{\mathrm{T}}\delta_k, \tag{15}$$

where U is the eigenvector matrix of the graph Laplacian L and $\lambda_1, \ldots, \lambda_{|V|}$ are the eigenvalues, $g_s(\lambda) = \exp(-\lambda s)$ is the heat kernel, and δ_k is the one-hot vector for k. By taking Ψ_k as a probability distribution, the spectral wavelet distribution pattern in Ψ_k is then encoded into its empirical characteristic function

$$\phi_k(t) = \frac{1}{|V|}\sum_{m=1}^{|V|} e^{it\Psi_{km}}. \tag{16}$$

Then v_k's low-dimensional representation is then obtained by sampling the 2-dimensional parametric function of $\phi_k(t)$ at d evenly separated points $t_1, t_2, \ldots, t_d$ as

$$\begin{aligned} f(v_k) = [&\mathrm{Re}(\phi_k(t_1)), \ldots, \mathrm{Re}(\phi_k(t_d)), \\ &\mathrm{Im}(\phi_k(t_1)), \ldots, \mathrm{Im}(\phi_k(t_d))]. \end{aligned} \tag{17}$$

Structural and Neighborhood Similarity Preserving Network Embedding (SNS). SNS [40] enhances a random walk based method with structural role proximity. To preserve vertex structural roles, SNS represents each vertex as a *Graphlet Degree Vector* with each element being the number of times the given vertex is touched by the corresponding orbit of graphlets. The *Graphlet Degree Vector* is used to measure the vertex structural role similarity.

Given a vertex v_i, SNS uses its context vertices $\mathcal{C}(v_i)$ and structurally similar vertices $\mathcal{S}(v_i)$ to predict its existence, which is achieved by maximizing the following probability:

$$\Pr(v_i|\mathcal{C}(v_i), \mathcal{S}(v_i)) = \frac{\exp(r'_{v_i} \cdot h_{v_i})}{\sum_{u\in V}\exp(r'_u \cdot h_{v_i})}, \tag{18}$$

where r'_{v_i} is the output representation of v_i and h_{v_i} is the hidden layer representation for predicting v_i, which is aggregated from the input representations r_u, for each u in $\mathcal{C}(v_i)$ and $\mathcal{S}(v_i)$.

Summary. struct2vec [38] and GraphWave [39] take advantage of structural role proximity to learn vertex representations that facilitate specific structural role dependent tasks, e.g., vertex classification in traffic networks, while SNS [40] enhances a random walk based microscopic structure preserving NRL algorithm with structural role proximity. Technically, random walk is employed by struct2vec and SNS, while matrix factorization is adopted by GraphWave.

4.1.3 Intra-Community Proximity Preserving NRL

Another interesting feature that real-world networks exhibit is the community structure, where vertices are densely connected to each other within the same community, but sparsely connected to vertices from other communities. For example, in social networks, people from the same interest group or affiliation often form a community. In citation networks, papers on similar research topics tend to frequently cite each other. Intra-community preserving NRL aims to leverage the community structure that characterizes key vertex properties to learn informative vertex representations.

Learning Latent Social Dimensions. The social dimension based NRL algorithms try to construct social actors' embeddings through their membership or affiliation to a number of social dimensions. To infer these latent social dimensions, the phenomenon of "community" in social networks is considered, stating that social actors sharing similar properties often form groups with denser within-group connections. Thus, the problem boils down to one classical network analytic task—community detection—that aims to discover a set of communities with denser within-group connections than between-group connections. Three clustering techniques, including modularity maximization [31], spectral clustering [32] and edge clustering [33] are employed to discover latent social dimensions. Each social dimension describes the likelihood of a vertex belonging to a plausible affiliation. These methods preserve the global community structure, but neglect local structure properties, e.g., the first-order and second-order proximity.

Modularized Nonnegative Matrix Factorization (M-NMF). M-NMF [28] augments the second-order and high-order proximity with broader community structure to learn more informative vertex embeddings $U \in \mathbb{R}^{|V|\times d}$ using the following objective:

$$\begin{aligned} &\min_{M,U,H,C} \|S - MU^{\mathrm{T}}\|_F^2 + \alpha\|H - UC^{\mathrm{T}}\|_F^2 - \beta tr(H^{\mathrm{T}}BH) \\ &s.t., \quad M \geq 0, U \geq 0, H \geq 0, C \geq 0, tr(H^{\mathrm{T}}H) = |V|, \end{aligned} \tag{19}$$

where vertex embedding U is learned by minimizing $\|S - MU^{\mathrm{T}}\|_F^2$, with $S \in \mathbb{R}^{|V|\times|V|}$ being the vertex pairwise proximity matrix, which captures the second-order and the high-order proximity when taken as representations. The community indicative vertex embedding H is learned by maximizing $tr(H^{\mathrm{T}}BH)$, which is essentially the objective of modularity maximization with B being the modularity matrix. The minimization on $\|H - UC^{\mathrm{T}}\|_F^2$ makes these two embeddings consistent with each other by importing a community representation matrix C.

Summary. The algorithms of learning latent social dimensions [31], [32], [33] only consider the community structure to learn vertex representation, while M-NMF [28] integrates microscopic structure (the second-order and high-order proximity) with the intra-community proximity. These methods primarily rely on matrix factorization to detect community structure, making them hard to scale up.

4.1.4 Macroscopic Structure Preserving NRL

Macroscopic structure preserving methods aim to preserve certain global network properties in a macroscopic view. Only very few recent studies are developed for this purpose.

Degree Penalty Principle (DP). Many real-world networks present the macroscopic scale-free property, which depicts the phenomenon that vertex degree follows a long-tailed distribution, i.e., most vertices are sparsely connected and only few vertices have dense edges. To capture the scale-free property, [41] proposes the degree penalty principle: penalizing the proximity between high-degree vertices. This principle is then coupled with two NRL algorithms (i.e., spectral embedding [5] and DeepWalk [6]) to learn scale-free property preserving vertex representations.

Hierarchical Representation Learning for Networks (HARP). To capture the global patterns in networks, HARP [42] samples small networks to approximate the global structure. The vertex representations learned from sampled networks are taken as the initialization for inferring the vertex representations of the original network. In this way, global structure is preserved in the final representations. To obtain smooth solutions, a series of smaller networks are successively sampled from the original network by coalescing edges and vertices, and the vertex representations are hierarchically inferred back from the smallest network to the original network. In HARP, DeepWalk [6] and LINE [1] are used to learn vertex representations.

Summary. DP [41] and HARP [42] are both realized by adapting the existing NRL algorithms to capture the macroscopic structure. The former tries to preserve the scale-free property, while the latter makes the learned vertex representations respect the global network structure.

4.2 Unsupervised Content Augmented Network Representation Learning

Besides network structure, real-world networks are often attached with rich content as vertex attributes, such as webpages in webpages networks, papers in citation networks, and user metadata in social networks. Vertex attributes provide direct evidence to measure content-level similarity between vertices. Therefore, network representation learning can be significantly improved if vertex attribute information is properly incorporated into the learning process. Recently, several content augmented NRL algorithms have been proposed to incorporate network structure and vertex attributes to reinforce the network representation learning.

4.2.1 Text-Associated DeepWalk (TADW)

TADW [7] first proves the equivalence between DeepWalk [6] and the following matrix factorization:

$$\min_{W,H} \|M - W^{\mathrm{T}}H\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{20}$$

where W and H are learned latent embeddings and M is the vertex-context matrix carrying transition probability between each vertex pair within k steps. Then, textual features are imported through inductive matrix factorization [54]

$$\min_{W,H} \|M - W^{\mathrm{T}}HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{21}$$

where T is vertex textual feature matrix. After (21) is solved, the final vertex representations are formed by taking the concatenation of W and HT.

4.2.2 Homophily, Structure, and Content Augmented Network Representation Learning (HSCA)

Despite its ability to incorporate textural features, TADW [7] only considers structural context of network vertices, i.e., the second-order and high-order proximity, but ignores the important homophily property (the first-order proximity) in its learning framework. HSCA [20] is proposed to simultaneously integrates homophily, structural context, and vertex content to learn effective network representations.

For TADW, the learned representation for the ith vertex v_i is $\left[W_{:i}^{\mathrm{T}}, (HT_{:i})^{\mathrm{T}}\right]^{\mathrm{T}}$, where $W_{:i}$ and $T_{:i}$ is the ith column of W and T, respectively. To enforce the first-order proximity, HSCA introduces a regularization term to enforce homophily between directly connected nodes in the embedding space, which is formulated as

$$\mathcal{R}(W,H) = \frac{1}{4}\sum_{i,j=1}^{|V|} S_{ij}\left\|\begin{bmatrix} W_{:i} \\ HT_{:i}\end{bmatrix} - \begin{bmatrix} W_{:j} \\ HT_{:j}\end{bmatrix}\right\|_2^2, \tag{22}$$

where S is the adjacent matrix. The objective of HSCA is

$$\min_{W,H} \|M - W^{\mathrm{T}}HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2) + \mu\mathcal{R}(W,H), \tag{23}$$

where λ and μ are the trade-off parameters. After solving the above optimization problem, the concatenation of W and HT is taken as the final vertex representations.

4.2.3 Paired Restricted Boltzmann Machine (pRBM)

By leveraging the strength of Restricted Boltzmann Machine (RBM) [57], [29] designs a novel model called Paired RBM (pRBM) to learn vertex representations by combining vertex attributes and link information. The pRBM considers the networks with vertices associated with binary attributes. For each edge $(v_i, v_j) \in E$, the attributes for v_i and v_j are $\mathbf{v}^{(i)}$

and $\mathbf{v}^{(j)} \in \{0,1\}^m$, and their hidden representations are $\mathbf{h}^{(i)}$ and $\mathbf{h}^{(j)} \in \{0,1\}^d$. Vertex hidden representations are learned by maximizing the joint probability of pRBM defined over $\mathbf{v}^{(i)}$, $\mathbf{v}^{(j)}$, $\mathbf{h}^{(i)}$ and $\mathbf{h}^{(j)}$

$$\begin{aligned}&\Pr(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}; \boldsymbol{\theta}) \\ &= \exp(-E(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}))/Z,\end{aligned} \tag{24}$$

where $\boldsymbol{\theta} = \{\mathbf{W} \in \mathbb{R}^{d\times m}, \mathbf{b} \in \mathbb{R}^{d\times 1}, \mathbf{c} \in \mathbb{R}^{m\times 1}, \mathbf{M} \in \mathbb{R}^{d\times d}\}$ is the parameter set and Z is the normalization term. To model the joint probability, the energy function is defined as

$$\begin{aligned}&E(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}) \\ &= -w_{ij}(\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{M}\mathbf{h}^{(j)} - (\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{W}\mathbf{v}^{(i)} - \mathbf{c}^{\mathrm{T}}\mathbf{v}^{(i)} - \mathbf{b}^{\mathrm{T}}\mathbf{h}^{(i)} \\ &\quad - (\mathbf{h}^{(j)})^{\mathrm{T}}\mathbf{W}\mathbf{v}^{(j)} - \mathbf{c}^{\mathrm{T}}\mathbf{v}^{(i)} - \mathbf{b}^{\mathrm{T}}\mathbf{h}^{(j)},\end{aligned} \tag{25}$$

where $w_{ij}(\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{M}\mathbf{h}^{(j)}$ forces the latent representations of v_i and v_j to be close and w_{ij} is the weight of edge (v_i, v_j).

4.2.4 User Profile Preserving Social Network Embedding (UPP-SNE)

UPP-SNE [43] leverages user profile features to enhance the embedding learning of users in social networks. Compared with textural content features, user profiles have two unique properties: (1) user profiles are noisy, sparse and incomplete and (2) different dimensions of user profile features are topic-inconsistent. To filter out noise and extract useful information from user profiles, UPP-SNE constructs user representations by performing a non-linear mapping on user profile features, which is guided by network structure.

The approximated kernel mapping [63] is used in UPP-SNE to construct user embedding from user profile features

$$\begin{aligned}f(v_i) = \varphi(\boldsymbol{x}_i) = \frac{1}{\sqrt{d}}\big[&\cos(\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{x}_i), \ldots, \cos(\boldsymbol{\mu}_d^{\mathrm{T}}\boldsymbol{x}_i), \\ &\sin(\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{x}_i), \ldots, \sin(\boldsymbol{\mu}_d^{\mathrm{T}}\boldsymbol{x}_i)\big]^{\mathrm{T}},\end{aligned} \tag{26}$$

where $\boldsymbol{x}_i$ is the user profile feature vector of vertex v_i and $\boldsymbol{\mu}_i$ is the corresponding coefficient vector.

To supervise the learning of the non-linear mapping and make user profiles and network structure complement each other, the objective of DeepWalk [6] is used

$$\min_f \ -\log \Pr(\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i | f(v_i)), \tag{27}$$

where $\{v_{i-t}, \ldots, v_{i+t}\} \backslash v_i$ is the context vertices of vertex v_i within t window size in the given random walk sequence.

4.2.5 Property Preserving Network Embedding (PPNE)

To learn content augmented vertex representations, PPNE [44] jointly optimizes two objectives: (i) the structure-driven objective and (ii) the attribute-driven objective.

Following DeepWalk, the structure-driven objective aims to make vertices sharing similar context vertices represented closely. For a given random walk sequence $\mathcal{S}$, the structure-driven objective is formulated as

$$\min D_T = \prod_{v\in\mathcal{S}} \prod_{u\in context(v)} \Pr(u|v). \tag{28}$$

The attribute-driven objective aims to make the vertex representations learned by Eq. (28) respect the vertex attribute similarity. A realization of the attribute-driven objective is

$$\min D_N = \sum_{v\in\mathcal{S}} \sum_{u\in pos(v)\cup neg(v)} P(v,u)d(v,u), \tag{29}$$

where $P(u,v)$ is the attribute similarity between u and v, $d(u,v)$ is the distance between u and v in the embedding space, and $pos(v)$ and $neg(v)$ is the set of top-k similar and dissimilar vertices according to $P(u,v)$, respectively.

Summary. The above unsupervised content augmented NRL algorithms incorporate vertex content features in three ways. The first, used by TADW [7] and HSCA [20], is to couple the network structure with vertex content features via inductive matrix factorization [54]. This process can be considered as a linear transformation on vertex attributes constrained by network structure. The second is to perform a non-linear mapping to construct new vertex embeddings that respect network structure. For example, RBM [57] and the approximated kernel mapping [63] is used by pRBM [29] and UPP-SNE [43], respectively, to achieve this goal. The third used by PPNE [44] is to add an attribute preserving constraint to the structure preserving optimization objective.

5 Semi-Supervised Network Representation Learning

Label information attached with vertices directly indicates vertices' group or class affiliation. Such labels have strong correlations, although not always consistent, to network structure and vertex attributes, and are always helpful in learning informative and discriminative network representations. Semi-supervised NRL algorithms are developed along this line to make use of vertex labels available in the network for seeking more effective vertex representations.

5.1 Semi-Supervised Structure Preserving NRL

The first group of semi-supervised NRL algorithms aim to simultaneously optimize the representation learning that preserves network structure and discriminative learning. As a result, the information derived from vertex labels can help improve the representative and discriminative power of the learned vertex representations.

5.1.1 Discriminative Deep Random Walk (DDRW)

Inspired by the discriminative representation learning [64], [65], DDRW [45] proposes to learn discriminative network representations through jointly optimizing the objective of DeepWalk [6] together with the following L2-loss Support Vector Classification objective

$$\mathcal{L}_c = C\sum_{i=1}^{|V|}(\sigma(1 - Y_{ik}\beta^{\mathrm{T}}r_{v_i}))^2 + \frac{1}{2}\beta^{\mathrm{T}}\beta, \tag{30}$$

where $\sigma(x) = x$, if $x > 0$ and otherwise $\sigma(x) = 0$.

The joint objective of DDRW is thus defined as

$$\mathcal{L} = \eta\mathcal{L}_{DW} + \mathcal{L}_c. \tag{31}$$

where $\mathcal{L}_{DW}$ is the objective function of Deekwalk. The objective (31) aims to learn discriminative vertex representations for binary classification for the kth class. DDRW is

generalized to handle multi-class classification by using the *one-against-rest* strategy [66].

5.1.2 Max-Margin DeepWalk (MMDW)

Similarly, MMDW [46] couples the objective of the matrix factorization version DeepWalk [7] with the following multi-class Support Vector Machine objective with $\{(r_{v_1}, Y_{1:}), \ldots, (r_{v_T}, Y_{T:})\}$ training set

$$\begin{aligned} &\min_{W,\xi} \mathcal{L}_{SVM} = \min_{W,\xi} \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{T}\xi_i, \\ &s.t.\ \mathrm{w}_{l_i}^{\mathrm{T}} r_{v_i} - \mathrm{w}_j^{\mathrm{T}} r_{v_i} \geq e_i^j - \xi_i,\ \forall i, j, \end{aligned} \tag{32}$$

where $l_i = k$ with $Y_{ik} = 1$, $e_i^j = 1$ for $Y_{ij} = -1$, and $e_i^j = 0$ for $Y_{ij} = 1$.

The joint objective of MMDW is

$$\begin{aligned} &\min_{U,H,W,\xi} \mathcal{L} = \min_{U,H,W,\xi} \mathcal{L}_{DW} + \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{T}\xi_i, \\ &s.t.\ \mathrm{w}_{l_i}^{\mathrm{T}} r_{v_i} - \mathrm{w}_j^{\mathrm{T}} r_{v_i} \geq e_i^j - \xi_i,\ \forall i, j. \end{aligned} \tag{33}$$

where $\mathcal{L}_{DW}$ is the objective of the matrix factorization version of DeepWalk.

5.1.3 Transductive LINE (TLINE)

Along similar lines, TLINE [47] is proposed as a semi-supervised extension of LINE [1] that simultaneously learns LINE's vertex representations and an SVM classifier. Given a set of labeled and unlabeled vertices $\{v_1, v_2, \ldots, v_L\}$ and $\{v_{L+1}, \ldots, v_{|V|}\}$, TLINE trains a multi-class SVM classifier on $\{v_1, v_2, \ldots, v_L\}$ by optimizing the objective

$$\mathcal{O}_{svm} = \sum_{i=1}^{L}\sum_{k=1}^{K} \max(0, 1 - Y_{ik}\mathrm{w}_k{}^{\mathrm{T}} r_{v_i}) + \lambda\|\mathrm{w}_k\|_2^2. \tag{34}$$

Based on LINE's formulations that preserve the first-order and second-order proximity, TLINE optimizes two objective functions

$$\mathcal{O}_{TLINE(1st)} = \mathcal{O}_{line1} + \beta\mathcal{O}_{svm}, \tag{35}$$

$$\mathcal{O}_{TLINE(2nd)} = \mathcal{O}_{line2} + \beta\mathcal{O}_{svm}. \tag{36}$$

Inheriting LINE's ability to deal with large-scale networks, TLINE is claimed to be able to learn discriminative vertex representations for large-scale networks with low time and memory cost.

5.1.4 Group Enhanced Network Embedding (GENE)

GENE [48] integrates group (label) information with network structure in a probabilistic manner. GENE assumes that vertices should be embedded closely in low-dimensional space, if they share similar neighbors or join similar groups. Inspired by DeepWalk [6] and document modeling [67], [68], the mechanism of GENE for learning group label informed vertex representations is achieved by maximizing the following log probability:

$$\begin{aligned} \mathscr{S} = \sum_{g_i \in \mathcal{Y}} \Bigg[&\alpha \sum_{W \in W_{g_i}} \sum_{v_j \in W} \log \Pr(v_j | v_{j-t}, \ldots, v_{j+t}, g_i) \\ &+ \beta \sum_{\hat{v}_j \in \hat{W}_{g_j}} \log \Pr(\hat{v}_j | g_i) \Bigg], \end{aligned} \tag{37}$$

where $\mathcal{Y}$ is the set of different groups, W_{g_i} is the set of random walk sequences labeled with g_i, $\hat{W}_{g_i}$ is the set of vertices randomly sampled from group g_i.

5.1.5 Semi-Supervised Network Embedding (SemiNE)

SemiNE [49] learns semi-supervised vertex representations in two stages. In the first stage, SemiNE exploits the DeepWalk [6] framework to learn vertex representations in an unsupervised manner. It points out that DeepWalk does not consider the order information of context vertex, i.e., the distance between the context vertex and the central vertex, when using the context vertex v_{i+j} to predict the central vertex v_i. Thus, SemiNE encodes the order information into DeepWalk by modeling the probability $\Pr(v_{i+j}|v_i)$ with j-dependent parameters

$$\Pr(v_{i+j}|v_i) = \frac{\exp(\Phi(v_i) \cdot \Psi_j(v_{i+j}))}{\sum_{u \in V} \exp(\Phi(v_i) \cdot \Psi_j(u))}, \tag{38}$$

where $\Phi(\cdot)$ is the vertex representation and $\Psi_j(\cdot)$ is the parameter for calculating $\Pr(v_{i+j}|v_i)$.

In the second stage, SemiNE learns a neural network that tunes the learned unsupervised vertex representations to fit vertex labels.

5.2 Semi-Supervised Content Augmented NRL

Recently, more research efforts have shifted to the development of label and content augmented NRL algorithms that investigate the use of vertex content and labels to assist with network representation learning. With content information incorporated, the learned vertex representations are expected to be more informative, and with label information considered, the learned vertex representations can be highly customized for the underlying classification task.

5.2.1 Tri-Party Deep Network Representation (TriDNR)

Using a coupled neural network framework, TriDNR [50] learns vertex representations from three information sources: network structure, vertex content and vertex labels. To capture the vertex content and label information, TriDNR adapts the Paragraph Vector model [67] to describe the vertex-word correlation and the label-word correspondence by maximizing the following objective:

$$\mathcal{L}_{PV} = \sum_{i \in L} \log \Pr(w_{-b} : w_b | c_i) + \sum_{i=1}^{|V|} \log \Pr(w_{-b} : w_b | v_i), \tag{39}$$

where $\{w_{-b} : w_b\}$ is a sequence of words inside a contextual window of length $2b$, c_i is the class label of vertex v_i, and L is the set of indices of labeled vertices.

TriDNR is then realized by coupling the Paragraph Vector objective with DeepWalk objective

$$\max\ (1 - \alpha)\mathcal{L}_{DW} + \alpha\mathcal{L}_{PV}, \tag{40}$$

where $\mathcal{L}_{DW}$ is the DeepWalk maximization objective function and α is the trade-off parameter.

5.2.2 Linked Document Embedding (LDE)

LDE [51] is proposed to learn representations for linked documents, which are actually the vertices of citation or

webpage networks. Similar to TriDNR [50], LDE learns vertex representations by modeling three kinds of relations, i.e., word-word-document relations, document-document relations, and document-label relations. LDE is realized by solving the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{D},\boldsymbol{Y}} & -\frac{1}{|\mathcal{P}|}\sum_{(w_i,w_j,d_k)\in\mathcal{P}}\log \Pr(w_j|w_i,d_k)\\ & -\frac{1}{|E|}\sum_{i}\sum_{j:(v_i,v_j)\in E}\log \Pr(d_j|d_i)\\ & -\frac{1}{|\mathcal{Y}|}\sum_{i:y_i\in\mathcal{Y}}\log \Pr(y_i|d_i)\\ & +\gamma(\|\boldsymbol{W}\|_F^2+\|\boldsymbol{D}\|_F^2+\|\boldsymbol{Y}\|_F^2).\end{aligned} \tag{41}$$

Here, the probability $\Pr(w_j|w_i,d_k)$ is used to model word-word-document relations, which means the probability that in document d_k, word w_j is a neighboring word of w_i. To capture word-word-document relations, triplets (w_i,w_j,d_k) are extracted, with the word-neighbor pair (w_i,w_j) occurring in document d_k. The set of triplets (w_i,w_j,d_k) is denoted by $\mathcal{P}$. The document-document relations are captured by the conditional probability between linked document pairs (d_i,d_j), $\Pr(d_j|d_i)$. The document-label relations are also considered by modeling $\Pr(y_i|d_i)$, the probability for the occurrence of class label y_i conditioned on document d_i. In (41), $\boldsymbol{W}$, $\boldsymbol{D}$ and $\boldsymbol{Y}$ is the embedding matrix for words, documents and labels, respectively.

5.2.3 Discriminative Matrix Factorization (DMF)

To empower vertex representations with discriminative ability, DMF [8] enforces the objective of TADW (21) with an empirical loss minimization for a linear classifier trained on labeled vertices

$$\begin{aligned}\min_{W,H,\boldsymbol{\eta}} & \frac{1}{2}\sum_{i,j=1}^{|V|}(M_{ij}-\boldsymbol{w}_i^{\mathrm{T}}H\boldsymbol{t}_j)^2+\frac{\mu}{2}\sum_{n\in\mathcal{L}}(Y_{n1}-\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{x}_n)^2\\ & +\frac{\lambda_1}{2}(\|H\|_F^2+\|\boldsymbol{\eta}\|_2^2)+\frac{\lambda_2}{2}\|W\|_F^2,\end{aligned} \tag{42}$$

where $\boldsymbol{w}_i$ is the ith column of vertex representation matrix W and $\boldsymbol{t}_j$ is jth column of vertex textual feature matrix T, and $\mathcal{L}$ is the set of indices of labeled vertices. DMF considers binary-class classification, i.e., $\mathcal{Y}=\{+1,-1\}$. Hence, Y_{n1} is used to denote the class label of vertex v_n.

DMF constructs vertex representations from W rather that W and HT. This is based on empirical findings that W contains sufficient information for vertex representations. In the objective of (42), $\boldsymbol{x}_n$ is set to $[\boldsymbol{w}_n^{\mathrm{T}},1]^{\mathrm{T}}$, which incorporates the intercept term b of the linear classifier into $\boldsymbol{\eta}$. The optimization problem (42) is solved by optimizing W, H and $\boldsymbol{\eta}$ alternately. Once the optimization problem is solved, the discriminative and informative vertex representations together with the linear classifier are learned, and work together to classify unlabeled vertices in networks.

5.2.4 Predictive Labels and Neighbors with Embeddings Transductively or Inductively from Data (Planetoid)

Planetoid [52] leverages network embedding together with vertex attributes to carry out semi-supervised learning. Planetoid learns vertex embeddings by minimizing the loss for predicting structural context, which is formulated as

$$\mathcal{L}_u=-\mathbb{E}_{(i,c,\gamma)}\log\sigma(\gamma \mathrm{w}_c^{\mathrm{T}}\mathbf{e}_i), \tag{43}$$

where (i,c) is the index for vertex context pair (v_i,v_c), $\mathbf{e}_i$ is the embedding of vertex v_i, w_c is the parameter vector for context vertex v_c, and $\gamma\in\{+1,-1\}$ indicates whether the sampled vertex context pair (i,c) is positive or negative. The triple (i,c,γ) is sampled according to both the network structure and vertex labels.

Planetoid then maps the learned vertex representations $\mathbf{e}$ and vertex attributes $\mathbf{x}$ to hidden layer space via deep neural network, and concatenates these two hidden layer representations together to predict vertex labels, by minimizing the following classification loss:

$$\mathcal{L}_s=-\frac{1}{L}\sum_{i=1}^{L}\log p(y_i|\mathbf{x}_i,\mathbf{e}_i), \tag{44}$$

To integrate network structure, vertex attributes and vertex labels together, Planetoid jointly minimizes the two objectives (43) and (44) to learn vertex embedding $\mathbf{e}$ with deep neural networks.

5.2.5 Label Informed Attribute Network Embedding (LANE)

LANE [30] learns vertex representations by embedding the network structure proximity, attribute affinity, and label proximity into a unified latent representation. The learned representations are expected to capture both network structure and vertex attribute information, and label information if provided. The embedding learning in LANE is carried out in two stages. During the first stage, vertex proximity in network structure and attribute information are mapped into latent representations $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(A)}$, then $\mathbf{U}^{(A)}$ is incorporated into $\mathbf{U}^{(G)}$ by maximizing their correlations. In the second stage, LANE employs the joint proximity (determined by $\mathbf{U}^{(G)}$) to smooth label information and uniformly embeds them into another latent representation $\mathbf{U}^{(Y)}$, and then embeds $\mathbf{U}^{(A)}$, $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(Y)}$ into a unified embedding representation $\mathbf{H}$.

5.3 Summary

We now summarize and compare the discriminative learning strategies used by semi-supervised NRL algorithms in Table 5 in terms of their advantages and disadvantages.

Three strategies are used to achieve discriminative learning. The first strategy (i.e., DDRW [45], MMDW [46], TLINE [47], DMF [8], SemiNE [49]) is to enforce classification loss minimization on vertex representations, i.e., fitting the vertex representations to a classifier. This provides a direct way to separate vertices of different categories from each other in the new embedding space. The second strategy (used by GENE [48], TriDNR [50], LDE [51] and Planetoid [52]) is achieved by modeling vertex label relation, such that vertices with same labels have similar vector representations. The third strategy used by LANE [30] is to jointly embed vertices and labels into a common space.

Fitting vertex representations to a classifier can take advantage of the discriminative power in vertex labels. Algorithms using this strategy only require a small number of

TABLE 5
A Summary of Semi-Supervised NRL Algorithms

Discriminative Learning Strategy	Algorithm	Loss function	Advantage	Disadvantage
fitting a classifier	DDRW [45]	hinge loss	a) directly optimize classification loss; b) perform better in sparsely labeled scenarios	prone to overfitting
	MMDW [46]	hinge loss		
	TLINE [47]	hinge loss		
	DMF [8]	square loss		
	SemiNE [49]	logistic loss		
modeling vertex label relation	GENE [48]	likelihood loss	a) better capture intra-class proximity; b) generalization to other tasks	require more labeled data
	TriDNR [50]	likelihood loss		
	LDE [51]	likelihood loss		
	Planetoid [52]	likelihood loss		
joint vertex label embedding	LANE [30]	correlation loss		

labeled vertices (e.g., 10 percent) to achieve significant performance gain over their unsupervised counterparts. They are thus more effective for discriminative learning in sparsely labeled scenarios. However, fitting vertex representations to a classifier is more prone to overfitting. Regularization and DropOut [69] are often introduced to overcome this problem. By contrast, modeling vertex label relation and joint vertex embedding requires more vertex labels to make vertex representations more discriminative, but they can better capture intra-class proximity, i.e., vertices belonging to the same class are kept closer to each other in the new embedding space. This allows them to have generalized benefits on tasks like vertex clustering or visualization.

6 Applications

Once new vertex representations are learned via network representation learning techniques, traditional vector-based algorithms can be used to solve important analytic tasks, such as vertex classification, link prediction, clustering, visualization, and recommendation. The effectiveness of the learned representations can also be validated through assessing their performance on these tasks.

6.1 Vertex Classification

Vertex classification is one of the most important tasks in network analytic research. Often in networks, vertices are associated with semantic labels characterizing certain aspects of entities, such as beliefs, interests, or affiliations. In citation networks, a publication may be labeled with topics or research areas, while the labels of entities in social network may indicate individuals' interests or political beliefs. Often, because network vertices are partially or sparsely labeled due to high labeling costs, a large portion of vertices in networks have unknown labels. The problem of vertex classification aims to predict the labels of unlabeled vertices given a partially labeled network [10], [11]. Since vertices are not independent but connected to each other in the form of a network via links, vertex classification should exploit these dependencies for jointly classifying the labels of vertices. Among others, collective classification proposes to construct a new set of vertex features that summarize label dependencies in the neighborhood, which has been shown to be most effective in classifying many real-world networks [70], [71].

Network representation learning follows the same principle that automatically learns vertex features based on network structure. Existing studies have evaluated the discriminative power of the learned vertex representations under two settings: unsupervised settings (e.g., [1], [6], [7], [20], [34]), where vertex representations are learned separately, followed by applying discriminative classifiers like SVM or logistic regression on the new embeddings, and semi-supervised settings (e.g., [8], [30], [45], [46], [47]), where representation learning and discriminative learning are simultaneously tackled, so that discriminative power inferred from labeled vertices can directly benefit the learning of informative vertex representations. These studies have proved that better vertex representations can contribute to high classification accuracy.

6.2 Link Prediction

Another important application of network representation learning is link prediction [13], [72], which aims to infer the existence of new relationships or emerging interactions between pairs of entities based on the currently observed links and their properties. The approaches developed to solve this problem can enable the discovery of implicit or missing interactions in the network, the identification of spurious links, as well as understanding the network evolution mechanism. Link prediction techniques are widely applied in social networks to predict unknown connections among people, which can be used to recommend friendship or identify suspicious relationships. Most of the current social networking systems are using link prediction to automatically suggest friends with a high degree of accuracy. In biological networks, link prediction methods have been developed to predict previously unknown interactions between proteins, thus significantly reducing the costs of empirical approaches. Readers can refer to the survey papers [12], [73] for the recent progress in this field.

Good network representations should be able to capture explicit and implicit connections between network vertices thus enabling application to link prediction. [19] and [35] predict missing links based on the learned vertex representations on social networks. [34] also applies network representation learning to collaboration networks and protein-protein interaction networks. They demonstrate that on these networks links predicted using the learned representations achieve better performance than traditional similarity-based link prediction approaches.

6.3 Clustering

Network clustering refers to the task of partitioning network vertices into a set of clusters, such that vertices are densely connected to each other within the same cluster, but connected to few vertices from other clusters [74]. Such cluster

structures, or communities widely occur in a wide spectrum of networked systems from bioinformatics, computer science, physics, sociology, *etc.*, and have strong implications. For example, in biology networks, clusters may correspond to a group of proteins having the same function; in the network of webpages, clusters are likely pages having similar topics or related content; in social networks, clusters may indicate groups of people having similar interests or affiliations.

Researchers have proposed a large body of network clustering algorithms based on various metrics of similarity or strength of connection between vertices. Min-max cut and normalized cut methods [75], [76] seek to recursively partition a graph into two clusters that maximize the number of intra-cluster connections and minimize the number of inter-cluster connections. Modularity-based methods (e.g., [77], [78]) aim to maximize the modularity of a clustering, which is the fraction of intra-cluster edges minus the expected fraction assuming the edges were randomly distributed. A network partitioning with high modularity would have dense intra-cluster connections but sparse inter-cluster connections. Some other methods (e.g., [79]) try to identify nodes with similar structural roles like bridges and outliers.

Recent NRL methods (e.g., GraRep [26], DNGR [9], M-NMF [28], and pRBM [29]) used the clustering performance to evaluate the quality of the learned network representations on different networks. Intuitively, better representations would lead to better clustering performance. These works followed the common approach that first applies an unsupervised NRL algorithm to learn vertex representations, and then performs k-means clustering on the learned representations to cluster the vertices. In particular, pRBM [29] showed that NRL methods outperform the baseline that uses original features for clustering without learning representations. This suggests that effective representation learning can improve the clustering performance.

6.4 Visualization

Visualization techniques play critical roles in managing, exploring, and analyzing complex networked data. [80] surveys a range of methods used to visualize graphs from an information visualization perspective. This work compares various traditional layouts used to visualize graphs, such as tree-, 3D-, and hyperbolic-based methods, and shows that classical visualization techniques are proved effective for small or intermediate sized networks; they however confront a big challenge when applied to large-scale networks. Few systems can claim to deal effectively with thousands of vertices, although networks with this order of magnitude often occur in a wide variety of applications. Consequently, a first step in the visualization process is often to reduce the size of the network to display. One common approach is essentially to find an extremely low-dimensional representation of a network that preserves the intrinsic structure, i.e., keeping similar vertices close and dissimilar vertices far apart, in the low-dimensional space [17].

Network representation learning has the same objective that embeds a large network into a new latent space of low dimensionality. After new embeddings are obtained in the vector space, popular methods such as t-distributed stochastic neighbor embedding (t-SNE) [81] can be applied to visualize the network in a 2-D or 3-D space. By taking the learned vertex representations as input, LINE [1] used the t-SNE package to visualize the DBLP co-author network after the authors are mapped into a 2-D space, and showed that LINE is able to cluster authors in the same field to the same community. HSCA [20] illustrated the advantages of the content-augmented NRL algorithm by visualizing citation networks. Semi-supervised algorithms (e.g., TLINE [47], TriDNR [50], and DMF [8]) demonstrated that the visualization results have better clustering structures with vertex labels properly imported.

6.5 Recommendation

In addition to structure, content, and vertex label information, many social networks also include geographical and spatial-temporal information, and users can share their experiences online with their friends for point of interest (POI) recommendation, e.g., transportation, restaurant, and sightseeing landmark, *etc.* Examples of such location-based social networks (LBSN) include Foursquare, Yelp, Facebook Places, and many others. For these types of social networks, POI recommendation intends to recommend user interested objects, depending on their own context, such as the geographic location of the users and their interests. Traditionally, this is solved by using approaches, such as collaborative filtering, to leverage spatial and temporal correlation between user activities and geographical distance [82]. However, because each user's check-in records are very sparse, finding similar users or calculating transition probability between users and locations is a significant challenge.

Recently, spatial-temporal embedding [14], [15], [83] has emerged to learn low-dimensional dense vectors to represent users, locations, and point-of-interests *etc.* As a result, each user, location, and POI can be represented as a low-dimensional vector, respectively, for similarity search and many other analysis. An inherent advantage of such spatial-temporal aware embedding is that it alleviates the data sparsity problem, because the learned low dimensional vector is typically much more dense than the original representation. As a result, it makes query tasks, such as top-k POI search, much more accurate than traditional approaches.

6.6 Knowledge Graph

Knowledge graphs represent a new type of data structure in database systems which encode structured information of billions of entities and their rich relations. A knowledge graph typically contains a rich set of heterogeneous objects and different types of entity relationships. Such networked entities form a gigantic graph and is now powering many commercial search engines to find similar objects online. Traditionally, knowledge graph search is carried out through database driven approaches to explore schema mapping between entities, including entity relationships. Recent advancement in network representation learning has inspired structured embeddings of knowledge bases [84]. Such embedding methods learn a low-dimensional vector representation for knowledge graph entities, such that generic database queries, such as top-k search, can be carried out by comparing vector representation of the query object and objects in the database.

In addition to using vector representation to represent knowledge graph entities, researchers have also proposed

TABLE 6
A Summary of Benchmark Datasets for Evaluating Network Representation Learning

Category	Dataset	Type	$\|V\|$	$\|E\|$	$\|\mathcal{Y}\|$	Multi-label	Vertex attr.
Social Network	BlogCatalog[a]	undirected, binary	10,312	333,983	39	Yes	No
	Flickr[b]	undirected, binary	80,513	5,899,882	195	Yes	No
	YouTube[b]	undirected, binary	1,138,499	2,990,443	47	Yes	No
	Facebook[c]	undirected, binary	4,039	88,234	4	No	Yes
	Amherst[d] [86]	undirected, binary	2,021	81,492	15	No	Yes
	Hamilton[d] [86]	undirected, binary	2,118	87,486	15	No	Yes
	Mich[d] [86]	undirected, binary	2,933	54,903	13	No	Yes
	Rochester[d] [86]	undirected, binary	4,145	145,305	19	No	Yes
Language Network	Wikipedia [1]	undirected, weighted	1,985,098	1,000,924,086	N/A	N/A	No
Citation Network	DBLP (PaperCitation) [1], [87]	directed, binary	781,109	4,191,677	7	No	Yes
	DBLP (AuthorCitation) [1], [87]	directed, weighted	524,061	20,580,238	7	No	No
	Cora[e]	directed, binary	2,708	5,429	7	No	Yes
	Citeseer[e]	directed, binary	3,312	4,732	6	No	Yes
	PubMed[e]	directed, binary	19,717	44,338	3	No	Yes
	Citeseer-M10[f]	directed, binary	10,310	77,218	10	No	Yes
Collaboration network	Arxiv GR-QC [88]	undirected, binary	5,242	28,980	N/A	N/A	No
Webpage Network	Wikipedia[e]	directed, binary	2,405	17,981	20	No	Yes
	WebKB[e]	directed, binary	877	1,608	5	No	Yes
	Political Blog [89]	directed, binary	1,222	16,715	2	No	No
Biological Network	Protein-Protein Interaction [90]	undirected, binary	4,777	184,812	40	Yes	No
Communication Network	Enron Email Network[g]	undirected, binary	36,692	183,831	7	No	No
Traffic Network	European Airline Networks[h]	undirected, binary	N/A	N/A	4	No	No

[a] http://www.public.asu.edu/~ltang9/
[b] http://socialnetworks.mpi-sws.org/data-imc2007.html
[c] https://snap.stanford.edu/data/egonets-Facebook.html
[d] https://escience.rpi.edu/data/DA/fb100/
[e] https://linqs.soe.ucsc.edu/data
[f] http://citeseerx.ist.psu.edu/
[g] https://snap.stanford.edu/data/email-Enron.html
[h] http://complex.unizar.es/~atnmultiplex/

to use such representation to further enhance and complete the knowledge graph itself. For example, knowledge graph completion intends to discover complete relationships between entities, and a recent work [85] has proposed to use graph context to find missing links between entities. This is similar to link prediction in social networks, but the entities are typically heterogeneous and a pair of entities may also have different types of relationships.

7 Evaluation Protocols

In this section, we discuss evaluation protocols for validating the effectiveness of network representation learning. This includes a summary of commonly used benchmark datasets and evaluation methods, followed by a comparison of algorithm performance and complexity.

7.1 Benchmark Datasets

Benchmark datasets play an important role for the research community to evaluate the performance of newly developed NRL algorithms as compared to the existing baseline methods. A handful of network datasets have been made publicly available to facilitate the evaluation of NRL algorithms across different tasks. We summarize a list of network datasets used by most of the published network representation learning papers in Table 6.

Table 6 summarizes the main characteristics of the publicly available benchmark datasets, including the type of network (directed or undirected, binary or weighted), number of vertices $|V|$, number of edges $|E|$, number of labels $|\mathcal{Y}|$, whether the network is multi-labeled or not, as well as whether network vertices are attached with attributes. In Table 6, according to the property of information networks, we classify benchmark datasets into eight different types:

Social Network. The BlogCatalog, Flickr and YouTube datasets are formed by users of the corresponding online social network platforms. For the three datasets, vertex labels are defined by user interest groups but user attributes are unavailable. The Facebook network is a combination of 10 Facebook ego-networks, where each vertex contains user profile attributes. The Amherst, Hamilton, Mich and Rochester [86] datasets are the Facebook networks formed by users from the corresponding US universities, where each user has six user profile features. Often, user profile features are noisy, incomplete, and long-tail distributed.

Language Network. The language network Wikipedia [1] is a word co-occurrence network constructed from the entire set of English Wikipedia pages. There is no class label on this network. The word embeddings learned from this network are evaluated by word analogy and document classification.

Citation Network. The citation networks are directed information networks formed by author-author citation relationships or paper-paper citation relationships. They are collected from different databases of academic papers, such as DBLP and Citeseer. Among the commonly used citation networks, DBLP (AuthorCitation) [1] is a weighted citation network between authors with the edge weight defined by the number of papers written by one author and cited by the other author, while DBLP (PaperCitation) [1], Cora, Citeseer, PubMed and Citeseer-M10 are the binary paper citation networks, which are also attached with vertex text attributes as the content of papers. Compared with user profile features

in social networks, the vertex text features here are more topic-centric, informative and can better complement network structure to learn effective vertex representations.

Collaboration Network. The collaboration network Arxiv GR-QC [88] describes the co-author relationships for papers in the research field of General Relativity and Quantum Cosmology. In this network, vertices represent authors and edges indicate co-author relationships between authors. Because there is no category information for vertices, this network is used for the link prediction task to evaluate the quality of learned vertex representations.

Webpage Network. Webpage networks (Wikipedia, WebKB and Political Blog [89]) are composed of real-world webpages and hyperlinks between them, where the vertex represents a webpage and the edge indicates that there is a hyperlink from one webpage to another. Webpage text content is often collected as vertex features.

Biological Network. As a typical biological network, the Protein-Protein Interaction network [90] is a subgraph of the PPI network for Homo Sapiens. The vertex here represents a protein and the edge indicates that there is an interaction between proteins. The labels of vertices are obtained from the hallmark gene sets [91] and represent biological states.

Communication Network. The Enron Email Network is formed by the Email communication between Enron employees, with vertices being employees and edges representing the email communicated between employees. Employees are labeled as 7 roles (e.g., CEO, president and manager), according to their functions.

Traffic Network. European Airline Networks used in [39] are constructed from 6 airlines operating flights between European airports: 4 commercial airlines (Air France, Easyjet, Lufthansa, and RyanAir) and 2 cargo airlines (TAP Portugal, and European Airline Transport). For each airline network, vertices are airports and edges represent the direct flights between airports. In all, 45 airports are labeled as hub airports, regional hubs, commercial hubs, and focus cities, according to their structural roles.

7.2 Evaluation Methods

It is difficult to directly compare the quality of the vertex representations learned by different NRL algorithms, due to the unavailability of ground truth. Alternatively, in order to evaluate the effectiveness of NRL algorithms on learned vertex representations, several network analytic tasks are commonly used for comparison studies.

Network Reconstruction. The aim of network reconstruction is to reconstruct the original network from the learned vertex representations by predicting the links between vertices based on the inner product or similarity between vertex representations. The known links in the original network serve as the ground truth for evaluating reconstruction performance. $precision@k$ and MAP [19] are often used as evaluation metrics. This evaluation method can check whether the learned vertex representations well preserve network structure and support network formation.

Vertex Classification. As an evaluation method for NRL, vertex classification is conducted by taking learned vertex representations as features to train a classifier on labeled vertices. The classification performance on unlabeled vertices is used to evaluate the quality of the learned vertex representations. Different vertex classification settings, including binary-class classification, multi-class classification, and multi-label classification, are often carried out, depending on the underlying network characteristics. For binary-class classification, F_1 score is used as the evaluation criterion. For multi-class and multi-label classification, $Micro\text{-}F_1$ and $Macro\text{-}F_1$ are adopted as evaluation criteria.

Vertex Clustering. To validate the effectiveness of NRL algorithms, vertex clustering is also carried out by applying k-means clustering algorithm to the learned vertex representations. Communities in networks are served as the ground truth to assess the quality of clustering results, which is measured by $Accuracy$ and NMI (normalized mutual information) [92]. The hypothesis is that, if the learned vertex representations are indeed informative, vertex clustering on learned vertex representations should be able to discover community structures. That is, good vertex representations are expected to generate good clustering results.

Link Prediction. Link prediction can be used to evaluate whether the learned vertex representations are informative to support the network evolution mechanism. To perform link prediction on a network, a portion of edges are first removed, and vertex representations are learned from the remaining network. Finally, the removed edges are predicted with the learned vertex representations. The performance of link prediction is measured by AUC and $precision@k$.

Visualization. Visualization provides a straightforward way to visually evaluate the quality of the learned vertex representations. Often, t-distributed stochastic neighbor embedding (t-SNE) [81] is applied to project the learned vertex representation vectors into a 2-D space, where the distribution of vertex 2-D mappings can be easily visualized. If vertex representations are of good quality, in the 2-D space, vertices within a same class or community should be embedded closely, and the 2-D mappings of vertices in different classes or communities should be far apart from each other.

In Table 7, we summarize the type of information networks and network analytic tasks used to evaluate the quality of vertex representations learned by existing NRL algorithms. We also provide hyperlinks for the codes of respective NRL algorithms if available to help interested readers to further study these algorithms or run experiments for comparison. Overall, social networks and citation networks are frequently used as benchmark datasets, and vertex classification is most commonly used as the evaluation method in both unsupervised and semi-supervised settings.

7.3 Empirical Results

We observe from the literature that empirical evaluation is often carried out on different datasets under different settings. There is a lack of consistency on empirical results to determine the best performing algorithms and their circumstances. Therefore, we perform benchmark experiments to fairly compare the performance of several representative NRL algorithms on the same set of datasets. Note that, because semi-supervised NRL algorithms are task-dependent: the target task may be binary or multi-class, or multi-label classification, or because they use different classification strategies, it would be difficult to assess the effectiveness of network embedding under the same settings. Therefore, our empirical study focuses on comparing seven

TABLE 7
A Summary of NRL Algorithms with Respect to the Evaluation Methodology

Category	Algorithm	Network Type	Evaluation Method	Code Link
Unsupervised	Social Dim. [31], [32], [33]	Social Network	Vertex Classification	
	DeepWalk [6]	Social Network	Vertex Classification	https://github.com/phanein/deepwalk
	LINE [1]	Citation Network Language Network Social Network	Vertex Classification Visualization	https://github.com/tangjianpku/LINE
	GraRep [26]	Citation Network Language Network Social Network	Vertex Classification Vertex Clustering Visualization	https://github.com/ShelsonCao/GraRep
	DNGR [9]	Language Network	Vertex Clustering Visualization	https://github.com/ShelsonCao/DNGR
	SDNE [19]	Collaboration Network Language Network Social Network	Network Reconstruction Vertex Classification Link Prediction Visualization	https://github.com/suanrong/SDNE
	node2vec [34]	Biological Network Language Network Social Network	Vertex Classification Link Prediction	https://github.com/aditya-grover/node2vec
	HOPE [35]	Social Network Citation Network	Network Reconstruction Link Prediction	
	APP [36]	Social Network Citation Network Collaboration Network	Link Prediction	
	GraphGAN [37]	Citation Network Language Network Social Network	Vertex Classification Link Prediction	
	M-NMF [28]	Social Network Webpage Network	Vertex Classification Vertex Clustering	http://git.thumedia.org/embedding/M-NMF
	struct2vec [38]	Traffic Network	Vertex Classification	
	GraphWave [39]	Traffic Network Communication Network	Vertex Clustering Visualization	http://snap.stanford.edu/graphwave
	SNS [40]	Social Network Language Network Biological Network	Vertex Classification	
	DP [41]	Social Network Citation Network Collaboration Network	Network Reconstruction Link Prediction Vertex Classification	
	HARP [42]	Social Network Collaboration Network Citation Network	Vertex Classification Visualization	
	TADW [7]	Citation Network Webpage Network	Vertex Classification	https://github.com/thunlp/tadw
	HSCA [20]	Citation Network Webpage Network	Vertex Classification Visualization	https://github.com/daokunzhang/HSCA
	pRBM [29]	Social Network	Vertex Clustering	
	UPP-SNE [43]	Social Network	Vertex Classification Vertex Clustering	
	PPNE [44]	Social Network Citation Network Webpage network	Vertex Classification Link Prediction	
Semi-supervised	DDRW [45]	Social Network	Vertex Classification	
	MMDW [46]	Citation Network Webpage Network	Vertex Classification Visualization	https://github.com/thunlp/MMDW
	TLINE [47]	Citation Network Collaboration Network	Vertex Classification Visualization	
	GENE [48]	Social Network	Vertex Classification	
	SemiNE [49]	Social Network	Network Reconstruction Vertex Classification Link prediction	
	TriDNR [50]	Citation Network	Vertex Classification Visualization	https://github.com/shiruipan/TriDNR
	LDE [51]	Social Network Citation Network	Vertex Classification	
	DMF [8]	Citation Network	Vertex Classification Visualization	https://github.com/daokunzhang/DMF_CC
	Planetoid [52]	Citation Network	Vertex Classification Visualization	https://github.com/kimiyoung/planetoid
	LANE [30]	Social Network	Vertex Classification	

unsupervised NRL algorithms (DeepWalk [6], LINE [1], node2vec [34], M-NMF [28], TADW [7], HSCA [20], UPP-SNE [43]) on vertex classification and vertex clustering, which are the two most commonly used evaluation methods in the literature.

Our empirical studies are based on seven benchmark datasets: Amherst, Hamilton, Mich, Rochester, Citeseer, Cora and Facebook. Following [28], for Amherst, Hamilton, Mich and Rochester, only the network structure is used and the attribute "year" is used as class label, which is a good indicator of community structure. For Citeseer and Cora, the research area is used as the class label. The class label of Facebook dataset is given by the attribute "education type".

7.3.1 Experimental Setup

For random walk based methods, DeepWalk, node2vec and UPP-SNE, we uniformly set the number of walks, walk length and window size as 10, 80, 10, respectively. For UPP-SNE, we use the implementation that is optimized by stochastic gradient descent. The parameter p and q of node2vec are set to 1, as the default setting. For M-NMF, we set α and β as 1. For all algorithms, the dimension of learned vertex representations is set to 256. For LINE, we learn 128-dimensional vertex representations with the first-order proximity preserving version and the second-order proximity preserving version respectively and concatenate them together to obtain 256-dimensional vertex representations. The other

TABLE 8
Vertex Classification Results on Seven Datasets

	Method	Training ratio = 5%							Training ratio = 50%						
		Amherst	Hamilton	Mich	Rochester	Citeseer	Cora	Facebook	Amherst	Hamilton	Mich	Rochester	Citeseer	Cora	Facebook
$Micro\text{-}F_1$	DeepWalk	0.7168	0.7127	0.3933	0.6795	0.5061	0.7333	0.6839	0.8106	0.8188	0.4829	0.7822	0.5927	0.8292	0.6782
	LINE	0.7351	0.7367	0.4101	**0.7163**	0.3842	0.5625	0.6832	0.8240	0.8415	**0.5046**	0.8067	0.5353	0.7572	0.6848
	node2vec	**0.7528**	**0.7622**	**0.4163**	0.7018	0.5135	0.7395	0.6911	0.8063	0.8239	0.4900	0.7625	0.5936	0.8126	0.6944
	M-NMF	0.7325	0.7471	0.3865	0.7047	0.4070	0.5704	0.6875	**0.8280**	**0.8476**	0.4827	**0.8076**	0.5979	0.7635	0.6849
	TADW					0.6206	0.7257	0.7260					0.7379	0.8648	**0.8748**
	HSCA					0.6309	0.7737	0.6827					**0.7396**	**0.8693**	0.6955
	UPP-SNE					**0.6579**	**0.7745**	**0.8467**					0.7105	0.8429	0.8711
$Macro\text{-}F_1$	DeepWalk	0.3372	0.2829	0.1726	0.1925	0.4487	0.7103	0.2431	**0.4628**	**0.3838**	0.2249	**0.2549**	0.5281	0.8203	0.2529
	LINE	**0.3420**	0.2912	0.1823	0.2043	0.3456	0.5321	0.2350	0.4107	0.3487	**0.2395**	0.2540	0.4851	0.7504	0.2460
	node2vec	0.3158	0.2912	**0.1825**	0.1893	0.4577	0.7193	0.2231	0.3568	0.3211	0.2214	0.2207	0.5370	0.8035	0.2207
	M-NMF	0.3206	**0.2951**	0.1774	**0.2050**	0.3665	0.5377	0.2183	0.3895	0.3684	0.2341	0.2540	0.5494	0.7554	0.2362
	TADW					0.5614	0.7031	0.2926					**0.6920**	0.8527	**0.4425**
	HSCA					0.5712	**0.7544**	0.2219					0.6909	**0.8571**	0.2459
	UPP-SNE					**0.5847**	0.7451	**0.4177**					0.6509	0.8277	0.4355

parameters of the above algorithms are all set to their default values.

Taking the learned vertex representations as input, we carry out vertex classification and vertex clustering experiments to evaluate the quality of learned vertex representations. For vertex classification, we randomly select 5 and 50 percent samples to train an SVM classifier (with the LIBLINEAR implementation [66]) and test it on the remaining samples. We repeat this process 10 times and report the averaged $Micro\text{-}F_1$ and $Macro\text{-}F_1$ values. We adopt k-means to perform vertex clustering. To reduce the variance caused by random initialization, we repeat the clustering process for 20 times and report the averaged $Accuracy$ and NMI values.

7.3.2 Performance Comparison

Tables 8 and 9 compare the performance of different algorithms on vertex classification and vertex clustering. For each dataset, the best performing method across all baselines is bold-faced. For the attributed networks (Citeseer, Cora and Facebook), the underlined results indicate the best performer among the structure only preserving NRL algorithms (DeepWalk, LINE, node2vec and M-NMF).

Table 8 shows that among structure only preserving NRL algorithms, when the training ratio is 5 percent, node2vec achieves the best classification performance overall, and when the training ratio is 50 percent, M-NMF performs best in terms of $Micro\text{-}F_1$ while DeepWalk is the winner of $Macro\text{-}F_1$. Here, M-NMF does not exhibit significant advantage over DeepWalk, LINE and node2vec. This is probably due to that the parameter α and β of N-NMF are not optimally tuned; their values must be carefully chosen so as to achieve a good trade-off between different components. On attributed networks (Citeseer, Cora and Facebook), the content augmented NRL performs much better than the structure only preserving NRL algorithms. This proves that vertex attributes can largely contribute to learning more informative vertex representations. When training ratio is 5 percent, UPP-SNE is the best performer. This indicates that the UPP-SNE's non-linear mapping provides a better way to construct vertex representations from vertex attributes than the linear mapping, as is done in TADW and HSCA. When training ratio is 50 percent, TADW achieves the best overall classification performance, although in some cases, it is slightly outperformed by HSCA. On citation networks (Citeseer and Cora), HSCA performs better than TADW, while it yields worse performance than TADW on Facebook. This might be caused by the fact that the homophily property of Facebook social network is weaker than that of citation networks. The homophily preserving objective should be weighted less to make HSCA achieve satisfactory performance on Facebook.

Table 9 shows that LINE achieves the best clustering performance on Amherst, Hamilton, Mich and Rochester. As LINE's vertex representations capture both the first-order and second-order proximity, it can better preserve the community structure, leading to good clustering performance. On Citeseer, Cora and Facebook, the content augmented NRL algorithm UPP-SNE performs best. As UPP-SNE constructs vertex representations from vertex attributes via a non-linear mapping, the well preserved content information favors the best clustering performance. On Citeseer and Cora, node2vec performs much better than other structure only preserving NRL algorithms, including its equivalent version DeepWalk. For each vertex context pair (v_i, v_j), DeepWalk and node2vec use two different strategies to approximate the probability $\Pr(v_j|v_i)$: hierarchical softmax [93], [94] and negative sampling [95]. The better clustering performance of node2vec over DeepWalk proves the advantage of negative sampling over hierarchical softmax, which is consistent with the word embedding results as reported in [67].

7.4 Complexity Analysis

To better understand the existing NRL algorithms, we provide a detailed analysis of their time complexity and underlying optimization methods in Table 10. A new notation I is introduced to represent the number of iterations and we use $nnz(\cdot)$ to denote the number of non-zero entries of a matrix. In a nutshell, four kinds of solutions are used to optimize the objectives of the existing NRL algorithms: (1) *eigen decomposition* that involves finding top-d eigenvectors of a

TABLE 9
Vertex Clustering Results on Seven Datasets

	Method	Amherst	Hamilton	Mich	Rochester	Citeseer	Cora	Facebook
Accuracy	DeepWalk	0.6257	0.6273	0.3944	0.5593	0.3365	0.5062	0.6953
	LINE	**0.6908**	**0.6718**	**0.4127**	**0.6070**	0.2806	0.3905	0.6952
	node2vec	0.6662	0.6328	0.4114	0.5777	0.4574	0.6216	0.6952
	M-NMF	0.6545	0.6374	0.3279	0.5071	0.2379	0.3640	0.6952
	TADW					0.2778	0.4731	0.6953
	HSCA					0.2794	0.4594	0.6957
	UPP-SNE					**0.5748**	**0.6832**	**0.8328**
NMI	DeepWalk	0.4873	0.4390	**0.1897**	0.3468	0.0896	0.3308	0.0142
	LINE	**0.5030**	**0.4529**	0.1858	**0.3547**	0.0511	0.1639	0.0113
	node2vec	0.4742	0.4144	0.1824	0.3193	0.2027	0.4333	0.0162
	M-NMF	0.4696	0.4330	0.1304	0.2971	0.0464	0.1201	0.0176
	TADW					0.0845	0.3001	0.0651
	HSCA					0.0902	0.3148	0.0151
	UPP-SNE					**0.3005**	**0.4911**	**0.2095**

matrix, (2) *alternative optimization* that optimizes one variable with the remaining variables fixed alternately, (3) *gradient descent* that updates all parameters at each iteration for optimizing the overall objective, and (4) *stochastic gradient descent* that optimizes the partial objective stochastically in an on-line mode.

Both unsupervised and semi-supervised NRL algorithms mainly adopt stochastic gradient descent to solve their optimization problems. The time complexity of these algorithms is often linear with respect to the number of vertices/edges, which makes them scalable to large-scale networks. By contrast, other optimization strategies usually involve higher time complexity, which is quadratic with regards to the number of vertices, or even higher with the scale of the number of vertices times the number of edges. The corresponding NRL algorithms usually perform factorization on a $|V| \times |V|$ structure preserving matrix, which is quite time-consuming. Efforts have been made to reduce the complexity of matrix factorization. For example, TADW [7], DMF [8] and HSCA [20] take advantage of the sparsity of the original vertex-context matrix. HOPE [35] and GraphWave [39] adopt advanced techniques [96], [97] to perform matrix eigen decomposition.

8 Future Research Directions

In this section, we summarize six potential research directions and future challenges to stimulate research on network representation learning.

Task-Dependence. To date, most existing NRL algorithms are task-independent, and task-specific NRL algorithms have primarily focused on vertex classification under the semi-supervised setting. Only very recently, a few studies have started to design task-specific NRL algorithms for link prediction [35], community detection [98], [99], [100], [101], class imbalance learning [102], active learning [103], and information retrieval [104]. The advantage of using network representation learning as an intermediate layer to solve the target task is that the best possible information preserved in the new representation can further benefit the subsequent task. Thus, a desirable task-specific NRL algorithm must preserve information critical to the specific task in order to optimize its performance.

Theory. Although the effectiveness of the existing NRL algorithms has been empirically proved through experiments, the underlying working mechanism has not been well understood. There is a lack of theoretical analysis with regard to properties of algorithms and what contributes to good empirical results. To better understand DeepWalk [6], LINE [1], and node2vec [34], [105] discovers their theoretical connections to graph Laplacians. However, in-depth theoretical analysis about network representation learning is necessary, as it provides a deep understanding of algorithms and helps interpret empirical results.

Dynamics. Current research on network representation learning has mainly concerned static networks. However, in real-life scenarios, networks are not always static. The underlying network structure may evolve over time, i.e., new vertices/edges appear while some old vertices/edges disappear. The vertices/edges may also be described by some time-varying information. Dynamic networks have unique characteristics that make static network embedding fail to work: (i) vertex content features may drift over time; (ii) the addition of new vertices/edges requires learning or updating vertex representations to be efficient; and (iii) network size is not fixed. The work on dynamic network embedding is rather limited; the majority of existing approaches (e.g., [106], [107], [108]) assume that the node set is fixed and deal with the dynamics caused by the deletion/addition of edges only. However, a more challenging problem is to predict the representations of new added vertices, which is referred to as "out-of-sample" problem. A few attempts such as [52], [109], [110] are made to exploit inductive learning to address this issue. They learn an explicit mapping function from a network at a snapshot, and use this function to infer the representations of out-of-sample vertices, based on their available information such as attributes or neighborhood structure. However, they have not considered how to incrementally update the existing mapping function. How to design effective and efficient representation learning algorithms in complex dynamic domains still requires further exploration.

Scalability. The scalability is another driving factor to advance the research on network representation learning. Several NRL algorithms have made attempts to scale up to

TABLE 10
Complexity Analysis

Category	Algorithm	Complexity	Optimization Method
Unsupervised	Social Dim. [31], [32], [33]	$O(d\|V\|^2)$	Eigen Decomposition
	GraRep [26]	$O(\|V\|\|E\| + d\|V\|^2)$	
	HOPE [35]	$O(d^2 I\|E\|)$	
	GraphWave [39]	$O(\|E\|)$	
	DeepWalk [6]	$O(d\|V\| \log \|V\|)$	Stochastic Gradient Descent
	LINE [1]	$O(d\|E\|)$	
	SDNE [19]	$O(dI\|V\|^2)$	
	node2vec [34]	$O(d\|V\|)$	
	APP [36]	$O(d\|V\|)$	
	GraphGAN [37]	$O(\|V\| \log \|V\|)$	
	struct2vec [38]	$O(\|V\|^3)$	
	SNS [40]	$O(d\|V\|)$	
	pRBM [29]	$O(dmI\|V\|)$	
	PPNE [44]	$O(d\|V\|)$	
	M-NMF [28]	$O(dI\|V\|^2)$	Alternative Optimization
	TADW [7]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	
	HSCA [8]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	
	UPP-SNE [43]	$O(I\|E\| \cdot nnz(X))$	Gradient Descent
Semi-supervised	DDRW [45]	$O(d\|V\| \log \|V\|)$	Stochastic Gradient Descent
	TLINE [47]	$O(d\|E\|)$	
	SemiNE [49]	$O(d\|V\|)$	
	TriDNR [50]	$O(d \cdot nnz(X) \log m + d\|V\| \log \|V\|)$	
	LDE [51]	$O(dI \cdot nnz(X) + dI\|E\| + dI\|\mathcal{Y}\|\|V\|)$	
	DMF [8]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	Alternative Optimization
	LANE [30]	$O(m\|V\|^2 + dI\|V\|^2)$	

large-scale networks with linear time complexity with respect to the number of vertices/edges. Nevertheless, the scalability still remains a major challenge. Our findings on complexity analysis show that random walk and edge modeling based methods that adopt stochastic gradient descent optimization are much more efficient than matrix factorization based methods that are solved by eigen decomposition and alternative optimization. Matrix factorization based methods have shown great promise in incorporating vertex attributes and discovering community structures, but their scalability needs to be improved to handle networks with millions or billions of vertices. Deep learning based methods can capture non-linearity in networks, but their computational cost is usually high. Traditional deep learning architectures take advantage of GPU to speed up training on euclidean structured data [111]. However, networks do not have such a structure, and therefore require new solutions to improve the scalability [112].

Heterogeneity and Semantics. Representation learning for heterogeneous information networks (HIN) is one promising research direction. The vast amounts of existing work has focused on homogeneous network embedding, where all vertices are of the same type and edges represent a single relation. However, there is an increasing need to study heterogeneous information networks with different types of vertices and edges, such as DBLP, DBpedia, and Flickr. An HIN is composed of different types of entities, such as text, images, or videos, and the interdependencies between entities are very complex. This makes it very difficult to measure rich semantics and proximity between vertices and seek a common and coherent embedding space. Recent studies by [16], [113], [114], [115], [116], [117], [118], [119], [120], [121] have investigated the use of various descriptors (e.g., metapath or meta structure) to capture semantic proximity between distant HIN vertices for representation learning. However, the research along this line is still at early stage. Further research requires to investigate better ways for capturing the proximity between cross-modal data, and their interplay with network structure.

Another interesting direction is to investigate edge semantics in signed networks, where vertices have both positive and negative relationships. Signed networks are ubiquitous in social networks, such as Epinions and Slashdot, that allow users to form positive or negative friendship/trust connection to other users. The existence of negative links makes the traditional homophily based network representation learning algorithms unable to be directly applied. Some studies [122], [123], [124] tackle signed network representation learning through directly modeling the polar of links. How to fully encode network structure and vertex attributes for signed network embedding remains an open question.

Robustness. Real-world networks are often noisy and uncertain, which makes traditional NRL algorithms unable to produce stable and robust representations. Adversarial Network Embedding (ANE) [125] and Adversarially Regularized Graph Autoencoder (ARGA) [126] learn robust vertex representations via enforcing an adversarial learning regularizer [58]. To deal with the uncertainty in the existence of edges, Uncertain Graph Embedding (URGE) [127] encodes the edge existence probability into the vertex representation learning process. It is of great importance to have more research efforts on enhancing the robustness of network representation learning.

9 Conclusion

This survey provides a comprehensive review of the state-of-the-art network representation learning algorithms in the

data mining and machine learning field. We propose a taxonomy to summarize existing techniques into two settings: unsupervised setting and semi-supervised settings. According to the information sources they use and the methodologies they employ, we further categorize different methods at each setting into subgroups, review representative algorithms in each subgroup, and compare their advantages and disadvantages. We summarize evaluation protocols used for validating existing NRL algorithms, compare their empirical performance and complexity, as well as point out a few emerging research directions and the promising extensions. Our categorization and analysis not only help researchers to gain a comprehensive understanding of existing methods in the field, but also provide rich resources to advance the research on network representation learning.

ACKNOWLEDGMENTS

The work was supported by the US National Science Foundation (NSF) through grant IIS-1763452, and the Australian Research Council (ARC) through grant LP160100630 and DP180100966. Daokun Zhang was supported by China Scholarship Council (CSC) with No. 201506300082 and a post-graduate scholarship from Data61, CSIRO in Australia.

REFERENCES

[1] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[2] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Rep.*, vol. 533, no. 4, pp. 95–142, 2013.

[3] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Sci.*, vol. 290, no. 5500, pp. 2319–2323, 2000.

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

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 585–591.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[7] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.

[8] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Collective classification via discriminative matrix factorization on sparsely labeled networks," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 1563–1572.

[9] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.

[10] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 487–494.

[11] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*. Berlin, Germany: Springer, 2011, pp. 115–148.

[12] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.

[13] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1169–1174.

[14] C. Zhang, K. Zhang, Q. Yuan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, and J. Han, "Regions, periods, activities: Uncovering Urban dynamics via cross-modal representation learning," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 361–370.

[15] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based POI embedding for location-based recommendation," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 15–24.

[16] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, and J. Ying, "Distance-aware DAG embedding for proximity search on heterogeneous graphs," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2355–2362.

[17] J. Tang, J. Liu, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 287–297.

[18] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2181–2187.

[19] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.

[20] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proc. 16th IEEE Int. Conf. Data Mining*, 2016, pp. 609–618.

[21] L. G. Moyano, "Learning network representations," *Eur. Phys. J. Special Topics*, vol. 226, no. 3, pp. 499–518, 2017.

[22] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, 2018.

[23] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bulletin*, vol. 40, no. 3, pp. 52–74, 2017.

[24] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, 2018.

[25] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Trans. Knowl. Data Eng.*, 2018.

[26] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.

[27] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. Nat. Academy Sci. United States America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[28] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.

[29] S. Wang, J. Tang, F. Morstatter, and H. Liu, "Paired restricted Boltzmann machine for linked data," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 1753–1762.

[30] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.

[31] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 817–826.

[32] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining Knowl. Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[33] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *Proc. 18th ACM Int. Conf. Inf. Knowl. Manage.*, 2009, pp. 1107–1116.

[34] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[35] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.

[36] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2942–2948.

[37] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2508–2515.

[38] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 385–394.

[39] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Spectral graph wavelets for structural role similarity in networks," arXiv:1710.10321, 2017.

[40] T. Lyu, Y. Zhang, and Y. Zhang, "Enhancing the network embedding quality with structural similarity," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 147–156.

[41] R. Feng, Y. Yang, W. Hu, F. Wu, and Y. Zhuang, "Representation learning for scale-free networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 282–289.

[42] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "HARP: Hierarchical representation learning for networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2127–2134.

[43] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "User profile preserving social network embedding," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3378–3384.

[44] C. Li, S. Wang, D. Yang, Z. Li, Y. Yang, X. Zhang, and J. Zhou, "PPNE: Property preserving network embedding," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 163–179.

[45] J. Li, J. Zhu, and B. Zhang, "Discriminative deep random walk for network classification," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguistics*, 2016, pp. 1004–1013.

[46] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-Margin DeepWalk: Discriminative learning of network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3889–3895.

[47] X. Zhang, W. Chen, and H. Yan, "TLINE: Scalable transductive network embedding," in *Information Retrieval Technology*. Berlin, Germany: Springer, 2016, pp. 98–110.

[48] J. Chen, Q. Zhang, and X. Huang, "Incorporate group information to enhance network embedding," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 1901–1904.

[49] C. Li, Z. Li, S. Wang, Y. Yang, X. Zhang, and J. Zhou, "Semi-supervised network embedding," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 131–147.

[50] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1895–1901.

[51] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 115–124.

[52] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 40–48.

[53] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006, Art. no. 036104.

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

[55] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv:1301.3781, 2013.

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[57] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Sci.*, vol. 313, no. 5786, pp. 504–507, 2006.

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[59] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[60] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[61] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[62] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu, "Scalable proximity estimation and link prediction in online social networks," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 322–335.

[63] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.

[64] J. Zhu, A. Ahmed, and E. P. Xing, "MedLDA: Maximum margin supervised topic models," *J. Mach. Learn. Res.*, vol. 13, no. Aug., pp. 2237–2278, 2012.

[65] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1033–1040.

[66] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, no. Aug., pp. 1871–1874, 2008.

[67] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[68] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati, "Hierarchical neural language models for joint representation of streaming documents and their content," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 248–255.

[69] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv:1207.0580, 2012.

[70] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, 2008, Art. no. 93.

[71] P. Kazienko and T. Kajdanowicz, "Label-dependent node classification in the network," *Neurocomput.*, vol. 75, no. 1, pp. 199–209, 2012.

[72] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[73] V. Martinez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Comput. Surveys*, vol. 49, no. 4, pp. 69, 2017.

[74] S. Fortunato, "Community detection in graphs," *Physics Rep.*, vol. 486, no. 3–5, pp. 75–174, 2010.

[75] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 107–114.

[76] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[77] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physics Rev.*, vol. 69, 2004, Art. no. 026113.

[78] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Academy Sci. United States America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[79] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 824–833.

[80] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 1, pp. 24–43, Jan.–Mar. 2000.

[81] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov., pp. 2579–2605, 2008.

[82] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. World Wide Web Int. Conf.*, 2009, pp. 791–800.

[83] P. Wang, J. Zhang, G. Liu, Y. Fu, and C. Aggarwal, "Ensemble-spotting: Prioritizing vibrant communities via POI embedding with multi-view spatial graphs," in *Proc. SIAM Int. Conf. Data Mining*, 2018, pp. 351–359.

[84] A. Bordes, J. Weston, R. Collobert, Y. Bengio et al., "Learning structured embeddings of knowledge bases," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 301–306.

[85] J. Feng, M. Huang, Y. Yang et al., "GAKE: Graph aware knowledge embedding," in *Proc. 26th Int. Conf. Comput. Linguistics*, 2016, pp. 641–651.

[86] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of Facebook networks," *Physica A: Statistical Mech. Appl.*, vol. 391, no. 16, pp. 4165–4180, 2012.

[87] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 990–998.

[88] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, 2007, Art. no. 2.

[89] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 US election: Divided they blog," in *Proc. 3rd Int. Workshop Link Discovery*, 2005, pp. 36–43.

[90] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, et al., "The BioGRID interaction database: 2008 update," *Nucleic Acids Res.*, vol. 36, pp. D637–D640, 2008.
[91] A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, and J. P. Mesirov, "Molecular signatures database (MSigDB) 3.0," *Bioinf.*, vol. 27, no. 12, pp. 1739–1740, 2011.
[92] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2003.
[93] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1081–1088.
[94] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 246–252.
[95] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *J. Mach. Learn. Res.*, vol. 13, no. Feb., pp. 307–361, 2012.
[96] M. Hochstenbach, "A Jacobi–Davidson type method for the generalized singular value problem," *Linear Algebra Appl.*, vol. 431, no. 3/4, pp. 471–487, 2009.
[97] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops*, 2011, pp. 1–8.
[98] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 377–386.
[99] L. Yang, X. Cao, and Y. Guo, "Multi-facet network embedding: Beyond the general solution of detection and representation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 499–506.
[100] Y. Zhang, T. Lyu, and Y. Zhang, "COSINE: Community-preserving social network embedding from information diffusion cascades," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2620–2627.
[101] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized hraph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.
[102] Z. Wang, X. Ye, C. Wang, Y. Wu, C. Wang, and K. Liang, "RSDNE: Exploring relaxed similarity and dissimilarity from completely-imbalanced labels for network embedding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 475–482.
[103] X. Huang, Q. Song, J. Li, and X. Hu, "Exploring expert cognition for attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 270–278.
[104] V. Misra and S. Bhatia, "Bernoulli embeddings for graphs," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3812–3819.
[105] 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 *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 459–467.
[106] D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, "From properties to links: Deep network embedding on incomplete graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 367–376.
[107] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 387–396.
[108] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 571–578.
[109] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances Neural Inform. Proces. Syst.*, pp. 1025–1035, 2017.
[110] J. Ma, P. Cui, and W. Zhu, "DepthLGP: Learning embeddings of out-of-sample nodes in dynamic networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 370–377.
[111] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep feature learning for graphs," arXiv:1704.08829, 2017.
[112] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[113] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 119–128.
[114] Z. Huang and N. Mamoulis, "Heterogeneous information network embedding for meta path based proximity," in *arXiv:1701.05291*, 2017.
[115] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 135–144.
[116] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 426–433.
[117] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin, "Multi-dimensional network embedding with hierarchical structure," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 387–395.
[118] Y. Zhang, Y. Xiong, X. Kong, and Y. Zhu, "Learning node embeddings in interaction graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 397–406.
[119] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1767–1776.
[120] T.-Y. Fu, W.-C. Lee, and Z. Lei, "HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1797–1806.
[121] Y. Chen and C. Wang, "HINE: Heterogeneous information network embedding," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 180–195.
[122] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 327–335.
[123] S. Wang, C. Aggarwal, J. Tang, and H. Liu, "Attributed signed network embedding," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 137–146.
[124] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "SHINE: Signed heterogeneous information network embedding for sentiment link prediction," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 592–600.
[125] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2167–2174.
[126] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018.
[127] J. Hu, R. Cheng, Z. Huang, Y. Fang, and S. Luo, "On embedding uncertain graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 157–166.

Daokun Zhang received the master's degree in computer science from Northwest A&F University, Yangling, Shaanxi, China, in 2015. Since August 2015, he has been working toward the PhD degree in the Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney. His research interests include data mining and machine learning.

Jie Yin received the PhD degree in computer science from the Hong Kong University of Science and Technology, Hong Kong. She is currently a senior lecturer with the Discipline of Business Analytics, University of Sydney, Australia. Her research interests include data mining, machine learning, and their applications to text mining, network analytics, health informatics, and decision support systems. She has published about 60 refereed journal and conference papers in these areas. She is a co-chair of the International Workshop on Social Web for Disaster Management (SWDM 2015, SWDM 2016, and SWDM 2018).

Xingquan Zhu (SM'12) received the PhD degree in computer science from Fudan University, Shanghai, China. He is currently a professor with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, Florida. His research interests include data mining, machine learning, and multimedia systems. Since 2000, he has authored or co-authored more than 230 refereed journal and conference papers in these areas, including two Best Paper Awards and one Best Student Award. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* (2008-2012, and 2014-date), and an associate editor of the *ACM Transactions on Knowledge Discovery from Data* (2017-date). He is a senior member of the IEEE.

Chengqi Zhang (SM'95) received the PhD degree from the University of Queensland, Brisbane, Australia, in 1991, and the DSc degree (higher doctorate) from Deakin University, Geelong, Australia, in 2002. Since February 2017, he has been a distinguished professor with the University of Technology Sydney (UTS), Sydney, Australia, and he has been appointed as an associate vice president (Research Relationships China) with the UTS since December 2017. His research interests mainly focus on data mining and its applications. He has in total more than 300 publications till date. He is a general co-chair of KDD 2015 in Sydney, the local arrangements chair of IJCAI-2017 in Melbourne, a fellow of the Australian Computer Society, and a senior member of the IEEE.