Probabilistic Graph Convolutional Network via Topology-Constrained Latent Space Model

Liang Yang, Yuanfang Guo, *Senior Member, IEEE*, Junhua Gu, Di Jin, Bo Yang, Xiaochun Cao, *Senior Member, IEEE*

***Abstract*—Although many Graph Convolutional Neural Networks (GCNNs) have achieved superior performances in semi-supervised node classification, they are designed from either the spatial or spectral perspective, yet without a *general* theoretical basis. Besides, most of the existing GCNNs methods tend to ignore the ubiquitous noises in the network topology and node content, and thus unable to model these uncertainties. These drawbacks certainly reduce their effectiveness in integrating network topology and node content. To provide a probabilistic perspective to the GCNNs, we model the semi-supervised node classification problem as a topology-constrained probabilistic latent space model, Probabilistic Graph Convolutional Network (PGCN). By representing the nodes in a more efficient distribution form, the proposed framework can seamlessly integrate the node content and network topology. When specifying the distribution in PGCN to be Gaussian distribution, the transductive node classification problems can be solved by the general framework and a specific method, named Probabilistic Graph Convolutional Network with Gaussian distribution representation (PGCN-G) is proposed. To overcome the overfitting problem in covariance estimation and reduce the computational complexity, PGCN-G is further improved to PGCN-G+ by imposing the covariance matrices of all the vertices to possess the identical singular vectors. The optimization algorithm based on Expectation Maximization indicates that the proposed method can iteratively denoise the network topology and node content with respect to each other. Besides of the effectiveness of this *top-down* framework demonstrated via extensive experiments, it can also be deduced to cover the existing methods, GCN, GAT and GMM and elaborate their characteristics and relationships, by specific derivations.**

***Index Terms*—Graph Convolutional Network, probabilistic model, node classification, semi-supervised learning.**

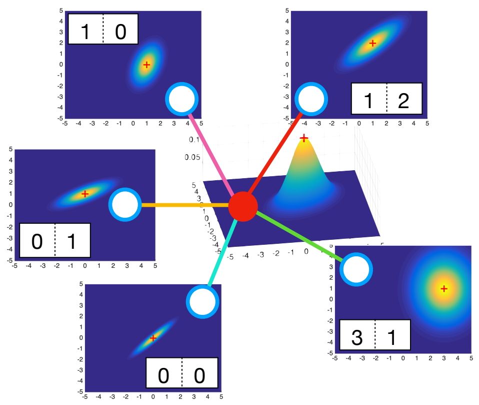


Fig. 1. An illustrative example of the proposed framework. The uncertainties of the links and node contents are modeled as distributions from probabilistic perspective. Each circle stands for a node in the network, while the two-dimensional vector represents its content. Connection between each pair of nodes denotes that there exists an observed edge between them. Although the observed node content is in the vector form, it is modeled as a parametric distribution $p(\mathbf{x}|\boldsymbol{\theta})$. For each observed edge, a distribution $p(\mathbf{z})$ is exploited to model the probability of its two endpoints belonging to the same class. Different node content obeys different distributions, which are represented as the two-dimensional color maps, except the center (red) node. To demonstrate the different probabilities of connected nodes which belong to the same classes, the connections are illustrated via different colors.

I. Introduction

Networks, including social networks, biological networks and technological networks, are ubiquitous in real world [1], [2], [3], [4], [5], [6]. They usually possess some common properties, such as community structure [7], [8], [9], scale-free property [10], etc. Network-based data analysis has become an interesting and challenging topic, and many specific problems have been explored. Among these problems, network partition (node classification) has been widely studied. According to various applications of the to-be-processed network data, traditional network partition techniques can be categorized into two categories: 1) Community detection, a.k.a. graph clustering, only exploits the network topology information [7]; 2) Node classification, a.k.a. semi-supervised community detection, usually resolves the problem by adopting both the network topology and label information [11].

This work was supported in part by the National Natural Science Foundation of China under Grant 61972442, Grant 61802282, Grant 61802391, Grant 61772361, Grant 61971016 and Grant U1936210, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D, in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Beijing Education Committee Cooperation Beijing Natural Science Foundation under Grant KZ201910005007, in part by the Hebei Province Innovation Capacity Enhancement Project under Grant 199676146H, and in part by the Fundamental Research Funds for Central Universities. (Corresponding author: Yuanfang Guo.)

Liang Yang is with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, China (e-mail: yangliang@vip.qq.com).

Junhua Gu is with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China, and also with Hebei Province Key Laboratory of Big Data Calculation, Hebei University of Technology, Tianjin, China (e-mail: jhgu@hebut.edu.cn).

Yuanfang Guo is with the School of Computer Science and Engineering, Beihang University, Beijing, China (e-mail: andyguo@buaa.edu.cn).

Di Jin is with the College of Intelligence and Computing, Tianjin Universitys, Tianjin, China (e-mail: jindi@tju.edu.cn).

Bo Yang is with the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, China, and also with College of Computer Science and Technology, Jilin University, China (e-mail: ybo@jlu.edu.cn).

Xiaochun Cao is with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, China (e-mail: caoxiaochun@iie.ac.cn).

Node classification in attributed network, which exploits the characteristics of network topology, meta-data in the nodes/edges, and the labels, has drawn significant attentions from researchers, because of the varieties of the meta-data. The majority of the existing techniques focuses on developing efficient approaches to extract semantic features from both the network topology and meta-data. Motivated by the feature learning strategy in deep learning [12] and the effectiveness of convolution neural networks (CNNs) [13], many efforts have been made to generalize CNNs to process the graph-structured data.

To handle various sizes of neighbourhoods and maintain the weight sharing property in CNNs, graph convolutional neural networks (GCNNs) [14], [15], [16], [17], [18] are invented and they can be classified into two categories, spatial and spectral approaches. Spatial approaches either learn different weight matrices for nodes with various degrees [15] or sample a fixed-size neighbourhood for each node [19], [20]. MoNet unifies the spatial convolution operation as a weighted mixture of all the neighbours and specifies the weights via a Gaussian kernel [21]. Graph attention networks (GAT) applies the attention mechanism to effectively process the variable sized neighbourhoods [22].

Spectral approaches employ the convolution operation in Fourier domain to effectively handle various neighbourhood sizes. Traditionally, they firstly compute the singular value decomposition (SVD) of the graph Laplacian $L = U\Lambda U^T$, where $L = D - A$ is the Laplacian matrix of graph A. Then, by applying the spectral filter to the singular values $Ug_\theta(\Lambda)U^T$, the spectral convolution is performed to the graph signal x as $g_\theta * x = Ug_\theta(\Lambda)U^T x$. Unfortunately, the huge computational complexity of SVD hinders their applications to the large scale networks. Therefore, Hammond et al. approximates $g_\theta(\Lambda)$ by a truncated expansion of Chebyshev polynomials [14]. Kipf and Welling further approximate it with the first Chebyshev polynomials [23]. This simplification, named Graph Convolutional Network (GCN), reduces the model complexity to be linearly proportional to the number of edges and outperforms many state-of-the-art techniques. Although its core mechanism is based on the spectral convolution, GCN can also be interpreted from the perspective of spatial convolution. Recent work indicates that GCN is equivalent to a Laplacian smoothing operation to all the neighbours [24].

Although many GCNNs algorithms have been proposed and some of them achieve superior performances [25], [26], [27], [28], [29], they are designed from either the spatial or spectral perspective, yet without a *general* theoretical basis. The lack of the theoretical basis hinders both the analysis of the existing methods and development of the novel methods. Besides, most of the existing GCNNs methods tend to ignore the ubiquitous noises in the network topology and node content, and thus unable to model these uncertainties. These drawbacks obviously reduce their effectiveness in integrating network topology and node content.

To provide a probabilistic perspective to the GCNNs methods, in this paper, we formulate the basic graph convolution operation by a topology-constrained latent space model, Probabilistic Graph Convolutional Network (PGCN), which serves as a general framework for the GCNNs. By representing the nodes and edges in the networks in a more efficient distribution form as shown in Figure 1, their uncertainties can also be formulated. When specifying the distribution in (PGCN) as Gaussian distribution, it is equivalent to applying the general framework to the specific type of problems, and a specific method, named Probabilistic Graph Convolutional Network with Gaussian distribution representation (PGCN-G), can be constructed. To overcome the overfitting problem in covariance estimation and reduce the computational complexity, PGCN-G is further improved to PGCN-G+ by imposing the covariance matrices of all the vertices to possess the identical singular vectors. The proposed formulation can be optimized by iteratively performing the Expectation Maximization (EM) algorithm and solving a logistic regression problem. Specifically, the optimization algorithm iteratively denoises the network topology and the node content with respect to each other. Besides of the effectiveness of this *top-down* framework, it can also be deduced to cover the existing methods, GCN, GAT and GMM and elaborate their characteristics and relationships, by specific derivations and comparing PGCN-G+'s iterative algorithms with the training processes of GNNs.

The contributions are summarized as follows:

- To simultaneously denoise the network topology and node content, we formulate GCNNs as a general latent space framework, Probabilistic Graph Convolutional Network (PGCN), and represent the node content and edge weight as parametric distributions for semi-supervised node classification.
- By assuming that node content obeys Gaussian distribution, we propose Probabilistic Graph Convolutional Network with Gaussian distribution representation (PGCN-G), which applies the general PGCN framework to the specific Gaussian distribution based problems. To overcome the overfitting in parameter estimation, we extend PGCN-G to its enhanced version PGCN-G+ by constraining the covariances of all the nodes to possess identical the same singular vectors.
- We analyze three existing spatial/spectral based methods, GCN, GAT and GMM, to prove that these methods can be derived from our proposed framework, and thus conclude that they are the special cases of the proposed PGCN-G+. Besides, our proposed framework provides a tool to analyze the characteristics and relationships of the existing methods.

II. Notations

A network can be represented by an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. $\mathcal{V} = \{v_i | i = 1, ..., N\}$ is a set of $|\mathcal{V}| = N$ vertices, where v_n is associated with a feature $\mathbf{x}_n \in \mathbb{R}^K$. $\mathbf{X} \in \mathbb{R}^{N \times K}$ is the collection of the features, each row of which corresponds to one node. $\mathcal{E}$ stands for a set of edges, each of which connects two vertices in $\mathcal{V}$. The adjacency matrix $\mathbb{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ is employed to represent the network topology, where $a_{ij} = 1$ if an edge exists between the vertices v_i and v_j, and vice versa. If the network is allowed to contain self-edges, then $a_{nn} = 1$, otherwise $a_{nn} = 0$. $\mathbf{a}_n$, which denotes the n^{th}

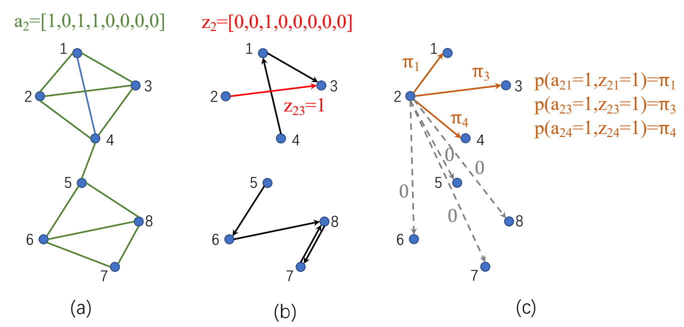


Fig. 2. The example of joint distribution $p(\mathbf{a}_n, \mathbf{z}_n)$. (a) Topology and $\mathbf{a}_2$. (b) An example of $\mathbf{z}_2$ where $z_{23} = 1$ represents v_3 is the most similar node to v_2. (c) The joint distribution of a_{2k} and z_{2k}.

column of $\mathbf{A}$, can be utilized to represent the neighbourhood of vertex v_n. $d_n = \sum_j a_{nj}$ is the degree of vertex v_n, and $\mathbf{D} = \mathrm{diag}(d_1, d_2, ..., d_N)$ is the degree matrix of the adjacency matrix $\mathbf{A}$. The graph Laplacian and its normalized form are defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, respectively.

For semi-supervised node classification task, we consider both the transductive and inductive learning scenarios.

Definition 1. For a network, given the labels $\mathbf{Y} = [y_{nf}] \in \mathbb{R}^{|\mathcal{V}_l| \times F}$ of a set of vertices $\mathcal{V}_l \subset \mathcal{V}$, where F is the number of classes, **transductive semi-supervised node classification** classifies other nodes $\mathcal{V} - \mathcal{V}_l$ according to the attributed graph $\mathcal{G}$ and given labels $\mathbf{Y}$.

Definition 2. Inductive semi-supervised node classification learns the model from a set of L attributed graphs $\{\mathcal{G}_i\}_{i=1}^{L}$ with labels, and classifies all the nodes of M unseen attributed graphs $\{\mathcal{G}_i\}_{i=L+1}^{L+M}$. All the attributed graphs $\{\mathcal{G}_i\}_{i=1}^{L+M}$ share the same set of attributes.

III. Probabilistic Graph Convolutional Network

In this section, we propose a general framework, Probabilistic Graph Convolutional Network (PGCN), which models the interactions between the network topology and node content. Then, a specific type of problems, where the distribution of each node can be modeled by a multivariate Gaussian distribution, is introduced.

A. The General Framework PGCN

Most existing work on semi-supervised node classification are discriminative algorithms which model the dependence of unknown variables with respect to the observed variables such as Logistic regression and Support Vector Machine. On the other hand, generative algorithms, which construct statistical models of the joint probability distribution of the unknown and observed variables, formulate the generation process of data, such as Latent Dirichlet Allocation (LDA) [30], stochastic block model (SBM) [31] and Generative adversarial networks (GANs) [32]. As shown in [33], generative algorithms usually outperform discriminative ones when the labelled data is limited. Therefore, in this paper, we pioneer to model the semi-supervised node classification as a generative model. In this model, variables are represented via distributions instead of vectors to characterize its properties and uncertainties.

Different from the existing generative models, which consider the node labels as latent variables and then model the conditional distribution of observed data based on the node labels, we consider the binary labels of directed edges as the latent variables and then model the conditional distribution of observed data based on the edge labels. For undirected graph, each undirected edge can be decomposed into two directed edges. If the label of a directed edge is one, the target node tends to be most similar node to the source node, among all the nodes. Specifically, for each vertex v_n, its observation is $\{\mathbf{a}_n, \mathbf{x}_n\}$, where $\mathbf{x}_n$ and $\mathbf{a}_n$ represent the content and topology information, respectively. To represent the similarity, a N-dimensional binary random variable (indicator) $\mathbf{z}_n \in \{0, 1\}^N$ is introduced, where $\sum_k z_{nk} = 1$. $z_{nk} = 1$ if and only if the nodes v_n and v_k are connected ($a_{nk} = 1$) and v_k is the most similar node to v_n among all the nodes. $z_{nn} = 1$ represents that the most similar node to nodes v_n is not is its neighbours. To model the prior, a variable π_k, which describes the node popularity and is normalized to $0 \le \pi_k \le 1$ with $\sum_k \pi_k = 1$, is assigned to each node. An example is shown in Figure 2(b), where v_3 is the most similar node to v_2. Assume the prior probability, which describes the possibility of node v_k being the most similar node to v_n if they are connected, is proportional to the popularity π_k of node v_k. Thus, the joint distribution over z_{nk} and a_{nk} is specified as

$$\begin{aligned} p(a_{nk} = 1, z_{nk} = 1) &= p(z_{nk} = 1 | a_{nk} = 1) p(a_{nk} = 1) \\ &= \pi_k \times 1 = \pi_k^{a_{nk} z_{nk}}. \end{aligned}$$

An example is shown in Figure 2(c). Similar to many existing generative models including SBM, our framework simplifies the construction of the edges to be independent of each other. Then, it can be reformed to

$$p(\mathbf{a}_n, \mathbf{z}_n) = \prod_k^N \pi_k^{a_{nk} z_{nk}}. \tag{1}$$

To facilitate the modeling of the node content, we represent the content of node v_n as a distribution $p(\mathbf{x}|\boldsymbol{\theta}_n)$ with a parameter $\boldsymbol{\theta}_n$, as shown in Figure 1. Note that the specific distribution will be discussed in the next sections. $p(\mathbf{x}_n|\boldsymbol{\theta}_n)$ denotes the likelihood of node v_n possessing the content $\mathbf{x}_n$, and $p(\mathbf{x}_n|\boldsymbol{\theta}_k)$ represents the content similarity between the nodes v_n and v_k. If nodes v_n and v_k are connected and belong to the same class, $p(\mathbf{x}_n|\boldsymbol{\theta}_k)$ should be large. If node v_n and none of its neighbours belong to the same class, $p(\mathbf{x}_n|\boldsymbol{\theta}_n)$ should be larger than any $p(\mathbf{x}_n|\boldsymbol{\theta}_k)$. Then, the likelihood of the content $\mathbf{x}_n$ is $p(\mathbf{x}_n|a_{nk} = 1, z_{nk} = 1) = p(\mathbf{x}_n|\boldsymbol{\theta}_k)$. Therefore, given the network topology $\mathbf{a}_n$ and indicator $\mathbf{z}_n$, the conditional distribution of $\mathbf{x}_n$ is

$$p(\mathbf{x}_n|\mathbf{a}_n, \mathbf{z}_n) = \prod_k^N p(\mathbf{x}_n|\boldsymbol{\theta}_k)^{a_{nk} z_{nk}}. \tag{2}$$

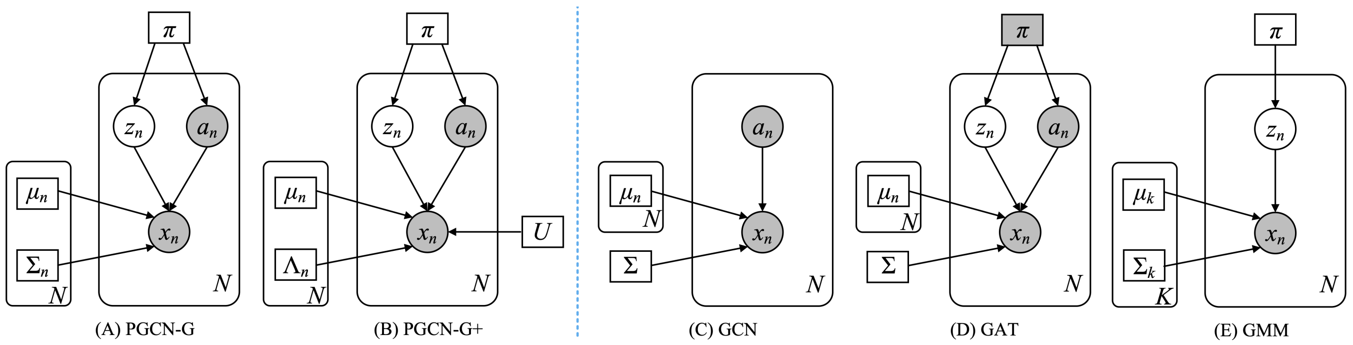


Fig. 3. Graphical representations of our proposed PGCN-G and PGCN-G+, along with PGCN-G+'s special cases GCN (one layer), GAT (one layer) and GMM, where the rectangles and circles denote parameters and random variables, respectively. The unshaded nodes represent the latent variables/parameters to be learned, while the shaded nodes represent the observed variables and fixed parameters.

Given the joint distribution as

$$\begin{aligned} p(\mathbf{a}_n, \mathbf{x}_n, \mathbf{z}_n) &= p(\mathbf{a}_n, \mathbf{z}_n) p(\mathbf{x}_n|\mathbf{a}_n, \mathbf{z}_n) \\ &= \prod_k^N (\pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k))^{a_{nk} z_{nk}}, \end{aligned} \tag{3}$$

the marginal distribution of the observation $\{\mathbf{a}_n, \mathbf{x}_n\}$ can be computed by summing the joint distribution over all the $\mathbf{z}_n$ as

$$\begin{aligned} p(\mathbf{a}_n, \mathbf{x}_n) = \sum_{\mathbf{z}_n} p(\mathbf{a}_n, \mathbf{x}_n, \mathbf{z}_n) &= \sum_{\mathbf{z}_n} \prod_k \{\pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k)\}^{a_{nk} z_{nk}}, \\ &= \sum_{k \in N(n) \cup \{n\}} \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k), \end{aligned} \tag{4}$$

where $N(n)$ denotes the neighbourhood of vertex v_n. For the attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with the adjacency matrix $\mathbf{A}$, the likelihood function (joint distribution) can be specified as

$$p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\Theta}) = \prod_{n=1}^N \sum_{k \in N(n) \cup \{n\}} \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k), \tag{5}$$

where $\boldsymbol{\pi} = \{\pi_1, ..., \pi_N\}$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N\}$ are the parameters, and the logarithm of the likelihood function is

$$\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\Theta}) = \sum_{n=1}^N \log \left\{ \sum_{k \in N(n) \cup \{n\}} \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k) \right\}. \tag{6}$$

Unfortunately, direct optimization of this (log) likelihood function $\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\Theta})$ is difficult, because the logarithm function outside the summation function will require a very large amount of computations. Instead, optimizing the complete data likelihood function $p(\mathbf{A}, \mathbf{X}, \mathbf{Z}|\pi, \boldsymbol{\Theta})$ is much easier. Here, the expectation maximization (EM) algorithm [34] is exploited to find the maximum likelihood solutions for the models with latent variables $\mathbf{Z}$ by iteratively constructing a lower-bound for $\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\Theta})$ (E-step) and then optimizing that (M-step). In E-step, the posterior distribution of the latent variables can be computed with respect to the current parameters $\{\boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}\}$ as

$$\begin{aligned} \gamma(z_{nk}) &= p(z_{nk} = 1|\mathbf{A}, \mathbf{X}, \boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}) \\ &= \frac{\pi_k^{old} p(\mathbf{x}_n|\boldsymbol{\theta}_k^{old})}{\sum_{j \in N(n) \cup \{n\}} \pi_j^{old} p(\mathbf{x}_n|\boldsymbol{\theta}_j^{old})}, \end{aligned} \tag{7}$$

which is one of the edges illustrated via different colors in Figure 1. Then, the expectation of the complete-data log likelihood function is calculated with estimated $p(z_{nk} = 1|\mathbf{A}, \mathbf{X}, \boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old})$ as

$$\begin{aligned} &\mathcal{Q}(\{\boldsymbol{\pi}, \boldsymbol{\Theta}\}, \{\boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}\}) \\ &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{A}, \mathbf{X}, \boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}) \log p(\mathbf{A}, \mathbf{X}, \mathbf{Z}|\boldsymbol{\pi}, \boldsymbol{\Theta}) \\ &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{A}, \mathbf{X}, \boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}) \sum_{n=1}^N \sum_{k \in N(n) \cup \{n\}} \log \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k). \end{aligned} \tag{8}$$

where the definition of $p(\mathbf{a}_n, \mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\pi}, \boldsymbol{\Theta})$ is shown in Eq. (3). In M-step, the new parameters $\{\boldsymbol{\pi}, \boldsymbol{\Theta}\}$ are determined by maximizing $\mathcal{Q}(\{\boldsymbol{\pi}, \boldsymbol{\Theta}\}, \{\boldsymbol{\pi}^{old}, \boldsymbol{\Theta}^{old}\})$ via

$$\{\boldsymbol{\pi}^{new}, \boldsymbol{\Theta}^{new}\} = \text{argmax}_{\{\boldsymbol{\pi}, \boldsymbol{\Theta}\}} \mathcal{Q}(\{\boldsymbol{\pi}, \boldsymbol{\Theta}\}, \{\boldsymbol{\pi}^{\text{old}}, \boldsymbol{\Theta}^{\text{old}}\}). \tag{9}$$

B. PGCN-G

After the general framework is introduced, it is applied to a specific type of problems where the distribution of each node v_n can be modeled by a multivariate Gaussian distribution as

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\theta}_n) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \\ &\propto \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_n)^T \boldsymbol{\Sigma}_n^{-1} (\mathbf{x} - \boldsymbol{\mu}_n) \right\}, \end{aligned} \tag{10}$$

where $\boldsymbol{\mu}_n$ is a K-dimensional mean vector and $\boldsymbol{\Sigma}_n$ is a $K \times K$ covariance matrix. This model is named as Probabilistic Graph Convolutional Network with Gaussian distribution representation (PGCN-G). Its graphical representation is shown in Figure 3(A). According to the EM algorithm in Eqs. (7) and (9), the following updating rules can be obtained:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j \in N(n) \cup \{n\}} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{11}$$

$$\boldsymbol{\mu}_n = \frac{1}{N_n} \sum_{k \in N(n) \cup \{n\}} \gamma(z_{nk}) \mathbf{x}_k, \tag{12}$$

$$\boldsymbol{\Sigma}_n = \frac{1}{N_n} \sum_{k \in N(n) \cup \{n\}} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \tag{13}$$

where $N_n = \sum_{k \in N(n) \cup \{n\}} \gamma(z_{nk})$ and $\pi_n = N_n/N$. For convenience, we omit the superscript $.^{old}$ in the parameters.

IV. PGCN for Semi-supervised Node Classification

In this section, the proposed Probabilistic Graph Convolutional Network (PGCN) is applied to the (transductive) semi-supervised node classification problem. Then, an illustrative explanation of its mechanism and validity is given. At last, the applicability of the general framework to the inductive learning task is demonstrated.

A. PGCN-G+

In Eq. (13), the number of variables in $\mathbf{\Sigma}_n$ is $K \times K$, which is much larger than that in $\boldsymbol{\mu}_n$. Meanwhile, the number of neighbours of the vertex v_n, i.e., its degree d_n, is very small for most of the vertices, because it obeys the power law distribution [10]. Therefore, the estimation of $\mathbf{\Sigma}_n$ in Eq. (13) may be insufficient. In this section, the problem of updating $\mathbf{\Sigma}_n$ is alleviated by enhancing the model and leveraging the label information.

Since the covariance matrix $\mathbf{\Sigma}_n$ is symmetric and positive-definite, without loss of generality, SVD can be expressed in the form of $\mathbf{\Sigma}_n = \mathbf{U}_n^T\mathbf{\Lambda}_n\mathbf{U}_n$, where $\mathbf{\Lambda}_n = \mathrm{diag}(\lambda_{n1}, ..., \lambda_{nK})$ is the singular value matrix and $\mathbf{U}_n \in \mathbb{R}^{K\times K}$ is the collection of the singular vectors. The SVD of $\mathbf{\Sigma}_n^{-1}$ is $\mathbf{\Sigma}_n^{-1} = \mathbf{U}_n^T\mathbf{\Lambda}_n^{-1}\mathbf{U}_n$ with $\mathbf{\Lambda}_n^{-1} = \mathrm{diag}(\lambda_{n1}^{-1}, ..., \lambda_{nK}^{-1})$. Then, the multivariate Gaussian distribution in Eq. (10) can be rewritten as

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \mathbf{\Lambda}_n) \propto \exp\left\{-\frac{1}{2}[\mathbf{U}_n(\mathbf{x}-\boldsymbol{\mu}_n)]^T\mathbf{\Lambda}_n^{-1}[\mathbf{U}_n(\mathbf{x}-\boldsymbol{\mu}_n)]\right\},$$

where $\mathbf{f} = \mathbf{U}_n(\mathbf{x}-\boldsymbol{\mu}_n)$ can be regarded as a new coordinate system defined by the singular vectors $\mathbf{U}_n$, as shown in Figure 4. Motivated by the previous distribution-based network embedding schemes, where the covariance matrix of Gaussian distribution is assumed to be diagonal [35], [36], we relax the covariance matrix for each node to only possess identical singular vectors,

$$\mathbf{\Sigma}_n^{-1} = \mathbf{U}^T\mathbf{\Lambda}_n^{-1}\mathbf{U},$$

and the nodes can be represented by

$$\begin{aligned}&\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \mathbf{U}^T\mathbf{\Lambda}_n\mathbf{U})\\ &\propto \exp\left\{-\frac{1}{2}[(\mathbf{U}\mathbf{x}-\mathbf{U}\boldsymbol{\mu}_n)]^T\mathbf{\Lambda}_n^{-1}[(\mathbf{U}\mathbf{x}-\mathbf{U}\boldsymbol{\mu}_n)]\right\}.\end{aligned} \tag{14}$$

This enhanced model is named as PGCN-G+, and its graphical representation is shown in Figure 3(B). Then, the logarithm of the likelihood function in Eq. (6) can be computed as

$$\begin{aligned}&\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Lambda}, \mathbf{U})\\ &= \sum_{n=1}^{N}\log\left\{\sum_{k\in N(n)\cup\{n\}}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{U}^T\mathbf{\Lambda}_k\mathbf{U})\right\},\end{aligned} \tag{15}$$

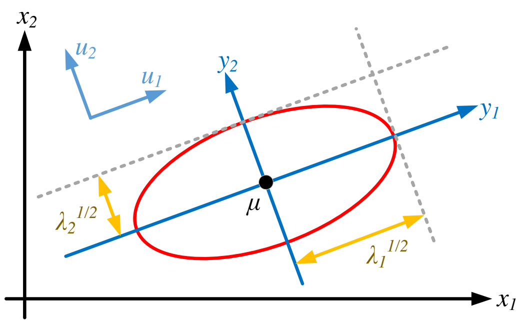


Fig. 4. The new coordinate system defined by the singular vectors $\mathbf{U}$ (Refer to Figure 2.7 in [37]).

and the expectation of the complete-data log likelihood function in Eq. (8) can be calculated as

$$\begin{aligned}&\mathcal{Q}(\{\boldsymbol{\pi}, \mathbf{\Theta}\}, \{\boldsymbol{\pi}^{old}, \boldsymbol{\mu}^{old}, \mathbf{\Lambda}^{old}, \mathbf{U}^{old}\})\\ &= \sum_{\mathbf{Z}}[p(\mathbf{Z}|\boldsymbol{\pi}^{old}, \boldsymbol{\mu}^{old}, \mathbf{\Lambda}^{old}, \mathbf{U}^{old})\times\\ &\qquad \sum_{n=1}^{N}\sum_{k\in N(n)\cup\{n\}}\log\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{U}^T\mathbf{\Lambda}_k\mathbf{U})].\end{aligned} \tag{16}$$

To iteratively optimize the problem, three variables, $\mathbf{U}$, $\mathbf{\Lambda}_n = \mathrm{diag}(\lambda_{n1}, ..., \lambda_{nK})$ and $\mathbf{U}\boldsymbol{\mu}_n$, are required to be separately updated. Here, $\mathbf{U}\boldsymbol{\mu}_n$ is directly updated instead of updating $\boldsymbol{\mu}_n$ followed by multiplication with $\mathbf{U}$ for computation and implementation considerations. $\mathbf{\Lambda}_n$ is updated by

$$\lambda_{nk} = \frac{1}{N_n}\sum_{j\in N(n)\cup\{n\}}\gamma(z_{nk})[\mathbf{u}_{k*}(\mathbf{x}_n-\boldsymbol{\mu}_j)]^2, \tag{17}$$

where $\mathbf{u}_{k*}$ denotes the k^{th} row of $\mathbf{U}$. Compared to the updating rule of $\mathbf{\Sigma}_n$ in Eq. (13), updating $\mathbf{\Lambda}_n$ in Eq. (17) reduces the number of parameters from K^2 to K, which prevents the overfitting problem. After updating $\mathbf{\Lambda}_n$, the corresponding $\mathbf{U}\boldsymbol{\mu}_n$ and $\gamma(z_{nk})$ are updated as

$$\gamma(z_{nk}) = \frac{\pi_k|\mathbf{\Lambda}_k|^{-1/2}\exp\left\{-\frac{1}{2}\mathbf{f}_{nk}^T\mathbf{\Lambda}_k^{-1}\mathbf{f}_{nk}\right\}}{\sum_{j\in N(n)\cup\{n\}}\pi_j|\mathbf{\Lambda}_j|^{-1/2}\exp\left\{-\frac{1}{2}\mathbf{f}_{nj}^T\mathbf{\Lambda}_j^{-1}\mathbf{f}_{nj}\right\}}, \tag{18}$$

$$\mathbf{U}\boldsymbol{\mu}_n = \frac{1}{N_n}\sum_{k\in N(n)\cup\{n\}}\gamma(z_{nk})\mathbf{U}\mathbf{x}_k, \tag{19}$$

where

$$\mathbf{f}_{nk} = \mathbf{U}(\mathbf{x}_n-\boldsymbol{\mu}_k) = \mathbf{U}\mathbf{x}_n-\mathbf{U}\boldsymbol{\mu}_k. \tag{20}$$

Eq. (19) can be reformed to

$$\mathbf{T}^{\text{PGCN-G+}} = \mathbf{OXW} \tag{21}$$

where $\mathbf{O} = [o_{nk}] \in \mathbb{R}^{N\times N}$ with $o_{nk} = \gamma(z_{nk})/N_n$, $\mathbf{X} \in \mathbb{R}^{N\times K}$ with n^{th} row being $\mathbf{x}_n^T$ and $\mathbf{W} = \mathbf{U}^T$.

Unfortunately, the matrix $\mathbf{U}$, which contains the singular vectors, cannot be directly obtained by setting the derivative of Eq. (16) with respect $\mathbf{U}$ equal to zero. Thus, we leverage the label information to effectively estimate $\mathbf{U}$. Recall that the mean $\boldsymbol{\mu}_n$ of $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \mathbf{\Lambda}_n)$ represents most of the information of vertex v_n, and the projection $\mathbf{U}\boldsymbol{\mu}_n$ represents the latent semantic of vertex v_n. Therefore, we intend to connect the projection $\mathbf{U}\boldsymbol{\mu}_n$ with the label of vertex v_n, i.e., the n^{th} row of $\mathbf{Y}$. By adopting $\mathbf{W}$ into $\mathbf{U}$ instead of learning a projection $\mathbf{W}$ from $\mathbf{U}$, we constrain $\mathbf{U} \in \mathbb{R}^{K\times F}$ and compute the cross-entropy over all the labelled nodes.

$$\mathcal{L} = -\sum_{n\in\mathcal{V}_l}\sum_{f=1}^{F} y_{nf}\log(t_{nf}) = -\sum_{n\in\mathcal{V}_l}\sum_{f=1}^{F} y_{nf}\log(\mathbf{u}_{f*}\boldsymbol{\mu}_n), \tag{22}$$

where $\mathcal{V}_l$ represents the set of labelled nodes, F stands for the number of classes and $\mathbf{u}_{f*}$ denotes the f^{th} row of $\mathbf{U}$. Then, $\mathbf{U}$ can be estimated by minimizing $\mathcal{L}$. Besides, the dimensions of $\boldsymbol{\Lambda}_n$ should also be adjusted to $F \times F$ according to the dimensions of $\mathbf{U}$.

B. Objective Function and Optimization

The overall objective function of PGCN-G+ is the combination of the following two objective functions

$$\mathbf{U}^* = \arg\min_{\mathbf{U}} -\sum_{n\in\mathcal{V}_l}\sum_{f=1}^{F} y_{nf}\log(\mathbf{u}_{f*}\boldsymbol{\mu}_n^*), \tag{23}$$

$$\begin{aligned}\left\{\boldsymbol{\pi}^*, \{\boldsymbol{\mu}_n^*, \boldsymbol{\Lambda}_N^*\}_{n=1}^N\right\} &= \arg\max_{\boldsymbol{\pi},\{\boldsymbol{\mu}_n,\boldsymbol{\Lambda}_N\}_{n=1}^N} \log p(\mathbf{A},\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda},\mathbf{U}^*)\\ &= \arg\max \sum_{n=1}^{N}\log\left\{\sum_{k\in N(n)\cup\{n\}} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{U}^{*T}\boldsymbol{\Lambda}_k\mathbf{U}^*)\right\}.\end{aligned} \tag{24}$$

In Eq. (23), $\boldsymbol{\mu}_n^*$'s are fixed and the optimal $\mathbf{U}^*$ can be obtained by minimizing the cross-entropy loss function with batch gradient descent algorithm. In Eq. (24), $\mathbf{U}^*$ is fixed and the optimal $\left\{\boldsymbol{\pi}^*, \{\boldsymbol{\mu}_n^*\}_{n=1}^N, \{\boldsymbol{\Lambda}_N^*\}_{n=1}^N\right\}$ are obtained via the Expectation Maximization (EM) algorithm. In the E-step, the posterior is computed via Eq. (18). In M-step, the model parameters $\boldsymbol{\mu}_n$'s, $\boldsymbol{\Lambda}_n$'s and π_n's are updated via Eqs. (19), (17) and $\pi_n = \sum_{k\in N(n)\cup\{n\}}\gamma(z_{nk})/N$, respectively. The overall optimal solution is obtained by iteratively optimizing Eqs. (23) and (24). Note that the EM algorithm in the inner loop guarantees that the results converge to a locally optimal. However, it is difficult to show that the outer loop guarantees that the results converge to a locally optimal, since our objective function is the combination of Eqs. (23) and (24). To speed-up the convergence, the inner loop is omitted as most of the alternating direction method of multipliers (ADMM) [38] approach, and the final PGCN-G+ algorithm for transductive semi-supervised node classification is shown in Algorithm 1.

There exists two reasons of separately learning $\mathbf{U}$ and other parameters. First, as shown in Eq. (14), the matrix $\mathbf{U}$, which contains the singular vectors, can also be considered as a mapping from the original node content $\mathbf{x}$ to its semantical version $\mathbf{Ux}$. Thus, estimating $\mathbf{U}$ by leveraging label information is much more effective and accurate than learning U via unsupervised schemes. Second, if $\mathbf{U}$ has been learned, the Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \mathbf{U}^T\boldsymbol{\Lambda}_n\mathbf{U})$ in Eq. (14) can then be reformed to $\mathcal{N}(\mathbf{Ux}|\mathbf{U}\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n)$, which considers $\mathbf{Ux}$ as a random variable. Then, we can estimate its mean $\mathbf{U}\boldsymbol{\mu}_n$ and diagonal covariance matrix $\boldsymbol{\Lambda}_n$ by maximizing Eq. (24) with EM, which guarantees that the results converge to a locally optimal.

Algorithm 1: PGCN-G+ for Transductive Learning (Speed-up)

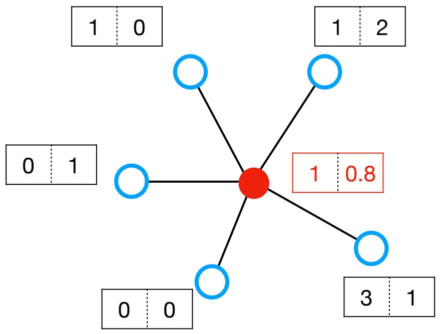
```
Input: Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, feature matrix
       $\mathbf{X} \in \mathbb{R}^{N\times K}$, node label matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}_l|\times F}$
       and the number of classes $F$.
Output: New adjacency matrix $\mathbf{O}$, mapping $\mathbf{U}$ and
        prediction $\mathbf{T}$.
1  Randomly initialize $\mathbf{U}$;
2  Initialize $\pi_n = d_n/\sum d_n$;
3  while not convergence do
4      %—— Exception-step ——
5      Update posterior $\gamma(z_{nk})$ via Eq. (18);
6      Update $\mathbf{O}$ with $o_{nk} = \gamma(z_{nk})/N_n$;
7      %—— Maximization-step ——
8      Update $\boldsymbol{\mu}_n$ via Eq. (19);
9      Update $\boldsymbol{\Lambda}_n$ via Eq. (17) ;
10     Update $\pi_n = \sum_{k\in N(n)\cup\{n\}}\gamma(z_{nk})/N$;
11     %—— Mapping-step ——
12     Update $\mathbf{U}$ by minimizing Eq. (22) via batch
         gradient decent;
13 end
14 $\mathbf{T}^{\text{PGCN-G+}} = \sigma(\mathbf{OXU})$;
15 return $\mathbf{O}, \mathbf{U}, \mathbf{T}$.
```

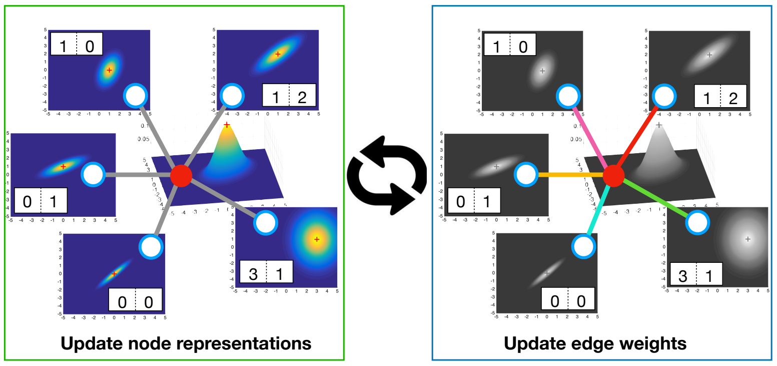
Remark: In Eq. (18), $\mathbf{f}_{nk}^T\boldsymbol{\Lambda}_k^{-1}\mathbf{f}_{nk}$ can be regarded as the weighted ℓ_2 norm of $\mathbf{f}_{nk}$. It assigns weights to various features in different nodes. Since different vertices tend to focus on different features, the learned $\boldsymbol{\Lambda}_k^{-1}$ is equivalent to the attentions obtained by the attention mechanism [39] in the node features. This is the main advantage to exploit the distribution-based node representation in PGCN-G+ compared to Graph Attention Network, which applies the same attention mechanism to all the nodes.

C. How and Why does PGCN-G+ Work?

PGCN-G+, as shown in Algorithm 1, is formulated as a latent space model and specifies the node content representations as Gaussian distributions. In this subsection, PGCN-G+ is compared to GCN [23] to provide an illustrative explanation of its mechanism and validity. As shown in Figure 5(a), GCN smoothes the observed node contents in the observed neighbourhoods [24] (where the details of GCN is given in Section V-A1). Unfortunately, both the network topology and node content may contain noises, thus the direct smoothing over observations may cause noise propagation. The PGCN framework represents node content and edge weights as distributions to model their uncertainties. Based on Expectation Maximization, Algorithm 1 iteratively updates the distribution-based node representations (parameterized by $\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$) and the edge weights $\gamma(z_{nk})$ shown in Figure 5(b).

The updates of distribution-based node representations are carried out based on the estimated edge weights, which





(a) GCN (b) PGCN-G+

Fig. 5. An illustrative explanation of the mechanism of our proposed PGCN-G+ work. (a) Graph Convolutional Network (GCN) [23] smoothes the node content over the neighbourhood based on the observed network topology and node content. (b) PGCN-G+ iteratively updates the distribution-based node representations ($\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$) and the edge weights ($\gamma(z_{nk})$).

represent the probabilities of the nodes in the same class being connected. As shown in Eq. (19), the mean $\boldsymbol{\mu}_n$ of node content distribution is the weighted average of its neighbours with the estimated edge weights $\gamma(z_{nk})$. Similarly, the variance λ_{nk}, which represents the confidence of the node content, is the weighted average of the squared differences between the node content $\mathbf{x}_n$ and the mean $\boldsymbol{\mu}_j$ of the neighbourhood distributions. Therefore, the updates of distribution-based node representations are equivalent to denoise the node contents according to the network topology.

Meanwhile, the updates of edge weights are performed according to the estimated node content distributions. As shown in Eq. (18), the edge weights are updated according to the weighted similarities between node content. The weights of the similarities are the variances of the node content distributions, which represent the confidences (certainties) of the node content. Therefore, the updates of edge weights is equivalent to denoise the network topology with respect to the node content.

In summary, our proposed PGCN-G+ iteratively denoises the node content and the network topology.

D. PGCN-G+ for Inductive Learning

Inductive semi-supervised node classification task is to apply the learned model to unseen graphs which possess the same node attributes yet without any labels. In Algorithm 1, which is the transductive version of the proposed PGCN-G+, only the projection parameter $\mathbf{U}$ is directly determined by the label $\mathbf{Y}$, while the other parameters are only affected by the adjacency matrix $\mathbf{A}$, feature matrix $\mathbf{X}$ and learned projection $\mathbf{U}$. Therefore, to apply the learned PGCN-G+ to unseen graphs, we fixed projection $\mathbf{U}$ and iteratively update $\mathbf{O}$, $\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$ as shown in Algorithm 2.

V. Relationship with Existing Methods

In this section, we analyze three existing models, GCN, GAT and GMM, and demonstrate that these models are the special cases of the proposed PGCN-G+ approach. Then, we compare the proposed PGCN framework with the existing MoNet [21] to better differentiate them.

Algorithm 2: PGCN-G+ for Inductive Learning

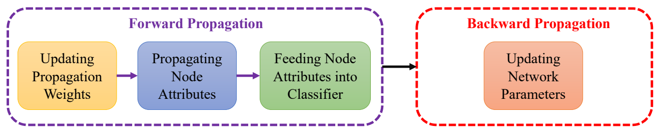
```
Input: Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, feature matrix
       $\mathbf{X} \in \mathbb{R}^{N\times K}$, node label matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}_l|\times F}$,
       the number of classes $F$ and learned projection
       $\mathbf{U}$.
Output: New adjacency matrix $\mathbf{O}$ and prediction $\mathbf{T}$.
1  Initialize $\pi_n = d_n / \sum d_n$;
2  while not convergence do
3      %—— Exception-step ——
4      Update posterior $\gamma(z_{nk})$ via Eq. (18);
5      Update $\mathbf{O}$ with $o_{nk} = \gamma(z_{nk})/N_n$;
6      %—— Maximization-step ——
7      Update $\boldsymbol{\mu}_n$ via Eq. (19);
8      Update $\boldsymbol{\Lambda}_n$ via Eq. (17) ;
9      Update $\pi_n = \sum_{k\in N(n)\cup\{n\}} \gamma(z_{nk})/N$;
10 end
11 $\mathbf{T}^{\text{PGCN-G+}} = \sigma(\mathbf{OXU})$;
12 return $\mathbf{O}, \mathbf{T}$.
```

A. Graph Neural Networks

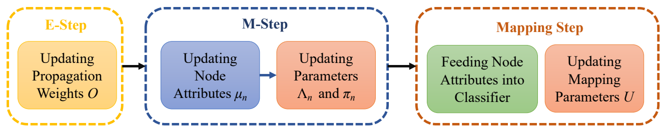
In this subsection, we first interpret the correspondences between the training processes of graph neural networks (GNNs) and our proposed PGCN-G+ as shown in Fig. 6.

On one hand, the training process of GNNs alternately perform the forward and backward propagations as shown in Fig. 6(a), which is similar to that of the deep neural networks on grid data, such as fully-connected neural network and convolutional neural networks (CNNs) [13]. In the forward propagation of GNN, the loss is computed in two steps. First, the node attributes are augmented by propagating them in the local neighbourhoods with specific weights [40]. Second, the augmented node attributes are fed into the classifier to compute the loss. In the backward propagation of GNN, the parameters of the neural networks are updated by back-propagating the gradients of the loss.

On the other hand, the alternating process of our proposed PGCN-G+, as shown in Algorithms 1 and 2, consists of three components, i.e., Exception-step (E-step), Maximization-


Forward Propagation
Updating Propagation Weights
Propagating Node Attributes
Feeding Node Attributes into Classifier
Backward Propagation
Updating Network Parameters

(a) Training Processes of Graph Neural Networks


E-Step
Updating Propagation Weights O
M-Step
Updating Node Attributes μ_n
Updating Parameters Λ_n and π_n
Mapping Step
Feeding Node Attributes into Classifier
Updating Mapping Parameters U

(b) Training Processes of PGCN-G+

Fig. 6. Correspondences between the training processes of graph neural networks (GNNs) and our proposed PGCN-G+. Boxes with the same color are possessing similar functionalities.

step (M-step) and Mapping-step, as shown in Fig. 6(b). The Exception-step computes the weights for the propagations. Updating $\boldsymbol{\mu}_n$ in Maximization-step propagates the node attributes in the local neighbourhoods. Thus, both of the two steps correspond to the forward propagations in GNNs. The Mapping-step (updating $\mathbf{U}$) and the operations (updating $\boldsymbol{\Lambda}_n$ and π_n) in Maximization-step will update the model parameters, which corresponds to the backward propagations in GNNs. In Mapping-step, $\mathbf{U}$ is updated by minimizing the cross-entropy loss. In Maximization-step, $\boldsymbol{\Lambda}_n$ and π_n are updated based on the obtained $\mathbf{U}$ and $\boldsymbol{\mu}_n$. Therefore, the alternating steps of our proposed PGCN-G+ correspond to the training process of graph neural networks.

According to these correspondences, we will demonstrate that two milestone GNNs, i.e., GCN and GAT, are the special cases of the proposed PGCN-G+ approach, as below.

1) Graph Convolutional Network (GCN): Graph convolution operation is the generalization of the convolution operation, which is applied to the regular grid such as image, to irregular graph. GCN [23] simplifies many previous models which possess high complexities, and defines the graph convolution operation with a parameter $\mathbf{w}$ on signal $\mathbf{s} \in \mathbb{R}^N$ as

$$g_w * \mathbf{s} = \mathbf{w}(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{s},$$

where $\mathbf{I}_N$ is the identity matrix with size N. Then, we renormalize $\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ to $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$,where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ and $\tilde{D}_{nn} = \sum_j \tilde{A}_{nj} = d_n + 1$. The normalized formula can be generalized from 1-dimensional signal $\mathbf{s} \in \mathbb{R}^{N\times 1}$ and one filter $\mathbf{w} \in \mathbb{R}^{K\times 1}$ to a K-channels signal $\mathbf{X} \in \mathbb{R}^{N\times K}$ and F filters $\mathbf{W} \in \mathbb{R}^{K\times F}$, each of which is for one class. Then, the graph convolution operation can be extended to

$$\mathbf{T}^{GCN} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}, \tag{25}$$

where $\mathbf{T}^{GCN} = [t_{nf}]$ is the convolved signal matrix. Li et al. conclude the mechanism and success of GCN that it actually performs a symmetric Laplacian smoothing ($\mathbf{H}_{GCN} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$) operation followed by a projection $\mathbf{T}^{GCN} = \mathbf{H}_{GCN}\mathbf{W}$ [24]. This mechanism can be implemented by constructing two layers, a smoothing layer and a fully-connected layer, in a neural network. The symmetric Laplacian smoothing operation serves as the key to GCN's performance improvement. On the other hand, asymmetric Laplacian smoothing operation

$$\mathbf{H} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}, \tag{26}$$

play a similar role. The parameter $\mathbf{W}$ is obtained by minimizing the cross-entropy between the given labels and the predictions

$$\mathcal{L} = -\sum_{n\in\mathcal{V}_l}\sum_{f=1}^{F} y_{nf}\log(t_{nf}). \tag{27}$$

Next, we will demonstrate that GCN is a special case of our proposed PGCN-G+.

Proposition 1. ***(Probabilistic Interpretation of GCN)*** *Regardless of the differences between asymmetric and symmetric Laplacian smoothings, the GCN model is equivalent to a special case of PGCN-G+ which omits the latent variables* $\mathbf{z}_n$ *and* π_n*, while fixes* $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}$ *for all the vertices and sets the graph to be a self-loop (*$a_{nn} = 1$ *for all* n*) as shown in Figure 3(C).*

Proof. Since the latent variables $\mathbf{z}_n$ and π_n are omitted, the marginal distribution of observation $\{\mathbf{A}, \mathbf{X}\}$ can be represented as

$$p(\mathbf{A}, \mathbf{X}) = \prod_n \prod_{k\in N(n)\cup\{n\}} \{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})\}^{a_{nk}}.$$

Different from Eq. (5), the only variables here are $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}$. Thus, they can be obtained by directly minimizing the logarithm of the likelihood function as

$$\boldsymbol{\mu}_n = \frac{1}{d_n + 1}\sum_{k\in N(n)\cup\{n\}} \mathbf{x}_k.$$

Then, it can be reformed to

$$\mathbf{H} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}, \tag{28}$$

where the n^{th} row of $\mathbf{H}$ and $\mathbf{X}$ is $\boldsymbol{\mu}_n$ and $\mathbf{x}_n$, respectively. Thus, computing the mean for each vertex is equivalent to the asymmetric Laplacian smoothing operation in GCN.

In PGCN-G+, $\mathbf{U}\boldsymbol{\mu}_n$ is employed to predict the labels, where $\boldsymbol{\Sigma}_n = \mathbf{U}^T\boldsymbol{\Lambda}_n\mathbf{U}$. Since $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}$ for all n, $\mathbf{Z} = \mathbf{H}\mathbf{U}^T = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}$ is exploited to predict the labels and $\mathbf{W} = \mathbf{U}^T$ is calculated by minimizing the cross-entropy between the labels and predictions. As can be concluded, GCN is equivalent to a special case of PGCN-G+. □

2) Graph Attention Networks (GAT): The success GCN is achieved by smoothing (averaging) the node content in a local neighbourhood. To consistently process the variable sized inputs and focus on the most serviceable parts of the inputs, the attention mechanism is introduced by GAT to handle the different sizes of the neighbourhoods. GAT replaces the smoothing layer in GCN with a graph attention layer. To obtain the nodes with more attentions, attention coefficients

$$o_{nk} = \frac{\exp\left(c(\mathbf{x}_n^T\mathbf{W}, \mathbf{x}_k^T\mathbf{W})\right)}{\sum_{k\in N(n)}\exp\left(c(\mathbf{x}_n^T\mathbf{W}, \mathbf{x}_k^T\mathbf{W})\right)}, \tag{29}$$

is computed to reveal the amount of attention obtained at node v_k from node v_n. $c(\mathbf{x}_n^T\mathbf{W}, \mathbf{x}_k^T\mathbf{W})$ represents a shared attention mechanism $c : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$. $N(n)$ denotes the neighbourhood of vertex v_n. GAT adopts the Leaky-ReLU nonlinearity mapping on $[\mathbf{x}_n^T\mathbf{W}||\mathbf{x}_k^T\mathbf{W}]\mathbf{b}$, where $[\mathbf{x}_n^T\mathbf{W}||\mathbf{x}_k^T\mathbf{W}]$ is the concatenation of $\mathbf{x}_n^T\mathbf{W}$ and $\mathbf{x}_k^T\mathbf{W}$, and $\mathbf{b} \in \mathbb{R}^{2F}$ stands for the shared parameters. $\mathbf{O} = [o_{nk}] \in \mathbb{R}^{N\times N}$ can be regarded as the re-weighting of the adjacency matrix $\mathbf{A} = [a_{nk}] \in \mathbb{R}^{N\times N}$. $o_{nk} \neq 0$ when $a_{nk} = 1$, i.e., the vertices v_n and v_k are connected. Then GAT can be represented as

$$\mathbf{T}^{GAT} = \mathbf{OXW}. \tag{30}$$

The parameter $\mathbf{W}$ can be computed by minimizing the cross-entropy between the given labels and predictions according to Eq. (27).

The following theorem illustrates the fact that GAT is also a special case of PGCN-G+.

Proposition 2. ***(Probabilistic Interpretation of GAT)*** *The GAT model with the attention mechanism $c(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2^2$ is equivalent to a special case of PGCN-G+, which fixes $\pi_n = \frac{1}{N}$ and $\mathbf{\Sigma}_n = \mathbf{\Sigma}$ for all the vertices and set the graph without any self-loops ($a_{nn} = 0$ for all n), as shown in Figure 3(D).*

Proof. Since $\pi_n = \frac{1}{N}$ and $\mathbf{\Sigma}_n = \mathbf{\Sigma}$, Eq. (18) can be reformed to

$$\begin{aligned} o_{nk} &= \frac{\exp\left\{(\mathbf{x}_n - \boldsymbol{\mu}_k)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)\right\}}{\sum_{j\in N(n)}\left\{(\mathbf{x}_n - \boldsymbol{\mu}_j)\mathbf{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j)^T\right\}} \\ &= \frac{\exp\left\{[\mathbf{W}(\mathbf{x}_n - \boldsymbol{\mu}_k)]^T[\mathbf{W}(\mathbf{x}_n - \boldsymbol{\mu}_k)]\right\}}{\sum_{j\in N(n)}\left\{[\mathbf{W}(\mathbf{x}_n - \boldsymbol{\mu}_j)][\mathbf{W}(\mathbf{x}_n - \boldsymbol{\mu}_j)]^T\right\}}, \end{aligned}$$

where $\mathbf{\Sigma}^{-1} = \mathbf{U}^T\mathbf{\Lambda}^{-1}\mathbf{U}$ and $\mathbf{W} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}$. Usually, $\boldsymbol{\mu}_k$ is initialized by $\mathbf{x}_k$, and $\gamma(z_{nk})$ becomes

$$o_{nk} = \gamma(z_{nk}) = \frac{\exp\left\{||\mathbf{W}(\mathbf{x}_n - \mathbf{x}_k)||_2^2\right\}}{\sum_{j\in N(n)}\left\{||\mathbf{W}(\mathbf{x}_n - \mathbf{x}_j)||_2^2\right\}}.$$

Then, the updating rule of $\boldsymbol{\mu}_n$ becomes

$$\boldsymbol{\mu}_n = \sum_{k\in N(n)} o_{nk}\mathbf{x}_k. \tag{31}$$

Eq. (31) is equivalent to the graph attention operation in Eq. (29) in GAT with $c(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2^2$ and can be reformed to the matrix form

$$\mathbf{H} = \mathbf{OX}, \tag{32}$$

where the n^{th} rows of $\mathbf{H}$ and $\mathbf{X}$ is $\boldsymbol{\mu}_n$ and $\mathbf{x}_n$, respectively, and $\mathbf{O} = [o_{nk}] \in \mathbb{R}^{N\times N}$ is the collection of posterior. Thus, computing the mean for each vertex is equivalent to the graph attention layer in GAT.

The equivalence between the projection $\mathbf{W}$ in GAT and the weighted singular vectors $\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}$ is similar to that in Theorem 1. Therefore, GAT is equivalent to a special case of PGCN-G+. □

Remark 1: According to the analysis of GCN and GAT, we can conclude that the vector-form node representation is essentially a special case of the distribution-form representation by assuming the covariances of all the nodes being the same. The weighted singular vectors $\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}$, where the weight $\mathbf{\Lambda}^{-\frac{1}{2}}$ is computed with respect to the singular value, can be regarded as the mapping W from the feature space to the semantic space (label space). Therefore, the learning of the mapping function (fully-connected network) is equivalent to the estimation of $\mathbf{U}$ in PGCN-G+.

Remark 2: The overall computational complexities of both GCN and GAT are $\mathcal{O}(MK + NFK)$, where N and M are the numbers of nodes and edges, respectively, and F and K are the dimensionalities of the input and output features of nodes, respectively. Note that $\mathcal{O}(NFK)$ and $\mathcal{O}(MK)$ operations are induced by node feature mappings and propagations, respectively. According to Theorems 1 and 2, the additional cost of our proposed PGCN-G+ in Algorithms 1 and 2 is induced by updating the covariances for all the nodes in Eq. (17), compared to GCN and GAT. Then, the additional complexity in each iteration is $\mathcal{O}(MK)$, which is the same as that of the propagations in GCN and GAT. Therefore, our proposed PGCN-G+ does not increase the computational complexity compared to existing GNNs.

B. Gaussian Mixture Models (GMM)

The Gaussian mixture model, which is represented by a graphical model in Figure 3(E), can be written as a linear combination of the Gaussian distributions in the form

$$p(\mathbf{x}) = \sum_{k=1}^{F}\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), \tag{33}$$

where π_k is the mixing coefficient and satisfies $\sum_{k=1}^{F}\pi_k = 1$, and F is the number of clusters to be determined in advance. GMM has been successfully applied to many real clustering problems including the background subtraction and the speaker identification etc. GMM can also be interpreted as a latent variable model. By introducing a F-dimensional latent binary variable $\mathbf{z}_n \in \{0,1\}^F$ with $\sum_{k=1}^{F} z_{nk} = 1$, the distribution of the observed data point $\mathbf{x}_n$ can be formulated via marginalizing the joint distribution

$$p(\mathbf{x}_n) = \sum_{\mathbf{z}_n} p(\mathbf{z}_n)p(\mathbf{x}_n|\mathbf{z}_n) = \sum_{k=1}^{F}\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), \tag{34}$$

where

$$p(\mathbf{z}_n) = \prod_{k=1}^{F}\pi_k^{z_{nk}}, p(\mathbf{x}_n|\mathbf{z}_n) \quad = \prod_{k=1}^{F}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)^{z_{nk}}.$$

Obviously, they are very similar to our proposed PGCN-G+ (Eqs. (1-4)). There exists constraints for the topological information a_{nk} in PGCN-G+ while there are none in GMM (Eq. (34)). Thus, GMM is a special case of PGCN-G+ with $a_{nk} = 1$ for all n and k, i.e., without the topological information constraint.

C. Comparison with MoNet

Mixture Model Network (MoNet) [21] is the most similar method to our proposed PGCN. It directly models the representation of vertex v_n as the weighted combination of its neighbours via

$$\hat{\mathbf{x}}_n = \sum_{k\in N(n)} w\big(u(v_n, v_k)\big)\mathbf{x}_k,$$

TABLE I
DATASETS.

Dataset	#Nodes	#Edges	#Classes	#Features
Texas	187	328	5	1,703
Cornell	195	304	5	1,703
Washington	230	446	5	1,703
Wisconsin	265	530	5	1,703
Wiki	3,363	45,006	19	4,972
CiteSeer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
PubMed	19,717	44,338	3	500
PPI	56,944	818,716	121	50

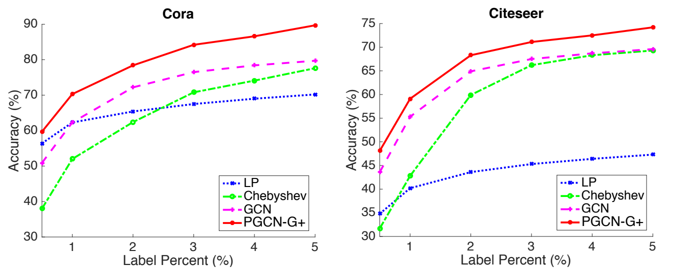


Fig. 7. The node classification results on three datasets with varying percentages of labelled nodes.

with the weights $w\big(u(v_n, v_k)\big)$ defined as

$$w\big(u(v_n, v_k)\big) = \exp\left\{-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)^T\right\},$$

where $u(v_n, v_k) = (\frac{1}{\sqrt{d_n}}, \frac{1}{\sqrt{d_k}})$, d_n denotes the degree of the vertex v_n, and $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ represent the learnable parameters. The differences between MoNet and our proposed PGCN are summarized as below.

- According to the detailed motivations of the two methods, the edge weights between the vertices v_n and v_k are determined by different kinds of information. MoNet assigns the weights with respect to only the topological information (e.g. degree), thus it can be regarded as an extension of GCN while remains differently compared to GAT. In PGCN, the neighbourhoods are determined by the network topology while the weights are computed according to the content of the vertices. Therefore, PGCN is more flexible and robust compared to MoNet.
- From the perspective of methodology, MoNet and PGCN are also different. MoNet is a heuristic *bottom-up* approach, which unifies the existing methods via one strategy and extends it with learnable parameters. PGCN is a *top-down* framework, which formulates the semi-supervised node classification task as a topology-constrained latent space model. This framework is essentially a more general framework which encompasses some existing methods as its special cases.

VI. EXPERIMENTAL RESULTS

A. Datasets

For the transductive learning task, the experiments are conducted on three commonly utilized citation networks, Cora, CiteSeer and PubMed, as shown in Table I. In each network, nodes and edges are research papers and undirected citations, respectively. In addition to the network structure, node content, which is represented by the bag-of-word representation of the documents, is available. According to the disciplines, papers are categorized into various classes. Besides, five more networks, including Cornell, Texas, Washington, Wisconsin and Wiki, are employed. Four of them, i.e., Texas, Cornell, Washington and Wisconsin, are the sub-networks of the WebKB network. Each of them is the collection of webpages from an American university. Similarly, nodes in Wiki network are webpages from Wikipedia.

For the inductive learning task, the protein-protein interaction (PPI) dataset [41] is employed as shown in the last row of Table I. PPI dataset consists of 24 attribute graphs, each of which corresponds to a different human tissue and contains 2,373 nodes in average. Each node possesses 50 features including positional gene sets, motif gene sets and immunological signatures. We employ 121 cellular functions from the gene ontology sets, which are collected from the Molecular Signatures Database, as labels. Algorithms are trained on 20 graphs, validated on 2 graphs and tested on 2 graphs. The 20 graphs for training and 2 graphs for validation are fully labelled, while the 2 graphs for testing are unseen during training and validation steages.

TABLE II
TRANSDUCTIVE NODE CLASSIFICATION RESULTS WITH DATASET SPLIT AS IN [42].

Methods	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg	59.5%	60.1%	70.7%
SemiEmb	59.0%	59.6%	71.7%
LP	68.0%	45.3%	63.0%
DeepWalk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
Chebyshev	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	79.0%
MoNet	81.7%	69.9%	78.8%
ST-GCN	81.7%	70.1%	79.2%
SGC	81.0%	71.9%	78.9%
APPNP	83.2%	71.8%	79.7%
GAT	83.0%	72.5%	79.0%
PGCN-G+	**84.5%**	**74.2%**	**80.4%**

B. Baselines

For the transductive learning task, we employ 14 state-of-the-art semi-supervised node classification algorithms, including multilayer perceptron (MLP), label propagation (LP) [43], semi-supervised embedding (SemiEmb) [44], manifold regularization (ManiReg) [45], graph embedding (DeepWalk) [46], iterative classification algorithm (ICA) [47], graph-based semi-supervised learning framework (Planetoid) [42], graph convolution with Chebyshev filters [14], graph convolutional network (GCN) [23], mixture model networks (MoNet) [21], self-training GCN (ST-GCN) [24], graph attention networks (GAT) [22], simplified GCN (SGC) [48], and approximated

TABLE III
TRANSDUCTIVE NODE CLASSIFICATION RESULTS (BOTH ACCURACIES AND RUNTIMES) WITH RANDOM DATASET SPLITS.

Methods	Cora	Citeseer	Pubmed
Chebyshev	76.8% (2.5s)	67.2% (5.2)	75.8% (2.9)
GCN	79.1% (1.3s)	68.2% (1.4s)	76.8% (0.9s)
MoNet	80.2% (3.6s)	69.1% (4.7s)	77.8% (9.3s)
ST-GCN	79.3% (1.4s)	67.7% (1.4s)	77.0% (1.0s)
SGC	80.0% (0.7s)	68.3% (0.4s)	76.6% (0.7s)
APPNP	82.2% (1.1s)	70.0% (1.2s)	78.9% (1.1s)
GAT	81.3% (5.8s)	69.1% (6.6s)	77.9% (15.4s)
PGCN-G+	**84.1%** (6.4s)	**73.5%** (7.1s)	**79.9%** (17.8s)

TABLE IV
COMPARISONS WITH DIFFERENT COMMUNITY DETECTION METHODS.

Datasets	SBM	MRF	LDC	SCI	MBP	GCN	Ours
Texas	48.1	30.6	38.8	49.7	53.6	57.1	**65.2**
Cornell	37.9	31.8	30.3	36.9	47.2	46.3	**55.9**
Washin.	31.8	35.0	30.0	46.1	42.9	54.9	**66.3**
Wiscon.	32.8	28.6	30.2	46.4	63.4	55.6	**67.9**
Wiki	2.6	31.1	28.8	29.5	**46.3**	16.4	44.7
CiteSeer	26.6	22.2	24.9	34.4	49.5	70.3	**74.2**
Cora	38.5	58.1	34.1	41.7	57.6	81.4	**84.5**
PubMed	53.6	55.5	63.6	-	65.7	79.0	**80.4**

TABLE V
INDUCTIVE CLASSIFICATION RESULTS.

Methods	PPI (split as [19])	PPI (random splits)
Random	0.396	0.388
Logistic Regression	0.422	0.436
Inductive GCN	0.500	0.496
GraphSAGE-mean	0.598	0.585
GraphSAGE-LSTM	0.612	0.615
GraphSAGE-pool	0.600	0.610
GAT	0.934	0.930
PGCN-G+	**0.979**	**0.977**

personalized propagation of neural predictions (APPNP) [49], for comparisons. All the baseline methods except DeepWalk are semi-supervised methods, while DeepWalk learns the node embeddings in an unsupervised manner and feeds the learned embeddings into a classifier, which is trained with part of the node labels. Besides, to give a comprehensive understanding, we also compare our method with another 5 community detection methods, which are all unsupervised methods, on attribute network. Degree-corrected Stochastic Block Model (SBM) [50] and MRF [51] only adopt network topology, while LDC [52], SCI [53] and MBP [54] utilize both the network topology and node attributes. All the results of the baseline methods are either from their original papers or produced by running the codes from the authors with their default settings.

For the inductive learning task, we employ 7 state-of-the-art algorithms, including random classifier, logistic regression based on node feature without network structure, inductive variant of the GCN [23], three variants of GraphSAGE [19] with different aggregator functions and GAT [22].

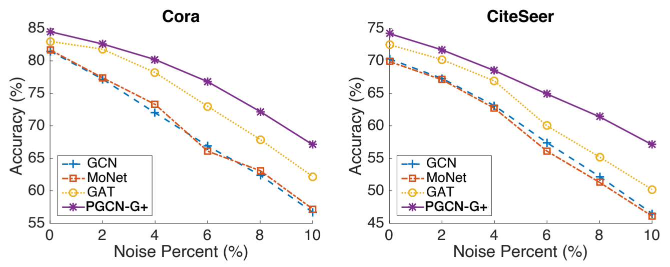
C. Results Analysis

1) Transductive Learning : In this task, PGCN-G+ is compared to 14 baseline methods by applying the experimental settings in [42], where 20 nodes per class, 500 nodes and 1000 nodes are employed for training, validation and performance evaluation, respectively. As shown in Table II, our PGCN-G+ outperforms all the baseline methods. The improvement of PGCN-G+ compared to GAT, which achieves the best performance except the proposed method, is moderately significant. Although the accuracy improvement is 1.53% in average, a large proportion of the error rates has been reduced. Specifically, the average error rates of GAT and our proposed PGCN-G+ are 21.84% and 20.3%, respectively, and thus the error rate reduction is (21.84% - 20.3%)/21.84% = 7.05%.

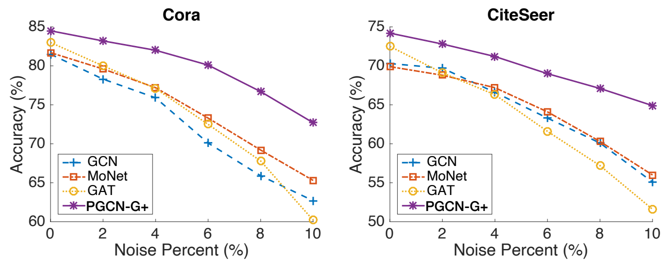
Besides, to provide a comprehensive comparison, we randomly split the network with the same numbers of nodes for training, validation and evaluation as [42] for 100 times and report both the average accuracies and runtime in Table III. Note that different algorithms are originally implemented with different framework, such as Tensorflow or Pytorch. For fair comparison, Pytorch Geometric [55], which implements most of the GNNs, is employed here. As can be observed the accuracies and runtime are the results of training with 200 epochs. The runtime is the sum of training time, validation time, and evaluation time. The average accuracies of PGCN-G+ also outperforms all the baseline methods. Compared to GAT, which share the same covariance matrix among all the nodes, the additional runtime of our PGCN-G+ are mainly spent on the calculation of the covariance matrix for each node, which is the main reason of the high accuracies and noise robustness of PGCN-G+. It not only demonstrates the effectiveness of our PGCN-G+ on jointly exploiting the node content and network topology, but also indicates the superiority of applying the distribution-based node representation.

In addition, we want to validate the effectiveness of PGCN-G+ on leveraging labelled data, especially limited labelled data. Thus, the performances of four representative methods including PGCN-G+ are provided with varying percentages of labelled nodes. As shown in Figure 7, the performance gains of PGCN-G+ are consistent and significant compared to the baseline approaches.

To comprehensively evaluate the proposed PGCN-G+, we compare it with the methods from community detection area. In addition to the three citation networks, five webpage networks, which is often adopted to evaluate the community detection methods, are employed. For the four medium webpage networks, including Cornell, Texas, Washington and Wsicsonsin, we adopt 20% labelled nodes for training, 10% labelled nodes for validation, and other nodes for testing. For large Wiki network, the percentages of nodes for training and validation are 3% and 3%, respectively. The results are shown in Table IV. As can be observed, GCN only significantly improves the existing community detection methods on three citation networks. However, its performances on the five webpage networks are only comparable to or even worse than the community detection methods. This phenomenon indicates that simply smoothing in the local neighbourhood (exploited by GCN) is not widely applicable. Its performances will be



(a) Performance with link noises



(b) Performance with content noises

Fig. 8. Noise impacts on performance.

degraded by the noises in topology and node content. By iteratively denoising the network topology and node content, PGCN-G+ remarkably improves the accuracies and possesses robustness against the interferences.

2) Inductive Learning : The performance on two unseen graphs is measured with the micro-averaged F1 score, which is already employed in the evaluation of GraphSAGE [19] and GAT [22]. The results shown in Table V are the averages of 10 runs for each methods. For comprehensive comparison, both the dataset split strategy in GraphSAGE [19] and the random dataset split are employed. Both GAT and PGCN-G+ significantly outperform other methods. The error rates of GAT and PGCN-G+ are 6.6% and 2.1%, respectively. Therefore, the error reduction of PGCN-G+ is (6.6% - 2.1%)/6.6% = 68.2%, which is quite significant. However, GAT models the content of all the nodes with identical covariance as shown in Figure 3(D) and Theorem 2. This hinders its ability to model different content uncertainties of different nodes. By modeling different nodes with different covariances, PGCN-G+ overcomes this difficulty and achieves a better performance.

D. Robustness Against Noises

To demonstrate the robustness of PGCN-G+ against noises, we manually add noises to links and node content, and quantitatively measure their impacts on performance. For links, the noises is added by randomly altering the existing connections. For node content in the bag-of-word form, we randomly remove some percentages of words. The results on Cora and Citeseer networks are shown in Figure 8. It reveals that the consistent performance improvements of PGCN-G+ compared to the baselines are more significant at high noise levels. Take the Cora network as an example. The improvements of PGCN-G+ over other methods is 3.2% when 2% noises are added, while the improvement is 7.4% when 10% noises are added.

Figure 8(a) gives the results of different methods with varying percentages of link noises. Compared to GCN and MoNet, GAT and PGCN-G+ are more robust, because both of them refine the network topology according to node content information. Due to the distribution-based node representation, which facilitates the modeling of the content uncertainties, PGCN-G+ outperforms GAT, especially at high noise levels. Figure 8(b) presents the results with different levels of the node content noises. As can be observed, the performances of PGCN-G+ are still much better than the other methods, especially at high noise levels, because PGCN-G+ iteratively denoises the network topology and node content. On the contrary, different from the case of link noises, the performance of GAT drastically decreases when the amount of the node content noise increases, be because GAT only refines the network topology with respect to the node content. When a certain amount of node content noises is added, this refinement strategy will not improve the classification performances, but provide potential negative impacts on the performances because the refinement may not be correct. According to the robustness test, PGCN-G+ possesses superior robustness against noises than the baseline methods.

VII. Conclusions and Future Work

In this paper, we propose a top-down latent space framework, Probabilistic Graph Convolution Network (PGCN), to provide a probabilistic perspective for the semi-supervised node classification problem. PGCN formulates the nodes in a distribution-form representation and models the graph convolution operation as a topology constraint. This model can iteratively denoise the network topology and node content with respect to each other. To be specific, we apply the general framework to a specific type of problems, which assumes the existence of Gaussian distribution, and propose PGCN with Gaussian distribution representation (PGCN-G). Then, we improve PGCN-G to PGCN-G+ by imposing the covariance matrices to possess identical singular vectors. Our derivations have proved that the existing GCN, GAT and GMM are the special cases of PGCN-G+, and our proposed framework can elaborate their characteristics and relationships. Extensive experiments demonstrate the effectiveness of the proposed method compared to many state-of-the-art approaches and its robustness against noises.

References

[1] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.
[2] W. Chen, D. Tan, Q. Yang, T. Gu, and J. Zhang, "Ant colony optimization for the control of pollutant spreading on social networks," *IEEE Transactions on Cybernetics*, pp. 1–13, 2019.
[3] W. Chen, Y. Jia, F. Zhao, X. Luo, X. Jia, and J. Zhang, "A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 842–857, 2019.
[4] T. Zhao, W. Chen, A. W. Liew, T. Gu, X. Wu, and J. Zhang, "A binary particle swarm optimizer with priority planning and hierarchical learning for networked epidemic control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–15, 2019.
[5] T. Zhao, W. Chen, S. Kwong, T. Gu, H. Yuan, J. Zhang, and J. Zhang, "Evolutionary divide-and-conquer algorithm for virus spreading control over networks," *IEEE Transactions on Cybernetics*, pp. 1–15, 2020.

[6] X. Wen, W. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, and J. Zhang, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 363–377, 2017.

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

[8] D. Jin, X. Wang, D. He, J. Dang, and W. Zhang, "Robust detection of link communities with summary description in social networks," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[9] D. Jin, K. Wang, G. Zhang, P. Jiao, D. He, F. Fogelman-Soulie, and X. Huang, "Detecting communities with multiplex semantics by distinguishing background, general and specialized topics," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[10] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009.

[11] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 3889–3895.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2012, NIPS 2012 ,Lake Tahoe, Nevada, United States, December 3-6, 2012,*, 2012, pp. 1106–1114.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2016, NIPS 2016, Barcelona, Spain, December 5-10, 2016,*, 2016, pp. 3837–3845.

[15] D. K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2015, Montreal, Quebec, Canada, December 7-12, 2015*, 2015, pp. 2224–2232.

[16] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 2014–2023.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[18] D. Jin, Z. Liu, W. Li, D. He, and W. Zhang, "Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2019, pp. 152–159. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.3301152

[19] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2017, NIPS 2017 Long Beach, CA, USA, December 4-9, 2017*, 2017, pp. 1025–1035.

[20] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2018, London, UK, August 19-23, 2018*, 2018, pp. 1416–1424.

[21] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5425–5434.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30-May 3, 2018*, 2018.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*, 2017.

[24] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence, AAAI 2018 , New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 3538–3545.

[25] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 2018, pp. 2609–2615.

[26] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2020.

[27] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised domain adaptive graph convolutional networks," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 2020, pp. 1457–1467.

[28] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 2019, pp. 1907–1913.

[29] M. Wu, S. Pan, X. Zhu, C. Zhou, and L. Pan, "Domain-adversarial graph neural networks for text classification," in *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, 2019, pp. 648–657.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[31] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[32] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2014, NIPS 2014 Montreal, Quebec, Canada, December 8-13 2014*, 2014, pp. 2672–2680.

[33] C. M. Bishop and J. Lasserre, "Generative or Discrimative? Getting the Best of Both Worlds," in *Bayesian Statistics 8*. Oxford University Pres, 2007, pp. 3–24.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[35] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in wasserstein space," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2018, London, UK, August 19-23, 2018*, 2018, pp. 2827–2836.

[36] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30-May 3, 2018*, 2018.

[37] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[38] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, Canada, May 7-9, 2015*, 2015.

[40] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1263–1272.

[41] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[42] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 40–48.

[43] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference, ICML 2003, 2003, Washington, DC, USA, August 21-24*, 2003, pp. 912–919.

[44] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade - Second Edition*, 2012, pp. 639–655.

[45] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[46] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, August 24 - 27, 2014, 2014, pp. 701–710.

[47] Q. Lu and L. Getoor, "Link-based classification," in *Proceedings of the 20th International Conference xon Machine Learning, ICML 2003, Washington, DC, USA, August 21-24, 2003*, 2003, pp. 496–503.

[48] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 6861–6871.

[49] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[50] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, p. 016107, 2011.

[51] D. He, X. You, Z. Feng, D. Jin, X. Yang, and W. Zhang, "A network-specific markov random field approach to community detection," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018 , New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 306–313.

[52] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: a discriminative approach," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, Paris, France, June 28 - July 1, 2009*, 2009, pp. 927–936.

[53] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large attribute networks," in *Proceedings of the 20th AAAI Conference on Artificial Intelligence, 2016, AAAI 2016, Phoenix, Arizona, USA., February 12-17*, 2016, pp. 265–271.

[54] D. He, Z. Feng, D. Jin, X. Wang, and W. Zhang, "Joint identification of network communities and semantics via integrative modeling of network topologies and node contents," in *Proceedings of the 21st AAAI Conference on Artificial Intelligence, AAAI 2017, San Francisco, California, USA., February 4-9, 2017*, 2017, pp. 116–124.

[55] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Liang Yang is an Assistant Professor in the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China. He received the B.E. and M.E. degrees both in computational mathematics from Nankai University, Tianjin, China, in 2004 and 2007, and the Ph.D. degree in computer science from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2016. His current research interests include community detection, graph neural networks, low-rank modeling, and data mining.

Yuanfang Guo (Senior Member, IEEE) received his B.E. degree in computer engineering and his Ph.D. degree in electronic and computer engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2009 and 2015, respectively. Then, he served as an Assistant Professor with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, for three years. He is currently an Assistant Professor with the Laboratory of Intelligent Recognition and Image Processing, School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include image/video security, compression, processing, and data analysis.

Junhua Gu was born in 1966. He is currently working at the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China. His main research interests include data mining, intelligent information processing, information acquisition and integration, intelligent computing and optimization, function and information display, software engineering and project management.

Di Jin Di Jin received his Ph.D. degree in computer science from Jilin University, Changchun, China, in 2012, He was a Post-Doctoral Research Fellow at the School of Design, Engineering, and Computing, Bournemouth University, Poole, U.K., from 2013 to 2014. He is currently an Associate Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. He has published more than 50 papers in international journals and conferences in the areas of community detection, social network analysis, and machine learning.

Bo Yang received the B.S., M.S., and Ph.D. degrees in computer science from Jilin University in 1997, 2000, and 2003, respectively. He is currently a Professor with the College of Computer Science and Technology, Jilin University. He is currently the Director of the Key Laboratory of Symbolic Computation and Knowledge Engineer, Ministry of Education, China. His current research interests include data mining, complex/social network modeling and analysis, and multi-agent systems. He is currently working on the topics of discovering structural and dynamical patterns from large-scale and time-evolving networks with applications to Web intelligence, recommender systems, and early detection/control of infectious events. He has published over 100 articles in international journals, including IEEE TKDE, IEEE TPAMI, ACM TWEB, DKE, JAAMAS, and KBS, and international conferences, including IJCAI, AAAI, ICDM, WI, PAKDD, and ASONAM. He has served as an associated editor and a peer reviewer for international journals, including Web Intelligence and served as the PC chair and an SPC or PC member for international conferences, including KSEM, IJCAI, and AAMAS.

Xiaochun Cao (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from Beihang University, Beijing, China, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA. After graduation, he spent about three years at ObjectVideo Inc. as a Research Scientist. From 2008 to 2012, he was a Professor with Tianjin University, Tianjin, China. He is currently a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen. He has authored and coauthored more than 100 journal and conference papers. Prof. Cao is a Fellow of the IET. He is on the Editorial Boards of the IEEE Transactions on Image Processing (Senior Area Editor), IEEE Transactions on Multimedia, IEEE Transactions on Circuits and Systems for Video Technology. His dissertation was nominated for the University of Central Florida's university-level Outstanding Dissertation Award. In 2004 and 2010, he was the recipient of the Piero Zamperoni Best Student Paper Award at the International Conference on Pattern Recognition.