Probabilistic Graph Convolutional Network via Topology-Constrained Latent Space Model

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

***Abstract*—Although many graph convolutional neural networks (GCNNs) have achieved superior performances in semisupervised 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 are 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 semisupervised 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 a Gaussian distribution, the transductive node classification problems can be solved by the general framework and a specific method, called PGCN with the 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 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 the effectiveness of this *top-down* framework demonstrated via extensive experiments, it can also be deduced to cover the existing methods, graph convolutional network, graph attention network, and Gaussian mixture model and elaborate their characteristics and relationships by specific derivations.**

***Index Terms*—Graph convolutional network (GCN), node classification, probabilistic model, semisupervised learning.**

Manuscript received September 9, 2019; revised March 9, 2020 and June 6, 2020; accepted June 18, 2020. 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. This article was recommended by Associate Editor Y. Jin. *(Corresponding author: Yuanfang Guo.)*

Liang Yang was with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, China. He is now with the School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China (e-mail: yangliang@vip.qq.com).

Yuanfang Guo is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: andyguo@buaa.edu.cn).

Junhua Gu is with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China, and also with the Hebei Province Key Laboratory of Big Data Calculation, Hebei University of Technology, Tianjin 300401, China (e-mail: jhgu@hebut.edu.cn).

Di Jin is with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, 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, Changchun 130012, China, and also with the College of Computer Science and Technology, Jilin University, Changchun 130012, 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).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.3005938

I. Introduction

NETWORKS, including social networks, biological networks, and technological networks, are ubiquitous in the real world [1]–[6]. They usually possess some common properties, such as community structure [7]–[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, also known as graph clustering, only exploits the network topology information [7] and 2) node classification, also known as semisupervised community detection, usually resolves the problem by adopting both the network topology and label information [11].

Node classification in an attributed network, which exploits the characteristics of network topology, metadata in the nodes/edges, and the labels, has drawn significant attention from researchers because of the varieties of the metadata. The majority of the existing techniques focuses on developing efficient approaches to extract semantic features from both the network topology and metadata. 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 neighborhoods and maintain the weight sharing property in CNNs, graph convolutional neural networks (GCNNs) [14]–[18] are invented and they can be classified into two categories: 1) spatial and 2) spectral

approaches. Spatial approaches either learn different weight matrices for nodes with various degrees [15] or sample a fixed-size neighborhood for each node [19], [20]. The mixture model network (MoNet) unifies the spatial convolution operation as a weighted mixture of all the neighbors and specifies the weights via a Gaussian kernel [21]. Graph attention networks (GATs) apply the attention mechanism to effectively process the variable-sized neighborhoods [22].

Spectral approaches employ the convolution operation in the Fourier domain to effectively handle various neighborhood sizes. Traditionally, they first 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 large-scale networks. Therefore, Defferrard *et al.* [14] approximated $g_\theta(\Lambda)$ by a truncated expansion of the Chebyshev polynomials. Kipf and Welling [23] further approximated it with the first Chebyshev polynomials. This simplification, called 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 neighbors [24].

Although many GCNN algorithms have been proposed and some of them achieve superior performances [25]–[29], they are designed from either the spatial or spectral perspective, yet without a *general* theoretical basis. The lack of a theoretical basis hinders both the analysis of the existing methods and development of the novel methods. Besides, most of the existing GCNN 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 article, we formulate the basic graph convolution operation by a topology-constrained latent space model, probabilistic GCN (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 Fig. 1, their uncertainties can also be formulated. When specifying the distribution in (PGCN) as a Gaussian distribution, it is equivalent to applying the general framework to the specific type of problems, and a specific method, called PGCN with the 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

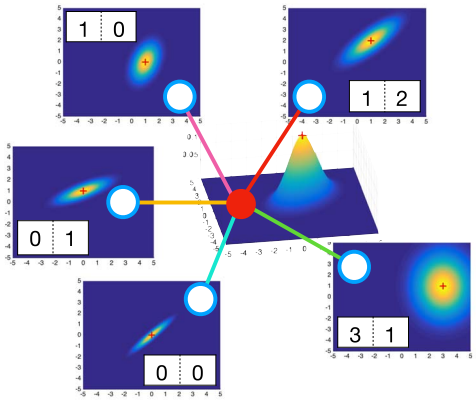


Fig. 1. Illustrative example of the proposed framework. The uncertainties of the links and node contents are modeled as distributions from a probabilistic perspective. Each circle stands for a node in the network, while the 2-D vector represents its content. The connection between each pair of nodes 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 contents obey different distributions, which are represented as the 2-D color maps, except the center (red) node. To demonstrate the different probabilities of connected nodes that belong to the same classes, the connections are illustrated via different colors.

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 Gaussian mixture model (GMM) and elaborate their characteristics and relationships, by specific derivations and comparing PGCN-G+'s iterative algorithms with the training processes of graph neural networks (GNNs).

The contributions are summarized as follows.

1) To simultaneously denoise the network topology and node content, we formulate GCNNs as a general latent space framework, PGCN, and represent the node content and edge weight as parametric distributions for semisupervised node classification.
2) By assuming that node content obeys the Gaussian distribution, we propose 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 nodes to possess identical the same singular vectors.
3) 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 $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. $\mathcal{V} = \{v_i | i = 1, \ldots, N\}$ 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 nth column of $\mathbf{A}$, can be utilized to represent the neighborhood of vertex v_n. $d_n = \sum_j a_{nj}$ is the degree of vertex v_n, and $\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_N)$ 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 $\mathbf{L} = \mathbf{D}^{-(1/2)}\mathbf{L}\mathbf{D}^{-(1/2)}$, respectively.

For the semisupervised 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 semisupervised 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 semisupervised node classification learns the model from a set of L attributed graphs $\{\mathcal{G}_i\}_{i=1}^{L}$ with labels and classifies all nodes of M unseen attributed graphs $\{\mathcal{G}_i\}_{i=L+1}^{L+M}$. All 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, 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. General Framework PGCN

Most existing works on semisupervised node classification are discriminative algorithms that 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 labeled data are limited. Therefore, in this article, we pioneer to model the semisupervised node classification as a generative model. In this model, variables are represented via distributions instead of vectors to characterize their 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 an 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 the 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, an 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 neighbors. 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 Fig. 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 Fig. 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 Fig. 1. Note that the specific distribution will be discussed in the next section. $p(\mathbf{x}_n|\boldsymbol{\theta}_n)$ 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 neighbors 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

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

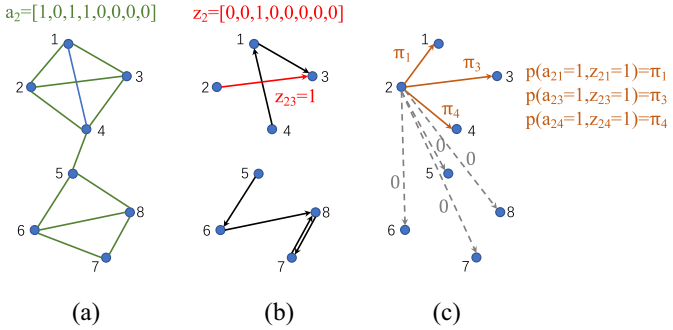


Fig. 2. Example of joint distribution $p(\mathbf{a}_n, \mathbf{z}_n)$. (a) Topology and $\mathbf{a}_2$. (b) Example of $\mathbf{z}_2$ where $z_{23} = 1$ represents v_3 is the most similar node to v_2. (c) Joint distribution of a_{2k} and z_{2k}.

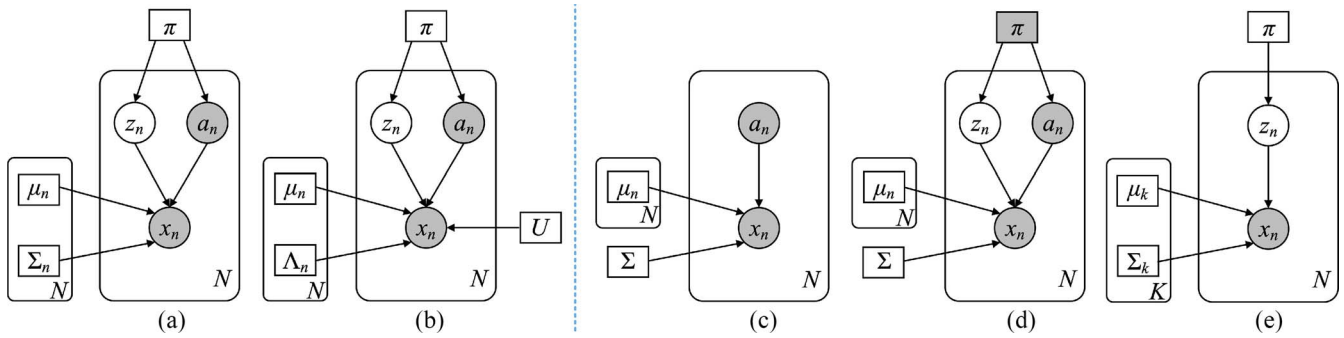


Fig. 3. Graphical representations of our proposed PGCN-G and PGCN-G+, along with PGCN-G+'s special cases GCN (one layer), GAT (one layer), and GMM, where the rectangles and circles denote parameters and random variables, respectively. The unshaded nodes represent the latent variables/parameters to be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

Given the joint distribution as

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

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

$$p(\mathbf{a}_n, \mathbf{x}_n) = \sum_{\mathbf{z}_n} p(\mathbf{a}_n, = \sum_{k\in N(n)\cup\{n}$$

where $N(n)$ denotes the attributed graph $\mathcal{G} = ($ the likelihood function

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

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

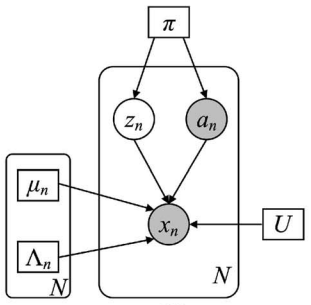
$$\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \Theta) = \sum_{n=1}^{N} \log \left\{ \sum_{k\in N(n)\cup\{n\}} \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k) \right\}. \quad (6)$$

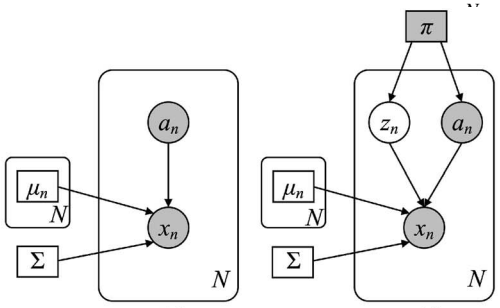
Unfortunately, direct optimization of this (log)-likelihood function $\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \Theta)$ is difficult because the logarithm function outside the summation function will require a very large amount of computations. Instead, optimizing the complete data-likelihood function $p(\mathbf{A}, \mathbf{X}, \mathbf{Z}|\pi, \Theta)$ is much easier. Here, the 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 $\log p(\mathbf{A}, \mathbf{X}|\boldsymbol{\pi}, \Theta)$ (E-step) and then optimizing that (M-step). In the E-step, the posterior distribution of the latent variables can be computed with respect to the current parameters $\{\boldsymbol{\pi}^{\text{old}}, \Theta^{\text{old}}\}$ as

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

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

$$\mathcal{Q}\left(\{\boldsymbol{\pi}, \Theta\}, \left\{\boldsymbol{\pi}^{\text{old}}, \Theta^{\text{old}}\right\}\right) = \sum_{\mathbf{Z}} p\left(\mathbf{Z}|\mathbf{A}, \mathbf{X}, \boldsymbol{\pi}^{\text{old}}, \Theta^{\text{old}}\right) \log p(\mathbf{A}, \mathbf{X}, \mathbf{Z}|\boldsymbol{\pi}, \Theta)$$





$$\sum_{\in N(n)\cup\{n\}} \log \pi_k p(\mathbf{x}_n|\boldsymbol{\theta}_k) \quad (8)$$

$\boldsymbol{\pi}, \Theta)$ is shown in (3). $, \Theta\}$ are determined by via

$$\Theta\}, \left\{\boldsymbol{\pi}^{\text{old}}, \Theta^{\text{old}}\right\}\right). \quad (9)$$

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

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

where $\boldsymbol{\mu}_n$ is a K-dimensional mean vector and $\boldsymbol{\Sigma}_n$ is a $K \times K$ covariance matrix. This model is called PGCN-G. Its graphical representation is shown in Fig. 3(a). According to the EM algorithm in (7) and (9), the following updating rules can be obtained as:

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

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

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

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 $.^{\text{old}}$ in the parameters.

IV. PGCN for Semisupervised Node Classification

In this section, the proposed PGCN is applied to the (transductive) semisupervised node classification problem. Then, an illustrative explanation of its mechanism and validity is given. Finally, the applicability of the general framework to the inductive learning task is demonstrated.

A. PGCN-G+

In (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 neighbors of the vertex v_n, that is, 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 (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 $\mathbf{\Lambda}_n = \text{diag}(\lambda_{n1}, \ldots, \lambda_{nK})$ 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 $\mathbf{\Lambda}_n^{-1} = \text{diag}(\lambda_{n1}^{-1}, \ldots, \lambda_{nK}^{-1})$. Then, the multivariate Gaussian distribution in (10) can be rewritten as

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

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 Fig. 4. Motivated by the previous distribution-based network embedding schemes, where the covariance matrix of the 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

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

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

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

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

$$\mathcal{Q}\Big(\{\boldsymbol{\pi}, \mathbf{\Theta}\}, \big\{\boldsymbol{\pi}^{\text{old}}, \boldsymbol{\mu}^{\text{old}}, \mathbf{\Lambda}^{\text{old}}, \mathbf{U}^{\text{old}}\big\}\Big) = \sum_{\mathbf{Z}}\left[p\Big(\mathbf{Z}|\boldsymbol{\pi}^{\text{old}}, \boldsymbol{\mu}^{\text{old}}, \mathbf{\Lambda}^{\text{old}}, \mathbf{U}^{\text{old}}\Big) \times \sum_{n=1}^{N}\sum_{k\in N(n)\cup\{n\}} \log \pi_k \mathcal{N}\big(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{U}^T\mathbf{\Lambda}_k\mathbf{U}\big)\right]. \quad (16)$$

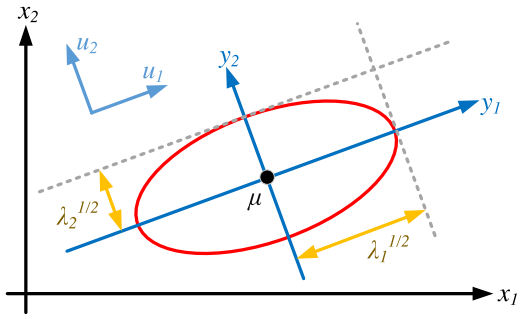


Fig. 4. New coordinate system defined by the singular vectors $\mathbf{U}$ (refer to [37, Fig. 2.7]).

To iteratively optimize the problem, three variables, $\mathbf{U}$, $\mathbf{\Lambda}_n = \text{diag}(\lambda_{n1}, \ldots, \lambda_{nK})$, 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

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

where $\mathbf{u}_{k*}$ denotes the kth row of $\mathbf{U}$. Compared to the updating rule of $\mathbf{\Sigma}_n$ in (13), updating $\mathbf{\Lambda}_n$ in (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

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

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

where

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

Equation (19) can be reformed to

$$\mathbf{T}^{\text{PGCN-G+}} = \mathbf{O}\mathbf{X}\mathbf{W} \quad (21)$$

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 the nth 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 (16) with respect $\mathbf{U}$ equal to 0. 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, that is, the nth 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 labeled nodes

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

where $\mathcal{V}_l$ represents the set of labeled nodes, F stands for the number of classes, and $\mathbf{u}_{f*}$ denotes the fth row of $\mathbf{U}$. Then, $\mathbf{U}$ can be estimated by minimizing $\mathcal{L}$. Besides, the dimensions of $\mathbf{\Lambda}_n$ 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:

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

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

In (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 (24), $\mathbf{U}^*$ is fixed and the optimal $\{\boldsymbol{\pi}^*, \{\boldsymbol{\mu}_n^*\}_{n=1}^N, \{\mathbf{\Lambda}_N^*\}_{n=1}^N\}$ are obtained via the EM algorithm. In the E-step, the posterior is computed via (18). In the M-step, the model parameters $\boldsymbol{\mu}_n$'s, $\mathbf{\Lambda}_n$'s and π_n's are updated via (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 (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 (23) and (24). To speed up the convergence, the inner loop is omitted as most of the alternating direction method of multipliers (ADMMs) [38] approach, and the final PGCN-G+ algorithm for transductive semisupervised node classification is shown in Algorithm 1.

There exist two reasons of separately learning $\mathbf{U}$ and other parameters. First, as shown in (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 $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_n, \mathbf{U}^T\mathbf{\Lambda}_n\mathbf{U})$ in (14) can then be reformed to $\mathcal{N}(\mathbf{Ux}|\mathbf{U}\boldsymbol{\mu}_n, \mathbf{\Lambda}_n)$, which considers $\mathbf{Ux}$ as a random variable. Then, we can estimate its mean $\mathbf{U}\boldsymbol{\mu}_n$ and diagonal covariance matrix $\mathbf{\Lambda}_n$ by maximizing (24) with EM, which guarantees that the results converge to a locally optimal.

Remark: In (18), $\mathbf{f}_{nk}^T\mathbf{\Lambda}_k^{-1}\mathbf{f}_{nk}$ 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 $\mathbf{\Lambda}_k^{-1}$ 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 GAT, which applies the same attention mechanism to all nodes.

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 $\mathbf{T}$.

```
Randomly initialize U;
Initialize π_n = d_n / Σ d_n;
while not convergence do
    %—— Exception-step ——
    Update posterior γ(z_nk) via Eq. (18);
    Update O with o_nk = γ(z_nk)/N_n;
    %—— Maximization-step ——
    Update μ_n via Eq. (19);
    Update Λ_n via Eq. (17) ;
    Update π_n = Σ_{k∈N(n)∪{n}} γ(z_nk)/N;
    %—— Mapping-step ——
    Update U by minimizing Eq. (22) via batch gradient decent;
end
T^PGCN-G+ = σ(OXU);
return O, U, T.
```

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 the Gaussian distributions. In this section, PGCN-G+ is compared to GCN [23] to provide an illustrative explanation of its mechanism and validity. As shown in Fig. 5(a), GCN smoothes the observed node contents in the observed neighborhoods [24] (where the details of GCN are 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 EM, Algorithm 1 iteratively updates the distribution-based node representations (parameterized by $\boldsymbol{\mu}_n$ and $\mathbf{\Lambda}_n$) and the edge weights $\gamma(z_{nk})$ shown in Fig. 5(b).

The updates of distribution-based node representations are carried out based on the estimated edge weights, which represent the probabilities of the nodes in the same class being connected. As shown in (19), the mean $\boldsymbol{\mu}_n$ of node content distribution is the weighted average of its neighbors 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 neighborhood distributions. Therefore, the updates of distribution-based node representations are equivalent to denoise the node contents according to the network topology.

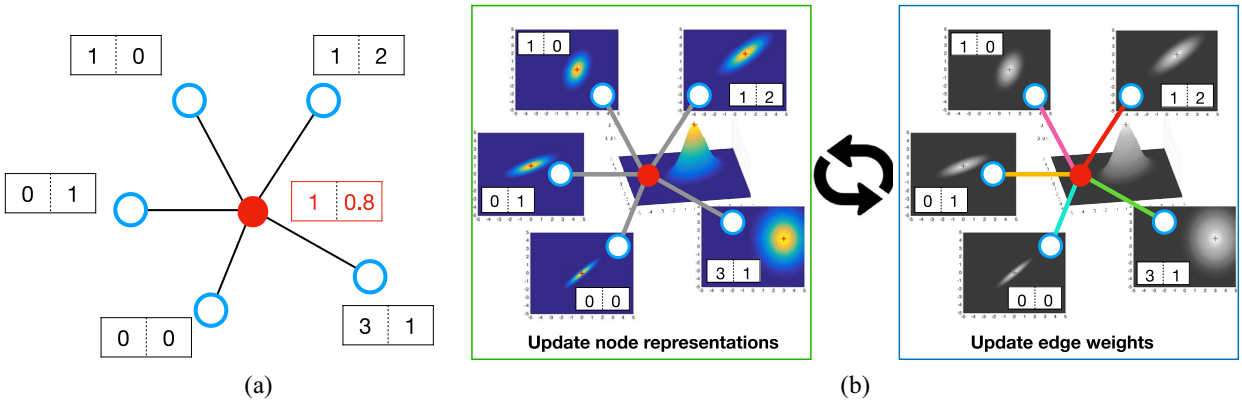


Fig. 5. Illustrative explanation of the mechanism of our proposed PGCN-G+ work. (a) GCN [23] smoothes the node content over the neighborhood based on the observed network topology and node content. (b) PGCN-G+ iteratively updates the distribution-based node representations ($\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$)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

Meanwhile, the updates of edge weights are performed according to the estimated node content distributions. As shown in (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 are 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

The inductive semisupervised 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.

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 $\mathbf{T}$.

Initialize $\pi_n = d_n / \sum d_n$;
while *not convergence* **do**
  %—— Exception-step ——
  Update posterior $\gamma(z_{nk})$ via Eq. (18);
  Update $\mathbf{O}$ with $o_{nk} = \gamma(z_{nk})/N_n$;
  %—— Maximization-step ——
  Update $\boldsymbol{\mu}_n$ via Eq. (19);
  Update $\boldsymbol{\Lambda}_n$ via Eq. (17) ;
  Update $\pi_n = \sum_{k\in N(n)\cup\{n\}} \gamma(z_{nk})/N$;
end
$\mathbf{T}^{\text{PGCN-G+}} = \sigma(\mathbf{OXU})$;
return $\mathbf{O}$, $\mathbf{T}$.

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.

A. Graph Neural Networks

In this section, we first interpret the correspondences between the training processes of GNNs and our proposed PGCN-G+ as shown in Fig. 6.

On the one hand, the training process of GNNs alternately performs 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 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 neighborhoods 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 backpropagating 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, that is, exception step (E-step), maximization 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 the maximization step propagates the node attributes in the local neighborhoods. 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 the maximization step will update the model parameters, which corresponds to the backward propagations in GNNs. In the mapping step, $\mathbf{U}$ is updated by minimizing the cross-entropy loss. In the 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

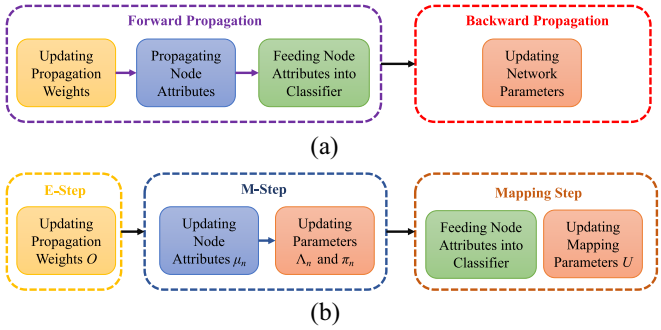


Fig. 6. Correspondences between the training processes of GNNs and our proposed PGCN-G+. Boxes with the same color are possessing similar functionalities.

our proposed PGCN-G+ correspond to the training process of GNNs.

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

1) Graph Convolutional Network: 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

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

where $\mathbf{I}_N$ is the identity matrix with size N. Then, we renormalize $\mathbf{I}_N + \mathbf{D}^{-(1/2)}\mathbf{A}\mathbf{D}^{-(1/2)}$ to $\tilde{\mathbf{D}}^{-(1/2)}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-(1/2)}$, 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-D 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

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

where $\mathbf{T}^{\mathrm{GCN}} = [t_{nf}]$ is the convolved signal matrix. Li *et al.* [24] concluded the mechanism and success of GCN that it actually performs a symmetric Laplacian smoothing ($\mathbf{H}_{\mathrm{GCN}} = \tilde{\mathbf{D}}^{-(1/2)}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-(1/2)}$) operation followed by a projection $\mathbf{T}^{\mathrm{GCN}} = \mathbf{H}_{\mathrm{GCN}}\mathbf{W}$. 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, an asymmetric Laplacian smoothing operation

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

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

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

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 $\mathbf{\Sigma}_n = \mathbf{\Sigma}$ for all vertices and sets the graph to be a self-loop ($a_{nn} = 1$ for all n) as shown in Fig. 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

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

Different from (5), the only variables here are $\boldsymbol{\mu}_k$ and $\mathbf{\Sigma}$. 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 nth rows of $\mathbf{H}$ and $\mathbf{X}$ are $\boldsymbol{\mu}_n$ 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 $\mathbf{\Sigma}_n = \mathbf{U}^T\mathbf{\Lambda}_n\mathbf{U}$. Since $\mathbf{\Sigma}_n = \mathbf{\Sigma}$ 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: The success GCN is achieved by smoothing (averaging) the node content in a local neighborhood. 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 neighborhoods. GAT replaces the smoothing layer in GCN with a graph attention layer. To obtain the nodes with more attentions, attention coefficients

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

are 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 neighborhood 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 reweighting of the adjacency matrix $\mathbf{A} = [a_{nk}] \in \mathbb{R}^{N\times N}$. $o_{nk} \neq 0$ when $a_{nk} = 1$, that is, the vertices v_n and v_k are connected. Then, GAT can be represented as

$$\mathbf{T}^{\mathrm{GAT}} = \mathbf{O}\mathbf{X}\mathbf{W}. \tag{30}$$

The parameter $\mathbf{W}$ can be computed by minimizing the cross-entropy between the given labels and predictions according to (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 $\pi_n = (1/N)$ and $\mathbf{\Sigma}_n = \mathbf{\Sigma}$ for all vertices and set the graph without any self-loops ($a_{nn} = 0$ for all n), as shown in Fig. 3(d).

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

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

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

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

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

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

Equation (31) is equivalent to the graph attention operation in (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 nth rows of $\mathbf{H}$ and $\mathbf{X}$ are $\boldsymbol{\mu}_n$ 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 $\mathbf{\Lambda}^{-(1/2)}\mathbf{U}$ 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 $\mathbf{\Lambda}^{-(1/2)}\mathbf{U}$, where the weight $\mathbf{\Lambda}^{-(1/2)}$ 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 (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

The GMM, which is represented by a graphical model in Fig. 3(e), can be written as a linear combination of the Gaussian distributions in the form

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

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, speaker identification, etc. GMM can also be interpreted as a latent variable model. By introducing an 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

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

where

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

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

C. Comparison With MoNet

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 neighbors via

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

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

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

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

1) 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

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

information (e.g., degree), thus it can be regarded as an extension of GCN while remains differently compared to GAT. In PGCN, the neighborhoods 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.

2) 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 semisupervised 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, that is, Texas, Cornell, Washington, and Wisconsin, are the subnetworks of the WebKB network. Each of them is the collection of webpages from an American university. Similarly, nodes in the 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. The PPI dataset consists of 24 attribute graphs, each of which corresponds to a different human tissue and contains 2373 nodes on 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 two graphs, and tested on two graphs. The 20 graphs for training and two graphs for validation are fully labeled, while the two graphs for testing are unseen during training and validation stages.

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 semisupervised node classification algorithms, including multilayer perceptron (MLP), label propagation (LP) [43], semisupervised embedding (SemiEmb) [44], manifold regularization (ManiReg) [45], graph embedding (DeepWalk) [46], iterative classification algorithm (ICA) [47], graph-based semisupervised learning framework (Planetoid) [42], graph convolution with Chebyshev filters [14], GCN [23], MoNet [21], self-training GCN (ST-GCN) [24], GATs [22], simplified GCN (SGC) [48], and approximated personalized propagation of neural predictions (APPNPs) [49], for comparisons. All baseline methods except DeepWalk are semisupervised 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 five community detection methods, which are all unsupervised methods, on attribute network. Degree-corrected 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 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 seven state-of-the-art algorithms, including random classifier, logistic regression based on node feature without the 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

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**

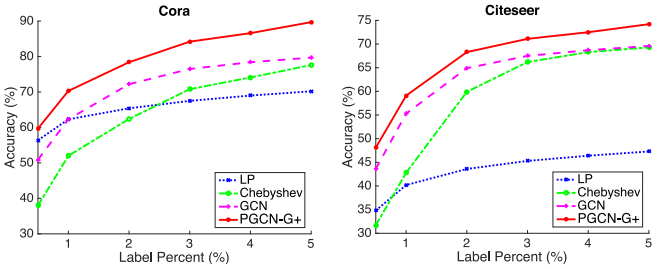


Fig. 7. Node classification results on three datasets with varying percentages of labeled nodes.

evaluation, respectively. As shown in Table II, our PGCN-G+ outperforms all baseline methods. The improvement of PGCN-G+ compared to GAT, which achieves the best performance except for the proposed method, is moderately significant. Although the accuracy improvement is 1.53% on 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 frameworks, such as Tensorflow or Pytorch. For a 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 the training time, validation time, and evaluation time. The average accuracies of PGCN-G+ also outperform all baseline methods. Compared to GAT, which shares the same covariance matrix among all the nodes, the additional runtime of our PGCN-G+ is 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 labeled data, especially limited labeled data. Thus, the performances of four representative methods including PGCN-G+ are provided with varying percentages of labeled nodes. As shown in Fig. 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 the community detection area. In addition to the three citation networks, five webpage networks, which are often adopted to evaluate the community detection methods, are employed. For the four medium webpage networks, including Cornell, Texas, Washington, and Wisconsin, we adopt 20% labeled nodes for training, 10% labeled nodes for validation, and other nodes for testing. For the 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 neighborhood (exploited by GCN) is not widely applicable. Its performances will be 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 microaveraged 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 ten runs for each method. For a comprehensive comparison, both the dataset split strategy in GraphSAGE [19] and the random dataset split is 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

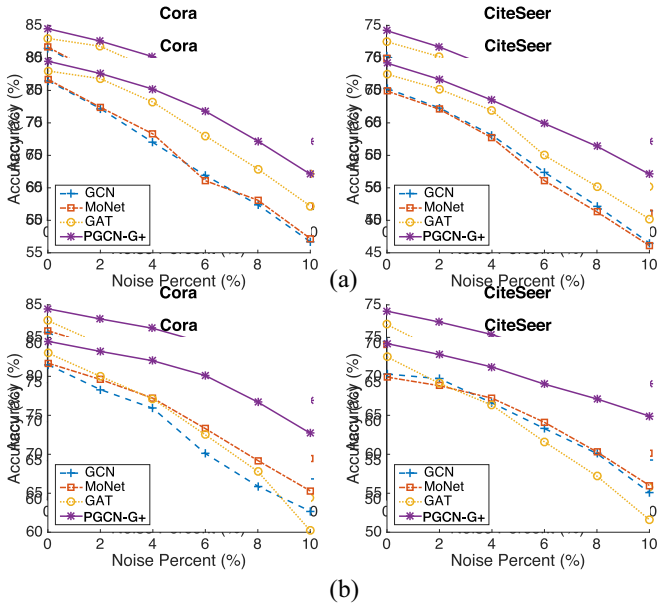


Fig. 8. Noise impacts on performance. Performance with (a) link noises and (b) content noises.

of all nodes with identical covariance as shown in Fig. 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 are 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 Fig. 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 are 3.2% when 2% noises are added, while the improvement is 7.4% when 10% noises are added.

Fig. 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. Fig. 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 because GAT only refines the network topology with respect to the node content. When a certain amount of node content noises are 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. Conclusion and Future Work

In this article, we proposed a top-down latent space framework, PGCN, to provide a probabilistic perspective for the semisupervised node classification problem. PGCN formulated the nodes in a distribution-form representation and modeled 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 applied the general framework to a specific type of the problems, which assumes the existence of Gaussian distribution, and proposed PGCN-G. Then, we improved 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 were the special cases of PGCN-G+, and our proposed framework can elaborate on their characteristics and relationships. Extensive experiments demonstrated 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 Rev.*, vol. 45, no. 2, pp. 167–256, 2003.

[2] W.-N. Chen, D.-Z. Tan, Q. Yang, T. Gu, and J. Zhang, "Ant colony optimization for the control of pollutant spreading on social networks," *IEEE Trans. Cybern.*, early access, Jul. 9, 2019, doi: 10.1109/TCYB.2019.2922266.

[3] W.-N. Chen, Y.-H. Jia, F. Zhao, X.-N. Luo, X.-D. Jia, and J. Zhang, "A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 842–857, Oct. 2019.

[4] T.-F. Zhao, W.-N. Chen, A. W.-C. Liew, T. Gu, X.-K. Wu, and J. Zhang, "A binary particle swarm optimizer with priority planning and hierarchical learning for networked epidemic control," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Oct. 30, 2019, doi: 10.1109/TSMC.2019.2945055.

[5] T. Zhao *et al.*, "Evolutionary divide-and-conquer algorithm for virus spreading control over networks," *IEEE Trans. Cybern.*, early access, Mar. 13, 2020, doi: 10.1109/TCYB.2020.2975530.

[6] X. Wen *et al.*, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 363–377, Jun. 2017.

[7] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, 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 Trans. Knowl. Data Eng.*, early access, Dec. 10, 2019, doi: 10.1109/TKDE.2019.2958806.

[9] D. Jin *et al.*, "Detecting communities with multiplex semantics by distinguishing background, general and specialized topics," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 11, 2019, doi: 10.1109/TKDE.2019.2937298.

[10] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, 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 *Proc. 21st Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2016, pp. 3889–3895.

[12] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, pp. 1106–1114.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Barcelona, Spain, Dec. 2016, pp. 3837–3845.

[15] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2015, pp. 2224–2232.

[16] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, Jun. 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 Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[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 *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI)*, Jan./Feb. 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 *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 1025–1035.

[20] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, London, U.K., Aug. 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 5425–5434.

[22] P. V. Kovià, G. Cucurull, A. Casanova, A. Romero, P. Lià, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–12.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 208–211.

[24] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 22nd AAAI Conf. Artif. Intell. (AAAI)*, Feb. 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 *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 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 Trans. Cybern.*, vol. 50, no. 6, pp. 2475–2487, Apr. 2020.

[27] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised domain adaptive graph convolutional networks," in *Proc. Web Conf. WWW*, Taipei, Taiwan, Apr. 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 *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 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 *Proc. IEEE Int. Conf. Data Min. (ICDM)*, Beijing, China, Nov. 2019, pp. 648–657.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.

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

[32] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2014, pp. 2672–2680.

[33] C. M. Bishop and J. Lasserre, "Generative or discriminative? Getting the best of both worlds," in *Bayesian Statistics 8*. Oxford, U.K.: Oxford Univ. 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," *J. Roy. Stat. Soc. B Methodol.*, vol. 29, no. 1, pp. 1–38, 1977.

[35] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in Wasserstein space," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, London, U.K., Aug. 2018, pp. 2827–2836.

[36] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–6.

[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," *Found. Trends Mach. Learn.*, 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 *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–6.

[40] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, Aug. 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 *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jun. 2016, pp. 40–48.

[43] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. (ICML)*, Aug. 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*. Cham, Switzerland: Springer, 2012, pp. 639–655.

[45] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006.

[46] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Aug. 2014, pp. 701–710.

[47] Q. Lu and L. Getoor, "Link-based classification," in *Proc. 20th Int. Conf. Xon Mach. Learn. (ICML)*, Washington, DC, USA, Aug. 2003, pp. 496–503.

[48] F. Wu, A. H. Souza, Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 6861–6871.

[49] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–6.

[50] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, Jan. 2011, Art. no. 016107.

[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 *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, Feb. 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 *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Paris, France, Jun./Jul. 2009, pp. 927–936.

[53] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large attribute networks," in *Proc. 20th AAAI Conf. Artif. Intell. (AAAI)*, Feb. 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 *Proc. 21st AAAI Conf. Artif. Intell. (AAAI)*, Feb. 2017, pp. 116–124.

[55] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 529–545.

Liang Yang received the B.E. and M.E. degrees in computational mathematics from Nankai University, Tianjin, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2016.

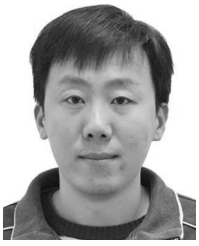
He is an Assistant Professor with the School of Artificial Intelligence, Hebei University of Technology, Tianjin. His current research interests include community detection, graph neural networks, low-rank modeling, and data mining.

Yuanfang Guo (Senior Member, IEEE) received the B.E. degree in computer engineering and the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2009 and 2015, respectively.

He served as an Assistant Professor with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, 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. His research interests include image/video security, compression, processing, and data analysis.

Junhua Gu was born in 1966. He received the Ph.D. degree in applied mathematics from Shanghai Jiao Tong University, Shanghai, China, in 1998.

He is currently working with 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 (Member, IEEE) received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2012.

He was a Postdoctoral Research Fellow with 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, Changchun, China, 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, Beijing, China. 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 the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the *ACM Transactions on the Web*, *Data and Knowledge Engineering*, the *Journal of Autonomous Agents and Multiagent Systems*, and *Knowledge-Based Systems*, and international conferences, including IJCAI, AAAI, ICDM, WI, PAKDD, and ASONAM. His current research interests include data mining, complex/social network modeling and analysis, and multiagent systems.

Prof. Yang has served as an Associate 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, in 1999 and 2002, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA, in 2006.

He spent about three years with ObjectVideo Inc., Tysons, VA, USA, 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, Beijing, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. He has authored and coauthored more than 100 journal and conference papers.

Prof. Cao was a recipient of the Piero Zamperoni Best Student Paper Award at the International Conference on Pattern Recognition in 2004 and 2010. His dissertation was nominated for the University of Central Florida's University-Level Outstanding Dissertation Award. He is on the editorial boards of the IEEE TRANSACTIONS ON IMAGE PROCESSING (Senior Area Editor), the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He is a fellow of IET.