IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 8, 2020, accepted April 22, 2020, date of publication May 4, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992269

Network Embedding Using Deep Robust Nonnegative Matrix Factorization

CHAOBO HE[1], HAI LIU[2], YONG TANG[2], XIANG FEI[3], HANCHAO LI[3], AND QIONG ZHANG[4]

[1]School of Information Science and Technology, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China
[2]School of Computer Science, South China Normal University, Guangzhou 510631, China
[3]Department of Computing, Coventry University, Coventry CV15FB, U.K.
[4]College of Computer Information and Engineering, Nanchang Institute of Technology, Nanchang 330044, China

Corresponding author: Qiong Zhang (qiong.zhang.1@outlook.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772211, in part by the Humanity and Social Science Youth Foundation of Ministry of Education of China under Grant 19YJCZH049, in part by the Natural Science Foundation of Guangdong Province of China under Grant 2019A1515011292, and in part by the Science and Technology Support Program of Guangzhou City of China under Grant 201905010006, Grant 201807010043, and Grant 201803020033.

ABSTRACT As an effective technique to learn low-dimensional node features in complicated network environment, network embedding has become a promising research direction in the field of network analysis. Due to the virtues of better interpretability and flexibility, matrix factorization based methods for network embedding have received increasing attentions. However, most of them are inadequate to learn more complicated hierarchical features hidden in complex networks because of their mechanisms of single-layer factorization structure. Besides, their original feature matrices used for factorization and their robustness against noises also need to be further improved. To solve these problems, we propose a novel network embedding method named DRNMF (deep robust nonnegative matrix factorization), which is formed by multi-layer NMF learning structure. Meanwhile, DRNMF employs the combination of high-order proximity matrices of the network as the original feature matrix for the factorization. To improve the robustness against noises, we use $\ell_{2,1}$ norm to devise the objective function for the DRNMF network embedding model. Effective iterative update rules are derived to resolve the model, and the convergence of these rules are strictly proved. Moreover, we introduce a pre-training strategy to improve the efficiency of convergence. Extensive experiments on several benchmarks of complex networks demonstrate that our proposed method DRNMF is effective and has better performance than the state-of-the-art matrix factorization based methods for network embedding.

INDEX TERMS Network embedding, deep nonnegative matrix factorization, network analysis, complex networks.

I. INTRODUCTION

The complex networks in real world (e.g., online social networks, co-authorship networks and hyperlink networks) often contain much valuable information, which has made network analysis become a hot research topic. A large number of researchers have engaged in studying various tasks of network analysis, such as node classification [1], node clustering [2], link prediction [3], visualization [4], etc. Owing to the fact that complex networks' data are very sparse and high dimensional, these network analysis tasks often suffer from troubles of high computational cost and low performance. To overcome these problems, network embedding has been proposed as an effective technique. This technique, also known as graph embedding or network representation, aims at learning low-dimensional node feature representations in the given network, while preserving structural and inherent properties of the network itself. The representations learnt can be input into analytical tasks as feature vectors. It has been proven by many existing works that better network embedding operations are beneficial to improve the performance of analysis tasks greatly [5]–[7].

As being a promising technical field, network embedding has attracted numerous endeavors on the studies of algorithms and methodologies. From algorithmic perspective, the existing methods for network embedding can be roughly

The associate editor coordinating the review of this manuscript and approving it for publication was Benyun Shi.

summarized into three main categories: deep learning based methods (e.g., SDNE [8] and DNGR [9]), random walk based methods (e.g., DeepWalk [10] and Node2vec [11]) and matrix factorization based methods (e.g., M-NMF [12] and GraRep [13]). The former two categories, especially the deep learning based ones, seem to have become more and more popular. Whereas, matrix factorization based methods have their own distinct advantages: better interpretability, less or no parameters, better flexibility to incorporate prior knowledge, etc. In [14], Qiu *et al.* proved that DeepWalk and Node2vec could be closely unified into the matrix factorization framework. Moreover, in [12] and [13], both M-NMF and GraRep present great competitiveness by comparing with other types of methods. All of these have stimulated great interests from researchers in matrix factorization based network embedding methods.

Recently, some network embedding methods based on matrix factorization have been proposed with improved performance obtained in different works, but they still suffer from the following problems:

- Most of the existing methods are with the mechanism of single-layer mapping from the original network to the final network embedding space, which limits their capabilities to learn more complex and useful features from the network. After all, real-world complex networks usually contain considerably complex hierarchical features, including microscopic node similarities and macroscopic community structures, which are quite difficult to be discovered by using shallow methods.
- Simple adjacent matrix of the network is normally used as the original feature matrix for factorization, nevertheless this feature matrix cannot represent enough local and global structure information of the network, which also reduces the performance of network embedding to some extent.
- The actual complex networks generally contain noises, such as casual followships in online social networks and unreal links in hyperlink networks, but the squared Frobenius norm is typically selected to measure the factorization error in the existing matrix factorization based network embedding methods, which makes them not robust enough against noises.

To solve these problems mentioned above and improve the performance of matrix factorization in network embedding, a novel embedding method called DRNMF for short is proposed. More specifically, the main work can be summarized as follows:

- We devise an embedding approach using deep robust nonnegative matrix factorization (DRNMF), which is made by multi-layer NMF and the combination of high-order proximity matrices as being the original factorization feature matrix. Thus, more informative and discriminative embedding effects can be obtained, meanwhile $\ell_{2,1}$ norm is applied to construct the objective function to improve the robustness against noises.
- We develop the effective iterative update rules to optimize DRNMF model and also prove their convergence. Besides, we introduce the pre-training strategy to expedite their iterative process.
- We evaluate DRNMF model on several benchmark datasets and different analysis tasks, including node classification, clustering and visualization. The results demonstrate that DRNMF is evidently superior over the state-of-the-art matrix factorization based network embedding methods.

The rest of this article is organized as follows. A brief review of related research on NMF, deep NMF (DNMF) and matrix factorization based network embedding is given in Section II. In Section III, the proposed network embedding method DRNMF is presented in detail. Experiments and specific analysis are reported in Section IV. Finally, conclusions are given in Section V.

II. RELATED WORK

In this section, we briefly review the related work regarding nonnegative matrix factorization (NMF), deep NMF (DNMF) and network embedding based on matrix factorization.

A. NMF AND DNMF

NMF is a popular low-rank matrix decomposition model that focuses on the analysis of data matrices whose elements are nonnegative [15]. Mathematically, it can be formulated as: given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}_+^{m\times n}$ composed of n data samples as columns, each with m features, $\mathbf{X}$ can be approximately decomposed into the product of two matrices as $\mathbf{X} \approx \mathbf{WH}$, where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_r] \in \mathbb{R}_+^{m\times r}$ is the basis matrix, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n] \in \mathbb{R}_+^{r\times n}$ is called the coefficient matrix or the encoding matrix, $r \ll \min(m, n)$, and $\mathbb{R}_+$ denotes the set of nonnegative elements. By applying nonnegativity constraints on $\mathbf{W}$ and $\mathbf{H}$, each data sample $\mathbf{x}_i$ can be represented as an additive linear combination of nonnegative basis vectors, which is $\mathbf{x}_i \approx \sum_{j=1}^{r} \mathbf{w}_j h_{ji}$. NMF provides more interpretable parts based decompositions than the other matrix factorization models, because it naturally complies with human intuition of ''combining parts to form a whole''. This characteristic makes NMF widely used in various representation learning tasks, such as image representation [16], [17], microbiome data representation [18], and even network embedding [12], [19], [20].

NMF is essentially a shallow method, which only contains single-layer mapping from $\mathbf{W}$ to $\mathbf{X}$ and $\mathbf{H}$, thus it cannot reveal more complex hierarchical features hidden in complex data objects. Inspired by deep learning, DNMF [21] was proposed to solve the problem left by traditional NMF, with principal idea as stacking single-layer NMF into l ($l > 1$) layers, thereby to obtain hierarchical mappings $(\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_l)$ and corresponding features $(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_l)$. An intuitive representation of this hierarchical factorization and the comparison between NMF and DNMF are respectively described

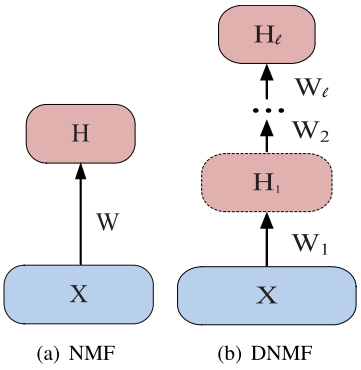


FIGURE 1. NMF vs. DNMF.

in Eq. (1) and Fig. 1.

$$\begin{aligned} \mathbf{X} &\approx \mathbf{W}_1\mathbf{H}_1, \\ \mathbf{H}_1 &\approx \mathbf{W}_2\mathbf{H}_2, \\ &\vdots \\ \mathbf{H}_{l-1} &\approx \mathbf{W}_l\mathbf{H}_l. \end{aligned} \tag{1}$$

Recently, there have already been some successful examples about DNMF on learning hierarchical features from complex data objects. For example, Trigeorgis *et al.* [21] used DNMF to automatically learn low-to-high level feature representations from face data, including pose, expression and identity features. Each representation level is suitable for clustering according to the uncovered corresponding attributes of data, and similar work can be found in [22]. In [23], Song *et al.* applied DNMF to document classification task and found that the process of hierarchical feature learning could eventually lead to a better classification performance. All of the above-mentioned works conducted comparison analysis among DNMF, NMF and other single-layer matrix factorization methods, and the results demonstrate that DNMF has stronger feature learning ability, which motivates us to apply DNMF to network embedding.

It should be noted that those works on DNMF all used Frobenius norm to construct the objective function, which were written like $||\mathbf{X}-\mathbf{W}_1\mathbf{W}_2\ldots\mathbf{W}_l\mathbf{H}_l||_F^2$. Noises with large errors in $\mathbf{X}$ are prone to dominate the objective function in the form of squared errors, which means that DNMF with Frobenius norm is not robust enough against noises. Normally, using $\ell_{2,1}$ norm, robust NMF (RNMF) has been considered to have better robustness against noises, comparing with the regular NMF methods based on Frobenius norm. Therefore, DRNMF is expected to perform better than DNMF, because of the advantage of $\ell_{2,1}$ norm over Frobenius norm, which will be verified in experiments.

B. NETWORK EMBEDDING BASED ON MATRIX FACTORIZATION

Owing that our goal is to boost the performance of matrix factorization for network embedding, here we are only concerned about matrix factorization based methods. Detailed information of other types of methods (e.g., deep learning based methods and random walk based methods) can be found in some survey papers, such as [24], [25] and [26]. In general, the existing matrix factorization based methods for network embedding mainly focus on two aspects: feature matrix construction and dimension reduction, which will be introduced respectively as follows.

The feature matrix of a given network is actually its original high-dimensional representation matrix used for factorization. At early stage, most methods select adjacent matrix as feature matrix, such as SocioDim (Social dimensions) [27] and GF (Graph factorization) [28]. However, many recent literatures have pointed out that high-order proximity matrix should be better to enhance the performance of network embedding than adjacent matrix and other low-order proximity matrices, because high-order proximity matrix is able to convey more local and global structure information, which is very useful to obtain more informative and discriminative embeddings. For example, GreRep [13] used k-step probability transition matrix as the high-order proximity matrix, and experimental results showed that it performs better than other methods using first-order or second-order proximity matrix, such as LINE [29]. In [30], MMDW method further improved the performance by using the average k-step probability transition matrix as the high-order proximity matrix. Similar works can also be found in [31]–[34].

Dimension reduction is used to obtain the final low-dimensional network representations by factorizing the original feature matrix. Factorization strategies vary according to matrix properties. If the obtained feature matrix is positive semi-definite, one can use SVD (singular value decomposition) method such as in GraRep [13] and HOPE [33]. For unstructured feature matrices, one can devise alternative optimization methods to obtain the embedding, such as in M-NMF [12], MMDW [30] and TADW [32], which are generally more efficient than SVD. Due to the complexity of calculating eigenvalues and eigenvectors, the computing process of SVD is very time consuming when encountering large-scale matrices.

Although the aforementioned matrix factorization based methods have achieved performance improvement at different levels, they are all shallow methods and still need to be improved. Our proposed method DRNMF has multi-layer factorization structure. Thus, DRNMF is fundamentally different from them. Moreover, to the best of our knowledge, at present there are still no works about network embedding using multi-layer matrix factorization.

III. METHODOLOGY

In this section, the proposed method DRNMF is described, starting from introducing the statement of problem. Then, DRNMF is presented in detail, including network embedding model, optimization solution, convergence proof and the embedding algorithm.

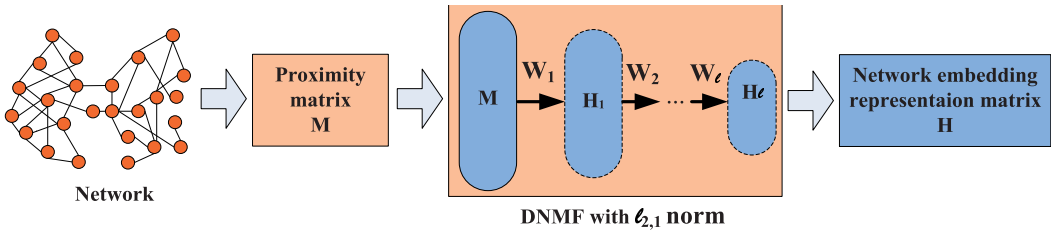


FIGURE 2. The framework of DRNMF network embedding model.

A. STATEMENT OF THE PROBLEM

Throughout this paper, matrices are denoted by bold uppercase letters. For a given matrix $\mathbf{X}$, its i-th column vector, (i, j)-th element, trace, and Frobenius norm are denoted by $\mathbf{x}_i$, x_{ij}, $tr(\mathbf{X})$ and $||\mathbf{A}||_F$, respectively. Meanwhile, the identity matrix is denoted by $\mathbf{I}$.

Without loss of generality, a given network can be formally represented as a directed and unweighted graph as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes, and $E = \{e_{ij} | v_i \in V \wedge v_j \in V\}$ is the directed edge set. The adjacent matrix is denoted as $\mathbf{A} = [a_{ij}]^{n\times n}$. If $e_{ij} \in E$, then $a_{ij} = 1$, else $a_{ij} = 0$. Following the idea presented in [13] and [29], we define k-order proximity matrix of the network as:

$$\mathbf{S}^k = \underbrace{\mathbf{S} \cdot \mathbf{S} \ldots \mathbf{S}}_{k}, \tag{2}$$

where $\mathbf{S} = [s_{ij}]^{n\times n}$ is also called as the first-order proximity matrix with $s_{ij} = \frac{a_{ij}}{\sum_{j=1}^{n} a_{ij}}$.

Thus, the problem of network embedding here can be formally stated as: given a network G with matrices $\mathbf{S}, \mathbf{S}^2, \ldots, \mathbf{S}^k$, it is aimed at learning a low-dimensional representation matrix $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n] \in \mathbb{R}_+^{r\times n}$ using DRNMF, where $\mathbf{h}_i$ is the r-dimensional feature representation of node v_i with $r \ll n$.

B. NETWORK EMBEDDING MODEL

The framework of DRNMF network embedding model can be depicted as in Fig. 2. As we can see, this model is comprised of two key components: proximity matrix construction and DNMF with $\ell_{2,1}$ norm, which can be described as follows.

1) PROXIMITY MATRIX CONSTRUCTION

Motivated by the work about the equivalence of DeepWalk and matrix factorization presented in [32], we select the mean value of all the k-order proximity matrices as the proximity matrix $\mathbf{M}$:

$$\mathbf{M} = \frac{\mathbf{S} + \mathbf{S}^2 + \ldots + \mathbf{S}^k}{k}. \tag{3}$$

In Eq. (3), $\mathbf{M}$ combines multiple high-order proximities, so it can be expected to capture more local and global structure features. In [32], k is advised to be set as 2 considering the balance between the computational speed and accuracy. Here our method DRNMF will also follow this suggestion.

2) DNMF WITH $\ell_{2,1}$ NORM

After obtaining the proximity matrix $\mathbf{M}$, we factorize $\mathbf{M}$ using DNMF with $\ell_{2,1}$ norm to produce an l-layer hierarchical feature representations for the original network: $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_l$, where $\mathbf{H}_i$ $(1 \leq i \leq l)$ denotes feature representation at the i-th level. Besides, the dimensionality of $\mathbf{H}_i$ will become much smaller along with the layer number increase, which implies a more abstract and more compact representation of the network. Similar to some other feature learning methods based on deep neural networks, this presented deep structure could also be expected to lead to more accurate network representation results, i.e., a better $\mathbf{H}_l$. In order to learn $\mathbf{H}_l$ and the other factor matrices (e.g., $\mathbf{W}_i$), we derive the following objective function using $\ell_{2,1}$ norm:

$$\begin{aligned} &\min \mathcal{J}(\mathbf{W}_i, \mathbf{H}_l) = ||\mathbf{M} - \mathbf{W}_1\mathbf{W}_2 \ldots \mathbf{W}_l\mathbf{H}_l||_{2,1}, \\ &s.t.\ \mathbf{H}_l \geq 0, \mathbf{W}_i \geq 0, \quad \forall i = 1, 2, \ldots, l, \end{aligned} \tag{4}$$

where $\mathbf{H}_l \in \mathbb{R}_+^{r\times n}$ is treated as the final network embedding representation matrix $\mathbf{H}$ and $\mathbf{W}_i \in \mathbb{R}_+^{d_{i-1}\times d_i}$ $(1 \leq i \leq l)$ is set as $n = d_0 > d_1 > \ldots > d_{l-1} > d_l = r$.

C. OPTIMIZATION SOLUTION

The minimization of the objective function in Eq. (4) is a typical constraint optimization problem and we can solve it by using alternating minimization strategy. Namely, all the variables can be fixed first at each iteration except for one unfixed to be updated. Through repeating these updating processes until the objective function achieve convergence, the final optimized results can be obtained. Next, we will present specific update rules for factor matrices $\mathbf{W}_i$ $(1 \leq i \leq l)$ and $\mathbf{H}_l$.

1) UPDATE RULE FOR $\mathbf{W}_i$

By fixing all the variables except for $\mathbf{W}_i$, the objective function in Eq. (4) is simplified to:

$$\begin{aligned} &\min \mathcal{J}(\mathbf{W}_i) = ||\mathbf{M} - \mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l||_{2,1}, \\ &s.t.\ \mathbf{W}_i \geq 0, \quad \forall i = 1, 2, \ldots, l, \end{aligned} \tag{5}$$

where $\mathbf{P}_{i-1} = \mathbf{W}_1\mathbf{W}_2 \ldots \mathbf{W}_{i-1}$ and $\mathbf{Q}_{i+1} = \mathbf{W}_{i+1}\mathbf{W}_{i+2} \ldots \mathbf{W}_l$. When $i = 1$, we set $\mathbf{P}_0 = \mathbf{I}$. Similarly, when $i = l$, we set $\mathbf{Q}_{l+1} = \mathbf{I}$.

To solve Eq. (5), we can firstly use the Lagrange multiplier method to devise the Lagrange function of $\mathcal{J}(\mathbf{W}_i)$:

$$\mathcal{L}(\mathbf{W}_i) = ||\mathbf{M} - \mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l||_{2,1} - tr(\Phi_i\mathbf{W}_i^{\mathbf{T}}), \tag{6}$$

where Φ_i is the Lagrange multiplier to $\mathbf{W}_i$. Then, $\mathcal{L}(\mathbf{W}_i)$ can be rewritten in the form of matrix traces as

$$\begin{aligned}\mathcal{L}(\mathbf{W}_i) &= tr((\mathbf{M}-\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l)\\ &\quad \mathbf{D}_i(\mathbf{M}-\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l)^{\mathbf{T}}) - tr(\Phi_i\mathbf{W}_i^{\mathbf{T}})\\ &= tr(\mathbf{M}\mathbf{D}_i\mathbf{M}^{\mathbf{T}}) - 2tr(\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{M}^{\mathbf{T}})\\ &\quad +tr(\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}}\mathbf{W}_i^{\mathbf{T}}\mathbf{P}_{i-1}^{\mathbf{T}})\\ &\quad -tr(\Phi_i\mathbf{W}_i^{\mathbf{T}}),\end{aligned} \tag{7}$$

where $\mathbf{D}_i = [d_{jj}]^{n\times n}$ is a diagonal matrix and $d_{jj} = \frac{1}{||\mathbf{m}_j-\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}(\mathbf{H}_l)_j||}$. The partial derivative of $\mathcal{L}(\mathbf{W}_i)$ with respect to $\mathbf{W}_i$ is:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{W}_i)}{\partial\mathbf{W}_i} &= -\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{M}\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}}\\ &\quad +\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}} - \Phi_i.\end{aligned} \tag{8}$$

Using Karush-Kuhn-Tucker (KKT) conditions [35], we have

$$\begin{aligned}(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}} - \mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{M}\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})\\ \odot\mathbf{W}_i = 0,\end{aligned} \tag{9}$$

where $\odot$ denotes element-wise product. Finally, by solving Eq. (9), we can obtain the following update rule for $\mathbf{W}_i$:

$$(\mathbf{W}_i)_{ab} = (\mathbf{W}_i)_{ab}\frac{(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{M}\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}}{(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}}. \tag{10}$$

2) UPDATE RULE FOR $\mathbf{H}_l$

Similarly, through fixing all the variables except for $\mathbf{H}_l$, the objective function in Eq. (4) is simplified to:

$$\min\mathcal{J}(\mathbf{H}_l) = ||\mathbf{M}-\mathbf{P}_l\mathbf{H}_l||_{2,1}, \quad s.t. \quad \mathbf{H}_l \geq 0. \tag{11}$$

Following the rule of Lagrange multiplier method presented above, we can also obtain the update rule for $\mathbf{H}_l$ as follows:

$$\mathbf{H}_l = \mathbf{H}_l \odot \frac{\mathbf{P}_l^{\mathbf{T}}\mathbf{M}\mathbf{D}_l}{\mathbf{P}_l^{\mathbf{T}}\mathbf{P}_l\mathbf{H}_l\mathbf{D}_l}. \tag{12}$$

D. CONVERGENCE PROOF

In this section, the convergence of the update rules shown in Eq. (10) and Eq. (12) are proved according to the following theorems.

Theorem 1: Updating $\mathbf{W}_i$ using the rule of Eq. (10) while fixing all the variables except for $\mathbf{W}_i$, the objective function $\mathcal{J}(\mathbf{W}_i)$ in Eq. (5) monotonically decreases to obtain the minimum.

Theorem 2: Updating $\mathbf{H}_l$ using the rule of Eq. (12) while fixing all the variables except for $\mathbf{H}_l$, the objective function $\mathcal{J}(\mathbf{H}_l)$ in Eq. (11) monotonically decreases to obtain the minimum.

The proof of convergence for $\mathbf{W}_i$ is similar to that for $\mathbf{H}_l$, thus we only focus on $\mathbf{W}_i$ here (i.e., the proof of Theorem 1). To prove Theorem 1, we need to employ the following lemma.

Lemma 1: Given nonnegative matrices $\mathbf{Y}$, $\mathbf{N}$ and $\mathbf{U}$, where $\mathbf{Y} = \mathbf{Y}^{\mathbf{T}}$ and $\mathbf{N} = \mathbf{N}^{\mathbf{T}}$, we have

$$tr(\mathbf{U}^{\mathbf{T}}\mathbf{Y}\mathbf{U}\mathbf{N}) \leq \sum_{a,b}(\mathbf{Y}\mathbf{U}'\mathbf{N})_{ab}\frac{\mathbf{U}_{ab}^2}{\mathbf{U}'_{ab}}, \tag{13}$$

and the equality holds when $\mathbf{U} = \mathbf{U}'$. The detailed proof of this Lemma can be found in [36].

Next, we select the widely used the auxiliary function approach proposed in [37] to prove theorem 1. Firstly, according to Lemma 1, we can obtain

$$\begin{aligned}\mathcal{J}(\mathbf{W}_i) &= tr((\mathbf{M}-\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l)\\ &\quad \mathbf{D}_i(\mathbf{M}-\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l)^{\mathbf{T}})\\ &= tr(\mathbf{M}\mathbf{D}_i\mathbf{M}^{\mathbf{T}}) - 2tr(\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{M}^{\mathbf{T}})\\ &\quad +tr(\mathbf{W}_i^{\mathbf{T}}\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})\\ &\leq tr(\mathbf{M}\mathbf{D}_i\mathbf{M}^{\mathbf{T}}) - 2tr(\mathbf{P}_{i-1}\mathbf{W}_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{M}^{\mathbf{T}})\\ &\quad +\sum_{a,b}((\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}'_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}\\ &\quad \frac{(\mathbf{W}_i)_{ab}^2}{(\mathbf{W}'_i)_{ab}}) = Z(\mathbf{W}_i,\mathbf{W}'_i).\end{aligned} \tag{14}$$

The equality holds when $\mathbf{W}_i = \mathbf{W}'_i$, hence $Z(\mathbf{W}_i, \mathbf{W}'_i)$ is an auxiliary function of $\mathcal{J}(\mathbf{W}_i)$. The first order and second order derivatives of $Z(\mathbf{W}_i, \mathbf{W}'_i)$ with respect to $\mathbf{W}_i$ are as follows:

$$\begin{aligned}&\frac{\partial Z(\mathbf{W}_i,\mathbf{W}'_i)}{\partial(\mathbf{W}_i)_{ab}}\\ &= -2(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{M}\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}\\ &\quad +\frac{2(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}'_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}(\mathbf{W}_i)_{ab}}{(\mathbf{W}'_i)_{ab}},\end{aligned} \tag{15}$$

$$\begin{aligned}&\frac{\partial Z^2(\mathbf{W}_i,\mathbf{W}'_i)}{\partial(\mathbf{W}_i)_{ab}\partial(\mathbf{W}_i)_{ab}}\\ &= \frac{2(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}'_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}}{(\mathbf{W}'_i)_{ab}}.\end{aligned} \tag{16}$$

By reason that each matrix involved is nonnegative, we have $\frac{\partial^2 Z(\mathbf{W}_i,\mathbf{W}'_i)}{\partial(\mathbf{W}_i)_{ab}\partial(\mathbf{W}_i)_{ab}} \geq 0$, which makes the Hessian matrix of $Z(\mathbf{W}_i, \mathbf{W}'_i)$ be positive semi-definite. This indicates that $Z(\mathbf{W}_i, \mathbf{W}'_i)$ is a convex function whose global minimum can be obtained by setting the gradient of $Z(\mathbf{W}_i, \mathbf{W}'_i)$ to 0 and solving it for $\mathbf{W}_i$. To this end, we set Eq. (15) to 0 and can obtain

$$(\mathbf{W}_i)_{ab} = (\mathbf{W}_i)'_{ab}\frac{(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{M}\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}}{(\mathbf{P}_{i-1}^{\mathbf{T}}\mathbf{P}_{i-1}\mathbf{W}'_i\mathbf{Q}_{i+1}\mathbf{H}_l\mathbf{D}_i\mathbf{H}_l^{\mathbf{T}}\mathbf{Q}_{i+1}^{\mathbf{T}})_{ab}}. \tag{17}$$

Obviously, Eq. (17) is the same to Eq. (10). Because $Z(\mathbf{W}_i, \mathbf{W}'_i)$ is the auxiliary function of $\mathcal{J}(\mathbf{W}_i)$, Eq. (17) can also make $\mathcal{J}(\mathbf{W}_i)$ converge to obtain the minimum. Therefore, Theorem 1 holds.

E. NETWORK EMBEDDING ALGORITHM

After repeating update rules shown in Eq. (10) and Eq. (12) until convergence, the final $\mathbf{H}_l$ is the result of network embedding. To expedite the iterative process, we pre-train each layer

in DRNMF model to attain an initial approximation to the factor matrices $\mathbf{W}_i$ and $\mathbf{H}_i$. The effectiveness of pre-training will be proven in the experimental part. To perform the pre-training, we first decompose $\mathbf{M} \approx \mathbf{W}_1\mathbf{H}_1$ by using robust NMF (RNMF), i.e., minimizing $||\mathbf{M} - \mathbf{W}_1\mathbf{H}_1||_{2,1}$. Then, we decompose $\mathbf{H}_1 \approx \mathbf{W}_2\mathbf{H}_2$ by minimizing $||\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||_{2,1}$. Doing the decomposition step by step, we can finish the pre-training work for all the layers.

Afterwards, each layer is fine-tuned by using iterative update rules to minimize the proposed objective function described in Eq. (4). Based on the above knowledge, we devise DRNMF network embedding algorithm composed of two stages: pre-training and fine-tuning. The entire algorithmic framework is shown in Algorithm 1.

Algorithm 1 DRNMF Network Embedding Algorithm

Input: $\mathbf{A}, l, r, d_i\ (1 \le i \le l)$;
Output: Network embedding representation matrix $\mathbf{H}$;
%Pre-training stage%
Compute the proximity matrix $\mathbf{M}$;
$\mathbf{W}_1, \mathbf{H}_1$ =RNMF($\mathbf{M}, d_1$);
for $i = 2$ ***to*** l **do**
  $\mathbf{W}_i, \mathbf{H}_i$ =RNMF($\mathbf{H}_{i-1}, d_i$);
%Fine-tuning stage%
while *not converged* **do**
  for $i = 1$ ***to*** l **do**

$$\mathbf{P}_{i-1} = \begin{cases} \mathbf{I} & \text{if } i = 1, \\ \prod_{y=1}^{i-1} \mathbf{W}_y & \text{otherwise;} \end{cases}$$

$$\mathbf{Q}_{i+1} = \begin{cases} \mathbf{I} & \text{if } i = l, \\ \prod_{y=i+1}^{l} \mathbf{W}_y & \text{otherwise;} \end{cases}$$

    Compute the diagonal matrix $\mathbf{D}_i$;
    Update $\mathbf{W}_i$ via Eq. (10);
  $\mathbf{P}_l = \mathbf{P}_{i-1}\mathbf{W}_i$;
  Update $\mathbf{H}_l$ via Eq. (12);
return $\mathbf{H}_l$;

Let d denote the maximal layer size out of all layers, t_p and t_f respectively denote the number of iterations to achieve convergence in pre-training stage and in fine-tuning stage, we can approximately analyze the time complexity of Algorithm 1. In the pre-training stage, the time complexity of computing the proximity matrix $\mathbf{M}$ is $\mathcal{O}(n^2)$ when we set $k = 2$ by following the suggestion presented in [32], and that for RNMF process is $\mathcal{O}(lt_p(nd^2 + n^2d) + t_pn^2d))$. We can deduce the time complexity for the pre-training stage to be $\mathcal{O}(lt_p(nd^2 + n^2d))$. In the fine-tuning stage, the time complexity for computing $\mathbf{P}_{i-1}$, $\mathbf{D}_i$ and $\mathbf{P}_l$ are all $\mathcal{O}(nd^2)$, for computing $\mathbf{Q}_{i+1}$ is $\mathcal{O}(d^3)$, for updating $\mathbf{W}_i$ and $\mathbf{H}_l$ are both $\mathcal{O}(n^2d)$. Therefore, the time complexity in the fine-tuning stage is $\mathcal{O}(lt_f(nd^2 + n^2d + d^3) + t_f(nd^2 + n^2d))$, which can be simplified to be $\mathcal{O}(lt_f(nd^2 + n^2d)$ because of the general condition $d \ll n$. To sum up, the overall time complexity of Algorithm 1 is $\mathcal{O}(l(t_p + t_f)(nd^2 + n^2d))$, which has the same order of magnitude as many state-of-the-art matrix factorization based network embedding methods, such as M-NMF [12] and GraRep [13]. Owing to the reason that $\mathbf{W}_i$ and $\mathbf{H}_l$ are very sparse, the practical time complexity can be reduced significantly if only the non-zero entries of matrices involved are computed.

IV. EXPERIMENTAL STUDY

In this section, the effectiveness of our proposed method DRNMF is evaluated. First, we provide an overview of experimental datasets, baseline methods and parameter settings. Then we conduct detailed comparative analysis with baseline methods on three network analysis tasks, including node classification, node clustering and visualization. Finally, we validate the robustness of DRNMF and the effectiveness of pre-training strategy. All of the experiments are conducted on a PC with 64-bits Windows-7 system, 3.4GHz Intel i7-6700 CPU and 32GB RAM.

A. DATASETS

In our experiments, we use the following four widely-used benchmark complex networks as the datasets:

- Political blog network (Polblog).[1] Polblog is a hyperlink network of websites, which is composed of 1224 blogs about US politics and 19025 hyperlinks related to them. Each blog is associated with a political label: liberal or conservative.
- Citeseer.[2] Citeseer is a citation network of academic papers from Citeseer digital library. It contains 3312 nodes (papers) and 4732 edges (citation links among papers). Each node has one category label indicating its topic.
- Cora[2]. Cora is also a citation network of academic papers. It consists of 2708 nodes and 5429 edges, and every node is assigned with a unique topic label.
- BlogCatalog[2]. BlogCatalog is an online social network between bloggers. It contains 10312 bloggers and 333983 friendships between them. Every blogger is assigned a group label indicating its topic interest. Although some bloggers have multiple labels, we only keep their foremost labels for the convenience of comparisons.

These four datasets all contain complex hierarchical features, and thus are suitable to be used to validate the effectiveness of DRNMF. Besides, on these network datasets, the label associated with each node can also be treated as the cluster label indicating its group/cluster membership. Therefore, these network datasets can support us to conduct performance evaluation tasks for node classification and node clustering at the same time. Detailed statistics for the datasets

[1] http://www-personal.umich.edu/~mejn/netdata
[2] https://linqs.soe.ucsc.edu/data

TABLE 1. Statistics of datasets.

Dataset	$\|V\|$	$\|E\|$	$\#labels$
Polblog	1224	19025	2
Citeseer	3312	4732	6
Cora	2708	5429	7
BlogCatalog	10312	333983	39

are summarized in Table 1, where #*labels* denotes the number of class labels.

B. BASELINE METHODS

By reason that the motivation of this paper is to improve the performance of matrix factorization for network embedding by using DRNMF, we specially select 3 state-of-the-art matrix factorization based methods for network embedding as baselines, including M-NMF [12], DNMF [21] and DANMF [38]. Besides, we also take into consideration the comparisions with another two classical network embedding methods, DeepWalk [10] and Node2vec [11], which are closely related to matrix factorization. Therefore, we have 5 baseline methods in total, which are respectively introduced as follows:

- M-NMF. M-NMF is based on modularized nonnegative matrix factorization and can incorporate community structure into network embedding. It uses the combination of first-order and second-order proximity matrices as the feature matrix for factorization, which belongs to single-layer matrix factorization method.
- DNMF. DNMF is the first method that formally introducing deep learning concept into the NMF model. Although it is used to learn attribute representations of images, its multi-layer matrix factorization structure can also help it learn complex hierarchical features hidden in the network. Here, we select the adjacent matrix as its feature matrix and use Frobenius norm to devise its cost function.
- DANMF. DANMF is a deep autoencoder-like NMF method. It is used in community discovery, but it also can be used in network embedding by naturally treating its community membership matrix as the network embedding representation matrix. Like DNMF, DANMF also uses the adjacent matrix as being feature matrix and uses the cost function based on Frobenius norm.
- DeepWalk. DeepWalk is a classical network embedding method. It first transforms a network into linear sequences by truncated random walk and then uses Skip-gram natural language model to obtain node representations. In [14], related theoretical analysis and proofs show that DeepWalk empirically produces a low-rank transformation of a network's normalized Laplacian matrix.
- Node2vec. Similar to DeepWalk, Node2vec is also on the basis of random walk and can learn richer node representations by exploring diverse node neighborhoods. Theoretically, Node2vec is regarded as factorizing a matrix related to stationary distribution and transition probability tensor of a 2nd-order random walk, which has also been proved in [14].

TABLE 2. Configuration of layers for DNMF, DANMF and DRNMF.

Dataset	Configuration of Layers
Polblog	1224-256-128-20
Citeseer	3312-512-256-60
Cora	2708-512-256-70
BlogCatalog	10312-2048-1024-390

C. PARAMETER SETTINGS

For fair comparisons, parameters in all the methods are tuned to be optimal or set to be their suggested values. For M-NMF, their regularizer parameters α and β are respectively set to be 0.5 and 5. As suggested in [10], for DeepWalk, we set walks per vertex $\gamma = 80$, window size $\omega = 10$ and walk length $t = 40$. For Node2vec, we use hyperparameter settings with $\gamma = 10$, $\omega = 10$ and $t = 80$. We employ a grid search over return parameter and in-out parameter $p, q \in \{0.25, 0.5, 1, 2\}$ for training. For GraRep, we set $k = 3$ on Polblog, Citeseer and Cora datasets, with $k = 6$ on BlogCatalog dataset. For DANMF, we set the graph regularizer parameter λ to be 1, which was suggested in [38]. It should be noted that all these methods use the same dimension of representation as $r = 10 * (\#labels)$ on the same dataset. For the sake of fairness, three methods with multi-layer learning structure (i.e., DNMF, DANMF and DRNMF) have the same layer configuration on the same dataset, which is shown in Table 2. Although we have tried to set more layers, the performance promotion is not significant while spending much more computation time. Under each setting of parameters for different methods, the experiments are repeated for 10 times and the average results are reported here.

D. COMPARISONS ON NETWORK ANALYSIS TASKS

1) NODE CLASSIFICATION

Node classification involves using the representations generated by network embedding methods to classify each given node into the category it belongs to. Here, the representation of each node is used as its feature vector and the label it associates with is treated as the true class label. In the experiments, we use support vector machine (SVM) implemented by Weka[3] as the classifier. For every dataset, when training the classifier, we randomly sample 10% to 90% of the labeled nodes as training data and the rest as test data. Since each node in each dataset has only one class label, we can simply use classification accuracy (i.e., the proportion of correctly classified nodes) as the performance metric. The evaluation results on each dataset are shown in Fig. 3, from which we have the following observations and analysis:

[3]https://www.cs.waikato.ac.nz/ml/weka

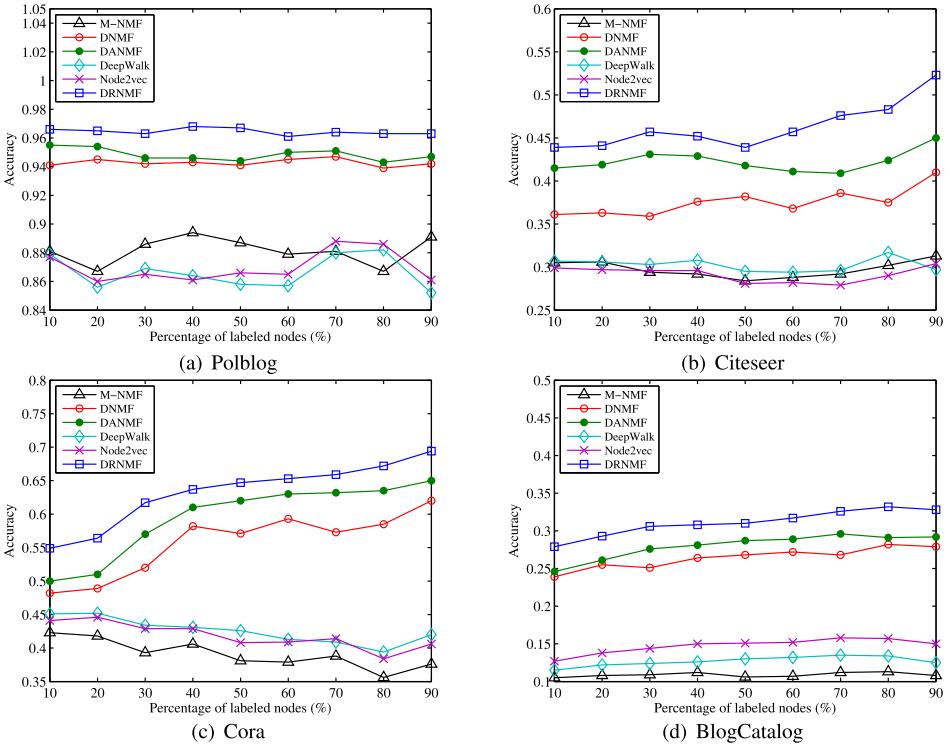


FIGURE 3. Node classification results on different datasets.

- The curve of our method DRNMF is consistently above the curves of baseline methods, which means that node representations learned by DRNMF are more informative and discriminative, thus the performance of node classification can be improved much better.
- Compared with shallow methods M-NMF, DeepWalk and Node2vec, multi-layer matrix factorization based methods DNMF, DRNMF and DANMF perform better, which demonstrates that the multi-layer factorization structure is indeed able to obtain better node representations from the process of learning hierarchical features of the original networks.
- Although DNMF, DRNMF and DANMF have multi-layer factorization structures, DRNMF performs better than DNMF and DANMF. The reasons might be concluded in two aspects. First, DRNMF uses the combination of high-order proximity matrices as the original feature matrix, which contains richer information about network structure than the adjacent matrix used by DNMF and DANMF. Second, DRNMF employs $\ell_{2,1}$ norm, which makes it more robust against noises existing in networks than DNMF and DANMF using Frobenius norm. It can be noted that the first reason undoubtedly plays a more important role, because network datasets used here almost have no noises. In Section IV.E, we will specially test the robustness of DRNMF by manually adding noises to datasets.

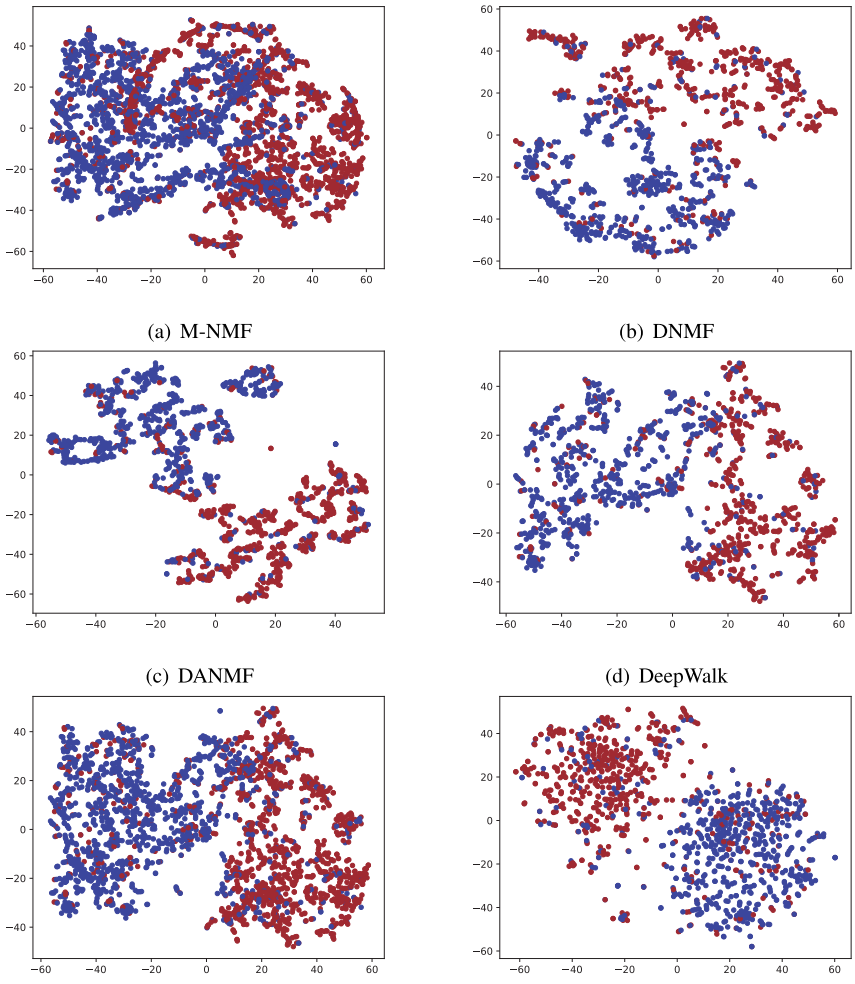
2) NODE CLUSTERING

Due to the fact that node labels of network datasets used here can also be treated as cluster labels, the performance evaluation for node clustering can thereby be conducted. To perform node clustering, we obtain node representations through using network embedding method at first, and then apply K-means algorithm to these learnt node representations to attain clusters. As each node has ground-truth cluster membership, we use the widely-used Purity and NMI (Normalized mutual information) [39] as metrics to measure the performance. The larger the Purity and NMI values are, the better the performance of node clustering will be. We run every method on four datasets and the evaluation results are shown in Table 3.

As is shown from Table 3, on Polblog dataset, Purity and NMI are of equal value as 1.0 only on DRNMF method. Besides, on every dataset, DRNMF is the only approach with all the Purity values larger than 0.9. In terms of performance improvement, compared with M-NMF, DNMF, DANMF, DeepWalk and Node2vec, on Citeseer dataset, the Purity value of DRNMF improves by 38%, 13.9%, 11.4%, 28.9% and 25.6% respectively and the NMI value improves by 44.8%, 22.8%, 16.9%, 36.6% and 32.9% respectively. Similar results can also be found on Cora and BlogCatalog datasets. All the results demonstrate that DRNMF performs the best in terms of node clustering, and it even has considerable advantages compared with the baseline methods.

TABLE 3. Performance evaluation results on node clustering (bold numbers represent the best results).

Dataset	Purity						NMI					
	M-NMF	DNMF	DANMF	DeepWalk	Node2vec	DRNMF	M-NMF	DNMF	DANMF	DeepWalk	Node2vec	DRNMF
Polblog	0.90	0.97	0.99	0.95	0.94	**1.0**	0.71	0.88	0.94	0.75	0.72	**1.0**
Citeseer	0.71	0.86	0.88	0.76	0.78	**0.98**	0.67	0.79	0.83	0.71	0.73	**0.97**
Cora	0.68	0.85	0.87	0.80	0.77	**0.93**	0.59	0.84	0.85	0.74	0.63	**0.91**
BlogCatalog	0.42	0.80	0.83	0.77	0.72	**0.91**	0.36	0.47	0.49	0.46	0.43	**0.67**



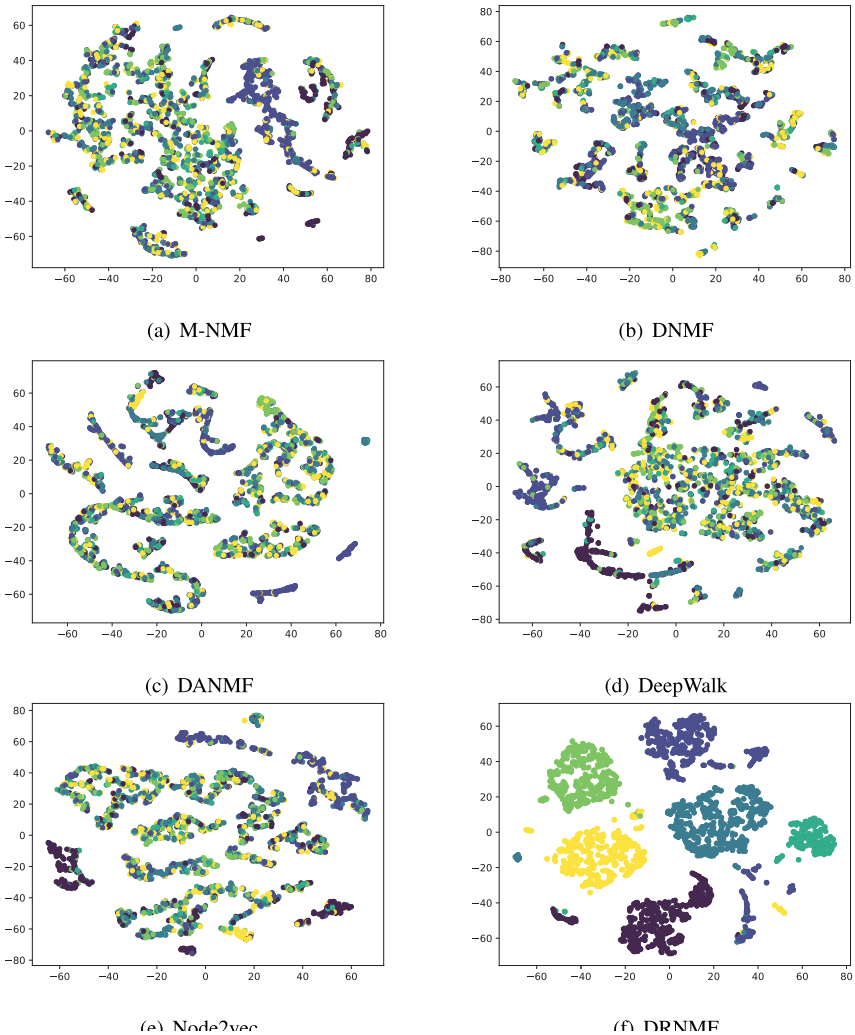
(a) M-NMF (b) DNMF (c) DANMF (d) DeepWalk (e) Node2vec (f) DRNMF

FIGURE 4. Visualization results on Polblog.

3) VISUALIZATION

Visualization is another important application for network embedding issues. It aims to display the given network in a two-dimensional space. If the learned node representations are more discriminative, the visualization results can display clearer boundaries between nodes with different labels. This also indicates the corresponding network embedding method performs better.

In our experiments, we use node representations learned from different network embedding methods as the input to the t-SNE visualization tool [4], where the nodes with the same label are highlighted with the same color. As representatives, the visualization results on Polblog and Citeseer datasets are given respectively as shown in Fig. 4 to Fig. 5. From Fig. 4, it can be seen that the visualization results of M-NMF, DNMF, DANMF, DeepWalk and Node2vec are not satisfactory, because lots of nodes with different labels are mixed with each other and the boundaries of different groups are not clear. For DRNMF, nodes with the same label aggregate together with clear boundary from the other cluster. Moreover, the number of clusters seems to be consistent with the number of ground truth's. Similar observations can also be found in Fig. 5. In general, the representations learned by DRNMF are more identifiable, which makes the



(a) M-NMF (b) DNMF

(c) DANMF (d) DeepWalk

(e) Node2vec (f) DRNMF

FIGURE 5. Visualization results on Citeseer.

visualization results of DRNMF perform better than the baseline methods.

E. ROBUSTNESS TEST

DRNMF network embedding model uses $\ell_{2,1}$ norm instead of the widely used Frobenius norm to devise its objective function, which can be expected to improve the model's robustness against noises. To validate the effectiveness of this mechanism, we specially conduct comparative experiments between DRNMF using $\ell_{2,1}$ norm (DRNMF_L21) and DRNMF using Frobenius norm (DRNMF_Fro). In our experiments, for each dataset, we first define a so-called cannot-link pairwise constraints which include pairs of nodes with different labels. The possible numbers ($\#CL$) of cannot-link pairwise constraints on a network with ground-truth classification results set $C = \{c_1, c_2, \ldots, c_{\#labels}\}$ can be denoted as:

$$\#CL = \sum_{i=1}^{\#labels} \sum_{j=i+1}^{\#labels} |c_i||c_j|. \tag{18}$$

Then, we randomly extract the fixed percentage values (including 6 levels: 0%, 1%, 2%, 4%, 6% and 8%) of cannot-link pairwise constraints to establish virtual links, which can be regarded as noises. Finally, we evaluate the performances of DRNMF_L21 and DRNMF_Fro on node classification tasks under different noise levels. Note that on each dataset, we only utilize 10% labeled nodes for the convenience of comparisons. The results on different datasets are shown in Fig. 6.

As we can see from Fig. 6, on each dataset, the performance degradation of DRNMF_L21 is much smaller than that of DRNMF_Fro with the percentage increase of noises. For example, when the noise rate is 8% on BlogCatalog dataset, the accuracy value of DRNMF_L21 is 0.21, which is 25.8% lower than that without noises, but the accuracy value of DRNMF_Fro is 0.13, which is 46.8% lower than that without noises. Similar results can also be obtained on the other three datasets. All the results illustrate that DRNMF_L21 has better robustness than DRNMF_Fro. This makes DRNMF with $\ell_{2,1}$ norm more suitable to be applied to real-world complex networks that contain noises in most cases.

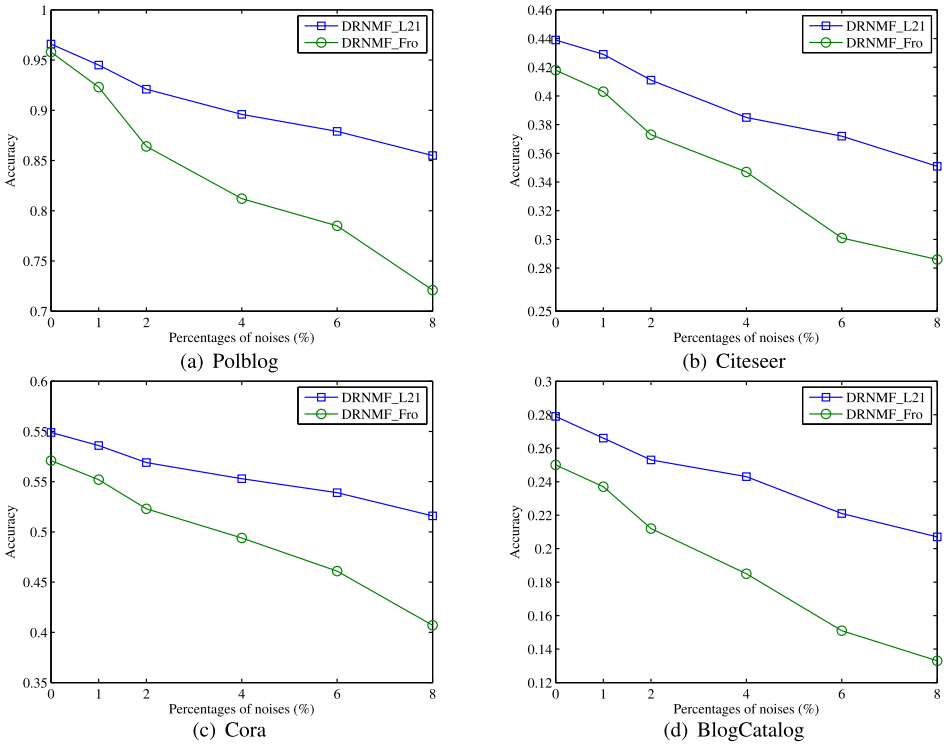


FIGURE 6. Comparison of robustness on node classification.

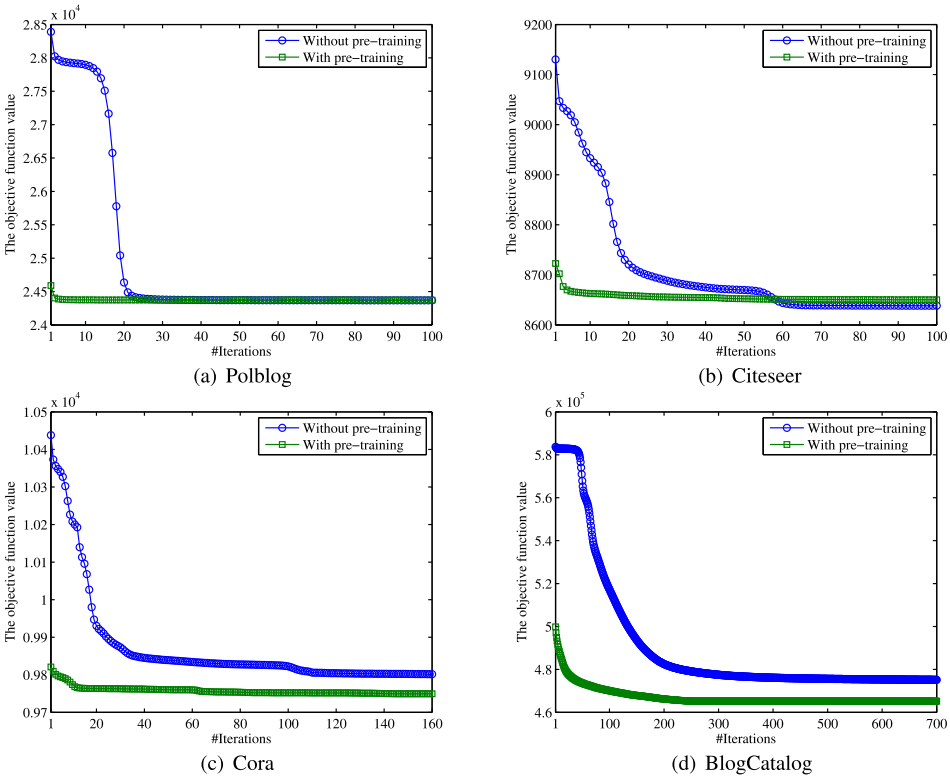


FIGURE 7. Comparison of convergence efficiency (#Iterations).

F. CONVERGENCE EFFICIENCY ANALYSIS USING PRE-TRAINING

In the proposed DRNMF network embedding algorithm (i.e., Algorithm 1), the purpose of pre-training stage is to expedite the iterative process in the subsequent fine-tuning stage. To validate the effectiveness of this strategy, we conduct convergence efficiency comparisons in the fine-tuning stage by using DRNMF with pre-training and DRNMF

TABLE 4. Comparisons of convergence efficiency (Time).

Dataset	Time (second)	
	Without pre-training	With pre-training
Polblog	6.2	2.1
Citeseer	183.6	101.3
Cora	209.8	107.6
BlogCatalog	11304	3799

without pre-training on every dataset. We evaluate the convergence efficiency from two aspects: the number of iterations (#Iterations) and the convergence time (Time). The results are shown in Fig. 7 and Table 4.

As shown in Fig. 7, DRNMF with pre-training needs 9, 28 and 67 iterations on Polblog, Citeseer and Cora datasets respectively till convergence, while DRNMF without pre-training respectively needs 28, 69 and 121 iterations. On larger dataset BlogCatalog, DRNMF with pre-training spending 251 iterations is more advantageous than DRNMF without pre-training spending 592 iterations. Less iterations, less convergence time. As shown in Table 4, compared with DRNMF without pre-training, DRNMF with pre-training saves time by 66.1%, 44.8%, 48.7% and 66.3% respectively on Polblog, Citeseer, Cora and BlogCatalog datasets. In particular, it takes about 1.05 hours running on BlogCatalog dataset, which is far less than 3.14 hours taken by DRNMF without pre-training. All these mentioned results demonstrate that, the proposed pre-training strategy in DRNMF accelerates the model within less iterations and time consumption till convergence, thereby effectively helps DRNMF improve the convergence efficiency.

V. CONCLUSION

In order to further increase the performance of matrix factorization for network embedding, in this paper, we propose a method called DRNMF which has multi-layer factorization structure. This structure makes DRNMF be able to learn more useful hierarchical features hidden in complex networks. Meanwhile, we select the combination of high-order proximity matrices of the network as the original feature matrix for factorization, and use $\ell_{2,1}$ norm to improve the robustness of the network embedding model. These strategies make our method perform better than the state-of-the-art matrix factorization based methods, which has been verified by extensive relevant experiments. In the future, there will still be some issues as follows to be solved further.

- There is no doubt that the performance of DRNMF will be affected by the configuration of layers, including the number of layers and the size of each layer. As most of the deep learning based methods, here their settings need to be tuned manually according to the change of performance. Therefore, an in-depth exploration of layer configuration is quite necessary.
- The time complexity of DRNMF is $\mathcal{O}(l(t_p + t_f)(nd^2 + n^2d))$, which restricts itself to be applied efficiently in large-scale complex networks. Hence, more efficient optimization algorithms for DRNMF model need to be developed.
- At present, it is generally believed that deep learning methods are all built on the basis of neural networks. However, DRNMF also has multi-layer learning structure, which makes it resemble a conventional deep learning method. It will be quite interesting to investigate whether DRNMF can reach or even outperform those network embedding methods established on the basis of deep neural networks. This needs us to conduct more research work with comparative experiments.

REFERENCES

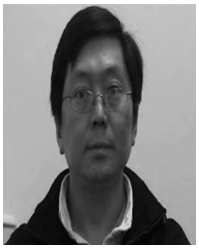
[1] S. Bhagat, G. Cormode, and S. Muthukrishnan, *Social Network Data Analytics*. London, U.K.: Springer-Verlag, 2011.
[2] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rep.*, vol. 659, no. 11, pp. 1–44, Nov. 2016.
[3] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Phys. A, Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, Mar. 2011.
[4] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[5] Y. Xie, M. Gong, S. Wang, W. Liu, and B. Yu, "Sim2vec: Node similarity preserving network embedding," *Inf. Sci.*, vol. 495, pp. 37–51, Aug. 2019.
[6] M. M. Keikha, M. Rahgozar, and M. Asadpour, "Community aware random walk for network embedding," *Knowl.-Based Syst.*, vol. 148, pp. 47–54, May 2018.
[7] W. Liu, Z. Liu, F. Yu, P.-Y. Chen, T. Suzumura, and G. Hu, "A scalable attribute-aware network embedding system," *Neurocomputing*, vol. 339, pp. 279–291, Apr. 2019.
[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 1225–1234.
[9] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 13th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 1145–1152.
[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, Aug. 2014, pp. 701–710.
[11] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 855–864.
[12] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 203–209.
[13] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Melbourne, VIC, Australia, 2015, pp. 891–900.
[14] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec," in *Proc. 11th ACM Int. Conf. Web Search Data Mining (WSDM)*, Los Angeles, CA, USA, Feb. 2018, pp. 1–9.
[15] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.
[16] H. Liu, Z. Wu, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1299–1311, Jul. 2012.
[17] Z. Shu, X. Wu, H. Fan, P. Huang, D. Wu, C. Hu, and F. Ye, "Parameter-less auto-weighted multiple graph regularized nonnegative matrix factorization for data representation," *Knowl.-Based Syst.*, vol. 131, pp. 105–112, Sep. 2017.
[18] X. Jiang, X. Hu, and W. Xu, "Microbiome data representation by joint nonnegative matrix factorization with Laplacian regularization," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 2, pp. 353–359, Mar. 2017.
[19] C. He, Q. Zhang, Y. Tang, S. Liu, and H. Liu, "Network embedding using semi-supervised kernel nonnegative matrix factorization," *IEEE Access*, vol. 7, pp. 92732–92744, 2019.

[20] S. Wang and W. Guo, "Sparse multigraph embedding for multimodal feature representation," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1454–1466, Jul. 2017.
[21] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 417–429, Mar. 2017.
[22] J. Yu, G. Zhou, A. Cichocki, and S. Xie, "Learning the hierarchical parts of objects by deep non-smooth nonnegative matrix factorization," *IEEE Access*, vol. 6, pp. 58096–58105, 2018.
[23] H. A. Song, B.-K. Kim, T. L. Xuan, and S.-Y. Lee, "Hierarchical feature extraction by multi-layer non-negative matrix factorization network for classification task," *Neurocomputing*, vol. 165, pp. 63–74, Oct. 2015.
[24] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
[25] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.
[26] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.
[27] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining Knowl. Discovery*, vol. 23, no. 3, pp. 447–478, Nov. 2011.
[28] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, Rio, Brazil, May 2013, pp. 37–48.
[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, Florence, Italy, May 2015, pp. 1067–1077.
[30] C. C. Tu, W. C. Zhang, Z. Y. Liu, and M. S. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, Jul. 2016, pp. 3889–3895.
[31] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Melbourne, VIC, Australia, Aug. 2017, pp. 3894–3900.
[32] C. Yang, Z. Y. Liu, D. L. Zhao, M. S. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, Buenos Aires, Argentina, Jul. 2015, pp. 2111–2117.
[33] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 1105–1114.
[34] S. Wang and W. Guo, "Robust co-clustering via dual local learning and high-order matrix factorization," *Knowl.-Based Syst.*, vol. 138, pp. 176–187, Dec. 2017.
[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Oxford, U.K.: Cambridge Univ. Press, 2004.
[36] C. H. Q. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.
[37] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. 13th Int. Conf. Neural Inf. Process. Syst.*, Denver, CO, USA, Dec. 2000, pp. 535–541.
[38] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Turin, Italy, Oct. 2018, pp. 1393–1402.
[39] C. He, Q. Zhang, Y. Tang, S. Liu, and J. Zheng, "Community detection method based on robust semi-supervised nonnegative matrix factorization," *Phys. A, Stat. Mech. Appl.*, vol. 523, pp. 279–291, Jun. 2019.

CHAOBO HE received the B.S., M.S., and Ph.D. degrees from South China Normal University, China, in 2004, 2007, and 2014, respectively. He is currently an Associate Professor with the Zhongkai University of Agriculture and Engineering, China, and is also a Visiting Scholar with the School of Data and Computer Science, Sun Yat-sen University, China. He has published over 20 articles on international journals and conferences. His research interests are data mining and social computing.

HAI LIU received the Ph.D. degree from the School of Data and Computer Science, Sun Yat-sen University, China, in 2010. He is currently an Associate Professor with the School of Computer Science, South China Normal University. His current research interests include machine learning, data mining, and big data.

YONG TANG received the B.S. degree in computer science from Wuhan University, in 1985, and the Ph.D. degree in computer science from the University of Science and Technology of China, in 2001. He is currently a Professor and the Dean of the School of Computer Science, South China Normal University, and also serves as the Director of the Services Computing Engineering Research Center of Guangdong Province. He has completed more than 30 research and development projects. He has authored or coauthored more than 100 publications in these areas. His research interests include database and cooperative software, temporal information processing, social network, and big data analytics.



XIANG FEI received the B.Sc. and Ph.D. degrees from Southeast University China, in 1992 and 1999, respectively. After graduation, he worked, as a Postdoctoral Research Fellow, on a number of projects including European IST Programs and EPSRC. He is currently working as a Senior Lecturer with the School of Computing, Electronics and Maths, Coventry University. His current research interests include machine learning and data mining in cyber-physical systems.

HANCHAO LI received the B.Sc. degree in mathematics from the University of Warwick, in 2013, and the M.Sc. degree in computing from Coventry University, in 2015, where he is currently pursuing the Ph.D. degree, working on music information retrieval, i.e., data mining in music subject area. He has published several conference and journal articles. His research interests are big data, data mining, machine learning, and any mathematics-related researches.

QIONG ZHANG received the Ph.D. degree from Laval University, QC, Canada in 2016. He worked as a Visiting Researcher with McGill University, QC, Canada, from 2015 to 2016. He is currently an Adjunct Professor with the Nanchang Institute of Technology and also a Research Director with Shenzhen Qihang Academy Company, Ltd. His research interests include computer vision, image processing, natural language processing, artificial intelligence, data mining, etc.