

Graph Convolutional Networks based Non-Negative Matrix Factorization aware Community detection

Pawan Meena^{1*}, Mahesh Pawar² and Anjana Pandey²

^{1*}Department of Computer Science, University Institute of
Technology, RGPV, Bhopal, 462036, MP, India.

²Department of Information Technology, University Institute of
Technology, RGPV, Bhopal, 462036, MP, India.

*Corresponding author(s). E-mail(s): pawan191423@gmail.com;

Contributing authors: mkpawar24@gmail.com;

anjanapandey@rgtu.net;

Abstract

Nonnegative matrix factorization (NMF) is widely used in community discovery because of its effectiveness and easy interpretability. However, most of the existing NMF-based community detection methods are linear. They cannot effectively deal with the nonlinear characteristics of complex networks, resulting in further improvement in community detection performance. Aiming at this problem, a convolution graph network (GCN) enhanced nonlinear NMF community discovery method NMFGCN is proposed. NMFGCN consists of two main modules: GCN and NMF, where GCN is used to learn network node representations, and NMF uses node representations as input to obtain network community representations. In addition, a joint optimization method is proposed to train NMFGCN, which enables NMFGCN to have nonlinear feature representation capabilities and enables GCN and NMF to promote each other and obtain better community segmentation results. Many experiments on artificial synthetic networks and entire networks show that NMFGCN is superior to current NMF-based community discovery methods, thus proving that NMFGCN can improve the performance of NMF community discovery methods. In addition, NMFGCN also outperforms Deep Walk and LINE standard graph representation learning methods.

Keywords: GCN, Community Detection, Nonnegative Matrix Factorization, Deep Walk, Deep Learning.

1 Introduction

Community detection[1] is a fundamental problem in network analysis that involves identifying groups of nodes in a network that are densely connected to each other and sparsely connected to nodes in other groups. Communities can be defined as subsets of nodes that have a higher density of edges within the subset compared to the density of edges between the subset and the rest of the network [2].

Community detection has a wide range of applications[3] in various fields, including social network analysis, biology, physics, and computer science. For example, in social network analysis, community detection can be used to identify groups of users with similar interests or behaviors, while in biology, community detection can be used to identify groups of proteins that interact with each other and perform similar functions.

There are several algorithms[4] and techniques used in community detection, including modularity-based methods, spectral clustering, and hierarchical clustering. Modularity-based methods aim to maximize a quality function known as modularity, which measures the degree of community structure in a network. Spectral clustering is a technique that involves partitioning a network based on the eigenvectors of its adjacency matrix. Hierarchical clustering involves recursively partitioning a network into smaller and smaller communities until a desired level of granularity is achieved. Effectively discovering the community structure in complex networks helps to understand the structural characteristics of the entire network and supports many data analysis applications based on complex networks, such as user point-of-interest recommendation[5] and telecommunications fraud gang discovery[6].

Community detection can provide insights into the structure and function of complex networks, and can help to identify important nodes and edges in a network. It can also be used to compare the community structure of different networks, and to study the evolution of communities over time.

In the past few decades, various community detection methods have emerged, including methods based on graph partitioning[7, 8], methods based on spectral clustering[9], methods based on modularity optimization[10], methods based on The technique of label propagation[11, 12] and the way based on deep learning[13]. It is worth pointing out that non-negative matrix factorization (NMF) is also widely used in community discovery due to its simplicity, easy scalability, and strong interpretability[14]. At present, many community discovery methods based on NMF have been proposed, for example, the community method based on naive non-negative matrix factorization[15], the

community discovery method SNMF based on symmetric NMF[16], the community discovery method SNMF-SS[17] based on semi-supervised NMF, NMF community discovery method based on node similarity and modularity M-NMF[18] and so on. Although the existing NMF-based community detection methods have achieved different degrees of performance improvement because NMF is essentially a linear model, these methods are still linear, and their expressive ability is weak. As a result, complex networks often fail to achieve satisfactory community discovery performance.

In recent years, deep neural networks (such as convolutional neural networks) have been widely used in natural language processing and computer vision due to their powerful nonlinear feature learning capabilities[19]. However, traditional deep neural networks can only be applied to data in Euclidean space, such as images, speech, and text data. Still, they cannot be effectively used for more common graph data because graphs are a typical non-Euclidean spatial data structure. A graph neural network[20] was proposed to extend the neural network model to the graph data processing to solve this problem. The graph neural network is developing rapidly and has produced many variants. Among them, the graph convolutional network GCN[21] has attracted much attention because it has the advantages of traditional convolutional networks and can be applied to graph data. Existing research shows that GCN is more suitable for processing graph data than conventional deep neural networks (including convolutional networks) and has a strong ability to learn nonlinear features of graph data[22]. Since complex networks can be naturally modelled as graph-structured data, the application of GCN can be expected to improve the performance of various complex network analysis tasks, including the community discovery that this paper focuses on. Based on the above analysis, this paper proposes to apply GCN to enhance the performance of the NMF community discovery method. The main work includes:

(1) A nonlinear NMF community discovery method NMFGCN is designed. By combining the respective advantages of NMF and GCN, this method enables it to have both nonlinear feature learning and community discovery capabilities of complex networks. (2) By designing the joint optimization framework of NMF and GCN, a standard training loss function is proposed, which makes full use of community information while considering the network structure so that GCN can learn network node representations that are more conducive to NMF community discovery results, Thereby promoting the improvement of community discovery performance. (3) Through a large number of experiments on artificial synthetic networks and real networks, the results show that NMFGCN is superior to existing NMF-based community discovery methods and commonly used network representation learning methods: DeepWalk[23] and LINE[24].

2 RELATED WORK

2.1 NMF-based community discovery algorithm

NMF-based community discovery algorithm is a method for detecting communities in networks using non-negative matrix factorization (NMF). NMF is a matrix factorization technique that factorizes a non-negative matrix into two non-negative matrices, which can be interpreted as basis and coefficient matrices.

The NMF-based community discovery algorithm starts by representing the network as an adjacency matrix A . The adjacency matrix is then factorized into two non-negative matrices W and H using NMF, such that $A \approx WH$. The columns of matrix W represent the basis vectors, and the rows of matrix H represent the coefficient vectors.

The algorithm then applies a clustering method, such as k-means or hierarchical clustering, to the columns of matrix H to group nodes with similar coefficient vectors into communities. The number of communities can be determined using methods such as silhouette analysis or the elbow method.

One advantage of NMF-based community discovery algorithm is that it can handle weighted and directed networks, as well as overlapping communities. It also allows for easy interpretation of the results, as the basis vectors represent the characteristic patterns of edges within communities.

However, one limitation of this method is that the choice of the number of communities can have a significant impact on the results, and there is no objective criterion for selecting the optimal number of communities. Additionally, NMF-based community detection may be computationally expensive for large networks.

NMF is a classic low-rank matrix factorization model[25]. Formally, given a matrix $X \in R_+^{n \times m}$, NMF can decompose X into two low-rank non-negative matrices $U \in R_+^{n \times k}$ and $H \in R_+^{k \times m}$ the approximate product $X \approx UH$, where U is the base matrix, H is the coefficient matrix, $k \ll \min(n, m)$. To make UH as close as possible to X , an objective function is needed to measure the error between X and UH and minimize the error during the iterative update solution. Usually the Frobenius norm can be used as the objective function, which is defined as follows:

$$\|X - UH\|_F^2 = \sqrt{\sum_{i=1}^n \sum_{j=1}^m (X_{ij} - [UH]_{ij})^2} \quad (1)$$

NMF has been proven to have an approximate equivalence relationship with K-means and spectral clustering models[26], so NMF has good data clustering capabilities. In addition, community discovery is essentially a node clustering problem, so NMF is also suitable for community discovery. At present, there are many community discovery methods based on NMF, mainly by

extending the basic NMF model to solve the community discovery problem of various complex networks, for example, community discovery method SNMF based on symmetric NMF[16, 27], community-based on joint NMF The discovery method S2-jNMF[28], the semi-supervised NMF-based community method SPOCD[29], SMpC[30] and the deep NMF-based community method DANMF[31], etc. Although these methods have achieved specific performance improvements to varying degrees, since NMF is essentially a linear model, these methods are still linear and do not have nonlinear expression capabilities. Moreover, they are found in complex network communities containing many nonlinear features. Therefore, there is still room for further improvement.

2.2 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) are a type of neural network architecture designed to operate on graph data. GCNs are inspired by convolutional neural networks (CNNs), which are widely used in computer vision tasks. However, while CNNs operate on grid-like data such as images, GCNs are designed to operate on graph-structured data such as social networks, biological networks, and knowledge graphs.

GCNs use a message-passing scheme to aggregate information from neighboring nodes in a graph. The nodes in a graph are represented as a matrix, where each row corresponds to a node and the columns correspond to the node's features. The adjacency matrix of the graph is used to define the relationships between the nodes.

In each layer of a GCN, a node's representation is updated by aggregating information from its neighboring nodes. This is done by computing a weighted sum of the features of the neighboring nodes, where the weights are determined by the adjacency matrix. The resulting aggregated features are then transformed using a neural network layer, such as a fully connected layer or a convolutional layer, before being passed on to the next layer.

GCNs can be used for a variety of tasks, such as node classification, link prediction, and community detection. They have been shown to be effective in capturing the complex dependencies between nodes in graphs and achieving state-of-the-art performance on various graph-related tasks.

One limitation of GCNs is that they can be computationally expensive for large graphs. Additionally, GCNs may struggle with graph data that is highly irregular or contains many isolated nodes. However, there are ongoing efforts to develop more efficient and scalable GCN variants to address these limitations.

However, convolution operations of convolutional neural networks can only be defined on Euclidean spatial data, such as images, texts, and sequences. Still, they cannot be determined on graphs that belong to non-Euclidean spatial data structures. To extend the convolution operation to graph data, Bruna et al. proposed the spectral domain graph convolution network Spectral GCN[32]. Spectral GCN exploits the convolution theorem[33] by computing the eigendecomposition of the Laplacian matrix of the graph, defining the convolution operation in the Fourier domain. However, Spectral GCN

has a high time complexity due to the need to calculate the eigenvalues of the graph Laplacian matrix. To improve the computational efficiency of Spectral GCN, literature[34] proposed the Chebyshev network ChebyNet. ChebyNet uses Chebyshev expansion to approximate convolution operation, which significantly improves the efficiency of the graph convolution operation.

Literature[21] further simplified the Chebyshev network to the first order and proposed a graph convolutional network GCN. The core idea of GCN is that a node defines a convolution operation by aggregating the representations of its neighbour nodes and nodes to obtain the expression of its nodes. Due to its simplicity and effectiveness, GCN has become the basic model of the current graph convolutional network.

The convolutional graph network is a semi-supervised method, which usually needs to know the label information of some nodes to train the network parameters. However, most real-world networks are without label information, so traditional graph convolutions cannot be directly used to handle community discovery tasks. Graph auto-encoder (graph auto-encoder, GAE)[35] is a variant of GCN. GAE does not need the label information of nodes to supervise the training network parameters. Its core idea is to learn the low-dimensional representation of nodes through graph convolution. Reconstruct the network and make the reconstructed network close to the original network during the training process. GAE is an unsupervised network representation learning method that can be applied to complex network tasks without accurate labels. For this reason, this paper proposes a community discovery method that combines graph convolution auto encoders with NMF.

3 Nonlinear community discovery methods

NMFGCN

3.1 Problem Definition

Without loss of generality, a given complex network containing n nodes can be modeled as an undirected graph $G = (V, E)$, where the node set is $V = \{v_1, v_2, \dots, v_n\}$, undirected. The edge set is $E = \{e_{ij} \mid v_i \in V, v_j \in V\}$. The adjacency matrix A corresponding to G is a binary matrix.

$$A_{ij} = \begin{cases} 0, & e_{ij} \notin E \\ 1, & e_{ij} \in E \end{cases} \quad (2)$$

Each node in G has a d -dimensional eigenvector $x \in \mathbb{R}^{1 \times d}$, and the eigenvectors of nodes in G constitute the node feature matrix $X \in \mathbb{R}^{n \times d}$. Let k be the number of communities in the network, $C = \{c_1, c_2, \dots, c_k\}$ represents the community division of G , where c_i is the set of nodes belonging to the i -th community, for example, $c_2 = \{1, 2, 5\}$ represents the nodes belonging to the i -th community. The nodes of the 2 districts are v_1, v_2 , and v_5 . Based on the above definitions, the problem this paper is dedicated to solving

is: for a given $G = (V, E)$, design a nonlinear unsupervised community discovery method NMFGCN, and learn the community member representation matrix $U \in R_+^{n \times k}$, $0 \leq u_{ij} \leq 1$, thus predicting the community partition C of a given network to improve the performance of NMF-based community discovery methods. Among them, the i -th row U_i of U is the probability that node i belongs to k different communities, and $\sum_{j=1}^k u_{ij} = 1$. Specifically, $u_{4,2} = 0.6$ means that the probability that node 4 belongs to community c_2 is 0.6

3.2 NMFGCN Community Discovery Model

The model framework of NMFGCN is shown in Figure 1. It mainly includes two modules: GCN Module and NMF Module. GCN takes the adjacency matrix A , the feature matrix X of the node as input, and the output node represents $Z \in R_+^{n \times m}$, where m is the dimension of the GCN output layer. On the other hand, NMF takes β node representation Z as input and outputs a node community member representation matrix U . The specific realization of each module is introduced below.

(1) GCN Module. The input of the GCN Module is the adjacency matrix A and the feature matrix X of the node, and its goal is to learn the low-dimensional representation matrix Z of the node. Considering the GCN of layer L , for layers 1 to $L - 1$, the hidden layer representation of nodes is calculated as follows:

$$H^{l+1} = \sigma \left(D^{-1/2} A D^{-1/2} H^l W^l \right) \quad (3)$$

H^1 is the node representation of layer 1 (specifically, $H^1 = X$). W^1 is the learnable weight matrix of layer 1, $A = A + I$, and I is the identity matrix. Matrix D is a diagonal matrix, the degree matrix of nodes, $D_{ii} = \sum_j A_{ij}$ - σ is a nonlinear activation function, which can be a commonly used ReLU. For the last layer, use sigmoid as the activation function to get the final representation Z of the nodes:

$$Z = \text{sigmoid} \left(D^{-1/2} A D^{-1/2} H^{(L-1)} W^{(L-1)} \right) \quad (4)$$

After the node representation Z is learned through graph convolution, make an inner product with Z^T to reconstruct the adjacency matrix:

$$A' = Z Z^T \quad (5)$$

The probability of reconstructing the adjacency matrix $A'_{ij} = 1$ is:

$$p(A'_{ij} = 1) = \text{sigmoid}(Z_i Z_j^T) \quad (6)$$

Correspondingly, there are:

$$p(A'_{ij} = 0) = 1 - \text{sigmoid}(Z_i Z_j^T) \quad (7)$$

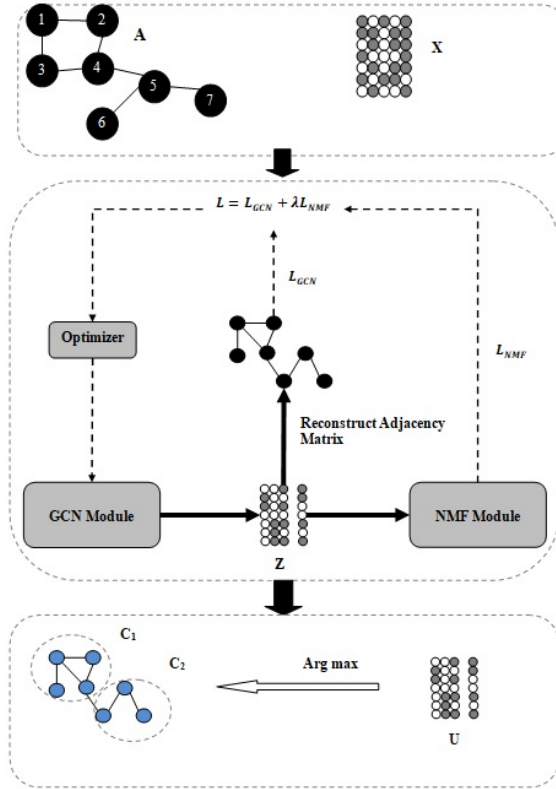


Fig. 1 NMFGCN model framework.

The purpose of reconstructing the adjacency matrix is to make A' approximate to A so that the low-dimensional dense matrix Z can represent the information of the graph. To this end, it is necessary to use a loss function to measure the error between A and A' and to make A' approximate to A by minimizing the error. This article uses cross entropy as the loss function:

$$L_{GCN} = -\frac{1}{n^2} \sum_{i,j} (A_{ij} \log p(A'_{ij} = 1) + (1 - A_{ij}) \log p(A'_{ij} = 0)) \quad (8)$$

(2) NMF Module. To get the community membership representation matrix U of a node. First, two low-rank matrices $U \in R_+^{n \times k}$ and $V \in R_+^{k \times m}$ are randomly initialized, where k is the number of communities, and NMF updates U and V in the iterative process and finally makes Z approximate to UV . To this end, it is necessary to use a loss function to measure the error between Z and UV and minimize the error. This article chooses to use the Frobenius norm:

$$\min L_{NMF}(Z \| UV) = \min \left(\frac{1}{2} \|Z - UV\|_F^2 \text{ s.t. } U \geq 0, V \geq 0 \right) \quad (9)$$

Using the gradient descent algorithm can get:

$$u_{ik} \leftarrow u_{ik} - \mu_{ik} \frac{\partial L_{NMF}}{\partial u_{ik}} \quad (10)$$

$$v_{kj} \leftarrow v_{kj} - \eta_{kj} \frac{\partial L_{NMF}}{\partial v_{kj}} \quad (11)$$

In:

$$\frac{\partial L_{NMF}}{\partial u_{ik}} = - [(Z - UV)V^T]_{ik} \quad (12)$$

$$\frac{\partial L_{NMF}}{\partial v_{kj}} = - [U^T(Z - UV)]_{kj} \quad (13)$$

Let $\mu_{ik} = \frac{u_{ik}}{[UVV^T]_{ik}}$, $\eta_{kj} = \frac{v_{kj}}{[U^TUV]_{kj}}$, then:

$$u_{ik} \leftarrow u_{ik} \frac{[ZV^T]_{ik}}{[UVV^T]_{ik}} \quad (14)$$

$$v_{kj} \leftarrow v_{kj} \frac{[U^TZ]_{kj}}{[U^TUV]_{kj}} \quad (15)$$

Iteratively update U and V respectively using Equation (14) and Equation (15) until L_{NMF} converges, then U is the node's community member representation matrix, and $u_{ij} \in U$ is the probability that node i belongs to community c_j . Algorithm 1 describes the process of obtaining the community member representation matrix U through the node representation Z as follows: Algorithm 1NMF obtains community member representation algorithm Input: node representation matrix $Z \in R_+^{n*m}$, the maximum number of iterations epoch, number of network communities k

Output: node community representation matrix U and community feature representation matrix V 1. Randomly initialize the non-negative matrices $U \in R_+^{n*k}$ and $V \in R_+^{k*m}$ 2. while $e < \text{epoch}$ do 3. $L_{NMF} = 0.5 \times Z - UV_F^2$; 4. if L_{NMF} converges do 5. End the iteration early; 6. end if 7. Update U through formula (14); 8. Update V through Equation (15); 9. $e \leftarrow e + 1$ 10. end while 11. return U, V ;

3.3 Loss function construction and optimization

(1) Loss function. The problem dealt with in this paper is unsupervised community discovery. The node representation obtained by training graph convolution using only L_{GCN} as a loss function cannot be used as a community representative. Therefore, this paper adopts the common training method of L_{GCN} and L_{NMF} :

$$L = L_{GCN} + \lambda \times L_{NMF} \quad (16)$$

Among them, λ is a coefficient used to weigh the size of L_{GCN} and L_{NMF} , which is 1 by default. By minimizing L , it can be expected that GCN can learn a node

representation Z that is more conducive to NMF to obtain better community division results. (2) Optimal solution. To optimize the resolution of the L function, it is necessary to iteratively update the GCN network parameters $W \in R^{h_{out}^l * h_{in}^l}$ and the NMF decomposition factor matrices U and V , where h_{out}^l and h_{in}^l are the output feature dimension and input feature dimension of the 1-th layer convolution, respectively. This paper uses a two-layer GCN. The input and output feature dimensions of the first layer of convolution are the dimension of the initial feature X of the node and 256, while the input and output feature dimensions of the second layer of convolution are 256 and 128, respectively. For W , update using stochastic gradient descent:

$$W^l = W^l - \gamma \frac{\partial L}{\partial W^l} \quad (17)$$

Among them, W^1 represents the parameter matrix of the first layer of GCN, γ is the learning rate, and $\partial L / \partial W$ is the gradient. For U and V , the corresponding iterative update rules use formulas (14) and (15), respectively.

In the iterative optimization solution process, the update of GCN can be controlled by setting the number of external iterations, and the update of NMF can be held by selecting the number of internal iterations. When the L function converges or exceeds the outer iterations, the iteration is stopped, and the node community member representation matrix U is obtained. The i -th row U_i of matrix U is the community representation of node i . U_{ij} is the probability that node i belongs to community c_j . Take the subscript j corresponding to the maximum value in U_i , then c_j is the community to which node i belongs. According to this idea, the design of the NMFGCN community discovery algorithm is shown in Algorithm 2.

Algorithm 2 NMFGCN community discovery algorithm

Input: adjacency matrix A , feature matrix X , number of communities k , number of iterations epoch, loss trade-off coefficient λ

Output: community division result C

1. while $e < \text{epoch}$ do
2. Obtain the node representation Z through the GCN Module;
3. $A' \leftarrow ZZ^T$; // Reconstruct the adjacency matrix
4. Calculate L_{CGN} through formula (8);
5. Obtain U and V through Algorithm 1;
6. Calculate L_{NMF} through formula (9);
7. $L \leftarrow L_{CGN} + \lambda L_{NMF}$;
8. for $1 \leftarrow 1$ to L ;
9. $W^l = W^l - \gamma \frac{\partial L}{\partial W^l}$
10. end for
11. $e \leftarrow e + 1$
12. end while
13. for $i \leftarrow 1$ to n
14. $j \leftarrow \arg \max_j (U_{ij})$;

15. $c_i \leftarrow c_j \cup \{v_i\}$
16. end for
17. return C ;

3.4 Time Complexity Analysis

Analyzing Algorithm 1 and Algorithm 2, it is easy to conclude that NMFGCN has two core steps: the first step is to optimize the convolutional graph network and NMF iteratively, and the second step is to obtain the community division result C from the community member representation matrix U . According to Algorithm 2, the time complexity of the second step is $O(nk)$, and the following mainly analyzes the time complexity of the first step.

In the specific implementation, a 2-layer GCN is used. For simplicity, the dimensions of each layer of GCN output are denoted by m . Assuming that the number of nodes in the network is n , the number of edges is E , the number of communities is k , the feature dimension of nodes is d , and the number of model iterations is T . The number of iterations inside NMF is t . The adjacency matrix A is stored in a sparse matrix; GCN The time complexity can be expressed as $O(mde)$. The time for iterative optimization of NMF is mainly spent on updating the community member matrix U and community feature matrix V in equations (14) and (15), and its time complexity is $O(tnmk)$. So the total time complexity of model training is $O(T(mde + tnmk))$ where $k \ll m, m \ll d$. Due to the complex relationship of most real networks, the connections between nodes are relatively dense, and the number n of network nodes is usually much smaller than the number of edges e . Therefore, the time complexity of NMFGCN mainly depends on the edge number e of the network.

4 Experimental analysis

To verify the effectiveness of NMFGCN, this paper conducts a comparative analysis with several representative methods on artificial synthetic and real networks. The experimental hardware platform is Intel Core i7-9700 CPU, the central frequency is 2.4 GHz, the memory is 32 GB, and the operating system is Windows 10. All comparison methods are implemented using Python 3.7.

4.1 Comparison Method

In this paper, several NMF-based community discovery methods are selected for comparison. In addition, the currently widely used network representation learning-based community discovery methods DeepWalk and LINE, are chosen for comparison. All these comparison methods are briefly introduced as follows:

1. **NMF[15]:** This method uses the basic NMF model to directly decompose the adjacency matrix A to obtain the matrices U and V , $A \approx UV^T$, where U represents the matrix as a community member.

2. **SNMF[16]**: SNMF is based on a symmetric non-negative matrix factorization model (symmetrical NMF, SNMF): $A \approx HH^T$, where H can directly represent the membership strength of community members.
3. **M-NMF[18]**: M-NMF is a modularity-based NMF community detection model, which integrates the network's community structure into the network embedding, and jointly optimizes the community detection model based on a non-negative community detection model.
4. **ONMF[36]**: ONMF is based on the orthogonal non-negative matrix factorization model (orthogonal NMF, ONMF). The basic idea is to impose an orthogonal constraint on W based on the NMF model $A \approx WH_F^2$ so that $W^T W = I$.
5. **HPNMF[37]**: HPNMF is based on a graph regularization NMF model, which can integrate network topology and node homogeneity information for community discovery.
6. **NSED[38]**: NSED is based on the joint NMF model, which includes an encoder and a decoder, and the community member representation matrix can be obtained through the encoder.
7. **DeepWalk[23]**: DeepWalk is a classic network representation learning method. The core idea of this method is to sample in the graph through the random walk and use the co-occurrence relationship between nodes in the diagram to learn the low-dimensionality of nodes. express.
8. **LINE[24]**: LINE is also a network representation learning method, which obtains low-dimensional representations of network nodes by preserving the first-order and second-order similarities of nodes.

It should be pointed out that for the NMF-based community discovery methods, namely NMF, SNMF, ONMF, M-NMF, HPNMF and NSED, the community discovery results can be obtained directly through the community member representation matrix, while for the network representation learning-based methods DeepWalk and LINE, you can use these methods to learn the low-dimensional picture of nodes first, and then cluster the low-dimensional phrases through K-means. Finally, the clustering results are used as the final community discovery results.

4.2 Dataset

The data sets used in the experiment include two parts: an artificial synthetic network and an entire network. These data sets are introduced as follows.

1. **Synthetic Network**: This paper uses the LFR benchmark network synthesis tool[39] to generate artificial networks with accurate community labels. The LFR tool has multiple adjustable parameters (Table 1). This paper creates various sets of different artificial networks by fixing some parameters and changing the remaining ones. Specifically, by selecting n , k , $\max_k$, $\min_k$, and $\max_c$, μ is changed from 0.1 to 0.3, and a set of 5 networks is generated with each increase of 0.05. Then by fixing k , $\max_k$, $\min_k$, $\max_c$ and μ , the number of nodes n increases from 1 000 to 5 000, rising by 1 000

Table 1 Adjustable parameters of LFR

Parameter	Explain
n	Number of network nodes
k	The average degree of nodes in the network
maxk	The maximum degree of a node in the network
mu	Confusion coefficient, the adjustable range is [0,1]
minc	The number of nodes contained in the smallest community in the network
maxc	The number of nodes contained in the largest community in the network

Table 2 Parameter settings of artificial synthesis network

Parameter	First Group	Second Group
n	3000	1000~5000
k	4	4
maxk	15	15
minc	50	n/20
maxc	100	n/40
mu	0.1~0.3	0.2

Table 3 Real Network parameters

Data Set	Number of Nodes	Number of Sides	Feature Dimension	Number of Communities
WebKB	877	1399	1703	4
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

each time to generate another group of 5 networks. The specific parameter settings of the two groups of networks are shown in Table 2.

- 2. Real Network:** This paper selects 4 real network datasets, namely WebKB, Cora, Citeseer and Pubmed[40]. The specific parameters of the entire data set are shown in Table 3. The above four datasets can be downloaded at <https://linqs.soe.ucsc.edu/data>.

Considering that most of the comparison methods are unsupervised community discovery methods for non-attribute networks, for the sake of fairness, the input feature X of MFGCN is obtained by decomposing the adjacency matrix A using NMF when comparing with the comparison methods in real networks and artificial networks.

4.3 Evaluation criteria and parameter settings

To evaluate the performance of community discovery results, this paper uses four currently commonly used evaluation criteria: mutual information (normalized mutual information, NMI), adjusted rand index (adjusted rand index,

ARI), accuracy (accuracy, ACC) and modularity Q, the specific definitions of these evaluation criteria can refer to [41] about community discovery performance evaluation an overview of pricing principles. This paper uses NMI, ARI, ACC and modularity Q for artificial synthetic networks to evaluate the results. Since there is no community division label for the entire network, this paper uses the modularity Q for the result evaluation. For all evaluation criteria, larger values indicate better performance and vice versa.

For a fair comparison, the parameters of all comparison methods in the experiment refer to their default values in the original text. Expressly, for M-NMF, set its regularization parameter $\alpha = 1, \beta = 5$; for HPNMF, set its regularization parameter $\lambda = 1$; for DeepWalk, set the window size $\omega = 10$, the number of iterations $\gamma = 80$, walk length = 40, feature dimension $d = 128$; for LINE, retain the second-order similarity, set the number of negative samples $k = 5$, feature dimension $d = 128$; For NMFGCN, set the loss trade-off factor $\lambda = 1$. The GCN Module uses a 2-layer graph convolution, and the output dimensions of the first and second layers are set to 256 and 128, respectively. In addition, each method was run ten times in the experiment, and the average value of each evaluation index was taken for evaluation.

4.4 Experimental results and analysis of synthetic network

The comparative study was carried out on the two sets of artificially synthesized network datasets shown in Table 2. The experimental results of the first set of networks are shown in Table 4

It can be seen from Table 4 that with the increase of the confusion coefficient μ , the community structure of the network becomes less and less clear, and the difficulty of community detection increases continuously. The performance of each method has a relatively apparent downward trend, but NMFGCN outperforms other models on each network. For example, when $\mu = 0.1$, compared with SNMF, the best-performing community detection algorithm based on NMF, the ACC, NMI, ARI and modularity Q of NMFGCN are improved by 7.7%, 3.1%, 8.2%, and 2.4%, respectively. Compared with LINE, the best community discovery algorithm based on graph representation learning, NMFGCN's ACC, NMI, ARI, and modularity Q have increased by 4.6%, 1.7%, 7.1%, and 1.3%, respectively. NMFGCN has different degrees of improvement in the four evaluation indicators on different data sets. The experimental results on the second set of artificial networks are shown in Figure 2, from which it can also be seen that NMFGCN achieves the best performance in all networks with different nodes.

The experimental results on artificial synthetic networks show that NMFGCN performs better than existing NMF-based community discovery methods. The assumption that GCN enhances the performance of NMF community discovery has been preliminarily verified.

Table 4 Performance comparison of synthetic networks with different confusion coefficients mu

mu	Parameter	NMF	SNMF	M-NMF	ONMF	HP-NMF	NSED	Deep Walk	LINE	NMF GCN
0.1000	ACC	0.8578	0.8680	0.5610	0.8405	0.8262	0.5029	0.7252	0.8945	0.9353
	NMI	0.8782	0.8813	0.6538	0.8680	0.8680	0.6334	0.7120	0.8935	0.9088
	ARI	0.7823	0.7874	0.3917	0.7609	0.7517	0.3662	0.5467	0.7956	0.8527
	Q	0.8813	0.9027	0.8048	0.8956	0.8945	0.7109	0.7844	0.9119	0.9241
	Factor									
0.1500	ACC	0.8191	0.7976	0.4753	0.8058	0.7721	0.3774	0.6610	0.8670	0.8905
	NMI	0.8170	0.8038	0.5702	0.8089	0.7946	0.5080	0.6477	0.8537	0.8609
	ARI	0.6916	0.6742	0.3060	0.6814	0.6487	0.2173	0.4692	0.7497	0.7711
	Q	0.8537	0.8568	0.7650	0.8588	0.8507	0.6365	0.7395	0.8690	0.8772
	Factor									
0.2000	ACC	0.7446	0.7283	0.3886	0.7171	0.6997	0.3376	0.3835	0.8058	0.8109
	NMI	0.7426	0.7354	0.4937	0.7283	0.7252	0.4631	0.4080	0.7691	0.7834
	ARI	0.5926	0.5783	0.2060	0.5661	0.5559	0.1918	0.1969	0.6691	0.6752
	Q	0.7976	0.8150	0.7222	0.8119	0.8089	0.6232	0.5641	0.8119	0.8323
	Factor									
0.2500	ACC	0.6650	0.6620	0.3254	0.6344	0.6293	0.3223	0.3580	0.7283	0.7334
	NMI	0.6589	0.6661	0.4213	0.6497	0.6457	0.4090	0.3907	0.6854	0.7028
	ARI	0.4886	0.4886	0.1550	0.4621	0.4539	0.1448	0.1765	0.5620	0.5671
	Q	0.7742	0.7844	0.6956	0.7772	0.7742	0.6049	0.5865	0.7925	0.7997
	Factor									
0.3000	ACC	0.5202	0.5222	0.1469	0.5324	0.5437	0.2305	0.1601	0.5916	0.6324
	NMI	0.4610	0.5131	0.2407	0.5151	0.5488	0.3427	0.2009	0.5671	0.5865
	ARI	0.3244	0.3193	0.0388	0.3407	0.3427	0.0969	0.0377	0.3509	0.3958
	Q	0.7385	0.7517	0.5916	0.7466	0.7436	0.5885	0.4172	0.7303	0.7538
	Factor									

4.5 Real network experiment results and analysis

To further verify the effectiveness of NMFGCN, a comparative analysis was carried out on four real networks, and the experimental results are shown in Table 5. It can be seen that the results of NMFGCN on each whole network are better than each comparison method. For example, on WebKB, Cora, Citeseer, and Pubmed, NMFGCN has improved modularity compared with HPNMF, the best NMF-based practice.

Compared with the best method, LINE, based on graph representation learning, the modularity has increased by 58.4%, 19.6%, 6.4%, and 2.4%, respectively. Real networks are usually more complex than synthetic networks and contain more nonlinear node features. Models SNMF and LINE that perform well on artificial networks perform poorly in three real data sets. At the same time, NMFGCN is nonlinear. The community discovery model performs better than other models in both natural and synthetic networks. The above results further prove that NMFGCN can enhance the representation ability of

Table 5 Comparison of Real Network Modularity Q

Dataset	NMF	SNMF	M-NMF	ONMF	HP-NMF	NSED	Deep	LINE Walk	NMF GCN
WebKB	0.5845	0.6436	0.6324	0.6548	0.6977	0.6803	0.6283	0.4641	0.7354
Cora	0.5416	0.5345	0.7171	0.6283	0.7283	0.6987	0.6610	0.6181	0.7395
Citeseer	0.6059	0.6181	0.6416	0.5947	0.6661	0.6028	0.5029	0.6844	0.7283
Pubmed	0.4631	0.4315	0.5161	0.5141	0.5437	0.4009	0.3305	0.5375	0.5508

nonlinear network features by integrating GCN and NMF modules, thereby improving the performance of NMF community discovery.

4.6 Verification of joint optimization effect of NMFGCN

As mentioned, NMFGCN can get better community segmentation results by jointly optimizing the NMF and GCN modules. To further verify the combined optimization effect of NMFGCN, the performance of NMFGCN and GCN+NMF is compared and analyzed.

For GCN+NMF, the node representation Z is first learned through GCN, and then NMF is used to decompose Z to obtain the community division result. Since joint training is no longer performed, the loss function of GCN is only the loss L_{GCN} of reconstructing the adjacency matrix, and the loss L_{NMF} of NMF decomposition is no longer considered; that is, its loss function is $L = L_{GCN}$.

In the comparison experiment, the experimental comparison was carried out in the whole network and the artificial synthetic network (the second group of data in Table 2). Each network data set was run ten times, and the average value was taken. The experimental results of the synthetic network are shown in Figure 3, and the experimental results of the entire network are shown in Figure 4. It can be seen that the modularity Q of NMFGCN on the four real datasets is higher than that of GCN+NMF. The reason is that NMFGCN considers the loss of adjacency matrix reconstruction and the failure of NMF decomposition while training so that the network's topology information and community information can be used simultaneously. As a result, NMF and GCN can promote each other: GCN can learn The node representation that is more favourable to NMF, and NMF can also obtain better community division results. On the synthetic network, as the number of nodes increases, the values of the four indicators of NMFGCN and GCN+NMF all decrease, but the performance of NMFGCN is still better than that of GCN+NMF. The above two sets of experiments fully demonstrate the effectiveness of NMFGCN for joint optimization.

5 Conclusion

To improve the performance of the NMF-based community discovery method, this paper proposes a GCN-enhanced nonlinear NMF community discovery method NMFGCN, which uses the powerful network node nonlinear feature

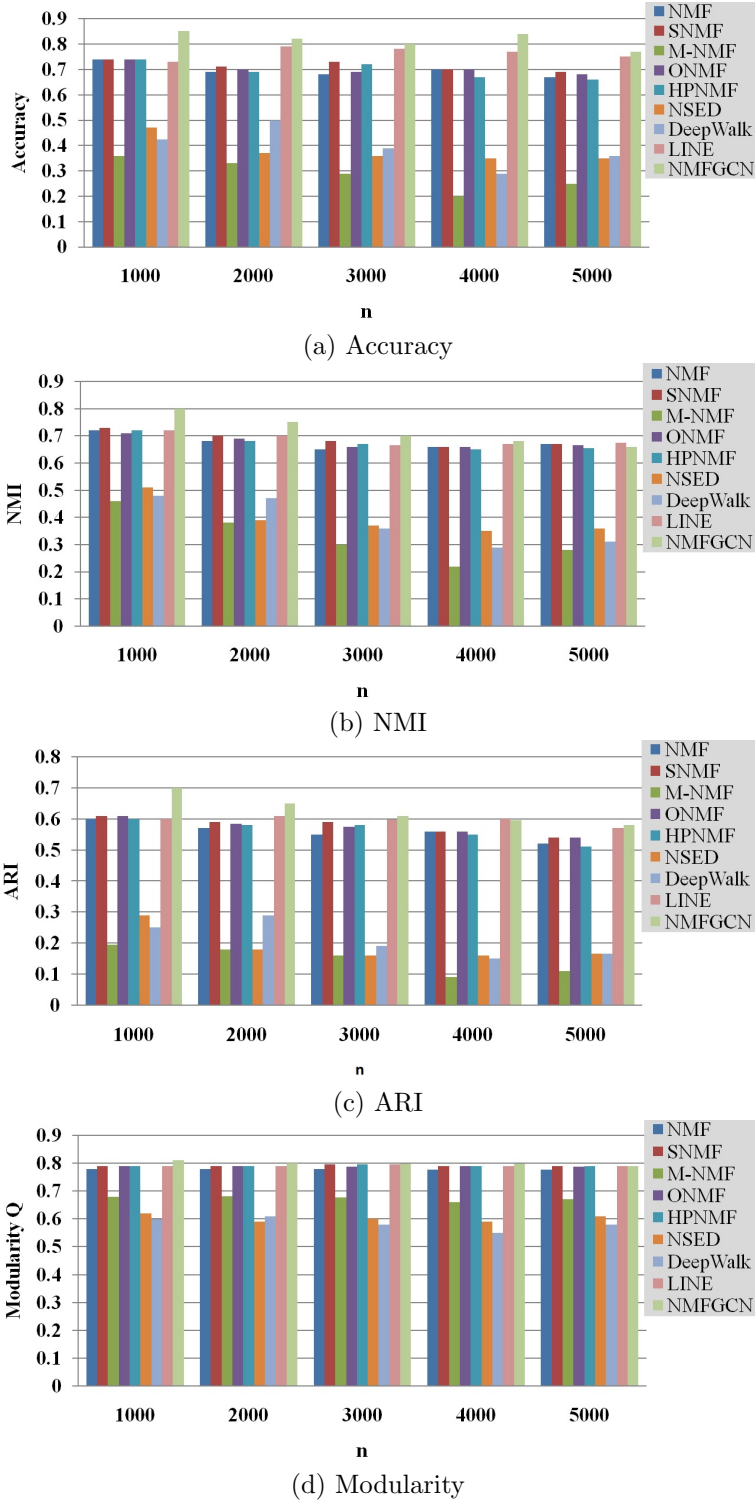
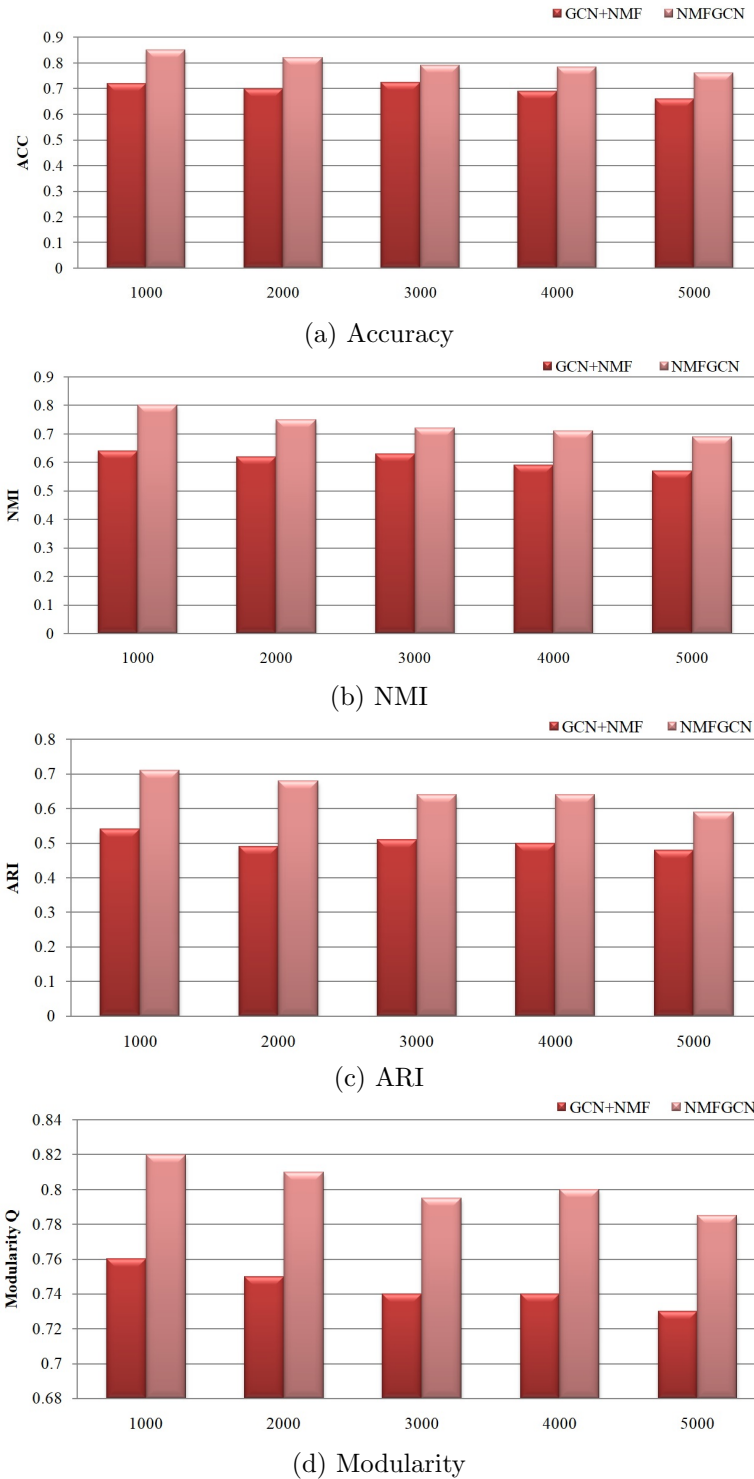


Fig. 2 Performance comparison of synthetic networks with different number of nodes n

**Fig. 3** Joint optimization comparison of artificial synthetic networks

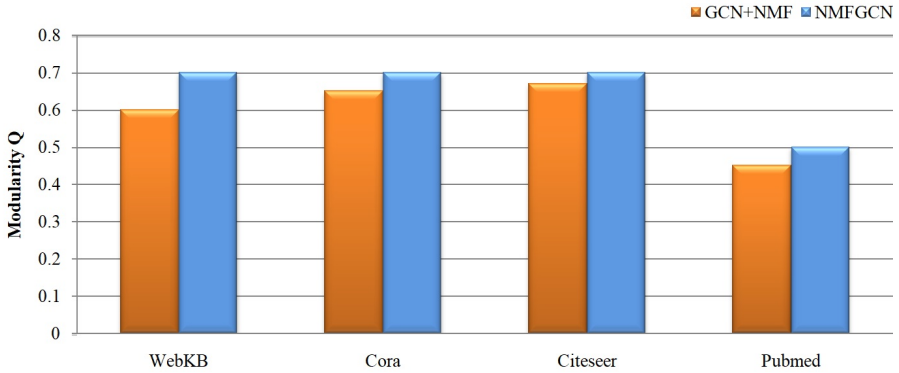


Fig. 4 Joint optimization comparison of real networks

learning ability of GCN to improve the performance of NMF community discovery. To verify the effectiveness of NMFGCN, this paper conducts many comparative experiments on artificial synthetic networks and entire networks. The results show that NMFGCN is superior to existing NMF-based community discovery methods and is also ideal for commonly used network representation learning methods DeepWalk and LINE. It solves the problem that the existing NMF community discovery method is challenging to deal with the characteristics of nonlinear network nodes, and the performance of community discovery is limited because it belongs to the linear model. It should be noted that, compared with the existing NMF-based community discovery model and some commonly used graph representation learning methods, NMFGCN has improved dramatically. However, there are still areas for improvement, specifically in two aspects: (1) The operation efficiency of NMFGCN is not high, and it takes a long time to train an extensive network. The training time of NMFGCN is mainly spent on graph convolution layer learning node representation Z and NMF decomposition Z . The latter can reduce the training time by reducing the size of the graph convolution layer's output dimension and the number of NMF iterations. (2) Since graph convolution is a transductive graph representation learning method, the model must be retrained when nodes are added to the network or the topological relationship is changed. To solve this problem, in the next step, the inductive graph representation learning method will be considered to learn the node representation Z .

6 Declarations

I hereby declare that the manuscript entitled "Graph Convolutional Networks based Non-Negative Matrix Factorization Aware Community Detection" is my own research article, written in my own words under the supervision of Dr. Mahesh Pawar and Dr. Anjana Pandey, UIT, RGPV, Bhopal (M.P.). This work is not funded by a Conflict of interest by co-authors.

- Funding
Not Applicable
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
No
- Ethics approval
Not Applicable
- Consent to participate
Yes
- Consent for publication
Yes
- Availability of data and materials
Yes
- Code availability
Yes
- Authors' contributions
Already described in declaration section.

References

- [1] Meena, P., Pawar, M., Pandey, A.: An important sampling over graph convolutional network for community detection. In: 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 1204–1209 (2022). <https://doi.org/10.1109/ICACCS54159.2022.9785268>
- [2] Wang, X., Zhong, Y., Zhang, L., Xu, Y.: Spatial group sparsity regularized nonnegative matrix factorization for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing* **55**(11), 6287–6304 (2017). <https://doi.org/10.1109/TGRS.2017.2724944>
- [3] Meena, P., Pawar, M., Pandey, A.: A survey on community detection algorithm and its applications. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* **12**(6), 4807–4815 (2021)
- [4] Meena, P., Pawar, M., Pandey, A.: Comparative analysis of overlap community detection techniques on social media platform. *The Computer Journal* (2022)
- [5] Hu, J., Xing, Y., Han, M., Wang, F., Zhao, K., Che, X.: Nonnegative matrix tri-factorization based clustering in a heterogeneous information network with star network schema. *Tsinghua Science and Technology* **27**(2), 386–395 (2022). <https://doi.org/10.26599/TST.2020.9010049>
- [6] Teng, X., Lan, L., Zhang, X., Dong, G., Luo, Z.: Transductive nonnegative matrix tri-factorization. *IEEE Access* **8**, 81331–81347 (2020). <https://doi.org/10.1109/ACCESS.2020.3000000>

[org/10.1109/ACCESS.2020.2989527](https://doi.org/10.1109/ACCESS.2020.2989527)

- [7] Guo, Z., Zhang, S.: Sparse deep nonnegative matrix factorization. *Big Data Mining and Analytics* **3**(1), 13–28 (2020). <https://doi.org/10.26599/BDMA.2019.9020020>
- [8] Debals, O., Van Barel, M., De Lathauwer, L.: Nonnegative matrix factorization using nonnegative polynomial approximations. *IEEE Signal Processing Letters* **24**(7), 948–952 (2017). <https://doi.org/10.1109/LSP.2017.2697680>
- [9] Wang, Y.-X., Zhang, Y.-J.: Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* **25**(6), 1336–1353 (2013). <https://doi.org/10.1109/TKDE.2012.51>
- [10] Lee, S., Park, S.H., Sung, K.-M.: Beam-space domain multichannel non-negative matrix factorization for audio source separation. *IEEE Signal Processing Letters* **19**(1), 43–46 (2012). <https://doi.org/10.1109/LSP.2011.2173192>
- [11] Huang, N., Xiao, L., Xu, Y., Chanussot, J.: A bipartite graph partition-based coclustering approach with graph nonnegative matrix factorization for large hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing* **60**, 1–18 (2022). <https://doi.org/10.1109/TGRS.2021.3097358>
- [12] Li, H.-C., Liu, S., Feng, X.-R., Zhang, S.-Q.: Sparsity-constrained coupled nonnegative matrix–tensor factorization for hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **13**, 5061–5073 (2020). <https://doi.org/10.1109/JSTARS.2020.3019706>
- [13] Xuan, J., Lu, J., Zhang, G., Xu, R.Y.D., Luo, X.: Doubly nonparametric sparse nonnegative matrix factorization based on dependent indian buffet processes. *IEEE Transactions on Neural Networks and Learning Systems* **29**(5), 1835–1849 (2018). <https://doi.org/10.1109/TNNLS.2017.2676817>
- [14] Wang, T., Yang, F., Yang, J.: Convolutional transfer function-based multichannel nonnegative matrix factorization for overdetermined blind source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **30**, 802–815 (2022). <https://doi.org/10.1109/TASLP.2022.3145304>
- [15] Zhu, W., Yan, Y.: Joint linear regression and nonnegative matrix factorization based on self-organized graph for image clustering and classification. *IEEE Access* **6**, 38820–38834 (2018). <https://doi.org/10.1109/ACCESS.2018.2854232>

- [16] Song, Y., Li, M., Luo, X., Yang, G., Wang, C.: Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: A triple factorization-based approach. *IEEE Transactions on Industrial Informatics* **16**(5), 3006–3017 (2020). <https://doi.org/10.1109/TII.2019.2908958>
- [17] Wang, D., Gao, X., Wang, X.: Semi-supervised nonnegative matrix factorization via constraint propagation. *IEEE Transactions on Cybernetics* **46**(1), 233–244 (2016). <https://doi.org/10.1109/TCYB.2015.2399533>
- [18] Ye, M., Qian, Y., Zhou, J.: Multitask sparse nonnegative matrix factorization for joint spectral–spatial hyperspectral imagery denoising. *IEEE Transactions on Geoscience and Remote Sensing* **53**(5), 2621–2639 (2015). <https://doi.org/10.1109/TGRS.2014.2363101>
- [19] Laroche, C., Kowalski, M., Papadopoulos, H., Richard, G.: Hybrid projective nonnegative matrix factorization with drum dictionaries for harmonic/percussive source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**(9), 1499–1511 (2018). <https://doi.org/10.1109/TASLP.2018.2830116>
- [20] Sekiguchi, K., Bando, Y., Nugraha, A.A., Yoshii, K., Kawahara, T.: Fast multichannel nonnegative matrix factorization with directivity-aware jointly-diagonalizable spatial covariance matrices for blind source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **28**, 2610–2625 (2020). <https://doi.org/10.1109/TASLP.2020.3019181>
- [21] Feng, J., Huo, X., Song, L., Yang, X., Zhang, W.: Evaluation of different algorithms of nonnegative matrix factorization in temporal psychovisual modulation. *IEEE Transactions on Circuits and Systems for Video Technology* **24**(4), 553–565 (2014). <https://doi.org/10.1109/TCSVT.2013.2280089>
- [22] Pei, X., Wu, T., Chen, C.: Automated graph regularized projective nonnegative matrix factorization for document clustering. *IEEE Transactions on Cybernetics* **44**(10), 1821–1831 (2014). <https://doi.org/10.1109/TCYB.2013.2296117>
- [23] Sørensen, M., Sidiropoulos, N.D., Swami, A.: Overlapping community detection via semi-binary matrix factorization: Identifiability and algorithms. *IEEE Transactions on Signal Processing* **70**, 4321–4336 (2022). <https://doi.org/10.1109/TSP.2022.3200215>
- [24] Tepper, M., Sapiro, G.: Compressed nonnegative matrix factorization is fast and accurate. *IEEE Transactions on Signal Processing* **64**(9), 2269–2283 (2016). <https://doi.org/10.1109/TSP.2016.2516971>

- [25] Guan, N., Tao, D., Luo, Z., Yuan, B.: Nnmf: An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing* **60**(6), 2882–2898 (2012). <https://doi.org/10.1109/TSP.2012.2190406>
- [26] Berahmand, K., Mohammadi, M., Saberi-Movahed, F., Li, Y., Xu, Y.: Graph regularized nonnegative matrix factorization for community detection in attributed networks. *IEEE Transactions on Network Science and Engineering* **10**(1), 372–385 (2023). <https://doi.org/10.1109/TNSE.2022.3210233>
- [27] Huang, R., Li, X., Zhao, L.: Hyperspectral unmixing based on incremental kernel nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing* **56**(11), 6645–6662 (2018). <https://doi.org/10.1109/TGRS.2018.2841036>
- [28] Zhang, J., He, Z., Zhang, J., Dai, T.: Cograpth regularized collective nonnegative matrix factorization for multilabel image annotation. *IEEE Access* **7**, 88338–88356 (2019). <https://doi.org/10.1109/ACCESS.2019.2925891>
- [29] Pan, J., Gillis, N.: Generalized separable nonnegative matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**(5), 1546–1561 (2021). <https://doi.org/10.1109/TPAMI.2019.2956046>
- [30] Kim, M., Yoo, J., Kang, K., Choi, S.: Nonnegative matrix partial co-factorization for spectral and temporal drum source separation. *IEEE Journal of Selected Topics in Signal Processing* **5**(6), 1192–1204 (2011). <https://doi.org/10.1109/JSTSP.2011.2158803>
- [31] Li, C., Xie, H.-B., Mengersen, K., Fan, X., Da Xu, R.Y., Sisson, S.A., Van Huffel, S.: Bayesian nonnegative matrix factorization with dirichlet process mixtures. *IEEE Transactions on Signal Processing* **68**, 3860–3870 (2020). <https://doi.org/10.1109/TSP.2020.3003120>
- [32] Truong, D.P., Skau, E., Desantis, D., Alexandrov, B.: Boolean matrix factorization via nonnegative auxiliary optimization. *IEEE Access* **9**, 117169–117177 (2021). <https://doi.org/10.1109/ACCESS.2021.3107189>
- [33] Ren, W., Li, G., Tu, D., Jia, L.: Nonnegative matrix factorization with regularizations. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **4**(1), 153–164 (2014). <https://doi.org/10.1109/JETCAS.2014.2298290>
- [34] Qin, Y., Li, B., Ni, W., Quan, S., Wang, P., Bian, H.: Affinity matrix learning via nonnegative matrix factorization for hyperspectral imagery

- clustering. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **14**, 402–415 (2021). <https://doi.org/10.1109/JSTARS.2020.3040218>
- [35] Guo, Z., Zhang, Y.: A label-embedding online nonnegative matrix factorization algorithm. *IEEE Access* **7**, 105882–105891 (2019). <https://doi.org/10.1109/ACCESS.2019.2932420>
- [36] Wang, J., Guan, S., Liu, S., Zhang, X.-L.: Minimum-volume multichannel nonnegative matrix factorization for blind audio source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **29**, 3089–3103 (2021). <https://doi.org/10.1109/TASLP.2021.3120603>
- [37] Al-Sharoha, E.M., Aviyente, S.: Community detection in fully-connected multi-layer networks through joint nonnegative matrix factorization. *IEEE Access* **10**, 43022–43043 (2022). <https://doi.org/10.1109/ACCESS.2022.3168659>
- [38] Shi, M., Yi, Q., Lv, J.: Symmetric nonnegative matrix factorization with beta-divergences. *IEEE Signal Processing Letters* **19**(8), 539–542 (2012). <https://doi.org/10.1109/LSP.2012.2205238>
- [39] Jia, Y., Liu, H., Hou, J., Kwong, S., Zhang, Q.: Self-supervised symmetric nonnegative matrix factorization. *IEEE Transactions on Circuits and Systems for Video Technology* **32**(7), 4526–4537 (2022). <https://doi.org/10.1109/TCSVT.2021.3129365>
- [40] Chen, B.-W.: Symmetric nonnegative matrix factorization based on box-constrained half-quadratic optimization. *IEEE Access* **8**, 170976–170990 (2020). <https://doi.org/10.1109/ACCESS.2020.3023557>
- [41] Lee, H., Yoo, J., Choi, S.: Semi-supervised nonnegative matrix factorization. *IEEE Signal Processing Letters* **17**(1), 4–7 (2010). <https://doi.org/10.1109/LSP.2009.2027163>