Neurocomputing 297 (2018) 71–81

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Incorporating network structure with node contents for community detection on large networks using deep learning

Jinxin Cao[a], Di Jin[a,*], Liang Yang[c], Jianwu Dang[a,b]

[a] School of Computer Science and Technology, Tianjin University, Tianjin 300350, China
[b] School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan
[c] School of Computer Science and Engineering, Hebei University of Technology, Tianjin 300401, China

ARTICLE INFO

Article history:
Received 3 June 2017
Revised 19 December 2017
Accepted 28 January 2018
Available online 2 February 2018

Communicated by Prof. FangXiang Wu

Keywords:
Community detection
Deep learning
Spectral clustering
Modularity maximization
Normalized-cut
Node contents

ABSTRACT

Community detection is an important task in social network analysis. In community detection, in general, there exist two types of the models that utilize either network topology or node contents. Some studies endeavor to incorporate these two types of models under the framework of spectral clustering for a better community detection. However, it was not successful to obtain a big achievement since they used a simple way for the combination. To reach a better community detection, it requires to realize a seamless combination of these two methods. For this purpose, we re-examine the properties of the modularity maximization and normalized-cut models and fund out a certain approach to realize a seamless combination of these two models. These two models seek for a low-rank embedding to represent of the community structure and reconstruct the network topology and node contents, respectively. Meanwhile, we found that autoencoder and spectral clustering have a similar framework in their low- rank matrix reconstruction. Based on this property, we proposed a new approach to seamlessly combine the models of modularity and normalized-cut via the autoencoder. The proposed method also utilized the advantages of the deep structure by means of deep learning. The experiment demonstrated that the proposed method can provide a nonlinearly deep representation for a large-scale network and reached an efficient community detection. The evaluation results showed that our proposed method outperformed the existing leading methods on nine real-world networks.

1. Introduction

The development of Internet has led to producing more and more variety of data, such as online comments, product reviews and co-author networks, which have affected all aspects of people's lives, and thus the analysis of those data has attracted more and more attention of researchers in various fields. One hot topic in the studies of such social media or online data is to discovery the underlying structure with group effect, which is the so-called community structure. The vertices (users) related to the communities in the network can be divided into groups, in which vertices have more multiple connection but the connections are relatively sparse in the whole network. Those individuals or users belonging to the same community share common profiles or have common interests. The identification of communities consisting of users with similarity is very important, and has been applied in many areas, e.g., sociology, biology and computer science. For example, in biology, some different units belonging to an organization have some related functions that are interconnected with special structures to characterize the whole effect of the organization. The interaction among a set of proteins in a cell can form an RNA polymerase for transcription of genes. In computer science, finding the salient communities from an organization of people can create a guide which helps to web marketing, behavior prediction of users belonging to a community and understanding the functions of a complex system [1].

For community detection, some methods have put forward, which can be cast as graph clustering. In normalized cut (n-cut) [4], the Laplacian matrix is the main objective to be processed. The eigenvectors with a non-zero eigenvalue, which are obtained by the eigenvalue decomposition (EVD) of graph Laplacian matrix, are treated as graph representation. Some other works can be also transformed to spectral clustering. For example, the modularity maximization model [10] first constructs a graph that is based on feature vectors, and then solves the top k eigenvectors as network representation for clustering. Here, we can deem modularity matrix as graph Laplacian matrix. We realize that, those two methods (i.e. modularity optimization and n-cut) can easily

* Corresponding author.
E-mail addresses: alfred718china@tju.edu.cn (J. Cao), jindi@tju.edu.cn (D. Jin), yangliang@iie.ac.cn (L. Yang), jdang@jaist.ac.jp (J. Dang).

https://doi.org/10.1016/j.neucom.2018.01.065
0925-2312/

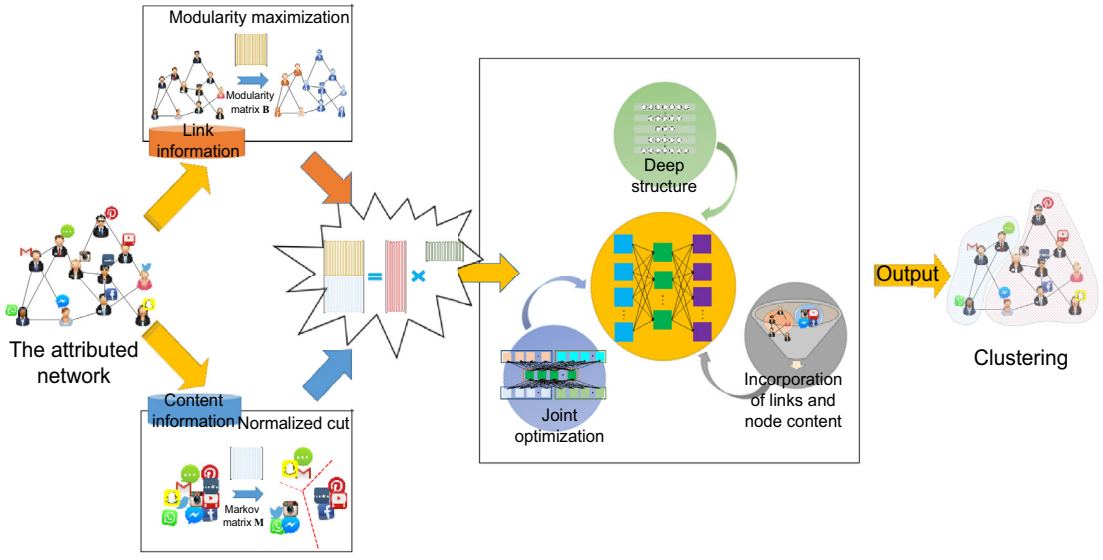


Fig. 1. The proposed framework. For community discovery on the attributed network, our input data consists of two parts which are modularity matrix and Markov matrix for the topological structure and node contents, respectively. Because of the similarity between spectral methods and autoencoder in terms of an approximation of the matrix reconstruction, the network representation is denoted by the low-dimensional encoding in the hidden layer of an autoencoder block. Finally, we achieve the fusion of network topology and node contents for community detection. Furthermore, we can append many autoencoder blocks and build a multiple layer deep neural network framework.

capture topology-based and content-based features by EVD of the corresponding spectral matrices separately, as shown in Fig. 1. However, those methods for community detection are often limited to obtain the important information about the structure of the communities in networks. It demonstrates that one of the techniques can overcome the problem, which only consider topological structure (or node contents), by fusing the vertex information (or say node contents) with linkage information for community detection [2].

When considering both topological and content information for community discovery, we can combine these two objective functions into one in the form of linearity directly. However, some classical graph embedding methods, such as locally linear embedding (LLE) [11], show that the relations among vertices in the real-world networks are not certainly linear. So, the model based on this linearly combination strategy is still limited on real-world networks. Moreover, although we could get a network representation by fusing those two types of information, the problem in the optimization of the combination model is, the low efficiency for deciding an appropriate ratio of two kinds of information due to manual tuning such ratios.

In the recent years, deep learning is used in many areas, such as speech recognition [6], image classification [7] and so on. As we known, neural network is a good framework for nonlinear computation with the elements that simulate the structure and properties of neurons [8]. Among them, autoencoder is proposed by Ng [5], which aims to obtain features from the input data. We found that autoencoder and spectral methods all intent to obtain the low-dimensional approximation of the corresponding matrix. Based on this similarity, we adopt autoencoder as a breakpoint method to solve the disadvantages of linear optimization, and to achieve the incorporation of these two different spectral methods.

In order to not only take the advantage of spectral methods but also achieve the incorporation of linkage and node content information, we propose an autoencoder-based method for community detection using the normalized-cut and modularity maximization. Our work is inspired by the similarity in theory between autoencoder and spectral methods in terms of getting an intrinsic structure of the spectral matrix. The framework of our main idea is shown in Fig. 1. We realized that autoencoder is a type of unsupervised learning methods, and thus only treat the low-dimensional encoding in the hidden layer as the network representation. In our method, we adopt modularity maximization model and normalized-cut to portray linkage and content information, separately, and construct the spectral matrices (i.e. modularity matrix and Markov matrix) as the input of the autoencoder. We design a unified objective function to get the best reconstruction of the combination matrix that consists of modularity matrix and Markov matrix, while make use of autoencoder to get a best encoding in the hidden layer as the network representation which is used to finding communities nicely. Furthermore, by building a multi-layers autoencoder, we adopt deep autoencoder to obtain a powerful representation by means of the deep structure, and combine with the intrinsic information of the original data to achieve an improvement for discovering communities. In total, our framework has three main contributions as follows:

- First, in theory both the autoencoder and spectral methods are related to the low-dimensional approximation of the specified corresponding matrix, i.e. the modularity matrix and Markov matrix. This study utilizes the autoencoder to obtain a low-dimensional encoding which can best reconstruct the joint matrix consisting of the modularity matrix and the Markov matrix, and treats this encoding as the graph representations for community detection. The important point is to propose an autoencoder-based method that can achieve the joint optimization of modularity model and normalized-cut without a seam.
- Second, this encoding supplies a nonlinear way to integrate the linkage and content information. This helps to further improve the performance of community detection, when using both those two types of information. The autoencoder not only encodes the important factors of the data in the process of reducing the dimension, but also automatically learns the weight of the relationship among the various factors to obtain the minimum of reconstruction error. In this framework, therefore, the performance improvement of our method is realized by its self-tuning characteristic, but not depend on adjusting balance factor.

- Furthermore, by stacking a series of autoencoders, we built a multi-layer autoencoder in favor of enhancing better generalization ability of the encoding in the hidden layer. Benefitting from the deep structure, we get a powerful encoding with both topological and content information, which can effectively aid network community detection.

The rest of the paper is organized as follows. In Section 2, we give a brief review of the related work. The proposed framework and the relevant algorithms are introduced in Section 3. Next, datasets and experimental setting are described in Section 4, and followed experimental evaluation and the analysis of the balance factor in this Section demonstrate the effectiveness of the proposed new method. The paper is then concluded in Section 5.

2. Related work

There exist three aspects of relevant works regarding the topic here, which are community detection with topological structure or content information alone, and the combination of links and node contents. As described above, it is not appropriate that node community memberships are denoted by using the network topology or content information alone. Combining topology and content achieves an improvement for community detection, as showed in studies [1,2,3,17,19,20]. However, they utilized those two types of information while did not take into account of their nonlinear combination. Besides, although those studies considered both linkage and content information, they usually modeled two different types of information separately, and then balanced these types of information by using a manually adjusted hyper-parameter. This may keep them from the optimal improvement.

As well-known, the conventional paradigm for generating features for vertices is based on feature extraction techniques which typically involve some features based on network properties. Among them, spectral approaches such as normalized cut [4] and modularity maximization model [10] exploit the spectral properties of various matrix representations of graphs, especially the Laplacian and adjacency matrices. These methods can be taken as the dimensionality reduction techniques under linear perspective, while they suffer from both nonlinear computation and statistical performance drawbacks. In terms of the collaborative optimization, spectral methods are deemed to as eigenvalue decomposition (EVD) of a data matrix, however, it is hard to design a unified framework or a general model fusing the different objective functions of the spectral methods without a seam.

As for deep learning, it has achieved great success in many applications. This may because deep neural network (DNN) can describe a complicated object using a huge nonlinear space and also have better generalization. Recently, there are several studies on community detection using deep learning methods, such as research works described in [21,34]. Although they improve the performance via DNN, only the structural or node content information was taken into account in their studies. They did not utilize the most important advantage of the DNN, which can combine the different modalities altogether. For this reason, we focus on incorporating those two kinds of information together for community detection via the DNN framework.

3. Framework of community detection using deep learning

To fully utilize the advantages of deep neural network (DNN) for combining network topology and content information, we re-examine the properties of the modularity maximization model and normalized cut, which are the leading models for community detection, and re-search the DNN framework to find out a certain approach appropriate to realize a seamless combination of the different modalities. These two models seek for a low-rank embedding to represent of the community structure and reconstruct the network topology and node contents respectively. In the DNN framework, we found that autoencoder has similar properties to spectral clustering in low-rank matrix reconstruction. Based on this property, we proposed a new approach to seamlessly combine the models of modularity and normalized-cut via the autoencoder. The autoencoder-based method is able to take the advantages of spectral methods and deep representation of the DNN, so that achieves the joint optimization of modularity and normalized-cut for community detection based on both links and node contents.

3.1. Spectral methods for community detection

Many community detection methods have been proposed. Among them, one type of the most popular methods are the spectral methods. Here, we explain two spectral approaches: modularity maximization and normalized cut.

3.1.1. Modularity maximization model

The topological structure of a network can be treated as a graph $G=(V, E)$, where V is the collection of vertices containing n nodes $\{v_1,\ldots,v_N\}$, and E the set of the edges in which each edge connects two vertices in V. We treat a nonnegative symmetric binary matrix $\mathbf{A}=\{a_{ij}\}\in\mathbb{R}_{+}^{N\times N}$ as the adjacency matrix of G, and set $a_{ij}=1$ if there is an edge between vertex v_i and v_j, and $a_{ij}=0$ otherwise.

Here, we make use of the modularity model to deal with community detection with network topology alone. This model is optimized through maximizing modularity, which means the division between the number of edges in the original graph and the expected number of edges in a random graph without community structure. We can define modularity matrix as $\mathbf{B}\in\mathbb{R}^{N\times N}\mathbb{R}$ with elements $b_{ij}=a_{ij}-k_ik_j/2m$, in which k_i is the degree of vertex v_i and m the number of edges. Newman [10] also relaxed this problem when considering a network with C communities, and then modularity maximization can be transformed to the follow optimization problem:

$$\begin{aligned}&\min_{\mathbf{X}\in\mathbb{R}^{N\times C}} tr\left(\mathbf{X}^T\mathbf{B}\mathbf{X}\right)\\ &s.t.\quad tr\left(\mathbf{X}^T\mathbf{X}\right)=N\end{aligned}\tag{1}$$

where $tr(\cdot)$ denotes the trace of a matrix, and $\mathbf{X}=\{x_{ij}\}\in\mathbb{R}^{N\times C}$ is an indicator matrix. Based on Rayleigh Quotient [34], the above problem can be solved by extracting the largest C eigenvectors of the modularity matrix $\mathbf{B}$. In this way, the optimization of modularity is taken as the eigenvalue decomposition of modularity matrix in terms of the spectral methods.

3.1.2. Normalized-cut

As to the content information of the network, we portray the content on vertex v_i by using a multi-dimensional word vector. We use $\mathbf{S}=\{s_{ij}\}\in\mathbb{R}^{N\times N}$ to represent the similarity matrix of the graph G, and s_{ij} is measured by using the cosine similarity between the word vector of the vertex v_i and that of the vertex v_j. Also, we have taken into account of the manifold structure of node content space, and dealt with the original similarity matrix by using *k-near neighbor consistency* method in [11]. It preserves the similarity value s_{ij} between vertex v_i and each of its k-near neighbor vertices and 0 otherwise.

By now, we got a similarity graph G in the form of similarity matrix $\mathbf{S}$ in which s_{ij} denotes the weight of the edge between vertices v_i and v_j. So the normalized-cut is defined so that, we want to cut the least edges to achieve that vertices belonging to the same partition have a high similarity while the vertices between different partitions have a low similarity. In this process, we measure the degree of vertices in a subset g of graph G by using $vol(g)=\sum_{v_i\in c}s_{ij}$. When considering to divide the graph G into C

groups, the normalized-cut (n-cut) is treated as the optimization problem in the following:

$$\begin{aligned} &\min_{\mathbf{Y}\in\mathbb{R}^{N\times C}} tr\left(\mathbf{Y}^T\mathbf{L}\mathbf{Y}\right) \\ &s.t. \quad \mathbf{Y}^T\mathbf{D}\mathbf{Y}=\mathbf{I} \\ &y_{ij}=\begin{cases}\dfrac{1}{\sqrt{vol\left(g_j\right)}}, & if \quad v_i\in g_j \\ 0, & otherwise\end{cases}\end{aligned} \tag{2}$$

Here, let $s_i=\sum_j s_{ij}$ be the diagonal elements of the diagonal matrix $\mathbf{D}$, and the graph Laplacian matrix is set to $\mathbf{D}^{-1}\mathbf{L}$. Like the optimization of modularity model, by relaxing the optimization problem, the indication matrix $\mathbf{Y}=\{y_{ij}\}\in\mathbb{R}^{N\times C}$ can be produced by eigenvalue decomposition of the Laplacian matrix $\mathbf{D}^{-1}\mathbf{L}$. Notice that $\mathbf{D}^{-1}\mathbf{L}=\mathbf{D}^{-1}(\mathbf{D}-\mathbf{S})=\mathbf{I}-\mathbf{D}^{-1}\mathbf{S}$. According to eigenvalue decomposition of the matrix $\mathbf{D}^{-1}\mathbf{S}$, the matrix $\mathbf{Y}$ is represented by the eigenvectors corresponding to the top C eigenvalues of the matrix $\mathbf{D}^{-1}\mathbf{S}$. Here, we set $\mathbf{M}=\mathbf{D}^{-1}\mathbf{S}$ which is the so-called Markov matrix.

3.2. Community detection via autoencoder reconstruction

As described in Section 3.1, both modularity maximization model and normalized cut can be converted into eigenvalue decomposition of their corresponding matrices. The *Eckart-Young-Mirsky Theorem* [9] illustrates this point that the eigenvalue decomposition of spectral matrix is referred to the low-rank approximation of a matrix.

Theorem 1. *(Eckart-Young-Mirsky) Given a matrix $\mathbf{\Theta}\in\mathbb{R}^{m\times n}$ with a rank-κ, let the singular value decomposition (SVD) of $\mathbf{\Theta}$ be $\mathbf{\Theta}=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$ where $\mathbf{P}^T\mathbf{P}=\mathbf{I}$ and $\mathbf{Q}^T\mathbf{Q}=\mathbf{I}$. We can optimize this problem*

$$\begin{aligned} &\arg\min_{\mathbf{\Theta}\in\mathbb{R}^{m\times n}} \|\mathbf{\Theta}-\mathbf{\Theta}^*\|_F^2 \\ &s.t. \quad rank(\mathbf{\Theta}^*)=\kappa \end{aligned} \tag{3}$$

by using $\mathbf{\Theta}^*=\mathbf{P}\mathbf{\Sigma}^*\mathbf{Q}^{\mathrm{T}}$. Here, the singular value matrix $\mathbf{\Sigma}$ only contains the κ largest singular values and 0 otherwise.

It explains that the SVD of a matrix aims to obtain the best low-rank approximation of the original matrix. In special, the modularity matrix $\mathbf{B}$ is symmetric. The SVD of $\mathbf{B}$ is accomplished by $\mathbf{B}=\mathbf{P}\mathbf{\Psi}\mathbf{P}^T$, regarded as the orthogonal decomposition of $\mathbf{B}$ where $\mathbf{P}^T\mathbf{P}=\mathbf{I}$ and the diagonal elements of the diagonal matrix $\mathbf{\Psi}$ are corresponding to the eigenvalues of $\mathbf{B}$ though the eigenvalue decomposition (EVD) of $\mathbf{B}$. Thus, the indicator matrix $\mathbf{X}$ can be obtained by constructing the best rank-k approximation of the matrix $\mathbf{B}$ under the Frobenuis norm.

As mentioned above, the eigenvalue decomposition is highly related to autoencoder in terms of matrix reconstruction. And thus, here we introduce an autoencoder with three layers to obtain a low-dimensional encoding that often leads to the best reconstruction of the modularity matrix $\mathbf{B}$. In this autoencoder, from the first layer to the hidden layer, the encoder is responsible to map the input $\mathbf{B}$ into a low-dimensional encoding $\mathbf{H}^{\mathrm{l}}=\{h_{ij}^{\mathrm{l}}\}\in\mathbb{R}^{r\times N}$ in which the ith column $\mathbf{h}_i^{\mathrm{l}}$ denotes a representation corresponding to vertex v_i in the latent space.

$$\mathbf{h}_i^{\mathrm{l}}=\phi(\mathbf{b}_i)=sig(\mathbf{W}\cdot\mathbf{b}_i+\mathbf{d}) \tag{4}$$

From the hidden layer to the third layer, the decoder maps the latent representation $\mathbf{H}^{\mathrm{l}}$ to the reconstruction of the input data,

$$\mathbf{b}_i^*=\varphi\left(\mathbf{h}_i^{\mathrm{l}}\right)=sig\left(\mathbf{W}^*\cdot\mathbf{h}_i^{\mathrm{l}}+\mathbf{d}^*\right) \tag{5}$$

where $\mathbf{b}_i$ and $\mathbf{b}_i^*$ means the ith column of the modularity matrix $\mathbf{B}$ and that of the reconstruction matrix $\mathbf{B}^*$ respectively, and $sig(\cdot)$ is an nonlinear mapping function (we adopt $sig(x)=\frac{1}{1+e^{-x}}$ in this article). The weight matrices $\mathbf{W}\in\mathbb{R}^{r\times N}$, $\mathbf{W}^*\in\mathbb{R}^{N\times r}$ and the bias vectors $\mathbf{d}\in\mathbb{R}^{r\times 1}$, $\mathbf{d}^*\in\mathbb{R}^{N\times 1}$ are all to be learned in the autoencoder, where r is the number of nodes in the hidden layer and N the number of nodes in the first or the third layer. The autoencoder can find the low-dimensional representation of a matrix by minimizing the reconstruction error between the data in the first layer and that in the third layer.

$$\breve{\delta}_{\mathbf{B}}=\arg\min_{\breve{\delta}_{\mathbf{B}}}\mathrm{O}(\mathbf{B},\mathbf{B}^*)=\arg\min_{\breve{\delta}_{\mathbf{B}}}\sum_{i=1}^{N}\mathrm{O}(\mathbf{b}_i,\varphi(\phi(\mathbf{b}_i))) \tag{6}$$

where $\breve{\delta}_{\mathbf{B}}=\{\mathbf{W},\mathbf{d},\mathbf{W}^*,\mathbf{d}^*\}$ is the parameter of the autoencoder-based model, and N the number of nodes in the first layer. After training the autoencoder, one can obtain the parameters $\mathbf{W}$ and $\mathbf{d}$, and then get a low-dimensional encoding based on Eq. (4) which is regarded as the graph representation of this network.

On the other hand, we can also utilize Markov matrix $\mathbf{M}$ as the input of autoencoder that achieves community detection with content information. In this autoencoder, the corresponding loss function of this reconstruction can be written as

$$\breve{\delta}_{\mathbf{M}}=\arg\min_{\breve{\delta}_{\mathbf{M}}}\mathrm{O}(\mathbf{M},\mathbf{M}^*)=\arg\min_{\breve{\delta}_{\mathbf{M}}}\sum_{i=1}^{N}\mathrm{O}(\mathbf{m}_i,\varphi(\phi(\mathbf{m}_i))) \tag{7}$$

where $\breve{\delta}_{\mathbf{M}}=\{\mathbf{W},\mathbf{d},\mathbf{W}^*,\mathbf{d}^*\}$ denotes the parameter of the autoencoder-based model to be learned.

3.3. Combination of links and node contents for community detection via autoencoder reconstruction

As described above, either modularity maximization or normalized cut aims to find the low-dimensional representation of the graph representation in the latent space. Their optimization problems are as follows:

$$\arg\min_{\mathbf{B}^*\in\mathbb{R}^{N\times N}}\|\mathbf{B}-\mathbf{B}^*\|_F^2 \tag{8}$$

$$\arg\min_{\mathbf{M}^*\in\mathbb{R}^{N\times N}}\|\mathbf{M}-\mathbf{M}^*\|_F^2 \tag{9}$$

So we can adopt the orthogonal decomposition $\mathbf{B}^*=\mathbf{X}\mathbf{\Sigma}_{\mathbf{B}}\mathbf{X}^T$ and $\mathbf{M}^*=\mathbf{Y}\mathbf{\Sigma}_{\mathbf{M}}\mathbf{Y}^T$ to solve (8) and (9), respectively.

For community detection with both network topology and node contents, we design a combination matrix $\mathbf{Z}=[\mathbf{B},\mathbf{M}]^T$ which consists of the modularity matrix $\mathbf{B}$ for linkage information as well as Markov matrix $\mathbf{M}$ for content information, and then get a low-dimensional encoding that can best lead to approximation of the combination matrix for the network representation.

$$\arg\min_{\mathbf{Z}^*\in\mathbb{R}^{2N\times N}}\|\mathbf{Z}-\mathbf{Z}^*\|_F^2=\arg\min_{[\mathbf{B}^*,\mathbf{M}^*]^T}\left\|[\mathbf{B},\mathbf{M}]^T-[\mathbf{B}^*,\mathbf{M}^*]^T\right\|_F^2 \tag{10}$$

According to the explanation in [36], we can construct approximation in terms of matrix factorization $\mathbf{Z}^*=\mathbf{P}\mathbf{H}$ where all columns of matrix $\mathbf{P}$ are denoted as the basis features and all columns of $\mathbf{H}$ mean the low-dimensional representation for data $\mathbf{Z}$. And thus, we convert the optimization problem in (10) into

$$\arg\min_{\mathbf{H}\in\mathbb{R}_+^{r\times N}}\|\mathbf{Z}-\mathbf{P}\mathbf{H}\|_F^2$$

By now, we got a new matrix reconstruction problem for community detection using the rich data consisting of both network topology and node contents.

Based on the above discussions, we first adopt an autoencoder to obtain a low-dimensional encoding of the combination matrix $\mathbf{Z}$ while we achieve collective optimization of those two different objectives. We then deal with each column of the combination matrix $\mathbf{Z}$ by fusing the topology-based and the content-based "features", and thus the obtained representation $\mathbf{H}^{\mathrm{a}}$ provides a good way of combination of those two types of information.

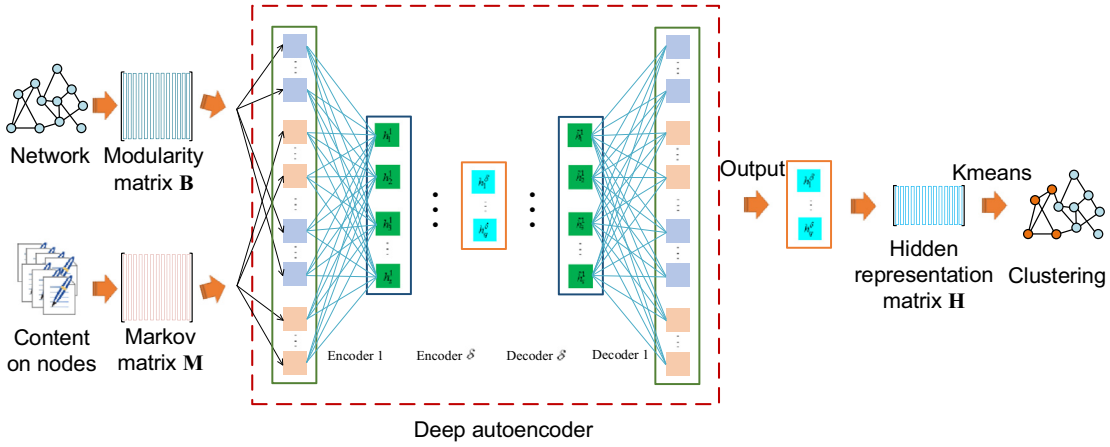


Fig. 2. Construction of a multiple layer deep autoencoder framework for obtaining the deep representation of the graph. The input matrix **Z** consists of two parts, modularity matrix **B** and Markov matrix **M**, where the low-dimensional encoding in the hidden layer of an autoencoder block is treated as a community-oriented graph representation. The deep autoencoder structure used here does not necessarily consist of only two stacked autoencoders. It can adopt three or more stacked autoencoders. In the experiment, the detailed information of the structure of the deep autoencoders is shown in Section 4.5.

So we use the combination matrix **Z** as the input of our autoencoder. The encoder maps the input data to a low-dimensional encoding $\mathbf{H}^{\mathrm{a}} = \{\mathbf{h}_i^{\mathrm{a}}\} \in \mathbb{R}^{r\times N}$ in which its i-th column $\mathbf{h}_i^{\mathrm{a}}$ represents a representation related to vertex v_i. In particular, the ith column of the combination matrix is denoted as $\mathbf{z}_i = [\mathbf{b}_i^T, \mathbf{m}_i^T]^T$. And the encoding can be written as

$$\mathbf{h}_i^{\mathrm{a}} = \phi(\mathbf{z}_i) = sig(\mathbf{W}\cdot\mathbf{z}_i + \mathbf{d}) \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{r\times 2N}$ is the weight matrix, $\mathbf{d} \in \mathbb{R}^{r\times 1}$ the bias vector, and $sig(\cdot)$ nonlinear mapping function $sig(x) = \frac{1}{1+e^{-x}}$. Otherwise, the decoder maps a representation $\mathbf{H}^{\mathrm{a}}$ to the reconstruction of the input matrix **Z**, as follows:

$$\mathbf{z}_i^* = \varphi\left(\mathbf{h}_i^{\mathrm{a}}\right) = sig(\mathbf{W}^*\cdot\mathbf{h} + \mathbf{d}^*) \tag{12}$$

And the parameters $\mathbf{W}^* \in \mathbb{R}^{2N\times r}$ and $\mathbf{d}^* \in \mathbb{R}^{2N\times 1}$ need to be learned in the decoder. So in the autoencoder, we can obtain the best low-dimensional representation by optimizing the following problem

$$\delta = \arg\min_{\delta} \mathrm{O}(\mathbf{Z}, \mathbf{Z}^*) = \arg\min_{\delta} \sum_{i=1}^{N} \mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i))) \tag{13}$$

where $\delta = \{\mathbf{W}, \mathbf{d}, \mathbf{W}^*, \mathbf{d}^*\}$ is the set of the parameters to be learned in the optimization of this proposed model. The workflow of the proposed framework is as follows.

3.4. Optimization of the autoencoder-based model

We apply stochastic gradient descent [35] to solve the problem in (13). To be specific, the parameters {**W, d, W***, **d***} of the model are updated in each iteration according to:

$$W_{ij} = W_{ij} - \eta\cdot\frac{\partial}{\partial W_{ij}}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*), \qquad d_i = d_i - \eta\cdot\frac{\partial}{\partial d_i}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*),$$
$$W_{ij}^* = W_{ij}^* - \eta\cdot\frac{\partial}{\partial W_{ij}^*}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*), \qquad d_i^* = d_i^* - \eta\cdot\frac{\partial}{\partial d_i^*}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*),$$

Because of the similarity in the inference processing of **W, d** and **W***, **d*** we only show the inference of the update rules for the parameters **W, d** as follows:

$$\frac{\partial}{\partial W_{ij}}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*) = \sum_{i=1}^{N}\frac{\partial}{\partial W_{ij}}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i)))$$
$$= \sum_{i=1}^{N}\frac{\partial}{\partial h_i^{\mathrm{a}}}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i)))\frac{\partial}{\partial W_{ij}}h_i^{\mathrm{a}} = \sum_{i=1}^{N}\rho_i\mathbf{z}_i^T,$$

$$\frac{\partial}{\partial d_i}\mathrm{O}(\mathbf{Z}, \mathbf{Z}^*) = \sum_{i=1}^{N}\frac{\partial}{\partial d_i}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i)))$$
$$= \sum_{i=1}^{N}\frac{\partial}{\partial h_i^{\mathrm{a}}}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i)))\frac{\partial}{\partial d_i}h_i^{\mathrm{a}} = \sum_{i=1}^{N}\rho_i,$$

where $\mathbf{h}^{\mathrm{a}} = \mathbf{W}\cdot\mathbf{z}_i + \mathbf{d}$, and $\rho_i = \frac{\partial}{\partial h_i^{\mathrm{a}}}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i)))$ denotes each node "responsible" for the whole error. To measure the difference between the input data **Z** and the reconstruction **Z***, ρ_i in the encoder is written as

$$\rho_i = \frac{\partial}{\partial h_i^{\mathrm{a}}}\mathrm{O}(\mathbf{z}_i, \varphi(\phi(\mathbf{z}_i))) = -\sum_{i=1}^{N}\frac{\partial}{\partial h_i^{\mathrm{a}}}\mathrm{O}\left(Z_{ij} - Z_{ij}^*\right)\cdot sig'\left(h_i^{\mathrm{a}}\right),$$

While, in the decoder, we have $\boldsymbol{\omega}_i = \mathbf{W}^*\cdot\mathbf{h} + \mathbf{d}$ and the "responsibility" ρ_i^* is

$$\rho_i^* = \left(\sum_{i=1}^{r} W_{ij}^*\cdot\rho_i\right)\cdot sig'(z_i),$$

where $sig'(\cdot)$ is obtained by taking the derivative of the function $sig(\cdot)$.

3.5. Construction of deep autoencoder

Deep learning is well known for its powerful data representation capability. By training the neural network layer by layer, the representation produced by each layer will better act on the next layer. To benefit from the deep structure, we build a deep autoencoder with a series of autoencoders, and train the model in the first autoencoder by reconstructing the input matrix **Z**, and then get a best latent representation $\mathbf{H}^1 \in \mathbb{R}^{r^1\times N}$. And finally, we train the next autoencoder by reconstructing the $\mathbf{H}^1$ from the above autoencoder, and get a new latent representation $\mathbf{H}^2 \in \mathbb{R}^{r^2\times N}$. In this deep architecture, the number of nodes in the next autoencoder is less than that in the above autoencoder, i.e. $r^2 > r^1$. The workflow of this deep framework is shown as Table 1. Besides, we show the configuration of the number of nodes in the deep autoencoder in Section 4.5.

In the following, we analyze the computational complexity of the proposed new algorithm. For a standard multilayer perceptron, which is the key component of the autoencoder, matrix multiplications occupy most of the computational time. In a single layer with

Table 1
Our algorithm framework.

Algorithm

Input
$\mathbf{B} = \{b_{ij}\} \in \mathbb{R}^{N\times N}$: Modularity matrix.
$\mathbf{M} = \mathbf{D}^{-1}\mathbf{S}$: Markov matrix.
Output
$\mathbf{H}^{\Delta} \in \mathbb{R}^{r^{\Delta}\times N}$: The network representation.
1. Set Δ: DNN layers number, $r^1 = 2N$, r^{σ}: The number of nodes in layer σ, $\mathbf{H}^1 = \mathbf{Z} = [\boldsymbol{B}, \boldsymbol{M}]^T$, $\mathbf{H}^{\sigma} \in \mathbb{R}^{r^{\sigma}\times N}$ is the input to layer σ.
2. Put the data matrix $\mathbf{H}^1$ into the deep autoencoder.
For $\sigma = 1$ to Δ
Build an autoencoder block with three layers, and put the data $\mathbf{H}^{\sigma}$ into it.
Train the autoencoder and obtain the hidden layer $\mathbf{u}^{\sigma}$ using the function (13).
Set $\mathbf{H}^{\sigma} = \mathbf{u}^{\sigma}$.
End

Table 2
Real-world datasets. "undirected" means that this is an undirected graph, and "0/1" denotes that the node content information is described by the *d*-dimensions binary vector. V denotes the number of nodes, E the number of edges, and C the number of communities. Here we applied all datasets uniformly in form of the undirected and unweighted graph.

Dataset	V	E	Type of G	F	Type of F	C
Twitter629863	171	796	directed	578	0/1	9
Texas	187	328	undirected	1,703	0/1	5
Washington	230	446	undirected	1,703	0/1	5
Wisconsin	265	530	undirected	1,703	0/1	5
Facebook107	1045	26,749	directed	576	0/1	7
Cora	2708	5,429	undirected	1433	0/1	7
Citeseer	3312	4,732	undirected	3703	0/1	6
Uai2010	3363	45,006	directed	4972	0/1	19
PubMed	19,729	44,338	directed	500	0/1	3

N input dimensions and $N/2$ output dimensions, back-propagation algorithm requires $O(N^2d/2)$ floating-point calculations, where d ($d << N$) is the number of mini-batchs. Besides, when the nodes are sparsely connected, the time complexity of back-propagation is reduced to be $O(NMd/2)$, in which M ($M << N$) is the number of links. And thus this step needs $O(N)$ time. Since the output functions of neurons are in general very simple to compute, one can assume that those costs are constant per neuron. Also, because the given neural network has N neurons, this step needs $O(N)$ time. Finally, the proposed new algorithm has the computational complexity $O(N)$ which is linear to the number of nodes in large graphs.

4. Experiments

Here we give the comparisons between our algorithm and some state-of-the-art community detection algorithms on a wealth of real-world networks. There are also some detailed descriptions on the baseline methods, networked datasets and experimental setups.

4.1. Datasets description

We test the performance of our method on nine publicly-available datasets in which each dataset is described as follows, and the key information of those datasets are listed in Table 2.

Citeseer [22]: This is a citation network of computer science publications that are labeled as six sub-fields, and thus we set the number of communities 6. In our predefined graph, nodes stand for publications and undirected edges indicate the citation relationships. Content information on each node is stemmed words from the research paper, which is represented as a 3703 dimensions' binary vector for a document. To be specific, in the word vector $f(v_i)$ of node v_i, $f(v_i) = 1$ means the jth word in the content on vertex v_i occurred, and $f(v_i) = 1$ otherwise.

Cora [22]: It is also a citation type dataset which includes science publications from 7 subfields of machine learning, so the number of clusters is set to 7 in the experiments.

PubMed [22]: The PubMed Diabetes dataset consists of 19,729 scientific publications from three classes. Each publication in this dataset is described by a *tf/idf* word vector with 500 unique words. In this paper, for the unification of dataset format, we adopt 500 dimensions' binary content feature vectors instead of *tf/idf* word vector, and set an element to 1 in the content feature vector corresponding to non-zero value of the *tf/idf* word vector.

Texas, *Washington*, *Wisconsin* [22]: These three subsets are all described by a lot of webpages and links are "generated" by some college students from those 3 Universities. These webpages are represented by vertices while the links among the webpages represent the edges. Vertices are labeled by webpages belonging to user's class. Here, each webpage in those three datasets is described by a 1703 dimensions' binary vector respectively.

Uai2010 [22]: This is a Wikipedia dataset of Wikipedia articles that appeared in the featured list. This dataset includes 3363 documents and 45,006 links. The documents are from 19 distinct categories and are represented by the *tf/idf* weighted vectors. Here, we make use of 4972 dimensions' binary feature vectors instead of the *tf/idf* vectors.

Facebook107, *Twitter629863* [23]: Those two datasets are extracted to form ego-networks. The ground-truth of communities is set up in accordance with the protocol as in [23]. Circles in those two datasets are defined as ground-truth. Thus, the number of groups on those two datasets is set to 7 and 9, respectively.

4.2. Baselines for comparison

The baselines can be divided into three types: (1) make use of network structural information alone, (2) consider node contents alone, and (3) utilize both those two types of information.

In the first class, CNM [24] and Louvian [25], which adopt the same greedy optimization as did by modularity maximization model, are used to partition network structure. Besides, Infomap [26], Linkcommn [27], Clique [13] and BigCLAM [28] are popular overlapping community detection methods also using network structure.

The second type focuses on modeling the contents on nodes, ignoring the network structure. In this class, we chose four methods which have their own characteristics, i.e., Affinity Propagation (AP) [29] and DP [30] are state-of-the-art non-parameter clustering algorithms, LDA [31] is a well-known topic model for data clustering algorithm, and MAC [14] is an overlapping community detection approach.

The third class of methods uses both network structure and node contents. They consist of two different types, this is the

overlapping and non-overlapping methods respectively. There are two overlapping algorithms which are CESNA [19] and DCM [32]. We also consider PCL-DC [33] and Block-LDA [12], which are non-overlapping approaches.

Finally, we also consider some autoencoder-based methods for comparison. We chose AE-link approach which is our method but using links alone, and AE-content approach which is also our approach but using node contents alone. To be specific, for AE-link we only set the modularity matrix as the input of the autoencoder, while AE-content is a normalized cut-based method optimized via autoencoder.

4.3. Description for metrics

In this paper, we focus on community detection with the predefined number of communities. We assume that there are C communities. So the community detection problem is simplified to divide the vertices into C different groups based on the network topology as well as content information. And thus, the community detection results are evaluated by measuring the degree of how the detected labels are consistent with the ground-truth communities. In the experiments, two sets of labels which are the ground-truth partitions C^* and the detected partitions $\bar{C}$ are considered. Because both overlapping and non-overlapping approaches exist in the baselines, we quantify the performance of the algorithms by using two types of performance metrics. One type includes normalized mutual information (*NMI*) and the accuracy (*AC*) for evaluating disjoint communities, and another type contains Jaccard similarity (*Jaccard*) and *F-scores* for evaluating overlapping communities.

1) *NMI* [15]: there are two sets of labels which are the ground-truth labels C^* and the detected labels $\bar{C}$ respectively. The entities c_{ij} of the matrix $\mathbf{C}$ denote the number of nodes belonging to group i of set C^*, which are also treated as the size of group j of set $\bar{C}$. So, the measure $NMI(C^*,\bar{C})$ can be written as

$$NMI\left(C^*,\bar{C}\right)=\frac{-2\cdot\sum_{i=1}^{c_{C^*}}\sum_{j=1}^{c_{\bar{C}}}c_{ij}\cdot\log\left(\frac{c_{ij}}{\mathbf{c}_{i\cdot}\mathbf{c}_{\cdot j}}\right)}{\sum_{i=1}^{c_{C^*}}\mathbf{c}_{i\cdot}\cdot\log\left(\frac{\mathbf{c}_{i\cdot}}{N}\right)+\sum_{j=1}^{c_{\bar{C}}}\mathbf{c}_{\cdot j}\cdot\log\left(\frac{\mathbf{c}_{\cdot j}}{N}\right)} \tag{14}$$

where the set $C^*(\bar{C})$ includes $c_{C^*}(c_{\bar{C}})$ communities. The value of $NMI(C^*,\bar{C})$ gets 1, when the detected labels $\bar{C}$ are agreement with the ground-truth labels C^*. Conversely, we obtain $NMI(C^*,\bar{C})=0$.

2) *AC* [15]: we also introduce a simple measure for evaluate the performance of algorithms:

$$AC\left(C^*,\bar{C}\right)=1-\frac{\left||C^*|-|\bar{C}|\right|}{|C^*|} \tag{15}$$

which is the relative error in predicting the number of communities.

3) *F-scores* [16]: we quantify the *F-scores* between C^* and $\bar{C}$ in term of measuring the error produced by using the partition $\bar{C}$ to predict the partition C^*. So the *F-scores* is:

$$F-scores\left(C^*,\bar{C}\right)=2\cdot\frac{precesion\left(C^*,\bar{C}\right)\cdot recall\left(C^*,\bar{C}\right)}{precesion\left(C^*,\bar{C}\right)+recall\left(C^*,\bar{C}\right)} \tag{16}$$

Like the text classification tasks, here we treat C^* as a set of "relevant" partitions, and $\bar{C}$ as a set of "retrieved" partitions, so the precision and recall are defined as:

$$precision\left(C^*,\bar{C}\right)=\frac{\left|C^*\cap\bar{C}\right|}{|C^*|},\ recall\left(C^*,\bar{C}\right)=\frac{\left|C^*\cap\bar{C}\right|}{\left|\bar{C}\right|}.$$

Jaccard [18]: considering the expensive computation, we estimate it as did in [19]. An unbiased estimator of Jaccard similarity of sets C^* and $\bar{C}$ can be obtained by

$$Jaccard\left(C^*,\bar{C}\right)=\frac{1}{\tau}\sum_{i=1}^{\tau}I\left(\min\left(\pi_i(C^*)\right)=\min\left(\pi_i\left(\bar{C}\right)\right)\right) \tag{17}$$

where $\pi_1,\ \pi_2,\ldots,\pi_\tau$ are τ permutations drawn randomly from a family of min-wise independent permutations defined on the universe C^* and $\bar{C}$ belong to, and I is the identity function.

4.4. Parameters setting

Many proposed methods considering both the topological and content information often rely on a balance factor to control the effectiveness of incorporating those two kinds of information. In our input matrix $[\mathbf{M},\ \mathbf{B}]^T$, we have a parameter λ, which is set to $0<\lambda<1$, to balance the proportion of those two types of information, i.e. $[(1-\lambda)\cdot\mathbf{M},\ \lambda\cdot\mathbf{B}]^T$. With different values of λ, we run 10 groups of the experiments on three datasets mentioned above, and get the accuracy curve with error bar. The results are shown in Fig. 3.

As we can see, when λ is set up to 0 or 1, the performance of our method gets a much lower accuracy than that of our method with $\lambda\in(0,\ 1)$. This confirms that, considering both topological and content information we can give a clearer view of community structure and help to achieve a big improvement for community detection. Furthermore, with the variation of λ in the area of [0.1, 0.9], the ratio between topological and content information is always variable, but the accuracy curve shows very small fluctuations. This also suggests that, the deep autoencoder provides a powerful capability for automatically tuning the balance factor. It may further mean that the encoding of the hidden layer in an autoencoder block can capture the most important information from the topological and content information, instead of collecting all information on the whole data. Thus, our method, different from others also using those two types of information, obtains better performance without adjusting the balance factor, and realizes the balance factor self-tuning. For this reason, we set $\lambda=0.5$ in our experiments in general.

As shown in Fig. 2, we run *Kmeans algorithm* on the low-dimensional encoding $\mathbf{H}$ to perform community detection. In general, many existing community detection approaches (especially those in the machine learning area) make the automatic determination of the number of communities as a separately solved problem, i.e. model selection, which has been discussed in many researches [37,38,39]. And also they require a given number (C) for the communities, so did the proposed new method in this work.

4.5. Setups in DNN structure

We implement the deep autoencoder structure with 2, 3, 4 and 5 layers on all datasets for detecting communities. The performance is shown in Fig. 4. In generally, the encoding of the deep autoencoder structure with 3 layers obtains a better performance of community detection than that of the deep autoencoder structure with 2 layers. This means that the deep structure plays an important role in yielding a good encoding for clustering. However, we also found that, in the Washington dataset, the encoding of the deep autoencoder structure with more layers got a lower *NMI* value. This may because, when the dimension of data decreases with the increase of the deep level of this structure, some important information hold in the data will be lost, causing the performance decrease. And thus, the deep autoencoder structure in this work utilize different number of stacked autoencoders for different real-world networks.

Table 3 demonstrates, on all datasets, the node number in each configuration in the deep autoencoders. There are 2 ∼ 4 layers

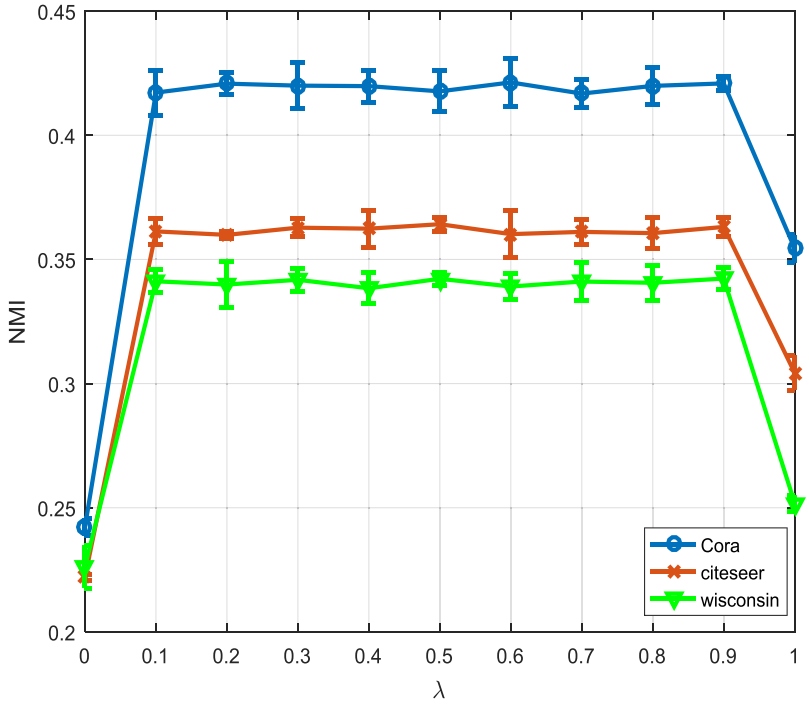


Fig. 3. The performance of community detection with variation of the parameter λ for our algorithms on three real-world networks (averaged over 10 repeated experiments).

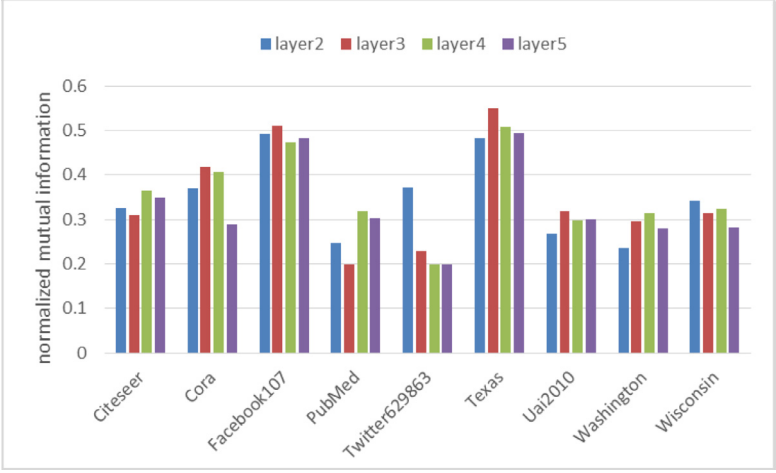


Fig. 4. Community detection performance in the deep autoencoder structure with 2, 3, 4 and 5 layers on the nine real-world networks.

Table 3
Neural network structures.

Datasets	Layer configuration
Texas	374–256
Wisconsin	530–256
Twitter629863	342–256–128
Facebook107	2090–1024–512
Cora	5416–4096–2,048
Uai2010	6726–4096–2048
Washington	460–256–128–64
Citeseer	6624–4096–2048–1024
PubMed	39458–8192–8192–4096

stacked autoencoders in the build-in deep framework, and the number of nodes in each hidden layer is set to half of that of its input or output layer. Besides, we set sigmoid function as activations of each layer in the deep autoencoders. For training the model, we adopted 10 random initializations in the deep autoencoders, and treated the low-dimensional representation in the hidden layer of a deep autoencoder as the output.

4.6. Experimental results

On the datasets, the results measured by four types of evaluation metrics are listed in Tables 4–9. The overlapping results in Tables 4 and 5 show the comparison of our method and others in terms of the overlapping communities, and comparison in terms of non-overlapping results are listed in Tables 6–9. In addition, the average performance of methods is also shown in these tables.

As we all know, neural networks with the deep structure have a powerful learning ability. In Tables 5 and 6, MAC method has better results in most cases than those obtained by AE-link or AE-content methods. This may because, the clustering results, obtained by the non-overlapping community detection methods (e.g., AE-link or AE-content), got bad accuracy when using the overlapping evaluation metrics. Our methods did not outperform all other

Table 4
Comparison of our method and some overlapping methods in terms of *F-scores* on the nine datasets. Here, **bold** means the best result, and **asterisk** corresponds to the second-best result.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Infomap	0.066	0.3058	0.3565	0.3148	0.1852	0.3761	0.2261	0.1583	0.1859	0.2416
	Linkcomm	0.0492	0.0389	0.1103	0.0689	0.1633	0.242	0.0483	0.135	0.1087	0.1072
	Clique	0.0454	0.1115	0.23	0.0857	0.1201	0.1925	0.0776	0.1432	0.1487	0.1283
	BigCLAM	0.0755	0.1429	0.2915	0.0464	0.1288	0.3382	0.1555	0.1334	0.1185	0.159
Content	MAC	**0.4248**	0.2385	0.2432	0.5893	**0.4997**	0.2577	0.2529	**0.5798**	**0.4402**	0.3918
Link + content	CESNA	0.176	0.406	0.3386	0.1947	0.2327	0.4082	0.2852	0.1882	0.1921	0.2691
	DCM	0.017	0.0231	0.1246	0.0023	0.1444	0.1874	0.0953	0.1612	0.107	0.0958
Autoencoder	AE-link	0.2866	0.3055	0.2634	0.036	0.2631	**0.4622**	0.2356	0.1699	0.1022	0.2361
	AE-content	0.3267	0.2864	0.023	0.5633	0.1533	0.3626	0.0346	0.1289	0.1521	0.2257
	Ours	0.3933*	**0.4526**	**0.4633**	**0.6621**	0.4399*	0.4055	**0.3655**	0.5533*	0.4225*	**0.462**

Table 5
The comparison of our method and some overlapping methods in terms of *Jaccard scores*.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Infomap	0.035	0.2227	0.271	0.2118	0.1063	0.2504	0.1489	0.0874	0.1075	0.1601
	Linkcomm	0.0744	0.0722	0.0797	0.1668	0.1654	0.1668	0.0474	0.1315	0.3288	0.137
	Clique	0.0242	0.0656	0.23	0.0476	0.0654	0.1116	0.042	0.0783	0.0826	0.083
	BigCLAM	0.0404	0.0797	0.197	0.0238	0.0713	0.2132	0.0896	0.0732	0.0645	0.0947
Content	MAC	**0.2768**	0.2066	0.1561	**0.4184**	**0.3547**	0.2355	0.1511	**0.4564**	0.2136	0.2744
Link + content	CESNA	0.1066	0.2754	0.2452	0.1082	0.1379	0.2751	0.1738	0.1041	0.1081	0.1705
	DCM	0.0086	0.0117	0.0684	0.0012	0.0838	0.1044	0.0552	0.0898	0.0572	0.0534
Autoencoder	AE-link	0.036	0.3254	0.2966	0.2566	0.0362	0.2239	0.1325	0.0985	0.0023	0.1564
	AE-content	0.2535	0.1189	0.0465	0.3956	0.3455	0.1569	0.1056	0.0685	**0.3522**	0.2048
	Ours	0.2489	**0.4122**	**0.3024**	0.3922	0.3388	**0.2725**	**0.2566**	0.3989*	0.31*	**0.3258**

Table 6
Comparison of our method and some non-overlapping methods in terms of *NMI*.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Louvain	0.3474	0.3976	**0.5582**	0.1766	0.2129	0.5464	0.2967	0.2069	0.1914	0.326
	CNM	0.3406	**0.4709**	0.3884	0.2184	0.0726	0.5305	0.214	0.0791	0.0927	0.2675
Content	AP	0.3169	0.3234	0.2624	0.138	0.284	0.5051	**0.3375***	0.3096	0.2731	0.3016
	DP	0.0197	0.0146	0.1834	0.0136	0.172	0.0514	0.1227	0.2632	0.2157	0.1174
	LDA	0.0257	0.0272	0.1357	0.0265	0.1066	0.3322	0.1104	0.1274	0.1873	0.1199
Link + content	PCL-DC	0.0295	0.195	0.4265	0.2684	0.066	0.5436	0.2692	0.0931	0.0407	0.2147
	Block-LDA	0.0058	0.0158	0.0563	0.007936	0.0353	0.1091	0.0387	0.1187	0.0459	0.0482
Autoencoder	AE-link	0.3043	0.3544	0.4587	**0.3421**	0.309	0.4467	0.2644	0.2185	0.2518	0.3278
	AE-content	0.222	0.2422	0.2743	0.2184	0.212	0.4045	0.2118	**0.3152**	0.222	0.258
	Ours	**0.3642**	0.4177*	0.5096*	0.3192*	**0.372**	**0.5491**	0.3197*	0.315*	**0.3422**	**0.3899**

Table 7
Comparison of our method and some non-overlapping methods in *AC*.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Louvain	0.0457	0.0716	0.5774	0.0746	0.1967	0.4938	0.2237	0.1244	0.1489	0.2174
	CNM	0.1907	0.4328	0.6004	0.4067	0.2568	0.6296	0.2449	0.2581	0.2176	0.3597
Content	AP	0.0059	0.0066	0.0418	0.002	0.0656	0.1358	0.0375	0.0737	0.0534	0.0469
	DP	0.1645	0.1935	0.0774	0.1889	0.1202	0.358	0.0506	0.0968	0.1718	0.158
	LDA	0.1118	0.1193	0.4289	0.1232	0.1858	0.2222	0.1704	0.2359	0.3282	0.214
Link + Content	PCL-DC	0.247	0.3778	0.4184	**0.6355**	0.3825	0.5679	0.2882	0.3917	0.3092	0.402
	Block-LDA	0.2024	0.2574	0.2664	0.3957	0.3989	0.2469	0.1474	0.447	0.3321	0.2994
Autoencoder	AE-link	0.0623	0.2755	0.5069	0.4533	0.4099	0.5213	0.1244	0.1355	0.3675	0.3174
	AE-content	0.1852	0.3266	0.3324	0.3133	0.0212	0.4655	0.2954	0.3655	0.1998	0.2783
	Ours	**0.3896**	**0.452**	**0.6013**	0.2299	**0.6543**	**0.6321**	**0.4699**	**0.4962**	**0.4852**	**0.4901**

baselines, such as MAC got better performance on some datasets. But actually, by comparing with MAC in terms of the average performance, our method achieved a gain of 18.73% in terms of *Jaccard* scores and a gain of 11.62% in terms of *F-scores*. Furthermore, our method beats AE-content, AE-link and other baselines using link or content alone. This observably verifies that our assertions (incorporating the topology and content) can indeed help to obtain an improvement for community detection.

In general, Tables 6–9 show that our method outperforms all baselines. On one hand, our method gets a better performance than that of AE-link and AE-content. It validates that the fusion of those two types of information can be very useful for community detection. On the other hand, with the comparison of the average performance of the methods, our method outperforms PDL-DC 10.99% and Block-LDA 28.34% on average. Because in Tables 5 and 6 there are the non-overlapping clustering results, they may get a better qualification comparison using non-overlapping evaluation

Table 8
Comparison of our method and some non-overlapping methods in *F-scores*.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Louvain	0.0453	0.0857	0.353	0.0634	0.1873	0.3704	0.1718	0.133	0.157	0.1741
	CNM	0.1411	0.3286	0.2216	0.2818	0.2083	0.4007	0.1741	0.2062	0.2027	0.2406
Content	AP	0.0089	0.0105	0.0811	0.0022	0.1859	0.2418	0.0529	0.1118	0.0896	0.0872
	DP	0.1251	0.1245	0.0895	0.1445	0.1355	0.1826	0.0549	0.1232	0.151	0.1256
	LDA	0.1347	0.1397	0.1525	0.1628	0.188	0.2294	0.1605	0.2074	0.2415	0.1796
Link + content	PCL-DC	0.2474	0.3891	0.2933	0.6305	**0.5071**	0.4019	0.2971	0.3366	0.3021	0.3786
	Block-LDA	0.1476	0.155	0.096	0.0147	0.1405	0.0638	0.0607	0.1964	0.1907	0.1184
Autoencoder	AE-link	0.2866	0.3055	0.2634	0.036	0.2631	**0.4622**	0.2356	0.1699	0.1022	0.2361
	AE-content	0.3267	0.2864	0.023	0.5633	0.1533	0.3626	0.0346	0.1289	0.1521	0.2257
	Ours	**0.3933**	**0.4526**	**0.4633**	**0.6621**	0.4399*	0.4055*	**0.3655**	**0.5533**	**0.4225**	**0.4732**

Table 9
Comparison of our method and some non-overlapping methods in *Jaccard*.

Algorithm\datasets		Citeseer	Cora	Facebook107	PubMed	Texas	Twitter629863	Uai2010	Washington	Wisconsin	Avg.
Link	Louvain	0.0239	0.0528	0.2648	0.0341	0.1121	0.2429	0.1085	0.0733	0.0894	0.1113
	CNM	0.0841	0.2455	0.1783	0.1912	0.1212	**0.2782**	0.1031	0.1198	0.1151	0.1596
Content	AP	0.0045	0.0053	0.0437	0.0011	0.1479	0.1743	0.0278	0.0617	0.0478	0.0571
	DP	0.0697	0.0697	0.0488	0.0845	0.0754	0.1077	0.0285	0.0668	0.0872	0.0709
	LDA	0.0731	0.0759	0.0885	0.0897	0.1059	0.1307	0.0897	0.1185	0.1504	0.1025
Link + content	PCL-DC	0.1437	0.2561	0.1877	**0.4613**	0.1926	0.2702	0.1917	0.212	0.1826	0.2331
	Block-LDA	0.0809	0.0868	0.0514	0.0074	0.0839	0.0332	0.0328	0.122	0.1134	0.068
Autoencoder	AE-link	0.036	0.3254	0.2966	0.2566	0.0362	0.2239	0.1325	0.0985	0.0023	0.1564
	AE-content	**0.2535**	0.1189	0.0465	0.3956	**0.3455**	0.1569	0.1056	0.0685	**0.3522**	0.2048
	Ours	0.2489*	**0.4122**	**0.3024**	0.3922	0.3388*	0.2725*	**0.2566**	**0.3989**	0.31*	**0.3355**

metrics. So this may further validate that, our method, by incorporating those two kinds of information in terms of the nonlinearity, obtains a powerful graph representation which really helps to aid to find communities.

5. Conclusion

In this paper, we proposed a new method that fuses the topological and content information for community detection using the deep learning framework. This study is inspired by the similarity between autoencoder and spectral methods in terms of a low-dimensional approximation of the spectral matrix. The proposed method provides a nice approach for finding a low-dimensional encoding of the community structure and achieving collective optimization of modularity and normalized-cut without a seam. This encoding supplies a nonlinear way to integrate the linkage and content information. As a result, the proposed method has better performance than the existing methods, also considers network structure and node contents. Experimental results demonstrate that the proposed new method provides better representation for community detection.

There are several questions to be further investigated in our method. For example, it would be very interesting to explore the other approaches of finding the manifold structure of the node content information space. In the future, we may also develop better strategies to apply the autoencoder to incorporate the linkage and content information, or explore some other ways of incorporating the topology and node contents from different but better viewpoints.

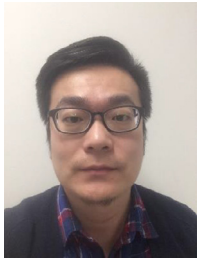
Acknowledgments

The work was supported by National Basic Research Program of China (2013CB329301), and Natural Science Foundation of China (61772361, 61503281, 61303110).

References

[1] M. Sachan, D. Contractor, T.A. Faruquie, L.V. Subramaniam, Using content and interactions for discovering communities in social networks, in: Proceedings of the 21st International Conference on World Wide Web, ACM, 2012, pp. 331–340.
[2] S. Ganguly, M. Gupta, V. Varma, V Pudi, Author2vec: learning author representations by combining content and link information, in: Proceedings of the 25th International Conference Companion on World Wide Web, 2016, pp. 49–50.
[3] D. He, Z. Feng, D. Jin, X. Wang, W. Zhang, Joint identification of network communities and semantics via integrative modeling of network topologies and node contents, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[4] J. Shi, J Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[5] A Ng, Sparse autoencoder, CS294A Lecture Notes 72 (2011) (2011) 1–19.
[6] L Deng, Deep learning: from speech recognition to language and multimodal processing, APSIPA Trans. Signal and Inf. Process. 5 (2016) 1–15.
[7] L. Shao, Z. Cai, L. Liu, K Lu, Performance evaluation of deep feature learning for RGB-D image/video classification, Inf. Sci. 385 (2017) 266–283.
[8] H. Bourlard, Y Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybern. 59 (4) (1988) 291–294.
[9] C. Eckart, G Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[10] M.E. Newman, Modularity and community structure in networks, Proc. Ntl. Acad. Sci. 103 (23) (2006) 8577–8582.
[11] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[12] R. Balasubramanyan, W.W. Cohen, Block-LDA: jointly modeling entity-annotated text and entity-entity links, in: Proceedings of the 2011 SIAM International Conference on Data Mining, 2011, pp. 450–461.
[13] I. Derényi, G. Palla, T Vicsek, Clique percolation in random networks, Phys. Rev. Lett. 94 (16) (2005) 160202.
[14] A.P. Streich, M. Frank, D. Basin, J.M. Buhmann, Multi-assignment clustering for Boolean data, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 969–976.
[15] L. Danon, A. Diaz-Guilera, J. Duch, A Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (2005) P09008.
[16] J. Mcauley, J Leskovec, Discovering social circles in ego networks, ACM Trans. Knowl. Discov. Data 8 (1) (2014) 4.
[17] Y. Zhang, E. Levina, J Zhu, Community detection in networks with node features, Electron. J. Stat. 10 (2) (2016) 3153–3178.
[18] A.Z. Broder, M. Charikar, A.M. Frieze, M Mitzenmacher, Min-wise independent permutations, J. Comput. Syst. Sci. 60 (3) (2000) 630–659.
[19] J. Yang, J. McAuley, J Leskovec, Community detection in networks with node attributes, in: Proceedings of the Data Mining (ICDM), IEEE, 2013, pp. 1151–1156.
[20] X. Wang, Di Jin, X. Cao, L. Yang, W. Zhang, Semantic community identification in large attribute networks, in: Proceedings of the AAAI'16, 2016.

[21] F. Tian, B. Gao, Q. Cui, et al., Learning deep representations for graph clustering, in: Proceedings of the International Conference on Artificial Intelligence, 2014, pp. 1293–1299.
[22] P. Sen, G. Namata, M. Bilgic, et al., Collective classification in network data, AI Mag. 29 (3) (2008) 93–106.
[23] J. Leskovec, J.J. Mcauley, Learning to discover social circles in ego networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 539–547.
[24] A. Clauset, M.E. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.
[25] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[26] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123.
[27] Y.Y. Ahn, J.P. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, Nature 466 (7307) (2010) 761–764.
[28] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 587–596.
[29] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[30] A. Rodriguez, A Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[31] L Griffiths T, M Steyvers, Finding scientific topics, Proc. Natl. Acad. Sci. 101 (2004) 5228–5235.
[32] S. Pool, F. Bonchi, M.V. Leeuwen, Description-driven community detection, ACM Trans. Intell. Syst. Technol. 5 (2) (2014) 28.
[33] T. Yang, R. Jin, Y. Chi, S Zhu, Combining link and content for community detection: a discriminative approach, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 927–936.
[34] L Yang, X Cao, D He, et al., Modularity based community detection with deep learning, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016, pp. 2252–2258.
[35] A. Ng, Sparse autoencoder, CS294A Lect. Notes 72 (2011) (2011) 1–19.
[36] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[37] X. Yan, C. Shalizi, J.E. Jensen, F. Krzakala, C. Moore, L. Zdeborová, Y. Zhu, Model selection for degree-corrected block models, J. Stat. Mech. Theory Exp. 2014 (5) (2014) P05007.
[38] K. Chen, J. Lei, Network cross-validation for determining the number of communities in network data, J. Am. Stat. Assoc. (2017) 1–11.
[39] A. Lancichinetti, M.I. Sirer, J.X. Wang, D. Acuna, K. Körding, L.A.N. Amaral, High-reproducibility and high-accuracy method for automated topic classification, Phys. Rev. X 5 (1) (2015) 011007.



Jinxin Cao received his B.S. degree from Shandong Normal University, China, in 2010. Since 2011, he has been a post-graduate and Ph.D. joint program student in school of Computer Science and Technology at Tianjin University, China. His research interests includes data mining and analysis of complex networks.

Di Jin received his B.S., M.S. and Ph.D. degree from College of Computer Science and Technology, Jilin University, China, in 2005, 2008 and 2012. Since 2012, he has been associate professor in Tianjin University. His current research interests include artificial intelligence, complex network analysis, and network community detection.

Liang Yang received his Ph.D. degree from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences in 2016. He has been an assistant professor in School of Information Engineering, Tianjin University of Commerce. His current research interests include community detection, machine learning and computer vision.

Jianwu Dang graduated from Tsinghua University, China, in 1982, and got his M.S. at the same university in 1984. He worked for Tianjin University as a lecture from 1984 to 1988. He was awarded the PhD from Shizuoka University, Japan in 1992. Since 2001, he has moved to Japan Advanced Institute of Science and Technology (JAIST). His research interests are in all the fields of speech production, speech synthesis, and speech cognition.