Using Deep Learning for Community Discovery in Social Networks

Di Jin[1], Meng Ge[1], Zhixuan Li[1], Wenhuan Lu[2*], Dongxiao He[1], and Francoise Fogelman-Soulie[2]
[1]School of Computer Science and Technology, Tianjin University, Tianjin, China
Email: jindi@tju.edu.cn, gemeng@tju.edu.cn, zhixuanli520@gmail.com, hedongxiao@tju.edu.cn,
[2]School of Computer Software, Tianjin University, Tianjin, China
Email: wenhuan@tju.edu.cn, soulie@tju.edu.cn

***Abstract*—Community detection is an important task in social network analysis. Existing methods typically use the topological information alone, and ignore the rich information available in the content data. Recently, some researchers have noticed that user profiles can also benefit to community detection, and hence the combination of topology and node contents has become a new hot topic. Some methods using both topology and content have been proposed. However, they often suffer from two drawbacks: 1) they cannot extract a potential deep representation of the network; 2) they cannot automatically weight different information sources with adequate balance parameters. To overcome these issues, we propose a deep integration representation (DIR) algorithm via deep joint reconstruction, which is motivated by the similarity between deep feedforward auto-encoders and spectral clustering in terms of matrix reconstruction. Thanks to spectral clustering which is one of the best community detection methods, the proposed new method is also good at community discovery task. In addition, DIR has further benefit because it not only provides a nonlinear and deep representation of the network, but also learns the most suitable balance between different components automatically. We compare the proposed new approach with nine state-of-the-art community detection methods on eight real relatively large networks. The experimental results show the definite superiority of this new approach.**

***Keywords*—community detection, deep learning, stacked auto-encoder, spectral clustering, social networks.**

I. Introduction

With the emergence of social media networks, social network analysis research has recently become more and more important. Graph is a popular data structure for describing social networks, in which nodes represent the users and edges reflect the relationships between users. Community detection is a significant research topic for characterizing and understanding complex social networks. It aims at finding a partition of the network into potential communities such that members of each community are mostly connected to members within that same community.

In the conventional scheme, most community detection algorithms [1]–[3] only use the topological information, and often ignore the rich information available in the content data. Recently, many researchers [4]–[10] have noticed that users profiles can be used to measure the similarity between different users, and that the more similar profiles are, the more likely the corresponding users should belong to the same group. Therefore, combining topology and node contents has become a new hot topic in the current community detection research.

Recently, many methods using both topology and content have been proposed. There are several different ways to measure the incorporation of different components in network: 1) *Linear incorporation*, which connects different components through additional parameters. Such methods follow the assumption that all components can be linearly combined. For instance, CODICIL [4] linearly combines network topology and similarities of node contents into a new network model, and then computes clusters by standard graph clustering algorithms. NMTF [11] is a unified clustering framework based on nonnegative matrix tri-factorization with three types of graph regularization, to integrate different social information, such as users profiles and social relations. SCI [12] integrates network topology and semantic information on nodes using community membership for community detection. 2) *Probabilistic incorporation* generates one information from the other. Such methods introduce the subjective preference, not fully achieving the purpose of effective integration of different information sources. For example, NEMBP [13] assumes the topological information as the basis to generate the text information. CESNA [5] is a probabilistic generative model, to obtain the relationship between node attributes and communities, where the sub-models using network topology and node attributes are connected by a sparse constraint. Moreover, a block coordinate ascent approach is adopted to update the model parameters. Yang [14] introduces an alternative discriminative probabilistic method to detect communities, different from generative models. In this model, the nodes' popularities are used to decide the link probability between two nodes and are then combined with content information in the condition of link model to estimate the community memberships. 3) *Heuristic incorporation*. For example, a heuristic framework aiming at finding communities and their corresponding descriptions by means of alternating between community generation and description induction was proposed in DCM [6]. An algorithm based on matrix decomposition was presented in [15], which constructs a latent representation of nodes and edges based on their mutual structural dependency, and incorporates edge content by two methods for community detection. A novel transformation framework is proposed in NEIWalk [16], based

*Corresponding Author: Wenhuan Lu, wenhuan@tju.edu.cn on the idea of transforming the content-based network into a new network where structure, node content and edge content are all embedded based on the relationship among them.

However, the above combinatorial methods still suffer from two drawbacks requiring more work to improve them: 1) they cannot extract a potential deep representation of the network. 2) they cannot automatically weight different information sources with adequate balance parameters. Therefore, we choose to put our focus on these two drawbacks of community detection.

To overcome the above-mentioned drawbacks of existing combinatorial methods, we notice that deep learning [17] is a powerful representation-learning framework, which has had great success in many applications such as speech recognition, image classification and natural language processing. It can easily explore deep non-linear relationships. Moreover, our proposal is motivated by the similarity between deep feed-forward auto-encoders [18] and spectral clustering [19] in terms of matrix reconstruction [3], [20]. Specifically, due to its solid theoretical foundation and global optimal solution, spectral clustering has come to the foreground in the past decades. For instance, the spectral method based on modularity considers topological information for community detection, and the content-based normalized-cut model for content clustering. It aims at finding a low-dimensional matrix to reconstruct the spectral matrix. Similarly, an autoencoder also encodes a low-rank embedding to reconstruct its original input.

Based on these above discussions, we propose a deep integration representation (DIR) algorithm via deep joint reconstruction. First, we use the modularity and the normalized-cut model to combine network topology and the node content because of their superior performance for topology/content in spectral clustering. Second, deep auto-encoders are adopted to combine different views of each node, to jointly generate a deep node representation for community detection. To the best of our knowledge, this may be the first work promoting deep auto-encoders for community detection to automatically weight the different information sources.

II. Preliminary

Given an undirected and attributed network $G = (\mathrm{A}, \mathrm{O})$, with n nodes, where $\mathrm{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is the adjacency matrix, whose elements $a_{ij} = 1$ if there is an edge joining nodes i and j, and 0 otherwise. Here $\xi_i = \sum_j a_{ij}$ is the degree of node i and $m = \frac{1}{2}\sum_i \xi_i$ is the total number of edges in the network. $\mathrm{O} = [o_{ij}] \in \mathbb{R}^{n \times n}$ is the similarity matrix, in which the similarity o_{ij} between nodes i and j is the cosine similarity between their corresponding content vectors. To avoid the problem caused by the manifold structure of node contents, an η-nearest neigbors method [21] was used to sparsify the similarity matrix O. We keep the η closest points for each node in O and set other similarity values to zero, and then obtain the sparse similarity matrix $\mathrm{S} = [s_{ij}] \in \mathbb{R}^{n \times n}$, in which the (weighted) degree of node i is defined as $d_i = \sum_j s_{ij}$. As said in [21], a suitable η can help to retain the local invariance of the original data, and hence can better respect the structure of the underlying manifold of node contents. According to the experiments and other literatures, we often set η at a small fixed integer, such as 6. Now our goal is to separate the n nodes into k communities, denoted as $C_1, C_2, ..., C_k$.

A. Modularity Model

Newman-Girvan's modularity function Q was introduced in [22]. It is by far the most used and best known quality function for community detection. Thus, optimization of modularity Q has become one of the major community detection methods. This function can be defined as follows, in the case of bisections (two communities only):

$$Q = \frac{1}{4m}\sum_{ij}(a_{ij} - \frac{\xi_i \xi_j}{2m})(\psi_i \psi_j) \tag{1}$$

where ψ_i equals 1 (or -1) if node i belongs to the first community (or second one).

Modularity can conveniently be optimized using eigenvectors and eigenvalues by defining the modularity matrix $\mathrm{B} = [b_{ij}] \in \mathbb{R}^{n \times n}$, with elements $b_{ij} = a_{ij} - \frac{\xi_i \xi_j}{2m}$. Thus, modularity Q can be rewritten as

$$Q = \frac{1}{4m}\psi^T \mathrm{B} \psi \tag{2}$$

where $\psi = [\psi_i] \in \{-1, 1\}^n$ is the nodes' community membership indicator. However, modularity maximization is a NP-hard problem. By relaxing the problem and allowing variables ψ_i to take any real value, this problem can be easily translated as:

$$\max Q = \max_{\Psi \in \mathbb{R}^{n \times k}} \mathrm{Tr}(\Psi^T \mathrm{B} \Psi) \tag{3}$$

where $\Psi = [\psi_{ij}] \in \mathbb{R}^{n \times k}$ is the community membership indicator matrix and $\mathrm{Tr}(\cdot)$ is the trace function. It can be shown that the solution to this problem is to get the k largest eigenvectors of modularity matrix B. Besides, solution space Ψ allows reconstructing the network topology in terms of community structure, and hence each row of matrix Ψ can be regarded as a good representation of the corresponding node in latent space for community detection.

B. Normalized-cut Model

Normalized-cut [23] is one of the most commonly used methods for content-based graph clustering. It computes the cut cost as a ratio between the total edge connections in one cluster to nodes outside and the total inner connections, i.e. the ratio of outer connection to inner connections. The objective function can be written as:

$$NCut(C_1, C_2, ..., C_k) = \sum_{t=1}^{k} \frac{link(C_t, \overline{C_t})}{vol(C_t)} \tag{4}$$

where $link(C_t, \overline{C_t}) = \frac{1}{2}\sum_{i \in C_t, j \in \overline{C_t}} s_{ij}$ is the total connections (content similarities) from nodes in C_t to all the nodes in $\overline{C_t}$ (not in C_t), $vol(C_t) = \sum_{i \in C_t} d_i$ is the total inner connections in C_t.

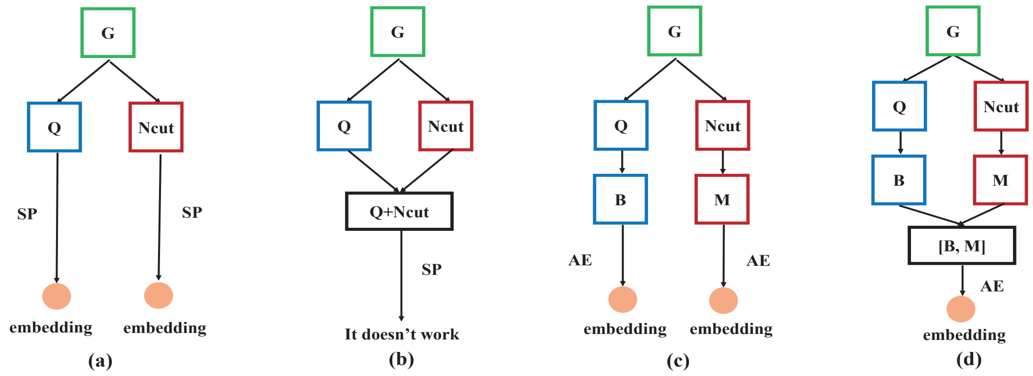

Fig. 1. (a) Q and $Ncut$ are solved by spectral clustering (SP) to obtain corresponding embeddings. (b) An ideal idea of linear combination of Q and $Ncut$ by spectral clustering, but it does not work. (c) We use B and M as the input feature matrix for the auto-encoder (AE) to separately optimize Q and $Ncut$. The process is similar to (a). (d) Q and $Ncut$ can be easily combined through deep learning to jointly reconstruct B and M using auto-encoder (AE).

To achieve the goal of minimizing the objective function, it is convenient to convert to the following optimization problem if node i is denoted as v_i:

$$\begin{aligned} \min_{\Phi \in \mathbb{R}^{n\times k}} \quad & \mathrm{Tr}(\Phi^T \mathrm{L} \Phi) \\ s.t. \quad & \mathrm{L} = \mathrm{D} - \mathrm{S}, \\ & \mathrm{D} = diag(d_1, d_2, ..., d_n), \\ & \phi_{ij} = \begin{cases} 1/\sqrt{vol(C_j)}, & \text{if } v_i \in C_j; \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Here, L is the Laplacian matrix of the similarity graph and its normalized form is $\mathrm{D}^{-1}\mathrm{L} = \mathrm{D}^{-1}(\mathrm{D} - \mathrm{S}) = \mathrm{I} - \mathrm{D}^{-1}\mathrm{S}$ where the identity matrix is denoted as I. Then $\mathrm{M} = \mathrm{D}^{-1}\mathrm{S}$ is called the Markov matrix. For this problem, the solution matrix Φ consists in the eigenvectors corresponding to the k smallest non-zero eigenvalues of the normalized Laplacian matrix $\mathrm{D}^{-1}\mathrm{L}$. In other words, the k largest eigenvalues of $\mathrm{D}^{-1}\mathrm{S}$ exactly span the solution representation in latent space. More important is that the solution matrix Φ is an extremely good representation to obtain (text or image) content clustering.

III. Model Description

In this section, we will introduce our deep integration representation (or DIR) model for community detection, including our motivation, the basic model, the deep structured model, as well as the time complexity.

A. Motivation

As mentioned in the preliminary section, the modularity model (Q) is exceptional for topology-based clustering, designed for representing network structure. On the other hand, the normalized-cut model ($Ncut$) is very effective for clustering content-based graphs, and best representing and reconstructing node contents. Each of these can be solved by spectral clustering (SP), which aims at finding a low-rank embedding to reconstruct the spectral matrix as shown in Fig. 1 (a). Therefore, as illustrated in Fig. 1 (b), an ideal solution may be to integrate these two models and combine their advantages. But it is difficult to optimize this new problem using spectral method. This is because the spectral method requires specific objectives, such as modularity function or normalized-cut separately. But when combining these two objectives, spectral clustering does not work due to its lack of flexibility. Besides, spectral clustering has a high time complexity, and it is very hard to incorporate additional regularization constraints which may also be needed to meet the requirements of real nonlinear networked data. Thus, in this paper we mainly focus on this difficulty and try to solve it.

However, we noted that Q and $Ncut$ can be regarded as aiming at reconstructing modularity matrix B and Markov matrix M respectively, using a low-rank approximation as shown in Theorem 1 below [20]. On the other hand, in theory, the non-linear auto-encoder (AE) [18] also seeks for a low-rank representation to reconstruct the network, and is, in practice, more flexible and efficient than spectral clustering. If we use B or M as the input feature matrix for the auto-encoder as shown in Fig. 1 (c), the process actually also best reconstructs the original input matrix, similarly to spectral clustering. Meanwhile, as illustrated in Fig. 1 (d), deep feed-forward auto-encoders are more flexible, and can easily jointly optimize Q and $Ncut$ through deep joint reconstruction of B and M as the input feature matrix. Moreover, deep feed-forward auto-encoders can further stack multiple layers to achieve additional representation accuracy from deep structures.

In addition, conventional models often integrate network topology and node contents through a linear combination strategy, which requires to tune the combination coefficient, i.e. the balance between topology and contents. In contrast, a non-linear auto-encoder can automatically learn the balance to seamlessly combine different data sources in a non-linear fashion, hence avoiding one of the disadvantages of linear combination methods. Thus, the above discussions motivate us to adopt deep learning auto-encoders to combine network topology and node contents for better community detection.

Theorem 1: (Eckart-Young-Mirsky). For a rank-r matrix $D \in \mathbb{R}^{m\times n} (m \geq n)$, there exists a singular value decomposition $D = U\Sigma V^T$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$ are the eigenvalues of D, $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_m) \in \mathbb{R}^{m\times n}$,

$U \in \mathbb{R}^{m\times m}$, and $V \in \mathbb{R}^{n\times n}$ are defined as follows:

$$U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$$

where U_1 is $m \times r$, Σ_1 is $r \times r$, and V_1 is $r \times n$. Then the low-rank approximation problem can be regarded as a process of minimizing the cost function between the given matrix D and an approximating matrix $\tilde{D}$, with Frobenius cost function as follows:

$$\underset{\tilde{D}:rank(\tilde{D})=k, k\leq r}{\arg\min} \|D - \tilde{D}\|_F = \sqrt{\sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_m^2}.$$

Thus, the reconstructed matrix results from the singular vectors corresponding to the k largest singular values and then can be written as:

$$\tilde{D}^* = U_1 \Sigma_1 V_1^T$$

B. Reconstruction based on Auto-Encoder

The auto-encoder is the key building block of our DIR model. The original input data representation is the same as the output representation in the auto-encoder. To be specific, we first construct modularity matrix B based on the network topology A. Next, the nearest neighbors algorithm is used to select, for each node, its η closest nodes in O, thus resulting into a matrix S sparser than the original similarity matrix O. We then construct the Markov matrix M using S and the mixed spectral matrix $\mathrm{X} = [\mathrm{B}, \mathrm{M}]$ which is fed to the auto-encoder: the collection of rows in X is the set of examples successively presented as inputs to the network. Later in the experiments Section IV, we will explain why the mixed spectral matrix X does not need a specified balance between B and M.

The auto-encoder always consists of two stages, the encoder and the decoder. The encoder maps a row in the mixed spectral matrix X to a low-dimensional (hidden layer) representation h and h can be regarded as a new representation of the corresponding row in latent space. Hence, in matrix form, X is mapped into a representation H with rows h, computed as:

$$\mathrm{H} = f(\mathrm{X}) = \Gamma(\mathrm{W}^{(\mathrm{H})}\mathrm{X} + \mathrm{b}^{(\mathrm{H})}) \quad (6)$$

where $\mathrm{W}^{(\mathrm{H})}, \mathrm{b}^{(\mathrm{H})}$ are the parameters of the encoder, $\Gamma(\cdot)$ is an encoder non-linear function such as $tanh(x) = 1/(1 + exp(-x))$ or $ReLU(x) = max(0, x)$ [17]. The decoder layer maps the latent representation H back into the original data space X, and the process can be represented as follows:

$$\mathrm{Y} = g(\mathrm{H}) = \Gamma(\mathrm{W}^{(\mathrm{Y})}\mathrm{H} + \mathrm{b}^{(\mathrm{Y})}) \quad (7)$$

where $\mathrm{W}^{(\mathrm{Y})}, \mathrm{b}^{(\mathrm{Y})}$ are the parameters to be learned in the decoder, $\Gamma(\cdot)$ is a decoder mapping function, which can be the same as the encoder function. The auto-encoder aims at learning a latent representation of the input data by minimizing the reconstruction loss between the original data and the data reconstructed from the representation. Then the optimization goal is formally represented as follows:

$$\hat{\theta} = \arg\min_{\theta} L_\theta(\mathrm{X}, \mathrm{Y}) = \arg\min_{\theta} \sum_{i=1}^{n} \|x_i - y_i\|_2. \quad (8)$$

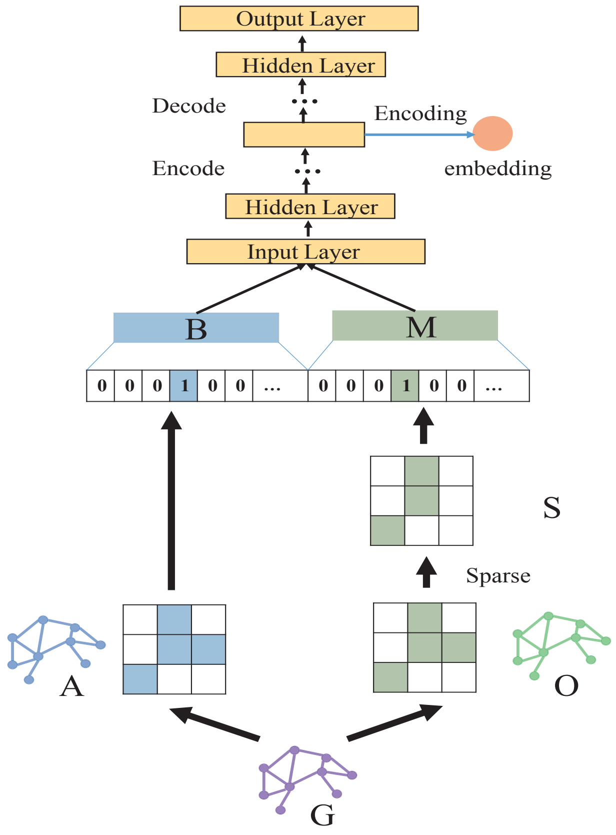

Fig. 2. The proposed deep integration representation (DIR) model which combines network topology and node contents to obtain the deep representations (embedding) for community detection. Given an attribute network $G = (\mathrm{A}, \mathrm{O})$, we construct a sparse similarity graph S from the original similarity graph O. Next, the modularity matrix B and the Markov matrix M can be constructed from A and S respectively, and concatenated into a mixed spetral matrix $\mathrm{X} = [\mathrm{B}, \mathrm{M}]$. Finally, we feed each row of the mixed spetral matrix into deep auto-encoders to obtain final deep reppresentations (embedding) for community detection.

where $\theta = \left\{\mathrm{W}^{(\mathrm{H})}, \mathrm{b}^{(\mathrm{H})}, \mathrm{W}^{(\mathrm{Y})}, \mathrm{b}^{(\mathrm{Y})}\right\}$ is the parameters vector of the auto-encoder and L_θ is a loss function which measures the reconstruction error. Here we adopt the Euclidean distance function as loss function and we do not use classical regularizers, such as L^1 regularization to enforce sparsity, and L^2 regularization to get robustness.

After training the auto-encoder, the parameters have been learned and can be used to generate the latent representations of the network with content. To date we have combined the best of modularity model and normalized-cut model, achieving to combine topology and node contents in a seamless way. After that, many traditional clustering methods can be used on the latent representations to find the communities. Here we use the k-means algorithm as many previous works did.

C. Optimization

Our goal is to minimize the objective function $J(\theta) = \sum_{i=1}^{n} \|x_i - y_i\|_2$. To solve this problem, we initialize each parameter randomly, and then apply the back-propagation algorithm with stochastic gradient descent [24]. Each iteration of gradient descent updates the parameters $\theta = \{\mathrm{W}^{(\mathrm{H})}, \mathrm{b}^{(\mathrm{H})}, \mathrm{W}^{(\mathrm{Y})}, \mathrm{b}^{(\mathrm{Y})}\}$ after presentation of one example (a row in the matrix) as follows:

$$\begin{aligned} W_{ji}^{*} &= W_{ji}^{*} - \alpha \frac{\partial}{\partial W_{ji}^{*}} J(\theta), \\ b_{j}^{*} &= b_{j}^{*} - \alpha \frac{\partial}{\partial b_{j}^{*}} J(\theta), \end{aligned} \tag{9}$$

where α is the learning rate, $*$ is a wildcard (here $* = \{(\mathrm{H}, \mathrm{Y})\}$). By defining $\mathrm{z}^{*} = \mathrm{W}^{*}\mathrm{x} + \mathrm{b}^{*}$, we have

$$\begin{aligned} \frac{\partial}{\partial W_{ji}^{*}} J(\theta) &= \sum_{i=1}^{n} \frac{\partial J(\theta)}{\partial \mathrm{z}_{j}^{*}} \cdot \frac{\partial \mathrm{z}_{j}^{*}}{W_{ji}^{*}} = \sum_{i=1}^{n} \delta_{j}^{*} \mathrm{x}_{i}^{T}, \\ \frac{\partial}{\partial b_{j}^{*}} J(\theta) &= \sum_{i=1}^{n} \frac{\partial J(\theta)}{\partial \mathrm{z}_{j}^{*}} \cdot \frac{\partial \mathrm{z}_{j}^{*}}{b_{j}^{*}} = \sum_{i=1}^{n} \delta_{j}^{*}, \end{aligned} \tag{10}$$

where $\delta_{j}^{*} = \partial J(\theta)/\partial \mathrm{z}_{j}^{*}$ is an error term that measures the reconstruction error between the activation and the true target value. The error terms are defined as follows:

$$\begin{aligned} \delta_{j}^{(\mathrm{Y})} &= -\sum_{i=1}^{n} (y_{ij} - h_{ij}) \cdot s'(\mathrm{z}_{j}^{(\mathrm{Y})}), \\ \delta_{j}^{(\mathrm{H})} &= (\sum_{i=1}^{n} W_{ji}^{(\mathrm{H})} \delta_{i}^{(\mathrm{Y})}) \cdot s'(\mathrm{z}_{j}^{(\mathrm{H})}), \end{aligned} \tag{11}$$

D. Stacked Auto-Encoders

Deep learning [17] is a layer-wise training architecture, meaning the original data can be presented through multiple layers, a large number of layers allows for better performance. To take advantage of deep architectures, we stacked a series of auto-encoders to construct the DIR model (Fig. 2). Recall that many community detection algorithms only involve one layer of representation. This may not be enough for the analysis of complex social networks in the real world. We aim to overcome this disadvantage by stacked auto-encoders. We trained the first auto-encoder to reconstruct the mixed spectral matrix $\mathrm{X} = [\mathrm{B}, \mathrm{M}]$, and obtained the latent representation $\mathrm{H}^{(1)}$ as well as the reconstructed data $\mathrm{Y}^{(1)}$. After that, we train the model layer by layer, training the i^{th} auto-encoder to reconstruct the hidden layer representation of the $(i-1)^{th}$ auto-encoder, and then obtaining the new latent representation $\mathrm{H}^{(i)}$. That is, the objective function of the i^{th} layer is

$$\hat{\theta}^{(i)} = \arg\min_{\theta^{(i)}} L_{\theta^{(i)}}(\mathrm{H}^{(i-1)}, \mathrm{Y}^{(i)}). \tag{12}$$

After this initialization stage, all the layers are stacked together and a few training passes are executed on this stacked architecture to fine-tune parameters. We summarize our layer-wise deep representation learning procedure in Algorithm 1. Based on the deep structure, the topology and node contents can thus be combined seamlessly, and their balance can be determined automatically. As a result, we obtain a deep and non-linear representation which can be further used for better community detection.

Algorithm 1: DIR Algorithm

```
input : adjacency matrix A, original similarity matrix O,
        number of layers $\mathcal{L}$, number of communities $k$, number of nearest neighbors $\eta$.
output: Final community structure $C_1, \cdots, C_k$.
1 Compute modularity matrix B based on network
  topology via $b_{ij} = a_{ij} - \xi_i \xi_j / (2m)$ (see Sec II).
2 Compute sparse similarity matrix S by $\eta$ nearest
  neighbors algorithm. Compute degree matrix D via
  $d_i = \sum_j s_{ij}$, and compute Markov matrix M using
  node contents via $\mathrm{M} = \mathrm{D}^{-1}\mathrm{S}$ (see Sec II).
3 Set mixed spectral matrix $\mathrm{X} = [\mathrm{B}, \mathrm{M}]$.
4 Set $\mathrm{X}^{(1)} = \mathrm{X}$
  for $l = 1$ to $\mathcal{L}$ do
5     Build single hidden layer auto-encoder with
      input data the rows in $\mathrm{X}^{(l)}$.
6     Apply Eq.(9) - (11) to optimize parameters,
      and obtain the latent representation $\mathrm{H}^{(l)}$.
7     Set $\mathrm{X}^{(l+1)} = \mathrm{H}^{(l)}$.
  end
8 Fine tune.
9 Run $k$-means on the rows of $\mathrm{H}^{(\mathcal{L})}$ to find the
  communities.
```

E. Time Complexity

To simplify the analysis, we denote by $\mathcal{L}$ the number of layers, n the number of nodes in the graph, m the number of edges. Thus $2m/n$ is the average degree of the graph. Let γ be the maximum number of hidden layer nodes in the stacked auto-encoders, t_p and t_q be the number of iterations for each auto-encoder in the stack and number of parameters random draws. It is not difficult to see that the complexity of the DIR model is $O(t_p t_q \gamma \mathcal{L}(m + nk))$ where η is the number of top-η nearest neighbors in the similarity network. Usually the values of parameter m, $\mathcal{L}$, t_p, t_q are bounded. Besides, parameter γ can be regarded as a fixed value, which is related to a predefined representation dimension, but not related to n. Therefore, the overall complexity is nearly linear in the number of nodes and links of the network.

IV. Experiments

We report the experimental results in this section. First, we introduce the experimental setup, including the data sets and the benchmark algorithms, and then analyze the experiments settings. After that, we present the experimental results to compare our proposed method with nine state-of-the-art benchmark approaches.

A. Datasets

We used seven publicly available datasets[1] with varying sizes and characteristics. Their domains cover social networks and document networks. The details of each dataset are as follows:

- **WebKB** includes 4 sub-datasets. It is a web page dataset gathered from four different universities, namely Cornell,

[1]http://linqs.umiacs.umd.edu/projects//projects/lbc/index.html

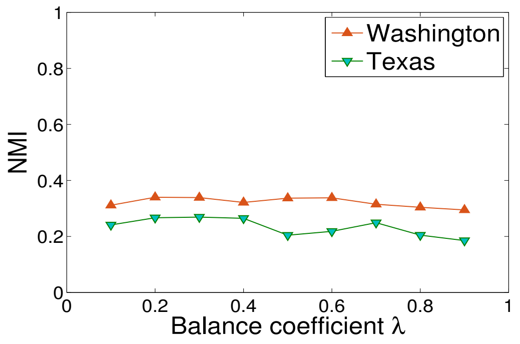

Fig. 3. The NMI (normalized mutual information) performance curves w.r.t balance coefficient λ on Washington and Texas datasets

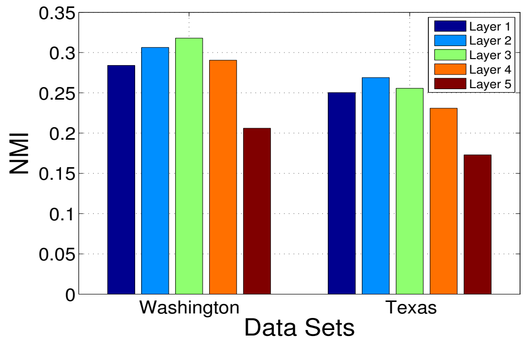

Fig. 4. The number of layers vs. the NMI on Washington and Texas datasets

Texas, Washington and Wisconsin, corresponding to 4 sub-datasets. Every university is portioned into 5 groups.

- **Citeseer** is a citation network, consisting of 3,312 scientific publications with 4,732 citation relationships classified into 6 sub-fields.
- **Cora** is a collection of 2,708 papers with 5,429 citation links. These papers are classified into seven classes.
- **UAI2010** is the articles information dumped from Wikipedia pages (October 2009) classified into 19 categories. The data contains 3,067 articles and 45,006 links.
- **Pubmed** consists of 19,717 scientific publications with three classes from Pubmed database and the citation network consists of 44,338 links.

B. Parameter Analysis

In this subsection, we first introduce the input data format. In our experiments, we constructed content-based edges by calculating the cosine similarity of nodes' content vectors and selected the top-η edges using the η nearest neighbors algorithm. To ensure that topology and content information of each node are in the same space, we normalized matrices B and M rows with $z\text{-norm}(\cdot)$ [4], which centers and normalizes the values to zero mean and unit variance:

$$z\text{-norm}(x) = \frac{x_i - \mu}{\sigma}, \mu = \frac{\sum_{i=1}^{|x|} x_i}{|x|}, \sigma^2 = \frac{\sum_{i=1}^{|x|}(x_i - \mu)^2}{|x|-1}.$$

First, we wanted to study the effect of a mixture balance parameter between network topology and node contents. We thus fed a combined matrix $\mathrm{X} = [\lambda \mathrm{B}, (1-\lambda)\mathrm{M}]$ which horizontally stacks modularity matrix B and Markov matrix M with a balance coefficient λ into the auto-encoder, row by row, and trained the auto-encoder to minimize the reconstruction error. For each network, we trained it with 10 random initializations of the networks' parameters and obtained the latent representations for each row. Then, we used k-means algorithm for clustering the representations obtained and produced our final result as the average performance of our 10 experimental results.

Here we show this analysis on two datasets in the WebKB collection (Washington and Texas), since the results on other datasets are also similar. We can then easily observe the changes in the balance of the two information sources in the experiment. The results are shown in Fig. 3, where performance is measured by NMI (normalized mutual information [25], described in next section). We found that there is very little influence on the final performance when the balance weight changed. For example, we calculated that the overall variances are 0.00026 on Washington and 0.00087 on Texas, where the variances are all less than 5%; the error between maximum and minimum is less than 5%. That is, the balance coefficient can be set at any value from 0.1 to 0.9, such as $\lambda = 0.5$ in all of the experiments as we did. The input data rows can thus be merged at 1:1 mixing ratio, thus stacking modularity matrix B and Markov matrix M horizontally, which we denoted as $\mathrm{X} = [\mathrm{B}, \mathrm{M}]$.

TABLE I
DEEP LEARNING NETWORK SETTING

Datasets	Nodes	Links	Attributes	Layer configuration
Texas	183	328	1,498	366-256-128-64-32-16
Cornell	195	304	1,588	390-256-128-64-32-16
Washington	217	446	1,578	434-256-128-64-32-16
Wisconsin	262	530	1,623	524-256-128-64-32-16
Citeseer	2,559	4,732	3,698	5,118-1,024-512-256-128-64
Cora	2,708	5,429	1,432	5,416-1,024-512-256-128-64
UAI2010	3,067	45,006	4,973	6,134-1,024-512-256-128-64
Pubmed	19,717	44,338	500	39,434-1,024-512-256-128-64

In our experiments, the number of nodes in each hidden layer tested is shown in Table I. The neural networks for all datasets are stacked auto-encoders with 4+1 hidden layers for encoding and 4 hidden layers for decoding, and the dimension of each hidden layer is set to a power of two less than input and output spaces. In practice, all auto-encoders were trained separately and the learned representation of every layer could thus be obtained. The final layers configuration of the network is decided by the result of each layer. For example, the layer configurations for Washington and Texas are 434-256-128-64-32-16 and 366-256-128-64-32-16 respectively, and the final experiment results in Fig. 4 show that the result of the third layer on Washington is best, and for Texas the third layer is second-best. So we set the final layer configuration as the third layer because a good result often appears in (or near) the third layer representation in all the datasets in our experiments.

In practice, we use Theano[2] deep learning tools to construct stacked auto-encoders, and set the learning rate as 0.1. Specifically, we use the $tanh(\cdot)$ as the activation function, set the training batch to the size of the network (i.e. the number of rows n), and ran at most 50,000 iterations. For each network, we train with 10 random initializations of the parameters. Finally, we obtain the latent representations of the n rows (i.e. nodes) for clustering. Here, we adopted the k-means algorithm for community detection and returned the performance results as the mean of the 10 experimental performances. The source code of the DIR is available online[3].

C. Evaluation Metrics

To assess the performance of community detection algorithms, we use the standard community comparison metrics: normalized mutual information (NMI) [25], accuracy (AC) [25], F-score [5], Jaccard [5] and generalized normalized mutual information (GNMI) [26].

Given the detected communities C and the ground-truth communities C^*, accuracy AC is defined as:

$$AC\left(C,C^*\right)=\frac{\sum_{i=1}^{n}\delta\left(map\left(C_i^*\right),map\left(C_i\right)\right)}{n}$$

where $\delta\left(a,b\right)$ is the delta function that equals 1 if $a=b$ and 0 otherwise, and $map(C_i^*)$, resp. $map(C_i)$, is the function that maps each community C_i^*, resp. C_i to the index of the community i belongs to in C^*, resp. C. The normalized mutual information (NMI) is defined as:

$$NMI\left(C,C^*\right)=\frac{\widehat{MI}\left(C,C^*\right)}{max\left(H\left(C\right),H\left(C^*\right)\right)}$$

where $H\left(C\right)=\sum\limits_{C_i}\frac{|C_i|}{|C|}log(\frac{|C_i|}{|C|})$ is the entropy of the set of communities C, and

$$\widehat{MI}\left(C,C^*\right)=\sum_{C_i,C_j}p\left(C_i,C_j^*\right)log\frac{p\left(C_i,C_j^*\right)}{p\left(C_i\right)p\left(C_j\right)}$$

is the mutual information between C and C^*. The F-score is defined as:

$$F\left(C,C^*\right)=\sum_{C_i\in C}\frac{|C_i|}{\sum_{C_j\in C}|C_j|}\max_{C_j^*\in C^*}F\left(C_i,C_j^*\right)$$

where $F\left(C_i,C_j^*\right)$ evaluates the F-score between C_i and C_j^*. The Jaccard metric is defined as:

$$JS\left(C,C^*\right)=\sum_{C_j^*\in C^*}\frac{\max\limits_{C_i\in C}JS\left(C_i,C_j^*\right)}{2\left|C^*\right|}+\sum_{C_i\in C}\frac{\max\limits_{C_j^*\in C^*}JS\left(C_i,C_j^*\right)}{2\left|C\right|}$$

[2]http://deeplearning.net/software/theano/
[3]http://git.oschina.net/dmai/dir

where $JS\left(C_i,C_j^*\right)$ is the Jaccard similarity between C_i and C_j^*. The generalized normalized mutual information (GNMI) is defined as:

$$GNMI\left(C,C^*\right)=1-\frac{1}{2}\left[H\left(C|C^*\right)_{norm}+H\left(C^*|C\right)_{norm}\right]$$

where $H\left(C|C^*\right)_{norm}$ is the normalized version of conditional entropy between C and C^*. For all metrics, largest is best.

D. Experimental Evaluation

To assess the effectiveness of our method DIR, we compared it with nine state-of-the-art methods for community detection, which are divided into three groups based on the data sources they used, for non-overlapping (Table II) and overlapping (Table III) community detection. Specifically, we compare DIR against two topology-based methods: SBM [1] and BigCLAM [2], two node contents-based methods: CAN [9] and SMR [10] and five methods that combine both topology and node contents: PCL-DC [14], Block-LDA [27], SCI [12], CESNA [5] and DCM [6].

TABLE II
PERFORMANCE (%) OF ALGORITHMS WITH DISJOINT COMMUNITIES. BOLD REPRESENTS BEST RESULT. N/A MEANS OUT OF MEMORY

Metrics(%)	**Methods**	**Cornell**	**Texas**	**Washington**	**Wisconsin**	**Citeseer**	**Cora**	**UAI2010**	**Pubmed**
NMI	SBM	9.69	16.65	9.87	3.14	4.13	17.07	31.22	12.28
	CAN	6.14	8.15	12.82	7.00	1.21	1.32	6.12	5.67e-4
	SMR	8.45	3.55	0.73	7.21	0.71	1.18	1.70	3.57e-4
	PCL-DC	7.23	10.37	5.66	5.01	2.99	17.54	26.92	**26.84**
	BLOCK-LDA	6.81	4.21	3.69	10.09	2.42	1.41	1.41	6.58
	SCI	11.44	17.84	12.37	17.04	4.88	19.26	24.79	N/A
	DIR	**27.50**	**26.89**	**31.79**	**29.65**	**27.42**	**40.30**	**31.86**	22.44(2)
AC	SBM	37.95	48.09	31.80	32.82	26.57	38.48	26.02	53.64
	CAN	41.54	49.73	51.61	46.56	23.21	30.21	17.87	39.96
	SMR	31.79	47.54	49.77	40.84	22.55	30.28	15.68	39.95
	PCL-DC	30.26	38.80	29.95	30.15	24.85	34.08	28.82	**63.55**
	BLOCK-LDA	**46.15**	54.10	39.17	49.62	24.35	25.52	16.04	16.04
	SCI	45.64	62.30	51.15	**50.38**	27.98	40.62	30.94	N/A
	DIR	45.79(2)	**64.03**	**60.51**	46.79(3)	**41.85**	**60.71**	**31.07**	59.60(2)

TABLE III
PERFORMANCE (%) OF ALGORITHMS WITH OVERLAPPING COMMUNITIES. BOLD REPRESENTS BEST RESULT.

Metrics(%)	**Methods**	**Cornell**	**Texas**	**Washington**	**Wisconsin**	**Citeseer**	**Cora**	**UAI2010**	**Pubmed**
F-score	BIGCLAM	13.23	20.64	13.35	12.84	9.30	18.89	16.99	7.72
	CESNA	23.48	23.54	21.91	23.17	3.38	**31.05**	32.32	27.97
	DCM	14.38	11.15	12.45	10.45	2.50	3.43	9.65	0.38
	DIR	**46.22**	**44.42**	**43.75**	**44.90**	**43.20**	28.37(2)	**33.76**	**61.18**
Jaccard	BIGCLAM	7.18	12.18	7.25	7.01	5.01	10.89	9.87	4.04
	CESNA	13.47	13.57	12.40	13.14	1.73	1.91	21.26	16.26
	DCM	7.95	6.03	6.72	5.54	1.27	1.76	5.77	0.19
	DIR	**31.23**	**31.74**	**31.66**	**30.66**	**28.37**	**43.94**	**22.36**	**44.82**
GNMI	BIGCLAM	0.59	0.75	0.77	0.44	0	0	11.94	0.57
	CESNA	2.22e-14	0.67	0.32	2.22e-14	2.22e-14	2.64	7.60	0
	DCM	0	1.17	0.17	0.51	0	1.11e-14	2.80	2.00e-14
	DIR	**11.50**	**13.74**	**13.94**	**13.66**	**11.16**	**29.36**	**10.43**	**16.82**

We compare our method DIR to the algorithms with disjoint or overlapping communities, respectively, shown in Table II and Table III. The metrics AC and NMI are for disjoint communities, and F-score, Jaccard and GNMI are for overlapping communities. For all the metrics, high value stands for better performance and the bold digits represent the best performance. For each result, we repeat 10 times and report the average result.

As shown in Table II, DIR outperforms other methods on all networks, except Pubmed, for NMI, and on all networks for AC, except Cornell, Wisconsin and Pubmed, where it is ranked 2 or 3. As shown in Table III, the performance of our method neatly achieves the best performance on all networks and metrics, except on Cora for F-score. In total, when measured in terms of metrics Jaccard and GNMI, our algorithm always achieved the best performance. Specifically, for dataset Citeseer for example, we increase the performance by around 22% in NMI, 13% in AC, 34% in F-score, 23% in Jaccard and 10% in GNMI. On average, we improve performances by more than 20%.

There are three possible reasons for these improvements. One is that both document networks and social networks have multi-level modalities, i.e., document, topic, vocabulary, words, etc. Besides, real-world networks have various nonlinear features, i.e., a relationship among users is not necessarily linear. The existing community detection methods typically involve one layer of representation, but our method can learn deep multi-levels representations for community detection. Another is that our method combines network topology and node contents without parameter-tuning, automatically learning the balance parameters to integrate the two different data sources for improved performance. More importantly, our method combines the advantages of modularity model and normalized-cut model, which are promising for community detection and content graph clustering respectively, and a nonlinear representation so that performance can improve a lot. However, for some networks, the performance improvement of our algorithm is less than for others: this is the case, for example, for Texas and Citeseer. A possible explanation for this may be that small networks have relatively less information; although deep structures can improve performance, the original information will be reduced if the structure is too deep. On the contrary, large-scale networks may have deeper representations. Then the deep integration representations will be better, and the performance will be improved more.

V. Conclusion

We proposed a deep nonlinear approach to combine topological information and node content information of networks, and effectively solve the seamless integration of two community detection models or objectives, i.e., network modularity and normalized-cut. In addition, our method can automatically learn the balance mixture weight between the different views of the original networked data, and obtain deep representations which allow improved community detection performances. Experimental results on real-world networks show that our method provides better performance in most networks, comparing with the existing state-of-the-art approaches.

Acknowledgment

The work was supported by Natural Science Foundation of China (61772361, 61502334, 61673293, 61471259).

References

[1] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, p. 016107, Mar. 2011.

[2] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *WSDM 2013*, Rome, Italy, Feb. 2013, p. 587596.

[3] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *IJCAI 2016*, New York, USA, 2016, p. 22522258.

[4] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *Proceedings of the 22nd international conference on World Wide Web*, Rio de Janeiro, Brazil, May 2013, pp. 1089–1098.

[5] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM 2013*, Dallas, Texas, Dec. 2013, p. 11511156.

[6] S. Pool, F. Bonchi, and M. V. Leeuwen, "Description-driven community detection," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, p. 28, 2014.

[7] S. Chang, W. Han, J. Tang, G. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *KDD 2015*, Sydney, Australia, Aug. 2015, p. 119128.

[8] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering," in *AAAI 2014*, Quebec, Canada, 2014, pp. 1293–1299.

[9] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *KDD 2014*, New York, USA, Aug. 2014, p. 977986.

[10] H. Hu, Z. Lin, J. Feng, and J. Zhou, "Smooth representation clustering," in *CVPR 2014*, Columbus, Ohio, USA, 2014, p. 38343841.

[11] Y. Pei, N. Chakraborty, and K. Sycara, "Nonnegative matrix tri-factorization with graph regularization for community detection in social networks," in *IJCAI 2015*, Buenos Aires, Argentina, 2015, pp. 2083–2089.

[12] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large attribute networks," in *AAAI*, Phoenix, Arizona, USA, Feb. 2016, p. 265271.

[13] D. He, Z. Feng, D. Jin, X. Wang, and W. Zhang, "Joint identification of network communities and semantics via integrative modeling of network topologies and node contents," in *AAAI 2017*, San Francisco, California, USA, Feb. 2017, p. 116124.

[14] T. Yang, R. Jin, Y. Chi, and S. Zhu, "A bayesian framework for community detection integrating content and link," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Montreal, Quebec, Canada, 2009, p. 615622.

[15] G. Qi, C. Aggarwal, and T. Huang, "Community detection with edge content in social media networks," in *ICDE 2012*, Washington, USA, Apr. 2012, pp. 534–545.

[16] C. Wang, J. Lai, and P. Yu, "Neiwalk: Community discovery in dynamic content-based networks," *IEEE Trans. Knowl. Data. Eng.*, vol. 26, p. 17341748, 2014.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, p. 291294, 1988.

[19] W. Chen, Y. Song, H. Bai, C. Lin, and E. Chang, "Parallel spectral clustering in distributed systems," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 33, p. 568586, Apr. 2011.

[20] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[21] N. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, pp. 175–185, Feb. 1992.

[22] M. E. J. Newman, "Modularity and community structure in networks," *the national academy of sciences*, vol. 103, p. 85778582, 2006.

[23] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 22, p. 888905, Aug. 2000.

[24] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, 1998, p. 948.

[25] H. Liu, Z. Wu, X. Li, D. Cai, and T. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 34, p. 12991311, Nov. 2012.

[26] A. Lancichinetti, S. Fortunato, and J. Kertesz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, p. 033015, Mar. 2009.

[27] R. Balasubramanyan and W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, Montreal, Quebec, Canada, Oct. 2011, p. 450461.