Information Sciences 493 (2019) 75–90

Contents lists available at ScienceDirect

Information Sciences

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

High-performance community detection in social networks using a deep transitive autoencoder

Ying Xie[a,b], Xinmei Wang[b], Dan Jiang[b], Rongbin Xu[a,b,*]

[a] *School of Information Engineering, Putian University, Putian, China*
[b] *The School of Computer Science and Technology, Anhui University, Hefei, China*

ARTICLE INFO

Article history:
Received 18 December 2018
Revised 8 April 2019
Accepted 10 April 2019
Available online 17 April 2019

Keywords:
Community detection
Complex networks
Deep autoencoder
Unsupervised transfer learning

ABSTRACT

Community structure is an important characteristic of complex networks. It determines where important functions of a network are located. Recently, discovering community structure in complex networks has become a hot topic of research. However, the continuous increase in network size has made network structure more complex, and community detection has become extremely difficult in real applications. In particular, the detection results are usually not accurate enough when classical clustering methods are applied to high-dimensional data matrices. In this paper, inspired by the relationship between vertices, we design a novel and effective network adjacency matrix transformation method to describe vertices' similarity in the network topology. On this basis, we propose a framework to extract nonlinear features: community detection with deep transitive autoencoder (CDDTA). This framework can obtain powerful nonlinear features of a real network to make community detection algorithms perform excellently in practice. We further incorporate unsupervised transfer learning into the CDDTA (Transfer-CDDTA) by minimizing the Kullback–Leibler divergence of embedded instances, to discover powerful low-dimensional representations. Finally, we propose a new training strategy and optimization method for our algorithm. Extensive experimental results indicate that our new framework can ensure good performance on both real-world networks and artificial benchmark networks, which outperforms most of the state-of-the-art methods for community detection in social networks.

1. Introduction

Complex networks are composed of vertices and edges (representing relationships) between those vertices [6,14]; examples include social, biological, and web networks. The community detection problem has attracted considerable attention from researchers within the domain of complex networks [30]. Research on community detection methods and network science is very important in text analysis, bibliometrics, and data analysis [2]. Some research achievements of community detection have been successfully applied to various fields, such as friend recommendation, personalized product promotion, protein function prediction, and public opinion analysis and processing. Community detection requires dividing a network into groups of tightly connected vertices, while vertices belonging to different groups have only sparse connections. An ad-

* Corresponding author.
E-mail addresses: xieyingscholar@126.com (Y. Xie), e16201092@stu.ahu.edu.cn (X. Wang), e17201021@stu.ahu.edu.cn (D. Jiang), xurb_910@126.com (R. Xu).

https://doi.org/10.1016/j.ins.2019.04.018
0020-0255/© 2019 Elsevier Inc. All rights reserved.

jacency matrix can be easily obtained from any network. However, using the adjacency matrix as a similarity representation of the network can only render the connections between vertices, which greatly affects the accuracy of community detection. Clustering method is very important and has a wide range of applications in exploring data relationships. However, if clustering is used as a method for community partition, the detection results are usually not accurate enough when these methods are used to deal with high-dimensional data.

Community detection aims to effectively detect the structure of communities in complex networks. There is a variety of approaches to developing network community detection methods. For example, the multilayer ant-based algorithm (MABA) [8] implements community detection by adopting the method of locally optimizing modularity with individual ants. The spectral algorithm (SP) [27] is used to detect community structure by spectral analysis of a Laplace or modularity matrix. In addition, many algorithms have been put forward to solve the detection problem of complex network topology, including: (1) Algorithms based on modularity optimization. For example, the unified link and content (ULC) model [23] is a heuristic searching algorithm that attempts to solve the modularity optimization problem. It optimizes the global parameters by adjusting local extreme values, to improve the efficiency of computation. Fast and accurate mining (FAM) [12] is a fast mining algorithm based on modularity, which belongs to the class of greedy agglomeration algorithms. The fast unfolding algorithm (FUA) [10] is a local optimization and hierarchical clustering algorithm, which takes less computing time than many other network clustering algorithms. (2) Algorithms based on label passing. For example, the label propagation algorithm (LPA) [24] is the earliest tag-based algorithm, whose basic aim is to predict the tag information of untagged vertices from tag information of a tagged vertex. (3) Dynamic algorithms. For example, finding and extracting communities (FEC) [32] is a community detection algorithm which considers both connection density and connection symbols of complex networks. It can effectively deal with the relationship between vertices and find more reasonable network community structures. The Infomap algorithm [25] adopts random walks as a process for information propagation on the network to generate the corresponding data flow. According to the network topology, these methods can solve the problem of community detection from different angles. However, in real-world applications, networks usually contain many nonlinear properties, so the detection ability of those models may be limited in real-world networks.

Deep learning research has made remarkable achievements in the fields of machine learning and artificial intelligence. Theoretical studies, algorithm designs, and application systems for neural networks have been widely proposed in many fields, such as speech identification, image classification, and natural language processing [5]. Multilayer neural networks can effectively reduce the data dimensionality [33] and achieve good community detection results. High-dimensional original data can be encoded into a new feature representation through a multilayer neural network, with a bottleneck layer to best approximate the original input data. This method can reduce data dimensionality better than classical dimensionality reduction methods. However, existing neural network methods cannot perform feature extraction well, in the feature representation of community detection. Therefore, we need to discover more effective deep neural network approaches to obtain low-dimensional features.

In pattern recognition, transfer learning aims to establish a target prediction model with efficient generalization ability. It aims to transfer effective information, which may contain a limited amount of valid label information, from a source domain to a related target domain. Therefore, it has become an important research problem in transfer learning methods to obtain more powerful feature representations for instances of source and target domains. In complex networks, a good feature representation can help detect the topological structure. However, low-dimensional feature representations are not strong enough in mainstream community detection methods based on deep learning. Traditional transfer methods inspired by deep learning methods are to learn a more powerful feature representation. However, most methods do not explicitly emphasize minimizing the differences between various domains. In addition, most transfer methods use supervised learning, and valid labels are limited to existing community detection tasks. To achieve better performance in regard to the problem of community detection, unsupervised transfer learning is urgently required to assist a deep autoencoder.

In summary, there are three main contributions of our paper:

(1) We propose a novel method to transform a network to an adjacency matrix, to describe the similarity of vertices in a network. A new method, which applies an effective algorithm based on a deep transitive autoencoder, is proposed to transmit similarity information and detect communities in complex networks. The method is named community detection with deep transitive autoencoder (CDDTA).
(2) We incorporate unsupervised transfer learning into the proposed algorithm by measuring Kullback–Leibler (KL) divergence of embedded instances, to ensure that the differences between different domains can be approximately equal when learning low-dimensional representations.
(3) We propose a training strategy for our novel framework, in which the target domain shares common parameters with the source domain, for encoding and decoding in the process of deep transitive autoencoder training. We also optimize the training algorithm by a back-propagation method with stochastic gradient descent.

The rest of this paper is organized as follows. Section 2 introduces the related work. The proposed CDDTA algorithm is introduced in Section 3. We further incorporate unsupervised transfer learning into the CDDTA algorithm (named Transfer-CDDTA) by KL divergence of embedded instances, in Section 4. Section 5 analyzes the detection results on various social networks. Finally, Section 6 presents the conclusions.

2. Related work

In real life, many complex relationships can be represented by networks with complex structural characteristics. The community structure of complex networks has the characteristic that tightly interconnected vertices form groups and there are sparse connections between vertices in different groups. We introduce some relevant community detection and transfer learning methods in this section.

2.1. Community detection methods

Yang et al. [30] proposed a deep nonlinear reconstruction (DNR) model, which adopted the modularity matrix as the similarity matrix and aimed to obtain a powerful representation for complex networks using a stacked autoencoder. The proposed DNR method was further improved as a semi-supervised DNR algorithm by incorporating pairwise constraints. Liu et al. [13] presented a community detection method based on label propagation, which used effective prior information to affect the detection ability of complex network structures. The label information of other vertices was gradually inferred according to the labels of known vertices in the network topology structure. Ma et al. [16] integrated pairs of mandatory and non-consecutive constraints into an adjacency matrix and solved the community detection problem by symmetric non-negative matrix decomposition. Amancio et al. [3] presented a method to improve the time performance of Walktrap and fast greedy algorithms, to implement community detection on incomplete networks.

Pasta et al. [21] focused on the structure and topology of networks and regularly evaluated the detection ability of community detection along with changes in network structure. He et al. [8] presented a local optimization-based algorithm, which could effectively detect high-resolution community structure and had the advantage of low computational complexity. Psorakis et al. [22] presented a stochastic model of Bayesian nonnegative matrix factorization, based on computational efficiency, which had good performance on community detection. Mall et al. [17] presented a parameter-free kernel spectral clustering method that used the projection structure in feature space to automatically identify the number of communities, to perform community detection in large-scale networks.

A high-dimensional similarity matrix cannot reflect the main features of network topology, and it has a weak ability to express community structure. Existing community detection algorithms generally only focus on connections between vertices and often ignore the transmission of structural information. We describe the similarity of vertices in network topology by a transformative network adjacency matrix during the process of community detection. Therefore, transmitting the structural information by our proposed deep transitive autoencoder can obtain a more efficient low-dimensional feature representation.

2.2. Transfer learning

Transfer learning, as the name suggests, is to transfer effective information from a source domain to a related target domain. Considering whether tasks of domains are the same, and whether samples from different domains are labeled, existing studies of transfer learning can be categorized into transductive transfer learning, unsupervised transfer learning, and inductive transfer learning. From experience and theories, transfer learning has the potential of achieving a better performance, particularly when target data contain a limited amount of valid label information.

Martín-Wanton et al. [18] proposed a transfer model that combined the target set with a mass of unlabeled tweets to help improve the clustering performance of the target set. Shao et al. [26] presented a transfer framework that mapped the different domains' data to a generalized subspace, and further maintained the structure of the two domains by adding low-rank constraints during the transfer process. Pan et al. [20] proposed a dimension reduction transfer learning method by minimizing the differences between different domains' data in a potential semantic space.

Considering the characteristics of nonlinear properties and high dimensionality existing in real-world networks, our proposed method in this paper is different from the above work. We adopt a deep autoencoder neural network to build our new model. In addition, we employ a network adjacency matrix transformation method to represent the similarity relationship between vertices in the network topology, and we use the adjacency matrix as the input of a deep autoencoder, to transmit similarity information. We call the network adjacency matrix a deep transitive autoencoder. To obtain a more efficient low-dimensional feature representation, we incorporate unsupervised transfer learning into the proposed algorithm by measuring the KL divergence of embedded instances, to ensure that the differences between the different domains are approximately equal. During the training process, we set the two domains to share the same model parameters.

3. Community detection with deep transitive autoencoder model

In this section, we introduce the proposed community detection method based on a deep transitive autoencoder. We also describe an adjacency matrix transformation method to represent the similarity of vertices in a network topology.

A social network can be modeled as an undirected and unweighted graph $G = \{V, E\}$, in which V represents the set of N vertices, where $V = \{v_1, v_2, \cdots, v_N\}$, and $E = \{e_{ij}\}$ represents the set of edges. Here, we use a symmetric matrix $A = [a_{ij}] \in R^{N\times N}$ as the adjacency matrix to characterize connection relationships between vertices, where each element of matrix A is 0 or 1. If there is an edge between vertices i and j, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. We also define $a_{ii} = 0$ for all $1 \le i \le N$.

In this paper, the community detection problem can be simulated to classify vertices into different groups or communities, $\{V_i\}_{i=1}^{C}$, based on the topological information in the networks, where C represents the number of communities in G. We mainly consider non-overlapping community structure, that is, $V_i \cap V_j = \emptyset$ for $i \neq j$.

3.1. Network adjacency matrix transformation

Our CDDTA algorithm takes the network adjacency matrix $A = [a_{ij}] \in R^{N\times N}$ as the input. In general, A can be used as the similarity matrix of the network, and the entries of A can be used to characterize the similarity relationships between vertices in the network. A sparse adjacency matrix A can characterize the edges between vertices in social networks. However, using A as a similarity matrix of a network can only represent the similarity relationships between directly connected pairs of vertices; it cannot represent the similarity relationships between vertices that are not directly connected. Therefore, we use the following transformation method to obtain a new similarity matrix that keeps the similarity relationship information of many pairs of vertices and reflects the local information of vertices, and further improves the accuracy of community detection. The transformation of the similarity matrix is inspired by normalized cuts based on low-dimensional continuous space representation for each vertex in the graph. The transformation is formulated as follows:

$$D_1 = DG\left(d_1^{-1/2}\right) \tag{1}$$

where $DG(\cdot)$ is equivalent to the $diag(\cdot)$ function in MATLAB, which is used to construct a diagonal matrix. d_1 is the vector whose elements are the sums of the rows of A, that is, the degrees of the network vertices. The exponentiation is carried out element-wise. Therefore, we construct matrices A_1 and A_2 as

$$\begin{aligned} A_1 &= D_1 A D_1 \\ A_2 &= A_1 - DG(DG(A_1)) \end{aligned} \tag{2}$$

We then compute $2k$ eigenvectors U_1, associated with the $2k$ largest eigenvalues of A_2. In MATLAB notation, this can be concisely expressed as

$$U_1 = Eig(A_2, 2k) \tag{3}$$

where $Eig(\cdot)$ is represented as the $eigs(\cdot)$ function in MATLAB and $k = C$, which indicates the size of the network communities. This value is the optimal value obtained through continuous training during the experiment.

Finally, we obtain the transformed similarity matrix M, which can transmit structure information more effectively:

$$M = \left(DG\left(d_2^{-1/2}\right)U_1\right)^T \tag{4}$$

where d_2 is the sum of the rows in matrix U_1^2 (exponentiation done element-wise). M is the input matrix to the autoencoder. The dimension of M is $2k \times N$: each vertex can be represented by a vector with length $2k$ on the network, which also determines the dimensions of the input and output layers of the deep autoencoder.

3.2. Community detection based on deep transitive autoencoder

An autoencoder is a neural network minimizing input and output information, which is a method for unsupervised learning. It has a good ability to obtain low-dimensional representation of data. A single-layer autoencoder consists of both encoding and decoding components. An autoencoder extracts the distinguishing features of the input data by reconstructing the original data. The deep transitive autoencoder we propose in this paper can transmit structure information more effectively by transforming the similarity matrix and can obtain an efficient low-dimensional feature representation, to improve the detection ability in complex social networks.

We employ a novel similarity matrix $M = [m_{ij}] \in R^{2k\times N}$ as the original input data of an autoencoder. $m_i \in R^{2k\times 1}$ represents the i-th vertex of the similarity matrix M. We adopt m_i as the i-th input vector for the autoencoder, and M is mapped to a low-dimensional embedding space $H = [h_{ij}] \in R^{d\times N}$ by

$$h_i = S(Wm_i + p) \tag{5}$$

where $S(\cdot)$ is an activation function in the encoder; here we use the sigmoid function $S(m) = \frac{1}{1+e^{-m}}$. This sigmoid function has a characteristic S-shaped curve which can return values increasing from 0 to 1 monotonically in the real number domain. In this paper, we assume that the network models probabilities of various events in the internal or output layers. $W \in R^{d\times 2k}$ and $p \in R^{d\times 1}$ are the weight and bias terms, respectively, to be learned by the encoder.

After we input the new representation $h_i \in R^{d\times 1}$ into the decoder layer, the low-dimensional embedding representation H is mapped back to the original data space by

$$\hat{m}_i = S(\tilde{W}h_i + q) \tag{6}$$

where $\tilde{W} \in R^{2k\times d}$ and $q \in R^{2k\times 1}$ are the weight and bias terms, respectively, to be learned by the decoder.

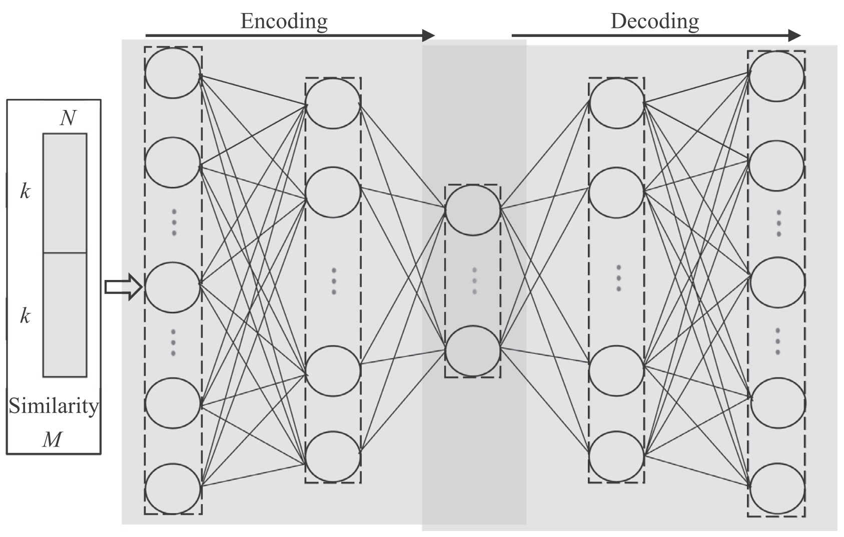


Fig. 1. A deep transitive autoencoder neural network, including a new similarity matrix that keeps the similarity relationship information of many pairs of vertices and reflects the local information of vertices. We use the novel transformed similarity matrix M to transmit structure information more effectively.

The autoencoder aims to learn a novel feature representation H, which best approximates the original input data M. In the process of training, the autoencoder automatically adjusts the weight and bias terms. The objective function is as follows:

$$\mathcal{J}(M, \hat{M}) = \min_{\{W, \tilde{W}, p, q\}} \sum_{i=1}^{N} \|\hat{m}_i - m_i\|_2^2 = \min_{\{W, \tilde{W}, p, q\}} \sum_{i=1}^{N} \|S(\tilde{W}S(Wm_i + p) + q)\|_2^2 \tag{7}$$

The weight W and bias term p can be obtained by training the autoencoder, and the new low-dimensional nonlinear feature representations can be obtained by Eq. (1).

To obtain the feature matrix with a strong expression ability for network topology, we introduce a deep transitive autoencoder with a powerful adjacency expression. Its structure is shown in Fig. 1.

4. Unsupervised transfer learning-inspired deep transitive autoencoder

4.1. Problem formalization

Transfer learning builds a target prediction model with good generalization performance by transferring knowledge from the source to the target domain. Because community detection may lack sufficient information in the real world, transfer learning is an effective way to solve this problem. In addition, to assist the deep transitive autoencoder to obtain a more powerful feature representation, we incorporate unsupervised transfer learning into our method and further introduce a novel framework, named Transfer-CDDTA. Suppose the source training set of N_s instances, $D_s = \{m_i^{(s)}\}|_{i=1}^{N_s}$, is the source domain, and the target training set of N_t instances, $D_t = \{m_i^{(t)}\}|_{i=1}^{N_t}$, is the target domain. Here, the two domains share common model parameters and feature space: that is, each input feature vector, $m_i \in R^N$. Fig. 2 shows the whole Transfer-CDDTA framework.

Therefore, as shown in the learning process of the overall framework in Fig. 2, we can easily obtain the overall loss function in our proposed Transfer-CDDTA learning framework; its expression is shown below:

$$\mathcal{J} = \mathcal{J}_r(M, \hat{M}) + \alpha\mathcal{L}(H^{(s)}, H^{(t)}) + \beta\Omega(\theta) \tag{8}$$

where $r \in (s, t)$ represents the source and target domains, respectively. α and β are parameters that can regulate and control the effect of each term in the overall objective function. We explain each of these loss functions in detail below.

In Eq. (8), the first term is the reconstruction error of the two domains in the objective function. Because the reconstructed error expression of the source domain is equivalent to the target domain, they can be integrated into one formula; the expression is shown below:

$$\mathcal{J}_r(M, \hat{M}) = \sum_{r \in (s,t)} \sum_{i=1}^{N_r} \|\hat{m}_i - m_i\|_2^2 \tag{9}$$

where each encoding and decoding process of the deep autoencoder can be obtained by the following formula:

$$o_i^{(r)} = S\big(W_1 m_i^{(r)} + p_1\big), \qquad h_i^{(r)} = S\big(\tilde{W}_1 o_i^{(r)} + q_1\big)\hat{o}_i^{(r)} = S\big(\tilde{W}_2 h_i^{(r)} + q_2\big), \qquad \hat{m}_i^{(r)} = S\big(W_2 \hat{o}_i^{(r)} + p_2\big) \tag{10}$$

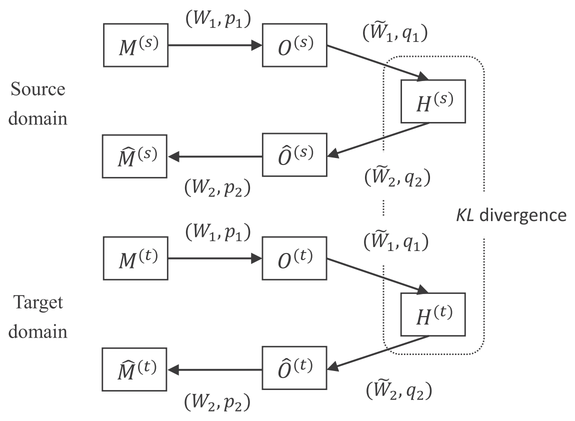


Fig. 2. Transfer Learning-inspired Community Detection with Deep Transitive Autoencoder (Transfer-CDDTA) framework. This framework incorporates unsupervised transfer learning into the proposed algorithm by measuring the Kullback–Leibler (KL) divergence of embedded instances to ensure that the differences between different domains can be approximately equal when learning low-dimensional representation.

The second term of Eq. (8) is the KL divergence of the embedded instances between two domains, which can ensure that the distribution similarity of different domains' data is in the low-dimensional space; this can be defined as

$$\mathcal{L}(H^{(s)}, H^{(t)}) = D_{KL}(Q_s \| Q_t) + D_{KL}(Q_t \| Q_s) \tag{11}$$

where the specific representation of each item in Eq. (11) is shown below:

$$\begin{aligned} Q_s &= \frac{Q_s^{'}}{\sum Q_s^{'}}, \qquad Q_s^{'} = \frac{1}{N_s} \sum_{i=1}^{N_s} h_i^{(s)} \\ Q_t &= \frac{Q_t^{'}}{\sum Q_t^{'}}, \qquad Q_t^{'} = \frac{1}{N_t} \sum_{i=1}^{N_t} h_i^{(t)} \end{aligned} \tag{12}$$

To reduce overfitting and improve the generalization ability of the model, we add a regularization term to the parameters, which can be defined as follows:

$$\Omega(\theta) = \|W_1\|^2 + \|p_1\|^2 + \|W_2\|^2 + \|p_2\|^2 + \|\tilde{W}_1\|^2 + \|q_1\|^2 + \|\tilde{W}_2\|^2 + \|q_2\|^2 \tag{13}$$

4.2. Optimization

We solve the minimization problem in Eq. (8) by back-propagation with the stochastic gradient descent method. During the iteration, we update the weighting parameter $\{W_1, p_1, W_2, p_2, \tilde{W}_1, q_1, \tilde{W}_2, q_2\}$ alternatively and iteratively, by applying the following rules until the solutions converge:

$$\begin{aligned} &\leftarrow W_1 - \gamma \frac{\partial \mathcal{J}}{\partial W_1}, \qquad p_1 \leftarrow p_1 - \gamma \frac{\partial \mathcal{J}}{\partial p_1} \\ W_2 &\leftarrow W_2 - \gamma \frac{\partial \mathcal{J}}{\partial W_2}, \qquad p_2 \leftarrow p_2 - \gamma \frac{\partial \mathcal{J}}{\partial p_2} \\ \tilde{W}_1 &\leftarrow \tilde{W}_1 - \gamma \frac{\partial \mathcal{J}}{\partial \tilde{W}_1}, \qquad q_1 \leftarrow q_1 - \gamma \frac{\partial \mathcal{J}}{\partial q_1} \\ \tilde{W}_2 &\leftarrow \tilde{W}_2 - \gamma \frac{\partial \mathcal{J}}{\partial \tilde{W}_2}, \qquad q_2 \leftarrow q_2 - \gamma \frac{\partial \mathcal{J}}{\partial q_2} \end{aligned} \tag{14}$$

where γ is the learning rate. Because the partial derivative expression of the overall loss function may be too long, and to simplify the mathematical expression, we first define the following custom functions:

$$\begin{aligned} g(x_i, \hat{x}_i) &= 2(\hat{x}_i - x_i)\hat{x}_i(1 - \hat{x}_i) \\ s(x_i) &= x_i(1 - x_i) \end{aligned} \tag{15}$$

Therefore, we can derive the partial derivative of the whole cost function J:

$$\frac{\partial \mathcal{J}}{\partial W_2} = \sum_{r \in (s,t)} \sum_{i=1}^{N_r} g\big(m_i^{(r)}, \hat{m}_i^{(r)}\big) \hat{o}_i^{(r)T} + 2\beta W_2 \tag{16}$$

$$\frac{\partial \mathcal{J}}{\partial p_2} = \sum_{r\in(s,t)} \sum_{i=1}^{N_r} g\left(m_i^{(r)}, \hat{m}_i^{(r)}\right) + 2\beta p_2 \tag{17}$$

$$\frac{\partial \mathcal{J}}{\partial \tilde{W}_2} = \sum_{r\in(s,t)} \sum_{i=1}^{N_r} W_2^T g\left(m_i^{(r)}, \hat{m}_i^{(r)}\right) s\left(\hat{o}_i^{(r)}\right) h_i^{(r)T} + 2\beta \tilde{W}_2 \tag{18}$$

$$\frac{\partial \mathcal{J}}{\partial q_2} = \sum_{r\in(s,t)} \sum_{i=1}^{N_r} W_2^T g\left(m_i^{(r)}, \hat{m}_i^{(r)}\right) s\left(\hat{o}_i^{(r)}\right) + 2\beta q_2 \tag{19}$$

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \tilde{W}_1} = & \sum_{r\in(s,t)} \sum_{i=1}^{N_r} \tilde{W}_2^T \left(W_2^T g(m_i^{(r)}, \hat{m}_i^{(r)}) s(\hat{o}_i^{(r)})\right) s\left(h_i^{(r)}\right) o_i^{(r)T} + \frac{\alpha}{N_s} \sum_{i=1}^{N_s} s\left(h_i^{(s)}\right)\left(1 - \frac{Q_t}{Q_s} + ln\left(\frac{Q_s}{Q_t}\right)\right) o_i^{(s)T} \\ & + \frac{\alpha}{N_t} \sum_{i=1}^{N_t} s\left(h_i^{(t)}\right)\left(1 - \frac{Q_s}{Q_t} + ln\left(\frac{Q_t}{Q_s}\right)\right) o_i^{(t)T} + 2\beta \tilde{W}_1\end{aligned} \tag{20}$$

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial q_1} = & \sum_{r\in(s,t)} \sum_{i=1}^{N_r} \tilde{W}_2^T \left(W_2^T g(m_i^{(r)}, \hat{m}_i^{(r)}) s(\hat{o}_i^{(r)})\right) s\left(h_i^{(r)}\right) + \frac{\alpha}{N_s} \sum_{i=1}^{N_s} s\left(h_i^{(s)}\right)\left(1 - \frac{Q_t}{Q_s} + ln\left(\frac{Q_s}{Q_t}\right)\right) \\ & + \frac{\alpha}{N_t} \sum_{i=1}^{N_t} s\left(h_i^{(t)}\right)\left(1 - \frac{Q_s}{Q_t} + ln\left(\frac{Q_t}{Q_s}\right)\right) + 2\beta q_1\end{aligned} \tag{21}$$

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial W_1} = & \sum_{r\in(s,t)} \sum_{i=1}^{N_r} W_2^T g\left(m_i^{(r)}, \hat{m}_i^{(r)}\right)\left(\tilde{W}_1^T (\tilde{W}_2^T s(\hat{o}_i^{(r)}) s(h_i^{(r)}))\right) s(o_i^{(r)}) m_i^{(r)T} + \frac{\alpha}{N_s} \sum_{i=1}^{N_s} \tilde{W}_1^T s(h_i^{(s)})\left(1 - \frac{Q_t}{Q_s} + ln\left(\frac{Q_s}{Q_t}\right)\right) s(o_i^{(s)}) m_i^{(s)T} \\ & + \frac{\alpha}{N_t} \sum_{i=1}^{N_t} \tilde{W}_1^T s\left(h_i^{(t)}\right)\left(1 - \frac{Q_s}{Q_t} + ln\left(\frac{Q_t}{Q_s}\right)\right) s\left(o_i^{(t)}\right) m_i^{(t)T} + 2\beta \tilde{W}_1\end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial p_1} = & \sum_{r\in(s,t)} \sum_{i=1}^{N_r} W_2^T g\left(m_i^{(r)}, \hat{m}_i^{(r)}\right)\left(\tilde{W}_1^T \left(\tilde{W}_2^T s\left(\hat{o}_i^{(r)}\right) s\left(h_i^{(r)}\right)\right)\right) s\left(o_i^{(r)}\right) + \frac{\alpha}{N_s} \sum_{i=1}^{N_s} \tilde{W}_1^T s\left(h_i^{(s)}\right)\left(1 - \frac{Q_t}{Q_s} + ln\left(\frac{Q_s}{Q_t}\right)\right) s\left(o_i^{(s)}\right) \\ & + \frac{\alpha}{N_t} \sum_{i=1}^{N_t} \tilde{W}_1^T s\left(h_i^{(t)}\right)\left(1 - \frac{Q_s}{Q_t} + ln\left(\frac{Q_t}{Q_s}\right)\right) s(o_i^{(t)}) + 2\beta p_1\end{aligned} \tag{23}$$

The specific description of our framework is shown in Algorithm 1. The time complexity is $O(k \times L)$, where k and L are the number of iterations and the number of layers of the autoencoder, respectively.

Algorithm 1 Unsupervised transfer learning-inspired community detection with Deep Transitive Autoencoder (Transfer-CDDTA).

```
Input: Adjacency matrix $A = [a_{ij}] \in R^{N\times N}$
Input: The trade-off parameters $\alpha$, $\beta$, $\gamma$
Output: The embedded layer $H \in R^{d\times N}$
1: Obtain a novel similarity matrix $M = [m_{ij}] \in R^{2k\times N}$ by Eq. (7) as original input data of autoencoder.
2: Divide the matrix $M$ into source domain $D_s$ and target domain $D_t$.
3: Initialize the weight matrix $\{W_1, \tilde{W}_1, \tilde{W}_2, W_2\}$ and the bias vector $\{p_1, q_1, q_2, p_2\}$.
4: Perform a feedforward propagation and compute the activation value of each layer according to Eq. (10).
5: repeat
6:     Compute the partial derivatives of all variables and iteratively update the variables by Eqs. (16)–(23).
7:     Iteratively update the variables by using Eq. (14).
8: until convergence
9: Calculate the obtained low-dimensional codes by Eq. (10).
10: return the embedded layer $H$
```

5. Experimental analysis

To evaluate the proposed CDDTA method and Transfer-CDDTA framework, we analyze their performance on some widely used real networks and two types of synthetic benchmark networks. We compare our framework with some efficient methods to prove the validity of our method. Here, we employ 20-iteration k-means as our clustering method. We conducted our experiments on an NVIDIA GeForce GTX 1070 Ti with MATLAB 2015a.

5.1. Experimental setup

5.1.1. Comparing methods

We mainly analyze the performance of our CDDTA method by comparing it with DNR [30], including both DNR-L2 and DNR-CE methods, which are novel deep nonlinear reconstruction approaches based on modularity. We also adopt seven existing community detection methods to help prove the effectiveness of our method: MABA [8], ULC [23], SP [27], FUA [10], Infomap [25], FEC [32], and FAM [12].

5.1.2. Parameter settings

In the process of training a neural network, selecting the right number of nodes in each layer can promote the autoencoder to achieve a good feature extraction effect. Because the experiment mainly selects the middle-hidden layer of the deep autoencoder, to perform cluster analysis, the deep neural network was set as a full autoencoder with dimensions 2k-30-C-30-2k for all datasets, to ensure that the results are accurate. Here, 2k-30-C-30-2k is the layer configuration of the deep autoencoder, which is the dimensional change of dataset followed by 2k, 30, C, 30, 2k, where $k = C$. The learning rate γ of the deep autoencoder was set to 0.01. During the training, we set the number of iterations to 100000. We trained our model with 50 random initializations and obtained new representations from the deep transitive autoencoder for clustering, so as to obtain new communities.

5.1.3. Evaluation metric

In the field of community detection, community evaluation indicators are usually used to measure the quality of the network partition [9]. Here, we employ two standard community evaluation methods to analyze the performance of community detection: normalized mutual information (NMI) and modularity.

The formula for NMI is:

$$NMI(Y_1, Y_2) = \frac{-2\sum_{ij} n_{ij} log\frac{n_{ij}n_t}{n_{i.}n_{.j}}}{\sum_i n_{i.} log\frac{n_{i.}}{n_t} + \sum_j n_{.j} log\frac{n_{.j}}{n_t}} \tag{24}$$

where Y_1 represents the ground-truth label information, Y_2 represents the computed label information, and n_{ij} represents the number of vertices in the ground-truth community i that are assigned to the computed community j. Here, $n_{i.}$ and $n_{.j}$ are the numbers of vertices in communities i and j, respectively. In addition, $n_t = \sum_i \sum_j n_{ij}$.

Another commonly used measure is modularity. The formula for modularity is:

$$Q = \frac{1}{2m}\sum_{ij}\left(a_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{25}$$

$$\delta(u, v) = \begin{cases} 1 & \text{u=v} \\ 0 & \text{otherwise} \end{cases}$$

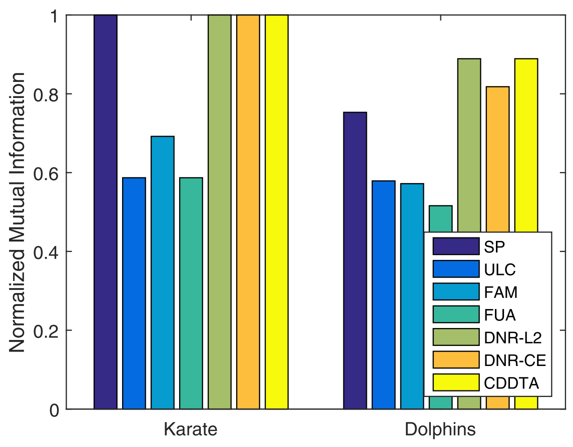
where k_i is the degree of vertex i, c_i represents the community of vertex i, and m is the number of edges in the graph. In experiments, we can also confirm that maximizing Q returns sensible divisions of networks in practice.

5.2. Community detection with deep transitive autoencoder results

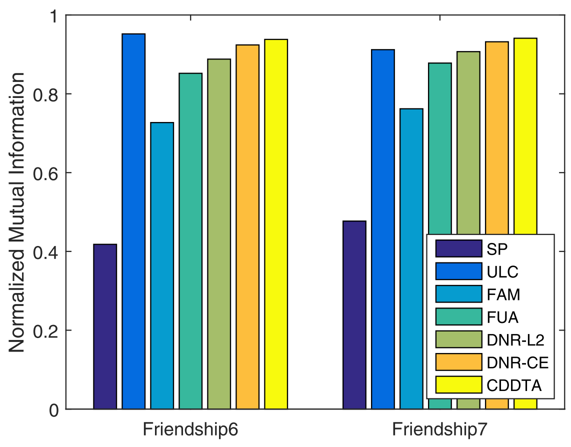
5.2.1. Real-world networks

We used eleven network datasets, with various sizes, that are widely used in community detection research, to assess the effectiveness of our framework on real-world networks. The information about these datasets is shown in Table 1, which lists the numbers of vertices, edges, and communities in the real networks. Each dataset includes both network topology and ground-truth information about community membership.

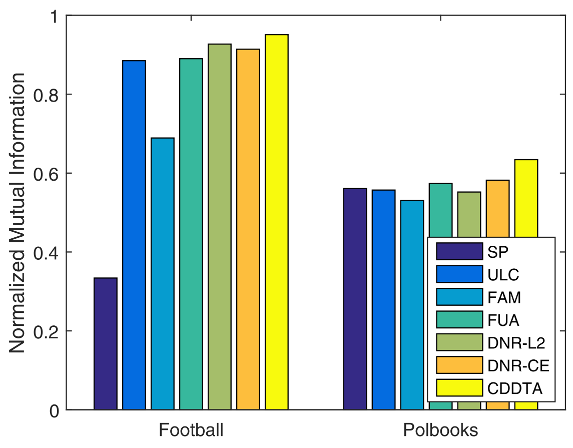
To assess the advantages of our new method, we compared CDDTA with six existing typical methods, including SP, ULC, FAM, FUA, DNR-L2, and DNR-CE; the results are shown in Fig. 3. We find that the detection effect of our method is competitive with DNR (DNR-L2 and DNR-CE), which is also better than the other four classical algorithms. This is because we adopt a novel pretreatment approach to effectively improve the local information of vertices. At the same time, the deep transitive autoencoder can effectively extract the characteristic information of the similarity matrix when dealing with the high-dimensional adjacency matrix. As a result, a low-dimensional feature matrix with more effective features can be obtained. From these results, we can conclude that our approach is more effective than other well-known community detection



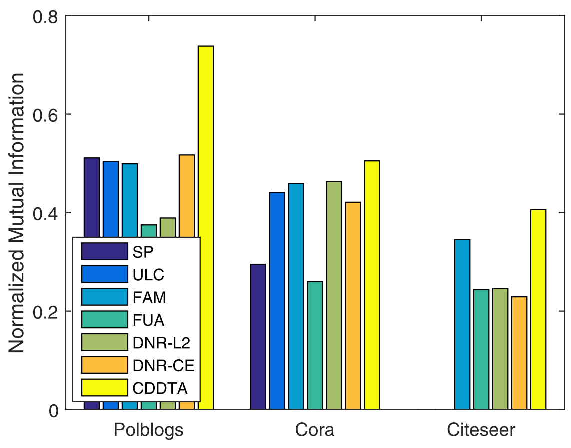
(a)



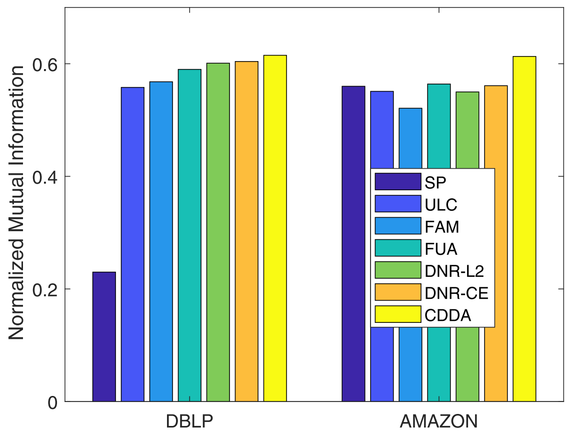
(b)



(c)



(d)



(e)

Fig. 3. Comparison of CDDTA with six existing methods on eleven real-world networks.

Table 1
Dataset information.

Dataset	Vertices	Edges	Communities	Description
Karate [31]	34	78	2	Zachary's karate club
Dolphins [15]	62	159	2	Dolphin social network
Friendship6 [28]	69	220	6	High school friendship
Friendship7 [28]	69	220	7	High school friendship
Football [7]	115	613	12	American Division I college football-2000
Polbooks [19]	105	441	3	Books about US politics
Polblogs [1]	1,490	16,718	2	Blogs about US politics
Cora [4]	2,708	5,429	7	Citation type dataset
Citeseer[a]	3,312	4,732	6	Citation network of computer science
Amazon [29]	334,863	925,872	75,149	Amazon product network
DBLP [29]	371,080	1,049,866	13,477	DBLP collaboration network publications

[a] http://citeseer.ist.psu.edu/.

Table 2
Running time (seconds) on nine datasets.

Dataset	DNR	CDDTA
Karate	4	4
Dolphins	6	4
Friendship6	23	8
Friendship7	27	8
Football	32	11
Polbooks	24	6
Polblogs	45	15
Cora	79	57
Citeseer	82	60

methods on real networks. Furthermore, we also measure statistics on the running time of our algorithm. Because CDDTA is an algorithm based on deep learning, we use the DNR algorithm as a comparative method. The running time of CDDTA and DNR algorithms on nine datasets is shown in Table 2. Experimental results demonstrate that the CDDTA algorithm is comparable to the community detection method based on deep learning.

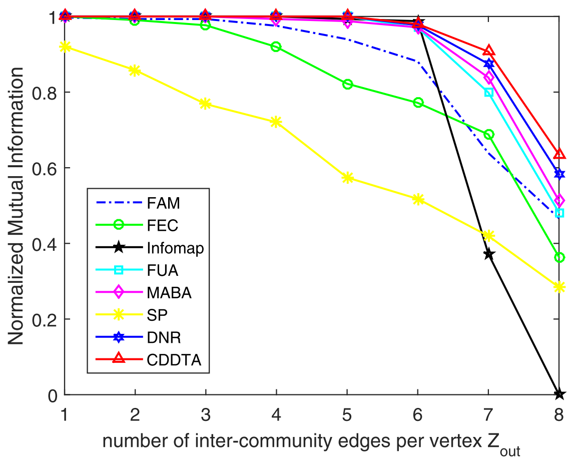
5.2.2. Artificial benchmark networks

The topological structure of synthetic networks may differ from real-world networks. We employed two widely used large-scale artificial benchmark networks to further prove the validity of our method: that is, the Girvan–Newman (GN) network [7] and the Lancichinetti–Fortunato–Radicchi (LFR) network [11].

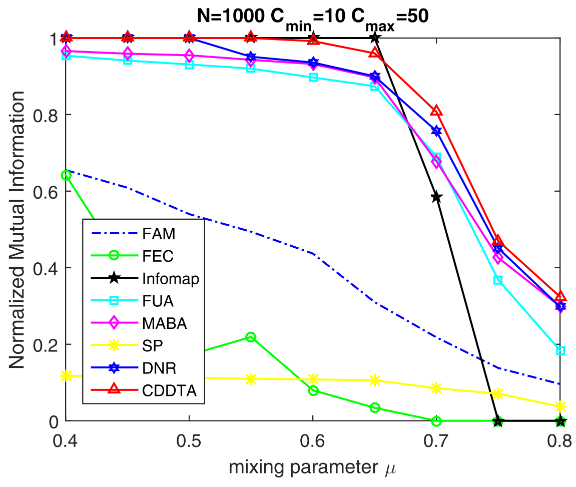
For the GN network, each network had 128 vertices, grouped into four communities, each containing 32 vertices. The average degree of each vertex was 16, of which an average of Z_{out} connected it to vertices in different communities and Z_{in} edges connected it to vertices in the same community, where Z_{out} and Z_{in} satisfy $Z_{out} + Z_{in} = 16$. We randomly generated eight different networks by using Z_{out} values between 1 and 8. Clearly, the community structure is clearer when Z_{out} is small but becomes more obscure, and the detection of communities becomes more difficult, as Z_{out} increases. We evaluated the CDDTA method by comparing it with seven methods on the GN network; the results are shown in Fig. 4(a). The performance of our CDDTA algorithm outperformed the seven methods on the GN network when $Z_{out} > 6$.

The LFR network has a more complicated structure than the GN network. It can generate more flexible networks, and is the most common simulation benchmark in current community detection research. With reference to the experimental setting in [11], we set the network size $N = 1000$ and the average degree of each vertex to 20. The variation range of mixing parameter μ was [0.4, 0.8], with spacing 0.05. The exponents of the community size and vertex degree distributions were set to -1 and -2, respectively. We generated two networks of different sizes. The minimum size of communities $C_{\min}$ was set to either 10 or 20, and the maximum size of communities $C_{\max}$ was $5 \times C_{\min}$. The experimental results for the two sets of networks, with the increasing of the mixing parameter μ, are shown in Figs. 4(b) and 4(c). The conclusion is that our CDDTA algorithm outperforms the other seven methods when $\mu > 0.65$ on LFR network ($N = 1000$, $C_{\min} = 10$, $C_{\max} = 50$), and $0.65 < \mu < 0.75$ on LFR network with larger communities ($N = 1000$, $C_{\min} = 20$, $C_{\max} = 100$). In addition, we increased the LFR network size to $N = 10000$; the experimental results are shown in Fig. 5. The conclusion is that our algorithm outperformed the comparative approaches when $0.65 < \mu < 0.7$ on LFR networks with two groups of community size ($N = 10000$, $C_{\min} = 10$, $C_{\max} = 50$ and $N = 10000$, $C_{\min} = 20$, $C_{\max} = 100$).

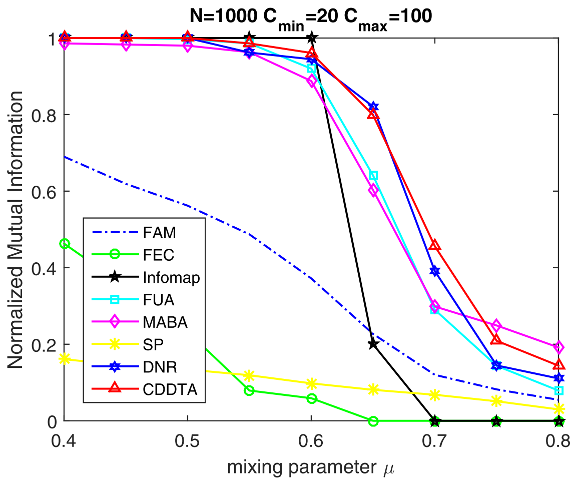
The experimental results on artificial benchmark networks indicate that our method is more effective than other community detection methods in detecting communities in networks with fuzzy community structure. It is also more competitive than most other popular comparative methods on community networks with clear structure.



(a) GN network.

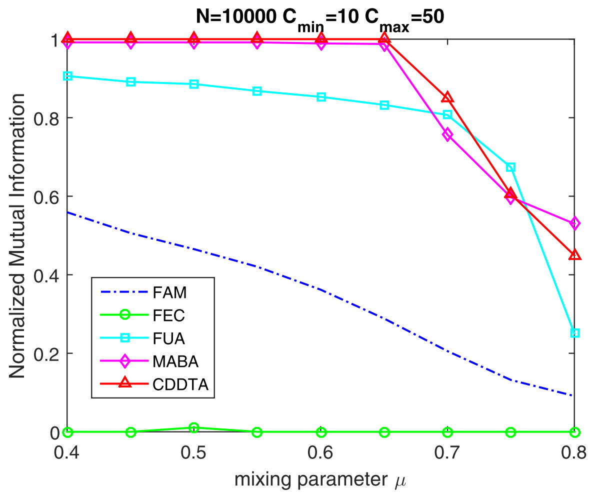


(b) LFR network (small communities).

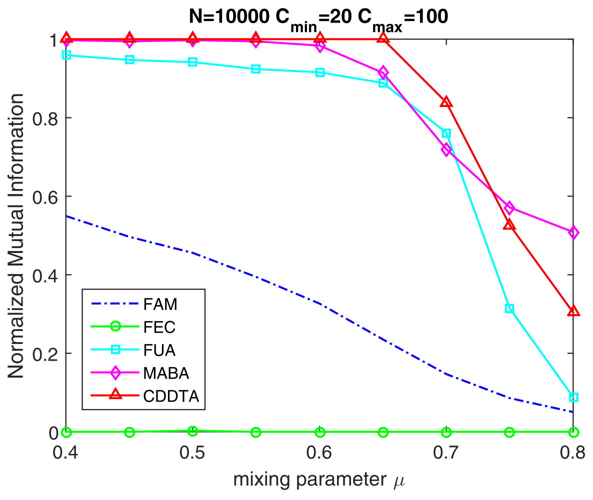


(c) LFR network (large communities).

Fig. 4. Comparison of CDDTA with seven existing methods on GN and LFR network.



(a)



(b)

Fig. 5. Experimental results on the two different sizes of LFR network ($N = 10000$, $C_{\min} = 10$, $C_{\max} = 50$ and $N = 10000$, $C_{\min} = 20$, $C_{\max} = 100$).

5.2.3. *Results of modularity*

We checked our CDDTA method against other baseline methods, using networks with no known ground truth partition. For a quantitative comparison between our method and others, we compared values of the Q modularity on nine datasets drawn from the literature. We compare results against seven previously published algorithms, which are widely used and have been incorporated into some of the more popular network analysis programs. The *modularity* values are shown in Table 3. In practice, the CDDTA method gives excellent performance.

In recent years, there has been particular interest in large networks. In our experiments, Table 3 reveals that the CDDTA algorithm clearly outperformed the baseline methods for all of the networks, particularly large networks, for the task of modularity optimization. It appears that the deep transitive autoencoder for detecting community structure may be the most effective of these methods.

5.3. *Transfer learning inspired community detection with deep transitive autoencoder results*

To evaluate our novel transfer-inspired deep autoencoder method, we compared Transfer-CDDTA with DNR and CDDTA, in terms of NMI accuracy with a varying proportion of source domain data.

We chose four LFR networks with different sizes (the first one was $N = 1000$, $C_{\min} = 10$, $C_{\max} = 50$, the second was $N = 1000$, $C_{\min} = 20$, $C_{\max} = 100$, the third was $N = 10000$, $C_{\min} = 10$, $C_{\max} = 50$, and the fourth was $N = 10000$, $C_{\min} = 20$, $C_{\max} = 100$). The trend in the experimental results from increasing the proportion of the data in the source domain is shown

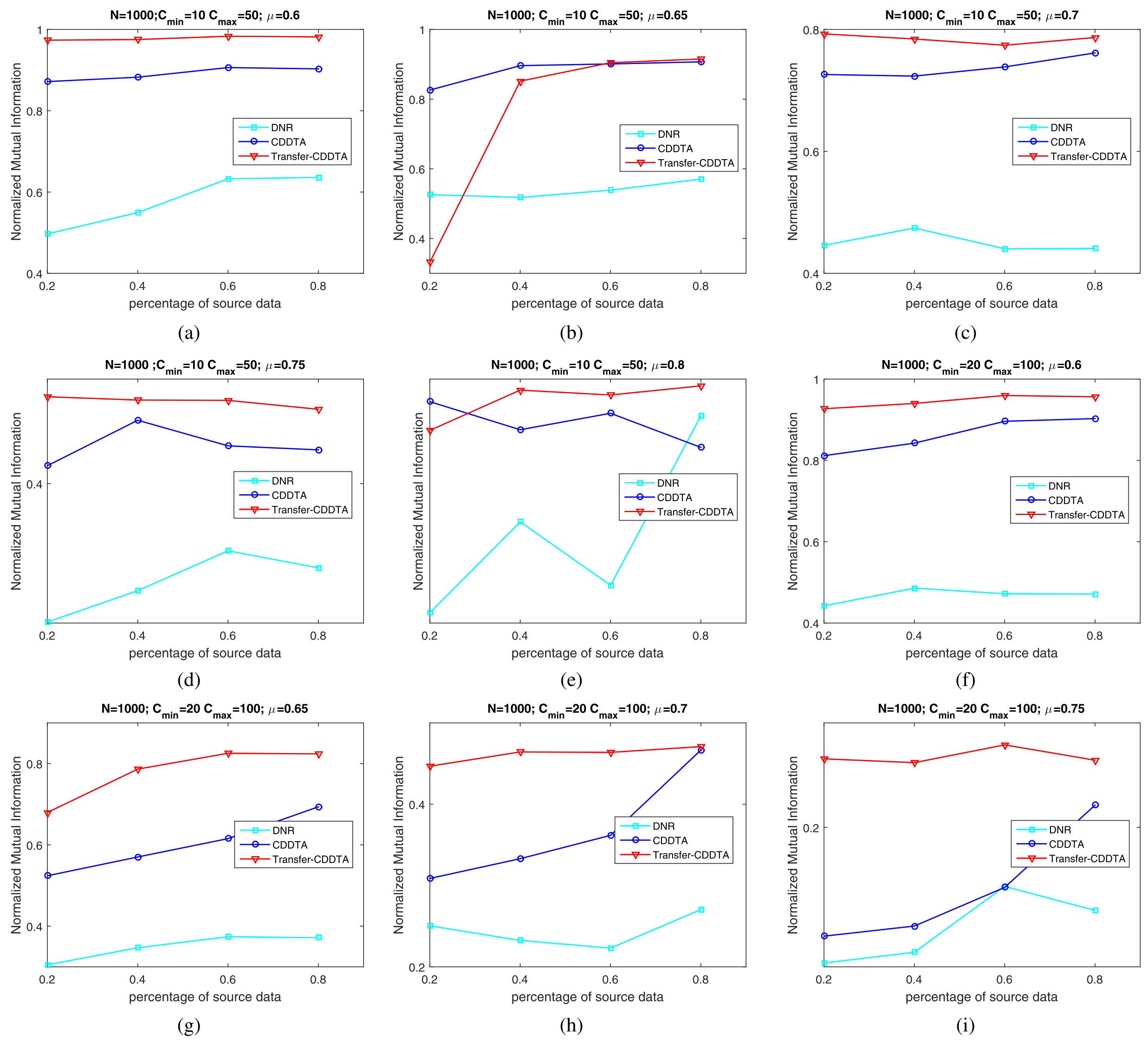
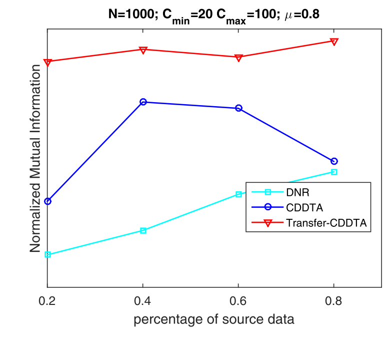
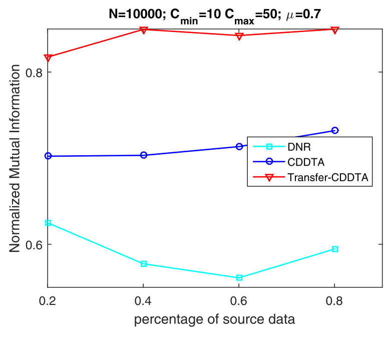


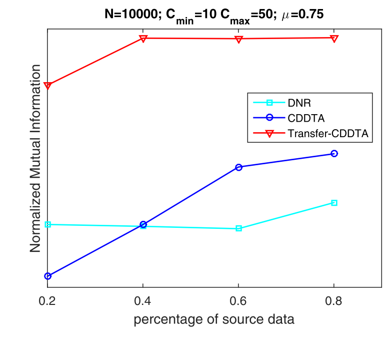
Fig. 6. The trend of NMI accuracy on the four sizes of LFR networks with increasing proportion of source domain data.



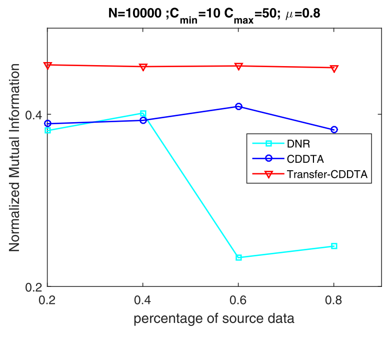
(j)



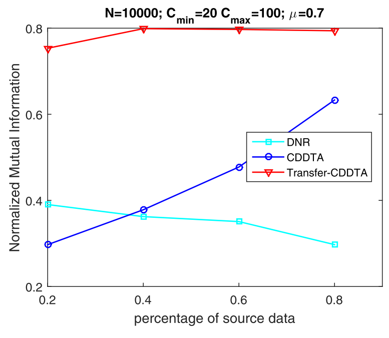
(k)



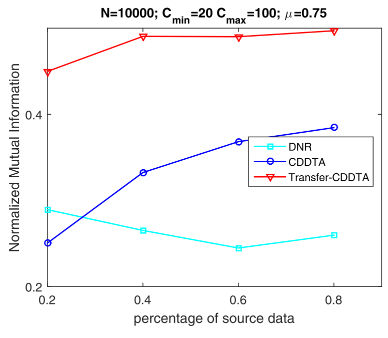
(l)



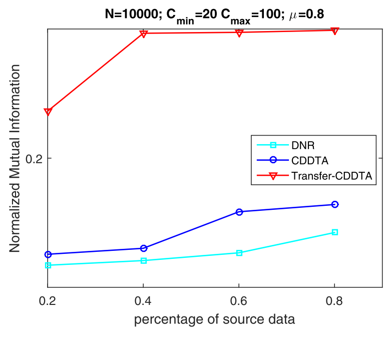
(m)



(n)



(o)



(p)

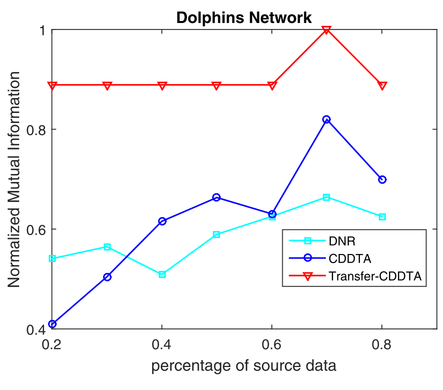
Fig. 6. Continued

in Fig. 6. The maximum NMI for both the proposed methods is very low as the mixing rate μ increases to greater than 0.7. This is because, with the increase of μ, the network structure becomes more obscure, and the detection rate of the algorithm becomes relatively low. Therefore, we conclude that the performance improves significantly with an increasing proportion of source domain data.

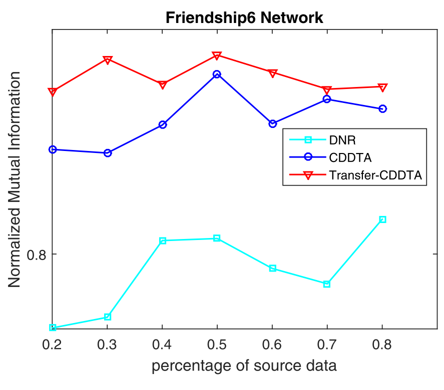
To sum up, Transfer-CDDTA was compared with two methods on real-world networks while varying the proportion of source domain data from 0.2 to 0.8. As shown in Fig. 7, the results show significant improvements on the eight real-world networks, which indicates that our Transfer-CDDTA is more effective on the same proportion of source domain data than other methods on these real networks.

6. Conclusion and future work

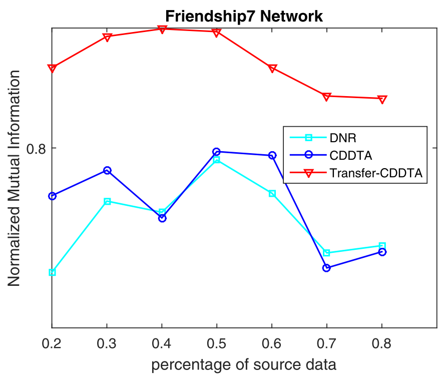
In this paper, to compensate for the defects of existing community detection methods, we proposed a novel CDDTA method for the feature representation of large complex networks. On this basis, we further incorporated unsupervised transfer learning into the proposed method, named Transfer-CDDTA, by measuring the KL divergence of embedded instances. Extensive experimental results show that our new methods are superior to most existing advanced methods available, on real-world networks and artificial benchmark networks used by the network community. In the future, we plan to use the potential ability of Transfer-CDDTA in the model selection task to discover how to make our method more robust than existing methods. We will study the accuracy of community detection techniques on incomplete networks generated by sampling



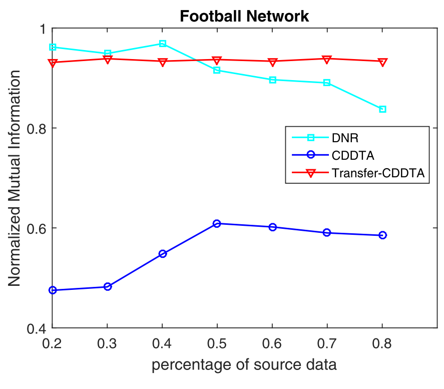
(a)



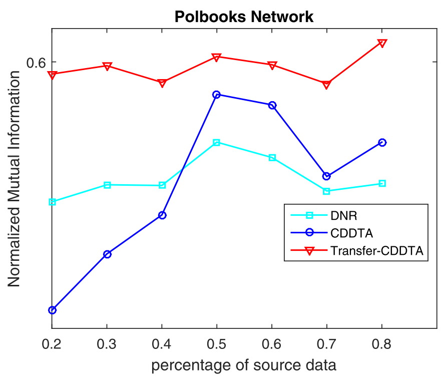
(b)



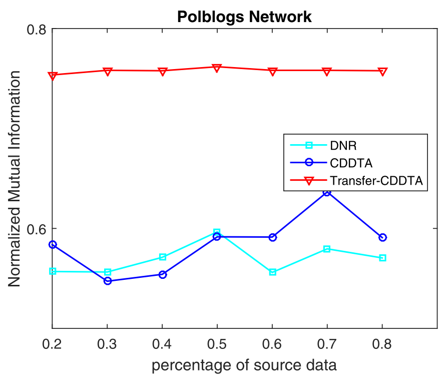
(c)



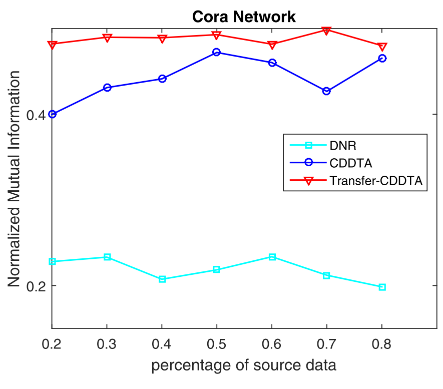
(d)



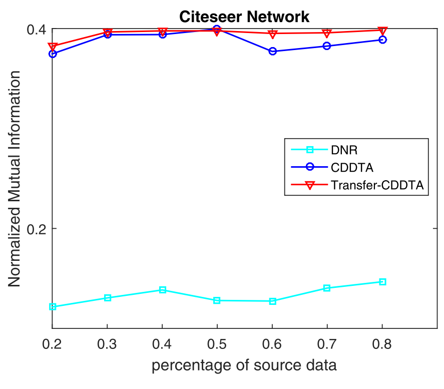
(e)



(f)



(g)



(h)

Fig. 7. Comparison of Transfer-CDDTA with CDDTA and DNR on eight real-world networks.

Table 3
Comparison of modularity Q for the network partitions found by CDDTA and seven other baseline algorithms.

Dataset	FEC	SP	Infomap	FAM	FUA	MABA	DNR	CDDTA
Karate	0.327	0.332	0.340	0.362	0.370	0.371	0.377	0.424
Dolphins	0.418	0.420	0.421	0.423	0.452	0.457	0.468	0.493
Friendship6	0.518	0.520	0.523	0.543	0.544	0.572	0.574	0.590
Friendship7	0.512	0.514	0.528	0.531	0.544	0.545	0.571	0.578
Football	0.513	0.523	0.536	0.541	0.546	0.554	0.566	0.597
Polbooks	0.411	0.423	0.427	0.466	0.484	0.493	0.501	0.518
Polblogs	0.318	0.320	0.323	0.327	0.353	0.393	0.394	0.426
Cora	0.621	0.642	0.704	0.707	0.709	0.714	0.721	0.732
Citeseer	0.620	0.634	0.640	0.641	0.643	0.652	0.682	0.701

processes. Because the algorithm in this paper is mainly aimed at undirected and unweighted networks, we plan to improve the algorithm according to the differences in network structure, so that it can have universal applicability to networks with different structures.

Conflict of interest

There are no conflicts of interest.

Acknowledgments

The research is supported by the National Natural Science Foundation of China (No. 61602005), Natural Science Foundation of Anhui Province (1608085MF130, 1808085MF199), The Natural Science Foundation of the Anhui Higher Education Institutions of China (KJ2018A0016).

References

[1] L.A. Adamic, N.S. Glance, The political blogosphere and the 2004 U.S. election: divided they blog, in: Proceedings of the 3rd international workshop on Link discovery, LinkKDD 2005, Chicago, Illinois, USA, August 21–25, 2005, 2005, pp. 36–43.
[2] C. Akimushkin, D.R. Amancio, O.N. Oliveira Jr, Text authorship identified using the dynamics of word co-occurrence networks, PLoS One 12 (1) (2017) e0170527.
[3] D.R. Amancio, O.N. Oliveira Jr, L. da F Costa, Robustness of community structure to node removal, J. Stat. Mech: Theory Exp. 2015 (3) (2015) P03003.
[4] J. Cao, D. Jin, L. Yang, J. Dang, Incorporating network structure with node contents for community detection on large networks using deep learning, Neurocomputing 297 (2018) 71–81.
[5] H.M. Fayek, M. Lech, L. Cavedon, Evaluating deep learning architectures for speech emotion recognition, Neural Netw. 92 (2017) 60–68.
[6] S. Fortunato, Community structure in complex networks, in: Extraction et Gestion des Connaissances, EGC 2018, Paris, France, January 23–26, 2018, 2018, pp. 5–6.
[7] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. U.S.A. 99 (12) (2002) 7821–7826.
[8] D. He, J. Liu, B. Yang, Y. Huang, D. Liu, D. Jin, An ant-based algorithm with local optimization for community detection in large-scale networks, Adv. Complex Syst. 15 (08) (2012) 1250036.
[9] M. Jebabli, H. Cherifi, C. Cherifi, A. Hamouda, Community detection algorithm evaluation with ground-truth data, Physica A 492 (2017) 651–706.
[10] R. Kanawati, Ensemble selection for community detection in complex networks, in: Social Computing and Social Media - 7th International Conference, SCSM 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2–7, 2015, Proceedings, 2015, pp. 138–147.
[11] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E Stat. Nonlin. Soft Matter Phys. 78 (2) (2008) 046110.
[12] H.J. Li, Z. Bu, A. Li, Z. Liu, Y. Shi, Fast and accurate mining the community structure: integrating center locating and membership optimization, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2349–2362.
[13] D. Liu, H.-Y. Bai, H.-J. Li, W.-J. Wang, Semi-supervised community detection using label propagation, Int. J. Mod. Phys. B 28 (29) (2014) 1450208.
[14] H. Long, Overlapping community detection with least replicas in complex networks, Inf. Sci. (Ny) 453 (2018) 216–226.
[15] D. Lusseau, M.E.J. Newman, Identifying the role that animals play in their social networks, Proc. R. Soc. London B: Biol. Sci. 271 (Suppl 6) (2004) S477–S481.
[16] X. Ma, L. Gao, X. Yong, L. Fu, Semi-supervised clustering algorithm for community structure detection in complex networks, Phys. A Stat. Mech. Appl. 389 (1) (2010) 187–197.
[17] R. Mall, R. Langone, J.A. Suykens, Self-tuned kernel spectral clustering for large scale networks, in: 2013 IEEE International Conference on Big Data, IEEE, 2013, pp. 385–393.
[18] T. Martín-Wanton, J. Gonzalo, E. Amigó, An unsupervised transfer learning approach to discover topics for online reputation management, in: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27, - November 1, 2013, 2013, pp. 1565–1568.
[19] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. U.S.A. 103 (23) (2006) 8577–8582.
[20] S.J. Pan, J.T. Kwok, Q. Yang, Transfer learning via dimensionality reduction, in: AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, Usa, July, 2008, pp. 677–682.
[21] M.Q. Pasta, F. Zaidi, Topology of complex networks and performance limitations of community detection algorithms, IEEE Access 5 (2017) 10901–10914.
[22] I. Psorakis, S. Roberts, M. Ebden, B. Sheldon, Overlapping community detection using Bayesian non-negative matrix factorization, Phys. Rev. E 83 (2011) 066114.
[23] Z. Qu, J. Yang, X. Wang, S. Yin, Combining link and content for community detection in social networks, in: 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018, Shanghai, China, January 15–17, 2018, 2018, pp. 607–610.
[24] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 76 (2) (2007) 036106.

[25] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. U.S.A. 105 (4) (2008) 1118–1123.
[26] M. Shao, D. Kit, Y. Fu, Generalized transfer subspace learning through low-rank constraint, Int. J. Comput. Vis. 109 (1–2) (2014) 74–93.
[27] K. Shin, T. Eliassi-Rad, C. Faloutsos, Patterns and anomalies in k-cores of real-world graphs with applications, Knowl. Inf. Syst. 54 (3) (2018) 677–710.
[28] J. Xie, S. Kelley, B.K. Szymanski, Overlapping community detection in networks:the state-of-the-art and comparative study, ACM Comput. Surv. 45 (4) (2013) 1–35.
[29] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[30] L. Yang, X. Cao, D. He, C. Wang, X. Wang, W. Zhang, Modularity based community detection with deep learning, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, 2016, pp. 2252–2258.
[31] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[32] H. Zhang, C.D. Wang, J.H. Lai, P.S. Yu, Modularity in complex multilayer networks with multiple aspects: a static perspective, Appl. Inf. 4 (1) (2017) 7.
[33] X.L. Zhang, Multilayer bootstrap networks, Neural Netw. 103 (2018) 29–43.