Modularity Based Community Detection with Deep Learning

Liang Yang,1,2,3 Xiaochun Cao,1 Dongxiao He,3,* Chuan Wang,1 Xiao Wang,4 Weixiong Zhang5,6
1State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences
2School of Information Engineering, Tianjin University of Commerce
3School of Computer Science and Technology,
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin University
4Department of Computer Science and Technology, Tsinghua University
5College of Math and Computer Science, Institute for Systems Biology, Jianghan University
6Department of Computer Science and Engineering, Washington University in St. Louis
{yangliang,caoxiaochun,wangchuan}@iie.ac.cn, hedongxiao@tju.edu.cn,
wxiaotju@gmail.com, weixiong.zhang@wustl.edu

Abstract

Identification of module or community structures is important for characterizing and understanding complex systems. While designed with different objectives, i.e., stochastic models for regeneration and modularity maximization models for discrimination, both these two types of model look for low-rank embedding to best represent and reconstruct network topology. However, the mapping through such embedding is linear, whereas real networks have various nonlinear features, making these models less effective in practice. Inspired by the strong representation power of deep neural networks, we propose a novel nonlinear reconstruction method by adopting deep neural networks for representation. We then extend the method to a semi-supervised community detection algorithm by incorporating pairwise constraints among graph nodes. Extensive experimental results on synthetic and real networks show that the new methods are effective, outperforming most state-of-the-art methods for community detection.

1 Introduction

Real-world systems often appear in networks, e.g., social networks in Facebook media, protein interaction networks, power grids and the Internet. Real-world networks often consist of functional units, which manifest in the form of network modules or communities, subnetworks with nodes more tightly connected with respect to the rest of the networks. Finding network communities is, therefore, critical for characterizing the organizational structures and understanding complex systems.

A great deal of effort has been devoted to developing network community finding methods, among which, two are widely adopted and thus worth mentioning. The stochastic model [Psorakis et al., 2011] focuses on deriving generative models of networks. Such a generative model in essence maps a network to an embedding in a low-dimensional latent space. The mapping can be done by, e.g., nonnegative matrix factorization (NMF) [Wang et al., 2008]. Unlike the stochastic model, the modularity maximization model [Newman, 2006], as the name suggests, attempts to maximize a modularity function on network substructures. The optimization can be done by eigenvalue decomposition (EVD), which is equivalent to reconstructing a low-rank modularity matrix.

In short, while appeared in different forms, the stochastic model and modularity maximization model share an essential commonality, i.e., mapping a network to a low-dimensional, latent space embedding. Motivated by the strong discrimination power of the modularity maximization model and the relationship between maximizing modularity and reconstructing modularity matrix by finding low-dimensional embedding, we aim to seek a more effective reconstruction algorithm for modularity optimization. However, all these types of embedding adopted in the two popular models are linear, e.g., NMF in the stochastic model and EVD in the modularity maximization model. This is in sharp contrast to the fact that real-world networks are full of nonlinear properties, e.g., a relationship (e.g., distance) among nodes may not necessarily be linear. As a result, the representation power of these linear mapping based models is limited on real-world networks.

Neural networks, particularly that with deep structures, are well known to provide nonlinear low-dimensional representations [Bourlard and Kamp, 1988]. They have been successfully applied to complex problems and systems in practice, such as image classification, speech recognition, and playing an ancient strategic board game of Go. To the best of our knowledge, however, deep neural networks have not yet been successfully applied to community detection.

Taking advantage of the nonlinear representation power of deep neural networks, we propose in this paper a nonlinear reconstruction (DNR) algorithm for community detection using deep neural networks.
It is known that information beyond network topology can greatly aid network community identification [Zhang, 2013; Yang et al., 2015]. Examples of such information include semantics on nodes, e.g., names and labels, and constraints on relationships among nodes, e.g., community membership constraints between adjacent nodes (pairwise constraints). In the current study, we extend our DNR method to a semi-supervised DNR (semi-DNR) algorithm to explicitly incorporates pairwise constraints among nodes to further improve community detection.

2 Reconstruction based Community Detection

We consider an undirected and unweighted graph \( G = (V, E) \), where \( V = \{v_1, v_2, ..., v_N\} \) is the set of \( N \) vertices, and \( E = \{e_{ij}\} \) the set of \( E \) edges of which connects two vertices in \( V \). The adjacency matrix of \( G \) is a nonnegative symmetric matrix \( \mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N} \) where \( a_{ij} = 1 \) if there is an edge between vertices \( i \) and \( j \), or \( a_{ij} = 0 \) otherwise, and \( a_{ii} = 0 \) for all \( 1 \leq i \leq N \). The degree of vertex \( i \) is defined as \( k_i = \sum_j a_{ij} \). The problem of community detection is to find \( K \) modules or communities \( \{V_i\}_{i=1}^K \) that are subgraphs whose vertices are more tightly connected with one another than with outside vertices. Here, we consider disjoint communities, i.e., \( V_i \cap V_j = \emptyset \) for \( i \neq j \).

2.1 Stochastic Model

In stochastic model [Psorakis et al., 2011; He et al., 2015; Jin et al., 2015], \( a_{ij} \) can be viewed as the probability that vertices \( i \) and \( j \) are connected. This probability can be further considered to be determined by the probabilities that vertices \( i \) and \( j \) generate edges belonging to the same community. We introduce latent variables \( \mathbf{H} = [h_{ik}] \in \mathbb{R}^{N \times K} \) with \( h_{ik} \) representing the probability that node \( i \) generates an edge belonging to community \( k \). This latent variable also captures the probability that node \( i \) belongs to community \( k \), and each row of \( \mathbf{H} \) can be considered as a community membership distribution of a vertex. The probability that vertices \( i \) and \( j \) is connected by a link belonging to community \( k \) is then \( h_{ik}h_{jk} \), and the probability that they are connected is:

\[
\hat{a}_{ij} = \sum_{k=1}^K h_{ik}h_{jk}.
\]

As a result, the community detection problem can be formulated as a nonnegative matrix factorization \( \mathbf{A} \approx \mathbf{HH}^T \). The NMF-based community detection approaches [Psorakis et al., 2011] aim to find a nonnegative membership matrix \( \mathbf{H} \) to reconstruct adjacency matrix \( \mathbf{A} \). There are two common objective (loss) functions to quantify the reconstruction error. The first is based on the square loss function [Wang et al., 2008; Zhang et al., 2007] which is equivalent to the square of the Frobenius norm of the difference between two matrices

\[
\mathcal{L}_{LSE}(\mathbf{A}, \mathbf{HH}^T) = \|\mathbf{A} - \mathbf{HH}^T\|_F^2.
\]

The second is based on the Kullback-Leibler divergence (KL-divergence) between two matrices

\[
\mathcal{L}_{KL}(\mathbf{A}, \mathbf{HH}^T) = KL(\mathbf{A}||\mathbf{HH}^T).
\]

The index of the largest element in the \( i^{th} \) row of \( \mathbf{H} \) indicates the community that node \( i \) belongs to.

There are many variations to the stochastic model, such as nonnegative matrix tri-factorization and stochastic block model. Nearly all of these models can be intuitively viewed as finding new representations in a low-dimensional space that can best represent and reconstruct the adjacency matrix.

2.2 Modularity Maximization Model

This model was introduced by Newman [Newman, 2006] to maximize a modularity function \( Q \), which is defined as the difference between the number of edges within communities and the expected number of such edges over all pairs of vertices. For example, consider a network with two communities, then

\[
Q = \frac{1}{4m} \sum_{ij} (a_{ij} - \frac{k_ik_j}{2m})(h_i h_j),
\]

where \( h_i \) equals to 1 (or -1) if vertex \( i \) belongs to the first (or second) group, \( \frac{k_ik_j}{2m} \) is the expected number of edges between vertices \( i \) and \( j \) if edges are placed randomly, \( k_i \) is the degree of vertex \( i \) and \( m = \frac{1}{2} \sum_k k_i \) is the total number of edges in the network. By defining modularity matrix \( \mathbf{B} = [b_{ij}] \in \mathbb{R}^{N \times N} \) whose element is \( b_{ij} = a_{ij} - \frac{k_ik_j}{2m} \), modularity \( Q \) can be written as

\[
Q = \frac{1}{4m} \mathbf{h}^T \mathbf{B} \mathbf{h},
\]

where \( \mathbf{h} = [h_i] \in \mathbb{R}^N \) is a community membership indicator vector. Maximizing Eq. (1) is NP-hard, for which many optimization algorithms have been proposed, such as extermal optimization [Duch and Arenas, 2005]. In practice, we can relax the problem by allowing variable \( h_i \) to take any real value and \( \mathbf{h}^T \mathbf{h} = N \). To generalize Eq. (1) to \( K > 2 \) communities, we can define an indicator matrix \( \mathbf{H} = [h_{ik}] \in \mathbb{R}^{N \times K} \) and obtain

\[
\mathcal{L}_{MOD}(\mathbf{H}, \mathbf{B}) = Q = \text{Tr}(\mathbf{H}^T \mathbf{BH}),
\]

s.t. \( \text{Tr}(\mathbf{H}^T \mathbf{H}) = N \),

where \( \text{Tr}(\cdot) \) is the trace of a matrix. Based on Rayleigh Quotient, the solution to this problem is the largest \( K \) eigenvectors of the modularity matrix \( \mathbf{B} \). Each row of matrix \( \mathbf{H} \) can be regarded as a new representation of the corresponding vertex in the latent space, and clustering algorithms, such as \( \ell \)-means, can be used to classify the nodes into disjoint groups in the latent space.

The Eckart-Young-Mirsky Theorem [Eckart and Young, 1936] explores the relationship between the reconstruction and singular value decomposition (SVD).

**Theorem 1.** [Eckart and Young, 1936] For a matrix \( \mathbf{D} \in \mathbb{R}^{m \times n}(m \geq n) \), if \( \mathbf{D} = \mathbf{U} \Sigma \mathbf{V}^T \) is the singular value decomposition of \( \mathbf{D} \), and \( \mathbf{U}, \mathbf{V}, \Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_m) \) where \( \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \) are as follows:

\[
\mathbf{U} = [\mathbf{U}_1 \mathbf{U}_2], \quad \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, \quad \mathbf{V} = [\mathbf{V}_1 \mathbf{V}_2],
\]

where \( \Sigma_1 \) is \( r \times r \), \( \mathbf{U}_1 \) is \( m \times r \) and \( \mathbf{V}_1 \) is \( n \times r \), then the optimal solution to the following problem

\[
\arg\min_{\mathbf{D} \in \mathbb{R}^{m \times n} \text{rank}(\mathbf{D}) \leq r} \|\mathbf{D} - \tilde{\mathbf{D}}\|_F
\]
\[ \hat{\mathbf{D}} = \mathbf{U}_1 \Sigma_1 \mathbf{V}_1^T \text{ and} \]
\[
\| \mathbf{D} - \hat{\mathbf{D}} \|_F = \sqrt{\sigma^2_{r+1} + \ldots + \sigma^2_m}.
\]

The above theorem means that the matrix reconstruction from the singular vectors corresponding to the \(K\) largest singular values is the best rank-\(K\) approximation to the original matrix under the Frobenius norm. Since modularity matrix \(\mathbf{B}\) is symmetric, there exists orthogonal decomposition \(\mathbf{B} = \mathbf{U} \Lambda \mathbf{U}^T\), where \(\Lambda\) is diagonal matrix with the eigenvalues of \(\mathbf{B}\) as the diagonal elements and \(\mathbf{U}^T \mathbf{U} = \mathbf{I}\). Thus, the matrix reconstructed by the eigenvectors corresponding to the \(K\) largest eigenvalues is also the best approximation to the input matrix \(\hat{\mathbf{B}}\) with rank \(K\). Therefore, the modularity maximization problem can be regarded as reconstructing the modularity matrix using a low-rank approximation.

Put together, stochastic model and modularity maximization can be intuitively interpreted as to find low-dimensional representations to best reconstruct given network structures. It is important to note that both of them only reconstruct original networks by linear reconstruction, using, e.g., NMF or SVD, and ignore nonlinear properties of the networks. It is unclear how NMF and SVD based approaches can be extended to accommodate nonlinear low-dimensional embedding. We aim to overcome this difficulty by using deep neural networks, the focus of the current paper.

3 Deep Nonlinear Reconstruction Model

We now present a novel deep nonlinear reconstruction (DNR) model for community detection. We first introduce an Auto-Encoder, which is a key building block of the model, and then describe a stacked Auto-Encoder. While in the following discussion we focus on finding a nonlinear embedding that best reconstructs the modularity matrix \(\mathbf{B}\), as inspired by the SVD-based modularity maximization (Section 2.2), our method can be readily applied to other network input forms, such as adjacency and Laplacian matrices.

3.1 Reconstruction based on Auto-Encoder

Auto-Encoder is a special neural network that is used to learn a new representation that can best approximate the original data [Bourlard and Kamp, 1988; Hinton and Zemel, 1994]. We adopt modularity matrix \(\mathbf{B} = [b_{ij}] \in \mathbb{R}^{N \times N}\) as the input to the Auto-Encoder. Here, the elements of \(\mathbf{B}\) are \(b_{ij} = \alpha_{ij} - \frac{k_i k_j}{2m}\), and the \(j^{th}\) column \(\mathbf{b}_j\) of \(\mathbf{B}\) represents vertex \(i\). The Auto-Encoder consists of two key components: encoder and decoder. The encoder maps the original data \(\mathbf{X}\) to a low-dimensional embedding \(\mathbf{H} = [h_{ij}] \in \mathbb{R}^{d \times N}\) where \(d < N\) and the \(j^{th}\) column of \(\mathbf{H}\), i.e., \(\mathbf{h}_j\), represents vertex \(i\) in the latent space

\[ h_i = f(b_i) = s(\mathbf{W}_H \mathbf{b}_i + \mathbf{d}_H), \tag{2} \]

where \(\mathbf{W}_H \in \mathbb{R}^{d \times N}, \mathbf{d}_H \in \mathbb{R}^{d \times 1}\) are the parameters to be learned in the encoder, and \(s(\cdot)\) is an element-wise nonlinear mapping, such as sigmoid function \(s_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}}\) or tanh function \(s_{\text{tanh}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}\). The decoder maps the latent representation \(\mathbf{H}\) back into the original data space, i.e., reconstructs the original data from the latent representation:

\[ m_i = g(h_i) = s(\mathbf{W}_M h_i + \mathbf{d}_M), \]

where \(\mathbf{W}_M \in \mathbb{R}^{N \times d}, \mathbf{d}_M \in \mathbb{R}^{N \times 1}\) are the parameters to be learned in the decoder and \(g(\cdot)\) is another element-wise nonlinear mapping similar to \(s(\cdot)\). Auto-Encoder aims at learning a low-dimensional nonlinear representation \(\mathbf{H}\) that can best reconstruct the original data \(\mathbf{B}\), i.e. minimize the difference between the original data \(\mathbf{B}\) and reconstruction data \(\mathbf{M}\) under parameters \(\theta = \{\mathbf{W}_H, \mathbf{d}_H, \mathbf{W}_M, \mathbf{d}_M\}\)

\[ \hat{\theta} = \arg\min_{\theta} L_\theta(\mathbf{B}, \mathbf{M}) = \arg\min_{\theta} \sum_{i=1}^{N} L_\theta(\mathbf{b}_i, m_i), \tag{3} \]

where \(L_\theta(\mathbf{b}_i, m_i)\) is a distance function that measures the reconstruction error. Here we adopt the Euclidean distance and sigmoid cross-entropy distance as distance functions. The sigmoid cross-entropy distance maps elements in \(\mathbf{b}_i = [b_{ij}] \in \mathbb{R}^{N \times 1}\) and \(\mathbf{m}_i = [m_{ij}] \in \mathbb{R}^{N \times 1}\) to \([0, 1]\) using sigmoid function \(\sigma(x) = \frac{1}{1 + e^{-x}}\), and then computes the cross-entropy of them as

\[ \sum_{j=1}^{N} (\sigma(m_{ji}) \log(\sigma(b_{ji})) + (1 - \sigma(m_{ji})) \log(1 - \sigma(b_{ji}))). \]

After training the Auto-Encoder, \(\mathbf{W}_H\) and \(\mathbf{d}_H\) are obtained and Eq. (2) can be used to generate the new representations for all vertices.

3.2 Optimization

Eq. (3) can be solved by back-propagation with stochastic gradient descent. In each iteration, the parameters \(\theta = \{\mathbf{W}_H, \mathbf{d}_H, \mathbf{W}_M, \mathbf{d}_M\}\) are updated as follows

\[ \frac{\partial}{\partial \mathbf{W}_{ji}^{\alpha}} L_\theta(\mathbf{X}, g(f(\mathbf{X}))) = \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{W}_{ji}^{\alpha}} L_\theta(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{z}_i^{\alpha}} L_\theta(\mathbf{x}_i, g(f(\mathbf{x}_i))) \]

where \(\mathbf{z}_i = \mathbf{W}_\alpha \mathbf{x} + \mathbf{d}_\alpha\), we have

\[ \frac{\partial}{\partial \mathbf{z}_i^{\alpha}} L_\theta(\mathbf{X}, g(f(\mathbf{X}))) = \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{z}_i^{\alpha}} L_\theta(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \sum_{i=1}^{N} \delta_i^{\alpha} \mathbf{x}_i, \]

\[ \frac{\partial}{\partial \mathbf{d}_{ji}^{\alpha}} L_\theta(\mathbf{X}, g(f(\mathbf{X}))) = \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{d}_{ji}^{\alpha}} L_\theta(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \sum_{i=1}^{N} \delta_i^{\alpha} \mathbf{x}_i. \]
where \( \delta^i_M \) denotes the contribution of a node to the overall reconstruction error. For Euclidean distance based \( L_\theta(B, M) \),

\[
\delta^i_M = -\sum_{i=1}^{N} (b_{ij} - m_{ij}) s'(z^i_M), \quad \delta^j_H = (\sum_{i=1}^{N} W^i_H \delta^i_M) s'(z^j_H),
\]

where \( s'(x) \) is the derivative of \( s(x) \).

### 3.3 Stacked Auto-Encoder

Recently, deep learning has been successful on various problems in many fields, such as image classification, semantic segmentation [Liu et al., 2015; Liang et al., 2015]. However, as the number of layers increases, the space of parameters grows exponentially, making optimization inefficient. A compromising strategy is to train the network layer by layer [Vincent et al., 2010].

To take advantage of a deep architecture, we stack a series of Auto-Encoders to form a DNR model. For a deep Auto-Encoder network, we train the first Auto-Encoder by reconstructing the original data, i.e., the modularity matrix \( B \) and obtain a new representation \( H^1 \in \mathbb{R}^{N \times t_1} \). We then train the \( i \)th Auto-Encoder by reconstructing the output of the \((i-1)\)th Auto-Encoder and obtain a representation \( H^i \in \mathbb{R}^{N \times t_i} \), where \( t_0 < t_1 < \ldots \). The number of Auto-Encoders we stack and the dimensions of new representations, i.e., \( t_i, s_i \), are discussed in Section 5.2.

### 4 Pairwise Constrained Semi-supervised Community Detection

We now incorporate pairwise constraints on vertices into the proposed DNR model and introduce a novel reconstruction space graph regularizer for semi-supervised DNR (semi-DNR). If we have a priori knowledge that vertices \( i \) and \( j \) belong to the same community, we can use this knowledge in two different ways. First, to classify two nodes into the same community, the new representations of nodes \( i \) and \( j \), i.e., \( h_i \) and \( h_j \), should be similar, since the new representations are used to cluster the two vertices after encoding. Second, this a priori information should also be encoded into the DNR model to further affect the embedding of other vertices. Therefore, instead of just modifying the embedding representation, we incorporate the pairwise constraints into the loss function of the Auto-Encoder in Eq. (3).

To measure the similarity of their latent representations, we can adopt either Euclidean distance or Kullback-Leibler divergence (KL-divergence), i.e.,

\[
D_{LSE}(h_i, h_j) = \|h_i - h_j\|_2^2 = \sum_{k=1}^{K} (h_{ik} - h_{jk})^2,
\]

\[
D_{KL}(h_i||h_j) = \sum_{k=1}^{K} (h_{ik} \log \left( \frac{h_{ik}}{h_{jk}} \right) - h_{ik} + h_{jk}).
\]

If nodes \( i \) and \( j \) are known to belong to the same community, we then try to minimize the difference between their new representations \( D_{LSE}(h_i, h_j) \). We define a pairwise constraint matrix \( O = [o_{ij}] \in \mathbb{R}^{N \times N} \), where \( o_{ij} = 1 \) if nodes \( i \) and \( j \) are known to be in the same community, or \( o_{ij} = 0 \) otherwise. Thus, we can write the pairwise constraints as

\[
R_{LSE}(O, H) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} o_{ij} \|h_i - h_j\|_2^2 = \text{Tr}(H^T D H) - \text{Tr}(H^T O H) = \text{Tr}(H^T L H),
\]

where \( \text{Tr}(\cdot) \) is the trace of a matrix, \( D = [d_{ij}] \in \mathbb{R}^{N \times N} \) a diagonal matrix whose entries are row summation of \( O \), i.e., \( d_{ii} = \sum_{j=1}^{N} o_{ij} \) and \( L = D - O \) the graph regularization matrix (Laplacian matrix) of a priori information \( O \). Similarly, the KL-divergence based constraints can be written as:

\[
R_{KL}(O, H) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} o_{ij} (D_{KL}(h_i||h_j) + D_{KL}(h_j||h_i)),
\]

which takes into account the asymmetry of KL-divergence and averages \( D_{KL}(h_i||h_j) \) and \( D_{KL}(h_j||h_i) \). By minimizing \( R_{LSE}(O, H) \) or \( R_{KL}(O, H) \), we expect the new representations of two nodes \( i \) and \( j \) are similar if we have some information indicating that these two nodes belong to the same community, i.e., the corresponding element \( o_{ij} = 1 \).

By incorporating the pairwise constraints in Eq (4) with the reconstruction error function in Eq (3), we obtain the overall loss function for semi-supervised DNR (semi-DNR) as

\[
\hat{\theta} = \argmin_{\theta} L_\theta(B, M) + \lambda \text{Tr}(H^T L H),
\]

where \( \lambda \) is a parameter for making a tradeoff between the reconstruction error (the first term \( L(B, M) \)) and the consistency of the new representations with a priori information (the second term \( \text{Tr}(H^T L H) \)). For semi-DNR, we can impose a graph regularization term on the reconstruction layers of all Auto-Encoders, and then evaluate the performance improvement as discussed in Section 5.3.

Eq. (4) can also be solved using back-propagation. Since the reconstruction space graph regularization term \( \text{Tr}(H^T L H) \) is independent of reconstruction of \( M \), it does not affect the update of \( W_M \) and \( b_M \). Therefore, we only need to modify \( \delta^i_H \) as \( \delta^i_{H(semi)} = \delta^i_H + \delta^i_{graph} \), where

\[
\delta_{graph,lse} = H(L + L^T) \circ s'(H),
\]

\[
\delta_{graph,kl} = H(L + L^T) \div H \circ s'(H),
\]

denote how much a node is responsible for the inconsistency of the derived representation and the pairwise constraints based on the Euclidean distance and KL-divergence, respectively. Here \( \circ \) and \( \div \) denote the element-wise multiplication and division, respectively.

### 5 Experimental Analysis

To evaluate the proposed DNR and semi-DNR methods, we analyzed their performance on widely used synthetic benchmarks and real-world networks. We first compared DNR with seven state-of-the-art approaches, which are divided into two categories based on whether they are based on modularity maximization. The modularity based methods include the spectral (SP) algorithm [Newman, 2006], the external optimization
we compared semi-DNR with two existing semi-supervised
community detection algorithms, ModLink [Zhang, 2013] and
GraphNMF [Yang et al., 2015]. The former transforms pair-
wise constraints into information of network topology, modi-
fies the adjacency matrix and uses a conventional algorithm to
find communities on new label-refined networks. The latter
incorporates the labels by constraining the nodes belonging to
the same community to have similar membership representa-
tions. The former transforms pairwise constraints into infor-
mation of network topology, modifies the adjacency matrix and
uses a conventional algorithm to find communities on new
label-refined networks. The latter incorporates the labels by
constraining the nodes belonging to the same community to have
similar membership representations. We adopted the Normalized
Mutual Information (NMI) for performance measure.

5.1 Experiment Setup

Table 2: Deep Nonlinear Reconstruction Network Setting

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$N$</th>
<th>Layers Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karate [Zachary, 1977]</td>
<td>34</td>
<td>34-32-16</td>
</tr>
<tr>
<td>Friendship6 [Xie et al., 2013]</td>
<td>68</td>
<td>68-32-16</td>
</tr>
<tr>
<td>Friendship7 [Xie et al., 2013]</td>
<td>68</td>
<td>68-32-16</td>
</tr>
<tr>
<td>Polbooks [Newman, 2006]</td>
<td>105</td>
<td>105-64-32-16</td>
</tr>
<tr>
<td>Polblogs [Adamic and Glance, 2005]</td>
<td>1,490</td>
<td>1,490-256-128-64</td>
</tr>
<tr>
<td>Cora [Yang et al., 2009]</td>
<td>2,708</td>
<td>2,708-512-256-128</td>
</tr>
<tr>
<td>LFR Network</td>
<td>1,000</td>
<td>1,000-512-256-128</td>
</tr>
</tbody>
</table>

Table 1: Performance on Real-world Networks (the best performance is in bold and the second best performance is in italics)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$N$</th>
<th>$M$</th>
<th>$K$</th>
<th>SP</th>
<th>EO</th>
<th>FN</th>
<th>FUA</th>
<th>DNR_L2</th>
<th>DNR_CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karate</td>
<td>34</td>
<td>78</td>
<td>2</td>
<td>1.000</td>
<td>0.587</td>
<td>0.692</td>
<td>0.587</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Dolphins</td>
<td>62</td>
<td>159</td>
<td>2</td>
<td>0.753</td>
<td>0.579</td>
<td>0.572</td>
<td>0.516</td>
<td>0.889</td>
<td>0.818</td>
</tr>
<tr>
<td>Friendship6</td>
<td>68</td>
<td>220</td>
<td>6</td>
<td>0.418</td>
<td>0.952</td>
<td>0.727</td>
<td>0.852</td>
<td>0.888</td>
<td>0.924</td>
</tr>
<tr>
<td>Friendship7</td>
<td>68</td>
<td>220</td>
<td>7</td>
<td>0.477</td>
<td>0.910</td>
<td>0.762</td>
<td>0.878</td>
<td>0.907</td>
<td>0.932</td>
</tr>
<tr>
<td>Football</td>
<td>115</td>
<td>613</td>
<td>12</td>
<td>0.334</td>
<td>0.885</td>
<td>0.698</td>
<td>0.890</td>
<td>0.927</td>
<td>0.914</td>
</tr>
<tr>
<td>Polbooks</td>
<td>105</td>
<td>441</td>
<td>3</td>
<td>0.561</td>
<td>0.557</td>
<td>0.531</td>
<td>0.574</td>
<td>0.552</td>
<td>0.582</td>
</tr>
<tr>
<td>Polblogs</td>
<td>1,490</td>
<td>16,718</td>
<td>2</td>
<td>0.511</td>
<td>0.501</td>
<td>0.499</td>
<td>0.375</td>
<td>0.389</td>
<td>0.517</td>
</tr>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,429</td>
<td>7</td>
<td>0.295</td>
<td>0.441</td>
<td>0.459</td>
<td>0.260</td>
<td>0.463</td>
<td>0.421</td>
</tr>
</tbody>
</table>

The layer configurations of the deep neural networks for
different problems tested are shown in Table 2. The networks
have at most 3 stacked Auto-Encoder, and the dimension of
each latent space is less than that of its input and output spaces.
For example, the stacked Auto-Encoder network for the LFR
network consists of three Auto-Encoders, where the first is
1,000-512-1,000, the second 512-256-512 and the third 256-128-256.
All Auto-Encoders were trained separately. We took a
modularity matrix as the input to the first Auto-Encoder, and
trained the it to minimize the reconstruction error, then
took the embedding result as the input to the second Auto-
Encoder, and so on. We set the training batch to the size of
the network and ran at most 100,000 iterations. For each network
we trained a DNR model with 10 random initializations, and
took the latent representations from three Auto-Encoders for
clustering. Here, we adopted the $k$-means for clustering and
returned the results with the maximum modularity.

5.2 Community Detection Results

We considered two types of synthetic networks, Girvan-
Newman (GN) networks [Girvan and Newman, 2002] and
Lancichinetti-Fortunato-Radicchi (LFR) networks [Lanci-
chinetti et al., 2008]. Each GN network consists of 128 nodes
divided into 4 communities of 32 nodes each. Each node
has on average 16 edges, among which $Z_{out}$ edges are inter-
community edges. The results are shown in Figure 1(a). As
shown, DNR outperforms all competing methods, especially
when $Z_{out} > 0$.

The LFR networks are more complicated than the GN net-
works. As suggested in [Lancichinetti et al., 2008], we set
the number of nodes to 1000, the average degree to 20, the
exponent of a vertex degree and the community size to -2
and -1 respectively, and varied the mixing parameter \( \mu \) from 0.6 to 0.8. To fully evaluate the performance on networks with different community sizes, we generated two groups of networks: one with community sizes from 10 to 50 while the other from 20 to 100. The results on these two groups of networks are shown in Figs 1(b) and 1(c). As shown, DNR can successfully detect small and large communities. It achieves the best performance when \( \mu > 0.65 \) on networks with small community size (middle figure, Figure 1) and \( \mu > 0.6 \) on networks with large community sizes (right figure, Figure 1). While Infomap performs slightly better than DNR when \( \mu \) is small, it completely fails when \( \mu \geq 0.75 \).

In summary, the experimental results on synthetic networks showed that the deep nonlinear model is more effective on difficult networks with vague community structures and is competitive on easy networks with clear community structures. While the factors for such a superb performance compared with other modularity-based methods remains to be further investigated, it may be partially attributed to the nonlinear structure of our new model, which helps to mitigate the resolution limit [Fortunato and Barthelemy, 2007] and the extreme degeneracy [Good et al., 2010] problems, both of which are often suffered from modularity optimization.

Nine widely-used real networks, listed in Table 1, were used for evaluation. As shown in previous work [Psorakis et al., 2011], no single unified loss function seemed to exist that can successfully detect communities in all networks. In our experiments, we also adopted the Euclidean distance and sigmoid cross-entropy as error functions. We used sigmoid cross-entropy instead of KL-divergence because the former can be integrated with the sigmoid function for back-propagation.

The results are shown in Table 1. Here, we compared DNR with other well-known modularity-based community detection methods. As shown in the table, DNR with the \( L_2 \) norm (DNR\_L2) and with the cross-entropy distance (DNR\_CE) outperforms most of the competing modularity-based optimization algorithms.

### 5.3 Semi-supervised Community Detection Results

To evaluate the new semi-supervised deep nonlinear reconstruction (semi-DNR) method, we used networks on which it is difficult to find satisfactory community structures without label information. As shown in Figure 2, the performance is mediocre on GN network with \( Z_{out} = 8 \) and LFR network with \( \mu = 0.75 \) and 0.8. Besides, we chose GN network with \( Z_{out} = 7 \) where the methods without labels can also achieve better results for comparison. Here, we set the balancing parameter \( \lambda = 1000 \). We also verified the effects of pairwise constraints on more than one layer i.e., only the bottom layer, both bottom and middle layers and all the three layers.

The results are shown in Figure 2. On the GN with \( Z_{out} = 7 \), all the methods have similar improvements by enforcing the same percent of labels. On the networks where unsupervised methods cannot obtain satisfactory results, in comparison, the semi-DNR achieves much better performance with the same number of labels. For example, with 20\% pairwise constraints, the NMI of semi-DNR achieves 0.95 while that of GraphNMF and ModLink only achieve 0.76 and 0.79, respectively, on LFR network with \( \mu = 0.8 \) (right figure in Figure 2). This means the semi-DNR is much more efficient on the use of pairwise constraints than other methods. Furthermore, the performance of enforcing pairwise constraints on multi-layers has similar improvements. It illustrates that semi-DNR can fully explore pairwise constraints by only one layer graph regularization.

### 6 Conclusion

In order to overcome the serious drawback of linear low-rank embedding used by the widely adopted stochastic model and modularity maximization model for network community identification, we proposed a nonlinear model in deep neural networks to gain representation power for large complex networks; developed an algorithm using the model for network community detection; and further extended this method to a semi-supervised deep nonlinear reconstruction algorithm by incorporating pairwise constraints. Extensive experimental results on synthetic and real networks illustrate that our new methods outperform the existing state-of-the-art methods for network community identification. In the future, we plan to study model selection using the latent space embedding from DNR to make the model and methods more robust.

### 7 Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61503281, 61502334, 61303110, 61422213), "Strategic Priority Research Program" of the Chinese Academy of Sciences (XDA06010701), Open Funding Project of Tianjin Key Laboratory of Cognitive Computing and Application, Foundation for the Young Scholars by Tianjin University of Commerce (150113), the Talent Development Program of Wuhan, the municipal government of Wuhan, Hubei, China (2014070504020241), and an internal research
References


