Incorporating Network Embedding into Markov Random Field for Better Community Detection

Di Jin,[1] Xinxin You,[1] Weihao Li,[2] Dongxiao He,[1] Peng Cui,[3,*] Françoise Fogelman-Soulié,[1] Tanmoy Chakraborty[4]

[1]College of Intelligence and Computing, Tianjin University, Tianjin 300350, China,
[2]Visual Learning Lab, Heidelberg University, Heidelberg 69120, Germany,
[3]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China,
[4]Indraprastha Institute of Information Technology Delhi, New Delhi 110020, India.
{jindi, xinxyou, hedongxiao}@tju.edu.cn , francoise.soulie@outlook.com
weihao.li@iwr.uni-heidelberg.de, cuip@tsinghua.edu.cn, tanmoy@iiitd.ac.in

Abstract

Recent research on community detection focuses on learning representations of nodes using different network embedding methods, and then feeding them as normal features to clustering algorithms. However, we find that though one may have good results by direct clustering based on such network embedding features, there is ample room for improvement. More seriously, in many real networks, some statistically-significant nodes which play pivotal roles are often divided into incorrect communities using network embedding methods. This is because while some distance measures are used to capture the spatial relationship between nodes by embedding, the nodes after mapping to feature vectors are essentially not coupled any more, losing important structural information. To address this problem, we propose a general Markov Random Field (MRF) framework to incorporate coupling in network embedding which allows better detecting network communities. By smartly utilizing properties of MRF, the new framework not only preserves the advantages of network embedding (e.g. low complexity, high parallelizability and applicability for traditional machine learning), but also alleviates its core drawback of inadequate representations of dependencies via making up the missing coupling relationships. Experiments on real networks show that our new approach improves the accuracy of existing embedding methods (e.g. Node2Vec, DeepWalk and MNMF), and corrects most wrongly-divided statistically-significant nodes, which makes network embedding essentially suitable for real community detection applications. The new approach also outperforms other state-of-the-art conventional community detection methods.

Introduction

Complex networks such as biological networks, communication networks and social networks, are abstract representations of biological systems, communication systems and interaction systems respectively. Networks model interaction relationships between units of such various complex systems. They are powerful representations which can be used for analyzing the nature and function of complex systems.

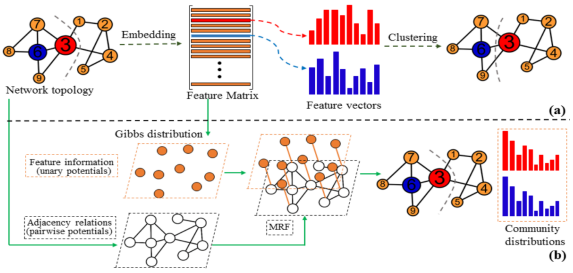


Figure 1: Comparison of (a) network embedding methods and (b) our MRF approach for community detection. The sizes of nodes are proportional to their degree centrality values. Each of the actual community structure (left) and the community structures derived by algorithms (right) is separated by a dotted line in the graph. (a) For embedding methods, the new feature representation in the form of individual vectors is first extracted and then taken as input to a data clustering algorithm. Nodes 3 and 6 both have high degree centralities and thus can be regarded as statistically-significant nodes. Here node 3 is wrongly divided. (b) For our MRF approach, the feature vectors obtained by embedding methods are modelled as unary potentials. The topological structure, which reflects the nodes dependency relationship, is modelled as pairwise potentials. They work together to form the final energy function, in which the statistically-significant node 3 is successfully assigned.

One of the most important properties of complex networks is their community structure, in which nodes are connected more densely within clusters than across clusters. Discovering communities is useful in many real applications such as targeted advertising, protein network analysis, recommendation system, etc. (Newman 2010).

Recently, discovering community structure in complex networks has attracted a great number of scholars from various research fields. A series of methods based on different theories and techniques have been proposed, as reviewed e.g. in (Fortunato and Hric 2016). They include spectral clustering (Newman 2010), modularity optimiza-

*Corresponding Author

tion (Yang et al. 2016), hierarchical clustering (Newman 2010), stochastic block model (Fortunato and Hric 2016), nonnegative matrix factorization (Karrer and Newman 2011; Wang et al. 2017; Zhang and Yeung 2012) and deep learning (Yang et al. 2016). These methods typically use network topology, often represented by the adjacency matrix. Network topology is a kind of relational data and represents the coupling relationship between pairs of nodes. These mutually related or constrained relationships between pairs of nodes offer a clue key for community detection.

Another line of related research which can identify community structure is to learn a new kind of network representation from network topology and feed the network representation as normal features to clustering algorithms such as k-means, as shown in Fig. 1(a). New network representations are extracted through embedding techniques which map the relational data from the original space to a low-dimensional feature space (Cui et al. 2018). Embedding techniques aim at learning the dense and continuous vectors of nodes in feature space, so that the noise or redundant information can be reduced while the intrinsic structural information can be preserved (Cui et al. 2018). Some are Node2Vec (Grover and Leskovec 2016), DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), MNMF (Wang et al. 2017), LINE (Tang et al. 2015), GraRep (Cao, Lu, and Xu 2015), ComE (Cavallari et al. 2017), AROPE(Zhang et al. 2018), etc. Such new network representation has at least three main advantages in network analysis tasks, namely low computational complexity, high capability for parallelization and applicability for traditional machine learning algorithms, thus it has received extensive attention recently (Cui et al. 2018).

However, we find that although one may have good results by directly clustering the new network representation, its community detection accuracy can still be improved. A more serious problem comes from the fact that, in many real networks, some statistically-significant nodes are often wrongly divided using the network embedding. This produces a worse effect in real applications than just the simple decrease in accuracy, since statistically-significant nodes often play pivotal roles in network analysis. For example, nodes with high degree are more likely to be the authoritative or active individuals which have an important effect in network and communities, while nodes with high betweenness are often bridges or hubs which play the pivotal role in connecting different communities. This may be because, although the distance is employed to capture the relationship between nodes by the embedding technique, nodes after mapping to feature vectors are not coupled anymore (Cui et al. 2018). Also, the lack of related or constraint relationship between nodes has an enormous impact on statistically-significant nodes in the network. It will thus be very important to not only improve the accuracy of network embedding methods, but also correct the wrongly-divided statistically-significant nodes, which may make embedding technology more suitable for real community detection applications.

To solve this problem, one may need a new framework that not only utilizes network embedding to play a dominant role in preserving its own advances (e.g. low computational complexity, high capability for parallelization, and applicability for machine learning algorithms), but also uses network topology to play a role of fine adjustments of the improper division of nodes (especially for the statistically-significant nodes) caused by missing dependency relationship. Fortunately, the pairwise Markov Random Field (MRF), which has been successfully used in image segmentation, meets this requirement. Specifically, its associated energy function can be factorized into a sum of potential functions, which consist in a set of unary potentials and a set of pairwise potentials (He et al. 2018). Unary potentials are defined to characterize the features of each individual having a main function, which is similar to the role of network embedding. On the other hand, pairwise potentials are defined based on pairs of objects and reveal inter-objects constraints to fine-tune the unary potentials, similar to the role of network relationship.

Based on the above idea, we define a new framework using pairwise MRF for community detection. Specifically, we use the new network embedding in the form of a low-dimensional vector of each node to define unary potentials, and further use relationships (derived from network topology) between nodes to define new pairwise potentials, as shown in Fig. 1(b). Under this framework, the advantage of network embedding can be preserved through the dependence on the network embedding, and the drawbacks of inadequate representations of dependencies can be mitigated by making up for the lack of coupling relationships.

Table 1: Datasets statistics and the overlapping ratio of edges between each original network and the network reconstructed via Node2Vec, DeepWalk or MNMF. Here n, m and K are the number of nodes, edges and communities, respectively.

Datasets	n	m	K	Rate (%)		
				Node2Vec	DeepWalk	MNMF
Zachary's karate club	34	78	2	38.46	41.96	32.62
Dolphin social network	62	160	2	48.71	62.95	30.64
High school friendship6	69	220	6	64.21	69.81	54.59
High school friendship7	69	220	7	64.21	69.81	54.59
Political books	105	441	3	54.71	61.15	34.47
American college football	115	613	12	72.52	76.63	72.27
Political blogs	1,490	16,717	2	14.17	20.71	17.12
Cora	2,708	5,429	7	65.59	69.12	60.33
Citeseer	3,312	4,732	6	59.14	63.91	57.77
UAI2010	3,363	45,006	19	41.98	47.15	38.48
North	13,882	381,935	7	34.41	39.44	32.25
PubMed diabetes	19,717	44,338	3	20.41	29.09	18.41

Motivated Observations

We first show that some statistically-significant nodes in real networks are often wrongly divided by using network embedding methods. We then show that the network reconstructed by embedding technologies loses many dependencies between nodes. The real networks with ground truth communities we use are shown in Table 1 (Newman 2017).

Observation Experiment 1

To validate that network embedding technologies often assign some statistically-significant nodes into incorrect communities, we first find the community structure of different networks based on network embeddings. We use three well known network embedding methods, Node2Vec (Grover and Leskovec 2016), DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), and MNMF (Wang et al. 2017) (a community preserving method) to extract the network embeddings for each of the twelve networks in Table 1. (Other methods such as LINE (Tang et al. 2015), GraRep (Cao, Lu, and Xu 2015) and ComE (Cavallari et al. 2017) were also tested. We do not show them since they have similar effects.) Then, as was done in many network embedding works (Cui et al. 2018), we use k-means to cluster these new network representations to get the community structures.

After finding the nodes which are wrongly divided by the above methods (by comparing with the ground truth), we further determine which nodes among them are statistically significant. To this end, we employ three largely used indices of network nodes statistical significance, i.e. degree centrality, eigenvector centrality and betweenness centrality (Newman 2010), to analyze each of the wrongly divided nodes.

Table 2 shows some representative nodes in each of the networks that are wrongly divided by Node2Vec, DeepWalk and MNMF. Due to space limit, we display no more than three wrongly-divided statistically-significant nodes for each network. As shown, there are 8 out of 12 networks which include wrongly-divided statistically-significant nodes by using Node2Vec, DeepWalk and MNMF. This demonstrates that when one applies embedding to find community structures in real-world networks, some statistically-significant nodes which play an important role in each network are often wrongly divided. This shows that embedding methods (including the community preserving method MNMF) have serious drawbacks for real community detection applications. But fortunately, almost all nodes shown in Table 2 can be corrected using our framework which will be introduced in the following sections.

Table 2: The statistically-significant nodes which are wrongly divided by Node2Vec, DeepWalk, MNMF and our new MRF framework (our approaches using Node2Vec, DeepWalk and MNMF as bases are named as MRF-N, MRF-D and MRF-M, respectively). "×" denotes a node which is which is wrongly divided by a network embedding method. The rank of a node is shown in terms of degree centrality, eigenvector centrality (eigen for short) and betweenness (between for short) centrality, respectively.

Datasets	Node ID	Node2 Vec	MRF-N	Deep Walk	MRF-D	MNMF	MRF-M	Degree (rank)	Eigen (rank)	Between (rank)
Karate	3	×		×		×		4	3	4
	4	×				×		17	8	6
Friend6	2			×				7	20	10
	27	×		×		×		8	26	21
	62	×				×		9	8	33
Friend7	7			×		×		5	4	7
	27	×				×		8	26	21
	62	×				×		9	8	33
Polbooks	50			×		×		38	35	2
	59	×		×		×		19	42	8
Polblogs	101	×		×				63	108	24
	123			×		×		42	21	85
	1,182	×				×		234	144	262
Cora	28			×		×		9	273	7
	73	×		×				8	203	9
	88	×				×		11	180	16
Cite	161	×		×	×			351	455	78
	632	×				×		24	28	66
	725			×		×		121	63	72
North	6,404	×		×		×		128	23	50
	7,073	×		×		×		52	76	31
	7,089	×				×		21	151	48
Pubmed	316	×		×		×		36	1053	21
	604	×		×		×		38	819	140
	3,657	×		×		×		70	818	28

Observation Experiment 2

According to the first experiment, the embedding technology will assign some statistically-significant nodes to incorrect communities. This may happen when the network embedding method loses some dependency relationships between pairs of nodes in the network topology. To validate this hypothesis, we first reconstruct a new network using its embedding, and then calculate the overlapping ratio of edges between the reconstructed and the original networks. Intuitively, the smaller the overlapping ratio, the more dependencies were lost reconstructing the network from embedding, i.e. the more embedding loses dependencies.

Again, we first use embedding methods, i.e. Node2Vec, DeepWalk and MNMF, to extract the feature vectors of nodes for each of the twelve networks. Then, for each node i in a network, we compute the feature similarities between node i and all other nodes by using the cosine similarity. Then we sort the similarity in decreasing order, select the top d_i (d_i is the degree of node i) nodes with the highest similarities and take them as structural neighbors (to distinguish from topological neighbors in the original network) of node i in the reconstructed network. Using the structural neighbors of all nodes, we finally reconstruct a new network.

The right part of Table 1 shows the overlapping ratio between the original network and the network reconstructed by Node2Vec, DeepWalk and MNMF. As shown, the average overlapping ratio between the reconstructed and the original networks are 46.76%, 52.90% and 40.81%, respectively, based on Node2Vec, DeepWalk and MNMF. This indicates that there are almost half of the dependency relationships in the original network topology, which are lost or do not appear directly in the networks reconstructed via embedding.

Finally, let us analyze the reason why some statistically-significant nodes are wrongly divided by the embedding methods. As discussed in (Fortunato and Hric 2016), traditional community detection approaches mainly focus on network topology in which statistically-significant nodes are often highly emphasized in the community detection process, so that it is not easy to wrongly divide them in this case. Unfortunately, after mapping network topology to feature space by embedding methods (including the community

preserving method MNMF), some dependency relationships between nodes are (directly or indirectly) lost, as shown in Table 1. Then, the statistical significance of nodes in the original network topology cannot be preserved. This will lead to the fact that the originally statistically-significant nodes are not necessarily protected as before, and hence they may be more easily divided wrongly in the embedding-based community detection process. Our work here is to correct network embeddings by utilizing network topology to essentially improve community detection performance.

The MRF Framework

We first define a structured pairwise Markov Random Field (MRF) to correct network embeddings by using network topology for community detection. We then present a belief propagation (BP) algorithm to learn the MRF model. Finally, we give its complexity analysis.

The Model

Consider an undirected and unweighted network G with n nodes and m edges, which is represented by an adjacency matrix $A = [a_{ij}]_{n\times n} \in R^{n\times n}$, where a_{ij} is 1 if nodes i and j are connected, or 0 otherwise. We divide the nodes into K communities, where each node i has a label $c_i \in \{1, \cdots, K\}$, indicating which of the K communities it belongs to.

Our MRF model consists of two parts (Fig. 1(b)). One is a set of unary potentials which are defined to force the network embeddings to play a dominant role. The other is a set of pairwise potentials which are defined based on pairs of nodes and reveal constraints to fine-tune the unary potentials. All these unary potentials and pairwise constraints in the energy function work together in order to characterize and exploit the network embeddings as well as network topology to achieve a global coherent community structure. Then, the complete energy function can be defined as:

$$E(C; A) = \sum_i \theta_i(c_i) + \sum_{<i,j>\in\varepsilon} \theta_{ij}(c_i, c_j) \tag{1}$$

where $C = (c_1, c_2, \cdots, c_n)$ is the community assignments of nodes, ε is the set of edges, θ_i and θ_{ij} define unary and pairwise potentials respectively (see below).

With the energy function for partition C defined, we can then adopt the Gibbs distribution, which is a function of inverse temperature β, to compute the posterior probability of partition C given network topology A. This is defined as:

$$P(C|A) = \frac{1}{Z} e^{-\beta E(C;A)} \tag{2}$$

where Z is a normalization coefficient. Then, the best community partition $\hat{C}$ can be obtained according to:

$$\hat{C} = \arg\max_C P(C|A) \tag{3}$$

Definition of Unary Potentials We first define unary potentials using network embeddings in order to characterize each individual's features.

Let $\theta_i(c_i)$ denote the unary potential of node i, where the value of c_i ranges from 1 to K. That is, $\theta_i(c_i)$ measures the cost of assigning community label c_i to node i. For example, if node i is more likely to belong to the first rather than the second community, we will have $\theta_i(c_1) < \theta_i(c_2)$. We need to model the network embeddings of all nodes into unary potentials in the energy function. However, network embedding is a type of vector representations of nodes, which is different from the real form of unary potentials. For this problem, we conversely employ the Gibbs distribution in (2). We use γ_{c_i} to denote the probability that node i belongs to community c_i. Then, the unary potential can be obtained by its corresponding probability distribution:

$$\theta(c_i) = \frac{-\log \gamma_{c_i}}{\beta} \tag{4}$$

where $\beta = 1$, and γ_{c_i} is the probability distribution of node i belonging to each community.

Specifically, we use the Gaussian Mixture Model (GMM) to approximate the probability distributions γ_{c_i} of all nodes belonging to various communities. Then, γ_{c_i} is obtained by the sum where y is the feature vector of the node:

$$P\left(y \,|\mu, \sigma^2\right) = \sum_{i=1}^{M} \alpha_i \phi\left(y \,|\mu_i, \sigma_i^2\right) \tag{5}$$

$$\phi\left(y \,|\mu_i, \sigma_i^2\right) = \frac{1}{\sqrt{2\pi}\sigma_i} exp\left(-\frac{(y-\mu_i)^2}{2\sigma_i^2}\right) \tag{6}$$

where $\phi(y|\mu_i, \sigma_i^2)$ is a component in GMM with mean μ_i and variance σ_i^2 $((\mu;\sigma) = (\mu_1, \cdots, \mu_M; \sigma_1, \cdots, \sigma_M))$, and α_i is the mixing coefficient. The GMM can be taken as a full-covariance Gaussian mixture with M components, in which each component characterizes a community, thus the number of components M should be equal to the number of communities K. We employ EM algorithm to infer the parameters, and then get the probability distributions γ_{c_i}, which can be converted into unary potentials via (4).

Definition of Pairwise Potentials We then define another type of potential functions, i.e. pairwise potentials, which play the role of correcting network embeddings. Different from using the fully connected network structure as done in (He et al. 2018), we use network topology directly as the underlying MRF structure since our purpose here is to use the coupling relationship between nodes to make up the drawbacks of inadequate dependencies of embeddings. This also makes the model efficient due to the sparsity of networks.

In addition, it is often believed that an observed link in a network is either formed by a direct connection (i.e. direct edge) or formed from indirect transmission (i.e. transitive edge). It is not an ideal way of using these links uniformly to form a MRF to correct network embeddings since the dependency strength of transitive edges is often overestimated. Here we use direct dependency strength proposed by (Feizi et al. 2013) to define the dependency relationship between linked nodes to correct network embeddings.

To be specific, let $G^{(h)}$ be the hth order direct dependency graph of the observed network G with adjacency matrix A. Then the interaction process between nodes proceeds as follows. Given the direct dependency graph G_{dir} (i.e.,

$G^{(1)}$) with edges representing the tie strength of the first order interactions, an individual's influence can spread when the nodes it interacted with connects with other nodes in the next step, with its strength diluted along with the increase of interaction order. We take the final observations to be the cumulative effects of all the order of interactions, that is $A = A_{dir} + A^{(2)} + A^{(3)} + ...$, where $A^{(h)}$ (including $A_{dir} = A^{(1)}$) denotes the weighted adjacency matrix of $G^{(h)}$ in which each element $a_{ij}^{(h)}$ represents the hth order connection strength between nodes i and j. From the high order interaction property of networks, we have $A^{(h)} = A^{(h-1)}A_{dir}$=$(A_{dir})^h$, $h = 2, 3, \ldots$ Based on the property of power of matrices, we have

$$A^{(h)} = (A_{dir})^h = U(\Lambda_{dir})^h U^{-1} \tag{7}$$

where $\Lambda_{dir} = diag(\lambda_1^{dir}, \cdots, \lambda_n^{dir})$ in which λ_i^{dir} denotes the ith eigenvalue of the direct dependency graph G_{dir}, and U denotes the eigenvectors shared by every $A^{(h)}$. The observation graph can then be defined as

$$\begin{aligned} A &= (A_{dir}) + A^{(2)} + A^{(3)} + ... \\ &= (A_{dir}) + (A_{dir})^2 + (A_{dir})^3 + ... \\ &= U\Lambda U^{-1} \end{aligned} \tag{8}$$

$\Lambda = diag(\lambda_1 = \sum_{i\geq 1}(\lambda_1^{dir})^i, \cdots, \lambda_n = \sum_{i\geq 1}(\lambda_n^{dir})^i)$ in which λ_i is the ith eigenvalue of A, and U denotes the eigenvectors. However, the real situation is that Λ and U can be calculated easily since A is observed, while Λ_{dir} and A_{dir} need to be derived. According to (Feizi et al. 2013), each eigenvalue of the direct dependency network is expressed as a nonlinear function of a single corresponding eigenvalue of the convolved observed network. We can then construct the direct dependency graph as

$$A_{dir} = U\Lambda_{dir}U^{-1} \tag{9}$$

where $\Lambda_{dir} = diag(\lambda_1^{dir} = \frac{\lambda_1}{1+\lambda_1}, \cdots, \lambda_n^{dir} = \frac{\lambda_n}{1+\lambda_n})$. Then A_{dir} can be easily derived based on Λ and U, in which each element a_{ij}^{dir} represents the direct dependency strength between nodes i and j.

Based on the above preparations, we define the pairwise potentials as follows. Let $\theta_{ij}(c_i, c_j)$ represent the pairwise potential of two connected nodes i and j with community labels c_i and c_j respectively. Using network topology as the underlying MRF structure and based on the direct dependency strength of links, the pairwise function is defined as:

$$\sum_{<i,j>\in\varepsilon} \theta_{ij}(c_i, c_j) = \sum_{<i,j>\in\varepsilon} a_{ij}^{dir} \times (-1)^{\delta(c_i,c_j)} \tag{10}$$

where $\delta(c_i, c_j)$ is 1 if $c_i = c_j$, and 0 otherwise, ε is the set of edges. The pairwise term in (10) achieves our goal that we encourage pairs of nodes with high direct dependency strength to be in the same community. Since the number of edges in the network is fixed, if more edges appear within a community, there will be fewer edges appearing across communities. This is consistent with the definition of community detection which aims at dividing nodes into communities with denser connections within communities and sparser connections between communities.

Model Inference

We now need to derive the best community partition $\hat{C}$ by maximizing the posterior probability $P(C|A)$ of this model. Here, instead of optimization of individual per-variable marginal probabilities as done in other community detection models, we use the belief propagation (BP) algorithm with a max-sum version to maximize joint probability distribution so as to derive the better configuration of community memberships.

The key idea of our BP algorithm is that: each node i sends a 'message' to node j directly connected with i in the network topology (j is i's neighbor), indicating the maximum negative energy achievable when fixing the community of i to c_i if j is absent. We use it to denote the maximum negative energy that i will be in community c_i in the absence of j, which is obtained by recursively computing the messages that i receives from other neighbors (k) of i:

$$\psi_{c_i}^{i\to j} \leftarrow \beta \log \gamma_{c_i} + \sum_{k\in N(i)/j} \left[\max_{c_k}\left[-\beta a_{ij}^{dir}(-1)^{\delta(c_i,c_k)} + \psi_{c_k}^{k\to i}\right]\right] \tag{11}$$

When the algorithm converges, the variable max-belief, that is $\mu_i(c_i)$, can be computed as:

$$\mu_i(c_i) \leftarrow \beta \log \gamma_{c_i} + \sum_{k\in N(i)} \left[\max_{c_k}\left[-\beta a_{ij}^{dir}(-1)^{\delta(c_i,c_k)} + \psi_{c_k}^{k\to i}\right]\right] \tag{12}$$

To recover the joint maximum posterior labeling, we select the state which corresponds to maximum max-belief for each variable c_i:

$$\hat{c}_i = \underset{c_i\in\{1,\ldots,K\}}{\arg\max}\ \mu_i(c_i) \tag{13}$$

Complexity Analysis

The pairwise structure of our framework is the same as network topology in which each node needs to send K 'messages' to its neighbors at each iteration, thus the total number of 'messages' to be updated in each iteration is $2mK$, where m and K are the number of edges and communities, respectively. Also, the complexity of updating a message is constant, i.e. $O(1)$, so that the complete complexity of our algorithm is $O(2mKT)$, where T is the number of iterations. Because T and K can be taken as constants compared with network scales (i.e. m), the complexity of our algorithm is finally $O(m)$ which is nearly linear on large sparse networks.

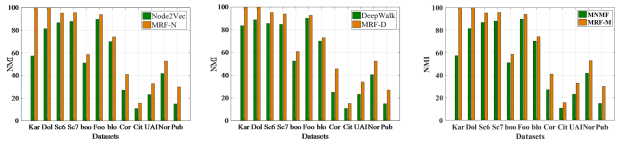
Experiments

We first compare our approach with three network embedding methods for community detection. We then compare it with some state-of-the-art community detection methods.

Compare with Network Embedding Algorithms

We mainly focus on whether our framework maintains the advantages of network embedding methods and overcomes their disadvantages, making it suitable for community detection. Specifically, we compare our approach with three well-known embedding methods (Node2Vec, DeepWalk and MNMF) on twelve real networks with known communities

listed in Table 1. We use the default parameter for all the methods. The feature dimension is set to 64. We use k-means algorithm (as in other network embedding papers for community detection (Cui et al. 2018)) to cluster the feature vectors derived from Node2Vec, DeepWalk and MNMF. Correspondingly, our framework using Node2Vec, DeepWalk and MNMF features is called MRF-N, MRF-D and MRF-M, respectively. We use normalized mutual information (NMI) as the accuracy metric and runtime as the efficiency metric.



(a) (b) (c)

Figure 2: Comparison of our method with three embedding methods, (a) Node2Vec, (b) DeepWalk, (c) MNMF, in terms of NMI on 12 real networks with ground truth communities in Table 1. The name of each network is represented with its first three characters.

Table 3: Comparison of our new MRF framework with three network embedding methods in terms of percentage of statistically significant nodes which were misclassified.

Datasets	Rate (%)		RER(%)	Rate (%)		RER(%)	Rate (%)		RER(%)
	Node2Vec	MRF-N		DeepWalk	MRF-D		MNMF	MRF-M	
Karate	75.00	0	100	100	0	100	80.00	0	100
Dolphin	50.00	0	100	0	0	100	50.00	0	100
Friend6	21.02	0	100	21.05	0	100	23.33	0	100
Friend7	66.67	25.00	66.30	33.33	0	100	48.87	0	100
Polbooks	38.89	21.55	44.59	37.65	21.90	41.83	38.89	21.55	44.59
Football	20.00	15.38	23.08	23.53	20.00	15.00	20.46	15.55	24.00
Polblogs	19.30	15.79	18.18	33.33	21.00	39.99	28.32	19.92	29.66
Cora	27.99	20.10	28.19	30.69	20.05	34.67	32.68	21.48	34.27
Citeseer	32.68	26.11	20.10	32.68	21.48	34.21	31.70	22.23	29.87
UAI2010	35.04	24.00	31.50	34.95	28.33	18.94	32.87	27.98	14.88
North	26.50	19.92	24.83	20.46	12.10	40.86	26.50	13.54	48.91
PubMed	26.11	18.88	27.69	24.48	18.88	22.88	24.59	16.71	32.05

The comparison results in terms of NMI are given in Fig. 2. As shown, our methods MRF-N, MRF-D and MRF-M have the best performance on all twelve networks. On average, MRF-N and MRF-D are 12.30% and 9.99% more accurate than Node2Vec and DeepWalk, respectively. Especially, MRF-M also improves MNMF which is a community preserving-based embedding technology (by $\sim$12.21%).

However, the runtime of our MRF is somewhat larger than that of the original embedding methods. But on average, MRF-N, MRF-D and MRF-M increased by only 16%, 25% and 21% over Node2Vec, DeepWalk and MNMF, respectively, in runtime from the results. The reason is that our approach needs to calculate embeddings to define unary potentials first before its inference, thus needs a little more time. This result further demonstrates that the high-efficiency property of embedding methods has been reasonably maintained in our MRF framework.

In addition, remember the Observation Experiment1 section. There are some statistically-significant nodes in different networks wrongly divided by embedding methods, such as Node2Vec, DeepWalk and MNMF. In Table 3, we show that the percentage of statistically-significant nodes which were misclassified in the whole network for each dataset, by embedding methods and our method. These results indicate that some statistically-significant nodes are wrongly divided using embedding methods, including the community-preserving approach MNMF. Based on this, we compute the relative error reduction (RER) for each network, given in Table 3. (RER = $(error_{m1} - error_{m2})/error_{m2} \times 100\%$, where $error_{m1}$ and $error_{m2}$ represent the percentage of statistically-significant nodes which were misclassified by our method and embedding method.) As shown, the RER value of each network is very high, which indicates that our approach can successfully correct most of the wrongly divided statistically-significant nodes. This may be because the dependency relationship which is neglected by embedding technology is actually made up by our framework, and the statistically-significant nodes which were wrongly divided for this reason can now be corrected by it. This further indicates that our framework not only improves the accuracy of network embedding methods for community detection, but also corrects the wrongly-divided statistically-significant nodes, making network embedding suitable for community detection essentially. In comparison, the community preserving-based embedding technology, such as MNMF, does not have this ability.

Compare with Community Detection Algorithms

Here we mainly validate whether our new MRF framework is competitive with or performs better than the existing community detection methods, particularly in the realm of statistical models. Besides the 12 network with ground truth communities in Table 1, here we also add 8 networks without known communities. We compare our approach with six baseline algorithms, i.e. 1) Karrer's method which is a degree-corrected stochastic block model (Karrer and Newman 2011); 2) SNMF (Wang et al. 2011), 3) BNMTF (Zhang and Yeung 2012) and 4) MNDP (Jin et al. 2015) which are nonnegative matrix factorization based methods; 5) DNR (Yang et al. 2016) which is an algorithm using deep learning; 6) NetMRF(He et al. 2018) the MRF-based community detection method. We use normalized mutual information (NMI) and accuracy (AC) (Fortunato and Hric 2016) as accuracy metrics when ground-truth of communities is known, and modularity Q (Newman 2010) as quality metrics when true communities are unknown.

We take the source codes of the baselines and report results using the default parameters, except DNR (Yang et al. 2016). For DNR, we use the results that were reported in the original paper since getting satisfactory results requires adjustment of too many parameters. All the methods converge to local minima, so that we run each method 20 times and report the result with the highest likelihood.

Real Networks with Known Communities The networks with known communities are listed in Table 1. Because all

the methods need the number of communities K to be given, we set K to the ground-truth when true communities are known. The methods that were compared in NMI and AC, along with the results on the twelve networks, are shown in Tables 4 and 5. In terms of NMI, our methods MRF-N, MRF-D and MRF-M show the best performance on 11 and 10 out of the 12 networks respectively. For example, using NMI index, our MRF-N is 13.65%, 10.07%, 9.15%, 7.93%, 7%, 5.84% and 4.62% more accurate than Karrer's method, SNMF, BNMTF, MNDP, DNR-L2, DNR-CE and NetMRF on average; MRF-D and MRF-M have similar effects. We also have similar results in AC. These further emphasize the superiority of our approach compared with the existing ones, including another MRF-based method NetMRF.

Real Networks without Known Communities We further compare these methods on eight real networks with no known communities (Newman 2017) which are listed in Table 6. When no "true" number of communities K is known, our framework can run on different K's (e.g. in a range of K_{min} to K_{max}), and get the best K as well as communities which correspond to the smallest energy function. However, the methods compared cannot find the number of communities automatically, which is typically an issue for statistical model-based methods. So for fair comparison, we ran Louvain method (Blondel et al. 2008) to estimate the numbers of communities and used the estimated numbers in all methods. We adopted modularity Q (Newman 2010) for evaluating the quality of a community structure.

Results are shown in Table 6. It is not a surprise that NetMRF has the best performance in terms of modularity Q in general, since it does optimize Q. But except for NetMRF, our MRF-N, MRF-D and MRF-M have the best performance on 6, 7 and 7 of 8 networks, respectively. On average, MRF-N is 0.1728, 0.0334, 0.0514 and 0.0293 better than Karrer's method, SNMF, BNMTF and MNDP; MRF-D is 0.1673, 0.0279, 0.0493 and 0.0238 better than Karrer's method, SNMF, BNMTF and MNDP; MRF-M is 0.1642, 0.0248, 0.0448 and 0.0207 better than Karrer's method, SNMF, BNMTF and MNDP. Particularly, as the Q-values are normally in the range of 0.3 to 0.8 (Newman 2010), our MRF framework obviously outperforms other methods, except for NetMRF which optimizes Q.

Runtime on Real Networks We validate the efficiency of our framework. We test it on all 20 networks used above, and compare it with Karrer's method, SNMF, BNMTF, MNDP and NetMRF. From the results, though our MRF-N, MRF-D and MRF-M need a little more time on small networks, they outperform or are at least competitive with the best baselines on larger networks. On average, MRF-M takes 6%, 74%, 0.16%, 3.4% and 16% of the runtime of Karrer's method, SNMF, BNMTF, MNDP and NetMRF, respectively, on large networks including Cora, Citeseer, UAI2010, NorthEastern, PubMed, E-mail, Power and Word. MRF-N and MRF-D also have similar results, which means that our approach is suitable to deal with large-scale networks. Especially, our approach is much faster than NetMRF since we use a sparser pairwise structure than NetMRF.

Table 4: Comparison with 6 methods in NMI on 12 real networks with ground-truth communities. DNR-L2 and DNR-CE are two versions of the deep learning method in (Yang et al. 2016). The former is DNR with L2 norm and the latter is that with cross-entropy distance. They did not give DNR's results on some networks so we mark them as 'N/A'. '-' denotes run time >100 hours.

Datasets	NMI index (%)									
	Karrer	SNMF	BNMTF	MNDP	DNR_{L2}	DNR_{CE}	NetMRF	&MRF_{N}	MRF_{D}	MRF_{M}
Karate	83.72	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Dolphin	88.88	81.41	81.41	88.88	88.9	81.8	88.88	**100**	**100**	**100**
Friend6	77.02	78.64	71.22	79.30	88.8	92.4	93.98	**95.21**	**94.24**	**95.21**
Friend7	85.10	82.11	84.30	84.26	90.7	93.2	**93.24**	**94.55**	**93.95**	92.77
Polbooks	54.20	56.48	51.18	53.01	55.2	58.2	56.88	**58.61**	**58.61**	**58.61**
Football	87.06	90.38	92.42	92.42	92.7	91.4	92.42	**93.91**	**92.77**	**92.71**
Polblogs	45.68	70.95	70.78	71.07	38.9	51.7	71.83	**74.21**	**73.09**	**74.27**
Cora	17.06	24.72	26.08	33.99	**46.3**	42.1	37.24	41.01	45.68	44.46
Citeseer	4.05	7.77	**15.51**	10.21	N/A	N/A	11.15	**15.66**	15.25	15.19
UAI2010	20.98	23.24	21.68	25.01	N/A	N/A	25.76	**32.92**	**34.15**	**28.30**
North	49.13	38.66	-	40.67	N/A	N/A	45.24	**52.84**	**52.53**	**51.07**
PubMed	12.28	13.80	-	14.96	N/A	N/A	16.89	**30.02**	**27.01**	**29.98**

Table 5: Comparison with 5 methods in AC on 12 real networks with ground-truth of communities. DNR does not appear here because they did not use AC in (Yang et al. 2016).

Datasets	Accuracy AC (%)							
	Karrer	SNMF	BNMTF	MNDP	NetMRF	MRF-N	MRF-D	MRF-M
Karate	97.06	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Dolphin	98.39	96.77	96.77	98.39	96.77	**100**	**100**	**100**
Friend6	81.16	78.26	60.87	78.26	95.65	**97.10**	**97.10**	**96.67**
Friend7	94.20	88.41	89.86	89.86	**95.65**	**95.65**	**95.65**	95.33
Polbooks	82.86	80.95	69.52	81.90	**83.81**	**84.76**	**84.76**	82.97
Football	84.35	87.83	91.30	91.30	91.30	**93.04**	**93.04**	**92.76**
Polblogs	87.18	94.69	94.61	94.69	95.01	**95.42**	**95.26**	**95.26**
Cora	37.70	42.25	40.95	44.39	58.05	**58.88**	**58.68**	**63.05**
Citeseer	26.57	32.79	27.12	25.79	25.79	**34.76**	**35.76**	**34.11**
UAI2010	27.78	28.52	25.51	28.92	31.14	**34.83**	**35.38**	**34.71**
North	66.36	58.87	-	56.4	65.11	**70.66**	**71.01**	**68.96**
PubMed	53.64	52.87	-	50.72	55.53	**66.50**	**65.30**	**66.41**

Table 6: Comparison with 4 methods on eight real networks without ground-truth. n is the number of nodes, m the number of edges and K the number of communities derived by Louvain method (Blondel et al. 2008).

Datasets	n	m	K	Modularity Q							
				Karrer	SNMF	BNMTF	MNDP	MRF_{N}	MRF_{D}	MRF_{M}	NetMRF
LesMis	77	254	6	0.4575	0.5453	0.5487	0.5434	**0.5556**	**0.5502**	**0.5533**	0.5600
Adjnoun	112	425	7	0.1041	0.2672	0.2634	0.2712	0.2602	**0.2818**	0.2588	0.2813
Jazz	198	2,742	4	0.3696	0.4348	0.4347	0.4377	**0.4409**	**0.4405**	**0.4405**	0.4495
Neural	297	2,148	5	0.2617	0.3701	0.3689	0.3811	**0.3938**	**0.3877**	**0.3911**	0.4120
Metabolic	453	2,025	10	0.2656	0.3879	0.3834	0.3796	**0.4119**	**0.4281**	**0.4201**	0.4579
E-mail	1,133	5,451	11	0.5126	0.5007	0.4685	0.5154	**0.5652**	**0.5652**	**0.5484**	0.5712
Power	4,941	6,594	39	0.1796	0.8649	0.7212	0.8683	**0.9207**	**0.8803**	**0.8901**	0.9266
Word	5,018	55,234	12	**0.4595**	0.3546	-	0.3613	0.4446	0.4147	0.4214	0.4266

Conclusion

We developed a general MRF framework, which employs coupling relationships to correct the drawbacks of inadequate network embeddings of dependencies. Our framework corrects most of the wrongly-divided statistically-significant nodes while maintaining the advantage of embedding technology, making embedding suitable for real community detection applications. More importantly, this new framework is applicable to nearly all types of network embedding methods. Experimental results verify above views well.

We also analyzed some community preserving-based network embedding methods presented recently to improve community detection (Cavallari et al. 2017; Wang et al. 2017; Sun et al. 2017). We found that these methods still possess the drawback, i.e. many statistically significant nodes are often wrongly divided (see Tables 1 and 3). This is because they still map nodes from the dependent network space into the independent feature space. Thus, in essence, there is no difference between them and other embedding techniques in solving this problem. They lose the coupling relationship between nodes, so they cannot solve this drawback of using network embedding for community detection.

Compared with NetMRF, our approach 1) adds the definition of unary potentials using network embeddings, 2) uses the sparse network structure instead of the fully connected node pairs to define the MRF structure, which makes it not only efficient but also suitable to correct network embeddings using coupling relationship, and 3) introduces direct dependency strength on links to alleviate the situation that indirect links are overestimated, in order to further improve the ability of correcting network embeddings.

Acknowledgments

This work was partially supported by the Natural Science Foundation of China (No. 61876128, 61772361, 61502334, 61772304), Ramanujan Faculty Fellowship, India and Early Career Research Award, SERB India.

References

Blondel, V. D.; Guillaume, J. L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *J. stat. Mech.* 2008(10):155–168.

Cao, S.; Lu, W.; and Xu, Q. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 891–900. ACM.

Cavallari, S.; Zheng, V. W.; Cai, H.; Chang, K. C.-C.; and Cambria, E. 2017. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, 377–386. ACM.

Cui, P.; Wang, X.; Pei, J.; and Zhu, W. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*.

Feizi, S.; Marbach, D.; Médard, M.; and Kellis, M. 2013. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature biotechnology* 31(8):726.

Fortunato, S., and Hric, D. 2016. Community detection in networks: A user guide. *Phys. Rep.* 659:1–44.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. San Francisco, USA: ACM.

He, D.; You, X.; Feng, Z.; Jin, D.; Yang, X.; and Zhang, W. 2018. A network-specific Markov random field approach to community detection. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*. San Francisco, USA: AAAI.

Jin, D.; Chen, Z.; He, D.; and Zhang, W. 2015. Modeling with node degree preservation can accurately find communities. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. Austin, Texas, USA.: AAAI.

Karrer, B., and Newman, M. E. J. 2011. Stochastic blockmodels and community structure in networks. *Phys. Rev. E* 83(1):016107.

Newman, M. E. J. 2010. *Networks. An introduction*. Oxford University Press, Inc.

Newman, M. E. J. 2017. Real world network data in newman's homepage. http://www-personal.umich.edu/-mejn/netdata.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Sun, B.; Shen, H.; Gao, J.; Ouyang, W.; and Cheng, X. 2017. A non-negative symmetric encoder-decoder approach for community detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 597–606. ACM.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. International World Wide Web Conferences Steering Committee.

Wang, F.; Li, T.; Wang, X.; Zhu, S.; and Ding, C. 2011. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery* 22(3):493–521.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, 203–209. AAAI.

Yang, L.; Cao, X.; He, D.; Wang, C.; Wang, X.; and Zhang, W. 2016. Modularity based community detection with deep learning. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2252–2258. New York, USA: IJCAI.

Zhang, Y. and Yeung, D. Y. 2012. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 606–614. ACM.

Zhang, Z.; Cui, P.; Wang, X.; Pei, J.; Yao, X.; and Zhu, W. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2778–2786. ACM.