Community-Centric Graph Convolutional Network for Unsupervised Community Detection

Abstract

Community detection, aiming at partitioning a network into multiple communities, is of practical importance. Graph Convolutional Networks (GCN), a new deep learning technique, has recently been developed for community detection. MRFasGCN, a new GCN method for community detection, goes one step further to incorporate Markov Random Fields (MRF) modeling of community structures to improve accuracy. However, the existing GCN methods are semi-supervised community-finding methods, even though community detection is in essence an unsupervised learning problem since little training data are available for most applications and information from one network can be hardly used for another network. To address this problem, we introduced a new GCN approach for *unsupervised community detection* under the framework of Autoencoder. We first cast MRFasGCN as an encoder and furthermore derived node community membership in the hidden layer of the encoder. We then introduced a community-centric dual decoder to reconstruct network topologies and node attributes separately in an unsupervised fashion, for faithful community detection in the input space. We specifically introduced a scheme of local structural enhancement to accommodate nodes to have more common neighbors and similar attributes with the similar community memberships. Experimental results on some real-world networks showed that our new method outperformed the state-of-the-art methods for community detection. We also showed the effectiveness of the novel decoding mechanism for generating links and attributes together over the commonly used methods for reconstructing links alone.

Introduction

Real-world systems often appear in the form of networks or graphs. Examples include social networks, networks of protein interaction, power grid and world trade networks. Real networks have modular structures or communities (Fortunato 2010), which are densely connected subgraphs with nodes of close relationships and similar properties (Li et al. 2017). For example, the World Wide Web can be regarded as a collection of web communities, each of which contains webpages describing the same or similar topics (Adamic and Adar 2003). Identification and analysis of network modular structures are an effective means to understanding the underlying organizational principles and functions of the system that the network represents. Community detection has been an active area of research, as surveyed in (Falih et al. 2018; Fortunato 2010; Fortunato and Hric 2016), and many community detection methods have been proposed, including that based on statistical modeling (Chen, Li, and Bruna 2018), modularity optimization (Yang et al. 2016), matrix factorization (Wang et al. 2017), and so on.

Deep learning has recently been adopted in network analysis (Kipf and Welling 2016b; 2016a; Yang et al. 2016; Jin et al. 2018; Pan et al. 2019b). In particular, Graph Convolutional Networks (GCN) has attracted a great deal of attention lately due to its success on *supervised* and *semi-supervised* classification of nodes in a graph (Li, Han, and Wu 2018; Kipf and Welling 2016a) which can be also adopted for community detection. Of particular interest and relevance to the current study is MRFasGCN (Jin et al. 2019), a state-of-the-art GCN-based *semi-supervised* community detection method, which incorporates a Markov Random Fields (MRF) modeling of communities in the GCN framework.

Community detection is in essence an *unsupervised* learning problem. Real-world networks are typically unique the training data from one network can be hardly used adequately for another network. As a corollary to this observation, in practice the problem of community finding has to be solved for individual networks in isolation; when finding communities in a network, the only data available for analysis are the information on the network itself. A scheme of semi-supervised learning, such as the MRFasGCN method, can apply in such a way that partial label information on some of the nodes in a given network can be used to predict the community identities of the remaining unlabeled nodes in the same network. However, for most real application problems, even such partial training data are costly and arduous to gather.

Therefore, in order to advance the state-of-the-art of network community detection, it is of great technical significance and paramount practical importance to develop GCN-based algorithm for *unsupervised* network community detection by exploiting the great power of automatic feature

learning and effective optimization that deep learning can offer. It also remains to be seen if an end-to-end deep neural network approach can outperform the existing methods that use statistical methods and other machine learning techniques for network community finding.

We developed a novel **G**CN-based approach for **U**nsupervised **C**ommunity **D**etection in attribute networks, which we referred to as GUCD. In this method, we incorporated network modeling method of MRFasGCN in the autoencoder framework and introduced specialized neural network architecture suitable for learning network structural and semantic information and predicting network topologies, along with network community structures, and node semantic contents at the same time. We further added a regularization of local enhancement to the latent communities, i.e. we first constructed an aggregated graph combining the information of both topology and attributes, and then made each pair of connected nodes in this graph have similar community distributions. We conducted extensive experiments on some real networks to compare the new approach with the best existing methods and analyzed the features of the new approach. To our best knowledge, GUCD is the first method using GCN for unsupervised network community finding, a difficult category learning problem.

Preliminaries

We first introduce some notations and define the problem of community detection, and then discuss GCN as well as MRFasGCN (Jin et al. 2019) which serve as the bases of our new method.

Notations and Problem Definition

An undirected and attributed network is represented as a graph $G = (V, E, W)$ over a set of n nodes $V = \{v_1, v_2, \ldots, v_n\}$, a set of e edges E with $e_{ij} = (v_i, v_j) \in E$ if an edge exists between nodes v_i and v_j, and a set of m node attributes $W = \{w_1, \ldots w_m\}$. The topological structure of G is represented by an $n \times n$ adjacency matrix $A = (a_{ij})_{n\times n}$, where $a_{ij} = 1$ if $e_{ij} \in E$, or 0 otherwise. An $n \times m$ attribute matrix X is used to denote the attributes of nodes, where $x_{it} = 1$ if a node has attribute w_t, or 0 otherwise.

Given an attribute network G, community detection is to partition the n nodes into K communities $C = \{C_1, C_2, \ldots, C_K\}$ so that each node v_i has a community identity or label $c_i \in L = \{l_1, l_2, \ldots, l_K\}$.

Graph Convolutional Networks

GCN is an innovative convolution neural network for solving semi-supervised classification problems on nodes of a graph or network (Kipf and Welling 2016a). Borrowing the concept of convolution filter for image pixels or a linear array of signals, GCN defines a spectral graph convolution by multiplying a graph signal with a spectral filter (using the connectivity structure of the graph as the filter) in the Fourier domain. A classic two-layer GCN can be elegantly summarized by the following expression:

$$X^{(2)} = \text{softmax}\left(\hat{A}\,\text{ReLU}\left(\hat{A}XH^{(0)}\right)H^{(1)}\right) \quad (1)$$

where $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ (and $\tilde{A} = A + I_n$, A is the adjacency matrix and I_n the identity matrix, and $\tilde{D}$ is a diagonal matrix with $\tilde{d}_{ii} = \sum_j \tilde{a}_{ij}$) captures network topology, X is the attribute matrix, $H^{(0)}$ and $H^{(1)}$ are weight parameters of the two convolutional layers to be trained, and $X^{(2)}$ is the final output of GCN for the assignment of node labels.

MRFasGCN

MRFasGCN (Jin et al. 2019) infers a partition of nodes in a network by 1) using the original two convolutional layers of GCN as the first two layers of the neural network and 2) incorporating a community-oriented MRF model as the third convolutional layer in the framework of GCN. The central piece of this MRF model is an energy function, $E(C|A, X)$, which has two parts: the sum of unary potentials over all nodes and the sum of pairwise potentials over all edges:

$$\begin{aligned} E(C|A,X) &= \alpha * \sum_i \phi_u(c_i) + (1-\alpha) * \sum_{i\neq j} \phi_p(c_i, c_j) \\ &= \alpha * \sum_i -p(c_i) + (1-\alpha) * \sum_{i\neq j} \mu(c_i, c_j)\tau(v_i, v_j) \end{aligned} \quad (2)$$

where α is a parameter for balancing the unary and pairwise potentials. The unary potential $\phi_u(c_i) = -p(c_i)$ measures the cost for node v_i taking label c_i, where $p(c_i) = x^{(2)}_{i,c_i}$ is the probability that v_i has label c_i, which is derived from GCN in Eq. (1). The pairwise potential $\phi_p(c_i, c_j) = \mu(c_i, c_j)\,\tau(v_i, v_j)$ measures the cost for assigning labels c_i and c_j to nodes v_i and v_j, where $\mu(c_i, c_j)$ denotes the semantic similarities between communities c_i and c_j, and $\tau(v_i, v_j)$ denotes the attribute similarity or consistency between nodes v_i and v_j. They are defined as:

$$\mu(c_i, c_j) = (-1)^{\delta(c_i,c_j)} h^{(2)}_{c_i c_j} \quad (3)$$

$$\tau(v_i, v_j) = \beta * \xi(v_i, v_j) + (1-\beta) * R_i(\zeta(v_i, v_j)) \quad (4)$$

where $h^{(2)}_{c_i c_j}$ is the parameter to be learned, $\delta(c_i, c_j) = 1$ if $c_i = c_j$, or 0 otherwise, $\xi(v_i, v_j) = d_i d_j / 2e - a_{ij}$ (d_i is the degree of node v_i and e the number of edges), and β is a tradeoff parameter that balances topology and attributes. $\zeta(v_i, v_j)$ is defined by using the cosine similarity between the attributes of nodes v_i and v_j, and then an asymmetric regularization term is used to balance the difference of the sum of similarity on every node, i.e., $R_i(\zeta(v_i, v_j)) = \zeta(v_i, v_j) / \sum_{t=1}^{n} \zeta(v_i, v_t)$.

However, minimizing the above energy function in order to yield the most probable community partition for a given network is intractable, since the pairwise potentials are defined over a complete graph rather than the sparse network. Thus, a mean filed approximation is adopted to approximate the exact distribution $P(C|A, X)$. The updating procedure for this approximation has four steps: initialization, message passing, adding unary potentials, and normalization. Following these steps, MRFasGCN can transform the MRFs inference into a layer of convolutional process that is compatible with GCN, defined as:

$$Z = \text{softmax}\left(X^{(2)} - \Upsilon X^{(2)} H^{(2)}\right) \quad (5)$$

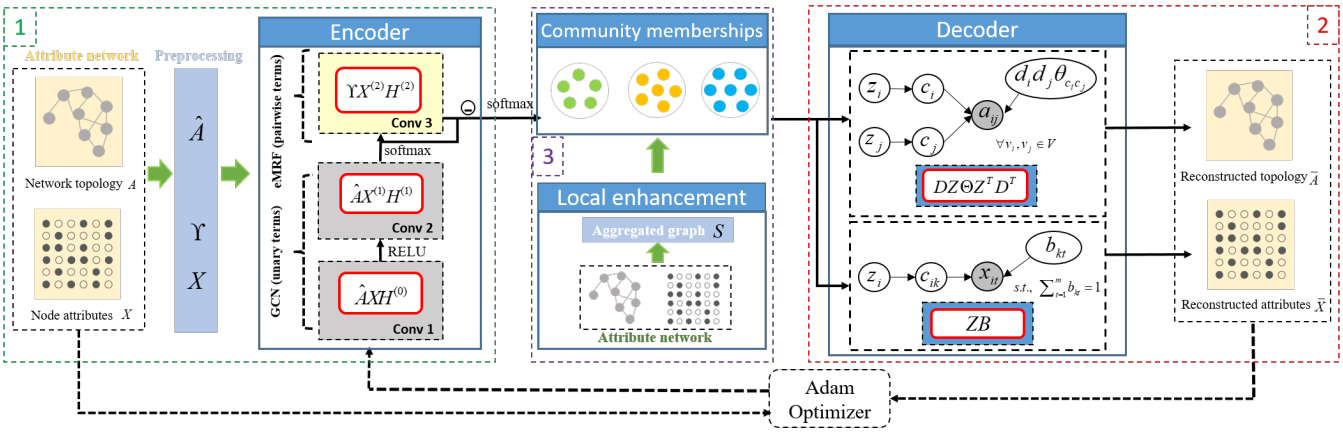

Figure 1: The architecture of GUCD. Part 1 to the left in the green box represents the encoder for deriving community membership of nodes. Part 2 to the right in the red box represents the dual decoder for reconstructing links and node attributes using community membership. Part 3 in the middle in the purple box represents the local structural enhancement on communities.

where $\Upsilon = (\tau(v_i, v_j))_{n\times n}$ is defined in Eq. (4), $X^{(2)}$ is from GCN defined in Eq. (1), $H^{(2)}$ is the weight parameters to be trained, and $Z = (z_{ic_i})_{n\times K}$ is the final community membership of nodes (where z_{ic_i} denotes the probability that node v_i belongs to community c_i). In essence, this is to take advantage of the MRF's pairwise potentials to make the new model community oriented and to perform a smooth refinement to the coarse results from GCN.

The Method

After briefly overview our new method, we will introduce an encoder based on MRFasGCN to derive node community memberships and a decoder for reconstructing links and attributes based on communities. We then discuss a local structural regularization to enhance community detection.

Overview

The new GCN-based approach for Unsupervised Community Detection, short-handed as GUCD, adopts Autoencoder as its overall architecture and includes three main parts (Fig. 1). In the first part (the green box to the left of Fig. 1), we adopted the three convolutional layers of MRFasGCN as the encoder of GUCD, where the first two layers were to learn a deep representation of the attribute network and the third layer was to model and derive node community membership using both the deep representation and network information. We then designed a dual decoder as the second part of GUCD (the red box to the right of Fig. 1), using the derived communities to separately reconstruct network topology and node attributes. We first reconstructed the network topology by requiring the nodes within the same community to maintain the same link pattern (with also considering the heterogeneity of node degrees) connecting to the rest of the network, which is suitable for generating the coupling relationships between nodes. We then generated node contents using topic modelling, i.e. we assumed that the contents of nodes in the same community share similar distributions of attribute words used. We formulated the dual decoder to fully utilize (structural and content) data from diverse sources, making the decoding process suitable for unsupervised community detection. In the third part (in purple box in the middle of Fig. 1), we added a regularization of local structural enhancement to the latent communities, i.e. we first constructed an aggregated graph combining the information of both topology and attributes, and then made each pair of connected nodes in this graph have similar community distributions. The model was trained as a whole using the Adam optimizer (Kingma and Ba 2015).

The Shared Encoder

We processed the network topology and node attributes together using the same encoder so as to extract hidden network characteristics, particularly latent community structures to be identified. To be specific, we adopted the three convolutional layers of MRFasGCN (Jin et al. 2019) as the encoder (the green box of Fig. 1). We used the first two layers to derive a general embedding and then derived an initial node community labels $X^{(2)}$ (defined in Eq. (1)) by using softmax on the embedding, which is not community-specific. We then used this initial solution $X^{(2)}$ to define unary potentials in MRF, and then defined pairwise potentials of MRF to model communities Z (defined in Eq. (5)) hidden in network topology and attributes, leading to the third convolutional layer. Through these three layers, the derived node community labels Z not only utilize the deep representation but also is smoothly community-oriented. It is important to note that the node to community assignments in Z will be learned when the whole model with encoder and decoder is trained together; the high quality of reconstructed network will require an accurate community structure at the end of the encoder.

The Dual Decoder

The dual decoder is the *core* part of the new method. It consists of two decoders, one for reconstructing network topol-

ogy and the other for reconstructing node attributes.

The decoder for reconstructing network topology. This novel decoder attempts to reconstruct the network topology based on the node community membership derived in the latent space. It not only makes the decoder community oriented, but also achieves an unsupervised learning for community detection. The idea is inspired by the block model for blocks, groups, or communities in networks. That is, if the community which the node v_i belongs to is denoted as c_i, we can then define a $K \times K$ block matrix Θ such that each element $\theta_{c_i c_j}$ in the matrix is the possibility of having an edge between nodes v_i and v_j. In this case, the nodes in each of the K communities (with label r) have the same link pattern, i.e., these nodes have the same link probability with any node v_j in the network, i.e. θ_{rc_j}. This idea naturally describes the coupling relationship between nodes in the network since it is defined to generate pairwise rather than individuals based on community structure. The model is further improved by considering the heterogeneity of node degrees, i.e. the nodes with higher degrees should be more likely to be connected. Therefore, the model can be revised such that the possibility that nodes v_i and v_j are connected is $d_i d_j \theta_{c_i c_j}$, where d_i is the degree of v_i. This mechanism is in concordance with the degree-corrected stochastic block model (DCSBM) (Karrer and Newman 2011) even though the formulation is different. Besides, a degree-corrected model can accommodate multi-edges and self-edges in the network. Similar to a random graph model of most sparse networks, the use of multi-edges and self-edges makes this model simpler and does not significantly affect the results.

Based on the above model, the expected number of links between nodes v_i and v_j, which respectively belong to communities c_i and c_j, can be written as

$$\overline{a}_{ij}^{c_i,c_j} = \text{sigmoid}\left(z_{ic_i} z_{jc_j} d_i d_j \theta_{c_i c_j}\right) \tag{6}$$

where z_{ic_i} is the probability that v_i belongs to community c_i, which is from the output of encoder, defined in Eq. (5). Considering all communities $\{(c_i, c_j) \,|c_i, c_j \in L\}$, we can then define the expected number of links between nodes v_i and v_j as:

$$\overline{a}_{ij} = \text{sigmoid}\left(\sum_{c_i=1}^{K}\sum_{c_j=1}^{K} z_{ic_i} z_{jc_j} d_i d_j \theta_{c_i c_j}\right) \tag{7}$$

By considering both the block modelling and heterogeneity degree of nodes, the model can well describe the coupling relationship among nodes in the network with community structures. The above model can be formulated as a layer of a neural network in order to incorporate it into the Autoencoder framework. As the K-dimensional vector $\vec{z}_i = (z_{ic_i})_{1\times K}$ denotes the probability distribution of v_i belonging to different communities, the link propensity between nodes v_i and v_j can be revised to $\overline{a}_{i,j} = \text{sigmoid}\left(d_i\left(\vec{z}_i \Theta \vec{z}_j^T\right) d_j\right)$,which can also be written in a matrix form as:

$$\overline{A} = \text{sigmoid}\left(DZ\Theta Z^T D^T\right) \tag{8}$$

where $D = \text{diag}\left(d_1, d_2, \cdots, d_n\right)$. Taking Θ as the weight parameters of the neural network, Eq. (8) can then be represented by a layer of the neural network, i.e., the topological decoder. The parameters Θ can be learned by minimizing the difference between the observed adjacency matrix A and the adjacency matrix $\overline{A}$ generated from the model. This difference can be defined by using the cross-entropy loss as:

$$L_{topo} = -\sum_{i,j=1}^{n}\left[a_{ij}\ln\overline{a}_{ij} + (1-a_{ij})\ln\left(1-\overline{a}_{ij}\right)\right] \tag{9}$$

The decoder for reconstructing attributes. Reconstruction of node attributes stems in topic modelling (Blei, Ng, and Jordan 2012). Nodes with similar attributes are believed to be more likely to belong to the same community. Therefore, the nodes in the same community are more likely to have similar distributions of attribute words, and different communities in a network can be characterized as representing different semantic topics.

Let $P(c_i|v_i) = z_{ic_i}$ be the probability that node v_i belongs to community c_i, where $Z = (z_{ic_i})_{n\times K}$ is the matrix of node community memberships from the encoder; and $P(x_{it} = 1|c_i = l_k) = b_{kt}$ be the probability that community/topic c_i selects an attribute word w_t from the entire word set $w_t \in W$, where $B = (b_{kt})_{K\times m}$ is to be learned. Assume that every node-attribute pair $\langle v_i, w_t\rangle$ in the context (with $x_{it} = 1$) is generated independently, where $v_i \in V$. Then, given the community c_i of node v_i, the probability for a node-attribute pair $\langle v_i, w_t\rangle$ will be:

$$\overline{x}_{it}^{c_i} = P\left(x_{it} = 1|c_i\right) P\left(c_i|v_i\right) \tag{10}$$

Considering all communities, the probability that $\langle v_i, w_t\rangle$ appears in the context will be:

$$\overline{x}_{it} = P\left(x_t|v_i\right) = \sum_{c_i\in L} P\left(x_{it} = 1|c_i\right) P\left(c_i|v_i\right) \tag{11}$$

The above generative process can be formulated as a layer of neural network to serve as an attributed decoder. Using $\overline{X} = (\overline{x}_{it})_{n\times m}$, the matrix form of the above model is:

$$\overline{X} = Z\cdot B \quad s.t., \quad \sum_{t=1}^{m} b_{kt} = 1 \tag{12}$$

where B denotes the weight parameters in this neural network layer to be trained. Then, the likelihood that the attribute matrix X is generated by the above model is:

$$\begin{aligned} P(X|V) &= \prod_{i=1}^{n}\prod_{t=1}^{m}\sum_{c_i\in L} P\left(x_{it} = 1|c_i\right) P\left(c_i|v_i\right) \\ &= \prod_{i=1}^{n}\prod_{t=1}^{m}\left(\sum_{c_i\in L} z_{ic_i} b_{c_i t}\right)^{x_{it}} = \prod_{i=1}^{n}\prod_{t=1}^{m}\left(\overline{x}_{it}\right)^{x_{it}} \end{aligned} \tag{13}$$

where $x_{it} = 1$ if node v_i has attribute w_t, or 0 otherwise. Since the objective of the network generative process is to maximize this likelihood, the negative log-likelihood is then taken as the loss function of this layer of the neural network,

$$L_{attr} = -\sum_{i=1}^{n}\sum_{t=1}^{m} x_{it}\ln\overline{x}_{it} \tag{14}$$

Local Enhancement

To improve community detection, we incorporated a scheme of local structural enhancement. The rationale underlying this scheme is that two nodes in a network are more likely to belong to the same community if they share more common neighbors or have more similar attributes. In other words, two nodes have similar node community membership if they are close to each other, topologically or semantically, in the attribute network.

We implemented this local enhancement scheme by introducing pairwise constraints on nodes and a graph regularization term to the objective function. We first built an aggregated graph based on network topology and node attributes. To measure the proximity of a pair of nodes, we introduced a measurement method to combine their topological and attribute similarities. We computed the set of the local neighbors for each node v_i,

$$\Gamma_i = \Gamma_i^{topo} \cup \Gamma_i^{attr} \tag{15}$$

where Γ_i^{topo} is the set of neighbors directly adjacent to v_i, and Γ_i^{attr} the set of top-k_{attr} most attribute-similar neighbors of v_i, where the TF-IDF cosine similarity is used to calculate the attribute similarity between two nodes. That is, let $\vec{x}_i$ be the attribute vector of node v_i, then for attribute unit x_{it}, its TF-IDF value in a term vector $\vec{x}_i$ is:

$$\begin{aligned} x'_{it} &= tf - idf\left(x_{it}, \vec{x}_i\right) \\ &= \sqrt{tf\left(x_{it}, \vec{x}_i\right)} \cdot \log\left(1 + \frac{|V|}{\sum_{j=1}^{|V|} tf\left(x_{it}, \vec{x}_j\right)}\right) \end{aligned} \tag{16}$$

The cosine similarity of attributes between nodes v_i and v_j is then defined as:

$$cosine\left(\vec{x}'_i, \vec{x}'_j\right) = \frac{\vec{x}'_i \cdot \vec{x}'_j}{\|\vec{x}'_i\| \cdot \|\vec{x}'_j\|} \tag{17}$$

Thereafter, we added weights to the aggregated graph. We calculated the proximity between a node and its direct neighbors in this new graph using:

$$S = \lambda zero{-}one\left(S^{topo}\right) + (1-\lambda) zero{-}one\left(S^{attr}\right) \tag{18}$$

where $S^{topo} = \left[s_{ij}^{topo}\right]$ (with $s_{ij}^{topo} = cosine\left(\vec{a}_i, \vec{a}_j\right)$) and $S^{attr} = \left[s_{ij}^{attr}\right]$ (with $s_{ij}^{attr} = cosine\left(\vec{x}_i, \vec{x}_j\right)$) are respectively the topological similarity and attribute similarity between node v_i and its direct neighbor $v_j \in \Gamma_i$, λ is a tradeoff parameter, and $zero{-}one(\cdot)$ plays the role of normalization which is defined as:

$$zero{-}one(\vec{y}) = \left(y_i - \min(\vec{y})\right) / \left(\max(\vec{y}) - \min(\vec{y})\right) \tag{19}$$

Given the similarity matrix S of the aggregated graph, we dened the pairwise constraint as:

$$L_{reg} = \sum_{i,j} s_{ij} \left\|\vec{z}_i - \vec{z}_j\right\|_2^2 = 2\,\mathrm{tr}\left(Z^T \Psi Z\right) \tag{20}$$

where $\Psi = F - S$ and F is a diagonal matrix with $f_{ii} = \sum_j s_{i,j}$. We then incorporated the pairwise constraint as a graph regularization term in the objective function to enhance community detection.

Algorithm 1 The algorithmic process of GUCD.

Input: Attribute network $G = (V, E, W)$ with adjacency matrix A and attribute matrix X
Output: Community partition C

```
Calculate similarity matrix K via (4) and pairwise constraint matrix S via (18)
Initialize parameters H = {H^(0), H^(1), H^(2), Θ, B}
Repeat
  Apply (1) & (5) in the encoder to derive the node community membership matrix Z (using A, X, ϒ, H)
  Apply (9) & (14) in the dual decoder to obtain the losses L_topo & L_attr of reconstructing links and attributes, respectively (using Z, Θ, B)
  Apply (20) to calculate the loss of pairwise constraint L_reg in local enhancement (using Z, S)
  Calculate the complete loss function via (21)
  Apply Adam optimizer with BP algorithm to update the neural network parameters H
Until converge or reach maximum number of iterations
Apply (22) to derive the community partition c
```

Finally, the objective of the model is to minimize the following objective function:

$$L = \gamma L_{topo} + (1-\gamma) L_{attr} + \eta L_{reg} \tag{21}$$

where γ and η are the parameters that control the tradeoff between different parts in the loss functions. Last, we used the back-propagation (BP) algorithm and Adam optimizer to train the model as done in MRFasGCN (Jin et al. 2019). The algorithm of GUCD is shown in Algorithm 1. At convergence, the algorithm produces the community label for each node:

$$\hat{c}_i = \arg\max_{c_i \in L} \quad z_{ic_i} \tag{22}$$

Experiments

We compared our approach with six state-of-the-art methods on nine widely-used benchmark datasets. We also analyze the features of our method to appreciate its effectiveness.

Experiment Setup

In order to carry out an accurate comparison of the seven methods, we used nine public datasets with known community structures (Table 1). Because the networks used have ground truth communities, we adopted two widely used performance metrics, i.e. accuracy (AC) (Liu et al. 2012) and normalized mutual information (NMI) (Danon et al. 2005), for performance evaluation.

We applied the widely-used Adam optimizer in our GUCD method using the TensorFlow framework. For training, we uniformly set the learning rate to 0.01, the maximum epoch to 1,000 and the balance parameters to $\alpha = \beta = 0.5$. We terminated the training process when the loss function failed to decrease in 10 consecutive epochs. For the encoder, we used the same number of hidden units and initialization to the weights as the three layers of MRFasGCN (except that we uniformly set the number of hidden units at the first layer

Table 1: Datasets descriptions.

Datasets	Nodes	Edges	Communities	Attributes
Texas	183	328	5	1,703
Cornell	195	304	5	1,703
Washington	217	446	5	1,703
Wisconsin	262	530	5	1,703
Twitter	171	796	7	578
Cora	2,708	5,429	7	1,433
Citeseer	3,312	4,732	6	3,703
UAI2010	3,363	45,006	19	4,972
Pubmed	19,729	44,338	3	500

32). For the decoder, we used the uniform distribution to initialize the weights.

Comparison with the Existing Methods

We first evaluated the GUCD method against six existing (unsupervised) community detection methods. Depending on what network information that they use, the existing methods can be grouped into three types. The first type uses only network topology, which includes DCSBM (Karrer and Newman 2011). The second uses only node attribute information, including LDA (Blei, Ng, and Jordan 2012). The third type uses network topology and node attributes together, including Block-LDA (Balasubramanyan and Cohen 2011), PCLDC (Yang et al. 2009), SCI (Wang et al. 2016) and TLSC (Zhang et al. 2018). All of the seven methods that we compared need to have the number of communities specified, which was set to the actual number of communities in our experiments. We adopted AC and NMI for performance measure for comparison against the ground-truth.

GUCD performs the best on 7 and 7 out of 9 datasets in terms of AC (Table 2) and NMI (Table 3), respectively. On the remaining networks where GUCD does not have the best performance, it still has the second best results and is competitive with the best baselines. To be specific, GUCD is on average 24.82%, 19.97%, 24.01%, 24.29%, 16.54% and 11.71% more accurate than DCSBM, LDA, Block-LDA, PCLDC, SCI and TLSC in terms of AC; and 20.49%, 11.97%, 33.89%, 21.19%, 21.08% and 14.02% more accurate than these methods in terms of NMI. These results demonstrate the superiority of our new approach over the existing methods.

Table 2: Comparison of the seven community detection methods in terms of AC. Bold font indicates the best result. The number in brackets denotes the rank of this algorithm.

Datasets	AC (%)						
	DCSBM	LDA	Block-LDA	PCLDC	SCI	TLSC	GUCD
Texas	48.09	56.28	54.10	38.80	62.30	65.02	**67.76**
Cornell	37.95	44.62	46.15	30.26	45.64	47.69	**71.28**
Washington	31.8	44.62	39.17	29.95	51.15	51.61	**73.85**
Wsicsonsin	32.82	44.62	49.62	30.15	50.38	49.23	**76.72**
Twitter	60.49	37.04	35.80	56.79	50.62	**62.87**	60.49 (2)
Cora	38.48	37.19	25.52	34.08	40.62	47.62	**50.59**
Citeseer	26.57	31.34	24.35	24.85	27.98	35.74	**54.47**
UAI2010	2.60	34.07	16.04	28.82	30.94	29.28	**37.58**
Pubmed	53.64	46.30	49.01	**63.55**	47.39	61.38	63.13 (2)
AVG	36.94	41.79	37.75	37.47	45.22	50.05	**61.76**

Table 3: Comparison of the seven community detection methods in terms of NMI.

Datasets	NMI (%)						
	DCSBM	LDA	Block-LDA	PCLDC	SCI	TLSC	GUCD
Texas	16.65	31.29	4.21	10.37	17.84	23.92	**29.65**
Cornell	9.69	21.09	6.81	7.23	11.44	13.61	**43.85**
Washington	9.87	38.48	3.69	5.66	12.37	17.63	**46.67**
Wsicsonsin	3.14	46.56	10.09	5.01	17.03	16.65	**47.56**
Twitter	57.48	31.10	0.00	52.64	43.00	49.14	**58.14**
Cora	17.07	14.61	2.42	17.54	19.26	**33.20**	32.33 (2)
Citeseer	4.13	9.13	1.41	2.99	4.87	23.16	**27.43**
UAI2010	31.21	**35.42**	5.70	26.92	24.80	22.87	33.35 (2)
Pubmed	12.28	10.55	6.58	26.84	5.59	19.63	**26.98**
AVG	17.95	26.47	4.55	17.25	17.36	24.42	**38.44**

Deep Analysis of GUCD

Similar to most existing deep learning algorithms, GUCD has multiple components that can significant impact on its performance. Moreover, MRFasGCN is a major component of GUCD and can be applied in different ways. Here we report the results from different combinations of different major components of GUCD.

Effects of individual components. We compared GUCD with six variations of GUCD. We first considered two unsupervised variants of MRFasGCN, a major component of GUCD: 1) the weight parameters $H^{(0)}$, $H^{(1)}$ and $H^{(2)}$ in MRFasGCN being set to 1 without training (as the most general way), which was named as MasG-U1, and 2) the weight parameters being set randomly without training (as suggested as an unsupervised usage in (Kipf and Welling 2016a)), which was named as MasG-U2. We included in our comparison another unsupervised variant of MRFasGCN by adding the inner product of node community membership for reconstructing network topology as the decoder, which was named as MasG-IN. (Note that this is the most common way used in the unsupervised GCN models for network embedding, a different albeit similar problem.) We also tested three direct variants of GUCD. The first two reconstruct network topology and node attributes, separately, named GUCD-1 and GUCD-2, respectively. The third is GUCD without the local structural enhancement, named as GUCD-3.

The experimental results on the ten benchmark problems revealed that GUCD performed the best on 7 and 7 out of the 9 networks in terms of AC (Table 4) and NMI (Table 5), respectively. Specifically, GUCD is on average 17.45% (and 24.49%) and 23.99% (and 29.18%) more ac-

Table 4: The comparison of our GUCD with the six variants (MasG-U1, MasG-U2, MasG-IN, GUCD-1, GUCD-2 and GUCD-3) in terms of AC.

Datasets	AC (%)						
	MasG-U1	MasG-U2	MasG-IN	GUCD-1	GUCD-2	GUCD-3	GUCD
Texas	57.22	56.99	57.30	61.62	**72.28**	69.57	67.76 (3)
Cornell	43.72	43.43	44.16	47.47	70.26	67.18	**71.28**
Washington	48.42	48.42	63.76	66.36	71.43	72.02	**73.85**
Wsicsonsin	54.17	46.04	46.97	53.21	71.76	73.66	**76.72**
Twitter	55.30	31.00	51.85	50.00	55.56	56.79	**60.49**
Cora	34.96	32.50	36.40	37.27	49.17	**51.29**	50.59 (2)
Citeseer	48.81	24.86	32.49	28.60	48.18	52.46	**54.47**
UAI2010	16.25	16.72	16.34	15.96	34.00	36.12	**37.58**
Pubmed	39.98	39.95	50.84	51.41	62.09	61.33	**63.13**
AVG	44.31	37.77	44.46	45.77	59.41	60.05	**61.76**

Table 5: The comparison of our GUCD with the six variants in terms of NMI.

Datasets	NMI (%)						
	MasG-U1	MasG-U2	MasG-IN	GUCD-1	GUCD-2	GUCD-3	GUCD
Texas	14.34	15.32	17.29	20.60	**43.45**	34.35	29.65 (3)
Cornell	9.33	9.05	6.80	9.22	41.65	38.10	**43.85**
Washington	10.35	10.35	24.25	28.07	46.16	37.59	**46.67**
Wsicsonsin	12.36	8.48	6.52	8.50	44.81	42.78	**47.56**
Twitter	38.02	21.36	48.60	35.18	55.36	47.60	**58.14**
Cora	13.03	7.08	12.43	19.38	27.31	29.76	**32.33**
Citeseer	22.43	5.74	12.07	9.08	21.40	27.03	**27.43**
UAI2010	5.35	5.61	5.50	5.42	32.50	**33.90**	33.35 (2)
Pubmed	0.34	0.37	9.55	8.89	23.76	24.27	**26.98**
AVG	13.95	9.26	15.89	16.04	37.38	35.04	**38.44**

curate than MasG-U1 and MasG-U2 in AC (and NMI). By some extra experiments we also observed that, to achieve the similar performances with GUCD, MRFasGCN needs almost 30%, 30%, 30%, 30%, 20%, 0.2%, 0.8%, 2% and 0.05% supervised information for the nine datasets in training. This validates not only the effectiveness of our unsupervised framework based on Autoencoder, but also the soundness of the community-oriented dual decoder, i.e. we use different while the most suitable mechanisms to reconstruct links and attributes respectively for community detection. Besides, GUCD is on average 17.3% (and 22.55%) more accurate than MasG-IN in AC (and NMI). This further validates the soundness of the new decoding mechanisms (i.e. we reconstruct links by modelling that nodes in the same community share the same link pattern, and reconstruct attributes based on topic modelling) over the existing methods for reconstructing links alone by using the inner product operation. In addition, GUCD is on average 15.99% (and 22.4%), 2.35% (and 1.06%), and 1.71% (and 3.4%) more accurate than GUCD-1, GUCD-2 and GUCD-3 in AC (and NMI). This demonstrates that the topological and attributed decoders both are effective. In addition, the strategy for local structural enhancement also helps meliorate overfitting caused by the sparsity of networked data.

Qualitative Analysis. To further validate the effective of our new decoding mechanism over the existing ones commonly used in network embedding, we further compared GUCD and MasG-IN by showing the results of AC and NMI as a function of the number of iterations. Consider the results on Twitter as an example. As shown in Fig. 2, the results of GUCD match more closely to the ground truth communities than MasG-IN and the result from former is more stable than that from the latter as training cycle increases. This suggests that the new decoding mechanisms are indeed suitable for unsupervised community detection.

Related Work

The current work of Graph Convolutional Networks (GCN) focuses primarily on node classification (Kipf and Welling 2016a), which has also been adopted for semi-supervised community detection (Li, Han, and Wu 2018; Jin et al. 2019). However, community detection is in essence an unsupervised problem since little training data are available for most applications. Therefore, it is imperative to develop novel GCN methods for unsupervised network community identification.

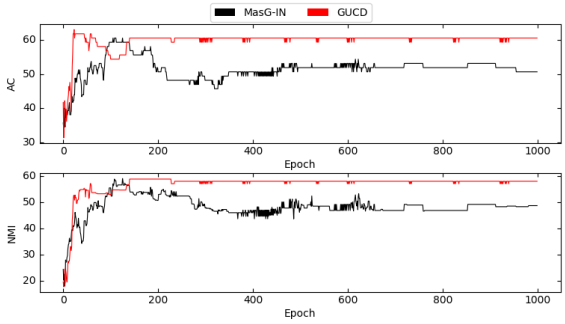

Figure 2: The values of AC (and NMI) as a function of the number of iterations on Twitter.

A line of work related to our GUCD method is network embedding using GCN. For example, the GAE method (Kipf and Welling 2016b) incorporates GCN into the Autoencoder framework and uses GCN as an encoder to derive a low-dimensional representation and embedding of a network, and then uses the inner product of the embedding to reconstruct links of the network in the decoder. The ARGA method (Pan et al. 2018) adopts an adversarially regularized graph Autoencoder, which uses the same encoder and decoder as GAE while introduces an adversarial module to force the node representation to follow a suitable prior distribution. The ARGA method has also been extended to use graph convolution, instead of inner product, as the decoder to reconstruct links; the resulting method is able to reconstruct links better than reconstructing both links and attributes (Pan et al. 2019a). For unsupervised learning, most of the existing methods focus primarily on reconstructing network structures alone, as their performance degrades when being used to reconstruct links and attributes together. All these existing methods seem to be suitable for representation learning tasks, such as embedding, but not adequate for unsupervised community detection, a difficult category learning task. This observation was experimentally shown by the comparison between GUCD and its variant MasG-IN (using the inner produce approach to reconstruct links alone), as shown in Fig. 2 and Tables 4 & 5.

Conclusion and Discussion

This is the first approach extending GCN effectively to unsupervised community detection. Using the Autoencoder framework, we focused on community identification rather than network embedding in the encoder; and used the most suitable mechanisms to reconstruct links and attributes separately. We also introduced a first-order structural enhancement to meliorate the issue of overfitting caused by the sparsity of networked data. We carried out experiments and demonstrated the superiority of the new approach. We further showed the effectiveness of the novel decoding mechanism for generating links and attributes over the most commonly used methods for reconstructing links alone. Even though the new approach was designed for community detection, the underlying idea can be readily extended to GCN-based network embedding as discussed in Related Work.

References

Adamic, L. A., and Adar, E. 2003. Friends and neighbors on the web. *Social Networks* 25(3):211–230.

Balasubramanyan, R., and Cohen, W. W. 2011. Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *SDM*.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2012. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Chen, Z.; Li, X.; and Bruna, J. 2018. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*.

Danon, L.; Díaz-Guilera, A.; Duch, J.; and Arenas, A. 2005. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* 2005(09):P09008–P09008.

Falih, I.; Grozavu, N.; Kanawati, R.; and Bennani, Y. 2018. Community detection in attributed network. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, 1299–1306.

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

Fortunato, S. 2010. Community detection in graphs. *Physics Reports* 486:75 – 174.

Jin, D.; Ge, M.; Yang, L.; He, D.; Wang, L.; and Zhang, W. 2018. Integrative network embedding via deep joint reconstruction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 3407–3413.

Jin, D.; Liu, Z.; Li, W.; He, D.; and Zhang, W. 2019. Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In *AAAI*.

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

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Kipf, T. N., and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. *CoRR* abs/1609.02907.

Kipf, T. N., and Welling, M. 2016b. Variational graph auto-encoders. *CoRR* abs/1611.07308.

Li, H.; Yan, G.; Liu, Z.; Li, G.; and Zhang, X. 2017. A linear community detection algorithm based on dynamical system in networks. *SCIENTIA SINICA Mathematica* 47(2):241–256.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR* abs/1801.07606.

Liu, H.; Wu, Z.; Cai, D.; and Huang, T. S. 2012. Constrained nonnegative matrix factorization for image representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 34(7):1299–1311.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 2609–2615. International Joint Conferences on Artificial Intelligence Organization.

Pan, S.; Hu, R.; fu Fung, S.; Long, G.; Jiang, J.; and Zhang, C. 2019a. Learning graph embedding with adversarial training methods.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2019b. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*.

Wang, X.; Jin, D.; Cao, X.; Yang, L.; and Zhang, W. 2016. Semantic community identification in large attribute networks. In *AAAI*.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *AAAI*.

Yang, T.; Jin, R.; Chi, Y.; and Zhu, S. 2009. Combining link and content for community detection: A discriminative approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, 927–936. ACM.

Yang, L.; Cao, X.; He, D.; Wang, C.; Wang, X.; and Zhang, W. 2016. Modularity based community detection with deep learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 2252–2258. AAAI Press.

Zhang, G.; Jin, D.; Gao, J.; Jiao, P.; Soulie Fogelman, F.; and Huang, X. 2018. Finding communities with hierarchical semantics by distinguishing general and specialized topics. 3648–3654.