Learning Community Structure with Variational Autoencoder

Jun Jin Choong
Department of Computer Science
Tokyo Institute of Technology
Tokyo, Japan
junjin.choong@net.c.titech.ac.jp

Xin Liu
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
xin.liu@aist.go.jp

Tsuyoshi Murata
Department of Computer Science
Tokyo Institute of Technology
Tokyo, Japan
murata@c.titech.ac.jp

***Abstract*—Discovering *community structure* in networks remains a fundamentally challenging task. From scientific domains such as biology, chemistry and physics to social networks the challenge of identifying community structures in different kinds of network is challenging since there is no universal definition of community structure. Furthermore, with the surge of social networks, content information has played a pivotal role in defining community structure, demanding techniques beyond its traditional approach. Recently, network representation learning have shown tremendous promise. Leveraging on recent advances in deep learning, one can exploit deep learning's superiority to a network problem. Most predominantly, successes in supervised and semi-supervised task has shown promising results in network representation learning tasks such as link prediction and graph classification. However, much has yet to be explored in the literature of community detection which is an unsupervised learning task. This paper proposes a deep generative model for community detection and network generation. Empowered with Bayesian deep learning, deep generative models are capable of exploiting non-linearities while giving insights in terms of uncertainty. Hence, this paper proposes Variational Graph Autoencoder for Community Detection (VGAECD). Extensive experiment shows that it is capable of outperforming existing state-of-the-art methods. The generalization of the proposed model also allows the model to be considered as a graph generator. Additionally, unlike traditional methods, the proposed model does not require a predefined community structure definition. Instead, it assumes the existence of latent similarity between nodes and allows the model to find these similarities through an automatic model selection process. Optionally, it is capable of exploiting feature-rich information of a network such as node content, further increasing its performance.**

***Index Terms*—community detection, variational autoencoder, generative model**

I. Introduction

Real-world complex systems are often projected into networks to observe complex patterns. Entities in a complex system can be represented as nodes (vertices) and their interactions represented as an edge (link). For instance, social interactions between people can be represented in the form of a social network. Publications by authors and their respective publication venues can be represented with a bipartite citation network. The flexibility of networks and its vast literature on graph theory makes networks very appealing to researchers. Although networks are merely represented in forms of nodes and edges, a large complex system could easily scale from hundreds to millions of nodes and edges. This poses a very challenging task in machine learning, especially tasks such as graph clustering or more commonly known as community detection [1] in the literature of network science. Given a network (graph) with its node content and structural (link) information, community detection aims to partition the nodes in the network into a number of disjoint groups. These partitions can be formulated depending on the given definition. For example, in modularity maximization [2], each partition is compared against a null model (random network). A partition is classified as good when the modularity score is greater than partitioning a random network. On the other hand, statistical methods such as the Stochastic Blockmodel (SBM) introduced Bayesian treatment of uncertainty when partitioning the network. Nodes with similar statistical similarity have higher probability to cluster together regardless of the cluster's density [3]. This is known as stochastic equivalence. In general, a universal definition of community structure does not exist. Nevertheless, the objective remains the same, i.e. to find a group of nodes that shares some form of similarity between one another. In this paper, such similarity is defined as latent similarity; the similarity measure is not predefined. Quantifying such similarity is arguably subjective and difficult especially when a given network can be feature-rich or structure-only; there is no one-size-fit-all solution for community detection (i.e. the *no free lunch theorem*). Therefore, it is essential that algorithms capture both higher-order information and structural information. To this end, we look at network representation learning as a potential solution.

In machine learning, representation learning has been successfully applied to various fields such as natural language processing and computer vision. Notably, successes of deep learning have surpassed human accuracy with ease [4]. However, these successes are difficult to be explained. More precisely, it is difficult to explain "why" deep learning model performs so well. In an attempt to solve this problem, researchers bridged the understanding gap by introducing probabilistic deep models (also known as Bayesian Deep Learning) [5]. Using fundamental building blocks from a probabilistic perspective, assumptions are given in forms of non-informative priors and the model is forced to correct these assumptions while learning. Consequently, it becomes less ambiguous than a typical deep learning model which is commonly known to be a black-box.

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00022

Stemming from the principals of representation learning, network representation learning targets a similar objective, but from a network perspective. Given a network, the objective is to find a latent representation that generalizes for various machine learning tasks such as classification, link prediction and clustering of nodes. Generally, a common choice for finding community structure in networks often involve a two-step approach. First, the network is embedded into a latent space (i.e. Euclidean space). Next, a general clustering algorithm such as Spectral Clustering [6] or k-means is applied to the learned embedding. For instance, Tian *et al.* proposed a network representation [7] learning model to learn a non-linear mapping of the original network using a Stacked Autoencoder by showing that spectral clustering and Autoencoders have the same optimization objectives. Yang *et al.* considers a Stacked Autoencoder as a modularity optimization problem and further introduced a semi-supervised approach through *must-pair* nodes for increased performance [8]. Assignment of communities are then obtained through k-means clustering from the latent representation that exhibits the highest modularity score. Inspired from Denoising Autoencoders [9], Wang *et al.* proposed Marginalized Graph Autoencoder for Graph Clustering (MGAE) [10] that artificially corrupts the feature matrix to increase the number of training data and provides a close-form solution for optimization. Spectral Clustering is then applied to the learned latent representation. Clearly, these methods all employ a two-step approach which is unsuitable for studying network generation or graph modeling [11].

Instead of a costly two-step approach and ignoring uncertainty in the modeling process, the problem can be solved from a Bayesian point of view; by encoding our latent beliefs and assumptions as probabilistic graphical models. Specifically, one can assume that nodes and edges are modeled from a mixture model such as the Gaussian Mixture Model (GMM). This effectively couples the learning of cluster assignment with respect to its network representation into a joint probability distribution. Additionally, it helps to capture network properties exhibited by common networks which consequently helps in better understanding of real-world networks.

Concretely, this paper proposes an extension to Variational Graph Autoencoder (VGAE) [12]. Originally, VGAE projects graph convolutions into a Univariate Gaussian latent space and has only been considered for semi-supervised task such as link prediction and graph classification. The proposed model, VGAECD relaxes this notion by introducing a Mixture of Gaussian. This is desirable as we would like to capture higher-order patterns from community structures and model its generative process. It is worth noting that, similar approaches have been applied to VAE in domains such as image recognition [13]. However, these approaches are not readily applicable for networks, especially in a community detection problem.

To summarize, this paper explores the idea of learning network representations using Bayesian treatment. We extend VGAE to include clustering-aware capability specifically targeting a community detection task. The contribution of this paper is summarized as follows:

- This paper proposes a novel generative model for community detection which is agnostic to the necessity of a predefined community structure definition. Through the process of automatic model selection, nodes are assigned a community based on the criterion that best reduces the loss function.
- The proposed model inherits the benefits of Variational Autoencoder Framework. The advantages are three-fold, (1) it provides a variational lower bound which is guaranteed to converge to a local minimum, (2) the lower bound is scalable and (3) the model is generative, allowing generation of synthetic networks.
- The proposed model outperforms the state-of-the-art models in community detection without requiring additional priors (unlike the Degree-Corrected SBM).

II. Problem Definition

A network pertaining to nodes, edges and node features can be formally defined as $G = (V, E, X)$, where $V = \{v_i, ..., v_N\}$ consists of a set of nodes $|V| = N$, $E = \{e_{ij}\}$ is a set of edges, and $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node features. Each $\mathbf{x}_i \in \mathbb{R}^d$ defines a vector of real-values associated with node v_i. From an Autoencoder's perspective, the inputs are given in terms of structural information $A \in \mathbb{R}^{N \times N}$, and node features $X \in \mathbb{R}^{N \times d}$, where A denotes the adjacency matrix of G and the node features are content information provided in forms of vector representation. In this work, we consider the undirected and unweighted network G, such that $A_{ij} = 1$ if $e_{ij} \in E$ otherwise 0.

Given the network G, the objective of community detection or graph clustering is to partition the nodes in G into K disjoint groups $\{c_1, c_2, ..., c_K\}$, such that nodes grouped within the same cluster are close to each other while nodes in different clusters are distant in terms of network structure. Vertices grouped within the same cluster are more likely to have similarities in node features.

Additionally, we consider the definition of a generative model. The discriminative model, $p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{A})$ infers the model parameters $\boldsymbol{\theta}$ from the observed network G. Subsequently, a network G' can be generated from a the same set of parameters. Concretely, $p(\mathbf{A} \mid \boldsymbol{\theta}) = G'$. Under the model selection criterion, the model is said to be a good model when $G' \approxeq G$ and satisfies the condition of having community structures i.e, G' is not an Erdős-Rényi network. By definition, generative models can be considered as an ensemble learning model.

III. Related Work

Recent work in community detection can be broadly categorized into two types of model; namely, discriminative and generative models. The former includes a class of methods that infers communities given an observed network and optionally, node features. Meanwhile, the latter considers the reconstruction of network while exploring plausible models that explains the observed phenomenon.

A. Discriminative Methods & Models

Predominantly, modularity maximization [2, 14] has been considered as the most successful method for detecting communities. However, it suffers from a resolution limit problem [15] and is known to exhibit degeneracies [16]. In terms of speed, label propagation [17] is capable of detecting communities in large-scale networks near linear time. Though, the solutions are usually non-unique. Other representation learning methods such as GraRep [18] considers the completion of its adjacency matrix and can be generally considered as matrix factorization problem. Meanwhile, others like DeepWalk [19], and node2vec [20] considers representation of each node via a bias random walk. It assumes that neighboring nodes share similarities from the pivot node. Hence, when nodes are clustered together, they tend to co-occur on short random walks over the network.

Besides standard linear methods mentioned previously, recent advances in deep learning revisited Autoencoders for networks. Particularly, GraphEncoder proposed by Tian *et al.* shows that optimizing the objective function of Autoencoder is similar to finding a solution for Spectral Clustering [7]. Leveraging on deep learning's non-linearity and recent advances in Convolutional Neural Networks, [21, 22] proposed the Graph Neural Network (GNN) and its generalization, the Graph Convolutional Neural Network (GCN) [21]. Defferrard *et al.* first cast the problem by projecting graph convolutions into spectral space, and convolving within this space.

B. Generative Methods & Models

Generative models can be further subdivided into algorithmic and statistical types. Examples of algorithmic models include the Kronecker Graphs [23] and Block Two-Level Erdős-Rényi (BTER) model [24]. One of the most popular generative models for capturing networks with group structure is the Stochastic Blockmodels (SBM) or also known as the planted partition model. First explored by Snijders *et al.* [25] two decades ago, the key idea behind SBM is stochastic equivalence. The probability that two nodes i and j are connected depends exclusively on their group memberships; two nodes within a cluster sharing the same stochasticity. However, the vanilla SBM exhibits a problem where high degree nodes are clustered into a community of its own. Karrer *et al.* proposed the Degree Corrected (D.C.) SBM [26] which introduces a normalizing prior. Extensions to SBM includes the Mixed Membership SBM (MMSBM) [27] for identifying mix community participation and bipartite SBM (biSBM) [28] for finding communities in bipartite networks. Today, SBM is well explored and its limitations has been widely studied [29, 30]. However, SBM is not a network representation learning model. Instead, SBM learns the latent variables $\mathbf{\Pi}$ and $\mathbf{Z}$ which describe the probabilities of cluster connectivity and cluster assignment of a particular node which differs from common representation learning method.

Unlike SBM, although Autoencoders consists of two parts (encoder and decoder), the representation learned cannot be generalized for generation of networks. One of the most recent work includes Variational Autoencoder specifically for graphs, i.e. Kipf *et al.* introduced a variant of Autoencoder for network link prediction that is generative. This is discussed in detail in section IV-A.

IV. Proposed Method

Algorithm 1 VGAECD

Input: Features $\mathbf{X}$, Adjacency matrix $\mathbf{A}$,
Hyperparameters: learning rate ϵ, epochs L, size of layer 1 and 2.
Output: Community Assignment Probability $\boldsymbol{\gamma}$ and Reconstructed Adjacency matrix $\tilde{\mathbf{A}}$

```
π ~ U(0, 1)
for l = 1, ..., L do
    for i = 1, ..., N do
        μ_i = GCN_μ(x_i, a_i)
        σ_i = GCN_σ(x_i, a_i)
        Sample c ~ Cat(c | π)
        Sample z_i ~ N(μ_c|_i, diag(σ²_c|_i))
        Obtain reconstructed ã_i = σ(z_i^⊤ z_j)
        Compute loss, L_ELBO                    ▷ From (14)
            and backpropagate gradients.
    end for
end for
Extract community assignment γ via z_i          ▷ From (18)
Return γ, Ã = {ã_1, ..., ã_N}
```

A. Variational Graph Autoencoder

Variational Graph Autoencoder (VGAE) [12] extends the problem of learning network embedding to a generative perspective by leveraging on the Variational Autoencoder (VAE) framework [31]. Consider a given network G with structural information $\mathbf{A}$ and node features $\mathbf{X}$, the inference model of VGAE parameterized by a two-layer GCN is defined as,

$$q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}) \tag{1}$$

$$q(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)). \tag{2}$$

Here, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the mean and standard deviation vectors for node i which is obtained from a GCN layer, $\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. The two-layer GCN is then defined as,

$$\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \hat{\mathbf{A}}\sigma(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1, \tag{3}$$

with $\mathbf{W}_0$ and $\mathbf{W}_1$ representing the weight matrices for the first layer and second layer respectively. $\mathbf{W}_0$ is shared between $\mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. $\sigma(\cdot)$ is a non-linear function such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ or $\mathrm{sigmoid}(t) = 1/(1 + e^{-t})$. $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ denotes the symmetric normalized adjacency matrix. The generative model is simply the inner product between the latent variables,

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij} = 1 \mid \mathbf{z}_i\mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j). \tag{4}$$

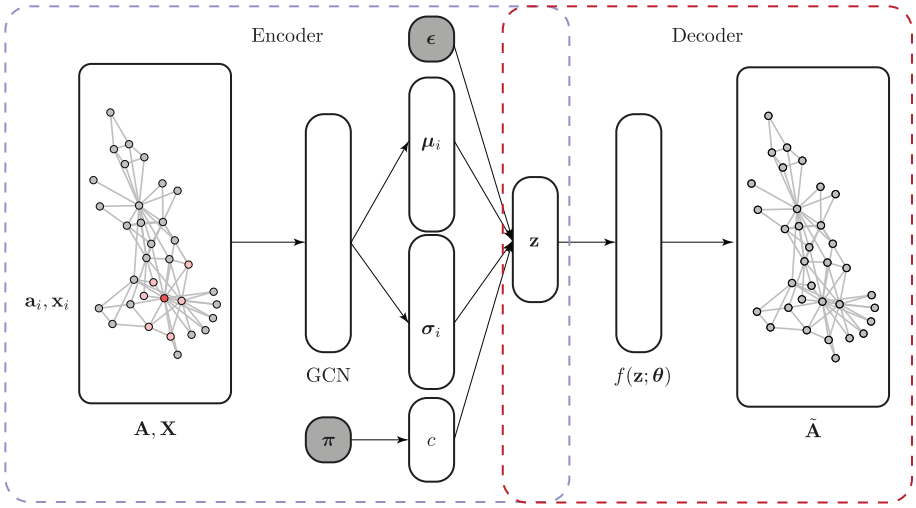


Fig. 1: Conceptual illustration of Variational Graph Autoencoder Framework for Community Detection (VGAECD). In the encoding phase, VGAECD first convolves on the network, learning structural and nodal features in the process. These information are then mapped into a latent representation, $\boldsymbol{\mu}_c|_i$ and $\boldsymbol{\sigma}_c|_i$ which are parameters to Mixture of Gaussian Model which is subsequently sampled to obtain a latent representation for each node $\mathbf{z}$. Finally, $\tilde{\mathbf{A}}$ can be reconstructed using a decoding function, $f(\cdot)$. The loss is calculated and backpropagated to the latent variables.

In accordance to the VAE framework, both models can be tied together and optimized by maximizing the variational lower bound $\mathcal{L}(\cdot)$,

$$\log p_{\boldsymbol{\theta}}(\mathbf{X}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{Z}|\mathbf{X},\mathbf{A})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{A} \mid \mathbf{Z})\right] - D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{Z} \mid \mathbf{X}) \parallel p_{\boldsymbol{\theta}}(\mathbf{Z})]. \tag{5}$$

$D_{KL}[q_{\boldsymbol{\phi}}(\cdot) \parallel p_{\boldsymbol{\theta}}(\cdot)]$ defines the Kullback-Leibler (KL) divergence between $q_{\boldsymbol{\phi}}(\cdot)$ and $p_{\boldsymbol{\theta}}(\cdot)$. The lower bound can be maximized with respect to the variational parameters $(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbf{W}_i$ via stochastic gradient descent, performed with a full-batch size. Here, the prior is define as $p_{\boldsymbol{\theta}}(\mathbf{Z}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i \mid 0, \mathbf{I})$, which is the isotropic Gaussian distribution, where gradients can backpropagate via a *reparametrization trick* [31].

In the absence of node features, $\mathbf{X}$ becomes the identity matrix. This relaxation allows the reconstruction of a structure-only network. When provided with node features, the accuracy of VGAE link prediction improves [12].

B. Variational Graph Autoencoder for Community Detection

A major drawback in VGAE's approach is its restriction on projected nodes to a Univariate Gaussian space. This restriction suggests that all generated nodes come from a single clustering space. More specifically, dissimilar nodes tend to stay away from the Gaussian mean (centroid) [12]. On the contrary, the mean of the Gaussian should be a better representative of each respective community such that nodes which are similar should stay closer to its represented mean.

Utilizing this fact, we consider the unsupervised learning problem of community detection while adhering to the VGAE framework. Suppose that each node originating from a particular community is similar in some way, we can encode their similarity into the node's representation vector $\mathbf{z}$ which is better described by the mixture's mean. The generative process then follows:

- For communities $C = \{c_1, ..., c_K\}$
  - Obtain a sample $c \sim \text{Cat}(\boldsymbol{\pi})$
  - where, K is the number of clusters hyperparameter, π_k is the prior probability for cluster $k, \boldsymbol{\pi} \in \mathbb{R}_+^K, \sum_{k=1}^{K} \pi_k = 1$. $\text{Cat}(\boldsymbol{\pi})$ is the categorical distribution parameterized by $\boldsymbol{\pi}$.
- For nodes $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$,
  - Obtain a latent vector $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \mathbf{I})$
  - where $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ are the mean and variance of the multivariate Gaussian distribution corresponding to cluster c.
- Obtain a sample $\mathbf{a}$ by,
  - Computing the expectation $\boldsymbol{\mu}_x = f(\mathbf{z}; \boldsymbol{\theta})$
  - Sample $\mathbf{a} \sim \text{Bern}(\boldsymbol{\mu}_x)$

The function $f(\mathbf{z}; \boldsymbol{\theta})$ is a non-linear function whose input is $\mathbf{z}$ and is parameterized by $\boldsymbol{\theta}$. Particularly, we use the $\sigma(\mathbf{z}_i^\top \mathbf{z}_j)$ inner product decoder. $\text{Bern}(\cdot)$ denotes the multivariate Bernoulli distribution parameterized by the latent vector $\boldsymbol{\mu}_x$. Then, the joint probability $p(\mathbf{a}, \mathbf{z}, c)$ can be factorized as:

$$p(\mathbf{a}, \mathbf{z}, c) = p(\mathbf{a} \mid \mathbf{z}) p(\mathbf{z} \mid c) p(c), \tag{6}$$

with $\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_N\}$. Since $\mathbf{a}$ and c are independently conditioned on $\mathbf{z}$, the factorized probabilities can be defined as:

$$p(c) = \text{Cat}(c \mid \boldsymbol{\pi}) \tag{7}$$

$$p(\mathbf{z} \mid c) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2 \mathbf{I}) \tag{8}$$

$$p(\mathbf{a} \mid \mathbf{z}) = \text{Bern}(\mathbf{a} \mid \boldsymbol{\mu}_x) \tag{9}$$

For brevity, $p_{\boldsymbol{\theta}}(\cdot) = p(\cdot)$ and $q_{\boldsymbol{\phi}}(\cdot) = q(\cdot)$, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \mathcal{L}_{\text{ELBO}}(\mathbf{x})$ we can rewrite the lower bound in (5) to include the new terms,

$$\log p(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z},c|\mathbf{x},\mathbf{a})}\left[\log \frac{p(\mathbf{a},\mathbf{z},c)}{q(\mathbf{z},c \mid \mathbf{x},\mathbf{a})}\right], \tag{10}$$

where $q(\mathbf{z}, c \mid \mathbf{x}, \mathbf{a})$ is the variational posterior which approximates the true posterior $p(\mathbf{z}, c \mid \mathbf{x}, \mathbf{a})$. Under the mean-field assumption, the approximate distribution can be factorized as:

$$q(\mathbf{z}, c \mid \mathbf{x}, \mathbf{a}) = q(\mathbf{z} \mid \mathbf{x}, \mathbf{a}) q(c \mid \mathbf{x}, \mathbf{a}). \tag{11}$$

Substituting (6) and (11) into (10), the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ can be rewritten as,

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z},c|\mathbf{x},\mathbf{a})}\left[\log \frac{p(\mathbf{a},\mathbf{z},c)}{q(\mathbf{z},c \mid \mathbf{x},\mathbf{a})}\right] \\
&= \mathbb{E}_{q(\mathbf{z},c|\mathbf{x},\mathbf{a})}\left[\log p(\mathbf{a},\mathbf{z},c)\right. \\
&\quad \left. - \log q(\mathbf{z},c \mid \mathbf{x},\mathbf{a})\right] \\
&= \mathbb{E}_{q(\mathbf{z},c|\mathbf{x},\mathbf{a})}\left[\log p(\mathbf{a} \mid \mathbf{z}) + \log p(\mathbf{z} \mid c)\right. \\
&\quad + \log p(c) - \log q(\mathbf{z} \mid \mathbf{x},\mathbf{a}) \\
&\quad \left. - \log q(c \mid \mathbf{x},\mathbf{a})\right].
\end{aligned} \tag{12}$$

The inference model $q(\mathbf{z} \mid \mathbf{x}, \mathbf{a})$ is then modeled using a two-layer GCN as follows,

$$\begin{aligned}
q(\mathbf{z} \mid \mathbf{x}, \mathbf{a}) &= \mathcal{N}(\mathbf{z}; \text{GCN}_{\boldsymbol{\mu}}(\mathbf{x}, \mathbf{a}), \text{GCN}_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{a})\mathbf{I}) \\
&= \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}, \log \tilde{\boldsymbol{\sigma}}\mathbf{I}).
\end{aligned} \tag{13}$$

Similar to VGAE, the first layer's weight matrix $\mathbf{W}_0$ is shared between $\tilde{\boldsymbol{\mu}}$ and $\log \tilde{\boldsymbol{\sigma}}$. Substituting the terms in, the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ can be further rewritten as:

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) =& \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N} \mathbf{x}_i \log \boldsymbol{\mu}_i^{(l)} + (1 - \mathbf{x}_i)\log(1 - \boldsymbol{\mu}_i^{(l)}) \\
&- \frac{1}{2}\sum_{c=1}^{K} \gamma_c \left(\log \boldsymbol{\sigma}_c^2 + \frac{\tilde{\boldsymbol{\sigma}}^2}{\boldsymbol{\sigma}_c} + \frac{(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}_c)^2}{\boldsymbol{\sigma}_c^2}\right) \\
&+ \sum_{c=1}^{K} \gamma_c \log \frac{\pi_c}{\gamma_c} + \frac{1}{2}\left(1 + \log \tilde{\boldsymbol{\sigma}}_c^2\right),
\end{aligned} \tag{14}$$

with L being the total number of samples through sampled using the Monte Carlo Stochastic Gradient Variational Bayes (SGVB) estimator [31]. $\mathbf{x}_i$ the vector of node i, K the number of clusters with π_c denoting the prior probability of cluster c, and γ_c denoting $q(c \mid \mathbf{x}, \mathbf{a})$ for brevity. $\boldsymbol{\mu}_x^{(l)}$ is computed as

$$\boldsymbol{\mu}_x^{(l)} = \sigma(\mathbf{z}_i^\top \mathbf{z}_j), \tag{15}$$

where $\mathbf{z}^{(l)}$ is the l^{th} sample from $q(\mathbf{z} \mid \mathbf{x}, \mathbf{a})$ as written in (13). To allow gradient backpropagation through the stochastic layer, the *reparameterization trick* is used, then $\mathbf{z}^{(l)}$ can be obtained via,

$$\mathbf{z}^{(l)} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\sigma}} \circ \boldsymbol{\epsilon}^{(l)}. \tag{16}$$

According to [31], $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$, $\circ$ is the Hadamard product operator. $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ are obtained through GCN$(\cdot)$.

If we consider regrouping the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ with like-terms, (12) can be rewritten as,

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}}(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z},c|\mathbf{x},\mathbf{a})}\left[\log \frac{p(\mathbf{a},\mathbf{z},c)}{q(\mathbf{z},c \mid \mathbf{x},\mathbf{a})}\right] \\
&= \int_{\mathbf{z}} \sum_{c} q(\mathbf{z} \mid \mathbf{x},\mathbf{a}) q(c \mid \mathbf{x},\mathbf{a}) \left[\log \frac{p(\mathbf{x},\mathbf{a} \mid \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right. \\
&\quad \left. + \log \frac{p(c \mid \mathbf{z})}{q(c \mid \mathbf{x})}\right] d\mathbf{z} \\
&= \int_{\mathbf{z}} q(\mathbf{z} \mid \mathbf{x},\mathbf{a}) \log \frac{p(\mathbf{x},\mathbf{a} \mid \mathbf{z}) p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x},\mathbf{a})} d\mathbf{z} \\
&\quad - \int_{\mathbf{z}} q(\mathbf{z} \mid \mathbf{x},\mathbf{a}) D_{KL}[q(c \mid \mathbf{x}) \parallel p(c \mid \mathbf{z})] d\mathbf{z}.
\end{aligned} \tag{17}$$

The first term in (17) has no dependency on c and from the definition of KL divergence, it is non-negative. Therefore, the $\mathcal{L}_{\text{ELBO}}(\mathbf{x})$ is maximized when $D_{KL}[q(c \mid \mathbf{x}) \parallel p(c \mid \mathbf{z})] \equiv 0$. From that, we follow [13], by defining $q(c \mid \mathbf{x}, \mathbf{a})$ as,

$$q(c \mid \mathbf{x}, \mathbf{a}) = p(c \mid \mathbf{z}) \equiv \frac{p(c) p(\mathbf{z} \mid c)}{\sum_{c'=1}^{K} p(c') p(\mathbf{z} \mid c')} \tag{18}$$

From (18) the information loss induced by the mean-field approximation can be mitigated by forcing its dependency on the posterior $p(c \mid \mathbf{z})$ and non-informative prior $p(c)$. The complete algorithm can be found in algorithm 1.

V. Experiments

Community detection algorithms are often evaluated against two kinds of network. Namely, synthetic and empirical networks (datasets). These are discussed in detail.

A. Synthetic Datasets

Two synthetic networks are used in our evaluation. We consider two most common benchmark graphs used for benchmarking community detection algorithm. Namely, we used the Girvan-Newman (GN) benchmark graph [1, 32, 33] and the LFR benchmark graph [34]. The GN benchmark graph is a variant of the planted l-partition. In our experiment, we vary the z_{out} value from a range of $\{1, ..., 8\}$. Each node has an average degree of $k = 16$, with 32 nodes in each communities (a total of 128 nodes), and 4 communities in total.

The LFR benchmark graph is an extension of the GN benchmark graph. It is considered to be more realistic than the GN benchmark graph. It introduces a skewed degree distribution and accounts for network heterogeneity, resulting in communities that are generated in different sizes. The LFR benchmark graph is generated using default parameters as suggested by Lancichinetti *et al.* [34]. These parameters are, number of nodes ($N = 1000$), average degree ($k = 15$), minimum ($c_{min} = 30$) and maximum ($c_{max} = 50$) number of nodes per community. The generation follows the *scale-free* parameters settings of exponents $\tau_1 = -2$ and $\tau_2 = -1$ respectively. On average, between 20 to 30 communities are generated.

TABLE I: Empirical Network Datasets

Dataset	Type	Nodes	Edges	Clusters (K)	Features
Karate	Social	34	78	2	*N/A*
Polblogs	Blogs	1,222	16,717	2	*N/A*
Cora	Citation	2,708	5,429	7	1,433
PubMed	Citation	19,717	44,338	3	500

B. Empirical Datasets

The empirical datasets are divided into two kinds. Networks with features and without features. The datasets are as follows:

- **Karate**: A social network represents friendship among 34 members of a karate club at a US University [35].
- **PolBlogs**: A network of political blogs assembled by Adamic *et al.* [36]. The nodes are blogs and web links between them are represented by its edge. These blogs have known political leanings and were labeled by hand by Adamic *et al.*
- **Cora**: A citation network with 2708 nodes and 5429 edges. Each node corresponds to a documents and the edges are citation links [37].
- **PubMed**: A network consisting 19,717 scientific publications from PubMed database pertaining to diabetes classified into one of three classes ("Diabetes Mellitus, Experimental", "Diabetes Mellitus Type 1", "Diabetes Mellitus Type 2"). The citation network consists of 44,338 links. Each publication in the dataset is described by a TF-IDF weighted word vector from a dictionary which consists of 500 unique words.

For starters, experiments are performed on datasets in accordance to Karrer *et al.* These networks (Karate and PolBlogs) are featureless and only contain structural information. The Karate network is a commonly studied empirical benchmark network for community detection. Similar to [26], only the largest connected component and its undirected form is considered for Polblogs. Next, two networks containing features are used (Cora and Pubmed) [22, 38].

C. Baseline Methods

We establish a baseline by comparing against several state-of-the-art methods. These methods are divided into two categories. The first category comprises of discriminative methods and the second category comprises of generative methods.

Discriminative Methods

- *Spectral Clustering* [6] is a commonly used approach for performing graph clustering. By identifying the Fiedler Vector of the Graph Laplacian, we can divide the network into two components. Repeating this process, the graph can be subdivided further, giving more clusters in the process.
- *DeepWalk* [19], proposed by Perozzi *et al.* is a network embedding method that performs a bias random walk on a given network.
- *node2vec* [20] is a generalization of DeepWalk. It leverages on homophily and structural roles in embedding.

Generative Methods

- *Stochastic Blockmodel (SBM)* [25, 26] is a state-of-the-art generative model. It models the likelihood of two nodes forming an edge on the basis of stochastic equivalence. Degree Correction eliminates the formation of single modules by normalizing the degree of nodes.
- *Variational Graph Autoencoder* [12] follows the Variational Autoencoder framework by leveraging on GCN layers.

D. Evaluation Metrics

Some of the common approaches to evaluate detected communities are Normalized Mutual Information (NMI), Variation of Information (VI) and Modularity. In some cases, accuracy can be accurately measured (i.e. when the number of clusters K is 2). Furthermore, these measures are only possible when ground-truth exists. Hence, we include other forms of measures which considers the quality of a partition without ground-truths.

1) Ground Truth:

- Accuracy measures the number of correctly classified clusters given the ground-truth. Formally, given two sets of community labels, i.e. C is the ground-truth and C' the detected community labels, the accuracy can be calculated by,

$$ACC(C') = \frac{\sum_{i=1}^{|C|} \delta(c_i, c_i')}{|C|} \times 100\%.$$

  $c_i \in C, c_i' \in C'$, where $\delta(\cdot)$ denotes the Kronecker delta, $\delta(c_i, c_i') = 1$ when both labels matches and $|\cdot|$ denotes the cardinality of a set. For clustering tasks, accuracy is usually not emphasized as labels are known to oscillate between clusters.
- NMI and VI are based on information theory. Essentially, NMI measures the 'similarity' between two community covers, while VI measures their 'disimilarity' in terms of uncertainty. Correspondingly, a higher NMI indicates a better match between both covers while VI indicates the opposite. Formally [39],

$$\text{NMI}(C, C') = \frac{2\text{I}(C, C')}{(\text{H}(C) + \text{H}(C'))}$$

  and

$$\text{VI}(C, C') = \text{H}(C) + \text{H}(C') - 2\text{I}(C, C'),$$

  where $\text{H}(\cdot)$ is the entropy function, and $\text{I}(C, C') = \text{H}(C) + \text{H}(C') - \text{H}(C, C')$ is the mutual information function.

2) Cluster Quality:

- Modularity (Q) [14] measures the quality of a particular community structure when compared to a null (random) model. Intuitively, intra-community links are expected to be stronger than inter-community links. Specifically,

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{4m} \right) \delta(c_i, c_j),$$

TABLE II: Experimental Results on Karate Dataset

	NMI (↑)	VI (↓)	ACC (↑)	Q (↑)	CON (↓)	TPR (↑)
Spectral Clustering	0.2297	2.0005	0.7353	0.1127	0.3702	0.7363
DeepWalk	0.7198	0.8812	0.9353	0.3582	0.1337	0.9353
node2vec	0.8372	0.8050	0.9706	0.1639	0.4239	0.4549
Stochastic Blockmodel	0.0105	1.1032	0.4412	-0.2084	0.7154	0.4034
Stochastic Blockmodel (D.C)	0.8372	0.8050	0.9706	**0.3718**	**0.1282**	**0.9412**
VGAE* + k-means	0.6486	0.8189	0.9647	0.3669	0.1295	0.9407
VGAECD*	**1.0000**	**0.6931**	**1.0000**	0.3582	0.1412	**0.9412**

TABLE III: Experimental Results on PolBlogs Dataset

	NMI (↑)	VI (↓)	ACC (↑)	Q (↑)	CON (↓)	TPR (↑)
Spectral Clustering	0.0014	1.1152	0.4828	-0.0578	0.5585	0.7221
DeepWalk	0.7367	1.0839	0.9543	0.0980	0.3873	0.6870
node2vec	0.7545	0.8613	0.9586	0.1011	0.3827	0.6863
Stochastic Blockmodel	0.0002	1.2957	0.4905	-0.0235	0.5329	0.5657
Stochastic Blockmodel (D.C)	0.7145	0.8890	0.9496	**0.4256**	**0.0730**	**0.8101**
VGAE* + k-means	0.7361	0.8750	0.9552	0.4238	0.0752	0.8089
VGAECD*	**0.7583**	**0.8583**	**0.9601**	0.4112	0.0880	0.7913

TABLE IV: Experimental Results on Cora Dataset

	NMI (↑)	VI (↓)	ACC (↑)	Q (↑)	CON (↓)	TPR (↑)
Spectral Clustering	0.0651	**2.0005**	0.1252	0.0189	0.1909	**0.6196**
DeepWalk	0.3796	2.7300	0.1626	0.6595	**0.0396**	0.4949
node2vec	0.3533	2.9947	0.1359	0.6813	0.1078	0.4902
Stochastic Blockmodel	0.0917	3.5108	**0.1639**	0.4068	0.4280	0.3376
Stochastic Blockmodel (D.C.)	0.1679	3.4547	0.1176	0.6809	0.1736	0.5112
VGAE* + k-means	0.2384	3.3151	0.1033	0.6911	0.1615	0.4906
VGAE + k-means	0.3173	3.1277	0.1589	0.6981	0.1517	0.5031
VGAECD*	0.2822	3.1606	0.1532	0.6674	0.1808	0.5076
VGAECD	**0.5072**	2.7787	0.1101	**0.7029**	0.1371	0.4987

TABLE V: Experimental Results on PubMed Dataset

	NMI (↑)	VI (↓)	ACC (↑)	Q (↑)	CON (↓)	TPR (↑)
Spectral Clustering	0.0382	1.4341	0.3261	0.0414	0.5645	**0.4935**
DeepWalk	0.2946	1.7865	0.3101	0.5766	**0.0499**	0.2461
node2vec	0.1197	1.9849	0.2228	0.3501	0.3170	0.2269
Stochastic Blockmodel	0.0004	1.9340	0.3080	-0.1620	0.1038	0.1965
Stochastic Blockmodel (D.C.)	0.1325	2.0035	0.3118	0.5622	0.8121	0.2441
VGAE* + k-means	0.2041	1.8096	0.3724	0.5273	0.1320	0.2898
VGAE + k-means	0.1981	1.8114	0.2751	0.5297	0.1283	0.2900
VGAECD*	0.1642	1.8320	0.1956	0.4966	0.1252	0.2692
VGAECD	**0.3252**	**1.7056**	**0.4216**	**0.6878**	0.1636	0.4827

where $A_{ij} - k_i k_j / 4m$ measures the actual edge connectivity versus the expectation at random and $\delta(c_i, c_j)$ defines the Kronecker delta, where $\delta(c_i, c_j) = 1$ when both node i and j belongs to the same community, and 0 otherwise. Essentially, Q approaches 1 when the partitions are considered good.

- Conductance (CON) [40, 41] measures the separability of a community across the fraction of outgoing local volume of links in the community, which is defined as,

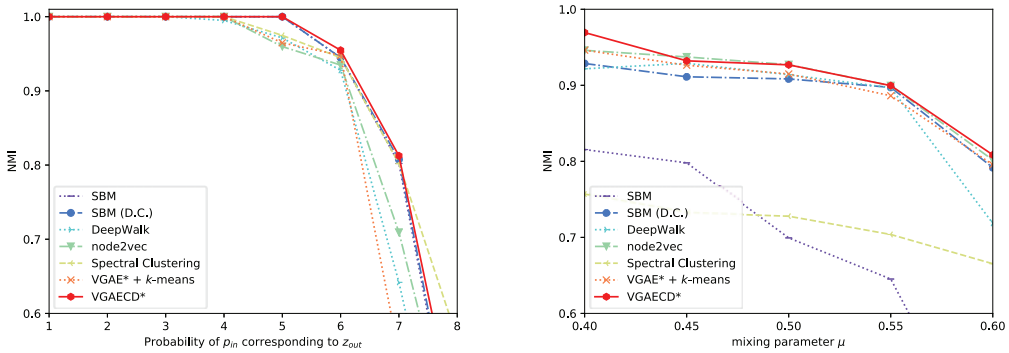
$$\text{CON}(C) = \frac{\sum_{i \in C, j \in C'} A_{ij}}{\min(a(C), a(C'))},$$

where the nominator defines the total number of edges within community C and $a(C) = \sum_{i \in C}(j \in V)$ defines the volume of set $C \subseteq V$. A better local separability of community is achieved when the overall conductance value is the smallest.

- Triangle Participation Ratio (TPR) [41] measures the fraction of triads within the community C.

$$\text{TPR}(C) = |\{v_i \in C, \{(v_j, v_k) : v_j, v_k \in C, (v_i, v_j), (v_j, v_k), (v_i, v_k) \in E\} \neq \emptyset\}| / |C|,$$

where, E denotes the total number of edges in the graph G. A larger TPR value indicates a denser community structure.



(a) GN Benchmark Graph

(b) LFR Benchmark Graph

Fig. 2: Comparative performance of VGAECD against other methods on Synthetic Networks

E. Experiment Settings

For discriminative models such as node2vec and DeepWalk, the latent representation is learned first. Next, k-means is applied to the learned latent vector with K given a priori. The parameters used for node2vec are performed using exhaustive search on variables $p, q \in \{0.25, 0.5, 1, 2, 4\}$ as suggested in [20]. Specifically, the parameters obtained were $(p = 0.5, q = 4)$, $(p = 0.25, q = 0.25)$, $(p = 1, q = 0.25)$, $(p = 0.5, q = 1)$ for Karate, PolBlogs, Cora and PubMed respectively. As for DeepWalk, the parameters used are $d = 128, r = 10, l = 80, k = 10$ which were the suggested values [19]. On the other hand, generative models like SBM (and D.C.) have several optimization strategies. In this case, we applied the Expectation-Maximization (EM) algorithm as suggested in [26].

For a fair comparison between VGAE and VGAECD, we used identical layer configurations for both models. The layer configurations are (32-16), (32-16), (32-8), (32-8) for Karate, PolBlogs, Cora and PubMed respectively. These configurations are determined empirically as suggested from [12]. Generally, we found the first layer to be insensitive and second layer to be sensitive. By reducing the size of the second layer with respect to the number nodes we found that 8 was ideal for Cora and PubMed. The hyperparameter K is given a priori for all methods. For a fair comparison, the average of 10 runs were taken for both discriminative and generative models. All experiments were conducted on a Ubuntu 16.06 LTS machine with 64 GB of RAM and two GeForce GTX 1080 Ti graphics card.

F. Experiment Results

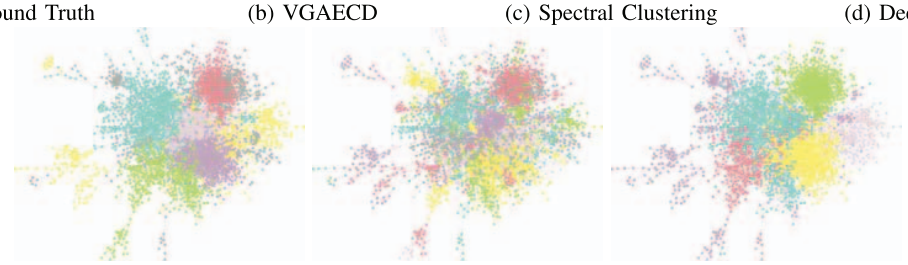
We first compare our result with 7 baseline methods on several state-of-the-art methods that employs unsupervised network embedding, except SBM; the only generative model that does learn a network embedding. Since VGAE is non-clustering, the two-step approach for clustering was applied, i.e. obtaining the latent vectors and subsequently applying k-means. The * symbol denote methods that were confined to structural information only.

1) Synthetic Dataset Performance: Fig. 2 depicts the performance of the proposed model in comparison to other methods. In Fig. 2a, VGAECD can be seen as a strong performer when $z_{\text{out}} \geq 4$. On the LFR benchmark graph in Fig. 2b, the performance of VGAECD is comparable to other methods. When $\mu < 0.4$, VGAECD is capable of outperforming other methods. When $\mu > 0.55$, VGAECD is seen to exhibit similar performance with other methods. In both cases, the performance were as expected since the mixing parameter (z_{out} and μ) is consistent with the study recoverability limit in planted partitions [29, 30].

2) Empirical Dataset Performance: Experiments performed on four different empirical datasets are shown in table II, III, IV and V for Karate, PolBlogs, Cora and PubMed respectively. We measure the performance of clusters found using metrics as proposed in section V-D and the best values are marked in bold.

Generally, the experiments revealed that our method outperforms other methods when ground truth is given. In terms of cluster quality, VGAECD performs relatively well in terms of modularity score (Q). However, it retains competitiveness on Conductance (CON) and Triangle Participation Ratio (TPR) measures. Since datasets such as Cora and PubMed have more than 2 clusters ($K > 2$), the accuracy of labels can be affected by label oscillation. Therefore, it is a less accurate measure for measuring cluster's label when compared to classification accuracy measures. However, accurate measures can still be obtained for datasets with only two clusters such as Karate and PolBlogs, which revealed that the proposed method is better than baseline methods. In most cases, the results of our method is comparable to SBM (D.C.). This is plausible since SBM (D.C.) has an advantage due to its prior knowledge on degree normalization. Regardless, when more than two clusters are given, the modularity score of VGAECD outperforms SBM (D.C.) as shown in Cora and PubMed datasets.

(a) Ground Truth (b) VGAECD (c) Spectral Clustering (d) DeepWalk



(e) VGAE + k-means (f) SBM (D.C.) (g) node2vec

Fig. 3: Visualization of community assignment on Cora Dataset (largest connected component)

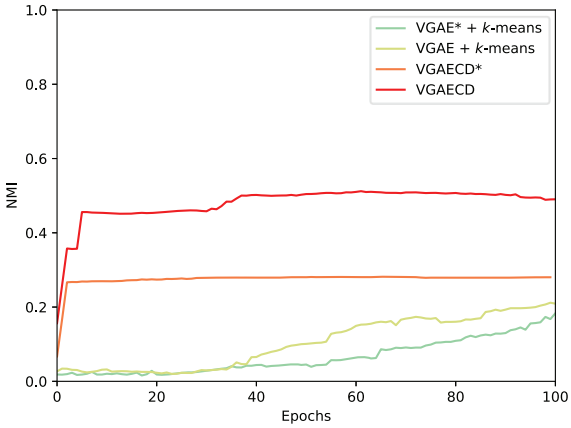


Fig. 4: NMI Convergence Comparison

3) Time Complexity Analysis: Since the proposed model follows the VAE framework, it employs a similar optimization method using SGVB. Therefore, it follows a linear-time complexity for one epoch, but requires L number of runs to achieve convergence. The convergence rate of NMI with respect to the number of epochs can be observed in Fig. 4 in comparison to VGAE. In contrast to VGAE, the proposed method can achieve convergence at a faster rate.

4) Synthetic Network Generation: The implication of a generative model is its ability to generate a graph when prescribed a certain set of parameters. Therefore, a synthetic network can be generated using the proposed VGAECD model. Given parameters c and ω, we can generate a network simply by following the generative process specified in section IV-B. However, in order to vary the community structure, we can follow the planted partition's approach by including the mixing of a random network model,

$$\omega = \lambda\omega^{\text{planted}} + (1-\lambda)\omega^{\text{random}}. \tag{19}$$

ω^{planted} defines the amount of actual draws from the Gaussian Mixture model and ω^{random} draws from the random model. For instance, ω^{planted} can be specify as,

$$\omega^{\text{planted}} = (n_1, n_2, n_3, n_4), \tag{20}$$

which n_c denotes the number of draws from the mixture model with μ_c and σ_c. In (20), we specify the number of nodes drawn for four different communities. A generated matrix $\tilde{\mathbf{A}}$ can be obtained as shown in Algorithm 1.

5) Network Visualization: Community assignments for Cora dataset is visualized in Fig. 3. Since Cora has several disconnected nodes, only the largest connected component is visualized. Among them, VGAECD has closer resemblance to the ground truth's cluster assignment. Notably VGAECD is able to recover a community structure in the center of the network. Additionally, it also has less tendency to cluster nodes that are far away which is seen in VGAE + k-means and SBM (D.C.). DeepWalk, however, appears to have a resolution problem, resulting in smaller clusters merging together. This can be seen as the number of clusters depicted in the largest component is fewer than $K = 7$. This problem is not observed in node2vec since the sampling strategies are generalization of DeepWalk; parameters p and q allows node2vec to explore more locations. In contrast, DeepWalk is restricted to visiting nodes within the pivot node's vicinity. However, to achieve the observed results, node2vec requires a costly parameter search which is not ideal. Among the baseline methods, Spectral Clustering appears to struggle in finding a community structure, even though it performed relatively very well on synthetic benchmark graphs.

VI. Conclusion

This paper proposes a novel Variational Graph Autoencoder for community detection, VGAECD which generalizes VGAE for community detection tasks. It is capable of learning both features and structural information and encode them into a community-aware latent representation. The representation learned are different from previous network representation methods. Specifically, the latent representations themselves are parameters to a probabilistic graphical model i.e. the Gaussian Mixture Model. Thus, allowing us to draw samples from the learned model itself and generate synthetic graphs like SBM. Additionally, the flexibility of the proposed method shows that by leveraging on more information, it is capable of outperforming other methods in community structure recovery. By employing a generative approach, our method is able to recover communities in a single step and simplify the process. Consequently, it does not require a second step for clustering such as applying k-means on the learned representation. Moreover, in comparison to existing state-of-the-art generative model such as SBM, VGAECD is community structure definition agnostic. In other words, nodes are not forced to be similar under a specific similarity measure. This is an advantage over other generative community detection algorithms where the definition of community structures are always assumed. This is desirable feature in cases where networks can have a mixture of community structure, i.e. multilayer networks.

Acknowledgement

This work was supported by JSPS Grant-in-Aid for Scientific Research (B)(Grant Number 17H01785), JST CREST (Grant Number JPMJCR1687) and NEDO (New Energy and Industrial Technology Development Organization).

References

[1] S. Fortunato, "Community Detection in Graphs," *Phys. Rep.*, vol. 486, no. 3, pp. 75–174, 2010.

[2] M. E. J. Newman, "Modularity and Community Structure in Networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, Jun. 6, 2006.

[3] S. Fortunato and D. Hric, "Community Detection in Networks: A User Guide," *Phys. Rep.*, vol. 659, pp. 1–44, Nov. 11, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep Into Rectifiers: Surpassing Human-level Performance on Imagenet Classification," *ICCV*, pp. 1026–1034, 2015.

[5] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" *NIPS*, pp. 5580–5590, 2017.

[6] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *NIPS*, pp. 849–856, 2002.

[7] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning Deep Representations for Graph Clustering.," in *AAAI*, 2014, pp. 1293–1299.

[8] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity Based Community Detection with Deep Learning.," *IJCAI*, pp. 2252–2258, 2016.

[9] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," *ICML*, pp. 1096–1103, 2008.

[10] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized Graph Autoencoder for Graph Clustering," *CIKM*, pp. 889–898, 2017.

[11] D. Chakrabarti and C. Faloutsos, "Graph Mining: Laws, Generators, and Algorithms," *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006.

[12] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," in *Bayesian Deep Learning Workshop*, Centre Convencions Internacional Barcelona, Barcelona, Spain, 2017.

[13] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering," *IJCAI*, 2017.

[14] M. E. J. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Phys. Rev. E*, vol. 69, no. 2, p. 026 113, 2004.

[15] S. Fortunato and M. Barthélemy, "Resolution Limit in Community Detection," *PNAS*, vol. 104, no. 1, pp. 36–41, Feb. 1, 2007.

[16] B. H. Good, Y.-A. de Montjoye, and A. Clauset, "The Performance of Modularity Maximization in Practical Contexts," *Phys. Rev. E*, vol. 81, no. 4, Apr. 15, 2010.

[17] U. N. Raghavan, R. Albert, and S. Kumara, "Near Linear Time Algorithm to Detect Community Structures in Large-scale Networks," *Phys. Rev. E*, vol. 76, no. 3, p. 036 106, 2007.

[18] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning Graph Representations with Global Structural Information," *CIKM*, pp. 891–900, 2015.

[19] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," *KDD*, pp. 701–710, 2014.

[20] A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," *KDD*, vol. 2016, pp. 855–864, Aug. 2016.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *NIPS*, pp. 3844–3852, 2016.

[22] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," *ICLR*, 2017.

[23] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker Graphs: An Approach to Modeling Networks," *JMLR*, vol. 11, pp. 985–1042, Feb 2010.

[24] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community Structure and Scale-free Collections of Erdős-Rényi Graphs," *Phys. Rev. E*, vol. 85, no. 5, p. 056 109, May 10, 2012.

[25] T. A. B. Snijders and K. Nowicki, "Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure," *J. Classif.*, vol. 14, no. 1, pp. 75–100, Jan. 1, 1997.

[26] B. Karrer and M. E. J. Newman, "Stochastic Blockmodels and Community Structure in Networks," *Phys. Rev. E*, vol. 83, no. 1, p. 016 107, 2011.

[27] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed Membership Stochastic Blockmodels," *JMLR*, vol. 9, pp. 1981–2014, Sep 2008.

[28] D. B. Larremore, A. Clauset, and A. Z. Jacobs, "Efficiently Inferring Community Structure in Bipartite Networks," *Phys. Rev. E*, vol. 90, no. 1, p. 012 805, 2014.

[29] E. Abbe and C. Sandon, "Community Detection in General Stochastic Block models: Fundamental Limits and Efficient Algorithms for Recovery," *FOCS*, pp. 670–688, Oct. 2015.

[30] E. Abbe, A. S. Bandeira, and G. Hall, "Exact Recovery in the Stochastic Block Model," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 471–487, Jan. 2016.

[31] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ICLR*, 2014.

[32] M. Girvan and M. E. J. Newman, "Community Structure in Social and Biological Networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[33] A. Condon and R. M. Karpy, "Algorithms for Graph Partitioning on the Planted Partition Model," 2000.

[34] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark Graphs for Testing Community Detection Algorithms," *Phys. Rev. E*, vol. 78, no. 4, p. 046 110, 2008.

[35] W. W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.

[36] L. A. Adamic and N. Glance, "The Political Blogosphere and the 2004 US Election: Divided They Blog," *LinkKDD*, pp. 36–43, 2005.

[37] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[38] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting Semi-supervised Learning with Graph Embeddings," *ICML*, pp. 40–48, 2016.

[39] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing Community Structure Identification," *J. Stat. Mech.*, vol. 2005, no. 09, P09008, 2005.

[40] R. Kannan, S. Vempala, and A. Vetta, "On Clusterings: Good, Bad and Spectral," *J. ACM*, vol. 51, no. 3, pp. 497–515, May 2004.

[41] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities Based on Ground-Truth," *KAIS*, vol. 42, no. 1, pp. 181–213, 2015.