Hindawi
Complexity
Volume 2018, Article ID 4867304, 13 pages
https://doi.org/10.1155/2018/4867304

Research Article

Variational Approach for Learning Community Structures

Jun Jin Choong,[1] **Xin Liu**,[2] **and Tsuyoshi Murata**[1]

[1]*Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan*
[2]*National Institute of Advanced Industrial Science and Technology, Tokyo, Japan*

Correspondence should be addressed to Jun Jin Choong; choong.junjin@gmail.com

Received 10 August 2018; Accepted 2 December 2018; Published 13 December 2018

Academic Editor: Pasquale De Meo

Discovering and modeling *community structure* exist to be a fundamentally challenging task. In domains such as biology, chemistry, and physics, researchers often rely on community detection algorithms to uncover community structures from complex systems yet no unified definition of community structure exists. Furthermore, existing models tend to be oversimplified leading to a neglect of richer information such as nodal features. Coupled with the surge of user generated information on social networks, a demand for newer techniques beyond traditional approaches is inevitable. Deep learning techniques such as network representation learning have shown tremendous promise. More specifically, supervised and semisupervised learning tasks such as link prediction and node classification have achieved remarkable results. However, unsupervised learning tasks such as community detection remain widely unexplored. In this paper, a novel deep generative model for community detection is proposed. Extensive experiments show that the proposed model, empowered with Bayesian deep learning, can provide insights in terms of uncertainty and exploit nonlinearities which result in better performance in comparison to state-of-the-art community detection methods. Additionally, unlike traditional methods, the proposed model is community structure definition agnostic. Leveraging on low-dimensional embeddings of both network topology and feature similarity, it automatically learns the best model configuration for describing similarities in a community.

1. 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 make network science 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-fits-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 [4, 5] as a potential solution.

In machine learning, representation learning [6] 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 [7]. 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) [8]. Using fundamental building blocks from a probabilistic perspective, assumptions are given in forms of noninformative priors and the model is forced to correct these assumptions while learning. Consequently, the models become less ambiguous than a typical deep learning model which is commonly known to be a black-box.

Leveraging on recent advances in representation learning, network representation learning aims at 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 involves 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 [9] or k-means is applied to the learned embedding. For instance, Tian *et al.* proposed a network representation [10] learning model to learn a nonlinear mapping of the original network using a Stacked Autoencoder by showing that spectral clustering and Autoencoders have the same optimization objectives. Yang *et al.* considered a Stacked Autoencoder as a modularity optimization problem and further introduced a semisupervised approach through *must-pair* nodes for increased performance [11]. Assignment of communities is then obtained through k-means clustering from the latent representation that exhibits the highest modularity score. Inspired from Denoising Autoencoders [12], Wang *et al.* proposed Marginalized Graph Autoencoder for Graph Clustering (MGAE) [13] 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 [14].

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) [15]. Originally, VGAE projects graph convolutions into a Univariate Gaussian latent space and have only been considered for semisupervised 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 [16]. 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:

(i) 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.

(ii) The proposed model inherits the benefits of Variational Autoencoder Framework. The advantages are threefold: (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.

(iii) The proposed model outperforms the state-of-the-art models in community detection without requiring additional priors (unlike the Degree-Corrected SBM).

2. Problem Definition

A network pertaining to nodes, edges, and node features can be formally defined as $G = (V, E, X)$, where $V = \{v_i, \ldots, v_N\}$ consists of a set of nodes $|V| = N$, $E = \{e_{ij}\}$ is a set of edges, and $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ 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 $e_{ij} \in E$ and otherwise it is equal to 0.

Given the network G, the objective of community detection or graph clustering is to partition the nodes in G into K disjoint groups $\{c_1, c_2, \ldots, c_K\}$, 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 the same set of parameters. Concretely, $p(\mathbf{A} \mid \boldsymbol{\theta}) = G'$. Under the model selection criterion, the model is said to be good when $G' \cong G$ 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.

3. Related Work

Recent work in community detection can be broadly categorized into two types of models, 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 explain the observed phenomenon.

3.1. Discriminative Methods and Models. Predominantly, modularity maximization [2, 17] has been considered as the most successful method for detecting communities. However, it suffers from a resolution limit problem [18] and is known to exhibit degeneracies [19]. In terms of speed, label propagation [20] is capable of detecting communities in large-scale networks near linear time, though the solutions are usually nonunique. Additionally, other approaches such as WalkTrap [21], Infomap [22], Louvain [23], and their empirical competitiveness are subjected to trade-off between accuracy and scalability [24]. Representation learning methods such as GraRep [25] and CFOND [26] consider the completion of their adjacency matrix and can be generally considered as matrix factorization problem. Meanwhile, others like DeepWalk [27] and node2vec [28] consider representation of each node via a biased random walk. It assumes that neighboring nodes share similarities from the pivot node. Hence, when nodes are clustered together, they tend to cooccur 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 [10]. Leveraging on deep learning's nonlinearity and recent advances in Convolutional Neural Networks, [29, 30] proposed the Graph Neural Network (GNN) and its generalization, the Graph Convolutional Neural Network (GCN) [29]. Defferrard *et al.* first cast the problem by projecting graph convolutions into spectral space, and convolving within this space.

3.2. Generative Methods and Models. Generative models can be further subdivided into algorithmic and statistical types. Examples of algorithmic models include the Kronecker Graphs [31], NetSim [32], and Block Two-Level Erdős-Rényi (BTER) model [33]. On the other hand, statistical methods attempt to approximate the true distribution via statistical inferencing or through statistical models (i.e., benchmark graphs such as GN [34], LFR [35], and mLFR [36, 37]). A widely known generative model for capturing networks with group structure is the Stochastic Blockmodel (SBM) or also known as the planted partition model. First explored by Snijders and Nowicki [38] 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 community memberships: two nodes within a community sharing the same stochasticity. However, the vanilla SBM exhibits a problem where high degree nodes are clustered into a community of their own. Karrer and Newman proposed the Degree Corrected (D.C.) SBM [39] which introduces a normalizing prior. Extensions to SBM include the Mixed Membership SBM (MMSBM) [40] for identifying mix community participation and bipartite SBM (biSBM) [41] for finding communities in bipartite networks. Today, SBM is well explored and its limitations has been widely studied [42, 43]. However, SBM is not a network representation learning model. Instead, SBM learns the latent variables $\boldsymbol{\Pi}$ and $\mathbf{Z}$ which describe the probabilities of cluster connectivity and cluster assignment, respectively, of a particular node which differs from common representation learning method.

Contrary to SBM, typically Autoencoders consists of two nongenerative steps (encoder and decoder). Consequently, the learned representation cannot be generalized for generation of networks. To alleviate this problem, most recent approaches consider generative models for representation learning such as Generative Adversarial Networks (GAN) or Variational Autoencoder (VAE). For graphs, Kipf and Welling [15] introduced a variant of VAE for link prediction tasks in graphs and for GAN, and Pan *et al.* [44] recently introduced adversarially regularized graph autoencoder (ARGA). In this work, we only consider the framework of VAE. We discuss this in Section 4.1.

4. Methodology

4.1. Variational Graph Autoencoder. Variational Graph Autoencoder (VGAE) [15] extends the problem of learning network embedding to a generative perspective by leveraging on the Variational Autoencoder (VAE) framework [45]. 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

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

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

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

$$\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \widehat{\mathbf{A}} \tau\left(\widehat{\mathbf{A}} \mathbf{X} \mathbf{W}_0\right) \mathbf{W}_1, \tag{3}$$

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 $\mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. $\tau(\cdot)$ is the nonlinear function

such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ or $\mathrm{sigmoid}(t) = 1/(1 + e^{-t})$. $\widehat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ denotes the symmetric normalized adjacency matrix. The generative model is simply the inner product between the latent variables:

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

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

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{X}) &\geq \mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{Z}\mid\mathbf{X},\mathbf{A})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{A} \mid \mathbf{Z})\right] \\ &\quad - D_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{Z} \mid \mathbf{X}) \parallel p_{\boldsymbol{\theta}}(\mathbf{Z})\right]. \end{aligned} \quad (5)$$

$D_{KL}[q_{\boldsymbol{\phi}}(\cdot) \parallel p_{\boldsymbol{\theta}}(\cdot)]$ 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 defined 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, whose gradients can backpropagate via a *reparametrization trick* [45].

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 [15].

4.2. Variational Graph Autoencoder for Community Detection (VGAECD). A major drawback in VGAE's approach is its restriction of nodes to be projected in 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) [15]. 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 their represented mean. Thus, nodes that are well represented by the mean representation hold equivalence in similarity. In this scenario, we can consider this as a relaxation of SBM which requires nodes in the same block to uphold stochastic equivalence.

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:

(i) For communities $C = \{c_1, \ldots, c_K\}$

  (a) Obtain a sample $c \sim \mathrm{Cat}(\boldsymbol{\pi})$

  (b) where K is the number of clusters hyperparameters and π_k is the prior probability for cluster $k, \boldsymbol{\pi} \in \mathbb{R}_+^K, \sum_{k=1}^{K}\pi_k = 1$. $\mathrm{Cat}(\boldsymbol{\pi})$ is the categorical distribution parameterized by $\boldsymbol{\pi}$.

(ii) For nodes $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$,

  (a) Obtain a latent vector $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2\mathbf{I})$

  (b) where $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ are the mean and variance of the multivariate Gaussian distribution corresponding to cluster c.

(iii) Obtain a sample $\mathbf{a}$ by

  (a) computing the expectation $\boldsymbol{\mu}_x = f(\mathbf{z}; \boldsymbol{\theta})$

  (b) sample $\mathbf{a} \sim \mathrm{Bern}(\boldsymbol{\mu}_x)$

The function $f(\mathbf{z}; \boldsymbol{\theta})$ is optionally a nonlinear function whose input is $\mathbf{z}$ and is parameterized by $\boldsymbol{\theta}$. Particularly, we use the $\tau(\mathbf{z}_i^{\top}\mathbf{z}_j)$ inner product decoder. $\mathrm{Bern}(\cdot)$ 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

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

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

$$p(c) = \mathrm{Cat}(c \mid \boldsymbol{\pi}) \quad (7)$$

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

$$p(\mathbf{a} \mid \mathbf{z}) = \mathrm{Bern}\left(\mathbf{a} \mid \boldsymbol{\mu}_x\right) \quad (9)$$

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

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

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

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

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

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

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

$$\begin{aligned} q(\mathbf{z} \mid \mathbf{x}, \mathbf{a}) &= \mathcal{N}\left(\mathbf{z}; \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{x}, \mathbf{a}), \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{a})\,\mathbf{I}\right) \\ &= \mathcal{N}\left(\mathbf{z}; \widetilde{\boldsymbol{\mu}}, \log \widetilde{\boldsymbol{\sigma}}\mathbf{I}\right). \end{aligned} \quad (13)$$

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, $\mathscr{L}_{\text{ELBO}}(\mathbf{x})$ can be further rewritten as

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

with L being the total number of samples through sampled using the Monte Carlo Stochastic Gradient Variational Bayes (SGVB) estimator [45]. $\mathbf{x}_i$ is the vector of node i, K is the number of clusters with π_c denoting the prior probability of cluster c, and γ_c denotes $q(c \mid \mathbf{x}, \mathbf{a})$ for brevity. $\boldsymbol{\mu}_x^{(l)}$ is computed as

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

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

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

Then, according to [45], $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$; $\circ$ is the Hadamard product operator. $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ are obtained through GCN$(\cdot)$.

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

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

The first term in (17) has no dependency on c and from the definition of KL divergence, it is nonnegative. Therefore, $\mathscr{L}_{\text{ELBO}}(\mathbf{x})$ is maximized when $D_{KL}[q(c \mid \mathbf{x}) \parallel p(c \mid \mathbf{z})] \equiv 0$. From that, we follow [16], 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 noninformative prior $p(c)$. The complete VGAECD algorithm can be found in Algorithm 1 and Figure 1 illustrates the conceptual idea of VGAECD.

5. Experiments

Community detection algorithms are often evaluated against two kinds of networks: synthetic and empirical datasets. These are discussed in detail in the following subsections.

5.1. 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, 34, 46] and the LFR benchmark graph [35]. 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 $\{1, \ldots, 8\}$. Each node has an average degree of $k = 16$, with 32 nodes in each community (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.* [35]. These parameters are number of nodes ($N = 1000$), average degree ($k = 15$), and 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 and 30 communities are generated.

5.2. Empirical Datasets. The empirical datasets are divided into two kinds: networks with features and without features. The datasets are as follows:

(i) **Karate**: a social network represents friendship among 34 members of a karate club at a US University [47].

(ii) **PolBlogs**: a network of political blogs assembled by Adamic and Glance [48]. The nodes are blogs and web links between them are represented by their edge. These blogs have known political leanings and were labelled by hand by Adamic and Glance.

(iii) **Cora**: a citation network with 2,708 nodes and 5,429 edges. Each node corresponds to a document and the edges are citation links [49].

(iv) **PubMed**: A network consisting of 19,717 scientific publications from PubMed database pertaining to diabetes was 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.

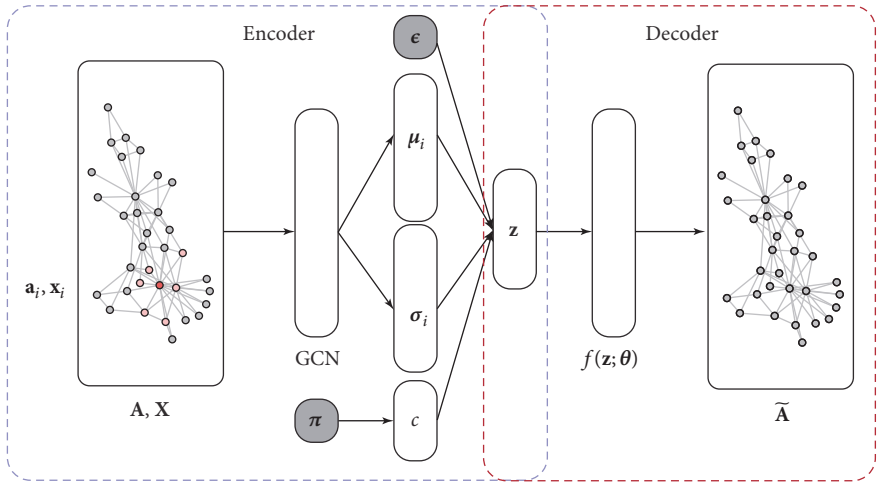


Figure 1: Conceptual illustration of Variational Graph Autoencoder Framework for Community Detection (VGAECD). In the encoding phase, VGAECD first convolves on the network, learning structural and nodal features in the process. These pieces of information are then mapped into a latent representation,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

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 $\widetilde{\mathbf{A}}$
$\boldsymbol{\pi} \sim \mathcal{U}(0, 1)$
for $l = 1, ..., L$ **do**
  for $i = 1, ..., N$ **do**
    $\boldsymbol{\mu}_i = \text{GCN}_{\boldsymbol{\mu}}(\mathbf{x}_i, \mathbf{a}_i)$
    $\boldsymbol{\sigma}_i = \text{GCN}_{\boldsymbol{\sigma}}(\mathbf{x}_i, \mathbf{a}_i)$
    Sample $c \sim \text{Cat}(c \mid \boldsymbol{\pi})$
    Sample $\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_c|_i, \text{diag}(\boldsymbol{\sigma}_c^2|_i))$
    Obtain reconstructed $\widetilde{\mathbf{a}}_i = \tau(\mathbf{z}_i^\top \mathbf{z}_j)$ ▷ Decoder
    Compute loss, $\mathcal{L}_{\text{ELBO}}$ ▷ From (14)
      and backpropagate gradients.
  end for
end for
Extract community assignment $\boldsymbol{\gamma}$ via $\mathbf{z}_i$ ▷ From (18)
Return $\boldsymbol{\gamma}$, $\widetilde{\mathbf{A}} = \{\widetilde{\mathbf{a}}_1, ..., \widetilde{\mathbf{a}}_N\}$

Algorithm 1: Variational Graph Autoencoder for Community Detection (VGAECD).

For starters, experiments are performed on datasets in accordance to Karrer and Newman. 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 [39], only the largest connected component and its undirected form are considered for Polblogs. Next, two networks containing features are used (Cora and Pubmed) [30, 50]. Table 1 summarizes the list of datasets and their respective properties.

5.3. 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 discriminative methods and the second category comprises generative methods.

Discriminative Methods

(i) *Spectral Clustering* [9] 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.

(ii) *Louvain* [23] is a greedy modularity optimization method for maximizing modularity score.

Table 1: 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

(iii) *DeepWalk* [27], proposed by Perozzi *et al.*, is a network embedding method that performs a bias random walk on a given network.

(iv) *node2vec* [28] is a generalization of DeepWalk. It leverages on homophily and structural roles in embedding.

Generative Methods

(i) *Stochastic Blockmodel (SBM)* [38, 39] 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 (D.C.) penalizes the formation of single node modules by normalizing the node degrees.

(ii) *Variational Graph Autoencoder* [15] follows the Variational Autoencoder framework by leveraging on GCN layers.

5.4. 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 consider the quality of a partition without ground truths.

5.4.1. Ground Truth

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

$$ACC\left(C'\right) = \frac{\sum_{i=1}^{|C|} \delta\left(c_i, c_i'\right)}{|C|} \times 100\%. \tag{19}$$

$c_i \in C$, $c_i' \in C'$, where $\delta(\cdot)$ denotes the Kronecker delta, $\delta(c_i, c_i') = 1$ when both labels match, and $|\cdot|$ denotes the cardinality of a set. For clustering tasks, accuracy is usually not emphasized as labels are known to oscillate between clusters.

(ii) NMI and VI are based on information theory. Essentially, NMI measures the "similarity" between two community covers, while VI measures their "dissimilarity" in terms of uncertainty. Correspondingly, a higher NMI indicates a better match between both covers while VI indicates the opposite. Formally [51]

$$\mathrm{NMI}\left(C, C'\right) = \frac{2\mathrm{I}\left(C, C'\right)}{\left(\mathrm{H}\left(C\right) + \mathrm{H}\left(C'\right)\right)} \tag{20}$$

and

$$\mathrm{VI}\left(C, C'\right) = \mathrm{H}\left(C\right) + \mathrm{H}\left(C'\right) - 2\mathrm{I}\left(C, C'\right), \tag{21}$$

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

5.4.2. Community Quality

(i) Modularity (Q) [17] measures the quality of a particular community structure when compared to a null (random) model. Intuitively, intracommunity links are expected to be stronger than intercommunity links. Specifically,

$$Q = \frac{1}{4m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{4m}\right)\delta\left(c_i, c_j\right), \tag{22}$$

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 nodes i and j belong to the same community, and 0 otherwise. Essentially, Q approaches 1 when the partitions are considered good.

(ii) Conductance (CON) [52, 53] measures the separability of a community across the fraction of outgoing local volume of links in the community, which is defined as

$$\mathrm{CON}\left(C\right) = \frac{\sum_{i\in C, j\in C'} A_{ij}}{\min\left(a\left(C\right), a\left(C'\right)\right)}, \tag{23}$$

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.

(iii) Triangle Participation Ratio (TPR) [53] measures the fraction of triads within the community C.

$$\mathrm{TPR}(C) = \frac{\left|\left\{v_i \in C,\ \left\{\left(v_j, v_k\right) : v_j, v_k \in C,\ \left(v_i, v_j\right), \left(v_j, v_k\right), \left(v_i, v_k\right) \in E\right\} \neq \emptyset\right\}\right|}{|C|}, \tag{24}$$

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

5.5. 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 [28]. Specifically, the parameters obtained were $(p = 0.5, q = 4)$, $(p = 0.25, q = 0.25)$, $(p = 1, q = 0.25)$, and $(p = 0.5, q = 1)$ for Karate, PolBlogs, Cora, and PubMed, respectively. As for DeepWalk, the parameters used are $d = 128$, $r = 10$, $l = 80$, and $k = 10$ which were the suggested values [27]. 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 [39].

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), and (32-8) for Karate, PolBlogs, Cora, and PubMed, respectively. These configurations are determined empirically as suggested in [15]. 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 was taken for both discriminative and generative models. All experiments were conducted on an Ubuntu 16.06 LTS machine with 64 GB of RAM and two GeForce GTX 1080 Ti graphics cards.

5.6. Experiment Results. We first compare our result with 8 baseline methods on several state-of-the-art methods that employ unsupervised network embedding, except SBM: the only generative model that does learn a network embedding. Since VGAE is nonclustering, the two-step approach for clustering was applied, i.e., obtaining the latent vectors and subsequently applying k-means. The $*$ symbol denotes methods that were confined to structural information only.

5.6.1. Synthetic Dataset Performance. Figure 2 depicts the performance of the proposed model in comparison to other methods. In Figure 2(a), VGAECD can be seen as a strong performer when $z_{\text{out}} \geq 4$. On the LFR benchmark graph in Figure 2(b), 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 to other methods. In both cases, the performance was as expected since the mixing parameter (z_{out} and μ) is consistent with the study recoverability limit in planted partitions [42, 43].

5.6.2. Empirical Dataset Performance. Experiments performed on four different empirical datasets are shown in Tables 2, 3, 4, and 5 for Karate, PolBlogs, Cora, and PubMed, respectively. We measure the performance of clusters found using metrics as proposed in the Section 5.4 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 Ratroio (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 are 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.

5.6.3. 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 Figure 3 in comparison to VGAE. In contrast to VGAE, the proposed method can achieve convergence at a faster rate.

5.6.4. 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 4.2. 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:

$$\omega = \lambda \omega^{\text{planted}} + (1 - \lambda)\, \omega^{\text{random}}. \tag{25}$$

Table 2: 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
Louvain	0.4900	1.5205	0.3235	**0.4188**	0.2879	0.7333
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 3: Experimental results on PolBlogs dataset.

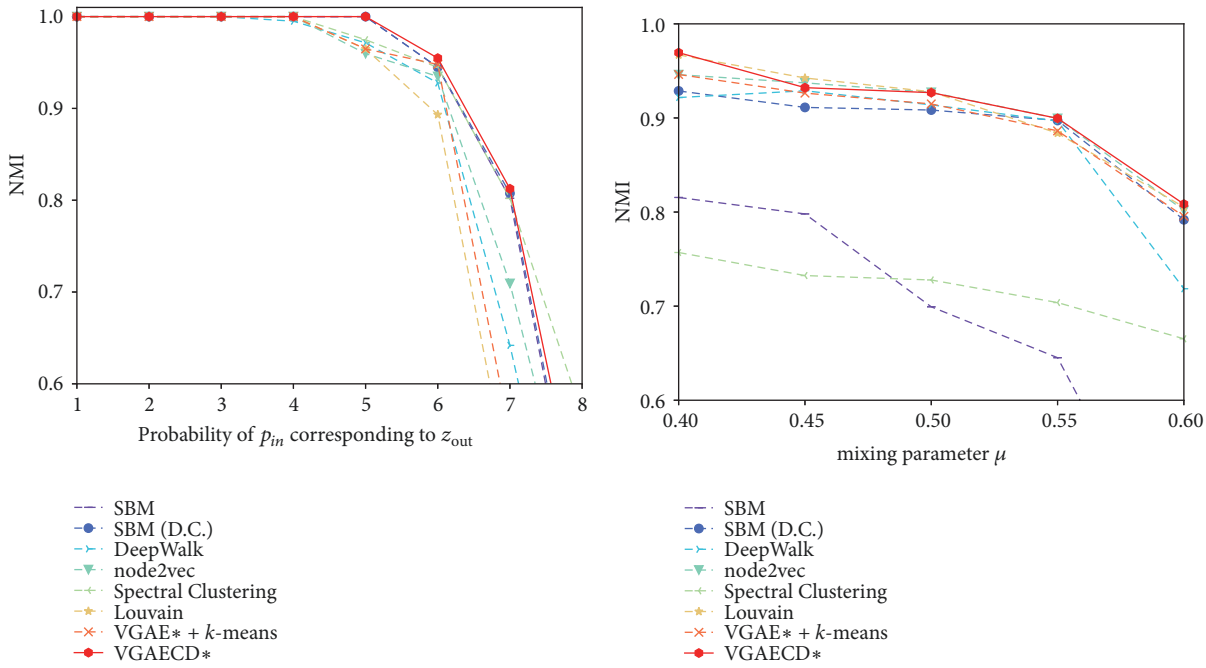
	NMI (↑)	VI (↓)	ACC (↑)	Q (↑)	CON (↓)	TPR (↑)
Spectral Clustering	0.0014	1.1152	0.4828	-0.0578	0.5585	0.7221
Louvain	0.6446	1.0839	0.9149	0.2987	0.8130	0.1922
DeepWalk	0.7367	1.0839	0.9543	0.0980	0.3873	0.6870
node2vec	0.7545	0.8613	0.9586	0.1011	0.3827	0.6863
Stochastic Blockmodel	0.0002	1.2957	0.4905	-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 4: 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**
Louvain	0.4336	4.0978	0.0081	**0.8142**	**0.0326**	0.2821
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 5: 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**
Louvain	0.1983	3.6667	0.0954	**0.7726**	0.1388	0.1592
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



(a) GN benchmark graph (b) LFR benchmark graph

Figure 2: Comparative performance of VGAECD against other methods on synthetic networks.

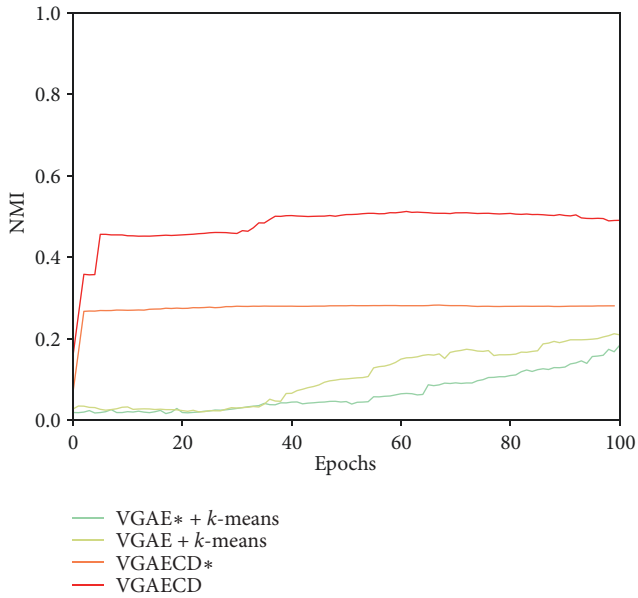


Figure 3: NMI convergence comparison.

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 specified as

$$\omega^{\text{planted}} = (n_1, n_2, n_3, n_4), \tag{26}$$

where n_c denotes the number of draws from the mixture model with μ_c and σ_c. In (26), we specify the number of nodes drawn for four different communities. A generated matrix $\widetilde{\mathbf{A}}$ can be obtained as shown in decoder part of Algorithm 1. Ideally, each node is represented by $\mathbf{z}$, and the Hadamard product between $\mathbf{z}_i$ and $\mathbf{z}_j$ determines the likelihood of edge connectivity between nodes i and j which is obtained after the nonlinearity $\sigma(\cdot)$ function.

5.6.5. Network Visualization. Community assignments for Cora dataset are visualized in Figure 4. 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 larger 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. This generalization of p and q allows node2vec to explore more locations. In contrast, DeepWalk is highly 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 and Louvain appear to struggle in finding a community structure, even though they performed very well on synthetic benchmark graphs. Louvain in particular had a very competitive NMI score, but visually, the results are not very satisfactory.

(a) Ground truth (b) VGAECD (c) Spectral clustering

(d) Louvain (e) DeepWalk (f) VGAE + k-means

(g) SBM (D.C.) (h) node2vec

Figure 4: Visualization of community assignment on Cora Dataset (largest connected component).

6. Conclusion

In this paper, we propose a novel community detection algorithm termed Variational Graph Autoencoder for community detection (VGAECD). It generalizes VGAE for community detection tasks. The model is capable of learning both features and structural information and encodes them into a community-aware latent representation. The low-dimensional representation learned differs from previous network representation methods. Concretely, the latent representations themselves are parameters to a probabilistic graphical model, i.e., the Gaussian Mixture Model. Therefore, this allows 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 feature information, it is capable of outperforming other methods in community structure recovery. Unlike other representation learning methods which require a two-step approach (applying k-means as the second step), VGAECD is a generative model capable of recovering communities in a single step. Moreover, in comparison to existing state-of-the-art generative models such as SBM, VGAECD is community structure definition agnostic. Specifically, nodes are not forced to be similar under a specific similarity measure. This is an advantage over other community detection algorithms where the definition of community structures is always assumed. This is a desirable feature in cases where networks can have a mixture of community structures, i.e., multilayer networks.

Data Availability

All data used in our research are publicly available data. Upon request, we could point them to their respective sources.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

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," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[2] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Acadamy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[3] S. Fortunato and D. Hric, "Community detection in networks: a user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.

[4] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 817–826, ACM, Paris, France, July 2009.

[5] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 1107–1116, Hong Kong, China, November 2009.

[6] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *Proceedings of the 15th IEEE International Conference on Computer Vision (ICCV '15)*, pp. 1026–1034, IEEE, Santiago, Chile, December 2015.

[8] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS '17)*, pp. 5580–5590, Long Beach, CA, USA, 2017.

[9] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," In Advances in Neural Information Processing Systems (NIPS '02), pp. 849–856, 2002.

[10] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1293–1299, 2014.

[11] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pp. 2252–2258, 2016.

[12] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, ACM, July 2008.

[13] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: marginalized graph autoencoder for graph clustering," in *Proceedings of the ACM on Conference on Information and Knowledge Management (CIKM '17)*, pp. 889–898, Singapore, November 2017.

[14] D. Chakrabarti and C. Faloutsos, "Graph mining: laws, generators, and algorithms," *ACM Computing Surveys*, vol. 38, no. 1, 2006.

[15] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proceedings of the Bayesian Deep Learning Workshop (NIPS '16)*, 2016.

[16] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 1965–1972, Melbourne, Australia, August 2017.

[17] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 2, Article ID 026113, 2004.

[18] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences of the United States of America* , vol. 104, no. 1, pp. 36–41, 2006.

[19] B. H. Good, Y.-A. de Montjoye, and A. Clauset, "Performance of modularity maximization in practical contexts," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 81, no. 4, Article ID 046106, pp. 1–20, 2010.

[20] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 76, no. 3, Article ID 036106, 2007.

[21] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proceedings of the International Symposium on Computer and Information Sciences*, pp. 284–293, Springer, 2005.

[22] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PLoS ONE*, vol. 6, no. 4, Article ID e18209, 2011.

[23] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, Article ID P10008, 2008.

[24] S. Agreste, P. De Meo, G. Fiumara et al., "An empirical comparison of algorithms to find communities in directed graphs and their application in web data analytics," *IEEE Transactions on Big Data*, vol. 3, no. 3, pp. 289–306, 2017.

[25] S. Cao, W. Lu, and Q. Xu, "GraRep: learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pp. 891–900, Melbourne, Australia, October 2015.

[26] T. Guo, S. Pan, X. Zhu, and C. Zhang, "CFOND: consensus factorization for co-clustering networked data," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 701–710, New York, NY, USA, August 2014.

[28] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 855–864, San Francisco, Calif, USA, August 2016.

[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral

filtering," in *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS '16)*, pp. 3844–3852, December 2016.

[30] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the World Wide Web Conference*, pp. 499–508, Lyon, France, April 2018.

[31] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: an approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, pp. 985–1042, 2010.

[32] A. W.-U. Ashraf, M. Budka, and K. Musial, "NetSim—The framework for complex network generator," in *Proceedings of the 22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES '18)*, September 2018, https://arxiv.org/abs/1805.10520.

[33] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of Erdős-Rényi graphs," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 85, no. 5, Article ID 056109, 2012.

[34] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Acadamy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[35] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 78, no. 4, Article ID 046110, 2008.

[36] P. Brodka and T. Grecki, "mLFR Benchamark: Testing Community Detection Algorithms in Multi-layered," *Multiplex and Multiple Social Networks*, 2014.

[37] P. Bródka, *A Method for Group Extraction and Analysis in Multilayer Social Networks [PhD Thesis]*, 2012, https://arxiv.org/abs/1612.02377.

[38] T. A. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[39] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 83, no. 1, Article ID 016107, 2011.

[40] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed Membership Stochastic Blockmodels," *Journal of Machine Learning Research*, vol. 9, pp. 1981–2014, 2008.

[41] D. B. Larremore, A. Clauset, and A. Z. Jacobs, "Efficiently inferring community structure in bipartite networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 90, no. 1, Article ID 012805, 2014.

[42] E. Abbe and C. Sandon, "Community detection in general stochastic block models: fundamental limits and efficient algorithms for recovery," *Foundations of Computer Science (FOCS)*, pp. 670–688, 2015.

[43] E. Abbe, A. S. Bandeira, and G. Hall, "Exact recovery in the stochastic block model," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 62, no. 1, pp. 471–487, 2016.

[44] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially Regularized Graph Autoencoder for Graph Embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pp. 2609–2615, Stockholm, Sweden, July 2018.

[45] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *In International Conference on Learning Representations (ICLR '14)*, 2014.

[46] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures & Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[47] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[48] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. Election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery (LinkKDD '05)*, pp. 36–43, ACM, 2005.

[49] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[50] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33rd International Conference on Machine Learning*, pp. 40–48, 2016.

[51] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, Article ID P09008, 2005.

[52] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: good, bad and spectral," *Journal of the ACM*, vol. 51, no. 3, pp. 497–515, 2004.

[53] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.