Pattern Recognition 122 (2022) 108334

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

# Graph Clustering via Variational Graph Embedding

Lin Guo, Qun Dai*

*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*

A R T I C L E I N F O

*Article history:*
Received 22 December 2020
Revised 2 August 2021
Accepted 17 September 2021
Available online 22 September 2021

*Keywords:*
Graph convolution neural network
Variational graph embedding
Graph clustering
Variational graph auto-encoder

A B S T R A C T

Graph clustering based on embedding aims to divide nodes with higher similarity into several mutually disjoint groups, but it is not a trivial task to maximumly embed the graph structure and node attributes into the low dimensional feature space. Furthermore, most of the current advanced methods of graph nodes clustering adopt the strategy of separating graph embedding technology and clustering algorithm, and ignore the potential relationship between them. Therefore, we propose an innovative end-to-end graph clustering framework with joint strategy to handle the complex problem in a non-Euclidean space. In terms of learning the graph embedding, we propose a new variational graph auto-encoder algorithm based on the Graph Convolution Network (GCN), which takes into account the boosting influence of joint generative model of graph structure and node attributes on the embedding output. On the basis of embedding representation, we implement a self-training mechanism through the construction of auxiliary distribution to further enhance the prediction of node categories, thereby realizing the unsupervised clustering mode. In addition, the loss contribution of each cluster is normalized to prevent large clusters from distorting the embedding space. Extensive experiments on real-world graph datasets validate our design and demonstrate that our algorithm has highly competitive in graph clustering over state-of-the-art methods.

## 1. Introduction

Graph is an important form of data representation that naturally exists in a variety of real-world scenarios, e.g., social graph and diffusion graph in social media networks, citation graph in research areas, knowledge graph and user interest graph in electronic commerce area etc. The point containing the property and the edge reflecting the nature of the connection between points are the main components of a graph. For example, in the social network graph, users or entities with different interests and preferences participate in the network to form points in the graph, and there are edges between nodes when there is an interaction (e.g., retweet/comment/follow in Twitter) between entities. Effective graph analysis can provide insight into the knowledge behind the data and benefit many applications. Examples include extracting medical entities from text [1], linking natural language text with entities in a knowledge graph [2], predicting the propagation user and identifying the domain expert according to information propagation between users [3].

Clustering, an efficient data analysis tool, can group similar graph nodes together according to the similarity of nodes and obtain the underlying attribute information of graph data, which plays an important role in user classification and community recommendation of social network graph. Graph clustering aims at dividing the graph nodes into several mutually disjoint groups. However, effective utilizing the node structures and features to graph clustering is not a trivial task.

Researchers have so far been working on graph clustering [4,5] and put forward many pioneering studies [6,7], such as nonnegative matrix factorization-based algorithms [8] and random walks-based algorithms [9]. Compared with most existing unsupervised learning models [10,11] based on graph structured data, the methods that also incorporate node features can be further categorized as spatial domain-based and spectral domain-based. The core of the spatial domain-based algorithms considering the spatial relationship between nodes lies in the fusion of node information, such as GraphSAGE [12], Graph Attention (GAT) [13] networks. Variational Graph Auto-Encoder (VGAE) [14], a typical spectral domain approach, learns meaningful embedded representations by utilizing Graph Convolutional Networks (GCN) [15] which is a layer-wise linear graph convolution model by first order approximation of localized spectral filters on the graph. VGAE converts a graph into a low dimensional space in which the graph information is maximally preserved. In view of the good embedded representation of VGAE, many variant algorithms based on this method have been developed, e.g., Marginalized Graph Auto-encoder (MGAE) [16], Adversarially Regularized Graph Auto-encoder (ARGA), Adversarially Regularized Variational Graph Auto-

* Corresponding author name: Jiang Jun road, Nanjing 211106, China.
*E-mail address:* daiqun@nuaa.edu.cn (Q. Dai).

https://doi.org/10.1016/j.patcog.2021.108334
0031-3203/© 2021 Elsevier Ltd. All rights reserved.

encoder (ARVGA) [17], Deep Graph Structured Clustering Network (DGSCN) [18], Variational Graph Autoencoder for Community Detection (VGAECD) [19] and Semi-implicit graph variational auto-encoder (SIG-VAE) [20]. These methods exploit the graph embedding approach to transform the graph data into a low-dimensional, compact, continuous feature space, and save the topological structure and vertex information. VGAE does have a powerful performance in learning interpretable latent representations for undirected graphs, but the current researches ignore the effect of node feature generation on the variational lower bound.

Most of these advanced graph node clustering algorithms convert the graph data into a low dimensional space first, and then utilize partitioned process such as K-means and spectral clustering to cluster in the feature space [21,22]. They ignore the potential relationship between graph embedding and clustering. In this paper, we reanalyze the graph clustering and ask whether graph embedding and clustering can be addressed jointly by graph data driven.

We take inspiration from recent work on deep embedding [23] which simultaneously optimizes deep embedding and clustering. This improvement, however, is obtained for traditional data, whereas our goal is to cluster the graph nodes by mining the topology structure and node features between the nodes.

In this paper, we propose an innovative end-to-end graph clustering framework which can simultaneously handle the graph embedding representation and nodes partition. The purpose of our framework is to cluster nodes with similar properties using the graph topology and node features. To this end, we define a parameterized nonlinear mapping to embed the graph space containing topology structure and node features into the low dimensional feature space, and optimize clustering objectives in the graph embedding space. To further refine the clusters, we implement a self-learning mechanism by constructing an auxiliary target distribution which is derived from the current soft cluster distribution [23], so as to realize the unsupervised clustering pattern. Moreover, the loss of each cluster is normalized to prevent large cluster from distorting the embedding space. In graph embedding respect, we propose a novel improved variational graph auto-encoder which takes into account the joint generative model about graph structure and node features. In a unified graph clustering framework, graph embedding and clustering centers are jointly optimized by taking advantage of the gradient optimization, with which a stable performance of clustering is achieved and the new graph embedding is more compact with respect to the cluster centers.

The main contributions of our work are summarized as follows:

- We introduce a novel graph auto-encoder, which considers the effect of federated generative model with respect to graph structure and node features on the variational lower bound, so as to acquire a relatively superior embedding for the initial step in the clustering framework.
- We design a new algorithm of graph nodes clustering, which exploits a self-training mode to optimize graph embedding and cluster centroids corporately.
- Extensive experiments on real-world graph datasets verify that our proposed method is highly competitive on the task of graph clustering and can significantly outperforms state-of-the-art methods.

## 2. Related Work

Graph clustering aims to reveal the inherent organizational structure of graph by grouping similar nodes together. In observations of whether to exploit the feature information of the node itself, we propose two taxonomies of graph clustering work, by categorizing graph clustering literature based on the structural graph clustering and the attributed graph clustering. In terms of structural graph clustering taxonomy, [24] assumes that every node is a linear combination of its neighbors in the embedding space, and the embedded representation of the node is obtained by optimizing the representation error of all nodes. Laplacian Eigenmaps [25] believes that nodes with higher similarity should be mapped closer, and describes the similarity by the Euclidean distance of the nodes in the embedded space. GraphEncoder [26] utilizes deep neural network (DNN) to learn graph similarity matrix. Graph Factorization [27] get the embedded nodes by factorizing the adjacency matrix. Deep Neural networks for Graph Representations (DNGR) [28] introduces a stacked denoising auto-encoder to factorize pointwise mutual information matrix for graph embedding. Another kind of random walk capture similarity between nodes. For instance, DeepWalk [9] first applies the language model to the network representation by using the connection relationship of nodes on the graph, generates the sequence nodes set of the language model through the short random walk sampling on the graph, and then utilizes the word vector technology to obtain the embedding of nodes. Similar to DeepWalk, node2vec [10] maintains high order proximity among nodes by maximizing the occurrence probability of subsequent nodes at a fixed length. However, these graph clustering methods only consider the topological structure of graph and ignore the essential information with respect to the graph node properties.

Attributed graph clustering focuses on both the graph topology and the nodes properties. The traditional method, like relational topic model (RTM) [29], a probabilistic generative model, defines a joint distribution over the words and citation to forecast document topic. Robust Multi-view Spectral Clustering (RMSC) [30] adopt a standard Markov chain applied to a shared low-rank transition probability matrix for multi-view clustering. On the basis of matrix factorization, Text-Associated DeepWalk (TADW) [31] merges text features of node into network representation learning. Based on the graph signal, Defferrard et al. [32] gives the Convolutional Neural Networks (CNN) form on the graph and designs the fast local convolution kernel. Kipf and Welling [15] implement GCN via a first-order approximation of localized spectral filters on graphs. GCN-based methods have demonstrated state-of-the-art performance on integrate node relationships and features. E.g., Graph Auto-Encode (GAE) [14] utilizes Variational Auto-Encoder (VAE) [33] and GCN to embed the graph nodes and achieves excellent realization in the task of link prediction. ARGA [17], on the other hand, adds adversarial regularization to GAE to learn the robust embedded representation. Whereas, GCN-based methods only take advantage of its effective feature extraction function and ignores the impact of node features included in graph embedding on the variational lower bound. At present, the node clustering methods based on graph embedding do not simultaneously learn low dimensional embedding space and optimize cluster centers. There is also some work [34,35] which puts a high value on the reconstruction of the graph structure and nodes features, through the method of graph autoencoder. However, this work focuses on the task of classification, and does not exploit the integrating graph embedding learning and clustering optimization.

## 3. Method

In this section, we formally study the high-efficiency Graph-based clustering method by utilizing the self-supervised variational graph embedding mechanism. Before diving into the details of the proposed approach, we first summarize the necessary notations in Table 1. Then the thorough explanation of GC-VGE is given. Fig. 1 presents the illustration for the proposed methodology. The graph clustering first utilizes the variational graph auto-encoder to obtain the initial low dimensional embedding in which the graph topolog-

**Table 1**
Definitions and mathematical symbols.

| Notations | Brief descriptions |
|---|---|
| $G = \langle V, E \rangle$ | An undirected graph |
| $V$ | Vertex-set in graph $G$ |
| $E$ | Edge-set in graph $G$ |
| $X$ | Node features matrix |
| $A$ | Adjacency matrix |
| $D = \mathrm{diag}(d_1, d_2, \cdots, d_n)$ | Degree matrix with $d_i = \sum_j A_{ij}$ |
| $Z$ | Node embedding matrix |
| $f(X, A)$ | Nonlinear mapping from the input layer to the embedding layer |
| $\phi_\theta = \mathrm{diag}(\theta)$ | Convolution kernel parameterized by $\theta$ |
| $U$ | Eigenvector matrix of the Laplacian $L$ |
| $\Gamma$ | Eigenvalue matrix with respect to the Laplacian $L$ |
| $\lambda_{\max}$ | Maximum eigenvalue of the Laplacian $L$ |
| $\Theta$ | Matrix of filter parameters |
| $W_k$ | Weight matrix in k-th encoder layer |
| $\varphi_k(\cdot)$ | Activation function in k-th encoder layer |
| $\zeta_j$ | The j-th clustering center |

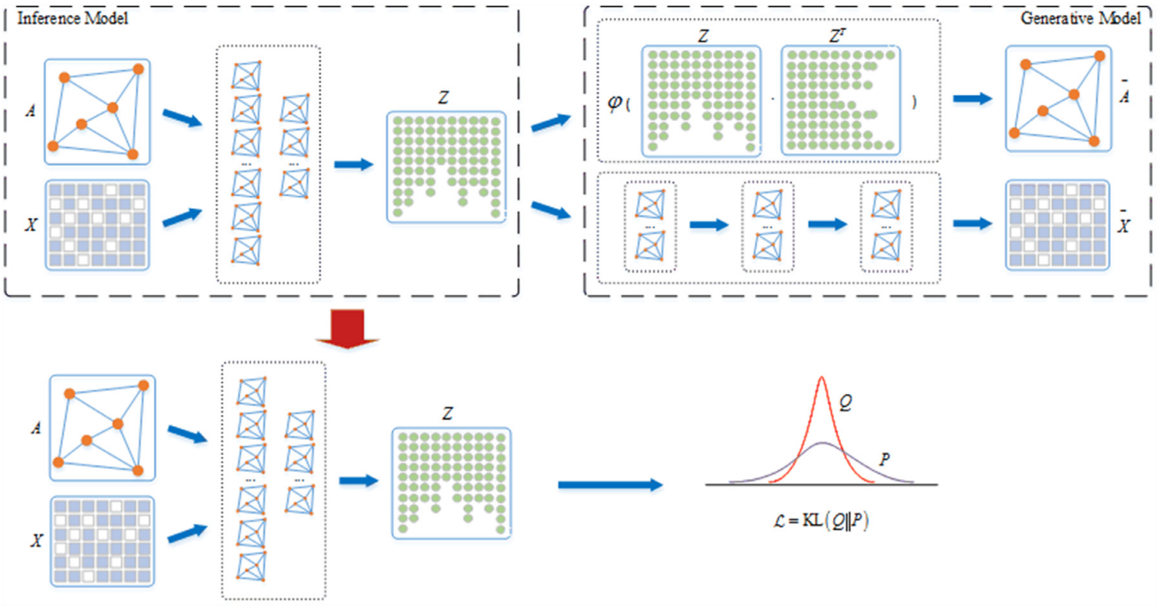

**Fig. 1.** The framework of graph clustering.

ical structural and nodes properties are preserved. After that, the initial graph embedding and clustering centers are updated jointly.

Given an undirected graph $G = \langle V, E \rangle$ with $n$ vertexes, where $V = \{v_1, v_2, \cdots, v_n\}$ is a vertex-set and $E$ is an edge-set. Node features are summarized in matrix $X = [x_1, x_2, \cdots, x_n]^T \in R^{n\times m}$, where $x_i \in R^m$ denotes a numeric feature vector of the vertex $v_i$. Edge structures information is stored in an adjacency matrix $A = \{A_{ij}\} \in R^{n\times n}$, where $A_{ij} = 1$ if $e_{ij} = \langle v_i, v_j \rangle \in E$, otherwise $A_{ij} = 0$. Here, the diagonal elements of adjacency matrix $A$ are set to 1, i.e., every node is connected to itself. The degree matrix $D = \mathrm{diag}(d_1, d_2, \cdots, d_n) \in R^{n\times n}$, where $d_i = \sum_j A_{ij}$.

Our main purpose is to divide n sample points of graph $G$ with feature information $X$ and node relation $A$ into different clusters. However, the probability distribution of data in the real world is usually defined on the data points in the high-dimensional space, so the cost of clustering such data directly is too high, and some meaningless features will affect the decision. Therefore, we take advantage of mapping the high-dimensional data to the low-dimensional space, where there's a host of conducive raw information. The commonly used methods, i.e., PCA and LDA, as the representative methods of feature reduction of linear mapping, can use attribute features to reduce the dimension of graph data by linear mapping. The linearization of graph embedding is more efficient for projection vector learning. However, if only linear mapping method is used to reduce the dimension of the graph data, more information of the original data may be lost during the mapping process. Moreover, LDA needs to take class information as input, which may not be easy to achieve intuitively in unsupervised learning. Secondly, for graph data, a kind of non-Euclidean structure data, we constructed a variational graph autoencoder which considers the enhancement effect of the joint generation model of graph structure and node attributes on the embedded output, in the nonlinear manner of neural network. Furthermore, a self-training mechanism was used to further enhance the prediction of node categories by constructing an auxiliary distribution, so as to realize the end-to-end unsupervised clustering task.

## 3.1. The Proposed Variational Graph Autoencoder

In this section, a detailed introduction of the proposed variational graph autoencoder is presented. The inference model and the generative model that we will construct act as two contributing factors to the final variational learning of the latent variable model. Theoretically, for an arbitrary point $x_i$, a corresponding latent variable $z_i$, which has a very similar relationship to $x_i$, can be generated. At the same time, for an arbitrary $A_{ij}$ representing the linking relation between nodes, it is ensured that there are corresponding latent variables $z_i$ and $z_j$ to generate structural information close to the homologous relationship $A_{ij}$. On account of the above measures, the important information stored in the hidden space can pave the way for dividing nodes into more separate partitions later. Specifically, the latent variable $z_i$ is sampled from an unknown distribution $p(Z)$ on an eigenspace $Z$. In order to formulate the concept accurately, we give the log-likelihood of $X$ and $A$:

$$\log p(X,A) = \log \int p(X,A,Z)dZ = \log \int p(X,A|Z)p(Z)dZ \quad (1)$$

The implicit representation in Eq. (1) reveals that the probability distribution of $X$ and $A$ in high-dimension could be dealt with the assistance of the hidden space $Z$. Due to the existence of non-essential information and the number of dimensions in real data, for solving the log-likelihood mathematically, $Z$ has to be tackled first. So the main goal is to automatically learn the latent variable $z_i$ by using the variational posteriori $q(Z|X,A)$ to estimate the true posteriori $p(Z|X,A)$. We use Jensen Inequality to transform Eq. (1):

$$\begin{aligned} \log p(X,A) &= \log \textstyle\int p(X,A,Z)dZ \\ &= \log \textstyle\int \frac{p(X,A,Z)}{q(Z|X,A)} q(Z|X,A)dZ \\ &= \log \left(\mathrm{E}_{q(Z|X,A)}\left[\frac{p(X,A,Z)}{q(Z|X,A)}\right]\right) \\ &\geq \mathrm{E}_{q(Z|X,A)}\left[\log \frac{p(X,A,Z)}{q(Z|X,A)}\right] = L_{low}(X,A) \end{aligned} \quad (2)$$

After obtaining the variational lower bound, Bayesian criterion is utilized for formula (2):

$$\begin{aligned} L_{\mathrm{low}}(X,A) &= \mathrm{E}_{q(Z|X,A)}\left[\log \frac{p(X,A,Z)}{q(Z|X,A)}\right] \\ &= \mathrm{E}_{q(Z|X,A)}\left[\log \frac{p(X,A|Z)p(Z)}{q(Z|X,A)}\right] \\ &= \mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z) + \log p(Z) - \log q(Z|X,A)] \\ &= \mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)] - \mathrm{E}_{q(Z|X,A)}[\log q(Z|X,A) - \log p(Z)] \\ &= \mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)] - \mathrm{KL}[q(Z|X,A)\| p(Z)] \end{aligned} \quad (3)$$

To handle Eq. (3), i.e., the maximum lower bound, the stochastic gradient descent (SGD) process is adopted to solve the optimization problem in Eq. (3) with the approximate correct $q(Z|X,A)$. It is worth mentioning that the right-hand side of Eq. (3) is the form of an auto-encoder, which encodes $X$ and $A$ as $Z$ through $q(Z|X,A)$ and decodes $Z$ to $X$ and $A$ via $p(X,A|Z)$. We will explore this extraordinary connection in more detail in the following subsections. In the generative process, the key is to sample the latent variables that may produce $x_i$ and $A_{ij}$. This means that we need a measurable $q(Z|X,A)$ which utilizes the observed values of $X$ and $A$ and gives us the generative values of $X$ and $A$. We expect the calculation of $\mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)]$ will meet with easier process in the case of the probability distribution $q(Z|X,A)$ obeyed by the latent variables.

### 3.1.1. Inference Model

Given the node features $X$ and the connection relationship matrix $A$ between nodes, it is also a challenge to make effective and efficient use of these information. We first construct a graph neural network $f(X,A)$ to embed the higher space into a low space, which is the primary step to project $q(Z|X,A)$. Specifically, we apply the convolution kernel $\phi_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in \mathrm{R}^n$ in the Fourier domain to the signal $x_i$ on the graph $G$, so as to obtain the convolution of the graph $G$:

$$\phi_\theta * x_i = U\phi_\theta U^T x_i \quad (4)$$

where $U$ is the eigenvector matrix of the normalized Laplacian $L = U\Gamma U^T$, especially $U^T x_i$ stands for the Fourier transform of $x_i$. $\phi_\theta$ represents a function of the eigenvalue $\Gamma$ with respect to $L$, which is recorded as $\phi_\theta(\Gamma)$. Since the time complexity of matrix multiplication with eigenvector matrix $U$ in Eq. (4) is $O(n^2)$, the eigendecomposition of $L$ will lead to highly expensive calculation for large graphs. Therefore, using the Chebyshev polynomials with K-th order to approximate the estimation could get $\phi_\theta(\Gamma) \approx \sum_{k=0}^{K} \theta_k T_k(2/\lambda_{\max}\Gamma - I)$ [36], where $\lambda_{\max}$ is the maximum eigenvalue of $L$, and $\theta_k$ denotes the K-th Chebyshev coefficient. Thus, the convolution of $G$ can be rewritten as:

$$\phi_\theta * x_i \approx \sum_{k=0}^{K} \theta_k T_k(\frac{2}{\lambda_{\max}} L - I)x_i \quad (5)$$

Formula (5) reduces the quadratic complexity to $O(n)$ which is linear in relation to the size of $G$. At this point, $L$ only needs to be obtained once, and the calculation of spectral convolution of $G$ is completely dependent on the K-th order calculation of nodes [32]. Further, the first order approximation of Chebyshev polynomials is applied with $\theta = \theta_0 = -\theta_1$ to prevent overfitting and $\lambda_{\max} \approx 2$ [15], that is, the convolution of a signal $x_i$ with a filter $\phi_\theta$ could be simplified as:

$$\phi_\theta * x_i \approx \theta_0 x_i + \theta_1(L - I)x_i = \theta x_i - \theta(L - I)x_i = \theta(D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x_i \quad (6)$$

The spectral convolution about the input matrix $F$ in multiple channels is denoted as $(D^{-\frac{1}{2}}AD^{-\frac{1}{2}})F\Theta$, where $\Theta$ is a matrix of filter parameters. So far, this convolutional filter function could be used to carry out the propagation of non-Euclidean signals on a neural network and further achieve the representation learning of the convolved signals. This also solves the difficulty of choosing a fixed convolution kernel in general neural networks to adapt to the irregularity of the whole graph when both of the number of nodes and the order of nodes are uncertain. Mathematically, we introduce the hierarchical propagation rule:

$$f_{k+1} = \varphi_k(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} f_k W_k) \quad (7)$$

where $W_k$ is a layer-specific trainable weight matrix, $\varphi_k(\cdot)$ denotes a layer-specific activation function, and $f_0 = X$ represents an input matrix of node feature vectors $x_i$. The simple and flexible model builds, in this way, a subtly and efficiently process for propagating associated information and node attributes from one layer to the next.

Based on the convolution theorem and the graph Fourier transform, the point-wise $L$-linearity rule for information, in each convolutional layer, is formulated, where multiple convolutional layers of Eq. (7) are superimposed to establish a graph-based neural network model. Define activate functions of $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ as $\mathrm{ReLU}(\cdot) = \max(0,\cdot)$ and a self-mapping function respectively [14]. Then, we could obtain:

$$f_1 = \varphi_1(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} f_0 W_0) = \mathrm{ReLU}(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} X W_0) \quad (8)$$

$$f_2 = \varphi_2(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} f_1 W_1) = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\mathrm{ReLU}(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} X W_0) W_1 \quad (9)$$

Up to now, the two-layer graph neural network is proposed based on layer-wise linear formulation for embedding the graph information into the continuous feature space. The next key point is how to reasonable use SGD to infer random latent variables in the embedding space. More specifically, the posteriori $q(Z|X,A)$ is identified as the crux during the sampling process. The commonly used Gaussian distribution is technically exploited for modeling. Then, this leaves us with the following expressions:

$$q(z_i|X,A) = N(z_i|\mu_i, \mathrm{diag}(\sigma_i^2)) \quad (10)$$

$$q(Z|X,A) = \prod_{i=1}^{n} q(z_i|X,A) \quad (11)$$

where $\mu_i$ and $\sigma_i$ are arbitrary functions with parameters which could be learned from the graph neural network, and $\mathrm{diag}(\sigma_i^2)$ is constructed as the diagonal matrix with respect to $\sigma_i$. By sampling multiple Gaussian distributions, the latent space inherits sufficient graph information, including the nodes attributes and the connected structure between nodes. Note that $Z$ embodies the main features of graph $G$. So far, the inference model embeds the discrete data points on the graph into a continuous space.

### 3.1.2. Generative Model

After solving the problem of how to transfer the graph information to the embedded space, the next thorny problem is how to select the latent variables to generate the graph information. First, we focus on how to select the latent variable $z_i$ to capture hidden information, that is, how to mathematically determine the specific prior distribution $p(Z)$ based on the absence of any prior knowledge of the latent variables. Considering that, in an arbitrary d-dimensional space, given the random variables under the normal distribution of d-dimensions, there exists a quite sophisticated function, through which these d-dimensional random variables can be mapped to the probability density distribution under the d-dimensional space to be constructed [37]. Therefore, the following assumptions are made:

$$p(z_i) = N(z_i|0, I) \tag{12}$$

$$p(Z) = \prod_{i=1}^{n} p(z_i) \tag{13}$$

Subsequently, the latent variable sample $z_i$ can be derived from a simple distribution. Using this universal function approximator and the process of simply learning a function, the independent and normally distributed values can be mapped to any latent variables that the model may require. And these latent variables are then mapped to $x_i$ and $A_{ij}$, on account of the phenomenon that the random latent variable $z_i$ in the embedding space can generate a variable close to $x_i$, and the arbitrary latent variables $z_i$ and $z_j$ correspond to the approximations of $A_{ij}$. According to this generated process, the joint probability distribution $p(X, A|Z)$ can be decomposed into:

$$p(X, A|Z) = p(X|Z)p(A|Z) \tag{14}$$

Here, $X$ and $A$ are conditional independent with respect to $Z$. Sampling $Z$ from $q(Z|X, A)$ gives an estimator for the expectation which generally converges much faster than from $N(0, I)$ as mentioned in the second paragraph of subsection 3.1. Therefore, the log-likelihood of joint probability density function (PDF) $p(X, A|Z)$ can be decomposed as:

$$\begin{aligned} \mathrm{E}_{q(Z|X,A)}[\log p(X, A|Z)] &= \mathrm{E}_{q(Z|X,A)}[\log p(X|Z) + \log p(A|Z)] \\ &= \mathrm{E}_{q(Z|X,A)}[\log p(X|Z)] + \mathrm{E}_{q(Z|X,A)}[\log p(A|Z)] \end{aligned} \tag{15}$$

In the light of the probability rule, the dependence of $X$ and $A$ on $Z$ could be confirmed, thus favorably computing Eq. (15). According to the above analysis that the linked relation $A_{ij}$ is determined by $z_i$ and $z_j$, we utilize the inner product between latent variables to represent the generated model of $A$, and adopt the Sigmoid function $S(x) = 1/(1 + e^x)$ to calculate the probability of $A_{ij}$ for obtaining the PDF of $A$, that is:

$$p(A_{ij} = 1|z_i, z_j) = S(z_i^T z_j) \tag{16}$$

$$p(A|Z) = \prod_{i=1}^{n} \prod_{j=1}^{n} p(A_{ij}|z_i, z_j) \tag{17}$$

Next, to infer the output distribution of $X$, we take advantage of different PDF according to multiple types of data which are divided into real-valued or binary. Specifically, when $x_i$ is a real valued instance, the Gaussian distribution is selected as the output distribution. When $x_i$ is binary, it can be a Bernoulli distribution. The mathematical expressions are:

$$p(x_i|z_i) = \begin{cases} N\left(x_i\middle|\hat{\mu}_i, \mathrm{diag}\left(\hat{\sigma}_i^2\right)\right), & \text{if } x_i \text{ is real} - \text{valued} \\ B\left(x_i\middle|\hat{\mu}_i\right), & \text{if } x_i \text{ is binary} \end{cases} \tag{18}$$

$$p(X|Z) = \prod_{i=1}^{n} p(x_i|z_i) \tag{19}$$

Accordingly, $Z$ under $q(Z|X, A)$ could determine the variational posteriori $p(A|Z)$ and $p(X|Z)$, that is, the foundation of Eq. (17) and Eq. (19) provides a relatively simple measure to calculate Eq. (15).

### 3.1.3. Variational Learning for the Latent Variable Model

The inference model $f(X, A)$ maps the graph node feature $X$ and the connected structure $A$ to a continuous hidden space where the latent variable samples in $Z$ comes from the approximate posterior $q(Z|X, A)$. Then, the latent variables $z_i$ are used as the input of $p(A|Z)$ and $p(X|Z)$, which are respectively equivalent to the probability density of data $X$ and adjacency matrix $A$ in the generative model. Hence, after the graph auto-encoder, the variational lower bound $L_{\mathrm{low}}$ is reformulated as follows:

$$\begin{aligned} L_{\mathrm{low}}(X, A) = {} & \mathrm{E}_{q(Z|X,A)}[\log p(X|Z)] + \mathrm{E}_{q(Z|X,A)}[\log p(A|Z)] \\ & - \mathrm{KL}[q(Z|X, A) \| p(Z)] \end{aligned} \tag{20}$$

Note that the first two items are regenerative items of $X$ and $A$ respectively, and the third term plays the role of regularization to prevent overfitting and ensure constructive capability of embedded space. To obtain the continuous latent space with large capacity of the graph information, the parameters and the latent variables learned in Eq. (20) are updated by utilizing the operation of full-batch gradient descent and reparameterization trick [33] respectively. At this point, the initial embedded space constructed by maximizing the variational lower bound provides the coding network (i.e., network weight) for the following joint clustering.

## 3.2. Graph Clustering with Manageable Latent Variables

After initializing the continuous hidden space $Z$ through the learning of the graph neural network, K-means, one of centroid-based clustering methods, is utilized to implement the clustering division in the embedded space, so as to obtain $c$ initial clustering centers $\zeta_j (j = 1, \cdots, c)$. This is a two-stage clustering method commonly utilized at present, while the feature space often obtained cannot meet the requirements of clustering. Certainly, it is possible to make superior partition decisions when the lower dimensional representation is close to perfection, and this is rarely simple. Hence, we exploit a self-supervised approach to simultaneously divide nodes and optimize low-dimensional space without supervision.

The clustering information of samples in the embedded space will be transmitted to the inference model by taking advantage of the back propagation, which controls the extracted coding part. For synchronously optimizing the hidden space and the decision of group division, $Z$ obtained by variational learning is regarded as the initial feature space, and $c$ cluster centers obtained by K-means play the role of initial cluster input.

Given the initial graph embedding and clustering centers, the graph clustering framework simultaneously optimizes them in an unsupervised manner to promote the clustering effect. Student's t-distribution is utilized to measure the similarity between embedded point $z_i$ and cluster center $\zeta_j$:

$$p(z_i, \zeta_j) = \frac{\left(1 + \left\|z_i - \zeta_j\right\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_j \left(1 + \left\|z_i - \zeta_j\right\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}} \tag{21}$$

where $\alpha$ are the degrees of freedom of the Student's t-distribution, $z_i \in Z$ corresponds to $x_i \in X$ after graph embedding and $p(z_i, \zeta_j)$ denotes the probability of assigning sample $z_i$ to cluster $\zeta_j$ (i.e., a soft assignment). Refer to the treatment of $\alpha$ by [23], we let $\alpha = 1$ for all experiments.

To update network parameters and optimize the clustering partition directly, the approach of auxiliary assignment [23] is implemented in an iterative learning way, so as to find the assignable decision with high confidence. Particularly, the assistant assignment $q(z_i, \zeta_j)$ is driven by soft assignment $p(z_i, \zeta_j)$, and the clustering partition of sample points in the embedded space is gradually updated with the help of optimizing the KL divergence between $p(Z, C)$ and $q(Z, C)$. This leaves us with the following expression:

$$L_{\text{clu}} = \text{KL}[q(Z, C) \| p(Z, C)] = \sum_j \sum_i q(z_i, \zeta_j) \log \frac{q(z_i, \zeta_j)}{p(z_i, \zeta_j)} \tag{22}$$

with

$$q(z_i, \zeta_j) = \frac{p^2(z_i, \zeta_j)/\sum_i p(z_i, \zeta_j)}{\sum_j (p^2(z_i, \zeta_j)/\sum_i p(z_i, \zeta_j))} \tag{23}$$

Here, the auxiliary distribution $q(z_i, \zeta_j)$, i.e., the approximate distribution of the original distribution $p(z_i, \zeta_j)$, could enhance the probability that the sample $z_i$ is a member of one partition, where the clustering centroid is $\zeta_j$. $\sum_{i=1}^{n} p(z_i, \zeta_j)$ represents the frequency of the soft cluster. Note that $L_{\text{clu}}$ exploits a self-supervised way to iteratively refine the embedding space and the clustering division without the constraint of supervisory information, as the auxiliary distribution $q(z_i, \zeta_j)$ is calculated by the target distribution $p(z_i, \zeta_j)$. Embedding samples dispersed by controlling $L_{\text{clu}}$ are divided in the same cluster with high confidence. For gradually optimizing the distribution of embeddings, the learned inference model is combined with clustering assignable learning, which assists in the co-establishment of the positive low-dimensional embedding and the assignment of graph nodes.

Eq. (22) shows how to perform clustering partition with high confidence in a data-driven manner. In detail, the nonlinear mapping $f(X, A)$ with beneficial attribute of the graph is learned by taking advantage of the inference model of the variational autoencoder, and then the clustering segmentation is carried out in the continuous embedded space. Repeat this operation with backpropagation until the desired clustering result is achieved, which requires that the constraint of the inference model is based on the parameters of $f(X, A)$ after initializing the nonlinear mapping.

### 3.3. The Error of the Variational Lower Bound

In the subsection 3.1, the Jensen's inequality is used to estimate the Variational Lower Bound of the log-likelihood. However, we don't know how many errors are introduced by such optimization. Next, let's take a closer look at this problem. In fact:

$$\log p(X, A) = L_{low}(X, A) + \text{KL}[q(Z|X, A) \| p(Z|X, A)] \tag{24}$$

That is, Eq. (24) has an extra term $\text{KL}[q(Z|X, A) \| p(Z|X, A)]$ compared to Eq. (2) (the proof is in Appendix A). Intuitively, $Z$'s variational posterior $q(Z|X, A)$ needs to be adopted to fit the true posterior $p(Z|X, A)$. In other words, the calculated lower bound $L_{low}(X, A)$ is to estimate the information loss caused by the joint log-likelihood of $X$ and $A$. More specifically, the information loss occurs during the generation of the hidden space. Note that the term $p(Z|X, A)$ in $\text{KL}[q(Z|X, A) \| p(Z|X, A)]$ cannot be calculated analytically. In consequence, we use an assumption that when an arbitrary high performance probability distribution $q(Z|X, A)$ is adopted, it will match $p(Z|X, A)$ as expected. In this case, the term $\text{KL}[q(Z|X, A) \| p(Z|X, A)]$ will be 0, and the log-likelihood $\log p(X, A)$ that we need to solve will be approximated to the calculated Variational Lower Bound $L_{low}(X, A)$.

**Table 2**
The statistics of the real-world graph datasets.

| Dataset | # Nodes | # Edges | # Features | # Classes |
|---|---|---|---|---|
| Cora | 2,708 | 5,429 | 1,433 | 7 |
| Citeseer | 3,327 | 4,732 | 3,703 | 6 |
| Pubmed | 19,717 | 44,338 | 500 | 3 |
| BlogCatalog | 5,196 | 171,743 | 8,189 | 6 |
| Flickr | 7,575 | 239,738 | 12,047 | 9 |
| Wiki | 2,405 | 17,981 | 4,973 | 17 |

### 3.4. Computational Complexity

GC-VGE consists of two parts: the first part contains both the pre-training of graph auto-encoder and the initialization of clustering nodes, and the second part is the training process. In the pre-training of graph auto-encoder, the time complexity of the inference model and the generative model are $O(|E|mhd)$ and $O(|E|d^2 + ndl)$ respectively with $|E|$ edges, $m$ input channels, $h$ feature maps of the hidden layer, $d$ filters of the output, $n$ nodes and $l$ depth of the network layer. In the initialization of clustering nodes, the time complexity is $O(ndct_1)$ with $t_1$ iterations and $c$ clusters. In the second part, the time complexity of KL divergence process is $O(nc)$. Since $d$ is usually much smaller than $m$, the time complexity with $t_2$ times is approximately $O(|E|mhdt_2 + ndlt_2 + nct_2 + ndct_1)$.

## 4. Experiments

In this section, the detailed experiments are presented to evaluate the proposed method of graph clustering which incorporates the encoding of graphs by means of variational graph embedding.

### 4.1. Datasets

In this work, we take advantage of six representative real datasets for experiments, including three commonly used mainstream citation network datasets, one webpage network dataset and two popular social network datasets.

The citation network data sets adopted are Cora, Citeseer and Pubmed respectively. For each dataset, every scientific publication plays as a node, citation relationships perform as edges, and the features of each document are words.

In Wiki data set, nodes are webpages from the cite of Wiki and are connected if one links the other.

The social network data sets utilized are BlogCatalog and Flickr respectively. The BlogCatalog data set consists of a network of social relationships among bloggers on the site of BlogCatalog. Users follow each other in the BlogCatalog to form social link information. Bloggers follow each other and form a network of relationships. We use the keywords in the description of the blog as attribute information for a short description of his/her blog. Labels represent the subject categories provided by the author, which are selected from a number of predefined categories that indicate the blog's interests.

The Flickr dataset is built from a social network of users interacting with each other on the online community platform of Flickr. Nodes in the network represent users, while edges denote the friendship formed by users sharing pictures and videos of their interest with each other in the network. Users under the same label have the same interests. The statistics for the benchmark dataset are listed in Table 2.

### 4.2. Baselines and Metrics

For analyzing the performance of GC-VGE, thirteen prominent algorithms, including K-means, spectral clustering (Spectral), DeepWalk [9], deep neural networks for graph representations (DNGR)

[28], robust multi-view spectral clustering (RMSC) [30], structured graph learning with multiple kernel (SGMK) [11], text-associate DeepWalk (TADW) [33], adversarially regularized graph autoencoder (ARGA) and adversarially regularized variational graph autoencoder (ARVGA) [17], dirichlet graph autoencoder (DGAE) and dirichlet graph variational autoencoder (DGVAE) [38], graph autoencoder (GAE) and graph variational auto-encoder (VGAE) [14] are selected as baselines. K-means and Spectral are the two commonly used foundation-laying clustering methods, with K-means utilizing the attribute of the graph and Spectral taking the topological structure of the graph as input. DeepWalk exploits the measure of truncated random walk to learn the social representation of a network from the input graph, which is able to adapt to changes in the network. DNGR adopts the graph structure in a low-dimensional vectorization manner, and instead of using traditional random sampling, it obtains the graph structure information through random surfing and extracts features in a manner of stacked autoencoder. RMSC uses the graph structure and node features to cluster the graph nodes in a multi-view way. SGMK utilizes both local and global structure information by self-expressiveness learning of samples. TADW treats DeepWalk as a process of matrix decomposition and adds nodes information to represent and learn from the graph in a matrix decomposition. In contrast to the above method, GAE establishes an auto-encoder based on GCN to learn graph with an unsupervised neuronal network. VGAE exploits a variational auto-encoder to learn the embedding graph. And on this basis, DGAE and DGVAE uses the Dirichlet distributions on the latent variables. In addition, ARGA and ARVGA make improvements on the basis of GAV and VGAE respectively, introducing a adversarial idea to learn the embedding.

In order to evaluate the performance of the proposed algorithm on the task of graph nodes clustering, we utilize five widely used metrics, namely, clustering accuracy (Acc), normalized mutual information (NMI), F-Score (F1), adjusted rand index (ARI), and precision (P).

### 4.3. Experimental Settings

For GC-VGE, we train 200 iterations, in both the pre-training stage and the clustering stage, with a learning rate of 0.01. Discriminatively, for Pubmed (considering the increase of the number of nodes and the inherited embedding parameters with the required attributes of relatively stable in the clustering phase), we set 1000 iterations in the pre-training stage and reduce the learning rate to 0.001 in the clustering stage. The initial number of K-means is set to 20 times to obtain relatively superior initial clustering centers. In all experiments, we construct a 32-dim hidden layer and a 16-dim embedding layer with the Adam [39] optimizer. Our method is implemented in tensorflow, a powerful deep learning system. In order to reduce the computation cost, we takes advantage of the sparse coding to meet the running conditions of the graph datasets we adopted. All models implemented in our experiments are performed on a machine equipped with an Intel Xeon CPU E5-2620 v4 @ 2.10 GHz and 16.00 GB RAM.

For the remaining baselines, we implement the codes released by the authors and tune the settings described in the corresponding papers. In particular, for DeepWalk, the number of random Walks is 10, the embedded dimension of each node is 128, and the path length of each random walk is 80. For DNGR, three encoders are set up, and the number of neurons in the first and second hidden layers is 512 and 256 respectively. The regularization parameters of RMSC is set to 0.005. SGMK adopts 12 kernels rescaled in [0, 1], i.e., seven Gaussian kernels, one linear kernel and four polynomial kernels. TADW sets the SVD to reduce the dimension to 200, and uses DeepWalk to embed the vector to 80 dimensions with the regularization as 0.2. For GAE and VGAE, an encoder with a 32-dimension hidden layer and a 16-dimension embedded layer is built, with 200 iterations and a learning rate of 0.01. DGAE and DGVAE set the Dirichlet prior to 0.001 on the basis of VGAE parameter settings. For ARGA and ARVGA, encoders with a 32-dimension hidden layer and a 16-dimension embedded layer are constructed, and the discriminators are constructed with two hidden layers which consist of 16-neuron and 64-neuron respectively Each experiment is repeated for 10 times.

**Table 3**
The values (%) of the clustering metric on the Cora data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 33.719 | 10.219 | 28.550 | 5.416 | 22.100 |
| Spectral | Graph | 31.466 | 9.699 | 29.674 | 0.351 | 18.078 |
| DeepWalk | Graph | 56.204 | 39.870 | 47.061 | 32.181 | 50.483 |
| DNGR | Graph | 44.398 | 33.316 | 34.686 | 15.869 | 27.863 |
| RMSC | Both | 36.298 | 17.072 | 26.689 | 12.287 | 29.162 |
| SGMK | Both | 50.435 | 37.942 | 43.831 | 27.534 | 39.490 |
| TADW | Both | 55.007 | 36.596 | 41.520 | 26.400 | 36.505 |
| ARGA | Both | 60.842 | 42.212 | 60.493 | 36.885 | 63.685 |
| ARVGA | Both | 62.832 | 45.930 | 63.174 | 38.002 | 64.807 |
| DGAE | Both | 61.258 | 44.297 | 59.421 | 36.014 | 62.837 |
| DGVAE | Both | 64.426 | 47.641 | 62.698 | 38.424 | 64.905 |
| GAE | Both | 60.347 | 44.858 | 58.723 | 36.738 | 61.399 |
| VGAE | Both | 63.567 | 47.454 | 63.754 | 39.427 | 65.648 |
| GC-VGE | Both | **70.676** | **53.574** | **69.482** | **48.154** | **70.517** |

**Table 4**
The values (%) of the clustering metric on the Citeseer data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 38.158 | 16.727 | 32.348 | 12.162 | 25.592 |
| Spectral | Graph | 37.893 | 17.690 | 32.764 | 12.862 | 25.615 |
| DeepWalk | Graph | 39.489 | 14.044 | 35.346 | 13.974 | 35.599 |
| DNGR | Graph | 34.271 | 18.497 | 29.136 | 4.174 | 19.889 |
| RMSC | Both | 43.818 | 19.818 | 31.840 | 17.580 | 32.685 |
| SGMK | Both | 49.764 | 27.619 | 37.031 | 25.718 | 42.902 |
| TADW | Both | 55.504 | 33.803 | 45.336 | 30.542 | 38.937 |
| ARGA | Both | 40.655 | 18.385 | 40.821 | 12.362 | 49.909 |
| ARVGA | Both | 44.881 | 21.971 | 43.686 | 17.174 | 49.021 |
| DGAE | Both | 44.582 | 22.008 | 41.956 | 16.053 | 46.139 |
| DGVAE | Both | 50.612 | 26.507 | 49.031 | 28.518 | 35.682 |
| GAE | Both | 43.992 | 21.407 | 41.461 | 16.960 | 45.658 |
| VGAE | Both | 44.370 | 21.165 | 42.917 | 15.206 | 49.206 |
| GC-VGE | Both | **66.607** | **40.909** | **63.390** | **41.516** | **64.894** |

### 4.4. Comparison between GC-VGE and the State-of-the-art Graph Methods

In this section, we analyze and report in detail the results of experiments compared with a variety of significant baseline algorithms. In order to visually compare the overall performance of each algorithm, we first present the boxplots of the overall results on the six data sets with respect to the five metrics, which are shown in Fig. 2. It is not difficult to find that GAE, VGAE, DGAE, DGVAE, ARGA, ARVGA and DeepWalk have similar performance in the second tier, and GC-VGE is always superior to the comparison experiments. The horizontal line in the box represents the median, and the square represents the mean. As can be seen from the figure, GC-VGE has the largest mean value among all experimental methods. The values of mean (the square) and median (the horizontal line in the box) reflect that GC-VGE has better stability in all experiments. This is because GC-VGE not only exploits the nodes features and graph typology to maximize the use of available information in the low-dimensional space, but also utilizes the architecture of data driven to simultaneously learn the embedding space and the clusters.

The specific experimental results are listed in Table 3-8. In particular, compared with approaches which only adopt a single at-

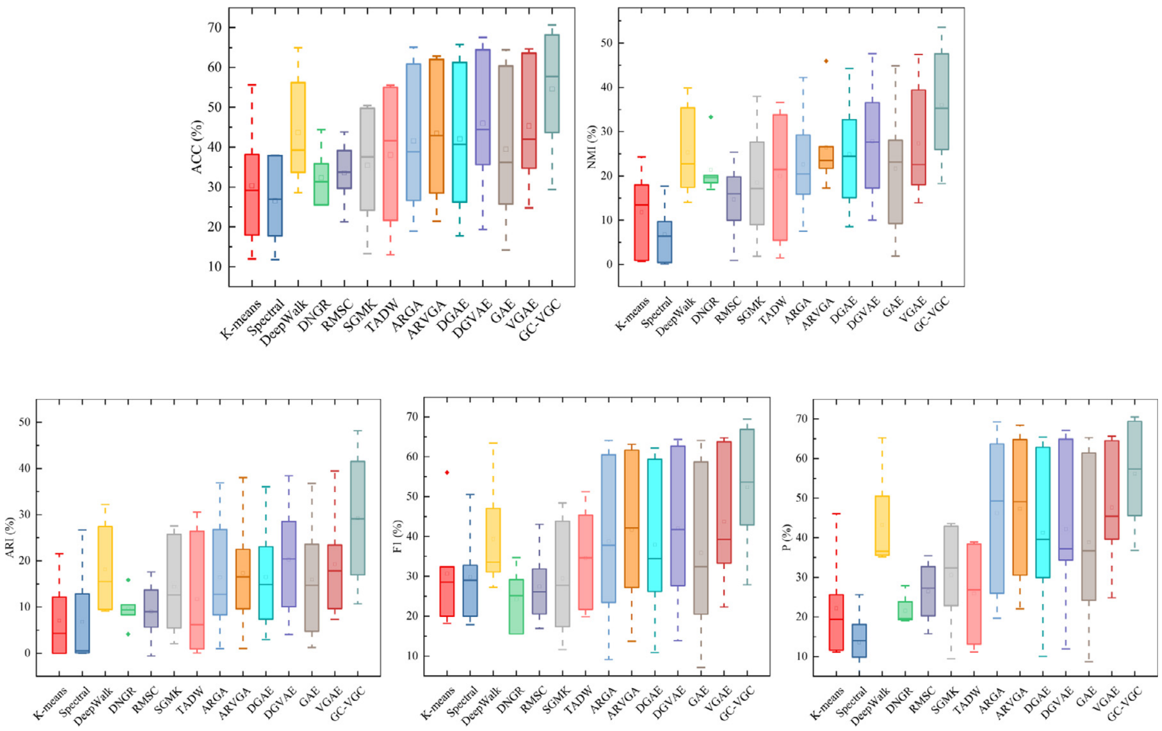

**Fig. 2.** The boxplots of all results on five metrics.

**Table 5**
The values (%) of the clustering metric on the BlogCatalog data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 17.950 | 0.912 | 28.556 | -0.004 | 16.694 |
| Spectral | Graph | 22.394 | 9.250 | 28.335 | 0.636 | 16.970 |
| DeepWalk | Graph | 33.684 | 19.064 | 31.093 | 9.484 | 35.884 |
| DNGR | Graph | 35.856 | 19.203 | 28.303 | 10.490 | 23.793 |
| RMSC | Both | 29.667 | 14.922 | 25.513 | 5.692 | 20.284 |
| SGMK | Both | 24.149 | 9.038 | 18.460 | 5.479 | 25.274 |
| TADW | Both | 21.624 | 5.477 | 27.792 | 0.967 | 17.137 |
| ARGA | Both | 37.052 | 22.479 | 34.776 | 13.121 | 48.620 |
| ARVGA | Both | 40.933 | 25.066 | 40.576 | 15.913 | 49.174 |
| DGAE | Both | 26.230 | 15.092 | 26.867 | 7.352 | 29.94 |
| DGVAE | Both | 35.614 | 17.246 | 34.504 | 10.063 | 38.724 |
| GAE | Both | 25.747 | 9.263 | 20.512 | 4.765 | 24.199 |
| VGAE | Both | 34.723 | 18.006 | 33.311 | 9.658 | 39.669 |
| GC-VGE | Both | **43.662** | **25.985** | **42.866** | **16.954** | **45.568** |

**Table 6**
The values (%) of the clustering metric on the Flickr data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 11.962 | 0.663 | 19.996 | 0.001 | 11.119 |
| Spectral | Graph | 11.790 | 0.439 | 20.000 | 0.001 | 11.119 |
| DeepWalk | Graph | 28.567 | 17.450 | 27.231 | 9.118 | 35.120 |
| DNGR | Graph | 28.407 | 16.984 | 21.982 | 10.469 | 19.002 |
| RMSC | Both | 21.247 | 9.977 | 16.919 | 5.658 | 15.760 |
| SGMK | Both | 13.251 | 1.841 | 11.618 | 2.081 | 9.418 |
| TADW | Both | 13.017 | 1.486 | 19.861 | 0.094 | 11.161 |
| ARGA | Both | 26.646 | 15.861 | 23.450 | 8.273 | 25.996 |
| ARVGA | Both | 28.479 | 17.250 | 27.214 | 9.594 | 30.537 |
| DGAE | Both | 17.736 | 8.504 | 10.883 | 2.952 | 10.084 |
| DGVAE | Both | 19.356 | 10.025 | 13.872 | 4.061 | 11.947 |
| GAE | Both | 14.206 | 1.914 | 7.140 | 1.224 | 8.725 |
| VGAE | Both | 24.749 | 13.956 | 22.268 | 7.319 | 24.826 |
| GC-VGE | Both | **29.386** | **18.271** | **27.874** | **10.706** | **36.812** |

tribute of graph as the input, it seems obvious that GC-VGE obtains the excellent results by jointly considering the graph topology and nodes features. Compared with the traditional method of RMSC, SGMK and the matrix decomposition method of TADW, GC-VGE improves the clustering results of the graph data set by relying on the strong performance of the explicable neural network. For example, in the Cora data set, GC-VGE improves 34.378%, 36.502%, 42.793%, 35.867%, and 41.355% compared with RMSC in the five metrics, namely ACC, NMI, F1, ARI and P. Depending on GCN, the graph structure and node features can be considered as the attributes of input information at the same time, so as to achieve good performance for GC-VGE, ARGA, ARVGA, DGAE, DGVAE, GAE and VGAE. However, GC-VGE exerts better in terms of performance based on an architecture of jointly optimizing graph embedded space and node assignment. For example, on the Citeseer data set, GC-VGE improves the performance of ARGA, ARVGA, DGAE, DGVAE, GAE and VGAE respectively by 25.952%, 21.726%, 22.025%,15.995%, 22.615% and 22.237% in terms of Acc. The reason might be that the joint form could promote each other and have a positive effect on the performance of the cluster. In addition, the end-to-end fusion mode is learned, which avoids noise introduction caused by low-dimensional embedding and cluster separation. Performance gap remains, of course, on real and approximate clustering partition, as there is a certain amount of data embedded in a low-dimensional compact space that could not be accurately estimated, but the overall structure of the clustering is significantly improved compared to previous advanced algorithms.

Remark: In Table 3-8, we use bold fonts to represent the best metric's values of each column.

## *4.5. Contribution of the Combined Optimum Design*

In this independent part, experiments are conducted to verify that cluster is the influential factor affecting the signal propagation in the network. We believe that the cluster structure plays a key role in the stable state of nodes in this propagation mechanism. Experiments are carried out to more clearly verify the validity of the target decision through jointly learning the embedding graph nodes and the nodes partition. First, we construct a

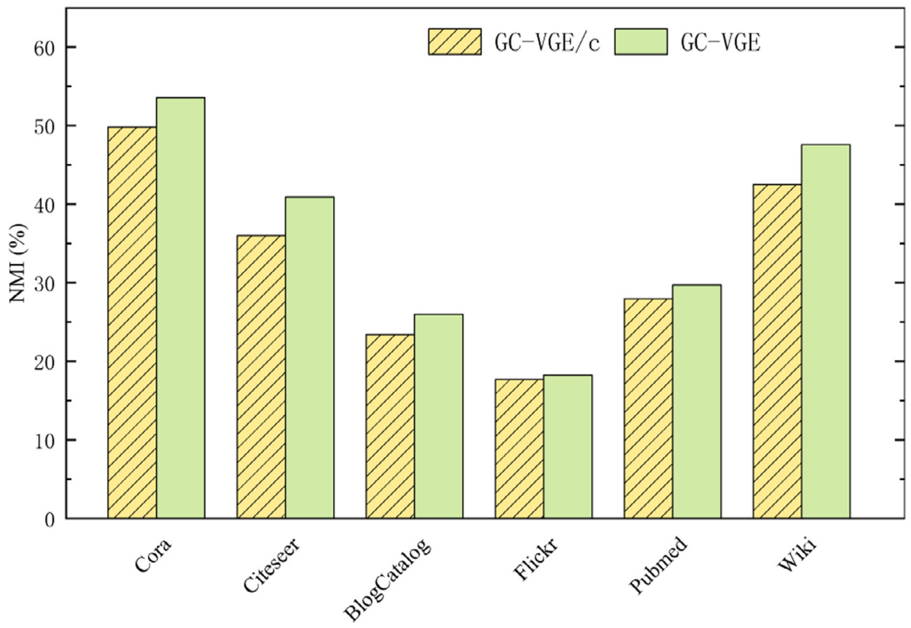

**Fig. 3.** The histogram shows the results of GV-VGE/c and GV-VGE.

**Table 7**
The values (%) of the clustering metric on the Pubmed data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 55.599 | 24.345 | 56.047 | 21.542 | 46.085 |
| Spectral | Graph | 37.986 | 0.103 | 50.549 | 26.678 | -0.023 |
| DeepWalk | Graph | 64.986 | 26.440 | 63.460 | 27.425 | 65.244 |
| DNGR | Graph | 25.539 | 20.113 | 15.575 | 8.293 | 19.267 |
| RMSC | Both | 39.175 | 0.903 | 43.061 | -0.562 | 35.424 |
| SGMK | Both | 47.910 | 12.465 | 48.397 | 15.160 | 43.568 |
| TADW | Both | 46.825 | 9.475 | 51.223 | 5.789 | 38.347 |
| ARGA | Both | 65.076 | 29.239 | 64.115 | 26.794 | 69.275 |
| ARVGA | Both | 62.011 | 26.622 | 61.665 | 22.468 | 68.416 |
| DGAE | Both | 65.742 | 26.870 | 62.155 | 23.061 | 65.418 |
| DGVAE | Both | 67.562 | 28.724 | 64.359 | 24.928 | 67.103 |
| GAE | Both | 64.430 | 24.853 | 64.076 | 23.571 | 65.267 |
| VGAE | Both | 64.679 | 23.947 | 64.775 | 23.415 | 64.531 |
| GC-VGE | Both | **68.182** | **29.706** | **66.875** | **29.763** | **69.393** |

**Table 8**
The values (%) of the clustering metric on the Wiki data set.

| Methods | Input | Acc | NMI | F1 | ARI | P |
|---|---|---|---|---|---|---|
| K-means | Feature | 24.599 | 17.965 | 18.163 | 3.202 | 11.607 |
| Spectral | Graph | 17.759 | 3.580 | 17.864 | 0.088 | 9.857 |
| DeepWalk | Graph | 39.077 | 35.358 | 31.722 | 17.018 | 37.276 |
| DNGR | Graph | 25.539 | 20.113 | 15.575 | 8.293 | 19.267 |
| RMSC | Both | 31.175 | 25.379 | 20.567 | 13.701 | 25.388 |
| SGMK | Both | 27.241 | 21.846 | 17.392 | 10.084 | 22.831 |
| TADW | Both | 36.412 | 33.451 | 21.711 | 6.532 | 13.141 |
| ARGA | Both | 18.952 | 7.541 | 9.122 | 1.047 | 19.665 |
| ARVGA | Both | 21.418 | 21.718 | 13.702 | 1.061 | 22.043 |
| DGAE | Both | 36.802 | 32.716 | 26.195 | 13.642 | 33.021 |
| DGVAE | Both | 38.254 | 36.552 | 27.635 | 15.831 | 34.330 |
| GAE | Both | 28.349 | 28.045 | 23.345 | 12.449 | 27.691 |
| VGAE | Both | 39.630 | 39.386 | 35.597 | 20.502 | 41.631 |
| GC-VGE | Both | **48.819** | **47.583** | **43.869** | **28.406** | **49.835** |

comparative group, where the constructed variational auto-encoder is utilized using the encoding to obtain a low-dimensional and dense vector representation of each node in the graph structure. Then K-means is implemented to embed sample points for clustering. This comparison experiment is marked as GC-VGE/c. The experimental parameters of GC-VGE/c are the same as those of GC-VGE. In Fig. 3, the histogram of GC-VGE/c and GC-VGE comparative experimental results are reported. On the whole, GC-VGE has certain advantages, mainly because the GC-VGE algorithm can not only enable the sample embedded in the space to inherit the node features of graph well, but also reflect the spatial attribute between nodes, so as to use the similar attribute between nodes to divide into a relatively dense cluster. More essentially, the traction effect of the downstream cluster centers on samples can react against the former, that is, the embedding space.

## 4.6. *Influence of Embedding Dimension*

In GC-VGE, the dimension of embedding layer is an important parameter. Since during the experiments, the other parameters in GC-VGE, such as the number of iterations, the learning rate and the initial number of K-means, are the most commonly used parameters. In order to save space, here we mainly discuss the sensitivity of embedding dimension. The quantity of hidden layer is set to twice the quantity of embedding layer. Fig. 4 exhibits the NMI values, with different number of embedding layer, on all the six real-word datasets.

From Figs. 4(a)-4(f), we notice that the clustering results improve steadily as the embedding dimension increased from 8 neurons to 16 neurons except for the visually difficult BlogCatalog dataset. This may mean that the increased embedding dimension in 8-16 facilitates the representation of more information about the data and hence good clustering. However, performance fluctuates as we enhance the number of neurons further, and sometimes we have access to a better score at higher dimension. Therefore, it could be inferred that the number of neurons in the embedded layer has an impact on the performance of GC-VGE, that is, the nonlinear information learned by the procedure through the dynamic interaction between the network structure and nodes affects the division of clusters. From the overall observation and analysis, we can roughly conclude that when the neuron number belongs to 16-128, the performance of GC-VGE varies in a small range due to the embedding dimension. That is, in this value range, the NMI score remains good on the whole.

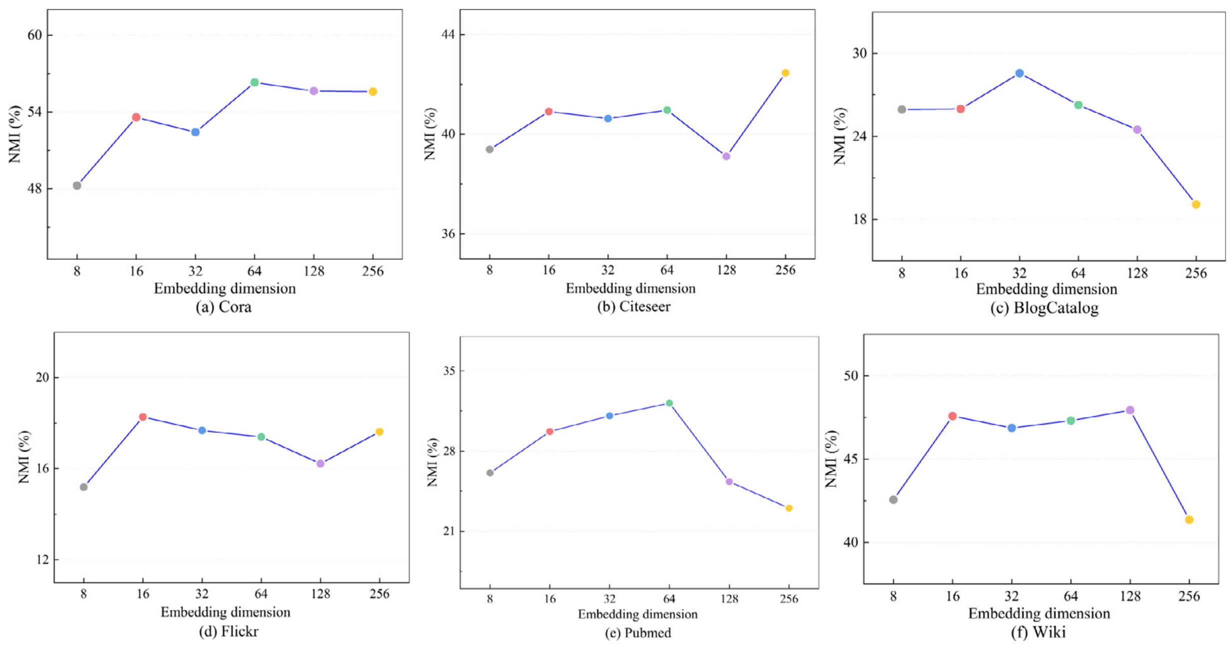

**Fig. 4.** Effect of different embedding dimensions.

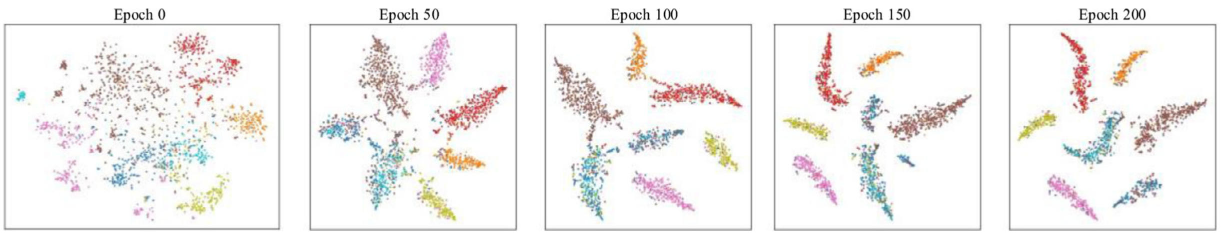

**Fig. 5.** Visualization of the proposed method on Cora data set. From left to right, the scatter diagram with different iterations is exhibited, and different colors indicate different clusters. Although the samples are somewhat mixed at epoch 200, they can still be divided in some way.

### 4.7. *Visualization*

In order to have a more intuitive understanding of how nodes are divided into a stable cluster when running the model GC-VGE, we chose the Cora data set to visualize the transformation of its embedding samples. Fig. 5 is a visual scatter diagram of embedded samples after reducing to 2-dim by t-SNE [40]. We utilize different colors to mark different ground-truth clusters. It can be seen that the nodes are scattered in the initial state and improved to some extent after pre-training. After adding the cluster information, the cluster gradually tends to the best state. Among them, the points in the same cluster have greater traction, forming several compact cluster sets, and different clusters are far away from each other.

## 5. Conclusion and Future Work

In this paper, we introduce a novel graph node clustering method with an improved graph variational auto-encoder method based on VGAE, which simultaneously learns the embedding and partitioning of non-Euclidean data. We believe that it is necessary to adopt the end-to-end clustering model compared with the embedding and clustering separation strategies. Upon the initialization of embedding and clustering centers, we establish a self-training framework, which exploit the KL divergence to refine the clustering assignment with high confidence. Obviously, it is extremely critical to have a low dimensional embedding succeeded the graph information, so we propose an innovative graph auto-encoder with generative graph structure and generative node properties. Comprehensive experimental results and extensive analysis demonstrate the effectiveness of our algorithms and methodological design choices, and the superior performance over the state-of-the-art unsupervised methods.

The proposed GC-VGE is suitable for the graph scenario, especially for the situation without available labels. Besides, the establishment of self-training mechanism and the prevention of distorting the embedded space of the large clusters are considered in the method, guiding to the scheme of an enhanced predictive ability.

Moreover, there is room for improvement. For instance, the unknown prior knowledge of the clusters is not taken into account in our algorithm. Directing at this flaws, hierarchical clustering will be considered in our future work. In this, the structural information will be discovered through the establishment of differentiable operators that can connect graph embedding to the segmentation criteria or aggregation criteria in the hierarchical algorithm. Additionally, we hope that future research will embark on such ideas for realistic issues, e.g., the digital image segmentation of users' portrait on a social network, the rapid and effective discovery of drugs, and the discovery of biological associations between species.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was supported by the National Key R&D Program of China (Grant Nos. 2018YFC2001600, 2018YFC2001602); and the National Natural Science Foundation of China under Grant no. 61473150.

## Appendix A

In this appendix, we give the error term $\mathrm{KL}[q(Z|X,A)\|p(Z|X,A)]$ between the variational lower bound $L_{low}(X,A)$ and $\log p(X,A)$ that we want to estimate, where $\mathrm{KL}[q(Z|X,A)\|p(Z|X,A)]$ reflects the similarity between $Z$'s variational posterior $q(Z|X,A)$ and the true posterior $p(Z|X,A)$. Then, we first have:

$$\begin{aligned}\mathrm{KL}[q(Z|X,A)\|p(Z|X,A)] &= \int q(Z|X,A)\log\frac{q(Z|X,A)}{p(Z|X,A)}dZ\\ &= \mathrm{E}_{q(Z|X,A)}[log\frac{q(Z|X,A)}{p(Z|X,A)}]\\ &= \mathrm{E}_{q(Z|X,A)}\log q(Z|X,A)-\log p(Z|X,A)\end{aligned}\tag{25}$$

where posterior distribution $q(Z|X,A)=\prod_{i=1}^{n}q(z_i|X,A)=\prod_{i=1}^{n}N(z_i|\mu_i,\mathrm{diag}(\sigma_i^2))$. Then we apply the Bayes Rule:

$$\begin{aligned}\log p(Z|X,A) &= \log\frac{p(X,A|Z)p(Z)}{p(X,A)}\\ &= \log p(X,A|Z)+\log p(Z)-\log p(X,A)\end{aligned}\tag{26}$$

where prior distribution $p(Z)=\prod_{i=1}^{n}p(z_i)=\prod_{i=1}^{n}N(z_i|0,I)$. According to Eq. (26), Eq. (25) can be rewritten as:

$$\begin{aligned}&\mathrm{KL}[q(Z|X,A)\|p(Z|X,A)]\\ &= \mathrm{E}_{q(Z|X,A)}[\log q(Z|X,A)-\log p(X,A|Z)-\log p(Z)+\log p(X,A)]\\ &= \mathrm{E}_{q(Z|X,A)}\left[\log\frac{q(Z|X,A)}{p(Z)}\right]-\mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)]+\log p(X,A)\\ &= \mathrm{KL}[q(Z|X,A)\|p(Z)]-\mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)]+\log p(X,A)\end{aligned}\tag{27}$$

Here, $\log p(X,A)$ comes out of the expectation because it does not depend on $Z$.

As $L_{low}(X,A)=\mathrm{E}_{q(Z|X,A)}[\log p(X,A|Z)]-\mathrm{KL}[q(Z|X,A)\|p(Z)]$, we then have:

$$\mathrm{KL}[q(Z|X,A)\|p(Z|X,A)]=\log p(X,A)-L_{low}(X,A)\tag{28}$$

The variational lower bound exerts an estimable part in calculating the loss of information generated by the joint log-likelihood of X and A. From the perspective of information theory, it is the amount of information loss generated during the generation of hidden space.

## References

[1] Zhilin Yang, William W Cohen, Ruslan Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, in: the International Conference on Learning Representations, 2016, pp. 40–48.
[2] Chenyan Xiong, Russell Power, Jamie Callan, Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding, in: the International World Wide Web Conferences, 2017, pp. 1271–1279.
[3] Na Zhao, Hanwang Zhang, Meng Wang, Richang Hong, Tat-Seng Chua, Learning content-social influential features for influence analysis, the International Journal of Multimedia Information Retrieval (2016) 137–149.
[4] Shudong Huang, Zhao Kang, Ivor W. Tsang, Zenglin Xu, Auto-weighted multi-view clustering via kernelized graph learning, Pattern Recognit 88 (2019) 174–184.
[5] Tong Wu, Graph regularized low-rank representation for submodule clustering, Pattern Recognit 100 (2020) 107145.
[6] Juan-Hui Li, Chang-Dong Wang, Pei-Zhen Li, Jian-Huang Lai, Discriminative metric learning for multi-view graph partitioning, Pattern Recognit 75 (2018) 199–213.
[7] Alper Aksac, Tansel Özyer, Reda Alhajj, CutESC: Cutting edge spatial clustering technique based on proximity graphs, Pattern Recognit 96 (2019) 106948.
[8] Fanhua Shang, L.C. Jiao, Fei Wang, Graph dual regularization non-negative matrix factorization for co-clustering, Pattern Recognit 45 (2012) 2237–2250.
[9] Bryan Perozzi, Rami Alrfou, Steven Skiena, DeepWalk: online learning of social representations, ACM Knowledge Discovery and Data Mining (2014) 701–710.
[10] Aditya Grover, Jure Leskovec, node2vec: Scalable Feature Learning for Networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[11] Zhao Kang, Chong Peng, Qiang Cheng, Xinwang Liu, Xi Peng, Zenglin Xu, Ling Tian, Structured graph learning for clustering and semi-supervised classification, Pattern Recognit 110 (2021) 107627.
[12] William L Hamilton, Zhitao Ying, Jure Leskovec, Inductive Representation Learning on Large Graphs, in: Annual Conference on Neural Information Processing Systems, 2017, pp. 1024–1034.
[13] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, Graph Attention Networks, the International Conference on Learning Representations, 2018.
[14] Thomas N. Kipf, Max Welling, Variational Graph Auto-Encoders, the Annual Conference on Neural Information Processing Systems Workshop on Bayesian Deep Learning, 2016.
[15] Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 889–898.
[16] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, Jing Jiang, MGAE: Marginalized Graph Autoencoder for Graph Clustering, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 889–898.
[17] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, Chengqi Zhang, Adversarially Regularized Graph Autoencoder for Graph Embedding, in: the International Joint Conference on Artificial Intelligence, 2018, pp. 2609–2615.
[18] Li Xunkai, Youpeng Hu, Yaoqi Sun, Ji Hu, Jiyong Zhang, Meixia Qu, A Deep Graph Structured Clustering Network, IEEEAccess 8 (2020) 161727–161738.
[19] Jun Jin Choong, Xin Liu, Tsuyoshi Murata, Variational Approach for Learning Community Structures, Complexity 2018 (2018) 1–13.
[20] Arman Hasanzadeh, Ehsan Hajiramezanali, Nick Duffield, Krishna Narayanan, Mingyuan Zhou, Xiaoming Qian, Semi-implicit graph variational auto-encoders, in: Annual Conference on Neural Information Processing Systems, 2019, pp. 10712–10723.
[21] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Philip S. Yu and Weixiong Zhang, A Survey of Community Detection Approaches:From Statistical Modeling to Deep Learning, arXiv: 2101.01669v1, (2021).
[22] Yaochen Xie, Zhao Xu, Zhengyang Wang, Shuiwang Ji, Self-Supervised Learning of Graph Neural Networks: A Unified Review, arXiv: 2102.10757v2, (2021).
[23] Junyuan Xie, Ross Girshick, Ali Farhadi, Unsupervised deep embedding for clustering analysis, in: the International Conference on Learning Representations, 2016, pp. 478–487.
[24] Sam T Roweis, Lawrence K Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 290 (2000) 2323–2326.
[25] Mikhail Belkin, Partha Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, in: Annual Conference on Neural Information Processing Systems, 2001, pp. 585–591.
[26] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, Tieyan Liu, Learning deep representations for graph clustering, in: the Pacific Rim International Conference on Artificial Intelligence, 2014, pp. 1293–1299.
[27] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, Alexander J Smola, Distributed large-scale natural graph factorization, in: the International World Wide Web Conferences, 2013, pp. 37–48.
[28] Shaosheng Cao, Wei Lu, Qiongkai Xu, Deep neural networks for learning graph representations, in: the Pacific Rim International Conference on Artificial Intelligence, 2015, pp. 1145–1152.
[29] Jonathan Chang, David M Blei, Relational Topic Models for Document Networks, in: the International Conference on Artificial Intelligence and Statistics, 2009, pp. 81–88.
[30] Rongkai Xia, Yan Pan, Lei Du, Jian Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: the Pacific Rim International Conference on Artificial Intelligence, 2014, pp. 2149–2155.
[31] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, Edward Y Chang, Network representation learning with rich text information, in: the Pacific Rim International Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[32] Michael Defferrard, Xavier Bresson, Pierre Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Annual Conference on Neural Information Processing Systems, 2016, pp. 3844–3852.
[33] Diederik P Kingma, Max Welling, Auto-Encoding Variational Bayes, the International Conference on Learning Representations, 2014.
[34] Amin Salehi, Hasan Davulcu, Graph Attention Auto-Encoders, in: International Conference on Tools with Artificial Intelligence, 2020, pp. 989–996.
[35] Keting Cen, Huawei Shen, Jinhua Gao, Qi Cao, Bingbing Xu, Xueqi Cheng, ANAE: Learning Node Context Representation for Attributed Network Embedding, arXiv: 1906.08745v3, 2019.
[36] David K. Hammond, Pierre Vandergheynst, Remi Gribonval, Wavelets on graphs via spectral graph theory, Applied and Computational Harmonic Analysis 30 (2011) 129–150.
[37] Luc Devroye, Sample-based non-uniform random variate generation, in: Proceedings of the 18th conference on Winter simulation, 1986, pp. 260–265.
[38] Jia Li, Tomas Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, Junzhou Huang, Dirichlet graph variational autoencoder, Annual Conference on Neural Information Processing Systems, 2020.

[39] Diederik P Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, the International Conference on Learning Representations, 2015.

[40] Laurens Van Der Maaten, Geoffrey E Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.

Lin Guo received the B.S. degree from Taiyuan Normal University, Taiyuan, China, in 2013, and the M.S. degree from Yunnan Minzu University, Yunnan, China, in 2016. She is working toward the Ph.D. degree in College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her research interests include pattern recognition, data mining and machine learning.

Qun Dai completed her M.S. degree in computer science at Nanjing University of Aeronautics and Astronautics (NUAA) in March, 2003, and then worked at the College of Information Science and Technology of NUAA as an assistant lecturer. There she received a Ph.D. degree in computer science in 2009. In 2010, she became an Associate Professor for the College of Computer Science and Technology of NUAA. Since 2015, she has become a Professor for the College of Computer Science and Technology of NUAA. Her research interests focus on neural computing, pattern recognition and machine learning.