ISSN: 1813-4890

A Community Detection Algorithm for Graph Autoencoders Based on Improved Encoding and Decoding Structure and Multiple Optimizations

Yiwen Shen

School of Information Technology and Engineering, Tianjin University of Technology and Education, Tianjin 300222, China

Abstract

With the widespread application of graph neural networks (GNNs) in graph representation learning, community detection, as an important task in graph analysis, has received increasing attention in recent years. This paper proposes a community detection method based on VGAE (Variational Graph Autoencoder) and K-means clustering, and experimentally evaluates this method on multiple academic graph datasets. The method first uses VGAE to extract the latent variable representation of nodes in the graph to capture the relationship between nodes and global dependency information in the graph structure, and then uses the K-means clustering algorithm for unsupervised community segmentation. We experimented with this method on multiple datasets, especially on academic literature datasets such as Cora, Citeseer, and PubMed, and comprehensively evaluated the algorithm performance through evaluation indicators such as NMI (Normalized Mutual Information), ARI (Adjusted Rand Index), and Modularity. Experimental results show that the VGAE + K-means method has better community division effect on these datasets than the classic DeepWalk + K-means and Node2Vec + K-means methods, especially in F1-Score. We further analyzed the clustering performance of Node2Vec and DeepWalk on different datasets, and pointed out that the VGAE model can better capture the complex structural relationships in the graph through the autoencoder structure, thereby effectively improving the quality of community division. This study not only demonstrates the potential of VGAE in the field of graph embedding, but also provides new ideas for community discovery tasks. By combining VGAE and K-means clustering algorithms, our model can automatically extract meaningful node representations from graph data and achieve accurate community division under unsupervised learning. Future work will be devoted to further optimizing the model, exploring more types of graph datasets, and studying how to use node features to further improve clustering effects.

Keywords

Community discovery, graph neural network, VGAE, K-means clustering, graph representation learning.

1. Introduction

In the real world, there are a wide variety of complex entity relationships. Since the graph structure can well express the complex associations between nodes and edges[1], the entity relationship can be abstractly expressed as a graph, and we can do more operations on this basis. For example, recommending friends to users through social networks, studying the associations between papers through citation networks, and analyzing the interaction relationships between proteins in biological systems through protein interaction networks. Among them, community discovery is an important research topic in the field of complex

ISSN: 1813-4890

networks[2]. Its goal is to divide complex networks into multiple subnetworks, and the nodes within each subnetwork are closely connected, while the connections between subnetworks are sparse. In recent years, convolutional neural networks have attracted attention due to their powerful modeling capabilities and have been widely used in image processing, natural language processing and other fields. However, traditional convolutional neural networks can only process Euclidean spatial data such as images, voices and texts, but cannot effectively process common graph data, because graph data is a typical non-Euclidean spatial data[3]. Therefore, graph convolutional networks were proposed and applied to graph data processing. With the continuous deepening of research and the increase in data scale, the computational complexity of graph data due to non-Euclidean spatiality and the spatial complexity of adjacency matrices are more prominent. We urgently need new deep neural network models[4]. Among them, graph representation learning is the mainstream method to solve these problems. It is necessary to apply graph representation learning methods to the field of community discovery. On the one hand, after the graph data is embedded in a specific vector space, the graph data can be transformed from non-Euclidean data to Euclidean data, which is helpful for the execution of downstream tasks. On the other hand, while realizing graph data compression, graph representation learning also alleviates the problem of data dimensionality disaster[5]. We usually use adjacency matrix to express the relationship between nodes in graph data. As the number of nodes in the graph increases, the dimension of the adjacency matrix increases in the form of power, which brings great computational complexity to traditional machine learning algorithms[6]. Graph representation learning can map high-dimensional sparse graph data into low-dimensional dense vectors, and compress the node feature vectors expressed by the adjacency matrix[7]. This means that we can use traditional machine learning algorithms to process and analyze graph data in the future[8].

Based on the existing research model, in order to obtain higher quality node representation, this paper makes the following improvements based on the existing problems of the VGAE model:

(1) Improve the encoding model: Improve the GCN encoding in the encoding stage to FAGCN module encoding, so that the weight coefficient changes in the encoding model are more flexible, and hidden variables with stronger representation capabilities can be obtained.

(2) Improve the decoding model: In the decoding stage, not only the adjacency matrix is reconstructed by using the inner product decoder, but also an inverse graph convolution decoder is constructed to reconstruct the feature matrix.

(3) Update the loss function: This study improves on the basis of VGAE. The loss function adds node feature matrix reconstruction loss and node feature distribution loss on the basis of the original loss function of VGAE, and multiple optimization objectives participate in the training process.

2. Introduction to Model

2.1. Current Model Analysis

Kipf and Welling proposed two graph auto-encoder models, GAE (Graph Auto-Encoder) and VGAE (Variational Graph Auto-Encoder), based on graph convolutional networks and variational Bayes[9]. GAE uses GCN as the encoding model, inputs the feature matrix and the adjacency matrix to obtain the embedded representation of the graph; in the decoding stage, the matrix inner product is used to perform the decoding operation and reconstruct the adjacency matrix of the original graph. The encoding operation of GAE is:

$$H = GCN(X, A) \tag{1}$$

ISSN: 1813-4890

where $H \in \mathbb{R}^{N\times f}$ is the embedded vector output by the GCN model in the encoding stage, $X \in \mathbb{R}^{N\times d}$ is the node feature matrix of the original graph, and $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix of the original graph. Formula (1) can only express the input and output elements of GCN in the encoding stage. In terms of the working principle of GCN, the literature [10] has formula (2):

$$GCN(X,A) = \tilde{A}ReLU\left(\tilde{A}XW_0\right)W_1 \quad (2)$$

Where $\tilde{A} = D^{-1/2}AD^{-1/2}$ is the normalization of the adjacency matrix, W_0 and W_1 are the parameter matrices of the two-layer convolutional network in the GCN model. In the literature [10], Kipf and Welling studied the application of the embedding vector obtained by the GCN model in downstream tasks such as link prediction and node classification. The difference is that in GAE, in addition to obtaining the embedding vector, it is also necessary to input the embedding vector into the decoding stage to reconstruct the adjacency matrix of the original graph. Unlike most autoencoder models, GAE does not invert the encoder model as the decoder model, but uses a simple matrix inner product operation as the decoder model to reconstruct the adjacency matrix, as shown in formula (3):

$$\hat{A} = \sigma(HH^T) \quad (3)$$

In formula (3), $\sigma(\cdot)$ represents a nonlinear activation function, HH^T is a matrix inner product operation, the stronger the embedding vector representation capability obtained by the encoder, the more similar the reconstructed adjacency matrix $\hat{A}$ is to the original graph adjacency matrix. Since the element value y of the adjacency matrix in the original graph is either 0 or 1, and the reconstructed adjacency matrix element $\hat{y} \in [0,1]$, the cross entropy loss function is used as the optimization target of the model, as shown in formula (4):

$$\mathcal{L}_{GAE} = -\frac{1}{N}\sum_{y\in A} y log\hat{y} + (1-y)log(1-\hat{y}) \quad (4)$$

like GAE, the GCN model responsible for encoding in VGAE combines the variational idea [11]. The embedding vector is calculated after sampling from the Gaussian distribution. To determine the Gaussian distribution of the input data, the mean and variance of the original graph data are required. Therefore, it is necessary to use GCN to fit the mean and variance of the input data in the encoding model of VGAE. The formulas are shown in equations (5) and (6) :

$$\mu = GCN_u(X,A) \quad (5)$$

$$log\sigma^2 = GCN_\sigma(X,A) \quad (6)$$

rrespondingly, due to the use of variational ideas in VGAE, the loss function also needs to be improved accordingly. The loss function of VGAE is composed of cross entropy and relative entropy (KL divergence), as shown in equation (7):

$$\mathcal{L}_{VGAE} = \mathbb{E}_{q(H|X,A)}[\log p(A|Z)] - KL\left[[q(Z|X,A) \parallel p(H)]\right] \quad (7)$$

Where $\mathbb{E}_{q(H|X,A)}[\log p(A|Z)]$ is the expectation, $KL\left[[q(Z|X,A) \parallel p(H)]\right]$ is the KL divergence, which is used to measure the difference between distributions. $q(Z|X,A)$ is the distribution calculated by the GCN function, and $p(H)$ uses the standard Gaussian distribution.

2.2. VGAE-ISMO Model

2.2.1. Encoder structure design

(1) Node feature update design

The input of the encoder includes the node feature matrix $X \in \mathbb{R}^{N\times F}$ and the adjacency matrix $A \in \mathbb{R}^{N\times N}$, where N represents the number of nodes and F represents the dimension of the node features. In the GCN network, the essence of each graph convolution operation is to perform a weighted summation operation on the neighboring nodes around the node. $\tilde{A} = D^{-1/2}AD^{-1/2}$ is a standardized operation on the adjacency matrix A. The node feature update formula of the graph convolution in GCN is:

ISSN: 1813-4890

$$h_i^{(l+1)} = \sum_{j \in N_i} \frac{1}{\sqrt{d_i d_j}} \sigma\left(W^{(l)} h_j^{(l)}\right) \quad (8)$$

As can be seen from the above formula, in the GCN encoding process, the weight coefficient between any two nodes is fixed to $\frac{1}{\sqrt{d_i d_j}}$, which affects the flexibility and generalization ability of the model. FAGCN introduces adaptive weights α_{ij}^G to dynamically determine the contribution of each neighbor node j to the feature update of the target node i. The adaptive weight enables the model to dynamically adjust the influence of neighbors according to the similarity of node features, thereby enhancing the flexibility and expression of feature aggregation. Therefore, the node feature update formula after introducing adaptive weights is:

$$h_i^{(l)} = \varepsilon h_i^{(0)} + \sum_{j \in N_i} \frac{\alpha_{ij}^G}{\sqrt{d_i d_j}} h_j^{l-1} \quad (9)$$

The first term $\varepsilon h_i^{(0)}$ is the residual connection, which is used to retain the initial features and prevent the features from being transitionally smoothed. The second term $\sum_{j \epsilon N_i} \frac{\alpha_{ij}^G}{\sqrt{d_i d_j}} h_j^{l-1}$ is the adaptive weighted aggregation of neighboring nodes.

(2) Latent variable generation

In traditional GAE and VGAE, latent variables are generated by GCN fitting[12]. The advantage of this is that it is easier to keep the generated latent variables consistent with the input data distribution, and then obtain an embedding vector that accurately matches the input data distribution. However, the additional convolutional layer increases the number of model parameters and computational complexity, resulting in longer model training and inference time, and if the dimension of the input data is too high, it will cause overfitting problems.

In this algorithm, we choose to generate the latent variable distribution parameters by linearly transforming the node features $h^{(L)}$ of the last layer:

$$\mu = h^{(L)} W_\mu \quad (10)$$

$$log\sigma^2 = h^{(L)} W_\sigma \quad (11)$$

is generation method directly maps the final feature $h^{(L)}$ extracted by FAGCN to the latent space distribution parameters without the need for additional graph convolution operations. Its complexity is concentrated in the convolution part of FAGCN, and the generation of distribution parameters is simplified to a layer of linear transformation, which reduces the number of parameters in the model encoding stage and improves computational efficiency. However, it lacks some accuracy in fitting the data distribution to a certain extent, which makes this generation method more dependent on the quality of the feature $h^{(L)}$ extracted by FAGCN[13].

(3) Reparameterization and Latent Variable Sampling

In a class of generative models such as (VAE and VGAE), we want to sample the latent variable Z from a latent distribution so that it obeys the distribution $q(Z|X)$ output by the encoder, that is:

$$Z \sim N(\mu, \sigma^2) \quad (12)$$

Sampling directly from the potential distribution will result in a non-differentiable training process. The reparameterization technique can solve this problem by treating the entire sampling process as a differentiable function, so that the gradient can be passed through the sampling process and the model can be trained through the standard back-propagation algorithm.

The reparameterization technique represents the random variable Z as a differentiable deterministic function:

ISSN: 1813-4890

$$Z = \mu + \sigma \odot \epsilon \tag{13}$$

Where μ is the mean of the latent distribution obtained from the encoder; σ is the standard deviation of the latent distribution obtained from the encoder, and $\sigma = exp(0.5 \cdot log\sigma^2)$; ϵ is a noise term that follows a standard normal distribution, and $\epsilon \sim N(0,1)$, ⊙represents element-by-element multiplication. Thus, the sampling of the latent variable Z can be re-expressed as a deterministic process:

$$Z = \mu + \sigma \cdot \epsilon \tag{14}$$

2.2.2. Decoder structure design

In this study, we designed a decoder structure different from the traditional model to reconstruct the adjacency matrix and node feature matrix of the input graph from the latent variable Z generated by the encoder[14]. The decoder consists of two parts: reconstructing the adjacency matrix through the inner product decoder, and reconstructing the node feature matrix through the inverted GCN decoder. In the GAE and VGAE models, it can be found from their loss functions that the training goal of the model is only to restore the adjacency matrix of the original graph, and does not consider the restoration of the node feature matrix. On this basis, this study adds node feature matrix loss terms and distribution loss terms.

(1) Inner Product Decoder

The inner product decoder reconstructs the adjacency matrix of the graph by calculating the inner product between the latent variables Z. This method is simple and effective, and can capture the similarity between nodes in the latent space, thereby generating edge probabilities. The inner product decoder operates as follows:

$$\hat{A} = \sigma(ZZ^T) \tag{15}$$

Where $\hat{A}$ is the reconstructed adjacency matrix, $Z \in \mathbb{R}^{N \times d}$ is the latent variable matrix generated by the encoder, and $\sigma(\cdot)$ is the activation function

(2) GCN Decoder

The GCN decoder reconstructs the node feature matrix from the latent variable Z through the reverse graph convolution operation, and uses the structural information of the graph to achieve feature reconstruction by reversely aggregating node features layer by layer. The GCN decoder operates as follows:

$$\hat{X}^{(l-1)} = \sigma\left(\hat{A}D^{-1/2}\sum_{j \in N_i} \alpha_{ij}^{G} \hat{X}^{(l)} W^{(l-1)} D^{-1/2}\right) \tag{16}$$

where $\hat{X}^{(l-1)}$ is the reconstructed feature representation of the $l-1$ layer, α_{ij}^{G} is the adaptive weight coefficient, $\hat{X}^{(l)} = Z$ represents the input of the decoding process starting from the latent variable Z, and $W^{(l-1)}$ is the learnable weight matrix of the decoder at the $l-1$ layer.

(3) Loss function design

The loss function consists of two parts. The first part is the loss function $\mathcal{L}_{VGAE}$ in the VGAE model, whose training goal is based on the adjacency matrix loss and the difference in input and output data distribution. The second part adds the reconstructed node feature loss and the divergence between the probability distribution of the reconstructed node feature matrix and the probability distribution of the original node feature matrix. Therefore, the total loss function is as follows:

$$\mathcal{L}_{VGAE-ISMO} = \lambda_{VGAE}\mathcal{L}_{VGAE} + \lambda_{feat}\frac{1}{N}\sum_{i=1}^{N}\left\|X_i - \hat{X}_i\right\|^2 + \lambda_{feat_{kl}} KL\left[P(\hat{X}|Z) \parallel P(X)\right] \tag{17}$$

ISSN: 1813-4890

Among them, λ_{VGAE}、 λ_{feat}、 and λ_{feat_kl} are the weight coefficients of the losses of each part respectively.

2.2.3. Model training algorithm

In the training process of this model, the number of input parameters is 7, including the input graph G, the node feature matrix X, the learning rate lr, the number of training iterations epochs, the hidden layer configuration hiddendims of the model, the loss function weight λ, and the regularization configuration parameter dropout. The output of the training is the embedding expression results of all nodes in the graph and the reconstruction results of the adjacency matrix. The model training process is shown in Algorithm 1 below:

Algorithm 1: *VGAE_ISMO training process*

Input:

Parameters:$G, X, lr, epochs, hidden\_dims, dropout, \lambda$

Output:

Parameters:Z

1: $A, X \leftarrow parseGraph(G)$
2: $\tilde{A} \leftarrow normalize(A)$
3: $model \leftarrow VGAE\_ISMO(hidden\_dims, dropout, \lambda)$
4: $optimizer \leftarrow Adam(model.parameters(), lr)$
5: $Z \leftarrow Matrix(X.shape[0], hidden\_dims[-1])$
6: $for\ epoch\ in\ range(epochs)\ do$
7: $\quad model \leftarrow model.train()$
8: $\quad optimizer.zero\_grad()$
9: $\quad \hat{A}, \hat{X}, Z \leftarrow model(X, A)$
10: $\quad L_{VGAE-ISMO} \leftarrow \lambda_{VGAE} L_{VGAE} + \lambda_{feat} \frac{1}{N} \sum_{i=1}^{N} \left\| X_i - \hat{X}_i \right\|^2 + \lambda_{feat\_kl} KL\left[P\left(\hat{X}|Z\right) \| P(X)\right]$
11: $\quad L_{VGAE-ISMO}.backward()$
12: $\quad optimizer.step()$
13: $\quad end\ for$

The first line of the algorithm inputs the graph G and parses out the feature matrix X and the adjacency matrix A. The second line normalizes the adjacency matrix A so that it satisfies $\tilde{A} = D^{-1/2} A D^{-1/2}$, where D is the degree matrix of the node. The purpose of normalization is to enhance the stability of the model. The third line instantiates the VGAE-ISMO model. The required input parameters include the hidden layer configuration hidden_dims (determines the structure of the encoder and decoder), the regularization parameter dropout, and the weight λ of the loss function. The fourth line constructs an Adam optimizer to optimize the parameters of the VGAE-ISMO model. The Adam optimizer combines the momentum method and adaptive learning rate adjustment to maintain the stability of learning during the parameter update process. The fifth line of the algorithm initializes the embedding matrix Z, whose size is the number of nodes N multiplied by the hidden variable dimension $hidden\_dims[-1]$. The seventh line enters the training phase of the model. The eighth line sets the gradient of the optimizer to 0 to ensure that each iteration does not accumulate the gradient of the previous time. The ninth line inputs the node feature matrix X and the adjacency matrix A into the VGAE-ISMO model for forward propagation to obtain the reconstructed adjacency matrix A^, the reconstructed feature matrix X^, and the hidden variable Z. The tenth line calculates the total loss $L_{VGAE-ISMO}$ based on the output of the model. The loss function consists of three parts:

ISSN: 1813-4890

The reconstruction loss L_{VGAE} of VGAE measures the similarity between the reconstructed adjacency matrix $\hat{A}$ and the original adjacency matrix A;

The feature matrix reconstruction error $\frac{1}{N}\sum_{i=1}^{N}\left\|X_i - \widehat{X}_i\right\|^2$ measures the reconstruction accuracy of the node feature matrix;

The KL divergence of the feature distribution $KL\left[P\left(\hat{X}|Z\right) \| P(X)\right]$ is used to evaluate the difference between the feature distribution generated by the latent variable and the true distribution.

Line 11 propagates the loss backward through the $backward()$ function and calculates the gradient; Line 12 updates the model parameters through the optimizer's $step()$ function. When the number of training iterations reaches the set value of the hyperparameter epochs, the model stops training and finally returns the node's embedded representation Z.

3. Experiment and result analysis

3.1. Experimental environment and evaluation indicators

The advantages of NSGA-Ⅱ are that it runs efficiently and the resulting solution set is well-distributed, and its main shortcomings are that it is difficult to find isolated points and it is easy to produce a large number of duplicate individuals. In order to overcome the above shortcomings, this paper proposes an improved NSGA-Ⅱ algorithm, which is mainly improved in the following two aspects.

Table 1 Dataset Information

Dataset	Node	Edge	Network	Category
Cora	2708	5069	引文	7
Citeseer	2110	3668	引文	6
Pubmed	19717	44324	引文	3
Acm	3025	13128	引文	3
DBLP	17716	52867	协作者	4

In order to verify the experimental results, this paper selects several mainstream similar algorithms for comparative testing, including: DeepWalk, Node2Vec, two classic unsupervised algorithms based on random walks (Experiment 1) and GAE, VGAE, ARGA, ARVGA and other graph autoencoders (Experiment 2) as basic models for comparison with NLGAE. In order to achieve fairness in the comparison, in the process of comparing with the unsupervised model algorithm, all classification models uniformly use the classic clustering algorithm: K-means algorithm. In the following experiment, three performance indicators, including normalized mutual information (NMI), adjusted Rand index (ARI), and F-score (F1), are used to evaluate the performance of each model.

3.2. Comparative experiment

3.2.1. Experiment 1: Comparison with random walk-based methods

In order to compare the effect of the VGAE-ISMO model on the data set, this paper first selected classic algorithms based on random walks for comparison, namely DeepWalk and Node2Vec. After the embedding vectors were obtained by these three algorithms, the K-means algorithm was uniformly used as the clustering algorithm for community division, and NMI, ARI, and Modularity were used as evaluation indicators to evaluate the community division results of each algorithm. Since both the DeepWalk and Node2Vec algorithms rely only on the structural information of the graph for representation learning and do not use the node feature matrix, in order to ensure the accuracy of the comparative test, the embedded representation vectors

ISSN: 1813-4890

obtained by them are spliced with the node feature matrix as the final representation of the node, and are added to the comparative test.

Table 2 Comparison of algorithms on four indicators: NMI, ARI, and F-score

模型	评价指标	Cora	Citeseer	Pubmed	ACM	DBLP
DeepWalk+K-means	NMI	0.5979	0.3788	0.3674	0.5124	0.3552
	ARI	0.5496	0.3729	0.3773	0.4772	0.3679
	F1	0.6238	0.4897	0.4281	0.5673	0.4363
DeepWalk+X+K-means	NMI	0.4173	0.4832	0.4217	0.5481	0.4163
	ARI	0.4865	0.3944	0.4186	0.5237	0.4373
	F1	0.5655	0.5363	0.5463	0.5452	0.5167
Node2Vec+K-means	NMI	0.4065	0.3768	0.3846	0.5036	0.3624
	ARI	0.4572	0.2677	0.3574	0.4213	0.3287
	F1	0.6742	0.4638	0.4773	0.5863	0.4038
Node2Vec+X+K-means	NMI	0.3728	0.4423	0.4176	0.5732	0.4236
	ARI	0.4782	0.3479	0.3998	0.4612	0.3937
	F1	0.4689	0.5127	0.4273	0.5375	0.5084
VGAE-ISMO+X+K-means	NMI	0.6612	0.6148	0.5661	0.5817	0.5699
	ARI	0.5973	0.5780	0.5173	0.5324	0.5824
	F1	0.6561	0.6831	0.6038	0.6129	0.6235

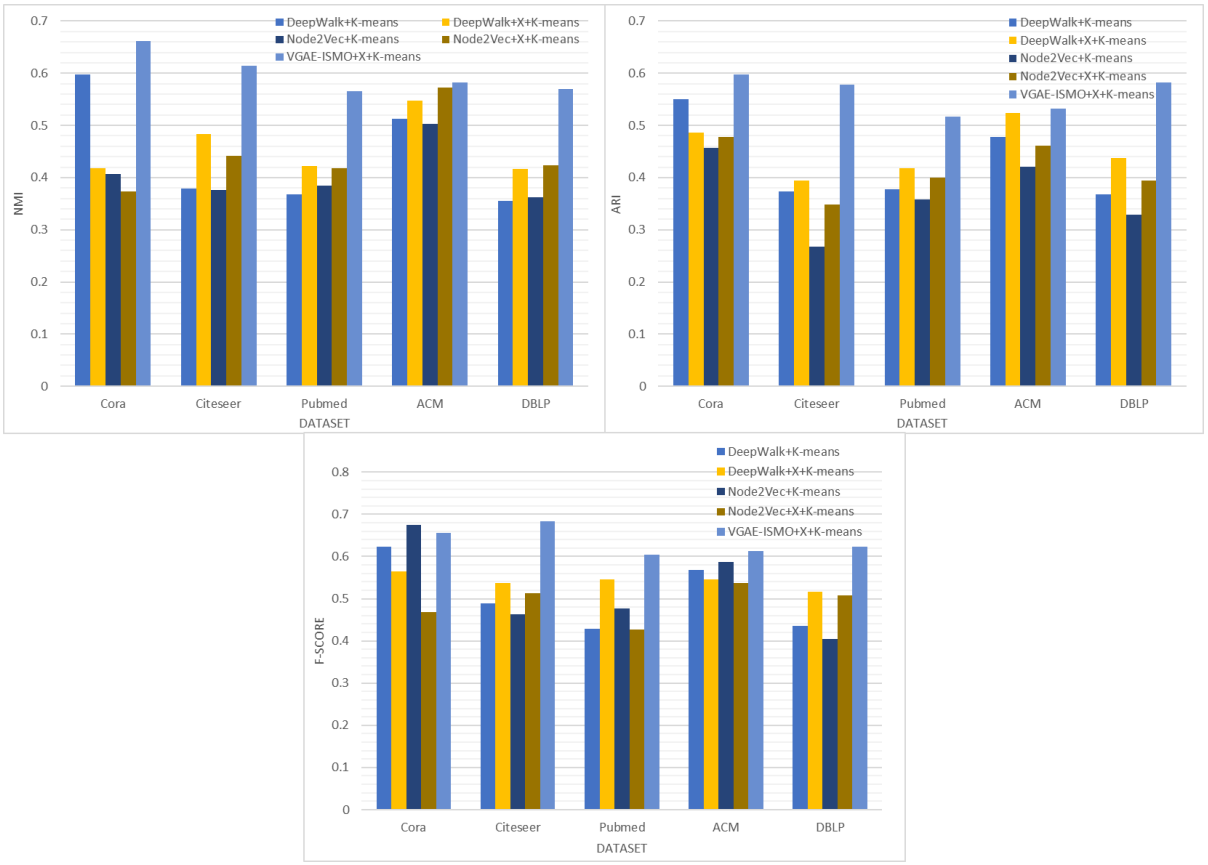

Fig. 1 Comparison of algorithms on four indicators: NMI, ARI, and F-score

As shown in Table 2, VGAE-ISMO and similar unsupervised algorithms use NMI, ARI, and F1 evaluation indicators to measure the community division effect on five data sets, including Cora, Citeseer, Pubmed, DBLP, and ACM. The bold values are the best values. Among them, DeepWalk+K-means means that the embedding vector is obtained by using DeepWalk and the clusterer uses K-means. DeepWalk+X+K-means means that the embedding representation

vector is obtained and then concatenated with the node feature matrix, and the community division is obtained by clustering using the K-means algorithm. The same is true for Node2Vec+K-means, Node2Vec+X+K-means and VGAE-ISMO +X+K-means.

As shown in Figure 2, DeepWalk+K-means, DeepWalk+X+K-means, Node2Vec+K-means, Node2Vec+X+K-means, and VGAE-ISMO +X+K-means are compared in terms of NMI, ARI, and F1 on each data set. We can draw the following conclusions:

1. DeepWalk+K-means obtains the best values for NMI and ARI on the Cora data set, and even obtains better numerical results than DeepWalk+X+K-means in the absence of node feature matrix input. This shows that for the Cora data set, the topological graph structure information contains more information that is helpful for community division than its node attribute information, and the latter has poor performance due to the integration of the node feature matrix. In this case, the performance can be adjusted by adjusting the hyperparameter λ in the loss function to increase the proportion of graph structure information in the optimization target; in contrast, Node2Vec+K-means obtains the best value on the F1 indicator because Node2Vec A more flexible random walk strategy can better capture the local and global structure of the graph and generate high-quality embedding representations. These embeddings enable K-means clustering to more accurately divide communities, improve Precision and Recall, and thus make the F1 index higher.

2. After adding the node feature matrix, the algorithms on the Citeseer, Pubmed, ACM, and DBLP datasets all showed numerical improvements, which shows that the node feature matrix of these datasets can enrich clustering information, and VGAE-ISMO has the best performance on these datasets. Taking the NMI index as an example, especially on the DBLP dataset, VGAE-ISMO+X+K-means has improved its performance by about 30% compared with the suboptimal Node2Vec+K-means. The effect is also significant on the Citeseer dataset. VGAE-ISMO+X+K-means has improved its performance by about 16% compared with DeepWalk+X+K-means. In addition, although the performance on the Cora dataset is not the best, all three evaluation indicators are suboptimal.

3.2.2. Experiment 2: Comparison with Graph Autoencoder Method

In this section, VGAE-ISMO is compared with mainstream graph autoencoder models such as GAE, VGAE, ARGA, and ARVGA. In order to ensure the uniformity of the comparison, K-means is used as the clustering algorithm. In terms of model parameters, the training iterations of GAE and VGAE, which are the basic models, are set to 200, the number of encoder layers is set to 2 layers, the hidden layer 1 is set to 32, the hidden layer 2 is set to 32, and the learning rate is 0.01; the parameters of ARGA and ARVGA are all set to the default parameters.

Table 3 Comparison of algorithms on four indicators: NMI, ARI, and F-score

模型	评价指标	Cora	Citeseer	Pubmed	ACM	DBLP
GAE	NMI	0.5612	0.4747	0.2532	0.3722	0.2155
	ARI	0.4983	0.4430	0.2484	0.3485	0.1974
	F1	0.6527	0.5631	0.3251	0.4623	0.2475
VGAE	NMI	0.6102	0.5472	0.4217	0.4721	0.3247
	ARI	0.5826	0.5682	0.4186	0.4273	0.3028
	F1	0.5489	0.5127	0.3963	0.4326	0.3367
ARGA	NMI	0.5173	0.5438	0.4638	0.4836	0.4083
	ARI	0.4928	0.4347	0.3675	0.4046	0.3359
	F1	0.5562	0.4876	0.3365	0.4363	0.3574
ARVGA	NMI	0.6204	0.5825	0.4572	0.4632	0.4363
	ARI	0.5638	0.5066	0.4251	0.4346	0.3675
	F1	0.6581	0.5527	0.4648	0.5065	0.4472

ISSN: 1813-4890

VGAE-ISMO	NMI	0.6783	0.6148	0.5661	0.5817	0.5699
	ARI	0.6382	0.5780	0.5373	0.5324	0.5824
	F1	0.6942	0.6831	0.6038	0.6129	0.6235

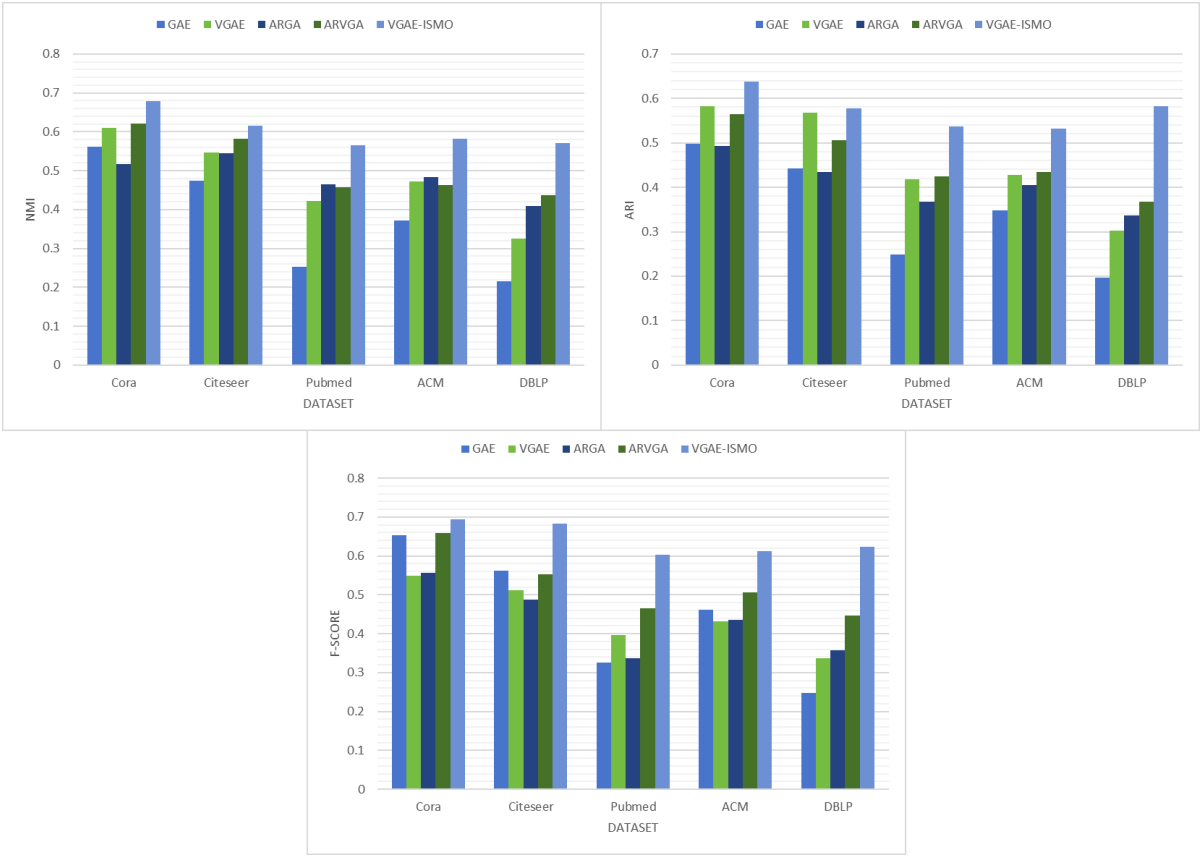

Fig. 2 Comparison of algorithms on four indicators: NMI, ARI, and F-score

As shown in Table 3, VGAE-ISMO is compared with GAE, VGAE, ARGA, and ARVGA algorithms on five datasets: Cora, Citeseer, Pubmed, ACM, and DBLP, using the three evaluation indicators NMI, ARI, and F1. We can draw the following conclusions:

1. Compared with GAE, VGAE, ARGA, and ARVGA algorithms, VGAE-ISMO has achieved the best values in the three evaluation indicators NMI, ARI, and F1 on five datasets: Cora, Citeseer, Pubmed, ACM, and DBLP, and has a certain performance advantage.

2. On the Cora dataset, VGAE-ISMO has a slight improvement over ARVGA. Taking the NMI index as an example, VGAE-ISMO has a 0.0579 improvement over ARVGA. However, compared with the basic models GAE and VGAE, the performance has been greatly improved. Taking the DBLP dataset as an example, VGAE-ISMO has improved 0.3544, 0.3850, and 0.3760 in NMI, ARI, and F1 indicators compared with GAE, and 0.2452, 0.2796, and 0.2868 respectively compared with the VGAE model; Taking the Pubmed dataset as an example, VGAE-ISMO has improved 0.3129, 0.2886, and 0.2787 in NMI, ARI, and F1 indicators compared with GAE, and VGAE-ISMO has improved 0.1444, 0.1187, and 0.2075 respectively compared with VGAE.

Through the content of Chapter 3, we can know the principles of GAE and VGAE models. The existing problems are: lack of restoration of node feature matrix, single decoder expression ability, and the problem of fixed weight coefficients between nodes. We designed the VGAE-ISMO model to address these problems. First, the FAGCN model was used to replace the GCN model in the encoding stage, and the problem of fixed weight coefficients between nodes was solved with the help of the GAT idea; secondly, the original decoding structure was improved, the reconstructed node feature matrix was supplemented, and relevant optimization objectives were added to the loss function, so that the model not only considers the graph structure

information when generating the embedded vector, but also considers the node attribute information, so as to obtain a higher quality embedded representation vector. These improvements make our VGAE-ISMO model have a leading performance advantage on five datasets: ora, Citeseer, Pubmed, ACM, and DBLP.

4. Conclusion

Community discovery is an important research topic in the field of complex networks, and its goal is to divide complex networks into multiple sub-networks. More and more deep learning algorithms are applied to this topic, and graph representation learning is one of them. GAE and VGAE models are classic graph representation learning algorithms. In this study, these two models are studied in depth, and on the basis of existing research, the problems of GAE and VGAE models are analyzed in detail. The existing problems are solved from three aspects: improving the encoding structure, decoding structure, and loss function. A graph autoencoder model VGAE-ISMO based on improved encoding and decoding structure and multi-optimization is proposed. First, in terms of improving the encoding structure, we use the FAGCN model instead of the GCN encoding to solve the problem of solidified weight coefficients between nodes. At the same time, FAGCN can divide the information in the graph data into low-frequency information and high-frequency information, and selectively and comprehensively obtain graph structure information; secondly, in terms of decoding structure, we added the reconstruction of the node feature matrix to make up for the vacancy of the original model in this regard and solve the problem of single decoder expression ability; finally, the improvement of the loss function enables the VGAE-ISMO model to simultaneously integrate node feature loss, graph structure loss, and data distribution loss for training and optimization. The comparative experimental results show that VGAE-ISMO, as an unsupervised model, can learn high-quality node embedding vectors. It is not only better than classic unsupervised algorithms such as DeepWalk and Node2Vec in community discovery tasks, but also has certain performance advantages in similar graph autoencoder algorithms.

References

[1] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.

[2] Fortunato S. Community detection in graphs[J]. Physics reports, 2010, 486(3-5): 75-174.

[3] Chen P, Redner S. Community structure of the physical review citation network[J]. Journal of Informetrics, 2010, 4(3): 278-290.

[4] Chen J, Yuan B. Detecting functional modules in the yeast protein–protein interaction network[J]. Bioinformatics, 2006, 22(18): 2283-2290.

[5] X. Su et al., "A Comprehensive Survey on Community Detection With Deep Learning," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2021.3137396.

[6] P. W. Holland, K. B. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” Social Networks, vol. 5, no. 2, pp. 109–137, 1983.

[7] Liu C, Zheng X, Zhao P. Overlapping Community Discovery Algorithm Based on Seed Node Importance Selection[C]//Proceedings of the 2023 8th International Conference on Multimedia and Image Processing. 2023: 113-117.

[8] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” in Computer and Information Sciences, 2005, pp. 284–293.

[9] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[10] Kipf T N, Welling M. Variational graph auto-encoders[J]. arXiv preprint arXiv:1611.07308, 2016.

ISSN: 1813-4890

[11] Choong J J, Liu X, Murata T. Learning community structure with variational autoencoder[C]//2018 IEEE international conference on data mining (ICDM). IEEE, 2018: 69-78.

[12] Zhao P, Zheng X, Liu C. Probability-Corrected Overlapping Community Detection Algorithm Based on GCN[C]//2023 Asia Symposium on Image Processing (ASIP). IEEE, 2023: 24-31.

[13] Fei R, Wan Y, Hu B, et al. A novel network core structure extraction algorithm utilized variational autoencoder for community detection[J]. Expert Systems with Applications, 2023, 222: 119775.

[14] Qiu C, Huang Z, Xu W, et al. VGAER: graph neural network reconstruction based community detection[J]. arXiv preprint arXiv:2201.04066, 2022.