doi:10.3772/j.issn.1006-6748.2022.01.012

Deep convolutional adversarial graph autoencoder using positive pointwise mutual information for graph embedding①

MA Xiuhui (马秀慧)[*,**], WANG Rong[***], CHEN Shudong②[*,**], DU Rong[*,**], ZHU Danyang[*,**], ZHAO Hua[***]

([*] University of Chinese Academy of Sciences, Beijing 100049, P. R. China)

([**] Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, P. R. China)

([***] Key Laboratory of Space Object Measurement Department, Beijing Institute of Tracking and Telecommunications Technology, Beijing 100094, P. R. China)

Abstract

Graph embedding aims to map the high-dimensional nodes to a low-dimensional space and learns the graph relationship from its latent representations. Most existing graph embedding methods focus on the topological structure of graph data, but ignore the semantic information of graph data, which results in the unsatisfied performance in practical applications. To overcome the problem, this paper proposes a novel deep convolutional adversarial graph autoencoder (GAE) model. To embed the semantic information between nodes in the graph data, the random walk strategy is first used to construct the positive pointwise mutual information (PPMI) matrix, then, graph convolutional network (GCN) is employed to encode the PPMI matrix and node content into the latent representation. Finally, the learned latent representation is used to reconstruct the topological structure of the graph data by decoder. Furthermore, the deep convolutional adversarial training algorithm is introduced to make the learned latent representation conform to the prior distribution better. The state-of-the-art experimental results on the graph data validate the effectiveness of the proposed model in the link prediction, node clustering and graph visualization tasks for three standard datasets, Cora, Citeseer and Pubmed.

Key words: graph autoencoder (GAE), positive pointwise mutual information (PPMI), deep convolutional generative adversarial network (DCGAN), graph convolutional network (GCN), semantic information

0 Introduction

Graph data has been widely used in various areas, including social media, e-commence, citation networks and protein-protein interaction networks. Analyzing the graph data has become a hot topic of current researches, such as link prediction[1], node classification[2], node clustering[3] and graph visualization[4]. However, the irregularity and complexity of graph data lead to the high computational complexity and low parallelism, which brings great challenges to existing graph data researches.

Graph embedding maps the complex graph data into the latent and low-dimensional feature representations, meanwhile it captures the network topological structure, vertex content, and other information of graph data, which facilitates the processing and analysis of graph data. With the graph embedding, traditional pattern recognition models (e.g., linear support vector machine (SVM)) can easily achieve graph analysis tasks (e.g., classification tasks). Motivated by this advantage, many improvement studies have been proposed[5-6].

In recent years, graph autoencoder (GAE)[7] combined with graph convolutional network (GCN)[2] has received more and more attention. Graph autoencoder employs GCN[2] to encode the node structure information and node feature information at the same time. Since the distribution of latent representations learned by simple GAE[7] has no restriction, which results in the poor embeddings. The variational graph autoencoder (VGAE)[7] applies Gaussian prior to learn the distribution of the latent representation, which greatly improves the effectiveness of the latent repre-

① Supported by the Strategy Priority Research Program of Chinese Academy of Sciences (No. XDC02070600).

② To whom correspondence should be addressed. E-mail: chenshudong@ime.ac.cn.

Received on Jan. 7, 2021

sentation. The adversarially regularized graph autoencoder (ARGA)[8] uses the generative adversarial network (GAN)[9] to enforce the latent code to approximate the prior distribution. Although these graph autoencoder models make full use of the structural information of graph data, they ignore the semantic information between nodes.

In this paper, a novel graph autoencoder model is proposed to capture the semantic information between nodes. The proposed model firstly constructs the postive pointwise mutual information (PPMI) matrix[10] through the random walk strategy[11], which learns the semantic information between nodes of the graph data. Then GCN[2] is employed to encode the PPMI matrix and node content of the graph into the latent representation. Finally, the decoder is used to reconstruct the topological structure of the graph data based on the learned latent representation. To further enhance the robustness of graph representation, this work introduces the deep convolutional adversarial training scheme to regularize the latent code. The purpose of the deep convolutional adversarial training module is to distinguish whether the latent code comes from the real prior distribution or from the graph encoder. In the unified framework, graph embedding learning and deep convolutional adversarial regularization are jointly optimized to benefit each other and eventually get better graph embedding. The experimental results on the benchmark datasets prove that the proposed algorithm shows the good performance on link prediction, node clustering and graph visualization tasks. The contribution of this paper can be summarized as follows.

A novel deep convolutional adversarial graph autoencoder model using positive pointwise mutual information is proposed.

The PPMI matrix and node content of the graph data are encoded into a latent and low-dimensional representation, which better captures the semantic information of graph data.

Deep convolutional adversarial regularized framework is used to further match the learned latent code to the prior distribution.

The network structure of the existing graph autoencoder is changed. The adjacency matrix is replaced by the PPMI matrix as the input of the autoencoder to reconstruct the graph structure.

1 Related work

DeepWalk[4] extends language modeling techniques from word sequences to paths in the graph, which uses the random walk strategy to learn the graph embedding. In this work, they treat the nodes obtained from the random walk strategy as context nodes. Since then, a number of probabilistic models have been proposed including LIKE[12], node2vec[13] and so on. Ref. [14] further extended DeepWalk [4] to encode the multi-scale node relationships in the graph. These methods mentioned above embed the graph data base on the random walk strategy, and they assume that the nodes are similar if they are close to each other in simulated random walks over the graph. However, these methods are only suitable to shallow models, which cannot capture the complex graph structure.

In recent years, deep learning networks are widely used for graph embedding, which can greatly explore the nonlinear graph structures. SDNEI[15] used stacked autoencoder to jointly preserve the first and second-order proximities of nodes in the graph. DNGR[16] exploited the stacked denoising autoencoder to reconstruct the PPMI matrix. But these methods only consider the topological structure information of the graph and fail to consider the content information of the nodes. GCN[2] embedded both the topological structure information of the graph and the feature information of the nodes, which greatly improves the performance of the graph networks on various graph tasks. Ref. [7] proposed GAE which uses GCN[2] as the encoder to reconstruct the adjacency matrix of the graph. To further enforce the latent representation to approximate the prior distribution, Ref. [8] proposed ARGA which employs the training scheme of GAN[9]. However, ARGA[8] ignores the semantic information. Inspired by this, the PPMI matrix and node content information of the graph are embedded in the encoder to capture the semantic information between nodes. Meanwhile, deep convolutional generative adversarial network (DCGAN)[17] applies convolutional layers to generative adversarial network, which greatly improves the stability of GAN[9] training and the quality of generated results. Motivated by this, a deep convolutional adversarial training scheme is incorporated to learn a more robust graph representation.

2 Problem definition and framework

2.1 Problem definition

This paper firstly defines some commonly used symbols to represent the graph structure and then presents the unsupervised representation learning problem of the nodes on the graph G.

$G = \{V, E, X\}$ is used to represent an undirected graph, where $V = \{v_1, \cdots, v_N\}$ means the set of nodes and N is the number of nodes. E denotes the set

of edges, where $e_{ij} \in E$ represents the edge between the node v_i and the node v_j. In addition, the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is used to represent the topological structure of the graph G, where $A_{ij} = 1$ if $e_{ij} \in E$, otherwise $A_{ij} = 0$. And $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ means the feature matrix of the nodes, where $x_i \in \mathbb{R}^M$ corresponds to the ith row of matrix $\boldsymbol{X}$, denoting the feature of node v_i, and M is the dimension of the feature.

Given a graph G, the goal is to learn the low-dimensional representation $\boldsymbol{Z} \in \mathbb{R}^{N \times D}$ of the nodes in the graph, where $z_i \in \mathbb{R}^D$ represents the D-dimensional representation of the node v_i, and $D \ll N$. It is hoped that $\boldsymbol{Z}$ as an embedded matrix can capture not only the content information and topology information but also the semantic information of nodes.

2.2 Overall framework

The framework (DCAG-AE) consists of three main parts: constructing PPMI matrix by the random walk strategy, graph autoencoder, and the deep convolutional adversarial network. Fig. 1 briefly displays the workflow of DCAG-AE.

Constructing PPMI matrix On the basis of the topological structure $\boldsymbol{A}$ of the graph, this work constructs the PPMI matrix with the random walk strategy.

Graph convolutional autoencoder The encoder encodes the PPMI matrix and the node content $\boldsymbol{X}$ into a latent representation $\boldsymbol{Z}$, and then the decoder reconstructs the topological structure $\boldsymbol{A}$ on the basis of $\boldsymbol{Z}$.

Deep convolutional adversarial regularization The deep convolutional adversarial network trains a discriminator to discriminate whether $z_i \in \boldsymbol{Z}$ comes from the encoder or the prior distribution, which further enhances the robustness of the encoder.

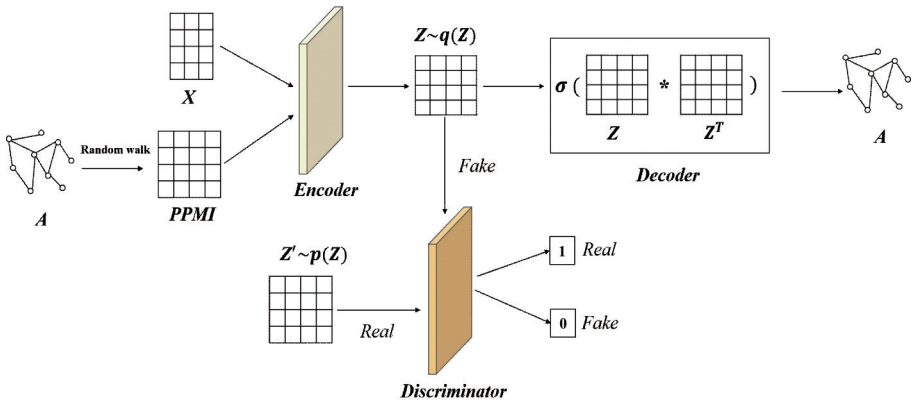


Fig. 1 The brief architecture of the deep convolutional adversarial graph autoencoder using positive pointwise mutual information for graph embedding (DCAG-AE). Top half of the network corresponds to the graph convolutional autoencoder which exploits the PPMI matrix and node content $\boldsymbol{X}$ to reconstruct the topological structure $\boldsymbol{A}$. Bottom half is a deep convolutional adversarial network, which discriminates whether a sample comes from the encoder or the prior distribution

3 Proposed model

3.1 Constructing PPMI matrix

Motivated by DGCN[18], which uses the random walk strategy to construct the PPMI matrix for the graph nodes classification task. Specifically, with the topological structure $\boldsymbol{A}$, this paper firstly exploits the random walk strategy to build a co-occurrence matrix $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ of nodes on the entire graph. Then, according to the matrix $\boldsymbol{C}$, this paper constructs the PPMI matrix and explains how to capture the semantic information between nodes of the graph G.

Constructing nodes co-occurrence matrix $\boldsymbol{C}$. DeepWalk[4] proves that in the connected graph, the degree of nodes and the frequency of nodes in the short random walk conform to the same distribution. So, in this work, the short random walk strategy is used to construct the nodes co-occurrence matrix $\boldsymbol{C}$.

Algorithm 1 shows the construction process of nodes co-occurrence matrix $\boldsymbol{C}$. First, randomly select a node v_1 from the graph as the current node, then a node v_2 is randomly selected from the neighbors of the current node and is made as the new current node. Repeating this random sampling process to obtain a random walk sequence T until the number of nodes in the sequence reaches the walk length s. Then sample the node pairs (v_a, v_b) from the sequence T uniformly, and add 1.0 to the value of $C_{a,b}$ and $C_{b,a}$ in the matrix $\boldsymbol{C}$. Finally repeat the above process γ times which is called the total walk. Note that the probability of reaching any of its neighbors from the current node in the algorithm is equal.

Algorithm 1 Construct nodes co-occurrence matrix $\boldsymbol{C}$

Input:

graph $G = \{V, E, X\}$ with adjacency matrix $\boldsymbol{A}$;

window size w;

walks per node γ;

walk length s;

Output: the nodes co-occurrence matrix $\boldsymbol{C} \in \mathbb{R}^{N \times N}$;

```
Initialization: initialize the matrix C ∈ R^{N×N} to 0;
for i = 0 to γ do
    O = Shuffle (V);
    for each v_i ∈ O do
        T = RandomWalk (A, v_i, s);
        Uniformly sample all node pairs (v_a, v_b) ∈ T
        within w;
        for each pair (v_a, v_b) do
            C_{a,b} += 1.0;
            C_{b,a} += 1.0;
        end for
    end for
end for
return nodes co-occurrence matrix C.
```

Constructing PPMI matrix. Based on the nodes co-occurrence matrix $\boldsymbol{C}$, the PPMI matrix is calculated as

$$p(i) = (\Sigma_j C_{i,j})/(\Sigma_{ij} C_{i,j}) \tag{1}$$

$$p(j) = (\Sigma_i C_{i,j})/(\Sigma_{ij} C_{i,j}) \tag{2}$$

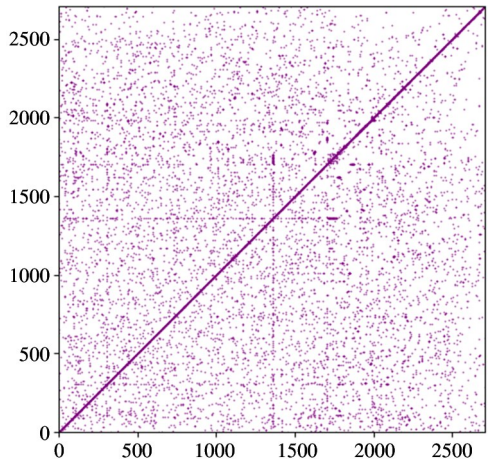
$$p(i,j) = C_{i,j}/(\Sigma_{ij} C_{i,j}) \tag{3}$$

$$PPMI(i,j) = \max\{\log_2(p(i,j)/(p(i)p(j))), 0\} \tag{4}$$

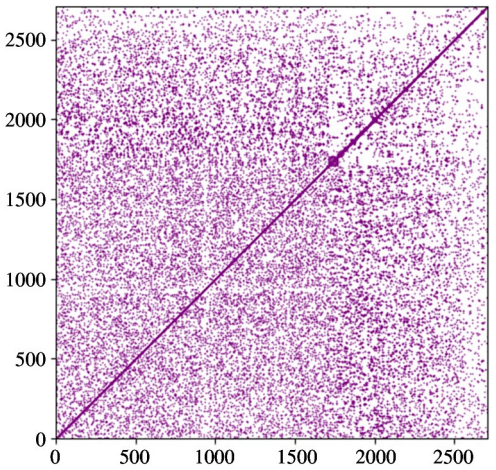
where $p(i)$ is the probability that node v_i appears, $p(j)$ is the probability that context c_j appears, $p(i,j)$ is the probability that node v_i and context c_j appear at the same time.

It can be seen from the calculation formula of the PPMI matrix that when the node v_i and the context c_j are independent, $p(i,j) = p(i)p(j)$, thus $PPMI(i,j) = 0$. When the node v_i and the context c_j are related, i.e., $p(i,j) > p(i)p(j)$, that is $PPMI(i,j) > 0$. Furthermore, the higher the correlation between the node v_i and the context c_j, the greater the value of $PPMI(i,j)$. Therefore, PPMI matrix can capture the semantic information between the node and its context nodes, which is not captured by the adjacent matrix $\boldsymbol{A}$. Additionally, when calculating the nodes co-occurrence matrix $\boldsymbol{C}$ using the random walk strategy, the PPMI matrix not only captures the information of the first-order neighbor nodes, but also is associated with higher-order neighbor nodes. That means the PPMI matrix can mine more potential relationships between the nodes.

Fig. 2 shows the visualization of the sparsity pattern of the normalized adjacency matrix and PPMI matrix of the Cora[19] dataset. From Fig. 2, it can be seen that the PPMI matrix is denser than the adjacency matrix, i.e., the PPMI matrix embeds more latent relations between nodes.



(a) The visualization of the sparsity pattern of the normalized adjacency matrix



(b) The visualization of the sparsity pattern of the normalized PPMI matrix

Fig. 2 The visualizations of the sparsity pattern of the normalized adjacency matrix and PPMI matrix of the Cora dataset

3.2 Graph convolutional autoencoder

The graph autoencoder is usually composed of an encoder and a decoder. In the proposed method, the encoder consists of a 2-layer GCN[2] which fuses the PPMI matrix and the node content $\boldsymbol{X}$ into a latent and low-dimensional representation, and the decoder uses above latent representation to reconstruct the graph structure $\boldsymbol{A}$.

3.2.1 Graph convolutional network

GCN[2] encodes the adjacency matrix $\boldsymbol{A}$ and node content $\boldsymbol{X}$ into the low-dimensional representation. Given a graph $G = \{V, E, X\}$ with its adjacency matrix $\boldsymbol{A}$ and the spectral convolutional function $f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}|$

$\boldsymbol{W}^{(l)}$), the output $\boldsymbol{Z}^{(l+1)}$ after convolution is

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A} \mid \boldsymbol{W}^{(l)}). \tag{5}$$

where $\boldsymbol{Z}^{(l)}$ is the output of lth hidden layer of the convolutional network, $\boldsymbol{W}^{(l)}$ is the filter parameter matrix of lth layer. And the spectral convolutional function is defined as

$$f(\boldsymbol{Z}^{(l)}, \boldsymbol{A} \mid \boldsymbol{W}^{(l)}) = \sigma(\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)}) \tag{6}$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, $\boldsymbol{I}$ is the identity matrix of $\boldsymbol{A}$, $\widetilde{\boldsymbol{D}}$ is the diagonal node degree matrix of $\widetilde{\boldsymbol{A}}$, $\widetilde{\boldsymbol{D}}_{ii} = \Sigma_j \widetilde{\boldsymbol{A}}_{ij}$, and $\sigma(\cdot)$ is an activation function such as ReLU or Sigmoid.

3.2.2 Graph convolutional autoencoder using PPMI matrix

In this work, a 2-layer GCN[2] is used as the encoder model $g(\boldsymbol{X}, \boldsymbol{P})$. Instead of encoding node feature $\boldsymbol{X}$ and the adjacency matrix $\boldsymbol{A}$, the proposed model embeds $\boldsymbol{X}$ and PPMI matrix into a low-dimensional space to better capture the semantic information of the graph data. The convolutional operation is defined as $f(\boldsymbol{Z}^{(l)}, \boldsymbol{P} \mid \boldsymbol{W}^{(l)})$:

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{P} \mid \boldsymbol{W}^{(l)}) = \sigma(\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{P}}\widetilde{\boldsymbol{D}}^{-1/2}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)}) \tag{7}$$

where $\boldsymbol{P}$ is the PPMI matrix and $\widetilde{\boldsymbol{D}}$ is the diagonal node degree matrix of $\widetilde{\boldsymbol{P}}$, i. e., $\widetilde{\boldsymbol{D}}_{ii} = \Sigma_j \widetilde{\boldsymbol{P}}_{ij}$.

The graph encoder $q(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{P})$ is

$$\boldsymbol{Z}^{(1)} = f_{\mathrm{ReLU}}(\boldsymbol{Z}^{(0)}, \boldsymbol{P} \mid \boldsymbol{W}^{(0)}) \tag{8}$$

$$\boldsymbol{Z}^{(2)} = f_{\mathrm{linear}}(\boldsymbol{Z}^{(1)}, \boldsymbol{P} \mid \boldsymbol{W}^{(1)}) \tag{9}$$

where $\boldsymbol{Z}^{(0)} = \boldsymbol{X}$. For the activation function, this paper employs the ReLU function in the first layer and the linear function in the second layer. The output of the graph encoder (i. e., the latent representation in the low-dimensional space) is $\boldsymbol{Z} = \boldsymbol{Z}^{(2)}$.

Based on the embedding $\boldsymbol{Z}$, this paper reconstructs the graph structure $\hat{\boldsymbol{A}}$ by decoder $p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z})$:

$$\hat{\boldsymbol{A}} = \sigma(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}) \tag{10}$$

where the activation function $\sigma(\cdot)$ is the linear function.

To optimize this network, this work minimizes the reconstruction loss of the graph structure by

$$L_R = E_{q(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{P})}[\log(p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z}))] \tag{11}$$

3.3 Deep convolutional adversarial regularization

To improve the matching of the latent representation $\boldsymbol{Z}$ and the prior distribution, the proposed model uses a deep convolutional adversarial training network, which is composed of a 2-layer GCN[2] and a fully connected layer. The adversarial model $D(\boldsymbol{Z})$ is formulated as

$$\boldsymbol{H}^{(1)} = f_{\mathrm{ReLU}}(\boldsymbol{H}^{(0)}, \boldsymbol{P} \mid \boldsymbol{W}^{(0)}) \tag{12}$$

$$\boldsymbol{H}^{(2)} = f_{\mathrm{ReLU}}(\boldsymbol{H}^{(1)}, \boldsymbol{P} \mid \boldsymbol{W}^{(1)}) \tag{13}$$

$$\boldsymbol{H}^{(3)} = \sigma(\boldsymbol{H}^{(2)}\boldsymbol{W}^{(2)} + b) \tag{14}$$

where $\boldsymbol{H}^{(0)} = \boldsymbol{Z}$ is the input of the adversarial network, $\boldsymbol{H}^{(l)}$ denotes the output of lth hidden layer of the deep convolutional adversarial network, $\boldsymbol{H}^{(3)}$ represents the binary output of the adversarial network, σ is the Sigmoid function, and b is the bias of the fully connected layer.

The adversarial model acts as a discriminator to distinguish whether the latent code $\boldsymbol{Z}$ comes from the prior distribution $p(\boldsymbol{Z})$ ($p(\boldsymbol{Z})$ conforms to the Gaussian distribution) or from the graph encoder model $g(\boldsymbol{X}, \boldsymbol{P})$. The loss function of the discriminator is a cross-entropy loss related to the binary classifier, and the discriminator is optimized by minimizing the loss function.

$$L_D = -\frac{1}{2}E_{\boldsymbol{Z}\sim p(\boldsymbol{Z})}\log D(\boldsymbol{Z}) - \frac{1}{2}E_{\boldsymbol{X}}\log[1 - D(g(\boldsymbol{X}, \boldsymbol{P}))]$$

The graph encoder $g(\boldsymbol{X}, \boldsymbol{P})$ is regarded as a generator in this paper, and the joint training of the generator $g(\boldsymbol{X}, \boldsymbol{P})$ and discriminator $D(\boldsymbol{Z})$ is as follows.

$$\max_g \min_D \{\frac{1}{2}E_{\boldsymbol{Z}\sim p(\boldsymbol{Z})}\log D(\boldsymbol{Z}) + \frac{1}{2}E_{\boldsymbol{X}}\log[1 - D(g(\boldsymbol{X}, \boldsymbol{P}))]\}$$

Algorithm 2 displays the overall flow of the framework.

Algorithm 2 DCAG-AE

Input:

graph $G = \{V, E, X\}$ with adjacency matrix $\boldsymbol{A}$;

the number of epochs L;

Output:

the latent representation $\boldsymbol{Z} \in \mathbb{R}^{N \times D}$;

the reconstructed graph structure $\hat{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$;

1. Calculate the co-occurrence matrix $\boldsymbol{C}$ by Algorithm 1;
2. Calculate PPMI matrix $\boldsymbol{P}$ by Eq. (4);
3. for $i = 0$ to L do
4. Encoder $g(\boldsymbol{X}, \boldsymbol{P}) \rightarrow \boldsymbol{Z}$ according to Eq. (9);
5. Sample m entities $\{a^{(1)}, \cdots, a^{(m)}\}$ from the prior distribution $p(\boldsymbol{Z})$;
6. Sample m entities $\{z^{(1)}, \cdots, z^{(m)}\}$ from the latent matrix $\boldsymbol{Z}$;
7. Calculate the output of the discriminator $D(\boldsymbol{Z})$ by Eq. (14);
8. Update the discriminator by $\nabla \frac{1}{m}\sum_{j=1}^{m}[\log D(a^{(j)}) + \log(1 - D(z^{(j)}))]$
9. Decoder $\sigma(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}) \rightarrow \hat{\boldsymbol{A}}$ according to Eq. (10);

10. Update encoder and decoder using Eq. (11);
11. end for
12. return the latent representation $\boldsymbol{Z}$ and the reconstructed graph structure $\hat{\boldsymbol{A}}$.

Given a graph G and its adjacency matrix $\boldsymbol{A}$, this work firstly calculates the nodes co-occurrence matrix $\boldsymbol{C}$ following Algorithm 1 and then calculates the PPMI matrix according to the learned $\boldsymbol{C}$. Then, the encoder model $g(\boldsymbol{X}, \boldsymbol{P})$ encodes the PPMI matrix $\boldsymbol{P}$ and node content $\boldsymbol{X}$ into the latent representation $\boldsymbol{Z}$. After that, the same number of samples are sampled from the generated representation $\boldsymbol{Z}$ and the real data distribution $p(\boldsymbol{Z})$ to train the discriminator with the cross-entropy loss. Then the decoder reconstructs the graph structure $\hat{\boldsymbol{A}}$ based on the latent representation $\boldsymbol{Z}$. Finally, the encoder and decoder are updated by Eq. (11).

4 Experiments

To verify the effectiveness of the proposed model, three unsupervised analysis tasks are performed on the graph data: link prediction, node clustering, and graph visualization.

4.1 Dataset

This paper conducts experiments on 3 benchmark datasets. The details of these datasets are shown in Table 1. Cora, Citeseer and Pubmed[19] are all composed of citation networks, whose nodes represent the documents and edges represent the citation relations. All datasets are divided into training set, validation set and testing set. Among them, the validation set accounts for 5% to optimize the hyperparameters, the testing set accounts for 10% to verify the performance of the model, and the rest is used for training.

Table 1 The statistics of different datasets

Dataset	Nodes	Edges	Classes
Cora	2708	5429	7
Citeseer	3327	4732	6
Pubmed	19717	44338	3

4.2 Link prediction

This paper uses the reconstructed graph structure obtained from Eq. (10) to predict the edges and non-edges between the nodes in the testing set.

4.2.1 Baseline

Several classic link prediction methods are selected as the compared methods.

DeepWalk[4] is a graph embedding method, which encodes social relations into a continuous vector space by training Skipgram model[20] using short random walk strategy.

Spectral Clustering[21] learns the social embedding by generating a representation from the normalized graph Laplacian.

GAE[7] uses the graph convolutional network as its encoder and leverages both topological and content information to reconstruct the graph structure.

VGAE[7] is the variational version of GAE which learns the distribution of the latent representation.

ARGA[8] is an adversarially regularized autoencoder which uses the graph autoencoder to learn the low dimensional embedding of graph.

ARVGA[8] is the variational version of ARGA which also uses the graph autoencoder to learn the low dimensional embedding of graph.

4.2.2 Metrics

To measure the effectiveness of the proposed model on the link prediction task, the following two indexes are adopted: AUC score (the area under a receiver operating characteristic curve) and average precision (AP) score[7]. This work performs experiments 10 times for each dataset, and adopts the mean value with the standard errors as the final result.

4.2.3 Parameter settings

For Cora and Citesseer datasets, they have similar parameter settings because they have a similar number of nodes and edges. Therefore, the autoencoder model is trained for 220 iterations. The learning rate and discriminator's learning rate are set as 0.005 and 0.001, respectively. Pubmed dataset has more nodes and edges, the model is trained for 350 iterations. The learning rate is set as 0.009 and discriminator's learning rate is 0.001. For all experiments, the number of neurons of hidden layer and embedding layer are both set as 32, and the two hidden layers in the discriminator are set as 32-neural and 64-neural, the walk length of the random walk strategy is set as 2 and the total walk is set as 100. For the other baselines, the parameter settings are the same as that in the corresponding papers.

4.2.4 Results

The experimental results of link prediction task are shown in Table 2. For link prediction task, the proposed model outperforms all other baseline methods, achieving state-of-the-art results on the three benchmark datasets. All the *AUC* and *AP* scores are as higher as 97%. For the Cora dataset, the method improves the *AUC* score by 5.6% compared with the ARGA or ARVGA and improves the *AP* score by 4.2% and 4.8%, respectively, compared with the ARGA and ARVGA. For Citeseer dataset, the proposed meth-

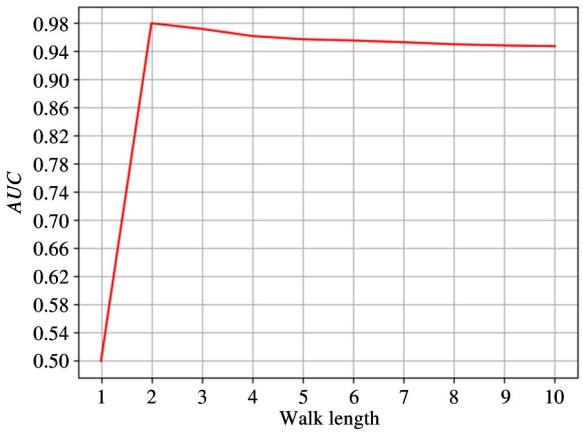
od improves the *AUC* score by 7.4% and the *AP* score by 6.1% compared with the ARGA, 18.8% and 15.5% compared with DeepWalk. On the Pubmed dataset, the method improves upon the ARGA by 1.9% in terms of the *AUC* score and 1.5% in the *AP* score. In addition, the standard errors of the proposed method are smaller, indicating that the model is more robust.

Table 2 The results of link prediction on various datasets

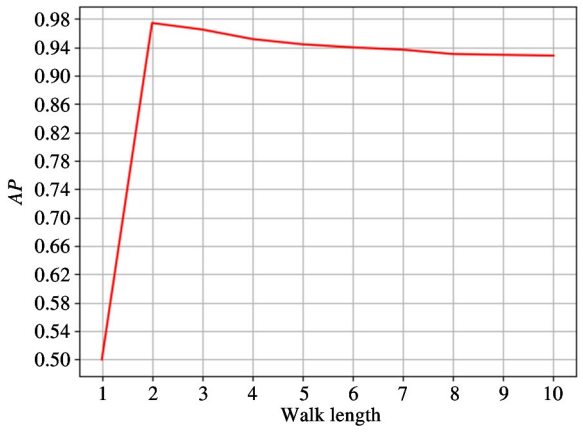
Approaches	Cora		Citeseer		Pubmed	
	AUC	*AP*	*AUC*	*AP*	*AUC*	*AP*
SC	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DeepWalk	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	96.4 ± 0.00	96.5 ± 0.00
VGAE	91.4 ± 0.01	92.6 ± 0.01	90.8 ± 0.02	92.0 ± 0.02	94.4 ± 0.02	94.7 ± 0.02
ARGA	92.4 ± 0.003	93.2 ± 0.003	91.9 ± 0.003	93.0 ± 0.003	96.8 ± 0.001	97.1 ± 0.001
ARVGA	92.4 ± 0.004	92.6 ± 0.004	92.4 ± 0.003	93.0 ± 0.003	96.5 ± 0.001	96.8 ± 0.001
DCAG-AE	98.0 ± 0.0002	97.4 ± 0.0002	99.3 ± 0.0001	99.1 ± 0.0002	98.7 ± 0.0002	98.6 ± 0.0002

4.2.5 Parameter study

Change the walk length of the random walk strategy from 1 to 10 to learn the effect of walk length on embedding. The results of the Cora dataset are shown in Fig. 3.



(a) AUC score



(b) Average precision score

Fig. 3 The embedded performance on different walk lengths of the Cora dataset

When the walk length is equal to 1, the PPMI matrix contains only the information of the node itself, which results in the learned embedding being invalid. When the walk length is 2, the embedding performance is the best. At this time, the PPMI matrix embeds the semantic information of the first-order and second-order neighbor nodes, which are also the neighbor nodes that has the greatest semantic correlation with the node. As the walk length increases, although the PPMI matrix captures the semantic information of higher-order neighbor nodes, it also embeds some semantic information with lower correlation. So, there must be a balance between more neighbor semantic information and neighbor information with higher semantic correlation.

4.3 Node clustering

After learning the latent embeddings, this paper uses the spectral clustering algorithm to cluster the nodes of the graph.

4.3.1 Baseline

In addition to the baselines compared in the link prediction task, it is also compared with other state-of-the-art clustering algorithms.

Kmeans is the basis of many clustering algorithms. Graph Encoder[22] uses a stacked autoencoder to learn representations for spectral graph clustering. DNGR[16] uses a stacked denoising autoencoder to reconstruct the PPMI matrix for learning a low dimensional embedding. RMSC[23] is a multi-view spectral clustering approach which considers both the information view of structure and content data. TADW[24] is based on matrix decomposition which adds node feature for the representation learning.

4.3.2 Metrics

In this paper, the following four indexes are used to measure the effectiveness of the proposed model on

the node clustering task: accuracy (ACC), normalized mutual information (NMI), precision, F-score (F1) and average rand index (ARI).

4.3.3 Results

The experimental results of node clustering tasks on the Cora and Citeseer datasets are shown in Table 3 and Table 4. Through the experimental results it can be seen that the proposed method performs better than other baselines for all evaluation metrics on the node clustering task. For instance, for the Cora dataset, the method outperforms ARGA by 7.1% in the *ACC* score, 8.0% in the *NMI* score and 16.5% in the *API* score. For the Citeseer dataset, the proposed method improves the *ACC* score by 8.2% and the *F*1 score by 5.6% compared with the ARGA, the precision score by 6.8% and the *NMI* score by 11.7% compared with the ARVGA.

Table 3 The results of node clustering on the Cora dataset

Approaches	*ACC*	*NMI*	*F*1	Precision	*ARI*
K-Means	0.492	0.321	0.368	0.369	0.230
SC	0.367	0.127	0.318	0.193	0.031
GraphEncoder	0.325	0.109	0.298	0.182	0.006
DeepWalk	0.484	0.327	0.392	0.361	0.243
DNGR	0.419	0.318	0.340	0.266	0.142
RMSC	0.407	0.255	0.331	0.227	0.090
TADW	0.560	0.441	0.481	0.396	0.332
GAE	0.596	0.429	0.595	0.596	0.347
VGAE	0.609	0.436	0.609	0.609	0.346
ARGA	0.640	0.449	0.619	0.646	0.352
ARVGA	0.638	0.450	0.627	0.624	0.374
DCAG-AE	0.711	0.529	0.688	0.730	0.517

Table 4 The results of node clustering on the Citeseer dataset

Approaches	*ACC*	*NMI*	*F*1	Precision	*ARI*
K-Means	0.540	0.305	0.409	0.405	0.279
SC	0.239	0.056	0.299	0.179	0.010
GraphEncoder	0.225	0.033	0.301	0.179	0.010
DeepWalk	0.337	0.088	0.270	0.248	0.092
DNGR	0.326	0.180	0.300	0.200	0.044
RMSC	0.295	0.139	0.320	0.204	0.049
TADW	0.455	0.291	0.414	0.312	0.228
GAE	0.408	0.176	0.372	0.418	0.124
VGAE	0.344	0.156	0.308	0.349	0.093
ARGA	0.573	0.350	0.546	0.573	0.341
ARVGA	0.544	0.261	0.529	0.549	0.245
DCAG-AE	0.655	0.378	0.602	0.617	0.374

4.4 Graph visualization

t-SNE algorithm[25] is used to realize the visualization of Cora dataset in the two-dimensional space. Fig. 4 shows the visualization results using different walk lengths on the Cora dataset.

As the walk length increases, the nodes in the cluster are close to each other. To verify this view, the intra-cluster distance is calculated, which is defined as the average value of the Euclidean distance from the node to the centroid in the cluster. As the walk length increases, the intra-cluster distance gradually decreases, therefore, the embeddings of the nodes in the cluster are more similar to each other. When the walk length increases, the nodes will obtain richer context information, and the nodes with similar contexts will obtain similar low-dimensional representations.

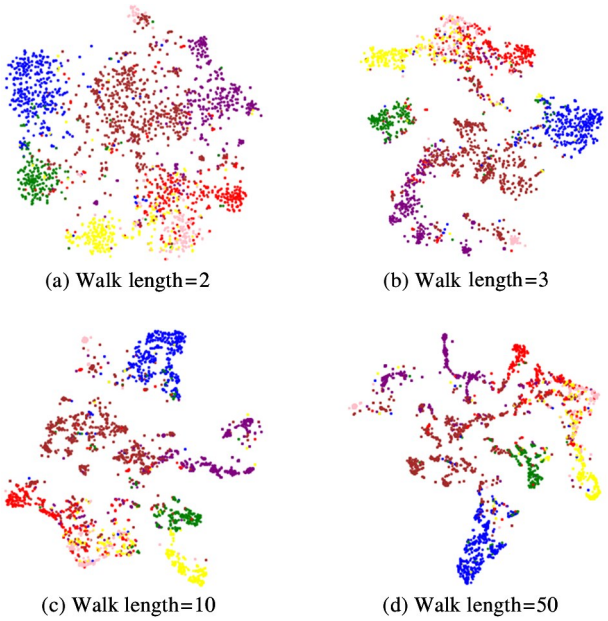


Fig. 4 Visualization of the Cora dataset on embeddings generated by DCAG-AE using different walk lengths. From left to right, the intra-cluster distance = {1.221, 0.902, 0.662, 0.548}

5 Conclusions

A novel deep convolutional adversarial graph autoencoder using PPMI matrix for graph embedding is proposed. Most of the existing graph embedding algorithms only focus on the topological information of the graph data, and ignore the semantic information of the graph data. To capture the semantic information between nodes, the proposed model encodes the PPMI matrix generated by the random walk strategy and the node content information into a latent representation, and then uses this latent representation to reconstruct the graph structure. Additionally, to enhance the robustness of embedding, this paper introduces the deep convolutional adversarial training schemes. The effectiveness of the proposed model is verified by evaluating

the performance of link prediction, node clustering and graph visualization tasks on the benchmark datasets.

References

[1] WANG Z, CHEN C, LI W. Predictive network representation learning for link prediction[C]// Proceedings of the 40th International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval, Tokyo, Japan, 2017: 969-972

[2] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks [EB/OL]. https://arxiv.org/pdf/1609.02907.pdf: arXiv, (2017-02-22), [2020-07-07]

[3] WANG C, PAN S, LONG G, et al. Mgae: marginalized graph autoencoder for graph clustering[C]// Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 2017: 889-898

[4] PEROZZI B, AL-RFOU R, SKIENA S. Deepwalk: online learning of social representations[C]// Proceedings of the 20th ACM Special Interest Group on Knowledge Discovery and Data Mining International Conference on Knowledge Discovery and Data Mining, New York, USA, 2014: 701-710

[5] CAI H, ZHENG V W, CHANG K C C. A comprehensive survey of graph embedding: problems, techniques, and applications[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2018, 30(9): 1616-1637

[6] GOYAL P, FERRARA E. Graph embedding techniques, applications, and performance: a survey[J]. *Knowledge-Based Systems*, 2018, 151: 78-94

[7] KIPF T N, WELLING M. Variational graph auto-encoders [EB/OL]. https://arxiv.org/pdf/1611.07308.pdf: arXiv, (2016-11-21), [2020-07-07]

[8] PAN S R, HU R Q, LONG G D, et al. Adversarially regularized graph autoencoder for graph embedding[C]// Proceedings of the 27th International joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018: 2609-2615

[9] GOODFELLOW I J, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial networks[EB/OL]. https://arxiv.org/pdf/1406.2661:arXiv, (2014-06-10), [2020-12-07]

[10] BULLINARIA J A, LEVY J P. Extracting semantic representations from word co-occurrence statistics: a computational study[J]. *Behavior Research Methods*, 2007, 39(3): 510-526

[11] PAN J Y, YANG H J, FALOUTSOS C, et al. Automatic multimedia cross-modal correlation discovery[C]// Proceedings of the 10th ACM Special Interest Group on Knowledge Discovery and Data Mining International Conference on Knowledge Discovery and Data Mining, Seattle, USA, 2004: 653-658

[12] TANG J, QU M, WANG M, et al. Line: large-scale information network embedding[C]// Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 2015: 1067-1077

[13] GROVER A, LESKOVEC J. node2vec: scalable feature learning for networks[C]// Proceedings of the 22nd ACM Special Interest Group on Knowledge Discovery and Data Mining International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2016: 855-864

[14] PEROZZI B, KULKARNI V, SKIENA S. Walklets: multiscale graph embeddings for interpretable network classification[EB/OL]. https://arxiv.org/pdf/1605.02115.pdf:arXiv, (2016-05-06), [2020-12-07]

[15] WANG D, CUI P, ZHU W. Structural deep network embedding[C]// Proceedings of the 22nd ACM Special Interest Group on Knowledge Discovery and Data Mining, San Francisco, USA, 2016: 1225-1234

[16] CAO S, LU W, XU Q. Deep neural networks for learning graph representations[C]// Proceedings of the Association for the Advance of Artificial Intelligence Conference on Artificial Intelligence, Phoenix, USA, 2016: 1145-1152

[17] RADFORD A, METZ L, CHINTALA S. Unsupervised representation learning with deep convolutional generative adversarial networks[EB/OL]. http://arxiv.org/pdf/1511.06434.pdf:arXiv, (2016-01-07), [2020-07-07]

[18] ZHUANG C, MA Q. Dual graph convolutional networks for graph-based semi-supervised classification[C]// Proceedings of the 2018 World Wide Web Conference, Lyon, France, 2018: 499-508

[19] SEN P, NAMATA G, BILGIC M, et al. Collective classification in network data[J]. *AI magazine*, 2008, 29(3): 93-93

[20] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[EB/OL]. https://arxiv.org/pdf/1301.3781.pdf: arXiv, (2013-09-07), [2020-07-07]

[21] TANG L, LIU H. Leveraging social media networks for classification[J]. *Data Mining and Knowledge Discovery*, 2011, 23(3): 447-478

[22] TIAN F, GAO B, CUI Q, et al. Learning deep representations for graph clustering[C]// Proceedings of the Association for the Advance of Artificial Intelligence Conference on Artificial Intelligence, Québec, Canada, 2014: 1293-1299

[23] XIA R, PAN Y, DU L, et al. Robust multi-view spectral clustering via low-rank and sparse decomposition[C]// Proceedings of the Association for the Advance of Artificial Intelligence Conference on Artificial Intelligence, Québec, Canada, 2014: 28(1)

[24] YANG C, LIU Z, ZHAO D, et al. Network representation learning with rich text information[C]// the International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 2015: 2111-2117

[25] VAN DER MAATEN L. Accelerating t-SNE using tree-based algorithms[J]. *The Journal of Machine Learning Research*, 2014, 15(1): 3221-3245

MA Xiuhui, born in 1997. She is a master student at the University of Chinese Academy of Sciences. She received her Bachelor degree from Hangzhou Dianzi University in 2018. Her research interests include graph neural network and computer vision.