Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Deep linear graph attention model for attributed graph clustering

Huifa Liao, Jie Hu [*], Tianrui Li, Shengdong Du, Bo Peng

School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, 611756, China

A R T I C L E I N F O

Article history:
Received 19 August 2021
Received in revised form 20 March 2022
Accepted 24 March 2022
Available online 31 March 2022

Keywords:
Attributed graph clustering
Linear model
Attention-based fusion
Graph neural network
Graph representation learning

A B S T R A C T

Attributed graph clustering is an important task for grouping the nodes in a graph. In recent years, the algorithms based on graph convolutional networks (GCN) have achieved promising performance. However, almost all existing methods ignore that the nonlinearity between two consecutive GCN layers is unnecessary for improving the performance of attributed graph clustering, and may even harm the efficiency of the model. In this paper, we propose a novel deep linear graph attention model for attributed graph clustering (DLGAMC), which consists of an attention-based aggregation module and a similarity preserve module. Specifically, we simply exploit cosine similarity to construct the attention for aggregation, which does not need to learn extra attention parameters. It is worth noting that the attention we designed not only explicitly considers the similarity of attribute information, but also implicitly takes into account the local graph structure. To select the proper order of aggregation, we propose an adaptive strategy to evaluate the smoothness of node representations, where intra-cluster distance and inter-cluster distance are the key indicators in this process. To learn node representations for clustering, we design a similarity preserve module to preserve local similarity and global dissimilarity of the smooth features obtained by multiple aggregations, which is different from the ideas in reconstruction-based methods. Finally, k-means is performed on the learned representations to obtain the cluster partition. Experiments on several datasets show that our algorithm achieves great performance in attributed graph clustering tasks.

1. Introduction

Attributed graph data widely exists in the real world, e.g., citation networks and social networks. Generally, an attributed graph consists of numerous nodes with attribute information, and the relationship between these nodes can be associated through edges. In order to mine useful information from attributed graph data, various important tasks in machine learning and data mining have been widely studied, including node classification [1], node clustering [2], link prediction [3], etc. Attributed graph clustering, which clusters the vertexes in a graph, can be applied to many scenarios, such as grouping on enterprise social network [4] and community detection [5].

A simple method for attributed graph clustering is to perform k-means [6] on the nodes directly. Unfortunately, such a traditional clustering algorithm fails to make full use of the rich information in the network topological structure, which usually leads to poor performance. For example, the referenced relationships play an important role in citation networks, and the edges in social networks directly reflect the friendship between users.

On the other hand, to exploit the structural information in the attributed graph, a lot of methods have been proposed to learn node representations for graph clustering, which are based on random walks [7], matrix factorization [8], autoencoder [9], etc. However, the above graph structure based methods only consider the topological structure and ignore node attributes, which always leads to the suboptimal attributed graph clustering performance. In general, the attribute information in citation networks and social networks describes the content of papers and the profiles of users respectively, which is also crucial for modeling attributed graphs.

Therefore, how to simultaneously use node attributes and graph structure to perform efficient clustering has aroused great research interest. Accordingly, a lot of algorithms for attributed graph clustering have been presented, including the methods based on discriminative models [10], generative models [11], matrix factorization [12], graph convolutional networks (GCN) [13], etc. Specifically, due to the powerful feature learning ability of deep neural networks, the algorithms based on GCN have achieved particularly excellent performance, e.g., deep attentional embedded graph clustering (DAEGC) [2], graph attention autoencoder (GATE) [14], structural deep clustering network (SDCN) [15], adversarially regularized variational graph autoencoder with reconstructing the node content and graph structure (ARVGA-AX)

[*] Corresponding author.
E-mail addresses: hfliao@my.swjtu.edu.cn (H. Liao), jiehu@swjtu.edu.cn (J. Hu), trli@swjtu.edu.cn (T. Li), sddu@swjtu.edu.cn (S. Du), bpeng@swjtu.edu.cn (B. Peng).

https://doi.org/10.1016/j.knosys.2022.108665
0950-7051/© 2022 Elsevier B.V. All rights reserved.

[16], variational graph autoencoder with Gaussian mixture models (GMM-VGAE) [17].

However, Wu et al. [18] demonstrated that the nonlinearities between GCN layers are unnecessary complexities for many downstream tasks, so they proposed a simple graph convolution (SGC) to simplify GCN. Inspired by their viewpoint, in this paper, we introduce SGC to construct a simple linear model for attributed graph clustering tasks. Actually, SGC cannot impose more weight on the more important adjacent nodes in the process of aggregation. Therefore, we further propose a linear graph attention network to improve SGC. We believe that the central node should impose more weight on the adjacent vertexes which share more similar attribute information with it. For example, if two papers refer to each other in a citation network, they are more likely to be of the same class when they share more similar content. Hence, we assign more attention to the adjacent nodes whose attribute information is more similar to the central vertex. Through this mechanism, each node will receive less noise from the adjacent vertexes belonging to other classes during the aggregation. On the other hand, cosine similarity is widely used to measure the similarity between two vectors. Therefore, in this article, we directly select cosine similarity to measure the similarity of attribute information and construct the attention for aggregation, which does not need to learn extra parameters and is simpler than graph attention network (GAT) [19]. In addition, with the progress of aggregation, the adjacent nodes sharing more local topological structure with the central vertex will mix more common attribute information, which further affects the attention value applied in the subsequent aggregation. The above analysis indicates that our attention also implicitly considers the local structural information. After multiple times of aggregation, we can get the smooth node representations, but their dimensions are as high as the original data. Therefore, we apply a single-layer neural network to learn low-dimensional meaningful representations.

In fact, the number of times for aggregation plays an important role in our linear graph attention network. Li et al. [20] showed that the graph convolution in GCN is actually a special form of Laplacian smoothing (i.e., mixing the representations of each node and its adjacent vertexes), which enables GCN to work, but it also suffers from over-smoothing (i.e., the nodes from different categories cannot be well distinguished, and even the representations of all vertexes will converge to a stable fixed point finally [21]). Therefore, it is necessary to select a proper number for the order of aggregation. Chen et al. [22] proposed two quantitative metrics to measure the smoothness and over-smoothness of node representations respectively, but new parameters are introduced meanwhile. Zhang et al. [23] exploited an intra-cluster distance without extra parameters to select the value of k for k-order graph convolution, but they ignored that inter-cluster distance is also important for evaluating the smoothness of node features. Based on the above work, we propose a new strategy that utilizes both inter-cluster distance and intra-cluster distance to determine an appropriate value for the order of aggregation adaptively.

In recent years, many GCN-based algorithms [2,14,16,17] learn node representations by reconstructing the original graph structure or node attributes. However, such reconstruction-based idea is not suitable for our linear graph attention network since they may destroy the smoothness of node representations. Different from the existing algorithms, we train the neural network by preserving both the local similarity and global dissimilarity of the smooth features. Specifically, as for the local similarity preserve module, we preserve the similarities between each node and its local most similar vertexes. Meanwhile, we preserve the dissimilarities between each node and its global most dissimilar vertexes in the global dissimilarity preserve module. After training the single-layer neural network in our linear graph attention network, k-means is performed on the learned representations to obtain clustering results. The overall framework of our model is shown in Fig. 1.

The main contributions of this paper can be summarized as follows:

- We introduce SGC into attributed graph clustering tasks and improve it by designing a novel linear graph attention network, which is conceptually simple and effective.
- To determine the number of times for aggregation, an adaptive strategy that exploits both intra-cluster distance and inter-cluster distance to evaluate the smoothness of node representations is proposed.
- Different from other algorithms which reconstruct the node attributes or original graph structure to learn graph representations, we obtain the node representations by preserving the local similarity and global dissimilarity of the smooth features.
- Experiments on several widely used datasets show that our simple linear model outperforms the GCN-based baselines with a certain margin on most of the cases.

The organization of this paper is described below. Section 2 reviews the related work of graph clustering methods. In Section 3, we introduce the preliminaries of our work. Section 4 gives the details of the proposed model. In Section 5, we conduct experiments from multiple perspectives to verify the effectiveness of our model. Finally, we conclude this paper in Section 6.

2. Related work

Generally, according to the information utilized in algorithms, graph clustering methods can be roughly divided into the following three types.

2.1. Using only node content

This category of algorithms only takes node attributes into consideration when clustering the nodes in the graph. There are many classical clustering methods belonging to this kind, e.g., k-means [6] and DBSCAN [24].

2.2. Using only network structure

Over the past several years, various methods which exploited only the structural information in the graph to learn node representations for graph clustering have been proposed. For instance, inspired by the word2Vec [25] in natural language processing, Perozzi et al. [7] modeled a set of linear sequences translated from network structure through random walks to learn graph representations. Grover et al. [26] designed biased random walks to improve Deepwalk [7]. Ribeiro et al. [27] explored a hierarchical network to learn latent representations based on the conception that the nodes with structurally similar neighborhoods should share similar representations. Tang et al. [28] thought that the two nodes are connected to each other or share more common neighbors should obtain more similar embedding, which is implemented by optimizing the objective function considering first-order proximity and second-order proximity simultaneously. Cao et al. [8] built the global representations by combining the different representations from different k-step local structural information based on the matrix factorization technique. Wang et al. [29] incorporated community structure into network representations by jointly optimizing a modularity-based community detection model and a representations learning model

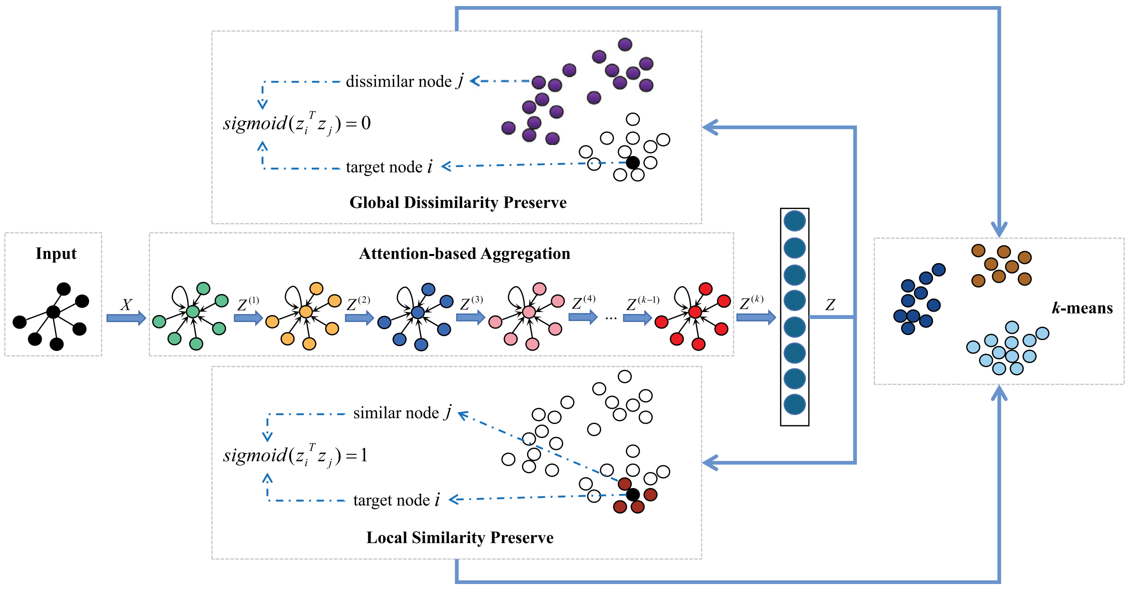


Fig. 1. The overall framework of the proposed deep linear graph attention model for attributed graph clustering (DLGAMC). Our model consists of three stages. Firstly, each node in the graph aggregates the representations of adjacent nodes k times based on the non-parametric attention to obtain the smooth features, where the number of times for aggregation can be determined by utilizing both inter-cluster distance and intra-cluster distance. Then, we feed the smooth features into a single-layer neural network, which is trained by preserving the local similarity and global dissimilarity of the smooth features. Finally, we get the clustering results by performing k-means on the learned node representations.

with nonnegative matrix factorization. Tian et al. [9] exploited a deep neural network with multiple stacked autoencoders to learn graph embedding. Cao et al. [30] built a random surfing model to capture network structural information and introduced a denoising autoencoder to learn low-dimensional representations. Wang et al. [31] constructed an autoencoder to learn node representations, where the first-order proximity and second-order proximity are retained together.

2.3. Using both node attributes and graph structure

The above two types of algorithms often lead to poor performance since they only consider node content or network structure. Therefore, a lot of methods have been studied to integrate graph structure and node attributes for attributed graph clustering. For example, Yang et al. [10] combined node content and connections for community detection based on a discriminative model. Chang et al. [32] exploited a generative model to explicitly tie links with the content of documents. Sun et al. [33] built a unified generative topic model to utilize both the structural information in the network and the text of documents. Xu et al. [34] presented a Bayesian probabilistic method to model network structure and attribute information jointly. Ruan et al. [35] integrated content information into community detection to strengthen community signal since the links in the network usually contain noise. To improve the performance of community detection, Yang et al. [36] modeled the interaction between node content and graph structure through a generative model. Liu et al. [37] introduced a content dissemination theory, which naturally combined content information with graph structure. He et al. [11] proposed a generative model to identify network communities and semantics, where both network topologies and the node contents with semantic information are modeled. Yang et al. [12] demonstrated that Deepwalk is actually equivalent to matrix factorization and then incorporated node attributes into graph embedding based on matrix factorization technology. Wang et al. [38] integrated attribute matrix and community membership matrix into the framework of nonnegative matrix factorization to detect community and infer semantics. Li et al. [39] combined the attribute matrix with the network structure embedding matrix and formulated their embedding approach as a nonnegative matrix factorization optimization problem. Li et al. [40] proposed a deep network representation learning framework consisting of a structural role proximity enhancement autoencoder and a neighbor-modified Skip-Gram component, which can effectively fuse the information captured from node attributes and network topological structure.

Owing to the development of graph neural network over recent years, a series of GCN-based methods have achieved the promising clustering performance in attributed graphs. For instance, Kipf et al. [13] proposed the graph auto-encoder (GAE) based on GCN to learn node representations and presented the variational graph auto-encoder (VGAE) based on GAE to exploit the latent variables. Wang et al. [41] performed a marginalization process in a single-layer graph autoencoder, where the randomness is introduced into node attributes. Pan et al. [42] enforced the latent representations to match prior distribution by introducing the generative adversarial module into graph autoencoder. Bo et al. [15] integrated the structural information into deep clustering by combining the basic autoencoder with GCN. Pan et al. [16] learned graph representations through an adversarially regularized model that represents the network structure and node attributes in a continuous vector space. Hui et al. [17] combined the variational graph autoencoder with Gaussian mixture models for attributed graph clustering. Zang et al. [43] proposed a node-to-node geodesic similarity-based model for learning graph embedding, where a network structure and a graph structure augmentation method are designed to alleviate the oversmoothing problem and enhance the stability of embedding respectively.

Besides, some methods tried to improve the performance of model from the perspective of attention mechanisms. For example, Wang et al. [2] built a simple graph attention autoencoder with network structure reconstruction to learn graph representations and performed deep embedding clustering [44] jointly to obtain clustering results. Salehi et al. [14] stacked multiple graph attention layers to construct the model and then learned node representations by reconstructing the graph structure and node attributes simultaneously. The above two models are all based on GAT, which can assign varying attention to different nodes in a neighborhood. Huo et al. [45] utilized a cross-attention module to fuse the two types of information extracted from a basic auto-encoder and a GCN-based auto-encoder respectively, and integrated deep embedding clustering into the model for clustering tasks. Unlike the above methods, Peng et al. [46] performed representations learning and cluster assignment jointly in a unified framework, which is equipped with attention mechanisms for dynamically fusing the features of node attributes and network structure, and adaptively concatenating the multi-scale features from different layers.

Unfortunately, all of the above-mentioned GCN-based methods ignore that the model without nonlinearities can still achieve the promising performance in attributed graph clustering tasks. Compared with SGC, although Zhang et al. [23] utilized an adaptive graph convolution (AGC) to obtain high-order node representations, they neglected that the collapsed weight matrix of k-layer GCN for learning low-dimensional meaningful representations is also critical.

3. Preliminaries

3.1. Basic notations

Given a graph $G = (V, E, X)$, where $V = \{v_i\}_{1,2,\ldots,n}$ is a set of $|N|$ nodes, E contains the links between nodes, $X = \{x_i\}_{1,2,\ldots,n}$ is an attribute matrix, and x_i denotes the attribute information of vertex i. Adjacency matrix A represents the structural information in the graph, where $A_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $A_{ij} = 0$.

3.2. Problem definition

The target of the attributed graph clustering algorithm is to partition all the nodes in a graph into $|C|$ disjoint clusters, where only the network topological structure and attribute information are available, excluding the labels of any nodes.

3.3. A simplified graph convolutional networks

Firstly, we review the principle of GCN. The layer-wise propagation rule of GCN is defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

where $\tilde{A} = A + I_N$ denotes the adjacency matrix of the graph with self-connection, I_N is an identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is a degree matrix, $\sigma(\cdot)$ is an activation function (e.g., Relu [47]), $H^{(l)}$ means the output representations of the lth layer ($H^{(0)} = X$), and $W^{(l)}$ denotes the learnable weight matrix in the lth layer.

Then, Wu et al. [18] considered that the nonlinearities between GCN layers are not critical in many downstream tasks. Therefore, they removed the nonlinear activation functions in GCN and only remained the final softmax. After that, the output of the k-layer GCN is represented as:

$$Y = softmax(P \ldots PPXW^{(1)}W^{(2)} \ldots W^{(k)}) \tag{2}$$

where $P = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.

Finally, Eq. (2) can be simplified by collapsing the normalized adjacency matrix P and reparameterizing the multiple weight matrices into a single matrix:

$$Y = softmax(P^kXW) \tag{3}$$

4. Proposed method

In this section, we introduce the details of our proposed deep linear graph attention model for attributed graph clustering (DLGAMC). We first construct a linear graph attention network, which can be divided into two stages, i.e., attention-based aggregation and representations learning. Then, we design an adaptive strategy to determine the number of times for aggregation, where the core is that evaluating the smoothness of node representations through intra-cluster distance and inter-cluster distance. Based on the smooth features obtained from multiple aggregations, a similarity preserve module is applied to learn node representations for clustering finally.

4.1. Linear graph attention network

Although SGC simplifies GCN, it cannot identify the more important neighbors in the process of graph convolution actually. In this paper, we improve SGC by devising a linear graph attention network, which can impose more weight on the more important links. The layer-wise aggregation principle of the linear graph attention network is defined as:

$$z_i^{(l)} = \sum_{\tilde{A}_{ij}=1} \alpha_{ij}^{(l)} z_j^{(l-1)} \tag{4}$$

where $z_i^{(l)}$ denotes the output representations of node i in the lth layer, $z_i^{(0)} = x_i$ is the original attribute value of vertex i, and $\alpha_{ij}^{(l)}$ represents the attention value applied to node j by vertex i in the lth aggregation.

Considering that, if we impose more attention on the edges connecting the two nodes that are more likely to belong to the same class, the central node will receive less noise from the vertexes belonging to other classes. Moreover, we think that the central node is more likely to belong to the same category as its adjacent vertexes which share more similar attribute information with it. Here, we select cosine similarity to measure the proximity between the attribute information of the central node and its adjacent vertexes. As we know, the cosine similarity between the features of node i and vertex j can be represented as:

$$s_{ij}^{(l)} = \frac{z_i^{(l-1)} \cdot z_j^{(l-1)}}{\left|z_i^{(l-1)}\right| \cdot \left|z_j^{(l-1)}\right|} \tag{5}$$

where $z_i^{(l-1)}$ and $z_j^{(l-1)}$ denote the representations of vertex i and node j after the $(l-1)$th aggregation respectively. Then, we exploit the aforementioned cosine similarity to construct the attention for aggregation. Specifically, the attention value imposed on node j by vertex i in the lth layer can be obtained by normalizing Eq. (5) with a softmax function:

$$\alpha_{ij}^{(l)} = softmax_j(s_{ij}^{(l)}) = \frac{\exp(s_{ij}^{(l)})}{\sum_{\tilde{A}_{it}=1,\ t\in\{1,2,\ldots,n\}} exp(s_{it}^{(l)})} \tag{6}$$

Notably, the adjacent nodes which share more local structural information with the central vertex can obtain more common attribute information during the aggregation, so they will be imposed the larger attention value in the subsequent aggregation. In other words, the above attention we designed also implicitly takes into account the similarity of local topological structure.

After k times of aggregation, we can get the smooth features $Z^{(k)}$. Finally, we apply a single-layer neural network to obtain the final node representations:

$$Z = Z^{(k)}W \tag{7}$$

where W denotes a trainable weight matrix.

4.2. How to determine k

The aggregation in the linear graph attention network enables the node features to smooth, but it also causes the node representations to suffer from over-smoothing after performing too many aggregations. Hence, it is necessary to choose an appropriate value for k.

AGC selects the order of graph convolution through intra-cluster distance:

$$intra(C) = \frac{1}{|C|}\sum_{c\in C}\frac{1}{|c|(|c|-1)}\sum_{v_i,v_j\in c,v_i\neq v_j}\left\|z_i - z_j\right\|_2 \tag{8}$$

where C denotes the cluster partition of all nodes, the number of clusters is represented as $|C|$, $|c|$ means the number of vertexes in cluster c, z_i and z_j denote the smooth representations of node i and vertex j respectively. In our model, we perform k-means on the smooth features to obtain the cluster partition.

The value of $intra(C)$ becomes smaller as k increases but gets larger for the first time when the phenomenon of over-smoothing begins to happen. The change of intra-cluster distance between two successive times can be expressed as:

$$d\_intra(t) = intra(C^{(t)}) - intra(C^{(t-1)}) \tag{9}$$

where $C^{(t)}$ denotes the cluster partition for the node features after tth aggregation. As t increases until $d\_intra(t) > 0$ for the first time, then AGC simply chooses k as $t_{select} = t - 1$.

The ideal smoothness of node representations for clustering should satisfy that the inter-cluster distance is large and the intra-cluster distance is small. However, the above method only considers intra-cluster distance and ignores that inter-cluster distance decreases gradually as k increases, which will have a negative effect on clustering. Therefore, t_{select} may not be the most appropriate value for k. Here, we introduce the inter-cluster distance as follows:

$$inter(C) = \frac{1}{|C|}\sum_{c\in C}\frac{1}{|c|(|N|-|c|)}\sum_{v_i\in c,v_j\notin c}\left\|z_i - z_j\right\|_2 \tag{10}$$

where $|N|$ represents the number of vertexes in the graph. We denote the difference between inter-cluster distance and intra-cluster distance as $inter\_intra(t)$, and it can be defined as:

$$inter\_intra(t) = inter(C^{(t)}) - intra(C^{(t)}) \tag{11}$$

We think that the more appropriate value for k should maximize the value of $inter\_intra(t)$ when $t \leq t_{select}$. Therefore, we determine the value of k by the following formula:

$$k = \underset{t\leq t_{select}}{argmax}(inter\_intra(t)) \tag{12}$$

4.3. Similarity preserve of smooth features

After performing multiple aggregations on the node features, we can get the smooth representation of each vertex. As for each node, we calculate the cosine similarity between it and other vertexes. Then, we choose the sim_num most similar node pairs for each vertex to generate similar set S_s and select the $dissim\_num$ most dissimilar vertex pairs for each node to generate dissimilar set S_d. As for each node pair, we give a label for it, where $label_{ij} = 1$ if $(v_i, v_j) \in S_s$, and $label_{ij} = 0$ if $(v_i, v_j) \in S_d$.

Generally, the dissimilar node pairs are much larger than similar node pairs. In order to balance the number of negative samples and positive samples, we randomly divide the dissimilar set S_d into $batch\_num$ equal batches, where $batch\_num = \frac{dissim\_num}{sim\_num}$. We set the number of iterations as $batch\_num$ and optimize the single-layer neural network by minimizing the following cross entropy loss function:

$$loss = -\frac{1}{2sim\_num}\sum_{(v_i,v_j)\in S_s\cup S_d^{iter}} \times\ (label_{ij}\ln y_{ij} + (1 - label_{ij})\ln(1 - \ln y_{ij})) \tag{13}$$

where S_d^{iter} is the subset of dissimilar set S_d for the $iter$th iteration. y_{ij} denotes the similarity between vertex i and node j, and it is defined as follows:

$$y_{ij} = sigmoid({z_i}^T z_j) = \frac{1}{1 + e^{-{z_i}^T z_j}} \tag{14}$$

where z_i and z_j represent the representations of node i and vertex j respectively. Finally, we apply k-means on the learned representations to obtain clustering results. The executive process of our complete method is shown in Algorithm 1.

Algorithm 1: Deep Linear Graph Attention Model for Attributed Graph Clustering (DLGAMC)

```
   Data: node attributes $X$, adjacency matrix $A$, $sim\_num$,
         $dissim\_num$
   Result: cluster partition $C$
   /* obtain the initial value with no aggregation
      */
1  Run $k$-means on $X$ to obtain the initial cluster partition
     $C^{(0)}$;
2  Obtain $intra(C^{(0)})$ and $inter(C^{(0)})$ by Eqs. (8) and (10)
     respectively;
3  Calculate $inter\_intra(0)$ by Eq. (11);
   /* get smooth features $Z^{(k)}$ through
      attention-based aggregation module */
4  Initialize: $d\_intra(0) = -inf$, $max\_inter\_intra = 0$, $t = 0$,
     $k = 0$;
5  while $d\_intra(t) < 0$ do
6      if $inter\_intra(t) > max\_inter\_intra$ then
7          $max\_inter\_intra = inter\_intra(t)$; // update the
             maximum $inter\_intra$
8          $k = t$; // update the selectd $k$
9      end
10     $t = t + 1$; // for the next aggregation
11     Obtain smooth features $Z^{(t)}$ by Eq. (4);
          // attention-based aggregation
12     Run $k$-means on $Z^{(t)}$ to get cluster partition $C^{(t)}$;
13     Obtain $intra(C^{(t)})$ and $inter(C^{(t)})$ by Eqs. (8) and (10)
         respectively;
14     Calculate $d\_intra(t)$ and $inter\_intra(t)$ by Eqs. (9) and
         (11) respectively;
15 end
   /* Sample the node pairs for training */
16 Calculate the similarity between all nodes based on $Z^{(k)}$;
17 Generate similar set $S_s$ and dissimilar set $S_d$;
18 Randomly divide the dissimilar set $S_d$ into
     $batch\_num = \frac{dissim\_num}{sim\_num}$ equal batches;
   /* training the single-layer neural network
      through similarity preserve */
19 for $iter = 1$ to $batch\_num$ do
20     Merge similar set $S_s$ and dissimilar subset $S_d^{iter}$;
          // node pairs for this iteration
21     Optimize neural network by minimizing Eq. (13);
22 end
23 Obtain the final cluster partition $C$ by performing $k$-means
     on the learned representations.
```

Table 1
Summary of datasets.

Datasets	Edges	Nodes	Features	Classes
Cora	5429	2708	1433	7
Citeseer	4732	3327	3703	6
Pubmed	44338	19717	500	3
Wiki	17981	2405	4973	17

5. Experiments

In this section, we first conduct experiments on several widely used datasets to evaluate the performance of the proposed method. Besides that, we perform extensive additional experiments to analyze the effectiveness of our model.

5.1. Datasets

We evaluate the performance of our model on three widely used citation network datasets (Cora, Citeseer, and Pubmed) and a webpage network dataset (Wiki). In each citation network, the nodes denote papers, and the reference relationships between papers are represented as edges. A webpage network generally includes many webpages and the hyperlinks connecting them, in which vertexes and edges are used to represent the webpages and hyperlinks respectively. The detailed information of these datasets is described as follows:

- **Cora** [1] [2]: The Cora dataset consists of 5429 edges and 2708 nodes with feature dimension 1433. There are 7 categories of papers in this dataset, including Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, and Theory.
- **Citeseer** [1] [2]: The Citeseer dataset is composed of 4732 edges and 3327 vertexes with feature dimension 3703, where each node belongs to one of the following 6 classes of papers: Agents, AI, DB, IR, ML, and HCI.
- **Pubmed** [1] [2]: The Pubmed dataset includes 44338 edges and 19717 nodes with feature dimension 500. The papers in this dataset can be divided into the following 3 categories: "Diabetes Mellitus, Experimental", "Diabetes Mellitus Type 1", and "Diabetes Mellitus Type 2".
- **Wiki** [3]: The Wiki dataset is made up of 17981 edges and 2405 vertexes with feature dimension 4973. In addition, each webpage in this network belongs to one of the 17 classes.

The statistical information of all datasets is listed in Table 1.

5.2. Evaluation metrics

In this paper, We utilize four widely used metrics to evaluate the performance of graph clustering methods: Accuracy (ACC), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and macro F1 score (F1).

ACC [44] is the most frequently-used evaluation metric among all these indices, and it is defined as follows:

$$ACC = \max_{m} \frac{\sum_{i=1}^{n} \{1|y_i == m(pred_i)\}}{n} \tag{15}$$

where m is a mapping function which matches the predicted cluster to true cluster by ranging over all the possible mapping, and it can be optimized by the Kuhn–Munkres algorithm [48]; n represents the number of all nodes; y_i means the true label of vertex i; $pred_i$ denotes the predicted label for node i. The target of ACC is to obtain the maximal matching of cluster assignment.

NMI [49] measures the normalized mutual information between the true clusters C and predicted clusters C', and it can be calculated by the following formula:

$$NMI(C, C') = \frac{MI(C, C')}{\sqrt{H(C)H(C')}} \tag{16}$$

where $H(C)$ and $H(C')$ denote the entropy of C and C' respectively. $MI(C, C')$ represents the mutual information between C and C', and it can be expressed as:

$$MI(C, C') = \sum_{c_i \in C, c_j \in C'} p(c_i, c_j) \log \frac{p(c_i, c_j)}{p(c_i)p(c_j)} \tag{17}$$

where $p(c_i)$ and $p(c_j)$ denote the probability that the node belongs to cluster c_i and c_j respectively, and $p(c_i, c_j)$ represents the joint probability that the vertex belongs to cluster c_i and c_j in the meantime.

ARI [50] calculates the similarity between the distributions of true labels and predicted labels, and it is defined as Eq. (18) in Box I

where $\begin{pmatrix} n \\ 2 \end{pmatrix} = C_n^2 = \frac{n(n-1)}{2}$, n_{ij} denotes the number of nodes belonging to true cluster i and predicted cluster j at the same time, a_i represents the number of nodes that belong to true cluster i and obtain the same label in the predicted clusters, and b_j denotes the number of vertexes belonging to predicted cluster j and getting the same label in the true clusters.

F1 [51] measures the clustering performance by the following formula:

$$F1_{macro} = \frac{2Precision_{macro}Recall_{macro}}{Precision_{macro} + Recall_{macro}} \tag{19}$$

where $Precision_{macro} = \frac{1}{|C|}\sum_{i=1}^{|C|} Precision_i$, and $Recall_{macro} = \frac{1}{|C|}\sum_{i=1}^{|C|} Recall_i$. $Precision_i$ and $Recall_i$ are the precision and recall of class i respectively.

5.3. Baselines

We compare the proposed DLGAMC with the following three types of graph clustering algorithms:

Using only node content

- ***k*-means** [6] is one of the most classic clustering methods using only node attributes.

Using only network structure

- **Deepwalk** [7] adopts random walks to construct a set of linear sequences for learning vertex representations.
- **GraphEncoder** [9] learns deep embedding for graph clustering through multiple stacked autoencoders.
- **DNGR** [30] introduces a random surfing model to extract the structural information of the graph and then exploits a stacked denoising autoencoder to learn node representations.
- **M-NMF** [29] integrates community structure into graph representations through a modularized nonnegative matrix factorization model.

Using both node attributes and graph structure

- **TADW** [12] fuses graph representations with attribute information based on matrix factorization.
- **GAE** [13] learns node representations by constructing a simple autoencoder that consists of a two-layer GCN and an inner product decoder.

[1] https://github.com/tkipf/gae/tree/master/gae/data
[2] https://linqs.soe.ucsc.edu/data
[3] https://github.com/albertyang33/TADW/tree/master/wiki

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}} \tag{18}$$

Box I.

Table 2
Clustering results on three citation network datasets and a webpage network dataset.

Method	Input	Cora				Citeseer				Pubmed				Wiki			
		ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
k-means	A	0.354	0.176	0.096	0.311	0.404	0.181	0.138	0.375	0.595	0.314	0.280	0.581	0.339	0.316	0.074	0.245
Deepwalk	S	0.529	0.384	0.291	0.435	0.390	0.131	0.137	0.305	0.647	0.238	0.255	0.530	0.385	0.324	–	0.257
GraphEncoder	S	0.301	0.059	0.046	0.230	0.293	0.057	0.043	0.213	0.531	0.210	0.184	0.506	–	–	–	–
DNGR	S	0.419	0.318	0.142	0.340	0.326	0.180	0.043	0.300	0.468	0.153	0.059	0.445	0.376	0.359	–	0.254
M-NMF	S	0.423	0.256	0.161	0.320	0.336	0.099	0.070	0.255	0.470	0.084	0.058	0.443	–	–	–	–
TADW	A & S	0.536	0.366	0.240	0.401	0.529	0.320	0.286	0.436	0.565	0.224	0.177	0.481	–	–	–	–
GAE	A & S	0.530	0.397	0.293	0.415	0.380	0.174	0.141	0.297	0.632	0.249	0.246	0.511	0.173	0.119	–	0.154
VGAE	A & S	0.592	0.408	0.347	0.456	0.392	0.163	0.101	0.278	0.619	0.216	0.201	0.478	0.287	0.303	–	0.205
ARGE	A & S	0.640	0.449	0.352	0.619	0.573	0.350	0.341	0.546	0.668	0.305	0.295	0.656	0.414	0.395	–	0.383
ARVGE	A & S	0.638	0.450	0.374	0.627	0.544	0.261	0.245	0.529	0.690	0.290	0.306	0.678	0.416	0.400	–	0.378
AGC	A & S	0.689	0.537	–	0.656	0.670	0.411	–	0.623	0.698	0.316	–	0.687	0.477	0.453	–	0.404
DAEGC	A & S	0.704	0.528	0.496	0.682	0.672	0.397	0.410	0.636	0.671	0.266	0.278	0.659	–	–	–	–
ARVGA-AX	A & S	0.711	0.526	0.495	**0.693**	0.581	0.338	0.301	0.525	0.640	0.239	0.226	0.644	–	–	–	–
GMM-VGAE	A & S	0.715	0.544	–	0.678	0.674	0.423	–	0.632	**0.710**	0.303	–	0.697	–	–	–	–
SGC	A & S	0.724	0.547	0.502	0.672	0.680	0.418	0.425	0.637	0.678	0.285	0.288	0.674	0.440	0.438	0.205	0.393
DLGAMC (Ours)	A & S	**0.737**	**0.560**	**0.522**	0.691	**0.687**	**0.431**	**0.436**	**0.641**	0.708	**0.318**	**0.332**	**0.700**	**0.490**	**0.454**	**0.284**	**0.423**

- **VGAE** [13] combines GAE with the principle of variational autoencoder to exploit latent variables.
- **ARGE and ARVGE** [42] introduce a generative adversarial module into GAE and VGAE respectively.
- **AGC** [23] exploits a *k*-order graph convolution to get node representations and then applies spectral clustering [52] on the obtained representations.
- **DAEGC** [2] learns node representations through a graph attention autoencoder and performs deep embedding clustering jointly in a unified framework.
- **ARVGA-AX** [16] is a variant of ARVGE, which learns vertex representations by reconstructing node content and graph structure.
- **GMM-VGAE** [17] integrates the Gaussian mixture models into variational graph autoencoder for attributed graph clustering.
- **SGC** [18] is a simplified linear version of GCN. We extend SGC into unsupervised learning tasks by designing a similarity preserve module and an adaptive strategy of determining *k* value.

5.4. Parameter settings

For our DLGAMC, the value of *k* is determined by the proposed strategy which considers inter-cluster distance and intra-cluster distance meanwhile, and we will discuss it in Section 5.7. In addition, we set the learning rate, *sim_num*, dropout rate, and the size of the single-layer neural network as 0.001, $0.01|N|$, 0.8, and 500 for all datasets, respectively. The number of dissimilar node pairs is much larger than similar vertex pairs, so we roughly fine tune the value of *dissim_num* in $\{0.6|N|, 0.7|N|, 0.8|N|, 0.9|N|\}$ by Davies–Bouldin index (DBI) [53], which can directly evaluate the quality of clustering results without ground truth. The parameter settings of SGC are the same as that of DLGAMC for a fair comparison.

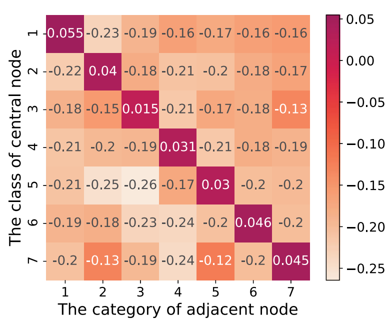
5.5. Clustering results

We run *k*-means, SGC, and DLGAMC 10 times. The average clustering results are recorded in Table 2, where *A* and *S* denote the node attributes and graph structure respectively. For other baselines, we quote that from the original papers if their authors conducted the algorithms on the graph clustering tasks, otherwise we quote that from Ref. [2] (Cora, Citeseer, and Pubmed) and Ref. [23] (Wiki), or mark it as '-'. From Table 2, we can observe that the attributed graph clustering methods which exploit both node attributes and graph structure generally outperform the algorithms using only node content or network structure. It shows that both node attributes and graph structure contain rich meaningful information for graph clustering, and the absence of either will lead to poor performance. In addition, SGC achieves competitive performance compared with GCN-based baselines, which demonstrates that the simple linear model can also obtain great results in attributed graph clustering tasks. Moreover, the proposed DLGAMC outperforms SGC and other baselines on most of the evaluation metrics. The above observations show that the overall design of our model is valid.

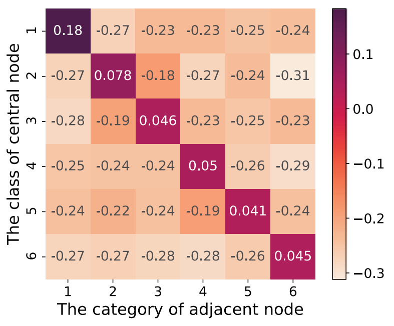
5.6. The effectiveness of cosine attention

In that case, if the central node imposes the same weight on its adjacent vertexes during the aggregation, the attention value applied to adjacent nodes by central vertex *i* is the reciprocal of degree, i.e., $average\_attention_i = 1.0/\sum_j \tilde{A}_{ij}$. The effective attention should try to satisfy that the attention values imposed on the edges which link the two nodes belonging to the same classes greater than average attention, and the weights applied to other links less than average attention. Therefore, we use the following metric to evaluate attention:

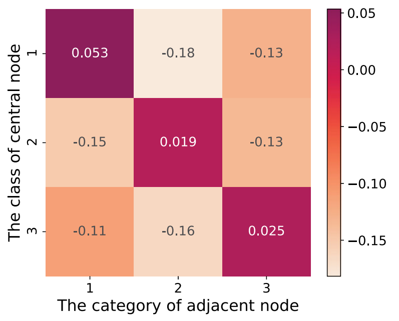
$$edge\_attention_{c_m c_n} = \frac{\sum_{i\in c_m, j\in c_n, \tilde{A}_{ij}=1} \frac{\alpha_{ij} - average\_attention_i}{average\_attention_i}}{\sum_{i\in c_m, j\in c_n} \tilde{A}_{ij}} \tag{20}$$



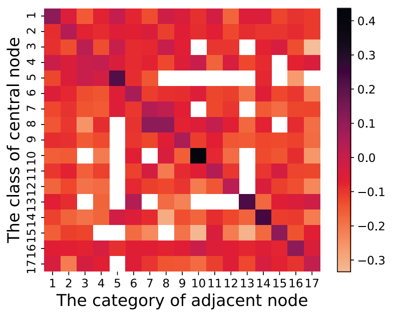
(a) Cora



(b) Citeseer

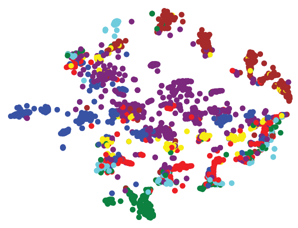


(c) Pubmed

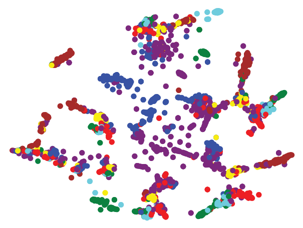


(d) Wiki (blank space means no edge)

Fig. 2. The edge attention values on Cora, Citeseer, Pubmed, and Wiki.



(a) SGC (50)



(b) SGC (100)

Fig. 3. Visualization of the node representations obtained by the graph convolution filter of SGC on Cora.

where c_m and c_n denote the true class m and n respectively. The $edge\_attention_{c_m c_n}$ represents the average normalized attention value imposed on the adjacent nodes belonging to class c_n by the central vertexes whose true class are c_m. α_{ij} is the attention value applied to vertex j by node i.

We can directly calculate the attention value at any time during the aggregation since no parameter needs to learn in our attention-based module. To verify the attention in the proposed DLGAMC, we exploit the attention value for the first aggregation (i.e., $\alpha_{ij} = \alpha_{ij}^{(1)}$) to calculate the value of *edge_attention* on all datasets and show the results in Fig. 2. In these heat maps, the values on the diagonal are generally higher than other values, which demonstrates that the edges connecting the two nodes in the same class obtain the attention value greater than other edges linking the two vertexes from the different categories. In other words, the cosine attention in our model is reasonable.

Moreover, we conduct additional experiments on Cora to examine the effect of our attention-based module when going deep. Firstly, we exploit t-SNE [54] to visualize the node representations obtained by performing 50, 100 aggregations on the original features respectively using the graph convolution filter in SGC, and the visualized results are shown in Fig. 3. Meanwhile, we apply 50, 100, 200 aggregations on the original features respectively through our attention-based aggregation module to get the smooth presentations and visualize them in Fig. 4. From Fig. 3, we can observe that the node representations have suffered from serious over-smoothing after only 50 graph convolutions, and almost all nodes are indistinguishable after performing 100 graph convolutions. As for our attention-based module, we can find from Fig. 4 that most vertexes are concentrated in their real clusters after applying 50 aggregations. It is worth mentioning that many nodes with the same class are still clustered together even after 200 aggregations. The above experiments demonstrate that the central node exactly receives less noise from the nodes belonging to other categories in our attention-based aggregation module, and our linear graph attention network has great potential in going deep.

5.7. The value of k for aggregation

In this section, we run k-means 10 times on the node representations among different orders of aggregation in our model and then calculate the average value of *d_intra* and *inter_intra*. The above values are shown in Fig. 5. According to our proposed strategy in Section 4.2, we determine the value of k for Cora, Citeseer, Pubmed, and Wiki as 9, 6, 25, and 1, respectively.

To verify the effectiveness of the proposed strategy, we run our method among different values of k on all datasets and show the clustering results in Fig. 6. From it, we can observe that each evaluation metric first increases and then decreases with the increase of k. The above observation shows that the node representations become smooth gradually and suffer from over-smoothing with the increase of aggregation times. In addition, DLGAMC obtains near the best clustering results on most of the cases through exploiting the value of k determined by our proposed strategy. The above results indicate that the proposed adaptive selection strategy is effective.

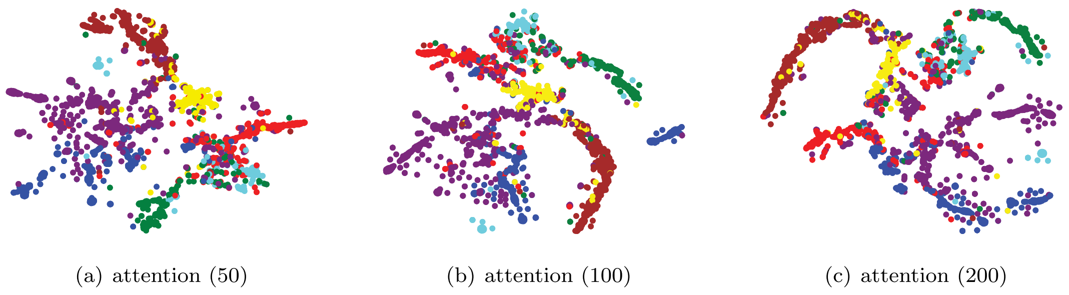


Fig. 4. Visualization of the node representations obtained by the attention-based aggregation of DLGAMC on Cora.

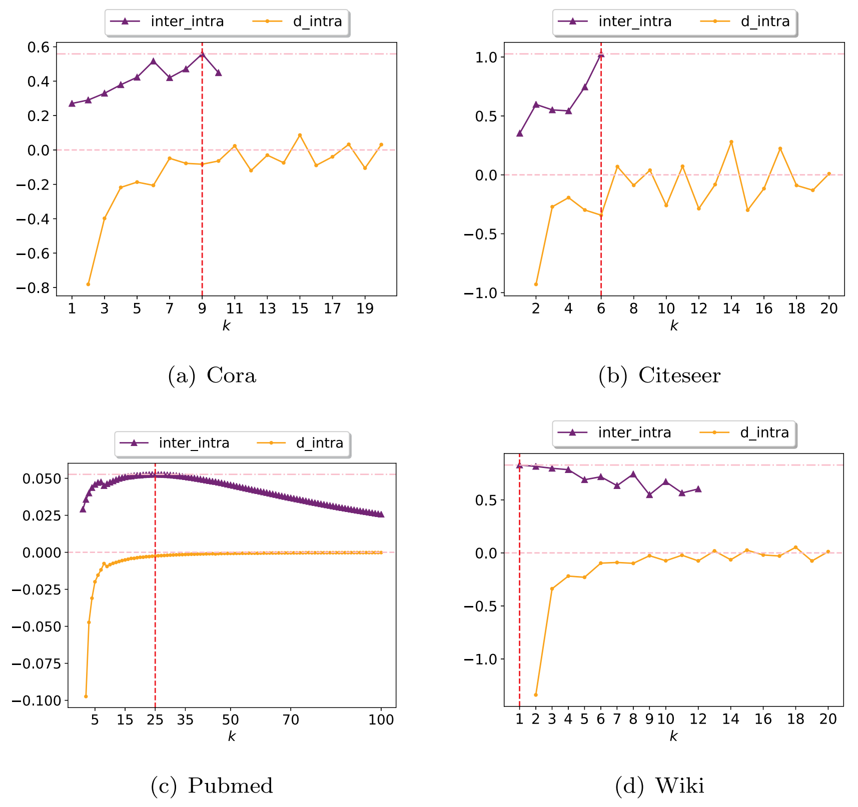


Fig. 5. The *inter_intra* and *d_intra* in our model among different values of k.

Moreover, we compare our strategy with the approach that considers only intra-cluster distance on Pubmed. From the results in Fig. 5, we can find that the value of *d_intra* is negative all the time when $k \leq 100$, which indicates that the value selected for k will larger than 100 if we only consider intra-cluster distance. However, in Fig. 6, we can observe that the node representations have suffered from over-smoothing after only 40 aggregations. As for Wiki, the value of k selected by only intra-cluster distance is 12, which produces worse results compared with our method. The above analysis shows that the method of determining k value by only intra-cluster distance is incomplete. By comparison, the proposed strategy which takes into account both inter-cluster distance and intra-cluster distance is more robust and efficient.

5.8. Ablation study

In order to analyze the influence of different components in our model, we conduct ablation study on all datasets. Specifically, we run all cases 10 times and show the average clustering results in Table 3. As can be seen, case 1 obtains the worst results on most datasets, because k-means only utilizes the high dimensional node attributes information for clustering. Comparing cases 2 and 3 with 1, we can find that the aggregation module improves the clustering performance a lot, except for case 2 on Pubmed. The reason for the above situation is that the node features become smooth through the aggregation module, but the central node receives more noise from adjacent vertexes on Pubmed in case 2. In addition, cases 4, 5, and 6 achieve better performance than cases 1, 2, and 3, respectively, which indicates that the similarity preserve module effectively retains the essential information in node features when reducing dimension. Next, comparing case 3 with 2 and case 6 with 5 respectively, one important phenomenon is that our attention-based aggregation module brings about a boost in attributed graph clustering performance. On the whole, the proposed model (case 6) nearly achieves the best performance of all cases.

5.9. Visualization

We visualize the node representations of Cora among different stages of our model in Fig. 7. In the initial stage, most nodes from the different classes cannot be distinguished. After k times of aggregation, node representations become smooth and most of

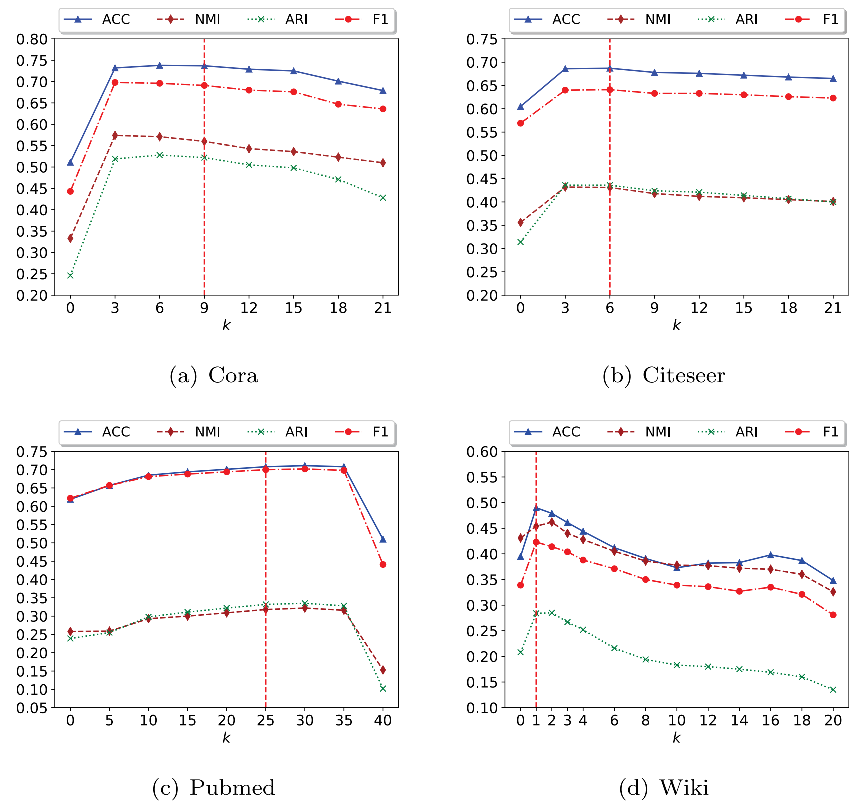


Fig. 6. Clustering results among different values of k.

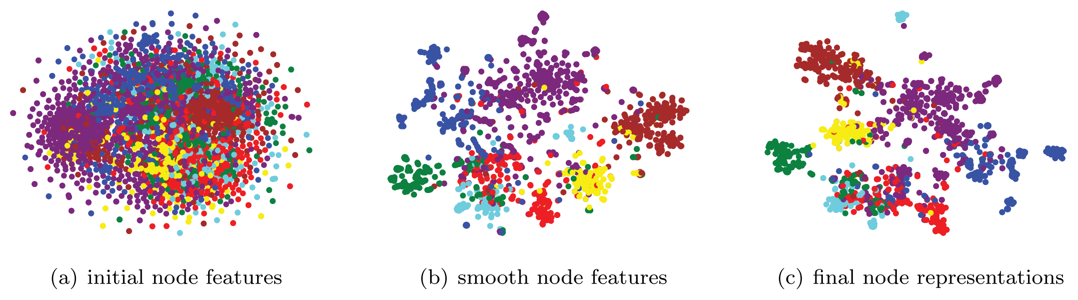


Fig. 7. Visualization of the node representations at different stages of DLGAMC on Cora.

the vertexes from the same class are close together. Finally, the nodes in each cluster become more compact after applying the similarity preserve module on the smooth features.

5.10. The extension of DLGAMC

In our model, we perform k-means on the learned embedding to obtain clustering results. As we know, k-means is a traditional clustering algorithm, so we conduct extended experiments to verify the effectiveness of our model with another advanced clustering method.

We find that many recent graph clustering models based on deep embedding clustering (DEC) [44] have achieved very great performance, e.g., DAEGC [2], SDCN [15], CaEGCN [45], and AGCN [46]. Different from other traditional algorithms such as k-means, DEC is developed for clustering with deep learning, and its core idea is to help the marginal nodes close to the correct cluster by taking the high confidential samples as supervision. We replace k-means in the proposed model with DEC and extend our model as DLGAMC-DEC. The framework of DLGAMC-DEC is shown in Fig. 8.

We run DLGAMC-DEC 10 times (In the stage of DEC, run 100 epochs each time) and show the average clustering results in Fig. 9. Note that we set the degree of freedom in DEC as 1, which is the same as the original paper. As can be seen, DLGAMC-DEC achieves the similar performance as DLGAMC on Cora, Citeseer, and Pubmed and outperforms DLGAMC on Wiki. The above observation shows that the node embeddings learned by our model are friendly to both DEC and k-means, and DEC brings a certain improvement on Wiki. In a word, besides k-means, our model has the potential to be extended to other clustering algorithms.

6. Conclusion

In this paper, we propose a deep linear graph attention model for attributed graph clustering, which is simple but efficient. Specifically, we exploit an attention-based aggregation module to obtain the smooth features, where the number of times for aggregation is determined by our proposed strategy which considers both inter-cluster distance and intra-cluster distance. Then, we design a similarity preserve module on the smooth features to learn node representations. Finally, we get the clustering results

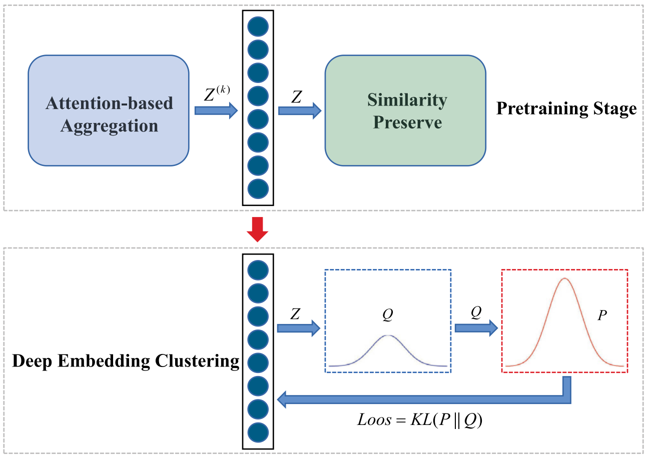
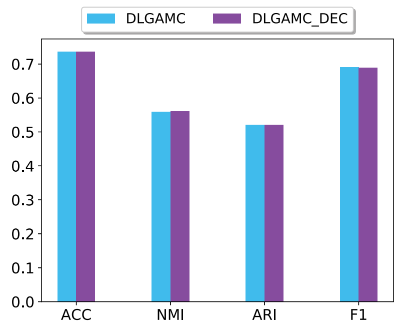


Fig. 8. The framework of extended model DLGAMC-DEC. It can be divided into two stages: pretraining and clustering. In the first stage, we execute attention-based aggregation to obtain smooth representations and then pretrain the single-layer neural network through the similarity preserve module, which is the same as DLGAMC. In the second stage, the above network is further trained by minimizing the KL divergence loss between soft assignments Q and auxiliary distribution P. According to soft assignments Q, the final cluster assigned to each node is the one with maximum probability.

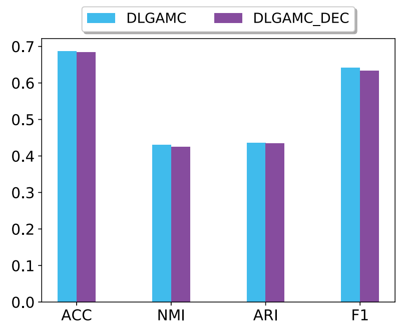
Table 3
Ablation study ('✔' means the usage of this component).

	Aggregation		Similarity preserve	*k*-means	Cora				Citeseer				Pubmed				Wiki			
	Filter[a]	Attention[b]			ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
1	–	–	–	✔	0.354	0.176	0.096	0.311	0.404	0.181	0.138	0.375	0.595	0.314	0.280	0.581	0.339	0.316	0.074	0.245
2	✔	–	–	✔	0.603	0.463	0.376	0.495	0.611	0.392	0.365	0.512	0.575	0.236	0.206	0.549	0.352	0.344	0.109	0.299
3	–	✔	–	✔	0.629	0.508	0.394	0.522	0.624	0.399	0.382	0.543	0.705	**0.321**	0.325	0.695	0.418	0.390	0.200	0.338
4	–	–	✔	✔	0.511	0.333	0.246	0.443	0.605	0.356	0.314	0.569	0.619	0.258	0.239	0.622	0.395	0.431	0.208	0.339
5	✔	–	✔	✔	0.724	0.547	0.502	0.672	0.680	0.418	0.425	0.637	0.678	0.285	0.288	0.674	0.440	0.438	0.205	0.393
6	–	✔	✔	✔	**0.737**	**0.560**	**0.522**	**0.691**	**0.687**	**0.431**	**0.436**	**0.641**	**0.708**	0.318	**0.332**	**0.700**	**0.490**	**0.454**	**0.284**	**0.423**

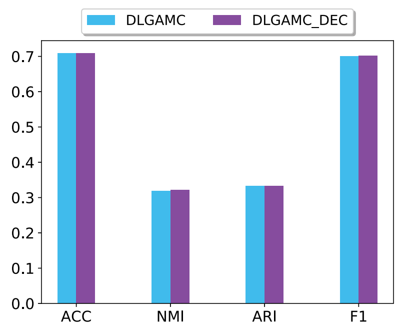
[a]The filter in SGC.
[b]Attention-based aggregation in our model.



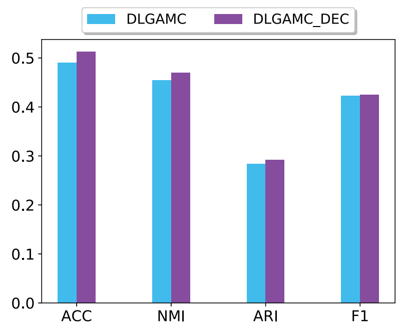
(a) Cora



(b) Citeseer



(c) Pubmed



(d) Wiki

Fig. 9. The comparison between DLGAMC and DLGAMC-DEC.

by performing *k*-means on the learned representations. Extensive experiments on several widely used datasets have verified the effectiveness of our method and show that our simple linear model can still achieve great clustering performance in attributed graphs. In future work, we will try to improve the similarity preserve module from the perspective of node pairs selection. Besides that, our linear graph attention network can be applied to inductive learning, which is not further discussed in this article and will be explored in the later work.

CRediT authorship contribution statement

Huifa Liao: Performed the data analyses, Wrote the manuscript, Revised the manuscript. **Jie Hu:** Contributed significantly to analysis and manuscript preparation, Helped revise the manuscript. **Tianrui Li:** Helped perform the analysis with constructive discussions. **Shengdong Du:** Helped revise the manuscript. **Bo Peng:** Helped revise the manuscript.

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.

Acknowledgments

This work is supported by the National Key R&D Program of China (No. 2020AAA0105101), the National Natural Science Foundation of China (Nos. 62176221, 61976182, 61772435), and Sichuan Key R&D project (2020YFG0035, 2021YFG0312).

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017.
[2] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2019, pp. 3670–3676.
[3] Z. Wang, C. Chen, W. Li, Predictive network representation learning for link prediction, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 969–972.
[4] R. Hu, S. Pan, G. Long, X. Zhu, J. Jiang, C. Zhang, Co-clustering enterprise social networks, in: Proceedings of the International Joint Conference on Neural Networks, 2016, pp. 107–114.
[5] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[6] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (2) (1982) 129–137.
[7] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[8] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the ACM International Conference on Information and Knowledge Management, 2015, pp. 891–900.
[9] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2014, pp. 1293–1299.
[10] T. Yang, R. Jin, Y. Chi, S. Zhu, Combining link and content for community detection: A discriminative approach, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 927–936.
[11] D. He, Z. Feng, D. Jin, X. Wang, W. Zhang, Joint identification of network communities and semantics via integrative modeling of network topologies and node contents, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2017, pp. 116–124.
[12] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Chang, Network representation learning with rich text information, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[13] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: NeurIPS Workshop on Bayesian Deep Learning, 2016.
[14] A. Salehi, H. Davulcu, Graph attention auto-encoders, in: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, 2020, pp. 989–996.
[15] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: Proceedings of the Web Conference, 2020, pp. 1400–1410.
[16] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE Trans. Cybern. 50 (6) (2020) 2475–2487.
[17] B. Hui, P. Zhu, Q. Hu, Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 4215–4222.
[18] F. Wu, T. Zhang, A. Souza, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: Proceedings of the International Conference on Machine Learning, 2019, pp. 6861–6871.
[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proceedings of the International Conference on Learning Representations, 2018.
[20] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 3538–3545.
[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.
[22] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 3438–3445.
[23] X. Zhang, H. Liu, Q. Li, X.M. Wu, Attributed graph clustering via adaptive graph convolution, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2019, pp. 4327–4333.
[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
[25] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Proceedings of the International Conference on Learning Representations, 2013.
[26] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[27] L.F.R. Ribeiro, P.H.P. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 385–394.
[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the International Conference on World Wide Web, 2015, pp. 1067–1077.
[29] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2017, pp. 203–209.
[30] S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, pp. 1145–1152.
[31] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.
[32] J. Chang, D. Blei, Relational topic models for document networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2009, pp. 81–88.
[33] Y. Sun, J. Han, J. Gao, Y. Yu, iTopicModel: Information network-integrated topic modeling, in: Proceedings of the IEEE International Conference on Data Mining, 2009, pp. 493–502.
[34] Z. Xu, Y. Ke, Y. Wang, H. Cheng, J. Cheng, A model-based approach to attributed graph clustering, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2012, pp. 505–516.
[35] Y. Ruan, D. Fuhry, S. Parthasarathy, Efficient community detection in large networks using content and links, in: Proceedings of the International Conference on World Wide Web, 2013, pp. 1089–1098.
[36] J. Yang, J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: Proceedings of the IEEE International Conference on Data Mining, 2013, pp. 1151–1156.
[37] L. Liu, L. Xu, Z. Wangy, E. Chen, Community detection based on structure and content: A content propagation perspective, in: Proceedings of the IEEE International Conference on Data Mining, 2015, pp. 271–280.

[38] X. Wang, D. Jin, X. Cao, L. Yang, W. Zhang, Semantic community identification in large attribute networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, pp. 265–271.
[39] Y. Li, C. Sha, X. Huang, Y. Zhang, Community detection in attributed graphs: An embedding approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 338–345.
[40] Z. Li, X. Wang, J. Li, Q. Zhang, Deep attributed network representation learning of complex coupling and interaction, Knowl.-Based Syst. 212 (2021) 106618.
[41] C. Wang, S. Pan, G. Long, X. Zhu, J. Jiang, MGAE: Marginalized graph autoencoder for graph clustering, in: Proceedings of the ACM International Conference on Information and Knowledge Management, 2017, pp. 889–898.
[42] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2018, pp. 2609–2615.
[43] Z. Zang, S. Li, D. Wu, J. Guo, Y. Xu, S.Z. Li, Unsupervised deep manifold attributed graph embedding, 2021, arXiv preprint arXiv:2104.13048.
[44] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 478–487.
[45] G. Huo, Y. Zhang, J. Gao, B. Wang, Y. Hu, B. Yin, CaEGCN: Cross-attention fusion based enhanced graph convolutional network for clustering, IEEE Trans. Knowl. Data Eng. (2021).
[46] Z. Peng, H. Liu, Y. Jia, J. Hou, Attention-driven graph clustering network, in: Proceedings of the ACM International Conference on Multimedia, 2021, pp. 935–943.
[47] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[48] H.W. Kuhn, The Hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1–2) (1955) 83–97.
[49] P.A. Estévez, M. Tesmer, C.A. Perez, J.M. Zurada, Normalized mutual information feature selection, IEEE Trans. Neural Netw. 20 (2) (2009) 189–201.
[50] K.Y. Yeung, W.L. Ruzzo, Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data, Bioinformatics 17 (9) (2001) 763–774.
[51] C. Goutte, E. Gaussier, A probabilistic interpretation of precision, recall and F-score, with implication for evaluation, in: Proceedings of the European Conference on Information Retrieval, 2005, pp. 345–359.
[52] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[53] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.
[54] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008) 2579–2605.