Knowledge and Information Systems (2024) 66:4549–4572
https://doi.org/10.1007/s10115-024-02097-4

REGULAR PAPER

Deep graph clustering via mutual information maximization and mixture model

Maedeh Ahmadi[1] · Mehran Safayani[1] · Abdolreza Mirzaei[1]

Received: 4 December 2022 / Revised: 12 December 2023 / Accepted: 6 March 2024 /
Published online: 10 April 2024

Abstract
Attributed graph clustering or community detection which learns to cluster the nodes of a graph is a challenging task in graph analysis. Recently contrastive learning has shown significant results in various unsupervised graph learning tasks. In spite of the success of graph contrastive learning methods in self-supervised graph learning, using them for graph clustering is not well explored. In this paper, we introduce a contrastive learning framework for learning clustering-friendly node embedding. We propose Gaussian mixture information maximization which utilizes a mutual information maximization approach for node embedding. Meanwhile, in order to have a clustering-friendly embedding space, it imposes a mixture of Gaussians distribution on this space. The parameters of the contrastive node embedding model and the mixture distribution are optimized jointly in a unified framework. Experiments show that our clustering-directed embedding space can enhance clustering performance in comparison with the case where community structure of the graph is ignored during node representation learning. The results on real-world datasets demonstrate the effectiveness of our method in community detection.

Keywords Graph neural network · Graph representation learning · Graph clustering · Contrastive learning

1 Introduction

Graphs provide a way of representing a wide-variety of complex data in real-world systems. Several graph analysis approaches have emerged to extract useful information hidden in graphs. Community detection as an essential tool for graph analysis has been applied to

✉ Mehran Safayani
safayani@iut.ac.ir

Maedeh Ahmadi
m.ahmadi@ec.iut.ac.ir

Abdolreza Mirzaei
mirzaei@iut.ac.ir

[1] Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

many real-world problems like social networks [1], citation networks [2], brain networks [3] and protein–protein interaction (PPI) network [4]. Many community detection algorithms have been proposed, from shallow approaches [5, 6] to deep ones [7, 8]. Several recent deep methods utilize graph convolutional network (GCN) [9] to extract features from graphs [10, 11]. Many of these methods rely on local graph information (e.g., adjacency matrix reconstruction) which is not appropriate for graph clustering [8].

Recently, contrastive approaches have demonstrated significant results in graph analysis tasks. These self-supervised methods mainly discriminate between positive and negative sample pairs from node or graph level representations [12–15]. Despite the high performance of contrastive methods in node representation learning, they have not received sufficient attention for community detection. Several contrastive approaches for node clustering have been proposed in the literature, but most of them isolate the node embedding step from the clustering task [16]. When community structure of the graph is ignored during node representation learning process, the resulted embedding space is suboptimal for the clustering task because the representation learning step is unaware of the downstream clustering task and is performed independent of it. Therefore, it is beneficial to define a clustering-oriented contrastive objective function to achieve better clustering performance.

To address this, we propose to learn a novel clustering-friendly node embedding framework which utilizes a contrastive method for node representation learning, and meanwhile, it considers the community structure of the graph for optimizing node representations. Therefore, the proposed method not only employs the great potential of contrastive node embedding, but also explores the cluster information of the embedding space. As the embedding method, we utilize a contrastive approach which relies on mutual information maximization to learn representations. In order to have a clustering-friendly node embedding space, our approach is to impose a Gaussian mixture distribution on the representation space. Combining Gaussian mixture embedding and deep models is not a straight-forward task. Several approaches have been introduced to do so. Jiang et al. [17], Uğur et al. [18] use variational approaches to have a mixture model in their latent space. Makhzani et al. [19] utilize an adversarial training procedure to make the latent space of their model follow a mixture of Gaussians distribution. Our approach is to assume that the learnt node embedding space follows a mixture of Gaussians (MoG) distribution and learn the parameters of this mixture distribution along with the parameters of the contrastive model in a unified framework by taking iterative single (or limited) steps of expectation–maximization (EM) and gradient descent. Moreover, since many message passing algorithms are restricted to local messages, it is beneficial to employ a method which goes beyond direct neighbors to capture higher order information in the graph. To do so, we employ graph diffusion convolution (GDC) [20] which may help with the task of clustering by providing a global view of the graph. However, since GDC does not perform well for some complicated graphs, we employ a clustering quality measure, the modularity [21], to decide whether to utilize GDC for clustering a given graph or not. Our code is available on https://github.com/MaedehAhmadi/GMIM.

Contributions of our method are summarized as follows:

1. We propose a clustering-oriented contrastive learning-based method, Gaussian mixture information maximization (GMIM), for learning node embedding. Different from other simple contrastive methods, which ignore the community structure of the graph during node embedding, GMIM learns a clustering-friendly node representation which cares about the downstream clustering task.
2. We utilize graph diffusion to benefit from the global view of the graph in the clustering task in cases it is beneficial. Diffusion makes it possible to surpass the limited information

of direct neighbors in message passing process and provides the proceeding contrastive learning algorithm with the global view of the graph.
3. Extensive experiments on six real-world datasets demonstrates the effectiveness of our method in comparison with the state-of-the-art deep graph clustering methods.

The rest of the paper is organized as follows. We review the related work of graph embedding and graph clustering in Sect. 2. Section 3 introduces a detailed description of the proposed method. Experimental results on six real-world datasets are presented in Sect. 4. The conclusions are given in Sect. 5.

2 Related works

2.1 Graph embedding

Recently, approaches based on deep learning have made great progress in many fields of graph learning specially graph embedding. Early deep learning based researches were mostly rely on random walk objectives [22, 23]. These methods take random walks along nodes and utilize neural language models (like SkipGram [24]) for node embedding. They assume that close nodes in the input graph, which co-occur in the same random sequence, should also be close in the embedding space.

Graph neural networks (GNNs) [25–27] have demonstrated strong representation power for attributed graph learning tasks. GNN-based methods follow a message passing mechanism to capture structural information of graph data. For unsupervised graph embedding, graph autoencoder-based methods [7, 9, 10] mainly try to reconstruct adjacency matrix so they impose closeness of fist-order neighbor nodes in the embedding space. Both of random walks and graph autoencoders-based methods over-emphasize the local proximity information [13].

Recently, contrastive approaches have achieved state-of-the-art results in graph data analysis [13, 14, 28]. They contrast samples from a desired distribution and another undesired one. Motivated by the excellent results of contrastive learning in visual representation learning [29, 30], graph contrastive algorithms propose to retain local and global structural information of graphs [13, 14].

2.2 Community detection

Many methods for detecting communities have been proposed. Early methods employ shallow approaches to community detection, mostly focusing on the information of network topology. Non-negative matrix factorization (NMF) [5, 6] and Laplacian eigenmaps [31] are two widely used approaches in this area. Stochastic block model-based methods [32] are also well explored. Modularity maximization is a popular goal to extract communities [33]. To exploit both of content and structural information, several extended algorithms based on topic models [34] and NMF [35, 36] are proposed.

As graph analysis problems and graph data get more complicated, deep learning-based methods have demonstrated great performance in graph analysis tasks including community detection. As baseline methods among deep approaches, applying well-known clustering algorithms on embedding results of GAE and VGAE [9] have better performance than many shallow algorithms. Some works present enhanced graph autoencoder-based methods with boosted results in graph clustering [7, 10, 11].

While some methods perform graph embedding and clustering in two independent stages, some other methods try to combine clustering and graph embedding goals. Wang et al. [8] co-optimize a graph attention-based reconstruction loss and the clustering loss of [37]. Tsitsulin et al. [38] maximize modularity on the embedding space of a GCN. A probabilistic generative model which learns node embedding and community assignment jointly is proposed in [39]. In [40], GCN is integrated with Bernoulli–Poisson probabilistic model [6] for overlapping community detection. Zhang et al. [41] train a graph autoencoder to find an appropriate embedding space for relaxed Kmeans. A variational framework for learning clustering and node embedding is introduced in [42].

There exist some contrastive approaches for graph clustering in the literature. SCGC [16] presents a contrastive approach for node clustering. But unlike our method, it performs node embedding and clustering in two independent steps. Therefore, the second step is totally unaware of the first one. Moreover, since it defines a neighbor-oriented contrastive objective function, it over-emphasizes on the limited local information of direct neighbors. CCGC [43] is a node-by-node contrastive learning method which utilizes the clustering result of each iteration for selecting positive and negative pairs of nodes in the next iteration. It defines a loss function to minimize the similarity of cross-view different high-confidence cluster centers. CONVERT [44] is also a contrastive approach with a label-matching module which aligns the pseudo-labels selected via clustering and semantic labels obtained from applying softmax on node embeddings.

3 Method

3.1 Problem formalization and method overview

We consider community detection in attributed networks in this paper. The input is a graph $G = (V, E, X)$, where $V = (v_1, v_2, \ldots, v_N)$ is the set of N nodes and $E = \{e_{ij}\}$ is the edge set. $X = \{x_1; x_2; \ldots; x_N\}$ are the attribute values where $x_i \in \mathbb{R}^F$ is the feature vector of node v_i. An adjacency matrix $A \in \mathbb{R}^{N \times N}$ encodes the structural connectivity of nodes where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise, $A_{i,j} = 0$.

The purpose of attributed community detection is to divide the nodes into K communities (or clusters) based on the attributes and structural information.

Our proposed method considers the clustering and node embedding tasks in a joint manner. To achieve this goal, we assume that the node embedding space flows a mixture of Gaussians distribution. We learn the parameters of the MoG and the contrastive method jointly. This results in a more cluster-friendly representation space which is more appropriate for Kmeans clustering algorithm to be applied to.

The proposed method includes two main parts: (1) node embedding part which utilizes contrastive learning for extracting embedding vectors of the nodes and (2) clustering part that tries to impose a Gaussian mixture distribution on the learned latent representation. The overall framework of our proposed method is shown in Fig. 1.

3.2 Node embedding

Our proposed framework for clustering-friendly node representation learning is not limited to a specific type of contrastive node embedding approach. contrasting representation vectors can be done in node versus node level [45] or node versus graph level [13, 14]. In this

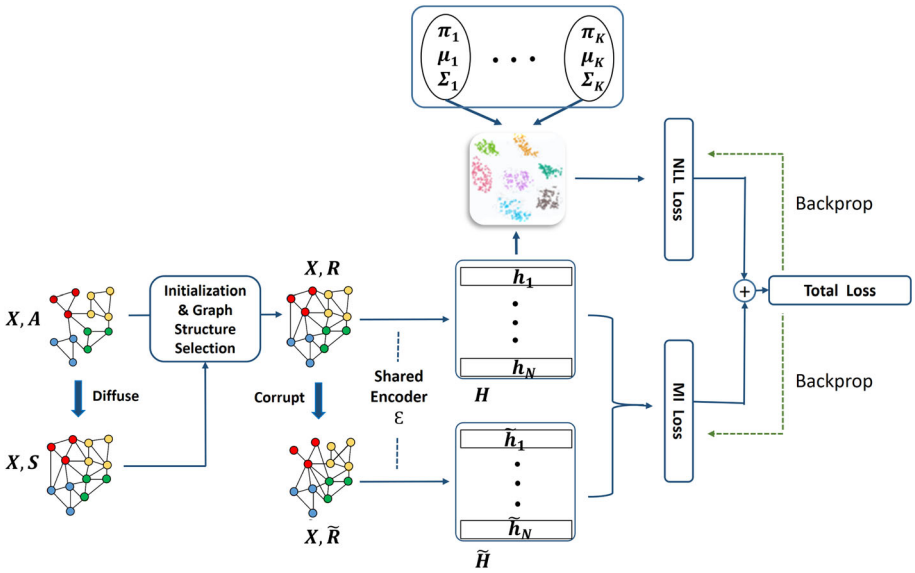


Fig. 1 Our GMIM framework. Given X and A as the input, we select our input graph structure according to the clustering quality of two initialized embedding spaces. The resulted graph and its corrupted version are fed in to a shared encoder. The output features construct information maximization embedding objective (bottom). The clustering module (top) aims to enforce this representation to follow a MoG distribution. The embedding and clustering modules are trained jointly

paper, we use the contrastive framework of [13] for learning node embedding on attributed networks. We maximize the mutual information between node representation vectors and a global graph summary vector. The objective is to train an encoder $\mathcal{E}$ such that $\mathcal{E}(X, A) = H = \{h_1, h_2, \ldots, h_N\} \in \mathbb{R}^{N \times F'}$ represent node representations $h_i \in \mathbb{R}^{F'}$ for each node i. We generate a negative graph $\tilde{G}$ by a corruption function $\tilde{G} = C(G)$ that shuffles the rows of X. The same encoder $\mathcal{E}$ is applied to the positive and negative graphs to obtain H and $\tilde{H}$ representation matrices. Summary vector s is obtained by the readout function $s = \mathcal{R}(H) = \sigma(1/N \sum_{i=1}^{N} h_i)$, with logistic sigmoid nonlinearity σ. Given representation vector h and s, The following discriminator $\mathcal{D}$ distinguishes between representations from positive and negative graphs by assigning higher probabilities to representation vectors that the summary contains them:

$$\mathcal{D}(h, s) = \sigma(h^T W s), \tag{1}$$

where W is a learnable scoring matrix. To maximize the mutual information between h_i and the summery vector s, the following cross-entropy loss is minimized:

$$\begin{aligned} \mathcal{L}_{MI} = -\frac{1}{2N}\Bigg(& \sum_{i=1}^{N} \mathbb{E}_{(X,A)}[\log \mathcal{D}(h_i, s)] \\ & + \sum_{j=1}^{N} \mathbb{E}_{(\tilde{X},\tilde{A})}[\log(1 - \mathcal{D}(\tilde{h}_j, s))]\Bigg). \end{aligned} \tag{2}$$

The encoder is the following single-layer GNC:

$$\mathcal{E}_A(X, A) = \text{PReLU}(\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}X\Phi), \tag{3}$$

where $\hat{A} = A + I_N$ is the adjacency matrix with self-connections and $\widehat{D}_{ii} = \sum_j \hat{A}_{ij}$ is the corresponding degree matrix. Φ is a learnable transformation matrix and PReLU represents parametric rectified linear unit function.

3.3 Graph diffusion

Message passing neural networks pass messages between immediate nodes of the graph. Although they try to aggregate the messages from higher-order neighbors in deep layers, most of them achieve their best performance with 2-layer networks because of over-smoothing phenomenon [46]. Limiting the messages of each layer to one-hop neighbors is restrictive, and some methods try to capture higher-order information in the graph. One of the successful methods in this regard is graph diffusion convolution (GDC) [20]. It replaces the adjacency matrix with a diffusion matrix which is formulated as:

$$S = \sum_{k=0}^{\infty} \theta_k T^k, \tag{4}$$

with generalized transition matrix T and weighting coefficients θ. One popular example of graph diffusion is Personalized PageRank (PPR) [47]. Given adjacency matrix A and related degree matrix $D_{ii} = \sum_j A_{ij}$, (PPR) chooses $T = AD^{-1}$ and $\theta_k = \alpha(1-\alpha)^k$ with teleport probability $\alpha \in [0, 1]$. The closed-form solution for PPR diffusion is as below:

$$S^{\text{PPR}} = \alpha \left(I_n - (1-\alpha)D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)^{-1}. \tag{5}$$

This diffusion matrix provides global view of a graph, acts as a low-pass filter and smooths out the neighborhood over the graph [20]. GDC can be integrated with any kind of graph-based model. We can utilize it as the input of our model instead of adjacency matrix. But for complicated datasets, GDC may not perform well for clustering. In order to make decision about using diffusion or adjacency matrices as our input graph structure, we utilize the modularity measure [21]. This score measures the clustering quality without regarding label information. To do so, we train two encoders using (2) as the loss function. Adjacency matrix is fed to one encoder and diffusion matrix is fed to the other. We then apply Kmeans clustering on the resulted representations from two encoders. The higher value of modularity indicates the better clustering quality. In GMIM framework of Fig. 1, we utilize the winner matrix and the winner encoder as the matrix R and the encoder $\mathcal{E}$, respectively. Also, the selected trained encoder is utilized for initialization as will be stated in Sect. 3.6. Note that we use the following GCN encoder in case of using diffusion matrix as the input structure:

$$\mathcal{E}_{PPR}(X, S^{PPR}) = \text{PReLU}(S^{PPR}X\Phi), \tag{6}$$

where Φ is a learnable transformation matrix.

3.4 Gaussian mixture modeling for community detection

Assume we have calculated a node embedding h_i for every node v_i of the graph by a node embedding model with parameters Ψ. We consider each node is generated from a multivariate Gaussian distribution. Then, the likelihood for all the nodes of the graph is a Gaussian mixture distribution:

$$p(V) = \prod_{i=1}^{|V|} \sum_{k=1}^{K} p(c_i = k) p(v_i | c_i = k; \Psi, \mu_k, \Sigma_k), \tag{7}$$

here c_i denotes the soft community assignment of node i and $p(c_i = k)$ indicates the probability of node i being assigned to community k. $p(v_i|c_i = k; \Psi, \mu_k, \Sigma_k)$ is a multivariate Gaussian distribution as follows:

$$p(v_i|c_i = k; \Psi, \mu_k, \Sigma_k) = N(h_i|\mu_k, \Sigma_k). \tag{8}$$

For simplicity of notations, we denote $p(c_i = k)$ as π_k where $\sum_{k=1}^{K} \pi_k = 1$. So the parameters of the Gaussian mixture are $\Theta = \{\Pi = \{\pi_k\}, M = \{\mu_k\}$ and $\sum = \{\Sigma_k\}\}$ for $i = 1, \ldots, |V|$ and $k = 1, \ldots, K$. We assume covariance matrices Σ_k are diagonal.

3.5 Clustering-friendly node embedding

We propose a clustering-promoting objective which outputs a latent space that is suitable for clustering. We assume that the learnt latent space follows a MoG distribution. Our defined objective function has two parts: embedding and clustering. The embedding part utilizes the self-learning objective of $\mathcal{L}_{MI}$ for node representation learning and the clustering module tries to enforce this representation to follow a MoG distribution. The later goal is achieved by minimizing the negative log-likelihood (NLL) under MoG distribution:

$$L_{NLL} = -\sum_{i=1}^{|V|} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(h_i|\mu_k, \Sigma_k). \tag{9}$$

Our total loss function is defined as:

$$\mathcal{L} = \omega \mathcal{L}_{MI} + \beta \mathcal{L}_{NLL}, \tag{10}$$

where $\mathcal{L}_{MI}$ and $\mathcal{L}_{NLL}$ are the mutual information loss and the negative log-likelihood (NLL), respectively. The weighs ω and β balance between two terms of the objective function. After optimizing our objective, we have a Kmeans-friendly latent space on which we apply Kmeans algorithm to obtain the final clusters of nodes.

3.6 Inference

The total loss function of (10) consists of two sets of parameters: node embedding parameters (Ψ) and MoG parameters $\Theta = \{\Pi, M$ and $\Sigma\}$. To optimize these parameters, we use an iterative approach by fixing one set and optimizing the other. We initialize the Ψ parameters by training the model using (2) as the loss function. To initialize MoG parameters, we apply Kmeans algorithm on the achieved embedding from Ψ initialization. We initialize (Π, M, Σ) using the hard assignment results of Kmeans algorithm. The details of this iterative approach are described below.

Fixing Ψ Parameters and Optimizing $\Theta = \{\Pi, M, \Sigma\}$

Fixing deep network parameters, we use expectation–maximization algorithm [48] to optimize (Π, M, Σ). The following equations are used iteratively to update these parameters:

$$\pi_k = \frac{N_k}{|V|}, \tag{11}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{|V|} \mathcal{V}_{ik} h_i, \tag{12}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{|V|} \mathcal{V}_{ik} (h_i - \mu_k)(h_i - \mu_k)^T, \tag{13}$$

where

$$\mathcal{V}_{ik} = \frac{\pi_k \mathcal{N}(h_i | \mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{i,k'} \mathcal{N}\left(h_i | \mu_{k'}, \Sigma_{k'}\right)}, \tag{14}$$

and

$$N_k = \sum_{i=1}^{|V|} \mathcal{V}_{ik} \quad 1 \leq k \leq K. \tag{15}$$

More precisely, we update $\mathcal{V}_{ik}$ in E-step and (Π, M, Σ) in the M-step of the EM algorithm. Note that we perform a limited steps of E and M in each iteration not until the convergence of EM. More details about the derivation of updating formulas are provided in Appendix A.

Fixing $\Theta = \{\Pi, M, \Sigma\}$ and Updating Ψ Parameters

Fixing MoG parameters, we optimize the total loss function of (10) with respect to Ψ parameters using gradient descent (GD):

$$\Psi = \Psi - \eta \left(\frac{\omega \partial \mathcal{L}_{MI}(\Psi)}{\partial \Psi} + \frac{\beta \partial \mathcal{L}_{NLL}(\Psi, \Theta)}{\partial \Psi} \right), \tag{16}$$

where η is the learning rate. Parameters Ψ are updated via PyTorch auto-grad. Green arrows in Fig. 1 denote the backpropagation process. Ψ consists of the learnable scoring matrix W of (1), encoder parameters Φ and PReLU parameters of (3). Our proposed method is summarized in Algorithm 1.

3.7 Computational complexity

In this section, we analyze the computational complexity of GMIM with N nodes, $|E|$ edges, K clusters, attribute and embedding dimensions of F and F' and T_i and T epochs for initialization and main optimization, respectively. Our algorithm consists of three main components: computing diffusion, initialization and Gaussian mixture modeling. We calculate the diffusion using Eq. 5 whose time complexity is $\mathcal{O}(N^3)$. In the initialization phase, we optimize $\mathcal{L}_{MI}$ for T_i epochs. In each epoch, encoder and discriminator constitute the main parts of $\mathcal{L}_{MI}$. The encoder can be implemented through sparse or dense multiplication with time complexities of $\mathcal{O}(NFF' + |E|F')$ or $\mathcal{O}(N^2F' + NFF')$, respectively. The complexity of computing the discriminator is $\mathcal{O}(NF'^2)$. The time complexity of calculating $\mathcal{L}_{MI}$ is the sum of complexities of the encoder and discriminator. Gaussian mixture modeling phase consists of T epochs. In each epoch, $\mathcal{L}_{MI}$ should be calculated with the complexity mentioned above and the parameters of MoG are updated once with the complexity of $\mathcal{O}(NKF')$. Since K is usually much smaller than F, we can ignore it and since we set T_i and T at the same order, we assume the same value of T for both.

To sum up, by combining these three components, the overall time complexity of our model is $\mathcal{O}(N^3 + NFF'T + |E|F'T)$ for sparse multiplication implementation and $\mathcal{O}(N^3 + N^2F'T + NFF'T)$ for dense one. It is worth noting that we can utilize an approximation (using Andersen algorithm [49]) with a linear runtime $\mathcal{O}(N)$ for calculating diffusion. In this

Algorithm 1 Gaussian Mixture Information Maximization.

Require: $G = (V, E, X)$, number of clusters K, weight ω, PPR parameter α and hidden dimension.
Ensure: Node embedding H, final community assignments.
1: initialize $H_A = \mathcal{E}_A(X, A)$ by optimizing (2)
2: Calculate S^{PPR} by (5)
3: Replace A by S^{PPR} in (2)
4: Initialize $H_{PPR} = \mathcal{E}_{PPR}(X, S^{PPR})$ by optimizing (2)
5: Apply Kmeans on H_A and H_{PPR}
6: **If** $Modularity_{Kmeans}(H_{PPR}) \geq Modularity_{Kmeans}(H_A)$ **then**
7: $R = S^{PPR}, \mathcal{E} = \mathcal{E}_{PPR}$
8: **else**
9: $R = A, \mathcal{E} = \mathcal{E}_A$
10: Initialize MoG parameters by the previously applied Kmeans on H
11: **for** $t = 0$ to T **do**
12: **for** $t_1 = 0$ to T_1 **do**
13: Update Π, M, Σ by equations (11), (12) and (13)
14: **end for**
15: **for** $t_2 = 0$ to T_2 **do**
16: Calculate mutual information loss by (2) using R and $\mathcal{E}$
17: Calculate negative log-likelihood loss by (9)
18: Update Ψ parameters by GD on (10)
19: **end for**
20: **end for**
21: Get the final community assignments by applying Kmeans on H.

case, the time complexity will be reduced to $\mathcal{O}(NFF'T + |E|F'T)$. We leave this task for our future work.

4 Experiments

4.1 Benchmark datasets

We conduct attributed graph community detection experiments on six standard widely used network datasets (Cora [50], PubMed [50], Wiki [51], ACM [52], Flickr [53] and Coauthor-Phys [54]). Cora and PubMed are two citation networks. Their nodes represent papers and edges correspond to citations. Wiki is a webpage network dataset in which nodes and edges are related to webpages and links between them, respectively. In both cases nodes are represented by bag-of-words vestors. Features of Cora are binary vectors while PubMed and Wiki are represented by tf-idf weights. ACM is a paper network in which nodes represent papers and two papers are connected by an edge if they are written by the same author. Node features representing papers are bag-of-words of keywords. Flickr is a social network in which users play as nodes and edges indicate friendship connection between users. The labels of nodes are user interest groups. Features of each node is a list of tags specified by the users to indicate their interests. Coauthor-Phys is a co-authorship network in which nodes are authors. Two nodes are connected if their corresponding authors have co-authored a paper. Node features are defined based on the set of keywords of author's papers. Class labels correspond to the field of research. Table 1 summarizes the detailed statistics of datasets.

Table 1 Datasets statistics

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2708	5429	1433	7
PubMed	19, 717	44, 338	500	3
Wiki	2405	17, 981	4973	17
Acm	3025	13, 128	1870	3
Flickr	7575	239, 738	12, 047	9
Coauthor-Phys	34, 493	247, 962	8415	5

4.2 Compared methods

We compare GMIM with the following methods. These approaches are categorized into three groups:

1. Methods which use node features only: Kmeans and spectral clustering [55] are two common clustering methods. Spectral-F is a spectral clustering method which considers the cosine similarity between node features as the similarity matrix.
2. Methods which use graph structure only: Spectral-G considers the adjacency matrix as the similarity matrix. DeepWalk [22] generates random paths along a graph and use them to train SkipGram language model to learn node embedding. GraphEncoder [56] trains a stacked sparse autoencoder to obtain node embedding. DNGR [57] uses stacked denoising autoencoders to learn each node representation. Kmeans in applied to the learnt latent space of the three later methods. vGraph [39] is a probabilistic generative model which performs graph clustering and node embedding jointly.
3. Methods which use both node features and graph structure: TADW [58] adds node features to DeepWalk framework. GAE and VGAE [9] integrate (variational) autoencoder and graph neural networks for node embedding. ARGA and ARVGA [7] use an adversarial training scheme to impose a prior distribution on latent space of GAE and VGAE. DGVAE [59] presents a graph variational generative model which uses the Dirichlet distributions as priors on the latent variables. AGC [11] designs a high-order graph convolution to take smooth node features for enhancing clustering results. CommDGI [60] incorporates contrastive learning to learn cluster assignment of the nodes. DAEGC [8] optimizes graph reconstruction loss and a clustering loss jointly. SENet [61] uses a spectral clustering loss to learn node embeddings. GC-VGE [42] introduces a joint framework for clustering and representation learning by utilizing a variational graph embedding mechanism. DBGAN [62] introduces an adversarial framework to learn node embeddings. SCGC [16] is a contrastive method for node clustering which perform node embedding and clustering tasks in two isolated steps. it defines an adjacency-oriented loss function to contrast between views. CCGC [43] proposes a cluster-guided contrastive learning method which uses high-confidence clustering information for selecting discriminative positive and reliable negative samples for contrastive learning. CONVERT [44] is a contrastive approach which guides the node embedding procedure via matching the pseudo-labels obtained from clustering and the semantic labels.

4.3 Evaluations metrics and experimental settings

We report three evaluation metrics to measure the performance of graph clustering: clustering accuracy (ACC), normalized mutual information (NMI), adjusted rand index (ARI). The higher values of all these metrics indicates the better results. We run our algorithm 10 times on each dataset and report the average and standard deviation of the obtained metrics.

For the encoder, we set the size of hidden dimension to 512 for Cora, Wiki and ACM and 256 for Flickr and PubMed and Coauthor-Phys. The weight ω is set to Cora:25,000, Wiki:15,000, PubMed:1000, ACM:15,000, Coauthor-Phys:10 and Flickr:2000, to balance two terms of objective function. At the start of training we set β to zero and as training progresses, we gradually increase it to reach one. We use the Adam GD optimizer with learning rate of 0.001 in both initialization and training phases for all datasets except the ACM dataset for which we use learning rate of 0.0001 in the training phase. T_1 and T_2 are set to one for all datasets. We train the model for 200 epochs on Cora, Wiki, ACM and Coauthor-Phys, 400 epochs on Flickr and 1000 epochs on PubMed.

We set $\alpha = 0.2$ for PPR diffusion on Cora, PubMed and ACM. We have not used diffusion for Wiki, Flickr and Coauthor-Phys since the decision process specifies the adjacency matrix as the appropriate input structure for these datasets.

4.4 Experimental results

Our experimental results are summarized in Tables 2, 3, 4, 5 and 6. F, G and F&G indicate the methods which use only node features, graph structure or both of features and structure information, respectively. Boldface indicates the best metric value in each column. According to these tables, we obtain the following observations:

1. Methods using both feature and structure generally outperform the methods using only one source of information. This indicates the importance of these information for the clustering task.
2. Our method significantly outperforms classic GNN-based methods GAE, VGAE, ARGA and ARVGA. These are two-stage methods perform node embedding and clustering stage independently. In addition, they basically try to reconstruct adjacency matrix.
3. Some methods (like SENet, CCGC and Convert) exploit clustering information of nodes in node embedding. Also, some competitors (including DAEGC, DGVAE, GC-CGE and CommDGI) present unified frameworks for clustering and representation learning. But, overall, they have inferior results compared to ours. The reason of this matter is different for various methods. For instance, DAEGC, DGVAE and GC-CGE mainly rely on adjacency matrix reconstruction which focuses too much on local proximity information and is not appropriate for the graph clustering goal.
4. Compared with some recent contrastive based methods like SCGC, CONVERT and CCGC, our method achieves significant performance improvements in all cases except ACC on Wiki which is comparable to GMIM. This confirms the positive effect of our clustering-oriented contrastive loss function.
5. GMIM consistently surpasses all of its competitors w.r.t to all metrics on Cora and ACM datasets. On other cases, it can be seen that in few cases that a competitor method has a higher performance than GMIM w.r.t a specific metric on one dataset, it is consistently outperformed by GMIM w.r.t. other metrics on that dataset and also, w.r.t. all metrics on all other datasets. To be more precise, *On PubMed*: GMIM absolutely outperforms all methods w.r.t. all metrics, excluding CommDGI which has higher NMI than ours.

Table 2 Clustering results on Cora dataset

Method	Info	ACC	NMI	ARI
Kmeans	F	49.2	32.1	23.0
Spectral-F	F	34.7	14.7	7.1
Spectral-G	G	31.46	9.69	0.35
DeepWalk	G	56.20	39.87	32.18
GraphEncoder	G	32.5	10.9	0.6
DNGR	G	44.39	33.31	15.86
vGraph	G	28.7	34.5	31.2
TADW	F&G	55.00	36.59	26.40
GAE	F&G	60.34	44.85	36.73
VGAE	F&G	63.56	47.45	39.42
ARGA	F&G	60.84	42.21	36.88
ARVGA	F&G	62.83	45.93	38.00
DGVAE	F&G	64.42	47.64	38.42
AGC	F&G	68.92	53.68	48.6
CommDGI	F&G	69.8	57.9	50.2
DAEGC	F&G	70.4	52.8	49.6
SENet	F&G	71.92	55.08	48.96
GC-VGE	F&G	70.67	53.57	48.15
DBGAN	F&G	74.8	56.0	54.0
CONVERT	F&G	73.99	55.50	50.49
SCGC	F&G	73.31	55.74	51.08
CCGC	F&G	73.88	56.45	52.51
GMIM	F&G	**75.61** $\pm$ **0.86**	**60.19** $\pm$ **0.68**	**54.74** $\pm$ **1.37**

This method is outperformed by GMIM in terms of other metrics on this dataset and also w.r.t. all metrics on other datasets. *On Flickr*: GMIM again consistently surpasses all methods w.r.t. all metrics, except GC-VGE with higher NMI value than ours. This method is outperformed by GMIM in terms of other metrics on this dataset and also w.r.t. all metrics on other datasets. *On Wiki*: GMIM exceeds all of the baselines in terms of NMI (1.93% higher than the best baseline). It also outperforms all of the methods with respect to ACC, excluding SCGC which is slightly better (0.25%) than ours on Wiki dataset in terms of ACC. However, GMIM significantly surpasses SCGC on all datasets (averagely by 7.57%, 8.98% and 9.28% in terms of ACC, NMI and ARI, respectively). In terms of ARI, ACG and DAEGC have higher results than GMIM, while both of these methods are outperformed by GMIM with respect to ACC and NMI. Moreover, GMIM exceeds both of the methods on all other datasets w.r.t. all metrics.

4.5 Ablation study

To evaluate the effectiveness of our unified framework for learning clustering-friendly node embedding, we have performed an ablation study which is shown in Table 7. We train the model using only the mutual information loss ($\mathcal{L}_{MI}$). We then apply Kmeans and Gaussian

Table 3 Clustering results on PubMed dataset

Method	Info	ACC	NMI	ARI
Kmeans	F	55.59	24.34	21.54
Spectral-F	F	60.20	30.90	27.7
Spectral-G	G	37.98	10.30	26.67
DeepWalk	G	64.98	26.44	27.42
GraphEncoder	G	53.1	20.9	18.4
DNGR	G	25.53	20.11	8.29
vGraph	G	26.00	22.40	18.50
TADW	F&G	46.82	9.47	5.78
GAE	F&G	64.43	24.85	23.57
VGAE	F&G	64.67	23.94	23.41
ARGA	F&G	65.07	29.23	26.79
ARVGA	F&G	62.01	26.62	22.46
DGVAE	F&G	67.56	28.72	24.92
AGC	F&G	69.78	31.59	31.19
CommDGI	F&G	69.90	**35.70**	29.2
DAEGC	F&G	67.10	26.60	27.8
SENet	F&G	67.59	30.61	29.66
GC-VGE	F&G	68.18	29.70	29.76
DBGAN	F&G	69.40	32.40	32.7
CONVERT	F&G	67.97	31.54	30.33
SCGC	F&G	41.96	2.88	0.73
CCGC	F&G	62.42	28.30	26.00
GMIM	F&G	**71.11** $\pm$ **0.8**	33.57 $\pm$ 1.3	**34.44** $\pm$ **1.3**

mixture model (GMM) on the learnt embedding, and the average result of 10 runs is reported in Table 7 as MI+Kmeans/GMM. Best results are shown in bold. The fact that GMIM outperforms MI+Kmeans/GMM confirms the effectiveness of jointly optimizing MI and NLL objectives.

4.6 Effect of embedding dimension

In this section, we investigate the influence of the embedding dimension on clustering performance. Figure 2 shows the clustering results of five datasets for different embedding dimensions. It is worth noting that all other hyper-parameters except embedding size are fixed according to Sect. 4.3. We have verified the embedding dimensions of [64, 128, 256, 512] for Cora, Wiki, ACM and Flickr and [16, 32, 64, 128, 256] for PubMed.

According to Fig. 2, the clustering results have small variations using different embedding dimensions. Mostly, the larger embedding dimension results in higher performance.

Table 4 Clustering results on Wiki dataset

Method	Info	ACC	NMI	ARI
Kmeans	F	40.43	42.91	15.03
Spectral-F	F	49.1	46.4	25.4
Spectral-G	G	22.04	18.17	1.46
DeepWalk	G	38.46	32.38	17.03
GraphEncoder	G	20.67	12.07	0.49
DNGR	G	37.54	35.85	17.97
TADW	F&G	30.96	27.13	4.54
GAE	F&G	32.85	29.02	7.80
VGAE	F&G	45.09	46.76	26.34
ARGA	F&G	38.05	34.45	11.22
ARVGA	F&G	38.67	33.88	10.69
DGVAE	F&G	38.25	36.55	15.83
AGC	F&G	47.65	45.28	**34.3**
DAEGC	F&G	48.2	44.8	33.1
GC-VGE	F&G	48.81	47.58	28.40
CONVERT	F&G	46.67	43.64	26.75
SCGC	F&G	**50.42**	46.71	28.24
CCGC	F&G	49.25	43.04	25.63
GMIM	F&G	50.17 ± 1.34	$\mathbf{49.51 \pm 0.4}$	30.80 ± 0.8

Table 5 Clustering results on ACM dataset

Method	Info	ACC	NMI	ARI
Kmeans	F	67.31	32.44	30.60
DeepWalk	G	73.04	35.58	33.97
GAE	F&G	84.52	55.38	59.46
VGAE	F&G	84.13	53.20	57.72
ARGA	F&G	86.29	56.21	63.37
ARVGA	F&G	83.89	51.88	57.77
DAEGC	F&G	86.94	56.18	59.35
CONVERT	F&G	84.64	55.55	60.06
SCGC	F&G	89.81	66.63	72.4
CCGC	F&G	88.75	64.27	69.86
GMIM	F&G	$\mathbf{90.61 \pm 0.1}$	$\mathbf{69.33 \pm 0.4}$	$\mathbf{74.46 \pm 0.4}$

4.7 Influence of hyper-parameter ω

In this section, we focus on the impact of weight ω which balances between two terms of our total loss function. The value of this hyper-parameter is proportional to the ratio of $\mathcal{L}_{NLL}$ to $\mathcal{L}_{MI}$ for each dataset. For obtaining best results, this value is selected such that both of $\mathcal{L}_{MI}$ and $\mathcal{L}_{NLL}$ decrease at the end of training with respect to their initial value.

Figure 3 depicts the clustering results for different values of on five datasets. The size of hidden dimension is set to 128 for PubMed and to the same values as stated in Sect. 4.3 for

Table 6 Clustering results on Flickr dataset

Method	Info	ACC	NMI	ARI
Kmeans	F	11.96	0.66	0.001
Spectral-G	G	11.79	0.43	0.001
DeepWalk	G	28.56	17.45	9.11
TADW	F&G	13.01	1.48	0.09
GAE	F&G	14.20	1.91	1.22
VGAE	F&G	24.74	13.95	7.31
ARGA	F&G	26.64	15.86	8.27
ARVGA	F&G	28.47	17.25	9.59
DGVAE	F&G	19.35	10.02	4.06
GC-VGE	F&G	29.38	**18.27**	10.70
CONVERT	F&G	23.62	12.13	6.53
SCGC	F&G	24.35	13.19	7.08
CCGC	F&G	20.01	9.19	4.08
GMIM	F&G	**30.19** $\pm$ **3.6**	17.36 $\pm$ 2.6	**11.51** $\pm$ **2.6**

Table 7 Ablation study

Method	Cora		Pubmed		Wiki	
	ACC	NMI	ACC	NMI	ACC	NMI
MI + Kmeans	72.15	56.50	67.49	29.15	48.82	49.45
MI + GMM	67.69	53.52	65.25	27.54	47.82	48.81
GMIM	**75.61**	**60.19**	**71.11**	**33.57**	**50.17**	**49.51**

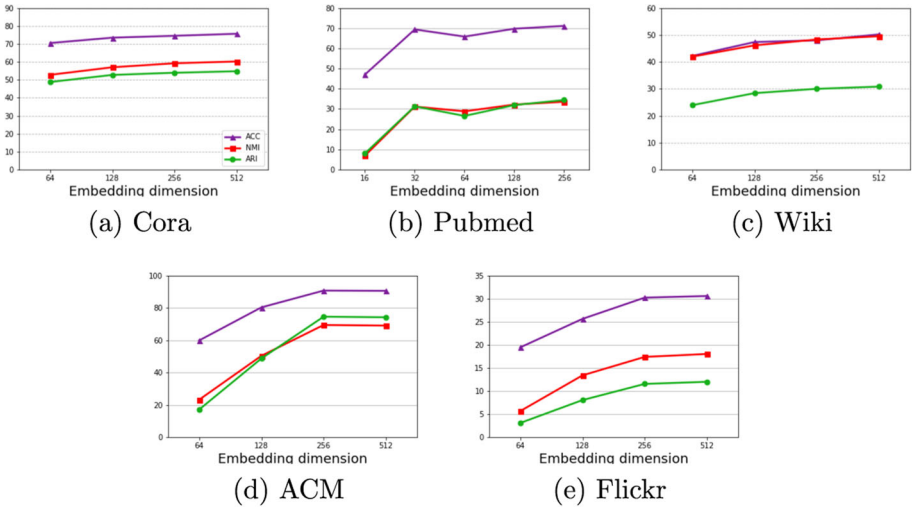


Fig. 2 Effect of different embedding dimensions on node clustering

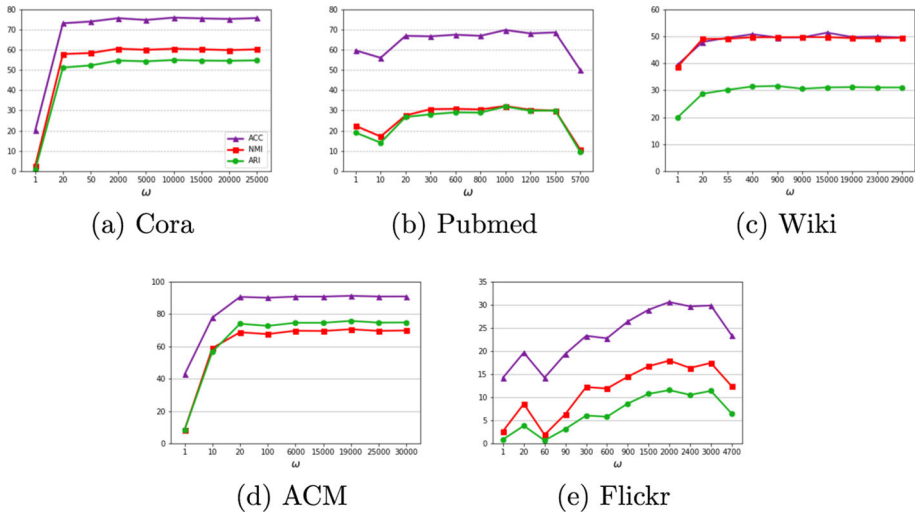


Fig. 3 Effect of different values of ω on node clustering

other datasets. All other hyper-parameters except ω are fixed. As is shown in this figure, the clustering performance is robust within a wide range of ω. However, for very large or small values of ω, one of the terms of the total loss function dominates the other, and consequently the performance degrades.

4.8 Influence of full covariance matrices

As stated in Sect. 3.4, we have used diagonal covariance matrices for Gaussian mixture modeling. Theoretically, even if the elements of feature vectors are not independent, the linear combination of diagonal covariance Gaussians can equally describe the correlations among features modeled by at least one full covariance matrix Gaussian [63]. To verify the effect of using full covariance Gaussians, we examined GMIM with full covariance matrices on Cora dataset. We obtained 75.67%, 60.36% and 54.75% in terms of ACC, NMI and ARI, respectively. Comparing these results with the diagonal covariance case shows that using full covariance Gaussians does not enhance the performance significantly, whereas diagonal covariance matrices result in simplified computations and need less memory.

4.9 Running time analysis

The running time of our model (the required time to train the model and generate the embeddings) on five datasets is shown in Fig. 4. To analyze the efficiency of GMIM, we performed clustering on a larger dataset, namely Coauthor-Phys. The statistics of this dataset is presented in Table 1. We evaluated the clustering results of GMIM and several baseline methods (including GAE, VGAE, ARGA, ARVGA and also SCGC, CCGC and CONVERT as three recent contrastive graph clustering methods) on this dataset. The running times versus clustering accuracy results are depicted in Fig. 5. The running time is in log-scale.

As shown in this figure, our method significantly outperforms all the competitors in terms of accuracy, while the running times of CCGC, SCGC and CONVERT are considerably

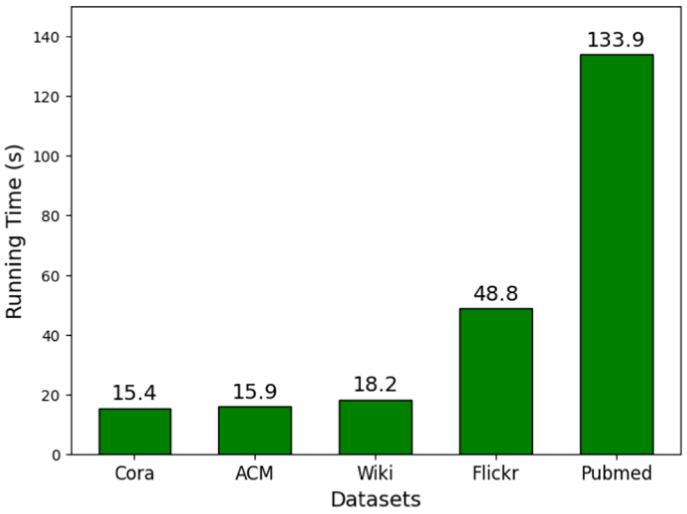


Fig. 4 Running time of GMIM on five datasets

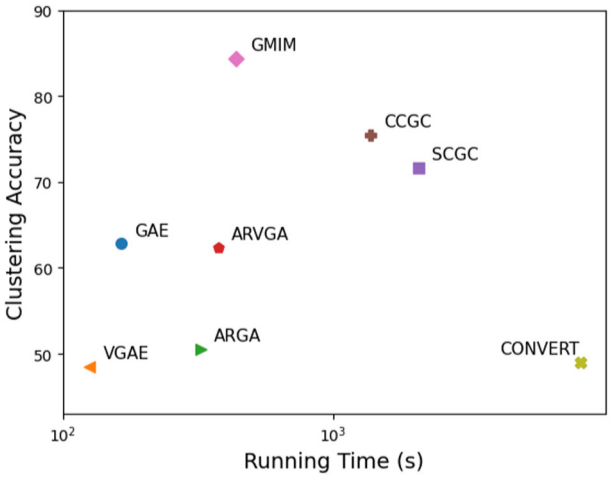
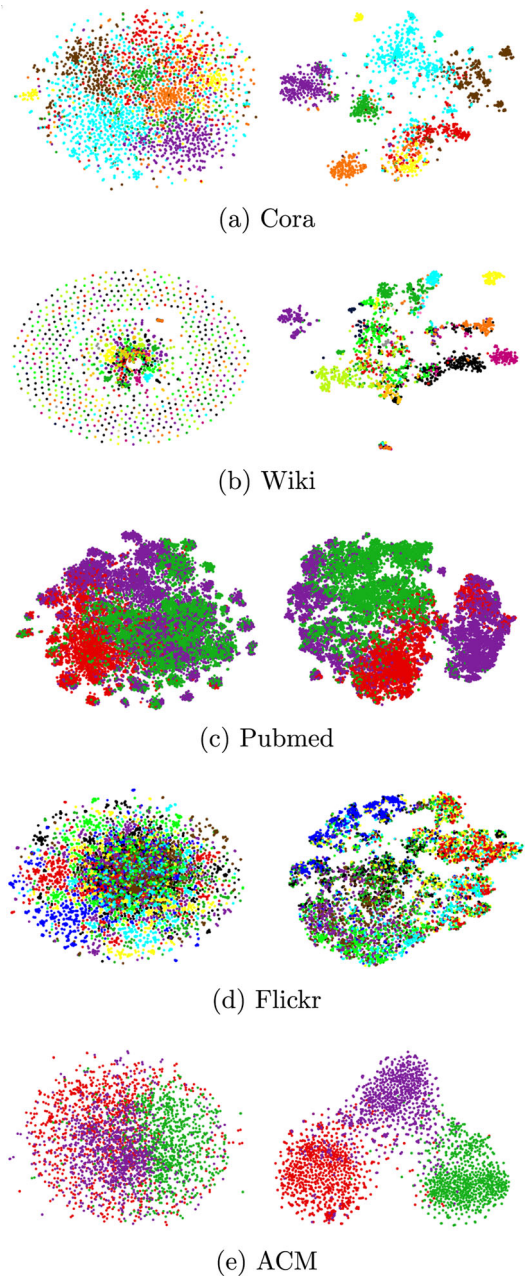


Fig. 5 Clustering accuracy versus running time on Coauthor-Phys dataset

greater than ours. Although our running time is greater than GAE, VGAE, ARGA and ARVGA, GMIM achieves more than 20–30% improvements in terms of ACC against these methods.



(a) Cora

(b) Wiki

(c) Pubmed

(d) Flickr

(e) ACM

Fig. 6 $2D$ visualization of node embeddings on five datasets. (Left) the raw features and (right) the learnt node representations. Different classes are shown by different colors

4.10 Visualization

To show the effectiveness of learnt node embedding, we visualize the node representations of five datasets in two-dimensional space using t-SNE [64] in Fig. 6. As illustrated in this figure, comparing raw features with the node representations in 2D space shows that the learnt representations are cluster-friendly and appropriate for Kmeans to be applied to.

5 Conclusions

In this paper, we introduce a clustering-promoting objective for node embedding. Our proposed method utilizes contrastive learning to produce a clustering-friendly latent space by assuming that the learnt representation follows a mixture of Gaussians distribution. The embedding and clustering-related objectives are optimized in a unified framework to benefit each other. Our experiments show that incorporating the clustering-directed objective function can enhance the clustering ability of graph contrastive learning. We evaluated the proposed method on six real-world datasets. Empirical results demonstrate the effectiveness of our method compared with state-of-the-art methods.

Author Contributions Maedeh Ahmadi wrote the main manuscript text and prepared all figures. All authors reviewed the manuscript.

Funding The authors did not receive support from any organization for the submitted work.

Data Availability The datasets analyzed during this study are available in the following public resources: https://github.com/thunlp/TADW; https://github.com/tkipf/gcn/tree/master/gcn; https://github.com/zhumeiqiBUPT/AM-GCN/tree/master/data.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Appendix A

In this appendix, we give the derivation of the formulas of expectation–maximization update rules in Eqs. 11–15. Suppose we have N data points $H = \{h_1, h_2, \ldots, h_N\}$. We consider each h_n is generated from a GMM with parameters $\Theta = \{\prod = \{\pi_k\}, M = \{\mu_k\}$ and $\sum = \{\Sigma_k\}$ for $k = 1, \ldots, K\}$ and the hidden variables $Z = \{z_1, \ldots, z_N\}$ such that $p(z_n) = \prod_{k=1}^{K} \pi_k^{z_{nk}}$, in which $z_{nk} = 1$ indicates that component k is selected for generating data point n. The joint likelihood is formulated as:

$$\begin{aligned} P(H, Z|\Theta) &= p(h_1, h_2, \ldots, h_N, z_1, z_2, \ldots, z_N|\Theta) \\ &= \prod_{n=1}^{N} p(h_n|z_n, \Theta) \prod_{n=1}^{N} p(z_n|\Theta) \\ &= \prod_{n=1}^{N} \prod_{k=1}^{K} N(h_n \mid \mu_k, \Sigma_k)^{z_{nk}} \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}}, \end{aligned} \tag{A1}$$

therefore:

$$\ln\, p(H, Z|\Theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln\, N(h_n|\mu_k, \Sigma_k) + \ln\, \pi_k. \tag{A2}$$

It can be shown that $\mathbb{E}_{p(Z|H,\Theta)}[\ln\, p(H, Z|\Theta)]$ is the lower bound of log likelihood$(\Theta) = log(p(H; \Theta))$. So we maximize this lower bound in order to maximize the likelihood(Θ). By applying the operator $\mathbb{E}_{p_{(Z|H,\Theta)}}$ to $\ln\, p(H, Z|\Theta)$, we have:

$$\begin{aligned} G(\Theta) &= \mathbb{E}_{p(Z|H,\Theta)}[\ln\, p(H, Z \mid \Theta)] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{p(Z|H,\Theta)}[z_{nk}]\{\ln\, N(h_n|\mu_k, \Sigma_k) + \ln\, \pi_k\}, \end{aligned} \tag{A3}$$

in which:

$$\begin{aligned} \mathbb{E}_{p(Z|H,\Theta)}[z_{nk}] &= \sum_{z_1} \cdots \sum_{z_N} z_{nk}\, p(Z|H, \Theta^{old}) \\ &= \sum_{z_n} z_{nk}\, p(z_n|h_n, \Theta^{old}) \\ &= \frac{\sum_{z_n} z_{nk} \prod_{k=1}^{K} N(h_n|\mu_k, \Sigma_k)^{z_{nk}} \pi_k^{z_{nk}}}{\sum_{z_{nk}} \prod_{k=1}^{K} N(h_n|\mu_k, \Sigma_k))^{z_{nk}} \pi_k^{z_{nk}}} \\ &= \frac{\pi_k N(h_n|\mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l N(h_n|\mu_l, \Sigma_l)} = \mathcal{V}_{nk}. \end{aligned} \tag{A4}$$

where Θ^{old} is the values of Θ in the previous iteration. According to the above equation, $G(\Theta)$ can be rewritten as:

$$\begin{aligned} G(\Theta) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathcal{V}_{nk} \left\{ -\frac{1}{2}(h_n - \mu_k)^T \Sigma_k^{-1}(h_n - \mu_k) - \frac{1}{2}(F \ln 2\pi + \ln \mid \Sigma_k \mid) + \ln\, \pi_k \right\}, \\ s.t. &\sum_{k=1}^{K} \pi_k = 1. \end{aligned} \tag{A5}$$

To maximize $G(\Theta)$, we take the derivatives and set them to 0:

$$\begin{aligned} \frac{dG(\Theta)}{d(\mu_k)} &= \sum_{n=1}^{N} \mathcal{V}(z_{nk}) \Sigma_k^{-1}(h_n - \mu_n) = 0 \\ &\Longrightarrow\ \mu_k = \frac{\sum_{n=1}^{N} \mathcal{V}(z_{nk}) h_n}{\sum_{n=1}^{N} \mathcal{V}(z_{nk})}, \end{aligned} \tag{A6}$$

$$\begin{aligned} \frac{dG(\Theta)}{d(\Sigma_k)} &= \sum_{n=1}^{N} \mathcal{V}(z_{nk})\{(h_n - \mu_k)(h_n - \mu_k)^T - \Sigma_k\} = 0 \\ &\Longrightarrow\ \Sigma_k = \frac{\sum_{n=1}^{N} \mathcal{V}(z_{nk})(h_n - \mu_k)(h_n - \mu_k)^T}{\sum_{n=1}^{N} \mathcal{V}(z_{nk})} \end{aligned} \tag{A7}$$

and

$$\frac{dL(\Theta)}{d(\pi_k)} = 0 \ \Longrightarrow\ \pi_k = \frac{\Sigma_{n=1}^{N} \mathcal{V}(z_{nk})}{N}. \tag{A8}$$

References

1. Yang J, McAuley J, Leskovec J (2013) Community detection in networks with node attributes. In: 2013 IEEE 13th international conference on data mining. IEEE, pp 1151–1156
2. Chen P, Redner S (2010) Community structure of the physical review citation network. J Informetr 4(3):278–290
3. Nicolini C, Bordier C, Bifone A (2017) Community detection in weighted brain connectivity networks beyond the resolution limit. Neuroimage 146:28–39
4. Chen J, Yuan B (2006) Detecting functional modules in the yeast protein-protein interaction network. Bioinformatics 22(18):2283–2290
5. Wang X, Cui P, Wang J, Pei J, Zhu W, Yang S (2017) Community preserving network embedding. In: Thirty-first AAAI conference on artificial intelligence
6. Yang J, Leskovec J (2013) Overlapping community detection at scale: a nonnegative matrix factorization approach. In: Proceedings of the sixth ACM international conference on web search and data mining, pp 587–596
7. Pan S, Hu R, Fung S-F, Long G, Jiang J, Zhang C (2019) Learning graph embedding with adversarial training methods. IEEE Trans Cybern 50(6):2475–2487
8. Wang C, Pan S, Hu R, Long G, Jiang J, Zhang C (2019) Attributed graph clustering: a deep attentional embedding approach. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 3670–3676
9. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv:1611.07308
10. Wang C, Pan S, Long G, Zhu X, Jiang J (2017) Mgae: marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 889–898
11. Zhang X, Liu H, Li Q, Wu X-M (2019) Attributed graph clustering via adaptive graph convolution. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 4327–4333
12. Sun F-Y, Hoffman J, Verma V, Tang J (2020) Infograph: unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In: International conference on learning representations
13. Velickovic P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD (2019) Deep graph infomax. ICLR (Poster) 2(3):4
14. Hassani K, Khasahmadi AH (2020) Contrastive multi-view representation learning on graphs. In: International conference on machine learning. PMLR, pp 4116–4126
15. Jiao Y, Xiong Y, Zhang J, Zhang Y, Zhang T, Zhu Y (2022) Scalable self-supervised graph representation learning via enhancing and contrasting subgraphs. Knowl Inf Syst 64(1):235–260
16. Liu Y, Yang X, Zhou S, Liu X, Wang S, Liang K, Tu W, Li L (2023) Simple contrastive graph clustering. IEEE Trans Neural Netw Learn Syst
17. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H (2016) Variational deep embedding: An unsupervised and generative approach to clustering. arXiv:1611.05148
18. Uğur Y, Arvanitakis G, Zaidi A (2020) Variational information bottleneck for unsupervised clustering: deep gaussian mixture embedding. Entropy 22(2):213
19. Makhzani A, Shlens J, Jaitly N, Goodfellow I, Frey B (2015) Adversarial autoencoders. arXiv:1511.05644
20. Klicpera J, Weißenberger S, Günnemann S (2019) Diffusion improves graph learning. In: Proceedings of the 33rd international conference on neural information processing systems, pp 13366–13378
21. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
22. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
23. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
24. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning. PMLR, pp 1188–1196
25. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
26. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: International conference on learning representations
27. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: International conference on learning representations
28. You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y (2020) Graph contrastive learning with augmentations. Adv Neural Inf Process Syst 33:5812–5823

29. Bachman P, Hjelm RD, Buchwalter W (2019) Learning representations by maximizing mutual information across views. Adv Neural Inf Process Syst 32
30. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning. PMLR, pp 1597–1607
31. Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E Stat Nonlinear Soft Matter Phys 74. https://doi.org/10.1103/PhysRevE.74.036104
32. Karrer B, Newman MEJ (2011) Stochastic blockmodels and community structure in networks. Phys Rev E Stat Nonlinear Soft Matter Phys 83. https://doi.org/10.1103/PhysRevE.83.016107
33. Zhang P, Moore C (2014) Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. Proc Natl Acad Sci USA 111. https://doi.org/10.1073/pnas.1409770111
34. Chang J, Blei D (2009) Relational topic models for document networks. In: Artificial intelligence and statistics. PMLR, pp 81–88
35. Pei Y, Chakraborty N, Sycara K (2015) Nonnegative matrix tri-factorization with graph regularization for community detection in social networks. In: Twenty-fourth international joint conference on artificial intelligence
36. Wang X, Jin D, Cao X, Yang L, Zhang W (2016) Semantic community identification in large attribute networks. In: Proceedings of the AAAI conference on artificial intelligence 30
37. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International conference on machine learning. PMLR, pp 478–487
38. Tsitsulin A, Palowitch J, Perozzi B, Müller E (2020) Graph clustering with graph neural networks. arXiv:2006.16904
39. Sun F-Y, Qu M, Hoffmann J, Huang C-W, Tang J (2019) vgraph: A generative model for joint community detection and node representation learning. Adv Neural Inf Process Syst 32
40. Shchur O, Günnemann S (2019) Overlapping community detection with graph neural networks. arXiv:1909.12201
41. Zhang H, Li P, Zhang R, Li X (2022) Embedding graph auto-encoder for graph clustering. IEEE Trans Neural Netw Learn Syst
42. Guo L, Dai Q (2022) Graph clustering via variational graph embedding. Pattern Recognit 122:108334
43. Yang X, Liu Y, Zhou S, Wang S, Tu W, Zheng Q, Liu X, Fang L, Zhu E (2023) Cluster-guided contrastive graph clustering network. In: Proceedings of the AAAI conference on artificial intelligence, 37
44. Yang X, Tan C, Liu Y, Liang K, Wang S, Zhou S, Xia J, Li SZ, Liu X, Zhu E (2023) Convert: contrastive graph clustering with reliable augmentation. In: Proceedings of the 31th ACM international conference on multimedia
45. Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L (2020) Deep graph contrastive representation learning. arXiv:2006.04131
46. Li Q, Han Z, Wu X-M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-second AAAI conference on artificial intelligence
47. Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab
48. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc: Ser B (Methodol) 39(1):1–22
49. Andersen R, Chung F, Lang K (2006) Local graph partitioning using pagerank vectors. In: 2006 47th annual IEEE symposium on foundations of computer science (FOCS'06). IEEE, pp 475–486
50. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93
51. Yang C, Liu Z, Zhao D, Sun M, Chang E (2015) Network representation learning with rich text information. In: Twenty-fourth international joint conference on artificial intelligence
52. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: The world wide web conference, pp 2022–2032
53. Li J, Hu X, Tang J, Liu H (2015) Unsupervised streaming feature selection in social media. In: Proceedings of the 24th ACM international on conference on information and knowledge management, pp 1041–1050
54. Shchur O, Mumme M, Bojchevski A, Günnemann S (2018) Pitfalls of graph neural network evaluation. arXiv:1811.05868
55. Ng A, Jordan M, Weiss Y (2002) On spectral clustering: analysis and an algorithm. Adv Neural Inf Process Syst 14
56. Tian F, Gao B, Cui Q, Chen E, Liu T-Y (2014) Learning deep representations for graph clustering. In: Proceedings of the AAAI conference on artificial intelligence 28
57. Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: Proceedings of the AAAI conference on artificial intelligence 30

58. Yang C, Liu Z, Zhao D, Sun M, Chang E (2015) Network representation learning with rich text information. In: Twenty-fourth international joint conference on artificial intelligence
59. Li J, Yu J, Li J, Zhang H, Zhao K, Rong Y, Cheng H, Huang J (2020) Dirichlet graph variational autoencoder. Adv Neural Inf Process Syst 33:5274–5283
60. Zhang T, Xiong Y, Zhang J, Zhang Y, Jiao Y, Zhu Y (2020) CommDGI: community detection oriented deep graph infomax. In: Proceedings of the 29th ACM international conference on information and knowledge management, pp 1843–1852
61. Zhang X, Liu H, Wu X-M, Zhang X, Liu X (2021) Spectral embedding network for attributed graph clustering. Neural Netw 142:388–396
62. Zheng S, Zhu Z, Zhang X, Liu Z, Cheng J, Zhao Y (2020) Distribution-induced bidirectional generative adversarial network for graph representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7224–7233
63. Reynolds DA (2009) Gaussian mixture models. Encycl Biom 741:659–663
64. Van Der Maaten L (2014) Accelerating t-SNE using tree-based algorithms. J Mach Learn Res 15(1):3221–3245

Maedeh Ahmadi received her B.Sc. degree in computer hardware from Isfahan University, Isfahan, Iran, and her M.Sc. degree in artificial intelligence from Isfahan University of Technology, Isfahan, Iran. She is currently a Ph.D. student in artificial intelligence at Isfahan University of Technology, Isfahan, Iran. Her research interests include machine learning and pattern recognition.

Mehran Safayani received his B.S. degree in computer engineering from Isfahan University, Isfahan, Iran, in 2002. Then, he received the M.Sc. and Ph.D. degrees from Sharif University of Technology, Tehran, Iran, in computer architecture and artificial intelligence in 2006 and 2011, respectively. Since 2012, he is an associate professor of electrical and computer engineering department at Isfahan University of Technology. His research interests include machine leaning, deep learning and large language model.

Abdolreza Mirzaei was born in Isfahan, Iran. He received the B.Sc. (first-class honors) degree in computer engineering from Isfahan University, in 2001, the M.Sc. degree in artificial intelligence from Iran University of Science and Technology, Tehran, Iran, in 2003, and the Ph.D. degree in artificial intelligence from Amirkabir University of Technology, Tehran, in 2009, respectively. He is currently in the department of electrical and computer engineering, Isfahan University of Technology. His research interests include statistical and structural classification methods, digital image processing, computer vision, multiple classifier systems and learning methods.