Deep Graph Clustering via Mutual Information Maximization and Mixture Model

Maedeh Ahmadi, Mehran Safayani, Abdolreza Mirzaei

Abstract

Attributed graph clustering or community detection which learns to cluster the nodes of a graph is a challenging task in graph analysis. In this paper, we introduce a contrastive learning framework for learning clustering-friendly node embedding. Although graph contrastive learning has shown outstanding performance in self-supervised graph learning, using it for graph clustering is not well explored. We propose Gaussian mixture information maximization (GMIM) which utilizes a mutual information maximization approach for node embedding. Meanwhile, it assumes that the representation space follows a Mixture of Gaussians (MoG) distribution. The clustering part of our objective tries to fit a Gaussian distribution to each community. The node embedding is jointly optimized with the parameters of MoG in a unified framework. Experiments on real-world datasets demonstrate the effectiveness of our method in community detection.

Index Terms

Graph neural network, graph representation learning, graph clustering, contrastive learning.

I. 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 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. For example, Deep Graph Infomax (DGI) [12] have made great progress in node embedding via maximizing mutual information (MI) between node representations and graph summary. Despite the high performance of contrastive methods in node representation learning, they have not received sufficient attention for community detection. Obviously, better node embedding results in enhanced community detection.

In this paper, we aim to utilize the great potential of contrastive node embedding in community detection. One naïve approach is to apply a clustering algorithm, e.g. K-means, on the embedding result of a contrastive representation learning method to extract communities. This 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. To address this, we propose to learn a clustering-friendly node embedding which utilizes a contrastive method for node representation learning. As the contrastive embedding method, we follow the approach of [12] which relies on mutual information maximization to learn representations. For clustering, we assume that the learnt node embedding space follows a Mixture of Gaussians (MoG) distribution. The parameters of the contrastive model and MoG are optimized in an iterative manner by gradient descent algorithm and Expectation Maximization (EM) [13]. 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) [14] which helps with the task of clustering by providing a global view of the graph.

The authors are with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran (e-mail: m.ahmadi@ec.iut.ac.ir)
Contributions of our method are summarized as follows: 1) we introduce a unified framework for learning clustering-friendly node embedding. 2) We utilize graph diffusion to benefit from the global view of the graph in the clustering task. 3) The experimental evaluations on real-world datasets demonstrate the effectiveness of our proposed method.

The rest of the paper is organized as follows. We review the related work of graph embedding and graph clustering in section II. Section III introduces a detailed description of the proposed method. Experimental results on two real-world datasets are presented section IV. The conclusions are given in section V.

II. RELATED WORKS

A. 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 [15], [16]. These methods take random walks along nodes and utilize neural language models (like SkipGram [17]) 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) [18]–[20] have demonstrated strong representation power for attributed graph learning tasks. They 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 first-order neighbor nodes in the embedding space. Both of random walks and graph autoencoders-based methods over-emphasize the local proximity information [12].

Recently, contrastive approaches have achieved state-of-the-art results in graph data analysis [12], [21]–[23]. They contrast samples from a desired distribution and another undesired one. Motivated by the excellent results of contrastive learning in visual representation learning [24], [25], graph contrastive algorithms propose to retain local and global structural information of graphs [12], [21].

B. 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 [26] are two widely used approaches in this area. Stochastic block model-based methods [27] are also well explored. Modularity maximization is a popular goal to extract communities [28]. To exploit both of content and structural information, several extended algorithms based on topic models [29] and NMF [30], [31] 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].

Recent methods try to combine clustering and graph embedding goals. [8] co-optimizes a graph attention-based reconstruction loss and the clustering loss of [32], [33] maximizes modularity on the embedding space of a GCN. A probabilistic generative model which learns node embedding and community assignment jointly is proposed in [34]. In [35], GCN is integrated with Bernoulli-Poisson probabilistic model [6] for overlapping community detection. [36] trains a graph auto-encoder to find an appropriate embedding space for relaxed K-means. A variational framework for learning clustering and node embedding is introduced in [37].

III. Method

A. 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 $x_i$ 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 K-means 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. 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 figure 1.

B. Node Embedding

We use the contrastive framework of [12] 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 \( s = R(H) = \sigma(1/N \sum_{i=1}^{N} h_i) \), with logistic sigmoid nonlinearity \( \sigma \). Given representation vector \( h \) and \( s \), the following discriminator \( D \) distinguishes between representations from positive and negative graphs by assigning higher probabilities to representation vectors that the summary contain them:

\[
D(h, s) = \sigma(h^T W s),
\]

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

\[
\mathcal{L}_{MI} = -\frac{1}{2N} \left( \sum_{i=1}^{N} \mathbb{E}_{(X,A)}[\log D(h_i, s)] + \sum_{j=1}^{N} \mathbb{E}_{(\tilde{X},\tilde{A})}[[\log(1 - D(\tilde{h}_j, s))] \right).
\]

The encoder is the following single-layer GNC:

\[
\mathcal{E}(X, A) = \text{PReLU}(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Phi),
\]

where \( \tilde{A} = A + I_N \) is the adjacency matrix with self-connections and \( \tilde{D}_{ii} = \sum_j A_{ij} \) is the corresponding degree matrix. \( \Phi \) is a learnable transition matrix and PReLU represents parametric rectified linear unit function.

C. 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 [38]. 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) [14]. It replaces the adjacency matrix with a diffusion matrix which is formulated as:

\[
S = \sum_{k=0}^{\infty} \Theta_k T^k,
\]

with generalized transition matrix \( T \) and weighting coefficients \( \Theta \). One popular example of graph diffusion is Personalized PageRank (PPR) [39]. Given adjacency matrix \( A \) and related degree matrix \( D \), (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:

\[
S^{PPR} = \alpha(I_n - (1 - \alpha)D^{-\frac{1}{2}} AD^{-\frac{1}{2}})^{-1}.
\]

This diffusion matrix provides global view of a graph, acts as a low-pass filter and smoothes out the neighborhood over the graph [14]. GDC can be integrated with any kind of graph-based model. In this work we utilize it as the input of our model instead of adjacency matrix.
Fig. 1: Our GMIM framework. A graph diffusion is applied to the graph. The resulted graph and its corrupted version are fed to a shared encoder. The output features construct information maximization embedding objective. The clustering module (top) aims to enforce this representation to follow a MoG distribution. The embedding and clustering modules are trained jointly.

D. 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 $\Psi$. 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:

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

(6)

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:

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

(7)

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

E. 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 $L_{MI}$ 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:

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

(8)
Our total loss function is defined as:

$$L = \omega L_{MI} + \beta L_{NLL},$$

(9)

where $L_{MI}$ and $L_{NLL}$ are the mutual information loss and the negative log-likelihood (NLL) respectively. The weights $\omega$ and $\beta$ balance between two terms of the objective function. After optimizing our objective, we have a K-means-friendly latent space on which we apply k-means algorithm to obtain the final clusters of nodes.

F. Inference

The total loss function of (9) consists of two sets of parameters: Node embedding parameters ($\Psi$) and MoG parameters ($\Pi, M$ and $\Sigma$). To optimize these parameters, we use an iterative approach by fixing one set and optimizing the other. We initialize the $\Psi$ parameters by training the model using (2) as the loss function. To initialize MoG parameters, we apply $K$-means algorithm on the achieved embedding from $\Psi$ initialization. We initialize ($\Pi, M, \Sigma$) using the hard assignment results of $K$-means algorithm. The details of this iterative approach is described below.

**Fixing $\Psi$ Parameters and Optimizing ($\Pi, M, \Sigma$)**

Fixing deep network parameters, we use expectation maximization algorithm [13] to optimize ($\Pi, M, \Sigma$). The following equations are used iteratively to update these parameters:

$$\pi_{i,k} = \frac{N_k}{|V|},$$

(10)

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

(11)

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

(12)

where

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

(13)

and

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

(14)

More precisely, we update $\mathcal{V}_{ik}$ in $E$-step and ($\Pi, M, \Sigma$) in the $M$-step of the EM algorithm.

**Fixing ($\Pi, M, \Sigma$) and Updating $\Psi$ Parameters**

Fixing MoG parameters, we optimize the total loss function of (6) with respect to $\Psi$ parameters using stochastic gradient descent (SGD). $\Psi$ consists of the learnable scoring matrix $W$ of (1), encoder parameters $\Phi$ and PReLU parameters of (3). Our proposed method is summarized in Algorithm 1.

IV. Experiments

A. Benchmark Datasets

We use two standard widely-used network datasets (Cora and Pubmed) for attributed graph community detection in our experiments. Cora and Pubmed are citation networks. Nodes represent papers and edges correspond to citations. Labels are papers topics and features are bag-of-word vectors. Features of Cora are binary vectors while Pubmed is represented by tf-idf weights. Table summarizes the detailed statistics of datasets.
TABLE I: Datasets statistics

| Dataset | Nodes | Edges | Features | Clusters |
|---------|-------|-------|----------|----------|
| Cora    | 2708  | 5429  | 1433     | 7        |
| Pubmed  | 19717 | 44338 | 500      | 3        |

Algorithm 1 Gaussian Mixture Information Maximization.

Require:
- Graph $G = (V, E, X)$, number of clusters $K$, weight $\omega$,
- PPR parameter $\alpha$ and hidden dimension.

Ensure:
- Node embedding $H$, final community assignments.

Calculate $S_{PPR}$ by (5)
Replace $A$ by $S_{PPR}$ in (2)
Initialize $H$ by optimizing (2)
Initialize MoG parameters by applying K-means on $H$

for $t = 0$ to $T$
  for $t_1 = 0$ to $T_1$
    Update $\Pi, M, \Sigma$ by equations (10), (11) and (12)
  end
  for $t_2 = 0$ to $T_2$
    Calculate mutual information loss by (2)
    Calculate negative log-likelihood loss by (8)
    Update $\Psi$ parameters by SGD on (9)
  end
end
Get the final community assignments by applying K-means on $H$.

B. Baseline Methods

We compare GMIM with the following baseline methods. These approaches are categorized into three groups:
1) Methods which use node features only: K-means and spectral clustering [40] 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 [15] generates random paths along a graph and use them to train SkipGram language model to learn node embedding. GraphEncoder [41] trains a stacked sparse auto-encoder to obtain node embedding. DNGR [42] uses stacked denoising autoencoders to learn each node representation. K-means in applied to the learnt latent space of the three later methods. vGraph [34] is a probabilistic generative model which performs graph clustering and node embedding jointly.
3) Methods which use both node features and graph structure: TADW [43] adds node features to DeepWalk framework. GAE and VGAE [9] integrate (variational) autoencoder and graph neural networks for node embedding. MGAE [10] introduces a marginalized graph autoencoder for graph clustering. ARGA and ARVGA [7] use an adversarial training scheme to impose a prior distribution on latent space of GAE and VGAE. DGVAE [44] 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 [45] incorporates contrastive learning to learn cluster assignment of the nodes. DAEGC [8] optimizes graph reconstruction loss and a clustering loss jointly. SENet [46] uses a spectral clustering loss to learn node embeddings. GC-VGE [37] introduces a joint framework for clustering and representation learning by utilizing a variational graph embedding mechanism. EGAE [36] learns graph auto-encoder and relaxed K-means simultaneously to obtain a latent space which is ideal for relaxed K-
means. This method has not reported its results for Pubmed dataset. GALA [47] proposes a symmetric graph convolutional autoencoder in which the encoder and decoder are based on Laplacian smoothing and sharpening respectively. DBGAN [48] introduces an adversarial framework to learn node embeddings.

C. Evaluations Metrics and Implementation Details

We report five evaluation metrics to measure the performance of graph clustering: clustering accuracy (ACC), normalized mutual information (NMI), adjusted rand index (ARI), F-Score (F1) and precision (P). The higher values of all these metrics indicates the better results. We run our algorithm 10 times on each dataset and report the average of obtained metrics. For the encoder, we set the size of hidden dimension to 512 and 256 for Cora and Pubmed respectively. Due to memory limitations we could not increase the hidden size for Pubmed. We set $\alpha = 0.2$ for PPR diffusion. The weight $\omega$ is set to 25000 and 1000 for Cora and Pubmed respectively, to balance two terms of objective function. At the start of training we set $\beta$ to zero and as training progresses, we gradually increase it to reach one. We use the Adam SGD optimizer with learning rate of 0.001 in both initialization and training phases for both datasets. $T_1$ and $T_2$ are set to one for both datasets. We train the model for 200 and 1000 epochs on Cora and Pubmed respectively.

D. Experimental Results

Our experimental results are summarized in tables II and III. F, G and F&G indicate the methods which use only node features, graph structure or both of features and structure information, respectively. As shown in these tables, methods using both feature and structure generally outperform the methods using only one source of information. This indicates the importance of these information for clustering task. Our method has superior performance to TADW. This method does not utilize GNN for node embedding. Our method significantly outperforms GAE, VGAE, MGAE, ARGA and ARVGA. These are two-stage methods which perform node embedding and clustering stage independently. DAEGC has a unified framework for clustering and representation learning but its objective rely on adjacency matrix reconstruction which over-emphasize proximity information. AGC captures the global structure of a graph by utilizing high-order graph convolution. This is similar to how our framework benefits from graph diffusion which gives a global view of the graph. However, this method performs representation learning and clustering independently. The proposed method has higher performance than GALA and DBGAN. These methods learn cluster assignment and node embeddings in two separate stages. CommDGI, GC-VGE and EGAE learn cluster assignments and node representations jointly but they have lower performance in comparison to our method.

E. Ablation Study

To study the effect of using diffusion graph, we train the model using only the mutual information loss ($L_{MI}$) without replacing the adjacency matrix with the diffusion graph. We apply K-means and Gaussian mixture model (GMM) on the learnt embedding and the average result of 10 runs is reported in table IV as MI+K-means/GMM. We then replace the adjacency matrix with diffusion matrix and repeat the above experience. MI+Diffusion+K-means/GMM shows the related results. Comparing MI+K-mean/GMM with MI+Diffusion+K-means/GMM shows that the diffusion can improve the clustering performance. To evaluate the effectiveness of our unified framework for learning clustering-friendly node embedding, we compare it against MI+Diffusion+K-means/GMM. The fact that our method outperforms DGI+Diffusion+K-means/GMM confirms the effectiveness of jointly optimizing MI and NLL objectives.

F. Visualization

We visualize the node representations of Pubmed dataset in a two-dimensional space using t-SNE [49]. The result is shown in Fig. 2. The latent space of MI+Diffusion and GMIM are shown in Fig. 2a and 2c respectively. Although both of these subfigures show more separable embedding in comparison to raw features, the latent space of GMIM has better fit to a mixture of Gaussians distribution. So, it is more clustering-friendly and is more appropriate for K-means to be applied to.
TABLE II: Clustering results on Cora dataset

| Method               | Info. | ACC  | NMI  | ARI  | F1   | P    |
|----------------------|-------|------|------|------|------|------|
| K-means              | F     | 49.2 | 32.1 | 23.0 | 36.8 | 36.9 |
| Spectral-F [40]      | F     | 34.7 | 14.7 | 7.1  | -    | -    |
| Spectral-G [40]      | G     | 31.46| 9.69 | 0.35 | 29.67| 18.07|
| DeepWalk [15]        | G     | 56.20| 39.87| 32.18| 47.6 | 5.48 |
| GraphEncoder [41]    | G     | 32.5 | 10.9 | 0.6  | 29.8 | 18.2 |
| DNGR [42]            | G     | 44.39| 33.31| 15.86| 34.68| 27.86|
| vGraph [34]          | G     | 28.7 | 34.5 | 31.2 | 30.5 | -    |
| TADW [43]            | F&G   | 55.00| 36.59| 26.40| 41.52| 36.50|
| GAE [9]              | F&G   | 60.34| 44.85| 36.73| 58.72| 61.39|
| VGAE [9]             | F&G   | 63.56| 47.45| 39.42| 63.75| 65.64|
| MGAE [10]            | F&G   | 63.43| 45.57| 38.01| 38.01| -    |
| ARGA [7]             | F&G   | 60.84| 42.21| 36.88| 60.49| 63.38|
| ARVGA [7]            | F&G   | 62.83| 45.93| 38.00| 63.17| 64.80|
| DGVAE [44]           | F&G   | 64.42| 47.64| 38.42| 62.69| 64.90|
| AGC [11]             | F&G   | 68.92| 53.68| 48.6 | 65.61| -    |
| CommDGI [45]         | F&G   | 69.8 | 57.9 | 50.2 | 68.4 | -    |
| DAEGC [8]            | F&G   | 70.4 | 52.8 | 49.6 | 68.2 | -    |
| SENet [46]           | F&G   | 71.92| 55.08| 48.96| -    | -    |
| GC-VGE [37]          | F&G   | 70.67| 53.57| 48.15| 69.48| 70.51|
| EGAE [36]            | F&G   | 72.42| 53.96| 47.22| -    | -    |
| GALA [47]            | F&G   | 74.6 | 57.7 | 53.2 | -    | -    |
| DBGAN [48]           | F&G   | 74.8 | 56.0 | 54.0 | -    | -    |
| GMIM (ours)          | F&G   | 75.65| 60.30| 54.74| 71.20| 78.13|

Fig. 2: 2D visualization of node embeddings on Pubmed dataset. Different classes are shown by different colors.

V. 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 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 real-world datasets to show its effectiveness. Empirical results demonstrate the favorable performance of our method compared with state-of-the-art methods.
### TABLE III: Clustering results on Pubmed dataset

| Method            | Info. | ACC  | NMI  | ARI  | F1   | P    |
|-------------------|-------|------|------|------|------|------|
| $K$-means         | F     | 55.59| 24.34| 21.54| 56.04| 46.08|
| Spectral-F [40]   | F     | 60.20| 30.90| 27.7 | -    | -    |
| Spectral-G [40]   | G     | 37.98| 10.30| 26.67| 50.54|-0.02 |
| DeepWalk [15]     | G     | 64.98| 26.44| 27.42| 63.46| 65.24|
| GraphEncoder [41] | G     | 53.1 | 20.9 | 18.4 | 50.6 | 45.6 |
| DNGR [42]         | G     | 25.53| 20.11| 8.29 | 15.57| -0.02|
| vGraph [34]       | G     | 26.00| 22.40| 18.50| 33.20|-    |
| TADW [43]         | F&G   | 46.82| 9.47 | 5.78 | 51.22| 38.34|
| GAE [9]           | F&G   | 64.43| 24.85| 23.57| 64.07| 65.26|
| VGAE [9]          | F&G   | 64.67| 23.94| 23.41| 64.77| 64.53|
| MGAE [10]         | F&G   | 43.88| 8.16 | 3.98 | 41.98|-    |
| ARGGA [7]         | F&G   | 65.07| 29.23| 26.79| 64.11| 69.27|
| ARVGGA [7]        | F&G   | 62.01| 26.62| 22.46| 61.66| 68.41|
| DGVVAE [44]       | F&G   | 67.56| 28.72| 24.92| 64.35| 67.10|
| AGC [11]          | F&G   | 69.78| 31.59| 31.19| 68.72|-    |
| CommDGI [45]      | F&G   | 69.90| 35.70| 29.2 | 69.2 | -    |
| DAEGC [8]         | F&G   | 67.10| 26.60| 27.8 | 65.9 | -    |
| SENet [46]        | F&G   | 67.59| 30.61| 29.66| -    | -    |
| GC-VGE [37]       | F&G   | 68.18| 29.70| 29.76| 66.87| 69.39|
| GALA [47]         | F&G   | 69.39| 32.73| 32.1 | -    | -    |
| DBGAN [48]        | F&G   | 69.40| 32.40| 32.7 | -    | -    |
| GMIM (ours)       | F&G   | 70.87| 32.43| 33.25| 69.19| 70.83|

### TABLE IV: Ablation study

| Method                   | Input | Cora ACC | NMI  | Pubmed ACC | NMI  |
|--------------------------|-------|----------|------|------------|------|
| MI+K-means               | X,A   | 72.15    | 56.50| 66.37      | 31.08|
| MI+GMM                   | X,A   | 67.69    | 53.52| 64.36      | 28.04|
| MI+Diffusion+K-means     | X,S   | 74.45    | 58.99| 67.87      | 31.40|
| MI+Diffusion+GMM         | X,S   | 69.47    | 56.32| 65.85      | 26.36|
| GMIM (ours)              | X,S   | 75.65    | 60.30| 70.87      | 32.43|

### REFERENCES

[1] J. Yang, J. McAuley, and J. Leskovec, “Community detection in networks with node attributes,” in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 1151–1156.

[2] P. Chen and S. Redner, “Community structure of the physical review citation network,” *Journal of Informetrics*, vol. 4, no. 3, pp. 278–290, 2010.

[3] C. Nicolini, C. Bordier, and A. Bifone, “Community detection in weighted brain connectivity networks beyond the resolution limit,” *Neuroimage*, vol. 146, pp. 28–39, 2017.

[4] J. Chen and B. Yuan, “Detecting functional modules in the yeast protein–protein interaction network,” *Bioinformatics*, vol. 22, no. 18, pp. 2283–2290, 2006.

[5] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, “Community preserving network embedding,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[6] J. Yang and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 587–596.

[7] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, and C. Zhang, “Learning graph embedding with adversarial training methods,” *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2019.

[8] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, “Attributed graph clustering: a deep attentional embedding approach,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3670–3676.
[44] J. Li, J. Yu, J. Li, H. Zhang, K. Zhao, Y. Rong, H. Cheng, and J. Huang, “Dirichlet graph variational autoencoder,” Advances in Neural Information Processing Systems, vol. 33, pp. 5274–5283, 2020.

[45] T. Zhang, Y. Xiong, J. Zhang, Y. Zhang, Y. Jiao, and Y. Zhu, “Commdgi: community detection oriented deep graph infomax,” in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1843–1852.

[46] X. Zhang, H. Liu, X.-M. Wu, X. Zhang, and X. Liu, “Spectral embedding network for attributed graph clustering,” Neural Networks, vol. 142, pp. 388–396, 2021.

[47] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, “Symmetric graph convolutional autoencoder for unsupervised graph representation learning,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6519–6528.

[48] S. Zheng, Z. Zhu, X. Zhang, Z. Liu, J. Cheng, and Y. Zhao, “Distribution-induced bidirectional generative adversarial network for graph representation learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7224–7233.

[49] L. Van Der Maaten, “Accelerating t-sne using tree-based algorithms,” The journal of machine learning research, vol. 15, no. 1, pp. 3221–3245, 2014.