AnchorGAE: General Data Clustering via $O(n)$ Bipartite Graph Convolution

Hongyuan Zhang, Jiankun Shi, Rui Zhang, Xuelong Li
School of Computer Science and School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi’an 710072, Shaanxi, P. R. China
hyzhang98@gmail.com, henusjk@163.com, ruizhang8633@gmail.com, xuelong_li@ieee.org

Abstract

Graph-based clustering plays an important role in clustering tasks. As graph convolution network (GCN), a variant of neural networks on graph-type data, has achieved impressive performance, it is attractive to find whether GCNs can be used to augment the graph-based clustering methods on non-graph data, i.e., general data. However, given $n$ samples, the graph-based clustering methods usually need at least $O(n^2)$ time to build graphs and the graph convolution requires nearly $O(n^2)$ for a dense graph and $O(|E|)$ for a sparse one with $|E|$ edges. In other words, both graph-based clustering and GCNs suffer from severe inefficiency problems. To tackle this problem and further employ GCN to promote the capacity of graph-based clustering, we propose a novel clustering method, AnchorGAE. As the graph structure is not provided in general clustering scenarios, we first show how to convert a non-graph dataset into a graph by introducing the generative graph model, which is used to build GCNs. Anchors are generated from the original data to construct a bipartite graph such that the computational complexity of graph convolution is reduced from $O(n^2)$ and $O(|E|)$ to $O(n)$. The succeeding steps for clustering can be easily designed as $O(n)$ operations. Interestingly, the anchors naturally lead to a siamese GCN architecture. The bipartite graph constructed by anchors is updated dynamically to exploit the high-level information behind data. Eventually, we theoretically prove that the simple update will lead to degeneration and a specific strategy is accordingly designed.

1. Introduction

Graph-based clustering methods [14,24,25,29,30,36] are important in the clustering field since they can group data points non-linearly via constructing a graph. For instance, spectral clustering [29] originates from a relaxed graph-cut problem, e.g., Ratio Cut [14], Normalized Cut [36], Balanced Cut [6], etc. Given $n$ data points, the graph-based clustering methods suffer the severe inefficiency in practice due to the construction of graphs, which usually needs $O(n^2)$ time without any acceleration tricks. For some methods such as spectral clustering [29], the eigen-value decomposition, an $O(n^3)$ operation, is required. To alleviate the phenomenon, plenty of variants of spectral clustering have been developed [12,37,45]. An important variant is to introduce some representative points [27] to accelerate both graph construction and optimization [3,7]. However, the graph constructed by original features may only contain low-level information. With the rise of deep learning, the utilization of deep neural networks (DNN) is widely regarded as an important way to exploit deep information of data [44]. Although some deep clustering models [13,34,46,47] have been proposed, few models focus on how to extend graph-based methods into deep version. Moreover, some models based on convolution neural networks (CNN) have achieved impressive improvement on clustering, but CNN can not be applied on irregular data, such as social networks, recommendation systems, signals, data mining data, etc. Compared with the regular data (e.g., images, videos, etc.), the most difference of irregular data is that features of irregular data are not relevant to the neighboring features. In other words, the features are disordered, which indicates that the classical convolution operation is unapplicable.

Graph convolution networks (GCN) [2,10,22], which attempt to extend the classical convolution operation into irregular data, have attracted much attention. GCN is designed for graph-type data, such as social networks, recommendation systems, citation networks, etc. According to the practical scenarios, links between two samples are cheap to obtain and thus graphs are provided as prior information. Several works [21,32,33,39,40] aim to utilize GCN for node clustering on graph-type data. As GCNs have shown impressive capacity on graph-type data, it is attractive to employ GCN to promote the capacity of graph-based clustering methods. Nevertheless, in the most scenarios (e.g., text, signals, etc.), graphs are not provided as the prior information, which indicates that the existing GCN-based models for node clustering are unavailable.
on these non-graph data without artificial construction of graphs. Meanwhile, inefficiency is also a well-known problem for GCNs [4, 15]. Forward propagation of each node in a GCN layer relies on neighboring samples such that the computation complexity of the graph convolution increases exponentially with the growth of layers. For a dense graph, the complexity of one GCN layer is almost $O(n^2)$ time. For a sparse one, the computational cost becomes $O(|E|)$ where $|E|$ represents the amount of edges. Accordingly, each training step approximately depends on all data points and the training is quite time-consuming. It also results in the unavailability of batch gradient descent. To accelerate GCNs, some works focus on how to apply batch gradient descent [4, 5, 8, 41].

Although SDCN [1] aims to extend GCNs into the data clustering task, it fails to exploit the high-level information of data and neglect the practicability in the real applications. In this paper, we propose a novel-based GCN clustering model, namely AnchorGAE. The contributions of this paper are concluded as follows

- Since no graph is provided for general data, we aim to estimate the connectivity distribution for each sample from a generative model perspective. Anchors are introduced to accelerate the construction and apply GCNs to the practical applications. With the produced anchors and connectivity distributions, we obtain a bipartite graph and its one-step probability transition graph to reduce the computational complexity of each forward propagation from $O(n^2)$ or $O(|E|)$ to $O(n)$.

- Furthermore, a siamese architecture is naturally designed to transform anchors into the embedding space. The distributions are rectified dynamically during training to exploit the high-level information via GCN.

Meanwhile, we devise a specific strategy to update the graph to avoid the collapse that is proved theoretically.

**Notations** Let $G = (V, E, W)$ be a graph where $V$, $E$, and $W$ represent nodes, edges, and weights respectively. For an unweighted matrix, $W$ can be ignored since weights of edges are all regarded as 1. All matrices and vectors are represented by upper-case words and bold lower-case words respectively. $|·|$ represents the size of set. $(·)_+ = \max(·, 0)$. For a real number $x$, $\lfloor x \rfloor$ represents the maximum integer that is not larger than $x$. $\mathbf{1}_n$ denotes an $n$-dimension vector whose all entries equal 1. The theoretical proof and details of experiments are shown in the supplementary.

## 2. Related Work

In this section, we introduce the existing work about graph convolution networks and fast clustering algorithms.

**Graph Convolution Networks** In recent years, how to effectively extract features of graphs has attracted a lot of attention [22, 42]. Although GCN has achieved outstanding performance [16, 22], it requires too much memory and time. Some methods [5, 8, 41] for accelerating training and reducing consumption of memory are proposed. For instance, SGC [41] attempts to accelerate by removing non-linear activations in GCN, which reduces the number of parameters. More importantly, it compresses multiple layers into one layer such that stochastic gradient descent can be used. ClusterGCN [8] tries to find an approximate graph with multiple connected components via METIS [19] to employ batch gradient descent. To extend GCN [22] into unsupervised learning, graph auto-encoders [21, 26, 32, 33, 48], based on auto-encoder [17, 20], are developed. Different from traditional auto-encoders, GCN is...
used as an encoder and decoder calculates inner-products of embedding to reconstruct graphs.

**Efficient Clustering** In traditional clustering, k-means [28] works efficiently on large scale data but it can only deal with spherical data, which limits the application of k-means. For graph-based clustering methods [14, 29, 36], both computational complexity and space complexity of graph construction are $O(n^2)$. In particular, solving spectral clustering needs eigen-value decomposition, which is also time-consuming. To apply spectral clustering on large scale data, a lot of variants [12, 27, 37, 45] have been developed. Anchor-based methods [3, 7, 27] are important extensions of spectral clustering. The anchor-based methods implicitly via some representative points, which are usually named as anchors. By constructing a bipartite graph between samples and anchors, they only need to perform singular value decomposition on a small matrix. The anchor-based spectral clustering provides an elegant scheme to improve the efficiency of graph-based models.

**Remark 1.** Although several CNN-based clustering methods [13, 46] have achieved good performance, they focus on vision data and the main improvement is caused by the classical convolution operator, such that they are unavailable on irregular data, e.g., signals, word vectors, data mining data, etc. Therefore, the CNN-based methods will not be discussed in the following part.

3. Methodology

In this section, we first show how to construct a bipartite graph on the non-graph data. Then, the core idea of AnchorGAE and its efficiency are elaborated. Finally, the rigorous analysis of the degeneration is provided. The core idea of AnchorGAE is illustrated in Figure 1.

3.1. Inefficiency of GCN Revisit

In computer science, the adjacency matrix, which is denoted by $A$ in this paper, is a common way to represent the edges of a graph. Specifically, for an unweighted graph $G = (V, E)$, $A_{ij} = 1$ if $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. In GCNs, a popular definition of the graph convolution layer is accordingly formulated as [22]

$$H = \varphi(\bar{A}XW),$$

where $\varphi(\cdot)$ is an activation function, $X \in \mathbb{R}^{n \times d}$ is the representation of data points, $W \in \mathbb{R}^{d \times d}$ denotes the coefficients to learn, $A = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$ is often viewed as a variant of the normalized Laplacian matrix, and $D$ is the degree matrix of $(A + I)$. Specifically, $D$ is a diagonal matrix where $D_{ii} = \sum_{j=1}^{n}(a_{ij} + 1)$.

In the forward propagation, it requires $O(|E|d)$ time to compute $\bar{A}X$. If $G$ is sparse enough, the computational complexity is acceptable. If there are plenty of edges in $G$, the computational complexity can be roughly regarded as $O(n^2d)$, such that GCNs are unavailable in this case.

3.2. Generative Perspective for Bipartite Graphs

**Generative Perspective for Graphs** For a graph $G = (V, E)$, an edge, $(v_i, v_j) \in E$, can be regarded as a sampling from an underlying connectivity distribution $p(\cdot|v_i)$ for $v_i$, where $v_i$ denotes the $i$-th node. Note that $\sum_{j=1}^{n} p(v_j|v_i) = 1$ where $n = |V|$. Therefore, a graph can be viewed as multiple samplings regarding connectivity distributions $\{p(\cdot|v_i)\}_{i=1}^{n}$. Specially, for a weighted graph $G = (V, E, W)$ where $W$ represents the weights on edges, $W$ can be viewed as an estimation of the underlying connectivity distributions. From the generative perspective, the underlying connectivity distributions can be supposed to exist in any dataset. Different from the graph data, the sampling is not performed on non-graph data. However, no matter which type of data is, it will not affect the existence of connectivity distributions. The vital idea that leads AnchorGAE to be applied to non-graph data is to estimate the distributions. Besides, they are also the target to reconstruct by our unsupervised model. The following assumption helps to estimate the connectivity distributions.

**Assumption 1.** Given an ideal metric of difference between two nodes $d(v_i, v_j)$, $p(v_j|v_i)$ is negatively related to $d(v_i, v_j)$.

In this paper, each node is described by a $d$-dimension vector, i.e., $x_i \in \mathbb{R}^{d}$. Therefore, the metric can be reformulated as

$$d(v_i, v_j) = \|f(x_i) - f(x_j)\|_2^2, \quad (2)$$

where $f(\cdot)$ represents an ideal mapping of the raw features.

**Bipartite Graph Simplification with Anchors** To efficiently apply graph convolution, we attempt to simplify the graph via some representative points, which are named anchors or landmarks. Let $U$ denote anchors where $|U| = m$ and the $i$-th anchor is also described by a $d$-dimension vector $c_i$. Then, we can get a new bipartite graph with anchors, $G_a = (V_a, E_a, W_a)$. Specifically, $V_a = V \cup U$. The bipartite property means that only edges between $V$ and $U$ are allowed in $E_a$. This bipartite property is from the following assumption.

**Assumption 2.** Given an ideal set of anchors $U$, $p(v_j|v_i)$ can be constructed by $\{p(u_i|v_i)\}_{i=1}^{n}$ and $\{p(v_j|u_i)\}_{i=1}^{n}$.

Since both connectivity distributions and anchors are unknown in general data, we need to estimate them. First, we intend to estimate anchors and $p(u_i|v_i)$ alternatively by

$$\min_{U, p(\cdot|v_i)} \sum_{i=1}^{n} \sum_{j=1}^{n} E_{u_j \sim p(\cdot|v_i)} d(v_i, u_j), \quad (3)$$
according to Assumption 1. Note that we only focus on the estimation of \( p(u_j|v_i) \) and overlook \( p(v_i|u_j) \) for now. However, the above problem has a trivial solution,

\[
p_0(u_j|v_i) = \begin{cases} 1, & j = \text{arg min}_j (d(u_j, v_i)) \\ 0, & \text{else} \end{cases}
\]

(4)

To avoid the ill anchor graph, we turn to the following regularized problem,

\[
\min_{u, \pi(\cdot|v_i)} \mathbb{E}_{u_j \sim p(\cdot|v_i)} d(v_i, u_j) + g(p(\cdot|v_i), \pi(\cdot|v_i)),
\]

(5)

where \( \pi(\cdot|v_i) \) represents the uniform distribution and \( g(\cdot, \cdot) \) is a metric of two discrete distributions. Intuitively, \( p(u_j|v_i) \) should be sparse such that anchors far from \( v_i \) are ignored. In practice, Kullback-Leibler divergence (KL-divergence) is usually used as \( g(\cdot, \cdot) \) to measure the divergence of two probability distributions. However, it will return a dense solution of \( p(\cdot|v_i) \). Instead, we employ a simple measure as follows

\[
\min_{u, \pi(\cdot|v_i)} \mathbb{E}_{u_j \sim p(\cdot|v_i)} d(v_i, u_j) + \gamma_i \sum_{j=1}^m (p(u_j|v_i) - \pi(u_j|v_i))^2.
\]

(6)

Let \( d_k(v_i, \cdot) \) be the \( k \)-th smallest value of \( \{d(v_i, u_j)\}_{j=1}^n \), and if \( \gamma_i = \frac{1}{2} (kd_k(v_i, \cdot) - \sum_{l=1}^k d_l(v_i, \cdot)) \), \( p(\cdot|v_i) \) will be \( k \)-sparse (only \( k \) entries are non-zero). Meanwhile, it can be transformed into [30] and solved by

\[
p(u_j|v_i) = \left( \frac{d_k(v_i, \cdot) - d(v_i, u_j)}{\sum_{l=1}^k d_l(v_i, \cdot) - d_l(v_i, u_j)} \right)_{+}.
\]

(7)

The detailed derivations can be found in supplementary.

When \( p(u_j|v_i) \) is fixed, anchors are computed by solving the following problem,

\[
\min_{f(c_j)} \mathbb{E}_{u_j \sim p(\cdot|v_i)} \|f(x_i) - f(c_j)\|^2_2,
\]

(8)

where \( c_j \) is representation of anchor \( u_j \) in the original space. If we take the derivative of the above equation regarding \( f(c_j) \) and set it to 0, then we have

\[
f(c_j) = \frac{\sum_{i=1}^n p(u_j|v_i) f(x_i)}{\sum_{i=1}^n p(u_j|v_i)}.
\]

(9)

In sum, problem (24) can be solved iteratively through Eq. (7) and (9). More details can be found in supplementary.

Then we turn to model \( p(v_i|u_j) \), which is used to compute \( p(v_j|v_i) \), the probability between two raw nodes. Rather than solving a problem like problem (24), \( p(v_i|u_j) \) is set by a simple normalized step,

\[
p(v_i|u_j) = \frac{p(u_j|v_i)}{\sum_{l=1}^n p(u_j|v_l)},
\]

(10)

via utilizing \( p(u_j|v_i) \) calculated by Eq. (7). To show how the above formulations construct a bipartite graph and simplify the succeeding discussion about the fast convolution, we further reformulate \( p(u_j|v_i) \) and \( p(v_i|u_j) \) via matrix form. Let \( B \in \mathbb{R}^{n \times m} \) be a matrix where \( b_{ij} = p(u_j|v_i) \). Then

\[
T = \begin{bmatrix} 0 & B \\ 0 & 0 \end{bmatrix} \text{ and } D_a = \begin{bmatrix} I & 0 \\ 0 & \Delta \end{bmatrix}
\]

(11)

represent an unnormalized bipartite graph and its degree matrix, respectively. Note that \( T \) does not conform to the generative perspective for graphs, though it is a bipartite graph. According to Eq. (10), the generative bipartite graph can be defined as

\[
P = D_a^{-1} T = \begin{bmatrix} 0 & B \\ \Delta^{-1} B^T & 0 \end{bmatrix} \in \mathbb{R}^{(n+m) \times (n+m)},
\]

(12)

which can be also regarded as a probability transfer matrix.

### 3.3. \( O(n) \) Bipartite Graph Convolution with Anchors

Since the above discussion is based on graph models, a multi-layer GCN is a rational scheme to implement \( f(x_i) \) which is defined in Eq. (2). We utilize the bipartite graph, defined in Eq. (11), to accelerate the graph convolution. In virtue of the Markov process [38], \( p(v_j|v_i) \) can be obtained by the one-step transition probability, i.e.,

\[
p(v_j|v_i) = \sum_{l=1}^m p(v_l|v_i) p(u_l|v_i).
\]

Similarly, \( p(u_j|u_i) = \sum_{l=1}^m p(u_j|u_i) p(u_l|u_i) \). Formally, the constructed graphs of \( \mathcal{V} \) and \( \mathcal{U} \) are defined as

\[
A_a = P^2 = \begin{bmatrix} B \Delta^{-1} B^T & 0 \\ 0 & \Delta^{-1} B^T B \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & A_t \end{bmatrix}.
\]

(13)
Algorithm 1 Optimization of AnchorGAE.

Input: Raw features $X$, initial sparsity $k_0$, the estimated upper-bound of sparsity $k_m$, maximum epochs $T$.

1: Initialize $B$ and $C$ via solving problem (24).
2: Initialize $A_u$ by Eq. (13).
3: $k \leftarrow k_0$.
4: for $i = 1, 2, \cdots, T$ do
5: \hspace{0.5cm} Update parameters of GCN layers by gradient descent. \hspace{0.5cm} // Obtain the mapping $f(\cdot)$.
6: \hspace{0.5cm} Update $B$ and $f(c_j)$ on $Z$ and $Z_t$ via problem (24).
7: \hspace{0.5cm} Update anchors in the original space for the bipartite graph convolution: $c_j = f^{-1}(f(c_j)) = X^T b_j$.
8: \hspace{0.5cm} Update $A_u$ by Eq. (13).
9: \hspace{0.5cm} $k \leftarrow k + \lceil \frac{k_m - k_u}{k_m} \rceil$.
10: end for
11: Perform fast clustering on $Z$.

Output: Clustering assignments and embedding $Z$.

Accordingly, $A$ is the adjacency matrix constructed by anchors, of $n$ samples, and $A_t$ is the adjacency of $m$ anchors.

Remark 2. In GCNs, the graph is conventionally preprocessed by adding self-loops as shown in Eq. (1). The graph constructed by Eq. (13) is equivalent to be assigned with adaptive self-loops, since $A_{ii}$ and $(A_t)_{ii}$ are non-zero in most cases.

Therefore, $A$ can be directly used as a part of the graph convolution operator. Since $A_1 = B \Delta^{-1} B^T 1 = B 1 = 1$, the degree matrix of $A$ is the identity matrix, i.e., $A = A$. Therefore, the output of GCN is formulated as

$$H = \varphi(B \Delta^{-1} B^T X W). \quad (14)$$

Here, we explain why the above equation accelerates the operation. It should be emphasized that $A$ is not required to be computed explicitly. The core is to design the order of matrix multiplications as follows

$$T_1 = B^T X \Rightarrow T_2 = \Delta^{-1} T_1$$
$$\Rightarrow T_3 = T_2 W \Rightarrow H = \varphi(B T_3). \quad (15)$$

According to the above order of computation, the computational complexity is reduced to $O(nmd + m^2d + mdd')$. Since $d, d'$, and $m$ are usually much smaller than $n$, the complexity can be regarded as $O(nm)$. Moreover, if the amount of anchors is small enough, the required time of each forward propagation of a GCN layer is $O(n)$.

3.4. Siamese Architecture

As the task is unsupervised, the whole architecture is based on graph auto-encoder (GAE). The encoder consists of multiple bipartite GCN layers defined in Eq. (14),

$$Z = f(X) = \varphi_L(A \varphi_{L-1}(\cdots \varphi_1(A X W_1) \cdots) W_L), \quad (16)$$

where $L$ denotes the amount of layers. The embedding (denoted by $Z$), which is produced by the encoder, is a non-linear mapping for the raw representation of $V$.

However, the mentioned encoder just transforms $V$ and there is no edge between $V$ and $U$. In other words, $U$ can not be projected via the network with the adjacency $A$. Meanwhile, GCN layers suffer from the out-of-sample problem. Accordingly, $U$ can not be directly projected into the deep feature space by the encoder. To map anchors into the same feature space, a siamese [9] encoder is thus designed. The encoder for anchors should share the same parameters but has its own graph structure. Surprisingly, the one-step probability transition matrix defined in Eq. (13) gives us a graph that is only composed of anchors, $A_t = \Delta^{-1} B^T B$. As $A_1 = \Delta^{-1} B^T B 1 = \Delta^{-1} B^T 1 = 1$, the corresponding degree matrix, $D_t$, is also the identity matrix, i.e.,

$$\tilde{A}_t = D_t^{-\frac{1}{2}} A_t D_t^{-\frac{1}{2}} = A_t. \quad (17)$$

Accordingly, the embedding of anchors is formulated as

$$Z_t = f(C) = \varphi_L(A_t \varphi_{L-1}(\cdots \varphi_1(A_t C W_1) \cdots) W_L), \quad (18)$$

where $C \in \mathbb{R}^{m \times d}$ represents $m$ anchors.

Loss: As the underlying connectivity distributions have been estimated, we use them, rather than the adjacency in the existing GAEs, as the target for the decoder to rebuild. According to Assumption 1, the connectivity probability should be reconstructed according to Euclidean distances between $V$ and $U$. Formally speaking, the decoder is

$$q(u_j | v_i) = \frac{\exp(-d(v_i, u_j))}{\sum_{i=1}^m \exp(-d(v_i, u_i))}. \quad (19)$$

Instead of using mean-square error (MSE) to reconstruct $A$, AnchorGAE intends to reconstruct the underlying connectivity distribution and employ the cross-entropy to measure the divergence,

$$\mathcal{L} = \sum_{i=1}^n \sum_{j=1}^m p(u_j | v_i) \log \frac{1}{q(u_j | v_i)}. \quad (20)$$

The GCN part of AnchorGAE is trained via minimizing $\mathcal{L}$.

So far, how to get anchors at the beginning is ignored. In deep learning, a common manner is to pretrain a neural network. Nevertheless, the pretrain of raw graph auto-encoders will result in nearly $O(n^2)$ time, which violates the motivation of our model. Therefore, we first compute anchors and transition probability on the raw features. Then anchors and transition probability will be rectified dynamically, which will be elaborated in the next subsection.

3.5. Adaptive Update to Avoid Degeneration

Since the initialized anchors and transition probability is calculated from the original features, the information hid-
AnchorGAE approximation of the inverse mapping of the encoder. To
the raw input from the embedding can be regarded as an
auto-encoders since the decoder that tries to reconstruct
an invertible mapping. A scheme is to utilize the classi-
cators under the deep representations,
are two problems to address.

Firstly, according to Eq. (7) and (9), we can obtain an-
chains under the learned embedding. There
are two problems to address.

Lastly, according to Eq. (7) and (9), we can obtain an-
chains under the deep representations, \{f(e_j)\}_{j=1}^n. How-
however, the input of our model requires anchors under the
raw features. In other words, it is necessary to compute
\(c_j = f^{-1}(f(e_j))\). Unfortunately, it is hard to ensure \(f(\cdot)
invertible mapping. A scheme is to utilize the clas-
cal auto-encoders since the decoder that tries to reconstruct the
raw input from the embedding can be regarded as an
approximation of the inverse mapping of the encoder.

To simplify the discussion in this paper, we simply set
\(c_j = f^{-1}(f(e_j)) = X^T b_j\),
\(c_j = f^{-1}(f(e_j)) = X^T b_j\),
\(c_j = f^{-1}(f(e_j)) = X^T b_j\)...

to estimate \(c_j\) under the original feature space.

Secondly, the simple update of anchors and transition
probability is unavailable. The performance of AnchorGAE
will become worse and worse after several updates of an-
chors. An interesting phenomenon is that a better recon-
struction indicates a more severe degeneration.

**Theorem 1.** Let \(p_k(\cdot|v_i)\) be the \(k\)-largest one of
\{\(p(u_j|v_i)\)\}_{j=1}^m and \{\(\hat{p}(u_j|v_i)\)\}_{j=1}^m be the updated distribu-
tion with the same \(k\) after step 6 of Algorithm 1. Suppose
that there exists a constant \(s\) such that \(\|f(c_j) - f(x_i)\|_2 \leq s\)
for any \(v_i\) in a cluster with \(\hat{p}(\cdot|v_i)\). If \(|q(\cdot|v_i) - p(\cdot|v_i)| \leq \varepsilon\), then there exists a con-
stant \(c\), which is irrelevant to \(\varepsilon\), such that when \(\varepsilon \leq c\),

\[ |\hat{p}(\cdot|v_i) - \frac{1}{k}| \leq h(\varepsilon) \text{ holds for any } j \leq k \text{ where } h(\cdot) \text{ is a continuous function that satisfies } \lim_{\varepsilon \to 0} h(\varepsilon) = 0. \]

The above conclusion means that \(\hat{p}(\cdot|v_i)\) is a sparse and
uniform distribution. In other words, the constructed graph
degenerates into an unweighted graph, which is usually
unexpected in clustering. The degeneration is caused by
projecting samples from the identical cluster into multiple
tiny and cohesive clusters as shown in Figure 3. These tiny
clusters have no connection with each other and they scatter
randomly in the representation space, such that the clus-
tering result is disturbed. To avoid this collapse, we propose
two strategies: (1) increase \(k\) which represents the sparsity of
neighbors; (2) decrease the amount of anchors, \(m\). In
this paper, we increase \(k\) dynamically for simplicity. Since
the amount of samples in each cluster is unknown, an esti-
ated upper-bound of \(k\) is required. Suppose that there are
\(c\) clusters and the number of samples in the smallest cluster
is represented by \(n_s\). Then we can define the upper-bound of \(k\) as
\(k_m = \left\lfloor m \times \frac{n_s}{m} \right\rfloor.\)

![Figure 3. Illustration of the strategy to avoid degeneration on MNIST-test: The left one is from AnchorGAE with fixed \(k\) while the right one is from AnchorGAE with dynamically increasing \(k\).](image-url)
Let $k_0$ denote the sparsity of the initialized anchor graph and $T$ be the number of iterations to update anchors. Then $k$ can be set adaptively by

$$k \leftarrow k + \left\lfloor \frac{k_m - k_0}{T} \right\rfloor.$$  

(23)

If we have no prior information of $n_s, n_a$ can be simply set as $\lfloor n/c \rfloor$ or $\lfloor n/(2c) \rfloor$. When AnchorGAE is trained, we can perform some fast clustering algorithms on $B$ or $Z$, which is elaborated in supplementary. The optimization of AnchorGAE is summarized in Algorithm 1.

### 4. Experiment

#### 4.1. Datasets and Comparative Methods

The performance of the AnchorGAE is evaluated on 6 non-graph datasets, including 2 UCI datasets (ISOLET [11] and SEGMENT [11]) and four image datasets (USPS [18], MNIST-test [23], MNIST-full [23], and Fashion-MNIST [43]). Note that Fashion-MNIST is denoted by Fashion in Table 2. The details of these datasets are shown in Table 2.

AnchorGAE is compared with 9 methods, including 5 anchor-based methods (Nystrom [12], CSC [37], KASP [45], LSC [3], and SNC [7]), 3 GAE-based methods (SGC [41], GAE [21] and ClusterGCN [8]), 2 deep methods (DEC [44] and SpectralNet [35]), and classical K-Means [28]. All codes are downloaded from the homepages of authors.

Figure 4. The influence of number of anchors to ACC, NMI, and TIME on SEGMENT, USPS, and MNIST-full.

![Graphs showing the influence of number of anchors on ACC, NMI, and TIME for SEGMENT, USPS, and MNIST-full datasets.](image)

Figure 5. Consuming time of AnchorGAE, SGC, GAE, ClusterGCN-L2, and ClusterGNC-L4.

![Bar chart showing consuming time for different datasets and methods.](image)

#### 4.2. Experimental Setup

There are three clustering evaluation criteria used in our experiments, including the clustering accuracy (ACC), normalized mutual information (NMI), and running time. In our experiments, the number of anchor points is set from 100 to 1000 and the increasing step is 100. The best results are reported in Table 1. The activation function of the last layer is set to linear, while the other layers use the ReLU function. Particularly, ClusterGCN recommends 4 layers convolution. To ensure the convolutional structure of ClusterGCN the same as others, we modify the 4 layers of convolutional structure (denoted by ClusterGCN-L4) to 2 layers (denoted by ClusterGCN-L2) and remain 4 layers of the convolutional structure as a competitor. To test the scalability of AnchorGAE in a deeper network, we design 2 layers (denoted by AnchorGAE-L2) and 4 layers (denoted by AnchorGAE-L4) convolutional structure separately. The 2 layers neurons are designed as 128 and 64 or 256 and 32. For the 4 layers neurons, we set the first three layers as 16 and set the last layer as $L+1$. $L$ is the number of labels. The maximum iterations to update network are 200, while the maximum epochs to update anchors are set as 5. The initial sparsity $k_0$ is set as 3.

To investigate the impact of anchors, we run AnchorGAE with diverse amount of anchors and initial sparsity, i.e., $m$ and $k_0$. We also conduct two ablation experiments.
Figure 6. The influence of number of anchors to ACC, NMI, and TIME on ISOLET, MNIST-test, and Fashion-MNIST.

Figure 7. The influence of $k_0$ to ACC and NMI on SEGMENT and MNIST-test.

based on AnchorGAE-L2 to study the impact of different parts. On the one hand, we fix anchors and transition probability during training, which is denoted by Ours-A. On the other hand, we fix the sparsity $k$, which is denoted by Ours-B. All methods are run 10 times and the average results are reported. All experiments are conducted on a PC with an NVIDIA GeForce GTX 1660 GPU SUPER.

4.3. Experimental Results

ACC and NMI are shown in Table 1 while the consuming time is reported in Figure 5. For ACC and NMI, the best results are bolded and suboptimal results are underlined. Especially, some methods either spend too much memory to run on MNIST-full and Fashion-MNIST, or the codes provided by authors fail to run on them. So they are denoted by a horizontal line in Table 1. Similarly, they are represented as $\infty$ in Figure 5. Besides, the visualization of SEGMENT is shown in Figure 2. More experimental figures are shown in supplementary. Figures 4 and 6 show the impact of the number of anchors while the effect of the initial sparsity is illustrated in Figure 7.

From the experimental results, we can conclude that: (1) On almost all datasets, AnchorGAE-L2 obtains the best results in all metrics. On MNIST-full, AnchorGAE obtains the suboptimal NMI, while DEC, which consists of much more layers, outperforms AnchorGAE. Meanwhile, AnchorGAE, with only 2 layers, achieves better ACC than DEC. AnchorGAE needs to compute anchors such that it requires more time. (2) On MNIST-full, normal GAE-based methods fail to run due to out-of-memory since these methods need to save the graph structure in each convolution. AnchorGAE can convolute without constructing a graph explicitly, which not only needs less memory but also reduce consuming time. (3) When sparsity $k$ is fixed or anchors and transition probability (i.e., $B$) is not updated, AnchorGAE usually gets poor results on many datasets. For instance, Ours-B shrinks about 55% in ACC and 55% in NMI on MNIST-full, while Ours-A reduces about 40% in ACC and reduces about 15% in NMI. (4) To ensure fair competition and distinguish between anchor-based methods and GAE-based methods, we set the same number of anchors for AnchorGAE-L4 and other anchor-based methods. Note that the first two layers of AnchorGAE-L4 are same as AnchorGAE-L2. Through observing the results of experiments, AnchorGAE-L4 obtains the acceptable performance in most of the datasets but not the best results. It may be caused by the too small dimension of the final embedding. (5) If the initial sparsity, $k_0$, is set as a too large number, then it results in dense connections. As a result, a great quantity of wrong information is captured.

5. Conclusion

In this paper, we propose an anchor-siamese GCN clustering with adaptive $O(n)$ bipartite convolution (AnchorGAE). The generative perspective for weighted graph helps us to build the relationship between the non-graph-type data and the graph-type data. Since the anchor-based bipartite graph factorizes the adjacency matrix, we can change the order of matrix multiplications to accelerate the graph convolution. Experiments show that AnchorGAE consumes less time especially on MNIST-full and obtains impressive performance. Ablation experiments also support the necessity of updating anchors, transition probability, and sparsity dynamically.
References

[1] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In WWW, pages 1400–1410. ACM / IW3C2, 2020. 2

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013. 1

[3] Deng Cai and Xinlei Chen. Large scale spectral clustering via landmark-based sparse representation. IEEE Trans Cybern, 45(8):1669–1680, 2015. 1, 3, 7

[4] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. 2

[5] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.15068, 2017. 2

[6] Xiaojun Chen, Weijun Hong, Feiping Nie, Joshua Zhexue Huang, and Li Shen. Enhanced balanced min cut. International Journal of Computer Vision, pages 1–14, 2020. 1

[7] Xiaojun Chen, Feiping Nie, Joshua Zhexue Huang, and Min Yang. Scalable normalized cut with improved spectral rotation. In Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017. 1, 3, 7

[8] Wei-Lin Chiang, Xuangqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 257–266, 2019. 2, 7

[9] S. Chopra, R. Hadsell, and Y. Lecun. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), 2005. 5

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems, pages 3844–3852, 2016. 1

[11] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. 7

[12] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(2):214–225, 2004. 1, 3, 7

[13] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In Proceedings of the IEEE international conference on computer vision, pages 5746–5745, 2017. 1, 3

[14] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. IEEE transactions on computer-aided design of integrated circuits and systems, 11(9):1074–1085, 1992. 1, 3

[15] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 1025–1035, 2017. 2

[16] Kaveh Hassani and Amir Hosein Khas Ahmadi. Contrastive multi-view representation learning on graphs. In ICML, 2020. 2

[17] G. E Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006. 2

[18] Jonathan J. Hull. A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(5):550–554, 2002. 7

[19] George Karypis and Vinod Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Siam Journal on Scientific Computing, 20(1):359–0, 1998. 2

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In ICLR, 2014. 2

[21] Thomas N. Kipf and Max Welling. Variational graph autoencoders. 2016. 1, 2, 7

[22] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017. 1, 2, 7

[23] Y Lecun and L Bottou. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. 7

[24] Xuelong Li, Mulin Chen, Feiping Nie, and Qi Wang. Locality adaptive discriminant analysis. In Carles Sierra, editor, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pages 2201–2207, 2017. 1

[25] Xuelong Li, Mulin Chen, Feiping Nie, and Qi Wang. A multiview-based parameter free framework for group detection. In Satinder P. Singh and Shaul Markovitch, editors, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, pages 4147–4153, 2017. 1

[26] Xuelong Li, Hongyuan Zhang, and Rui Zhang. Adaptive graph auto-encoder for general data clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2021. 2

[27] Wei Liu, Junfeng He, and Shih Fu Chang. Large graph construction for scalable semi-supervised learning. In International Conference on Machine Learning, 2010. 1, 3

[28] J Macqueen. Some methods for classification and analysis of multivariate observations. In Proc of Berkeley Symposium on Mathematical Statistics & Probability, 1965. 3, 7

[29] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In Advances in neural information processing systems, pages 849–856, 2002. 1, 3

[30] Feiping Nie, Heng Huang, and Chris Ding. Low-rank matrix recovery via efficient schatten p-norm minimization. In Twenty-sixth AAAI conference on artificial intelligence, 2012. 1, 4, 11

[31] Feiping Nie, Xiaqian Wang, Cheng Deng, and Heng Huang. Learning a structured optimal bipartite graph for co-clustering. In Advances in Neural Information Processing Systems, pages 4129–4138, 2017. 14
[32] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, pages 2609–2615, 2018. 1, 2

[33] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6519–6528, 2019. 1, 2

[34] Xi Peng, Jiashi Feng, Shijie Xiao, Wei-Yun Yau, Joey Tianyi Zhou, and Songfan Yang. Structured autoencoders for subspace clustering. *IEEE Transactions on Image Processing*, 27(10):5076–5086, 2018. 1

[35] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587*, 2018. 7

[36] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 1, 3

[37] Hiroyuki Shinnou and Minoru Sasaki. Spectral clustering for a large data set by reducing the similarity matrix size. In *International Conference on Language Resources & Evaluation*, 2008. 1, 3, 7

[38] Frank Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2013. 4

[39] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: a deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3670–3676. AAAI Press, 2019. 1

[40] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017. 1

[41] Felix Wu, Tianyi Zhang, Amaur Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *Proceedings of Machine Learning Research*, 2019. 2, 7

[42] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2020. 2

[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv: 1708.07747*, 2017. 7

[44] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016. 1, 7

[45] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2009. 1, 3, 7

[46] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. 1, 3

[47] Rui Zhang, Xuelong Li, Hongyuan Zhang, and Feiping Nie. Deep fuzzy k-means with adaptive loss and entropy regularization. *IEEE Transactions on Fuzzy Systems*, 28(11):2814–2824, 2020. 1

[48] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao Ming Wu. Attributed graph clustering via adaptive graph convolution. In *28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 4327–4333, 2019. 2
A. Derivation of Equation (7) in Main Paper

Specifically, the problem regarding \( p(\cdot|v_i) \) can be separated into \( n \) subproblems,

\[
\min_{p(\cdot|v_i)} \mathbb{E}_{u_j \sim p(\cdot|v_i)} d(v_i, u_j) + \gamma \sum_{j=1}^{m} (p(u_j|v_i) - \frac{1}{m})^2, \tag{24}
\]

where \( \pi(u_j|v_i) = \frac{1}{m} \). To simply the discussion, the subscript \( i \) is neglected. Let \( p = \{p(u_1|v), p(u_2|v), \ldots, p(u_m|v)\} \) and \( d = [d(v, u_1), d(v, u_2), \ldots, d(v, u_m)] \) Accordingly, the above problem is formulated as

\[
\min_{p^T \mathbf{1} = 1} p^T \mathbf{d} + \gamma \sum_{j=1}^{m} (p_j - \frac{1}{m})^2. \tag{25}
\]

Note that

\[
\sum_{j=1}^{m} (p_j - \frac{1}{m})^2 = \|p\|^2 + \frac{1}{m} - 2 \sum_{j=1}^{m} \frac{p_j}{m} = \|p\|^2 - \frac{1}{m}. \tag{26}
\]

Therefore, problem (25) is equivalent to [30]

\[
\min_{p^T \mathbf{1} = 1} p^T \mathbf{d} + \gamma \|p\|^2. \tag{27}
\]

The Lagrangian of the above problem is

\[
\mathcal{L} = p^T \mathbf{d} + \gamma \|p\|^2 + \xi (1 - \sum_{i=1}^{n} \alpha_i) - \sum_{i=1}^{n} \beta_i \alpha_i, \tag{28}
\]

where \( \xi \) and \( \beta_i \) are Lagrangian variables. The KKT conditions are given as

\[
\begin{cases}
\frac{\partial \mathcal{L}}{\partial p_j} = d_j + 2 \gamma p_j - \xi - \beta_i = 0 \\
\beta_j p_j = 0 \\
\sum_{j=1}^{n} p_j = 1, \beta_j \geq 0, p_j \geq 0.
\end{cases} \tag{29}
\]

Then we consider the following cases

\[
\begin{cases}
p_j = 0 \Rightarrow \xi - d_j = -\beta_j \leq 0 \\
p_j \geq 0 \Rightarrow p_j = \frac{1}{2\gamma} (\xi - d_j) \tag{30}
\end{cases}
\]

which means

\[
p_j = \frac{(\xi - d_j)_+}{2\gamma}. \tag{31}
\]

Without loss of generality, assume that \( d_1 \leq d_2 \leq \cdots \leq d_m \). If \( \xi - d_{k+1} \leq 0 < \xi - d_k \), then \( \alpha \) is \( k \)-sparse. Due to \( \sum_{j=1}^{m} p_j = 1 \), we have

\[
\sum_{j=1}^{n} p_j = \frac{1}{2\gamma} (k\xi - \sum_{i=1}^{k} d_j) = 1, \tag{32}
\]

which means

\[
\xi = \frac{1}{k} \sum_{i=1}^{k} f_i + \frac{2\gamma}{k}. \tag{33}
\]

Combine with our assumption and we have

\[
\frac{1}{k} \sum_{j=1}^{k} d_j + \frac{2\gamma}{k} - d_{k+1} \leq 0 < \frac{1}{k} \sum_{i=1}^{k} d_j + \frac{2\gamma}{k} - d_k \\
\Rightarrow d_k < \frac{1}{k} \sum_{j=1}^{k} d_j + \frac{2\gamma}{k} \leq d_{k+1} \\
\Rightarrow \frac{kd_k}{2} - \frac{1}{2} \sum_{j=1}^{k} d_j < \gamma \leq \frac{kd_{k+1}}{2} - \frac{1}{2} \sum_{j=1}^{k} d_j. \tag{34}
\]

Hence, if \( \gamma \) is set within the above range, then \( \alpha \) will be \( k \)-sparse. In other words, \( \lambda \) is converted into the amount of neighbors \( k \).

If we simply set \( \lambda \) as its upper bound, i.e.,

\[
\gamma = \frac{kd_{k+1}}{2} - \frac{1}{2} \sum_{j=1}^{k} d_j, \tag{35}
\]

we have

\[
p_j = \left( \frac{\sum_{i=1}^{k} d_j + 2\gamma}{2k\gamma} - \frac{d_j}{2\gamma} \right)_+ = \left( \frac{d_{k+1} - d_j}{kd_{k+1} - \sum_{i=1}^{k} d_i} \right)_+. \tag{36}
\]

The main body of the derivation is from [30].

B. Explanation of Equation (9) in Main Paper

Take the derivative of Eq. (8) regarding \( f(c_j) \) and we have

\[
\nabla f(c_j) \mathbb{E}_{u_j \sim p(\cdot|v_i)} f(x_i) - f(c_j) \|^2 \\
= 2\left( \sum_{i=1}^{m} p(u_j|v_i) f(c_j) - p(u_j|v_i) f(x_i) \right). \tag{37}
\]

Set it to 0,

\[
f(c_j) = \frac{\sum_{i=1}^{m} p(u_j|v_i) f(x_i)}{\sum_{i=1}^{m} p(u_j|v_i)}. \tag{38}
\]

C. How to Solve Problem (7)

Problem (7) can be solved by the alternative method. Algorithm 2 summarizes the process.

D. Proof of Theorem 2

For an arbitrary probability distribution \( \{p(u_j|v_i)\}_{j=1}^{m} \), let \( p_k(\cdot|v_i) \) be the \( k \)-largest one. For simplicity, let \( d_{ij} =
Algorithm 2 Iterative update of the connectivity distribution and anchors, i.e., details of Step 6 of Algorithm 1 in main paper.

Input: \( \{f(x_i)\}_{i=1}^m, \{f(c_j)\}_{j=1}^m \), sparsity \( k \).

\( t = 0, f^{(0)}(u_j) = f(c_j), d^{(0)}(v_i, u_j) = \|f(x_i) - f^{(0)}(u_j)\|_2^2 \).

repeat

\( t = t + 1 \).

\[
p^{(t)}(u_j|v_i) = \left( \frac{d^{(t-1)}(v_i, c_j) - d^{(t-1)}(v_i, u_j)}{\sum_{j'=1}^m d^{(t-1)}(v_i, c_j) - d^{(t-1)}(v_i, u_{j'}))} \right)^+.\]

\[
f^{(t)}(c_j) = \frac{\sum_{i=1}^m p^{(t)}(u_j|v_i) f(x_i)}{\sum_{i=1}^m p^{(t)}(u_j|v_i)}.
\]

\[
d^{(t)}(v_i, u_j) = \|f(x_i) - f^{(t)}(u_j)\|_2^2.
\]

until convergence.

Output: \( f^{(t)}(c_j) \) and \( p^{(t)}(\cdot|v_i) \).

\[
d(u_j, v_i) = \|f(x_i) - f(c_j)\|_2^2 \] where \( f(\cdot) \) represents the mapping function implemented by the trained GCN encoder. Note that the connectivity distribution is estimated through the current representations, i.e.,

\[
p^{(0)}(\cdot|v_i) = \arg\min_{p(\cdot|v_i)} \mathbb{E}_{u_i \sim p(\cdot|v_i)} \hat{d}_{ij} + \gamma_1 \sum_{j=1}^m (p(u_j|v_i) - \frac{1}{m})^2.
\]

(39)

where \( \hat{d}_{ij} = \|\hat{f}(x_i) - f(c_j)\|_2^2 \) and \( \hat{f}(\cdot) \) denotes the mapping function before the new training of GCN encoders.

We define \( p^{(t)}(u_j|v_i) \) and \( f^{(t)}(c_j) \) as the solutions at the \( t \)-th iteration. To prove the theorem clearly, the iterative update of \( p(\cdot|v_i) \) and anchors is shown in Algorithm 2. The following lemma aims to depict the first estimated connectivity distribution after one perfect training of GCNs.

Lemma 1. Suppose that \( p^{(1)}(\cdot|v_i) = \arg\min_{p(\cdot|v_i)} \mathbb{E}_{u_i \sim p(\cdot|v_i)} d_{ij} + \gamma_1 \sum_{j=1}^m (p(u_j|v_i) - \frac{1}{m})^2 \) where \( \gamma_1 \) ensures the sparsity as \( k \). If \( |p^{(0)}(\cdot|v_i) - q_j(\cdot|v_i)| \leq \varepsilon \) and \( p_k(\cdot|v_i) \geq \sqrt{\varepsilon} \) for any \( i \), then for any \( j \),

\[
|p^{(1)}_j(\cdot|v_i) - \frac{1}{k}| \leq \frac{1}{k} \cdot \frac{1}{\log(\sqrt{\varepsilon} - 1)} - 1.
\]

(40)

Proof. Without loss of generality, we focus on the connectivity distribution of \( u \) and suppose that \( p^{(0)}(u_1|v) \geq p^{(0)}(u_2|v) \geq \cdots \geq p^{(0)}(u_k|v) \geq 0 = p^{(0)}(u_{k+1}|v) = \cdots = p^{(0)}(u_m|v) \). Let \( p_i = p^{(0)}(u_i|v) \) and \( q_i = q(u_i|v) \). According to the definitions,

\[
q_i = \frac{\exp(-d_j)}{\sum_{j=1}^m \exp(-d_j)}.
\]

(41)

If for any \( i \), we have \( |p_i - q_i| \leq \varepsilon \). Clearly, \( p_{k+1} = 0 \). Suppose that \( q_{k+1} = \tau \leq \varepsilon \). Therefore, we have

\[
\frac{\exp(-d_{k+1})}{\sum_{j=1}^m \exp(-d_j)} = \tau \Leftrightarrow d_{k+1} = \log \frac{1}{\tau} - \log C.
\]

(42)

where \( C = \sum_{j=1}^m \exp(-d_j) \). Combine with the condition, \( p_k \geq \sqrt{\varepsilon} \), and we have

\[
\frac{\exp(-d_k)}{\sum_{j=1}^m \exp(-d_j)} \geq p_k - \varepsilon \geq \sqrt{\varepsilon} - \varepsilon \Rightarrow d_k \leq -\log C - \log(\sqrt{\varepsilon} - \varepsilon).
\]

(43)

Similarly, since \( p_1 \geq \frac{1}{k} \),

\[
\frac{\exp(-d_1)}{\sum_{j=1}^m \exp(-d_j)} \leq 1 \Rightarrow d_1 \geq -\log C.
\]

(44)

If we update the connectivity distribution based on \( d^{(t)}(\cdot|v_i) \), then for any \( i \leq k \),

\[
p^{(1)}_i = \frac{d_{k+1} - d_i}{\sum_{j=1}^k (d_{k+1} - d_j)}.
\]

(45)

Furthermore, for any \( i, j \leq k \),

\[
|p^{(1)}_i - p^{(1)}_j| = \frac{|d_j - d_i|}{\sum_{j=1}^k (d_{k+1} - d_j)} \leq \frac{|d_k - d_i|}{\sum_{j=1}^k (d_{k+1} - d_j)} \leq \frac{-\log(\sqrt{\varepsilon} - \varepsilon)}{\sum_{j=1}^k (\log \frac{1}{t} - \log C - d_j)} \leq \frac{-\log(\sqrt{\varepsilon} - \varepsilon)}{\sum_{j=1}^k (\log \frac{1}{t} - \log C - d_k)} \leq \frac{1}{k} \cdot \frac{1}{\log(\sqrt{\varepsilon} - \varepsilon)} - 1 \leq \frac{1}{k} \cdot \frac{1}{\log(\sqrt{\varepsilon} - \varepsilon)} - 1.
\]

(46)

With \( \varepsilon \to 0 \), \( |p^{(1)}_i - p^{(1)}_j| \to 0 \).

The proof is easy to extend to other nodes. Hence, the theorem is proved.

The following lemma shows that the updated anchors will approximate the mean vectors of the connected nodes if \( p^{(t)}(u_j|v_i) \) approaches the sparse and uniform distribution.

Lemma 2. Let \( N_j \) denote the nodes that connect the anchor \( u_j \) and \( p(u_j|v_i) \) represent a \( k \)-sparse solution given by Theorem 1. If \( |p_j(\cdot|v_i) - \frac{1}{k}| \leq \varepsilon \) for any \( j \leq k \), then there exists a function \( h(\cdot) \) such that

\[
\|\sum_{i=1}^n \frac{p(u_j|v_i) f(x_i)}{\sum_{i=1}^n p(u_j|v_i)} - \frac{1}{n_j} \sum_{i \in N_j} f(x_i)\|_2^2 \leq h(\varepsilon),
\]

(47)

where \( n_j = |N_j| \) and \( h(\cdot) \) satisfies that \( \lim_{\varepsilon \to 0} h(\varepsilon) = 0 \).
Proof. To keep the notation uncluttered, let \( f_i = f(x_i) \) and
\[
\mathbf{h}_j = \sum_{i=1}^{n_j} p(u_j|v_i) f(x_i) \frac{p(u_j|v_i)}{\sum_{i=1}^{n_j} p(u_j|v_i)}. \tag{48}
\]
Note that \( \sum_{i=1}^{n_j} p(u_j|v_i) f(x_i) = \sum_{i\in\mathcal{N}_j} p(u_j|v_i) f(x_i) \).

Therefore, we have
\[
\|\mathbf{h}_j - \frac{1}{n_j} \sum_{i\in\mathcal{N}_j} \mathbf{f}_i\|_2 = \frac{1}{n_j} \sum_{i\in\mathcal{N}_j} \|p(u_j|v_i) - \frac{s_j}{n_j}\|_2, \tag{49}
\]
where \( s_j = \sum_{i=1}^{n_j} p(u_j|v_i) \). According to the condition, we have
\[
\frac{1}{k} - \varepsilon \leq p(u_j|v_i) \leq \frac{1}{k} + \varepsilon
\]
\[
\Rightarrow \frac{n_j}{k} - n_j \cdot \varepsilon \leq s_j \leq \frac{n_j}{k} + n_j \cdot \varepsilon. \tag{50}
\]

Accordingly, we have
\[
\|\mathbf{h}_j - \frac{1}{n_j} \sum_{i\in\mathcal{N}_j} \mathbf{f}_i\|_2
\]
\[
\leq \sum_{i\in\mathcal{N}_j} \frac{1}{s_j} |p(u_j|v_i) - \frac{s_j}{n_j}| \cdot \|\mathbf{f}_i\|_2
\]
\[
\leq \sum_{i\in\mathcal{N}_j} \frac{\|\mathbf{f}_i\|_2}{s_j} \max\{|p(u_j|v_i) - \frac{1}{k} + \varepsilon|, |p(u_j|v_i) - \frac{1}{k} - \varepsilon|\}
\]
\[
\leq \sum_{i\in\mathcal{N}_j} \frac{\|\mathbf{f}_i\|_2}{s_j} (|p(u_j|v_i) - \frac{1}{k}| + |\varepsilon|)
\]
\[
\leq \sum_{i\in\mathcal{N}_j} \frac{2\|\mathbf{f}_i\|_2}{s_j} \varepsilon. \tag{51}
\]

Hence, the theorem is proved. \( \square \)

With the following lemma, we can prove Theorem 2 via mathematical induction method.

Lemma 3. Suppose that \( q_j(\cdot|v_i) \leq \varepsilon \) for \( j > k \), and \( d(v_i, u_j) \leq s \). Then there exists \( c \) such that if \( \varepsilon \leq c \) and \( v_i \) does not connect with \( u_j \), then for any \( v' \in \mathcal{N}_j \), \( v_i \) and \( v' \) share no anchors.

Proof. At first, let \( \varepsilon' \leq \min[p_k(\cdot|v_i)]^2 \). Hence, \( p_k(\cdot|v_i) \geq \frac{\varepsilon}{\varepsilon'} \). For an arbitrary node \( v_i \) connected with \( u_a \) and \( u_b \) and an anchor \( u_c \) which disconnects with \( v_i \), suppose that there exists a node \( v_j \) such that \( v_j \) connects with \( u_b \) and \( u_c \) but disconnects with \( u_a \). According to the triangular inequality, we have
\[
\frac{\exp(-d(u_a, u_c))}{\exp(-d(v_i, u_a))} \leq \frac{\exp(-d(u_b, u_c))}{\exp(-d(v_i, u_a))}
\]
\[
= \frac{\exp(-d(v_i, u_a))}{\exp(-2d(v_i, u_a))}
\]
\[
= \frac{q(u_c|v_i)}{q(u_a|v_i) \exp(-d(v_i, u_a))}
\]
\[
\leq \frac{\varepsilon}{\varepsilon' + \varepsilon^s}. \tag{52}
\]

Let \( g(\varepsilon) = \frac{\varepsilon}{\varepsilon' + \varepsilon^s} \). According to the triangular inequality,
\[
d(v_i, u_a) + \log \frac{1}{g(\varepsilon)} - L
\]
\[
\leq d(v_a, u_c) < d(u_a, u_b) + d(u_b, u_c)
\]
\[
< d(v_i, u_a) + d(v_i, u_b) + d(v_j, u_c) + d(v_j, u_b)
\]

Therefore,
\[
\log \frac{1}{g(\varepsilon)} < 4s \tag{53}
\]

If we define \( c \) as
\[
c = \min\{g^{-1}(\exp(-4s)), c'\}, \tag{55}
\]
the Ineq. (54) results in a contradiction when \( \varepsilon \leq c \). \( \square \)

Proof of Theorem 2. The proof is to analyze each iteration of step 6. Ensure \( c \leq \min[p_k(\cdot|v_i)]^2 \) and we can obtain that \( |p_j^{(1)}(\cdot|v_i) - \frac{1}{k}| \leq h_1(\varepsilon) \) according to Lemma 1. Certainly, the conclusion construct the preconditions of Lemma 2.

Let \( \mu_j = \frac{1}{|\mathcal{N}_j|} \sum_{x_i \in \mathcal{N}_j} f(x_i) \). According to Lemma 2, we have \( \|f^{(1)}(c_j) - \mu_j\|_2^2 \leq h_2(\varepsilon) \) and we can further derive that
\[
\|f(x_i) - f^{(1)}(c_j)\|_2^2 \leq \|f(x_i) - \mu_j\|_2^2 + \|f^{(1)}(c_j) - \mu_j\|_2^2
\]
\[
\leq \|f(x_i) - \mu_j\|_2^2 + \sqrt{h_2(\varepsilon)}, \tag{56}
\]

and
\[
\|f(x_i) - f^{(1)}(c_j)\|_2^2 \geq \|f(x_i) - \mu_j\|_2^2 - \|f^{(1)}(c_j) - \mu_j\|_2^2
\]
\[
\geq \|f(x_i) - \mu_j\|_2^2 - \sqrt{h_2(\varepsilon)}. \tag{57}
\]

Therefore, for \( d^{(1)}(v_i, u_j) \), we have
\[
h_2(\varepsilon) - 2\sqrt{h_2(\varepsilon)} \||f(x_i) - \mu_j\|_2
\]
\[
\leq d^{(1)}(v_i, u_j) - \|f(x_i) - \mu_j\|_2^2 \tag{58}
\]
\[
\leq h_2(\varepsilon) + 2\sqrt{h_2(\varepsilon)} \|f(x_i) - \mu_j\|_2.
\]

Again, the above conclusions are provided as the preconditions of Lemma 3. Based on Lemma 2 and 3, we know that for any \( l \leq k \),
\[
d^{(1)}_{k+1}(v_i, \cdot) - d^{(1)}_{k+1}(v_i, \cdot) \leq d^{(1)}_{k+1}(v_i, \cdot) - c_1^2
\]
\[
- h_2(\varepsilon) + 2\sqrt{h_2(\varepsilon)c_1}, \tag{59}
\]

here \( c_1 \) is the minimum value of the function \( g(\varepsilon) = \frac{\varepsilon}{\varepsilon' + \varepsilon^s} \) for \( \varepsilon \leq c \).
and
\[ d_{k+1}^{(1)}(v_i, \cdot) - d_{l}^{(1)}(v_i, \cdot) \geq d_{k+1}^{(1)}(v_i, \cdot) - c_1^2 - h_2(\varepsilon) - 2\sqrt{h_2(\varepsilon)}c_1 \quad (60) \]

Furthermore, as
\[ p^{(2)}(u_j | v_i) = \frac{(d_{k+1}^{(1)}(v_i, \cdot) - d_{l}^{(1)}(v_i, \cdot))_+}{\sum_{i=1}^{k}(d_{k+1}^{(1)}(v_i, \cdot) - d_{l}^{(1)}(v_i, \cdot))} \]
\[ \leq \frac{d_{k+1}^{(1)}(v_i, \cdot) - c_1^2 - h_2(\varepsilon) + 2\sqrt{h_2(\varepsilon)}c_1}{\sum_{i=1}^{k}[d_{k+1}^{(1)}(v_i, \cdot) - c_1^2] - kh_2(\varepsilon) - 2kc_1\sqrt{h_2(\varepsilon)}} \quad (61) \]

we have
\[ p^{(2)}(u_j | v_i) \leq \frac{4c_1\sqrt{h_2(\varepsilon)}}{k[d_{k+1}^{(1)}(v_i, \cdot) - c_1^2 - h_2(\varepsilon) - 2c_1\sqrt{h_2(\varepsilon)}]} \quad (62) \]

Similarly, we can also infer that
\[ p^{(2)}(u_j | v_i) \geq \frac{-4c_1\sqrt{h_2(\varepsilon)}}{k[d_{k+1}^{(1)}(v_i, \cdot) - c_1^2 - h_2(\varepsilon) + 2c_1\sqrt{h_2(\varepsilon)}]} \quad (63) \]

Let
\[ h_3(\varepsilon) = \frac{4c_1\sqrt{h_2(\varepsilon)}}{k[d_{k+1}^{(1)}(v_i, \cdot) - c_1^2 - h_2(\varepsilon) - 2c_1\sqrt{h_2(\varepsilon)}]} \quad (64) \]
then there exists a constant \( c \) such that when \( \varepsilon \leq c \),
\[ p^{(2)}(u_j | v_i) \] approximate the sparse and uniform distribution,
\[ |p^{(2)}(u_j | v_i) - \frac{1}{k}| \leq h_3(\varepsilon) \quad (65) \]
We can repeat the above proof for every iteration to update \( p^{(1)}(u_j | v_i) \) and \( f^{(1)}(c_j) \), and therefore, the theorem is proved.

E. Obtain Clustering Assignments

After training AnchorGAE, we can simply run \( k \)-means on the learned embedding. Due to Assumption 1, Euclidean distances of similar points will be small such that it is appropriate to use \( k \)-means on the deep representations.

Another method is to perform spectral clustering on the constructed graph, which is same as anchor-based spectral clustering. Let
\[ L = I - D^{-\frac{1}{2}}A D^{-\frac{1}{2}} = I - A = I - B \Delta^{-\frac{1}{2}}B^T \quad (66) \]
be the normalized Laplacian matrix. It should be emphasized that \( L \) is also the unnormalized Laplacian matrix. Accordingly, the objective of spectral clustering is
\[ \max_{F^T F = I} \text{tr}(F^T B \Delta^{-\frac{1}{2}}B^T F), \quad (67) \]
where \( F \in \mathbb{R}^{n \times c} \) is the soft indicator matrix. Let \( \hat{B} = B \Delta^{-\frac{1}{2}} \). Then the eigen-value decomposition of \( A \) can be transformed into the singular value decomposition of \( B \).

Finally, one can perform clustering on the bipartite graph directly rather than the constructed graph. Let the bipartite graph be
\[ W = \left[ \begin{array}{cc} 0 & B^T \\ B & 0 \end{array} \right] \in \mathbb{R}^{(n+m) \times (n+m)}. \quad (68) \]
The degree matrix is represented as
\[ D = \left[ \begin{array}{cc} D_v & 0 \\ 0 & D_u \end{array} \right]. \quad (69) \]
Then the normalized cut used spectral clustering is to optimize
\[ \max_{F^T F = I} \text{tr}(F^T D^{-\frac{1}{2}}W D^{-\frac{1}{2}}F^T), \quad (70) \]
where
\[ F = \left[ \begin{array}{c} V \\ U \end{array} \right]. \quad (71) \]
Since \( D_v = I \), the above problem can be further reformulated as
\[ \max_{U^T U + V^T V = I} \text{tr}(V^T B D_u^{-\frac{1}{2}}U). \quad (72) \]
The closed-form solution can be calculated by
\[ \left\{ \begin{array}{l} V = \frac{1}{\sqrt{2}} \tilde{V} \\ U = \frac{1}{\sqrt{2}} \tilde{U} \end{array} \right. \quad (73) \]
where \( \tilde{V} \) and \( \tilde{U} \) are \( c \) leading left and right singular vectors of \( B D_u^{-\frac{1}{2}} \). The details can be found in [31].

F. Limitations

In AnchorGAE, the original images of anchors are estimated roughly. In other words, it may be imprecise to use \( f^{-1}(f(c_j)) = X^T b_j \) to estimate \( f^{-1}(\cdot) \). It may result in the performance bottleneck of AnchorGAE. Therefore, we will focus on how to exactly compute the original images from the embedding space in future work. To further speed up the model, it is worthy to develop stochastic gradient descent for AnchorGAE in the future.