Adaptive Hypergraph Auto-Encoder for Relational Data Clustering

Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, Yining Zhao, Chenggang Yan, Jian Yin, and Yue Gao, *Senior Member, IEEE*

Abstract—The embedded representation and clustering tasks both play important roles in relational data analysis and mining. Traditional methods mainly employ graph structure to describe relational data, but intuitive pairwise connections among nodes are insufficient to model high-order data in the real-world, such as the relations between proteins and polypeptide chains. Hypergraphs are a generalization of graphs, and hypergraphs can well model high-order data. When modeling relational data in the real world, hypergraphs are often accompanied by node attributes, i.e., attributed hypergraphs. Besides this, how to integrate the structural information and attribute information appropriately is another important task, while has not been investigated systematically. In this paper, we propose Adaptive Hypergraph Auto-Encoder(AHGAE) to learn node embeddings in low-dimensional space. Our method can utilize the high-order relation to generate embedding for clustering. It is composed of two procedures, i.e., the adaptive hypergraph Laplacian smoothing filter and the relational reconstruction auto-encoder. It has the advantage of integrating more complex data relations compared with graph-based methods, which leads to better modeling and clustering performance. The proposed method has been evaluated on hypergraph datasets and benchmark graph datasets. Experimental results and comparison with the state-of-the-art methods have demonstrated the effectiveness of our proposed method.

Index Terms—Hypergraph, clustering, representation learning, neural network

1 Introduction

NON-EUCLIDEAN data frequently appears in our lives, which can be described with relations, i.e., relational data. Graph structures, whose nodes represent entities and edges represent relations between entities, usually are utilized to describe the relational data. With the development of deep learning, the Graph Neural Network (GNN), a general term for a series of algorithms based on deep learning in the field of graphs, is proposed. GNN has efficient performance [1], a series of representative models [2], [3], [4] and many meaningful applications [5], [6].

However, each edge in graphs only connects two nodes, which limits the graph's representation ability. It is the characteristic that leads the simple graph structure unable to represent high-order relational data. Suppose we need to describe the relationship between the authors of some papers, i.e., the co-author network, we can naturally use edges to connect authors who once had cooperative relationships, thus form the graph structure, as shown in Fig. 1a. However, the relationships between the authors in the source data are described as the author lists of the paper, and we inevitably lose the association information between authors and papers. In order to completely retain this high-order information, we need to get rid of the limitation of edge degree, i.e., each edge can be permitted to connect two or more nodes, so we can use the edge to represent the paper and the sub-nodes of the edge to represent its authors, as shown in Fig. 1b. Such structures that do not limit the degree of nodes are called as hypergraphs [7], and such edges are called as hyperedges. Hypergraphs, as high-order representations of graphs, can well model high-order data without loss of information [8].

Besides, hypergraphs are generalizations of graphs, so many graph-related problems can also be solved using hypergraph structure. In general, hypergraphs have a wider perceptual domain and more reasonable interpretability in many scenes, so the introduction of hypergraph structures can obtain better performance in certain graph tasks[9]. It is because of the powerful capabilities of hypergraphs that they have received increasing attention to applications, such as the visual tasks [10] and the recommender systems [11], etc.

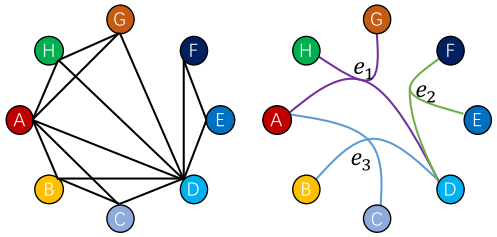
In the real-world, Hypergraph data is often accompanied by node attributes, namely, attributed hypergraphs, which have two accepted assumptions:

- Nodes in the same hyperedge have similar attributes.
- Nodes with similar features have similar attributes.

In other words, nodes belonging to the same hyperedge are more inclined to have the same preference. There are many interesting applications based on these assumptions, such as recommendations for products or users, and online

- *Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, and Jian Yin are with the School of Mechanical and Information Engineering, Shandong University, Weihai 264209, China. E-mail: {yooooooohu, iswangyj.cn}@gmail.com, {lxk_yb, beixuan_wh}@163.com, yinjian@sdu.edu.cn.*
- *Yining Zhao and Yue Gao are with the BNRist, THUIBCS, KLISS, School of Software, Tsinghua University, Beijing 100084, China. E-mail: zhaoyn17@mails.tsinghua.edu.cn, kevin.gaoy@gmail.com.*
- *Chenggang Yan is with the School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China, and also with the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China. E-mail: cgyan@hdu.edu.cn.*

Manuscript received 5 November 2020; revised 21 June 2021; accepted 24 August 2021. Date of publication 30 August 2021; date of current version 3 February 2023.
(Corresponding authors: Yue Gao and Chenggang Yan.)
Recommended for acceptance by G. Chen.
Digital Object Identifier no. 10.1109/TKDE.2021.3108192



(a) Graph (b) Hypergraph

Fig. 1. The graph and hypergraph representation of co-author network. (a) represents the graph structure connecting all the cooperating authors. (b) represents the hypergraph structure that uses the association information between authors and papers to describe the cooperative relationship, where e_1, e_2 and e_3 respectively represent different papers.

personalized services. Meanwhile, it derives many based tasks: clustering, representation learning, link prediction or classification. In recent years, with the gradual improvement of hypergraph theory, some works related to hypergraph learning have gradually attracted attention [12], [13], [14], [15]. However, the related works for clustering tasks still are little exploration.

Inspired by researches related to representation learning and clustering tasks in the graph field, we propose an Adaptive Hypergraph Auto-Encoder (AHGAE) model for hypergraph clustering tasks, which is also perfectly compatible with graph clustering tasks. Unlike other graph-based deep clustering or autoencoder models, the design of the AHGAE model consists of two steps, which can be understood as a decoupling operation, as shown in Fig. 2. For the first step, the hypergraph Laplacian smoothing filters are utilized to integrate node information and its neighbor information by the association information between hyperedges and nodes, and adaptively select the optimal order to form node representations optimized for clustering tasks. For the second step, the relationship reconstruction auto-encoder focuses on the relations between node features and learns the appropriate low-dimensional node embeddings while retaining the hypergraph structure.

In summary, the main contributions of this paper are as follows:

- A hypergraph Laplacian smoothing filter is proposed, which achieves the effect of smoothing node features by fusing their features and the adjacent features in the same hyperedges. Its purpose is to better reduce the impact of high-frequency noise and enhance more essential attributes. We provide its spatial derivation and analyze its filtering characteristics in the frequency domain.
- An Adaptive Hypergraph Auto-Encoder (AHGAE) is proposed, which is an embedded model specifically for hypergraph clustering tasks. Our model can adaptively integrate the node and structural information in the hypergraph to generate the node embeddings. Because of the superior representation ability of hypergraphs, our model is well compatible with the attributed graph.
- We construct an attributed hypergraph dataset DBLP-HG provided by DBLP official website. And through clustering experiments on the DBLP-HG dataset and some benchmark graph datasets, the considerable performance of our proposed model has been verified.

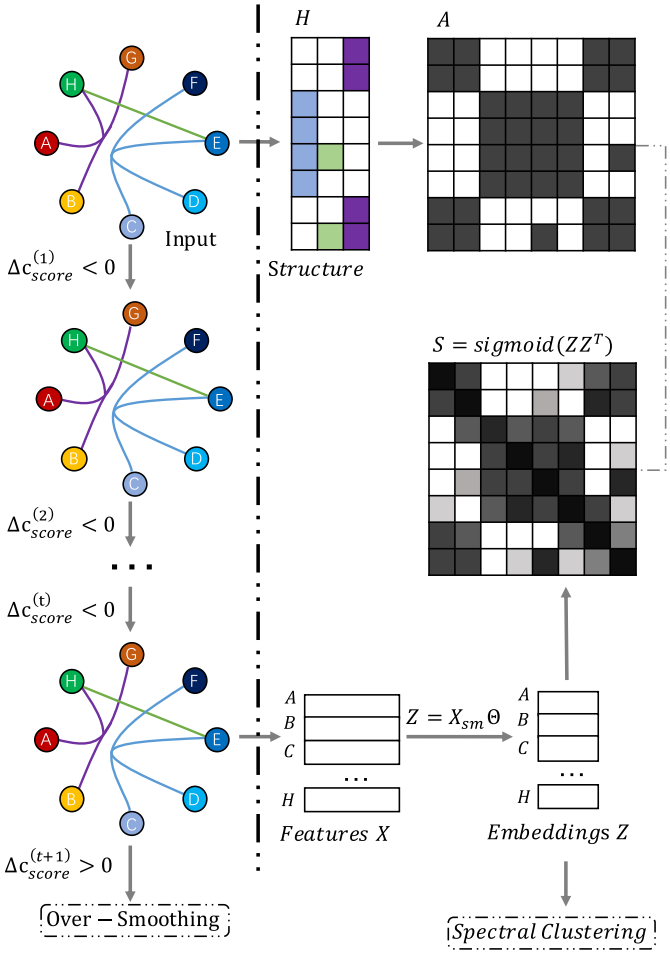


Fig. 2. Adaptive Hypergraph Auto-Encoder framework. First, the optimal smoothing feature of each node is obtained through an adaptive high-order hypergraph adaptive filter. The node features incorporating structural information are obtained through a simple relational reconstruction auto-encoder. $\Delta c_{score}^{(t)}$ represents the change value of the unlabeled clustering metric DBI, which is used to select the optimal order t.

2 Related Work

According to the data information used by models, graph clustering tasks are divided into structural graph clustering tasks and attributed graph clustering tasks.

Structural graph clustering tasks only utilize structural information. The spectral clustering [16] method directly takes the graph structure as input, and by slicing the graph, the weights of the edges between the different sub-graphs after the cut are as low as possible to achieve the purpose of clustering. [17] proposes to decompose the adjacency matrix of the nodes into node representations, and then uses K-means or other methods to obtain the clustering result. DeepWalk [18] is a method that uses SkipGram to learn node representations by randomly walking on the graph and maximizing the probability of each node's neighborhood, and then obtains clustering results. Moreover, some methods based on auto-encoder [19] are used for the structural graph clustering task.

Graphs with both graph structure and node features are called as attributed graphs. The research attention of attributed graph clustering [20] is how to balance graph structural information and node features information. The most common pipeline is to learn node representations that

incorporate structural information first, and then implement common clustering methods, such as K-means or spectral clustering to obtain the final results. GAE and VGAE [21] are the models that combine graph convolutional network and auto-encoder or variational auto-encoder to extract node representations. AGC [22] is an adaptive spectral graph convolution method, which uses high-order graph convolution operations to capture the global structure, and uses the intra-class distance to adaptively select the appropriate order, but this order selection method sometimes fails. SDCN [23] is a new structural deep clustering network, which uses dual self-supervised modules to effectively combine the advantages of auto-encoder and graph convolution networks. Based on the above work, DGSCN [24] is a deep graph structural clustering network based on a triple self-supervised module, which adds a graph auto-encoder structure and integrates denseGCN [25] to alleviate the over-smoothing problem, but its complex structures do not improve the performance obviously. AGE [26] uses the graph smoothing filter operator to filter the graph node features, and then calculates the similarity matrix of the filtered features to select pairs of positive and negative training samples as the supervision information to train the encoder. Although this approach is very innovative, many hyper-parameters cannot be fine-tuned to restrict its application.

The above works are all based on simple graphs, but they only describe the pairwise relations. Modeling high-order data using the graph structure inevitably loses the high-order information [8]. Therefore, it is unreasonable to use graph-based methods to mine or analyze high-order data. Hypergraph structures, as high-order representations of graph structures, can model complete high-order data. Meanwhile, because hypergraphs have wider node perceptual range and more reasonable interpretability, they achieve a considerable performance in certain graph tasks. How to mine potential information from hypergraph structures attracts increasing attention. [27] puts forward the concept of learning with hypergraphs, which can be used in clustering, classification, and embedding and achieves the performance beyond the ordinary graph. Hypergraph neural networks (HGNN) proposed in [28], which are similar to the GCNs in graphs [2], extend the convolution operation to the process of hypergraph learning. [12] proposes a new unsupervised feature selection method to jointly learn the similarity matrix and conduct both subspace learning and feature selection, which can use on clustering tasks. But its complex optimization details do not make it perform better than the above graph methods for clustering tasks.

However, it is a lack of works that directly cluster for hypergraphs or high-order data using deep learning. Our paper takes the hypergraph node clustering tasks as the research object, and proposes a novel AHGAE model, which is well compatible with the related tasks of graphs.

3 Proposed Method

In this section, we introduce our proposed Adaptive Hypergraph Auto-Encoder (*AHGAE*) model in detail. The overall framework of the model is shown in Fig. 2.

First, the basic concepts of hypergraphs and related symbolic representations are introduced in Section 3.1. Then in Section 3.2, a hypergraph Laplacian smoothing filter is proposed. We derive it in the spatial domain and analyze its low-pass characteristics in the frequency domain. In Section 3.3, we analyze the influence of the weight ratio of each node to its adjacent node features passing through hyperedges on the hypergraph, and how to select the order of Laplacian smoothing filtering layers is shown in Section 3.4. Finally, the detailed process of relational reconstruction auto-encoders is explained in Section 3.5. The overall algorithm of *AHGAE* is given in the end.

3.1 Hypergraph Definition

Given an undirected attributed hypergraph $\mathbf{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{W})$, where $\mathcal{V}$ represents the node set, $\mathcal{E}$ represents the hyperedge set, $\mathbf{X}$ represents the feature matrix and $\mathbf{W}$ represents the weighted diagonal matrix composed of weights of hyperedges. The feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$ is composed of the feature vectors of all nodes, where $|\mathcal{V}|$ is the number of nodes, and F is the dimension of the node features. v_i is used to denote the ith node in the nodes set $\mathcal{V}$, and e_k is used to denote the kth hyperedge in the hyperedge set $\mathcal{E}$. For the hypergraph $\mathbf{G}$, the incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ is used to describe the relations between nodes and hyperedges, where $|\mathcal{E}|$ represents the number of hyperedges. Each element $h(i,k)$ in $\mathbf{H}$, i.e., whether node v_i belongs to hyperedge e_k, is calculated as follows:

$$h(i,k) = \begin{cases} 1, & if\ v_i \in e_k \\ 0, & else \end{cases}. \tag{1}$$

3.2 Hypergraph Laplacian Smoothing Filter

The hypergraph Laplacian smoothing filter can be regarded as information aggregation between nodes in the spatial domain, and their information is transmitted through the hyperedges, as shown in Fig. 3.

The given information includes the node feature matrix $\mathbf{X}$, the incidence matrix $\mathbf{H}$, and the hyperedge-weighted matrix $\mathbf{W}$. Through calculation, the node degree v_i is $d_v(i) = \sum_{e_k \in \mathcal{E}} w(k) h(i,k)$, and the hyperedge degree e_k is $d_e(k) = \sum_{v_i \in \mathcal{V}} h(i,k)$. Further, $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the diagonal matrices of the node degrees and the hyperedge degrees, respectively.

For the node information aggregation process, we pass the average value of the node features to hyperedge. The feature of hyperedge e_k is defined as

$$\begin{aligned} \mathbf{E}_k^{(t)} &= \frac{1}{|N(e_k)|} \sum_{v_j \in N(e_k)} \mathbf{X}_j^{(t)} \\ &= \sum_{v_j \in \mathcal{V}} \frac{h(j,k)}{d_e(k)} \mathbf{X}_j^{(t)}, \end{aligned} \tag{2}$$

where t represents the order, $N(e_k)$ represents the set of all sub-nodes belonging to the hyperedge e_k, $\mathbf{E}_k$ and $\mathbf{X}_j$ represent the feature of the hyperedge e_k and the feature of the node v_j, respectively.

After each hyperedge aggregates the features of all sub-nodes, we need to pass the information back to all nodes through the weight of the hyperedge. Then, we implement

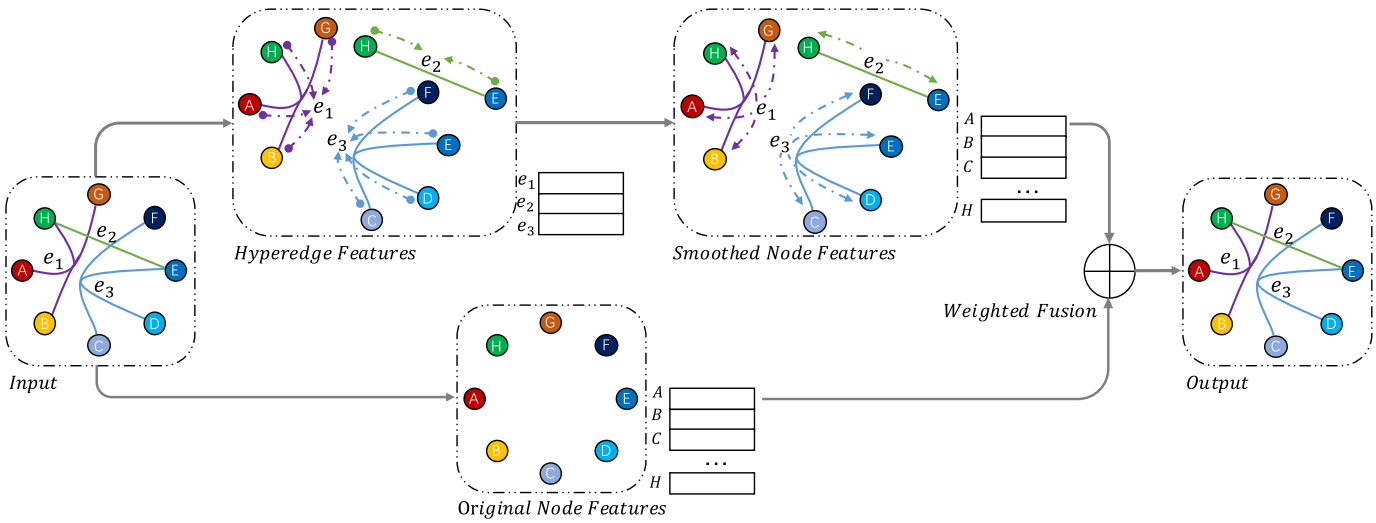


Fig. 3. Hypergraph Laplacian Smoothing Filter. First, the node features are merged into hyperedge features, and then the hyperedge features are propagated to the nodes to form smoothed node features. The smoothed node features are weighted fusion with the original node features to form a new node feature. Then, they are combined with the initial graph structure to generate the output.

weighted fusion with the original node information

$$\begin{aligned}\mathbf{X}_i^{(t+1)} &= (1-\gamma)\mathbf{X}_i^{(t)} + \gamma \sum_{e_k \in N(v_i)} \frac{h(i,k)w(k)}{d_v(i)} \mathbf{E}_k^{(t)} \\ &= (1-\gamma)\mathbf{X}_i^{(t)} + \gamma \sum_{v_j \in \mathcal{V}} \sum_{e_k \in \mathcal{E}} \frac{h(i,k)w(k)h(j,k)}{d_v(i)d_e(k)} \mathbf{X}_j^{(t)},\end{aligned} \tag{3}$$

where $N(v)$ represents all the hyperedges adjacent to the node v, and $\gamma \in [0,1]$ is the weight coefficient of the filter.

We then simplify the above formula and express it in a matrix form

$$\mathbf{X}^{(t+1)} = (1-\gamma)\mathbf{X}^{(t)} + \gamma \mathbf{D}_v^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{X}^{(t)}, \tag{4}$$

where $\mathbf{I}$ is the identity matrix whose dimension is the number of nodes $|\mathcal{V}|$.

However, the spectral radius of $\mathbf{D_v^{-1}HWD_e^{-1}H^T}$ is not less than 1, which leads to an unstable state of features, and increases the risk of feature explosion or disappearance when stacking multi-layer filters. We replace it with symmetric normalized form $\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}$ [27], and the equation is represented as

$$\begin{aligned}\mathbf{X}^{(t+1)} &= (1-\gamma)\mathbf{X}^{(t)} + \gamma \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}\mathbf{X}^{(t)} \\ &= \mathbf{X}^{(t)} - \gamma(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2})\mathbf{X}^{(t)}.\end{aligned} \tag{5}$$

Therefore, we obtain the symmetric hypergraph Laplacian matrix

$$\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}. \tag{6}$$

Here $\mathbf{L}$ is a positive semi-definite matrix[29]. Therefore, the eigenvalues of $\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}$ are no larger than 1, which solve the problem of feature instability.

So the formula of multi-order hypergraph Laplacian smoothing filter is

$$\mathbf{X}^{(t)} = (\mathbf{I} - \gamma\mathbf{L})^t\mathbf{X}. \tag{7}$$

A reasonable order t enables nodes to obtain the most suitable perception range to improve clustering performance. The hypergraph Laplacian smoothing filter with an order selected by node features is called an adaptive hypergraph Laplacian smoothing filter. The specific selection strategy is explained in Section 3.4.

Then we discuss the low-pass filtering characteristics of the hypergraph smoothing Laplacian filter in the frequency domain. We decompose the eigenvalue of the hypergraph Laplacian operator $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix, and the diagonal elements are the eigenvalues of $\mathbf{L}$. The frequency response function

$$p(\mathbf{\Lambda}) = diag(p(\lambda_1), \ldots, p(\lambda_{|\mathcal{V}|})), \tag{8}$$

so the hypergraph Laplacian smoothing filter can be set to $\mathbf{G} = \mathbf{U}p(\mathbf{\Lambda})\mathbf{U}^{-1}$. The essence of the smoothing filter is a low-pass filter. The frequency response function is set to a linear form

$$p(\lambda) = 1 - \gamma\lambda, \ \gamma \in [0,1]. \tag{9}$$

Because of the eigenvalue of the hypergraph Laplacian $\lambda \in [0,1]$, $p(\lambda)$ is negatively correlated with λ, and $p(\mathbf{\Lambda})$ is positive semi-definite matrix. Therefore, the hypergraph Laplacian smoothing filter $\mathbf{G}$ can suppress high-frequency signals and preserve the low-frequency information containing rich semantic information. It is similar to the graph signal filter, and the corresponding proof for graph structure is provided in [22]. The hypergraph Laplacian smoothing filter can be simplified as

$$\mathbf{G} = \mathbf{U}p(\mathbf{\Lambda})\mathbf{U}^{-1} = \mathbf{U}(\mathbf{I} - \gamma\mathbf{\Lambda})\mathbf{U}^{-1} = \mathbf{I} - \gamma\mathbf{L}. \tag{10}$$

So the update function of feature matrix $\mathbf{X}$ is the same as Eq. (5).

3.3 Weight γ

For the feature update function in Eq. (3), the value of γ determines the proportion of each node's features and adjacent features. When $\gamma = 0$, the node information is no longer

updated, i.e., $X^{(t+1)} = X^{(t)}$, which is obviously invalid. When $\gamma = 1$

$$\mathbf{X}^{(t+1)} = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}\mathbf{X}^{(t)}, \tag{11}$$

which is the hypergraph convolution operation [28]. Here, the node information is completely replaced by the information obtained by neighbor aggregation. Suppose the two nodes v_i, v_j with the same incidence information, i.e., $h(i,:) = h(j,:)$, their updated features become the same and no longer have discrimination in downstream calculations.

Hyperedges often cover more nodes in the real world, and the nodes have a greater probability of having the same incidence relationship so that the above situation appears frequently. For example, in a social personalized tag network, even though there are so many personalized tags, many users still have the same tag sets. Therefore, using Eq. (11) will cause the user portrait information to become invalid. For simple graphs, when they have the same adjacent relationship, the node features are indistinguishable. Currently, the best solution is to introduce residual structure or retain original information [30].

Each hyperedge in hypergraphs has its own interpretation, but this often cannot be a decisive factor for clustering tasks. So the disappearance of feature discrimination is not appropriate for clustering tasks. Especially when the order increases, the influence range increases exponentially, which leads to the over-smoothing problem more serious. Therefore, we need to compromise between their weights, i.e., the graph filter kernel

$$\mathbf{G} = (1-\gamma)\mathbf{I} + \gamma\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}. \tag{12}$$

The experimental analysis of the weight γ is explained in Section 4.5.1.

3.4 Choice of Order t

A suitable order can cause that each node perceives the most suitable range of adjacent features, so a reasonable metric becomes the key issue. Davies-Bouldin Index (DBI) [31] is the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. It has the advantage that is computed only quantities and features inherent to the dataset. The minimum score of DBI is zero, and the values closer to zero indicate a better partition. For the t-order smoothing filter, we utilize the DBI index of the smoothed node feature similarity matrix as an evaluation index for the quality of clustering

$$c_{score}^{(t)} = DBI(\mathbf{X}^{(t)}(\mathbf{X}^{(t)})^T), \tag{13}$$

where $DBI(\cdot)$ is the DBI algorithm. Note that the purpose of using the similarity matrix $\mathbf{X}\mathbf{X}^T$ instead of the original feature matrix $\mathbf{X}$ is to eliminate the influence of features and only focus on the similarities between nodes.

In detail, the formula for the DBI index is $DBI = \max_{i\neq j}\frac{S_i+S_j}{M_{i,j}}$, where S_i denotes the average distance between each of the samples within the class i and the center of the class, and $M_{i,j}$ denotes the inter-class distance from the center of cluster i to j. An overly high order is likely to cause the over-smoothing phenomenon of node features, i.e., all node feature tend to be similar and coalesce in the feature space. Therefore, the intra-class distances S_i decreases sharply, leading to a decrease in DBI score. Therefore, the global lowest of $c_{score}^{(t)}$ cannot obtain the optimal clustering performance in general. So we suppose the change of score is $\Delta c_{score}^{(t)} = c_{score}^{(t)} - c_{score}^{(t-1)}$. When $\Delta c_{score}^{(t)} > 0$, the order of $t-1$ is the local minimum and the clustering performance achieves or nears the best order, so we regard $t-1$ as the optimal order and obtain the optimal smoothed feature matrix $\mathbf{X}_{sm} = \mathbf{X}^{(t-1)}$.

The experimental analysis of the order t is further explained in Section 4.5.2.

3.5 Relational Reconstruction Auto-Encoder

After obtaining the smoothed feature matrix, we utilize the relational reconstruction auto-encoder to further learn node representations in low-dimensional spaces without losing structural information, as shown in Fig. 2. First, the adjacency matrix is constructed through the incidence matrix

$$\mathbf{A} = \varepsilon(\mathbf{H}\mathbf{H}^{\mathbf{T}}), \tag{14}$$

where binarization function $\varepsilon(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \end{cases}$.

The filtered feature matrix is compressed through a single-layer fully connected layer

$$\mathbf{Z} = scale(\mathbf{X_{sm}}\mathbf{\Theta}), \tag{15}$$

where Z is the node embedding matrix which contains both feature information and structural information, $scale(\cdot)$ represents a normalization function to rescale the range of node features to [0,1], i.e., $scale(\mathbf{x}) = \frac{\mathbf{x}-min(\mathbf{x})}{max(\mathbf{x})-min(\mathbf{x})}$, and $\mathbf{\Theta} \in \mathbb{R}^{F\times F'}$ is the learnable parameter that is applied over the nodes to extract features.

We further calculate the similarity matrix between the node features

$$\mathbf{S} = sigmoid(\mathbf{Z}\mathbf{Z}^T), \tag{16}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$. Eq. (16) is equivalent to calculate the normalized cosine distance matrix between node features. It is also called as an inner product decoder, which is used to reconstruct the adjacent relations between nodes. Our goal is to minimize the error between the adjacency matrix $\mathbf{A}$ and the similarity matrix $\mathbf{S}$. When converting an incidence matrix to an adjacency matrix by Eq. (14), the nodes in each hyperedge are connected in pairs. When converting a hypergraph to a normal graph, a hyperedge with degree d will be converted into $\frac{d\times(d-1)}{2}$ normal edges. Therefore, as the hyperedge degrees increase, the number of edges increases sharply. In some cases, the adjacency matrix is too dense, which will cause a serious imbalance between positive and negative samples in matrix $\mathbf{A}$.

In order to solve the above problems, we choose to weight each element in $\mathbf{A}$

$$\mathbf{W}_{ij} = \begin{cases} \frac{|\mathcal{V}|^2 - \sum\sum \mathbf{A}_{ij}}{\sum\sum \mathbf{A}_{ij}}, & \mathbf{A}_{ij} = 1 \\ 1, & \mathbf{A}_{ij} = 0 \end{cases}. \tag{17}$$

The weighted binary cross-entropy function is used to calculate the reconstruction loss

$$L_{re} = \frac{1}{|\mathcal{V}|^2} \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} -\mathbf{W}_{ij}[\mathbf{A}_{ij} \cdot log\ \mathbf{S}_{ij} + (1 - \mathbf{A}_{ij}) \cdot log\ (1 - \mathbf{S}_{ij})]. \tag{18}$$

By introducing weights to the cross-entropy loss function, the imbalance was effectively mitigated and the model had str onger compatibility with the tasks in denser relational networks.

Through a period of training to the relational reconstruction auto-encoder, the learned node embeddings is obtained. For clustering tasks, the spectral clustering algorithm [16] is utilized to obtain the final cluster distribution.

The overall algorithm flow is provided in Algorithm 1, where Lines 1-8 represent the adaptive hypergraph Laplacian smoothing filter, and Lines 9-15 represent the relational reconstruction auto-encoder.

Algorithm 1. AHGAE

Input: Node set $\mathcal{V}$, incidence matrix $\mathbf{H}$, feature matrix $\mathbf{X}$, hyperedge weight matrix $\mathbf{W}$, iteration training times N;
Output: Node Embedding Matrix $\mathbf{Z}$;
1 Set $\mathbf{X}^{(0)} = \mathbf{X}, t = 0, c^{(0)}_{score} = -\infty$;
2 **repeat**
3 $t = t + 1$;
4 $\mathbf{X}^{(t)} = (\mathbf{I} - \gamma\mathbf{L})\mathbf{X}^{(t-1)}$;
5 Calculate $c^{(t)}_{score}$ by Eq. (13);
6 $\Delta c^{(t)}_{score} = c^{(t)}_{score} - c^{(t-1)}_{score}$;
7 **until** $\Delta c^{(t)}_{score} > 0$
8 Set $\mathbf{X}_{sm} = \mathbf{X}^{(t-1)}$;
9 Obtain Adjacency matrix $\mathbf{A}$ from Eq. (14);
10 **for** $iter \in 0, 1, \ldots, N$ **do**
11 Get Embedding Matrix $\mathbf{Z}$ from Eq. (15);
12 Get Similarity matrix $\mathbf{S}$ from Eq. (16);
13 Train Relational Reconstruction Auto-Encoder with loss function Eq. (18);
14 **end**
15 Output Node Embedding Matrix $\mathbf{Z}$.

4 Experiments

In this section, we evaluate our proposed model on clustering tasks for hypergraph data and graph data, and also compare the impact of different graph filtering kernels on our model. Meanwhile, we also discuss the influence of hyperparameters including weight γ and order t, and verify the validity of the proposed order selection method.

4.1 Datasets

4.1.1 Attributed Hypergraph Dateset

To demonstrate that the proposed model *AHGAE* has considerable performance on attributed hypergraph datasets, the article-author hypergraph network *DBLP-HG* is constructed through the XML interface provided by the DBLP (DataBase system and Logic Programming) website.[1] In *DBLP-HG*, each node represents a paper, whose features are extracted by applying the bert-as-service [32] to the title of the paper, and each hyperedge represents a different author. Certain authors are used as the parent nodes, and adopt the BFS algorithm to mine the information of authors and their papers. The author IDs with a higher frequency is extracted to construct hyperedges, and use the research field of the journal/conference proceedings (refer to the attention of researchers in different fields and SCI partition) as the paper (node) label. Here we define 3 themes: *Artificial Intelligence and its Applications*, *Power Electronics and Circuit Systems*, and *Network and Communication*. The paper numbers of each theme are 1,984, 1,668, and 1,411, respectively.

Apart from the clustering tasks, this dataset is also utilized hypergraph supervised node classification and other hypergraph tasks. The data acquisition method is acquired in the appendix for detail, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2021.3108192.

4.1.2 Attributed Graph Datesets

Some additional graph datasets with different implications are utilized to verify the considerable performance of our model in simple graph datasets, such as citation networks, webpage redirection networks, paper co-author networks, and author networks. They all provide complete attributed graph information, including graph structure, node features, and node labels. A detailed description of the dataset is provided below:

- *Cora* [33]: This is a citation network dataset describing the citation relations between papers. Each node represents a paper and its feature is the word vector of the article. Each edge represents the citation relations between the two connected papers. Node labels indicate different research fields and the node features are binary word vectors.
- *Wiki* [34]: This is a webpage redirection network dataset where each node represents a web page, and each node feature is represented as a tf-idf weighted word vector. If two web pages can be accessed through hyperlinks, they are connected.
- *ACM* [23][2]: This is a paper network dataset selected from the ACM official dataset. If two papers have the same author, they will be connected by edges. The node features represent the word vectors of keywords, and categories represent three research fields: database technology, wireless communication, and data mining.
- *DBLP* [23][3]: This dataset is based on an author network provided by the DBLP platform, which can be understood as a social network of a specific group of people (researchers related to engineering disciplines such as computers). Each node represents an author, each edge represents that the two connected authors collaborated once, i.e., co-author, and the node features represent the feature of the keywords of the author's research.

1. https://dblp.org/xml/
2. https://dl.acm.org/
3. https://dblp.uni-trier.de/

TABLE 1
The Statistics of the Hypergraph/Graph Dataset

Dataset	Nodes	Edges	Hyperedges	Features	Classes
DBLP-HG	5,193	-	3607	768	3
Cora	2,708	5,429	-	1,433	7
Wiki	2,405	17,981	-	4,973	17
ACM	3,025	13,128	-	1,870	3
DBLP	4,058	3,528	-	334	4

Table 1 provides statistical information for the above datasets.

4.2 Baselines

We compare *AHGAE* with representative methods proposed in recent years. Their brief introductions are shown in the following:

- Methods using only node features include K**-means** [35], *Spectral-F* [16].
- *GAE* and *VGAE* [21] are models that combine graph convolutional network and auto-encoder or variational auto-encoder to extract node representations.
- *AGC* [22] is an adaptive spectral graph convolution method, which uses high-order graph convolution operations to capture the global structure, and uses the intra-class distance to adaptively select the appropriate order.
- *SDCN* [23] is a new structural deep clustering network, which uses dual self-supervised modules to effectively combine the advantages of auto-encoder and GCN.
- *DGSCN* [24] is a deep graph structural clustering network based on a triple self-supervised module, which adds a graph auto-encoder structure and integrates DenseGCN [25] to alleviate the over-smoothing problem.
- *AGE* [26] first uses the graph smoothing filter operator to filter the graph node features, and then calculates the similarity matrix of the filtered features to select pairs of positive and negative training samples as the supervision information to train the encoder.

4.3 Implementation

4.3.1 Hypergraph Construction

When data is inputted into the model, we need to perform hypergraph construction and feature normalization in turn. The difference between graphs and hypergraphs is edges and hyperedges, i.e., the degree limitation of edges. How to construct a hypergraph to make the hyperedge interpretable is a key problem. Constructing hyperedges based on adjacent relations has reasonable interpretability. For example, each node and the neighbor nodes are jointly constructed into a hyperedge: for citation networks, each hyperedge represents all citation relationships of a specified paper; for web-related networks, each hyperedge represents all linked pages in a specified webpage; for co-author networks, each hyperedge represents all co-authors of a specified paper; for author networks, each hyperedge represents all co-authors of a specified author, etc.

Considering both the complexity and the intuitive semantics of the hyperedges, we connect first-order neighbors to construct hyperedges. In other words, each hyperedge is composed of each node and its neighbor nodes. In this way, more detailed information can be fully perceived and learned.

4.3.2 Metrics and Parameter Settings

To evaluate the performance of each model, the following metrics are used in the clustering tasks: clustering accuracy (ACC), normalized mutual information (NMI) and adjusted rand index (ARI). For all the metrics, a higher value represents better performance.

To verify the robustness and considerable adaptive ability of our model, we remove the fixed random seeds to implement random parameter initialization. Additionally, we repeat each experiment at least 20 times and report the average clustering performance of the model *AHGAE* and the latest models. For the graph filter kernel of the adaptive hypergraph Laplacian filter in Eq. (12), the weight γ is set to $\frac{4}{5}$ for *Cora* and $\frac{2}{3}$ for other datasets. A further discussion about the choice of γ is in Section 4.5.1. The Adam optimizer is utilized for the reconstruction encoder part, and out of experience, the learning rate is set to 10^{-4} for *DBLP-HG* and 10^{-3} for other datasets. The encoder is a single fully connected layer whose input dimension is the node feature dimension and the output dimension is 500. This setting helps us well to compare with baselines. The iteration numbers of training are fixed at 5,000 for *DBLP-HG* and fixed at 400 for others to generate our final node embeddings. The spectral clustering result of node embeddings is used as the final clustering result to verify our model.

Note that since this paper does not consider the weight difference between the hyperedges, we give the hyperedges equal weights, i.e., $\mathbf{W} = \mathbf{I}$ on the following experiments. Moreover, it is emphasized that the spectral clustering algorithm is utilized to verify our performance.

4.4 Results

4.4.1 Attributed Hypergraph Clustering

Here we discuss the experiments for attributed hypergraph datasets. For the graph-based clustering methods, we use Eq. (14) to generate its corresponding adjacency matrix for downstream calculations. The experiment results of different models on the *DBLP-HG* are summarized in Table 2, where the bold values indicate the best performance.

The proposed model *AHGAE* exceeds representative models proposed in recent years. Compared with the latest AGE model, AHGAE still has 11.65%, 5.01%, and 13.15% performance improvements on ACC, NMI, and ARI respectively.

To demonstrate the validity of our proposed model, we replace the hypergraph Laplacian smoothing filter $\mathbf{G}$ calculated by Eq. (12) in our model with the graph convolution kernel proposed by [2]

$$\mathbf{G}_1 = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}, \tag{19}$$

TABLE 2
Clustering Results on DBLP-HG

Model	ACC	NMI	ARI
GAE(2016)	53.11	10.47	13.06
VGAE(2016)	49.97	9.45	8.54
AGC(2019)	60.97	22.77	26.03
SDCN(2020)	60.92	19.52	24.76
DGSCN(2020)	61.54	20.09	25.77
AGE(2020)	61.22	31.89	30.41
AHGAE(our)	**72.87**	**36.90**	**43.56**

the adaptive graph convolution kernel proposed by [22]

$$\mathbf{G}_2 = \frac{1}{2}\mathbf{I} + \frac{1}{2}\widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}, \tag{20}$$

the graph filter kernel of the graph Laplacian smoothing filter proposed by [26]

$$\mathbf{G}_3 = \frac{1}{3}\mathbf{I} + \frac{2}{3}\widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}, \tag{21}$$

and the hypergraph convolution kernel proposed by [28]

$$\mathbf{G}_4 = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}. \tag{22}$$

In the above formulas, $\mathbf{A}$ is obtained through Eq. (14), and $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$. For each graph filter kernel, we repeat the experiment 20 times to get the average value, and the experimental results are shown in Table 3, where RAE represents the relational reconstruction auto-encoder, and the bold values indicate the best performance.

By comparing with different graph filter kernels on the hypergraph dataset *DBLP-HG*, the superiority of the proposed kernel is proved. Although the performance is comparable when only the adaptive smoothing filter is used, the introduction of RAE prompts our hypergraph Laplacian smoothing filter kernel perform better than other filter kernels. Compared with the latest graph Laplacian smoothing filter kernel $\mathbf{G}_3$, the proposed filter kernel still has 0.30%, 1.23%, and 1.15% performance improvements on ACC, NMI, and ARI respectively.

The hypergraph convolution kernel $\mathbf{G}_4$ also directly uses high-order data as the object, which still achieves a certain improvement compared with the graph convolution kernel $\mathbf{G}_1$. But compared with $\mathbf{G}_4$, the proposed kernel G solves the problem of a rapid decline in the discrimination between hypergraph nodes by retaining a certain weight of the node's information, which has 8.70%, 15.79%, and 17.55% performance improvements on ACC, NMI, and ARI respectively.

Note that sometimes the introduction of RAE reduce the performance because the over-smoothing phenomenon leads to the problem that node discrimination disappears seriously, which makes it difficult to reconstruct the adjacent relations by over-smoothed features.

4.4.2 Attributed Graph Clustering

The proposed model is also compatible with attributed graph datasets. We compare *AHGAE* with the latest representative models on attributed graph datasets, including *Cora*, *Wiki*, *ACM* and *DBLP*. The graph-based models completely adopted the original parameter settings in their work. But for a fair comparison, we also remove the random seed of the mentioned models to test their robustness and adaptive ability. The experimental results are presented in Table 4, where the bold values indicate the best performance. The model AHGAE, which combines the above two parts, further improves the performance and in performance, our model exceeds the graph-based models proposed in recent years. Furthermore, the experiments on DBLP with sparser edges and Wiki with denser edges verify the advantages and powerful generalization ability of our proposed method.

TABLE 3
Clustering Results of Different Kernels on DBLP-HG

Model	kernel	ACC	NMI	ARI
Basline(Spectral-F)	-	50.08	14.79	15.10
Only RAE	-	69.12	31.98	38.37
Only Adaptive-Smoothing	$\mathbf{G}_1$	59.81	18.41	20.60
Only Adaptive-Smoothing	$\mathbf{G}_2$	65.45	28.20	32.87
Only Adaptive-Smoothing	$\mathbf{G}_3$	65.75	29.59	34.49
Only Adaptive-Smoothing	$\mathbf{G}_4$	60.41	19.78	24.51
Only Adaptive-Smoothing	$\mathbf{G}$	65.12	28.40	33.69
Adaptive-Smoothing + RAE	$\mathbf{G}_1$	61.14	19.76	21.14
Adaptive-Smoothing + RAE	$\mathbf{G}_2$	70.61	31.84	38.09
Adaptive-Smoothing + RAE	$\mathbf{G}_3$	72.57	35.67	42.41
Adaptive-Smoothing + RAE	$\mathbf{G}_4$	63.57	21.11	26.01
AHGAE(our)				
Adaptive-Smoothing + RAE	$\mathbf{G}$	**72.87**	**36.90**	**43.56**

Comparing with *AGE* [26], which consists of the Laplacian smoothing filter and the adaptive encoder, the proposed model has some advantages:

- The hypergraph Laplacian smoothing filter has a larger perception domain, which means that each node can more easily aggregate a wider range of node information.
- The relation reconstruction auto-encoder introduces weights to solve the imbalance problem, and compared with the adaptive filter in AGE, it has a simpler hyperparameter adjustment to achieve considerable performance.

Similarly, they are the main advantages of our model compared to the graph-based methods.

Meanwhile, the ablation experiments are conducted and the results are also shown in Table 4. Compared with the baseline *Spectral-F*, performance improvements are obtained apparently after introducing either the adaptive hypergraph Laplacian smoothing filter or the relation reconstruction auto-encoder. The proposed model *AHGAE*, which is combined by the above two parts, obtains further improvement and achieves state-of-the-art. It shows that the two parts of our model are meaningful and indispensable.

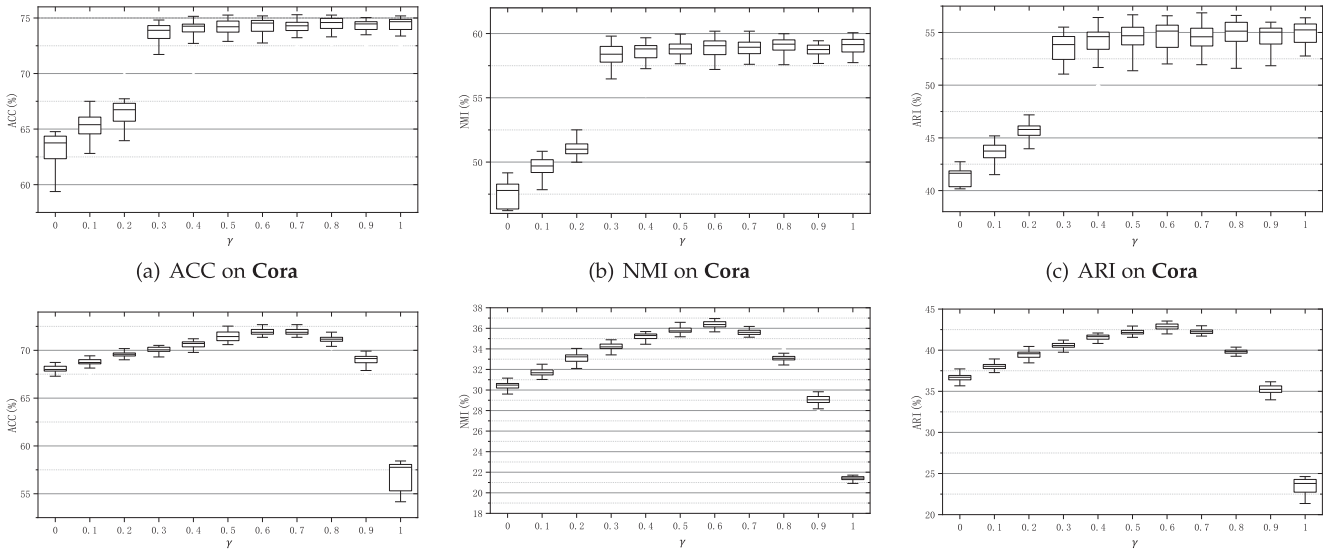
4.5 Discussion

4.5.1 Influence of Weight γ

In this section, to discuss the effect of the value of weight γ on the results, we plot the clustering performance *w.r.t.*

TABLE 4
Clustering Results for Attributed Graph Datasets Including *Cora*, *Wiki*, *ACM* and *DBLP*

	Cora			Wiki			ACM			DBLP		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means	50.30	31.70	24.40	41.70	44.00	15.10	67.31	32.44	30.60	38.65	11.45	6.97
Spectral-F	34.70	14.70	7.10	49.10	46.40	25.40	71.80	37.49	38.67	58.52	27.50	23.78
GAE(2016)	61.10	48.20	30.20	37.90	34.50	18.90	84.52	55.38	59.46	61.21	30.80	22.02
VGAE(2016)	59.20	40.80	34.70	45.10	46.80	26.30	84.13	53.20	57.72	58.59	26.92	17.92
AGC(2019)	63.85	49.60	37.62	47.65	45.28	27.11	68.76	35.58	37.78	52.18	22.88	18.18
SDCN(2020)	63.12	45.24	36.12	41.21	39.12	23.82	90.45	68.31	73.91	68.05	39.50	39.15
DGSCN(2020)	63.04	45.36	36.94	46.61	42.60	25.69	89.59	66.55	71.89	71.11	36.11	39.26
AGE(2020)	70.08	56.78	49.27	56.49	56.05	39.15	92.10	72.51	78.05	78.11	45.07	50.64
Adaptive Smoothing	64.18	49.93	36.99	53.56	53.24	34.38	79.93	52.01	52.57	66.95	39.17	33.79
RAE	63.86	47.74	41.52	54.01	53.95	36.46	91.80	71.59	77.28	76.83	45.58	48.86
AHGAE(both)	**74.58**	**59.12**	**55.09**	**59.49**	**56.52**	**42.80**	**92.79**	**74.63**	**79.86**	**78.62**	**48.88**	**52.23**



(a) ACC on **Cora** (b) NMI on **Cora** (c) ARI on **Cora**

(d) ACC on **DBLP-HG** (e) NMI on **DBLP-HG** (f) ARI on **DBLP-HG**

Fig. 4. Influence of weight γ on **Cora** and *DBLP-HG*. (a) (b) (c) represent the relationship between γ and the metrics ACC, NMI, ARI on **Cora** respectively, and (d) (e) (f) are for *DBLP-HG*.

weights γ on the graph dataset *Cora* and the hypergraph dataset *DBLP-HG*. We design 11 sets of experiments for each dataset which have different γ in the range of [0,1] with a step of 0.1. Each set of experiments is repeated 40 times, and the ACC, NMI, ARI metrics of the results of each experiment are recorded. The result is plotted as a box plot in Fig. 4. The best performance is obtained in $\gamma = 0.8$ and $\gamma = 0.6$ on *Cora* and *DBLP-HG*, respectively.

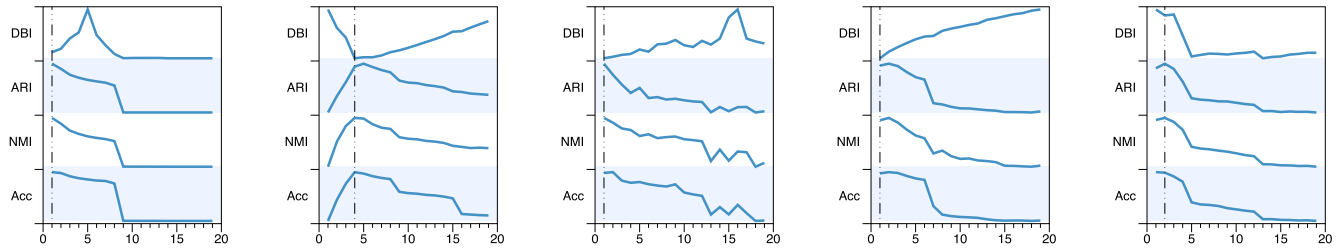
As for the hypergraph dataset *DBLP-HG*, overly high or low weights cause significant drops in performance. If γ is set too high, the discrimination between nodes will decrease sharply, even lead to lower performance than the performance of directly clustering the original data. By contrast, if γ is set too low, the smoothing order will increase apparently, and each node tends to learn the overall information, which makes it difficult to accomplish clustering tasks. It explicitly demonstrates the importance of a suitable weight ratio between node information and its neighbor information. As for the graph dataset *Cora*, a similar conclusion is obtained.

In addition, by observing the relative positions of the quartiles of the box plot, we can also find that an appropriate γ can also ensure the relative stability of the clustering performance. Moreover, the choice of the best γ will depend on the importance of the structural information relative to the attribute information.

4.5.2 Over-Smoothing Analysis and Order Selection

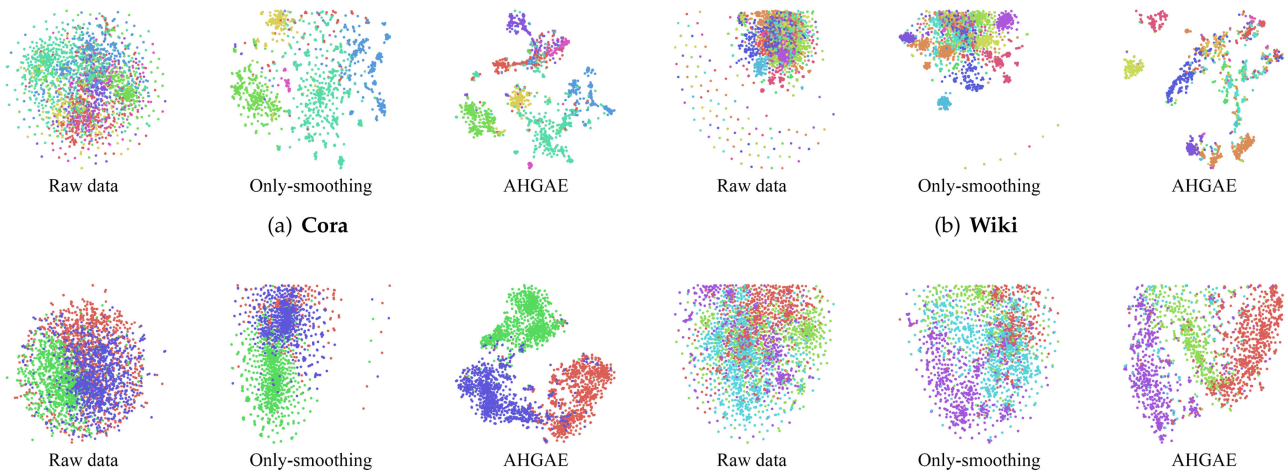
In this section, we discuss the over-smoothing problem and analyze the relations between the multi-order hypergraph Laplacian smoothing filter and DBI calculated by Eq. (13).

To demonstrate the validity of this selection method, we plot the clustering performance *w.r.t.* the order t on hypergraph dataset *DBLP-HG* and graph datasets *Cora*, *Wiki*, *ACM*, *DBLP*. The relations between the changing trend of different metrics and the order t when $\gamma = \frac{2}{3}$ are shown in Fig. 5. The order of the local lowest DBI is the same or near to the order of the optimal clustering effect. As shown in Fig. 5, the orders of the local lowest DBI on different



(a) **DBLP-HG** (b) **Cora** (c) **Wiki** (d) **ACM** (e) **DBLP**

Fig. 5. Over-Smoothing Analysis and Choice of Smoothing Order. The vertical axis represents the changing trend about different metrics, and the horizontal axis represents the order t.



(a) **Cora** (b) **Wiki**

(c) **ACM** (d) **DBLP**

Fig. 6. Two-dimensional visualization of node representations using t-SNE on graph datasets. For each sub-figure, the scatter plots represent the low-dimensional distribution of the original data, the smoothed data, and the embeddings learned by AHGAE, respectively.

datasets are 1, 4, 1, 1, 2 respectively. The best performance is obtained at orders 1, 4, 1, 2, 2 respectively, which are almost the same as the selected order.

By observing the trend of the ACC, NMI, and ARI in Fig. 5, we can find that almost everyone is rise first and then fall. It is no doubt that the increase is a manifestation of the effectiveness of the introduction of structural information, and the decline in performance indicates the over-smoothing phenomenon. Over-smoothing refers to the phenomenon that each node tends to have the same feature so that the distinction between nodes is greatly reduced with the increasing layers of graph filtering. Over-smoothing is an inevitable problem in graph or hypergraph learning. Fig. 5 shows that the DBI of the feature similarity matrix can stably select or close to the optimal order. This selection method prevents the occurrence of the over-smoothing phenomenon, which is very meaningful in practical applications.

4.5.3 Visualization

Finally, to intuitively reflect the performance of our model on clustering tasks, the representation in two-dimensional space is visualized using a t-SNE algorithm [36].

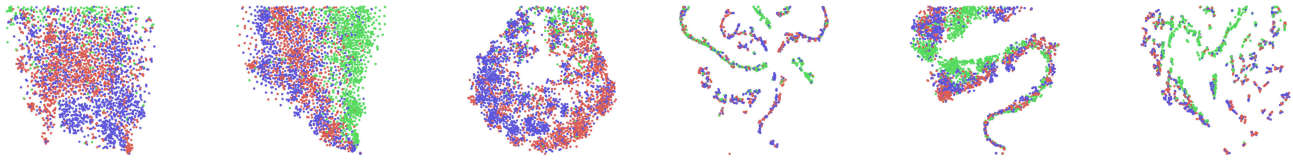
First, the experiments are conducted on all datasets, as shown in Fig. 6, where the different colors represent different labels, and the coordinates indicate the relative positions of different node features in two-dimensional space. For each sub-figure in Fig. 6, the scatter plots represent the two-dimensional distribution of the original data, the adaptive-smoothed data, and the embeddings learned by AHGAE, respectively.

It is obvious that the adjacent data tend to have the same features when applying the adaptive hypergraph Laplacian smoothing filter, which is reflected on scatter plots that the data points having the same attributes are close to each other. Then when applying the relationship reconstruction auto-encoder, the distance between the clusters gradually increases to achieve our final AHGAE performance. It is proven that the decoupling design of our proposed model is effective. The adaptive hypergraph Laplacian smoothing filter only pays attention to intra-class relations, and the relationship reconstruction auto-encoder focuses on inter-class relations while maintaining adjacent relations.

We also have similar conclusions for the hypergraph dataset *DBLP-HG*, as shown in Figs. 7a, 7b, and 7c. Moreover, the two-dimensional visualization learned by AGE, AGC, and DGSCN are shown in Figs. 7d, 7e, and 7f. It shows that our model has considerable clustering performance compared with other models, but other models have irregular distributions to weaken clustering performance.

5 Conclusion

In this paper, we propose a framework of Adaptive Hypergraph Auto-Encoders(AHGAE), which is used to learn the node embeddings of relational data for clustering tasks,



(a) Raw data(50.08%) (b) Smoothing(65.12%) (c) AHGAE(72.87%) (d) AGE(61.22%) (e) AGC(60.97%) (f) DGSCN(61.54%)

Fig. 7. Two-dimensional visualization of node representations using t-SNE on **DBLP-HG**. (a) (b) represent the low-dimensional distribution of the original data, the smoothed data, respectively. (c) (d) (e) (f) represent the low-dimensional distribution of the embeddings learned by AHGAE, AGE, AGC, and DGSCN, respectively.

whether data has a graph structure or hypergraph structure. Through the adaptive hypergraph Laplacian smoothing filter and the relational reconstruction auto-encoder, the node information and structural information are fused adaptively. Because hypergraphs are regarded as generalizations of graphs, AHGAE innovatively unifies the hypergraphs and graphs, which is a more general model for clustering tasks and embedded learning. By the clustering experiments on the attributed graph or hypergraph datasets, the proposed model outperforms state-of-the-art proposed in recent years. In future works, it is worthy to study how to Unify hypergraph learning and graph learning. Besides, embedded learning or clustering tasks of large-scale hypergraphs are also very meaningful research orientation.

ACKNOWLEDGMENTS

This work was supported in part by the National Nature Science Foundation of China under Grants U1701262, 61971268, 61931008, 61671196, 62071415, 62001146, 61701149, 61801157, 61901145, 61901150, and 61972123, in part by the National Key Research and Development Program of China under Grant 2020YFB1406604, in part by the Zhejiang Province Nature Science Foundation of China under Grants LR17F030006 and Q19F010030, and in part by 111 Project No. D17019.

REFERENCES

[1] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019, *arXiv:1810.00826*.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv:1609.02907*.
[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[4] F. Feng, X. He, H. Zhang, and T.-S. Chua, "Cross-GCN: Enhancing graph convolutional network with k-order feature interactions," *IEEE Trans. Knowl. Data Eng.*, early access, May 4, 2021, doi: 10.1109/TKDE.2021.3077524.
[5] J. Zhang, X. Shi, S. Zhao, and I. King, "STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4264–4270.
[6] W. Fan *et al.*, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.
[7] C. Berge, *Hypergraphs - Combinatorics of Finite Sets* (North-Holland mathematical library), vol. 45. Amsterdam, The Netherlands: North-Holland, 1989.
[8] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Hypergraph learning with line expansion," 2020, *arXiv: 2005.04843*.
[9] H. Chen, H. Yin, X. Sun, T. Chen, B. Gabrys, and K. Musial, "Multi-level graph convolutional networks for cross-platform anchor link prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1503–1511.
[10] H. Shi *et al.*, "Hypergraph-induced convolutional networks for visual classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, pp. 2963–2972, Oct. 2019.
[11] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, "Dual channel hypergraph collaborative filtering," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2020–2029.
[12] X. Zhu, Y. Zhu, S. Zhang, R. Hu, and W. He, "Adaptive hypergraph learning for unsupervised feature selection," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3581–3587.
[13] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2635–2641.
[14] F. Feng, X. He, Y. Liu, L. Nie, and T.-S. Chua, "Learning on partial-order hypergraphs," in *Proc. World Wide Web Conf.*, 2018, pp. 1523–1532.
[15] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Nov. 19, 2020, doi: 10.1109/TPAMI.2020.3039374.
[16] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 849–856.
[17] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.
[18] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[19] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1393–1402.
[20] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 927–936.
[21] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[22] X. Zhang, H. Liu, Q. Li, and X. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4327–4333.
[23] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proce. Web Conf.*, 2020, pp. 1400–1410.
[24] X. Li, Y. Hu, Y. Sun, J. Hu, J. Zhang, and M. Qu, "A deep graph structured clustering network," *IEEE Access*, vol. 8, pp. 161727–161738, 2020.
[25] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *Proc. EEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9266–9275.
[26] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 976–985.
[27] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 1601–1608.
[28] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3558–3565.
[29] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 17–24.
[30] J. You, J. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 10737–10745.

[31] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
[32] H. Xiao, "Bert-as-service," 2018. [Online]. Available: https://github.com/hanxiao/bert-as-service
[33] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, 2008, Art. no. 93.
[34] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.
[35] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," *J. Roy. Statist. Soc. Ser. C*, vol. 28, no. 1, pp. 100–108, 1979.
[36] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 93, pp. 3221–3245, 2014.

Youpeng Hu was born in Jiujiang, Jiangxi, China, in 1997. He received the BS degree in automation from Hangzhou Dianzi University, China, in 2019. He is currently working toward the MS degree in computer science from Shandong University, China. His research interests include graph-related learning and intelligent information processing.

Xunkai Li was born in Harbin, Heilongjiang, China, in 2000. He is currently working toward the BS degree in computer science from Shandong University, China. His research interests include machine learning and information processing.

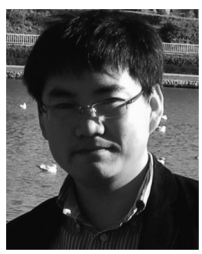
Yujie Wang was born in Jinan, Shandong, China, in 1998. He received the BS degree in computer science and technology from the China University of Mining & Technology, Beijing, in 2020. He is currently working toward the MS degree in computer science and technology with Shandong University, China. His research interests include computer vision and graph-related learning.

Yixuan Wu was born in Handan, Hebei, China, in 1999. From 2017 to 2018, she studied digital media from the School of Mechatronics and Information Engineering, Shandong University, where she is currently studying software engineering with the School of Mechatronics and Information Engineering. Her research interests include recommender systems and graph learning.

Yining Zhao is currently working toward the BE degree with the School of Software, Tsinghua University, Beijing, China.

Chenggang Yan received the BS degree in computer science from Shandong University in 2008 and the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2013. He is currently the director of Intelligent Information Processing Lab, Hangzhou Dianzi University. Before that, he was an assistant research fellow with Tsinghua University. He has authored or coauthored more than 50 refereed journal and conference papers. His research interests include intelligent information processing, machine learning, image processing, computational biology, and computational photography. As a co-author, he was the recipient of the best paper awards in Pacific Rim Conference on Multimedia 2018, the International Conference on Internet Multimedia Computing and Service 2018, the International Conference on Game Theory for Networks 2014, and the Best Student Paper in International Conference on Multimedia and Expo 2019.

Jian Yin received the PhD degree from Shandong University, China, in 2015. He is currently a professor with the School of Mechanical, Electrical and Information Engineering, Shandong University. His research interests include recommender systems and intelligent information processing.



Yue Gao (Senior Member, IEEE) received the BS degree from the Harbin Institute of Technology, Harbin, China, and the ME and PhD degrees from Tsinghua University, Beijing, China. He is currently an associate professor with the School of Software, Tsinghua University.