Hypergraph Joint Representation Learning for Hypervertices and Hyperedges via Cross Expansion

Yuguang Yan[1], Yuanlin Chen[1], Shibo Wang[1], Hanrui Wu[2], Ruichu Cai[1,3*]

[1]School of Computer Science, Guangdong University of Technology, Guangzhou, China
[2]College of Information Science and Technology, Jinan University, Guangzhou, China
[3]Guangdong Provincial Key Laboratory of Public Finance and Taxation with Big Data Application, Guangzhou, China
ygyan@gdut.edu.cn, {chenyaunlin27,wangshibo795,cairuichu}@gmail.com, hrwu@outlook.com

Abstract

Hypergraph captures high-order information in structured data and obtains much attention in machine learning and data mining. Existing approaches mainly learn representations for hypervertices by transforming a hypergraph to a standard graph, or learn representations for hypervertices and hyperedges in separate spaces. In this paper, we propose a hypergraph expansion method to transform a hypergraph to a standard graph while preserving high-order information. Different from previous hypergraph expansion approaches like clique expansion and star expansion, we transform both hypervertices and hyperedges in the hypergraph to vertices in the expanded graph, and construct connections between hypervertices or hyperedges, so that richer relationships can be used in graph learning. Based on the expanded graph, we propose a learning model to embed hypervertices and hyperedges in a joint representation space. Compared with the method of learning separate spaces for hypervertices and hyperedges, our method is able to capture common knowledge involved in hypervertices and hyperedges, and also improve the data efficiency and computational efficiency. To better leverage structure information, we minimize the graph reconstruction loss to preserve the structure information in the model. We perform experiments on both hypervertex classification and hyperedge classification tasks to demonstrate the effectiveness of our proposed method.

Introduction

Graph structure models pairwise relations between objects appearing in many real-world applications (Wu et al. 2020; Yang, Ma, and Cheng 2021), in which the relation between two vertices is indicated by if there exists an edge connecting them. Nevertheless, higher-order relationships among objects are ubiquitous in practice and cannot be captured by a standard graph structure. To tackle this, the hypergraph is proposed to model complicated relationships among vertices (Gao et al. 2020). In a hypergraph, a hyperedge can connect more than two hypervertices, and two hypervertices can have more than one kind of relation. For example, in a co-author network, a hypervertex represents an author, and a hyperedge represents a paper co-authored by multiple hypervertices. One paper (hyperedge) usually has more than two co-authors (hypervertices), and two authors (hypervertices) may have more than one collaborative paper (hyperedge).

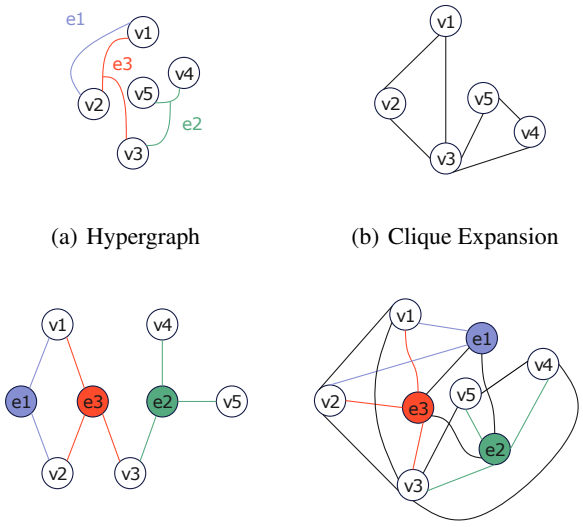
Compared with standard graphs, hypergraph learning has not been extensively explored. In order to borrow well-established graph approaches for hypergraphs, some expansion methods are proposed to transform a hypergraph structure into a graph structure, so that effective graph learning methods such as graph convolutional networks (GCN) (Kipf and Welling 2017) can be applied directly. Clique expansion and star expansion are two frequently used expansion approaches (Dong, Sawin, and Bengio 2020). However, clique expansion lacks higher-order information in the hypergraph, and star expansion ignores connections between two hypervertices or two hyperedges, resulting in information decaying in feature aggregation from a hypervertex to another hypervertex since information passing through an intermediate hyperedge is required. Neither of them can take advantage of the high-order information involved in hypergraphs.

Another kind of approach applies hypergraph convolutional operation based on hypergraph Laplacians to learn representations for hypergraphs (Feng et al. 2019). To better leverage hypergraph structure, (Dong, Sawin, and Bengio 2020) propose to learn representations for hypervertices and hyperedges alternately. (Wu, Yan, and Ng 2022) further extends this idea to learn representations for hypervertices and hyperedges from the information of both hypervertices and hyperedges. However, these methods learn separate embedding spaces for hypervertices and hyperedges without exploring a shared embedding space, which can capture common knowledge involved in hypervertices and hyperedges.

To address the above challenges, in this paper, we propose an expansion method for transforming a hypergraph to a standard graph with preserved higher-order information as well as direct connections between hypervertices or hyperedges. Figure 1 illustrates an example hypergraph and its expansions. Based on the expanded graph, we propose to learn a joint representation space for both hypervertices and hyperedges named Hypergraph Joint Representation Learning (HJRL), which extracts common knowledge involved in the representations of hypervertices and hyperedges, and at the same time, improves the data efficiency and reduces the computational cost.

In specific, we propose an expansion method called cross expansion, which transforms both hypervertices and hyper-

*Corresponding author.



(a) Hypergraph (b) Clique Expansion

(c) Star Expansion (d) Cross Expansion

Figure 1: Illustrations of expansion methods.

edges of a hypergraph to vertices in a standard graph, while the relations between two hypervertices, two hyperedges, a hypervertex and a hyperedge are preserved. Compared with the clique and star expansions, richer relationships are involved in our expansion to enhance information aggregation and graph reconstruction. Based on this expansion, we propose a joint learning model to embed both hypervertices and hyperedges (which are vertices in the expanded graph) into a shared representation space, so that more training samples are fed into the model to improve the data efficiency. In this learning paradigm, information from hypervertices and hyperedges are passed to both hypervertices and hyperedges without intermediate connectors, avoiding the information decaying during aggregation, and common knowledge involved in hypervertices and hyperedges are extracted in the joint representation space. In addition, we further preserve structural information in the model by minimizing the reconstruction loss for the hypergraph.

We summarize our major contributions as follows:

- We propose an expansion method for hypergraph by considering rich relationships between hypervertices and hyperedges.
- We propose to learn a joint representation space for hypervertices and hyperedges for extracting common information involved them, and train the model by minimizing the classification and the hypergraph reconstruction loss.
- We conduct experiments on both hypervertex classification and hyperedge classification tasks to show the effectiveness of our method.

Related Works

In this section, we discuss the related works regarding hypergraph learning. In a hypergraph, one hyperedge can connect more than two hypervertices, and two hypervertices can have more than one hyperedge connecting them (Gao et al. 2020, 2022; Lee and Shin 2023). Hypergraphs can model complex high-order relationships, thus have drawn much attention in recent years. In (Feng et al. 2019), HyperGraph Neural Network (HGNN) extends the spectral convolutional operation of standard graphs to propose a hypergraph convolutional model based on hypergraph Laplacian. In (Yadati et al. 2019), HyperGraph Convolutional Network (HyperGCN) further considers weighted pair-wise edges to propose a convolutional network for hypergraphs. Hypergraph has been applied in many applications, such as citation network (Jiang et al. 2019), social network (Yang et al. 2019), medical data analysis (Di et al. 2021), recommendation systems (Wang et al. 2020; Xia et al. 2021), etc.

To employ well-established methods of standard graphs (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018), some works of hypergraphs consider how to expand a hypergraph to obtain a graph, so that graph neural networks can be borrowed to address the problem of hypergraphs. Clique expansion constructs a graph whose vertices are the set of hypervertices, and a hyperedge is represented by a clique of hypervertices connected by it (Sun, Ji, and Ye 2008; Zhou, Huang, and Schölkopf 2006; Tu et al. 2018; Zhang et al. 2018). Clique expansion loses the high-order structural information. For example, multiple hyperedges between two hypervertices are reduced to only one edge. Star expansion (Zien, Schlag, and Chan 1999) constructs a bipartite graph whose vertices include hypervertices and hyperedges. Compared with clique expansion, star expansion preserves multiple connections between hypervertices, since each hyperedge is transformed to a vertex in the expanded graph. However, the direct connections between two hypervertices and two hyperedges are ignored. As a result, information passing from one hypervertex to another hypervertex goes through a hyperedge, which suffers from the issue of information decay (Wu, Yan, and Ng 2022). The similar issue also exists for information passing between two hyperedges. Compared with them, our cross expansion method involves richer relationships. The direct connections between two hypervertices and hyperedges are preserved, so that information passing between them does not rely on an intermediate.

To better leverage hyperedges, some methods consider how to learn representations for hyperedges. In (Wu and Ng 2022), Hypergraph Convolution on Nodes-Hyperedges (HCNH) learns separate representation spaces for hypervertices and hyperedges. In (Dong, Sawin, and Bengio 2020), Hypergraph Network with Hyperedge Neurons (HNHN) learns separate representations for hypervertices and hyperedges alternately. Based on this, in (Wu, Yan, and Ng 2022), both hypervertices and hyperedges are leveraged to learn representations for hypervertices and hyperedges. Different from these methods considering separate spaces for hypervertices and hyperedges, based on our constructed graph by cross expansion, we propose to learn a joint representation space for both hypervertices and hyperedges, so that shared information between them can be captured in the embedding space, which is beneficial for improving the performance.

notation	description
$\mathcal{G}$	a hypergraph
$\mathcal{V}$	the set of hypervertices
v	a hypervertex in $\mathcal{G}$
$\mathcal{E}$	the set of hyperedges
e	a hyperedge in $\mathcal{G}$
$\mathbf{H}$	the incidence matrix of $\mathcal{G}$
$\tilde{\mathcal{G}}$	the standard graph by cross expansion
$\tilde{\mathcal{V}}$	the set of vertices in $\tilde{\mathcal{G}}$
$\tilde{v}$	a vertex in $\tilde{\mathcal{G}}$
$\tilde{\mathcal{E}}$	the set of edges in $\tilde{\mathcal{G}}$
$\tilde{e}$	an edge in $\tilde{\mathcal{G}}$
$\mathbf{A}$	the adjacency matrix of $\tilde{\mathcal{G}}$

Table 1: Notations and descriptions.

Hypergraph Expansion

In this section, we present the notation used in the paper, and then present our cross expansion method for hypergraphs.

Preliminary of Hypergraph

Table 1 lists the notations used in this paper. A hypergraph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of hypervertices, and $\mathcal{E}$ is the set of hyperedges. The structure of the hypergraph $\mathcal{G}$ is represented by an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$. From the perspective of hypervertices, the element $\mathbf{H}(v,e)$ indicates if the hypervertex v is in the hyperedge e, *i.e.*,

$$\mathbf{H}(v,e) = \begin{cases} 1 & \text{if } v \in e, \\ 0 & \text{if } v \notin e. \end{cases} \tag{1}$$

We consider transductive classification problem on a hypergraph following (Wu, Yan, and Ng 2022). For hypervertex classificaiton, the label space is $\mathcal{Y}_v = \{1, \ldots, C_v\}$, where C_v is the number of classes. $\mathcal{V}^l$ is the set of labeled hypervertices, and $\mathcal{V}^u$ is the set of unlabeled hypervertices. The target of hypervertex classification is to train a classifier on $\mathcal{V}$ to predict labels for $\mathcal{V}^u$. Similarly, for hyperedge classificaiton, the label space is $\mathcal{Y}_e = \{1, \ldots, C_e\}$, where C_e is the number of classes. $\mathcal{E}^l$ is the set of labeled hyperedges, and $\mathcal{E}^u$ is the set of unlabeled hyperedges. The target of hyperedge classification is to train a classifier on $\mathcal{E}$ to predict labels for $\mathcal{E}^u$.

Cross Expansion

In this part, we propose a cross expansion method to transform a hypergraph to a graph with higher-order information preserved. Figure 1 illustrates various hypergraph expansion methods, including clique expansion, star expansion, and our proposed method cross expansion.

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ is induced by the cross expansion. The vertex set $\tilde{\mathcal{V}}$ includes the original hypervertices and hyperedges, *i.e.*, $\tilde{\mathcal{V}} = \mathcal{V} \cup \mathcal{E}$. The edge set $\tilde{\mathcal{E}}$ includes all the edges in the expanded graph $\tilde{\mathcal{G}}$. The edge $\tilde{e}(\tilde{v}, \tilde{v}')$ connects two vertices $\tilde{v}$ and $\tilde{v}'$, where either $\tilde{v}$ or $\tilde{v}'$ could be a hypervertex or a hyperedge in the original hypergraph $\mathcal{G}$. The corresponding adjacency matrix $\mathbf{A} \in \{0,1\}^{|\tilde{\mathcal{V}}|\times|\tilde{\mathcal{V}}|}$ is defined by the following rules:

1. for $\tilde{v} = v \in \mathcal{V}$, $\tilde{v}' = v' \in \mathcal{V}$, if $\exists e \in \mathcal{E}$ such that $\tilde{v} \in e$ & $\tilde{v}' \in e$, then the edge $\tilde{e}(\tilde{v}, \tilde{v}')$ exists, and $\mathbf{A}(\tilde{v}, \tilde{v}') = 1$;
2. for $\tilde{v} = e \in \mathcal{E}$, $\tilde{v}' = e' \in \mathcal{E}$, if $\exists v \in \mathcal{V}$ such that $\tilde{v} \in v$ & $\tilde{v}' \in v$, then the edge $\tilde{e}(\tilde{v}, \tilde{v}')$ exists, and $\mathbf{A}(\tilde{v}, \tilde{v}') = 1$;
3. for $\tilde{v} = v \in \mathcal{V}$, $\tilde{v}' = e \in \mathcal{E}$, if $v \in e$, then the edge $\tilde{e}(\tilde{v}, \tilde{v}')$ exists, $\mathbf{A}(\tilde{v}, \tilde{v}') = \mathbf{A}(\tilde{v}', \tilde{v}) = 1$;
4. otherwise, the edge $\tilde{e}(\tilde{v}, \tilde{v}')$ does not exist, and $\mathbf{A}(\tilde{v}, \tilde{v}') = 0$;

The adjacency matrix $\mathbf{A}$ includes following four blocks

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{v-v} & \mathbf{A}_{v-e} \\ \mathbf{A}_{e-v} & \mathbf{A}_{e-e} \end{bmatrix}, \tag{2}$$

where adjacency matrix $\mathbf{A}_{\alpha-\beta}$ denotes the relations from α to β, where α and β are v (*i.e.*, hypervertices) or e (*i.e.*, hyperedges). Based on this adjacency matrix, connections between two hypervertices (*resp.*, two hyperedges) are constructed without an intermediate connector. Based on our expanded graph, we propose to learn a joint embedding space for both hypervertices and hyperedges for better representation ability, which is given in the next section.

Comparison with Clique Expansion and Star Expansion. Now we discuss the relationship between our cross expansion and traditional clique expansion and star expansion. Rule 1 constructs the connections between two hypervertices, Rule 2 constructs the connections between two hyperedges, and Rule 3 constructs the connections between a hypervertex and a hyperedge. Clique expansion considers only hypervertices and Rule 1, and the edges constructed by Rules 2 and 3 do not exist in the clique expanded graph. Star expansion involves both hypervertices and hyperedges, while the edges are constructed by only Rule 3 without considering direct connections between two hypervertices or connections between two hyperedges. In star expansion, information passing between two hypervertices is through an intermediate hyperedge without direct connection from these two hypervertices, resulting in information decaying between them. The same issue exists for feature aggregation among hyperedges if hyperedge representation learning is required.

We take Figure 1 as an example to describe the higher-order information captured by our method. (i) a hyperedge ($e3$) connects more than two hypervertices ($v1$, $v2$, and $v3$), which is preserved in cross expansion. (ii) Multiple hyperedges ($e1$ and $e3$) connect hypervertices ($v1$ and $v2$), which are preserved in cross expansion while ignored in clique expansion, since only one edge exists between $v1$ and $v2$ in Figure 1(b). (iii) Direct connections between hypervertices ($v1$ and $v2$) or hyperedges ($e1$ and $e3$) are preserved in cross expansion while ignored in star expansion.

Hypergraph Joint Representation Learning

After obtaining the transformed graph by cross expansion, we train a graph convolutional network to learn a joint representation space for all the vertices, including hypervertices

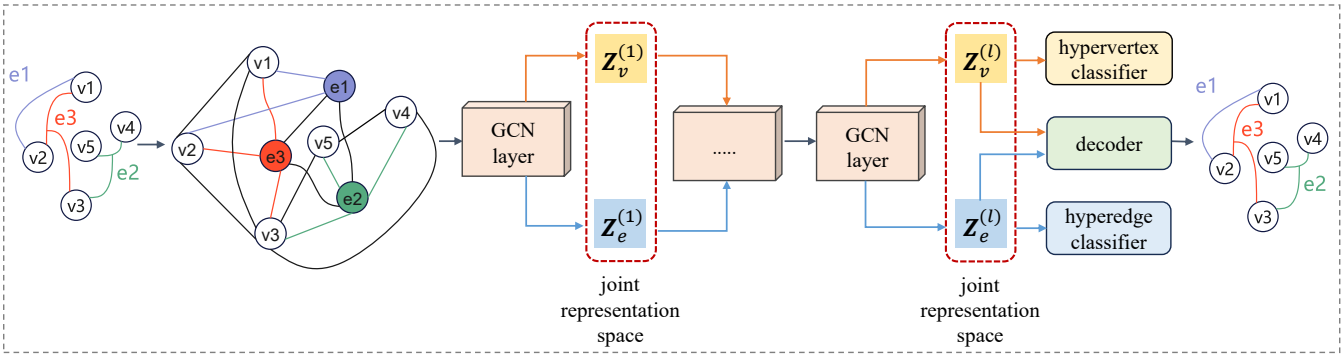


Figure 2: Illustrations of hypergraph joint representation learning for hypervertices and hyperedges.

and hyperedges. Figure 2 illustrates the architecture of our method. A hypergraph is first expanded to a graph by our cross expansion method. After that, hypervertices and hyperedges are transformed as vertices in the expanded graph and embedded into a joint representation space, which is trained by classification loss and hypergraph reconstruction loss.

In specific, given the embeddings of hypervertices and hyperedges in the τ-th layer $\mathbf{Z}_v^{(\tau)}$ and $\mathbf{Z}_e^{(\tau)}$, we apply GCN to learn embeddings for hypervertices and hyperedges in a joint representation space as follows

$$\mathbf{Z}_v^{(\tau+1)} = \sigma(\mathbf{A}_{v-v}\mathbf{Z}_v^{(\tau)}\mathbf{\Theta}^{(\tau)}) + \sigma(\mathbf{A}_{v-e}\mathbf{Z}_e^{(\tau)}\mathbf{\Theta}^{(\tau)}), \quad (3)$$

$$\mathbf{Z}_e^{(\tau+1)} = \sigma(\mathbf{A}_{e-v}\mathbf{Z}_v^{(\tau)}\mathbf{\Theta}^{(\tau)}) + \sigma(\mathbf{A}_{e-e}\mathbf{Z}_e^{(\tau)}\mathbf{\Theta}^{(\tau)}), \quad (4)$$

where $\sigma(\cdot)$ is the activation function, and $\mathbf{\Theta}$ is the parameters of the network. By doing this, higher-order information involved in the hypergraph is leveraged for joint representation learning, and information of a hypervertex (*resp.*, hyperedge) can be directly passed to another hypervertex (*resp.*, hyperedge) without information decay caused by an intermediate. For the initial features of a featureless hyperedge, one can adopt a mean-pooling or max-pooling operation based on its connected hypervertices to obtain features. Usually, a normalized adjacency matrix is applied in GCN (Kipf and Welling 2017). Normalization operation for hypergraphs is given in (Feng et al. 2019) and (Dong, Sawin, and Bengio 2020).

Learning for Classification and Reconstruction

For the hypervertex classification task, we employ the cross-entropy loss to define the following classification loss on labeled hypervertices

$$\mathcal{L}_{cls}^{v} = -\frac{1}{|\mathcal{V}^l|}\sum_{\tilde{v}\in\mathcal{V}^l}\mathbf{y}_{\tilde{v}}\log\hat{\mathbf{y}}_{\tilde{v}}, \quad (5)$$

where $\mathbf{y}_{\tilde{v}}$ is the ground-truth label vector, and $\hat{\mathbf{y}}_{\tilde{v}}$ is the predicted label vector.

For the hyperedge classification task, we reuse the notation $\tilde{v}$ to denote a hyperedge, since it is also a vertex in the graph by cross expansion. Since one hyperedge can associate with multiple classes, we employ the binary cross-entropy loss on labeled hyperedges considering each class individually:

$$\mathcal{L}_{cls}^{e} = -\frac{1}{|\mathcal{E}^l|}\sum_{\tilde{v}\in\mathcal{E}^l}\sum_{k=1}^{C_e} y_{\tilde{v},k}\log\hat{y}_{\tilde{v},k} + (1-y_{\tilde{v},k})\log(1-\hat{y}_{\tilde{v},k}), \quad (6)$$

where $y_{\tilde{v},k}$ is the label of the k-th class, and $\hat{y}_{\tilde{v},k}$ is the predicted value of the k-th class calculated by sigmoid.

Besides the loss functions for classification, we also apply a decoder to reconstruct the hypergraph for preserving structure information in the joint embedding space. Since expanded graph structure $\mathbf{A}$ is constructed based on the hypergraph structure $\mathbf{H}$, we only need to reconstruct $\mathbf{H}$. In specific, let $\mathbf{z}_v$ and $\mathbf{z}_e$ be the embeddings of a hypervertex v and a hyperedge e of the last layer, respectively. To reconstruct $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, we calculate the similarity $\mathbf{z}_v^\top\mathbf{z}_e$ and scale it to obtain $p(v,e)\in(0,1)$ by

$$p(v,e) = \text{sigmoid}(\mathbf{z}_v^\top\mathbf{z}_e), \quad (7)$$

which models the possibility of $\mathbf{H}(v,e)$ being 1 (Kipf and Welling 2016). The reconstruction loss is given as follows:

$$\mathcal{L}_r = \sum_v\sum_e \ell_{bce}\big(H(v,e), p(v,e)\big), \quad (8)$$

where the binary cross-entropy loss is defined as

$$\ell_{bce}\big(H(v,e), p(v,e)\big) = -H(v,e)\log p(v,e) - (1-H(v,e))\log(1-p(v,e)). \quad (9)$$

In summary, for hypervertex classification, we train the model by minimizing the following loss function

$$\mathcal{L}_v = \mathcal{L}_{cls}^{v} + \lambda_v\mathcal{L}_r, \quad (10)$$

where λ_v is the trade-off parameter. For hyperedge classification, we train the model by minimizing the following loss function

$$\mathcal{L}_e = \mathcal{L}_{cls}^{e} + \lambda_e\mathcal{L}_r, \quad (11)$$

where λ_e is the trade-off parameter.

We analyze the computational complexity of our proposed model. Let $|\mathcal{V}|$, $|\mathcal{E}|$, d_0, d_1 and d be the numbers of hypervertices, hyperedges, initial features, hidden-layer

Data	#Vertices	#Edges	#Classes	#Features	Label rate
Citeseer	1498	1107	6	3703	0.150
Pubmed	3840	7963	3	500	0.020
Cora	16313	7389	10	1000	0.052
DBLP	41302	22363	6	1425	0.042

Table 2: Statistical information on the datasets.

features and final-layer features, respectively. The complexities of the normalization operation, one-layer graph convolution, and graph reconstruction are $O(|\mathcal{V}|^2|\mathcal{E}| + |\mathcal{V}||\mathcal{E}|^2)$, $O(|\mathcal{V}|^2 d_0 + |\mathcal{V}|d_0 d_1 + |\mathcal{E}|d_0 d_1 + |\mathcal{E}|^2 d_0 + |\mathcal{V}||\mathcal{E}|d_0)$ and $O(|\mathcal{V}||\mathcal{E}|d)$, respectively.

Experiments

Datasets

We follow (Dong, Sawin, and Bengio 2020) to consider four benchmark datasets, including the co-citation datasets Citeseer (Bhattacharya and Getoor 2007) and Pubmed (Namata et al. 2012), the co-authorship datasets Cora (Sen et al. 2008), and DBLP (Rossi and Ahmed 2015). A co-citation dataset consists of a collection of papers and their citation links. To construct a hypergraph, a paper is represented by a hypervertex, and a set of hypervertices are connected to a hyperedge e if the papers corresponding to these hypervertices are cited by e. A co-authorship dataset consists of a collection of papers and their authors. To construct a hypergraph, a paper is represented by a hypervertex, and an author is represented by a hyperedge. A set of hypervertices are connected to a hyperedge e if the papers corresponding to these hypervertices are written by the author corresponding to e.

The vertex features of Citeseer and DBLP are bag-of-words vector representations, and the vertex features of Cora and Pubmed are based on the term frequency-inverse document frequency (TF-IDF). More detailed information can be found in (Dong, Sawin, and Bengio 2020; Wu, Yan, and Ng 2022). For fair comparisons, we strictly follow the experimental setting in (Wu, Yan, and Ng 2022), in which hypervertex classification is a multi-class problem, and hyperedge classification is a multi-label problem. Table 2 lists the statistical information of the used datasets.

Compared Methods

SVM (Cortes and Vapnik 1995): Support Vector Machine predicts labels based on features without considering structural information. **NN**: Nearest Neighbor predicts labels for a testing sample according to the label of its nearest neighbor in the training set. **GCN** (Kipf and Welling 2017): Graph Convolutional Network is a method for standard graphs. We adopt clique expansion to transform a hypergraph into a graph and then apply GCN on it. **HyperGCN** (Yadati et al. 2019): HyperGraph Convolutional Network considers weighted pair-wise edges to propose a convolutional network for hypergraphs. **HGNN** (Feng et al. 2019): HyperGraph Neural Network generalizes the spectral convolution operation for graphs to hypergraphs and proposes a hypergraph convolution model. **HNHN** (Dong, Sawin, and Bengio 2020): Hypergraph Network with Hyperedge Neurons learns separate representations for hypervertices and hyperedges alternately. **LEGCN** (Yang et al. 2022): Line Expansion Graph Convolutional Network proposes an expansion method to construct (vertex, edge) pair in a standard graph, and then apply GCN on it. **TriCL** (Lee and Shin 2023): Tridirectional Contrastive Learning applies contrastive learning for hypergraphs to learn embeddings for hypervertices. **HCNH** (Wu and Ng 2022): Hypergraph Convolution on Nodes-Hyperedges learns separate representation spaces for hypervertices and hyperedges, and a hypergraph reconstruction term is used to preserve structural information. **HCoN** (Wu, Yan, and Ng 2022): Hyper Collaborative Network learns separate representation spaces for hypervertices and hyperedges from previous representations of both hypervertices and hyperedges. In addition, a hypergraph reconstruction loss is also used to train the model.

Following (Wu, Yan, and Ng 2022), for each experiment, we run 100 trials and report the results of mean and standard derivation. We tune the hyperparameters λ_v and λ_e in the range $\{0.001, 0.01, 0.1, 1, 10\}$ on one trial, and apply the selected values for the remaining trials. All the experiments are conducted on the PyTorch platform. For our proposed method, we employ a two-layer graph convolutional network, and the hidden layer dimension is set to 512. We train the model for 200 epochs by the Adam optimizer and apply early stopping with a window of 100. The learning rate and weight decay factor are selected in the $\{0.0001, 0.001, 0.01, 0.1\}$. LeakyReLU with a negative input slope 0.2 is adopted as the activation function.

Evaluation Metrics

For the multi-class hypervertex classification problem, we adopt accuracy as the evaluation metric. For the multi-label hyperedge classification problem, we adopt six commonly used metrics, including precision, recall, F1, accuracy, exact match ratio, and 0/1 loss. For the last metric, the lower the better. While for the other metrics, the higher the better. Please refer to (Zhang and Zhou 2013) for the details of the evaluation metrics in multi-label problems.

Results on Hypervertex Classification

Table 3 presents the results of hypervertex classification in terms of the mean and standard derivation of accuracy. We strictly follow the setting in (Wu, Yan, and Ng 2022) and conduct 100 trials for LEGCN, TriCL, and HJRL to obtain the mean and standard derivation of accuracy, and quote the

Method	co-citation		co-authorship	
	Citeseer	Pubmed	Cora	DBLP
SVM	21.0 ± 1.3	41.3 ± 4.1	52.1 ± 1.1	27.2 ± 0.1
NN	27.0 ± 9.0	41.2 ± 2.7	28.9 ± 8.1	20.6 ± 6.2
GCN	63.0 ± 1.9	69.6 ± 6.3	40.0 ± 3.3	83.2 ± 2.6
HyperGCN	54.7 ± 9.8	60.0 ± 10.7	55.0 ± 0.9	71.3 ± 1.2
HGNN	61.1 ± 2.2	63.3 ± 2.2	58.2 ± 0.3	77.6 ± 0.4
HNHN	64.8 ± 1.6	75.9 ± 1.5	63.9 ± 0.8	85.1 ± 0.2
LEGCN	69.0 ± 1.4	74.2 ± 3.8	57.9 ± 1.4	88.4 ± 0.4
TriCL	72.0 ± 0.9	77.7 ± 2.6	61.1 ± 0.8	$\underline{88.9 \pm 0.3}$
HCNH	71.4 ± 1.2	77.1 ± 3.6	65.9 ± 0.5	–
HCoN	71.2 ± 2.7	80.4 ± 1.1	$\underline{66.5 \pm 0.5}$	88.0 ± 0.1
HJRL($\lambda_v = 0$)	70.3 ± 0.9	80.0 ± 2.2	65.7 ± 0.6	88.4 ± 0.2
HJRL(Clique)	71.4 ± 1.1	79.5 ± 2.1	62.9 ± 0.9	88.7 ± 0.1
HJRL(Star)	72.1 ± 1.0	78.5 ± 1.6	64.3 ± 0.9	86.8 ± 0.3
HJRL(Combined)	$\underline{72.7 \pm 0.8}$	$\underline{80.5 \pm 1.6}$	66.1 ± 0.8	88.6 ± 0.1
HJRL	$\mathbf{73.1 \pm 0.7}$	$\mathbf{81.0 \pm 1.5}$	$\mathbf{67.1 \pm 0.6}$	$\mathbf{89.1 \pm 0.1}$

Table 3: Accuracy of different methods on the semi-supervised hypervertex classification problem.

results of the other methods from (Wu, Yan, and Ng 2022). We draw the following observations. First, in general, hypergraph methods outperform the other methods, which verifies that high-order information involved in hypergraphs is beneficial for hypervertex classification. Second, HCoN obtains better performance compared with other hypergraph methods in general, which indicates that representation learning for hyperedges also helps to learn better hypervertex representations for classification. Third, our proposed method achieves the best performance compared with others, which demonstrates that our joint representation learning method is able to extract shared information of hypervertices and hyperedges for better hypergraph learning. We further perform significance test based on one-tailed and two-tailed t-test with the level 0.05. HJRL significantly outperforms the compared methods in all the comparisons.

Results on Hyperedge Classification

Table 4 presents the results of hyperedge classification in terms of the mean and standard derivation. We follow the setting in (Wu, Yan, and Ng 2022) and conduct 100 trials for our proposed method HJRL to obtain the mean and standard derivation of the metrics, and quote the results of the other methods from (Wu, Yan, and Ng 2022). We do not conduct experiments on DBLP here since it costs too much time for all the methods. We observe that our proposed method achieves the best performance or highly competitive performance on hyperedge classification tasks. This also demonstrates the effectiveness of our hypergraph joint representation learning method for both hypervertices and hyperedges.

Ablation Studies

In order to investigate the impact of the hypergraph reconstruction regularization term and the effectiveness of cross expansion, we conducted ablation studies for hypervertex classification and hyperedge classification tasks.

For the hypergraph reconstruction, we report the results with hypergraph reconstruction and without hypergraph reconstruction in Table 3 and Table 4, in which $\lambda_v = 0$ and $\lambda_e = 0$ represent that hypergraph reconstruction is removed. The inclusion of reconstruction achieves better results compared to that without reconstruction, which suggests that hypergraph reconstruction is able to preserve hypergraph structure information, leading to enhanced learning performance.

For the cross expansion, we applied clique expansion, star expansion, combined expansion of clique and star expansions to our learning model. The clique expansion cannot be applied to hyperedge classification since hyperedges are removed after expansion. The results in Table 3 and Table 4 demonstrate that our proposed cross expansion outperforms other hypergraph expansion methods, demonstrating the effectiveness of our approach.

Running Time Results

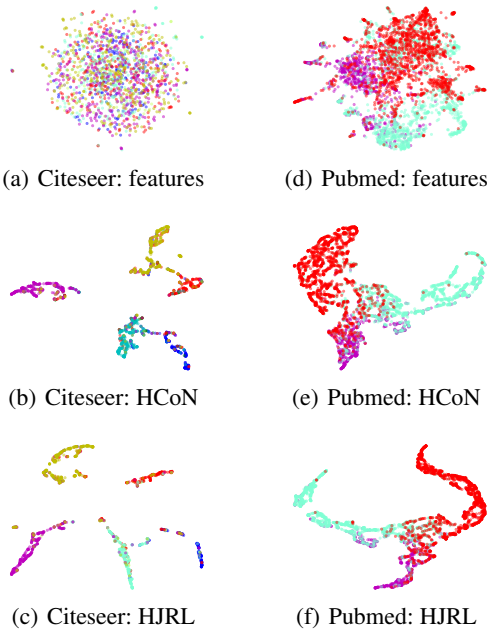
We evaluate the time efficiency of various methods for hypervertex classification tasks. This experiment was conducted on a Linux server with an NVIDIA RTX 4090 (24GB) graphics card. We run each method for 200 epochs without early stopping. The results measured in seconds are presented in Table 5, which includes training time, forward pass, calculation for objective, and backward pass. We observe that GCN requires the least time because of its simplicity, while suffering from limited performance. TriCL requires the longest runtime since the contrastive learning needs to construct pairs for training. Our method shows superior time efficiency compared to HCoN and TriCL.

Visualization

We apply the UMAP tool to visualize the original features and embeddings learned by HCoN and HJRL on Citeseer and Pubmed. Figure 3 shows the results, and different

Data	Method	Precision	Recall	F1-score	Accuracy	Exact Match Ratio	0/1 Loss
Citeseer co-citation	GCN	69.5 ± 1.7	68.6 ± 1.7	66.2 ± 1.2	59.9 ± 1.3	41.9 ± 1.8	58.1 ± 1.8
	HGNN	52.9 ± 1.7	46.4 ± 1.9	47.9 ± 1.7	44.3 ± 1.6	33.9 ± 1.6	66.1 ± 1.6
	HCoN	**82.4 ± 1.3**	73.4 ± 1.5	75.5 ± 1.1	70.7 ± 1.1	56.6 ± 1.3	43.4 ± 1.3
	HJRL($\lambda_e = 0$)	76.7 ± 1.5	84.8 ± 1.3	78.1 ± 1.3	72.7 ± 1.3	56.6 ± 1.7	43.4 ± 1.7
	HJRL(Star)	75.5 ± 1.2	85.9 ± 1.3	78.2 ± 1.0	73.2 ± 1.1	58.5 ± 1.4	41.5 ± 1.4
	HJRL(Combined)	76.0 ± 1.4	**90.0 ± 1.0**	80.0 ± 1.0	74.5 ± 1.2	58.7 ± 1.7	41.3 ± 1.7
	HJRL	79.0 ± 1.2	88.7 ± 1.0	**81.2 ± 0.9**	**75.6 ± 1.0**	**59.2 ± 1.4**	**40.8 ± 1.4**
Pubmed co-citation	GCN	87.1 ± 1.7	85.0 ± 1.7	82.7 ± 0.6	76.1 ± 0.8	56.6 ± 1.8	43.4 ± 1.8
	HGNN	83.8 ± 2.1	81.9 ± 2.3	79.7 ± 1.2	73.1 ± 1.4	53.9 ± 2.7	46.1 ± 2.7
	HCoN	90.4 ± 1.4	87.4 ± 1.4	85.9 ± 0.5	79.9 ± 0.7	61.9 ± 1.5	38.1 ± 1.5
	HJRL($\lambda_e = 0$)	88.3 ± 1.3	86.8 ± 1.7	84.3 ± 0.8	77.7 ± 1.1	57.8 ± 2.1	42.2 ± 2.1
	HJRL(Star)	**92.9 ± 3.0**	78.4 ± 2.9	81.2 ± 1.6	73.2 ± 2.0	47.6 ± 3.8	52.4 ± 3.8
	HJRL(Combined)	89.4 ± 2.3	85.1 ± 2.7	84.0 ± 1.1	76.9 ± 1.5	55.9 ± 3.3	44.1 ± 3.3
	HJRL	87.6 ± 1.2	**90.4 ± 1.3**	**86.1 ± 0.6**	**80.2 ± 0.7**	**62.7 ± 1.4**	**37.3 ± 1.4**
Cora co-authorship	GCN	35.4 ± 2.6	33.7 ± 4.9	32.0 ± 2.3	28.3 ± 2.0	18.9 ± 4.3	81.1 ± 4.3
	HGNN	17.0 ± 4.4	14.1 ± 3.4	14.9 ± 3.7	14.1 ± 3.4	11.8 ± 2.8	88.2 ± 2.8
	HCoN	52.8 ± 0.7	45.2 ± 0.9	46.6 ± 0.7	43.5 ± 0.6	35.3 ± 0.9	64.7 ± 0.9
	HJRL($\lambda_e = 0$)	61.0 ± 4.1	62.3 ± 1.4	58.2 ± 1.5	53.4 ± 1.1	40.0 ± 1.9	**59.0 ± 1.9**
	HJRL(Star)	56.3 ± 5.5	57.3 ± 3.8	53.6 ± 3.9	48.8 ± 3.2	36.1 ± 1.8	63.9 ± 1.8
	HJRL(Combined)	**62.1 ± 3.5**	60.0 ± 2.1	56.9 ± 1.9	51.5 ± 1.7	37.6 ± 2.2	62.4 ± 2.1
	HJRL	61.8 ± 4.1	**62.9 ± 1.5**	**58.7 ± 1.3**	**54.0 ± 1.1**	**41.1 ± 2.0**	**59.0 ± 2.0**

Table 4: Results in terms of *Precision* (↑), *Recall* (↑), *F1-score* (↑), *Accuracy* (↑), *Match Ratio* (↑), and *0/1 Loss* (↓) of different methods on the semi-supervised hyperedge classification problem.



(a) Citeseer: features (d) Pubmed: features

(b) Citeseer: HCoN (e) Pubmed: HCoN

(c) Citeseer: HJRL (f) Pubmed: HJRL

Figure 3: Visualization of original features and embeddings learned by HJRL of hypervertices on the Citeseer and Pubmed datasets. The color represents the class label.

	GCN	LEGCN	TriCL	HCoN	HJRL
Citeseer	0.75	2.20	16.85	3.42	2.13
Pubmed	0.93	2.63	104.65	7.73	5.25
Cora	3.29	3.00	-	19.14	16.17

Table 5: Wall-clock time (s) of different methods.

classes are indicated by colors. We observe that the original features of different classes distribute together, while HCoN and HJRL obtain more separatable embeddings. Compared with HCoN, the embeddings learned by HJRL have larger inter-class distances and smaller intra-class distances. This further demonstrates the effectiveness of our proposed method for hypergraph representation learning.

Conclusion

In this paper, we propose an expansion method to transform a hypergraph to a standard graph, and then design a joint learning model to embed both hypervertices and hyperedges into a shared representation space. Our expansion method preserves rich relationships in the hypergraph, and common knowledge involved in hypervertices and hyperedges are captured in the joint representation space. We also employ a hypergraph reconstruction objective to preserve structure information in the model. We conduct experiments on hypervertex and hyperedge tasks. The experimental results demonstrate the effectiveness of our proposed method.

Acknowledgments

This research was supported in part by National Natural Science Foundation of China (62206061, 62206111, 61876043, 61976052, 62206064), National Key R&D Program of China (2021ZD0111501), National Science Fund for Excellent Young Scholars (62122022), Guangzhou Basic and Applied Basic Research Foundation (2023A04J1700, 2023A04J1058), and Guangdong Provincial Science and Technology Innovation Strategy Fund (2019B121203012).

References

Bhattacharya, I.; and Getoor, L. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1): 5–es.

Cortes, C.; and Vapnik, V. 1995. Support-vector networks. *Machine learning*, 20: 273–297.

Di, D.; Shi, F.; Yan, F.; Xia, L.; Mo, Z.; Ding, Z.; Shan, F.; Song, B.; Li, S.; Wei, Y.; et al. 2021. Hypergraph learning for identification of COVID-19 with CT imaging. *Medical Image Analysis*, 68: 101910.

Dong, Y.; Sawin, W.; and Bengio, Y. 2020. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3558–3565.

Gao, Y.; Feng, Y.; Ji, S.; and Ji, R. 2022. HGNN+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3181–3199.

Gao, Y.; Zhang, Z.; Lin, H.; Zhao, X.; Du, S.; and Zou, C. 2020. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5): 2548–2566.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Jiang, J.; Wei, Y.; Feng, Y.; Cao, J.; and Gao, Y. 2019. Dynamic Hypergraph Neural Networks. In *International Joint Conference on Artificial Intelligence*, 2635–2641.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. In *Neural Information Processing Systems*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Lee, D.; and Shin, K. 2023. I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8456–8464.

Namata, G.; London, B.; Getoor, L.; Huang, B.; and EDU, U. 2012. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, volume 8.

Rossi, R.; and Ahmed, N. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI Conference on Artificial Intelligence*, volume 29.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Sun, L.; Ji, S.; and Ye, J. 2008. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 668–676.

Tu, K.; Cui, P.; Wang, X.; Wang, F.; and Zhu, W. 2018. Structural deep embedding for hyper-networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, J.; Ding, K.; Hong, L.; Liu, H.; and Caverlee, J. 2020. Next-item recommendation with sequential hypergraphs. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 1101–1110.

Wu, H.; and Ng, M. K. 2022. Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4): 1–19.

Wu, H.; Yan, Y.; and Ng, M. K.-P. 2022. Hypergraph collaborative network on vertices and hyperedges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3245–3258.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xia, X.; Yin, H.; Yu, J.; Wang, Q.; Cui, L.; and Zhang, X. 2021. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In *AAAI Conference on Artificial Intelligence*.

Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; and Talukdar, P. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32.

Yang, C.; Wang, R.; Yao, S.; and Abdelzaher, T. 2022. Semi-supervised hypergraph node classification on hypergraph line expansion. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2352–2361.

Yang, D.; Qu, B.; Yang, J.; and Cudre-Mauroux, P. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *The world wide web conference*, 2147–2157.

Yang, H.; Ma, K.; and Cheng, J. 2021. Rethinking graph regularization for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4573–4581.

Zhang, M.; Cui, Z.; Jiang, S.; and Chen, Y. 2018. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhang, M.-L.; and Zhou, Z.-H. 2013. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8): 1819–1837.

Zhou, D.; Huang, J.; and Schölkopf, B. 2006. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19.

Zien, J. Y.; Schlag, M. D.; and Chan, P. K. 1999. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 18(9): 1389–1399.