EAI Endorsed Transactions
on e-Learning

Research Article EAI.EU

A Review of Hypergraph Neural Networks

Xinke Zhi[1,*]

[1] School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, Henan 454000, P R China

Abstract

In recent years, Graph Neural Networks (GNNs) have seen notable success in fields such as recommendation systems and natural language processing, largely due to the availability of vast amounts of data and powerful computational resources. GNNs are primarily designed to work with graph data that involve pairwise relationships. However, in many real-world networks, the relationships between entities are complex and go beyond simple pairwise connections, as seen in scientific collaboration networks, protein networks, and similar domains. If these complex relationships are directly represented as pairwise relationships using graph structures, it can lead to information loss. A hypergraph is a special kind of graph-structured data, can represent higher-order relationships that cannot be fully captured by graphs, thereby addressing the limitations of graphs. In light of this, researchers have begun to focus on how to extend neural networks to hypergraphs, leading to the proposal of hypergraph neural network (HGNN) models for downstream tasks. Therefore, this paper reviews the existing hypergraph neural network models. The review is conducted from two perspectives: spectral analysis methods and neural network methods on hypergraphs, discussing both unfolded and non-unfolded methods and further subdividing them based on their algorithm characteristics and application scenarios. Subsequently, the design concepts of various algorithms are analyzed and compared, and the advantages and disadvantages of each type of algorithm are summarized based on experimental results. Finally, potential future research directions in hypergraph learning are discussed.

Keywords: Graph Neural Networks, Hypergraph Neural Network, Graph Structure, Hypergraph Structure

Received on 30 April 2024, accepted on 5 September 2024, published on 14 October 2024

doi: 10.4108/eetel.7064

1. Introduction

Traditional deep learning models like Convolutional Neural Networks (CNNs) [1] and Recurrent Neural Networks (RNNs) [2] have shown exceptional performance in tasks involving images, audio, and various other data types. This success is largely attributed to their capacity to process Euclidean data, taking advantage of translational invariance and local connectivity. The core ideas behind CNNs include local connections, parameter sharing, pooling, and multi-layer usage [3]. These techniques not only reduce the complexity of the network model and decrease the number of weights but also maintain the model's expressive capability. RNNs, on the other hand, excel at handling variable-length sequential data, allowing them to extract temporal and semantic information from the data. Long Short-Term Memory (LSTM) [4] improves upon traditional RNNs through a gating mechanism, while the Gated Recurrent Unit (GRU) [5], a simplified variant of LSTM, contains only a reset gate and update gate, streamlining LSTM architecture.

Despite the effectiveness of these traditional deep learning methods in extracting features from Euclidean data, many real-world applications involve non-Euclidean data that must be represented using graph structures, such as social networks, transportation networks, knowledge graphs, and protein networks [6]. As a result, researchers have increasingly become interested in extending deep learning methods to graph data.

In recent years, inspired by CNNs and RNNs, researchers have innovatively designed deep learning methods specifically for handling graph data, collectively referred to as Graph Neural Networks (GNNs). Graphs, as an

*Corresponding author. Email: zxk@home.hpu.edu.cn

efficient structure for expressing relationships, are widely used for modeling pairwise relationships, such as citation relationships between papers, social networks, protein interactions, and more. However, in many scenarios, besides pairwise relationships, there are numerous non-pairwise relationships that simple graph structures struggle to express, such as community structures in social networks and cluster structures in feature relationships. In these scenarios, researchers often find it difficult or even impossible to distinguish the interactions between samples within different structures. Due to their ability to include any number of nodes in a single edge, hypergraphs are better suited to represent these complex data relationships. Specifically, a hypergraph is a graph structure where an edge can contain any number of nodes, formally expressed as follows: a hypergraph $H = (X, E)$, where X is a set of elements called nodes, and E is a non-empty subset of X, referred to as hyperedges or edges. Figure 1 illustrates a simple example of a hypergraph.

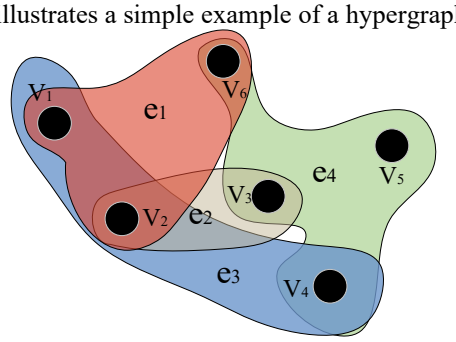

Figure 1. Example of hypergraph

Researchers have gradually be[gun] to explor[e] areas such as higher-order interactions in hyp[ergraphs], dy[namic] hypergraphs [8], and indec[omposable] hypergraphs [9], leveraging the powerful le[arning] capabilit[ies] and flexibility of neural networks. F[rom a] methodologica[l] erspective, spectral analysis m[ethods] h[ave] traditionally been the mainstream in gra[ph] learning and [held] a significant position in hypergraph l[earning] as well. Be[fore] the introduction of neural netwo[rks] into hypergraph learning, most studies were based on [spe]ctral the[ory]. However, with the application of neural net[works] in [hy]pergraph learning, researchers have proposed a se[ries] of studies t[hat] extend traditional pairwise g[raph] work l[earn]ing als[o] thms to hypergraphs [10–12]. Hypergra[ph] learni[ng] a process that propagates over hypergra[ph] structures, as introduced in the work of Zhou et al. [10], [...] adopted a spectral analysis method based [on] ypergraphs. These methods typically involve rigorous m[athe]matical derivations but also face several limitations due to th[eir].

T[he] advent of neural networks has revitalized hypergraph learning, making it an increasingly popular research area. Applying neural network architectures to hypergraph learning has led to notable advancements, with methods like Hyperedge-Based Embedding (HEBE) [13] standing out. HEBE aims to create representations for entities within hypergraphs, influencing subsequent neural network approaches. Despite its contributions, HEBE has been criticized for poor performance with sparse hypergraphs, as highlighted by Tu et al. [9]. This led to the development of the Deep Hyper-Network Embedding (DHNE) model. However, DHNE's reliance on multilayer perceptrons to model tuple relationships limits its applicability to uniform hypergraphs. When applying DHNE to [non-]uniform heterogeneous hypergraphs, the co[nsu]mption [...] of computational resources increases signi[fi]cantly, and [the] model's generalization ability is compr[omi]sed. To addres[s] these challenges, Zhang et al. [14] propose[d] Hyper-SAGNN, which introduced a self-attention [mec]hanism a[im]ed at bette[r] handling the learning and [ex]ploration of [non-]unif[orm] heterogeneous hypergraphs.

However, most of the [aforementi]oned works overlook a potential issue when d[eal]ing [w]ith [var]ious chara[cte]ristics of hypergraphs: their d[eco]mposabil[ity]. Hypergra[ph] unfolding is a classic metho[d] [for] hypergraph a[na]lysi[s], which expands hyperedges in[to] ordi[nary] edges, such as [clique] expansion [15] and star expansion [...]. However, these expanding techniques rely on the [de]composability of hypergraphs, mean[ing] that any subset of a hyperedge can form a new hy[per]edge. For indecomposable hypergraphs, Tu et al. [9] p[ro]posed the DHN[E] model and theoretically proved that in ex[ist]ing methods, l[in]ear similarity measures in standard em[bed]ding spaces [can]not maintain the indecomposability of hyper[gra]ph n[et]works. Therefore, regarding the decompos[abil]ity characteristics of model algorithms, this [...] will further review the existing hypergraph learning [me]thods.

2. Definition of hypergraph Learning

2.1. Symbol Definition

Table 1. Description of symbols

Sign	Meaning
G	Hypergraph
G_D	Directed Hypergraph
H_G	Incidence Matrix of the Hypergraph
H_{G_D}	Incidence Matrix of the Directed Hypergraph
$H_{G_D}^{head}$	Head Incidence Matrix of the Directed Hypergraph
$H_{G_D}^{tail}$	Tail Incidence Matrix of the Directed Hypergraph
X_G	Feature Matrix of the Vertices
x_i	Feature of Vertex i
$L(G)$	Line Graph of the Hypergraph G
$A_{L(G)}$	Adjacency Matrix of the Line Graph $L(G)$ of G
$\mathcal{L}_G$	Laplacian Matrix of G
$\mathcal{L}_{G_D}$	Laplacian Matrix of G_D
I	Identity Matrix
$\odot$	Hadamard Product
σ, σ_{att}	Activation Function

2.2. Hypergraph definition

Let $G = (V_G, E_G, W_G)$ represent a hypergraph, where V_G is the vertex set, and E_G is the hyperedge set. G contains $|V_G| = N_G$ vertices and $|E_G| = M_G$ hyperedges. If v_G^i is a vertex, then a hyperedge $e_G^i = \{v_G^{m_e}, \dots, v_G^{n_e}\}$, where $1 \le m_e \le N_G, 1 \le n_e \le N_G$. W_G represents the weight matrix of the hyperedges.

Typically, a hypergraph is represented by the incidence matrix $H_G \in \mathbb{R}^{(N_G, M_G)}$, defined as:

$$h_{i,j} = h(v_G^i, e_G^j) = \begin{cases} 1, if\, v_G^i \in e_G^j \\ 0, if\, v_G^i \notin e_G^j \end{cases} \quad (1)$$

where $h_{i,j}$ is an element of H_G, $i = 1,2, \dots, N_G$, and $j = 1,2, \dots, M_G$.

If $v_G^i \in V_G$, the degree of the vertex, denoted by $d(v_G^i)$, is defined as the number of hyperedges containing v_G^j:

$$d(v_G^i) = \sum_{e_G \in E_G} h(v_G^i, e_G)\, W_G(e_G) \quad (2)$$

If $e_G^j \in E_G$, the degree of the hyperedge, denoted by $d(e_G^j)$, is defined as the number of vertices contained in the hyperedge e_G^j:

$$d(e_G^j) = \sum_{v_G^i \in V_G} h(v_G^i, e_G^j) \quad (3)$$

D_{V_G} and D_{E_G} represent the degree matrices of the vertices and hyperedges, respectively.

In fact, a graph is essentially a specific type of hypergraph. When a hypergraph has hyperedges that each connect only two vertices, it simplifies into a standard graph. Figure 2(a) illustrates a standard graph where every edge links two vertices, while Figure 2(b) depicts a hypergraph with hyperedges connecting three or four vertices. If the hyperedges in Figure 2(b) were reduced to connect only two vertices each, the hypergraph would revert to a graph, as shown in Figure 2(c) and (d).

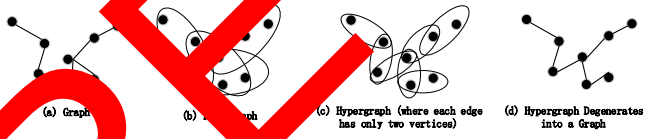

Figure 2. Graph and hypergraph

3. Hypergraph Learning Methods

Due to the more complex and higher-dimensional structure of hypergraphs, traditional graph algorithms are challenging to apply directly to hypergraph problems. With the advancement of hypergraph research and applications, several typical hypergraph-solving algorithms have emerged, such as spectral analysis, neural networks, and other methods. These can be further categorized into expansion-based methods and non-expansion-based methods, depending on their modeling approach.

3.1. Spectral Analysis Methods

Spectral analysis on hypergraphs is a traditional mainstream method in the field of hypergraph learning, involving matrix analysis based on spectral theory. These methods typically solve the optimal solution of the objective function through rigorous mathematical derivations, with the underlying theoretical foundations playing a crucial role in guiding algorithm model design. In recent years, the widely studied Graph Convolutional Networks (GCNs) [16,17] have gradually evolved from spectral theory combined with deep learning. This section focuses on spectral analysis methods on hypergraphs.

3.1.1 Expansion-Based Spectral Analysis Methods

Expansion-based spectral analysis methods generally simplify hypergraph learning problems by converting hypergraphs into traditional pairwise graph networks and then solving them through the spectral properties of the Laplacian matrix. These methods are often used in the study of homogeneous hypergraphs.

Star Expansion (SE) and Clique Expansion (CE) are two classic hypergraph expansion methods and have been widely applied in the design of many hypergraph algorithms. The earliest standard star expansion and clique expansion were proposed based on spectral theory [15,7].

The definition of the Star Expansion algorithm is as follows: For a hypergraph $G(V, E)$, the Star Expansion establishes a corresponding graph $G^*(V^*, E^*)$. In the graph $G^*(V^*, E^*)$, a new node $e \in E$ is introduced for each hyperedge in the hypergraph $G(V, E)$, resulting in $V^* = V \cup E$. Subsequently, each new node e is connected to all the nodes in its corresponding hyperedge, i.e., $E^* = \{(u, e): u \in e, e \in E\}$. It is important to note that each hyperedge in the set E corresponds to a star structure in the graph $G^*(V^*, E^*)$, making G^* a pairwise graph network. The Star Expansion redistributes the weight of each hyperedge to the corresponding edges in the pairwise graph as follows:

$$w^*(u, e) = \frac{w(e)}{\delta(e)} \quad (4)$$

where $\delta(e)$ is the number of nodes in the hyperedge e. After obtaining the expanded pairwise relationship graph G^*, the normalized Laplacian matrix L^* of G^* can be defined as follows [7]:

$$L^* = \begin{bmatrix} I & -A \\ -A^T & I \end{bmatrix} \quad (5)$$

Where A is a $|V| \times |E|$ matrix:

$$A_{ue} = \frac{h(u,e) w^*(u,e)}{\sqrt{d^*(u)}\sqrt{d^*(e)}} \quad (6)$$

Where $d^*(u)$ and $d^*(e)$ represent the sum of the weights of all connecting edges for node u and the newly added hyperedge node e in G^*. Formally, for $u \in V$, $d^*(u) = \sum_{e \in E} h(u,e) w^*(u,e)$, and for $e \in E$, $d^*(e) = \sum_{u \in V} h(u,e) w^*(u,e)$. The schematic diagram of the Star Expansion is shown in Figure 3(a).

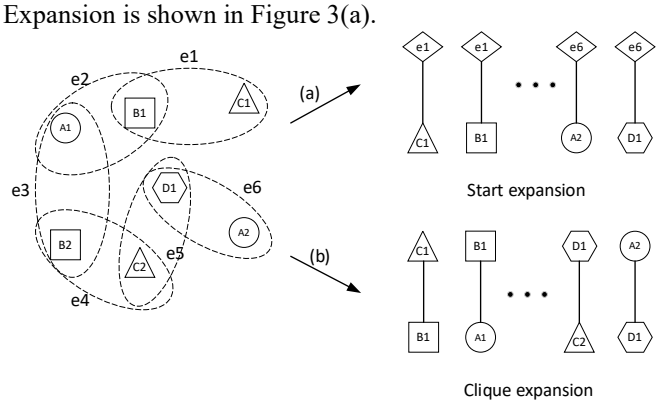

Figure 3. Start expansion and Clique expansion

The clique expansion algorithm constructs a graph $G^c(V^c, E^c)$ from the original hypergraph $G(V, E)$. Each hyperedge $e = (u_1, \ldots, u_{\delta(e)}) \in E$ in the original hypergraph is replaced in G^c by a fully connected structure formed by connecting every pair of nodes within the hyperedge, resulting in $E^c = \{(u,v): u, v \in e, e \in E\}$. It's important to note that each hyperedge forms a clique structure in G^c. The weight $w^c(u,v)$ of an edge formed by nodes u and v in G^c needs to minimize the difference between $w^c(u,v)$ and the weight of the original hyperedge e that contains u and v, which is expressed as $w^c(u,v) = argmin \sum_{e \in E; u,v \in e} (w^c(u,v) - w(e))^2$. This means after the clique expansion, the weight of each edge in G^c is related to the weight $w(u)$ of the original hyperedge e, specifically:

$$w^c(u,v) = \mu \sum_{u,v \in e} w(e) = \mu \sum_e h(v,e) h(v,e) w(e) \quad (7)$$

where μ is a fixed scalar. Similarly, after obtaining the graph G^c, the normalized Laplacian matrix L^c for the clique expansion is defined as follows [6]:

$$L^c = I - D_c^{-1/2} W H D_c^{-1/2} \quad (8)$$

where D_c is the degree matrix of G^c, H is the incidence matrix of the original hypergraph, and W is the weight matrix represented by w^c. A schematic diagram is shown in Figure 3(b). The difference between the two expansion methods lies in the following: in the clique expansion, nodes within the same hyperedge are directly connected, clearly reflecting the similarity between nodes; whereas in the star expansion, nodes within the same hyperedge are indirectly connected through the hyperedge node, exhibiting implicit similarity. The commonality between both methods is that they convert the hypergraph into a pairwise graph. In a k-uniform hypergraph, the eigenvectors of the pairwise graph $G_{ex}^*(V^*, E^*)$ obtained by scaling the weights of the star-expanded graph by $(\delta(e) - 1)\delta(e)$ times are consistent with those of the normalized clique-expanded graph G_c. Although the structure of the pairwise graphs obtained through star expansion and clique expansion differs significantly in uniform hypergraphs, the results of the Laplacian matrix eigen-decomposition of the two expanded graphs are mathematically similar. However, in non-uniform hypergraphs, clique expansion and star expansion differ in that clique expansion assigns more weight to larger hyperedges, leading to differences in the eigen decomposition results [7].

3.1.2 Non-Expansive Spectral Analysis Methods

Unlike expansion-based methods, non-expansive spectral analysis methods directly model the hypergraph, constructing the Laplacian matrix directly on the hypergraph. This modeling process ensures the completeness of the hypergraph information.

Carletti et al. [18] demonstrated in their study on random walks on hypergraphs (RWH) that there are significant differences between their proposed method, which directly models random walks on hypergraphs, and the traditional random walk algorithms used on the projection graph (i.e., the pairwise graph obtained after clique expansion). Through algorithmic derivation on constructed graph data, Carletti et al. [18] concluded that their hypergraph random walk algorithm is more sensitive to higher-order structures compared to traditional methods. This study indirectly suggests that algorithms designed based on expanded graphs and non-expansive methods directly designed on hypergraphs may perform differently.

Bolla [19] proposed one of the early non-expansive matrix decomposition methods, known as the Laplacian method. Bolla [19] aimed to find a k-partition of nodes on a connected hypergraph, minimizing the number of edges in the corresponding cut set. To this end, he defined a hypergraph Laplacian matrix L^o for an unweighted hypergraph as follows:

$$L^o = D_v - H D_e^{-1} H^T \quad (9)$$

where D_v is the diagonal matrix of node degrees, D_e is the diagonal matrix of hyperedge degrees, and H is the incidence matrix of the hypergraph. The eigenvectors of L^o define the optimal Euclidean distance embedding for hypergraph nodes. Bolla [19] demonstrated that the matrix decomposition method based on L^o can effectively solve the minimum cut problem for hypergraphs.

Subsequently, Zhou et al. [10] extended traditional matrix decomposition methods on undirected graphs to hypergraphs, proposing the definition of the N-cut algorithm on hypergraphs, further advancing the development of non-expansive matrix decomposition methods on hypergraphs. They formalized the expression for hypergraph partitioning. As an NP-complete problem, Zhou et al. [10] relaxed it into a real-valued optimization problem and then introduced spectral theory. Analogous to the Laplacian matrix of

traditional graphs, they defined the hypergraph Laplacian matrix Δ as:

$$\Delta = \mathrm{I} - D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} \tag{10}$$

Where I is the identity matrix, D_v is the node degree matrix, H is the incidence matrix of the hypergraph, W is the hyperedge weight matrix, and D_e is the hyperedge degree matrix.

According to standard linear algebra theory, the solution to the hypergraph minimum cut problem corresponds to the eigenvector associated with the smallest non-zero eigenvalue of the Laplacian matrix Δ, and thus, the minimum cut problem for hypergraphs can be solved through matrix decomposition methods. By left- and right-multiplying Bolla's [19] Laplacian matrix by $D_v^{-1/2}$, and comparing it with Zhou et al.'s [10] Laplacian matrix, it is not difficult to see that Zhou et al. [10] essentially extended Bolla's [19] method from unweighted hypergraphs to weighted hypergraphs, without changing the matrix's essence, but adding a step of matrix normalization. However, this extension made Zhou et al.'s [10] Laplacian matrix more broadly applicable, further extending the related theory to fields such as hypergraph embedding, clustering, and propagation inference. Subsequently, Zhou et al.'s [10] work became a classic algorithm, widely adopted by many researchers.

Huang et al. [20], when addressing the problem of image segmentation, used over-segmented image blocks as vertices to construct a hypergraph and then applied the N-cut algorithm proposed by Zhou et al. [10] to segment the hypergraph, thereby solving the problem of over-segmentation in images. Similarly, Purkait et al. [21] utilized the N-cut algorithm to cluster and segment the graph after obtaining a clustered hypergraph, while designing a large-scale hyperedge clustering algorithm. Huang et al. [22] combined the propagation inference theory proposed by Zhou et al. [10] with a probability matrix on the hypergraph and proposed a probability-based non-expansive hypergraph ranking matrix decomposition algorithm. Ma et al. [23], based on Zhou et al.'s [10] standard hypergraph Laplacian operator, further proposed its nonlinear extension—Hypergraph p-Laplacian regularization (HpLapR), to preserve the geometric probability distribution of the data.

Additionally, there is also a class of hypergraph algorithms based on random walks and spectral analysis methods. J. Huang et al. [24] proposed the Hyper2vec algorithm, based on a biased random walk strategy on hypergraphs. In Hyper2vec, the probability transition formula for a node is as follows:

$$\rho_2'(x|v \cdot U) = \frac{\alpha(x|v,U)\cdot\beta(x)\sum_{e\in E} w(\mathrm{e})\frac{h(v,\mathrm{e})h(x,e)}{\mathrm{d}(v)\ \ \delta(\mathrm{e})}}{Z} \tag{11}$$

where $h(v,\mathrm{e})$ and $h(x,e)$ are elements of the hypergraph incidence matrix H, $w(\mathrm{e})$ is the hyperedge weight, $\mathrm{d}(v)$ is the degree of node v, $\delta(\mathrm{e})$ is the degree of hyperedge e, and $\alpha(x|v,U)$ and $\beta(x)$ are piecewise functions that guide the random walk process. The matrix expression of the above formula is $P = ABD_v^{-1}HWD_e^{-1}H^T/Z$. It can be seen that matrix PPP is essentially the random walk form of the hypergraph Laplacian matrix proposed by Zhou et al. [10], multiplied by a guiding function. Hyper2vec adds a guiding function to the Laplacian operator proposed by Zhou et al. [10], allowing the method to adapt to different network structures and better preserve the network's structural and inherent properties.

Recently, Carletti et al. [18] proposed a hypergraph random walk algorithm driven by a generalized Laplacian operator. This algorithm draws inspiration from the property in microscopic physical models where multiple neighboring objects within the same hyperedge are more likely to exchange. The Laplacian operator constructed by this method is equivalent to the hypergraph Laplacian operator proposed by Zhou et al. [10] with the hyperedge weight allocation function being $C_{\alpha\alpha}(C_{\alpha\alpha}-1)$. In other words, the random walk algorithm proposed by Carletti et al. [18] is essentially a special case of the work by Zhou et al. [10]. Compared to traditional random walks on pairwise graphs, this method is more sensitive to higher-order hyperedges; that is, when the scale of the hyperedge changes, the change in node importance within the hyperedge calculated by this method is greater than that of traditional methods.

3.2. Neural Network Methods

In recent years, with the deepening of neural network research, researchers have introduced neural network methods into various fields, including hypergraph learning. The advantage of spectral analysis methods lies in their strong mathematical interpretability, but this also leads to a lack of flexibility, making them applicable to a relatively limited range of hypergraph scenarios. Additionally, many spectral analysis methods face challenges when applied directly to large-scale hypergraph mining due to their inherent characteristics. Neural network algorithms have effectively addressed these limitations. This section will explore how neural network approaches are applied to hypergraphs.

3.2.1 Expansive Neural Network Methods

Drawing inspiration from graph convolutional networks, Feng et al. [25] introduced a hypergraph neural network framework (HGNN). This framework integrates the hypergraph Laplacian matrix $\Delta = \mathrm{I} - \theta$, as proposed by Zhou et al. [10], into conventional graph convolutional neural networks. On this basis, they defined the graph convolution on hypergraphs as $g * x = \Phi g(\Lambda)\Phi^T x$. To accelerate the computation, Feng et al. [25] followed the approach of Defferrard et al. [17], using a second-order Chebyshev inequality to approximate the solution of $g * x$, ultimately expressing the graph convolution calculation on the hypergraph as follows:

$$Y = D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} X\theta \tag{12}$$

where $X \in \mathbb{R}^{n \times C_1}$ represents the node feature matrix, n is the number of nodes, C_1 is the dimension of the node features, $W = diag(w_1, \ldots, w_n)$ denotes the hyperedge weights, $\theta \in \mathbb{R}^{C_1 \times C_2}$ represents the learning parameter, and $Y \in \mathbb{R}^{n \times C_2}$ is the output vector of the layer, utilized for classification. The design of the model's overall convolutional layer is illustrated in Figure 4.

Inspired by the HGNN model, S. Ji et al. [26] proposed the DHCF (Dual-Channel Collaborative Filtering) algorithm. This algorithm extracts structural information of users and items through a defined JHConv structure, then uses a shared weight matrix to associate the two extracted representation matrices, ultimately obtaining the representation matrices for both items and users. The JHConv structure is defined as follows:

$$X^{(l+1)} = \sigma(D_v^{-\frac{1}{2}} H D_e^{-1} H^T D_v^{-\frac{1}{2}} X^{(l)} \theta^{(l)} + X^{(l)}) \quad (13)$$

Where σ is the activation function, $X^{(l)}$ is the node feature matrix at layer l, $\theta^{(l)}$ is the learnable weight matrix, D_v is the node degree matrix, H is the incidence matrix of the hypergraph, and D_e is the hyperedge degree matrix.

The calculation process of the JHConv structure incorporates the representation results from the previous layer $X^{(l)}$, which accelerates the model's convergence speed. This dual-channel structure, while implementing collaborative filtering, preserves the distinct characteristics of h[illegible]er and item representations. However, because the p[illegible]el cons[illegible]ts relationships between different entities sol[illegible] through shar[illegible] weights, the connections between di[illegible]nt entities rel[illegible] entirely on the quality of the data. There[illegible] it may face challenges when handling noisy d[illegible].

Yi et al.[27] extended [illegible]NN by integrat[illegible] it [illegible]n RNNs to develop the HGC[illegible]NN model[illegible]his appro[illegible]irst employs a graph convoluti[illegible]l netw[illegible]k to extract features from time slices of a te[illegible]poral [illegible]raph, which[illegible]e then fed into an RNN for ti[illegible]e-series pre[illegible]ion. Co[illegible]ared to other GNN-based str[illegible]d time-series [illegible]del[illegible]HGC-RNN has fewer parame[illegible]s and[illegible]ws improved r[illegible]ustness in complex networks. However, it [illegible]limited to unweighted temporal hypergra[illegible]

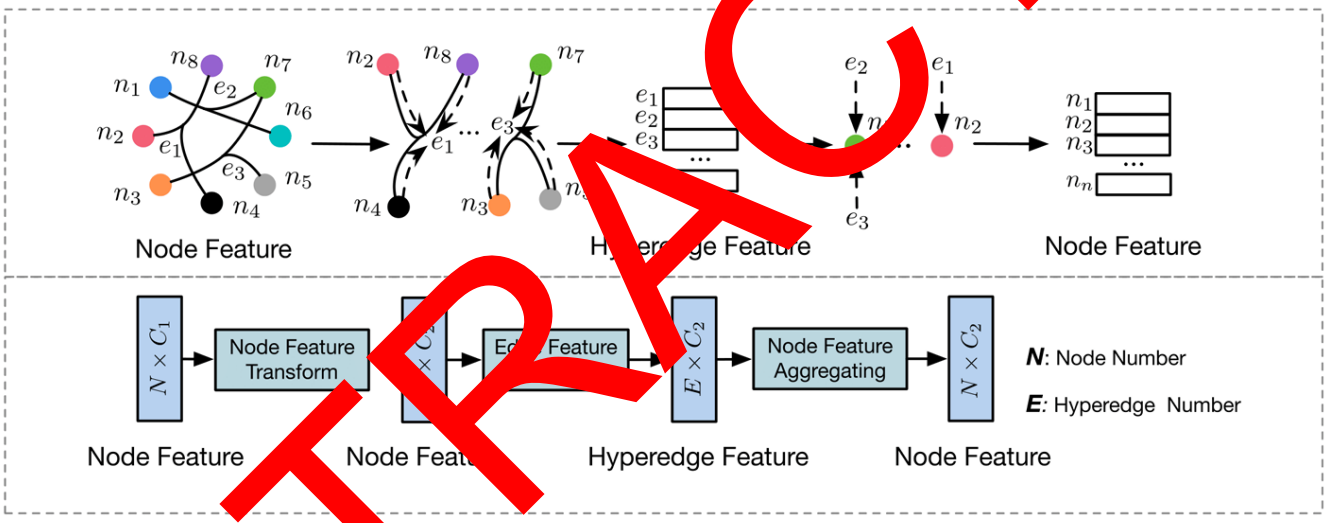

Figure 4. Illustration of the hyperedge convolutional layer [25]

Yadati et al. [28] pointed out that the HGNN model might introduce excessive noise during the information fusion process, which could have a negative impact in scenarios of hypergraph-based semi-supervised learning. To address this, Yadati et al. [28], inspired by the sampling-based expansion ideas from BTR [29] and MediSyn [30], proposed the HyperGCN, FastHyperGCN, and Mediator-HyperGCN models. These models filter the binary edges generated from the expanded hypergraph through a sampling process, thereby reducing potential data noise to some extent. However, this approach risks missing valuable information. Although sampling the network structure simplifies each training iteration, it might necessitate more iterations to achieve the desired results.

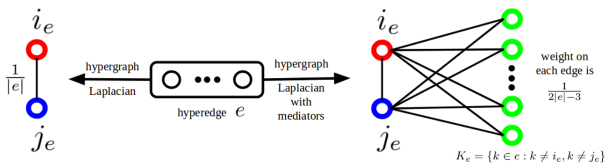

Figure 5. Supergraph Laplacian and Mediate-augmented Supergraph Laplacian [28]

3.2.2 Non-Expansion Neural Network Methods

Tu et al. [9] pointed out that in some hypergraphs, the subsets of a node set within a hyperedge cannot form independent hyperedges; such hyperedges are called non-decomposable hyperedges. In this case, certain expansion

methods are no longer applicable, leading to the proposal of the DHNE model, which is specifically designed for non-decomposable hyperedges. Tu et al. [9] defined first-order similarity in a hypergraph, which refers to the proximity similarity between nodes within the same hyperedge, and second-order similarity, which refers to the similarity between nodes with similar neighborhood structures. To preserve the second-order similarity of the hypergraph, they adopted an autoencoder structure to reconstruct the Laplacian matrix $A = HH^T - D_v$, which contains information about the neighborhood structure. The reconstructed node feature vectors are then used as inputs to calculate the first-order similarity of the hypergraph. The calculation formula is as follows:

$$\left.\begin{aligned} L_{ijk} &= \sigma(W_a^2 * X_i^a + w_b^2 * X_j^b + w_c^2 * X_k^c + b^2) \\ S_{ijk} &= S(X_i^a, X_j^b, X_k^c) = \sigma(w^3 * L_{\text{ijk}} + b^3) \end{aligned}\right\} \quad (14)$$

In the DHNE model, X_i^a, X_j^b, X_k^c represent the reconstructed feature vectors, and σ is the sigmoid function. The matrix A used in the DHNE model to preserve second-order similarity is actually Bolla's Laplacian matrix [22] mentioned earlier. Although DHNE is designed for heterogeneous hypergraphs and can retain first-order and second-order information of nodes, its model structure imposes strict requirements on the number of input tuples, making it applicable only to uni[illegible] hypergraphs and difficult to extend to arbitr[illegible] hypergraphs.

To address the limitation of the DHNE method, which is restricted to processing fixed type[illegible]zes of heterogeneous hyperedges, R. Zhang e[illegible]. [14] p[illegible]posed the Hyper-SAGNN model. They i[illegible]oduced th[illegible] classic self-attention mechanism [31,32] [illegible] the [illegible] aggregate hypergraph inform[illegible]n, co[illegible]ng pairwise attention coefficients betw[illegible] nodes as [illegible]ir dynamic features, while also inco[illegible]g the nodes' o[illegible]inal static features to describe th[illegible]. The [illegible]l calculation [illegible]mula is as follows:

$$o_i = W\left(\left(\vec{d}_i - \vec{s}_i\right)^{\circ 2}\right) + b \quad (15)$$

where [illegible] sents the [illegible]ynamic feature and $\vec{s}_i$ re[illegible]s the st[illegible] feature. [illegible]ce Hyper-SAGNN imposes [illegible] restri[illegible]ons on [illegible] ty[illegible] and number of nodes in the input tu[illegible]es, it has [illegible]etter generalization capabilities compa[illegible] DHNE. However, Hyper-SAGNN's on[illegible]uction of [illegible]merous intermediate features results in [illegible]er computational complexity for the model.

[illegible]imilarly, Bai et al. [33] introduced the attention mecha[illegible]m into hypergraphs, but unlike previous works, they combined the attention mechanism with graph convolution. Although the convolution computation defined by Bai et al. [33] is the same as that proposed by Feng et al. [25], they pointed out that the inherent attention mechanism within graph convolution is neither learnable nor trainable once the incidence matrix H is defined. Therefore, Bai et al. [33] re-calculated the incidence matrix H by introducing an attention mechanism and used it as the input for the convolutional neural network. Figure 6 shows a schematic diagram of the attention mechanism in the model. Under the assumption that nodes and hyperedges belong to the same homogeneous domain, the incidence relation H_{ij} between node x_i and its associated [illegible]peredge x_j is defined as follows:

$$H_{ij} = \frac{\exp\left(\sigma(sim(x_i P, x_j, P))\right)}{\sum_{k \epsilon \varkappa_i} \exp\left((sim(x_i P, x_k, P))\right)} \quad (16)$$

where $\sigma(\cdot)$ is a nonlin[illegible] activation func[illegible] si[illegible] is a function that comput[illegible] the simil[illegible]y betwee[illegible]des, and $\varkappa_i$ is the set of hyp[illegible]dges a[illegible]cent to node i. The incidence matrix [illegible]ained [illegible]gh these [illegible]culations, influenced by t[illegible] attention m[illegible]anism, [illegible] be learned during the ba[illegible]agation traini[illegible]pr[illegible]ss in the neural network. T[illegible]refor[illegible]ompared to t[illegible]tional hypergraph convolutional neural [illegible]tworks, this model has higher flexibili[illegible]. However, it [illegible] important to ensure that the si[illegible]arity between nodes a[illegible] hyperedges is comparable [illegible]en constructing the attention mechanism.

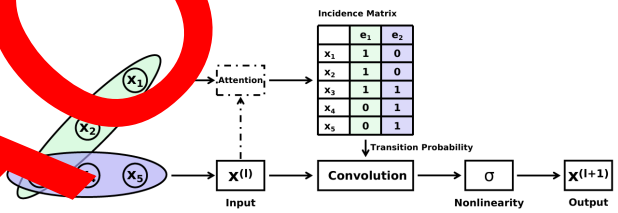

Figure 6. Attention Mechanism [33]

The summary of the above methods reiterates that hypergraph structures are ubiquitous in the real world, and the study of hypergraph learning methods holds universal practical significance.

4. Applications of Hypergraphs

Although hypergraph neural networks (HGNNs) have a relatively short history of development, they have already been widely applied in modeling various types of data interaction relationships and have achieved significant success. This section will detail their practical applications in different fields.

4.1. Recommendation Systems

As e-commerce continues to expand globally, the number of product categories has rapidly increased. Recommending the right products to customers has become a major challenge for e-commerce platforms, leading to the emergence of recommendation systems. As one of the most closely related application areas between

academia and industry, research on recommendation systems has been abundant.

In response to the complexity of recommendation scenarios, many studies have proposed using hypergraphs to model multivariate information [34–36]. This approach not only ensures the completeness of the information but also effectively models higher-order relationships, thereby improving information integration. Hypergraph-based recommendation algorithms have been proven to be effective in scenarios such as music recommendation [34,36], text recommendation [35], and movie recommendation [26].

DHCN [37], SHARE [38], and HGNN [39] are hypergraph neural network models based on session-based recommendations. DHCN is a dual-channel hypergraph convolutional network that combines hypergraphs and line graphs, where hypergraph convolution captures higher-order relationships at the item level, and line graph convolution learns relationships at the session level. S2-DHCN is a variant of DHCN that incorporates self-supervised learning. SHARE models each session as a hypergraph, with items as vertices and hyperedges linking all items within a contextual window. It employs a hypergraph attention mechanism to assess the significance of items to sessions and the influence of sessions on items. HGNNA, a recommendation model based on hypergraph neural networks and attention mechanisms, first utilizes a hypergraph neural network to learn item associations. It then applies a self-attention mechanism to aggregate session information and a graph attention mechanism to reveal the relevance between sessions.

Table 2. Datasets of Session-Based Recommendation

Dataset	Number of Training Sessions	Number of Test Sessions	Number of Products	Average Length of Sessions
YooChoose1/4	5[illegible]745	55898	[illegible]618	5.71
Diginetica	719470	60858	43097	5.12

Table [illegible] Results of Different Models for Session-Based Recommendation (MRR@K)

Model	Dataset	
	YooChoose1/4	Diginetica
STAMP	30.00	14.32
SR-GNN	31.89	17.59
HGNNA	30.81	17.59
SHARE	32.11	18.05
DHCN	36.72	30.58
S2-DHCN	40.13	30.94

Short-Term Attention/Memory Priority (STAMP) [40] and Session-based Recommendation with Graph Neural Networks (SR-GNN) [41] are two advanced baseline models that enhance session-based recommendations by introducing attention mechanisms and GNNs, respectively. Table 2 records the commonly used datasets for session-based recommendation, YooChoose, and Diginetica. As shown in Table 3, hypergraph neural network models SHARE, DHCN, and S2-DHCN outperform SR-GNN in terms of the mean reciprocal ranking (MRR@K) on YooChoose1/4, with improvements of 0.22%, 4.83%, and 8.24%, respectively. On Diginetica, the MRR@K values improved by 0.46%, 12.99%, and 13.35%, respectively. HGNNA also outperformed STAMP, further proving the significant effectiveness of hypergraph neural network models in this task.

4.2. Clustering

Network clustering involves partitioning network vertices into several clusters, with the goal of making vertices within the same cluster closely connected while having weaker connections between different clusters. This clustering structure is widespread in fields such as bioinformatics, computer science, physics, and sociology, and it has significant importance.

In fact, hypergraph partitioning can also achieve clustering objectives, so in many studies, hypergraph partitioning algorithms are classified as clustering algorithms. Research on custom edge weight functions is also an important method in hypergraph clustering. Additionally, projecting hypergraph structure information and node attribute information into vector space through hypergraph representation methods, and then clustering based on spatial distance, is also an effective approach for hypergraph clustering. This method preserves graph structure characteristics while also considering node attribute information.

4.3. Node Classification

In addition to hypergraph partitioning and clustering analysis, node classification is another important research direction in hypergraph learning. The node classification task is based on the assumption that similar nodes have similar labels. Most algorithms achieve classification by generalizing the data analysis of labeled nodes to unlabeled nodes, which is also applicable in hypergraphs.

Label propagation algorithms are a class of classic semi-supervised graph node classification methods. Zhou et al. [10] were the first to propose this method on hypergraphs, aiming to minimize the label differences between vertices sharing the same hyperedge. Unlike traditional label propagation, recent studies have proposed models that map hypergraph structure information into vector space before performing node classification, hoping to uncover higher-order structural information to improve

classification effectiveness. However, the main drawback of these methods is that they only use the initial hypergraph structure, neglecting the impact of vector space mapping on hypergraph structure. Therefore, some dynamic hypergraph models have been proposed that optimize graph structure during training to achieve better classification results.

4.4. Visual Tasks

Hypergraph learning algorithms are also widely applied in visual tasks. In computer vision, hypergraphs are used to describe relationships between visual features, such as visual classification [42], image retrieval [22], and video object segmentation [20]. Additionally, hypergraphs are used in 3D model classification for label propagation [43] and in social media for multimodal data processing [44]. The application of hypergraph learning algorithms in visual tasks is relatively flexible, with the algorithms needing to be designed based on the specific characteristics of the task objectives. The main reason for introducing hypergraph structures is that they can effectively express complex relationships between objects and uncover higher-order relationships.

4.5. Biological Networks

In biological networks, nodes often represent entities li proteins, metabolites, or genes, while edges denot functional relationships or interactions, such inding," "catalyzing," or "converting." Tradition graph which link only two nodes per edge, fall s rt in rep enting processes involving multiple partici ts. Hyper by contrast, provide a more suitabl fram ork capturing these complex interactions.

Tsuyuzaki et al. [45] osed a sensor address the problem of previous da ana is methods fo ng only on one-to-one interact ns betw two cell types, using it to extract repr tative hyper hs of interaction relationships a discover new cell raction patterns. Similarly, Y et al. [46] oposed HCIS based on the intra-class scat matrix, a yzing the interaction relationships of higher-o bi gical mo les. These methods not only overcom he limitat ns of traditional binary i ract analys but al provide new perspectives for understa ing comp ular networks.

rospects

In t paper, we systematically reviewed the concepts, definit s, methods, and applications of hypergraph learning in different fields. First, we introduced the definition of hypergraphs and their related symbols, clarifying the unique advantages of hypergraphs in handling higher-order relationships, making them an important tool for modeling complex data structures. We then explored the main current methods of hypergraph learning, particularly spectral analysis methods and neural network methods. By categorizing and comparing expansion-based and non-expansion-based methods, we revealed their applicability and limitations in different application scenarios. On the application level, we discussed in detail the widespread use of hypergraphs in recommendation systems, clustering, node classification, visual tasks, and biological networks, de strati the strong capability and flexibility of hyper phs in handl complex and high-dimensional data.

Although hypergraphs have demo ated powerfu modeling capabilities across v ous dom and hav significantly improved t performance of m y application tasks, there are ll many c lenges to ess. For instance, in areas su as large ale data processing, optimization of hyp graph e construct n, and the integration of he ogeneous n, the eff tiveness and scalability hypergraph dels need further enhanceme Mor er, improving nterpretability and explainability of th models is crucial for better understanding their cision-making mechanisms. Th efore, future research s ld focus on addressing these e issues, exploring more efficient algorithms, timizing hyper ph structure construction methods, and vancing theoret l analysis and application expansion to f unleash the tential of hypergraph neural networks acr a broad ange of fields.

nces

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J/OL]. Communications of the ACM, 2017, 60(6): 84-90.

[2] ELMAN J L. Distributed representations, simple recurrent networks, and grammatical structure[J]. Machine learning, 1991, 7: 195-225.

[3] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

[4] GRAVES A, GRAVES A. Long short-term memory[J]. Supervised sequence labelling with recurrent neural networks, 2012: 37-45.

[5] CHUNG J. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[6] BRUNA J, ZAREMBA W, SZLAM A, et al.. Spectral Networks and Locally Connected Networks on Graphs[M/OL]. arXiv, 2014[2023-06-03].

[7] AGARWAL S, BRANSON K, BELONGIE S. Higher order learning with graphs[C]//Proceedings of the 23rd international conference on Machine learning. 2006: 17-24.

[8] JIANG J, WEI Y, FENG Y, et al.. Dynamic Hypergraph Neural Networks[C/OL]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. Macao, China: International Joint Conferences on Artificial Intelligence Organization, 2019: 2635-2641[2023-05-24].

[9] TU K, CUI P, WANG X, et al.. Structural deep embedding for hyper-networks[C]//Proceedings of the AAAI conference on artificial intelligence: 32. 2018.

[10] ZHOU D, HUANG J, SCHÖLKOPF B. Learning with Hypergraphs: Clustering, Classification, and Embedding[M/OL]//SCHÖLKOPF B, PLATT J, HOFMANN T. Advances in Neural Information Processing Systems 19. The MIT Press, 2007: 1601-1608[2023-07-20].

[11] LIU Y, SHAO J, XIAO J, et al.. Hypergraph spectral hashing for image retrieval with heterogeneous social contexts[J]. Neurocomputing, 2013, 119: 49-58.

[12] WU F, HAN Y H, ZHUANG Y T. Multiple hypergraph clustering of web images by miningword2image correlations[J]. Journal of Computer Science and Technology, 2010, 25(4): 750-760.

[13] GUI H, LIU J, TAO F, et al.. Large-scale embedding learning in heterogeneous event data[C]//2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016: 907-912.

[14] ZHANG R, ZOU Y, MA J. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs[M/OL]. arXiv, 2019[2023-05-24].

[15] SUN L, JI S, YE J. Hypergraph spectral learning for multi-label classification[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008: 668-676.

[16] KIPF T N, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[M/OL]. arXiv, 2017[2023-06-05].

[17] DEFFERRARD M, BRESSON X, VANDERGHEYNST P. Convolutional neural networks on graphs with fast localized spectral filtering[J]. Advances in neural information processing systems, 2016, 29.

[18] CARLETTI T, BATTISTON F, CENCETTI G, et al.. Random walks on hypergraphs[J]. Physical review E, 2020, 101(2): 022308.

[19] BOLLA M. Spectra, euclidean representations and clusterings of hypergraphs[J]. Discrete Mathematics, 1993, 117(1-3): 19-39.

[20] HUANG Y, LIU Q, METAXAS D.]Video object segmentation by hypergraph [C]//[OL]. 2009 IEEE Conference on Computer Vision and Pattern Recognition. Miami, FL: IEEE, 2009: 1738-1745[2023-05-24].

[21] PURKAIT P, CHIN T, REID I, et al.. Clustering with hypergraphs: the case for large hyperedges[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(9): 1697-1711.

[22] HUANG Y, LIU Q, ZHANG S, et al.. Image retrieval via probabilistic hypergraph ranking[C]//2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010: 3376-3383.

[23] MA X, WANG W, LI S, et al.. Hypergraph p -Laplacian regularization for remotely sensed image recognition[J]. IEEE Transactions on Geoscience and Remote Sensing, 2018, 57(3): 1585-1595.

[24] LIANG J, CHEN C, YE F, et al.. Hyper2vec: Biased random walk for hyper-network embedding[C]//Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24. Springer, 2019: 273-277.

[25] FENG Y, YOU H, ZHANG Z, et al.. Hypergraph Neural Networks[M/OL]. arXiv, 2019[2023-05-24].

[26] JI S, FENG Y, JI R, et al.. Dual Channel Hypergraph Collaborative Filtering[C/OL]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Virtual Event CA USA: ACM, 2020: 2020-2029[2023-05-24].

[27] YI J, PARK J. Hypergraph convolutional recurrent neural network[C]//Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020: 3366-3376.

[28] YADATI N, NIMISHAKAVI M, YADAV P, et al.. Hypergcn: A new method for training graph convolutional networks on hypergraphs[J]. Advances in neural information processing systems, 2019, 32.

[29] LOUIS A. Hypergraph markov operators, eigenvalues and approximation algorithms[C]//Proceedings of the forty-seventh annual ACM symposium on Theory of computing, 2015: 713-722.

[30] CHAN T H H, LIANG Z. Generalizing the hypergraph laplacian via a diffusion process with mediators[J]. Theoretical Computer Science, 2020, 806: 416-428.

[31] VASWANI A. Attention is all you need[J]. arXiv preprint arXiv:1706.03762, 2017.

[32] VELIČKOVIĆ P, CUCURULL G, CASANOVA A, et al.. Graph attention networks[J]. stat, 2017, preprint arXiv:1710.10903, 2017.

[33] BAI S, ZHANG F, TORR P H. Hypergraph convolution and hypergraph attention[J]. Pattern Recognition, 2021, 110: 107637.

[34] LIU S, BU J, CHEN C, et al.. Using rich social media information for music recommendation via hypergraph model[J]. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2011, 7(1): 1-22.

[35] ZHU Y, GUAN Z, TAN S, et al.. Heterogeneous hypergraph embedding for document recommendation[J]. Neurocomputing, 2016, 216: 150-162.

[36] BU J, TAN S, CHEN C, et al.. Music recommendation by unified hypergraph: combining social media information and music content[C]//Proceedings of the 18th ACM international conference on Multimedia. 2010: 391-400.

[37] HAN J, CHENG B, WANG X. Two-Phase Hypergraph Based Reasoning with Dynamic Relations for Multi-Hop KBQA.[C]//IJCAI. 2020: 3615-3621.

[38] WANG J, DING K, ZHU Z, et al.. Session-based recommendation with hypergraph attention networks[C]//Proceedings of the 2021 SIAM international conference on data mining (SDM). SIAM, 2021: 82-90.

[39] DING M, LIN X, ZENG B, et al.. Hypergraph neural networks with attention mechanism for session-based recommendation[C]//Journal of Physics: Conference Series: 2082. IOP Publishing, 2021: 012007.

[40] LIU Q, ZENG Y, MOKHOSI R, et al.. STAMP: short-term attention/memory priority model for session-based recommendation[C]//Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018: 1831-1839.

[41] WU S, TANG Y, ZHU Y, et al.. Session-based recommendation with graph neural networks[C]//Proceedings of the AAAI conference on artificial intelligence: 33. 2019: 346-353.

[42] WANG M, LIU X, WU X. Visual Classification by $\ell _1$ -Hypergraph Modeling[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(9): 2564-2574.

[43] ZHANG Z, LIN H, GAO Y, et al.. Dynamic hypergraph structure learning.[C]//IJCAI. 2018: 3162-3169.

[44] JI R, CHEN F, CAO L, et al.. Cross-modality microblog sentiment prediction via bi-layer multimodal hypergraph learning[J]. IEEE Transactions on Multimedia, 2018, 21(4): 1062-1075.

[45] TSUYUZAKI K, ISHII M, NIKAIDO I. Uncovering hypergraphs of cell-cell interaction from single cell RNA-sequencing data[J]. BioRxiv, 2019: 566182.
[46] YU L, SHEN X, JIANG X, et al.. Hypergraph clustering based on intra-class scatter matrix for mining higher-order microbial module[C]//2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2019: 240-243.