Hypergraph Learning: Methods and Practices

Yue Gao, *Senior Member, IEEE*, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, *Member, IEEE*, and Changqing Zou, *Member, IEEE*

Abstract—Hypergraph learning is a technique for conducting learning on a hypergraph structure. In recent years, hypergraph learning has attracted increasing attention due to its flexibility and capability in modeling complex data correlation. In this paper, we first systematically review existing literature regarding hypergraph generation, including distance-based, representation-based, attribute-based, and network-based approaches. Then, we introduce the existing learning methods on a hypergraph, including transductive hypergraph learning, inductive hypergraph learning, hypergraph structure updating, and multi-modal hypergraph learning. After that, we present a tensor-based dynamic hypergraph representation and learning framework that can effectively describe high-order correlation in a hypergraph. To study the effectiveness and efficiency of hypergraph generation and learning methods, we conduct comprehensive evaluations on several typical applications, including object and action recognition, Microblog sentiment prediction, and clustering. In addition, we contribute a hypergraph learning development toolkit called THU-HyperG.

Index Terms—Hypergraph learning, hypergraph generation, hypergraph learning tool, tensor-based dynamic hypergraph learning, classification and clustering

1 Introduction

A hypergraph composing of a set of vertices and hyperedges is a generalization of a graph. As illustrated in Fig. 1, different from a simple graph where two vertices are joined together by one edge, each hyperedge can connect any number of vertices in a hypergraph. The edge degree in a hypergraph could be much higher than that of a simple graph, which is two. Compared to the graph structure being able to model pairwise connections with its 2-degree edges, a hypergraph has significant advantages in modeling the correlation of the practical data which could be far more complicated than pairwise relations.

Here we take two examples to introduce the application scenario of the hypergraph. To formulate the correlation among researchers, we construct a hypergraph where each vertex indicates a researcher and each hyperedge represents an article co-authored by several researchers. Given an article with k authors, a k-degree hyperedge can be generated to connect these k vertices corresponding to the k-order relationship. In this way, the researchers with closer cooperation can be connected by more hyperedges, and vice versa. Another example is the brain network modeling. Given a human brain network where a brain region predominantly interacts with one or more other brain regions neurologically [1], a hypergraph can be generated, where the vertices denote brain regions, and the hyperedges characterize the interactions among multiple regions.

Hypergraph learning is related to graph learning since a hypergraph is generalized by a graph. Similar to graph learning [2], learning on the hypergraph can be seen as the process of passing information along the hypergraph structure in analyzing the structured data and solving problems such as node classification [3], link prediction [4], community detection [5] and so on. From this perspective, graph learning is a special case of hypergraph learning, it considers only pair-wise connection between data. Unlike graph learning, hypergraph learning models explore the high-order correlation among data, which extend the graph learning models to a high dimensional and more complete nonlinear space, leading to higher capabilities of correlation modeling and thus better performance in practice.

Due to its advantages, hypergraph learning has many real world applications. In the field of computer vision, hypergraph learning has been extensively used in tasks including image retrieval [6], 3D object classification [7], video segmentation [8], person re-identification [9], hyperspectral image analysis [10], landmark retrieval [11] and visual tracking [12]. In these application tasks, a wide range of subjects can be embedded into a hypergraph structure. For example, in the image retrieval work [6], each vertex in the hypergraph is used to represent an image and feature-based correlations are used to generate hyperedges. In the 3D object classification work [7], 3D objects are represented by vertices and the view relationship information is used for hyperedge construction. In the person re-identification work [9], each vertex denotes one person, and the feature correlation is used to construct the hypergraph structure. In the field of bioinformatics and medical image analysis, hypergraph learning has been used to select genes [13], [14], predict diseases [15], [16],

- *Yue Gao and Zizhao Zhang are with the BNRist, KLISS, School of Software, and THUIBCS, Tsinghua University, Beijing 100084, China. E-mail: kevin.gaoy@gmail.com, zhangziz18@mails.tsinghua.edu.cn.*
- *Haojie Lin and Xibin Zhao are with the BNRist, KLISS, School of Software, Tsinghua University, Beijing 100084, China. E-mail: linhj18@mails.tsinghua.edu.cn, zxb@tsinghua.edu.cn.*
- *Shaoyi Du is with the Institute of Artificial Intelligence and Robotics, College of Artificial Intelligence, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: dushaoyi@gmail.com.*
- *Changqing Zou is with the Sun Yat-sen University, Guangzhou 510275, China. E-mail: aaronzou1125@gmail.com.*

Manuscript received 28 Apr. 2020; revised 6 Sept. 2020; accepted 2 Nov. 2020. Date of publication 19 Nov. 2020; date of current version 1 Apr. 2022. (Corresponding author: Changqing Zou.) Recommended for acceptance by O. Veksler. Digital Object Identifier no. 10.1109/TPAMI.2020.3039374

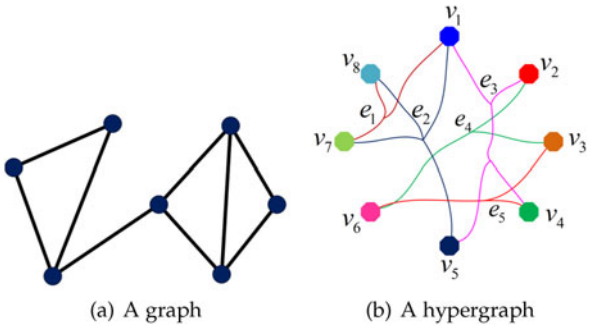


Fig. 1. Illustration of a graph and hypergraph.

identify sub-types [17], and analyze functional networks [18]. Moreover, hypergraph learning has also been used for recommendation systems [19], train self-driving systems [20], or predict social network links [21].

There have been many efforts to explore new hypergraph learning methods. So far, there is no systematic investigation of hypergraph learning methods and practices. This paper systematically reviews existing hypergraph generation methods, including distance-based, representation-based, attribute-based, and network-based approaches. We also introduce the learning methods on a hypergraph in detail, including transductive hypergraph learning, inductive hypergraph learning, hypergraph structure updating, and multi-modal hypergraph learning. Previous hypergraph learning methods have two main limitations. One is that the constructed hypergraphs may not be optimized or under-optimized so that the established model cannot fit the data well. The other is that the computational cost can be excessively high, especially when the hypergraph structure is updated simultaneously, making it difficult to apply to large-scale datasets. Considering that the modeling capability of the hypergraph structure has high influence on learning performance, and learning efficiency is crucial for large-scale data, we propose a tensor-based dynamic hypergraph learning method as a more effective and lower computational solution. We have conducted a comparative study of the effectiveness and efficiency of existing hypergraph learning methods and the proposed tensor-based dynamic hypergraph learning method on multiple applications.

The main contributions of this work can be summarized as follows:

1) We provide a systematic literature review of hypergraph learning methods, from hyperedge generation to learning on a hypergraph. We conduct a comparative study of existing methods on multiple applications, including 3D object classification, action recognition, Microblog sentiment prediction, and general clustering tasks, and study the advantages and disadvantages of existing methods.
2) We propose a tensor-based hypergraph learning method, which presents a flexible tensor representation for a dynamic hypergraph, introduces a bi-convex optimization model, and has the potential of large-scale data application due to its high effectiveness and low computational cost.
3) We have developed and released a toolbox called THU-HyperG, which provides a collection of hypergraph generation, learning algorithms and best practices.

The remainder of this paper is organized as follows. Section 2 presents preliminary knowledge of a hypergraph. Section 3 introduces hypergraph generation methods. Section 4 provides an overview of learning methods on the hypergraph. The proposed tensor-based hypergraph learning method is given in Section 5. Applications and evaluations are provided in Sections 6 and 7. The tool of THU-HyperG is presented in Section 8. We finally conclude this paper and discuss some open issues in Section 9.

2 Preliminary of Hypergraph

We first briefly introduce the preliminary knowledge of hypergraphs. A hypergraph denoted as $\mathcal{G}$, consists of a set of vertices $\mathcal{V}$ and a set of hyperedges $\mathcal{E}$. In a weighted hypergraph, each hyperedge $e \in \mathcal{E}$ is assigned a weight $w(e)$, representing the importance of the connection relationship in the whole hypergraph. Let $\mathbf{W}$ denote the diagonal matrix of the hyperedge weights, i.e., $\mathrm{diag}(\mathbf{W}) = [w(e_1), w(e_2), \ldots, w(e_{|\mathcal{E}|})]$. Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, the structure of the hypergraph is usually represented by an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, with each entry $\mathbf{H}(v,e)$ indicating whether the vertex v is in the hyperedge e

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

In some cases, the incidence matrix is not a simple (0,1)-matrix, but a continuous matrix whose elements are in the range of 0 to 1. The entry $\mathbf{H}(v,e)$ can be indicated as the possibility of vertex v assigning to hyperedge e, or the importance of vertex v for hyperedge e.

We can define the degree of hyperedge e and the degree of vertex v by

$$\delta(e) = \sum_{v\in\mathcal{V}} \mathbf{H}(v,e), \tag{2}$$

and

$$d(v) = \sum_{e\in\mathcal{E}} w(e) * \mathbf{H}(v,e), \tag{3}$$

respectively. Let $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}|*|\mathcal{E}|}$ and $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}|*|\mathcal{V}|}$ be the diagonal matrix of the hyperedge weights and vertex degrees, respectively.

Laplacian matrix plays an important role in graph theory. For example, spectral analysis (spectral clustering, spectral partitioning) of the graph is based on finding eigenvalues and eigenvectors of the graph's Laplacian matrix. For a simple graph, the Laplacian matrix is defined as $\Delta = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix of vertex degrees and $\mathbf{A}$ is the adjacency matrix. For a hypergraph, the Laplacian matrix is more complicated, which is defined as

$$\Delta = \mathbf{D}_v - \mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T. \tag{4}$$

This Laplacian matrix can be normalized as

$$\Delta = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}. \tag{5}$$

Table 1 summarizes the notations and definitions throughout this paper for clarity.

TABLE 1
Notations and Definitions

Notation	Definition
$\mathbf{X}$	The feature vectors of subject set. $\mathbf{X}(v)$ indicates the feature vector for vertex v.
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$	$\mathcal{G}$ represents the hypergraph, and $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{W}$ represent the set of vertices, the set of hyperedges, and diagonal matrix of hyperedge weights, respectively.
$d(v)$	The degree of vertex v.
$w(e)$	The weight of hyperedge e.
$\delta(e)$	The degree of hyperedge e.
$\mathbf{H}$	The $\lvert\mathcal{V}\rvert \times \lvert\mathcal{E}\rvert$ incidence matrix of hypergraph structure. $\mathbf{H}(v, e)$ indicates the connection strength between the vertice v and the hyperedge e.
$\mathbf{D}_v$	The diagonal matrix of vertex degrees.
$\mathbf{D}_e$	The diagonal matrix of hyperedge degrees.
Δ	The Laplacian matrix of hypergraph.

TABLE 2
Hyperedge Categories

Hyperedge Category	Hyperedge Subtypes	Methods
Implict Hyperedge	Distance Based	Nearest Neighbor (k-NN [6],ϵ-ball)
		Clustering (k-means [7])
	Representation Based	$l1$ Reconstruction ($l1$-hypergraph [23])
		Elastic Net (Elastic net hypergraph [24])
		$l2$-hypergraph [25]
Explict Hyperedge	Attribute Based	Attribute Hypergraph [26], [27]
	Network Based	Social Media Network [19]
		Brain Network [28]
		Reaction Network [30]
		Cellular Network [31]

3 HyperGraph Generation

To formulate the data correlation with a hypergraph, the first step is to construct a hypergraph from the data. The quality of the generated hypergraph structure directly affects the effectiveness of the data correlation modeling. The problem of how to construct an effective hypergraph is not trivial and thereby has been extensively investigated [7], [8], [22]. Hypergraph generation methods can generally be divided into four categories, i.e., those distance-based methods [6], [7], representation-based methods [23], [24], [25], attributed-based methods [26], [27] and network-based methods [19], [28], [29]. Next, we revisit these hypergraph generation methods, which is summarized in Table 2.

3.1 Distance-Based Hypergraph Generation

Distance-based hypergraph generation approaches exploit the relations among vertices using the distance in the feature space. The main objective is to find neighboring vertices in the feature space and construct a hyperedge to connect them. Generally, there are two ways to construct this hyperedge, i.e., nearest-neighbor searching and clustering, as shown in Figs. 2a and 2b. In nearest-neighbor based methods, hyperedge construction needs to find the nearest neighbors for each vertex. That is, given a vertex (i.e., centroid), a hyperedge can connect itself and its nearest neighbors in the feature space. The number of the selected neighbors is a predefined parameter k [6] or decided by the number of vertices within the distance range of ϵ. This type of hyperedge can connect a group of similar vertices connecting to the same centroid. Different from nearest-neighbor based methods, the clustering-based methods [7] aim to directly group all vertices into clusters by a clustering algorithm such as k-means, and connect all vertices in the same cluster by a hyperedge (note that different scales of clustering results can be used to generate multiple hyperedges).

For these hyperedges, how to measure effectiveness is essential. The weight or the confidence of each hyperedge can be measured by

$$w(e) = \sum_{u,v \in e} \exp\left(-\frac{d(\mathbf{X}(u), \mathbf{X}(v))^2}{\sigma^2}\right), \tag{6}$$

where u and v denote a pair of vertices in the hyperedge e and $d(\mathbf{X}(u), \mathbf{X}(v))$ is the distance between u and v. The parameter σ can be empirically set to the median value of the distances of all vertex pairs.

Apart from the similarity in feature space, location (spatial) information can also be used in distance-based hyperedge generation, i.e., each vertex can find its spatial neighbors, as shown in Fig. 2c. In this example, the selected pink vertex and its four orange neighbors can be connected by a hyperedge.

Distance-based hyperedges can represent vertex connection in the feature space. The main limitation of this line of method is the distance among vertices may be inaccurate in some situations due to noise and outliers. Another issue is how many neighbors should be connected as the scale of nearest neighbors may affect the performance of hypergraph learning, but adaptively learning this number is not trivial in practice.

3.2 Representation-Based Hypergraph Generation

Representation-based hypergraph generation methods formulate the relations among vertices through feature reconstruction. For instance, $l1$-hypergraph [23] is a work employing sparse representation to model the relationship among vertices in hypergraph generation. More specifically, in $l1$-hypergraph, each vertex acts as the centroid and can be represented by its nearest neighbors as

$$\begin{aligned} &\underset{\mathbf{z}}{\operatorname{argmin}} \|\mathbf{B}\mathbf{z} - \mathbf{X}(v_c)\|_2^2 + \gamma\|\mathbf{z}\|_1 \\ &\text{s.t. } \forall i, \mathbf{z}_i \geq 0, \end{aligned} \tag{7}$$

where $\mathbf{X}(v_c)$ and $\mathbf{B}$ denote the feature vectors of the centroid sample v_c and its k nearest neighbors, respectively. $\mathbf{z}_i$ is the learned reconstruction coefficients associated with the ith nearest neighbor. Based on this representation, the vertices in the neighbors with non-zero reconstruction coefficients to the centroid are grouped to generate a hyperedge, and the connection strength between the hyperedge and each vertex can be set as the coefficient $\mathbf{z}_i$.

$l1$-hypergraph is weak in revealing the grouping information of samples due to the $l1$-norm regularization. To tackle this problem, Liu *et al.* introduces an elastic net [24] to

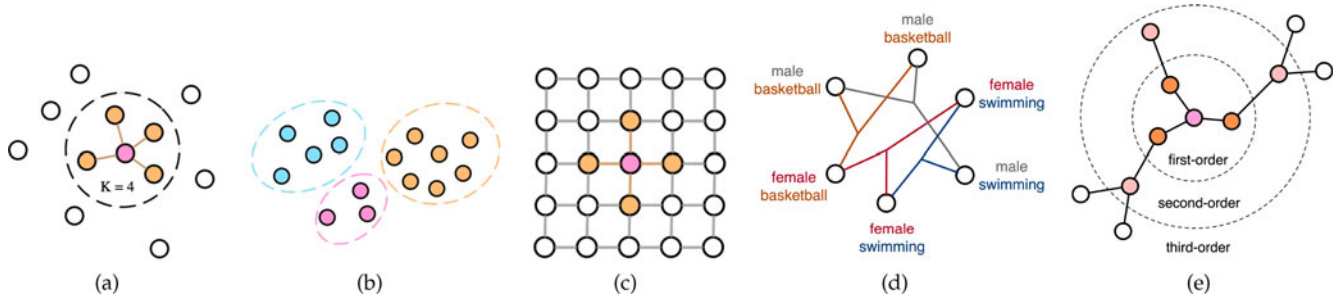


Fig. 2. Distance-based hyperedge generation methods include (a) nearest-neighbor based methods and (b) clustering-based methods. (c) Distance-based hypergraph generation methods construct hyperedges using spatial information [32]. (d) Attribute-based hypergraph generation methods construct a hypergraph by linking the vertices sharing the same attributes with a hyperedge. (e) Network-based hypergraph generation methods could construct a hypergraph using not only first-order information, but also higher-order (e.g., second-order and third-order) information.

enhance the grouping effect by combining a $l2$-norm penalty to the $l1$-norm constraint, which can be formulated as

$$\begin{aligned} &\underset{\mathbf{z}}{\operatorname{argmin}} \|\mathbf{B}\mathbf{z} - \mathbf{X}(v_c)\|_2^2 + \gamma\|\mathbf{z}\|_1 + \beta\|\mathbf{z}\|_2^2. \\ &\text{s.t. } \forall i, \mathbf{z}_i \geq 0. \end{aligned} \tag{8}$$

By leveraging both $l2$-norm and $l1$-norm penalties, the elastic net can group more relevant neighbors to construct a hyperedge whose weight can be derived from the reconstruction coefficients as it does in [23].

Note that these two representation-based methods have two limitations: (1) these methods employ a $l2$-norm based metric to measure the reconstruction errors, which are still sensitive to sparse reconstruction errors. (2) these methods generate hyperedges by solving a linear problem, which cannot deal with the non-linear data. To solve these problems, $l2$-hypergraph [25] is introduced by separating the sparse noise component from the original data and integrating the locality, preserving constraint to the linear regression framework

$$\begin{aligned} &\underset{\mathbf{C},\mathbf{E}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{X}\mathbf{C} - \mathbf{E}\|_F^2 + \frac{\gamma_1}{2}\|\mathbf{C}\|_F^2 + \frac{\gamma_1}{2}\|\mathbf{Q} \odot \mathbf{C}\|_F^2 + \beta\|\mathbf{E}\|_1 \\ &\text{s.t. } \mathbf{C}^T\mathbf{1} = \mathbf{1}, \operatorname{diag}(\mathbf{C}) = \mathbf{0}, \end{aligned} \tag{9}$$

where $\mathbf{C}$ is the coefficients matrix, $\mathbf{E}$ is the data errors matrix, $\mathbf{Q}$ is the locality adapter matrix used to preserve the local manifold structures, $\mathbf{Q}_{ij} = \frac{\exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2)}{\sum_{t \neq i} \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2)}$, and $\odot$ denotes element-wise multiplication. Then, the coefficients matrix $\mathbf{C}$ can be used to generate hyperedges.

Representation-based hyperedges can evaluate the reconstruction capability of each vertex in the feature space. The correlation between the feature vectors can be calculated and used for building the connections among the vertices. Like those distance-based methods, this line of work may also suffer from possible data noise and outliers. Another limitation of this line of work comes from the data sampling procedure: only part of the relevant samples are selected for reconstruction, and the generated hyperedge may be incapable of fully representing the data correlation.

3.3 Attribute-Based Hypergraph Generation

Attribute-based hypergraph generation methods are those methods that employ the attribute information to construct a hyperedge [26], [27]. Fig. 2d illustrates the hyperedges of an attribute-based hypergraph. In this example, the vertices sharing the same attribute are linked by a hyperedge.

Each hyperedge in an attribute-based hypergraph is regarded as a clique, and then the mean of the heat kernel weights of pairwise edges in this clique can be used as the hyperedge weight

$$w(e) = \frac{1}{\delta(e)(\delta(e)-1)} \sum_{u,v \in e} \exp\left(-\frac{\|\mathbf{X}(u) - \mathbf{X}(v)\|_2^2}{\sigma^2}\right), \tag{10}$$

where $\delta(e)$ denotes the degree of hyperedge e. As the attributes could be hierarchical, the generated hyperedges can also have different levels, leading to different hyperedges representing multiple-scale attribute connections. Although the attribute information has a significant advantage in representing data, such information may not be available in some cases. A possible solution is to define a group of attributes [33] that can be learned from existing data.

3.4 Network-Based Hypergraph Generation

Network data are available in many applications, such as social networks [21], reaction networks [30], cellular networks [31] and brain networks [28]. For these data, the network information can be used to generate subject correlations. For example, the vertices in [19] represent users and images. The hyperedges, including homogeneous and heterogeneous hyperedges, are utilized to capture multi-type relations including visual-textual content relations among images, and social links between users and images. The homogeneous hyperedges representing the visual and textual relations among images are constructed by using nearest-neighbor-based and attribute-based hyperedge generation methods. The heterogeneous hyperedges are constructed using the social link relations connecting images and users.

In location-based social networks, Yang *et al.* [21] use the user friendship and mobility information simultaneously to construct the hypergraph. Specifically, it generates the classical friendship hyperedges within the social domain and the check-in hyperedges across the social, semantic, temporal, and spatial domains. In a protein-protein interaction network, tandem affinity purification (TAP) data naturally span a hypergraph [31], where the complexes can represent the subsets (hyperedges) of the ground set of proteins.

Generally, apart from the first order correlation, higher order (e.g., second- and third- order) correlation in the

network can also be used for hyperedge generation. Fig. 2e illustrates a hyperedge example of network data. In this example, given a center vertex, its first order and high order neighbors (i.e., the vertices whose shortest path to the centroid is larger than 1), can be connected with the center vertex by a hyperedge. Intuitively, if attention is focused on the local connection of a vertex in the network, we only need to consider its low-order neighbors. For example, in the recommendation network [34], where users are connected according to similar preferences on items, only first-order and second-order correlations are used to generate the hypergraph and conduct collaborative filtering for recommendation. Conversely, if the information of a vertex travels a long distance in the network, we need to generate hyperedges using higher-order correlations.

3.5 Comparison of Hypergraph Generation Methods

The four types of hyperedge generation above can be further grouped into implicit and explicit categories, depending on whether the hyperedge construction is implicit or explicit. The implicit hyperedges are defined as the hyperedges that are not able to be directly obtained from raw data and need to be reconstructed by establishing representations and measurements. Thus, distance-based and representation-based methods can be classified into the implicit category. These methods can be applied to the tasks where we can create the representation of each subject and metrics to describe the similarity between samples. In contrast, explicit hyperedges can be established straightforwardly because the input data may inherently contain some structural information. Therefore, the attribute-based and network-based methods belong to the explicit category, where the hyperedges can be directly obtained using either attribute or network connections.

Each method has its strengths and limitations. Distance-based methods are straightforward and effective in a large variety of applications. However, they are sensitive to the similarity measurement and hyper-parameter settings (e.g., the number of neighbors in the nearest-neighbor based schema or the number of clusters in the clustering procedure). Compared to distance-based methods, representation-based methods can avoid the effect of the noise vertices via sparse representation. For example, the vertices with zero reconstruction coefficients are considered as noise vertices in the $l1$-hypergraph and are excluded from hyperedges. Although the representation-based methods have shown robustness to the noise data, computing the reconstruction coefficients could introduce an extra computational cost. Attribute-based approaches are appropriate for the samples with specific attributes. These attributes could be given by the input, or defined manually, or even automatically generated. This line of method is straightforward but has the limitation caused by considering only single-attribute features (i.e., only the vertices sharing the same attribute are connected with hyperedges). The relationships among different attributes are underutilized in this way. Network-based approaches are supposed to be suitable for the data that can be represented by graphs naturally. It is necessary to construct a hypergraph with specific domain knowledge, e.g., [21], [31].

In general, given a specific task, there may be a most suitable hyperedge generation method. Lastly, due to the scalability of a hypergraph's adjacency hypermatrices, it is common to create a more complicated hypergraph by merging several hypergraphs created by different approaches together.

4 Learning on Hypergraph

After a hypergraph is constructed, learning tasks, such as label prediction of unlabeled vertices, dynamic structure update, and multi-modal learning on this hypergraph, can be performed.

4.1 Label Prediction

This section introduces hypergraph learning methods for classification, which have attracted much interest due to its effectiveness and ability to consider the high-order correlation among subjects. We start by introducing a general learning model on the hypergraph, and then representing the learning process on the hypergraph in transductive and inductive manners in detail, respectively.

We would like the hypergraph learning model as general as possible while allowing useful analysis. Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, the training vertices are those vertices with labels. The task of label prediction is to predict the labels of the remaining unlabeled vertices. Let $\mathbf{F} : V \rightarrow \mathbb{R}^{|L|}$ denote the classification function that assigns a label vector $\mathbf{F}(v) \in \mathbb{R}^{|L|}$ to a vertex $v \in V$. The ith element in this label vector denotes the confidence score of classifying the corresponding subject as the ith class. Normally, the predicted label of an unlabeled vertex is set to the one with the highest confidence score.

In a general hypergraph learning model, there are two parts in the objective function. The first part is an empirical loss of learned labels $R_{\text{emp}}(\mathbf{F})$, like a hinge loss or a least square loss. The second part is a regularizer on the hypergraph $\Omega(\mathbf{F})$, indicating the smoothness of the label distribution on the constructed hypergraph structure. By a linear combination, a general model can be formulated as

$$\underset{\mathbf{F}}{\operatorname{argmin}} \Psi(\mathbf{F}) := \big\{\Omega(\mathbf{F}) + \lambda R_{\text{emp}}(\mathbf{F})\big\}, \tag{11}$$

where λ is a trade-off parameter. Next, we discuss two typical hypergraph learning models in accord with this general model: the transductive learning method and the inductive learning method.

4.1.1 Transductive Learning

The seminal work of [22] is the first transductive inference based hypergraph learning work. Label prediction can be regarded as a partition of $\mathcal{V}$ into several parts, i.e., a normalized hypergraph cut. The vertices with the same labels are divided into the same part. We first consider the two-class label prediction problem and then extend to the multi-class problem.

Naturally, without considering the empirical loss of learned labels, the binary classification model aims to learn a partition in which the connections among the vertices within the same part are dense while the connections between two parts are sparse. In other words, the objective function of the classification is to minimize the partition cost on the

hypergraph, which can be written as

$$\underset{\emptyset \neq S \subset \mathcal{V}}{\operatorname{argmin}} c(S) := \sum_{e \in \mathcal{E}} w(e) \frac{|e \cap S||e \cap S^c|}{\delta(e)} \left(\frac{1}{|S|} + \frac{1}{|S^c|} \right), \tag{12}$$

where S is a vertex subset of $\mathcal{V}$ and S^c is the compliment of S. And the classification function $\mathbf{F}$ can be written as

$$\mathbf{F}(v) = \begin{cases} 1 & v \in S \\ 0 & v \in S^c \end{cases}. \tag{13}$$

Thus, the regularizer on the hypergraph is defined as the partition cost

$$\begin{aligned} \Omega(\mathbf{F}) = c(S) = & \sum_{e \in E} \sum_{\{u,v\} \subseteq e} \frac{w(e)}{\delta(e)} \mathbf{F}(u)(1 - \mathbf{F}(v)) \\ & \left(\frac{1}{\sum_{v \in \mathcal{V}} \mathbf{F}(v)} + \frac{1}{\sum_{v \in \mathcal{V}} (1 - \mathbf{F}(v))} \right). \end{aligned} \tag{14}$$

The optimization of Eq. (14) is NP-complete. It could be relaxed into a real-valued optimization problem according to [22]. Finally, the regularizer $\Omega(\mathbf{F})$ in the general model can be defined as

$$\Omega(\mathbf{F}) = \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \subseteq e} \frac{w(e)}{\delta(e)} \left(\frac{\mathbf{F}(u)}{\sqrt{d(u)}} - \frac{\mathbf{F}(v)}{\sqrt{d(v)}} \right)^2 = \mathbf{F}^T \Delta \mathbf{F}, \tag{15}$$

where $\Delta = \mathbf{I} - \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2}$ is the Laplacian matrix. In the multi-class classification problem, we can achieve the same model as above by extending the 2-way partitioning of the hypergraph to a k-way partitioning.

The empirical loss of the general model could be defined as the least square loss between the predicted label vectors $\mathbf{F}$ and the initial label vectors $\mathbf{Y}$. For a labeled vertex v, the jth element of $\mathbf{Y}(v)$ is 1 if the subject belongs to the jth class. Otherwise, it is 0. For an unlabeled vertex u, all elements of $\mathbf{Y}(u)$ are set to 0.5, indicating that there is no prior information about the category of this subject. The whole transductive learning model can be defined as

$$\underset{\mathbf{F}}{\operatorname{argmin}} \Psi(\mathbf{F}) := \left\{ \mathbf{F}^T \Delta \mathbf{F} + \lambda \|\mathbf{F} - \mathbf{Y}\|^2 \right\}. \tag{16}$$

The physical meaning of Eq. (16) can be explained by analogy with passing messages on a hypergraph. The state of a vertex is more affected by its neighborhoods with closer and stronger connections than those remote vertices. The optimization iteration process is like conducting diffusion on the hypergraph. When reaching convergence, the optimal solution of the function corresponds to the balanced state of message passing on the hypergraph. Thus, Eq. (16) has a closed-form solution, that is

$$\mathbf{F} = (\mathbf{I} - \mu \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2})^{-1} \mathbf{Y}. \tag{17}$$

4.1.2 Inductive Learning

In the transductive hypergraph learning methods, both the training and testing stages are online, i.e., all the training data and the testing data are taken into consideration during the hypergraph construction and learning process. This online learning procedure has a high computational cost. Unlike the transductive learning schema, inductive learning contains an off-line training and an online classification process. The classification function $\mathbf{F}$ is defined as a projection from the original features $\mathbf{X}$ to label vectors, i.e., $\mathbf{F} = \mathbf{X}^T \mathbf{M}$, where $\mathbf{M}$ is the projection matrix. At the training stage, only the training data are employed to construct the hypergraph and learn an optimal $\mathbf{M}$. At the test stage, given an unlabeled subject $\mathbf{X}(v)$, its label vector can be obtained by projecting the subject into the subspace spanned by $\mathbf{M}$, i.e., $\mathbf{F}(v) = \mathbf{X}(v)^T \mathbf{M}$.

More formally, in inductive learning, the general model of learning $\mathbf{F}$ in Eq. (11) is converted to the problem of learning the projection matrix $\mathbf{M}$. A constraint to $\mathbf{M}$ could be further introduced to avoid over-fitting. Then, the optimization can be expressed in terms of $\mathbf{M}$ as

$$\underset{\mathbf{M}}{\operatorname{argmin}} \Psi(\mathbf{M}) := \left\{ \Omega(\mathbf{M}) + \lambda \mathcal{R}_{emp}(\mathbf{M}) + \mu \Phi(\mathbf{M}) \right\}. \tag{18}$$

Similar to transductive learning, the empirical loss term on $\mathbf{M}$ is defined as $\mathcal{R}_{emp}(\mathbf{M}) = \left\| \mathbf{X}^{\mathrm{T}} \mathbf{M} - \mathbf{Y} \right\|^2$, and the hypergraph Laplacian regularizer can be written in quadratic form of $\mathbf{M}$ as

$$\begin{aligned} \Omega(\mathbf{M}) = & \frac{1}{2} \sum_{k=1}^{c} \sum_{e \in \mathcal{E}} \sum_{u,v \in \mathcal{V}} \frac{\mathbf{W}(e) \mathbf{H}(u,e) \mathbf{H}(v,e)}{\delta(e)} \\ & \left(\frac{(\mathbf{X}^{\mathrm{T}} \mathbf{M})(u,k)}{\sqrt{d(u)}} - \frac{(\mathbf{X}^{\mathrm{T}} \mathbf{M})(v,k)}{\sqrt{d(v)}} \right)^2 \\ & = \operatorname{tr}\left(\mathbf{M}^{\mathrm{T}} \mathbf{X} \Delta \mathbf{X}^{\mathrm{T}} \mathbf{M} \right). \end{aligned} \tag{19}$$

The constraint on the projection matrix $\Phi(\mathbf{M})$ can be chosen accordingly, such as the $l2$-norm constraint in [26] or the $l_{2,1}$-norm in [35] to produce row sparsity to select more informative features.

Both transductive and inductive learning methods have their strengths and limitations. Transductive learning methods have the advantage of using both labeled and unlabeled data simultaneously. However, this line of work also suffers from high computational cost and weak scalability to new data. In contrast, inductive learning is able to deal with new coming data efficiently and effectively, thereby more suitable for applications with large-scale data. However, they do not use the test data in the training process as the transductive learning methods do, and may result in low classification accuracy when the training data size is too small.

4.2 Dynamic Hypergraph Structure Learning

In this section, we discuss dynamically learning an optimal structure for a hypergraph during the learning process. Different from the learning process on a static hypergraph structure as we discussed above, the dynamic structure update of a hypergraph is to dynamically adjust the hypergraph components, including the hyperedge weights, the vertex weights, and the incidence matrix during the learning process, to make the data correlation modeling more accurate. This problem is very important for hypergraph learning since the data correlation modeling strongly depends on the quality of the hypergraph structure.

4.2.1 Hyperedge Weights

Hyperedge weight is used to indicate the importance of different hyperedges. It is crucial to weigh a hyperedge according to its representative capability since different hyperedges may have different accuracy levels in representing the connections among vertices. A typical solution to this problem was an adaptive hyperedge weighting method proposed by Gao *et al*. in [36] where the impact of different relevancy (i.e., high-order connections among subjects) can be automatically modulated by simultaneously updating the weights of both hyperedges and label vectors during the learning process. By setting the sum of all hyperedge weights as 1, this adaptive hyperedge weighting method is a dual-optimization problem that can be formulated as

$$\begin{aligned}\underset{\mathbf{F},\mathbf{W}}{\operatorname{argmin}}\Psi(\mathbf{F}) := &\left\{\mathbf{F}^T\Delta\mathbf{F} + \lambda\|\mathbf{F}-\mathbf{Y}\|^2 + \mu\sum_{e\in\mathcal{E}}\mathbf{W}(e)^2\right\}\\ &\text{s.t.}\ \sum_{e\in\mathcal{E}}\mathbf{W}(e) = 1.\end{aligned} \tag{20}$$

It can be solved by employing the Lagrange multiplier method and the alternating optimization strategy.

4.2.2 Vertex Weights

Different vertices may play different importances in hypergraph learning. Vertex weight is used to weigh the importance of different vertices in the hypergraph. The pioneering work of vertex weighting was proposed by Su *et al*. in [37]. In this work, once the hypergraph is generated, the initial weight of each vertex is computed based on the distribution of training samples of each class. After that, a learning process is performed on the vertex-weighted hypergraph, to estimate the optimal label vectors and vertex weights simultaneously.

Mathematically, let $label_v$ be the category of vertex v, and $\hat{d}_v$ denote the mean distance from the vertex v to all other intra-class training vertices; the weight for vertex v can be written as $\mathbf{U}(v) = \frac{\hat{d}_v}{\sum_{\{u|label_u=label_v\}}\hat{d}_u}$. The Laplacian matrix of the vertex-weighted hypergraph is $\Delta = \mathbf{U} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}$. Then, the learning task can be written as

$$\begin{aligned}\underset{\mathbf{F},\mathbf{W}}{\operatorname{argmin}}\Psi(\mathbf{F}) := &\left\{\mathbf{F}^T\Delta\mathbf{F} + \lambda\|\mathbf{F}-\mathbf{Y}\|^2 + \mu\sum_{e\in\mathcal{E}}\mathbf{W}(e)^2\right\}\\ &\text{s.t.}\ \mathbf{W}(e)\geq 0, \sum_{e\in\mathcal{E}}\mathbf{H}(v,e)\mathbf{W}(e) = \mathbf{D}_v(v).\end{aligned} \tag{21}$$

It can also be solved by employing the Lagrange multiplier method and alternative optimization as Eq. (20).

4.2.3 Hypergraph Structure

Different from hyperedge or vertex weight learning, which are based on a given hypergraph structure (i.e., fixed incidence matrix), hypergraph structure learning is to update the hypergraph structure, i.e., optimizing the incidence matrix. Hypergraph structure learning could be performed together with the label prediction of unlabeled vertices within the same framework. We next introduce a typical hypergraph structure learning method [38] proposed by Zhang *et al*. , where a dual-optimization framework is introduced to learn the label vectors and the incidence matrix jointly. The objective function is formulated as

$$\underset{\mathbf{F},0\preceq\mathbf{H}\preceq 1}{\operatorname{argmin}}\Psi(\mathbf{F}) := \left\{\Omega(\mathbf{F},\mathbf{H}) + \lambda\mathcal{R}_{\text{emp}}(\mathbf{F}) + \mu\Phi(\mathbf{H})\right\}. \tag{22}$$

Here, the regularizer term $\Omega(\mathbf{F},\mathbf{H})$ and loss term $\mathcal{R}_{\text{emp}}(\mathbf{F})$ apply the formation of the transductive model in Eq. (16). $\Phi(\mathbf{H})$ denotes the constraint of $\mathbf{H}$ with respect to the input features $\mathbf{X}$. Based on the fact that there should be more and stronger connections between two subjects sharing similar features, the regularizer on $\mathbf{H}$ is formulated as

$$\Phi(\mathbf{H}) = \operatorname{tr}\left(\left(\mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{D}_v^{-\frac{1}{2}}\right)\mathbf{X}\mathbf{X}^{\mathrm{T}}\right). \tag{23}$$

An alternative optimization scheme is employed to solve the model in Eq. (22).

Hypergraph structure learning is effective in modeling data correction since it correlates the feature space and label space. This line of method inherently has the merit of dynamically updating the hypergraph structure during the learning process, but its disadvantages are also obvious: the non-convex model and high computational cost.

4.3 Hypergraph Learning for Multi-Modal Data

To apply hypergraph learning to multi-modal data, both the transductive hypergraph learning methods (see Section 4.1.1) and inductive hypergraph learning methods (see Section 4.1.2) can be further extended to the multi-modal cases.

4.3.1 Multi-Modal Transductive Hypergraph Learning

Suppose a subject has m modalities; for each modality, we can generate hyperedges and construct a hypergraph by considering each subject as a vertex. In this way, we can construct m hypergraphs $\mathcal{G}_1 = (\mathcal{V}_1,\mathcal{E}_1,\mathbf{W}_1), \mathcal{G}_2 = (\mathcal{V}_2,\mathcal{E}_2,\mathbf{W}_2),\ldots,\mathcal{G}_m = (\mathcal{V}_m,\mathcal{E}_m,\mathbf{W}_m)$ for the data with m modalities. To achieve a global effectiveness, an optimal weight α_i for the hypergraph $\mathcal{G}_i$ can be jointly learned with the vertex label vectors [7]. The learning objective function can be written as

$$\underset{\mathbf{F},\alpha}{\operatorname{argmin}}\Psi(\mathbf{F},\alpha) := \left\{\Omega(\mathbf{F},\alpha) + \lambda\mathcal{R}_{emp}(\mathbf{F}) + \mu\Phi(\alpha)\right\}. \tag{24}$$

Here, $\Phi(\alpha)$ denotes the constraint on the hypergraph weights. Extending the regularizer term for a single hypergraph in Eq. (15), the term on multiple hypergraphs can be defined as

$$\Omega(\mathbf{F},\alpha) = \mathbf{F}^T\sum_{i=1}^{m}\alpha_i(\mathbf{I}-\Theta_i)\mathbf{F}, \tag{25}$$

where $\Theta_i = \mathbf{D}_{vi}^{-1/2}\mathbf{H}_i\mathbf{W}_i\mathbf{D}_{ei}^{-1}\mathbf{H}_i^T\mathbf{D}_{vi}^{-1/2}$. Regarding the fact that the sum of the hypergraph weights is 1, Eq. (24) can be further written as

$$\begin{aligned}&\underset{\mathbf{F},\alpha}{\operatorname{argmin}}\Psi(\mathbf{F},\alpha) :=\\ &\left\{\mathbf{F}^T\sum_{i=1}^{m}\alpha_i(\mathbf{I}-\Theta_i)\mathbf{F} + \lambda\|\mathbf{F}-\mathbf{Y}\|^2 + \mu\sum_{i=1}^{m}\alpha_i^2\right\},\\ &\text{s.t.}\ \sum_{i=1}^{m}\alpha_i = 1.\end{aligned} \tag{26}$$

Eq. (26) can be solved by alternative optimization scheme.

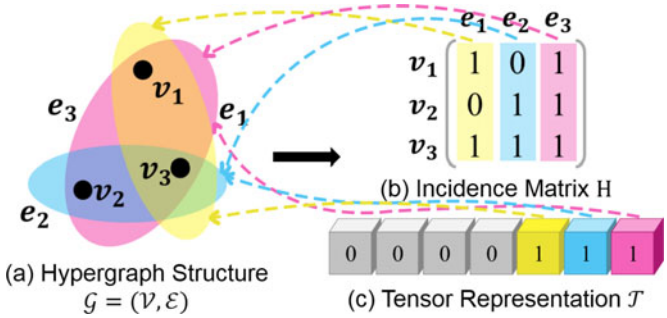


Fig. 3. (a) Hypergraph structure. (b) Traditional incidence matrix representation of a hypergraph structure. (c) Proposed tensor representation of a hypergraph structure.

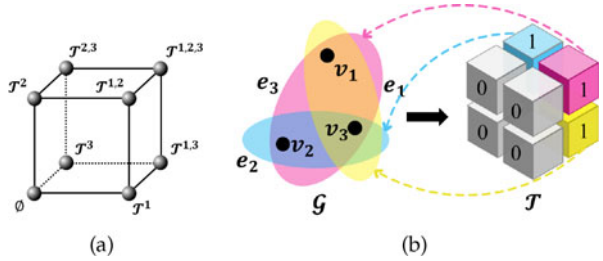


Fig. 4. (a) A 3-order tensor. (b) Tensor representation of a hypergraph with 3 vertices.

4.3.2 Multi-Modal Inductive Hypergraph Learning

As discussed in Section 4.1.2, the basic idea of inductive hypergraph learning is to learn a projection matrix from the feature space to the label space. We use $\{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^m\}$ to denote the feature vectors of the subjects in the m modalities. In the offline training stage, m hypergraphs are constructed to model the high-order correlation of the training data in different modalities. For each modality, an individual projection matrix, $\mathbf{M}_i$, is learned to project subjects from the feature space, $\mathbf{X}^i$, to the label space $\mathbf{F}$. To achieve global effectiveness, the weight α_i of each modality can be jointly learned with the projection matrix $\mathbf{M}_i$. Therefore, the inductive learning task on the multi-hypergraph can be written as

$$\begin{aligned} &\underset{\mathbf{M}_i, \alpha_i \leq 0}{\operatorname{argmin}} \Psi(\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_m, \alpha) := \\ &\sum_{i}^{m} \alpha_i \left\{ \Omega(\mathbf{M}_i) + \lambda \mathcal{R}_{emp}(\mathbf{M}_i) + \mu_1 \Phi(\mathbf{M}_i) \right\} + \mu_2 \sum_{i=1}^{m} \alpha_i^2, \quad (27) \\ &\text{s.t.} \sum_{i=1}^{m} \alpha_i = 1. \end{aligned}$$

This problem can be solved by first optimizing each $\mathbf{M}_i$ individually and then optimizing α_i [35]. With the learned $\mathbf{M}_i$ and α_i, in the online inference stage, the final label vector of a test vertex v can be achieved by

$$\mathbf{F}(v) = \sum_{i=1}^{m} \alpha_i \mathbf{X}^i(v)^T \mathbf{M}_i, \quad (28)$$

where $\mathbf{X}^i(v)$ is the ith feature vector for vertex v.

5 Tensor-Based Dynamic Hypergraph Learning

Although there have been attempts on optimizing the hypergraph structure, there still exists a long way to model the dynamic correlations in the learning process well. The primary limitations that stem from the challenges in the dynamic hypergraph learning problem are threefold.

- *Fixed number and order of hyperedges.* Regardless of optimizing the hyperedge weights [36] or the correlation between vertices and hyperedges [38], the number and order of hyperedges are fixed during the learning process, and thus the hypergraph structure is just partially optimized. We need to find a more flexible way to optimize the hypergraph in the complete space, which means the hyperedges of every order should be considered.
- *Non-convex optimization model.* The optimization models of existing dynamic hypergraph structure learning methods are mostly non-convex. The Laplacian of the hypergraph is complicated and it is a non-convex function about the hyperedge weights $\mathbf{W}$ or the incidence matrix $\mathbf{H}$. Thus, the models mentioned above may suffer from serious convergence issues in practice.
- *High computational cost.* The time complexity of existing methods , especially the DHSL method [38], is very high, which may take much more time to update the incidence matrix. The high representation complexity of the incidence matrix makes it expensive to compute gradients and optimize the objective function.

To tackle the above problems, we leverage a tensor representation to characterize the dynamic hypergraph structure more flexibly, and propose the tensor-based dynamic hypergraph learning method (t-DHL) to efficiently learn on the dynamic hypergraph. Unlike the incidence matrix, not only weights, but the number and order of hyperedges can also be changed when optimizing the tensor representation. It is due to that the tensor representation can describe the hyperedges of all orders, which is a complete representation of the hypergraph structure, as shown in Fig. 3. The tensor-based dynamic hypergraph learning is formulated as a bi-convex model to ensure that the model can converge to the optimal solution efficiently, which is illustrated in Fig. 4. Regarding that the tensor representation has several redundant 0's, we adopt several sampling strategies to reduce the complexity of the model so that it can be scaled to large datasets. After sampling, the number of 0 is very limited and will not have much impact on space complexity and time complexity. In addition to the novel t-DHL framework, we also discuss the potential of the tensor-based hypergraph in deep hypergraph learning applications.

5.1 Tensor Representation of Dynamic Hypergraph

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, we use an N-order tensor in 2-D space to represent a hypergraph with N vertices. For example, a hypergraph with 3 vertices can be represented by a $2 \times 2 \times 2$ tensor (see Fig. 5). For the sake of brevity, we show the tensor $\mathcal{T}$ in an unfolded form, i.e., a $2^N - 1$ dimensional vector, as shown in Fig. 3c. Let $\bar{\Psi}_N$ be the power set of $\Omega = \{1, 2, \ldots, N\}$, and let $\Psi_N = \bar{\Psi}_N - \{\emptyset\}$

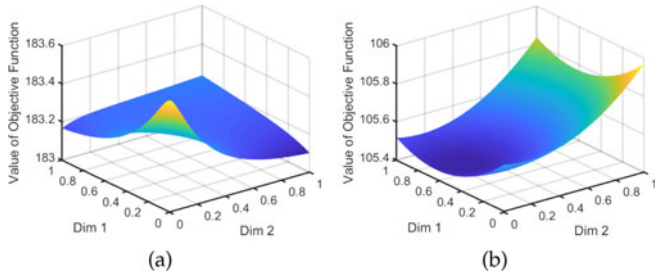


Fig. 5. (a) 2-D projection of the optimization space of DHSL model with respect to $\mathbf{H}$. The optimization model of DHSL is non-convex. (b) 2-D projection of the optimization space of t-DHL model with respect to $\mathcal{T}$. The optimization model of t-DHL is bi-convex.

index the 2^N-1 elements in tensor $\mathcal{T}$. Let ω denote a set in Ψ_N, also called as a clique hereinafter, and $\mathcal{T}(\omega)$ denote the connection strength of clique ω. In this way, $\mathcal{T}$ records all possible cliques and the associated connection strengths. There exists a hyperedge connecting all vertices in clique ω if $\mathcal{T}(\omega)\neq 0$. Let $\mathcal{T}_0$ represent the initial hypergraph structure; the adjustment of the hypergraph structure can be achieved by directly modifying the values of the corresponding elements. By means of tensor representation, the hypergraph structure can be optimized more flexibly, since the number, order, and weights of hyperedges are all variable during the learning process, which breaks through the limitations of the traditional incidence matrix.

5.2 Formulation of Dynamic Hypergraph Learning

For each clique, we use a potential function to estimate the data distribution on it. Since we have both the predicted label information and the input features, the cost function is with respect to two spaces, i.e., the label space and the feature space. Cliques with dense data distribution should have higher potential than those with the scattered distribution. Therefore, the potential function of ω can be written as

$$\xi(\omega)=\sum_{u,v\in\omega}\frac{\left(\|\mathbf{F}(u)-\mathbf{F}(v)\|^2+\alpha\|\mathbf{X}(u)-\mathbf{X}(v)\|^2\right)}{(1+\alpha)\delta(\omega)}, \tag{29}$$

where α is the trade-off parameter to balance the importance of label information and feature information. The hypergraph-based loss term encourages cliques with higher potential associating with lower weights and vise versa

$$\Omega(\mathbf{F},\mathcal{T})=\sum_{\omega\in\Psi_N}\mathcal{T}(\omega)\xi(\omega). \tag{30}$$

During the learning process, the learned hypergraph structure and label vector are expected not to change significantly within the assumed range. Therefore, we further apply the empirical loss of $\mathcal{T}$ and $\mathbf{F}$ as the regularity terms

$$\mathcal{R}_{emp}(\mathbf{F})=\|\mathbf{F}-\mathbf{Y}\|_F^2, \tag{31}$$

and

$$\mathcal{R}_{emp}(\mathcal{T})=\|\mathcal{T}-\mathcal{T}_0\|^2. \tag{32}$$

Then, the objective function for learning on a dynamic hypergraph can be written as the following dual optimization problem:

$$\begin{aligned}&\arg\min_{\mathbf{F},0\preceq\mathcal{T}\preceq 1}\Psi(\mathbf{F},\mathcal{T})\\&:=\Omega(\mathbf{F},\mathcal{T})+\lambda\mathcal{R}_{emp}(\mathbf{F})+\mu\mathcal{R}_{emp}(\mathcal{T})\\&:=\sum_{\omega\in\Psi_N}\mathcal{T}(\omega)\xi(\omega)+\lambda\|\mathbf{F}-\mathbf{Y}\|_F^2+\mu\|\mathcal{T}-\mathcal{T}_0\|^2,\end{aligned} \tag{33}$$

where λ and μ are parameters to balance different components in the objective function.

5.3 Solution of the Optimization Task

The optimization problem in Eq. (33) is a biconvex model with respect to the label vectors $\mathbf{F}$, and the dynamic hypergraph structure $\mathcal{T}$. It is pointed out by Gorski *et al.* in [39], that the biconvex function will converge to the optimal solution if the alternative optimization method converges. Therefore, we employ the alternative optimization scheme to update each variable when fixing the other.

5.3.1 Optimization on $\mathbf{F}$

We first fix the tensor $\mathcal{T}$, let $\Psi_{uv}=\{\omega\in\Psi_N|u\in\omega,v\in\omega\}$, the sub-problem on optimizing $\mathbf{F}$ can be written as

$$\begin{aligned}&\arg\min_{\mathbf{F}}\Psi(\mathbf{F}):=\Omega(\mathbf{F})+\lambda\mathcal{R}_{emp}(\mathbf{F})\\&:=c\sum_{\omega\in\Psi_N}\frac{\mathcal{T}^{\omega}}{\delta(\omega)}\sum_{u,v\in\omega}\|\mathbf{F}(u)-\mathbf{F}(v)\|^2+\lambda\|\mathbf{F}-\mathbf{Y}\|_F^2\\&:=c\sum_{u,v}\|\mathbf{F}(u)-\mathbf{F}(v)\|^2\sum_{\omega\in\Psi_{uv}}\frac{\mathcal{T}^{\omega}}{\delta(\omega)}+\lambda\|\mathbf{F}-\mathbf{Y}\|_F^2\\&:=c\sum_{u,v}\|\mathbf{F}(u)-\mathbf{F}(v)\|^2\mathbf{S}_{uv}+\lambda\|\mathbf{F}-\mathbf{Y}\|_F^2\\&:=2c\mathrm{tr}(\mathbf{F}^{\mathrm{T}}(\mathbf{D}_{\mathbf{S}}-\mathbf{S})\mathbf{F})+\lambda\|\mathbf{F}-\mathbf{Y}\|_F^2,\end{aligned} \tag{34}$$

where $c=\frac{1}{1+\alpha}$ and $\mathbf{S}_{uv}=\sum_{\omega\in\Psi_{uv}}\frac{\mathcal{T}^{\omega}}{\delta(\omega)}$. $\mathbf{D}_{\mathbf{S}}$ is a diagonal matrix whose elements are the sums of each row of $\mathbf{S}$. $\mathbf{S}$ can be regarded as the similarity matrix between vertices.

Similar to traditional hypergraph learning, the optimization of Eq. (34) has a closed-form solution. $\mathbf{F}$ can be obtained directly as

$$\mathbf{F}=\left(\mathbf{I}+\frac{2c}{\lambda}(\mathbf{D}_{\mathbf{S}}-\mathbf{S})\right)^{-1}\mathbf{Y}. \tag{35}$$

5.3.2 Optimization on $\mathcal{T}$

After the optimization of $\mathbf{F}$, we fix $\mathbf{F}$ and optimize $\mathcal{T}$, and the learning objective function is written as

$$\begin{aligned}&\arg\min_{0\preceq\mathcal{T}\preceq 1}\Psi(\mathcal{T}):=\Omega(\mathcal{T})+\mu\mathcal{R}_{emp}(\mathcal{T})\\&:=\sum_{\omega\in\Psi_N}\mathcal{T}(\omega)\xi(\omega)+\mu\|\mathcal{T}-\mathcal{T}_0\|^2.\end{aligned} \tag{36}$$

Eq. (36) is a bound constraint optimization problem and can be solved by the projected gradient method [40]. The derivation with respect to $\mathcal{T}$ can be written as

$$\nabla\Psi(\mathcal{T}) = \xi + 2\mu(\mathcal{T} - \mathcal{T}_0). \tag{37}$$

Then, $\mathcal{T}$ can be updated by iterating

$$\mathcal{T}_{k+1} = \boldsymbol{P}[\mathcal{T}_k - h\nabla\Psi(\mathcal{T}_k)], \tag{38}$$

where h is the optimal step size at each iteration and $\boldsymbol{P}$ is the projection onto the feasible set $\{\mathcal{T}|0 \preceq \mathcal{T} \preceq 1\}$. Here $\boldsymbol{P}$ is set as

$$\boldsymbol{P}[\mathcal{T}^\omega] = \begin{cases} \mathcal{T}^\omega & if\ 0 \le \mathcal{T}^\omega \le 1 \\ 0 & if\ \mathcal{T}^\omega < 0 \\ 1 & if\ \mathcal{T}^\omega > 1 \end{cases}. \tag{39}$$

With the updated $\mathcal{T}$, the label projection matrix $\mathbf{F}$ is further optimized with the fixed $\mathcal{T}$. The procedure repeats until the objective function converges.

We note that in Eq. (36), the optimization function on $\mathcal{T}$ is convex. To show this, we randomly select two elements of $\mathcal{T}$ and let them change from 0 to 1, while fixing the other elements. We illustrate the change of the objective function value with respect to the two elements in Fig. 5b. As a comparison, we similarly demonstrate the 2-dimensional projection of the optimization space of the DHSL model with respect to $\mathbf{H}$ in Fig. 5a. Obviously, by using the tensor representation of the hypergraph, the t-DHL model can solve the non-convex problem better.

5.4 Complexity Analysis

Given N samples, there are $2^N - 1$ possible connections and the dimension of $\mathcal{T}$ is $2^N - 1$. The optimization of $\mathcal{T}$ requires exploring the complete space, which is infeasible for large-scale hypergraphs. To address this issue, we conduct sampling on all possible connections to reduce the search space. Specifically, we only record and update part of the elements in tensor $\mathcal{T}$ rather than learning the full model. In many real-world applications, there is an assumption that the hypergraph has a small size of large cliques. Based on this assumption, we only consider the cliques containing no more than $\lceil \ln N \rceil$ vertices. What's more, the information from local neighborhoods plays a more important role than that from distant neighborhoods. Therefore, for each vertex, only cliques containing the center vertex and its k nearest neighborhoods, i.e., $k = 1, 2, \ldots, \lceil \ln N \rceil$, are selected.

In this way, the dimension of $\mathcal{T}$ (the space complexity) is reduced to $\lceil \ln N \rceil N$. When optimizing label projection matrix $\mathbf{Y}$, we first calculate the similarity matrix $\mathbf{S}$, then calculate $\mathbf{Y}$. The computational complexities of these two steps are $\mathcal{O}(\lceil \ln N \rceil^3 N)$ and $\mathcal{O}(N^3)$, respectively. The computational complexity of optimizing tensor representation $\mathcal{T}$ is $\mathcal{O}(\eta \lceil \ln N \rceil N)$, where η is the iteration number of optimizing $\mathcal{T}$. Hence, the overall computational complexity is $\mathcal{O}(\kappa N(\lceil \ln N \rceil^3 + N^2 + \eta \lceil \ln N \rceil))$, where κ is the number of alternate iterations.

Computational complexity is a common issue for graph or hypergraph based learning methods. Updating the hypergraph structure would increase the computational cost compared with traditional methods, but it is acceptable considering its benefit on improving accuracy. Moreover, our tensor-based hypergraph learning method has less computational complexity than the state-of-the-art hypergraph-based learning method based on the incidence matrix [38]. In Table 3, we summarize the time complexity of existing hypergraph learning methods for single modality data. We note that the time complexity of updating the incidence matrix once in [38] is $\mathcal{O}(N^3 + EN^2 + cN^2 + dN^2)$, while that of updating the tensor representation once in our method is $\mathcal{O}(\lceil \ln N \rceil N)$. And such calculation will be repeated $\kappa * \eta$ times in the whole learning process. Thus, the tensor-based dynamic hypergraph learning is much more efficient in optimizing the hypergraph structure than the conventional method based on the incidence matrix in [38].

TABLE 3
The Time Complexity of Hypergraph Learning Methods

Methods		Time Complexity
Transductive Hypergraph Learning [22]		$\mathcal{O}(N^3 + EN^2 + cN^2)$
Inductive Hypergraph Learning [35]	Training	$\mathcal{O}(\kappa(EN^2+dN^2+d^2N+cdN+d^3))$
	Testing	$\mathcal{O}(cdN)$
Structure Update	Hyperedge Weight [36]	$\mathcal{O}(\kappa(N^3 + EN^2 + cN^2))$
	Vertex Weight [37]	$\mathcal{O}(\kappa(N^3 + EN^2 + cN^2))$
	Incidence Matrix [38]	$\mathcal{O}(\kappa\eta(N^3 + EN^2 + cN^2 + dN^2))$
Tensor-based Hypergraph Learning		$\mathcal{O}(\kappa N(\lceil \ln N \rceil^3 + N^2 + \eta \lceil \ln N \rceil))$

N denotes the number of training/testing/total subjects, E denotes the number of hyperedges, d denotes the dimension of features, c denotes the number of classes, η denotes the inner iteration times if applicable, and κ denotes the outer iteration times if applicable.

5.5 Potential Application to Deep Hypergraph-Based Learning

Hypergraph-based deep learning methods have attracted much attention recently. In hypergraph neural networks (HGNN) [41], a spectral convolutional layer was introduced to handle the high-order data correlation for representation learning. However, this framework only employed a static initially constructed hypergraph structure. To further dynamically update the hypergraph structure with the adjusted feature embedding, Jiang *et al.* [42] proposed dynamic hypergraph neural networks (DHGNN), in which the hypergraph reconstruction modules and the hypergraph convolution modules were stacked alternatively. Unfortunately, although a new hypergraph was regenerated after each convolution operation, the change of hypergraph structure was not continuous and differentiable under such an update strategy, which might lead to non-convergence.

Tensor-based hypergraphs have great potential in designing new dynamic hypergraph convolutional networks. The main limitation of previous methods is that the hypergraph convolution operation based on the incidence matrix prevents the hypergraph structure from being optimized during the learning process. If we construct the convolution operation using the tensor representation of the hypergraph, this problem will be significantly alleviated. The tensor representation can be assigned with a gradient in the backpropagation process and optimized step by step.

It allows us to flexibly increase, decrease, strengthen, or weaken the hyperedges to optimize the hypergraph structure when training the networks.

Another potential application of a tensor-based hypergraph is the deep hypergraph generative model. As discussed in Section 3, traditional hypergraph generation methods are hand-engineered to construct a hypergraph of a specific order, and thus do not have the capability of learning a hypergraph generative model from observed data automatically. Although there have been a body of graph generative models [43], [44], [45], little efforts have been made on hypergraph generative models. Tensor representation makes it possible to estimate distribution over possible hypergraph structures without assuming a fixed set of hyperedges. The deep hypergraph generative models may play an essential role in real-world applications, such as the discovery of new molecular structures and the establishment of knowledge graphs.

6 Applications

Hypergraph learning based methods have a wide range of applications. The scale of data that can be processed can range from hundreds to tens of thousands. Moreover, these methods can be applied to not only the single modality data but also the multi-modal cases. In this section, several typical applications are selected as examples to show how hypergraph learning methods are used in various situations. These applications are classified into two categories: the classification and clustering based applications.

6.1 Classification

6.1.1 3D Object Classification

Three-dimensional object classification has become an increasingly important problem in recent years. The key task of 3D object classification is to identify the category for a 3D object captured by 3D photography equipment. In general, there are four different ways to describe a 3D object, including multi-view, point cloud, mesh, and voxel. Many efforts have been made to extract features from these four representations of 3D objects [46], [47], [48], [49]. However, each representation has its own advantages and disadvantages, and there is no most suitable representation. Under such circumstances, it is necessary to consider how to combine the information from multiple representations together for classification. To apply hypergraph learning to this task, a multi-hypergraph can be generated by considering each 3D object as a vertex and constructing a sub-hypergraph for each modality. Given a set of labeled objects and test objects, the objective of this task is to classify the labels of the test objects. Specifically, we have the label matrix of labeled objects, and the incidence matrices of multi-hypergraph structure. The label matrix of testing objects can be learned via multi-modal transductive/inductive hypergraph learning.

6.1.2 Microblog Sentiment Analysis

As social network users often express sentiments on events on the Internet, microblogs, such as Twitter and Weibo, have accumulated an abundance of social media data. Therefore, analyzing the sentiments in these microblogs has become an important problem for many applications, such as opinion mining [50]. In this task, microblog data are usually composed of three modality data, i.e., images, text, and emoticons. Visual features, textual features, and emoji features can be extracted for each microblog, and three hypergraphs can be generated accordingly. Given microblogs with positive or negative sentiment labels known for training, a primary goal of sentiment analysis is to predict the sentiments of test microblog data, which is a multi-modal node classification problem and can be solved by multi-modal transductive or inductive hypergraph learning methods [51]. In practice, the number of microblogs to be processed can be enormous, and this requires high computational efficiency for the prediction model. Therefore, inductive learning methods are more practical than transductive learning methods on this application.

6.1.3 Action Recognition

Action recognition is an important task with many applications like video surveillance, video indexing, and social sciences. The goal of this task is to identify different actions from video clips captured by various sensors such as RGB-D sensors and RGB sensors, which can be addressed with hypergraph learning. In this task, given a set of training and testing actions, gesture recognition methods like HON4D [52] can be used to extract the action features which characterize both 4-D (time and spatial coordinates) motion and geometry information. After feature extraction, each action can be represented by a vector of features, and the distance between two actions in the feature space can be calculated. Then, a hypergraph can be constructed based on the distance-based hyperedge generation scheme, where each vertex represents an action. Based on this hypergraph, existing hypergraph learning methods and the proposed tensor-based dynamic hypergraph learning method can be used for gesture recognition. As shown in [38], the recognition accuracy of hypergraph-based learning methods can significantly outperform traditional gesture recognition methods, which can be attributed to its capability of modeling high-order correlation among different action samples.

6.2 Clustering

Clustering is an important fundamental problem and has many applications, such as handwritten digit recognition [53], face recognition [54], and graphical pattern grouping [55]. We take face recognition as an example to explain how a hypergraph can be applied to clustering problems. A major goal of face recognition is to identify a unique face image from the face image dataset taken under different view conditions. To this end, a hypergraph based learning method can be used. First, a hypergraph can be built on the face image dataset where each face photo is associated with a vertex in the hypergraph. After extracting the features of all face images, distance-based or representation-based hyperedges can be generated as discussed in Section 3, and a hypergraph can be constructed accordingly. Then, the eigenvectors $\Upsilon = [\Upsilon_1 \dots \Upsilon_k]$ associated with the k smallest eigenvalues of the Laplacian matrix can be calculated to obtain a k-way partition. The row vectors of Υ are the hypergraph embedding of vertices, which are expected to

be well separated in the k-dimensional Euclidian space. Lastly, a clustering algorithm like k-means can be run on these vectors to divide them into k clusters, each one containing the images of a unique face.

To sum up, to apply hypergraph learning to a specific classification or clustering task, the first step is to model the problem with the hypergraph structure (e.g., defining the object that a vertex of the hypergraph refers to). The second step is to construct the hypergraph based on the features, attributes, or prior correlations of the objects. The last step is to conduct learning on this hypergraph with an appropriate hypergraph learning method.

7 Experimental Evaluations

In this section, we mainly evaluate the performances of hypergraph learning methods, including the proposed t-DHL method on several classification and clustering tasks.

7.1 On Classification Tasks

7.1.1 Experimental Settings

To investigate the effectiveness of existing hypergraph generation and learning methods and the proposed t-DHL method, we conduct experiments on four datasets, including the Princeton ModelNet dataset (ModelNet40) [56], the Microblog dataset [57], the Facebook Page-Page dataset [58], and the MSR Gesture 3D dataset [59]. These four datasets are employed for the tasks of 3D object classification, microblog sentiment analysis, and gesture recognition, respectively. The description of these datasets is shown in Table 4.

TABLE 4
Datasets Used for the Classification Task

Dataset	Vertex	Class	Modality	Feature	Task
ModelNet40	12,311	40	meshnet mlvcnn pointnet pvrnet	1,024 512 512 256	①
Microblog	4,012	2	text emoji visual	2,547 49 1,553	②
Facebook Page-Page	22,470	4	single	16	②
MSRGesture3D	333	12	single	2,400	③
MNIST	2,000	10	single	784	④
YALE B	2,414	38	single	1,024	④

Task ①: 3D object classification. Task ②: microblog sentiment analysis. Task ③: action recognition. Task ④: clustering.

The Princeton ModelNet dataset contains 127,915 CAD models from 662 object categories. In our experiments, its subset, i.e., ModelNet40, which consists of 12,311 CAD models from 40 object categories, is used. The split of training and test setting follows [56]. In our experiments, 3D model features are extracted using different shape representation methods, including MeshNet for mesh data [47], MLVCNN for view data [48], and PointNet for point cloud data [46]. The experimental settings follow the original works.

The Microblog dataset [57] is a large-scale multi-modal microblog sentiment prediction dataset, which is collected from Sina Weibo and covers trivial affairs, weather, work, stars, films, and international events. There are over 5,500 microblogs with sentiment labels, including 4,196 positive and 1,354 negative tweets in this dataset. Each microblog contains textual, visual, and emoticon data. Experiments are conducted on ten sets of data, and the average prediction performance is calculated for comparison. For each set of data, 80 percentage samples from the dataset are randomly selected as the training set, and the remaining 20 percentage samples are used for testing.

The Facebook Page-Page dataset is a webgraph including 22,470 Facebook pages and 171,002 mutual links between them. Node embedding is conducted using a graph convolutional network (GCN) [60] to extract features for each page. All pages can be divided into four categories, i.e., politicians, governmental organizations, television shows, and companies. Besides the accuracy rate, the test performance is further evaluated by weighted, micro, and macro F_1 scores [58]. In this experiment, 80 percent of nodes are used for training and the remaining 20 percent are used for testing.

The MSR Gesture 3D (MSRGesture3D) dataset [61] is a hand gesture dataset of depth image sequences. There are 12 classes of gestures defined by American sign language (bathroom, blue, finish, green, hungry, milk, past, pig, store, where, j, and z) and performed by 10 subjects in this dataset. In total, this dataset consists of 333 depth image sequences. We conduct ten instances of experiments. For the training/testing data splitting, we randomly select $N = 1, 2, 3, 4, 6, 8, 10$ samples as the training data and use the remaining samples as testing data for each gesture category. For each number $\in N$, we conduct ten times of data splitting and use the average recognition precision on these data for the comparison.

7.1.2 Compared Methods

To fairly compare hypergraph generation methods, we first construct hypergraphs based on different comparison hypergraph generation schemes and then evaluate the classification performances of performing the same hypergraph learning method [22] on those hypergraphs generated by the different hypergraph generation schemes. In this set of experiments, we mainly compare those implicit hypergraph generation schemes include k-NN, ϵ-ball, k-means, and $l1$-hypergraph.

We conduct two sets of experiments to investigate the performances of hypergraph learning methods. The first set of experiments compare the proposed tensor-based hypergraph learning (t-DHL) method and the following four representative single modality learning methods:

- Traditional hypergraph learning (HL) method [22],
- Hypergraph learning (HL-W) with learn-able subhyperedge weights [36],
- Inductive hypergraph learning (IHL) method [35],
- Dynamic hypergraph structure learning (DHSL) [38].

This set of experiments are conducted on the Microblog, Facebook Page-Page, and MSRGesture3D datasets.

The second set of experiments compare three multi-modality learning methods:

TABLE 5
Prediction Accuracy Comparison on the Microblog Dataset

Modality	HL(ϵ-ball)	HL(k-means)	HL($l1$)	HL(k-NN)	IHL	HL-W	DHSL	t-DHL
text	0.7381	0.789	0.7746	0.7751	0.7534	0.7751	0.791	**0.7993**
emojicon	0.9297	0.9192	0.9372	0.9337	**0.9414**	0.9337	0.9384	0.9364
visual	0.7933	0.794	0.7963	0.8042	0.7945	0.8042	0.7983	**0.8044**

TABLE 6
Classification Accuracy and F_1 Scores on the Facebook Page-Page Dataset

Evaluation Metric	GCN	HL(ϵ-ball)	HL(k-means)	HL($l1$)	HL(k-NN)	IHL	HL-W	DHSL	t-DHL
Acc	0.9211	0.9206	0.9183	0.9223	0.9219	0.9206	0.9219	0.9228	**0.9239**
Weighted F_1	0.9209	0.9204	0.9183	0.9224	0.922	0.9203	0.922	0.9227	**0.924**
Micro F_1	0.9211	0.9206	0.9183	0.9223	0.9219	0.9206	0.9219	0.9228	**0.9239**
Macro F_1	0.9167	0.9149	0.9123	0.9171	0.9162	0.9143	0.9162	0.9172	**0.9182**

TABLE 7
Classification Accuracy Comparison on the MSRGesture3D Dataset

Number of Training Samples per Class	HL(ϵ-ball)	HL(k-means)	HL($l1$)	HL(k-NN)	IHL	HL-W	DHSL	t-DHL
1	0.4044	0.3006	0.5037	0.5044	0.2704	0.504	**0.5199**	0.5134
2	0.5216	0.409	0.5867	0.5887	0.4362	0.5887	0.6016	**0.6026**
3	0.5377	0.4401	0.6003	0.6064	0.5013	0.6064	**0.6239**	0.6222
4	0.574	0.4621	0.646	0.6533	0.5881	0.6533	0.6667	**0.6698**
6	0.6188	0.5579	0.7023	0.6958	0.6897	0.6958	0.6985	**0.7195**
8	0.646	0.6122	0.7498	0.7485	0.7224	0.7489	0.7494	**0.7717**
10	0.6718	0.6254	0.7765	0.7751	**0.7958**	0.7751	0.7831	0.7892

- Multi-hypergraph learning (MHL) [7]. In this method, each sub-hypergraph is assigned with equal weights.
- Multi-hypergraph learning with learn-able sub-hypergraph weights (MHL-W) [7].
- Inductive multi-hypergraph learning (IMHL) [35].

This set of experiments are conducted on the Microblog and ModelNet40 datasets.

7.1.3 Experimental Results and Discussion

From the experimental results of investigating the performances of different hypergraph generation methods, we observe there is neither a method demonstrating a consistent superiority over other methods, nor any method being superior in a particular modality. On the Facebook Page-Page dataset and MSRGesture3D dataset, HL($l1$) and HL(k-NN) obtains the best performance, respectively. However, on the Microblog dataset, HL(k-means), HL($l1$) and HL(k-NN) surpass other methods in the text, emojicon and visual modalities, respectively. We note that both k-NN and $l1$ hypergraph generation schemes can achieve relatively promising classification results in most cases. The ϵ-ball and k-means hypergraph generation schemes are, however, more sensitive to the data type and less robust. For example, the classification accuracy of HL(ϵ-ball) and HL(k-means) are 19.71 and 40.32 percent lower than HL($l1$), respectively, on the MSRGesture3D dataset when the number of training samples per class is 1.

Comparison results of single modality hypergraph learning methods are shown in Tables 5, 6, and 7. We can see the t-DHL method can achieve the highest performance in most cases. On the Microblog dataset, the accuracy of t-DHL taking as input the text modality is 79.93 percent, which is 3.12, 6.09, 3.12, and 1.05 percentage points higher than HL(k-NN), IHL, HL-W and DHSL, respectively. On the Facebook Page-Page dataset, t-DHL has better performance on classification accuracy and F_1 values. On the MSRGesture3D dataset, t-DHL outperforms HL(k-NN), IHL, HL-W, and DHSL by 3.1, 6.82, 3.04 and 2.98 percentage points, respectively, when the number of training samples is 8. HL-W achieves a very close accuracy performance with HL, which indicates that the accuracy gains of learning hyperedge weights are very limited. In most cases, both DHSL and t-DHL outperform HL, IHL, and HL-W consistently, which indicates the superiority of the dynamic hypergraph structure over the static hypergraph structure. Since the dynamic hypergraph structure is optimized in both the label space and the feature space, the connections among the vertices with the same label and similar features can be stronger than those with different labels or dissimilar features. This dynamic hypergraph strategy can model the data more effectively and lead to better performance compared with traditional hypergraph learning methods.

When the size of training data is not large enough, transductive hypergraph learning methods, including HL, HL-W, DHSL and t-DHL, outperform the inductive hypergraph learning method IHL. For example, on the MSRGesture3D

TABLE 8
Classification Accuracy Comparison on Multi-Modality Datasets

Dataset	Modality	MHL	MHL-W	IMHL
Microblog	text+emojicon	0.881	0.8818	**0.9232**
	emojicon+visual	0.8711	0.894	**0.9317**
	text+visual	0.8067	0.8065	**0.8015**
	all	0.8516	0.8693	**0.9349**
ModelNet40	meshnet+mlvcnn	0.9395	0.9391	**0.9456**
	mlvcnn+pointnet	0.9245	0.9253	**0.9403**
	meshnet+pointnet	0.9156	0.9156	**0.9282**
	all	0.9351	0.9428	**0.9476**

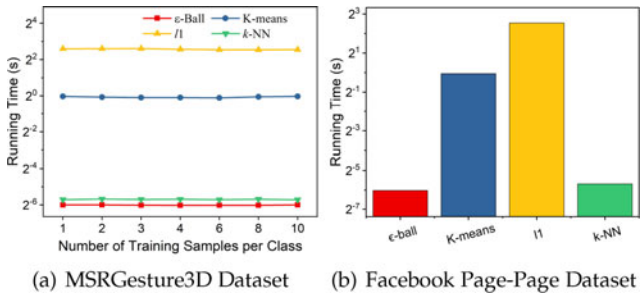


Fig. 6. Running time comparison of hypergraph generation methods.

dataset, when 1, 2, and 3 training clips are employed for each gesture, HL achieves respectively the improvement of 86.54, 34.96 and 20.97 percent compared with IHL. This may be because semi-supervised learning can use the unlabeled data to improve the learning task when the labeled data are scarce. However, when there are enough training data, the inductive scheme has the potential to learn a better projection from feature space to label space using only the labeled subjects. We can see the evidence by comparing the results of IHL and HL(k-NN) on the MSRGesture3D dataset, i.e., the advantage of HL is narrowing when the number of training clips goes up, and it is surpassed by IHL by about two percent when the number of training clips goes up to 10.

The comparison results of the multi-modality methods, i.e., MHL, MHL-W, and IMHL, on the Microblog dataset and ModelNet40 dataset, are demonstrated in Table 8. In general, IMHL achieves significant accuracy improvement compared with MHL and MHL-W. On the ModelNet40 dataset, the accuracy of IMHL is 1.34 percent higher than that of MHL when all three modalities are used. The results also indicate that more modalities would not necessarily improve the performance. For example, the highest classification accuracy on the Microblog dataset is achieved by IHL using only single emojicon modality (see Table 5), which is 0.7 percent higher than using all three modalities. The noise from the text and visual modality may be a possible explanation for this finding. Although there have been some attempts to impose higher weights on effective modalities, it is still challenging for the existing multi-modality methods to suppress the negative effects of the ineffective modalities.

7.1.4 Comparison of Running Time

In this section, we study the computational efficiency of the hypergraph generation and learning methods. All the methods are run on a server with two Intel Xeon E5-2637 3.50 GHz CPUs and 192G memory. In Fig. 6, we show the running time of the comparison hypergraph generation schemes on the MSRGesture3D and Facebook Page-Page datasets. In Table 3, we give the time complexity of different hypergraph learning methods. The running time of the comparison methods on each dataset is shown in Figs. 7 and 8.

As shown in the results, we have the following observations:

- From the fastest to the slowest, the speed of the hypergraph generation methods can be ranked in the following order: ϵ-ball, k-NN, k-means, and $l1$. This can be explained by the fact that the distance-based methods only need to search the nearest neighbors or perform clustering, while the representation-based methods need to learn an optimal representation with a gradient descent algorithm. Thus, it may be a good practice to select the k-NN hypergraph generation method to achieve a good trade-off between effectiveness and efficiency.
- When the scale of the dataset is large, inductive learning methods are more computationally efficient than transductive learning methods. The testing running time of IHL and IMHL is less than 2^{-6} second in all the comparison experiments. IHL is 300,000+ faster than HL on the Facebook Page-Page dataset, which indicates that supervised learning is more practical for applications with large-scale data.
- For the transductive learning methods, the running time of the structure update methods, including HL-W, MHL-W, DHSL, and t-DHL, is larger than HL and MHL. Compared to DHSL, t-DHL is much more efficient on large-scale data. With both the classification accuracy and the running time taken into account, t-DHL may be much more competitive than HL-W and DHSL in practice.

7.1.5 Discussion About t-DHL

Efficiency and Convergence. We study the efficiency and convergence issues of tensor-based dynamic hypergraph learning by comparing to previous dynamic hypergraph structure learning methods. From Tables 5, 6, 7 and Fig. 7, we can see that t-DHL can achieve comparable, if not better, performance compared to DHSL with a much lower time cost. We go deeper and randomly select 1k, 2k, 3k, 4k, 5k, 6k, 7k and 8k samples from the Facebook Page-to-Page dataset and draw the running time growth curves of DHSL and

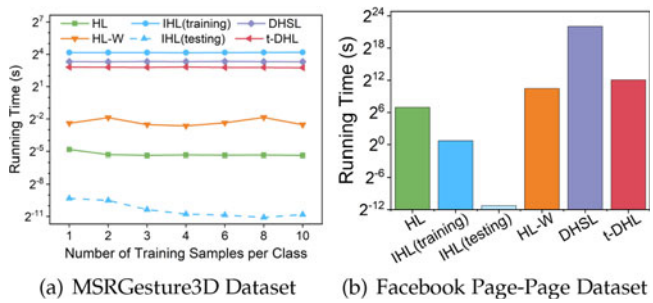


Fig. 7. Running time comparison of hypergraph learning methods.

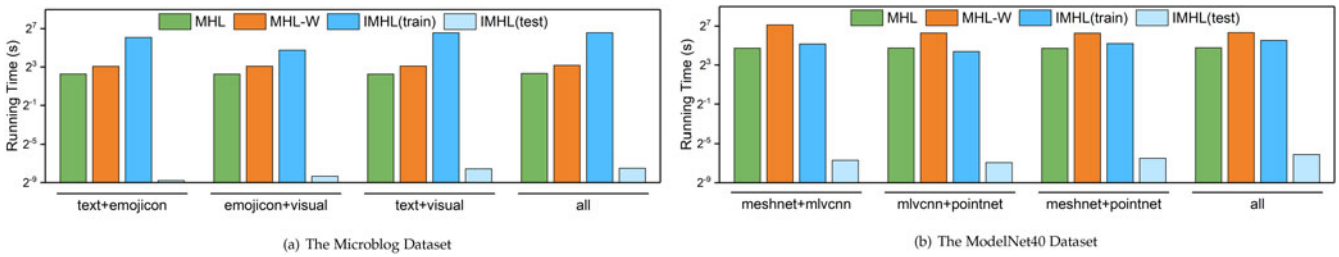


Fig. 8. Running time comparison on multi-modality datasets.

t-DHL in Fig. 9a. As the scale of the dataset becomes larger, the running time of DHSL increases dramatically, while t-DHL presents a limited running time growth. From what we can see, t-DHL is much more suitable for large-scale datasets than DHSL. We further study the convergences of DHSL and t-DHL by conducting experiments on the MSRGesture3D dataset. With the step sizes of both methods set to 0.01. we get the convergence curves of DHSL and t-DHL as shown in Fig. 9b, where we can see that t-DHL can achieve convergence within about 100 iterations while DHSL needs 1,000 iterations or even more. Thanks to the characteristic of bi-convex, our t-DHL model can reach the optimal solution much faster and more stably than DHSL.

Parameters. There are three parameters, i.e., λ, α, and μ, in the objective function of t-DHL. α is used to balance the impacts of label information and feature information towards the hypergraph structure. μ and λ are used to weigh the empirical losses of $\mathcal{T}$ and $\mathbf{F}$, respectively. Different from λ, which is a parameter that has been widely explored in many existing methods [35], [36], [38], α and μ are the parameters used for the t-DHL model in particular. Therefore, we first empirically set λ to 100 and only investigate the influences of α and μ to the classification accuracy. We then keep the optimal α and μ fixed and study how the varying λ affects the results. Experimental results on the three datasets are demonstrated in Fig. 10. For μ and α, we can see that the proposed method can achieve steady performance when these two parameters vary in a large range and show relatively poor performance when μ is too small. We have an interesting observation that, on all the three datasets, the proposed method can achieve the highest performance when μ is no smaller than 10 and α is no larger than 0.1. The performance of t-DHL is stable when λ is over 50 while it drops dramatically when λ is too small, which is consistent with the previous studies [35], [36], [38]. The results indicate that the proposed t-DHL is insensitive to parameter perturbations.

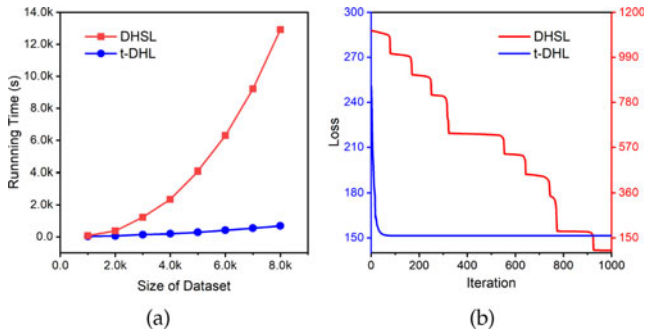


Fig. 9. (a) The growth curve of running time with the size of dataset on the Facebook Page-to-Page dataset. (b) The convergence curve of DHSL and t-DHL on the MSRGesture3D dataset when the step size is set to 0.01, respectively.

7.2 On Clustering Tasks

To investigate the effectiveness of applying different hypergraph construction schemes to clustering tasks, we conduct experiments on the MNIST digit dataset [62], and the Extended Yale B face dataset [63]. We compare four schemes: ϵ-ball hypergraph, k-means hypergraph, $l1$ hypergraph, and k-NN hypergraph. The MNIST dataset contains 70,000 handwritten digit images, from which 2,000 images (200 samples per digit) are randomly selected for the experiments. The Extended Yale B face dataset contains 38 individuals, each of which has 64 near-frontal images under different illuminations. To evaluate the clustering performance of these methods, we employ prediction accuracy (AC) and normalized mutual information (NMI) for evaluation metrics.

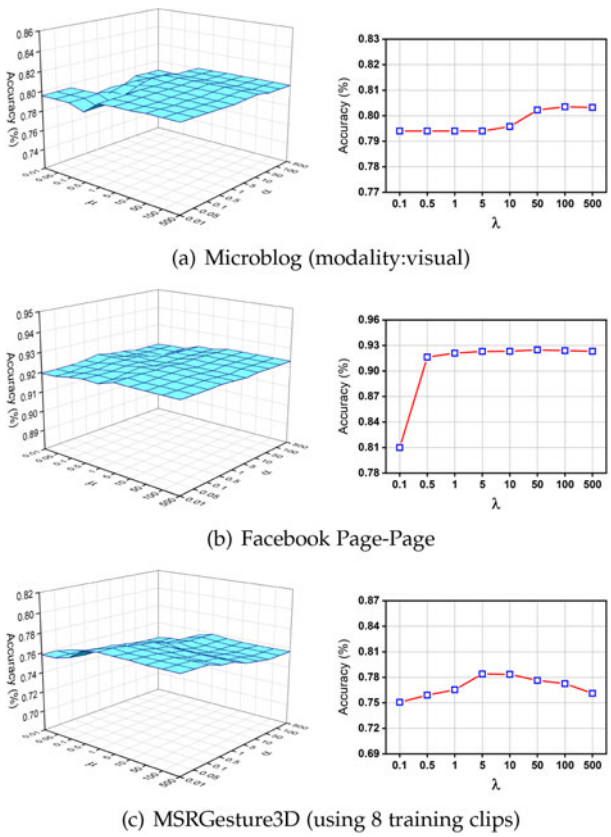


Fig. 10. Left: performances of t-DHL by varying α and μ on the three datasets; λ is set to 100. Right: performances of t-DHL by varying λ on the three datasets; α and μ are set as 0.1 and 5, respectively.

TABLE 9
Clustering Results on the MNIST and Extended Yale B Datasets

Method	MNIST		Yale B	
	AC	NMI	AC	NMI
ϵ-ball hypergraph	0.4644	0.4406	0.0457	0.0304
k-means hypergraph	0.546	0.5097	0.0838	0.1124
k-NN hypergraph	0.6237	0.6352	0.3451	0.364
$l1$ hypergraph	**0.6756**	**0.6922**	**0.392**	**0.4006**

The average clustering results of five runs on these two datasets are summarized in Table 9, from which we have the following observations:

- Representation-based $l1$ hypergraph significantly outperforms the distance-based hypergraphs (ϵ-ball, k-means, and k-NN) on both AC and NMI consistently. Specifically, compared to ϵ-ball, k-means and k-NN hypergraph, $l1$ hypergraph has increased by 45.48, 23.74 and 8.32 percent on AC, and 57.1, 35.81 and 8.97 percent on NMI, respectively, on the MNIST dataset. This indicates that the representation-based methods are more suitable for clustering tasks in practice. It may be because the representation-based hypergraph automatically learns the weights of different vertices connecting to the same hyperedge. The weights of the noise data are lowered, although this hyperedge may connect vertices from different classes. This makes the representation-based hypergraph ($l1$ hypergraph) more robust to the noise and more distinguishable in the clustering task.
- On the Extended Yale B Face dataset, the AC and NMI of the ϵ-ball and k-means hypergraphs are much lower than the other two hypergraphs. The poor performance of the ϵ-ball hypergraph may be due to its sensitivity to the parameter value of ϵ that decides the searching range of nearest neighbors. If ϵ is too small, each hyperedge may connect only one or two vertices. If ϵ is too large, the hyperedge may connect the vertices from different classes. The value of ϵ which enables effective learning is limited to a small range, and therefore makes the learning failure unexpected in some cases.

8 Learning Toolbox

8.1 Introduction to THU-HyperG

Although hypergraph learning has shown its superiority in modeling complex data and has been widely applied to various tasks recently, there is no available hypergraph learning toolbox so far, which significantly limits its application. To this end, we develop an open-source Python library called THU-HyperG for hypergraph learning. This toolbox has been released under the MIT license and is available on Github.[1]

THU-HyperG provides the implementations of various hypergraph learning methods, and it allows users to apply hypergraph learning to the tasks of classification and clustering. To facilitate the development of hypergraph learning based applications, we also provide example codes about how to apply THU-HyperG on the classification and clustering tasks. With the help of THU-HyperG, we believe hypergraph learning could be more easily applied to various applications. In addition, we have also decomposed the hypergraph learning algorithms into different modules in this toolbox. The structure of THU-HyperG is shown in Fig. 11. Researchers can easily implement their methods with this toolbox.

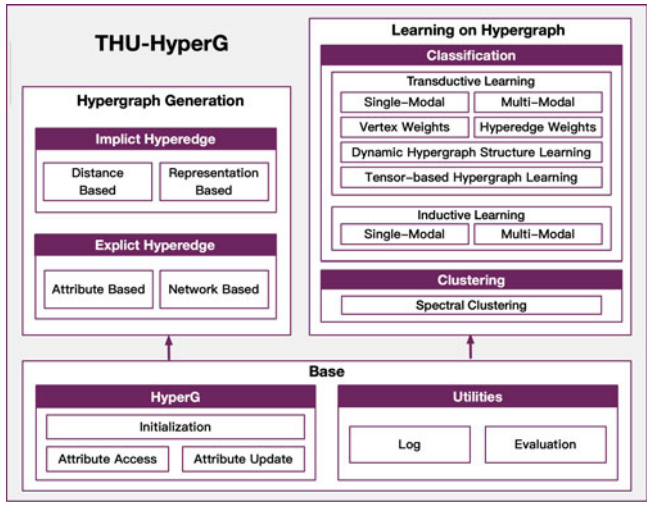


Fig. 11. Structure of THU-HyperG.

8.2 Implementation of THU-HyperG

THU-HyperG is implemented by Python and built on the scientific computing library Scipy [64] and NumPy [65], convex optimization tool CVXPY [66], and machine learning toolbox scikit-learn [67]. To reduce the computation cost, it is enhanced with the sparse matrix techniques.

Since the pipeline of applying hypergraph learning algorithms can be divided to hypergraph generation and learning, THU-HyperG is accordingly decoupled into two parts. For hypergraph generation, THU-HyperG supports various hypergraph generation algorithms, including distance-based, representation-based, attribute-based and network-based methods. For hypergraph learning, methods including transductive hypergraph learning, inductive hypergraph learning, cross-diffusion on hypergraph, tensor-based hypergraph learning, and the hypergraph structure update methods have been implemented in THU-HyperG.

9 Conclusion

In this paper, we systematically review recent progresses on hypergraph learning, with a focus on hypergraph generation, learning models, and their applications. We have the following interesting findings.

1) If the correlation is latent and ambiguous, hypergraphs can be implicitly generated from the observed data using distance-based methods and representation-based methods. We can also construct explicit hypergraphs using inherent attributes or network information from the data.

1. https://github.com/iMoonLab/THU-HyperG

2) The distance-based methods are more efficient in terms of running time and more suitable for online scenarios while the representation-based methods consume more time but are more robust to noisy data.
3) Each hypergraph learning method has its own strengths and weaknesses. Different methods may perform differently over different datasets. For example, the inductive hypergraph learning methods meet the requirements of online tests, while the transductive approaches do not apply.
4) Dynamic hypergraph learning may open up new prospects since the weight of importance can be defined not only on the hyperedges but also on vertices and incidence matrices.

Previous dynamic hypergraph structure learning methods have the problems of being non-convex, under-optimized and a high computational cost in real applications. To address these limitations, we have also proposed a tensor-based dynamic hypergraph learning method called t-DHL which can achieve a good balance between effectiveness and efficiency. The proposed t-DHL method uses a tensor representation of dynamic hypergraphs to model high-order correlations among data. Based on this tensor representation, we propose an efficient algorithm for learning on the dynamic hypergraph. It allows the hypergraph structure to be dynamically updated during the learning process by leveraging the data distribution from both the label space and the feature space. Experimental results demonstrate that the proposed method has the potential to achieve overall superior performance over traditional hypergraph learning methods and state-of-the-art approaches in most cases. More importantly, the proposed t-DHL is bi-convex, which converges fast and can be scalable to large-scale datasets. In addition, to facilitate the research of hypergraph learning, we have released THU-HyperG, which is a comprehensive toolbox for hypergraph learning methods and practices.

Hypergraph learning is generally a new field and there are still many open problems and directions that have not been explored. One interesting problem is how to build a more systematic and general mechanism for hypergraph generation. Since there are several hypergraph generation methods, given a new dataset, it is not easy to know which hypergraph generation method is the best. It may be a potential direction to figure out how we can efficiently find a hypergraph generation method suiting the data best. Another issue is about analyzing the semantic information of hyperedges. In previous works, the vertex features and structure information have been used for hypergraph learning, but the hyperedge information is often ignored. The types of hyperedges may be different in some cases like heterogeneous networks, thus raising an interesting problem about the meaning and interpretation of hyperedges during the learning process. The last important problem is about modeling and analyzing the high-order correlation on ultra-large-scale datasets. We have seen that time complexity is a crucial issue for graph/hypergraph-based methods. Therefore, studying the acceleration mechanism is crucial for applying hypergraph learning to ultra-large-scale networks, such as social networks, brain networks, etc.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2017YFC0113000).

References

[1] O. Sporns, "The human connectome: A complex network," *Ann. New York Acad. Sci.*, vol. 1224, no. 1, pp. 109–125, 2011.
[2] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2004, pp. 321–328.
[3] Q. Fang, J. Sang, C. Xu, and Y. Rui, "Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning," *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 796–812, Apr. 2014.
[4] D. Li, Z. Xu, S. Li, and X. Sun, "Link prediction in social networks based on hypergraph," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 41–42.
[5] L.-E. Martinet *et al.*, "Robust dynamic community detection with applications to human brain functional networks," *Nat. Commun.*, vol. 11, no. 1, pp. 1–13, 2020.
[6] Y. Huang, Q. Liu, and D. Metaxas, "Video object segmentation by hypergraph cut," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1738–1745.
[7] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai, "3-D object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4290–4303, Sep. 2012.
[8] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3376–3383.
[9] W. Zhao *et al.*, "Learning to map social network users by unified manifold alignment on hypergraph," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5834–5846, Dec. 2018.
[10] F. Luo, B. Du, L. Zhang, L. Zhang, and D. Tao, "Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2406–2419, Jul. 2019.
[11] L. Zhu, J. Shen, H. Jin, R. Zheng, and L. Xie, "Content-based visual landmark search via multimodal hypergraph learning," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2756–2769, Dec. 2015.
[12] D. Du, H. Qi, L. Wen, Q. Tian, Q. Huang, and S. Lyu, "Geometric hypergraph learning for visual tracking," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4182–4195, Dec. 2017.
[13] Z. Tian, T. Hwang, and R. Kuang, "A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge," *Bioinformatics*, vol. 25, no. 21, pp. 2831–2838, 2009.
[14] X. Zheng, W. Zhu, C. Tang, and M. Wang, "Gene selection for microarray data classification via adaptive hypergraph embedded dictionary learning," *Gene*, vol. 706, pp. 188–200, 2019.
[15] Y. Gao *et al.*, "MCI identification by joint learning on multiple MRI data," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Interv.*, 2015, pp. 78–85.
[16] W. Shao, Y. Peng, C. Zu, M. Wang, and D. Zhang, "Hypergraph based multi-task feature selection for multimodal classification of Alzheimer's disease," *Computerized Med. Imag. Graph.*, vol. 80, 2020, Art. no. 101663.
[17] E. Ramadan, S. Perincheri, and D. Tuck, "A hyper-graph approach for analyzing transcriptional networks in breast cancer," in *Proc. ACM Int. Conf. Bioinf. Comput. Biol.*, 2010, pp. 556–562.
[18] L. Xiao *et al.*, "Multi-hypergraph learning based brain functional connectivity analysis in fMRI data," *IEEE Trans. Med. Imag.*, vol. 39, no. 5, pp. 1746–1758, May 2020.
[19] Q. Fang, J. Sang, C. Xu, and Y. Rui, "Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning," *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 796–812, Apr. 2014.
[20] G. Xu *et al.*, "Precise point set registration using point-to-plane distance and correntropy for LiDAR based localization," in *Proc. IEEE Intell. Vehicles Symp.*, 2018, pp. 734–739.
[21] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in LBSNs: A hypergraph embedding approach," in *Proc. World Wide Web Conf.*, 2019, pp. 2147–2157.

[22] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1601–1608.
[23] M. Wang, X. Liu, and X. Wu, "Visual classification by ℓ_1-hypergraph modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2564–2574, Sep. 2015.
[24] Q. Liu, Y. Sun, C. Wang, T. Liu, and D. Tao, "Elastic net hypergraph learning for image clustering and semi-supervised classification," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 452–463, Jan. 2017.
[25] T. Jin, Z. Yu, Y. Gao, S. Gao, X. Sun, and C. Li, "Robust ℓ_2- hypergraph and its applications," *Inf. Sci.*, vol. 501, pp. 708–723, 2019.
[26] S. Huang, M. Elhoseiny, A. Elgammal, and D. Yang, "Learning hypergraph-regularized attribute predictors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 409–417.
[27] S. Aksoy, D. Arendt, L. S. Jenkins, B. Praggastis, E. Purvine, and M. Zalewski, "High performance hypergraph analytics of domain name system relationships," in *Proc. HICSS Symp. Cybersecur. Big Data Analytics*, 2019.
[28] C. Zu *et al.*, "Identifying high order brain connectome biomarkers via learning on hypergraph," in *Proc. Int. Workshop Mach. Learn. Med. Imag.*, 2016, pp. 1–9.
[29] F. Amato, G. Cozzolino, and G. Sperli, "A hypergraph data model for expert-finding in multimedia social networks," *Information*, vol. 10, 2019, Art. no. 183.
[30] N. Franzese, A. Groce, T. M. Murali, and A. Ritz, "Hypergraph-based connectivity measures for signaling pathway topologies," *PLOS Comput. Biol.*, vol. 15, no. 10, pp. 1–26, 2019.
[31] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLOS Comput. Biol.*, vol. 5, no. 5, pp. 1–6, 2009.
[32] R. Ji, Y. Gao, R. Hong, Q. Liu, D. Tao, and X. Li, "Spectral-spatial constraint hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1811–1824, Mar. 2014.
[33] Y. Fang and Y. Zheng, "Metric learning based on attribute hypergraph," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3440–3444.
[34] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, "Dual channel hypergraph collaborative filtering," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2020–2029.
[35] Z. Zhang, H. Lin, X. Zhao, R. Ji, and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3D object classification," *IEEE Trans. Image Process.*, vol. 27, no. 12, pp. 5957–5968, Dec. 2018.
[36] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, and X. Wu, "Visual-textual joint relevance learning for tag-based social image search," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 363–376, Jan. 2013.
[37] L. Su, Y. Gao, X. Zhao, H. Wan, M. Gu, and J. Sun, "Vertex-weighted hypergraph learning for multi-view object classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 2779–2785.
[38] Z. Zhang, H. Lin, and Y. Gao, "Dynamic hypergraph structure learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3162–3169.
[39] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Math. Methods Operations Res.*, vol. 66, no. 3, pp. 373–407, 2007.
[40] E. G. Birgin, J. M. Martínez, and M. Raydan, "Nonmonotone spectral projected gradient methods on convex sets," *SIAM J. Optim.*, vol. 10, no. 4, pp. 1196–1211, 2000.
[41] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3558–3565.
[42] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2635–2641.
[43] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016.
[44] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 5708–5717.
[45] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2434–2444.
[46] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
[47] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "MeshNet: Mesh neural network for 3D shape representation," in *Proc. AAAI Conf. Artif. Intell.*, 2018.
[48] J. Jiang, D. Bao, Z. Chen, X. Zhao, and Y. Gao, "MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 8513–8520.
[49] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 264–272.
[50] A. Weichselbraun, S. Gindl, and A. Scharl, "Enriching semantic knowledge bases for opinion mining in big data applications," *Knowl.-Based Syst.*, vol. 69, pp. 78–85, 2014.
[51] F. Chen, Y. Gao, D. Cao, and R. Ji, "Multimodal hypergraph learning for microblog sentiment prediction," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2015, pp. 1–6.
[52] O. Oreifej and Z. Liu, "HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 716–723.
[53] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[54] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang, "Diversity-induced multi-view subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 586–594.
[55] Z. Lun *et al.*, "Learning to group discrete graphical patterns," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 225.
[56] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
[57] R. Ji, F. Chen, L. Cao, and Y. Gao, "Cross-modality microblog sentiment prediction via Bi-layer multimodal hypergraph learning," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 1062–1075, Apr. 2019.
[58] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," 2019.
[59] A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," in *Proc. Eur. Signal Process. Conf.*, 2012, pp. 1975–1979.
[60] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[61] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu, "Robust 3D action recognition with random occupancy patterns," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 872–885.
[62] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[63] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
[64] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nat. Methods*, vol. 17, pp. 261–272, 2020.
[65] T. E. Oliphant, *A Guide to NumPy*, vol. 1. USA: Trelgol Publishing, 2006.
[66] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.
[67] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

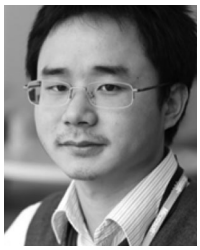
Yue Gao (Senior Member, IEEE) received the BS degree from the Harbin Institute of Technology, Harbin, China, and the ME and PhD degrees from Tsinghua University, Beijing, China. He is currently an associate professor at the School of Software, Tsinghua University.

Zizhao Zhang received the BE degree in software engineering from Tsinghua University, Beijing, China, in 2018. She is currently working toward the PhD degree from the School of Software, Tsinghua University, Beijing, China.

Haojie Lin received the BE degree in software engineering from the South China University of Technology, Guangzhou, China, in 2018. He is currently working toward the master's degree from the School of Software, Tsinghua University, Beijing, China.

Xibin Zhao received the BS, ME, and PhD degrees from the School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, China, in 1994, 2000, and 2004 respectively. He is currently an associate professor at the School of Software, Tsinghua University. His research interests include reliability analysis of hybrid network systems and information system security.

Shaoyi Du (Member, IEEE) received the double bachelor's degrees in computational mathematics and computer science, in 2002, the MS degree in applied mathematics, in 2005, and the PhD degree in pattern recognition and intelligence system from Xi'an Jiaotong University, Xi'an, China, in 2009. He is currently a professor at Xi'an Jiaotong University. His research interests include computer vision, machine learning, and pattern recognition.



Changqing Zou (Member, IEEE) received the BE degree from the Harbin Institute of Technology, Harbin, China, the ME degree from the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, China, and the PhD degree from the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China. His research interests include computer vision, computer graphics, and machine learning.