Artificial Intelligence: Foundations, Theory, and Algorithms

Qionghai Dai
Yue Gao

Hypergraph Computation

Springer

Artificial Intelligence: Foundations, Theory, and Algorithms

Series Editors

Barry O'Sullivan, Dep. of Computer Science, University College Cork, Cork, Ireland

Michael Wooldridge, Department of Computer Science, University of Oxford, Oxford, UK

Qionghai Dai • Yue Gao

Hypergraph Computation

Springer

Qionghai Dai
Department of Automation
Tsinghua University
Beijing, China

Yue Gao
School of Software
Tsinghua University
Beijing, China

This work was supported by Tsinghua University

ISSN 2365-3051 ISSN 2365-306X (electronic)
Artificial Intelligence: Foundations, Theory, and Algorithms
ISBN 978-981-99-0184-5 ISBN 978-981-99-0185-2 (eBook)
https://doi.org/10.1007/978-981-99-0185-2

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

Artificial Intelligence is now everywhere and fuels both industry and daily life all over the world. We are in the era of “big data,” and huge sums of information can be obtained which are too cumbersome for people to process themselves. These big data are even with much complex correlations behind them in various areas, such as computer vision and social media. For example, the complex correlations among pixels in an image reveal its semantic information, and different types of correlations among social posts infer the users’ emotions. Therefore, developing effective AI methods to exploit such complex data correlations has become an urgent but challenging task.

Graph has been widely used to formulate data correlations. A graph is a non-linear data structure which is composed of groups of vertices and edges, representing the pairwise correlations among vertices. Graph learning and graph neural networks have attracted much attention in both research and industrial fields and become very hot topics in these years. It is noted that the world is far more complex than just pairwise connections, and thus graph-based methods still have limitations on high-order correlation modeling.

Hypergraph, as a generation of graph, is able to formulate such high-order correlations among the data and has been investigated in last decades. Recent years have witnessed a great popularity of research on hypergraph-related AI methods, which have been used in computer vision, social media analysis, and etc. We noticed that there still has not been a theoretical book to systematically introduce the recent achievements in this field and then started preparation of this book. We summarize these attempts as a new computing paradigm, called hypergraph computation, which is to formulate the high-order correlations underneath the data using hypergraph, and then conduct semantic computing on the hypergraph for different applications.

In this book, we introduce recent progress in hypergraph computation, from hypergraph modeling to hypergraph neural networks. The applications of hypergraph computation are also discussed. We also summarize the recent achievements and useful tools in hypergraph computation. This book can be regarded as both a theoretical book and a manual on how to use hypergraph computation in practice.

Book Organization

This book includes 13 chapters with 3 parts. The first part introduces the fundamental knowledge of hypergraph computation. In this part, Chap. 1 depicts the basic knowledge, applications, and history of hypergraph. The mathematical foundations of hypergraph are introduced in Chap. 2. Three general paradigms of hypergraph computation are provided in Chap. 3.

The second part focuses on hypergraph modeling and learning techniques. The first step of hypergraph computation is to construct a hypergraph to formulate the high-order correlations among data, which is provided in Chap. 4. Typical hypergraph computation tasks are then provided in Chap. 5, including label propagation, data clustering, cost-sensitive learning, and link prediction. We further introduce the hypergraph structure evolution methods for hypergraph optimization in Chap. 6. The neural networks on hypergraph are introduced in Chap. 7. The practical applications of hypergraph computation require the capability of handling large-scale data. Therefore, we give an extensive introduction to large-scale hypergraph computation in Chap. 8.

The third part introduces the applications of hypergraph computation in several fields, including social media analysis in Chap. 9, medical and biological applications in Chap. 10, and computer vision in Chap. 11. This part also introduces the DeepHypergraph library, a hypergraph computation library based on Python, in Chap. 12, and the future advancement of hypergraph computation research in Chap. 13.

Prerequisites

This book is designed for advanced undergraduate and graduate students, postdoctoral researchers, lecturers, researchers, and industrial engineers, as well as anyone interested in AI, especially hypergraph computation. The readers are expected to have basic knowledge in probability, linear algebra, and machine learning. Graph theory could be a good prior before reading this book, but not mandatory. Besides the theoretical part from Section 3 to Section 8, we have also provided a series of applications from Section 9 to Section 11, which can be used as guidelines for the deployment of hypergraph computation in practice.

Contact Information

We welcome any feedback, corrections, and suggestions on the book, which may be sent to *gaoyue@tsinghua.edu.cn*. The readers can also find updates about the book from the personal homepage at www.gaoyue.org.

Beijing, China Qionghai Dai
December 2022 Yue Gao

Acknowledgments

The authors would like to acknowledge the support and contributions of research collaborators, who have provided insightful comments and suggestions. For the whole book, we thank Yifan Feng, Shuyi Ji, Yutong Jiang, Qingmei Tang, Jielong Yan, Xinwei Zhang, Yubo Zhang, Zizhao Zhang, and Hao Zhong for the preparation of initial drafts, and thank Jialiang Cheng, Yue Dai, Lou Fang, Jiashu Han, Jiangang Huang, Kejie Huang, Tao Jin, Renjie Li, Zhi Li, Jun Ma, Bohua Wang, Yuehang Wang Xu Wu, Chengwu Yang, Yifan Zhang, and Zhikuan Zhou for proofreading and corrections.

We started the plan for this book in 2018 and finished the preparation in 2022. We sincerely thank the Springer Senior Editor, Dr. Celine Lanlan Chang, for providing insightful comments and suggestions during the preparation of this book. We are also grateful to Springer's production editor, Jayesh Kalleri, for offering invaluable help during the preparation of this manuscript.

Also special thanks to Dr. Shihui Ying, for patiently discussing the book framework and reviewing early versions of this manuscript and high-quality suggestions which have significantly improved this book.

We also give our appreciations to our organizations, Department of Automation at Tsinghua University, School of Software at Tsinghua University, Institute for Brain and Cognitive Sciences at Tsinghua University (THUIBCS), and the Broadband Network and Digital Multimedia Lab at Tsinghua University, who have provided outstanding supports and facilities for preparing this book.

Finally, and most importantly, a very heartfelt thank you to our families, for their constant support, encouragement, patience, and understanding during the whole journey.

This book is supported by the Natural Science Foundation of China (62088102 and U1701262).

Contents

Acronyms

AD	Alzheimer's disease
AHGAE	Adaptive hypergraph auto-encoder
ASD	Autistic spectrum disorder
BCR	Bayesian concordance readjust
b-HGFN	Big-hypergraph factorization neural network
BIC	Bayesian information criteria
CF	Collaborative filtering
CHL	Centralized hypergraph learning
CT	Computed tomography
DGCNN	Dynamic graph CNN
DHCF	Dual-channel hypergraph collaborative filtering
DHG	Deep hypergraph
DHGNN	Dynamic hypergraph neural networks
DTI	Drug-target interaction
FC	Functional connectivity
GCNs	Graph convolutional networks
GVCNN	Group-view convolutional neural networks
HeteHG-VAE	Hypergraph variational autoencoder
HGNN	Hypergraph neural networks
HGNN+	General hypergraph neural networks
HHDTI	Heterogeneous hypergraph learning method for the DTI prediction
HHGNN	Hyperbolic hypergraph neural network
HHPL	Hierarchical hypergraph patch labeling
HINGE	Hyper-relational knowledge graph embedding
Hyper-Atten	Hypergraph convolution with attention
Hyper-SAGNN	Self-attention-based hypergraph neural network
IGL	Incremental graph learning
iMHL	Inductive multi-hypergraph learning
JHyConv	Jump hypergraph convolution
MAS	Multi-atlas segmentation

MC	Microblog clique
MCI	Mild cognitive impairment
MHG	Multi-modal hypergraph learning
MHGNN	Multi-hypergraph neural networks
MRI	Magnetic resonance imaging
m-TransH	Multi-fold TransH
MVCNN	Multi-view convolutional neural networks
MVHL	Multi-modal vertex-weighted hypergraph learning
NaLP	N-ary link prediction
NC	Normal control
PER	Personalized emotion recognition
TAP	Tandem affinity purification
WSIs	Whole-slide histopathological images

Chapter 1
Introduction

Abstract High-order correlations among data exist widely in various practical applications. Compared with the simple graph which can only model the pairwise relationship between two subjects, hypergraph is a flexible and representative model to formulate high-order correlations. Based on the hypergraph model, there have been many efforts to design the computation framework and analyze the high-order correlations. In this chapter, we briefly introduce the hypergraph computation, including its background, definition, history, recent challenges, and objectives.

1.1 Background

The basic elements of many natural and artificial systems have dependencies on each other and call for correlation modeling and analytic methods to study these. The graphs are all around us from different perspective, and in general all the objects in the real world are defined based on their connections with other objects. These connections can be described as a graph, which is a common data structure in many cases. For example, graphs can depict the path in a city, where each path is represented with an edge to show the spatial connections between two locations. Graphs are also employed in the airline route map, in which each vertex is an airport and each edge is an airline.

Recently, the most challenging data processing problem comes from the connected data, not just from the discrete ones. How to exploit the underneath connections behind the data has become an urgent and important task in many applications. Generally, graph has been used to formulate such correlations among data. A graph is a nonlinear data structure which is composed of a group of vertices and edges. Here, the vertices in a graph represent the subjects to be analyzed, and the edges in a graph are the lines connecting two vertices in the graph. Figure 1.1 shows an example of a graph.

As a common way to model pairwise correlations among data, the components in a system can be represented by the vertices of a graph, and the associations between components are described by the edges. In this way, the association pattern is abstracted by the topological structure of the graph. In the past decades,

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_1

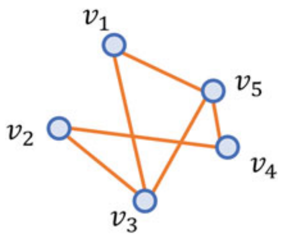


Fig. 1.1 An example of a graph

it was not easy to apply graph theory in practice because of the limitation of computing power. In recent years, with the advancement of information technology and computing power, graph theory has demonstrated its practical values. As scales of data grow, scientists have come up with the concept of network science. The study of network science can be applied in various fields. For example, by studying the connection relationship between terminals on the Internet, the efficiency of data transmission in a network can be estimated. The study of interpersonal relationships can help understand the way people communicate with each other, disseminate information, and generate community. Studying the transmission chain of infectious diseases can help predict risks in time, thus interrupt transmission, and prevent their spread. People have also found that many biological, social, information, and other real networks have nontrivial structural patterns in the connections among their elements. These patterns reflect meaningful features of the whole network. For example, the small-world phenomenon (the average path length in the network does not increase significantly with the increase of the network size) widely exists in social networks [1]. Another example is scale-free network [2], in which the vertex degree distribution follows a power-law distribution, and this phenomenon is known in some biological metabolic networks [3].

It is noted that the world is far more complex than just pairwise connections. Typical examples include social networks, protein–protein interaction networks, and brain networks. In social networks, the individual characteristics of users are related to the interactive patterns among users. The users with similar characteristics are more likely to connect with each other to form a social group. The social relationships of users also affect their profiling portraits. We notice that the correlations among these uses are not just pairwise connections but also group-like connections, which are more complex than these pairwise connections. Figure 1.2 shows an example of social connections, in which each user could have different types of connections with two or more other users or items.

In human brain networks, the cerebral cortex contains more than 10^{11} neurons and a cluster of neurons with similar functions and connections forms a nucleus. The nuclei can be further divided into different brain regions, resulting in a multilevel and multi-scale complex brain network. For example, the whole brain map includes Insula and Cingulate Gyri, Frontal Lobe, Occipital Lobe, Parietal Lobe, and other regions, which can be further divided into 90 brain regions that are provided in AAL atlas [4], such as Hippocampus and Parahippocampus. Each neuron can have more than 10,000 synapses, which can connect the neurons in the brain to other neurons in

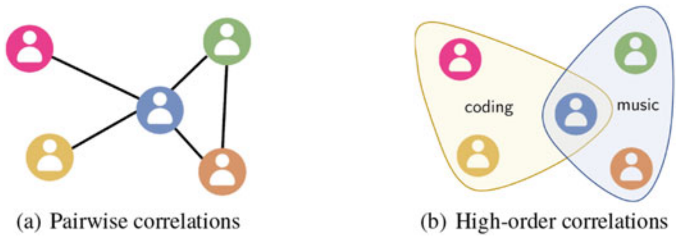


Fig. 1.2 An illustration of pairwise correlations and high-order correlations between/among users

the rest of the body or connect the neurons to the muscles. The connections among the neurons are complex and hard to be formulated in a graph, although graph is a typical way to model such correlations in the brain.

Such complex correlations, i.e., the high-order correlation rather than pairwise ones, are very common in real-world data. To study these complex systems, it is necessary to characterize and analyze high-order relationships between their elements. Empirical studies have shown that the correlation patterns of a system often play an important role in functions of the system. In recent years, more researchers have begun to pay attention to this field and apply high-order correlation modeling and analytic methods.

At the beginning of the development of machine learning on graph and network science, only graph has been used to model the network or the correlations, and the associations between the elements of the system were generally described by the topological structure of the graph. As a result, the pairwise connections can be described in the graph, while a large amount of semantic information in the system could be lost, and descriptive features in the network could not be extracted. Some well-discussed network properties, such as degree centrality, semi-locality centrality, and closeness centrality, were all based on such a static single network model. The underneath high-order information behind the data has to be degenerated to pairwise ones for processing, which may lead to serious information loss. With the development of big data, the explosive growth of data demonstrates their complexity and diversity, which calls for more complex data modeling methods. The network modeling methods for complex data types, complex topological structures, and complex connection patterns emerge. For example, the social closeness between individuals in a social network can be strong or weak, and a system with weight distribution for the association between vertices can be modeled using a weighted network [5]. Also, the power network and the communication network are inter-dependent in infrastructure construction. The vertices of the communication network provide control signals to the vertices of the power network, whereas the vertices of the power network supply power to the vertices of the communication network. The interdependence between different networks can be modeled using an inter-dependent graph [6]. Another example is the air transportation network, where the routes between the vertices may belong to different airlines. For the heterogeneity of object types and association relationships, the concept of multi-layer network or graph has been proposed [7]. The last example is that the ecological food chain in the species network changes with the change of seasonal environmental conditions. For dynamic systems, the concept of temporal network has been introduced [8] to formulate the correlation among the subjects.

Although graph-based methods have been developed for decades and great progresses have been achieved, they still have limitations. These graph models can better formulate the binary relationships between the elements in the system, while they may ignore the high-order correlations among three or more elements. In recent years, many studies have shown that modeling and optimizing high-order correlations are even more important in most of the applications [9–11]. For example, in the biosphere system, the high-order interactions between species ensure stable diversity of species [10]. The high-order characteristics of different networks can effectively distinguish their fields [11]. With the rapid development of network science, the complexity of data and correlation increase rapidly. In the fields of biological information, social computing, and image processing, there are a large number of multi-modal, heterogeneous, high-level data, and there are needs for effective high-order correlation modeling and optimization methods.

As the subject of interdisciplinary study in many different fields including computer science, physics, and biology, high-order correlation modeling and optimization have attracted much attention in recent decades. There are a large number of high-order relationships in many systems in the real world [12]. For example, in social networks, people form groups of three or more to communicate, and in academic networks, multiple authors cooperate to write an article. Protein interactions in biological networks may occur between multiple proteins, and gene expression is driven by high-order interactions between biomolecules [13]. High-order associations among elements are difficult to be described by the topology of simple graphs. Under such circumstances, the corresponding mathematical expressions have been introduced, such as set systems [14], simplicial complexes, and hypergraphs [15]. However, how to deploy the mathematical expressions in computation paradigm is still an open problem. The complexity of high-order correlations is much higher than that of pairwise correlations, which brings about new challenges to computation paradigms.

Hypergraph, as a generation of graph, which is able to formulate high-order correlations among the data, has been investigated recently. In this book, we introduce recent progress on hypergraph computation, from hypergraph modeling to hypergraph neural networks. Below we first introduce the basic definitions of hypergraph and then show the applications and research history of hypergraph. Finally, we provide the summary of our works in hypergraph computation and the structure of this book.

1.2 The Definition of Hypergraph

The hypergraph is an important concept in discrete mathematics, which is a generalization of the graph. Therefore, many concepts of hypergraphs can be defined related to the well-known definition of graphs. A *hypergraph* is defined as a pair of hypervertex set and hyperedge set. The *hypervertex set*, also called the *vertex set*, is a finite set, whereas the *hyperedge* represents the subset of the vertex set. As the hyperedge can connect any number of vertices, more general types of relationships could be modeled by hypergraphs rather than graphs. The order and the size of the hypergraph can be defined based on the vertex set and hyperedge set, i.e., the *order of the hypergraph* represents the cardinality of the vertex set, and the *size of the hypergraph* denotes the cardinality of the hyperedge set.

Similar to graphs, two specific types of hypergraphs can be defined, including the empty hypergraph and the trivial hypergraph.

- *The empty hypergraph* is the hypergraph with empty vertex set and empty hyperedge set.
- *The trivial hypergraph* is the hypergraph with nonempty vertex set and empty hyperedge set.

Generally speaking, unless stated otherwise, hypergraphs have a nonempty vertex set and nonempty hyperedge set and do not contain empty hyperedges.

The *isolated vertex* denotes the vertex which is not contained in any of the hyperedges. Two vertices are *adjacent* if there exists a hyperedge containing both of these two vertices. Two hyperedges are *incident* if they have a nonempty intersection.

The sub-hypergraph and partial hypergraph can be defined as follows:

- *An induced sub-hypergraph* of given hypergraph is the hypergraph whose vertex set is the subset of the given hypergraph, and the hyperedges have only one element or the intersection of the vertex set no less than two.
- *A sub-hypergraph* of the given hypergraph is the hypergraph whose both the vertex set and the hyperedge set are the subset of that of the given hypergraph.
- *A partial hypergraph* is a hypergraph whose hyperedge set is the subset of the given hypergraph.

Two special types of the hypergraph can be defined based on the degree:

- *A regular hypergraph* is the hypergraph in which all of the vertices have the same degree.
- *A uniform hypergraph* is the hypergraph in which all of the hyperedges have the same degree.

The concept of connectivity is defined as follows. The *loop* denotes the hyperedge with only one element. The *path* is a vertex–hyperedge alternative sequence, where the vertex belongs to the consecutive hyperedge in the sequence. The *cycle* is a path whose first vertex is the same as the last vertex. The *length* of a path is the

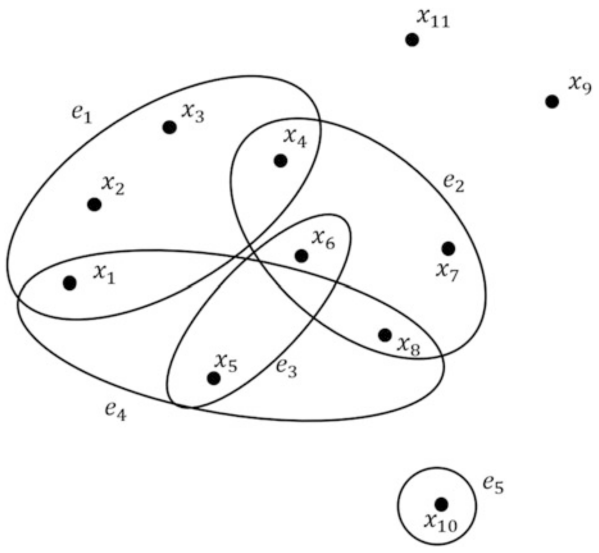


Fig. 1.3 An example of a hypergraph

number of vertices in the path. A path *connects* two vertices if these two vertices are in the path. A hypergraph is *connected* if any pair of vertices is connected, otherwise it is *disconnected*. The *distance* between two vertices is the minimum length of the path connecting these two vertices. The *diameter* of the hypergraph is the maximum distance among all pairs of vertices.

Here, we provide an example of a hypergraph in Fig. 1.3. In this hypergraph, there are 11 vertices and 5 hyperedges. In this hypergraph, the hyperedge e_1 connects vertices x_1, x_2, x_3, and x_4. The hyperedge e_2 connects x_4, x_6, x_7, and x_8. The hyperedge e_3 connects x_5 and x_6. The hyperedge e_4 connects x_1, x_5, and x_8. The hyperedge e_5 is a loop, which only connects vertex x_{10} itself. Vertices x_9 and x_{11} are two isolated vertices. The hypergraph is disconnected since x_{11} is not connected with any other vertex. $x_3 \to e_1 \to x_1 \to e_3 \to x_8 \to e_2 \to x_7$ is a path from x_3 to x_7, with length 4. The distance between x_4 and x_5 is 3 since the shortest path from x_4 to x_5 is $x_4 \to e_2 \to x_8 \to e_4 \to x_5$.

Besides Fig. 1.3, there are also other typical illustrations of hypergraph, which are shown in Fig. 1.4. In Fig. 1.4a, each circular represents a hyperedge. In Fig. 1.4b, all the lines with the same color represent a hyperedge, which connect the vertices in the hyperedge. In Fig. 1.4c, each hollow circle indicates a hypergraph and the lines with the same color link the vertices in the hyperedge.

It is noted that the hypergraph-type structures may be not explicit in many applications and they are hidden behind the data which can be observed directly. In some cases, we may only capture some pairwise correlations among the data, while the high-order correlation is needed to the regenerated based on these observations. For example, some popular citation networks, such as Cora, Citeseer, and PubMed [16], are widely used for analysis, while all these datasets only contain graph-type data, which treat the articles as vertices and the citation relationships as

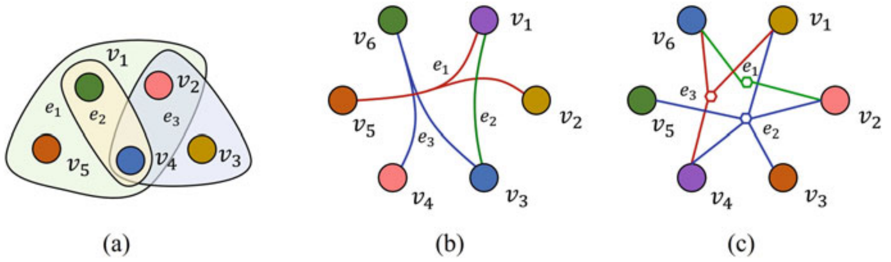


Fig. 1.4 Three typical hypergraph illustrations

links. Under such circumstances, to exploit the high-order correlation among these data, we need to transform these data to a hypergraph. As a typical method, a co-authorship hypergraph can be generated, which formulates the articles as vertices, and articles with the same authors are connected by a hyperedge. In a similar way, a co-citation hypergraph can be generated, which treats the articles as vertices as well, and the articles with the same citation are treated as a hyperedge.

1.3 Applications of Hypergraph

Hypergraph has been applied across several disciplines, including biology, economics, and sociology, due to its superiority in complex correlations modeling, which has promoted intelligent applications. In this part, we introduce several typical applications of hypergraphs to help understand this powerful tool.

One representative application is social computing. The social media data have been increasing rapidly over the past couple of decades, which can provide potential population-level insights. The hypergraph [17] is a useful tool for discovering the complex and hidden correlations from the data, in which the hypergraph structure can be used to formulate the high-order correlation in social networks.

In recommender system, the hypergraph is used to model the user–item network, to profile the user, and to further predict the preferences (future interactions). Given the raw user–item network without other information than the historical interactions between users and items, hypergraph [17] can be used to discriminatively formulate the high-order connectivities among users and items separately and conduct the collaborative filtering task. Sometimes the users and the items may be attached with different attributes or properties. For example, the user-side information may include the gender, age, and personality, and the item-side information may contain the category, text description, and image. This attribute information can help capture the user's preference. Therefore, another application of hypergraphs in recommender system is attribute modeling and inference.

Another popular yet challenging social media computing application is sentiment analysis, with the goal of recognizing the real emotions and attitudes of people in social media contexts. Nevertheless, the multi-modality and complexity of social media data have made the task more difficult. For example, the text, images, and videos may coexist in one tweet. Additionally, there are intricate relationships between posts, such as in the dimensions of time, location, and user preferences. Therefore, how to find out the complex relationship between tweets and analyze the user sentiment has become an urgent issue. To this end, hypergraph [18] can be used to formulate the correlation among each sample and conduct robust and accurate multi-modal sentiment prediction, taking into consideration different moods having their own characteristics, and that sentiment analysis should be based on the joint analysis of multiple information. As far as social event detection is concerned, exploring a set of highly related posts becomes more important because of noise and insufficient content in a single post that fails to convey clear and comprehensive information. Hypergraph [19] can be used to characterize the relationship between heterogeneous data among different tweets for its superiority in modeling high-order correlations between data of various posts, modalities, and times, therefore enabling real-time social event detection. Specifically, each microblog is connected with its several textual-related and visual-related microblogs and forms two hyperedges. Next, the microblog clique, a basic unit consisting of a set of highly related tweets, is produced by using the hypergraph cut method to put together microblogs that are about the same subject.

Hypergraph has also shown its advantage in medical and biological applications. In the past few decades, massive amounts of biological and medical data have been produced. The data is complex, heterogeneous, and multi-modal, with interwoven inter- and intra-data correlations. By concatenating hyperedge groups, the hypergraph [20–22] can naturally accommodate multi-modal or heterogeneous data. Moreover, in doing so, it can discriminatively use the complementary information among these data. The pipeline below can be used to describe how hypergraph computation is used in biological and medical tasks: (1) modeling the medical image, patches, or biological entities as vertices and connecting them with hyperedges based on their feature similarity or high-order topological links and (2) learning high-order correlations between data using a series of hypergraph computation methods. In this type of applications, hypergraph has been used for mild cognitive impairment (MCI) identification using magnetic resonance imaging (MRI) [23], COVID-19 identification using CT imaging [24], ASD identification using brain functional networks [25], medical image retrieval [26], etc.

The aforementioned examples are just a small part of hypergraph applications. Hypergraph computation techniques can be used in any cases where there exist high-order and complex correlations among data, such as computer vision, knowledge graph, and so on.

1.4 The History of Studies on Hypergraph

1.4.1 Topology and Coloring on Hypergraph

The studies of utilization on hypergraph have a long history. In 1943, Prenowitz et al. [27] first illustrated several kinds of geometries (projective, descriptive, and spherical) as hypergroup or multigroup. Prenowitz et al. [28] created Geometries on Join Spaces, a unique hypergroup that has been proven to be a valuable tool in the study of a variety of topics, including graphs, hypergraphs, binary relations, fuzzy sets, and rough sets. In 1996, Rosenberg et al. [29] first addressed the relationships between Hyperstructures (hypergraphs) and Binary Relations in the broadest sense. Later, they were also studied by Corsini and Leoreanu [30]. Rosenberg et al. [29] first developed join spaces related to fuzzy sets in 1996. Corsini, Leoreanu, and Tofan [31] have all reexamined these structures. Zahedi et al. [32] also advanced the concepts of linking a hypergraph with a fuzzy set and examining algebraic structures equipped with a fuzzy structure.

Hypergraph coloring is a typical and important task, which has attracted much attention since last century. It is fundamental to combinatorics and can be used to determine bounds for the chromatic number of some graphs as described by Kierstead et al. [33]. Lu et al. [34] suggested these algorithms to solve different optimization problems, such as divide and conquer and partition problems, in which hypergraph coloring can also be used to find monochromatic paths and cycles. Voloshin et al. [35, 36] described how to color mixed hypergraphs, which are divided into hyperedge and anti-hyperedge families. In such a case, they further applied it to energy supply problem.

The problem of finding large matches is closely related to the problem of bounding the chromatic index of a hypergraph (notice that the color classes of a proper edge-coloring form a matching). As a classical subject in the study of graphs, matching theory is very well developed and goes back to the work [37] in the 1930s. Tutte's theorem [38] is a characterization of graphs that contains perfect matchings. Edmonds et al. [39] proposed the Blossom algorithm, which uncovers a maximum matching in a graph in a polynomial amount of time for graphs containing a perfect matching. The above methods are early works on hypergraph-related research.

1.4.2 Hypergraph Partitioning, Clustering, and Machine Learning

Hypergraph partition is another important problem on hypergraph. It is defined in the Encyclopedia of Parallel Computing[1] that hypergraph partitioning involves

[1] https://link.springer.com/referenceworkentry/10.1007/978-0-387-09766-4_1.

dividing a hypergraph into two or more roughly equal parts in such a way that the cost function of the hyperedge connecting vertices in the different parts is minimized. In many cases, this definition is too restrictive and requires more than two parts. Karypis et al. [40] proposed the hMetis algorithm, which is based on multilevel coarsening of hypergraphs. The method iteratively bisections coarsened hypergraphs, starting with the smallest. George et al. [41] further developed the hMeTiS-Kway algorithm, which directly constructs a K-way partitioning of a hypergraph with coarse–uncoarse paradigm to solve the K-way hypergraph partitioning problem.

Besides, Papa et al. [42] provided several methods of partitioning hypergraphs and defines clustering as "the process of merging vertices into larger groups of vertices known as clusters to compute a coarser hypergraph from an input hypergraph." A number of applications of partitioning and clustering are also given, including VLSI design, numerical linear algebra, automated theorem proving, and formal verification. Several applications and methods have been described in the literature. For more details, a survey of clustering ensemble techniques has been published in [43], which includes hypergraph partitioning techniques as well. Multilevel strategies are often required in clustering and partitioning, which have been well studied in previous works. It has been extensively used in VLSI design [40], parallel scientific computing [44–46], image categorization [47], and social networks [48, 49].

In this century, hypergraph has been used in machine learning. Transductive hypergraph learning [48] is introduced to give the basic mathematical formulation of the objective function for predicting labels of vertices on a hypergraph. Since the performance of hypergraph learning is related to the modeling quality of the hypergraph, there are some efforts to further assign weights to the components in the hypergraph, including hyperedges, vertices, and hyperedge-dependent vertex weights [50, 51]. To accelerate the label propagation process on hypergraph, the cross diffusion on multiple hypergraphs is further introduced to model the high-order correlations among multi-modal data and conduct multi-modal information fusion [52].

1.4.3 Deep Learning on Hypergraph

Research on high-order representations of hypergraph structures has also been inspired by deep learning's powerful learning and modeling abilities. Generally speaking, most deep learning methods on hypergraph can be divided into spectral-based methods and spatial-based methods.

As for the spectral-based methods, Feng et al. [53] proposed Hypergraph Neural Networks (HGNNs) to model non-pairwise relations based on the hypergraph Laplacian. Multi-modal data can be naturally modeled using the proposed methods. It is also possible to classify images using hypergraph neural networks[54]. Using tools from the spectral theory of hypergraphs, Yadati et al. [55] proposed

HyperGCN to train a GCN for semi-supervised learning on hypergraphs using graph convolutional networks (GCNs). As for the spatial-based method, by extending the dynamic hypergraph learning, Jiang et al. [56] proposed a dynamic hypergraph neural network, which can adaptably change the hypergraph structure at each layer. As opposed to hypergraph convolution, where the underlying structure is defined beforehand, Bai et al. [57] proposed a hypergraph attention mechanism strategy to learn a dynamic connection of hyperedges, which propagates and gathers information in the task-relevant parts of the graph, thereby generating more discriminative vertex embeddings. Moreover, Gao et al. [58] proposed a general hypergraph neural network framework, which can be applied to multiple types of hypergraphs like undirected hypergraph, directed hypergraph, probabilistic hypergraph, vertex/hyperedge weighted hypergraph, etc.

For homogeneous and heterogeneous hypergraphs, Zhang et al. [59] proposed a self-attention-based hypergraph neural network (Hyper-SAGNN). By mapping the hypergraph to a weighted attribute line graph, Bandyopadhyay et al. [60] achieved a bi-injective hypergraph structure. Huang et al. [61] proposed UniGNN, which can generalize general GNN models into hypergraphs by interpreting the message passing process in graph and hypergraph neural networks. These neural network methods on hypergraph enable the representation learning by incorporating high-order correlation in process.

1.5 Hypergraph Computation: Challenges and Objectives

Hypergraph has its advantage on high-order correlation modeling compared with graph and other structures. To take this advantage in practice, hypergraph can be used to formulate such correlations and the conduct computing task accordingly. In this part, we summarize the objective of hypergraph computation, especially the main challenges and the tasks inside.

Below we give the definition of hypergraph computation: **hypergraph computation** is to formulate the high-order correlations underneath the data using hypergraph and then conduct semantic computing on the hypergraph for different applications.

The main challenges and objectives in hypergraph computation are from three parts, including how to generate a hypergraph, how to deal with large scale data, and how to conduct learning on hypergraph.

1. **How to generate a hypergraph.** In most cases, the hypergraph structure is not explicitly existed. What can be observed could be non-structure data, such as images, videos and discrete signals, and pairwise relationships between two subjects. To reveal the underneath high-order correlation as a hypergraph, it is needed to define how to generate it. More importantly, the observed data could be noisy, missing, and tend to be multi-modal. How to describe these data is also challenged. Under such circumstances, it is difficult to generate an accurate hypergraph structure based on these data. Therefore, how to generate a

hypergraph, especially a good hypergraph structure for specific task, is the first challenge in practice.

2. **How to deal with large scale data.** Computational complexity is a major issue for graph data, which is also very serious for hypergraph. The data in many applications, such as social media and brain neurons, are in million level or more. Confronting such large scale data, how to effectively and efficiently conduct storage and computing on hypergraph require further research.
3. **How to conduct learning on hypergraph.** Given a hypergraph, learning task can be conducted on the hypergraph structure, and it is important to design label propagation method on hypergraph. Besides traditional feature representation methods, the connections can also be used as representation. Given such high-order correlation by hypergraph, it is useful to learn new representations on hypergraph. Therefore, how to conduct representation learning on hypergraph is an important topic.

Hypergraph modeling can be briefly divided into two categories, i.e., the intra-correlation modeling and the inter-correlation modeling, as shown in Fig. 1.5. Here, the intra-correlation modeling regards the high-order correlations inside the subject. The components of the subject are represented as the vertices, and the correlations among these components are represented as hyperedges in the hypergraph. In these cases, the hypergraph, named intra-hypergraph, aims to represent the subject itself. The inter-correlation modeling concentrates on the high-order correlations among different subjects. A group of subjects is represented as the vertices, and the

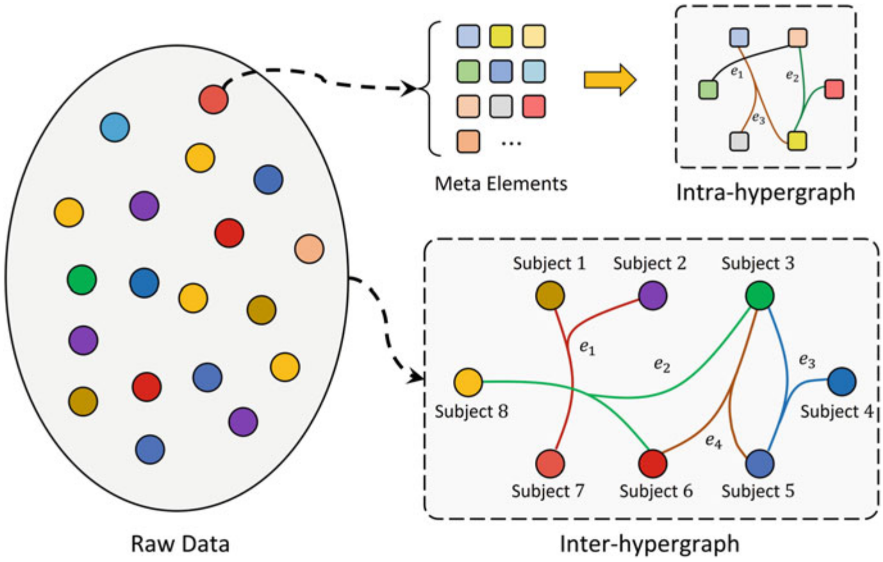


Fig. 1.5 The intra-hypergraph and the inter-hypergraph based on the intra-correlations and the inter-correlations among components and subjects

correlations among these subjects are represented as hyperedges in the hypergraph, named inter-hypergraph. The objective is to learn the representation or connections of the target subject with the help of its correlations to other subjects. Here we take image representation as an example. When an image is selected as the subject, the correlations among the pixels or the patches in the image are intra-correlations, and the corresponding intra-hypergraph can be generated for image representation. On the other side, we can also observe other images for processing. The correlations among the subject image and other images are inter-correlations, and the corresponding inter-hypergraph can be generated for image representation too. That is to say, the intra- and inter-correlations can be regarded as the views from different scales. If we take the subject itself as the target system, the correlations of the subject and other subjects are inter-correlations of the subject, corresponding to an inter-hypergraph. If we take the group of subjects as the target system, the correlations of these subjects are intra-correlations, leading to an intra-hypergraph accordingly.

1.6 Structure of This Book

This book is composed of 13 chapters and the structure of the remainders is introduced here.

- Chapter 2. Mathematical Foundations of Hypergraph. This chapter introduces the fundamental mathematics of hypergraph and presents the mathematical notations that are used to facilitate deep understanding and analysis of hypergraph structure.
- Chapter 3. Hypergraph Computation Paradigm. This chapter introduces three typical hypergraph computation paradigms, including inter-representation computing, inter-representation computing, and group correlation computing.
- Chapter 4. Hypergraph Modeling. This chapter introduces different hypergraph modeling methods, including implicit hypergraph modeling and explicit hypergraph modeling. Examples on computer vision, recommender system, and other applications are also provided in this chapter.
- Chapter 5. Typical Hypergraph Computation Tasks. This chapter introduces the typical hypergraph computation tasks, including label propagation on hypergraph, data clustering on hypergraph, imbalanced learning on hypergraph, and link prediction on hypergraph.
- Chapter 6. Hypergraph Structure Evolution. This chapter introduces the structure evolution methods on the hypergraph, which optimize the hypergraph structure accordingly, including both the hypergraph component optimization and hypergraph structure optimization. We briefly introduce the incremental learning method on growing data.
- Chapter 7. Neural Networks on Hypergraph. This chapter introduces recent progresses on hypergraph neural networks, including the spectral-based methods

and the spatial-based methods. The comparison between graph neural networks and hypergraph neural networks is also provided in this chapter.

- Chapter 8. Large Scale Hypergraph Computation. This chapter introduces how to deal with large scale data. More specifically, two kinds of large scale hypergraph computation methods, i.e., factorization-based hypergraph reduction and hierarchy-based hypergraph learning, are provided in this chapter.
- Chapter 9. Hypergraph Computation for Social Media Analysis. This chapter introduces applications of hypergraph computation on social media analysis, including recommender system, sentiment analysis, and emotion recognition.
- Chapter 10. Hypergraph Computation for Medical and Biological Applications This chapter introduces applications of hypergraph computation on medical and biological applications, including computer-aided diagnosis, survival prediction with histopathological image, drug discovery, and medical image segmentation.
- Chapter 11. Hypergraph Computation for Computer Vision. This chapter introduces applications of hypergraph computation on computer vision, including visual classification, 3D object retrieval, and tag-based social image retrieval.
- Chapter 12. The DeepHypergraph Library. This chapter introduces the DeepHypergraph Library, a hypergraph computation library based on Python.
- Chapter 13. Conclusions and Future Work. This chapter concludes this book and introduces three further research directions of hypergraph computation.

1.7 Summary

In this chapter, we introduce the basic ideas and background of hypergraph computation. We also provide the applications and the related research history on hypergraph. The idea of hypergraph computation is detailed introduced and discussed in this chapter. We also summarize our studies on hypergraph computation and present the organization of this book.

References

1. D.J. Watts, S.H. Strogatz, Collective dynamics of ‘small-world’ networks. Nature **393**(6684), 440–442 (1998)
2. A.L. Barabási, R. Albert, Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
3. A.D. Broido, A. Clauset, Scale-free networks are rare. Nat. Commun. **10**(1), 1–10 (2019)
4. F. Crivello, O. Étard, N. Delcroix, B. Mazoyer, M. Joliot, Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. NeuroImage **15**, 273–289 (2002)
5. E. Almaas, B. Kovacs, T. Vicsek, Z.N. Oltvai, A.L. Barabási, Global organization of metabolic fluxes in the bacterium Escherichia coli. Nature **427**(6977), 839–843 (2004)
6. A. Bashan, S. Havlin, The combined effect of connectivity and dependency links on percolation of networks. J. Statist. Phys. **145**(3), 686–695 (2011)

7. P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, J.P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–878 (2010)
8. A.L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, T. Vicsek, Evolution of the social network of scientific collaborations. Phys. A Statist. Mech. Appl. **311**(3), 590–614 (2002)
9. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: simple building blocks of complex networks. Science **298**(5594), 824–827 (2002)
10. J. Grilli, G. Barabás, M.J. Michalska-Smith, S. Allesina, Higher-order interactions stabilize dynamics in competitive network models. Nature **548**(7666), 210–213 (2017)
11. A.R. Benson, R. Abebe, M.T. Schaub, A. Jadbabaie, J. Kleinberg, Simplicial closure and higher-order link prediction. Proc. Natl. Acad. Sci. **115**(48), E11221–E11230 (2018)
12. A.R. Benson, D.F. Gleich, J. Leskovec, Higher-order organization of complex networks. Science **353**(6295), 163–166 (2016)
13. S. Basu, K. Kumbier, J.B. Brown, B. Yu, Iterative random forests to discover predictive and stable high-order interactions. Proc. Natl Acad. Sci. **115**(8), 1943–1948 (2018)
14. P. Frankl, Extremal set systems, in *Handbook of Combinatorics*, vol. 2 (Elsevier, Amsterdam, 1996), pp. 1293–1329
15. C. Berge, *Hypergraphs: Combinatorics of Finite Sets* (Elsevier, Amsterdam, 1984)
16. P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
17. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (2020), pp. 2020–2029
18. R. Ji, F. Chen, L. Cao, Y. Gao, Cross-modality microblog sentiment prediction via bi-layer multimodal hypergraph learning, IEEE Trans. Multimedia. **21**(4), 1062–1075 (2019)
19. S. Zhao, Y. Gao, G. Ding, T.S. Chua, Real-time multimedia social event detection in microblog, IEEE Trans. Cyber. **48**(11), 3218–3231 (2018)
20. D. Di, S. Li, J. Zhang, Y. Gao, Ranking-based survival prediction on histopathological whole-slide images, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, (2020), pp. 428–438
21. D. Di, J. Zhang, F. Lei, Q. Tian, Y. Gao, Big-hypergraph factorization neural network for survival prediction from whole slide image. IEEE Trans. Image Process. **31**, 1149–1160 (2022)
22. D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, Y. Gao, Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction. IEEE Trans. Pattern Analy. Mach. Intell. 1–16 (2022). https://doi.org/10.1109/TPAMI.2022.3209652
23. Y. Gao, C. Wee, M. Kim, P. Giannakopoulos, M. Montandon, S. Haller, D. Shen, MCI identification by joint learning on multiple MRI data, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 78–85
24. D. Di, F. Shi, F. Yan, Liming Xia, Z. Mo, Z. Ding, F. Shan, B. Song, S. Li, Y. Wei, Y. Shao, M. Han, Y. Gao, H. Sui, Y. Gao, D. Shen, Hypergraph learning for identification of COVID-19 with CT imaging. Med. Image Analy. **68**, 101910 (2021)
25. Z. Zhang, J. Liu, B. Li, Y. Gao, Diagnosis of childhood autism using multi-modal functional connectivity via dynamic hypergraph learning, in *Proceedings of the CAAI International Conference on Artificial Intelligence* (2021), pp. 123–135
26. Y. Gao, M. Kim, P. Giannakopoulos, S. Haller, D. Shen, Medical image retrieval using multi-graph learning for MCI diagnostic assistance, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, (2015), pp. 86–93
27. W. Prenowitz, Projective geometries as multigroups. Amer. J. Math. **65**(2), 235–256 (1943)
28. J. Jantosciak, W. Prenowitz, Geometrics and join spaces. J. Fur Die Reine Und Angewandte Mathematik **257**, 100–128 (1972)
29. I. G. Rosenberg, Wall monoids, in *New Frontiers in Hyperstructures*, vol. 166 (Hadronic Press, Palm Harbor, 1996)
30. V. Leoreanu, Direct limit and inverse limit of join spaces associated with fuzzy sets. Pure Math. Appl. **11**(3), 509–516 (2000)

31. I. Tofan, A.C. Volf, On some connections between hyperstructures and fuzzy sets. Ital. J. Pure Appl. Math. **7**, 63–68 (2000)
32. R. Ameri, M.M. Zahedi, Hypergroup and join spaces induced by a fuzzy subset. Pure Math. Appl. **8**(2–4), 155–168 (1997)
33. H.A. Kierstead, V. Rodl, Applications of hypergraph coloring to coloring graphs not inducing certain trees. Discrete Math. **150**(1–3), 187–193 (1996)
34. C.J. Lu, Deterministic hypergraph coloring and its applications. SIAM J. Discrete Math. **18**(2), 320–331 (2004)
35. V.I. Voloshin, The mixed hypergraphs. Computer Science J. Moldova. **1**(1), 1 (1993)
36. V.I. Voloshin, *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications*, vol. 17, (American Mathematical Society, Providence, 2002)
37. H. Philip, On representatives of subsets. J. London Math. Soc. **10**(1), 26–30 (1935)
38. W.T. Tutte, The factorization of linear graphs. J. London Math. Soc. **1**(2), 107–111 (1947)
39. J. Edmonds, Paths, trees, and flowers. Can. J. Math. **17**, 449–467 (1965)
40. G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans. Very Large Scale Integr. Syst. **7**(1), 69–79 (1999)
41. G. Karypis, V. Kumar, Multilevel k-way Hypergraph Partitioning. VLSI Design **11**(3), 285–300 (2000)
42. D. A. Papa, I. L. Markov, Hypergraph partitioning and clustering, in *Handbook of Approximation Algorithms and Metaheuristics* (CRC Press, Boca Raton, 2007), pp. 61–1–61–19
43. R. Ghaemi, M.N. Sulaiman, H. Ibrahim, N. Mustapha, A survey: clustering ensembles techniques. Int. J. Comput. Inf. Eng. **3**(2), 365–374 (2009)
44. U.V. Catalyurek, C. Aykanat, Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans. Parallel Distributed Syst. **10**(7), 673–693 (1999)
45. K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, U.V. Catalyurek, Parallel hypergraph partitioning for scientific computing, in *Proceedings of the IEEE International Parallel & Distributed Processing Symposium* (2006)
46. G. Ballard, A. Druinsky, N. Knight, O. Schwartz, Hypergraph partitioning for sparse matrix-matrix multiplication. ACM Trans. Parallel Comput. **3**(3), 1–34 (2016)
47. Y. Huang, Q. Liu, F. Lv, Y. Gong, D.N. Metaxas, Unsupervised image categorization by hypergraph partition. IEEE Trans. Pattern Analy. Mach. Intell. **33**(6), 1266–1273 (2011)
48. D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: Clustering, classification, and embedding, in *Proceedings of the Advances in Neural Information Processing Systems* (2006), pp. 1601–1608
49. W. Yang, G. Wang, M.Z.A. Bhuiyan, K.K.R. Choo, Hypergraph partitioning for social networks based on information entropy modularity. J. Netw. Comput. Appl. **86**, 59–71 (2017)
50. Y. Gao, M. Wang, Z.J. Zha, J. Shen, X. Li, X. Wu, Visual textual joint relevance learning for tag-based social image search. IEEE Trans. Image Process. **22**(1), 363–376 (2013)
51. Z. Zhang, H. Lin, Y. Gao, Dynamic hypergraph structure learning, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2018), pp. 3162–3169
52. Z. Zhang, H. Lin, J. Zhu, X. Zhao, Y. Gao, Cross-diffusion on multi-hypergraph for multi-modal 3D object recognition, in *Proceedings of the Pacific-Rim Conference on Multimedia* (2018), pp. 38–49
53. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
54. H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao, J. Sun, Hypergraph-induced convolutional networks for visual classification. IEEE Trans. Neural Netw. Learn. Syst. **30**(10), 2963–2972 (2018)
55. N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, P. Talukdar, HyperGCN: A new method for training graph convolutional networks on hypergraphs, in *Proceedings of the Advances in Neural Information Processing Systems* (2019), pp. 1511–1522
56. J. Jiang, Y. Wei, Y. Feng, J. Cao, Y. Gao, Dynamic hypergraph neural networks, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2019), pp. 2635–2641

57. S. Bai, F.Zhang, P.H. Torr, Hypergraph convolution and hypergraph attention. Pattern Recog. **110**, 107637 (2021)
58. Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN$^+$: General hypergraph neural networks. IEEE Trans. Pattern Analy. Mach. Intell. **45**(3), 3181–3199 (2023)
59. R. Zhang, Y. Zou, J. Ma, Hyper-SAGNN: A self-attention based graph neural network for hypergraphs, in *Proceedings of the International Conference on Learning Representations* (2020)
60. S. Bandyopadhyay, K. Das, M.N. Murty, Line hypergraph convolution network: applying graph convolution for hypergraphs (2020). Preprint arXiv:2002.03392
61. J. Huang, J. Yang, Unignn: A unified framework for graph and hypergraph neural networks, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2021), pp. 2563–2569

Chapter 2
Mathematical Foundations of Hypergraph

Abstract In this chapter, we introduce the mathematical foundations of hypergraph and present the mathematical notations that are used to facilitate deep understanding and analysis of hypergraph structure. A hypergraph is composed of a set of vertices and hyperedges, and it is a generalization of a graph, where a weighted hypergraph quantifies the relative importance of hyperedges or vertices. Hypergraph can also be divided into two main categories, i.e., the undirected hypergraph representation and the directed hypergraph representation. The latter one further divides the vertices in one hyperedge into the source vertex set and the target vertex set to model more complex correlations. Additionally, we discuss the relationship between hypergraph and graph from the perspective of structural transformation and expressive ability. The most intuitive difference between a simple graph and a hypergraph can be observed in the size of order and expression of adjacency. A hypergraph can be converted into a simple graph using clique expansion, star expansion, and line expansion. Moreover, the proof based on random walks and Markov chains establishes the relationship between hypergraphs with edge-independent vertex weights and weighted graphs.

2.1 Introduction

The importance of high-order complex network modeling has been discussed in Chap. 1. In this chapter, we introduce the basic knowledge of hypergraph. In a hypergraph, the edge degree is usually higher than that of a simple graph, which is two for a simple graph. Different from a graph structure that can model pairwise connections with its 2-degree edges, a hypergraph can model correlations between practical data that are much more complex than pairwise relationships. As a result of its versatility and usefulness of modeling complex correlations of data, machine learning on hypergraph has attracted increasing attention.

Machine learning methods on hypergraph have been used in many real-world applications due to its advantages. A wide variety of tasks have been performed with hypergraph in computer vision, including image retrieval [1] and 3D object classification [2], video segmentation [3], re-identification of people [4], hyper-spectral image analysis [5], landmark retrieval [6], and visual tracking [7]. It is possible to embed a wide range of subjects into a hypergraph structure for these tasks. In different tasks, the hypergraph structure can be used to formulate the correlation among a variety of subjects. In image retrieval [3], the correlation among different images can be modeled in a hypergraph, where each vertex denotes an image and the hyperedges can be generated by finding similar image features. In 3D object classification [2], the correlation among different 3D objects can be modeled in a hypergraph, where each vertex denotes a 3D object and the hyperedges can be generated based on the similarity among these 3D objects. In person re-identification [4], a hypergraph structure can be constructed, where each vertex represents a personal image and the hyperedges can be generated based on the similarities in the feature space. Similar modeling attempts have been deployed in medical image analysis and bio-informatics studies to identify genes [8, 9], predict diseases [10, 11], identify sub-types [12], and analyze functional networks [13].

Before detailed introduction of the hypergraph computation paradigm, hypergraph modeling, and other related methods and applications, in this chapter, we first present preliminary knowledge of hypergraph and multiple representations of hypergraph. We also compare the hypergraph structure with the graph structure from four aspects.

2.2 Preliminary Knowledge of Hypergraph

The basic concepts of hypergraph are hereby briefly discussed. Table 2.1 provides the main notations and definitions of hypergraphs throughout this chapter. We first introduce undirected hypergraph and directed hypergraph, respectively, and then introduce the K-uniform hypergraph, probabilistic hypergraph, the relationship between hypergraph and bipartite graph, and the weights on hypergraph.

2.2.1 Undirected Hypergraph

Let $\mathcal{G}$ be an indication of a hypergraph (undirected hypergraph), which 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 with a weight $w(e)$, symbolizing the importance of the connection relationship throughout the whole hypergraph. Let $\mathbf{W}$ denote the diagonal matrix of the hyperedge weights, i.e., $\text{diag}(\mathbf{W}) = \left[w\left(e_1\right), w\left(e_2\right), \ldots, w\left(e_{|\mathcal{E}|}\right)\right]$. 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:

$$\mathbf{H}(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{if } v \notin e, \end{cases} \tag{2.1}$$

Table 2.1 Notations and definitions of hypergraphs

Notation	Definition
$\mathscr{G}$	The hypergraph
$\mathscr{V}$	The set of vertices
$\mathscr{E}$	The set of hyperedges
$\mathbf{W}$	The diagonal matrix of the hyperedge weights
$\mathbf{U}$	The diagonal matrix of the vertex weights
$\mathbf{X}$	The vertex feature matrix
$\mathbf{Y}$	The vertex label matrix
$\mathbf{H}$	The $\lvert\mathscr{V}\rvert \times \lvert\mathscr{E}\rvert$ incidence matrix of undirected hypergraph structure. $\mathbf{H}(v, e)$ indicates the connection strength between vertex v and 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
$\mathbf{x}_i$	The feature vector of vertex v_i
$d(v)$	The degree of vertex v
$\delta(e)$	The degree of hyperedge e
$w(e)$	The weight of hyperedge e
$u(v)$	The weight of vertex v

where $\mathbf{H}(v, e)$ indicates the possibility of vertex v assigned to hyperedge e or the importance of vertex v for hyperedge e. The degree of hyperedge e and the degree of vertex v are defined as follows:

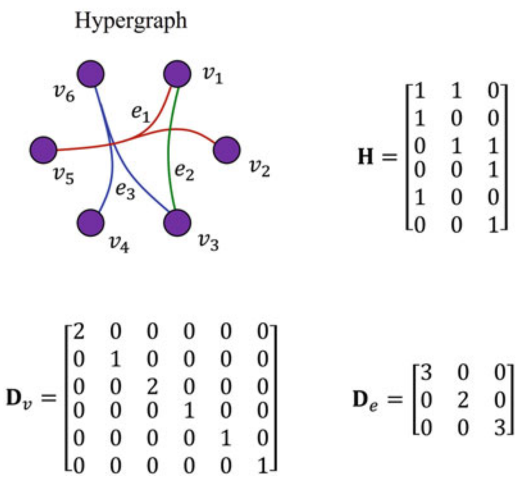
$$\delta(e) = \sum_{v \in \mathscr{V}} \mathbf{H}(v, e), \tag{2.2}$$

and

$$d(v) = \sum_{e \in \mathscr{E}} w(e) * \mathbf{H}(v, e). \tag{2.3}$$

The traditional hypergraph structure creates associations among vertices, with a single hyperedge connecting multiple vertices that have associations. All vertices on the same hyperedge are given a value of 1 in the incidence matrix $\mathbf{H}$. The adjacency matrix $\mathbf{H}$ is calculated as in (2.1), whose elements are valued by 0 or 1. Each row represents each vertex in the hypergraph and the columns represent all hyperedges. Each column represents the set of vertices on this hyperedge.

Figure 2.1 shows an undirected hypergraph, including the hypergraph itself, the incidence matrix $\mathbf{H}$, the vertex set $\mathscr{V}$, the hyperedge set $\mathscr{E}$, and the weight matrix $\mathbf{W}$. In the illustrated undirected hypergraph, there are 3 hyperedges e_1, e_2, and e_3 with 6 vertexes. The degree of the hyperedge e_3 is 3, which contains vertices $\{v_3, v_4, v_6\}$. By the same token, other elements of $\mathbf{D}_v$ can be inferred. Vertex v_3 belongs to the hyperedges e_2 and e_3, and the degree of the vertex is 2. The incidence matrix $\mathbf{H}$ of



$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D}_v = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{D}_e = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Fig. 2.1 An example of an undirected hypergraph

hypergraph is readily obtained by the rules of construction, which are shown on the right side of Fig. 2.1.

Given the incidence matrix $\mathbf{H}$ as calculated as in Eq. (2.1), all elements are valued by either 0 or 1. It is noted that the connection weights of different vertices on a hyperedge could be different. For example, some vertices are highly connected in the hyperedge and with high weights, while others may be with low weights. That is to say, the sum of each column of $\mathbf{H}$ is 1 (or not, due to different applications and objectives) and its values represent the vertex importance on this hyperedge.

There are various rules that can be used to determine whether vertices are associated with one another. Hyperedge groups can be generated from the data with a graph structure by using pairwise edges and k-hops; for the data without a graph structure, they can be generated by using neighbors in feature space. A detailed description of these methods is provided in Chap. 4.

2.2.2 *Directed Hypergraph*

The real world is incompatible with traditional undirected hypergraph representation in that hyperedges may be directional. Therefore, the representation of directed hypergraph structures is important. In each hyperedge, the vertex can be further divided into two sets: the source vertex set and the target vertex set. On directed hypergraph, a trivial definition [14] for the incidence matrix is defined as follows:

$$\hat{\mathbf{H}}(v, e) = \begin{cases} -1 & \text{if } v \in T(e) \\ 1 & \text{if } v \in S(e) \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

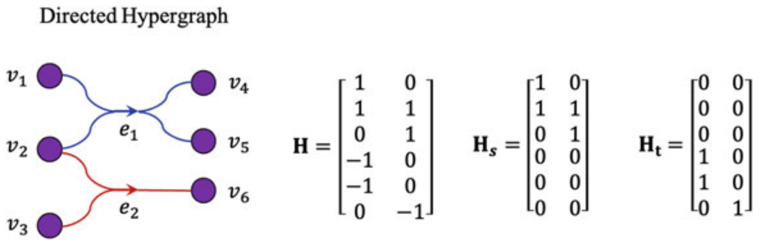

Directed Hypergraph
v_1
v_2
v_3
v_4
v_5
v_6
e_1
e_2
$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$
$\mathbf{H_s} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$
$\mathbf{H_t} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

Fig. 2.2 An example of a directed hypergraph

where $T(e)$ and $S(e)$ are the target and source vertices for hyperedge e, respectively. The incidence matrix $\mathbf{H}$ is split into two matrices, $\mathbf{H_s}$ and $\mathbf{H_t}$, describing the source and target vertices for all hyperedges, respectively. When passing messages with these two incidence matrices, it is important to maintain the directional information. Two different incidence matrices guide message passing in the directed hypergraph, $\mathbf{H_s}$ and $\mathbf{H_t}$, unlike in the undirected hypergraph. The average aggregation of messages is normalized by $\mathbf{D_s}$ and $\mathbf{D_t}$ as two matrices, and it can be formulated as follows:

$$\begin{cases} \mathbf{D_s} = \text{diag}(\text{col\_sum}(\mathbf{H}_s)) \\ \mathbf{D_t} = \text{diag}(\text{col\_sum}(\mathbf{H}_t)), \end{cases} \tag{2.5}$$

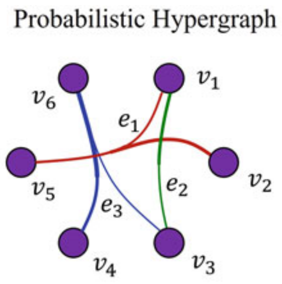
where $\text{diag}(v)$ is a function that converts a vector v to a diagonal matrix. The $\text{col\_sum}(\cdot)$ is a column accumulation function.

Figure 2.2 shows an example of directed hypergraph including the directed hypergraph itself, the incidence matrix $\mathbf{H}$, the source incidence matrix $\mathbf{H}_s$, and the target incidence matrix $\mathbf{H}_t$. The illustrated directed hypergraph contains six vertices and two hyperedge e_1 and e_2. e_1 connects four vertices and e_2 connects three vertices. In hyperedge e_1, the source vertices are v_1 and v_2, and the target vertices are v_4 and v_5. As for the hyperedge e_2, the source vertices are v_2 and v_3, and the target vertices are only v_6.

2.2.3 *Probabilistic Hypergraph*

In the real-world correlations, the intensity of the connection can not only be a binary number but also be a continuous number from zero to one. Consequently, the incidence matrix may be a continuous matrix with elements ranging from 0 to 1, which is adopted to denote a probabilistic hypergraph.

As shown in Fig. 2.3, the probabilistic hypergraph consists of six vertices and three hyperedges. The hyperedge e_1 connects three vertices v_1, v_2, and v_5. The intensity of the connection in this hyperedge is not the same. As shown in the right side of the figure, e_1 connects v_1 with an intensity of 0.3, connects v_2 with



$$\mathbf{H} = \begin{bmatrix} 0.3 & 0.7 & 0.0 \\ 0.8 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.2 \\ 0.0 & 0.0 & 0.8 \\ 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.9 \end{bmatrix}$$

$$\mathbf{D}_v = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 \end{bmatrix} \qquad \mathbf{D}_e = \begin{bmatrix} 1.6 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.9 \end{bmatrix}$$

Fig. 2.3 An example of a probabilistic hypergraph

an intensity of 0.8, and connects v_5 with an intensity of 0.5. The degree of vertex and hyperedge in this type of hypergraph is computed by the sum of the row or column of the hypergraph incidence matrix **H**, as shown in the bottom of Fig. 2.3.

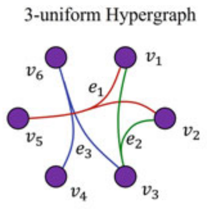
2.2.4 K-Uniform Hypergraph

In many applications, hyperedges in a hypergraph may connect the same number of vertices, which is known as the k-uniform hypergraph. In the k-uniform hypergraph, each hyperedge contains precisely k vertices, as shown in Fig. 2.4. Under this definition, a simple can be regarded as a spatial case of hypergraph, a 2-uniform hypergraph, where each hyperedge only connects two vertices.

Figure 2.4 illustrates an example of 3-uniform hypergraph. The hypergraph consists of six vertices and three hyperedges, and each hyperedge contains precisely 3 vertices. Hyperedge e_1 connects vertices v_1, v_2, and v_5. Hyperedge e_2 connects vertices v_1, v_2, and v_3. The degree of all hyperedges in this type of hypergraph is consistent k.

2.2.5 Hypergraph and Bipartite Graph

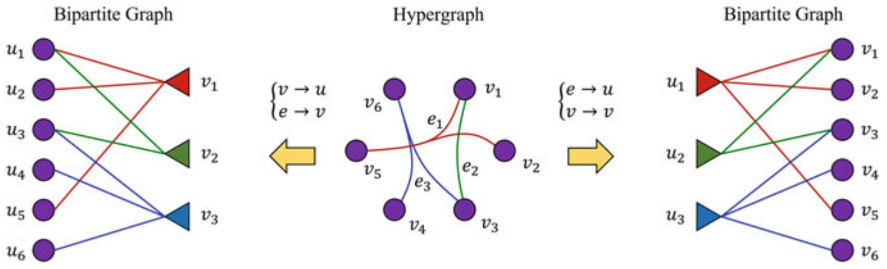
The bipartite graph can be indicated by $\mathscr{G} = \{\mathscr{U}, \mathscr{V}, \mathscr{E}\}$. Unlike the simple graph, vertices in the bipartite can be divided into two disjoint and independent sets $\mathscr{U}$



$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D}_v = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{D}_e = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Fig. 2.4 An example of a 3-uniform hypergraph



$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Leftarrow \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Fig. 2.5 The relationship between hypergraph and bipartite graph

and $\mathscr{V}$. Every edge only connects one vertex in set $\mathscr{U}$ and another vertex in set $\mathscr{V}$. Obviously, an undirected hypergraph can be regarded as a bipartite graph if the hyperedges are treated as another vertex set, as shown in Fig. 2.5.

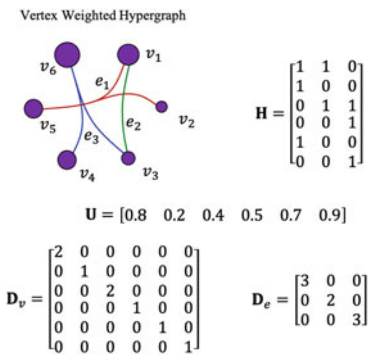
Figure 2.5 illustrates examples of converting hypergraph to bipartite graph. The bipartite graph can be generated by two strategies: the vertices and hyperedges are treated as vertices in $\mathscr{U}$ and vertices in $\mathscr{V}$ (as illustrated in the left part), and the vertices and hyperedges are treated as vertices in $\mathscr{V}$ and vertices in $\mathscr{U}$ (as illustrated in the right part). Similarly, a bipartite graph can also be transformed to an undirected hypergraph with set $\mathscr{U}/\mathscr{V}$ as the hyperedges. It is not mean that the hypergraph is the same as or can be replaced with the bipartite graph. The transformation only exists in the undirected hypergraph and the probabilistic hypergraph. Confronting more complex hypergraph like directed hypergraph, the transformation will be invalid.

2.2.6 The Weights on Hypergraph

It is noted that there are different weights on a hypergraph, which provide additional information to assign values to a hypergraph structure. This is a more semantically preferred way of representing a hypergraph, as different components of a hypergraph, such as a vertex, a hyperedge or even a sub-hypergraph, should have different impact on the relationship modeling. For example, in a recommender system, the weights in the user profile influence the categorization of user attributes. If the attributes are not categorized accurately, the accuracy of the recommendations and marketing based on the profile could be questionable. The main types of weight information on a hypergraph are hyperedge weights and vertex weights, with the magnitude of the values indicating the relative importance of hyperedges and vertices, respectively.

First, let us show how the weights on vertex can be used. Different vertices may have varying importance on hypergraph modeling, and vertex weights are used in a hypergraph to determine the importance of different vertices. If a vertex is connected on the hypergraph strongly (with high correlations), it should be with a large vertex weight. Otherwise, it should be with a small vertex weight. For those vertices which have a 0 weight value in the incidence matrix, it can also be regarded as it is connected by the corresponding hyperedge with a weight of 0. Here, the diagonal elements of $\mathbf{U}$ to represent the weights of vertices, which are between 0 and 1, which reveal the relative importance of these hyperedges. Figure 2.6 shows an example hypergraph with vertex weights. In this figure, the weight of each vertex is denoted by the size of the vertex node. Vertex v_6 has a weight of 0.9, which is larger than all other vertices, and vertex v_2 is the smallest among the six vertices.

Then, let us focus on the weights on hyperedge. Hyperedge weights reflect the importance of different hyperedges in a hypergraph. As different hyperedges may have different importance in representing connections among vertices, it is crucial that hyperedges be weighted corresponding to their representative capabilities. In some cases, a part of hyperedges are more reliable due to its generation method or the features employed in this task, and these hyperedges should be given a large



$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{U} = [0.8 \quad 0.2 \quad 0.4 \quad 0.5 \quad 0.7 \quad 0.9]$$

$$\mathbf{D}_v = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{D}_e = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Fig. 2.6 An example of a vertex weighted hypergraph

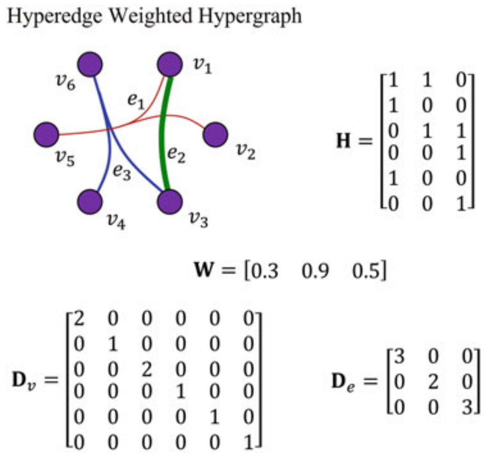

Hyperedge Weighted Hypergraph
v_1
v_2
v_3
v_4
v_5
v_6
e_1
e_2
e_3
$\mathbf{H} = \begin{bmatrix}1&1&0\\1&0&0\\0&1&1\\0&0&1\\1&0&0\\0&0&1\end{bmatrix}$
$\mathbf{W} = [0.3 \quad 0.9 \quad 0.5]$
$\mathbf{D}_v = \begin{bmatrix}2&0&0&0&0&0\\0&1&0&0&0&0\\0&0&2&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}$
$\mathbf{D}_e = \begin{bmatrix}3&0&0\\0&2&0\\0&0&3\end{bmatrix}$

Fig. 2.7 An example of a hyperedge weighted hypergraph

weight during the learning process. Here, the diagonal element values of **W** can be used to represent the weights of vertices, which are between 0 and 1, revealing the relative importance of these hyperedges. Figure 2.7 shows an example of the hyperedge weighted hypergraph. In the illustration, the three hyperedges have the weights 0.3, 0.9, and 0.5, respectively.

2.3 Comparison Between Graph and Hypergraph

As a generalization of graph, the relationship between graph and hypergraph is a fundamental question. In this part, we detailedly introduce the relationship between graph and hypergraph from four aspects, i.e., the order of correlations, the representation methods, the structure transformation and random work on both of them.

2.3.1 Low-Order Versus High-Order Correlations

First, we define the *interaction* as a set $I = [p_0, p_1, \cdots, p_{k-1}]$ containing k basic elements of the system being studied, which can also be called vertices or nodes. Various real-world interactions can be described by such interactions, such as coauthors of a scientific paper, genes required to perform a specific function, neurons co-activating during a specific task, and more. We then denote the order (or dimension) of interactions among vertices as an order-0 interaction for a vertex interacting with itself only, an order-1 interaction for two vertices interacting with each other, an order-2 interaction for three vertices interactions, and so on.

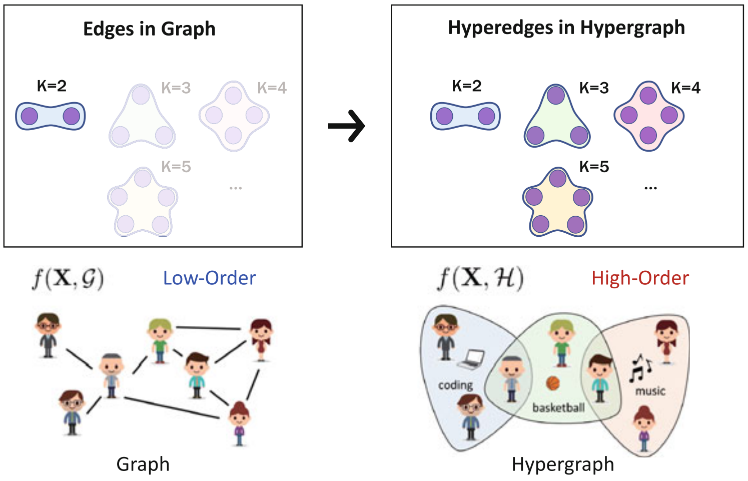


Fig. 2.8 The expressive ability comparison of graph and hypergraph

Furthermore, high-order interactions are considered k-interactions with $k \geq 2$. Low-order interactions, on the other hand, are those characterized by $k \leq 1$.

Figure 2.8 shows the comparison of hypergraph and graph on the modeling of different orders of correlations. We notice that a graph can only represent the order-1 interactions between two vertices. Different from graph, a hypergraph can represent any order-k interactions through its flexible hyperedges. From this direction, hypergraph is more effective on modeling high-order correlation among subjects compared with graph.

2.3.2 Adjacency Matrix Versus Incidence Matrix

A graph with N vertices can be described by an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N\times N}$, where $\mathbf{A}_{i,j} = 1$ denotes that there is an edge connecting vertex v_i and vertex v_j. In most cases, the adjacency matrix $\mathbf{A}$ is a symmetry matrix.

A hypergraph with N vertices and M hyperedges can be described by an incidence matrix $\mathbf{H} \in \{0, 1\}^{N\times M}$, where $\mathbf{H}_{i,j} = 1$ denotes that the hyperedge e_j connects vertex v_i.

By comparison of adjacency matrix and incidence matrix, a graph can be regarded as a 2-uniform hypergraph. In this case, each hyperedge can only connect two vertices. Given the possible $N \times N$ order-1 hyperedges $\mathbf{H}$ in the 2-uniform hypergraph, they can be directly projected to the $N \times N$ elements in adjacency matrix $\mathbf{A}$. The hypergraph incidence and the simple graph adjacency matrix can be bi-transformed as follows:

$$\mathbf{H}\mathbf{H}^\top = \mathbf{A} + \mathbf{D}. \tag{2.6}$$

The adjacency matrix for graph and the incidence matrix for hypergraph have different processing styles when confronting multi-modal data or multiple types of connections. Given m adjacency matrices representing m graphs $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m$, there are two typical ways to combine these data for graph. The first way is to combine different graphs into one graph $\mathcal{G}$ and then conduct other tasks. The second way is to conduct the task in each graph individually and then combine all these results. Figures 2.9 and 2.10 show these two types of methods. In either method, it is required to perform fusion, either in the graph structure part or in the result part. In recent years, a series of graph fusion methods [15, 16] have been introduced, while it is still a challenging task to optimally combine different graphs. On the other side, the multi-modal graph fusion is also with high computational complexity, which may limit the applications on multi-modal data.

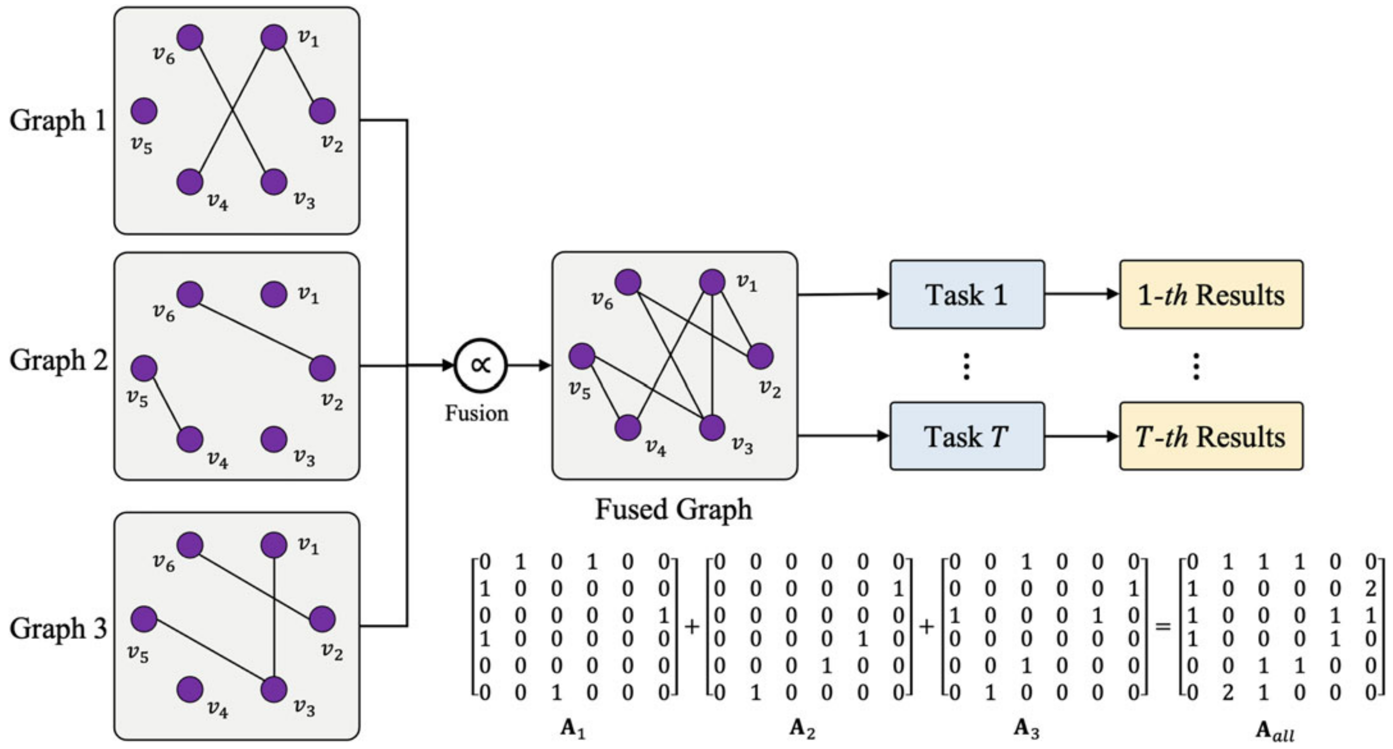
Different from the processing method in graph, hypergraph can handle such types of different connections in an easy and direct way, due to its flexible hyperedges. As shown in Fig. 2.11, when there are multiple types of connections available, it is possible to generate multiple hyperedge groups with m incidence matrices $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_m$, and these m incidence matrices can be directly concatenated to generate the overall hypergraph structure $\mathbf{H}$. In this way, all these multi-modal data or multiple types of connections can be easily modeled in one hypergraph and all further processing can be directly deployed on this hypergraph structure. Under such circumstances, it is not required to conduct multi-modal fusion in an explicit way, while it could be jointly included in the hypergraph computation process.

2.3.3 *Structure Transformation from Hypergraph to Graph*

A hypergraph can encode high-order data correlation (beyond pairwise) using its degree-free hyperedges compared to a simple graph, where the degree for all edges has to be 2. In a sense, a simple graph can be viewed as a special case, where all hyperedges on a hypergraph are of degree 2. Therefore, hypergraph and graph are interconvertible. Currently, there are a number of methods for converting a hypergraph to a simple graph. The common ones are clique expansion, star expansion, and line expansion, which are shown in Figs. 2.12, 2.13 and 2.14, respectively.

(1) Clique Expansion
Figure 2.12 shows an example of transforming a hypergraph to a graph with clique expansion. The clique expansion algorithm constructs a graph $\mathcal{G}^x(\mathcal{V}, E^x)$ from the original hypergraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ by replacing each hyperedge e with edges, whose degree is 2, for each pair (u, v) of vertices in the hyperedge [17]: $\mathcal{E}^x = \{(u, v) : u, v \in e, e \in \mathcal{E}\}$.

It is interesting to note that the vertices in hyperedge e form a clique in the graph $\mathcal{G}^x$, exactly where the name comes from. $\mathcal{G}^x$ preserves the structure of the vertices of $\mathcal{G}$, so that the information on the edges needs to be reduced as far as possible to



$$\underbrace{\begin{bmatrix}0&1&0&1&0&0\\1&0&0&0&0&0\\0&0&0&0&0&1\\1&0&0&0&0&0\\0&0&0&0&0&0\\0&0&1&0&0&0\end{bmatrix}}_{\mathbf{A}_1}+\underbrace{\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&1\\0&0&0&0&0&0\\0&0&0&0&1&0\\0&0&0&1&0&0\\0&1&0&0&0&0\end{bmatrix}}_{\mathbf{A}_2}+\underbrace{\begin{bmatrix}0&0&1&0&0&0\\0&0&0&0&0&1\\1&0&0&0&1&0\\0&0&0&0&0&0\\0&0&1&0&0&0\\0&1&0&0&0&0\end{bmatrix}}_{\mathbf{A}_3}=\underbrace{\begin{bmatrix}0&1&1&1&0&0\\1&0&0&0&0&2\\1&0&0&0&1&1\\1&0&0&0&1&0\\0&0&1&1&0&0\\0&2&1&0&0&0\end{bmatrix}}_{\mathbf{A}_{all}}$$

Fig. 2.9 An example of the graph structure fusion for the multi-modal data

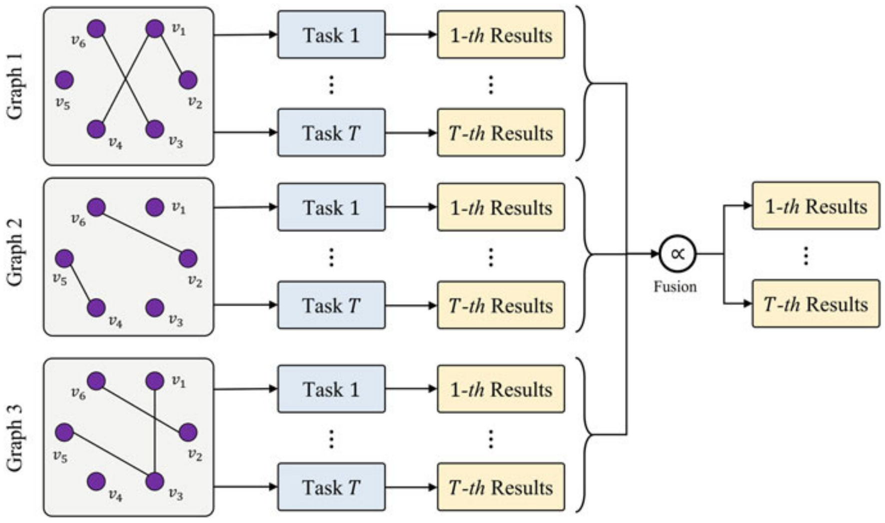


Fig. 2.10 An example of the results fusion for the multi-modal data

the higher order associations of the hyperedges. That is, the difference between the weights of any two edges that contains both u and v on $\mathscr{G}^x$ and the weights of the hyperedge connections should be as small as possible. Thus we use the following formula when assigning weights $w^x(u, v)$ to edges on $\mathscr{G}^x$:

$$w^x(u, v) = \underset{w^x(u,v)}{\arg\min} \sum_{e \in \mathscr{E}: u, v \in e} \left(w^x(u, v) - w(e)\right)^2 . \tag{2.7}$$

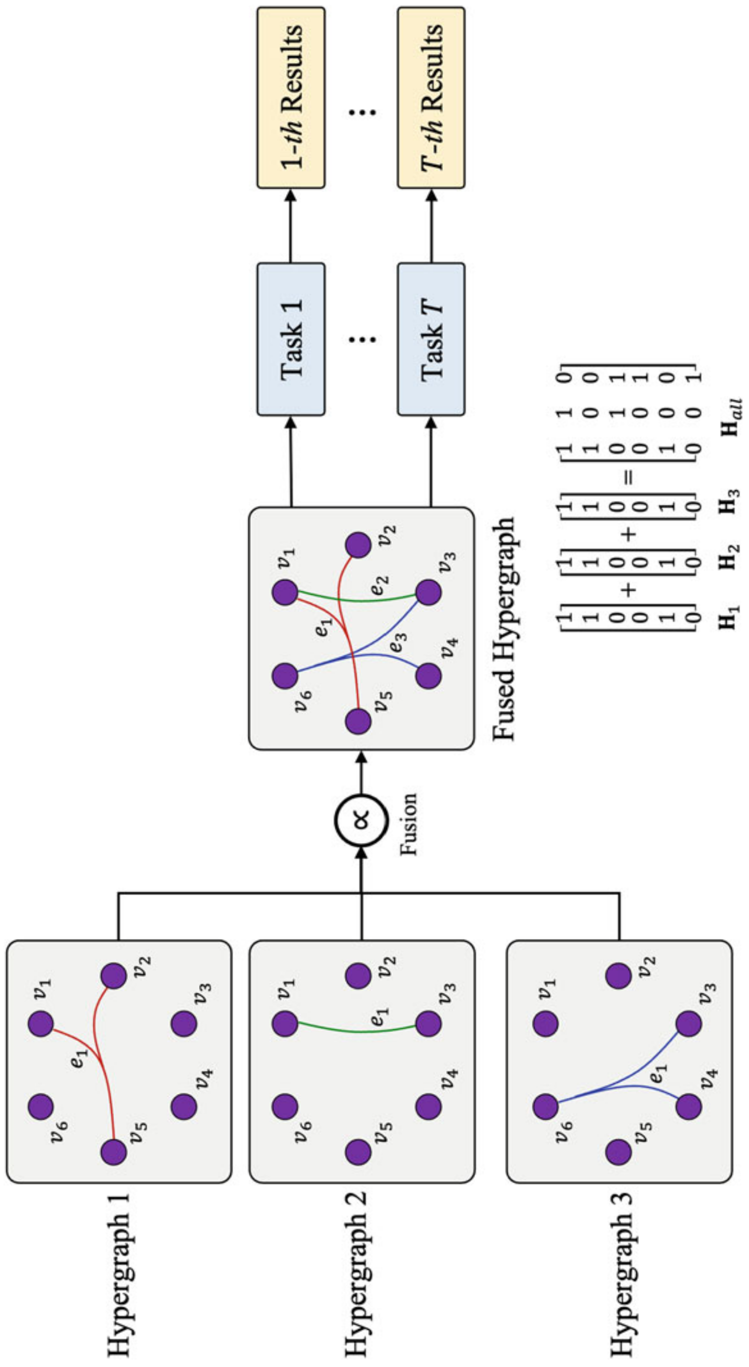
Hence, clique expansion uses the discriminative model, where every edge in the clique of $\mathscr{G}^x$ associated with hyperedge e has weight $w(e)$. This criterion has the following minimizer:

$$w^x(u, v) = \mu \sum_{e \in \mathscr{E}: u, v \in e} w(e) = \mu \sum_{e} h(u, e)h(v, e)w(e), \tag{2.8}$$

where μ is a fixed scalar. Equivalently, from the point of view of edges, the weight between two vertices u and v is derived from the sum of the weights assigned by the hyperedge that contains all of them simultaneously.

(2) Star Expansion

Figure 2.13 shows an example of transforming a hypergraph to a graph with star expansion. By star expansion, a graph $\mathscr{G}^*$ $(\mathscr{V}^*, \mathscr{E}^*)$ can be constructed from hypergraph $\mathscr{G}(\mathscr{V}, \mathscr{E})$ by regarding every hyperedge $e \in \mathscr{E}$ as a new vertex, thus $\mathscr{V}^* = \mathscr{V} \cup \mathscr{E}$ [17]. Each vertex in the hyperedge is connected to the new graph vertex e, i.e., $\mathscr{E}^* = \{(u, e) : u \in e, e \in \mathscr{E}\}$.



$$\begin{bmatrix}1\\1\\0\\0\\1\\0\end{bmatrix}+\begin{bmatrix}1\\1\\0\\0\\1\\0\end{bmatrix}+\begin{bmatrix}1\\1\\0\\0\\1\\0\end{bmatrix}=\begin{bmatrix}1&1&0\\1&0&0\\0&1&1\\0&0&1\\1&0&0\\0&0&1\end{bmatrix}$$

$$\mathbf{H}_1 \quad \mathbf{H}_2 \quad \mathbf{H}_3 \quad \mathbf{H}_{all}$$

Fig. 2.11 An example of the hypergraph structure fusion for the multi-modal data

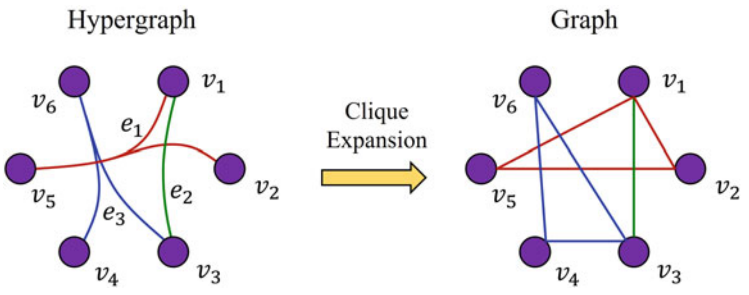


Fig. 2.12 An example of transforming a hypergraph to a graph with clique expansion

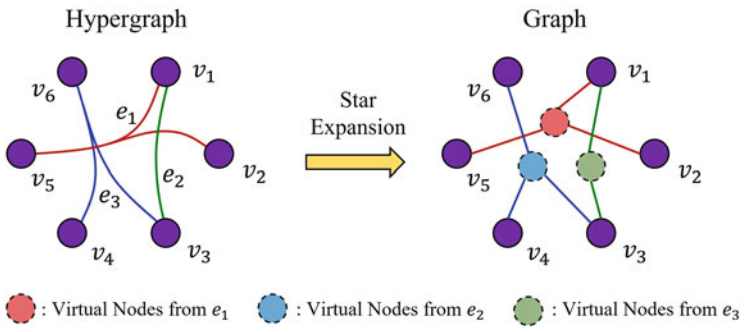

Hypergraph
Graph
Star Expansion
v_1
v_2
v_3
v_4
v_5
v_6
e_1
e_2
e_3
: Virtual Nodes from e_1
: Virtual Nodes from e_2
: Virtual Nodes from e_3

Fig. 2.13 An example of transforming a hypergraph to a graph with star expansion

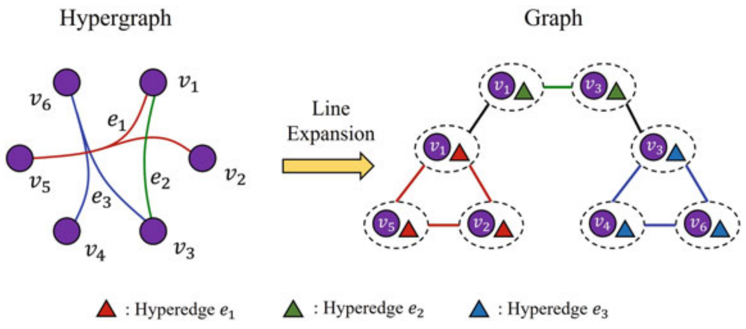


Fig. 2.14 An example of transforming a hypergraph to a graph with line expansion

There are different types of vertices in graph $\mathscr{G}^*$ and each hyperedge in $\mathscr{E}$ corresponds to a star in graph G. With star expansion, the scaled hyperedge weight is assigned to each graph edge $w^*(u, e)$ that corresponds to each hyperedge in $\mathscr{E}$ as follows:

$$w^*(u, e) = w(e)/\delta(e). \tag{2.9}$$

For each vertex representing a hyperedge, the weights of edges connecting to it are equivalent for equally dividing the superside weights into $|\delta(e)|$ parts.

(3) Line Expansion
Figure 2.14 shows an example of transforming a hypergraph to a graph with line expansion. In the case of line expansion algorithm, the vertices of the graph $\mathcal{G}^l = \left(\mathcal{V}^l, \mathcal{E}^l\right)$ are constructed by reconstructing the structure of the data stored in the vertices of the hypergraph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each line vertex (u, e) in $\mathcal{G}^l$ can be viewed as a vertex in a context of a hyperedge or a hyperedge in a context of a vertex [18]. For each point on each hyperedge, a vertex is created to represent it. The vertex v in the line expended graph indicates the property of the vertex in the hyperedge, to each vertex in the hyperedge to it, i.e., $\mathcal{V}^* = \{(u, e) : u \in e, u \in \mathcal{V}, e \in \mathcal{E}\}$. This means that $\left|\mathcal{V}^l\right| = \sum_e \delta(e)$.

Therefore the vertexes in $\mathcal{G}^l$, which contain the same vertex or the same hyperedge, can be defined as the neighborhood. Consider both connections to be equally important, so $W^l = diag(1, \ldots, 1)$, $|W^l| = |\mathcal{V}^l| \times |\mathcal{V}^l|$. The mapping between a hypergraph $\mathcal{G}$ and its line expansion $\mathcal{G}^l$ is bijective under the construction.

2.3.4 Random Walks on Graph and Hypergraph

Random walks propagate the information stored in the vertices based on the links among the vertices in the graph or hypergraph. These links constitute the path of different vertices. In the hypergraph, each vertex's neighbor vertex messages are aggregated to update itself based on the "path" between the central vertex and each vertex in its neighborhood. A hypergraph's path between vertices v_1 and v_k is defined as a sequence, called hyperpath [19]:

$$P(v_1, v_k) = (v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_k, v_k), \tag{2.10}$$

where v_j and v_{j+1} are both part of the same vertex subset described by a hyperedge e_j. We say that a hyperpath separates two neighboring vertices by a hyperedge. In a hypergraph, messages between vertices are propagated through hyperedges, which are higher-order relationships than those in graphs. It is first necessary to extend the Neighbor Relation definition among vertices to the Inter-Neighbor Relation N over vertex set $\mathcal{V}$ and hyperedge set $\mathcal{E}$ for message propagation from vertex to hyperedge and hyperedge to hyperedge on the hyperpath.

Definition 1 The Inter-Neighbor Relation $N \subset \mathcal{V} \times \mathcal{E}$ on a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with incidence matrix $\mathbf{H} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{E}|}$ is defined as

$$N = \{\, (v, e) \mid \mathbf{H}(v, e) = 1,\ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\}. \tag{2.11}$$

The hyperedge inter-neighbor set $N_e(v)$ of vertex v and the vertex inter-neighbor set $N_v(e)$ of hyperedge e are defined based on the Inter-Neighbor Relation.

Definition 2 The hyperedge inter-neighbor set of vertex $v \in \mathcal{V}$ is defined as

$$N_e(v) = \{\, e \mid vNe,\ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\}. \tag{2.12}$$

Definition 3 The vertex inter-neighbor set of hyperedge $e \in \mathcal{E}$ is defined as

$$N_v(e) = \{\, v \mid vNe,\ v \in \mathcal{V} \text{ and } e \in \mathcal{E} \,\}. \tag{2.13}$$

With hypergraph learning, in contrast to graph learning, data are correlated at a higher level, and correlation models are expanded to a high level, resulting in improved performance in practice. This is just an apparent part of the nature of graph and hypergraph. Next, we delve deeper into the relationship between graphs and hypergraph from the point of view of mathematical derivations with the help of random walks [20] and Markov chain [21]. We then provide a mathematical comparison between hypergraph and graph. The proof concludes that, from random walks' aspect, a hypergraph with edge-independent vertex weights is equivalent to a weighted graph, and a hypergraph with edge-dependent vertex weights cannot be reduced to a weighted graph.

Two types of hypergraphs can be constructed to accurately represent real-world correlations, that is, hypergraph with vertex weights independent of edge and hypergraph with vertex weights dependent on edge. By using the binary hypergraph incidence matrix $\mathbf{H} \in \{0, 1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, where vertices in each hyperedge share the same weight, hypergraph with edge-independent vertex weights ($\mathcal{G}_{\text{in}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$) can model beyond pairwise correlations. Alternatively, the weighted hypergraph incidence matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ is used to model the variable correlation intensity in each hyperedge for the hypergraph with edge-dependent vertex weights ($\mathcal{G}_{\text{de}} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}, \gamma\}$). We assume that hyperedge e includes vertex v, where $\gamma_e(v)$ denotes the connection intensity and $w(e)$ the weight of hyperedge e.

In hypergraph with edge-independent vertex weights, the definition of binary hypergraph incidence matrix $\mathbf{H}$, vertex degree $d(v)$, and hyperedge degree $\delta(e)$ is the same as in Sect. 2.1. In hypergraph with edge-dependent vertex weights, define the $d(v)$ and $\delta(e)$ as follows:

$$\begin{cases} d(v) = \sum\limits_{\beta \in \mathcal{N}_e(v)} w(\beta) \\ \delta(e) = \sum\limits_{\alpha \in \mathcal{N}_v(e)} \gamma_e(\alpha), \end{cases} \tag{2.14}$$

where $\mathcal{N}_v(\cdot)$ and $\mathcal{N}_e(\cdot)$ are defined in Eqs. (2.12) and (2.13), respectively.

Then, we will introduce the random walks and the Markov chain in hypergraph. First, we define the random walk in a hypergraph following papers [20–23]. At time t, a random walker at vertex v_t does the following:

- Pick an edge e containing vertex $v_t = v$, with probability $p_{v\to e}$.
- Pick vertex u from e, with probability $p_{e\to u}$.
- Move to vertex $v_{t+1} = u$, at time $t + 1$.

We then define the transition probability $p_{v,u}$ of the corresponding Markov chain on $\mathscr{V}$ as $p_{v,u} = \sum_{e\in\mathscr{N}_e(v,u)} p_{v\to e}p_{e\to u}$, where $\mathscr{N}_e(v,u) = \mathscr{N}_e(v) \cap \mathscr{N}_e(u)$ denotes the hyperedge $\beta \in \mathscr{N}_e(v,u)$ containing vertices v and u, simultaneously. In hypergraph with edge-independent vertex weights, we have $p_{v\to e} = w(e)/d(v)$ and $p_{e\to u} = 1/\delta(e)$. The transition probability $p_{v,u}$ can be written as $p_{v,u} = \sum_{\beta\in\mathscr{N}_e(v,u)} \frac{w(\beta)}{d(v)} \cdot \frac{1}{\delta(\beta)}$. In hypergraph with edge-dependent vertex weights, we have $p_{v\to e} = w(e)/d(v)$ and $p_{e\to u} = \gamma_e(u)/\delta(e)$, and the transition probability $p_{v,u}$ can be written as $p_{v,u} = \sum_{\beta\in\mathscr{N}_e(v,u)} \frac{w(\beta)}{d(v)} \cdot \frac{\gamma_\beta(u)}{\delta(\beta)}$.

The following lemmas and definitions are used to compare the graph and the two types of hypergraphs [21].

Definition 4 Let M be a Markov chain with state space X and transition probability $p_{x,y}$, for $x, y \in S$. It can be said that M is reversible if there exists a probability distribution π over S such that $\pi_x p_{x,y} = \pi_y p_{y,x}$.

Lemma 5 *Let M be an irreducible Markov chain with finite state space S and transition probability $p_{x,y}$ for $x, y \in S$. M is reversible if and only if there exists a weighted undirected graph $\mathscr{G}$ with vertex set S such that random walks on $\mathscr{G}$ and M are equivalent.*

Proof of Lemma 5 Note that π indicates the stationary distribution [21, 24] of a given edge-independent/edge-dependent hypergraph. The transition probability $p_{v,u}$ of vertices in hypergraph with edge-independent vertex weights is defined as

$$p_{v,u} = \sum_{\beta\in\mathscr{N}_e(v,u)} \left(\frac{w(\beta)}{d(v)}\right)\left(\frac{1}{\delta(\beta)}\right). \tag{2.15}$$

Moreover, the transition probability $p_{v,u}$ of vertices in hypergraph with edge-dependent vertex weights is defined as

$$p_{v,u} = \sum_{\beta\in\mathscr{N}_e(v,u)} \left(\frac{w(\beta)}{d(v)}\right)\left(\frac{\gamma_\beta(u)}{\delta(\beta)}\right). \tag{2.16}$$

"$\Rightarrow$": Suppose M is reversible with transition probability $p_{x,y}$. We then construct a graph $\mathscr{G}$ with vertex set S and edge weights $w_{x,y} = \pi_x p_{x,y}$. Because M is irreducible, $\pi_x \neq 0$ and $p_{x,y} \neq 0$ for all states x and y. Thus, the edge weight $w_{x,y} \neq 0$ and the graph $\mathscr{G}$ are a connected graph. Due to the reversibility of M that $w_{x,y} = \pi_x p_{x,y} = \pi_y p_{y,x} = w_{y,x}$, the constructed graph $\mathscr{G}$ is an undirected graph. Random walks on $\mathscr{G}$ from x to y in one-time step satisfy the following:

$$\frac{w_{x,y}}{\sum_{z\in S} w_{x,z}} = \frac{\pi_x p_{x,y}}{\sum_{z\in S} \pi_x p_{x,z}} = \frac{p_{x,y}}{\sum_{z\in S} p_{x,z}} = p_{x,y}, \tag{2.17}$$

since $\sum_{z \in S} p_{x,z} = 1$. Thus, if M is reversible, the stated claim holds.
"$\Leftarrow$": Random walks on an undirected graph are always reversible.

Definition 6 A Markov chain is reversible if and only if its transition probability satisfies

$$p_{v_1,v_2} p_{v_2,v_3} \cdots p_{v_n,v_1} = p_{v_1,v_n} p_{v_n,v_{n-1}} \cdots p_{v_2,v_1} \tag{2.18}$$

for any finite sequence of states $v_1, v_2, \cdots v_n \in S$. The definition is also known as Kolmogorov's criterion. For more detailed proofs, please refer to [25].

Theorem 1 *Let $\mathcal{G}_{in} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ be a hypergraph with edge-independent weights, and then there exists a weighted undirected graph $\mathcal{G}$ such that a random walk on $\mathcal{G}$ is equivalent to a random walk on $\mathcal{G}_{in}$.*

Proof of Theorem 1 The probability $p_{v,u}$ of $\mathcal{G}_{\text{in}}$ is defined in Eq. (2.15). By Definition 6, the following equation can be deduced:

$$\begin{aligned}
& p_{v_1,v_2} p_{v_2,v_3} \cdots p_{v_n,v_1} \\
&= \sum_{\beta \in \mathcal{N}_e(v_1,v_2)} \left(\frac{w(\beta)}{d(v_1)} \cdot \frac{1}{\delta(\beta)} \right) \cdots \sum_{\beta \in \mathcal{N}_e(v_n,v_1)} \left(\frac{w(\beta)}{d(v_n)} \cdot \frac{1}{\delta(\beta)} \right) \\
&= \left(\frac{1}{d(v_1)} \sum_{\beta \in \mathcal{N}_e(v_1,v_2)} \frac{w(\beta)}{\delta(\beta)} \right) \cdots \left(\frac{1}{d(v_n)} \sum_{\beta \in \mathcal{N}_e(v_n,v_1)} \frac{w(\beta)}{\delta(\beta)} \right) \\
&= \frac{1}{d(v_2)} \sum_{\beta \in \mathcal{N}_e(v_1,v_2)} \frac{w(\beta)}{\delta(\beta)} \cdots \frac{1}{d(v_1)} \sum_{\beta \in \mathcal{N}_e(v_n,v_1)} \frac{w(\beta)}{\delta(\beta)}.
\end{aligned} \tag{2.19}$$

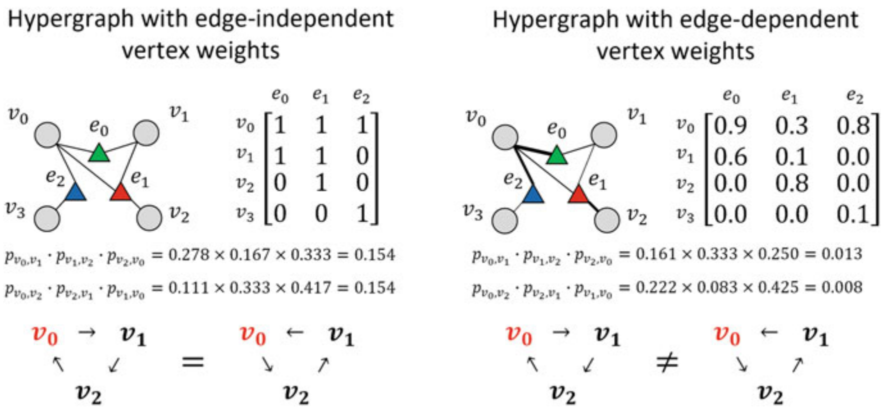
For any v_i and v_j, $\sum_{\beta \in \mathcal{N}_e(v_i,v_j)} \frac{w(\beta)}{\delta(\beta)} = \sum_{\beta \in \mathcal{N}_e(v_j,v_i)} \frac{w(\beta)}{\delta(\beta)}$. Thus, the reversibility can be proven by

$$\begin{aligned}
& p_{v_1,v_2} p_{v_2,v_3} \cdots p_{v_n,v_1} \\
&= \frac{1}{d(v_2)} \sum_{\beta \in \mathcal{N}_e(v_2,v_1)} \frac{w(\beta)}{\delta(\beta)} \cdots \frac{1}{d(v_1)} \sum_{\beta \in \mathcal{N}_e(v_1,v_n)} \frac{w(\beta)}{\delta(\beta)} \\
&= p_{v_2,v_1} p_{v_3,v_2} \cdots p_{v_1,v_n} \\
&= p_{v_1,v_n} p_{v_n,v_{n-1}} \cdots p_{v_2,v_1}.
\end{aligned} \tag{2.20}$$

We say that a random walk on $\mathcal{G}_{\text{in}}$ is reversible. Furthermore, by Lemma 5, a random walk on $\mathcal{G}_{\text{in}}$ is equivalent to a random walk on a weighted undirected graph $\mathcal{G}$.

The proof of Theorem 1 can be processed as follows:

1. A random walk on $\mathcal{G}_{\text{in}}$ is equivalent to a random walk on a reversible Markov chain (according to **Definition 6**).


Hypergraph with edge-independent vertex weights
Hypergraph with edge-dependent vertex weights
v_0 v_1 v_2 v_3 e_0 e_1 e_2
$p_{v_0,v_1} \cdot p_{v_1,v_2} \cdot p_{v_2,v_0} = 0.278 \times 0.167 \times 0.333 = 0.154$
$p_{v_0,v_2} \cdot p_{v_2,v_1} \cdot p_{v_1,v_0} = 0.111 \times 0.333 \times 0.417 = 0.154$
$p_{v_0,v_1} \cdot p_{v_1,v_2} \cdot p_{v_2,v_0} = 0.161 \times 0.333 \times 0.250 = 0.013$
$p_{v_0,v_2} \cdot p_{v_2,v_1} \cdot p_{v_1,v_0} = 0.222 \times 0.083 \times 0.425 = 0.008$
=
≠

$$\begin{array}{c|ccc} & e_0 & e_1 & e_2 \\ \hline v_0 & 1 & 1 & 1 \\ v_1 & 1 & 1 & 0 \\ v_2 & 0 & 1 & 0 \\ v_3 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|ccc} & e_0 & e_1 & e_2 \\ \hline v_0 & 0.9 & 0.3 & 0.8 \\ v_1 & 0.6 & 0.1 & 0.0 \\ v_2 & 0.0 & 0.8 & 0.0 \\ v_3 & 0.0 & 0.0 & 0.1 \end{array}$$

Fig. 2.15 An example of two types of random walks on the hypergraph with edge-independent vertex weights and the hypergraph with edge-dependent vertex weights. This figure is from [26]

2. A random walk on a reversible Markov chain is equivalent to a random walk on a weighted undirected graph $\mathscr{G}$ (according to **Lemma 5**).

Theorem 2 *Let $\mathscr{G}_{de} = \{\mathscr{V}, \mathscr{E}, \mathbf{W}, \gamma\}$ be a hypergraph with edge-dependent weights, and then there does not exist a weighted undirected graph $\mathscr{G}$ such that a random walk on $\mathscr{G}$ is equivalent to a random walk on $\mathscr{G}_{de}$.*

Proof of Theorem 2 Figure 2.15 provides an example that a random walk on $\mathscr{G}_{\mathrm{de}}$ is not equivalent to a random walk on a reversible Markov chain. According to the second step of **Theorem 1**'s proof, **Theorem 2** holds.

A simple illustration is shown in Fig. 2.15 to make it easier to understand. There is no difference in the connection structure between the two hypergraphs, but there is a difference in the intensity of the connections. For two types of hypergraphs, the transition probabilities $p_{v,u}$ can be computed accordingly. As a consequence, two random walks from vertex v_0 are conducted: "$v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_0$" and "$v_0 \rightarrow v_2 \rightarrow v_1 \rightarrow v_0$." Having obtained $p_{v_0,v_1} \cdot p_{v_1,v_2} \cdot p_{v_2,v_0}$ and $p_{v_0,v_2} \cdot p_{v_2,v_1} \cdot p_{v_1,v_0}$ for the two paths, the cumulative transition probability can then be calculated. This type of hypergraph is reversible according to **Theorem 1** and **Lemma 5**. Thus, from the two reversible paths, the same accumulated transition probability can be obtained. Alternatively, two different accumulated transition probabilities are obtained from two reversible paths in the hypergraph with edge-independent vertex weights.

2.4 Summary

In this chapter, we present the mathematical definition of the foundations of hypergraph and their interpretation. We then also show the representation of directed hypergraph, different from undirected hypergraph, which represents the relationships between vertices within a hyperedge. Finally, we discuss the relationship between graph and hypergraph in conversions and expressive ability perspectives. The most intuitive differences between graph and hypergraph can be seen in low-order versus high-order representations and adjacency matrix versus incidence matrix. Clique expansion, star expansion, and line expansion are methods for converting hypergraph into simple graph. We also show the relationship between graph and hypergraph from the random walk view. A hypergraph with edge-independent vertex weights is equivalent to a weighted graph, and a hypergraph with edge-dependent vertex weights cannot be reduced to a weighted graph from the information propagation process on graph/hypergraph.

References

1. Y. Huang, Q. Liu, S. Zhang, D.N. Metaxas, Image retrieval via probabilistic hypergraph ranking, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 3376–3383
2. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
3. Y. Huang, Q. Liu, D. Metaxas, Video object segmentation by hypergraph cut, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1738–1745
4. W. Zhao, S. Tan, Z. Guan, B. Zhang, M. Gong, Z. Cao, Q. Wang, Learning to map social network users by unified manifold alignment on hypergraph. IEEE Trans. Neural Netw. Learn. Syst. **29**(12) 5834–5846 (2018)
5. F. Luo, B. Du, L. Zhang, L. Zhang, D. Tao, Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image. IEEE Trans. Cyber. **49**(7), 2406–2419 (2018)
6. L. Zhu, J. Shen, H. Jin, R. Zheng, L. Xie, Content-based visual landmark search via multimodal hypergraph learning. IEEE Trans. Cyber. **45**(12), 2756–2769 (2015)
7. D. Du, H. Qi, L. Wen, Q. Tian, Q. Huang, S. Lyu, Geometric hypergraph learning for visual tracking. IEEE Trans. Cyber. **47**(12), 4182–4195 (2017)
8. Z. Tian, T. Hwang, R. Kuang, A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. Bioinformatics. **25**(21), 2831–2838 (2009)
9. X. Zheng, W. Zhu, C. Tang, M. Wang, Gene selection for microarray data classification via adaptive hypergraph embedded dictionary learning. Gene. **706**, 188–200 (2019)
10. Y. Gao, C.-Y. Wee, M. Kim, P. Giannakopoulos, M.L. Montandon, S. Haller, D. Shen. MCI identification by joint learning on multiple MRI data, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 78–85
11. W. Shao, Y. Peng, C. Zu, M. Wang, D. Zhang, Hypergraph based multi-task feature selection for multimodal classification of Alzheimer's disease. Comput. Med. Imaging Graph. **80**, 101663 (2020)
12. E. Ramadan, S. Perincheri, D. Tuck, A hyper-graph approach for analyzing transcriptional networks in breast cancer, in *Proceedings of the ACM International Conference on Bioinformatics and Computational Biology* (2010), pp. 556–562
13. L. Xiao, J. Wang, P.H. Kassani, Y. Zhang, Y. Bai, J.M. Stephen, T.W. Wilson, V.D. Calhoun, Y. Wang, Multi-hypergraph learning based brain functional connectivity analysis in fMRI data. IEEE Trans. Med. Imaging **39**(5), 1746–1758 (2019)

14. G. Gallo, G. Longo, S. Pallottino, S. Nguyen, Directed hypergraphs and applications. Discrete Appl. Math. **42**(2–3), 177–201 (1993)
15. K. Zhan, C. Niu, C. Chen, F. Nie, C. Zhang, Y. Yang, Graph structure fusion for multiview clustering. IEEE Trans. Knowl. Data Eng. **31**(10), 1984–1993 (2018)
16. Z. Kang, G. Shi, S. Huang, W. Chen, X. Pu, J.T. Zhou, Z. Xu, Multi-graph fusion for multi-view spectral clustering. Knowl.-Based Syst. **189**, 105102 (2020)
17. Y.J. Zien, M. Schlag, P. Chan, Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. IEEE Trans. Comput.-Aided Design Integr. Circ. Syst. **18**(9), 1389–1399 (1999)
18. C. Yang, R. Wang, S. Yao, T. Abdelzaher, Hypergraph learning with line expansion (2020). Preprint arXiv:2005.04843
19. R. Dharmarajan, K. Kannan, Hyper paths and hyper cycles. Ital. J. Pure Appl. Math. **98**(3), 309–312 (2015)
20. T. Carletti, F. Battiston, G. Cencetti, D. Fanelli, Random walks on hypergraphs, Phys. Rev. E **101**(2), 022308 (2020)
21. U. Chitra, B. Raphael, Random walks on hypergraphs with edge-dependent vertex weights, in *Proceedings of the Machine Learning Research* (2019), pp. 1172–1181
22. D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: Clustering, classification, and embedding, in *Proceedings of the Advances in Neural Information Processing Systems* (2007)
23. A. Ducournau, A. Bretto, Random walks in directed hypergraphs and application to semi-supervised image segmentation. Comput. Vision Image Understand. **120**, 91–102 (2014)
24. J. Li, J. He, Y. Zhu, E-tail product return prediction via hypergraph-based local graph cut, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 519–527
25. P.F. Kelly, *Reversibility and Stochastic Networks* (Cambridge University Press, Cambridge, 2011)
26. Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN+: General hypergraph neural networks. IEEE Trans. Pattern Analy. Mach. Intell. **45**(3), 3181–3199 (2023)

Chapter 3
Hypergraph Computation Paradigms

Abstract This chapter introduces three hypergraph computation paradigms, including intra-hypergraph computation, inter-hypergraph computation, and hypergraph structure computation. Intra-hypergraph computation representation aims to conduct representation learning of a hypergraph, where each subject is represented by a hypergraph of its components. Inter-hypergraph computation is to conduct representation learning of vertices in the hypergraph, where each subject is a vertex in the hypergraph. Hypergraph structure computation is to conduct hypergraph structure prediction, which aims to find the connections among vertices. This chapter is a general introduction of hypergraph computation paradigms to show how to formulate the task in the hypergraph computation framework.

3.1 Introduction

Hypergraph computation can be roughly divided into three types: representation learning of a hypergraph, where each subject is represented by a hypergraph of its components, representation learning of vertices in the hypergraph, where each subject is a vertex in the hypergraph, and hypergraph structure prediction, which aims to find the connections among vertices. There are three types of computation paradigms that can be named intra-hypergraph computation, inter-hypergraph computation, and hypergraph structure computation. In this chapter, we introduce the generalized computation paradigms corresponding to these three directions and show how to formulate practical tasks in these hypergraph computation frameworks. We note that specific implementations of generalized functions in the paradigm are not introduced here, as they are parts of specifically defined functions or modules in the hypergraph computation framework and will be introduced in subsequent chapters.

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_3

3.2 Intra-hypergraph Computation

Intra-hypergraph computation targets on learning the representation of a single subject using the inside component information, in which the correlations among the components of this subject are formulated in a hypergraph. In this hypergraph, the components of this subject are regarded by the set of vertices, and their high-order correlations are modeled by hyperedges. In this way, the individual subject is transformed into a hypergraph. As this hypergraph is generated by the subject's components themselves, we can name this hypergraph as the *intra-hypergraph* of this subject.

Image representation and understanding [1–3] are typical intra-hypergraph computation applications. For example, an image can be split into a group of patches, and each patch is denoted by a vertex in the hypergraph. The hypergraph can be generated according to the semantic and spatial information of these patches. The information of these patches and their high-order correlations can be then used simultaneously to learn the representation for the image.

The general paradigm of intra-hypergraph computation can be described as follows. Given a target subject that contains n components, that are represented by feature vectors $\mathbf{X} \in \mathbb{R}^{n\times d}$. An intra-hypergraph $\mathcal{G}$ can be generated to formulate the high-order correlations inside the subject, whose incidence matrix is denoted by $\mathbf{H}$. The representation of the individual subject can be learned by

$$\mathbf{Z}_{\mathcal{G}} = f_{\Theta}(\mathbf{H}, \mathbf{X}), \tag{3.1}$$

where Θ denotes the to-be-learned parameters. The function $f_{\Theta}(\cdot)$ can be the neural network layers or other computing operators that aggregate the information of vertices together based on the hypergraph structure. Intra-hypergraph computation integrates the complex correlations among components into the learned representation, which can extract more information than simple aggregation operations.

In this paradigm, the subject to be analyzed is regarded as a whole system, and the intra-hypergraph is to model the correlation inside the system. This process is shown in Fig. 3.1.

3.3 Inter-hypergraph Computation

Inter-hypergraph computation targets at learning the representation of a subject by considering its correlations with other subjects. In this hypergraph, each subject, including the target one, is regarded by the set of vertices, and their high-order correlations are modeled by hyperedges. In this way, this group of subjects is transformed into a hypergraph. As this hypergraph is generated by the cross-subject correlations, we can name this hypergraph as the *inter-hypergraph* of this subject. Subject classification and retrieval [4–7] are typical inter-hypergraph computation

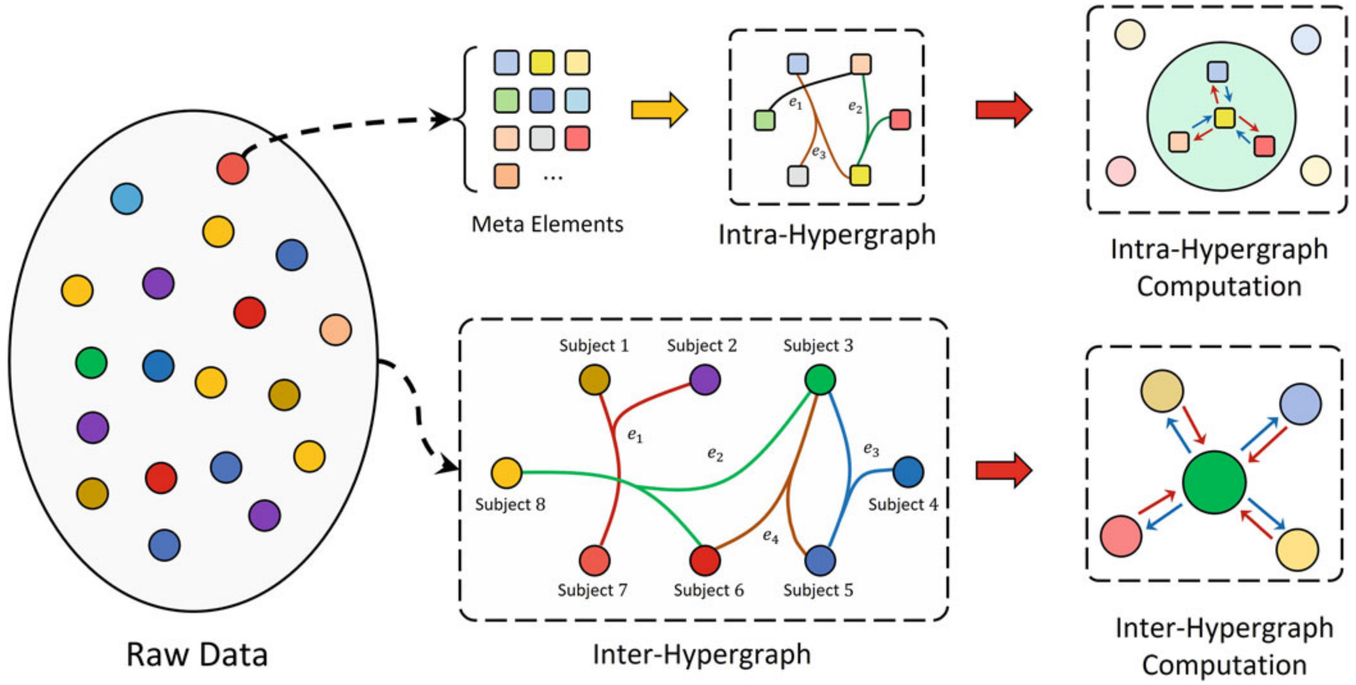


Fig. 3.1 An illustration of intra-hypergraph computation and inter-hypergraph computation

applications. For example, we take an image as the target subject, and we can also have a pool of images for processing. Each image can be denoted by a vertex in the hypergraph. The hypergraph can be generated according to the semantic and spatial information of these images. The information of these images and their high-order correlations can be then used simultaneously to learn the representation of the target image.

The general paradigm of inter-hypergraph computation can be described as follows. Given a target subject and other $n-1$ subjects, represented by feature vectors $\mathbf{X} \in \mathbb{R}^{n \times d}$, an inter-hypergraph $\mathcal{G}$ can be generated to formulate the high-order correlations among these subjects, whose incidence matrix is denoted by $\mathbf{H}$. The representation of the target subject can be learned by

$$\mathbf{Z}_{\mathcal{V}} = f_{\Theta}(\mathbf{H}, \mathbf{X}). \tag{3.2}$$

The vertex embedding can be further used for the downstream tasks, such as vertex classification, where the vertices are associated with pre-defined labels $Y \in [K]^n$. This process is also shown in Fig. 3.1.

It is noted that a hypergraph structure can be either homogeneous or heterogeneous, depending on the definition of vertices. Given multiple types of data, or multi-modal data, another way to formulate such correlations is to generate multiple hypergraphs accordingly. For example, supposing that there are m types of features or modalities, denoted by $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m$, we can construct one hypergraph for each modality respectively. In this way, we can have m hypergraphs $\mathcal{G}_1 = (\mathcal{V}_1; \mathcal{E}_1; \mathbf{W}_1)$; $\mathcal{G}_2 = (\mathcal{V}_2; \mathcal{E}_2; \mathbf{W}_2)$; ...; $\mathcal{G}_m = (\mathcal{V}_m; \mathcal{E}_m; \mathbf{W}_m)$ for the data with m modalities. The general paradigm for multi-modal inter-hypergraph computation can be described as

$$\mathbf{Z}_{\mathcal{V}} = f_{\Theta}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_m, \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m), \tag{3.3}$$

where $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_m$ are the incidence matrices of the m hypergraphs.

3.4 Hypergraph Structure Computation

Hypergraph structure computation aims to learn the high-order correlations among data in the presence of missing links and inaccurate initial structure. There are two scenarios in which hypergraph structure computation is performed: either the set of hyperedges is incomplete or the affiliation relationships between vertices and hyperedges are incomplete. Recommender system and drug discovery [8–10] are typical hypergraph structure computation applications. For example, in recommender system, the hyperedges describe the connections between items and users with specific semantics. The number of hyperedges is fixed, and the features of both vertices and hyperedges can be obtained as the input. Here, the target of hypergraph structure computation is to predict whether a vertex belongs to a hyperedge or not. If a new hyperedge is predicted, we can have new link to indicate the connections. However, in a knowledge hypergraph, the hyperedges display the facts in the real world, which are usually highly incomplete. The missing links are expected to be inferred based on existing links by hypergraph structure computation. Therefore, in the second case, the objective of hypergraph structure computation is not only optimizing existing links but also inferring the unobserved links.

In the following, we describe the computation paradigms of these two cases separately. The first scenario is that the set of hyperedges is complete and the affiliation relationships between vertices and hyperedges are incomplete. In this case, we usually can extract a feature vector for each hyperedge representation. Given the input of vertex features $\mathbf{X}_{\mathcal{V}}$ and hyperedge features $\mathbf{X}_{\mathcal{E}}$, we can calculate the incidence matrix by the function related to the vertex and hyperedge features as

$$\mathbf{H}^* = f_{\Theta}(\mathbf{X}_{\mathcal{V}}, \mathbf{X}_{\mathcal{E}}). \tag{3.4}$$

For example, the attention score can be used as an instance of the function in practice.

In the second scenario, if there are missing hyperedges in the observed hypergraph and the semantics of hyperedges are ambiguous, it is difficult to directly describe the hyperedges by features. Consequently, only the initial incomplete hypergraph structure and the features of vertices can be available as the input. We denote the incidence matrix of the initial hypergraph structure by $\mathbf{H}^{(0)}$. The computation paradigm can be written as

$$\mathbf{H}^* = f_{\Theta}(\mathbf{X}_{\mathcal{V}}, \mathbf{H}^{(0)}), \tag{3.5}$$

which indicates that the new hypergraph structure is updated based on the original hypergraph structure following specific prior information.

To guide the evolution of hypergraph structure to more accurately model data correlation, it is necessary to evaluate the quality of hypergraph structure based on the training data and prior information. If there is part of ground truth information about the hypergraph structure, the performance of correlation modeling can be evaluated directly. However, there is no golden standard for hypergraph structure in most cases. Therefore, we may need to perform downstream tasks using the new hypergraph and indirectly evaluate hypergraph computation performance through the downstream task results. Here, we refer to Fig. 3.1, and hypergraph structure computation can be conducted under the intra- and inter-hypergraph computation frameworks.

3.5 Summary

In this chapter, we introduce three hypergraph computation paradigms for different scenarios. These three paradigms are intra-hypergraph computation, inter-hypergraph computation, and hypergraph structure computation, which focus on learning the representation of a single subject using the inside component information, learning the representation of a subject by considering its correlations with other subjects, and learning the high-order correlations among data in the presence of missing links and inaccurate initial structure. This chapter provides an overview of how to use hypergraph computation, and the detailed hypergraph computation theory, methods, and application will be introduced in the following chapters.

References

1. D. Di, S. Li, J. Zhang, Y. Gao, Ranking-Based survival prediction on histopathological whole-slide images, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2020), pp. 428–438
2. D. Di, J. Zhang, F. Lei, Q. Tian, Y. Gao, Big-hypergraph factorization neural network for survival prediction from whole slide image. IEEE Trans. Image Process. **31**, 1149–1160 (2022)
3. D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, Y. Gao, Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction. IEEE Trans. Pattern Analy. Mach. Intell. 1–16 (2022). https://doi.org/10.1109/TPAMI.2022.3209652
4. Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, C. Zou, Hypergraph learning: methods and practices. IEEE Trans. Pattern Analy. Mach. Intell. **44**(5), 2548–2566 (2022)
5. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**, 4290–4303 (2012)
6. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
7. Y. Gao, M. Wang, Z.J. Zha, J. Shen, X. Li, X. Wu, Visual textual joint relevance learning for tag-based social image search. IEEE Trans. Image Process. **22**(1), 363–376 (2013)
8. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 2020–2029
9. H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, Q. Dai, Heterogeneous hypergraph variational autoencoder for link prediction. IEEE Trans. Pattern Analy. Mach. Intell. **44**(8), 4125–4138 (2021)
10. D. Ruan, S. Ji, C. Yan, J. Zhu, X. Zhao, Y. Yang, Y. Gao, C. Zou, Q. Dai, Exploring complex and heterogeneous correlations on hypergraph for the prediction of drug-target interactions. Patterns **2**(12), 100390 (2021)

Chapter 4
Hypergraph Modeling

Abstract Hypergraph modeling is the fundamental task in hypergraph computation, which targets on establishing a high-quality hypergraph structure to accurately formulate the high-order correlation among data. In this section, we introduce different hypergraph modeling methods to show how to build hypergraphs using various pieces of information, such as features, attributes, and/or graphs. These methods are organized into two broad categories, depending on whether these correlations are explicit or implicit, to distinguish the similarities and differences. We then further discuss different hypergraph structure optimization and generation methods, such as adaptive hypergraph modeling, generative hypergraph modeling, and knowledge hypergraph generation.

4.1 Introduction

Although there are complex correlations among data in many applications, it is difficult to discover such complex correlations in many cases due to the limitations of observation technologies. Taking social networks as an example, the group information is a kind of high-order correlation that connects a number of people based on certain criteria. However, it is intractable to investigate all the groups when there are millions or even billions of vertices in a social network. Another typical example is the human brain network. Apparently, some functions of the brain are implemented by the communications among multiple brain regions rather than just two regions, which means that there exist high-order correlations among brain regions. Nevertheless, much manpower and material resources would be required to directly record such high-order correlations by neuroscience experiments. Therefore, it is necessary to study how to model such high-order correlations based on existing information in practical applications.

Hypergraph has shown its superiority on high-order correlation modeling. Hypergraph structure generation has attracted much attention and is still an open problem due to complex correlations among non-standard data. In this chapter, we systematically review the existing hypergraph modeling methods, including both the implicit hypergraph modeling strategy and the explicit hypergraph modeling

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_4

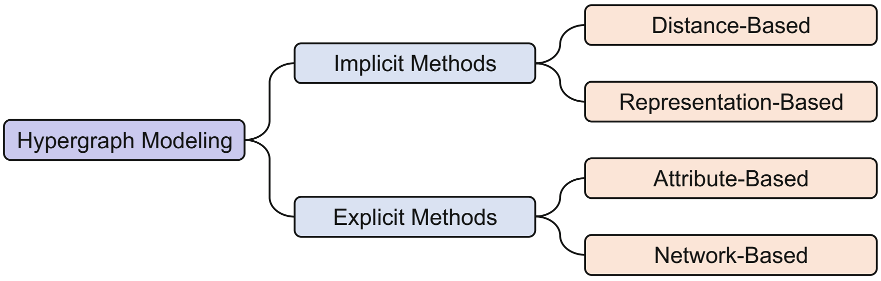


Fig. 4.1 Different categories of hypergraph modeling methods

strategy. The implicit hypergraph modeling strategy aims to generate the hypergraph structure using vertex representations based on their distances or similarities, in which the correlations are not directly provided. The explicit hypergraph modeling strategy targets at the data with explicit high-order correlation information, such as attributes and pairwise connections. For the implicit hypergraph modeling strategy, we mainly introduce the distance-based and representation-based hypergraph structure generation methods. For the explicit hypergraph modeling strategy, we focus on the attribute-based and the network-based hypergraph generation approaches. Figure 4.1 illustrates the hypergraph modeling methods.

We further give four examples in computer vision, recommender system, computer-aided diagnosis, and brain network for hypergraph modeling in this chapter. In the last part, we discuss the topics of further research of hypergraph modeling, which have the potential of going beyond the limitations of current methods that are difficult to be adaptive to complex data. Part of the work introduced in this chapter has been published in [1–4].

4.2 Implicit Hypergraph Modeling

In implicit hypergraph modeling, the correlations among data are not directly provided. Under such circumstances, we need to explore different representations of the data to build the correlations. Two typical methods for implicit hypergraph modeling are distance-based methods and representation-based methods. In distance-based methods, we can explore the neighborhood information for each sample in some specific feature spaces, and the samples with high similarity/low distance can be connected by a corresponding hyperedge. In representation-based methods, the representation among different feature vectors for the samples is used to measure the neighborhood information, which can be used to generate hyperedges.

4.2.1 Distance-Based Hypergraph Generation

Distance-based hypergraph generation methods construct the hyperedges based on the distances in the feature space for all the vertices. In general, the construction of the hypergraph can be divided into two steps: the incidence matrix generation and the hyperedge weight generation. For the incidence matrix generation, the connectivity on the hypergraph, i.e., the hyperedge, is determined with the consideration of the neighborhood relationships, where the neighbors of the vertices in the feature space are connected by these hyperedges. For the hyperedge weight generation, the weights of these hyperedges are calculated based on the distance information.

The incidence matrix is generated based on the neighbors of the vertices. There are two major approaches to determine the neighbors [1], i.e., the nearest-neighbor-based hyperedge generation strategy (shown in Fig. 4.2) and the clustering-based hyperedge strategy (shown in Fig. 4.3). The nearest-neighbor-based hyperedge generation strategy searches the nearest vertices for the given vertex, i.e., the centroid, and connects these vertices by the hyperedges. The clustering-based hyperedge generation strategy groups the vertices with the features and constructs a hyperedge to connect all vertices fallen into the same cluster.

The nearest-neighbor-based hyperedge generation strategy starts out with calculating the distances between all pairs of vertices in the feature space. Subsequently,

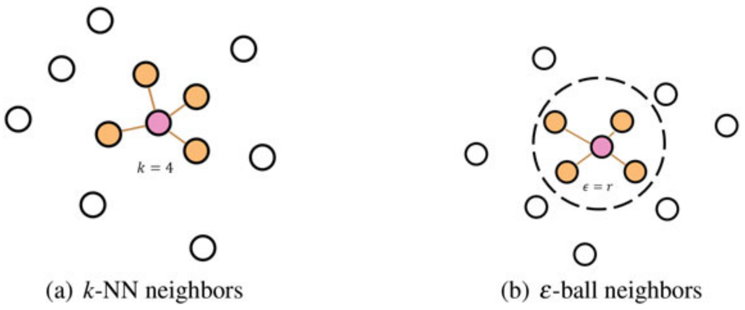


Fig. 4.2 An illustration of the nearest-neighbor-based hyperedge generation strategy. (**a**) shows the k-NN neighbors of the given vertex, and (**b**) shows the ϵ-ball neighbors

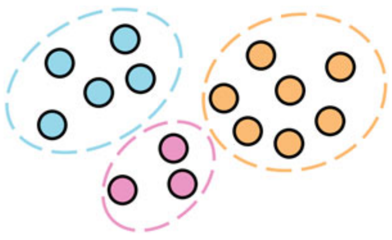


Fig. 4.3 Illustration of the cluster-based hyperedge generation strategy. This figure is from [1]

two commonly used criteria [5] are applied to determine the neighbors of the given centroid, i.e., the k-NN neighbors [6] and the ϵ-ball neighbors [2]. The given centroid and the selected neighbors are connected together by a hyperedge.

Here we denote $\mathcal{V}$ as the vertices set, $u \in \mathcal{V}$ as the given centroid, $X(u)$ as the feature vector of u, $d(x_1, x_2) = ||x_1 - x_2||_2$ as the Euclidean distance between the vectors x_1 and x_2, $\mathcal{N}_k(u)$ as the k-NN neighbors set of u, and $\mathcal{N}_\epsilon(u)$ as the ϵ-ball neighbors set of u. $\mathcal{N}_k(u)$ contains k vertices with the smallest distance to u, while $\mathcal{N}_\epsilon(u)$ contains the vertices with distance smaller than ϵ, i.e.,

$$\mathcal{N}_\epsilon(u) = \{v | d(X(u), X(v)) \le \epsilon\}. \tag{4.1}$$

The vertex u and the neighbors $\mathcal{N}(u)$ (either $\mathcal{N}_k(u)$ or $\mathcal{N}_\epsilon(u)$) are grouped together to generate a hyperedge $e(u)$:

$$e(u) = \mathcal{N}(u) \cup \{u\}, \tag{4.2}$$

and the hyperedge set $\mathcal{E}$ is formulated as

$$\mathcal{E} = \{e(u) | u \in \mathcal{V}\}. \tag{4.3}$$

The clustering-based hyperedge generation strategy starts out with grouping the vertices according to the corresponding features using the clustering algorithms, such like k-means. Subsequently, the vertices belonging to the same cluster are connected together using a hyperedge. Here we assume that the k-means algorithm clusters the vertex set $\mathcal{V}$ into K groups $\mathcal{V}_1, \ldots, \mathcal{V}_K$. Then, K hyperedges can be constructed using these clustering results:

$$\forall 1 \le k \le K, e_k = \mathcal{V}_k = \{v_{k_1}, v_{k_2}, \ldots\}, \tag{4.4}$$

and the hyperedge set $\mathcal{E}$ is formulated as

$$\mathcal{E} = \{e_k | \forall 1 \le k \le K\}. \tag{4.5}$$

Besides the similarity/distance in the feature space, other types of information, which can be used to measure the correlation in some specific space, such as the spatial information, can also be applied for hyperedge generation. For example, the spatial information of pixels in an image can be used to select a group of neighbor pixels for one centroid, which can be connected by a hyperedge, as shown in Fig. 4.4.

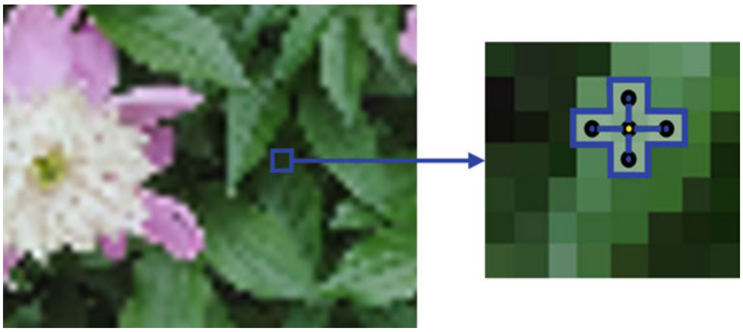


Fig. 4.4 An illustration of using spatial information of pixels to generate a hyperedge

Typically, an incidence matrix **H** is used to represent the structure of the hypergraph, i.e.,

$$\mathbf{H}_{ue} = \begin{cases} 1 & \text{if } u \in e \\ 0 & \text{otherwise} \end{cases}, \tag{4.6}$$

where $u \in \mathscr{V}$ and $e \in \mathscr{E}$.

The weight matrix of the hypergraph represents the importance of each hyperedge. A commonly used method for the hyperedge weight measurement is based on the Gaussian kernel, where the scores of each pair of vertices belonging to the same hyperedge are calculated using the distance between the vertices in the pair and the average score can be used as the weight of the hyperedge, i.e.,

$$w(e) = \sum_{u,v \in e} \exp\left(-\frac{d(X(u), X(v))}{\sigma^2}\right), \tag{4.7}$$

where $w(e)$ denotes the weight of hyperedge e, and σ is the band width of the Gaussian kernel.

In this way, if the vertices connected by a hyperedge are with relatively higher similarity, the corresponding hyperedge weight could be larger and vice versa. Then, the hyperedge weights can represent whether this hyperedge is trustable for further processing.

In practice, σ can be set as the median value of the distances among all vertices by

$$\sigma = \text{median}_{u,v \in \mathscr{V}} d\left(X\left(u\right), X\left(v\right)\right), \tag{4.8}$$

where median denotes the median value. It is noted that the hyperedge weight can be set in other ways following the purpose of evaluating the importance of each hyperedge.

The main limitation of the distance-based hypergraph generation method is the inaccurate distances due to noise and outliers of data, which may further introduce noise to the structure of hypergraphs. In practice, the feature representation for the data is still a challenging task. It is not easy to conduct effective feature extraction under certain application scenario. The metric for distance calculation also matters. Although the Euclidean distance is commonly used, there still exist some other metrics, such as the L_1-norm and the negative cosine distance. The decision making of these metrics requires experimental evaluation. Therefore, the distance-based hypergraph generation method may suffer under such circumstances.

The nearest-neighbor-based hyperedge generation strategy is the most simple one to be deployed in practice. The limitations of this strategy are as follows. First, the hyperparameter, i.e., k for the k-NN neighbors and ϵ for the ϵ-ball neighbors, may significantly affect the structure of the hypergraph and further influence the performance of hypergraph learning. Unfortunately, there are still no general principles for the selection of k and ϵ, and the adaptive justification of these hyperparameters is not trivial in practice. Second, the calculation of the k-NN neighbors is expensive for large scaled data in both time and memory.

Regarding the clustering-based hyperedge generation strategy, there is no common way to determine how many clusters should the vertex set be divided into, as the scale of the clustering results also affects the structure of the hypergraph. A possible solution is to conduct clustering multiple times in different scales, which generates multiple hypergraphs with different k values and then composes these hypergraphs together for multilevel representation.

4.2.2 Representation-Based Hypergraph Generation

As introduced above, the distance-based hypergraph generation has some disadvantages. For the kNN hypergraph, the hypergraph, which connects the centroid sample and its k nearest samples, is uniform. Its structure may not be sufficiently adaptive. Also, the distance-based hypergraph is sensitive to noise. To solve this problem, the hypergraph can be generated by the representation-based methods.

Different from the distance-based methods, which generate hyperedges through some metrics in the feature space, the relations among the vertices in representation-based methods are from the feature reconstruction, as shown in Fig. 4.5. In reconstruction, different strategies have different generation effects. Here we introduce three representation-based main branches to construct hypergraphs, i.e., l_1-hypergraph [7], l_2-hypergraph [8], and the combination of them both. The details of these methods are described as follows.

(1) l_1-Hypergraph Generation

For the l_1-hypergraph construction, as introduced in [7], sparse representation method can be used to formulate the relation between the hyperedge and its vertices, and the sparse representation is embodied in the coefficients that linearly combine

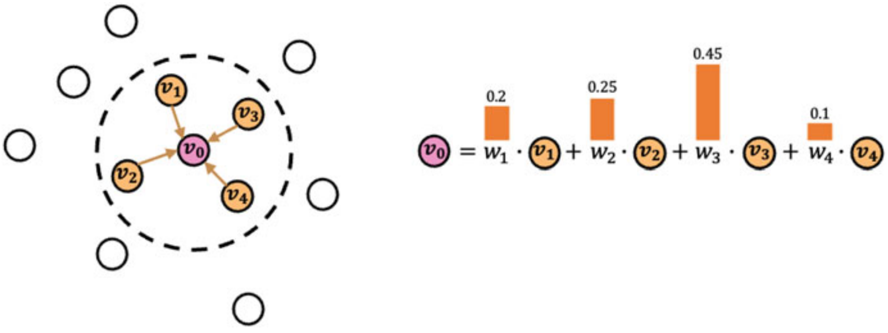

v_1
v_2
v_3
v_4
v_0
0.2
0.25
0.45
0.1
$v_0 = w_1 \cdot v_1 + w_2 \cdot v_2 + w_3 \cdot v_3 + w_4 \cdot v_4$

Fig. 4.5 An illustration of the representation-based methods

the basic vectors to reconstruct the input vector. In the hyperedge construction, the centroid vertex is reconstructed by the other vertices in the same hyperedge. We use the coefficients to present the incidence matrix of hypergraph. Mathematically, we denote the centroid vertex in the l_1-hypergraph by v_c, and it can be represented as

$$\begin{aligned}&\arg\min_{\mathbf{z}} \|\mathbf{B}\mathbf{z} - \mathbf{X}(v_c)\|_2^2 + \gamma\|\mathbf{z}\|_1,\\ &s.t.\ \forall i, \mathbf{z}_i \geq 0,\end{aligned} \tag{4.9}$$

where $\mathbf{X}(v_c)$ denotes the feature vector of the centroid vertex, $\mathbf{B}$ denotes the feature of its k nearest vertices, and $\mathbf{z}_i$ is the reconstruction coefficient vector. The first term in Eq. (4.9) is the reconstruction term that makes a good representation of input vector $\mathbf{X}(v_c)$ with the basic vectors $\mathbf{B}$. The second term is the l_1-regularization, which forces the coefficient $\mathbf{z}$ to sparse. γ is a hyperparameter that balances the influences of the two terms. The constraint $\mathbf{z}_i \geq 0$ makes the reconstruction coefficients non-negative. Note that each sample may act as a centroid vertex to generate a hyperedge. For the dataset containing n samples, the optimization problem is solved for n times. The non-zero reconstruction coefficients in the representation can be seen as the connection weights of the neighborhood vertices in the hyperedge, and the neighborhood vertices with zero reconstruction coefficients are outside of the hyperedge. The connection weight between the hyperedge and the neighborhood vertices can be set as the vector of coefficients $\mathbf{z}_i$. The incident matrix $\mathbf{H}$ of this hypergraph is defined as

$$\mathbf{H}(v_j, e_i) = \begin{cases} \mathbf{z}_i^j & \text{if } v_j \in e_i \\ 0 & \text{otherwise} \end{cases}, \tag{4.10}$$

where e_i is generated with the centroid vertex v_i, and $\mathbf{z}_i^j$ is the jth element of representation coefficients $\mathbf{z}$.

(2) *Elastic*-Hypergraph Generation

The l_1-regularization in l_1-hypergraph can generate sparse and effective hypergraphs, despite that fact that it is hard to reveal the grouping information of samples. To enhance the effect of grouping, the elastic net [9] is introduced to combine an l_2-norm penalty with the l_1-norm constraint. The objective function of elastic net can be formulated as

$$\begin{aligned} &\arg\min_{\mathbf{z}} \|\mathbf{B}\mathbf{z} - \mathbf{X}(v_c)\|_2^2 + \gamma\|\mathbf{z}\|_1 + \beta\|\mathbf{z}\|_2^2, \\ &s.t.\ \ \forall i. \mathbf{z}_i \geq 0. \end{aligned} \tag{4.11}$$

The elastic net can create a hyperedge whose weight can be determined by the reconstruction coefficients by using both the l_2-norm and the l_1-norm penalties to group more relevant and important neighbors.

(3) l_2-Hypergraph Generation

Note that there are two drawbacks of the above two representation-based approaches: (1) They use a l_2-norm-based metric to measure the reconstruction errors, which makes them still sensitive to sparse reconstruction errors. (2) Since these methods create hyperedges by linearization, they are unable to handle nonlinear data. By eliminating the sparse noise component from the original data, integrating the locality, and maintaining the constraint to the linear regression framework, the l_2-hypergraph [9] is created to address these issues as

$$\begin{aligned} &\arg\min_{\mathbf{z}} \|\mathbf{X} - \mathbf{X}\mathbf{C} - \mathbf{E}\|_F^2 + \frac{\gamma_1}{2}\|\mathbf{C}\|_F^2 + \frac{\gamma_2}{2}\|\mathbf{Q} \odot \mathbf{C}\|_F^2 + \beta\|\mathbf{E}\|_1, \\ &s.t.\ \ \mathbf{C}^T\mathbf{1} = \mathbf{1}, \mathtt{Diag}(\mathbf{C}) = \mathbf{0}, \end{aligned} \tag{4.12}$$

where $\odot$ stands for element-wise multiplication, $\mathbf{C}$ is the coefficient matrix, $\mathbf{E}$ is the data error matrix, and $\mathbf{Q}$ is the locality adapter matrix used to retain the local manifold structures. Hyperedges can then be created using the coefficient matrix $\mathbf{C}$.

The ability of each vertex being able to be reconstructed in the feature space can be evaluated via representation-based hyperedges. It is possible to calculate and use the correlation between the feature vectors to create connections among the vertices. Similar to the distance-based methods, this field of study may encounter the issue of data noise and outliers. Another drawback of this type of hypergraph generation methods is that only a portion of the relevant samples is chosen for reconstruction during the computing process, and the resulting hyperedge may not be able to accurately capture the data correlation through the complete data distribution.

4.3 Explicit Hypergraph Modeling

Different from implicit hypergraph modeling, in some cases, there are existing connections among data. Explicit hypergraph modeling focuses on these scenarios and generates hypergraph structure using attribute information or networks.

4.3.1 Attribute-Based Hypergraph Generation

The data in real world may be associated with attributes in many cases. For example, the users in social network could have profiles, such as gender, age, and interests. The visual objects in images could have different characteristics, such as color, shape, and texture. Given the data assigned with different attributes, attribute-based hypergraph generation methods can be adopted to construct the hypergraph based on the attribute information, which provides an explicit way to encode semantic properties and diffuse knowledge [10]. As such a construction schema leverages the apparent correlations among objects directly, it can be categorized as explicit hyperedge methods.

To generate a hypergraph using attributes, the following steps are needed: the hypergraph structure construction and the hyperedge weight assignation. The first step is to generate the vertex set $\mathcal{V}$ and hyperedge set $\mathcal{E}$ based on the attribute information, and the second step is to assign different weights to the hyperedges and acquire the weight matrix $\mathbf{W}$.

When constructing the hypergraph from the attribute data, the samples to be explored are first modeled as vertices in a hypergraph, denoted as the vertex set $\mathcal{V}$. The same attribute shared by different vertices effectively indicates that these samples share common characteristics, which may be an objective tag or a subjective evaluation. Therefore, each attribute can be regarded as the semantic information on a connection, i.e., a hyperedge. In attribute-based hypergraph generation methods, a group of hyperedges (called a hyperedge group) are generated by linking all the vertices associated with the attribute space. It is obvious that the number of hyperedges equals to the number of attributes in this way. Such a hyperedge group generated based on the attribute information is denoted by

$$\mathcal{E}_{\text{attribute}} = \left\{ N_{\text{att}}(a) \mid a \in \mathcal{A} \right\}, \tag{4.13}$$

where $N_{\text{att}}(a)$ is the subset of vertex set $\mathcal{V}$ sharing the attribute a, and $\mathcal{A}$ is a set containing all defined attributes. Sometimes the attribute could be hierarchical, *e.g.*, the car within the vehicles. In this case, the $\mathcal{A}$ and $\mathcal{E}_{\text{attribute}}$ can be extended to involve the subtypes of the attributes.

Here we give one simple example to show how to construct the hypergraph structure using the attribute information, as shown in Fig. 4.6. Given a social network data with user profiles, the users in the social network are first modeled as

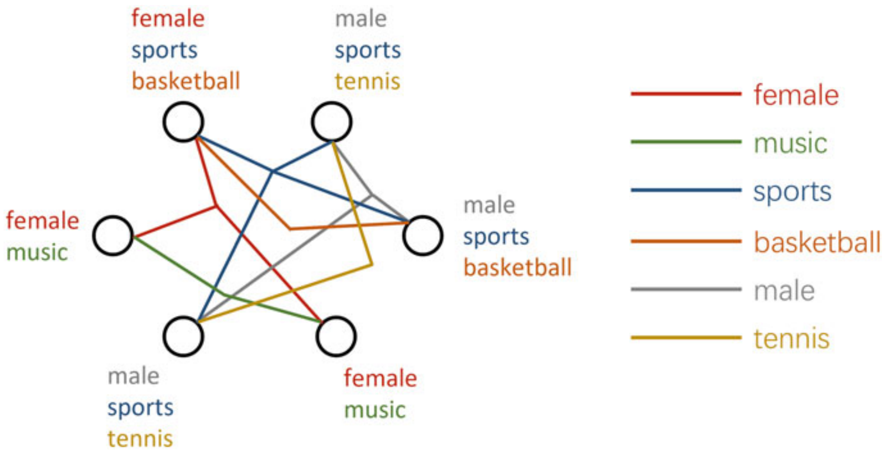


Fig. 4.6 An illustration of the attribute-based hyperedge generation method

vertices $\mathscr{V}$. The user profiles contain the objective reality such as gender and age as well as the subjective characteristics such as interests and knowledge, both of which can be adopted to generate the hyperedge groups. For example, we can have e_{female} hyperedge connecting all female users and e_{sports} linking users who like sports. Additionally, as discussed above, sometimes the attributes are hierarchical. Under such circumstances, we can generate hyperedges with different levels to characterize multiple-scale attribute connections. For instance, we have users A, B, C, and D who all like sports, among them both users A and B like playing basketball, and users C and D like playing tennis. In this case, we first generate e_{sports} connecting users A, B, C, and D, and then $e_{basketball}$ and e_{tennis} are generated to link A, B and C, D, respectively. The hyperedge set in this example can be written as

$$\mathscr{E} = \{e_{female}, e_{sports}, e_{basketball}, e_{tennis} \cdot\}.$$

The hyperedge weights are also important here. For attribute-based hypergraph, the number of shared attributes among the samples connected by the hyperedge can quantitatively reflect the relative correlation strength. Specifically, the more the attributes that the samples share, the stronger connections exist among these corresponding vertices, and the bigger weight that the hyperedges are assigned. Here each hyperedge e here can be seen as a clique. The mean of the heat kernel weights $w(e)$ of the pairwise edges in this clique is considered as the corresponding hyperedge weight:

$$w(e) = \frac{1}{\delta(e)(\delta(e)-1)} \sum_{u,v\in e} \exp\left(-\frac{\|\mathbf{X}(u)-\mathbf{X}(v)\|_2^2}{\sigma^2}\right), \tag{4.14}$$

where $\delta(e)$ indicates the degree of hyperedge e, and $\mathbf{X}(u)$ and $\mathbf{X}(v)$ denote the feature vectors of vertices u and v, respectively.

The attribute-based hypergraph generation method can capture the semantic properties apart from the structural information conveyed by the hypergraph structures themselves. The attributes serve as a type of intermediate-level feature representation of vertices and can provide another description for vertices beyond the low-level representations. However, the attributes are not available all the time. When there is no natural attribute descriptor for vertices, some extra solutions need to be applied to conduct attribute-based hypergraph generation. One possible solution is to manually design attribute tags, which may be both cumbersome and time-consuming. The other alternative is extracting attribute information from the raw low-level features by machine learning models [11]. Such a schema is more time-saving than manual definition, whereas the results rely heavily on the accuracy of the machine learning model. We also note that the attributes can be nameable, which indicates the semantic information can be directly understood, while they can also be non-nameable, which means the semantic information is not explicit.

4.3.2 Network-Based Hypergraph Generation

There are many applications of network data, including social networks [12], reaction networks [13], cellular networks [13], and human brain networks [3]. It is possible to generate subject correlations using the network information for these data. In a typical work of social media analysis [14], the vertices on hypergraph represent users and images. In addition to visual–textual relationships among images, hyperedges can be used to capture social links between users and images, also called homogeneous and heterogeneous hyperedges. The nearest-neighbor-based and attribute-based hyperedge generation methods are used to construct homogeneous hyperedges representing the visual and textual relations among images. Users and images are connected through social link relations to construct heterogeneous hyperedges. For example, both friendship and mobility information in location-based social networks can be used to generate hypergraphs using [12]. As a result, friendship hyperedges are generated within the social domain, and check-in hyperedges are generated across the social, semantic, temporal, and spatial domains. A protein–protein interaction network is naturally represented by a hypergraph [15], whose subsets (hyperedges) can be represented by tandem affinity purification (TAP) data.

Aside from the first-order correlation, high-order correlations, e.g., the second- and third-order correlations, within the network can also be used as a means for generating hyperedges. A center vertex can be connected with its first-order and high-order neighbors (i.e., vertices whose shortest path to the centroid is greater than 1) through a hyperedge. A vertex's low-order neighbors need only to be considered if attention is focused on its local connection in the network. As an example, users who have similar preferences on items are able to be connected

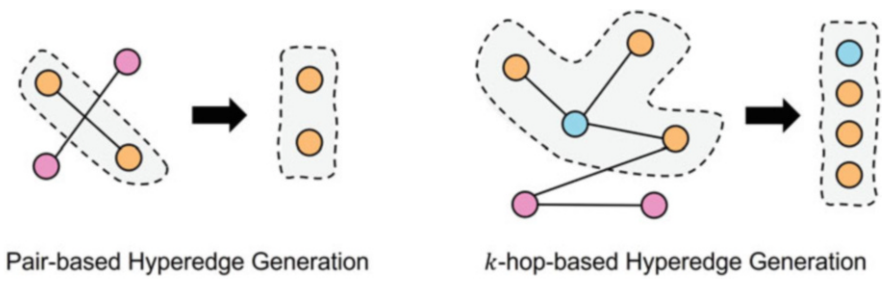

Pair-based Hyperedge Generation
k-hop-based Hyperedge Generation

Fig. 4.7 An illustration of the network-based hyperedge generation method

within the recommendation network [4] according to first-order and second-order correlations, which will be used in order to generate a hypergraph as well as to perform collaborative filtering for the recommendation. Alternatively, if information of a vertex travels a long distance in the network, higher-order correlation is required to generate hyperedges.

We then introduce two typical approaches to construct hyperedges from network/graph structure, i.e., pair-based and k-hop-based. Figure 4.7 illustrates the profile of these two approaches. In this example, $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ represents the graph structure with $v_i \in \mathcal{V}_s$ representing a vertex and $e_{s_{ij}} \in \mathcal{E}_s$ representing an edge connecting v_i and v_j. We let $\mathbf{A}$ indicate the adjacency matrix of $\mathcal{G}_s$. As a result of such a graph structure, two types of hyperedge groups can be generated (Fig. 4.7).

(1) Pair-Based Hyperedge Generation Strategy

The $\mathcal{E}_{\text{pair}}$ is adopted to indicate the hyperedges constructed by pair correlations in the network/graph. $\mathcal{E}_{\text{pair}}$ targets at directly transforming the graph structure into a group of 2-uniform hyperedges, which can be formulated as follows:

$$\mathcal{E}_{\text{pair}} = \Big\{ \{v_i, v_j\} \mid (v_i, v_j) \in \mathcal{E}_s \Big\}. \tag{4.15}$$

As a result, $\mathcal{E}_{\text{pair}}$ covers the low-order (pairwise) correlations in the graph structure, which is the basic information for high-order correlation modeling.

(2) k-Hop-Based Hyperedge Generation Strategy

$\mathcal{E}_{\text{hop}}$ is adopted to indicate the hyperedges constructed by the k-hop neighbors in the network/graph. First, we define the k-hop neighborhoods of a vertex v in graph $\mathcal{G}_s$ as follows:

$$N_{\text{hop}_k}(v) = \{u \mid \mathbf{A}^k_{uv} \neq 0, u \in \mathcal{V}_s\}.$$

Based on the k-hop's reachable positions in the graph structure, $\mathcal{E}_{\text{hop}}$ aims to find the related vertices for a central vertex. The range of the values of k is $[2, n_v]$, where n_v refers to the number of vertices in $\mathcal{G}_s$. The following is an example of a hyperedge group $\mathcal{E}_{\text{hop}}$ with k-hop:

$$\mathcal{E}_{\text{hop}_k} = \left\{ N_{\text{hop}_k}(v) \mid v \in \mathcal{V} \right\}. \tag{4.16}$$

The hyperedge generated by $\mathcal{E}_{\text{hop}}$ can be exploited by extending the search radius to the external vertices, which also leads to groups of vertices rather than just two vertices, as opposed to two vertices only in the graph structure. As compared with just the pairwise correlation in $\mathcal{E}_{\text{pair}}$, it can provide more information about correlations.

Here, we discuss the advantages and limitations of the two types of hyperedges using network data, respectively. As far as the pair-based construction is concerned, clearly this type of hyperedge can only model low-order correlations, which cannot naturally explore high-order correlations in some scenarios. In contrast, hyperedges generated from the k-hop-based methods have the high-order information built-in of the original network. However, the high-order information in this type of hyperedges may be redundant and ambiguous. This is because the connection details in the k-hop-based hyperedges may be lost, which means that you cannot reconstruct the original network/graph from this type of hyperedge. Additionally, the k-hop-based hyperedges may lead to irreversible over-smoothing in each hyperedge, which is caused by the k-hop neighbors with exponential growth as k grows.

4.4 Typical Examples of Hypergraph Modeling

Here we give several examples of hypergraph modeling in real applications, including computer vision, recommender systems, computer-aided diagnosis, and brain networks, to demonstrate how to construct hypergraphs from data.

4.4.1 Computer Vision

Computer vision has attracted much attention in recent decades. In computer vision, there are multi-modal data, such as images, point clouds, etc. Both low-level vision tasks and high-level vision tasks have been deeply investigated. In these tasks, an important but challenging issue is the complex data correlation behind the vision data. For example, from the aspect of images, the pixels or patches are the elements of an image, while the semantic information for the image is represented by these pixels or patches. Terrence Joseph Sejnowski [16] mentioned that "*In a task such as face recognition, in which **important information may be contained in the high-order relationships among pixels**, it seems reasonable to expect that better basis images may be found by methods sensitive to these high-order statistics*." Similar situations occur when facing multi-modal 3D object representation. Usually, a 3D object can be represented by different ways, such as one single image, multi-view, point clouds, voxel, and mesh. Under such circumstances, the correlation among these objects becomes even more complicated. To model such high-order relationship among pixels/patches in one image, or among different 3D objects, simple graph is not capable to conduct this task.

We first look into the high-order correlation modeling for an image. A 2D image is composed of a set of pixels, and each pixel owns a feature vector (channels). To generate a hypergraph to model the correlation behind this image, we can take each patch in the image as a vertex in the hypergraph, and the objective is to generate a group of hyperedges to connect these vertices (patches). Here we can employ the distance-based hypergraph generation method, in which each patch is selected as the centroid, and its nearest neighbors in the feature space are connected by a hyperedge. This process is shown in Fig. 4.8. Furthermore, we can also employ the spatial information to build connection among these patches. The patches with closed spatial locations in the image could be connected with a hyperedge. Figure 4.9 shows an example of hypergraph modeling for image patches using spatial information.

For 3D visual objects, there are complex correlations among them. For example, different furniture, such as tables and chairs, have legs, and different vehicles, such as cars and bicycles, have wheels. Another challenging issue comes from the multi-modality aspect. Given different modal data of 3D objects, the correlations are composed of inter-modal correlations and the cross-modal correlations, as shown

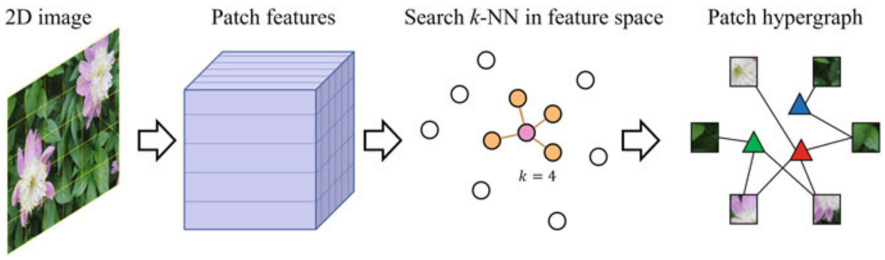


Fig. 4.8 An example of hypergraph modeling for image patches using feature information

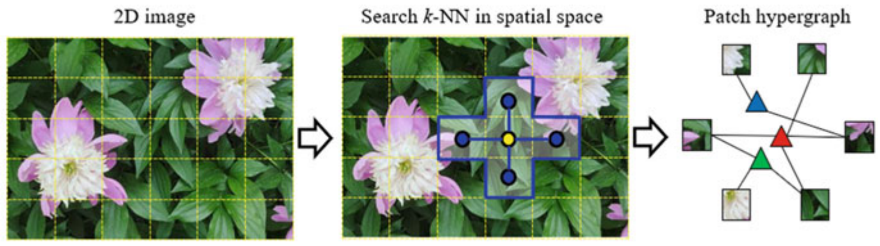


Fig. 4.9 An example of hypergraph modeling for image patches using spatial information

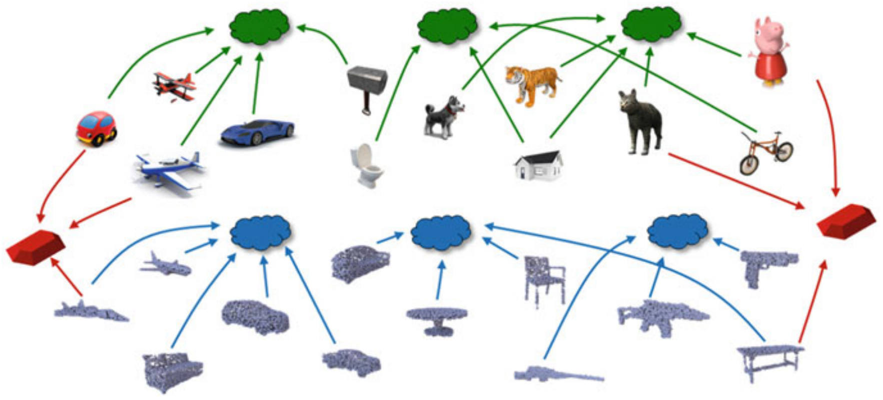


Fig. 4.10 The complex correlations among multi-modal 3D objects

in Fig. 4.10. Given a large number of 3D objects, it is difficult to accurately and completely manually describe all these correlations.

In order to efficiently build a hypergraph structure, we usually extract the features of 3D objects and then build implicit hypergraphs. 3D objects can be described by multiple modalities, including point clouds, views, grids, and voxels. We can extract the descriptors of their respective modalities through the corresponding deep neural networks, such as dynamic graph CNN (DGCNN) [17] and PointNet (PointNet) [18] for point cloud data, multi-view convolutional neural networks (MVCNN) [19], and group-view convolutional neural networks (GVCNN) [20] for the multi-view data. When multi-modal features have been obtained, we can build a hypergraph structure for each kind of features.

Here, each 3D object can be represented by a vertex in the hypergraph. Each time, one object is selected as the centroid in a feature space, and its nearest neighbors can be connected by a corresponding hyperedge. This process is repeated until all objects have been selected as the centroid once in this feature space. Every feature and possible feature combination can be used in this process. In this way, we can achieve multiple hypergraphs, represented by incidence matrices $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_m$ to formulate their correlations under different modalities. The pipeline is demonstrated in Fig. 4.11. We can further concatenate these incidence matrices along the axes of hyperedges to integrate these hypergraphs and obtain the complete hypergraph structure, as shown in Fig. 4.12.

4.4.2 Recommender System

In a recommender system, the relationship between users and items can be represented by a bipartite graph, that is, if an item is in a user's recommendation list, then we connect the user vertex with the item vertex. This bipartite graph can

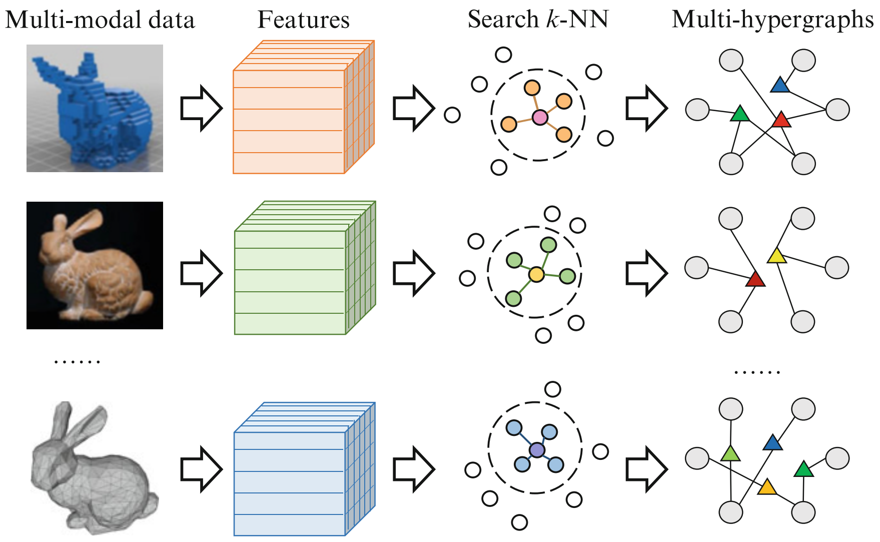


Fig. 4.11 An example of hypergraph modeling for multi-modal 3D objects

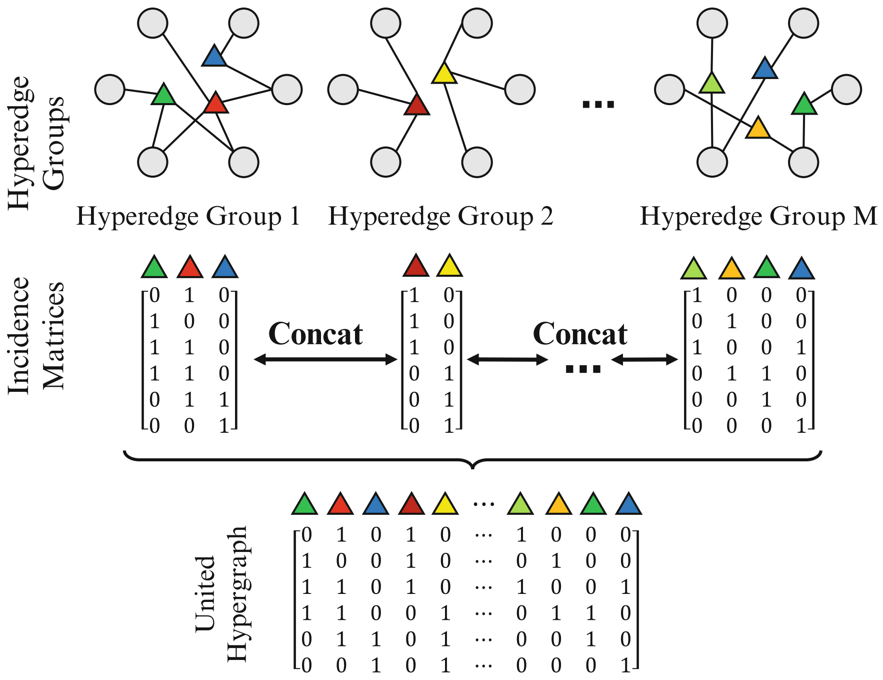


Fig. 4.12 An illustration of multi-hypergraph combination. This figure is from [5]

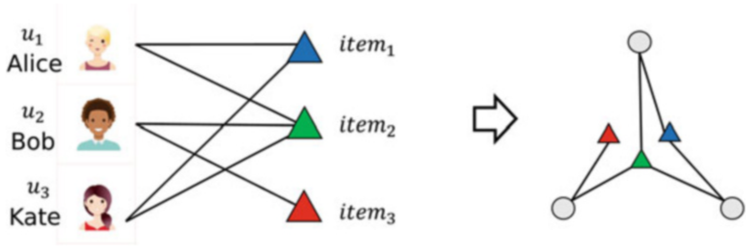


Fig. 4.13 An example of hypergraph modeling for a recommender system

be simply transformed into a hypergraph, where vertices on one side remain and vertices on the other side become hyperedges, as shown in Fig. 4.13. In this way, each user can be represented as a vertex in the hypergraph, and the users shared the same items can be connected by a corresponding hyperedge here. If the item is regarded as a vertex, then the hyperedges are generated using shared users. This hypergraph generation procedure follows the attribute-based strategy.

Mathematically, the ranking matrix of the recommender system equals to the incidence matrix of the corresponding hypergraph. With this transformation, we can solve the problem in recommender systems via hypergraph learning methods. In fact, undirected bipartite graph modeling and hypergraph modeling are interchangeable in some cases. If the edges in bipartite graph are weighted, we can use the hyperedge-dependent vertex weights accordingly.

4.4.3 Computer-Aided Diagnosis

In computer-aided diagnosis, the main objective is to measure whether a coming patient has some specific disease or not, or how serious the disease it is. For diagnosis, the experience and knowledge are from previous medical records. Case-based diagnosis has shown importance in practice. For AI-based computer-aided diagnosis, it is important to explore the existing labeled training data, which could be very few in some cases. These medical records may contain different examine files, MR images, CT images, and other types of data.

A conventional pipeline for computer-aided diagnosis is first extracting features from clinical text or medical imaging data and then applying computer programs to automatically categorize healthy people and patients. The commonly used techniques involve natural language processing, medical imaging analysis, machine learning, etc. It is worth noting that the existing methods mostly focus on individual subject classification. Under such circumstances, how to model the correlation among these subjects, including the training data and the coming patient (the testing data), is an important but difficult task.

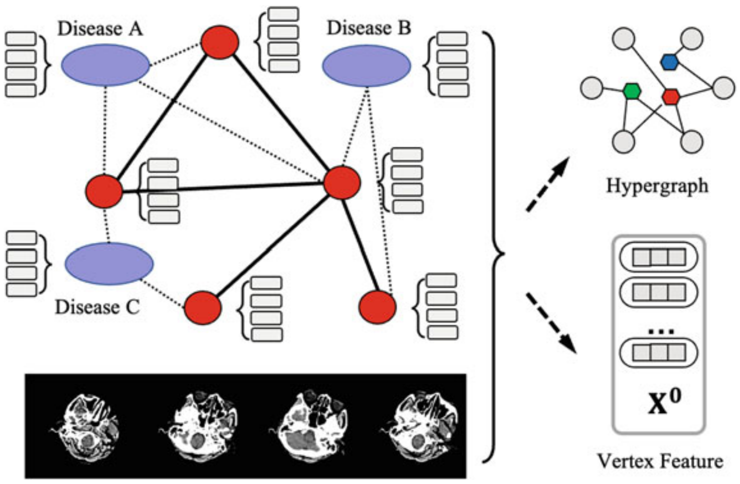


Fig. 4.14 An example of hypergraph modeling for computer-aided diagnosis

Here, a hypergraph at the subject level, i.e., each vertex stands for a subject, can be generated, where the hyperedges can be created using the distance-based method or attribute-based method. Given the MR images or other medical data, the features can be used to measure the distance between each two subjects. Then, the k-NN scheme can be used to select nearest neighbors for a centroid vertex and then generate a corresponding hyperedge, as shown in Fig. 4.14.

Another type of applications is to model the inter-correlation in one medical image, such as gigapixel whole-slide histopathological images (WSIs). Survival prediction is an important task in medical image analysis, which targets on modeling the life duration of a patient using WSIs. Different from traditional images, WSIs are with very large size and rich details. Therefore, traditional image representation methods do not work well in this task. To formulate the inter-correlation inside a WSI, a hypergraph can be generated, which the patch correlations are generated. A group of patches can be sampled from the original WSI, such as 2000 or 8000 patches. Then, these patches are represented as vertices in the corresponding hypergraph. The hyperedges can be generated based on either the visual feature of these patches or the spatial information, or both of them, using the distance-based hypergraph generation methods.

4.4.4 Brain Network

Recently, the development of neuroimaging techniques has provided a way to understand the brain network on a large scale. Studies have shown that the interaction relationships in the brain, from neuronal information flow to whole-brain functional networks, are the basis of its functionality. Therefore, formulating

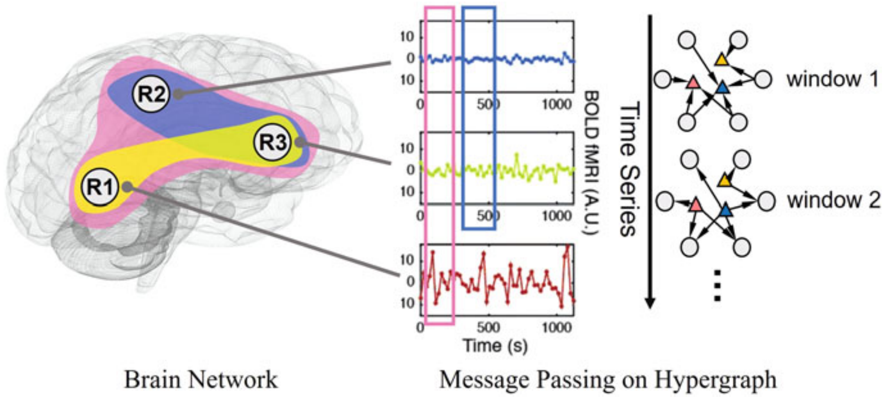


Fig. 4.15 An example of hypergraph modeling for brain network

the brain as a complex network and decoding its signals may further deepen our understanding of the human cognitive processes. The conventional functional network is usually modeled and represented based on pairwise correlations between two brain regions. However, neurologically, a brain region predominantly interacts with one or more other brain regions.

When using hypergraphs to model a single brain network, the vertices denote brain regions, and the hyperedges represent the interactions among multiple regions. Each element in the incidence matrix corresponds to the contribution of the brain region to the specific function, as shown in Fig. 4.15. In this process, each region can be selected as the centroid, and its nearest neighbor regions in the feature space can be selected and connected by a corresponding hyperedge.

4.5 Hypergraph Modeling in Next Stage

In this part, we discuss future research topics of hypergraphs modeling to render them more accurate and flexible, including adaptive hypergraph modeling, generative hypergraph modeling, and knowledge hypergraph generation.

4.5.1 Adaptive Hypergraph Modeling

Having initialized the hypergraph structure, the structure is fixed during the learning process. However, the initial hypergraph structure constructed by existing hypergraph modeling methods contains many noisy connections that may be destructive for the learning process. Therefore, the original structure needs to be optimized according to the data and downstream tasks to cut down on structure noise. Although there are some existing work on hypergraph structure optimization, these efforts are still far from reaching the goal of accurately modeling of complex data correlations.

At this stage, the selection of hypergraph generation methods still depends on experience, rather than a theoretical strategy. A possible route to conduct automate hypergraph generation is to create various hypergraphs via different approaches and then group them together to obtain a more complex but relatively complete hypergraph. The grouping weights can be learned in the training stage. Another way is to update the incidence matrix of hypergraph structure, which can be either directly optimized as learnable parameters or indirectly optimized via metric learning.

4.5.2 Generative Hypergraph Modeling

The generative models are a set of models that learn the distribution from the observed data and generate new data instances based on probability. They have been widely used in different tasks such as generation, synthesis, translation, reconstruction, prediction, etc. In recent years, with the development of deep graph representation learning, deep graph generative models have attracted much attention. Given a series of training graph data (assumed to be taken from the same distribution), the neural network is trained as a graph generation model. Inspired by these generative models, building a hypergraph by estimating the distribution of latent structures from observed data may be a viable way. Given a set of training hypergraphs or sampling signals from every vertex, the distribution can be implicitly or explicitly derived by combining hypergraph embeddings and generative models.

However, there is still a long way to go for hypergraph generative models to become practical. Unlike simple graphs whose distributions are the joint distributions of all pairwise correlations between data, the distribution of a hypergraph structure is the joint distribution of all high-order correlations among data. Therefore, the joint distribution is high dimension, and the variables are dependent on each other. Estimating the density function is intractable with considerable complexity. Furthermore, due to the high-dimensional issue, a large amount of observed data is required to make the density estimate closer to the true distribution, which is difficult to obtain in practical applications. Despite the above obstacles, generative hypergraph modeling is an area worth exploring in the future and will become useful in many areas, such as simulations of complex physical systems, trajectory tracking system identification, and community detection.

4.5.3 Knowledge Hypergraph Generation

Knowledge hypergraph has attracted much attention in recent years since it can store facts using high-arity relations. In a knowledge hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the vertices represent the set of entities, and hyperedges demonstrate the high-arity relations. The basic unit is a fact based on a high-arity relation. Unlike knowledge graph that only uses binary relations, the relations in knowledge hypergraph are defined on any number of entities.

Although there have been several pieces of work targeting at knowledge hypergraph embedding and completion, such as Multi-fold TransH (m-TransH) [21], Hyper-relational Knowledge Graph Embedding (HINGE) [22], N-ary Link Prediction (NaLP) [23], they are mostly based on the assumption that there exists an initial knowledge hypergraph or some hyper-relational links. Few efforts have been made on the initial knowledge hypergraph generation. Actually, manually mining the hyper-relations among entities requires much time and effort. Therefore, it is of great significance to study the knowledge hypergraph generation methods for efficient and comprehensive knowledge inference.

4.6 Summary

In this section, we introduce the hypergraph modeling methods, which are categorized as the implicit type and the explicit type. The implicit hyperedges can be used in tasks in which we can represent each subject and develop metrics to evaluate sample similarity. By using the sparse representation, representation-based approaches might mitigate the impact of the noise vertices in comparison with distance-based ones. Explicit hyperedges are more appropriate when input data may already have certain structural details. In general, choosing a suitable hyperedge generation method is important for a specific task. Finally, adaptive and generative hypergraph modeling are worth further exploring to adjust hypergraph structures based on the data and the on-going tasks.

References

1. Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, C. Zou, Hypergraph learning: methods and practices. IEEE Trans. Pattern Analy. Mach. Intell. **44**(5), 2548–2566 (2022)
2. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**, 4290–4303 (2012)
3. C. Zu, Y. Gao, B. Munsell, M. Kim, Z. Peng, Y. Zhu, W. Gao, D. Zhang, D. Shen, G. Wu, Identifying high order brain connectome biomarkers via learning on hypergraph, in *Proceedings of the Machine Learning in Medical Imaging* (2016), pp. 1–9
4. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 2020–2029
5. Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN$^+$: general hypergraph neural networks. IEEE Trans. Pattern Analy. Mach. Intell. **45**(3), 3181–3199 (2023)
6. Y. Huang, Q. Liu, D. Metaxas, Video object segmentation by hypergraph cut, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1738–1745

7. M. Wang, X. Liu, X. Wu, Visual classification by ℓ_1-hypergraph modeling. IEEE Trans. Knowl. Data Eng. **27**, 2564–2574 (2015)
8. T. Jin, Z. Yu, Y. Gao, S. Gao, X. Sun, C. Li, Robust ℓ_2-hypergraph and its applications. Inf. Sci. **501**, 708–723 (2019)
9. Q. Liu, Y. Sun, C. Wang, T. Liu, D. Tao, Elastic net hypergraph learning for image clustering and semi-supervised classification. IEEE Trans. Image Process. **26**, 452–463 (2017)
10. S. Huang, M. Elhoseiny, A. Elgammal, D. Yang, Learning hypergraph-regularized attribute predictors, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 409–417
11. Y. Fang, Y. Zheng, Metric learning based on attribute hypergraph, in *Proceedings of the IEEE International Conference on Image Processing* (2017), pp. 3440–3444
12. D. Yang, B. Qu, J. Yang, P. Cudre-Mauroux, Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach, in *Proceedings of the World Wide Web Conference* (2019), pp. 2147–2157
13. N. Franzese, A. Groce, T.M. Murali, A. Ritz, Hypergraph-based connectivity measures for signaling pathway topologies. PLOS Comput. Biol. **15**(10), e1007384 (2019)
14. Q. Fang, J. Sang, C. Xu, Y. Rui, Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning. IEEE Trans. Multimedia **16**, 796–812 (2014)
15. S. Klamt, U.-U. Haus, F. Theis, Hypergraphs and cellular networks. PLoS Comput. Biol. **5**(5), e1000385 (2009)
16. M.S. Bartlett, J.R. Movellan, T.J. Sejnowski, Face recognition by independent component analysis. IEEE Trans. Neural Netw. **13**(6), 1450–1464 (2002)
17. Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. **38**, 1–12 (2019)
18. R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 652–660
19. H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 945–953
20. Y. Feng, Z. Zhang, X. Zhao, R. Ji, Y. Gao, GVCNN: Group-view convolutional neural networks for 3d shape recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 264–272
21. J. Wen, J. Li, Y. Mao, S. Chen, R. Zhang, On the representation and embedding of knowledge bases beyond binary relations, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2017), pp. 1300–1307
22. P. Rosso, D. Yang, P. Cudré-Mauroux, Beyond triplets: Hyper-relational knowledge graph embedding for link prediction, in *Proceedings of the Web Conference* (2020), pp. 1885–1896
23. S. Guan, X. Jin, Y. Wang, X. Cheng, Link prediction on n-ary relational data, in *Proceedings of the World Wide Web Conference* (2019), pp. 583–593

Chapter 5
Typical Hypergraph Computation Tasks

Abstract After hypergraph structure generation for the data, the next step is how to conduct data analysis on the hypergraph. In this chapter, we introduce four typical hypergraph computation tasks, including label propagation, data clustering, imbalance learning, and link prediction. The first typical task is label propagation, which is to predict the labels for the vertices, i.e., assigning a label to each unlabeled vertex in the hypergraph, based on the labeled information. In general cases, label propagation is to propagate the label information from labeled vertices to unlabeled vertices through structural information of the hyperedges. In this part, we discuss the hypergraph cut on hypergraphs and random walk interpretation of label propagation on hypergraphs. The second typical task is data clustering, which is formulated as dividing the vertices into several parts in a hypergraph. In this part, we introduce a hypergraph Laplacian smoothing filter and an embedded model for hypergraph clustering tasks. The third typical task is cost-sensitive learning, which targets on learning with different mis-classification costs. The fourth typical task is link prediction, which aims to discover missing relations or predict new coming hyperedges based on the observed hypergraph.

5.1 Introduction

In previous chapters, we have introduced how to generate the hypergraph structure given observed data. After the hypergraph generation step, how to use this hypergraph for different applications becomes the key task. Hypergraph has the potential to be used in different areas, such as social medial analysis, medical and biological applications, and computer vision. We notice most of the applications can be categorized into several typical tasks and follow similar application patterns. In this chapter, we introduce several typical hypergraph computational tasks, which can be used for different applications.

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_5

More specifically, four typical tasks, including label propagation, data clustering, cost-sensitive learning, and link prediction, are introduced in this chapter. The first typical task is label propagation, which is also one of the most widely used methods in machine learning. The objective of label propagation is to assign a label to each unlabeled data. In general cases, label propagation on hypergraph is to propagate the label information from labeled vertices to unlabeled vertices through structural information of the hyperedges. Random walk is a basic processing for information propagation, which also plays a fundamental role in this process. We then review the hypergraph cut on hypergraphs and random-walk-based label propagation on hypergraphs. We introduce the label propagation process on single hypergraph and multi-hypergraphs [1, 2], respectively, in this part.

The second typical task is data clustering, targeting on grouping data into different clusters. We introduce how to conduct data clustering using hypergraph computation. The hypergraph structure can be used as guidance to the clustering criteria. Two types of hypergraph clustering methods are introduced, including structural hypergraph clustering and attribute hypergraph clustering, due to the different data information in the hypergraph. In structural hypergraph, the clustering tasks only use structural information, while in attribute hypergraph, each vertex is usually accompanied by attribute information from the real world. We introduce a hypergraph Laplacian smoothing filter and an embedded model specifically for hypergraph clustering tasks that named adaptive hypergraph auto-encoder (AHGAE) [3].

The third typical task is cost-sensitive learning, which is to solve the learning task under the scenario with different mis-classification costs, such as confronting the imbalanced data distribution issue. Here, we introduce two hypergraph computation methods, i.e., cost-sensitive hypergraph computation [4] and cost interval optimization for hypergraph computation [5]. First, we introduce a cost-sensitive hypergraph modeling method, in which the cost for different objectives is fixed in advanced. As the exact cost value may be not easy to be determined, we then introduce a cost interval optimization method, which can utilize the cost chosen inside the interval while generating data with high-order relations.

The fourth typical task is link prediction, which is to predict data relationship and can be used for recommender system and other applications. Here, the hypergraph link prediction is to mine the missing hyperedges or predict new coming hyperedges based on the observed hypergraph. We introduce a variational auto-encoder for heterogeneous hypergraph link prediction [6]. It aims to learn the low-dimensional heterogeneous hypergraph embedding based on the Bayesian deep generative strategy. The heterogeneous encoder generates the vertex embedding and hyperedge embedding, and the hypergraph embedding is the combination of them. The hypergraph decoder reconstructs the incidence matrix based on the vertex embedding and the hyperedge embedding, and the heterogeneous hypergraph is generated based on the reconstructed incidence matrix.

Part of the work introduced in this chapter has been published in [1–6].

5.2 Label Propagation on Hypergraph

This section mainly introduces the label propagation task on hypergraph. We first introduce the basic assumptions of the label propagation process. Given a set of vertices on a hypergraph, a part of vertices is labeled, while other vertices are unlabeled. The task is to predict the label information of these unlabeled data given the label information and the hypergraph structure. Figure 5.1 shows that the label propagation process is to propagate the label information from these labeled vertices to the unlabeled vertices.

When propagating label information, vertices within the same hyperedge are more likely to have the same label because they characterize themselves with similar attributes in some aspects, and therefore, they have a higher probability of sharing the same label. Under this assumption, the label propagation task can be transformed into a hypergraph cut. In a hypergraph cut, the goal is to make the cut edges as sparse as possible, with each vertex set after the cut as dense as possible. After cutting the hypergraph, different sets of vertices have different labels. This approach satisfies the goal based on the above assumption. The form of the hypergraph cut can be described below.

Suppose a vertex set $S \in \mathcal{V}$ and its compliment $\overline{S}$. There is a cut that splits the $\mathcal{V}$ into S and $\overline{S}$. A hyperedge e is cut if it is incident with the vertices in both S and $\overline{S}$. Define the hyperedge boundary ∂S as the cut hyperedges, i.e., $\partial S = \{e \in \mathcal{E} | e \cap S \neq \varnothing, e \cap \overline{S} \neq \varnothing\}$, and the volume of S, $vol(S)$, be the sum of the degrees of vertices in S, i.e., $vol(S) = \sum_{v \in S} \mathbf{D}_v(v)$. It can be shown as

$$vol(\partial S) = \sum_{e \in \partial S} w(e) \frac{|e \cap S||e \cap \overline{S}|}{\mathbf{D}_e(e)}. \tag{5.1}$$

The derivation is shown as follows, and the details can be found in [7]. Suppose that hyperedge e is a clique, i.e., a fully connected graph. To avoid confusion, the edges in the clique are called subedges. Then, the weight $\frac{w(e)}{\mathbf{D}_e(e)}$ is assigned to each subedge. When the hyperedge e is cut, $|e \cap S| \times |e \cap \overline{S}|$ subedges are cut. The volume of the cut is the sum of the weights over these subedges. Recall that our goal is to make the cut edges as sparse as possible, with each vertex set after the cut as dense

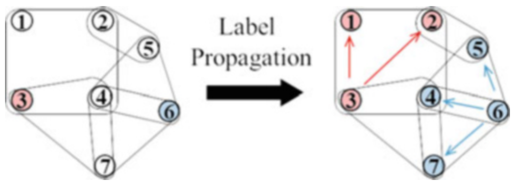
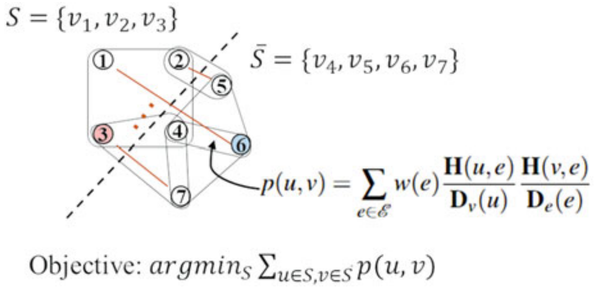


Fig. 5.1 An illustration of the label propagation on hypergraphs



$S = \{v_1, v_2, v_3\}$
$\bar{S} = \{v_4, v_5, v_6, v_7\}$
1
2
3
4
5
6
7
$p(u,v) = \sum_{e\in\mathcal{E}} w(e) \frac{\mathbf{H}(u,e)}{\mathbf{D}_v(u)} \frac{\mathbf{H}(v,e)}{\mathbf{D}_e(e)}$
Objective: $argmin_S \sum_{u\in S, v\in \bar{S}} p(u,v)$

Fig. 5.2 An illustration of the hypergraph label propagation based on random walks

as possible. Based on the goal, the objective partition formula is written as

$$\arg\min_{S\subset\mathcal{V}} c(S) = vol(\partial S)\left(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})}\right). \tag{5.2}$$

There are many methods to propagate label information on a hypergraph, and the propagation based on random walks is the most widely used. The following describes the label propagation by random walk, and the illustration is shown as Fig. 5.2. Suppose that the current position is $u \in \mathcal{V}$, and at first, we walk to a hyperedge e over all hyperedges incident with u with probability $w(e)$, and then we sample a vertex $v \in e$ uniformly. By generalizing from typical random walks on graphs, we use **P** as the transition probability matrix of the random walk on a hypergraph, and the element $p(u, v)$ is defined as follows:

$$p(u, v) = \sum_{e\in\mathcal{E}} w(e) \frac{\mathbf{H}(u, e)}{\mathbf{D}_v(u)} \frac{\mathbf{H}(v, e)}{\mathbf{D}_e(e)}. \tag{5.3}$$

The formula can be organized into a matrix form as $\mathbf{P} = \mathbf{D}_v^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top$. The stationary distribution π of the random walk is defined as

$$\pi(v) = \frac{d(v)}{vol(\mathcal{V})}, \tag{5.4}$$

where $\mathbf{D}_v(v)$ is denoted by $d(v)$ for short and $vol(.)$ is the volume of the vertices in set S, defined as $vol(S) = \sum_{v\in S} d(v)$. The formula can be derived from

$$\begin{aligned}
\sum_{u\in\mathcal{V}} \pi(u)p(u, v) &= \sum_{u\in\mathcal{V}} \frac{d(u)}{vol(\mathcal{V})} \sum_{e\in\mathcal{E}} w(e) \frac{\mathbf{H}(u, e)}{\mathbf{D}_v(u)} \frac{\mathbf{H}(v, e)}{\mathbf{D}_e(e)} \\
&= \frac{1}{vol(\mathcal{V})} \sum_{e\in\mathcal{E}} w(e) \sum_{u\in\mathcal{V}} \mathbf{H}(u, e) \frac{\mathbf{H}(u, e)}{\mathbf{D}_e(e)} \\
&= \frac{1}{vol(\mathcal{V})} \sum_{e\in\mathcal{E}} w(e)\mathbf{H}(v, e) = \frac{d(v)}{vol(\mathcal{V})}.
\end{aligned} \tag{5.5}$$

The objective function Eq. (5.2) can be written as

$$c(S) = \frac{vol(\partial S)}{vol(\mathscr{V})}\left(\frac{1}{vol(S)/vol(\mathscr{V})} + \frac{1}{vol(\overline{S})/vol(\mathscr{V})}\right), \tag{5.6}$$

and then we arrive at

$$\frac{vol(S)}{vol(\mathscr{V})} = \sum_{v\in S}\frac{d(v)}{vol(\mathscr{V})} = \sum_{v\in\mathscr{V}}\pi(v), \tag{5.7}$$

where $\frac{vol(S)}{vol(\mathscr{V})}$ is the probability of random walks to vertex in S. It can then be shown as

$$\begin{aligned}\frac{vol(\partial S)}{vol(\mathscr{V})} &= \sum_{e\in\partial S}\frac{w(e)}{vol(\mathscr{V})}\frac{|e\cap S||e\cap\overline{S}|}{\delta(e)}\\ &= \sum_{e\in\partial S}\sum_{u\in e\cap S}\sum_{v\in e\cap\overline{S}}\frac{w(e)}{vol(\mathscr{V})}\frac{\mathbf{H}(u,e)\mathbf{H}(v,e)}{\delta(e)}\\ &= \sum_{e\in\partial S}\sum_{u\in e\cap S}\sum_{v\in e\cap\overline{S}}w(e)\frac{d(u)}{vol(\mathscr{V})}\frac{\mathbf{H}(u,e)}{d(u)}\frac{\mathbf{H}(v,e)}{\delta(e)}\\ &= \sum_{u\in e\cap S}\sum_{v\in e\cap\overline{S}}\frac{d(u)}{vol(\mathscr{V})}\sum_{e\in\partial S}w(e)\frac{\mathbf{H}(u,e)}{d(u)}\frac{\mathbf{H}(v,e)}{\delta(e)}\\ &= \sum_{u\in S}\sum_{v\in\overline{S}}\pi(u)p(u,v),\end{aligned} \tag{5.8}$$

where the ratio $\frac{vol(\partial S)}{vol(\mathscr{V})}$ is the probability with the random walk from a vertex in S to $\overline{S}$ under the stationary distribution. It can be seen that the hypergraph normalized cut criterion is to search a cut such that the probability with which the random walk crosses different clusters is as small as possible, while the probability with which the random walk stays in the same cluster is as large as possible.

Let us review the objective function Eq. (5.2). Note that it is NP complete, while it can be relaxed into the following optimization problem as

$$\begin{aligned}&\arg\min_{\mathbf{f}\in\mathbb{R}^{|V|}}\Omega(\mathbf{f}) = \frac{1}{2}\sum_{e\in\mathscr{E}}\sum_{\{u,v\}\in e}\frac{w(e)}{\delta(e)}\left(\frac{\mathbf{f}(u)}{\sqrt{d(u)}} - \frac{\mathbf{f}(v)}{\sqrt{d(v)}}\right)^2,\\ &s.t.\ \sum_{v\in\mathscr{V}}\mathbf{f}^2(v) = 1, \sum_{v\in\mathscr{V}}\mathbf{f}(v)\sqrt{d(v)} = 0,\end{aligned} \tag{5.9}$$

where $\mathbf{f}$ is the to-be-learned score vector. Since the goal is label propagation, it can be arrived at for some labeled data. The optimization problem becomes the transductive inference problem as

$$\arg\min_{\mathbf{f}\in\mathbb{R}^{|\mathcal{V}|}} \{\Omega(\mathbf{f}) + \lambda R_{emp}(\mathbf{f})\}, \tag{5.10}$$

where the regularizer term is $\Omega(\mathbf{f})$, the empirical loss term is $R_{emp}(\mathbf{f}) = \|f - y\|^2 = \sum_{v\in\mathcal{V}}(\mathbf{f}(v) - \mathbf{y}(v))^2$, $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}|}$ is the label vector, and λ is the balance parameter. Let us assume that the i-th vertex is labeled, and the elements of $\mathbf{y}$ are all 0 except the i-th value that is 1. The regularizer $\Omega(\mathbf{f})$ can be turned into

$$\begin{aligned}\Omega(\mathbf{f}) =& \frac{1}{2}\sum_{e\in\mathcal{E}}\sum_{\{u,v\}\in e}\frac{w(e)}{\delta(e)}\left(\frac{\mathbf{f}(u)}{\sqrt{d(u)}} - \frac{\mathbf{f}(v)}{\sqrt{d(v)}}\right)^2\\ =&\sum_{e\in\mathcal{E}}\sum_{\{u,v\}\in\mathcal{V}}\frac{w(e)\mathbf{H}(u,e)\mathbf{H}(v,e)}{\delta(e)}\left(\frac{\mathbf{f}^2(u)}{d(u)} - \frac{\mathbf{f}(u)\mathbf{f}(v)}{\sqrt{d(u)d(v)}}\right)\\ =&\sum_{u\in\mathcal{V}}\mathbf{f}^2(u)\sum_{e\in\mathcal{E}}\frac{w(e)\mathbf{H}(u,e)}{d(u)}\sum_{v\in\mathcal{V}}\frac{\mathbf{H}(v,e)}{\delta(e)}\\ &-\sum_{e\in\mathcal{E}}\sum_{u,v\in\mathcal{V}}\frac{\mathbf{f}(u)\mathbf{H}(u,e)w(e)\mathbf{H}(v,e)\mathbf{f}(v)}{\sqrt{d(u)d(v)}\delta(e)}\\ =&\mathbf{f}^\top(\mathbf{I}-\Theta)\mathbf{f},\end{aligned} \tag{5.11}$$

where $\Theta = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}$. The hypergraph Laplacian is denoted by $\Lambda = \mathbf{I} - \Theta$. Therefore, the objective function can be rewritten as

$$\Omega(\mathbf{f}) = \mathbf{f}^\top\Lambda\mathbf{f}. \tag{5.12}$$

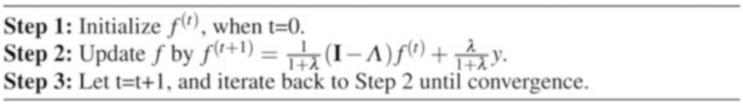
The optimization function can be turned into

$$\arg\min_{\mathbf{f}\in\mathbb{R}^{|\mathcal{V}|}} \{\mathbf{f}^\top\Lambda\mathbf{f} + \lambda\|\mathbf{f}-\mathbf{y}\|^2\}. \tag{5.13}$$

There are two ways to solve the above problem. The first one is differentiating the objective function in Eq. (5.13) with respect to f, and it can be obtained as

$$\mathbf{f} = \left(\mathbf{I} + \frac{1}{\lambda}\Lambda\right)^{-1}\mathbf{y}. \tag{5.14}$$

The second one is an iterative method. Similar to the iterative approach in [8], Eq. (5.13) can be efficiently solved by an iterative process. The process is illustrated in Fig. 5.3. The $\mathbf{f}^{t+1}$ can be obtained from the last iterative $\mathbf{f}^t$ and $\mathbf{y}$, and the procedure is repeated until convergence.



Step 1: Initialize $f^{(t)}$, when t=0.
Step 2: Update f by $f^{(t+1)} = \frac{1}{1+\lambda}(\mathbf{I} - \Lambda)f^{(t)} + \frac{\lambda}{1+\lambda}y$.
Step 3: Let t=t+1, and iterate back to Step 2 until convergence.

Fig. 5.3 The iterative solution of Eq. (5.13). This figure is from [1]

This process will converge to the solution Eq. (5.14). To prove it, we first prove that the eigenvalues of Θ are in $[-1, 1]$. Since $\Theta = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$, we find that its eigenvalues are in $[-1, 1]$. Therefore, $(\mathbf{I} \pm \Theta)$ are positive semi-definite.

The convergence of the iterative process is proved in [1]. Without loss of generality, we assume $\mathbf{f}^{(0)} = y$. From the iterative process, it can be obtained that

$$\begin{aligned}\mathbf{f}^{(t)} &= \left(\frac{\lambda}{1+\lambda}\right)\sum_{i=0}^{t-1}\left(\frac{1}{1+\lambda}\Theta\right)^i \mathbf{y} + \left(\frac{1}{1+\lambda}\Theta\right)^t \mathbf{y} \\ &= (1-\zeta)\sum_{i=0}^{t-1}(\zeta\Theta)^i\mathbf{y} + (\zeta\Theta)^t\mathbf{y},\end{aligned} \tag{5.15}$$

where $\zeta = \frac{1}{1+\lambda}$. Since $0 < \zeta < 1$, and the eigenvalues of Θ are in $[-1, 1]$, it can be derived that

$$\lim_{t\to\infty}(\zeta\Theta)^t = 0 \tag{5.16}$$

and

$$\lim_{t\to\infty}\sum_{i=0}^{t-1}(\zeta\Theta)^i = (\mathbf{I} - \zeta\Theta)^{-1}. \tag{5.17}$$

Then, it turns out

$$\mathbf{f} = \lim_{t\to\infty}\mathbf{f}^{(t)} = (1-\zeta)(\mathbf{I} - \zeta\Theta)^{-1}\mathbf{y} = \left(\mathbf{I} + \frac{1}{\lambda}\Delta\right)^{-1}\mathbf{y}. \tag{5.18}$$

Therefore, the convergence of **f** is proved to be equal to the closed-form solution Eq. (5.14).

The random-walk-based method is the most commonly used approach in label propagation on hypergraphs. It has the advantages of being simple to implement and theoretically verifiable.

In many cases, different hypergraphs may be generated based on different criteria. Under such circumstances, we need to conduct label propagation on multi-hypergraph. Here, we briefly introduce the cross diffusion method on multi-hypergraph [2]. We assume that there are T hypergraphs, and the t-th hypergraph is denoted as $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \mathbf{W}^t)$, where $\mathcal{V}^t$ is the vertex set, $\mathcal{E}^t$ is the hyperedge set, and $\mathbf{W}^t$ is a diagonal matrix, representing the weights of hyperedges.

The transition matrix is first generated for each hypergraph. The label propagation process on hypergraph is based on the assumption that the local similarities could approximate the long-range similarities, and therefore, the local similarities are more important than far-away vertices. The similarity matrix among vertices of the t-th hypergraph is shown as follows:

$$\Lambda^t(u, v) = \sum_{e \in \mathcal{E}^t} \frac{\mathbf{W}^t(e)\mathbf{H}^t(u, e)\mathbf{H}^t(v, e)}{\delta(e)}, \tag{5.19}$$

or in the matrix form:

$$\Lambda^t = \mathbf{H}^t\mathbf{W}^t\mathbf{D}_e^{t^{-1}}\mathbf{H}^{t^\top}. \tag{5.20}$$

The transition matrix $\mathbf{P}^t$ is the normalized similarity matrix:

$$\mathbf{P}^t(i, j) = \frac{\Lambda_t(i, j)}{\sum_{w \in \mathcal{V}^t} \Lambda_t(i, w)} \tag{5.21}$$

and

$$\mathbf{P}^t = \mathbf{D}^{t^{-1}}\Lambda^t, \tag{5.22}$$

where $\mathbf{D}^t$ is a diagonal matrix with the i-th diagonal element $\mathbf{D}^t(i, i) = \sum_{j=1}^{|\mathcal{V}^t|} \Lambda^t(i, j)$.

The element of the transition matrix $\mathbf{P}^t(i, j)$ represents the probability of transition from the vertex i to the vertex j, and $\mathbf{P}^t$ could be regarded as the Parzen window estimators on hypergraph structure. After the generation of the transition matrix, the cross label propagation process is applied to the multi-hypergraph structure.

Denote $\mathbf{Y}_0$ as the initial label matrix. For labeled vertices, the i-th row of $\mathbf{Y}_0$ is the one-hot label of the i-th vertex, while for the unlabeled vertices, all elements of the i-th row are 0.5, indicating that there is no prior knowledge of the label. We denote the labeled part of the initial label matrix as $\mathbf{Y}_0^L$.

For simplicity, we assume the number of hypergraphs T is 2. The label propagation process for multi-hypergraph uses the output of one hypergraph as the input of the other hypergraph, which repeats until the output converges. The process could be formulated as

$$\mathbf{Y}_{d+1}^1 \leftarrow \mathbf{P}^1\mathbf{Y}_d^2, \tag{5.23}$$

$$\mathbf{Y}_{d+1}^{1L} \leftarrow \mathbf{Y}_0^L \tag{5.24}$$

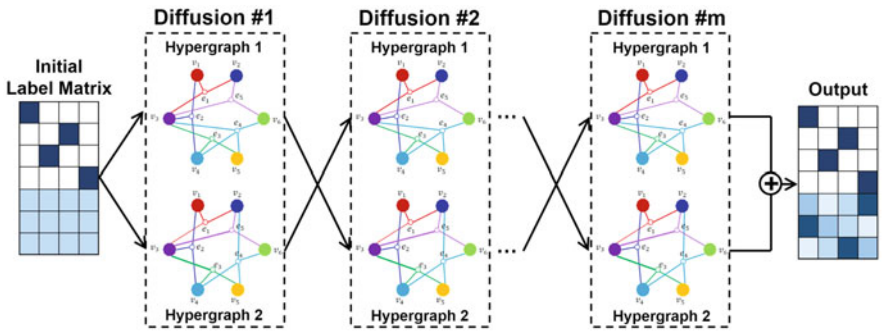


Fig. 5.4 An illustration of the diffusion process on multi-hypergraph. This figure is from [2]

and

$$\mathbf{Y}^2_{d+1} \leftarrow \mathbf{P}^2\mathbf{Y}^1_d, \tag{5.25}$$

$$\mathbf{Y}^{2L}_{d+1} \leftarrow \mathbf{Y}^L_0, \tag{5.26}$$

where $\mathbf{Y}^k_d$ denotes the label matrix of the k-th hypergraph after d times of label propagation. This process is shown in Fig. 5.4.

The overall matrix could be calculated according to the label matrix of each hypergraph after convergence:

$$\mathbf{Y}_{final} = \frac{1}{T}\sum_{i=1}^{T}\mathbf{Y}^i_d. \tag{5.27}$$

For more complicated scenarios, where more than two hypergraphs are available, the label propagation process can repeat that, and the output of one hypergraph can be used as the input of other hypergraphs.

This diffusion process can also be used for a single hypergraph, and the framework can be described in Fig. 5.5.

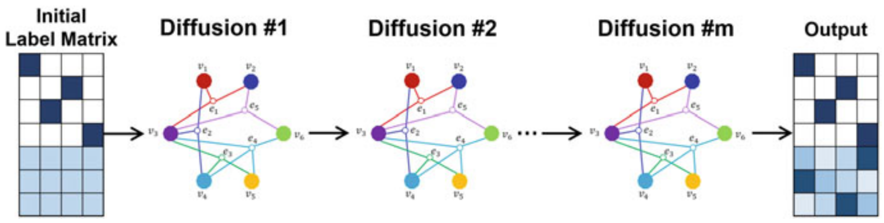


Fig. 5.5 An illustration of the diffusion process on a single hypergraph

5.3 Data Clustering on Hypergraph

Data clustering is a typical machine learning task that aims to group data into clusters. In this section, we introduce hypergraph-based data clustering methods, which can utilize the hypergraph structure for better finding correlations behind the data. For hypergraph clustering, two types of information can be used, including structural hypergraph clustering and attribute hypergraph clustering according to the data information in the hypergraph. In structural hypergraph, the clustering tasks only use structural information. For example, the hypergraph spectral clustering method[7] is extended on the basis of graph, which uses the hypergraph Laplacian to learn complex relations between nodes in the hypergraph. And some auto-encoder-based techniques[9] are also applied to structural clustering. In attribute hypergraph, each vertex is usually accompanied by attribute information from the real world. There are two assumptions as follows:

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

How to balance graph structure information and node feature information is a study focus of attributed graph clustering [10]. In this way, hypergraphs can utilize the features, attributes, and structured information of vertices to conduct data clustering task.

In this section, we introduce a hypergraph Laplacian smoothing filter and an embedded model called adaptive hypergraph auto-encoder (AHGAE) that is designed specifically for hypergraph clustering tasks [3]. First, we describe the hypergraph Laplacian smoothing filter and derive its low-pass filtering properties in the frequency domain. Then, we analyze the influence of each vertex on the attributes of its connected hyperedges and the feature of neighbor vertices. Finally, we introduce the detailed procedure and framework of the adaptive hypergraph auto-encoder.

The hypergraph Laplacian smoothing filter, as shown in Fig. 5.6, first merges the vertex features into hyperedge features, and the feature of hyperedge e_k is defined as

$$\mathbf{E}_k^{(t)} = \frac{1}{|N(e_k)|} \sum_{v_j \in N(e_k)} \mathbf{X}_j^{(t)} = \sum_{v_j \in \mathcal{V}} \frac{h(j,k)}{d_e(k)} \mathbf{X}_j^{(t)}, \tag{5.28}$$

where e_k denotes the k-th hyperedge in the hyperedge set $\mathcal{E}$, v_i denotes the i-th vertex in the vertex set $\mathcal{V}$, t represents the order, $N(e_k)$ is the vertex set in hyperedge e_k, $\mathbf{E}_k$ describes the hyperedge e_k feature, and $\mathbf{X}_j$ describes the feature of the vertex v_j.

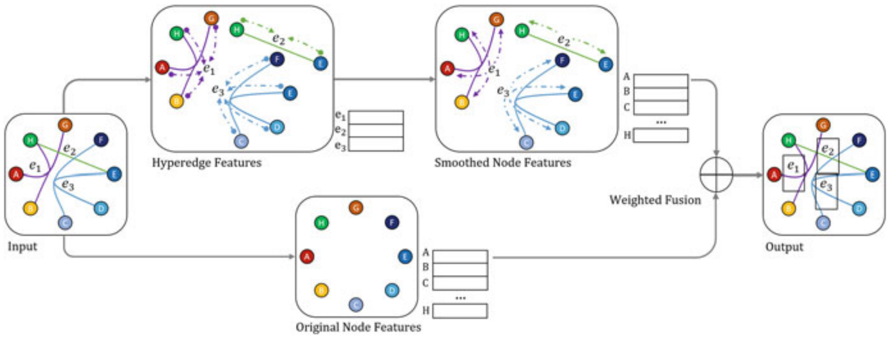


Fig. 5.6 An illustration for hypergraph Laplacian smoothing filter. This figure is from [3]

After aggregating the vertex features to get the hyperedge features, we can further combine the vertex features according to the hyperedge weights:

$$\begin{aligned}\mathbf{X}_i^{(t+1)} &= (1-\gamma)\mathbf{X}_i^{(t)} + \gamma \sum_{e_k \in N(v_i)} \frac{h(i,k)w(k)}{d_v(i)}\mathbf{E}_k^{(t)} \\ &= (1-\gamma)\mathbf{X}_i^{(t)} + \gamma \sum_{v_j \in \mathcal{V}} \sum_{e_k \in \mathcal{E}} \frac{h(i,k)w(k)h(j,k)}{d_v(i)d_e(k)}\mathbf{X}_j^{(t)}, \\ \mathbf{X}^{(t+1)} &= (1-\gamma)\mathbf{X}^{(t)} + \gamma \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{X}^{(t)},\end{aligned} \tag{5.29}$$

where $N(v)$ represents the hyperedge connected to vertex v, and $\gamma \in [0, 1]$ is the weight coefficient of the filter. $\mathbf{D}_v$ denotes the diagonal matrix of the vertex degrees, $\mathbf{D}_e$ denotes the diagonal matrix of the hyperedge degrees, and $\mathbf{H}$ is the incidence matrix of the hypergraph. In order to make the spectral radius less than 1, we can replace $\mathbf{D}_{\text{v}}^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top$ with symmetric normalized form:

$$\begin{aligned}\mathbf{X}^{(t+1)} &= (1-\gamma)\mathbf{X}^{(t)} + \gamma \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\mathbf{X}^{(t)} \\ &= \mathbf{X}^{(t)} - \gamma\left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top \mathbf{D}_v^{-1/2}\right)\mathbf{X}^{(t)}.\end{aligned} \tag{5.30}$$

Then, the multi-order hypergraph Laplacian smoothing filter can be written as

$$\mathbf{X}^{(t)} = (\mathbf{I} - \gamma \mathbf{L})^t \mathbf{X}. \tag{5.31}$$

After decomposing the eigenvalues of the hypergraph Laplacian operator $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$, the diagonal elements of the diagonal matrix $\boldsymbol{\Lambda}$ are eigenvalues of $\mathbf{L}$. The

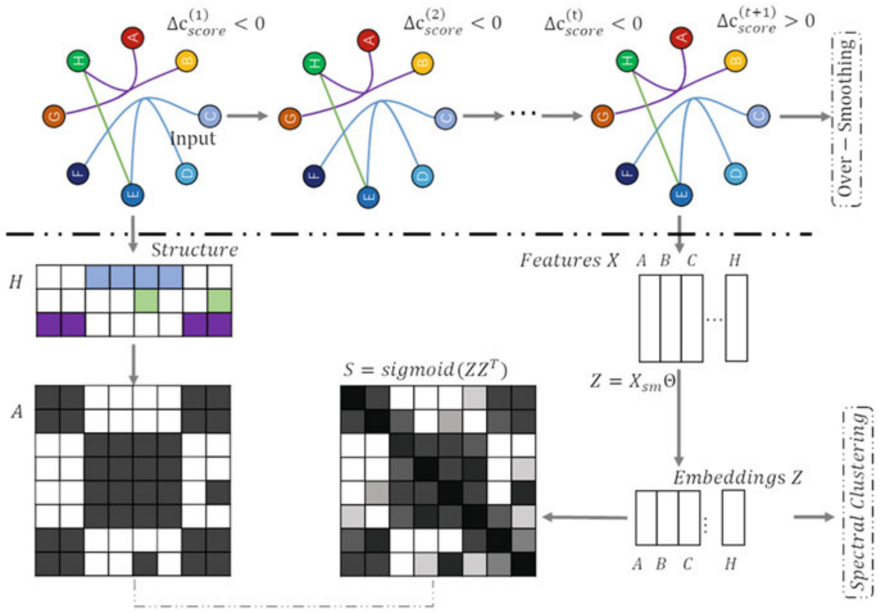

$\Delta c_{score}^{(1)} < 0$
$\Delta c_{score}^{(2)} < 0$
$\Delta c_{score}^{(t)} < 0$
$\Delta c_{score}^{(t+1)} > 0$
Input
Over – Smoothing
Structure
H
A
Features X
$S = sigmoid(ZZ^T)$
$Z = X_{sm}\Theta$
Embeddings Z
Spectral Clustering

Fig. 5.7 The framework of the adaptive hypergraph auto-encoder framework. This figure is from [3]

frequency response function is as

$$p(\boldsymbol{\Lambda}) = \operatorname{diag}\left(p\left(\lambda_1\right), \ldots, p\left(\lambda_{|\mathcal{V}|}\right)\right), \tag{5.32}$$

$$p(\lambda) = 1 - \gamma\lambda, \gamma \in [0, 1]. \tag{5.33}$$

Due to the eigenvalue of the hypergraph Laplacian $\lambda \in [0, 1]$, $p(\Lambda)$ is a positive semi-definite matrix, and the value of $p(\lambda)$ decreases as λ increases. Therefore, the hypergraph Laplacian smoothed filtered can effectively suppress high-frequency signals:

$$\mathbf{F} = \mathbf{U}p(\boldsymbol{\Lambda})\mathbf{U}^{-1} = \mathbf{U}(\mathbf{I} - \gamma\boldsymbol{\Lambda})\mathbf{U}^{-1} = \mathbf{I} - \gamma\mathbf{L}. \tag{5.34}$$

Figure 5.7 illustrates how to use the relational reconstruction auto-encoder after getting the smoothed feature matrix to conduct vertex representation learning in low-dimensional environments without losing information. First, the incidence matrix is used to generate the adjacency matrix:

$$\mathbf{A} = \varepsilon\left(\mathbf{H}\mathbf{H}^{\top}\right), \tag{5.35}$$

$$\varepsilon(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \end{cases}. \tag{5.36}$$

A single fully connected layer is used to compress the filtered feature matrix:

$$\mathbf{Z} = \text{scale}\left(\mathbf{X}_{\mathbf{sm}}\boldsymbol{\Theta}\right), \tag{5.37}$$

$$\text{scale}(\mathbf{x}) = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}, \tag{5.38}$$

where $\mathbf{Z}$ represents the vertex embedding matrix, which includes both structural and feature information, and $\boldsymbol{\Theta}$ is the learnable parameter that is used to extract features from the vertices. In order to rescale the range of vertex characteristics to [0, 1], scale $(\cdot)$ represents a normalization function. So the following is the similarity matrix for vertex features:

$$\mathbf{S} = \text{sigmoid}\left(\mathbf{Z}\mathbf{Z}^{\top}\right), \tag{5.39}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{5.40}$$

This is the inner product decoder used to reconstruct vertex and its neighbors. The objective is to minimize the error between the adjacency matrix $\mathbf{A}$ and the similarity matrix $\mathbf{S}$. However, using Eq. (5.35) to construct an adjacency matrix leads to a problem: the number of edges is too large when the hyperedge degree increases. To solve this problem, the elements in matrix $\mathbf{A}$ are weighted as

$$\mathbf{W}_{ij} = \begin{cases} \frac{|\mathcal{V}|^2 - \sum\sum \mathbf{A}_{ij}}{\sum\sum \mathbf{A}_{ij}} & , \mathbf{A}_{ij} = 1 \\ 1 & , \mathbf{A}_{ij} = 0 \end{cases}. \tag{5.41}$$

The reconstruction loss can be calculated by using the weighted binary cross-entropy function:

$$L_{re} = \frac{1}{|\mathcal{V}|^2} \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} -\mathbf{W}_{ij}\left[\mathbf{A}_{ij} \log \mathbf{S}_{ij} + \left(1 - \mathbf{A}_{ij}\right) \log\left(1 - \mathbf{S}_{ij}\right)\right]. \tag{5.42}$$

The relational reconstruction auto-encoder can be trained to produce the learned vertex embeddings, and the spectral clustering technique can be further used to obtain the final clustering results.

5.4 Cost-Sensitive Learning on Hypergraph

Most of the machine learning applications may suffer from cost-sensitive scenarios. It is noted that different types of faults in real-world jobs might result in losses with varying severity. In diagnostic work, for example, misdiagnosing a patient as a healthy person is significantly more erroneous than classifying a healthy individual as a patient, as shown in Fig. 5.8. Similar cases also happen in the application of software defect prediction. Misjudging the flaws of software modules as a good one may destroy the software system and have disastrous repercussions in software defect prediction. In these cases, cost-sensitive learning methods [11–13] have been developed to deal with these issues.

In many cases, the data from a group of categories may be enough, while the data from other categories may be very limited. These imbalanced data distributions lead to different costs for the classification performance of different categories. Under such circumstances, imbalanced learning [13, 14] attracts much attention, which aims to attain a predictive prediction using imbalanced sampling data. In traditional methods, sampling methods [15, 16] are used to over-sample the minority class and under-sample the majority class to solve the imbalanced sample problem. Another way is to conduct cost-sensitive learning that can focus more on the minority class.

To confront the cost-sensitive issue in hypergraph computation, in this section, we introduce cost-sensitive hypergraph computation framework [4] and cost interval optimization for hypergraph computation [5], respectively. First, we describe how to quantify cost in the hypergraph modeling procedure [4], in which a fixed cost value is provided for modeling, and thereafter, we illustrate how to use the cost-sensitive hypergraph computation approach to tackle imbalanced problems. As the cost value for mis-classification results may not be feasible in practice, we then introduce the hypergraph computation method with cost interval optimization [5], which can utilize the cost chosen inside the interval while generating data with high-order relations. Figure 5.9 shows the frameworks of hypergraph computation under cost-sensitive scenarios, from traditional hypergraph modeling, hypergraph modeling with cost matrix, to hypergraph modeling with cost matrix using cost interval.

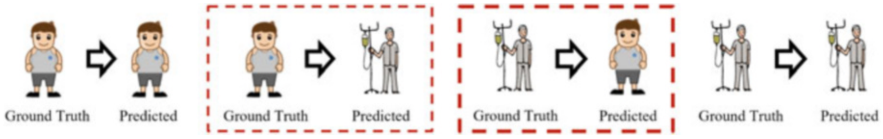


Fig. 5.8 A medical example of cost-sensitive classification scenario

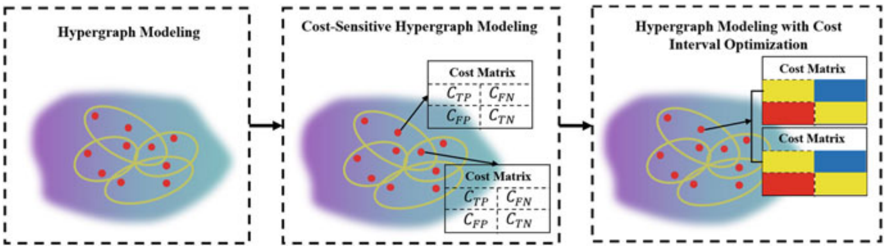


Fig. 5.9 The frameworks of hypergraph computation under cost-sensitive scenarios

(1) Cost-Sensitive Hypergraph Computation

In this part, we introduce a cost-sensitive hypergraph computation method [4], and Fig. 5.10 shows the framework of this method. This framework consists of two stages to handle the cost-sensitive issue: F-measure is used in the initial step to calculate candidate cost information for cost-sensitive learning, and then the hypergraph structure is utilized to model the high-order correlations among the data in the second stage.

First, we introduce the hypergraph modeling with cost matrix. In traditional hypergraph modeling, each vertex represents a subject, and the hyperedges connect related vertices. To introduce cost information in hypergraph modeling, a cost matrix is associated with each vertex, indicating different costs for misclassification, as shown in Fig. 5.11 for a binary classification task. The definition of cost matrix is as follows.

As shown in Fig. 5.11, the cost matrix is a 2×2 matrix, including the true positive cost C_{TP}, the true negative cost C_{TN}, the false positive cost C_{FP}, and the false negative cost C_{FN}, respectively. The true positive cost and the true negative cost are mostly 0 in the matrix since that denotes the correct prediction. The cost-sensitive hypergraph's propensity for each class is achieved by giving various values to the false positive cost and the false negative cost in the cost matrix. A special case is that, if the false positive cost and the false negative cost are equal, then the cost-sensitive hypergraph reduces to traditional hypergraph modeling.

We generate candidate cost information at first and then apply F-measure to reduce the expense for both binary and multi-class data. For a classifier h, we can define the error profile as

$$\Psi(h) = \left(\mathrm{FN}_1(h), \mathrm{FP}_1(h), \ldots, \mathrm{FN}_{N_c}(h), \mathrm{FP}_{N_c}(h)\right), \tag{5.43}$$

where N_c represents the number of classes, and FN and FP represent the false negative and the false positive probabilities. For simplicity, we let ψ_{2k-1} represent the FN possibility of the k-th class and ψ_{2k} represent the FP possibility of the k-th

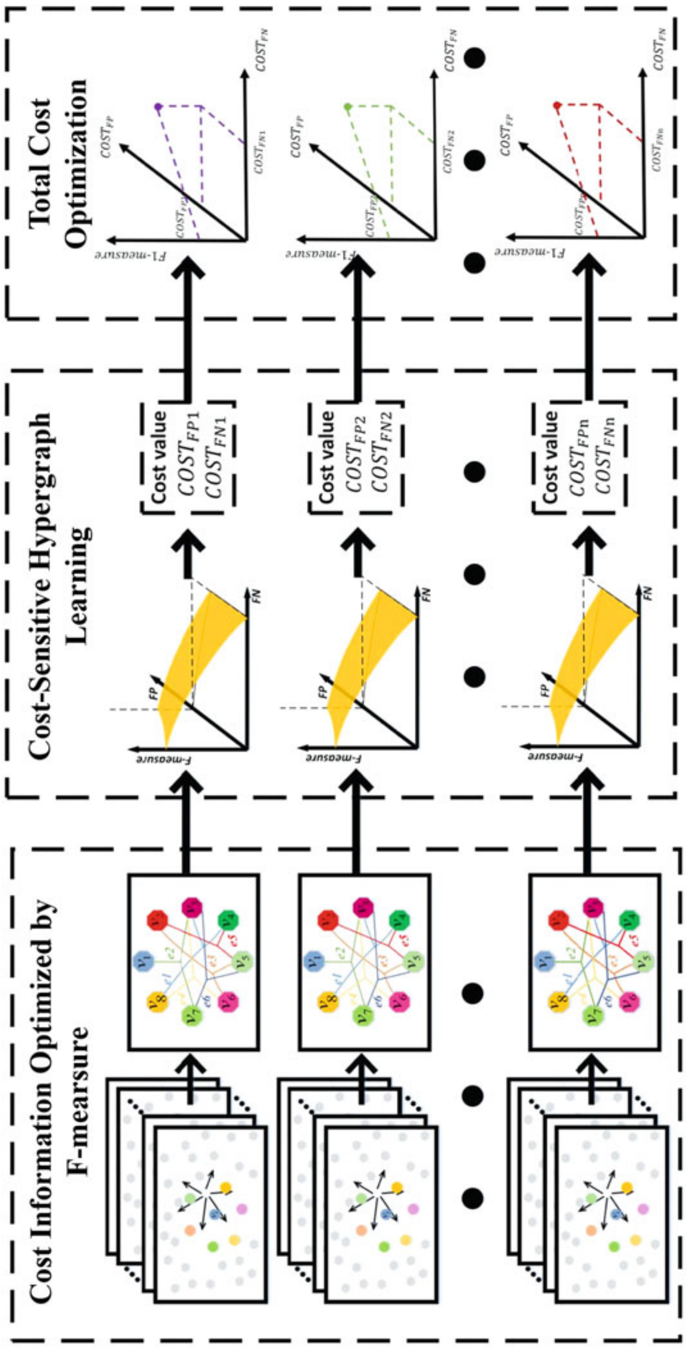


Fig. 5.10 The framework of cost-sensitive hypergraph computation framework. This figure is from [4]

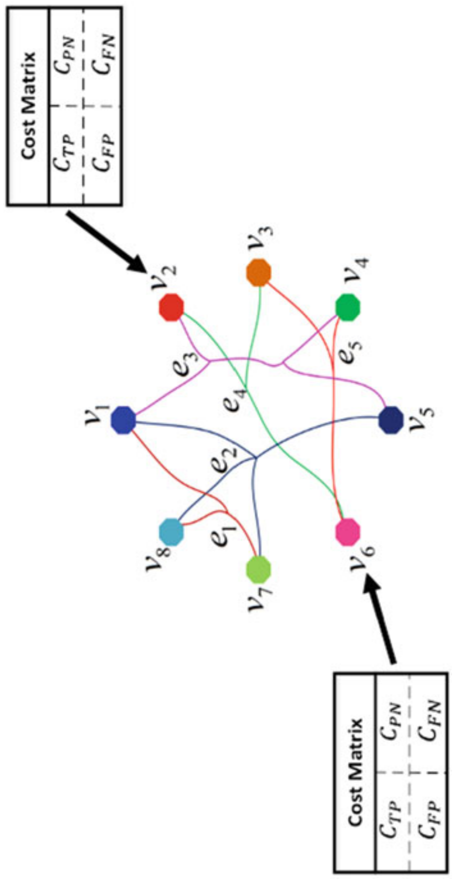


Fig. 5.11 An illustration of hypergraph modeling with cost matrix

class. The F-measure for binary classification can be defined as

$$F_\beta(\Psi) = \frac{\left(1+\beta^2\right)(P_1 - \psi_1)}{\left(1+\beta^2\right) P_1 - \psi_1(h) + \psi_2(h)}, \tag{5.44}$$

where P_k represents the marginal probability of class k. Similarly, the micro-F-measure for multi-class classification can be defined as

$$mcF_\beta(\Psi) = \frac{\left(1+\beta^2\right)\left(1 - P_1 - \sum_{k=2}^{C} \psi_{2k-1}\right)}{\left(1+\beta^2\right)(1-P_1) - \sum_{k=2}^{C} \psi_{2k-1} + \psi_1}. \tag{5.45}$$

We can further divide the F-measure values in the region [0, 1] into a collection of equally spaced values $F = \{f_i\}$ to calculate the cost of various mis-classifications. The cost function Υ is then used to construct the cost vector using every f_i. For binary classification, we constrain the denominator of Eq. (5.44) to be positive and $F_\beta(\Psi) \leq f_i$ for a value c of the F-measure:

$$\left(1+\beta^2 - f\right)\psi_1 + f\psi_2 + \left(1+\beta^2\right) P_1(f-1) \geq 0. \tag{5.46}$$

Therefore, the cost of ψ_1 and ψ_2 can be allocated according to f and $1+\beta^2-f$, and the cost function can be written as follows:

$$\Upsilon_i^{F_\beta} = \begin{cases} 1+\beta^2-f, & \text{if sample from class 1} \\ f, & \text{if sample from class 2} \\ 0, & \text{otherwise} \end{cases}. \tag{5.47}$$

Similarly, the cost function of multi-class classification can be written as follows:

$$\Upsilon_i^{mlF_\beta} = \begin{cases} 1+\beta^2-f, & \text{if sample from odd class and not from class 1} \\ f, & \text{if sample from class 1} \\ 0, & \text{otherwise} \end{cases}. \tag{5.48}$$

The cost of F-measure optimization is added to the optimization function to increase the efficacy of the hypergraph computation method in imbalanced data. We first regard each data to be a vertex of the hypergraph and then apply the k nearest neighbor algorithm to construct the hypergraph. The cost-sensitive hypergraph differs in that it includes the cost matrix information of each vertex in addition to the original hypergraph correlation structure. With training and testing samples represented by $\mathbf{O}$, cost-sensitive hypergraph computation function can be expressed as

$$\begin{aligned} &\arg\min_{\omega,\mathbf{W}}\Big\{\mu\Omega(\omega) + \mathscr{R}_{emp}(\omega) + \lambda\Phi(\mathbf{W})\Big\}, \\ &s.t. \sum_{j=1}^{N}\mathbf{W}_{j,j} = 1, \forall\, \mathbf{W}_{j,j} \geq 0, \end{aligned} \tag{5.49}$$

where $\Omega(\omega) = (\mathbf{O}\omega)^\top \Delta(\mathbf{O}\omega)$ represents the hypergraph Laplacian regularized with hypergraph Laplacian Δ, $\mathscr{R}_{emp}(\omega) = \|\Upsilon(\mathbf{O}\omega - \mathbf{y})\|_2^2 = \sum_{i=1}^{N}\left(\Upsilon_{i,i}\left(\mathbf{o}_i\omega - \mathbf{y}_i\right)\right)^2$ is the empirical loss using cost information with diagonal matrix Υ that $\Upsilon_{i,i}$ represents the cost of the i-th data, $\Phi(\mathbf{W}) = \lambda\|\mathbf{W}\|_{\mathrm{F}}^2$ stands for the hypergraph regularization, ω represents the mapping vector to be learnt, $\mathbf{W}$ is a diagonal matrix representing hyperedge weights, and μ and λ are the trade-off hyperparameter. We first fix $\mathbf{W}$ to optimize ω, and then the optimization equation can be expressed as

$$\arg\min_{\omega}\left\{\|\Upsilon(\mathbf{O}\omega) - \mathbf{y}\|_2^2 + \mu(\mathbf{O}\omega)^\top\Delta(\mathbf{O}\omega)\right\}. \tag{5.50}$$

The optimal ω can be obtained as

$$\omega = \left(\mathbf{O}^\top\Upsilon^2\mathbf{O} + \mu\mathbf{O}^\top\Delta\mathbf{O}\right)^{-1}\left(\mathbf{O}^\top\Upsilon\mathbf{y}\right). \tag{5.51}$$

Following that, we fix ω to enhance $\mathbf{W}$:

$$\begin{aligned} &\arg\min_{\mathbf{W}}\Big\{\mu(\mathbf{O}\omega)^\top\Delta(\mathbf{O}\omega) + \lambda\|\mathbf{W}\|_F^2\Big\}. \\ &s.t. \sum_{j=1}^{N}\mathbf{W}_{j,j} = 1, \forall\, \mathbf{W}_{j,j} \geq 0. \end{aligned} \tag{5.52}$$

We can have $\mathbf{W}$ as

$$\mathbf{W} = \frac{\mu\Lambda^\top\Lambda\left(\mathbf{D}_e\right)^{-1} - \eta\mathbf{I}}{2\lambda}, \tag{5.53}$$

where η can be calculated as $\eta = \frac{\mu\Lambda(\mathbf{D}_e)^{-1}\Lambda^\top - 2\lambda}{N}$, and Λ can be calculated as $\Lambda = (\mathbf{O}\omega)^\top\left(\mathbf{D}_v\right)^{-1/2}\mathbf{H}$. The optimized mapping vector ω allows sample ζ_i in the test set to obtain the classification result $\gamma = \zeta_i\omega$.

Each piece of potential cost information c_i generates a cost matrix Υ, which is then used to build a cost-sensitive hypergraph structure $\mathscr{G}_i$. The model then employs an efficient collection to choose the cost-sensitive hypergraph with the greatest F-measure as the best choice.

(2) Cost Interval Optimization for Hypergraph Computation

As the cost value for cost-sensitive hypergraph modeling is not easy to be determined in practice, in this part, we introduce a cost interval optimization method for hypergraph computation [5], in which the fixed cost value is replaced by a cost interval, which is much easier to be provided than a fixed cost value.

Given a hypergraph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathbf{W})$, the regularization foundation of the cost-sensitive hypergraph can be divided into three components, i.e., empirical loss using cost information, the hypergraph Laplacian regularizer, and the hypergraph regularization, in order to optimize the overall cost by adding the mis-classification costs of various categories to the hypergraph framework.

The empirical loss using cost information can be formulated as

$$\mathscr{R}_{emp}(\omega) = \|\Phi(\mathbf{S}\omega - \mathbf{y})\|_2^2 = \sum_{i=1}^{N_v} \left(\Phi_{i,i}\left(\mathbf{s}_i\omega - \mathbf{y}_i\right)\right)^2, \tag{5.54}$$

where ω represents the mapping vector, and Φ is a diagonal matrix representing mis-classification cost weights. The hypergraph Laplacian regularizer can be written as

$$\begin{aligned}\Omega(\omega) &= \frac{1}{2}\sum_{e\in\mathscr{E}}\sum_{v_i,v_j\in\mathscr{V}} \frac{\mathbf{W}(e)\mathbf{H}(v_i,e)\mathbf{H}\left(v_j,e\right)}{\delta(e)}\left(\frac{\omega\mathbf{s}_i}{\sqrt{d(v_i)}} - \frac{\omega s_j}{\sqrt{d\left(v_j\right)}}\right)^2 \\ &= (\mathbf{S}\omega)^\top\Delta(\mathbf{S}\omega).\end{aligned} \tag{5.55}$$

To adjust the hyperedges weights and hence the hypergraph classification ability, the hypergraph regularization is written as $\Psi(\mathbf{W}) = \|\mathbf{W}\|_F^2$. It is noted that this part can be removed in different applications, if not required.

Combining the above three, the whole optimization task for cost-sensitive hypergraph computation can be written as

$$\begin{aligned}&\arg\min_{\omega,\mathbf{W}}\left\{\|\Phi(\mathbf{S}\omega - \mathbf{y})\|_2^2 + \mu(\mathbf{S}\omega)^\top\Delta(\mathbf{S}\omega) + \lambda\|\mathbf{W}\|_\mathrm{F}^2\right\}, \\ &s.t. \sum_{j=1}^{N_e}\mathbf{W}_{j,j} = 1, \forall\, \mathbf{W}_{j,j} \geq 0,\end{aligned} \tag{5.56}$$

where μ and λ are the trade-off hyperparameters.

The precise cost of each category is required for cost-sensitive hypergraph computation, but the cost is frequently impossible to be obtained, and it can only be known that the cost is within a cost interval $[C_{max}, C_{min}]$. Therefore, a simple idea is to attempt all values inside the cost interval and minimize the overall cost. However, this is inefficient given the possibly huge cost interval. As the actual cost is difficult to establish, we need to find a surrogate cost c^* to guide the optimization procedure, and the surrogate classifier h^* is supposed to be as successful as the true cost classifier h^t. In this way, the problem can be formulated as

$$\begin{aligned} &\min_{h,c^*} L(h, c^*), \\ &s.t.\ p(L(h,c) < \theta) > 1 - \varphi, \forall\, c \in [C_{min}, C_{max}], C_{min} \leq c^* \leq C_{max}, \end{aligned} \tag{5.57}$$

where $L(h,c)$ is the empirical risk. $L(h,c)$ is formulated as $L(h,c) = \sum_{i=1}^{N_v} cI(\rho_i \neq y \wedge y = +) + I(\rho_i \neq y \wedge y = -)$, where $\rho_i = s_i\omega$ is the i-th data labeling in the test set, and $+$ and $-$ represent the label of the important class and the unimportant class, respectively.

The worst-case risk is considered first to guarantee that all limitations can be fulfilled. The worst-case classifier h^* can be written as

$$h^* = \arg \min_h \sup_c L(h, c) \tag{5.58}$$

and

$$p\left(\sup_c L\,(h_*, c) < \theta\right) > 1 - \varphi. \tag{5.59}$$

We have $p\,(L\,(h_*, c) < \theta) > 1 - \varphi$ for any c. The worst-case risk is attained when the surrogate cost c^* equals C_{max}. However, only a solution that meets the requirements can be acquired in this manner, and the cost cannot be guaranteed to be close to the true cost. As the average cost is the smallest maximum distortion of the genuine risk, it is another good choice, which can be calculated as $C_{mean} = 0.5(C_{max} + C_{min})$.

With the use of alternative costs C_{max} and C_{mean}, we can conduct cost interval optimization. First, C_{max} is used as a surrogate cost, and a collection of cost-sensitive hypergraph structures with varying parameter values is learned in the first stage. Then, C_{mean} is used as a surrogate cost to determine the lowest overall cost on the valid dataset, and then we choose the hypergraph structure as the final solution.

In this section, we describe cost-sensitive hypergraph computation methods. Imbalanced data issue is very common in many applications. The cost-sensitive hypergraph computation methods introduce cost matrix in hypergraph modeling, and both fixed cost value and cost interval can be used in the learning process.

5.5 Link Prediction on Hypergraph

Link prediction is a fundamental task in network analysis. The objective of link prediction is to predict whether two vertices in a network may have a link. Link prediction has wide applications in different domains, such as social relation exploration [17, 18], protein interaction prediction [19, 20], and recommender system [21, 22], which has attracted much attention in the past decades.

Link prediction on hypergraph aims to discover missing relations or predict new coming hyperedges based on the observed hypergraph, where hypergraph computation can be used to deeply exploit the underneath high-order correlations among these data. Unlike the link prediction task on the graph structure [23, 24], the hypergraph models the high-order correlation among the data, which is heterogeneous in many applications, as the vertices are in different types. For example, in a bibliographic network, the vertex can represent a paper, an author, or a venue, while the hyperedge represents the relation where the paper is written by multiple authors and published in a venue. These different types of vertices do not necessarily share the same representation space. The heterogeneous hypergraph consists of two kinds of vertex in the view of the hypergraph event, i.e., identifier vertex and slave vertex. Identifier vertex is the vertex that determines a hyperedge uniquely, while slave vertex is the other vertex except for the identifier vertex. In this section, we introduce the Heterogeneous Hypergraph Variational Auto-encoder (HeteHG-VAE) method [6] for heterogeneous hypergraph link prediction task.

The overview of HeteHG-VAE can be found in Fig. 5.12. HeteHG-VAE aims to learn the low-dimensional heterogeneous hypergraph embedding based on the Bayesian deep generative strategy. The input hypergraph is represented by the incidence matrix $\mathbf{H}$, whose sub-hypergraph represents the hypergraph generated by different types of slave vertices. The heterogeneous encoder can project the vertices and the hyperedges to the vertex embedding and hyperedge embedding, respectively. The hypergraph embedding is the combination of the vertex embedding and the hyperedge embedding, which can be used for reconstructing the incidence matrix by the hypergraph decoder.

In the following part of this section, we first introduce the variational evidence lower bound with the task specific derivation. Then, the inference model, including the heterogeneous vertex encoder and the heterogeneous hyperedge encoder, is presented. At last, the generative model and the link prediction method are introduced.

Denote $\{x_k\}_{k=1}^{K}$ as the observed data with the total number K, $\mathbf{Z}_k^V$ as the latent vertex embedding, and Z^E as the latent hyperedge embedding. HeteHG-VAE assumes that $\mathbf{Z}_k^V$ and $\mathbf{Z}^E$ are drawn *i.i.d.* from a Gaussian prior, i.e., $\mathbf{Z}_k^V \sim p_0(\mathbf{Z}_k^V)$ and $\mathbf{Z}^E \sim p_0(\mathbf{Z}^E)$, and x_k are drawn from the conditional distribution, $x_k \sim p(x_k|\mathbf{Z}_k^V, Z^E; \lambda_k)$, where λ_k is the parameter of the distribution. The objective of HeteHG-VAE is to maximize the log-likelihood of the observed data by optimizing

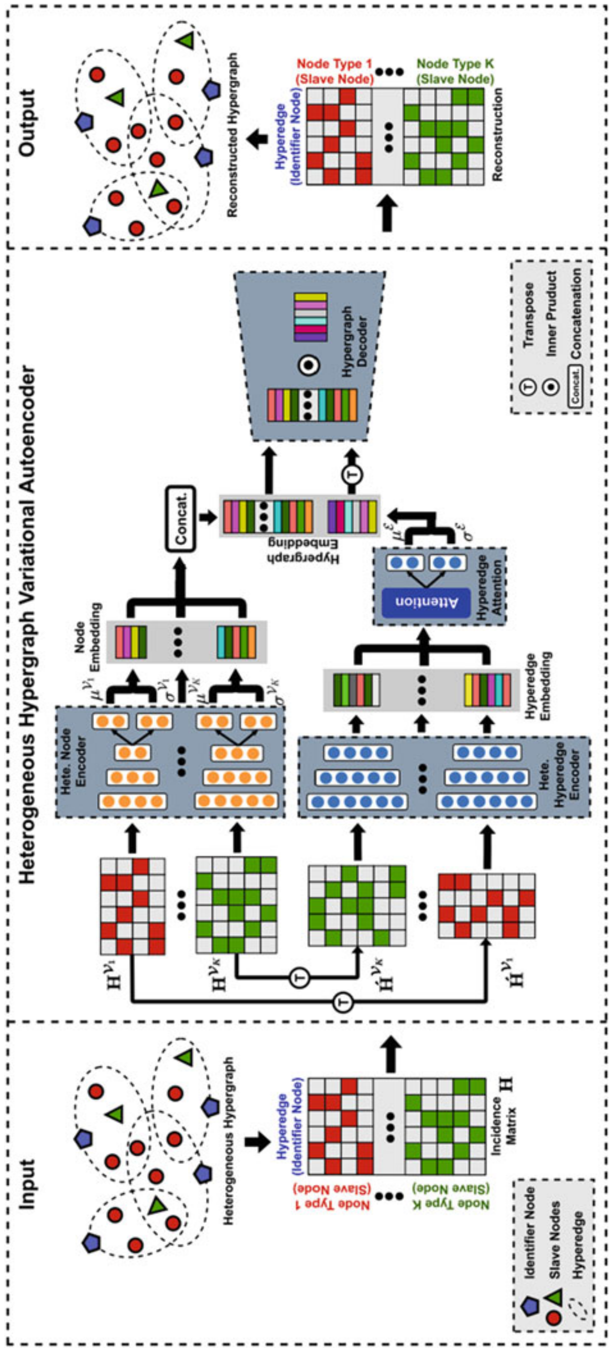


Fig. 5.12 An illustration of the HeteHG-VAE method. This figure is from [6]

λ_k as follows:

$$\begin{aligned}
&\log p(x_1, \cdots, x_K; \lambda) \\
&= \log \int_{\mathbf{Z}_1^V} \cdots \int_{\mathbf{Z}_K^V} \int_{\mathbf{Z}^E} p(x_1, \cdots, x_K, \mathbf{Z}_1^V, \cdots, \mathbf{Z}_K^V, \mathbf{Z}^E; \lambda) d\mathbf{Z}_1^V \cdots d\mathbf{Z}_K^V d\mathbf{Z}^E \\
&\geq \mathbb{E}_q \left(\log \frac{p(x_1, \cdots, x_K, \mathbf{Z}_1^V, \cdots, \mathbf{Z}_K^V, \mathbf{Z}^E; \lambda)}{q(\mathbf{Z}_1^V, \cdots, \mathbf{Z}_K^V, \mathbf{Z}^E | x_1, \cdots, x_K; \theta)} \right) \\
&:= \mathscr{L}(x_1, \cdots, x_K; \theta, \lambda),
\end{aligned} \tag{5.60}$$

where $q(\cdot)$ is the variational posterior for the estimation of the true posterior $p(\mathbf{Z}_1^V, \ldots, \mathbf{Z}_K^V, \mathbf{Z}^E | x_1, \ldots, x_K)$, which is inaccessible, and θ is the parameter to be estimated. Then, $\mathscr{L}(x_1, \ldots, x_K; \theta, \lambda)$ is the evidence lower bound of the log marginal likelihood. Based on the evidence lower bound, an inference encoder is presented to parameterize q, and a generative decoder is used to parameterize p.

The inference encoder of HeteHG-VAE consists of two main parts, i.e., the heterogeneous vertex encoder and the heterogeneous hyperedge encoder. Heterogeneous vertex encoder first maps the observed data x_k to a latent space $\tilde{\mathbf{Z}}_k^V$, which can be written as

$$\tilde{\mathbf{Z}}_k^V = f^V(x_k \mathbf{W}_k^V + b_k^V), \tag{5.61}$$

where $\mathbf{W}_k^V$ and b_k^V are the to-be-learned weights of the model, and f^V is a nonlinear activation function. Two separated linear layers map the latent representation of the means μ_k^V and variances σ_k^V of q:

$$\mu_k^V = \tilde{\mathbf{Z}}_k^V \mathbf{W}_k^{V\mu} + b_k^{V\mu}, \tag{5.62}$$

$$\sigma_k^V = \tilde{\mathbf{Z}}_k^V \mathbf{W}_k^{V\sigma} + b_k^{V\sigma}, \tag{5.63}$$

where $\mathbf{W}_k^{V\mu}$, $b_k^{V\mu}$, $\mathbf{W}_k^{V\sigma}$, and $b_k^{V\sigma}$ are learnable parameters. The vertex embedding is the sample from the Gaussian distribution $\mathcal{N}(\mu_k^V, \sigma_k^V)$.

Heterogeneous hyperedge encoder first maps the observed data x_k to a latent space $\tilde{\mathbf{Z}}_k^E$, which can be written as

$$\tilde{\mathbf{Z}}_k^E = f^E(x_k^\top \mathbf{W}_k^E + b_k^E), \tag{5.64}$$

where $\mathbf{W}_k^E$ and b_k^E are the to-be-learned weights of the model, and f^E is a nonlinear activation function. Then, the importance of different types of vertices is learned by the hyperedge attention mechanism, which can be written as

$$\tilde{\alpha}_k = \mathrm{Tan}h(\tilde{\mathbf{Z}}_k^E \mathbf{W}_k^{E\alpha} + b_k^{E\alpha})\mathbf{P}, \tag{5.65}$$

where $\mathbf{W}_k^{E\alpha}$, $b_k^{E\alpha}$, and $\mathbf{P}$ are learnable parameters. The attention score α_k is obtained by normalizing $\tilde{\alpha}_k$, and the hyperedge embedding can be written as

$$\tilde{\mathbf{Z}}^E = \sum_{k=1}^{K} \alpha_k \tilde{\mathbf{Z}}_k^E. \tag{5.66}$$

Similarly, two separated linear layers map the latent representation of the means μ^E and variances σ^E of the distribution q:

$$\mu^E = \tilde{\mathbf{Z}}^E \mathbf{W}^{E\mu} + b^{E\mu}, \tag{5.67}$$

$$\sigma^E = \tilde{\mathbf{Z}}^E \mathbf{W}^{E\sigma} + b^{E\sigma}, \tag{5.68}$$

where $\mathbf{W}^{E\mu}$, $b^{E\mu}$, $\mathbf{W}^{E\sigma}$, and $b^{E\sigma}$ are learnable parameters. The vertex embedding is the sample from the Gaussian distribution $\mathscr{N}(\mu^E, \sigma^E)$.

The incidence matrix is sampled from a Bernoulli distribution parameterized by $\mathscr{H}_k$:

$$p(\mathbf{H}_{ij} | \mathbf{Z}_{k,i}^V, \mathbf{Z}_{k,j}^E; \lambda_k) = Ber(\mathscr{H}_{ij}), \tag{5.69}$$

where $\mathscr{H}_{ij}$ is the dot product of the vertex embedding and the hyperedge embedding:

$$\mathscr{H}_{ij} = Sigmoid(\mathbf{Z}_{k,i}^V (\mathbf{Z}_j^E)^\top). \tag{5.70}$$

The likelihood of the connection among vertices could be obtained based on the vertex embedding and hyperedge embedding as follows:

$$p_{conn}(\mathbf{Z}_i^V, \mathbf{Z}_j^E) = ||\mathbf{Z}_i^V, \mathbf{Z}_j^E||_2. \tag{5.71}$$

In this section, we have introduced the Heterogeneous Hypergraph Variational Auto-encoder method [6] for the task of link prediction on hypergraph, which captures the high-order correlations among the data while preserving the origin low-order topology. Link prediction on hypergraph has shown superior performance in different experiments and can be further used in other applications.

5.6 Summary

In this chapter, we introduce four typical hypergraph computation tasks, including label propagation, data clustering, imbalance learning, and link prediction. Label propagation on hypergraph is to predict the labels for the vertices on a hypergraph, i.e., assigning a label to each unlabeled vertex in the hypergraph, based on the labeled information. Data clustering on hypergraph divides the vertices in a hypergraph into several groups. Imbalanced learning on hypergraph considers the imbalanced data distributions and introduces cost-sensitive hypergraph computation methods. Link prediction on hypergraph discovers missing relations or predicts new coming hyperedges based on the observed hypergraph. We note that these four tasks are typical ways to use hypergraph computation in practice. Other tasks can also be deployed under the hypergraph computation framework, such as data regression, data completion, and data generation. Following these typical hypergraph computation tasks, we can use them in different applications, such as social media analysis and computation vision.

References

1. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
2. Z. Zhang, H. Lin, J. Zhu, X. Zhao, Y. Gao, Cross diffusion on multi-hypergraph for multi-modal 3d object recognition, in *Proceedings of Pacific Rim Conference on Multimedia* (2018), pp. 38–49
3. Y. Hu, X. Li, Y. Wang, Y. Wu, Y. Zhao, C. Yan, Y. Gao, Adaptive hypergraph auto-encoder for relational data clustering. IEEE Trans. Knowl. Data Eng. (2021)
4. N. Wang, R. Liang, X. Zhao, Y. Gao, Cost-sensitive hypergraph learning with F-measure optimization. IEEE Trans. Cyber. (2021) pp. 1–12
5. X. Zhao, N. Nan, H. Shi, H. Wan, J. Huang, Y. Gao, Hypergraph learning with cost interval optimization, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (2018), pp. 4522–4529
6. H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, Q. Dai, Heterogeneous hypergraph variational autoencoder for link prediction. IEEE Trans. Pattern Analy. Mach. Intell. **44**(8), 4125–4138 (2021)
7. D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: clustering, classification, and embedding, in *Proceedings of Advances in Neural Information Processing Systems* (2006), pp. 1601–1608
8. D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in *Proceedings of the Advances in Neural Information Processing Systems* (2003), pp. 321–328
9. F. Ye, C. Chen, Z. Zheng, Deep autoencoder-like nonnegative matrix factorization for community detection, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1393–1402
10. T. Yang, R. Jin, Y.Chi, S. Zhu, Combining link and content for community detection: a discriminative approach, in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), pp. 927–936
11. C. Elkan, The foundations of cost-sensitive learning, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2001), pp. 973–978
12. C. Zhang, K. Tan, H. Li, G. Hong, A cost-sensitive deep belief network for imbalanced classification. IEEE Trans. Neural Netw. Learn. Syst. **30**(1), 109–122 (2018)
13. D. Tomar, S. Agarwal, Prediction of defective software modules using class imbalance learning, in *Proceedings of the Applied Computational Intelligence and Soft Computing* (2016)
14. N. Wang, X. Zhao, Y. Jiang, Y. Gao, Iterative metric learning for imbalance data classification, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), pp. 2805–2811

15. S. Barua, Md.M. Islam, X. Yao, K. Murase, MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans. Knowl. Data Eng. **26**(2), 405–425 (2012)
16. P. Sobhani, H. Viktor, S. Matwin, Learning from imbalanced data using ensemble methods and cluster-based undersampling, in *Proceedings of the International Workshop on New Frontiers in Mining Complex Patterns* (2014), pp. 69–83
17. D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, in *Proceedings of the 2003 ACM International Conference on Information and Knowledge Management* (2003), pp. 556–559
18. L. Liao, X. He, H. Zhang, T. Chua, Attributed social network embedding. IEEE Trans. Knowl. Data Eng. **30**(12), 2257–2270 (2018)
19. A. Clauset, C. Moore, M.E. Newman, Hierarchical structure and the prediction of missing links in networks. Nature **453**(7191), 98–101 (2008)
20. M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks. Bioinfor. **34**(13), 457–466 (2018)
21. W. Feng, J. Wang, Incorporating heterogeneous information for personalized tag recommendation in social tagging systems, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2012), pp. 1276–1284
22. C. Shi, B. Hu, W. X. Zhao, P.S. Yu, Heterogeneous information network embedding for recommendation. IEEE Trans. Knowl. Data Eng. **31**(2), 357–370 (2019)
23. A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 855–864
24. M. Zhang, Y. Chen, Link prediction based on graph neural networks, in *Proceedings of the Advances in Neural Information Processing Systems* (2018), pp. 5165–5175

Chapter 6
Hypergraph Structure Evolution

Abstract In practice, noise exists in the process of data collection and hypergraph construction. Therefore, missing, abundant, and noisy connections may be introduced into the generated hypergraph structure, which may lead to inaccurate inference on hypergraph. Another issue comes from the increasing data stream, which is also very common in many applications. It is important to consider the structure evolution methods on the hypergraph, which optimize the hypergraph structure accordingly. Early hypergraph computation methods mainly rely on static hypergraph structure, which may suffer from the limitation of the static mechanism when confronting random and increasing data scenarios. In this chapter, we introduce dynamic hypergraph structure evolution methods, including both hypergraph component optimization and hypergraph structure optimization. Finally, we briefly introduce the incremental learning method on growing data.

6.1 Introduction

The hypergraph structure models the high-order and complex correlations among data, and thus the quality of topology structure plays an important role in learning tasks on hypergraph. As shown in the previous chapter, there have been implicit and explicit methods of hypergraph generation from observed data. However, the generated hypergraph may contain abundant, missing, and noisy connections due to the disturbances in the process of data collection and hypergraph construction. In other words, there may exist biases between the generated hypergraph and the ground truth structure. Under such circumstances, it is essential to optimize the hypergraph structure to make it fit the ground truth high-order correlation more accurately. The quality of a hypergraph can be directly qualified by comparing with the ground truth structure if available, or indirectly evaluated by the performance of downstream applications. Most existing hypergraph computation methods rely on static hypergraph structure, such as k-nn-based method [1], cluster-based

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_6

method [2], and spare representation-based method [3]. These methods may suffer from the inaccurate hypergraph structure that exists in practice. In this chapter, we introduce hypergraph structure evolution methods under the dynamic hypergraph structure learning mechanism. Hypergraph structure evolution can be divided into two main categories, i.e., hypergraph component optimization and hypergraph structure optimization. The problem of hypergraph structure evolution is usually integrated with the learning process and formulated as a bi-level optimization problem. Part of the work introduced in this chapter has been published in [4–7].

6.2 Hypergraph Component Optimization

Besides the main structure of a hypergraph, i.e., the incidence matrix, a hypergraph is also composed of a group of components such as the weights for hyperedges, vertices, and even sub-hypergraphs, which play an important role on the hypergraph structure. Hypergraph component optimization aims to explore the optimal components of the hypergraphs, i.e., hyperedge weights, vertices weights, and sub-hypergraph weights. The hyperedge weights represent the strength of each high-order correlation among data, while the vertex weights represent the importance of different samples on the structure. In many cases, we may construct multiple hypergraphs using multi-modal data or different criteria, which can be regarded as sub-hypergraphs. The sub-hypergraph weights are used to measure the importance of different sub-hypergraphs on the overall structure. The optimization procedure adjusts the hyperedge weights, the vertex weights, and sub-hypergraph weights during the training process in order to improve the performances on the downstream applications.

6.2.1 Hyperedge Weight Optimization

The hyperedge is a basic component of the hypergraph, representing the high-order complex correlation among data. The initial hypergraph usually assigns an identical weight to all hyperedges. However, hyperedges actually have different effects for a given task. The hyperedge weights indicate the importance of different hyperedges contributing to the whole structure. In this section, we introduce the hyperedge weight learning methods [4], in which the weights of hyperedges are adaptively adjusted during the training process, and thus the importance of different hyperedges can be automatically modulated.

We assume that there are m hyperedges in the hypergraph, denoted by $\{e_1, e_2, \ldots, e_n\}$. The weights of the hyperedges are defined by the $n \times 1$ vectors $w = [w_1, w_2, \ldots, w_n]^\top$. There is usually a constraint on the hypergraph weights that their sum is equal to one, i.e., $\sum_{i=1}^{n} \omega_i = 1$. We use $\mathbf{F}$ to denote the output of hypergraph learning. The problem of learning hyperedge weights can be formulated in a dual-optimization form mathematically

$$\begin{aligned} \arg\min_{\mathbf{F}, w} \Psi(\mathbf{F}) &:= \left\{ \Omega(\mathbf{F}) + \lambda R_{\text{emp}}(\mathbf{F}) + \mu \Phi(w) \right\}, \\ s.t. \ \sum_{e \in \mathscr{E}} \mathbf{W}(e) &= 1. \end{aligned} \tag{6.1}$$

Here, $\Omega(\mathbf{F})$ and $R_{\text{emp}}(\mathbf{F})$ are the regularizer and empirical loss of $\mathbf{F}$, respectively. $\Phi(w)$ is the regularizer on w. λ and μ are the scalars controlling the relative importance of these three items.

The general formulation can be implemented by specifying the functions $\Omega(\cdot)$, $R_{\text{emp}}(\cdot)$ and $\Phi(\cdot)$. As said before, $\mathbf{F}$ is the to-be-learned labels in the node classification task. The regularizer $\Omega(\mathbf{F})$ can be defined as $\mathbf{F}^\top \Delta \mathbf{F}$, where Δ is the Laplacian matrix. The empirical loss $R_{\text{emp}}(\mathbf{F})$ in the general form can be instantiated by the difference between the learned $\mathbf{F}$ and observed labels of training data $\mathbf{Y}$, which are called the least residuals. The regularizer on w is a 2-norm. The general formulation can be written as

$$\begin{aligned} \arg\min_{\mathbf{F}, w} \Psi(\mathbf{F}) &:= \left\{ \mathbf{F}^\top \Delta \mathbf{F} + \lambda \|\mathbf{F} - \mathbf{Y}\|^2 + \mu \sum_{i=1}^{n} w_i^2 \right\}, \\ s.t. \ \sum_{i=1}^{n} w_i^2 &= 1. \end{aligned} \tag{6.2}$$

The aim of the learning process is to search the optimal solution of $\mathbf{F}$ and w to minimize the cost function in Eq. (6.2).

There are two variables to be optimized in Eq. (6.2), which can be solved by the alternating optimization algorithm. For each instant in time, one variable is optimized, while the other is kept constant for the to-be-learned two variables $\mathbf{F}$ and w. The details of the alternating optimization strategy are introduced as follows.

Given the initial hyperedge weights, the first step is fixing w and optimizing $\Omega(\mathbf{F})$. The sub-problem is written as

$$\arg\min_{\mathbf{F}} \Psi(\mathbf{F}) = \arg\min_{\mathbf{F}} \left\{ \mathbf{F}^\top \Delta \mathbf{F} + \lambda \|\mathbf{F} - \mathbf{Y}\|^2 \right\}. \tag{6.3}$$

A closed-form solution of Eq. (6.3) has already been achieved from the traditional hypergraph learning. The solution is written as

$$\begin{aligned}\mathbf{F} &= \left(I + \frac{1}{\lambda}\Delta\right)^{-1}\mathbf{Y} \\ &= \left(\mathbf{I} + \frac{1}{\lambda}(\mathbf{I} - \Theta)\right)^{-1}\mathbf{Y} \\ &= \frac{\lambda+1}{\lambda}\left(\mathbf{I} - \frac{1}{\lambda+1}\Theta\right)^{-1}\mathbf{Y}.\end{aligned} \tag{6.4}$$

Let $\zeta = \frac{1}{\lambda+1}$, and Eq. (6.4) can be rewritten as

$$\mathbf{F} = \frac{1}{1-\xi}(\mathbf{I} - \xi\Theta)^{-1}\mathbf{Y}. \tag{6.5}$$

With the updated $\mathbf{F}$, the next step is fixing $\mathbf{F}$ while optimizing w, and the sub-problem about w is

$$\begin{aligned}\arg\min_{w}\Psi(\mathbf{F}) &= \arg\min_{\mathbf{F}}\left\{\mathbf{F}^\top\Delta\mathbf{F} + \mu\sum_{i=1}^{n} w_i^2\right\}, \\ s.t.\ &\sum_{i=1}^{n} w_i = 1, \mu > 0.\end{aligned} \tag{6.6}$$

The Lagrangian multipliers method is employed here, and the sub-problem is replaced with

$$\begin{aligned}&\arg\min_{w,\eta}\mathbf{F}^\top\Delta\mathbf{F} + \mu\sum_{i=1}^{n} w_i^2 + \eta\left(\sum_{i=1}^{n} w_i - 1\right) \\ =\quad &\arg\min_{w,\eta}\mathbf{F}^\top\left(\mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}\right)\mathbf{F} + \mu\sum_{i=1}^{n} w_i^2 + \eta\left(\sum_{i=1}^{n} w_i - 1\right).\end{aligned} \tag{6.7}$$

Let $\Gamma = \mathbf{D}_0^{-\frac{1}{2}}\mathbf{H}$, and it can be shown that

$$\eta = \frac{\mathbf{F}^\top\Gamma\mathbf{F} - 2\mu}{n} \tag{6.8}$$

and

$$w_i = \frac{1}{n} - \frac{\mathbf{F}^\top \Gamma \mathbf{D}_e^{-1} \Gamma^\top \mathbf{F}}{2n\mu} + \frac{\mathbf{F}^\top \Gamma_i \mathbf{D}_e^{-1}(i,i) \Gamma_i^\top \mathbf{F}}{2\mu}. \tag{6.9}$$

Here, Γ_i defines the i-th column of Γ.

In this way, $\mathbf{F}$ and w are alternatively updated until convergence. Finally, the optimal values of $\mathbf{F}$ and w are obtained. We note that the above method is a typical way to optimize the hyperedge weights using the l_2-norm. Other methods can also be used to learn the hyperedge weights using different constraints.

6.2.2 *Vertex Weight Optimization*

Early hypergraph computation methods may not take the importance of vertices into account and mainly focus on the weights of hyperedges. However, the vertex set in the hypergraph may have heterogeneous, unbalanced, and outlier problems, resulting in performance degeneration of learning process. Therefore, it is highly required to consider the weights of vertices to define the impact of different subjects during the learning process. For example, vertices belonging to the minority class may require larger weights and vice versa for imbalanced data. In this part, we introduce the vertex-weighted hypergraph learning method [5], which can update the vertex weights during the learning process.

The aim of vertex-weighted hypergraph learning algorithm is to emphasize the vertices with distinguishable information and disregard the redundant vertices that bring in bias and noise instead of useful information. On the basis of learning hyperedge weights, vertex-weighted learning algorithm further considers the vertex weights. Here let $\{v_1, v_2, \dots, v_n\}$ denote all n vertices in the hypergraph. The corresponding weight for vertex v_i is represented by u_i. Let $\mathbf{U}$ denote the diagonal matrix of vertex weights. The overall cost function is similar to learning hyperedge weights, but with the impact of $\mathbf{U}$ simultaneously taken into consideration. The general formulation is written as

$$\begin{aligned} &\arg\min_{\mathbf{F},w} \Psi_{\mathbf{U}}(\mathbf{F}) := \left\{\Omega_{\mathbf{U}}(\mathbf{F}) + \lambda R_{\text{emp}}(\mathbf{F}) + \mu\Phi(w)\right\}, \\ &s.t.\ \mathbf{W}(e) \leq 0, \sum_{e\in\mathcal{E}} \mathbf{H}(v,e)\mathbf{W}(e) = \mathbf{D}_v(v). \end{aligned} \tag{6.10}$$

The key point of vertex weight optimization is to design a reasonable vertex weighting scheme that scores the importance of each subject during the learning process. First, the pairwise distances between vertices are calculated based on the features. Let d_{ij} denote the distances between vertices v_i and v_j, and $\hat{d}_i$ declares the mean distance between v_i and all other training vertices with the same label. The vertex weight is then defined as

$$u_i = \frac{\hat{d}_i}{\sum_{j=1}^{n_{train}} \hat{d}_j}, \tag{6.11}$$

where n_{train} denotes the number of training samples. It is noted that only the training data are labeled and further weighted. The unlabeled vertices are initialized with an identical weight. Normalization is then applied to the vertex weights. This weighting scheme can assign higher weights to vertices that are far from other intra-class vertices and vice versa. Therefore, the importance of repeated/close samples is relatively smaller than the outliers during the hypergraph learning process.

Since the hypergraph structure is updated with vertex weights, the hypergraph structure regularizer is different from the initial one. As stated already, the hypergraph regularizer is defined based on the cut cost. Here, the cut cost is related to not only just the hyperedge weights but to the vertex weights. In general, the higher the weight of two vertices, the higher the cut cost. Therefore, the regularizer of the hypergraph structure $\Psi_{\mathbf{U}}(\mathbf{F})$ is rewritten as

$$\begin{aligned}
\Omega(\mathbf{F}) &= \sum_{k=1}^{C}\sum_{e\in\mathcal{E}}\sum_{u,v\in\mathcal{V}} \frac{\mathbf{W}(e)\mathbf{U}(u)\mathbf{H}(u,e)\mathbf{U}(v)\mathbf{H}(v,e)}{2\delta(e)}\left(\frac{\mathbf{F}(u,k)}{\sqrt{d(u)}}-\frac{\mathbf{F}(v,k)}{\sqrt{d(v)}}\right)^2\\
&= \sum_{k=1}^{C}\sum_{e\in\mathcal{E}}\sum_{u,v\in\mathcal{V}} \frac{\mathbf{W}(e)\mathbf{U}(u)\mathbf{H}(u,e)\mathbf{U}(v)\mathbf{H}(v,e)}{\delta(e)}\\
&\quad\times\left(\frac{\mathbf{F}(u,k)^2}{d(u)}-\frac{\mathbf{F}(u,k)\mathbf{F}(v,k)}{\sqrt{d(u)d(v)}}\right)\\
&= \sum_{k=1}^{C}\left\{\sum_{u\in\mathcal{V}}\mathbf{U}(u)\mathbf{F}(u,k)^2\sum_{e\in\mathcal{E}}\frac{\mathbf{W}(e)\mathbf{H}(u,e)}{d(u)}\sum_{v\in\mathcal{V}}\frac{\mathbf{H}(v,e)\mathbf{U}(v)}{\delta(e)}\right.\\
&\quad\left.-\sum_{e\in\mathcal{E}}\sum_{u,v\in\mathcal{V}}\frac{\mathbf{F}(u,k)\mathbf{U}(u)\mathbf{H}(u,e)\mathbf{W}(e)\mathbf{H}(v,e)\mathbf{U}(v)\mathbf{F}(v,k)}{\sqrt{d(u)d(v)}\delta(e)}\right\}\\
&= \sum_{k=1}^{C}\mathbf{F}(:,k)^\top\boldsymbol{\Delta}_{\mathbf{U}}\mathbf{F}(:,k)\\
&= \mathbf{F}^\top\boldsymbol{\Delta}_{\mathbf{U}}\mathbf{F}.
\end{aligned} \tag{6.12}$$

Here, $\mathbf{F}(:,k)$ is the k-th column of $\mathbf{F}$ and C is the number of data categories. $\boldsymbol{\Delta}_{\mathbf{U}}$ is the vertex-weighted hypergraph Laplacian, which can be defined as

$$\boldsymbol{\Delta}_{\mathbf{U}} = \mathbf{U} - \Theta = \mathbf{U} - \mathbf{D}_v^{-1/2}\mathbf{U}\mathbf{H}\mathbf{W}\mathbf{D}_e{}^{-1}\mathbf{H}^\top\mathbf{U}\mathbf{D}_v{}^{-1/2}. \tag{6.13}$$

Compared with the traditional hypergraph Laplacian $\boldsymbol{\Delta} = \mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e{}^{-1}\mathbf{H}^\top$ $\mathbf{D}_v{}^{-\frac{1}{2}}$, the hypergraph Laplacian with weighted vertices takes different weights of vertices into consideration during the evaluation of the cost on the hypergraph structure. Therefore, the learning task can be further defined as

$$\begin{aligned} \arg\min_{\mathbf{F},\mathbf{W}} \Psi(\mathbf{F}) &:= \left\{\mathbf{F}^\top \boldsymbol{\Delta}_{\mathbf{U}}\mathbf{F} + \lambda\|\mathbf{F}-\mathbf{Y}\|^2 + \mu \sum_{e\in\mathcal{E}} \mathbf{W}(e)^2\right\}, \\ s.t.\ \mathbf{W}(e) &\geq 0, \sum_{e\in\mathcal{E}} \mathbf{H}(v,e)\mathbf{W}(e) = \mathbf{D}_v(v). \end{aligned} \tag{6.14}$$

The above optimization problem can be solved by the alternative optimization algorithm. The sub-problem about **F** has the closed-form solution as in traditional hypergraph learning. The sub-problem about **W** is written as

$$\begin{aligned} \arg\min_{\mathbf{F},\mathbf{W}} \Psi(\mathbf{F}) &:= \left\{\mathbf{F}^\top \boldsymbol{\Delta}_{\mathbf{U}}\mathbf{F} + \mu \sum_{e\in\mathcal{E}} \mathbf{W}(e)^2\right\}, \\ s.t.\ \mathbf{W}(e) &\geq 0, \sum_{e\in\mathcal{E}} \mathbf{H}(v,e)\mathbf{W}(e) = \mathbf{D}_v(v). \end{aligned} \tag{6.15}$$

The above optimization task can be solved via quadratic programming, since it is convex on **W**. Through vertex weight optimization, the vertex-weighted hypergraph structure takes the contribution of each vertex to the whole hypergraph structure into consideration, and thus it can model the high-order relevance among objects more accurately. During the learning process, the impact of low-quality training samples on the structure and subsequent classification tasks decreases continuously, while high-quality training data, which account for a minority, can be given greater importance. On the other hand, the minority of training data can have greater importance. The additional vertex weights lead to an optimal Laplacian matrix of hypergraph that measures data correlation better than the traditional one and consequently lead to improvement of the classification performance.

6.2.3 Sub-hypergraph Weight Optimization

Given multiple sub-hypergraphs that are used to jointly formulate the correlation among data, it is important to measure how these sub-hypergraphs work in the main task. Sub-hypergraph weight optimization adjusts the importance of the sub-hypergraphs, which models the complex correlation among the multi-model data. In this part, we introduce the inductive multi-hypergraph learning (iMHL) [7] to learn the weights of the model and adjust the weights of the sub-hypergraphs during the training process simultaneously, which models the high-order correlation of

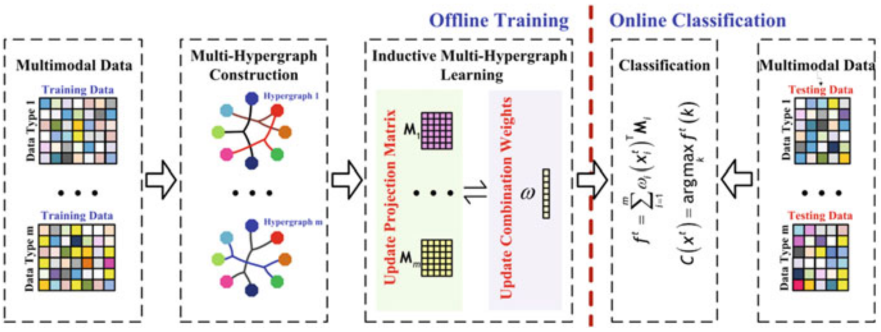


Fig. 6.1 The framework of inductive multi-hypergraph learning method. This figure is from [7]

the multi-model data with the multi-hypergraph, diffuses the sub-hypergraphs as the modality weight, and learns the map from the data to the labels under the supervised setting. Given testing data, the learning projection can be used to predict corresponding labels. The framework of iMHL is illustrated in Fig. 6.1, where the offline training and online training are both supported by the inductive learning process, which can easily handle new coming data efficiently.

Here, we denote m as the total number of all sub-hypergraphs and $\mathscr{G}_i = (\mathscr{V}_i, \mathscr{E}_i, \mathbf{W}_i)$ as the i-th hypergraph for the i-th modality. The projection matrices $\mathbf{M}_i$ are combined as per the sub-hypergraph weights and are used to map the data to the label for prediction. The combination weights $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_m]$ are another object to be optimized, which represents the weight of the corresponding modality, subject to $\sum_{i=1}^{m} \omega_i = 1$ and $\boldsymbol{\omega} \geq 0$.

The loss function $\bar{\Psi}$ for learning all $\mathbf{M}_i$ can be formulated as

$$\bar{\Psi} = \sum_{i=1}^{m} \omega_i \{\Omega(\mathbf{M}_i) + \lambda R_{emp}(\mathbf{M}_i) + \mu \Phi(\mathbf{M}_i)\} + \eta \Gamma(\boldsymbol{\omega}), \tag{6.16}$$

which consists of two main parts, i.e., the summation of the cost of each sub-hypergraph and the regularization on the sub-hypergraph weights $\boldsymbol{\omega}$. $\Phi(\mathbf{M})$ is the regularizer on the projection matrix. We assume that the vertices with similar labels are connected strongly, and $\Omega(\mathbf{M})$ can then be written as

$$\begin{aligned}\Omega(\mathbf{M}) &= \frac{1}{2} \sum_{k=1}^{c} \sum_{e \in \mathscr{E}} \sum_{u,v \in \mathscr{V}} \frac{\mathbf{W}(e)\mathbf{H}(u,e)\mathbf{H}(v,e)}{\delta(e)} \left(\frac{\mathbf{X}^\top \mathbf{M}(u,k)}{\sqrt{d(u)}} - \frac{\mathbf{X}^\top \mathbf{M}(v,k)}{\sqrt{d(v)}} \right)^2 \\ &= \mathrm{tr}(\mathbf{M}^\top \mathbf{X} \Delta \mathbf{X}^\top \mathbf{M}),\end{aligned} \tag{6.17}$$

where Δ denotes the normalized hypergraph Laplacian,

$$\Delta = \mathbf{I} - \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2}. \tag{6.18}$$

The empirical loss term $R_{emp}(\mathbf{M})$ can be written as

$$R_{emp}(\mathbf{M}) = ||\mathbf{X}^\top \mathbf{M} - \mathbf{Y}||^2. \tag{6.19}$$

$\Phi(\mathbf{M})$ can be formulated as the $\ell_{2,1}$-norm of $\mathbf{M}$,

$$\Phi(\mathbf{M}) = ||\mathbf{M}||_{2,1}, \tag{6.20}$$

which produces row sparsity for more informative features. $\Gamma(\omega)$ is the ℓ_2-norm of the sub-hypergraph weights

$$\Gamma(\omega) = ||\omega||^2, \tag{6.21}$$

which aims to learn the optimal weights for each sub-hypergraph.

The inductive multi-hypergraph learning task can be formulated as

$$\begin{gathered}\arg \min_{\mathbf{M}_i, \omega \geq 0} \sum_{i=1}^{m} \omega_i \left(\Omega(\mathbf{M}_i) + \lambda R_{emp}(\mathbf{M}_i) + \mu \Phi(\mathbf{M}_i)\right) + \eta \Gamma(\boldsymbol{\omega}), \\ s.t. \ \sum_{i=1}^{m} \omega_i = 1.\end{gathered} \tag{6.22}$$

It is observed that Eq. (6.22) could be split into $m+1$ independent sub-problems, each $\mathbf{M}_i$ is optimized individually, and the combination weights ω are optimized to fuse all multi-hypergraphs.

The optimization of $\mathbf{M}_i$ shown below can be solved by iterative algorithm.

$$\arg \min_{\mathbf{M}_i} \Omega(\mathbf{M}_i) + \lambda R_{emp}(\mathbf{M}_i) + \mu \Phi(\mathbf{M}_i). \tag{6.23}$$

The optimization problem of ω can then be written as

$$\begin{gathered}\arg \min_{\omega \geq 0} \sum_{i=1}^{m} \omega_i \left(\Omega(\mathbf{M}_i) + \lambda R_{emp}(\mathbf{M}_i) + \mu \Phi(\mathbf{M}_i)\right) + \eta ||\boldsymbol{\omega}||^2, \\ s.t. \ \sum_{i=1}^{m} \omega_i = 1.\end{gathered} \tag{6.24}$$

We denote $\Upsilon_i = \Omega(\mathbf{M}_i) + \lambda R_{emp}(\mathbf{M}_i) + \mu\Phi(\mathbf{M}_i)$, and Eq. (6.24) can be simplified to

$$\begin{aligned} &\arg\min_{\omega\geq 0} \sum_{i=1}^{m} \omega_i \Upsilon_i + \eta||\boldsymbol{\omega}||^2, \\ &s.t. \sum_{i=1}^{m} \omega_i = 1. \end{aligned} \tag{6.25}$$

The Lagrangian algorithm can be applied to solve Eq. (6.25), which can be formulated as

$$\arg\min_{\boldsymbol{\omega},\zeta} \sum_{i=1}^{m} \omega_i \Upsilon_i + \eta||\boldsymbol{\omega}||^2 + \zeta\left(\sum_{i=1}^{m} \omega_i - 1\right). \tag{6.26}$$

Then, we can have

$$\zeta = \frac{-\sum_{i=1}^{m} \Upsilon_i - 2\eta}{m} \tag{6.27}$$

and

$$\omega_i = \frac{1}{m} + \frac{\sum_{i=1}^{m} \Upsilon_i}{2m\eta} - \frac{\Upsilon_i}{2\eta}. \tag{6.28}$$

Given the testing sample $x^t = \{x_1^t, \cdots, x_m^t\}$ features for each modality, the prediction of the corresponding label can be achieved by

$$C(x^t) = \arg\max_{k} \sum_{i=1}^{m} \omega_i {x_i^t}^\top \mathbf{M}_i. \tag{6.29}$$

The overall algorithm is shown in Fig. 6.2. The optimization of sub-hypergraph weights is effective as the incorporation of the multi-modal data via multiple sub-hypergraphs can make it flexible to investigate the contributions of different data or information on the learning process.

6.3 Hypergraph Structure Optimization

Although the above component optimization methods can modify the weights of hyperedges, vertices, or sub-hypergraphs, it is not easy to precisely adjust the inappropriate or wrong connections since the intersections between vertices and hyperedges cannot be changed, i.e., the incidence matrix of the hypergraph is

```
Input: The training data $\mathbf{X} = [x_1, \cdots, x_n] \in \mathbb{R}^{n\times c}$,
    the label matrix $\mathbf{Y} = [y_1, \cdots, y_n] \in \mathbb{R}^{n\times c}$ for $\mathbf{X}$,
    the testing sample $x^t$.
Output: The predicted class for $x^t$.
1: Multi-hypergraph construction.
2: Update $\mathbf{M}_i$ for each hypergraph.
3: Initialize $\mathbf{U}$ as an identity matrix.
4: while not converges do
5:     Update $\mathbf{M}_i$ and $\mathbf{U}_i$ alternately.
6: Update $\boldsymbol{\omega}$ by Eq. (6.28).
7: Predict the class of $x^t$ by Eq. (6.29).
8: return result
```

Fig. 6.2 The workflow for the sub-hypergraph weight optimization method

fixed. To solve this challenge and further optimize the hypergraph structure, it is imperative to investigate how to finely optimize the hypergraph structure and dynamically learn the high-order relationship. It can be regarded as finding the optimal hypergraph structure in a hypergraph space, as shown in Fig. 6.3.

In this part, we introduce the dynamic hypergraph structure learning method [6], and Fig. 6.4 shows the framework of this method. Different from the above methods, structure optimization on incidence matrix aims to optimize the incidence matrix **H**.

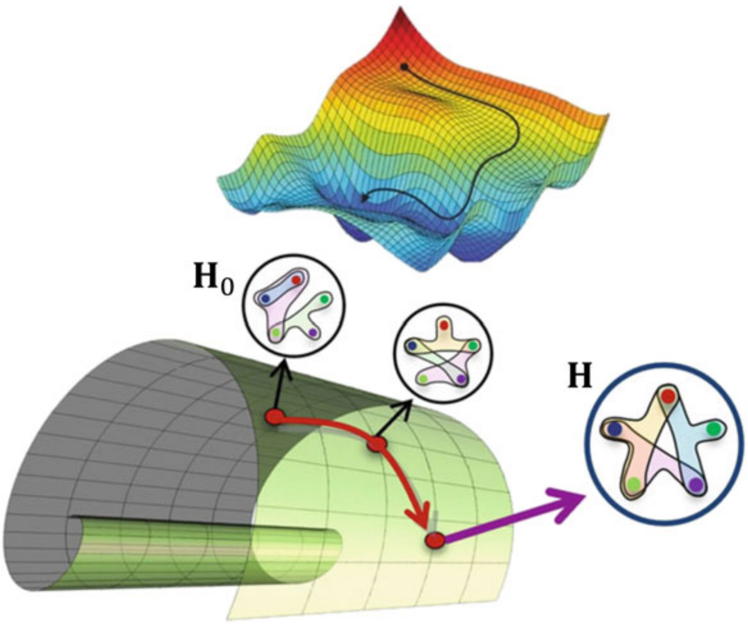


Fig. 6.3 An illustration of hypergraph structure evolution

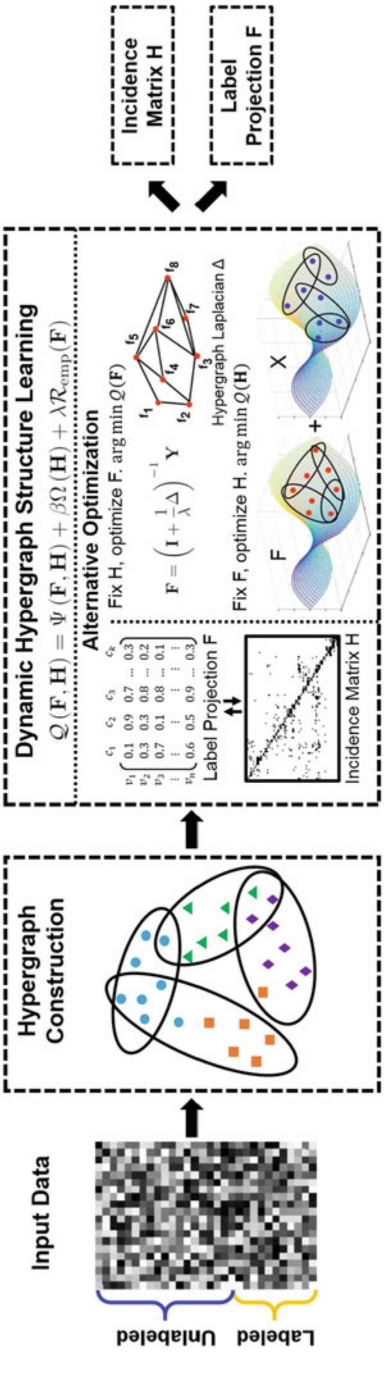

Unlabeled
Labeled
Input Data
Hypergraph Construction
Dynamic Hypergraph Structure Learning
$\mathcal{Q}(\mathbf{F},\mathbf{H}) = \Psi(\mathbf{F},\mathbf{H}) + \beta\Omega(\mathbf{H}) + \lambda\mathcal{R}_{\mathrm{emp}}(\mathbf{F})$
Alternative Optimization
Label Projection F
Incidence Matrix H
Fix H, optimize F. arg min $\mathcal{Q}(\mathbf{F})$
$\mathbf{F} = \left(\mathbf{I} + \frac{1}{\lambda}\Delta\right)^{-1}\mathbf{Y}$
Hypergraph Laplacian Δ
Fix F, optimize H. arg min $\mathcal{Q}(\mathbf{H})$
F
X
Incidence Matrix H
Label Projection F

Fig. 6.4 The framework of dynamic hypergraph structure learning method. This figure is from [6]

The output **F** and the incidence matrix **H** are jointly optimized by the dual-optimization method. The objective function of the joint learning can be formulated as

$$\arg\min_{\mathbf{F},0\preceq\mathbf{H}\preceq 1} \Psi(\mathbf{F}) := \left\{\Omega(\mathbf{F},\mathbf{H}) + \lambda\mathscr{R}_{\text{emp}}(\mathbf{F}) + \mu\Phi(\mathbf{H})\right\}. \tag{6.30}$$

There are three terms in the objective function, explained as follows:

- First, $\Omega(\mathbf{F},\mathbf{H})$ is the regularizer related to **F** and **H**. The output **F** is the to-be-learned label vectors of vertices. Therefore, smoothness is expected to be conducted on the hypergraph structure, where the commonly used regularizer of hypergraph smoothness can be written as

$$\Omega(\mathbf{F},\mathbf{H}) = \operatorname{tr}\left(\mathbf{F}^\top\left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\right)\mathbf{F}\right). \tag{6.31}$$

  However, the regularizer in the previous methods is a function only of **F**, while **H** is a stable parameter. Here, the regularizer is a function of both **F** and **H**.
- Second, the empirical loss $\mathscr{R}_{\text{emp}}(\mathbf{F})$ is the l_2-norm between **F** and **Y**.
- Third, $\Phi(\mathbf{H})$ is the regularizer only related to **H** to additionally constrain **H** to satisfy the prior knowledge. For instance, given the feature information of data, the hypergraph structure is expected to preserve smoothness not just in the label space but in the feature space as well. Let **X** denote the features of vertices, and the regularizer can be formulated as

$$\Phi(\mathbf{F}) = \operatorname{tr}\left(\mathbf{X}^\top\left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\right)\mathbf{X}\right). \tag{6.32}$$

To summarize, the general objective function in Eq. (6.30) for dynamic hypergraph structure learning is instantiated as

$$\begin{aligned}\arg\min_{\mathbf{F},0\preceq\mathbf{H}\preceq 1} \Psi(\mathbf{F}) := \operatorname{tr}&\left(\left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\right)\left(\mathbf{F}\mathbf{F}^\top + \mu\mathbf{X}\mathbf{X}^\top\right)\right)\\ &+ \lambda\|\mathbf{F}-\mathbf{Y}\|^2.\end{aligned} \tag{6.33}$$

Similar to the previous methods, the alternative optimization algorithm is applied to solve the dual-optimization problem. The sub-problem about **F** has the same closed-form solution as traditional hypergraph learning [8].

The most important point that is different from the previous one is the sub-problem about **H**, which is written as

$$\begin{aligned}\arg\min_{0\preceq\mathbf{H}\preceq 1} \mathscr{Q}(\mathbf{H}) &= \Omega(\mathbf{H}) + \mu\Phi(\mathbf{H})\\ &= \operatorname{tr}\left(\left(\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\right)\mathbf{K}\right),\end{aligned} \tag{6.34}$$

where $\mathbf{K} = \mathbf{F}\mathbf{F}^\top + \mu\mathbf{X}\mathbf{X}^\top$. The projected gradient method is employed here because Eq. (6.34) is a complex function of $\mathbf{H}$ with a bound constraint. The gradient is derived as

$$\begin{aligned}\nabla\mathscr{Q}(\mathbf{H}) =& \mathbf{J}\left(\mathbf{I}\otimes\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{D}_v^{-1/2}\mathbf{H}\right)\mathbf{W}\mathbf{D}_e^{-2} \\ &+ \mathbf{D}_v^{-3/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{J}\mathbf{W} - 2\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{D}_\mathbf{v}^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1},\end{aligned} \tag{6.35}$$

where $\mathbf{J} = \mathbf{1}\mathbf{1}^\top$. The detailed derivation process can be found in [6]. The step size of learning $\mathbf{H}$ is set as α. Since $\mathbf{H}$ is required to be in the range of [0, 1], the projection P on the feasible set is conducted after each update. Therefore, $\mathbf{H}$ is updated by

$$\mathbf{H}_{k+1} = \boldsymbol{P}\left[\mathbf{H}_k - \alpha\nabla\mathscr{Q}\left(\mathbf{H}_k\right)\right], \tag{6.36}$$

where

$$\boldsymbol{P}\left[h_{ij}\right] = \begin{cases} h_{ij} & \text{if } 0 \leq h_{ij} \leq 1 \\ 0 & \text{if } h_{ij} < 0 \\ 1 & \text{if } h_{ij} > 1 \end{cases}. \tag{6.37}$$

In this way, we can alternately optimize $\mathbf{F}$ and $\mathbf{H}$ until the objective function converges.

The dynamic hypergraph structure learning method can outperform the traditional hypergraph learning consistently. This is due to the fact that the dynamic hypergraph structure can fit the data better and formulate the high-order correlation more effectively. Furthermore, both the feature and the label information are applied for the hypergraph structure optimization. Therefore, the learned hypergraph structure is smooth on the feature space and the label space. In other words, the vertices with the same labels have stronger high-order connections, which benefit the downstream task. We also note that the above dynamic hypergraph structure optimization method is with relatively high computational complexity, as it optimizes the whole incidence matrix $\mathbf{H}$.

6.4 Incremental Learning on Growing Data

Most of the existing methods consider the static structures with fixed sets of vertices and edges, while the data are generally dynamic in real-world applications. Under such circumstances, the vertices and connections can be added or removed, and the vertex attributes and connects weights change during the dynamic procedure. Generally, there are two typical ways of dynamic structure learning, i.e., using recurrent architectures [9, 10] and capturing temporal patterns [11, 12]. However, the efficient learning of temporally growing structure has not been explored yet, where the vertex and edge sets are expanding over time. Taking the citation network into consideration, new publications and citation links are continuously added into the network.

The incremental subgraph is the subgraph with the newly appeared vertices and related new edges in the given growing graph at each time step. The edges connecting the vertices from the same incremental subgraph are denoted as intra-edges, while the edges connecting the vertices from different incremental subgraphs are denoted as inter-edges. The incremental learning method aims to update the model based on the incremental subgraphs at each time step and perform on the entire graph consistently. The challenge of the incremental graph learning method is how to design the efficient strategy to update the model with incremental data and maintain the performance on the whole dataset.

The main differences between incremental graph learning and existing incremental learning methods are as follows:

- Incremental learning on growing graphs should store the observed vertices, which may be connected with the newly coming vertices, while existing incremental learning methods always drop the old samples under some scenarios.
- Considering the effect of the inter-edges on training, it is also essential to use previous data when updating models with newly appeared data.

There are two straightforward solutions of incremental graph learning. First, the static graph learning methods can directly be applied on the whole graph at each time step, which suffers from a high computation cost. Second, only learn from the incremental subgraph, which leads to bias to the newly coming subgraphs and loses the information of the inter-edges.

In this section, we introduce incremental learning for graphs on the growing data. During training, a graph $\mathcal{G}_t^L$ with a smaller number of vertices and edge sets from the growing graph $\{\mathcal{G}_t\}$ is generated for updating current model, which can be implemented by existing GNN methods for specified graph learning task and can perform on the entire observed graph at any time. Vertices and edges within restricted numbers from the old graph are selected and combined with new subgraph into $\mathcal{G}_t^L$. Therefore, $\mathcal{G}_t^L$ is unbiased to the entire graph and enough inter-edges are preserved. The overview of the IGL is shown in Fig. 6.5.

To address these issues of subgraph bias and inter-edges missing, the following conditions should be considered for generating learning.

Unbiased Estimation of Neighboring Aggregation To alleviate the bias of subgraph, the aggregation results of vertices in $\mathcal{G}_t^L$ should be unbiased estimations of them in the entire graph, i.e., $\forall \mathbf{v} \in \mathcal{V}_t$,

$$\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L\right) = agg\left(\mathbf{v}, \mathcal{N}_t(\mathbf{v})\right), \tag{6.38}$$

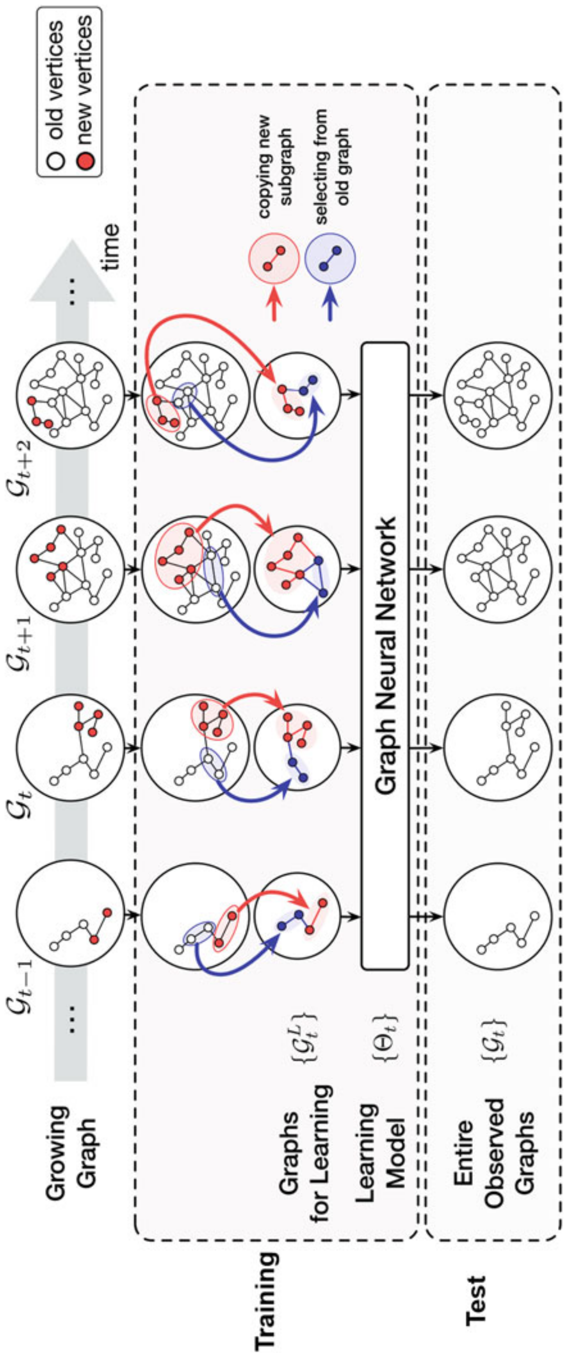


Fig. 6.5 The framework of the incremental graph learning method

where $agg(\mathbf{v}, \mathcal{N})$ is the aggregator function of GNN to aggregate vertex embeddings from $\mathcal{N}$ to $\mathbf{v}$, and $\mathcal{N}_t(\mathbf{v}) = \{\mathbf{u} \in \mathcal{V}_t \mid (\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t\}$ is the neighborhood set of vertex $\mathbf{v}$. Thus, $\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L$ represents the sampled neighboring vertices in $\mathcal{G}_t^L$.

Preservation of Inter-edges Since the missing of inter-edges may seriously affect training, we aim at preserving more edges of $\mathcal{E}_t^{inter}$ in $\Delta\mathcal{E}_t^L$, which can be formulated as

$$\begin{aligned} &\max_{\Delta\mathcal{E}_t^L} |\Delta\mathcal{E}_t^L \cap \mathcal{E}_t^{inter}|. \\ &s.t.\ \ |\Delta\mathcal{E}_t^L| \le E_{max}. \end{aligned} \tag{6.39}$$

The edge preservation can be required as a definite optimization problem in Eq. (6.39) or sampling problem with priority to vertices with higher degrees so that $P(\mathbf{u} \in \mathcal{V}_t^L) \propto |\{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter} \mid \mathbf{v} \in \mathcal{V}_t^{new}\}|$.

IGL is based on the unbiased and edge-preserved conditions. In the presentation of method, we follow the memory constraint V_{max} and set $E_{max} = (|\mathcal{V}_t^{new}| + V_{max})^2 - |\mathcal{E}_t^{intra}|$ by default. The generated edges can be uniformly sampled if a smaller E_{max} is required. The sample-based strategy is presented to select a subgraph from the previous graph for learning. The following cluster-based strategy is presented to construct a cluster graph that satisfies both the unbiased and edge-preserved conditions in midway. The presented strategies are illustrated in Fig. 6.6.

(1) Sample-Based Strategy

The strategy of sampling a representative subgraph from previous data based on the required conditions is studied first. We assume that a subset $\Delta\mathcal{V}_t^L$ from $\mathcal{V}_{t-1}$ in size of V_{max} is sampled, and all the related edges are preserved, i.e.,

$$\begin{aligned} \Delta\mathcal{V}_t^L &= Sample\ (\mathcal{G}_{t-1}, V_{max}), \\ \Delta\mathcal{E}_t^L &= \{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter} \mid \mathbf{u} \in \Delta\mathcal{V}_t^L, \mathbf{v} \in \mathcal{V}_t^{new}\}, \end{aligned} \tag{6.40}$$

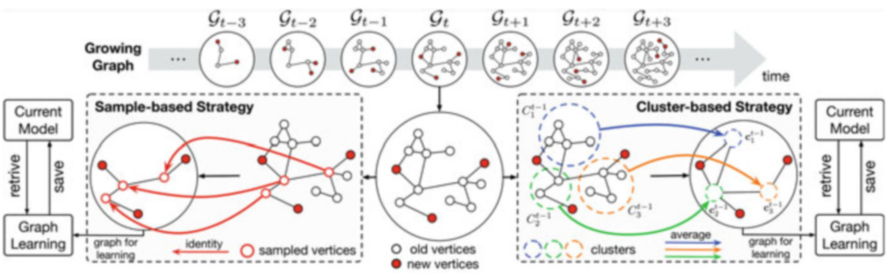


Fig. 6.6 An illustration of the sample-based and cluster-based strategies

where *Sample*() denotes the sampling function. Considering the required conditions, we explore the following pragmatic methods for appropriate sampling:

- *Random selection* is for the unbiased condition that uniformly selects V_{max} vertices from $\mathscr{V}_{t-1}$. However, it cannot preserve enough edges for efficient training, especially in sparse graphs.
- *Random jump* is a traversal-based sampling, and we adapt it in the following steps. Starting out with any vertex in $\mathscr{V}_t^{new}$, we either randomly walk to a neighboring vertex in $\mathscr{V}_{t-1}$ with probability p and select it, or randomly jump to a vertex in $\mathscr{V}_t^{new}$ with probability $(1-p)$. We repeat to fill the sampled set. It has been proved that the probability of sampling a vertex tends to be proportional to its degree, which works under the edge-preserved condition.
- *Degree-based selection* is for the edge-preserved condition that samples vertices with priority to those connected with more inter-edges. Let $D_t(\mathbf{u}) = |\{(\mathbf{u}, \mathbf{v}) \in \mathscr{E}_t\}|$ be the degree of $\mathbf{u}$, and we define $D_t^{new}(\mathbf{u}) = \frac{D_t(\mathbf{u}) - D_{t-1}(\mathbf{u})}{D_t(\mathbf{u})}$, $\forall \mathbf{u} \in \mathscr{V}_{t-1}$ as the new degree of vertices to measure their closeness to the new subgraph through inter-edges. We then select top-V_{max} vertices in $\mathscr{V}_{t-1}$ by their new degrees.

The above methods take into consideration only part of required conditions. It can be proved that, ignoring the ideal case when all the vertices in $\mathscr{V}_{t-1}$ connect with the same number of vertices in $\mathscr{V}_t^{new}$, sampling in Eq. (6.40) satisfies the two required conditions when all the vertices have been sampled, i.e., joint training.

(2) Cluster-Based Strategy

The sample-based strategy selects a subgraph from the previous graph for learning. However, in such a process, $\mathscr{G}_{t-1}$ is not completely covered, and some important vertices might be dropped. Then, the selected subgraph cannot perform full communication with the new subgraph. The assumption of sampling that $\mathscr{G}_t^L$ must be a subgraph of $\mathscr{G}_t$ is relaxed, and a cluster graph is constructed. Technically, we first arrange vertices in $\mathscr{V}_{t-1}$ into K cluster sets $\{\mathscr{C}_i^{t-1}\}_{i=1}^K$ with centers $\{\mathbf{c}_i^{t-1}\}_{i=1}^K$ in average values of clusters. We set the number of clusters $K = V_{max}$. The cluster graph is therefore defined as

$$\begin{aligned}
\Delta\mathscr{V}_t^L &= \{\mathbf{c}_1^{t-1}, ..., \mathbf{c}_K^{t-1}\},\\
\Delta\mathscr{E}_t^L &= \{(\mathbf{c}_i^{t-1}, \mathbf{v}) \mid \mathbf{v} \in \mathscr{V}_t^{new}, \exists\, \mathbf{u} \in \mathscr{C}_i^{t-F1}, (\mathbf{u}, \mathbf{v}) \in \mathscr{E}_t^{inter}\} \cup \\
&\quad \{(\mathbf{c}_i^{t-1}, \mathbf{c}_j^{t-1}) \mid \exists\, \mathbf{u}_1 \in \mathscr{C}_i^{t-1}, \mathbf{u}_2 \in \mathscr{C}_j^{t-1}, (\mathbf{u}_1, \mathbf{u}_2) \in \mathscr{E}_{t-1}\},
\end{aligned} \tag{6.41}$$

which suggests that the cluster centers be added as new cluster vertices, and the edges connecting to any vertex in $\mathscr{V}_{t-1}$ be directly transferred to the corresponding cluster vertex. It is noted that the additional edge sets in Eq. (6.41) represent $\mathscr{E}_t^{inter}$ and $\mathscr{E}_{t-1}$, respectively.

Due to the continued growth of the graph, direct clustering on the entire graph is time-consuming. For an approximate but efficient clustering with a balanced size, we first conduct clustering on the new vertices $\mathscr{V}_t^{new}$ into cluster sets $\{\Delta\mathscr{C}_i^t\}_{i=1}^K$ with centers $\{\hat{\mathbf{c}}_i^t\}_{i=1}^K$. The bipartite matching algorithm is applied to optimize a bijective matching function $M(\cdot) : \{1, ..., K\} \rightarrow \{1, ..., K\}$ for the objective: $\min_{m(\cdot)} \Sigma_{k=1}^{K} \|\mathbf{c}_k^{t-1} - \hat{\mathbf{c}}_{m(k)}^{t}\|_2^2$, which assigns new clusters to be closer with old clusters. Then, we merge the clusters as $\mathscr{C}_k^t = \mathscr{C}_k^{t-1} \cup \Delta\mathscr{C}_{m(k)}^t$ and update the value of centers $\mathbf{c}_k^t$.

In a word, incremental graph learning (IGL) is a general framework for efficient learning on growing graphs in an incremental manner, which has the following advantages. First, IGL is more suitable in real-world applications, since the dynamic graphs are commonly appeared. Second, the sample-based and cluster-based strategies significantly improve the efficiency when the large scale graph grows. However, only the incremental of the nodes and edges are considered, while the deletion are ignored, which limits the application of the method. The general dynamic patterns are worth studying in the future works.

6.5 Summary

In this chapter, we introduce hypergraph structure evolution methods, i.e., hyperedge weight optimization, vertex weight optimization, sub-hypergraph weight optimization, dynamic hypergraph learning, and the techniques for incremental learning on growing graphs. The hyperedge weight optimization adjusts weights of each hyperedge for different contributions, while the vertex weight optimization considers the different importance of vertices on hypergraph. The sub-hypergraph weight optimization method further combines multiple hypergraphs for multi-modal data with learned weights. Dynamic hypergraph learning optimizes the hypergraph structure by modifying the inappropriate connections, which can partially solve the missing and incorrect connection issue. Finally, we introduce the incremental learning method on growing graphs, which can update the data structure under the incremental scenario.

It is noted that the optimization of hypergraph, either component or the structure, will bring in extra computational cost and lead to potentially high computation complexity in practice. How to effectively and efficiently adjust the hypergraph structure is still a challenging problem, which requires further investigation in future.

References

1. Y. Huang, Q. Liu, D. Metaxas, Video object segmentation by hypergraph cut, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1738–1745
2. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
3. Q. Liu, Y. Sun, C. Wang, T. Liu, D. Tao. Elastic net hypergraph learning for image clustering and semi-supervised classification. IEEE Trans. Image Process. **26**(1), 452–463 (2017)

4. Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, X. Wu, Visual textual joint relevance learning for tag-based social image search. IEEE Trans. Image Process. **22**(1), 363–376 (2013)
5. L. Su, Y. Gao, X. Zhao, H. Wan, M. Gu, J. Sun, Vertex weighted hypergraph learning for multi-View object classification, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2017), pp. 2779–2785
6. Z. Zhang, H. Lin, Y. Gao, Dynamic hypergraph structure learning, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (2018), pp. 3162–3169
7. Z. Zhang, H. Lin, X. Zhao, R. Ji, Y. Gao, Inductive multi-hypergraph learning and its application on view-based 3D object classification. IEEE Trans. Image Process. **27**(12), 5957–5968 (2018)
8. D. Zhou, J. Huang, B. Scholkopf. Learning with hypergraphs: clustering, classification, and embedding, in *Proceedings of the Advances in Neural Information Processing Systems* (2007), pp. 1601–1608
9. F. Manessi, A. Rozza, M. Manzo, Dynamic graph convolutional networks. Pattern Recogn. **97**, 107000 (2020)
10. A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T.B. Schardl, C.E. Leiserson, EvolveGCN: Evolving graph convolutional networks for dynamic graphs, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), pp. 5363–5370
11. S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), pp. 7444–7452
12. D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, in *Proceedings of the International Conference on Learning Representations* (2020)

Chapter 7
Neural Networks on Hypergraph

Abstract With the development of deep learning on high-order correlations, hypergraph neural networks have received much attention in recent years. Generally, the neural networks on hypergraph can be divided into two categories, including the spectral-based methods and the spatial-based methods. For the spectral-based methods, the convolution operation is formulated in the spectral domain of graph, and we introduce the typical spectral-based methods, including hypergraph neural networks (HGNN), hypergraph convolution with attention (Hyper-Atten), and hyperbolic hypergraph neural network (HHGNN), which extend hypergraph computation to hyperbolic spaces beyond the Euclidean space. For the spatial-based methods, the convolution operation is defined in groups of spatially close vertices. We then present spatial-based hypergraph neural networks of the general hypergraph neural networks (HGNN+) and the dynamic hypergraph neural networks (DHGNN). Additionally, there are several convolution methods that attempt to reduce the hypergraph structure to the graph structure, so that the existing graph convolution methods can be directly deployed. Lastly, we analyze the association and comparison between hypergraph and graph in the two areas described above (spectral-based, spatial-based), further demonstrating the ability and advantages of hypergraph on constructing and computing higher-order correlations in the data.

7.1 Introduction

Hypergraph has demonstrated its ability to model and learn complex correlations in recent years. Zhou et al. [1] introduced the hypergraph learning, which conducts transductive learning and propagates information on the hypergraph structure. Transductive inference on the hypergraph aims to minimize the label difference between vertices with stronger connections. There has been extensive development and application of hypergraph learning in several fields over the past few years.

In addition, hypergraph has been investigated in deep learning applications. Based on the hypergraph Laplacian and the Chebyshev formula, Feng et al. [2] first introduced hypergraph neural networks (HGNN). The hypergraph Laplacian is synthesized using predictions in Yadati et al. [3], while Bai et al. [4] defined two

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_7

neural hypergraph operators based on [5, 6]. However, they do not implement high-order learning algorithms by introducing only vertex functions, even though they construct a simple weighted graph and apply mature graph learning algorithms. A lack of powerful tools for expressing hyperstructure and a wealth of graph literature motivated the work of [7]. Additionally, recent successes with graph representation learning have been achieved by using neural operators (convolution, attention, spectral, etc.). Generally, the neural networks on hypergraph can be divided into two categories, including the spectral-based methods and the spatial-based methods.

For the spectral-based methods, Feng et al. [2] introduced the hypergraph neural networks (HGNN) for modeling and learning beyond pairwise complex correlations. Different from the traditional graph neural networks (GNN), HGNN learns its data representation by iteratively propagating the vertex–hyperedge–vertex information pattern. Additionally, the hypergraph Laplacian is first approximated and introduced into the deep hypergraph learning method to speed up the learning process. Following [2], Bai et al. [4] developed an attention module based on hypergraph convolution patterns (Hyper-Atten). Hyper-Atten introduced a hyperedge–vertex attention learning module that adaptively identifies the importance of different vertices in a hyperedge, thus revealing the intrinsic correlations between vertices.

Using the spatial methods, Atwood et al. [8] made use of transition matrices to determine where vertices are located. The generalization of convolution in the spatial domain is achieved using Gaussian mixture models based on local path operators. A graph-based attention-based architecture was built in work [9] for analyzing vertices on graph using attention mechanism. A dynamic change in hypergraph structure was taken into considerations in [6]. The framework introduced in [6], which is more versatile than HGNN [2]. A unified hypergraph is then constructed by merging the correlations from different modalities/types using an adaptive hyperedge grouping strategy. To learn a general data representation for various tasks, a hypergraph convolution scheme [6] was performed in the spatial domain.

Hypergraph spectral graph theory [10] has been explored far less in other methods. The concept of hypergraph learning was first introduced by Zhou et al. [1], where it was presented as a propagation process. The Laplacian matrix, however, is equivalent to pairwise operations according to [11]. There have been several studies addressing non-pairwise relationships since then, including developing non-linear Laplacian operators [12, 13], learning the optimal parameters of hyperedges [13, 14], as well as utilizing random walking techniques [10]. Hyperedges can be regarded as connectors in these algorithms, which explicitly break the bipartite property of hypergraph by focusing on vertices.

In this chapter, we systematically introduce the above three types of neural networks on hypergraph and show the comparison between graph neural networks and hypergraph neural networks from both spectral and spatial aspects. Part of the work introduced in this chapter has been published in [2, 15, 16].

7.2 Spectral-Based Neural Networks on Hypergraph

The spectral neural networks methods have attracted much attention since Bruna et al. [17] and Kipf et al. [18] simplified them in a graph convolutional network pattern. The data are transformed from the common domain to the spectral domain to be processed with according to map theory and the convolution theorem, and it then gets transformed back to the common domain. In other words, first we convert the signal from the common domain to the frequency domain (Fourier transform implementation) and then multiply it by the phase. Then, we convert the result of the phase multiplication back to the common domain again (Fourier inverse transform implementation). We will present spectral-based hypergraph neural networks methods, including hypergraph neural networks (HGNN) [2], hypergraph convolution with attention (Hyper-Atten) [3], and hyperbolic hypergraph neural networks (HHGNN) [19]. In particular, HHGNN extends hypergraph learning to the hyperbolic spaces beyond the Euclidean space.

7.2.1 Hypergraph Neural Networks

Given a hypergraph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \varDelta)$ with N vertices, the hypergraph Laplacian $\varDelta$ is an $N \times N$ positive semi-definite matrix. The orthonormal eigen vectors $\varPhi = \text{diag}(\phi_1, \ldots, \phi_N)$ and a diagonal matrix $\varLambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$, which contains the corresponding non-negative eigenvalues, are obtained by employing the eigendecomposition $\varDelta = \varPhi \varLambda \varPhi^\top$. $\hat{x} = \varPhi^\top x$ defines the Fourier transform for a signal $x = (x_1, \ldots, x_N)$ in the hypergraph. It is assumed that the eigenvectors represent the Fourier bases and the eigenvalues represent the frequencies. The spectral convolution of signal x and filter g can be denoted as

$$g \star x = \varPhi((\varPhi^\top g) \odot (\varPhi^\top x)) = \varPhi g(\varLambda) \varPhi^\top x, \tag{7.1}$$

where $\odot$ denotes the element-wise Hadamard product and $g(\varLambda) = \text{diag}(g(\lambda_1), \ldots, g(\lambda_n))$ indicates a function of the Fourier coefficients. However, in the forward and inverse Fourier transforms, the computational cost is $O(n^2)$, which is high. To solve this problem, Defferrard et al. [20] parameterize $g(\varLambda)$ with K-order polynomials, and one such polynomial is the truncated the Chebyshev expansion. Chebyshev polynomials $T_k(x)$ are computed by the formula of $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. After that, the $g(\varLambda)$ can be computed by

$$g \star x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\varDelta}) x, \tag{7.2}$$

where $T_k(\tilde{\Delta})$ denotes the Chebyshev polynomial of order k with scaled Laplacian $\tilde{\Delta} = \frac{2}{\lambda_{max}}\Delta - I$. In Eq. (7.2), matrix powers, additions, and multiplications are combined instead of expansive computation of Laplacian Eigen vectors, thus improving computation complexity even further. Since that the Laplacian in hypergraph can already represent the high-order correlations among nodes, it can further limit the order of convolution operation to $K = 1$. It is suggested by Kipf et al. [18] that $\lambda_{max} \approx 2$ for the scale adaptability of neural networks. After that, the convolution operation can be simplified to

$$g \star x \approx \theta_0 x - \theta_1 \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}x, \tag{7.3}$$

where θ_0 and θ_1 represent the parameters of all node filters. In addition, a single parameter θ is used to avoid the overfitting problem, which is defined as

$$\begin{cases} \theta_1 = -\frac{1}{2}\theta \\ \theta_0 = \frac{1}{2}\theta\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}. \end{cases} \tag{7.4}$$

Thereafter, the convolution process can be simplified to the following function:

$$\begin{aligned} g \star x &\approx \frac{1}{2}\theta\mathbf{D}_v^{-1/2}\mathbf{H}(\mathbf{W}+\mathbf{I})\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}x \\ &\approx \theta\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}x, \end{aligned} \tag{7.5}$$

where $(\mathbf{W}+\mathbf{I})$ can be regarded as the weight of the hyperedges. In the initialization of $\mathbf{W}$, the hyperedges can be all assigned with equal weights as an identity matrix.

When having a hypergraph signal $\mathbf{X}^t$ for the t-th layer, the hyperedge convolution layer **HGNNConv** can be formulated by

$$\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^t\Theta), \tag{7.6}$$

where Θ is the parameter to be learned during the training process. To extract features from a hypergraph, the filter Θ is applied to the vertices. After convolution, $\mathbf{X}^{t+1}$, which can be used for further processing.

The framework of the abovementioned HGNN model is shown in Fig. 7.1. HGNN is able to address the challenges of learning representations for complex data by incorporating such data structures into hypergraph, which are more flexible and effectively confronting practical data.

The HGNN calculation stages are shown in Fig. 7.2, and the three processes are directly projected to the functions. We can observe that there are vertex feature transform, hyperedge feature gathering, and vertex feature aggregating steps in this framework.

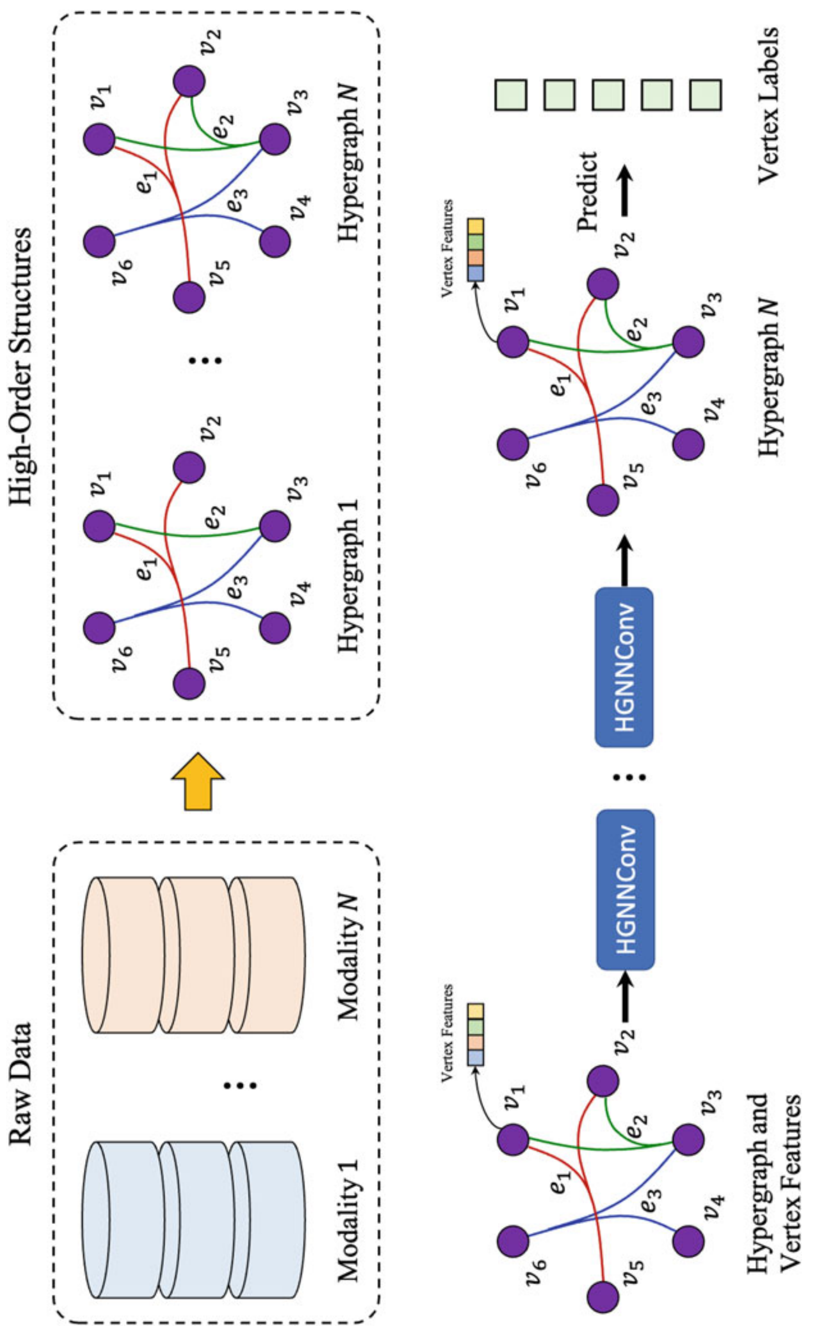


Fig. 7.1 The framework of the HGNN model

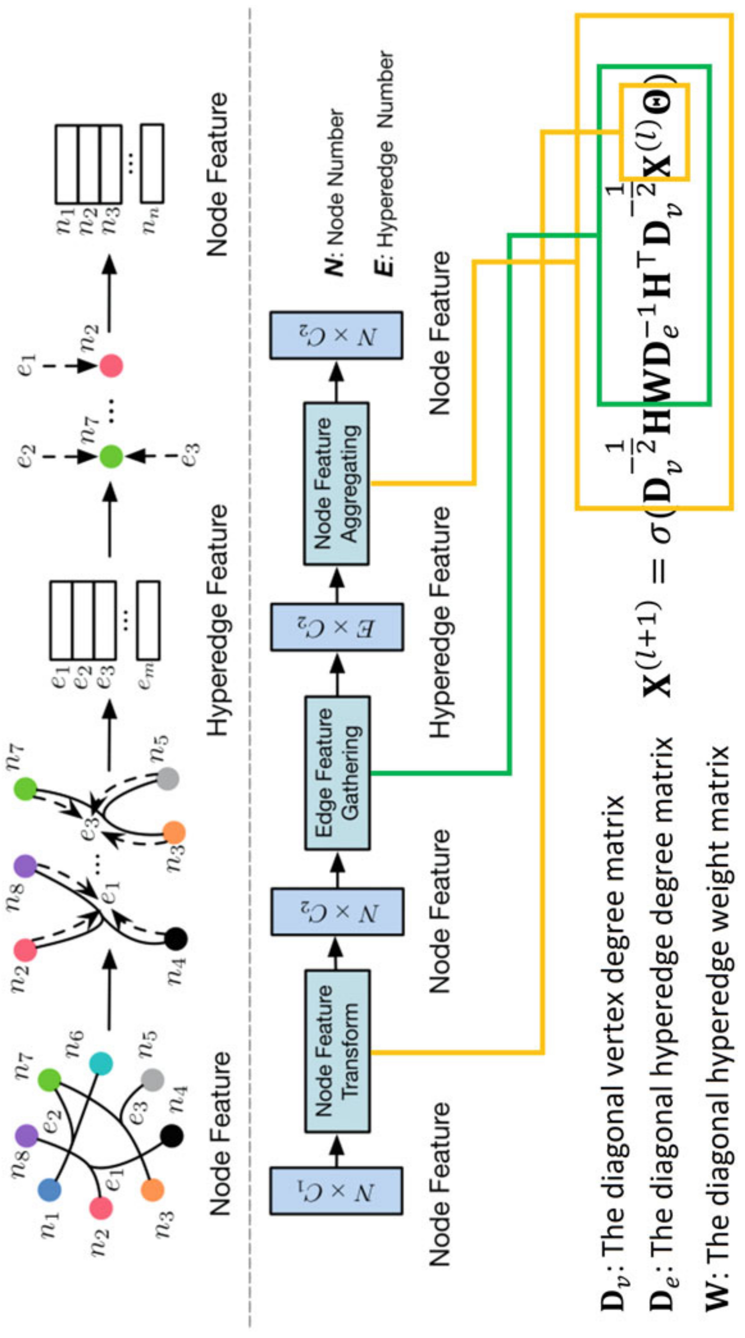

Node Feature
Hyperedge Feature
Node Feature
$N \times C_1$
Node Feature Transform
$N \times C_2$
Edge Feature Gathering
$E \times C_2$
Node Feature Aggregating
$N \times C_2$
Node Feature
Node Feature
Hyperedge Feature
Node Feature
N: Node Number
E: Hyperedge Number
$\mathbf{X}^{(l+1)} = \sigma(\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^{\top}\mathbf{D}_v^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{\Theta})$
$\mathbf{D}_v$: The diagonal vertex degree matrix
$\mathbf{D}_e$: The diagonal hyperedge degree matrix
$\mathbf{W}$: The diagonal hyperedge weight matrix

Fig. 7.2 The calculation process of the HGNN framework

7.2.2 Hypergraph Convolution and Hypergraph Attention

Based on the study of hypergraph neural networks [2], Bai et al. [4] introduced hypergraph convolution and hypergraph attention (Hyper-Atten) by introducing attention mechanism in the framework.

In this method, an explicit magnitude of importance is assigned to the afferent and efferent information flow for non-binary values of the transition probability between vertices for a given vertex. However, such an attention mechanism must work after the graph structure (the incidence matrix **H**) is given, instead of learning a dynamic incidence matrix. It is easier to reveal the intrinsic relationship between vertices using a dynamic transition matrix than by using a fixed incidence matrix. An attention learning module could be imposed on **H**, which does not treat each vertex as being connected by a hyperedge or which does not assign non-binary and real values when measuring the degree of connectivity. Following [6] when the vertex set and the edge set are comparable, the attention score between a given vertex x_i and its associated hyperedge x_j can be written as

$$\mathbf{H}_{ij} = \frac{\exp\left(\sigma\left(\operatorname{sim}\left(x_i\mathbf{P}, x_j\mathbf{P}\right)\right)\right)}{\sum_{k\in N_i}\exp\left(\sigma\left(\operatorname{sim}\left(x_i\mathbf{P}, x_k\mathbf{P}\right)\right)\right)}, \tag{7.7}$$

where $\sigma(\cdot)$ is a nonlinear activation function. The weight matrix between the (l)-th and $(l+1)$-th layers is denoted as $\mathbf{P} \in \mathbb{R}^{F^{(l)}\times F^{(l+1)}}$. N_i is the neighborhood set of x_i. The pairwise similarity of two vertices is computed with this similarity function $\operatorname{sim}(\cdot)$:

$$\operatorname{sim}\left(x_i, x_j\right) = \mathbf{a}^\top\left[x_i \| x_j\right]. \tag{7.8}$$

Operation $[,\|,]$ indicates concatenation, and notation **a** is a weight vector for outputting a scalar similarity value.

When following Eq. (7.6) to learn the intermediate embedding of vertices layer by layer, hypergraph attention also propagates gradients to **H** in addition to $\mathbf{X}^{(l)}$ and Θ. Therefore, Eq. (7.7) means the share of hyperedge x_j in the neighbors of the vertex x_i, which indicates the relative importance x_j of x_i. More categorical embeddings can be learned by the probabilistic model, and the relationship between vertices can be described more accurately.

In order to further enhance the capability of representation learning, the method uses hypergraph attention mechanisms based on the basic formulation of performing convolutions.

7.2.3 Hyperbolic Hypergraph Neural Networks

The hyperbolic space is a manifold with constant Gaussian negative curvature everywhere, which has several models. Similar to [21, 22], the work is based on the Poincaré ball model for its well-suited for gradient-based optimization. The Poincaré ball model with constant negative curvature $-1/k (k > 0)$ corresponds to the Riemannian manifold $\left(\mathbb{P}^{n,k}, g_{\mathbf{x}}^{\mathbb{P}}\right)$. $\mathbb{P}^{n,k} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$ is an open n-dimensional unit ball, where $\|.\|$ denotes the Euclidean norm. Its metric tensor is $g_{\mathbf{x}}^{\mathbb{P}} = \lambda_{\mathbf{x}}^2 g^{\mathbb{E}}$, where $\lambda_{\mathbf{x}} = \frac{2}{1-k\|\mathbf{x}\|^2}$ is the conformal factor and $g^{\mathbb{E}} = \mathbf{I}_n$ is the Euclidean metric tensor. Then, we define the Möbius addition of two points $\mathbf{x}, \mathbf{y} \in \mathbb{P}^{n,k}$ as follows:

$$\mathbf{x} \oplus_k \mathbf{y} = \frac{\left(1 + 2k\langle \mathbf{x}, \mathbf{y} \rangle + k\|\mathbf{y}\|^2\right)\mathbf{x} + \left(1 - k\|\mathbf{x}\|^2\right)\mathbf{y}}{1 + 2k\langle \mathbf{x}, \mathbf{y} \rangle + k^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \tag{7.9}$$

The distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{P}^{n,k}$ is calculated by integration of the metric tensor, which is given as

$$d_{\mathbb{P}}^k(\mathbf{x}, \mathbf{y}) = (2/\sqrt{k}) \tanh^{-1}\left(\sqrt{k}\,\|-\mathbf{x} \oplus_k \mathbf{y}\|\right). \tag{7.10}$$

Here we can denote point $\mathbf{z} \in \mathscr{T}_{\mathrm{x}}\mathbb{P}^{n,k}$ as the tangent (Euclidean) space centered at any point $\mathbf{x}$ in the hyperbolic space. For the tangent vector $\mathbf{z} \neq \mathbf{0}$ and the point $\mathbf{y} \neq \mathbf{0}$, the exponential map $\exp_{\mathbf{x}} : \mathscr{T}_{\mathbf{x}}\mathbb{P}^{n,k} \to \mathbb{P}^{n,k}$ and the logarithmic map $\log_{\mathbf{x}} : \mathbb{P}^{n,k} \to \mathscr{T}_{\mathbf{x}}\mathbb{P}^{n,k}$ are given for $\mathbf{y} \neq \mathbf{x}$ by

$$\exp_{\mathbf{x}}^k(\mathbf{z}) = \mathbf{x} \oplus_k \left(\tanh\left(\sqrt{k}\frac{\lambda_{\mathbf{x}}^k\|\mathbf{z}\|}{2}\right)\frac{\mathbf{z}}{\sqrt{k}\|\mathbf{z}\|}\right) \tag{7.11}$$

and

$$\log_{\mathbf{x}}^k(\mathbf{y}) = \frac{2}{\sqrt{k}\lambda_{\mathbf{x}}^k}\tanh^{-1}\left(\sqrt{k}\,\|-\mathbf{x} \oplus_k \mathbf{y}\|\right)\frac{-\mathbf{x} \oplus_k \mathbf{y}}{\|-\mathbf{x} \oplus_k \mathbf{y}\|}. \tag{7.12}$$

The transformation between the tangent space and the hyperbolic space is shown in Fig. 7.3. Leverage the operations of exp and log maps, so that we can use the tangent space $\mathscr{T}_{\mathbf{x}}\mathbb{P}$ to perform transformations such as convolution and activation in Euclidean space. In the convolution, vertex information is first gathered to the hyperedge for storage, and then each vertex aggregates the information of the connected hyperedge.

It is noted that initial data are on the Euclidean space and need to be converted into embeddings on the hyperbolic space, so then first project the data on the previously obtained Euclidean space onto the hyperbolic manifold space in order to use the spectral-based hypergraph hyperbolic convolutional network to learn the information to update the node embeddings. Set $t := \{\sqrt{k}, 0, 0, \ldots, 0\} \in \mathbb{P}^{d,k}$

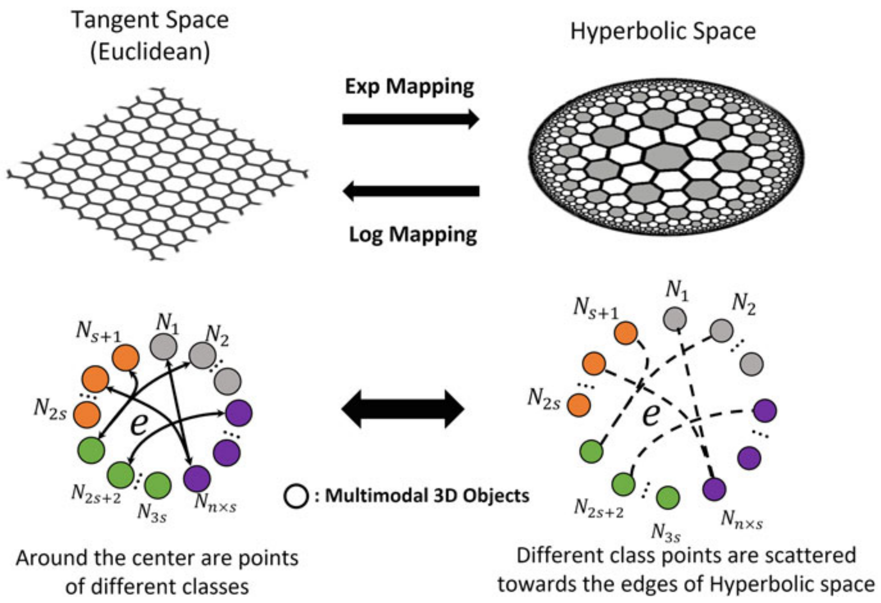

Tangent Space (Euclidean)
Hyperbolic Space
Exp Mapping
Log Mapping
N_{s+1}
N_1
N_2
N_{2s}
e
N_{2s+2}
N_{3s}
$N_{n\times s}$
○ : Multimodal 3D Objects
Around the center are points of different classes
Different class points are scattered towards the edges of Hyperbolic space

Fig. 7.3 The transformation between the tangent space and the hyperbolic space

as a reference point to perform tangent space operations. The above condition makes $\langle(0, \mathbf{x}^{0,\mathbb{E}}), t\rangle = 0$ hold, so $(0, \mathbf{x}^{0,\mathbb{E}})$ can be regarded as the initial embedding representation of the hypergraph structure on the tangent plane of the hyperbolic manifold space $\mathscr{T}_t\mathbb{P}^{d,k}$. The initial hypergraph structure embedding is then mapped onto the hyperbolic manifold space $\mathbb{P}$ following [19]:

$$\begin{aligned}\mathbf{x}^{0,\mathbb{P}} &= \exp_t^k\left((0, \mathbf{x}^{0,\mathbb{E}})\right)\\ &= \left(\sqrt{k}\cosh\left(\frac{\|\mathbf{x}^{0,\mathbb{E}}\|_2}{\sqrt{k}}\right), \sqrt{k}\sinh\left(\frac{\|\mathbf{x}^{0,\mathbb{E}}\|_2}{\sqrt{k}}\right)\frac{\mathbf{x}^{0,\mathbb{E}}}{\|\mathbf{x}^{0,\mathbb{E}}\|_2}\right).\end{aligned} \tag{7.13}$$

Unlike the previous study [23] that simply generates the hyperedge structure for common domain convolution, combined with the inspiration provided by HGNN [2], hypergraph computation from the perspective of spectral convolution can be conducted.

Given hyperbolic curvatures $-1/k_{\ell-1}$, $-1/k_\ell$ at layers $\ell-1$ and ℓ, respectively, then the hyperbolic hypergraph convolution of the hypergraph input signal $x^{\mathbb{P}}$ with filter g can be defined as

$$\begin{aligned}\mathbf{x}^{\mathbb{P}} * \mathrm{g} &= \exp_x^{k_\ell}\left(\Phi\left(\left(\Phi^\top\left(\log_x^{k_{\ell-1}}\left(x^{\mathbb{P}}\right)\right)\right)\odot\left(\Phi^\top \mathrm{g}\right)\right)\right)\\ &= \exp_x^{k_\ell}\left(\Phi g(\Lambda)\Phi^\top\left(\log_x^{k_{\ell-1}}\left(x^{\mathbb{P}}\right)\right)\right),\end{aligned} \tag{7.14}$$

where $\odot$ is the element-wise product, $g(\Lambda) = \mathrm{diag}(\theta)$, and $\theta = [\theta_1, \cdots, \theta_n]$ is the parameters to be learned. Leverage the operations of exp and log maps, so that the tangent space $\mathscr{T}_0\mathbb{P}^{d,k}$ can be used to perform Euclidean transformations. It operates in the tangent space of each center point $x^{\mathbb{P}}$ because the Euclidean approximation is best [19].

Considering the high computational complexity of the Fourier transform and inverse Fourier transform, this convolution method is very expensive to calculate. Convolutions can be computed more efficiently by truncating Chebyshev polynomials as [2]. It can be simply expressed as

$$\mathbf{x}^{\mathbb{P}} * \mathrm{g} \approx \exp_x^{k_\ell}\left(\theta \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\left(\log_x^{k_{\ell-1}}\left(x^{\mathbb{P}}\right)\right)\right), \tag{7.15}$$

where $\mathbf{W}$ is the initial weight of hyperedges. The above equation uses the hypergraph Laplacian matrix to calculate the total gain obtained after a small perturbation of a point. For a hypergraph with n vertices, the convolution layer can be denoted as following formulation:

$$\mathbf{X}^\ell = \exp_{\mathbf{x}^{\ell,\mathbb{E}}}^{k_\ell}\left(\sigma\left(\mathbf{A}\left(\log_{\mathbf{x}^{\ell-1,\mathbb{P}}}^{k_{\ell-1}}\left(\mathbf{X}^{\ell-1,\mathbb{P}}\right)\right)\Theta\right)\right), \tag{7.16}$$

where $\Theta \in \mathbb{R}^{c(\ell-1)\times c(\ell)}$ is the parameter to be learned during the training process, which is applied over the vertices in the hypergraph to extract features. c indicates the size of the embedding dimension, σ denotes the nonlinear activation function, and $\mathbf{A} = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$.

The hyperbolic operation is accomplished by conducting feature mapping between the Euclidean space and the hyperbolic space. The framework of the above spectral-based hyperbolic hypergraph convolution is shown in Fig. 7.4.

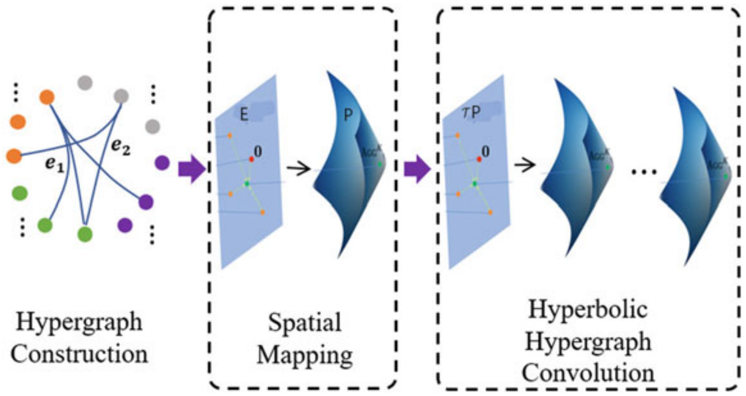


Fig. 7.4 The framework of the spectral-based hyperbolic hypergraph convolution method

7.3 Spatial-Based Neural Networks on Hypergraph

To show the spatial-based neural networks on hypergraph, we first briefly review the definition of spatial-based graph convolution. The processing on an image is taken as an example. The pixel in an image can be represented as vertices in a grid graph, where each vertex only connects its neighbor vertices in the spatial–closed region where it is located. A C-channel feature can be accordingly generated for each vertex (pixel) in the image. The process of filtering an image can be viewed as an average aggregation of neighbors' features after a central vertex transforms their features. Similar to convolution neural networks in image processing, spatial-based graph convolution combines the neighbors of the central vertex to produce a new representation. Spatial-based graph convolution runs from neighbor vertices to center vertices, which is similar to the definition of a path in a simple graph. A path in graph is defined as $P(v_1, v_k) = (v_1, v_2, \ldots, v_k)$. Vertices in the sequence are adjacent to each other, so that every vertex in the sequence is adjacent to every other vertex. It means that all the vertex pairs of i and $i + 1$ $(1 \leq i \leq k - 1)$ have the neighbor relation.

Similar to the spatial-based graph convolution, spatial-based hypergraph neural networks also consider the neighbor information when learning representation. Following, we introduce two typical spatial-based hypergraph neural networks, including general hypergraph neural networks (HGNN$^+$) [16] and dynamic hypergraph neural networks (DHGNN) [15].

7.3.1 General Hypergraph Neural Networks

In this part, the general framework [16] for modeling representation learning using hypergraph neural networks on given raw data is introduced. Figure 7.5 demonstrates the framework of general hypergraph neural networks, which also consists of two procedures, i.e., hypergraph modeling and hypergraph convolution. In the hypergraph modeling step, data issued to generate the high-order correlations, which are represented as a hypergraph. Similar to previous tasks, hyperedge groups can be generated as pairwise edges, k-Hop, and neighbors in the feature space. As a result of this procedure, all types of hyperedge groups (if they are available) are generated and concatenated in a hypergraph for the purpose of data correlations modeling. Hypergraph convolution is the process of creating a set of hypergraph convolutions on a given set of hypergraph, i.e., the spectral-based convolution and the spatial-based hypergraph convolution for representation learning on hypergraph. As a result of these convolution procedures, they can generate much more accurate representations of multi-modal data and high-order correlations using this information.

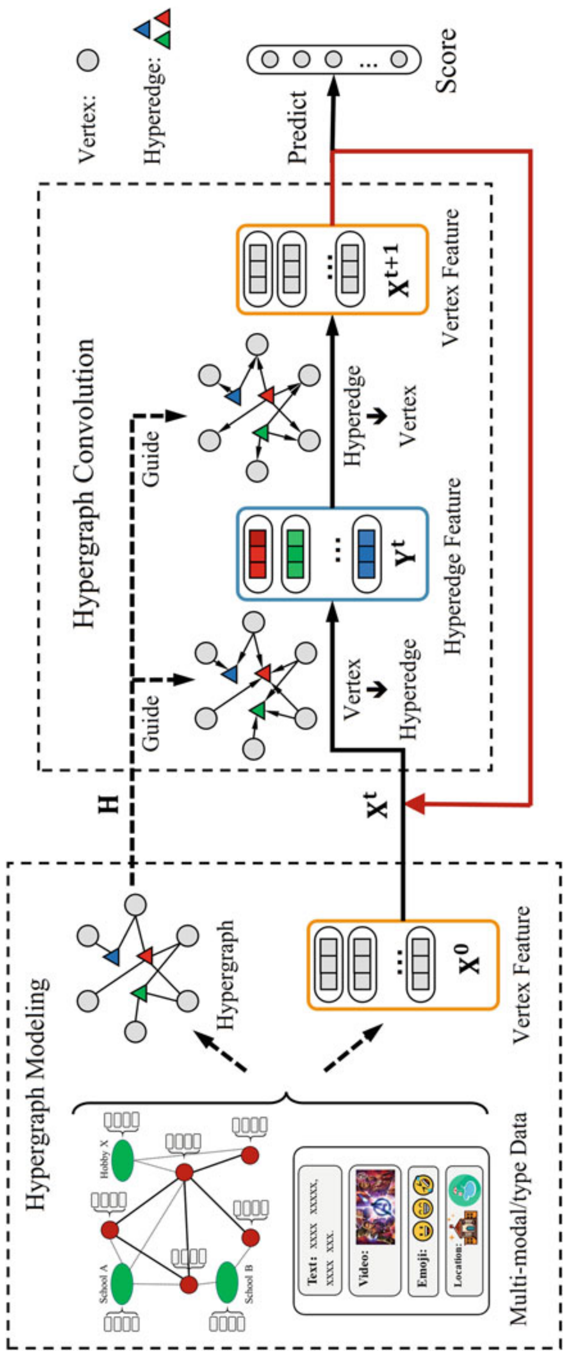


Fig. 7.5 An illustration of the general hypergraph neural network framework (HGNN$^+$). This figure is from [16]

(1) Hypergraph Modeling

The first step is to construct a flexible hypergraph from raw data if there is no hypergraph existed, and the data correlations can be modeled using a hypergraph structure. The ability to generate a suitable hypergraph structure is critical to exploit the high-order correlations among the data. Generally, hypergraph structures are not explicit in most cases. Therefore, different strategies are needed to generate the hypergraph. Hypergraph generation from scratch usually involves a combination of three scenarios, namely, data with graph structure, data without graph structure, and data with multi-modal/multi-type representations. Hyperedge generation strategies, which employ pairwise edges, k-Hop, and neighbors in the feature space, respectively, are introduced here. The strategies of using pairwise edges and k-Hop are utilized for hyperedge group generation from the data with a graph structure, and those of using neighbors in feature space are employed for hyperedge group generation from the data without graph structure. Finally, all the hyperedge groups are further concatenated to generate the overall hypergraph.

The above strategies can be used here to generate a number of hyperedge groups. A final hypergraph is then generated by further combining generated or natural hyperedge groups. Supposing there are K hyperedge groups $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K\}$, K indicates incidence matrices $\mathbf{H}_k \in \{0, 1\}^{N \times M_k}$, respectively. For the hypergraph $\mathcal{G}$, the simplest fusion way to construct the incidence matrix is directly concatenating all the hyperedge groups as $\mathbf{H} = \mathbf{H}_1 || \mathbf{H}_2 || \cdots || \mathbf{H}_K$, where $\cdot || \cdot$ is the matrix concatenation operation. These hyperedges weight matrices of hypergraph can be assigned a value of 1 in order to treat them equally. This simplest fusion way can be called as coequal fusion.

It is noted that other combination strategies can be also used according to different application scenarios. As the multi-modal hybrid high-order correlations cannot be fully exploited by a simple coequal fusion, due to differences in information richness between hyperedge groups, an adaptive strategy for the fusion of hyperedge groups, namely Adaptive Fusion, was introduced in [16]. Specifically, each hyperedge group is associated with a trainable parameter that can be used to adjust the effect of multiple hyperedge groups on the final vertex embedding in an adaptive manner, which can be defined as

$$\begin{cases} \mathbf{w}_k &= \text{copy}(\text{sigmoid}(w_k), M_k) \\ \mathbf{W} &= \text{diag}(\mathbf{w}_1^1, \ldots, \mathbf{w}_1^{M_1}, \ldots, \mathbf{w}_K^1, \ldots, \mathbf{w}_K^{M_K}) \\ \mathbf{H} &= \mathbf{H}_1 || \mathbf{H}_2 || \cdots || \mathbf{H}_K \end{cases}, \tag{7.17}$$

where $\mathbf{w}_k \in \mathbb{R}$ is a trainable parameter that is shared by all hyperedges inside a specified hyperedge group k. sigmoid$(\cdot)$ is an element-wise normalization function. $\mathbf{w}_k = (\mathbf{w}_k^1, \cdots, \mathbf{w}_k^{M_k}) \in \mathbb{R}^{M_k}$ denotes the generated weight vector for hyperedge group k. copy(a, b) function returns a vector of size b, and the value of which is padded by copying a by b times. Let $M = M_1 + M_2 + \cdots + M_K$ denote the summation of the hyperedges in all hyperedge groups. $\mathbf{W} \in \mathbb{R}^{M \times M}$ is a diagonal matrix that indicates the weight matrix of hypergraph, and each entry $\mathbf{W}^{ii}$ denotes the weight of the corresponding hyperedge e_i. By concatenating $(\cdot||\cdot)$ the incidence matrices of multiple hyperedge groups, $\mathbf{H} \in \{0, 1\}^{N\times M}$ can denote the incidence matrix of the hypergraph generated.

Multi-model/multi-type data can be analyzed to generate multiple hyperedge groups. From the constructed hyperedge groups, the hypergraph incidence matrix $\mathbf{H}$ and hyperedge weight matrix $\mathbf{W}$ can be generated, which can then be fed into the hypergraph convolution layer for further processing.

(2) Hypergraph Convolution

Following **Definitions 1, 2, 3**, an aggregation of neighbor vertex messages via hyperpath is introduced for one spatial hypergraph convolution layer. Given a vertex $\alpha \in \mathcal{V}$ of hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, aggregating messages from its hyperedge inter-neighbor set $\mathcal{N}_e(\alpha)$ is the aim. In order to obtain those hyperedge messages of each hyperedge β in the hyperedge inter-neighbor set $\mathcal{N}_e(\alpha)$, aggregating messages from its vertex inter-neighbor set $\mathcal{N}_v(\beta)$. After that, the two steps of hypergraph convolution make a closed loop from vertex feature sets X^t to X^{t+1}. A general spatial hypergraph convolution in the t-th layer can be defined as

$$\left\{\begin{array}{l} \left.\begin{array}{l} m_\beta^t = \sum\limits_{\alpha\in\mathcal{N}_v(\beta)} M_v^t(x_\alpha^t) \\ y_\beta^t = U_e^t(w_\beta, m_\beta^t) \end{array}\right\} \text{Stage 1} \\ \left.\begin{array}{l} m_\alpha^{t+1} = \sum\limits_{\beta\in\mathcal{N}_e(\alpha)} M_e^t(x_\alpha^t, y_\beta^t) \\ x_\alpha^{t+1} = U_v^t(x_\alpha^t, m_\alpha^{t+1}) \end{array}\right\} \text{Stage 2} \end{array}\right. , \tag{7.18}$$

where $x_\alpha^t \in \mathbf{X}^t$ denotes the input feature vector of vertex $\alpha \in \mathcal{V}$ in layer $t = 1, 2, \ldots, T$, and x_α^{t+1} denotes the updated feature of vertex α. m_β^t denotes the message of hyperedge $\beta \in \mathcal{E}$, and w_β denotes a weight associated to hyperedge β. m_α^{t+1} denotes the message of vertex α. y_β^t denotes the hyperedge feature of hyperedge β that denotes an element of hyperedge feature set $Y^t = \{y_1^t, y_2^t, \ldots, y_M^t\}$, $y_i^t \in \mathbb{R}^{C_t}$ in layer t. $M_v^t(\cdot), U_e^t(\cdot), M_e^t(\cdot), U_v^t(\cdot)$ are the vertex message function, hyperedge update functions, hyperedge message function, and vertex update function in t_{th} layer, respectively, which can be defined for specified applications.

With the high-order relationship in the hypergraph structure, the spatial hypergraph convolution layer is designed for high-level representation learning. In comparison with the graph convolution that consists of a single stage of message passing, the spatial hypergraph convolution is composed of four flexible operations with learned differentiable functions. As neighbor relations in graph, there is no natural ordering in inter-neighbors between vertices and hyperedges. Therefore, a summation operation is used to aggregate vertex–hyperedge messages from $M_v^t(\cdot)/M_e^t(\cdot)$ operation.

A simple spatial hypergraph convolution layer (named HGNNConv$^+$) via specifying the message-update functions (vertex message function $M_v^t(\cdot)$, hyperedge update function $U_e^t(\cdot)$, hyperedge message function $M_e^t(\cdot)$, and vertex update function $U_v^t(\cdot)$) is introduced as

$$\begin{cases} M_v^t(x_\alpha^t) & = \frac{x_\alpha^t}{|\mathcal{N}_v(\beta)|} \\ U_e^t(w_\beta, m_\beta^t) & = w_\beta \cdot m_\beta^t \\ M_e^t(x_\alpha^t, y_\beta^t) & = \frac{y_\beta^t}{|\mathcal{N}_e(\alpha)|} \\ U_v^t(x_\alpha^t, m_\alpha^{t+1}) & = \sigma(m_\beta^{t+1} \cdot \Theta^t) \end{cases}, \tag{7.19}$$

where $\Theta^t \in \mathbb{R}^{C^t \times C^{t+1}}$ denotes a trainable parameter of layer t, learned in training phase. $\sigma(\cdot)$ denotes an arbitrary nonlinear activation function such as $ReLU(\cdot)$, etc. Note that in Eq. (7.19), $x_\alpha^t/|\mathcal{N}_v(\beta)|$ and $y_\beta^t/|\mathcal{N}_e(\alpha)|$ denote the normalized vertex–hyperedge feature, of which convergence is accumulated and jittering is somewhat minimized.

For faster forward propagation of HGNNConv$^+$ in GPU/CPU devices, here rewrite it in the matrix format. Consider $\mathbf{X}^t$ as the input vertex feature set of layer t. From **Definitions 1, 2**, $\mathbf{H}^\top \in \{0, 1\}^{M \times N}$ can control the hyperedge inter-neighbor of each vertex feature in $\mathbf{X}^t$. Hence, it can be used to guide each vertex to aggregate and generate the hyperedge feature set Y^t, which can be formulated as $\mathbf{Y}^t = \mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{X}^t$. In a similar way, the process of updating vertex feature set $\mathbf{X}^{t+1}$ from hyperedge feature set $\mathbf{Y}^t$ can be formulated as $\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1}\mathbf{H}\mathbf{Y}^t\Theta^t)$. Thus, the matrix format of HGNNConv$^+$ can be written as

$$\mathbf{X}^{t+1} = \sigma(\mathbf{D}_v^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{X}^t\Theta^t). \tag{7.20}$$

Similar to HGNN, $\mathbf{X}^{t+1}$ can be obtained after convolution, which can be used for further learning. As an extension of HGNN [2], this method employs a broad multi-modal/multi-type data correlation model to learn an adaptive weight for each modality/type representation using a single hypergraph model.

7.3.2 Dynamic Hypergraph Neural Networks

Dynamic hypergraph neural networks (DHGNN) [15] is a kind of neural networks modeling dynamically evolving hypergraph structures, which is composed of the stacked layers of two modules: dynamic hypergraph construction and hypergraph convolution. The dynamic hypergraph construction module dynamically updates hypergraph structures on each layer as initially constructed hypergraph may not be an appropriate representation for data. After that, hypergraph convolution is introduced as a means of encoding high-order correlations between data points within a hypergraph. There are two phases in the hypergraph convolution module:

vertex convolution and hyperedge convolution, each of which is designed to aggregate features among vertices and hyperedges, respectively.

(1) Dynamic Hypergraph Construction
Symbol $\mathrm{Con}(e)$ is used to denote the vertex set that a hyperedge e contains, and the symbol $\mathrm{Adj}(v)$ is used to denote the hyperedge set where all hyperedges containing the vertex v:

$$\mathrm{Con}(e) = \left\{v_1, v_2, \ldots, v_{k_e}\right\}, \tag{7.21}$$

$$\mathrm{Adj}(v) = \left\{e_1, e_2, \ldots, e_{k_v}\right\} \tag{7.22}$$

where k_e and k_v are the number of vertices in hyperedge e, and the number of hyperedges containing vertex v. v is defined as the centroid vertex of the hyperedge set $\mathrm{Adj}(v)$. Here, traditional k-NN methods and k-means clustering methods can be combined for dynamic hypergraph construction to exploit local and global structures. On the one hand, it has computed the k-1 nearest neighbors for each vertex v. These neighborhood vertices, along with the vertex v, form a hyperedge in $\mathrm{Adj}(v)$. On the other hand, it has conducted k-means algorithm on the whole feature map of each layer according to the Euclidean distance. For each vertex, the nearest $S - 1$ clusters are assigned as to be the adjacent hyperedges of this vertex. Here, $|Adj(v)|$ denotes the size of adjacent hyperedge set, $\mathbf{x}_e$ denotes adjacent hyperedge features, and $\mathbf{x}_v$ denotes centroid vertex feature. $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters.

Such a procedure on the feature embedding of each layer is performed. Especially, it initializes hypergraph structures with the input feature embedding. Therefore, the hyperedge set is dynamically adjusted as the feature embedding evolves with network going deeper. In this way, it is able to obtain better hypergraph structures for high-order data correlation modeling with deep neural networks.

(2) Dynamic Hypergraph Convolution
Hypergraph convolution is composed of two sub-modules: vertex convolution sub-module and hyperedge convolution sub-module. By using vertex convolution, vertex features are aggregated to the hyperedge, and then by using hyperedge convolution, adjacent hyperedge features are aggregated to the center vertex.

There are several methods of pooling that can be used, including maximum pooling and average pooling. Vertex aggregation in state-of-the-art algorithms involves a fixed, pre-computed transform matrix generated from graph or hypergraph structure. Nevertheless, such methods cannot effectively model discriminative information among vertex features. For feature permutation and weighting, learn the transform matrix $\mathbf{T}$ from the vertex features. Information can flow within and between channels using the transform matrix. Using multi-layer perception (MLP), obtain the transform matrix $\mathbf{T}$ and compress the transformed features by using convolution as follows:

$$\mathbf{T} = \mathrm{MLP}\left(\mathbf{X}_v\right) \tag{7.23}$$

and

$$\mathbf{x}_e = \mathrm{conv}\left(\mathbf{T} \cdot \mathrm{MLP}\left(\mathbf{X}_v\right)\right). \tag{7.24}$$

(3) Hyperedge Convolution
Here, the hyperedge convolution is following the spatial convolution strategy, which consists of the aggregation of hyperedge features to center vertex features. Hyperedge convolution employs multi-layer perception to generate weight scores for each hyperedge. As a weighted sum of input hyperedge features, the center vertex feature is computed as an output. This procedure can be formulated as follows:

$$w = \mathrm{softmax}\left(\mathbf{x}_e \mathbf{W} + \mathbf{b}\right) \tag{7.25}$$

and

$$\mathbf{x}_v = \sum_{i=0}^{|\mathrm{Adj}(v)|} w^i \mathbf{x}_e^i. \tag{7.26}$$

As a result of these deep learning techniques, graph/hypergraph structure is taken into consideration as prior knowledge to the training of the model. There are, however, a number of hidden and important relationships that are not directly represented in the inherent structure. For vertex convolution, a transform matrix is employed to permute and weight vertices within hyperedges; for hyperedge convolution, an attention mechanism is employed to aggregate adjacent hyperedge features. Figure 7.6 shows the architecture of the DHGNN. The first part of the figure illustrates the process of the hyperedge construction. There are two hyperedges generated from two clusters (dashed ellipses), for example. In the second part, vertices within a hyperedge are aggregated to form a hyperedge feature through vertex convolution, and vertices within adjacent hyperedges are aggregated to form a center vertex feature via hyperedge convolution. In the third part, after performing such operations on all vertices in the current layer feature embedding, the new layer feature embedding and the new hypergraph structure can be constructed.

7.4 Comparison Between Graph and Hypergraph Neural Networks

After the previous introduction to the spectral-based and spatial-based hypergraph neural networks methods, we have a basic understanding of the implementation of these methods. In this section, we compare hypergraph neural networks with simple graph neural networks according to spectral and spatial areas to discover the

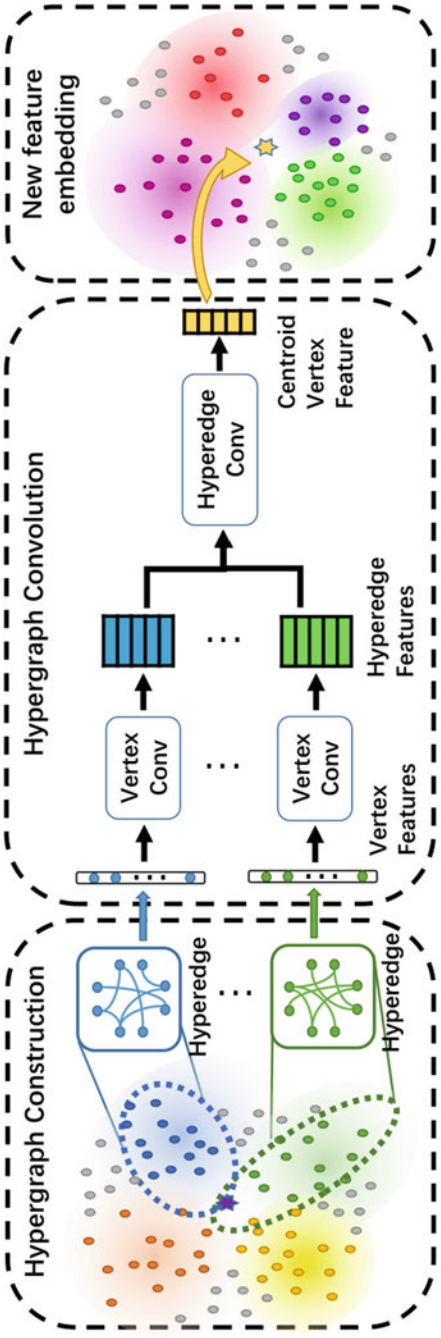


Fig. 7.6 The DHGNN framework. This figure is from [15]

connections and differences between them. The most typical methods of the two neural networks are chosen, the hypergraph neural networks model and the graph neural networks model, as a way to compare the most typical relationships and differences. HGNN [2] and HGNN$^+$ [16] are used to compare them in the spectral and spatial domains, respectively. In terms of convolution, GNN is the classical operator designed to operate on graph, such as [6, 18, 24, 25]. In this subsection, the HGNN [2] and HGNN$^+$ [16] are compared with GNN [18] from the spectral perspective and spatial perspective, respectively. Furthermore, the extended learning domain of the hypergraph emphasizes the connection.

7.4.1 Spectral Perspective

It can be proved that the GNN can be mathematically viewed as a special case of HGNN. Based on the assumption that every hyperedge connects only two nodes and has a weight equal to that of others, the simple hypergraph (2-uniform hypergraph) can also be expressed as a graph that has a graph adjacency matrix $\mathbf{A}$ and a vertex degree matrix $\mathbf{D}$, which is a construction similar to $\mathscr{E}_{\text{pair}}$. It is indicated by the hypergraph incidence matrix $\mathbf{H}$, the vertex degree matrix $\mathbf{D}_v$, the hyperedge degree matrix $\mathbf{D}_e$, and the hyperedge weight matrix $\mathbf{W}$. Under such circumstances, then the following formulations can reduce the simple hypergraph:

$$\begin{cases} \mathbf{H}\mathbf{H}^\top &= \mathbf{A} + \mathbf{D} \\ \mathbf{D}_e^{-1} &= \frac{1}{2}\mathbf{I} \\ \mathbf{W} &= \mathbf{I} \end{cases} . \tag{7.27}$$

This can be reduced as follows using the hypergraph convolution:

$$\begin{aligned} \mathbf{X}^{t+1} &= \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^t\Theta^t) \\ &= \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}(\tfrac{1}{2}\mathbf{I})\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^t\Theta^t) \\ &= \sigma(\tfrac{1}{2}\mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{D})\mathbf{D}^{-1/2}\mathbf{X}^t\Theta^t) \\ &= \sigma(\tfrac{1}{2}(\mathbf{I}+\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{X}^t\Theta^t) \\ &= \sigma(\mathbf{D}^{-1/2}\hat{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{X}^t\hat{\Theta}^t) \end{aligned} , \tag{7.28}$$

where $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\hat{\Theta}^t = \frac{1}{2}\Theta^t$. The extra $\frac{1}{2}$ can be absorbed by the learnable parameter Θ. It appears that in modeling the simple graph, the spectral-based hypergraph convolution in HGNN [2] exhibits the same formation as the graph convolution in GCN [18]. Due to its powerful expressive capabilities, the hypergraph convolution not only models and learns the high-order correlation in the hypergraph, but also it has the ability to handle simple graph.

7.4.2 Spatial Perspective

Learning to embed the rooted subtree in low-dimensional space can be viewed as a powerful GNN model [26]. Not only can rooted subtree [27] describe the connections of local vertices, but it can also describe message passing paths in a graph. The rooted subtree can therefore be used to compare HGNN$^+$ [16] with GNN [18]. In hypergraph, the node in the rooted subtree of hypergraph can either be a vertex or a hyperedge in order to satisfy the path definition (also known as the message passing path).

Comparing graph structures that are isomorphic is more straightforward. Therefore, 2-uniform hypergraph (each hyperedge connects only two vertices) is compared. Figure 7.7 displays the rooted subtree for HGNN$^+$ [16] and GNN [18] for a specified vertex, which can also be expressed as the message path in graph and hyperpath in hypergraph. It is obvious that in graph convolution, the vertex features of the neighbors are taken into account. These features are then aggregated to update the central vertex feature at the end of the process. This layer can be described as a hierarchical structure that enables the development of more powerful expressions and modeling capabilities. HGNN$^+$ [16] performs a two stage, i.e., vertex–hyperedge–vertex, transformation. As formulated in Eq. (7.18), the first stage of the procedure generates a hyperedge feature based on the vertex inter-neighboring of the vertex. As a result, the hyperedge inter-neighbor's features are aggregated to get the updated features of the vertices. Additionally, multi-layer hypergraph convolution has much more message interactions than graph convolution. The rooted vertex appears more frequently in the HGNN$^+$ [16] path of subtrees (like a latent extra self-loop), which accounts for its better performance. In comparison with graph convolution, hypergraph convolution can efficiently extract low- and high-order correlations on hypergraph via vertex–hyperedge–vertex transformation.

7.5 Summary

In this chapter, we introduce two types of hypergraph neural networks learning: spectral-based and spatial-based methods. In spectral-based methods, the hypergraph transforms the nodes in the common and spectral domains by computing the Laplacian matrix. In the spatial-based methods, each node is updated by aggregating information from the nodes on the spatial domain. Then, we consider that most learning methods in graph learning are still simple graph neural networks.

Finally, we also compare hypergraph neural networks and graph neural networks on the previous spectral-based spatial-based and others. According to the comparison of the convolutional computation coefficients, the hypergraph convolution can not only have the comparable expressive ability of GCN when handling a simple graph, but also is capable of modeling and learning high-order correlations within

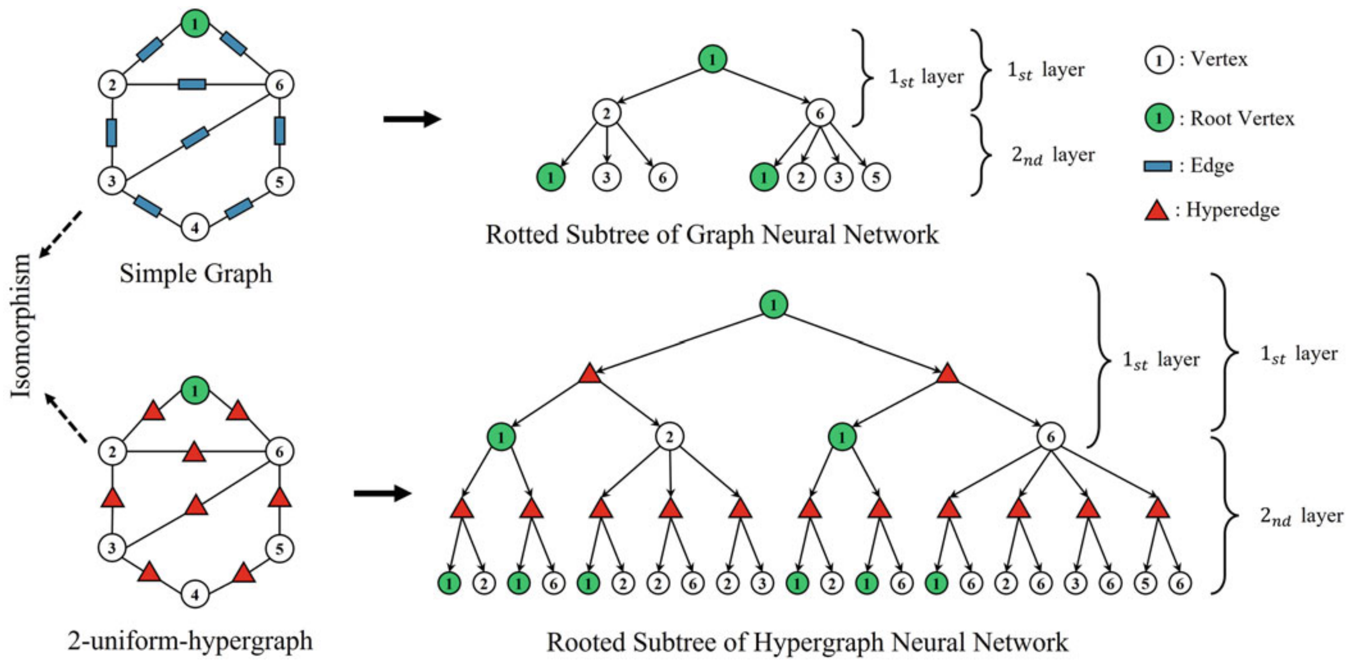


Fig. 7.7 Comparison of rooted subtree of graph and 2-uniform hypergraph. This figure is from [16]

the hypergraph. Comparing hypergraph convolution with graph convolution based on spatial domain comparison, we can find that hypergraph convolution layer can efficiently extract both low-order and high-order correlations on hypergraph using the vertex–hyperedge–vertex transformation.

References

1. D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: Clustering, classification, and embedding, in *Proceedings of the Advances in Neural Information Processing Systems* (2006), pp. 1601–1608
2. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
3. N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, P. Talukdar, HyperGCN: A new method for training graph convolutional networks on hypergraphs, in *Proceedings of the Advances in Neural Information Processing Systems* (2019)
4. S. Bai, F. Zhang, P.H. Torr, Hypergraph convolution and hypergraph attention. Pattern Recogn. **110**, 107637 (2021)
5. M. Welling, T.N. Kipf, Semi-supervised classification with graph convolutional networks, in *Proceedings of the International Conference on Learning Representations* (2016)
6. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in *Proceedings of the International Conference on Learning Representations* (2017)
7. C. Yang, R. Wang, S. Yao, T. Abdelzaher, Hypergraph learning with line expansion (2020). Preprint arXiv:2005.04843
8. J. Atwood, D. Towsley, Diffusion-convolutional neural networks, in *Proceedings of the Advances in Neural Information Processing Systems* (2016)
9. F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model cnns, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5115–5124
10. U. Chitra, B. Raphael, Random walks on hypergraphs with edge-dependent vertex weights, in *Proceedings of the International Conference on Machine Learning* (2019), pp. 1172–1181
11. S. Agarwal, K. Branson, S. Belongie, Higher order learning with graphs, in *Proceedings of the International Conference on Machine Learning* (2006), pp. 17–24
12. T.-H.H. Chan, A. Louis, Z.G. Tang, C. Zhang, Spectral properties of hypergraph laplacian and approximation algorithms. J. Appl. Comput. Mech. **65**(3), 1–48 (2018)
13. P. Li, O. Milenkovic, Inhomogeneous hypergraph clustering with applications, in *Proceedings of the Advances in Neural Information Processing Systems* (2017)
14. P. Li, O. Milenkovic, Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering, in *Proceedings of the International Conference on Machine Learning* (2018), pp. 3014–3023
15. J. Jiang, Y. Wei, Y. Feng, J. Cao, Y. Gao, Dynamic hypergraph neural networks. in *Proceedings of the International Joint Conference on Artificial Intelligence* (2019), pp. 2635–2641
16. Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN+: General hypergraph neural networks. IEEE Trans. Pattern Analy. Mach. Intell. **45**(3), 3181–3199 (2023)
17. J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs (2013). Preprint arXiv:1312.6203
18. T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in *Proceedings of the International Conference on Learning Representations* (2017)
19. I. Chami, Z. Ying, C. Ré, J. Leskovec, Hyperbolic graph convolutional neural networks, in *Proceedings of the Advances in Neural Information Processing Systems* (2019)

20. M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in *Proceedings of the Advances in Neural Information Processing Systems* (2016)
21. M. Nickel, D. Kiela, Poincaré embeddings for learning hierarchical representations, in *Proceedings of the Advances in Neural Information Processing Systems* (2017)
22. O. Ganea, G. Bécigneul, T. Hofmann, Hyperbolic neural networks, in *Proceedings of the Advances in Neural Information Processing Systems* (2018)
23. A. Li, B. Yang, H. Chen, G. Xu, Hyperbolic neural collaborative recommender. IEEE Trans. Knowl. Data Eng. 1–12 (2021). https://doi.org/10.1109/TKDE.2022.3221386
24. J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in *Proceedings of the International Conference on Machine Learning* (2017), pp. 1263–1272
25. W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in *Proceedings of the Advances in Neural Information Processing Systems* (2017)
26. K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in *Proceedings of the International Conference on Learning Representations* (2019)
27. H.E. Manoochehri, S.S. Kadiyala, M. Nourani, Predicting drug-target interactions using weisfeiler-lehman neural network, in *Proceedings of International Conference on Biomedical & Health Informatics* (2019), pp. 1–4

Chapter 8
Large Scale Hypergraph Computation

Abstract As introduced in the previous chapters, the complexity of hypergraph computation is relatively high. In practical applications, the hypergraph may not be in a small scale, where we often encounter the scenario that the size of the hypergraph is very large. Therefore, hypergraph computation confronts the complexity issues in many applications. Therefore, how to handle large scale data is an important task. In this chapter, we discuss the computation methods for large scale hypergraphs and their applications. Two types of hypergraph computation methods are provided to handle large scale data, namely the factorization-based hypergraph reduction method and hierarchical hypergraph learning method. In the factorization-based hypergraph reduction method, the large scale hypergraph incidence matrix is reduced to two low-dimensional matrices. The computing procedures are conducted with the reduced matrices. This method can support the hypergraph computation with more than 10,000 vertices and hyperedges. On the other hand, the hierarchical hypergraph learning method splits all samples as some sub-hypergraphs and merges the results obtained from each sub-hypergraph computation. This method can support hypergraph computation with millions of vertices and hyperedges.

8.1 Introduction

Hypergraph computation has been used in many areas such as image analysis [1–3] and recommendation [4–6]. In practical applications, the hypergraph may not be in a small scale, and the size of the hypergraph could be very large in many cases, where hypergraph computation confronts the complexity issues [7–13]. For instance, in medical image analysis, hypergraphs can be used to model the relationship among case patches within an image or different images. Here we take the gigapixel whole-slide histopathological images (WSIs) as an example. The large scale of pixels in WSIs leads to a great challenge for medical image analysis. If we generate a hypergraph for such pixels in WSIs, the scale of vertices tends to be in billion level. Even we sample patches in WSIs, this number can be still around tens of thousands, or in million level. The conventional hypergraph modeling methods are

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_8

highly unlikely able to analyze such large scale pixels. Another example is the recommender system. In recommender system, graphs or hypergraphs have been very widely used with their superior structural modeling capabilities. Meanwhile, the number of uses and items in the Internet or the recommender systems can be in million to billions level, and even keep increasing. Consequently, recommender systems are one of the typical playgrounds for large scale hypergraph applications. The large scale problem of hypergraphs is encountered in many other areas, such as social network analysis, protein relations prediction, and so on.

Under such circumstances, hypergraph computation confronts the large scale issue, as the modeling and computing on hypergraph are with high complexity in general. To help solve the large scale problem, we introduce two types of hypergraph computational methods to handle large scale data in this chapter, namely the factorization-based hypergraph reduction method and hierarchical hypergraph computation method. We also introduce their applications in medical image analysis and recommender systems, respectively. The factorization-based hypergraph reduction reduces the large scale hypergraph incidence matrix $\mathbf{H}$ to two low-dimensional matrices, leading to the reduction of the complexity. This method can support the hypergraph computation with tens of thousands vertices. The other method, i.e., the hierarchical hypergraph computation, splits the vertices to several subsets and computes each sub-hypergraph, respectively. The results from these sub-hypergraphs can be further combined following a hierarchical strategy. This method can support the hypergraph computation with millions of vertices and hyperedges. Part of the work introduced in this chapter has been published in [8].

8.2 Factorization-Based Big-Hypergraph Modeling

The complexity of the incident matrix $\mathbf{H} \in \mathbb{R}^{N\times E}$ is $\mathscr{O}(N^2)$, which rises rapidly with respect to the increasing of the number of vertices ($|\mathscr{V}| = N$) and the number of hyperedges ($|\mathscr{E}| = N$). Although hypergraphs can model high-order complex associations well, the incidence matrix cannot take up a sizable number of vertices in traditional hypergraph modeling and transductive computation strategy. This is one typical bottleneck that limits the applications of hypergraph computation. To address this problem, the factorization-based hypergraph reduction method [8] is introduced to handle hypergraph modeling and computing with tens of thousands vertices.

It is an effective way to reduce dimensionality by conducting matrix decomposition of matrices with high dimensionality into the product of matrices with small dimensionality and has been applied in different areas such as spectral clustering [14] and recommendation algorithms [15]. For a large-dimensional incidence matrix $\mathbf{H}$ for a hypergraph, matrix decomposition can also be used to find the low-dimensional embeddings of each vertex and hyperedge and support large scale hypergraph computation.

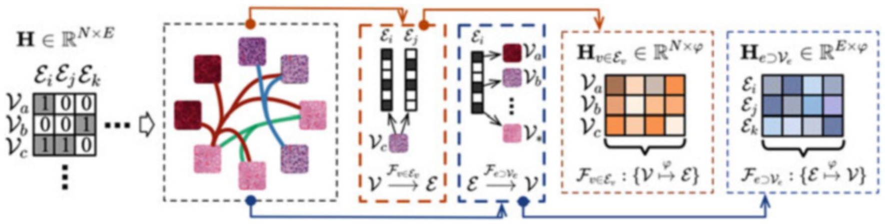

$\mathbf{H} \in \mathbb{R}^{N\times E}$
$\mathcal{E}_i \mathcal{E}_j \mathcal{E}_k$
$\mathcal{V}_a$ 1 0 0
$\mathcal{V}_b$ 0 0 1
$\mathcal{V}_c$ 1 1 0
$\mathcal{V} \xrightarrow{\mathcal{F}_{v\in\mathcal{E}_v}} \mathcal{E}$
$\mathcal{E} \xrightarrow{\mathcal{F}_{e\supset\mathcal{V}_e}} \mathcal{V}$
$\mathbf{H}_{v\in\mathcal{E}_v} \in \mathbb{R}^{N\times\varphi}$
$\mathcal{F}_{v\in\mathcal{E}_v} : \{\mathcal{V} \overset{\varphi}{\mapsto} \mathcal{E}\}$
$\mathbf{H}_{e\supset\mathcal{V}_e} \in \mathbb{R}^{E\times\varphi}$
$\mathcal{F}_{e\supset\mathcal{V}_e} : \{\mathcal{E} \overset{\varphi}{\mapsto} \mathcal{V}\}$

Fig. 8.1 The pipeline of the factorization-based hypergraph reduction method. This figure is from [8]

As illustrated in Fig. 8.1, the factorization-based hypergraph reduction incorporates a factor embedding component that encodes the relationships between hyperedges and vertices into two latent semantic spaces. Due to the low dimension of the latent semantic space, it can handle more vertices and hyperedges accordingly.

The purpose of factorization is to reduce the dimension of the incident matrix $\mathbf{H}$ to two semantic spaces, including vertex-belonging hyperedge represented by $\mathbf{H}_{v\in\mathscr{E}_v} \in \mathbb{R}^{N\times\varphi}$ and hyperedge-containing-vertices represented by $\mathbf{H}_{e\supset\mathscr{V}_e} \in \mathbb{R}^{E\times\varphi}$, where $\mathscr{E}_v$ and $\mathscr{V}_e$ represent the hyperedge set containing vertex v and vertex set in hyperedge e, respectively, and φ is a hyperparameter that represents the latent semantic space dimension. Figure 8.1 illustrates that the two latent semantic spaces aim to express all connections between vertices and hyperedges. This procedure is formulated as below:

$$\arg \min_{\mathbf{H}_{v\in\mathscr{E}_v}, \mathbf{H}_{e\supset\mathscr{V}_e}} \left\{ ||\mathbf{H} - \mathbf{H}_{v\in\mathscr{E}_v}\mathbf{H}_{e\supset\mathscr{V}_e}^\top||_2^2 \right\}. \tag{8.1}$$

Consequently, the corresponding loss generated by the hypergraph dimensionality reduction can be written as

$$\mathscr{L}_\gamma = ||\mathbf{H} - \mathbf{H}_{v\in\mathscr{E}_v}\mathbf{H}_{e\supset\mathscr{V}_e}^\top||_2^2. \tag{8.2}$$

The hypergraph Laplacian matrix $\mathbf{L}$ is another crucial component of hypergraph computation, with the ordinary form is $\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$. Since the incident matrix $\mathbf{H}$ has two low-dimensional latent semantic spaces, the low-dimensional hypergraph factorization-based Laplacian $\mathbf{L}_F$ is formulated as

$$\mathbf{L}_F = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}_{v\in\mathscr{E}_v} \underbrace{\mathbf{H}_{e\supset\mathscr{V}_e}^\top \mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}_{e\supset\mathscr{V}_e}}_{\Sigma\in\mathbb{R}^{\varphi\times\varphi}} \mathbf{H}_{v\in\mathscr{E}_v}^\top \mathbf{D}_v^{-1/2}, \tag{8.3}$$

where $\Sigma = \mathbf{H}_{e\supset\mathscr{V}_e}^\top \mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}_{e\supset\mathscr{V}_e}$ is an intermediate latent feature multiplication term of dimension φ. Because the latent semantic space dimension φ is significantly smaller than the total amount of vertices and hyperedges, the multiplication intermediate term Σ functions as an extended control coefficient matrix.

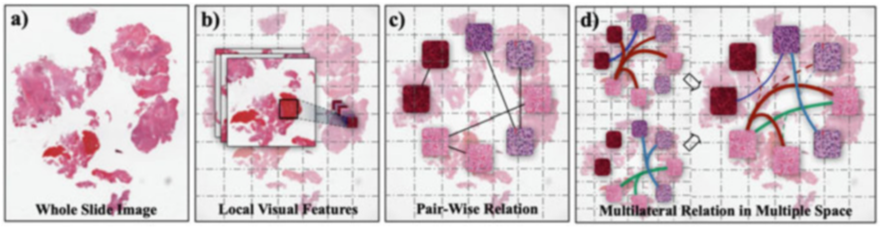


Fig. 8.2 (**a**) The whole-slide image for survival prediction; (**b**) Local feature extraction with convolution networks; (**c**) Feature aggregation with pairwise relation; (**d**) Global feature representation with high-order relation and multiple spaces. This figure is from [1]

The factorization-based hypergraph reduction can be used in hypergraph neural networks to support large scale computation, which can be used for more than 10,000 vertices and hyperedges.

Here we illustrate an application of hypergraph computation for large scale medical image analysis using whole-slide histopathology images for survival prediction. The goal is to make predictions by extracting valid survival-specific features reflecting the survival status of a patient based on a whole section histopathology image. Unlike conventional images, WSI data can be very large, i.e., a single image may have billions of pixels, and the correlations of these data are very complicated. Therefore, hypergraph computation in this application meets the large scale issue. The existing medical image analysis models are designed for analyzing natural images with a much smaller size, such as $256px \times 256px$ or more. In order for the model to handle these WSI data, a number of patches of a moderate size are usually sampled first. Some patches of a moderate size (e.g., 256×256) are extracted from each WSI, and then these patches are stacked up and fed into a CNN-based feature extractor (e.g., VGG) to generate a global representation, as shown in Fig. 8.2. Subsequently, a regression model is applied to the global features to predict the survival score. These methods have an obvious drawback that the structure of the entire histopathological image is broken into pieces by patch sampling.

It may be unrealistic to extract all of the structural information at the cellular level from gigapixel images because there is an apparently massive amount of pixel data that are included in a single histopathological image. A small number of image patches can be selected to generate graph-based models. The global feature can be extracted by this method. However, the number of sampled patches limits the sampling area's coverage to the original image's informative regions, which causes a serious portion of fields with pathological features to be missing. The incident matrix, which represents the connectivity between vertices and hyperedges, is an essential component of the hypergraph neural network. The large scale vertices and hyperedges in the constructed hypergraph limit the application of HGNN [16].

Here, we introduce the Big-Hypergraph Factorization Neural Network (b-HGFN) [8], which uses factorization-based hypergraph reduction to address the above issue. It incorporates a factor embedding component that encodes the relationships between hyperedges and vertices into two latent semantic spaces, as illustrated in Fig. 8.3. Due to the low dimension of the latent semantic space, b-HGFN can handle more vertices and hyperedges. With the hypergraph reduction, b-HGFN can provide more accurate feature representations of histopathological images from more densely sampled patches. Consequently, the first loss generated by the hypergraph dimensionality reduction can be written as Eq. (8.2). The hypergraph Laplacian matrix $\mathbf{L}$ is another crucial component of b-HGFN, and the low-dimensional hypergraph factorization Laplacian $\mathbf{L}_F$ is formulated as Eq. (8.3). A standard hypergraph neural network layer is represented as

$$\text{HGFConv}(\cdot) = D\Big[\sigma(\Theta^{(\cdot)}\mathbf{X}^{(\cdot)}(\mathbf{I} - \mathbf{L}_F))\Big], \tag{8.4}$$

where σ stands for the nonlinear activation function, and D represents the dropout layer. Convolution operations are embedded into the implicit latent semantic space by modifying the convolution network's specifics, which are denoted as

$$\begin{cases} \text{HGFConv}(0) & = D\Big[\sigma(\Theta^{(0)}\mathbf{X}^{(0)}\mathbf{D}_v^{-1/2}\mathbf{H}_{v\in\mathscr{E}_v}\Sigma)\Big] \\ \text{HGFConv}(1) & = D[\sigma(\Theta^{(1)}\mathbf{X}^{(1)}\Sigma)] \\ \ldots & \\ \text{HGFConv}(L-1) & = D[\sigma(\Theta^{(L-1)}\mathbf{X}^{(L-1)}\Sigma)] \\ \text{HGFConv}(L) & = D\Big[\sigma(\Theta^{(L)}\mathbf{X}^{(L)}\Sigma\mathbf{H}_{v\in\mathscr{E}_v}^{\top}\mathbf{D}_v^{-1/2})\Big] \end{cases}. \tag{8.5}$$

According to the HGFConv mentioned above, the hypergraph's high-dimensional connection relations can be embedded in the low-dimensional latent semantic spaces. To represent global features (i.e., $\mathbf{X} \in \mathbb{R}^{1\times C_{L+1}}$) at the histopathological image level, the output of the last layer of HGFConv (i.e., $\mathbf{X}^{(L+1)}$) is squeezed by a pooling layer after a complete b-HGFN.

The patient survival duration prediction is calculated using a fully connected neural network after obtaining the histopathological image's feature representation. The hierarchical loss, which incorporates list-wise loss, pairwise loss, and point-wise loss, has been experimentally demonstrated to be more effective for b-HGFN than using the simply pairwise Bayesian Concordance Readjust (BCR) loss function. The point-wise loss function applies negative Cox log partial likelihood loss as

$$\mathscr{L}_\alpha = \sum \delta_i \left(-s_i + \log \sum_{j\in\{j:t_j\le t_i\}} \exp(t_j) \right), \tag{8.6}$$

where s_i and t_i represent the predicted duration and the truth, while the pairwise loss and list-wise loss refer to NDCGLoss2 derived by LambdaLoss [17] and BCR loss [2]. Taken into consideration the loss function of hypergraph dimension reduction,

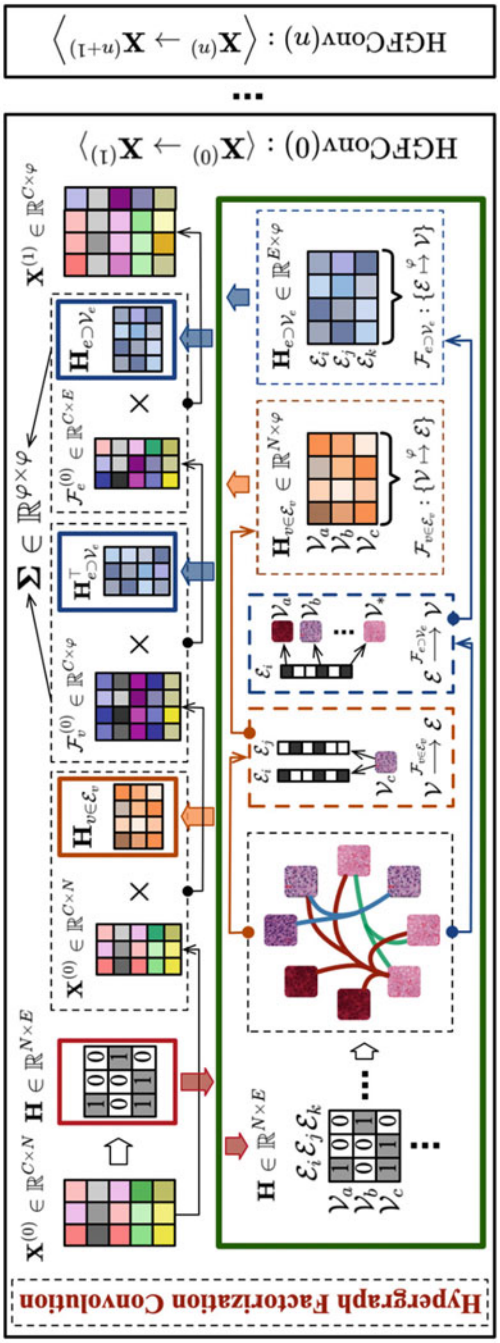

Hypergraph Factorization Convolution
HGFConv(0) : $\langle \mathbf{X}^{(0)} \rightarrow \mathbf{X}^{(1)} \rangle$
HGFConv(n) : $\langle \mathbf{X}^{(n)} \rightarrow \mathbf{X}^{(n+1)} \rangle$
$\mathbf{X}^{(0)} \in \mathbb{R}^{C\times N}$
$\mathbf{H} \in \mathbb{R}^{N\times E}$
$\mathbf{H}_{v\in\mathcal{E}_v}$
$\mathcal{F}_v^{(0)} \in \mathbb{R}^{C\times\varphi}$
$\mathbf{H}_{e\supset\mathcal{V}_e}^{\top}$
$\mathbf{\Sigma} \in \mathbb{R}^{\varphi\times\varphi}$
$\mathcal{F}_e^{(0)} \in \mathbb{R}^{C\times E}$
$\mathbf{H}_{e\supset\mathcal{V}_e}$
$\mathbf{X}^{(1)} \in \mathbb{R}^{C\times\varphi}$
$\mathcal{V} \xrightarrow{\mathcal{F}_{v\in\mathcal{E}_v}} \mathcal{E}$
$\mathcal{E} \xrightarrow{\mathcal{F}_{e\supset\mathcal{V}_e}} \mathcal{V}$
$\mathbf{H}_{v\in\mathcal{E}_v} \in \mathbb{R}^{N\times\varphi}$
$\mathcal{F}_{v\in\mathcal{E}_v} : \{\mathcal{V} \overset{\varphi}{\mapsto} \mathcal{E}\}$
$\mathbf{H}_{e\supset\mathcal{V}_e} \in \mathbb{R}^{E\times\varphi}$
$\mathcal{F}_{e\supset\mathcal{V}_e} : \{\mathcal{E} \overset{\varphi}{\mapsto} \mathcal{V}\}$

Fig. 8.3 An illustration of the Big-Hypergraph Factorization Neural Network (b-HGFN) that extracts global representation features from informative sampling patches. This figure is from [8]

the combination of all loss functions can be expressed as

$$\begin{cases} \mathscr{L}_\lambda &= \lambda \mathscr{L}_\alpha + (1-\lambda)\mathscr{L}_\beta \\ \mathscr{L}_\beta &= \{\text{NDCGLoss2}(\mathbb{S}, \mathbb{G}), \text{BCRLoss}(\mathbb{S}, \mathbb{G})\} \\ \mathscr{L}_{\text{all}} &= \mathscr{L}_\gamma + \mathscr{L}_\lambda \end{cases} . \tag{8.7}$$

The factorization-based hypergraph reduction incorporates a factor embedding component that encodes the relationships between hyperedges and vertices into two latent semantic spaces. Due to the low dimension of the latent semantic space, it can handle more vertices and hyperedges. The factorization-based hypergraph reduction can be used in HGNN [16] to solve the large scale problem. The method can effectively solve the hypergraph analysis problem with almost 10,000 vertices and hyperedges.

8.3 Hierarchical Hypergraph Modeling

The factorization-based hypergraph reduction can effectively analyze the hypergraph with almost 10,000 vertices and hyperedges, while it stretches its limit when the size extends to hypergraph with millions of vertices or hyperedges. Figure 8.4 shows a hierarchical hypergraph learning method for large scale hypergraphs with hierarchical labels. The hierarchical hypergraph can handle the hypergraph neural network with millions of data points. In the following, it is introduced in detail.

For million-scale unstructured data, it is impractical to convert the whole dataset into a single large hypergraph to represent the correlations of samples or conduct the factorization-based reduction, which would require an unrealistically large incidence matrix or a significant cost of computing memory. If there are hierarchical labels in the dataset, hierarchical hypergraph learning can be adopted to solve the

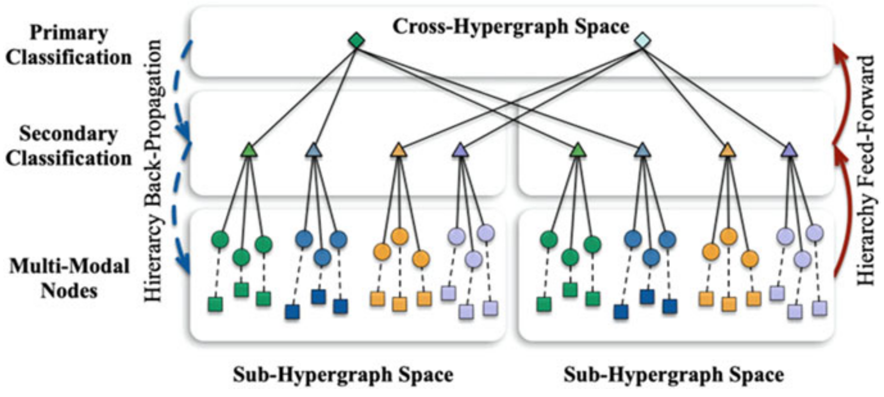


Fig. 8.4 An illustration of the hierarchical hypergraph learning

problem. The original dataset $\mathbf{X} \in \mathbb{R}^{N\times d}$ can be randomly divided uniformly into several subsets with smaller and more affordable scales, with that N denotes the scale of dataset and d denotes the dimension of sample. Then, each sample in the dataset forms vertices and hyperedges. In each subset, we construct a sub-hypergraph using the K nearest neighbors algorithm (kNN), which is based on the Euclidean distance between the representations of each pair of vertices. The incidence matrix $\mathbf{H}_i \in \mathbb{R}^{|\mathcal{V}_i|\times|\mathcal{E}_i|}$ serves as the role of indicating the correlation among vertices and the hyperedges, of values consisting of 0 and 1.

Given the initial feature matrix of vertices $\mathbf{X}$ as well as the corresponding incidence matrix $\mathbf{H}$, we use $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i \rangle$, $(i = 1, 2, 3, \ldots, m)$ to represent the i-th hypergraph that contains $|\mathcal{V}_i|$ vertices and $|\mathcal{E}_i|$ hyperedges. In order to weaken the loss of feature over-smooth in the convolutional operations, the residual connection [4] can be adopted to generate the updated vertex representations for the next layer of convolution, formulated as follows:

$$\widehat{\mathbf{X}}_i = \sigma(D_i^{-1/2} H_i W_i \mathcal{D}_i^{-1} H_i^{\top} D_i^{-1/2} \mathbf{X}_i \Theta_i + \mathbf{X}_i), \tag{8.8}$$

where $D_i \in \mathbb{R}^{|\mathcal{V}_i|\times|\mathcal{V}_i|}$ and $\mathcal{D}_i \in \mathbb{R}^{|\mathcal{E}_i|\times|\mathcal{E}_i|}$ are degree matrices of vertex and hyperedge. $W_i = diag(w_1, w_2, \ldots, w_{|\mathcal{E}_i|})$ and $\Theta_i \in \mathbb{R}^{d\times d}$ indicate the trainable weight parameters of the hyperedges and trainable weight matrix for feature transformation.

Note that here we assume that each sample has two hierarchical labels, named secondary label and primary label, and in which secondary label is the fine-grained category of the primary label. One special component in this first step is the "vertex belonging matrix," denoted as $\Gamma_i \in \mathbb{R}^{|\mathcal{V}_i|\times \mathcal{N}_2}$, where $\mathcal{N}_2$ is the number of secondary labels. The matrix Γ_i is generated by the labels in the training set and serves as the input for the transductive learning method.

The global labels shared by all the subsets are usually in the magnitude of hundreds, making it feasible to combine the independently learned label features of different groups. Obtaining the local latent high-order representations of subsets in the previous hypergraph learning step, two aggregating operations can be conducted here for primary and secondary labels classification, respectively. The aggregation of local secondary labels can be formulated as follows:

$$\mathbf{S}_i = \Gamma_i^{\top} \widehat{\mathbf{X}}_i, \tag{8.9}$$

where $\mathbf{X}_i$ denotes the aggregated local representation for secondary label, whose dimension is $\mathbb{R}^{\mathcal{N}_2\times d}$. Each row of the matrix $\mathbf{S}_i$ represents the latent feature for each specific category of secondary label in the i-th subset.

We then concatenate all of the local high-order vertices' features $\widehat{\mathbf{X}}_i$ to generate the global high-order vertices' features $\widehat{\mathbf{X}} \in \mathbb{R}^{|\mathbf{V}|\times d}$ as follows:

$$\widehat{\mathbf{X}} = \left[\widehat{\mathbf{X}}_1^{\top} \| \widehat{\mathbf{X}}_2^{\top} \| \cdots \| \widehat{\mathbf{X}}_m^{\top}\right]^{\top}, \tag{8.10}$$

where $\cdot\|\cdot$ denotes the concatenating operation between two matrices. The local aggregated secondary features $\mathbf{S}_i \in \mathbb{R}^{\mathcal{N}_2 \times d}$ can be further fused to form the global secondary features $\mathbf{S} \in \mathbb{R}^{\mathcal{N}_2 \times d}$ by average pooling, formulated as follows:

$$\mathbf{S} = \mathbf{S}_1 \oplus \mathbf{S}_2 \oplus \cdots \oplus \mathbf{S}_m, \tag{8.11}$$

where $\oplus$ denotes the average pooling operation, calculating the mean value of the corresponding latent features from the local secondary labels.

The global high-order representation of primary labels ($\mathbf{P} \in \mathbb{R}^{\mathcal{N}_1 \times d}$) is yielded from the global features of secondary labels, formulated as

$$\mathbf{P} = \Phi \mathbf{S}, \tag{8.12}$$

where $\Phi \in \mathbb{R}^{\mathcal{N}_1 \times \mathcal{N}_2}$ denotes the owning relations between secondary and primary labels.

Based on the results of the hypergraph convolution and global aggregation, the classifier consisting of the fully connected layers can be trained by concatenating the updated vertices' high-order representations and the global classification. The augmented representations of vertices are shown below:

$$\begin{cases} \widetilde{\mathbf{X}}_i^{<1>} = \widehat{\mathbf{X}}_i \parallel \frac{1}{\mathcal{N}_1} \sum_{j=1}^{\mathcal{N}_1} \mathbf{P}_j \\ \widetilde{\mathbf{X}}_i^{<2>} = \widehat{\mathbf{X}}_i \parallel \frac{1}{\mathcal{N}_2} \sum_{j=1}^{\mathcal{N}_2} \mathbf{S}_j \end{cases}. \tag{8.13}$$

Then the aggregated features can be used for some tasks and trained with the hierarchical labels in training set. In the following, we introduce the hierarchical hypergraph learning in recommendation.

Here, we introduce an application of hierarchical hypergraph learning for large scale user retrieval intention detection. Figure 8.5 shows the layout, which mainly consists of three steps: data division and local hypergraph modeling, latent high-order feature aggregation, and user intention prediction, respectively.

First, we randomly divide the original dataset uniformly into several subsets. In our work, every query log and the relationships between numerous query logs form vertices and hyperedges. As shown in Fig. 8.5, the whole original dataset and the divided subsets are, respectively, denoted as $\mathbf{V}$ and $\mathcal{V}_i$, where $i \in [1, 2, \ldots, m]$. And in each subset, a sub-hypergraph can be constructed, which is introduced above. Note that the initial semantic embeddings of vertices ($\mathbf{X}_i \in \mathbb{R}^{|\mathcal{V}_i| \times d}$) are extracted by the well-known pre-trained models, such as BERT [18], where d denotes the dimension of embeddings.

The hierarchical hypergraph learning can then be used to conduct the user intention prediction. In our research, the user intentions are categorized into two levels, i.e., the primary label and the secondary label, which is the fine-grained category of the primary label. After applying the hierarchical model, the features $\widetilde{\mathbf{X}}_i^{<1>}$ and $\widetilde{\mathbf{X}}_i^{<2>}$ can be obtained.

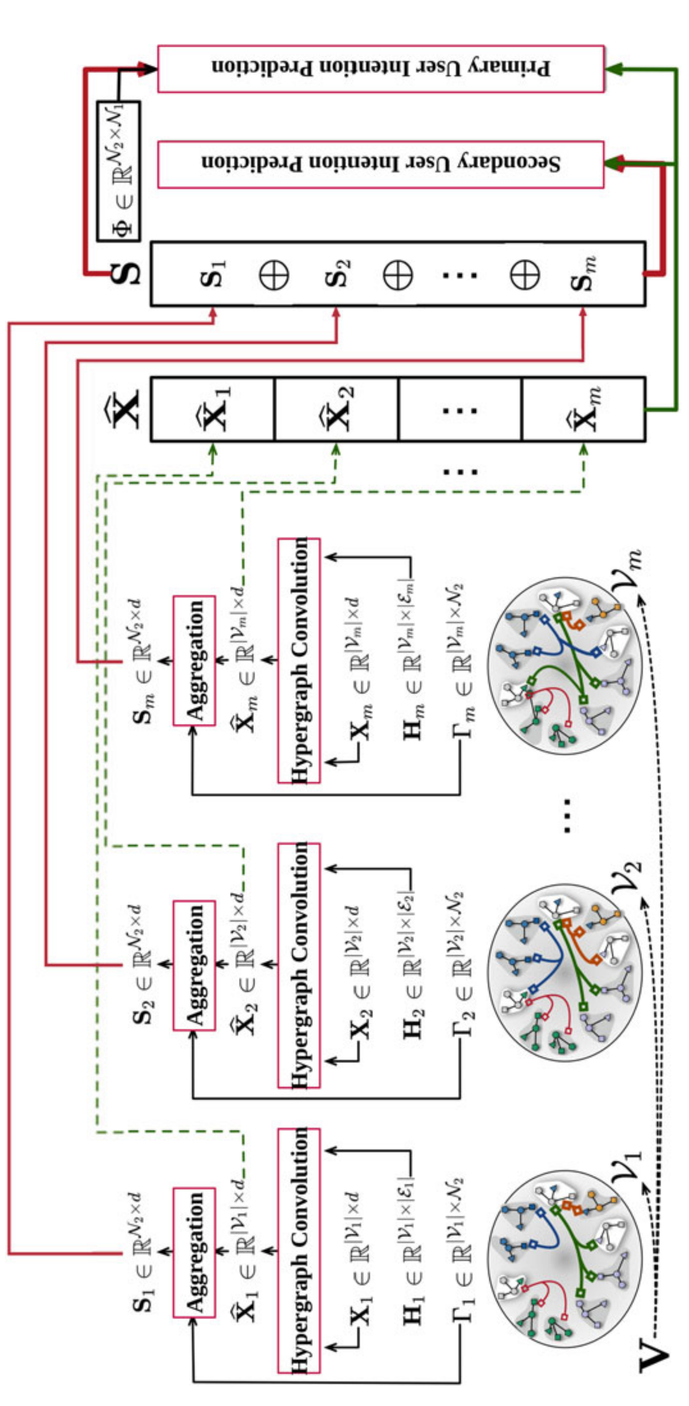

Primary User Intention Prediction
Secondary User Intention Prediction
$\Phi \in \mathbb{R}^{\mathcal{N}_2 \times \mathcal{N}_1}$
$\mathbf{S}$
$\mathbf{S}_1 \oplus \mathbf{S}_2 \oplus \cdots \oplus \mathbf{S}_m$
$\widehat{\mathbf{X}}$
$\widehat{\mathbf{X}}_1$ $\widehat{\mathbf{X}}_2$ $\cdots$ $\widehat{\mathbf{X}}_m$
$\mathbf{S}_1 \in \mathbb{R}^{\mathcal{N}_2 \times d}$
Aggregation
$\widehat{\mathbf{X}}_1 \in \mathbb{R}^{|\mathcal{V}_1| \times d}$
Hypergraph Convolution
$\mathbf{X}_1 \in \mathbb{R}^{|\mathcal{V}_1| \times d}$
$\mathbf{H}_1 \in \mathbb{R}^{|\mathcal{V}_1| \times |\mathcal{E}_1|}$
$\Gamma_1 \in \mathbb{R}^{|\mathcal{V}_1| \times \mathcal{N}_2}$
$\mathbf{S}_2 \in \mathbb{R}^{\mathcal{N}_2 \times d}$
Aggregation
$\widehat{\mathbf{X}}_2 \in \mathbb{R}^{|\mathcal{V}_2| \times d}$
Hypergraph Convolution
$\mathbf{X}_2 \in \mathbb{R}^{|\mathcal{V}_2| \times d}$
$\mathbf{H}_2 \in \mathbb{R}^{|\mathcal{V}_2| \times |\mathcal{E}_2|}$
$\Gamma_2 \in \mathbb{R}^{|\mathcal{V}_2| \times \mathcal{N}_2}$
$\mathbf{S}_m \in \mathbb{R}^{\mathcal{N}_2 \times d}$
Aggregation
$\widehat{\mathbf{X}}_m \in \mathbb{R}^{|\mathcal{V}_m| \times d}$
Hypergraph Convolution
$\mathbf{X}_m \in \mathbb{R}^{|\mathcal{V}_m| \times d}$
$\mathbf{H}_m \in \mathbb{R}^{|\mathcal{V}_m| \times |\mathcal{E}_m|}$
$\Gamma_m \in \mathbb{R}^{|\mathcal{V}_m| \times \mathcal{N}_2}$
$\mathbf{V}$
$\mathcal{V}_1$
$\mathcal{V}_2$
$\mathcal{V}_m$

Fig. 8.5 An illustration of heterogeneous hypergraph neural network for extracting the high-order user behavior representations from the page-view level data

In this application, the multi-classification can be converted into multiple binary classification problems to improve the effect of the model. We use $\mathscr{C} = \{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_{\mathscr{N}}\}$, $\mathscr{N} \in (\mathscr{N}_1, \mathscr{N}_2)$ to denote the collection of the user intentions. Therefore, the original multiple labels are converted into two labels: 0 and 1. For instance, we traverse all the data with l intentions to label 1, and others to label 0. Each classifier is trained using multi-layer perceptron (MLP) and the sigmoid activation function to implement label prediction based on the newly allocated binary label, formulated as follows:

$$\begin{cases} \widehat{\mathscr{Y}_1} = \sigma(\widetilde{\mathbf{X}}_i^{<1>} \Theta_{f1} + b_1) \\ \widehat{\mathscr{Y}_2} = \sigma(\widetilde{\mathbf{X}}_i^{<2>} \Theta_{f2} + b_2), \end{cases} \tag{8.14}$$

where Θ_{f1} and Θ_{f2} are the trainable transformation matrices. b_1 and b_2 are the biases. σ is the activation function. $\widehat{\mathscr{Y}_1}$ and $\widehat{\mathscr{Y}_2}$ denote the prediction of the primary and secondary user intentions, respectively.

To supervise and optimize the trainable parameters, we apply the cross-entropy loss function in the training procedure:

$$\mathscr{L} = \mathbb{CE}(\mathscr{Y}_1, \hat{\mathscr{Y}_1}) + \mathbb{CE}(\mathscr{Y}_2, \hat{\mathscr{Y}_2}), \tag{8.15}$$

where $\mathscr{Y}_1$ and $\mathscr{Y}_2$ denote the ground truth of the primary and secondary user intentions, respectively. When all of the classifiers have been trained completely, each test sample can be predicated to obtain a list of scores for both primary and secondary user intentions.

To summarize, the hierarchical hypergraph learning method can handle large scale hypergraphs with hierarchical labels, which divides a dataset into multiple sub-hypergraphs, and hierarchical aggregation is performed based on hierarchical labels. The hierarchical hypergraph can integrate with the hypergraph neural network to handle millions of data points.

8.4 Summary

This chapter describes two kinds of large scale hypergraph computation methods, i.e., factorization-based hypergraph reduction and hierarchical hypergraph learning. The factorization-based hypergraph reduction is based on the strategy of factorization, which decomposes the large scale hypergraph into low-dimensional embeddings of vertices and hyperedges. It can support the processing of hypergraphs with nearly 10,000 vertices or hyperedges. The hierarchical hypergraph learning is used to analyze hypergraphs with hierarchical labels, which divides a dataset into multiple sub-hypergraphs, and hierarchical aggregation is performed based on hierarchical labels. This method can support millions of data points. We also introduce two applications as examples, i.e., whole-slide image analysis and recommendation, to illustrate the usage of these two algorithms in practice. There are some other large scale hypergraph application scenarios, such as community discovery [19], spectral clustering [20], etc.

References

1. D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, Y. Gao, Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction. IEEE Trans. Pattern Analy. Mach. Intell. (2022), pp. 1–16
2. D. Di, S. Li, J. Zhang, Y. Gao, Ranking-based survival prediction on histopathological whole-slide images, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention* (2020), pp. 428–438
3. J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification. IEEE Trans. Image Process. **21**(7), 3262–3272 (2012)
4. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2020), pp. 2020–2029
5. Y. Liu, F. Wu, Y. Zhang, J. Shao, Y. Zhuang, Tag clustering and refinement on semantic unity graph, in *Proceedings of the IEEE International Conference on Data Mining* (2011), pp. 417–426
6. E. Poirson, C.D. Cunha, A recommender approach based on customer emotions. Expert Syst. Appl. **122**, 281–288 (2019)
7. Y. Zhang, S. Ji, C. Zou, X. Zhao, S. Ying, Y. Gao, Graph learning on millions of data in seconds: label propagation acceleration on graph using data distribution. IEEE Trans. Pattern Analy. Mach. Intell. **45**(2), 1835–1847 (2022)
8. D. Di, J. Zhang, F. Lei, Q. Tian, Y. Gao, Big-hypergraph factorization neural network for survival prediction from whole slide image. IEEE Trans. Image Process. **31**, 1149–1160 (2022)
9. M. Liu, N. Veldt, H. Song, P. Li, D.F. Gleich, Strongly local hypergraph diffusions for clustering and semi-supervised learning, in *Proceedings of the Web Conference* (2021), pp. 2092–2103
10. S. Maleki, U. Agarwal, M. Burtscher, K. Petrank, BiPart: a parallel and deterministic hypergraph partitioner, in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (2021), pp. 161–174
11. S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, P. Sanders, High-quality hypergraph partitioning. ACM J. Exper. Algorith. **27**, 1–39 (2022)
12. C. Mayer, R. Mayer, S. Bhowmik, L. Epple, K. Rothermel, HYPE: massive hypergraph partitioning with neighborhood expansion, in *Proceedings of the IEEE International Conference on Big Data* (2018), pp. 458–467
13. W. Jiang, J. Qi, J.X. Yu, J. Huang, R. Zhang, HyperX: a scalable hypergraph framework. IEEE Trans. Knowl. Data Eng. **31**(5), 909–922 (2019)
14. M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering. Pattern Recogn. **41**(1), 176–190 (2008)
15. H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFM: a factorization-machine based neural network for CTR prediction, in *Proceedings of the International Joint Conference on Artificial Intelligence* (2017), pp. 1725–1731
16. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
17. X. Wang, C. Li, N. Golbandi, M. Bendersky, M. Najork, The lambdaloss framework for ranking metric optimization, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1313–1322

18. J. Devlin, M. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019), pp. 4171–4186
19. P. Chodrow, N. Veldt, A. Benson, Generative hypergraph clustering: from blockmodels to modularity. Sci. Adv. **7**(28), eabh1303 (2021)
20. Y. Yang, S. Deng, J. Lu, Y. Li, Z. Gong, L. Hou U, Z. Hao, GraphLSHC: towards large scale spectral hypergraph clustering. Inf. Sci. **544**, 117–134 (2021)

Chapter 9
Hypergraph Computation for Social Media Analysis

Abstract Social media, such as Twitter and Weibo, have grown rapidly over the past decade. Large numbers of active social media users produce a voluminous amount of data each day, from which important insights can be drawn. Several applications, such as recommender system and sentiment analysis, have been developed to help study the users' intension and portrait. One common challenge faced by these social media applications is how to leverage the complex and multi-modal data on social networks and model the higher-order associations hidden in the data. Hypergraph computation has the huge potential to be used in such analysis. In this chapter, we introduce three typical applications of hypergraph computation, i.e., recommender system, sentiment analysis, and emotion recognition, from which hypergraph computation has shown great value on social media analysis.

9.1 Introduction

With the fast development of information technologies, social media data have increased rapidly. Social media platforms provide new ways to produce and receive content, especially user-generated content. Users can shop, watch movies, and instantly participate in the propagation, interaction, and sharing of news events on the Internet. Rich behavioral data on social media platforms are generated by great numbers of users every day, which support different downstream applications and provide insights for better understanding of users' intension and portrait.

A typical social media analysis application is the so-called recommender system [1]. When listening to music, shopping, watching movies on the Internet, or looking for friends on social network services, users are likely to be drowned in an unprecedented amount of information. This is what we call "information overload." To address this issue, recommender systems have been developed for decades. The main goal of recommender systems is to forecast how users would react to a product by better understanding their preferences based on the user's historical interaction data, user profile, item attributes, context data, and other information. This could help predict whether the users like an item or not. For example, in the movie recommender system, the user profile may contain user ID, age, gender, income,

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_9

marital status, and more. The movie (item) attributes' information may include movie ID, name, genre, director, released time, actors, and more. Interactive data contain the movies which users have seen and even provided comments. The goal of the movie recommender system is to integrate this information to recommend movies that users might like.

Another popular social media analysis application is sentiment analysis [2]. The uses on social media platforms have generated a large amount of opinion data every moment in recent years, which helps to decode and mine users' attitudes on specific topics. Researchers have begun to look at sentiment analysis of users in social media contexts. In economics, stock price fluctuations can be forecast to some extent by analyzing the sentiment of social media users. In politics, social media posts can reflect public opinions. Users' sentiments may also affect their behaviors, for example, emotionally charged people are more likely to forward and repost tweets. Therefore, sentiment analysis plays an important role in social media analysis. However, sentiment analysis is challenging due to the multi-modality and complexity of social media data. For example, a tweet may include text, images, videos, and possibly more. Furthermore, there exist complex correlations among posts in various areas, such as the dimensions of time, location, and user preferences. The interaction among these users further increases the challenges in this task.

In addition to the posts on social media, physiological signals can also be used to analyze the emotion of people [3]. Compared with text, facial expressions, and other data, physiological signals are not easy to disguise and can better reflect real emotions of people. Therefore, emotion recognition based on physiological signals plays an important role in many applications such as clinical diagnosis, which also has played a significant role on social media analysis when these data are available. Physiological signals of different modalities contain complementary information representations of human emotions. It is of great significance to discover and utilize the correlations among these representations to improve the accuracy of emotion recognition.

From the above examples, it can be readily seen that one important issue of social media analysis is how to model complex correlations among data and to make use of the complementary information among multi-modal representations to better understand data. Hypergraphs have been widely used in social media computing in recent years because of their usefulness in complex data modeling. In the following, we discuss three applications of hypergraph computation in social media analysis: recommender system, sentiment analysis, and emotion recognition. In the recommender system, we discuss hypergraph-based collaborative filtering [4] and attribute inference. We then present sentiment prediction [5] and social event detection using hypergraph computation [6] for sentiment analysis. In the third section, we introduce two different hypergraph computation methods of emotion recognition using multi-modal physiological signals [7–9]. Part of the work introduced in this chapter has been published in [4–9].

9.2 Recommender System

In recent years, the Internet has become an integral part of people's daily life—shopping, watching the news, listening to music, etc., on the Internet. However, with the explosion of information, people find it is increasingly difficult to sift through the massive volumes of data on the Internet to access the needed information. For example, when a user wants to watch a movie online and access the movie site, the user is likely to drown in thousands of movies and cannot find the one in mind. This is called the "information overload" issue.

Recommender systems emerge under such circumstances. Recommender systems are a powerful tool for reducing the problem of information overload since they may assist users to find useful information and assist service providers to boost profits. Recommender systems have been used in many online systems, from general platforms including e-commerce, social media, and content sharing to vertical services such as movie, news, and music websites.

The core of the recommender system is to understand the users through their attribute information and historical interactions and then predict whether they would like one item. It is worth noticing that the user-side information, the item-side information, as well as the interaction data, play a vital role in this process. The user-side information, including gender, age, personality, etc., often reflects the users' preference. For example, male users may be more likely to read military and political news, while female users may prefer fashion and entertainment news. The item-side information, such as the category, text description, image, etc., can characterize the attribute of the item. Such attribute information may suggest potential consumer groups. For instance, health supplements may be bought more often by the elderly, while electronics are more likely to be purchased by younger people. Historical interactions also involve potential users' preferences, which are suggested by the assumption that "behavioral similar users may have similar preferences on items." Figure 9.1 shows an example for recommender system based on similar patterns.

We can find from these examples that what recommender systems actually do is to distinguish similar users from different perspectives based on complex, multi-modal given data. Therefore, one key problem is how to model and learn the complex relationship between users and items. Recently, hypergraph computation has attracted much attention and has been applied to recommender systems to help solve this problem. The hypergraph can naturally integrate the user-side information, item-side information, as well as the interaction data, thanks to flexible hyperedges and especially hyperedge groups. Therefore, similar users/items can be connected in different areas. In this section, we discuss two examples of applying hypergraph computation in recommender system, i.e., collaborative filtering and attribute inference.

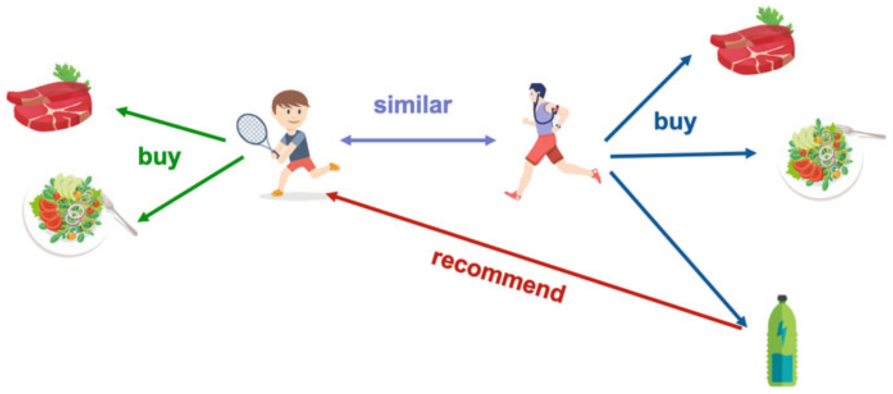


Fig. 9.1 An example of recommender system based on similar patterns

9.2.1 Collaborative Filtering

In the past decades, collaborative filtering (CF), a crucial, popular recommendation technique, has been extensively used in various recommender systems. The fundamental assumption of CF is that consumers who engage in similar behaviors, for example, reading the same kind of news frequently, are likely to have similar tastes for items, such as games, movies, and commodities. A common CF-based solution goes through the following two steps: first, it uses historical interactions to identify similar users and items; and second, it makes suggestions for users based on the information acquired in the last step.

Since people and things have topological links that the network can describe, graph-based CF approaches have attracted a lot of interest in recent years. Although graph-based CF approaches have been explored for a long time and produced respectable performance, there are still certain restrictions. First, the high-order correlations in the user–item network are modeled and utilized insufficiently. For example, CF methods hope to find a group of behavior-similar users. Such associations between users are group-level (beyond pairwise) and cannot be well-captured by the graph structure since only pairwise correlations can be modeled in a graph. Second, when users and things are represented by a graph in graph-based approaches, there are no fundamental distinctions between them. When an item has many users connected to it, it is a popular item. In contrast, being connected to a variety of items does not necessarily mean that a user is well-liked.

Under these circumstances, more adaptable and appropriate user and item modeling is required. Thanks to its adaptable hyperedges, the hypergraph structure, as opposed to the graph structure, offers a more natural approach for representing such high-order and intricate relationships. In this subsection, we present a dual channel hypergraph collaborative filtering (DHCF) framework [4] to solve the

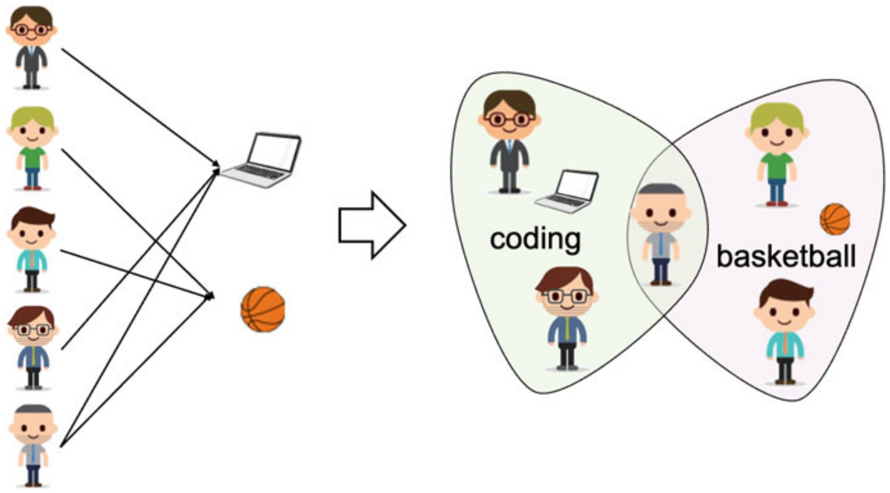


Fig. 9.2 An example of hypergraph modeling for user–item network

aforementioned problems. In the following, we introduce how to model the user–item interactions and learn the high-order connectivity with dual hypergraphs.

(1) Hypergraph Modeling of High-Order Connectivity
Given a user–item network, the high-order connectivity is captured by some self-defined association rules. Based on these rules, several hyperedge groups can be constructed, which can capture higher-order correlations rather than pairwise relationships, e.g., by linking users who behave similarly but without direct connections. For example, we can connect the users who have purchased the same item with a hyperedge, as shown in Fig. 9.2. In addition to the interactions that are apparently visible in the observed data, these rules may also be thought of as a high-order perspective to describe the otherwise raw data. Here we introduce a way to capture the high-order connectivity with hypergraphs for users and items, separately.

User Hypergraph Construction We first define the k-order neighbors for items. If there is a path between $item_a$ and $item_b$ that consists of a series of adjacent vertices and has fewer users than k, then we can say $item_a$ ($item_b$) is $item_b$ ($item_a$)'s k-order reachable neighbor in the user–item network.

We then define the k-order neighbor users for items. If there are direct paths between $user_a$ and $item_a$ and $item_a$ is $itemb$'s k-order neighbor, then $usera$ is k-order neighbor for $itemb$.

The $B_u^k(i)$ symbol represents the set of k-order $B_u^k(i)$ users for item i. A hypergraph can be defined mathematically as a set family where each set indicates a hyperedge. As a result, a hypergraph may be built using the k-order neighbor users set of an object. By using the above definitions, the corresponding hyperedge group may be constructed as follows:

$$\mathscr{E}_{B_u^k} = \{B_u^k(i) \mid i \in I\}. \tag{9.1}$$

The k-order accessible matrix of items is denoted by $\mathbf{A}_i^k \in \{0, 1\}^{M \times M}$, which can be written as follows:

$$\mathbf{A}_i^k = \text{Min}(1, \text{power}(\mathbf{H}^\top \cdot \mathbf{H}, k)), \tag{9.2}$$

where the function $\text{pow}(M, k)$ determines the k power of the matrix M in question. The incidence matrix of the user–item network is represented by $\mathbf{H} \in \{0, 1\}^{N \times M}$, where N and M are the numbers of users and items, respectively. Then, the incidence matrix of the hyperedge group has the following form:

$$\mathbf{H}_{B_u^k} = \mathbf{H} \cdot \mathbf{A}_i^{k-1}. \tag{9.3}$$

The hypergraph $\mathscr{G}_u$ can capture the overall high-order correlations among users by fusing multiple hyperedge groups that are constructed via k-order reachable rule. Therefore, the $\mathbf{H}_u$ can be written as

$$\mathbf{H}_u = f\left(\mathscr{E}_{B_u^{k_1}}, \mathscr{E}_{B_u^{k_2}}, \ldots, \mathscr{E}_{B_u^{k_a}}\right) = \underbrace{\mathbf{H}_{B_u^{k_1}} || \mathbf{H}_{B_u^{k_2}} || \ldots || \mathbf{H}_{B_u^{k_a}}}_{a}, \tag{9.4}$$

where $\cdot||\cdot$ is the concatenation operation, which is an example of hyperedge groups fusion function $f(\cdot)$.

Item Hypergraph Construction Here the high-order connectivity for items is defined in a similar way. The k-order accessible matrix of user $\mathbf{A}_u^k \in \{0, 1\}^{N \times N}$ is defined as

$$\mathbf{A}_u^k = \text{Min}(1, \text{power}(\mathbf{H} \cdot \mathbf{H}^\top, k)). \tag{9.5}$$

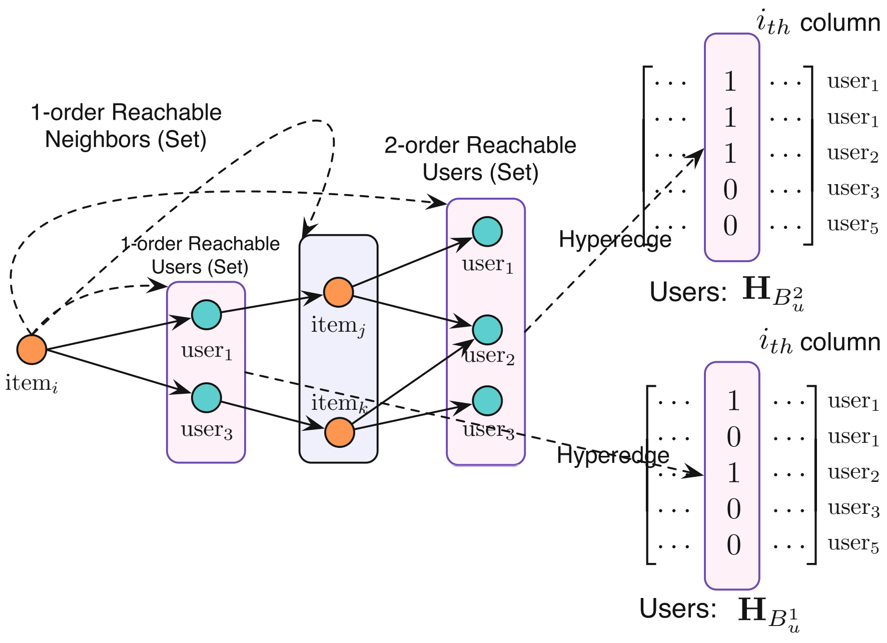
The incidence matrix $\mathbf{H}_{B_i^k} \in \{0, 1\}^{M \times N}$ can be written as

$$\mathbf{H}_{B_i^k} = \mathbf{H}^\top \cdot \mathbf{A}_u^{k-1}. \tag{9.6}$$

By assuming that we have b hyperedge groups, the item's hypergraph incidence matrices $\mathbf{H}_i$ are similarly formulated as follows:

$$\mathbf{H}_i = f\left(\mathscr{E}_{B_i^{k_1}}, \mathscr{E}_{B_i^{k_2}}, \ldots, \mathscr{E}_{B_i^{k_b}}\right) = \underbrace{\mathbf{H}_{B_i^{k_1}} || \mathbf{H}_{B_i^{k_2}} || \ldots || \mathbf{H}_{B_i^{k_b}}}_{b}. \tag{9.7}$$

In this way, the high-order connectivity for both users and items is captured with a hypergraph. Figure 9.3 gives one example of the defined high-order connectivity


1-order Reachable Neighbors (Set)
2-order Reachable Users (Set)
1-order Reachable Users (Set)
item$_i$
user$_1$
user$_3$
item$_j$
item$_k$
user$_1$
user$_2$
user$_3$
Hyperedge
Hyperedge
i_{th} column
Users: $\mathbf{H}_{B_u^2}$
i_{th} column
Users: $\mathbf{H}_{B_u^1}$

$$\begin{array}{c|c}
\begin{bmatrix} \cdots & 1 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 0 & \cdots \end{bmatrix} & \begin{array}{l} \text{user}_1 \\ \text{user}_1 \\ \text{user}_2 \\ \text{user}_3 \\ \text{user}_5 \end{array}
\end{array}$$

$$\begin{array}{c|c}
\begin{bmatrix} \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 0 & \cdots \end{bmatrix} & \begin{array}{l} \text{user}_1 \\ \text{user}_1 \\ \text{user}_2 \\ \text{user}_3 \\ \text{user}_5 \end{array}
\end{array}$$

Fig. 9.3 The illustration of high-order connectivity for users

for users [4]. Subsequently, two embedding look-up tables ($\mathbf{E}_u = [\mathbf{e}_{u_1}, \ldots, \mathbf{e}_{u_N}]$ and $\mathbf{E}_i = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_M}]$) are constructed to describe both users and items, which, together with the hypergraph structure, are prepared for later learning.

(2) High-Order Information Passing

When mixed high-order correlations have been obtained, the neighboring messages are aggregated using the high-order information passing technique, which can be expressed as

$$\begin{cases} M_u = \mathrm{HyConv}(E_u, H_u) \\ M_i = \mathrm{HyConv}(E_i, H_i) \end{cases}, \tag{9.8}$$

where HyConv$(\cdot, \cdot)$ can be any hypergraph convolution operation as that specified in HGNN (HGNNConv for short). Through information passing from high-order neighbors, the complex correlations between vertices have been encoded to the aggregated messages of users (M_u') and items (M_i'), respectively. It should be noted that the high-order neighbor mentioned here is not a fixed concept of the direct interactions in user–item network, but an abstract description that can link the similar users/items in latent behavior–attribute space.

To provide an example of high-order information passing, we present the jump hypergraph convolution (JHyConv) in this part. Inspired by some previous work [10], the JHyConv operator creates the learned representations by concatenating a vertex's current representation with its aggregated neighborhood representation. The JHyConv is written as

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\Theta^{(l)} + \mathbf{X}^{(l)}\right), \tag{9.9}$$

where all symbols follow existing notations consistently.

In contrast to conventional HGNNConv, the jump hypergraph convolution enables the model to take into account both its representation and aggregated high-order representations. The messages M_u and M_i are then used to jointly update E_u and E_i.

(3) Joint Information Updating

The goal of the joint information updating is to extract information that is discriminatory for users and items, which is formulated by

$$\begin{cases} E_u' = \mathrm{JMU}(M_u, M_i) \\ E_i' = \mathrm{JMU}(M_i, M_u) \end{cases}, \tag{9.10}$$

where any learnable feed-forward neural network may be used for $\mathrm{JMU}(\cdot,\cdot)$. Updated embeddings for users and items are termed as E_u' and E_i', respectively. Here, a shared fully connected layer is applied.

(4) Overall DHCF Layer

The two stages of DHCF framework are illustrated in Figs. 9.4 and 9.5, respectively. The high-order information passing and joint information updating constitute an integrated DHCF layer, which, thanks to its powerful hypergraph structure, can directly model and encode the high-order connectivity.

With the specified HyConv and JMU, a DHCF configuration can be formulated as follows:

$$\begin{cases} f(\dots) = \cdot||\cdot \\ \mathrm{HyConv}(\cdot,\cdot) = \mathrm{JHyConv}(\cdot,\cdot) \\ \mathrm{JMU}(\cdot,\cdot) = \mathrm{MLP}_1(\cdot) \end{cases}, \tag{9.11}$$

where $\mathrm{MLP}_1(\cdot)$ is a fully connected layer, Θ is trainable parameters, and $\cdot||\cdot$ is the concatenation operation.

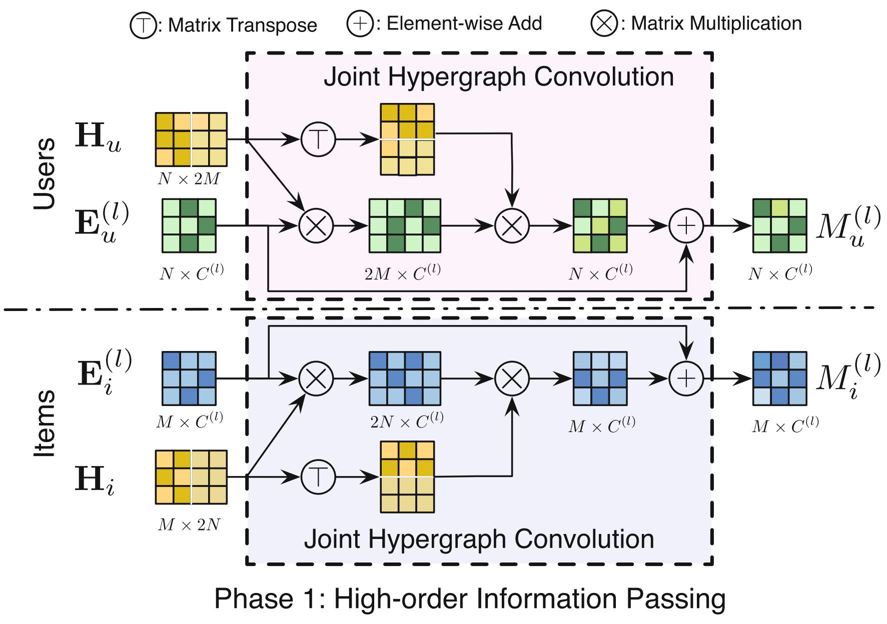


Fig. 9.4 The first stage of the DHCF framework

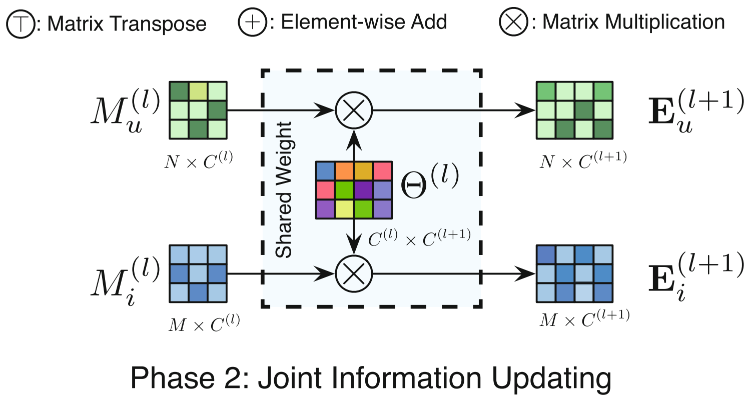


Fig. 9.5 The second stage of the DHCF framework

The matrix form of the embedding propagation on hypergraph can be written as follows:

$$\begin{cases} \left.\begin{aligned} \mathbf{H}_u &= \mathbf{H} || \left(\mathbf{H}(\mathbf{H}^\top \mathbf{H})\right) \\ \mathbf{H}_i &= \mathbf{H}^\top || \left(\mathbf{H}^\top (\mathbf{H}\mathbf{H}^\top)\right) \end{aligned}\right\} \text{hypergraph setup} \\ \left.\begin{aligned} \mathbf{M}_u^{(l)} &= \mathbf{D}_{u_v}^{-1/2} \mathbf{H}_u \mathbf{D}_{u_e}^{-1} \mathbf{H}_u^\top \mathbf{D}_{u_v}^{-1/2} \mathbf{E}_u^{(l)} + \mathbf{E}_u^{(l)} \\ \mathbf{M}_i^{(l)} &= \mathbf{D}_{i_v}^{-1/2} \mathbf{H}_i \mathbf{D}_{i_e}^{-1} \mathbf{H}_i^\top \mathbf{D}_{i_v}^{-1/2} \mathbf{E}_i^{(l)} + \mathbf{E}_i^{(l)} \end{aligned}\right\} \text{Phase 1} \\ \left.\begin{aligned} \mathbf{E}_u^{(l+1)} &= \sigma(\mathbf{M}_u^{(l)} \Theta^{(l)}) \\ \mathbf{E}_i^{(l+1)} &= \sigma(\mathbf{M}_i^{(l)} \Theta^{(l)}) \end{aligned}\right\} \text{Phase 2} \end{cases}, \tag{9.12}$$

where $\mathbf{D}_{u_v}$, $\mathbf{D}_{u_e}$ and $\mathbf{D}_{i_v}$, $\mathbf{D}_{i_e}$ are vertex degree and hyperedge degree matrices of user hypergraph $\mathbf{H}_u$ and item hypergraph $\mathbf{H}_i$, respectively. $\mathbf{E}_u^{(l)}$ and $\mathbf{E}_i^{(l)}$ are the inputs for layer l, while $\mathbf{E}_u^{(l+1)}$ and $\mathbf{E}_i^{(l+1)}$ are the outputs for layer l.

With the introduced framework, the collaborative signals in the user–item network are modeled and captured, thus achieving better representation.

9.2.2 Attribute Inference

A CF-based recommender system has the cold-start problem when there is a lack of historical behavior data of users, making it challenging to personalize recommendations to individual users. Making use of user and item attribute data is a potential answer to this issue. The attribute information of users usually includes gender, age, occupation, etc. The attribute information of an item can be the genre of a movie or music, or the classification of an item on an e-commerce website, etc. According to the principle of CF, similar users will choose similar items, and the attribute information can then be used to establish the similarity between users or items. The addition of attribute information can build up the association between users and items in the absence of user historical behaviors, which can well alleviate the cold-start problem. In other words, attribute information can assist in collaborative filtering.

However, attribute information is often insufficient, as many people are reluctant to provide true personal information. Therefore, attribute inference becomes an important task. It is mutually reinforcing with the recommendation task, as high-quality attributes can help better with collaborative filtering, while more accurate user behavior can also help infer attributes of users and items.

In this section, we discuss a framework of multi-task learning that combines the attribute inference task with the recommendation task. The framework first utilizes multi-channel hypergraph CF for representation learning, performs two downstream

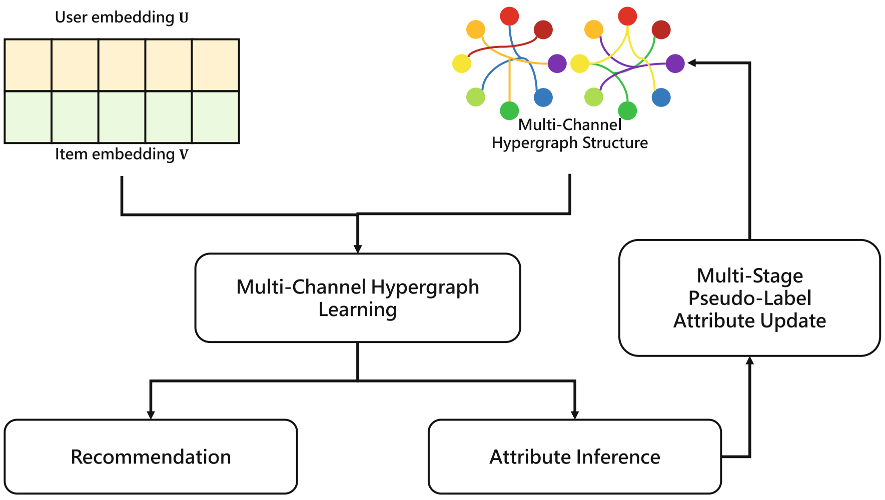

User embedding U
Item embedding V
Multi-Channel Hypergraph Structure
Multi-Channel Hypergraph Learning
Multi-Stage Pseudo-Label Attribute Update
Recommendation
Attribute Inference

Fig. 9.6 The pipeline of multi-channel hypergraph neural networks for recommendation and attribute inference

tasks simultaneously, and lastly optimizes the model by downstream tasks. The pipeline of the framework is presented in Fig. 9.6.

(1) Multi-Channel Hypergraph Collaborative Filtering

Multi-Channel Hypergraph Construction In order to model the higher-order interactions and attributes between users and items, two hypergraphs, named Interaction Hypergraph and Attribute Hypergraph, are constructed and denoted as I and A for simplicity.

The structure of I is generated through the interaction between users and items. The implicit interaction matrix is represented as $\mathbf{R} \in \mathbb{R}^{n_u \times n_v}$, where n_u and n_v denote user and item numbers, respectively. With the k-order reachable rule introduced in the previous subsection, we generate the hyperedges by connecting the user's and item's 1-order reachable users and items. The incidence matrix can be expressed as

$$\begin{aligned} \mathbf{H}_I^u(i,j) &= \begin{cases} 1 & \text{user}_i \text{ interacted with item}_j \\ 0 & \text{otherwise.} \end{cases} \\ \mathbf{H}_I^v(i,j) &= \begin{cases} 1 & \text{item}_i \text{ interacted with user}_j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{9.13}$$

It is obvious that $\mathbf{H}_I^u = \mathbf{R}$ and $\mathbf{H}_I^v = \mathbf{R}^\top$.

The structure of A is generated through the attribute information of users and items. The user and item binary attribute matrices are denoted by $\mathbf{X} \in \mathbb{R}^{n_u \times n_p}$ and $\mathbf{Y} \in \mathbb{R}^{n_v \times n_q}$, where n_p and n_q denote user and item attribute numbers, respectively.

Attributes represent hyperedges, and vertices with the same attributes are connected by hyperedges. The incidence matrix can be formulated as

$$\begin{aligned}\mathbf{H}_A^u(i,j) &= \begin{cases} 1 & \text{user}_i \text{ has attribute}_j \\ 0 & \text{otherwise.} \end{cases} \\ \mathbf{H}_I^v(i,j) &= \begin{cases} 1 & \text{item}_i \text{ has attribute}_j \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{9.14}$$

Here we can have $\mathbf{H}_A^u = \mathbf{X}$ and $\mathbf{H}_I^v = \mathbf{Y}$.

Multi-Channel Hypergraph Learning When the hypergraph structure has been generated, the multi-channel hypergraph convolution is performed separately. It can be written as

$$\mathbf{X}^{(k+1)} = \sigma(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(k)}), \tag{9.15}$$

where $\mathbf{X}^{(k)}$ denotes the vertex embeddings after k-layer convolution, and it should be replaced by $\mathbf{U}_c^{(k)}$ and $\mathbf{V}_c^{(k)}$ for user and item embeddings on channel $c \in \{A, I\}$ in our case. To bypass the over-smoothing problem, the results obtained from K-layer propagation are averaged as below:

$$\mathbf{U}_c^* = \frac{1}{K+1}\sum_{l=0}^{K}\mathbf{U}_c^{(k)}, \mathbf{V}_c^* = \frac{1}{K+1}\sum_{l=0}^{K}\mathbf{V}_c^{(k)}. \tag{9.16}$$

Moreover, to aggregate information from different channels, a channel attention mechanism is leveraged to generate the comprehensive user and item embeddings. It is defined as

$$\boldsymbol{\alpha}_u^c = f_a(\mathbf{U}_c^*) = \frac{\exp(\mathbf{a}_u^\top \cdot \mathbf{W}_a^{c,u}\mathbf{U}_c^*)}{\sum_c \exp(\mathbf{a}_u^\top \cdot \mathbf{W}_a^{c,u}\mathbf{U}_c^*)}, \tag{9.17}$$

$$\boldsymbol{\alpha}_v^c = f_a(\mathbf{V}_c^*) = \frac{\exp(\mathbf{a}_v^\top \cdot \mathbf{W}_a^{c,v}\mathbf{V}_c^*)}{\sum_c \exp(\mathbf{a}_v^\top \cdot \mathbf{W}_a^{c,v}\mathbf{V}_c^*)}, \tag{9.18}$$

where $\mathbf{W}_a \in \mathbb{R}^{d\times d}$ is the trainable parameter, and d denotes the embedding dimension. The comprehensive representations can be formulated as

$$\mathbf{U}^* = \sum_c \alpha_u^c \mathbf{U}_c^*, \mathbf{V}^* = \sum_c \alpha_v^c \mathbf{V}_c^*, \tag{9.19}$$

where $c \in \{A_u, I_u, A_v, I_v\}$.

The graph convolution is leveraged in order to further exploit the interaction data between users and items. It can be formulated as

$$\begin{pmatrix} \mathbf{U}^{*(j+1)} \\ \mathbf{V}^{*(j+1)} \end{pmatrix} = \mathbf{D}^{-1/2} \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{pmatrix} \mathbf{D}^{-1/2} \begin{pmatrix} \mathbf{U}^{*(j)} \\ \mathbf{V}^{*(j)} \end{pmatrix}, \tag{9.20}$$

$$\hat{\mathbf{U}} = \frac{1}{J+1} \sum_{l=0}^{J} \mathbf{U}^{*(j)}, \hat{\mathbf{V}} = \frac{1}{J+1} \sum_{l=0}^{J} \mathbf{V}^{*(j)}, \tag{9.21}$$

where J is the number of graph convolution layers.

(2) Recommendation Task and Attribute Inference Task
Following up the representation learning through multi-channel hypergraph collaborative filtering, the two downstream tasks can be performed simultaneously.

First, based on the idea of matrix factorization, the user and item interaction can be predicted as

$$\hat{\mathbf{R}} = \hat{\mathbf{U}}\hat{\mathbf{V}}^\top. \tag{9.22}$$

Next, we consider the nature of the relationship between attributes and vertices, and a subtle method of attribute inference is discussed. Also inspired by matrix factorization, the attribute matrix can be regarded as the product of two low-rank matrices. It can be formulated as

$$\hat{\mathbf{X}} = \hat{\mathbf{U}}\mathbf{P}^\top, \hat{\mathbf{Y}} = \hat{\mathbf{V}}\mathbf{Q}^\top, \tag{9.23}$$

where $\mathbf{P} \in \mathbb{R}^{n_p \times d}$ and $\mathbf{Q} \in \mathbb{R}^{n_q \times d}$ are the user and item attribute representations. The use of matrix factorization for attribute inference is very reasonable because attributes are influenced by the properties of vertices and the properties of attribute themselves; one cannot be presented without the other. In conclusion, the benefit of processing two distinct tasks concurrently with this method is that it permits information sharing while allowing a high degree of autonomy between the two training activities.

(3) Joint Optimization
A paired loss called Bayesian Personalized Ranking (BPR) promotes observable behavior predictions to outperform unobserved ones, and it is utilized to optimize the recommendation task. It can be written as

$$\mathscr{L}_r = \sum_{j \in \mathscr{I}(i), k \notin \mathscr{I}(i)} -log\sigma(\hat{r}_{i,j} - \hat{r}_{i,k}) + \lambda \|\Phi_r\|_2^2, \tag{9.24}$$

where Φ_r represents the model parameters and $\hat{\mathbf{r}}_{i,j} = \mathbf{u}_i^\top \mathbf{v}_j$ represents the probability that user$_i$ is interested in item$_j$. The sigmoid function is denoted as $\sigma(\cdot)$.

Next, the attribute inference task can be regarded as an attribute categories classification problem. The cross-entropy loss is then leveraged for optimizing the attribute inference task. It can be written as

$$\begin{aligned}
\mathscr{L}_i^u &= -\frac{1}{n_u}\sum_i\sum_{j=1}^{n_p} x_{ij} log(\hat{x}_{ij}),\\
\mathscr{L}_i^v &= -\frac{1}{n_v}\sum_i\sum_{j=1}^{n_q} y_{ij} log(\hat{y}_{ij}),\\
\mathscr{L}_i &= \mathscr{L}_i^u + \mathscr{L}_i^v,
\end{aligned} \tag{9.25}$$

where $\hat{\mathbf{x}}_{i,j} = \mathbf{u}_i^\top \mathbf{p}_j$ is the inference score of user$_i$ on user attribute$_j$, and $\hat{\mathbf{y}}_{i,j} = \mathbf{v}_i^\top \mathbf{q}_j$ is the inference score of item$_i$ on item attribute$_j$.

Finally, the sum of the losses from the two tasks is the overall loss. It can be written as

$$\mathscr{L} = \mathscr{L}_r + \gamma \cdot \mathscr{L}_i, \tag{9.26}$$

where γ is the hyperparameter for balancing the two different losses.

Although in this section we only discuss two instances, i.e., collaborative filtering and attribute inference, applications of hypergraph computation in recommender system do not end there. In collaborative filtering, only the historical interaction data are utilized, and the hypergraph is constructed upon the similarity of users/items in behavior space. In attribute inference, the attribute information of users and items is further utilized to solve the cold-start problem. In this case, the hypergraph is constructed based on both behaviors and attributes. In addition to the behavior and attribute data, the context data, such as the time, location, weather, etc., can also be integrated, and hypergraph can also be applied to model the complex correlations among these data. Also, the user–item network sometimes can be multiplex, that is, there may exist various kinds of interactions between users and items, e.g., a user may view, click, and purchase an item. How to adopt the hypergraph to model such multiplex connections also remains to be explored.

9.3 Sentiment Analysis

The emergence of Twitter and Sina Weibo has given social media users a place to share their thoughts and emotions about particular occurrences. At the same time, this information is rapidly and widely disseminated throughout social networks. Therefore, how to analyze the information in social media becomes an important issue.

First, sentiment dimension, event monitoring, social network analysis, and business advice all have numerous potential applications for microblog sentiment research. By analyzing the sentiment of massive data, we can get the emotional attitude of netizens toward relevant events. Second, real-time multimedia data may travel quickly and widely throughout the social network in terms of the temporal dimension, having a significant impact on society. Therefore, efficient real-time temporal detection can help government organizations with macroeconomic control and marketing management at huge corporations.

There are multi-modal data among Twitter data, including text, images, emojis, videos, etc. The higher-order association between different modalities can be well modeled by hypergraphs to extract sentiment information. In the following subsections, we provide two examples to analyze the sentiment of microblog data in two dimensions using hypergraphs, respectively, [5, 6].

9.3.1 Sentiment Prediction

Predicting multi-modal sentiment of tweets is not an easy task. Most sentiment analysis models focus on textual or visual channels only. However, in human emotional perception, different moods have their own characteristics so that sentiment analysis should be based on multiple perspectives. Even with multi-channel data, it is uncertain whether the emotions of different channels are related. Moreover, there are cases where some channels are missing. To address these problems, a two-layer multi-modal hypergraph learning framework [5] is introduced to create a multi-modal sentiment prediction.

This framework's objective is to forecast the sentiment of provided multi-modal microblog data (e.g., a Weibo tweet) that include text, visuals, and emoticons. The bag-of-textual-words feature $F_i^{botw} = \{w_i^t, \ldots, w_{m_t}^t\}$ is extracted for textual modality. The visual modality feature $F_i^{bovw} = \{w_i^v, \ldots, w_{m_v}^v\}$ is extracted from the i-th image. Furthermore, an emoticon dictionary is defined for the emotical modality, which forms the bag-of-emoticon-words feature $F_i^{boew} = \{w_i^e, \ldots, w_{m_e}^e\}$. A corresponding sentiment score s_k^t, s_k^v, s_k^e is assigned to w_k^t, w_k^v, w_k^e, respectively. Consequently, the tweet x_i can be denoted as $\{F_i^{botw}, F_i^{bovw}, F_i^{boew}\}$. Through investigating F_i^{botw}, F_i^{bovw} and F_i^{boew} simultaneously, the sentiment of x_i can be predicted.

(1) Multi-Modal Hypergraph Learning

To create the incidence matrix of the hypergraph, the correlation between each tweet and the "centroid" tweets of various modalities is first computed. Each tweet is treated as a vertex and the hyperedges connecting its k nearest neighbors in each modality. It is important to note that each vertex can be thought of as a centroid. The incidence matrix can be defined as

$$\mathbf{H}(v_i, e_j) = \begin{cases} s(j,i) \text{ if } v_i \in e_j \\ 0 \qquad \text{otherwise} \end{cases}, \tag{9.27}$$

where $s(j,i) = \exp(-\frac{dist(i,j)^2}{\sigma \hat{d}^2})$ is the correlation between v_i and e_j. $dist(i,j)$ is the distance in Euclidean terms between v_i and the centroid vertex of e_j. $\hat{d}$ is the average pairwise distance for the corresponding modality, and the parameter σ is empirically set to modify the normalization of the tweet relevance. Each hyperedge's weight starts out at 1.

In multi-modal hypergraph learning (MHG) [5], guided inference is used to perform hypergraph learning. It calculates the relevance scores of tweets with varying attitudes by iteratively updating the relevance score vector **f** and the hyperedge weights **W**. It accomplishes the aforementioned objectives by optimizing the loss functions:

$$\begin{gathered}\arg\min_{\mathbf{f},\mathbf{W}}\{\Omega(\mathbf{f}) + \lambda \mathscr{R}_{emp}(\mathbf{f}) + \mu \sum_{i=1}^{n_e} w_i^2\},\\ \text{s.t. } \textstyle\sum_{i=1}^{n_e} w_i = 1,\end{gathered} \tag{9.28}$$

where **f** is the learned relevance score, $\Omega(\mathbf{f})$ is a regularizer built on the Hypergraph Normalized Laplacian, $\mathscr{R}_{emp}(\mathbf{f}) = \|f - y\|^2$ denotes the empirical loss, and $\sum_{i=1}^{n_e} w_i^2$ is the regularizer. $\Omega(\mathbf{f})$ can be formulated as

$$\Omega(\mathbf{f}) = \frac{1}{2}\sum^{e\in\mathscr{E}}\sum_{u,v\in\mathscr{V}} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \times \left(\frac{\mathbf{f}(u)}{\sqrt{d(u)}} - \frac{\mathbf{f}(v)}{\sqrt{d(v)}}\right)^2, \tag{9.29}$$

where $d(v) = \sum_{e\in\mathscr{E}} \mathbf{W}(e)h(v,e)$ denotes vertex degree and $\delta(e) = \sum_{v\in\mathscr{V}} h(v,e)$ denotes hyperedge degree. Let $\Theta = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$ and $\Delta = \mathbf{I} - \Theta$ be the hypergraph Laplacian. The diagonal matrices of $d(v)$ and $\delta(e)$ are represented as $\mathbf{D}_v$ and $\mathbf{D}_e$, respectively. The normalized cost function can be expressed as

$$\Omega(\mathbf{f}) = \mathbf{f}^\top \Delta \mathbf{f}. \tag{9.30}$$

The two parameters **W** and **f** are optimized iteratively using the following two functions:

$$\arg\min_{\mathbf{f}} \Phi(\mathbf{f}) = \arg\min_{\mathbf{f}}\{\mathbf{f}^\top \Delta \mathbf{f} + \lambda \|f - y\|^2\}, \tag{9.31}$$

$$\begin{gathered}\arg\min_{\mathbf{W}} \Phi(\mathbf{W}) = \arg\min_{\mathbf{W}}\{\mathbf{f}^\top \Delta \mathbf{f} + \mu \sum_{i=1}^{n_e} w_i^2\},\\ \text{s.t. } \sum_{i=1}^{n_e} w_i = 1.\end{gathered} \tag{9.32}$$

As shown above, MHG simulates the sample–sample relation for the purpose of hypergraph construction. The properties of modalities and their relevance to one another, however, are not fully utilized.

(2) Dual-Layer Multi-Modal Hypergraph Learning
Dual-layer multi-modal hypergraph learning is composed of 2 hypergraph layers, $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathbf{W})$ for tweet-level hypergraph and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathbf{M})$ for feature-level hypergraph, respectively.

To allow multi-modal features to be adopted more explicitly and to directly construct multi-modal hypergraphs for modal correlation, each hypergraph layer of dual-layer multi-modal hypergraph learning uses relations between vertex and hyperedge to represent sample features or relations between features and samples, rather than relations between samples in MHG.

The sentiment label vector of tweets and the sentiment label vector of multi-modal sentiment words are denoted, respectively, by y and t in distinct hypergraph layers. Therefore, in two hypergraph layers, $\mathbf{f}$ and $\mathbf{g}$ started out originally as vectors representing the relevance scores of tweets and multi-modal features/words, respectively. It is said that $\mathbf{M}$ can be regarded as the confidence ratings of the sentiment labels $\mathbf{y}$, which correspond to $\mathbf{f}$ in the hypergraph of tweet level. Two hypergraph layers are connected, and the multi-modal relevance of features is transferred to the tweet-level hypergraph in order to help predict tweet sentiment.

The probabilistic incidence matrix of a hypergraph is written as

$$\mathbf{H}_*(v_i, e_j) = \begin{cases} 1 \text{ if } v_i \in e_j \\ 0 \text{ otherwise} \end{cases}, \tag{9.33}$$

where $*$ denotes either 1 or 2, and the same applies below.

The following loss function can be optimized to represent the learning process:

$$\arg\min_{\mathbf{f},\mathbf{g},\mathbf{W},\mathbf{M}} \{\Omega_1(\mathbf{f}) + \lambda_1 \mathcal{R}_{emp1}(\mathbf{f}) + \mu_1 \sum_i^{n_{e1}} \mathbf{W}_i^2 + \Omega_2(\mathbf{g}) + \lambda_2 \mathcal{R}_{emp2}(\mathbf{g}) + \mu_2 \sum_i^{n_{e2}} \mathbf{M}_i^2\},$$
$$\text{s.t.} \begin{cases} \sum_{i=1}^{n_e 1} \mathbf{W}_i = 1 \\ \sum_{i=1}^{n_e 2} \mathbf{M}_i = 1 \end{cases}, \tag{9.34}$$

where $\Omega_1(\mathbf{f})$ and $\Omega_2(\mathbf{g})$ are regularizers based on the normalized Laplacian on hypergraph, $\mathcal{R}_{emp1}(\mathbf{f}) = \|\mathbf{f} - \mathbf{y} \circ \mathbf{M}\|^2$ and $\mathcal{R}_{emp2}(\mathbf{g}) = \|\mathbf{g} - \mathbf{t}\|^2$ are the empirical losses, and $\sum_{i=1}^{n_e 1} \mathbf{W}_i$ and $\sum_{i=1}^{n_e 2} \mathbf{M}_i$ are the L_2 regularizers on the hyperedge weights. In this scenario, empirical loss is represented as $\mathscr{R}_{emp1}(\mathbf{f}) = \|\mathbf{f} - \mathbf{y} \circ \mathbf{M}\|^2$ and $\mathscr{R}_{emp2}(\mathbf{g}) = \|\mathbf{g} - \mathbf{t}\|^2$, and $\sum_{i=1}^{n_e1} \mathbf{W}_i$ and $\sum_{i=1}^{n_e2} \mathbf{M}_i$ are the L_2 regularizers on the hyperedge weights. The normalized Laplacian on hypergraph regularizers $\Omega_1(\mathbf{f})$ and $\Omega_2(\mathbf{g})$ are further described as follows:

$$\begin{aligned} \Omega_1(\mathbf{f}) &= \mathbf{f}^\top(\mathbf{I} - \mathbf{D}_{v1}^{-1/2}\mathbf{H}_1\mathbf{W}\mathbf{D}_{e1}^{-1}\mathbf{H}_1^\top\mathbf{D}_{v1}^{-1/2})\mathbf{f}, \\ \Omega_2(\mathbf{g}) &= \mathbf{g}^\top(\mathbf{I} - \mathbf{D}_{v2}^{-1/2}\mathbf{H}_2\mathbf{M}\mathbf{D}_{e2}^{-1}\mathbf{H}_2^\top\mathbf{D}_{v2}^{-1/2})\mathbf{g}. \end{aligned} \tag{9.35}$$

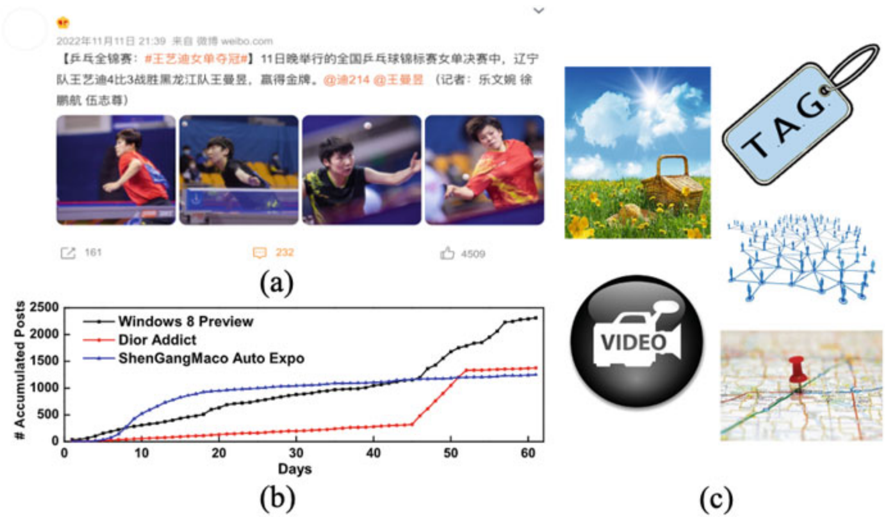
The loss function then has the following form in terms of **f**, **W**, **g**, and **M**:

$$\begin{aligned} \mathscr{L}(\mathbf{f}, \mathbf{W}, \mathbf{g}, \mathbf{M}) =& \Omega_1(\mathbf{f}) + \lambda_1\mathscr{R}_{emp1}(\mathbf{f}) + \mu_1 \sum_{i}^{n_{e1}} \mathbf{W}_i^2 \\ &+ \Omega_2(\mathbf{g}) + \lambda_2\mathscr{R}_{emp2}(\mathbf{g}) + \mu_2 \sum_{i}^{n_{e2}} \mathbf{M}_i^2 \\ &+ \eta_1 \left(\sum_{i=1}^{n_e1} \mathbf{W}_i - 1\right) + \eta_2 \left(\sum_{i=1}^{n_e2} \mathbf{M}_i - 1\right). \end{aligned} \tag{9.36}$$

To summarize, we introduced a two-layer multi-modal hypergraph learning framework that models correlations among visual, textual, and emoji modalities while allowing input from missing modalities to achieve document sentiment prediction for multi-modal tweets.

9.3.2 Social Event Detection

The expanding visual content of microblogs and the inter-connectedness of diverse data have received less attention from existing methods, while social event identification as a crucial social media analysis problem has received much attention in recent years. Figure 9.7 presents an example of real-time social event. In social media platforms, event detection is a difficult issue due to the distinctiveness of social media data for the following reasons. First, it is required to explore a set of posts that are significantly related to one another and discuss a common issue because social media postings are noisy and do not include enough substantial material to provide full information. Second, social media posts can come in a variety of multimedia formats and include information such as images, timestamps, locations, user preferences, and social connections in addition to text. Finally, social posts are real time, and these large scale, real-time data make social events difficult to detect. Hypergraph, due to its natural structural advantages, can establish higher-order correlations between data of different posts, different modalities, and different times, thus enabling real-time event detection. In this subsection, we


2022年11月11日 21:39 来自 微博 weibo.com

【乒乓全锦赛：#王艺迪女单夺冠#】11日晚举行的全国乒乓球锦标赛女单决赛中，辽宁队王艺迪4比3战胜黑龙江队王曼昱，赢得金牌。@迪214 @王曼昱 （记者：乐文婉 徐鹏航 伍志尊）

Fig. 9.7 An example of a real-time social event. (**a**) Conversational text. (**b**) Heterogeneous content. (**c**) Continuously growing real-time data. Parts of this figure are from [6]

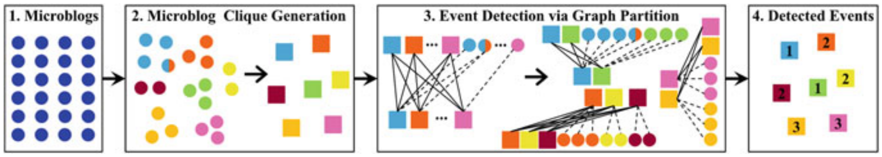

1. Microblogs
2. Microblog Clique Generation
3. Event Detection via Graph Partition
4. Detected Events

Fig. 9.8 Overall framework of the real-time social event detection. This figure is from [6]

introduce a hypergraph-based method for real-time social event detection. The overall framework is shown in Fig. 9.8.

(1) Microblog Clique Generation

Microblog clique (MC), which consists of a collection of closely connected tweets, is constructed as a basic unit rather than a single microblog in order to make up for the lack of information. These microblogs cover the same subject in short time.

A hypergraph is used to describe the relationship between heterogeneous data of various tweets. A set of microblogs is denoted as $M = \{m_1, m_2, \ldots, m_n\}$. The constructed hypergraph $\mathcal{G}_H = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, where a vertex v represents a microblog and a hyperedge e represents a subset of microblogs. The hyperedge weight is denoted as $w(e)$, and its diagonal matrix is formed as $\mathbf{W}$. The similarity between two microblogs m_i and m_j is first determined using the following heterogeneous features in order to generate hyperedges.

The cosine similarity function is used for computing textual and visual similarities. The Haversine formula is used for measuring the geographical similarity. The pairwise temporal similarity is calculated by $s_{TI}(m_i, m_j) = 1 - \frac{|ti_i, tj_i|}{\tau}$. The timestamps of m_i and m_j are ti_i and tj_i, while τ denotes a normalized constant. Measures of the pairwise social similarity are

$$s_S(m_i, m_j) = \begin{cases} 1, & \text{if } u_i = u_j \\ 0.5, & \text{if } u_i \text{ and } u_j \text{ are linked through the social platform} \\ 0, & \text{otherwise} \end{cases}, \tag{9.37}$$

where u_i is the owner of m_i.

Two hyperedges are created by connecting each microblog m_i with its neighbors as per geographic distance and middle position of location and time information. For each microblog m_i, the top N nearest microblogs in terms of textual information and visual content are chosen. Finally, all microblogs of the same user are connected to generate a hyperedge. The incidence matrix, vertex degree, and edge degree of the hypergraph are defined in the same way as above.

Next, MC is generated by dividing microblogs into groups of the same topic through the hypergraph cut approach. Assume S and $\bar{S}$ are the results of $\mathcal{G}_H$ through the two-way partition, and the hypergraph cut can be described as

$$\begin{aligned} \text{Cut}_H(S, \bar{S}) &:= \sum_{e \in \partial S} w(e) \frac{|e \cap S||e \cap \bar{S}|}{d(e)}, \\ \partial S &:= \{e \in E | e \cap S \neq \emptyset, e \cap \bar{S} \neq \emptyset\}. \end{aligned} \tag{9.38}$$

The definition of the two-way normalized partition is

$$N\text{Cut}_H(S, \bar{S}) := \text{Cut}_H(S, \bar{S}) \left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(\bar{S})} \right), \tag{9.39}$$

where the volume of S is denoted by $\text{vol}(S) = \sum_{v \in S} D(v)$. A real-valued optimization work can be relaxed from the normalized cut issue. By choosing the eigenvectors corresponding to the smallest non-zero eigenvalues of the hypergraph Laplacian, $\Delta = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$, and the solution can be found. The input tweets M are split into two groups, and then a bidirectional normalized partitioning is carried out recursively in each new set until the best partitioning outcome is attained. Based on the representation capacity of the various partitions as achieved by Bayesian Information Criteria (BIC), this best partitioning result is determined.

BIC is used to choose the optimal hypergraph partitioning results. For $M = \{m_1, m_2, \ldots, m_n\}$, with $P = \{P_1, P_2, \ldots, P_m\}$ as a set of partitions, the BIC score is determined by

$$\begin{aligned}
&\mathrm{BIC} = \mathrm{llh}(M) - \tfrac{N_p}{2} \log n, \\
&\mathrm{llh}(M) = \sum_i \left(\tfrac{1}{\sqrt{2\pi}\hat{\theta}^{N_p}} - \tfrac{1}{2\hat{\theta}^2} \left\| d(m_i, c_{m_i}) \right\|^2 + \log \tfrac{n_i}{n} \right), \\
&\hat{\theta}^2 = \tfrac{1}{n-m} \sum_i d(m_i, c_{m_i})^2,
\end{aligned} \tag{9.40}$$

where N_p represents the parameter number and the microblog features' dimension, n is the microblogs number, and n_i is the count of corresponding partition of m_i.

Following the division of the provided microblogs into a group of MCs, the MCs offer more sensible information by examining a collection of strongly correlated microblogs rather than individual microblogs, which can express more meaningful and pertinent material in the succeeding event detection technique.

(2) Detection of Social Events in Real Time

Event Detection by Using MC For MC $= \{\mathrm{MC}_1, \ldots, \mathrm{MC}_p\}$ and corresponding microblogs M $= \{m_1, \ldots, m_n\}$, there are two observations as follows. First off, inside a single MC, and microblogs frequently refer to the same event (MC cues). Second, MCs with similar features tend to be associated with the same event (smoothness cues).

If a microblog is integrated into an MC, it is connected to the MC to impose MC cues. In order to enforce smoothness cues, pairwise MCs that are close to one another in feature space are connected. Formally, a bipartite graph $\mathcal{G}_{\mathcal{B}} = \{X, Y, B\}$ is used to express MC and M, and two vertex sets are expressed as X and Y, where $X := \mathrm{MC} \cap \mathrm{M}$, $Y := \mathrm{MC}$, with $|X| = |\mathrm{MC}| + |\mathrm{M}|$ and $|Y| = |\mathrm{MC}|$ vertices, respectively. The definition of the across-affinity matrix B between X and Y is as follows:

$$B_{ij} = \begin{cases} \eta, & \text{if } x_i \in \mathrm{M}, x_i \in y_j, y_j \in \mathrm{MC} \\ e^{-\gamma d_{ij}}, & \text{if } x_i \in \mathrm{MC}, y_j \in \mathrm{MC} \\ 0, & \text{otherwise} \end{cases}, \tag{9.41}$$

where d_{ij} is the distance between two MCs, and η and γ are the two parameters that balance the inner-MC correlation and the between-MC smoothness.

The bipartite graph $\mathcal{G}_{\mathcal{B}}$ and the necessary number of partitions K are used as the basis for the transfer cut method to partition MCs. First, assume $\mathcal{G}_{\mathcal{BY}} = \{Y, \mathbf{W}_Y\}$ contains only vertices of the MC. $\mathbf{L}_Y = \mathbf{D}_Y - \mathbf{W}_Y$ is the graph Laplacian of $\mathcal{G}_{\mathcal{BY}}$, where $\mathbf{D}_Y = \mathrm{diag}(\mathbf{B}^\top \mathbf{1})$, $\mathbf{W}_Y = \mathbf{B}^\top \mathbf{D}_X^{-1} \mathbf{B}$. Assume that $\{\lambda_i, \mathbf{v}_i\}_1^K$ are the K smallest eigenpairs of $\mathcal{G}_{\mathcal{B}}$. They can be calculated as

$$\begin{aligned} &0 \le \xi_i \le 1, \xi_i(2-\xi_i) = \lambda_i, \\ &\mathbf{u}_i = \frac{1}{1-\xi_i}\mathbf{Q}\mathbf{v}_i, \mathbf{f}_i = (\mathbf{u}_i^\top, \mathbf{v}_i^\top)^\top, \end{aligned} \tag{9.42}$$

where $\mathbf{Q} = \mathbf{D}_{\mathbf{X}}^{-1}\mathbf{B}$ is the corresponding transition probability matrix from X to Y.

Second, $\{\mathbf{f}_1, \ldots, \mathbf{f}_K\}$ are K-spectra clustered and the best K is selected by BIC. Assume that K_0 is the count of existing events. It is started at 0. Furthermore, suppose that the biggest number for incoming data is not larger than $K_0 + n_{new}/t_m$, where the threshold t_m is used to decide the minimum microblog number. Therefore, the bipartite graph is segmented $n_{new}/t_m + 1$ times, and the segmentation result is selected as the event detection result using BIC. Suppose $\{\varGamma_1, \ldots, \varGamma_K\}$ are the detected K events in the last process. The key MCs are found by MC selection for each $\varGamma_i$, and the number of each MC is measured in terms of importance. Finally, the top n_{sMC} MCs are selected to describe each $\varGamma_i$.

Detection of Incremental Social Events The real-time detection method is defined as follows. Assume that event detection is run at time t_0, with generated MCs, i.e., MC $= \{\text{MC}_1, \ldots, \text{MC}_p\}$, detected events $\{\varGamma_1, \ldots, \varGamma_q\}$, and noisy data. New data arrive continuously from moment t_0, and it can be processed a short time gap t. In other words, event detection can be run at every $t_0 + x \times t$, where x equals to $1, 2, \ldots$. In this instance, $t_0 + \Delta_t$ is used as an example, and M_{new} stands for newly arriving microblogs. The two steps that make up event detection are MC generation and event partition.

To generate new MCs for previous time periods, $MC^* = \{MC_1^*, MC_2^*, \ldots.,$ $MC_{n_e}^*\}$ were used as known samples. MC^* and M_{new} are used to construct the incremental microblog hypergraph $\mathcal{G}_H^{t_0+\Delta_t}$. However, it is challenging because there is no clear distinction between a microblog collection and a microblog. No more than $3n_e$ representative microblogs get to be chosen since only the three most representative tweets for each MC are chosen, depending on the amount of retweets and comments. To create the incremental microblog hypergraph $\mathcal{G}_H^{t_0+\Delta_t}$, they are merged with M_{new}. New MCs (MC_{new0}) are then created from these data using the hypergraph partition. Based on the representative microblogs, MC_{new0} and MC* are combined together. In this way, $n_{\text{MC}_{\text{new}}}$ new MCs (MC_{new0}) are constructed and utilized for event detection.

For detection in real time, the past events $\varGamma = \{\varGamma_1, \ldots, \varGamma_K\}$are used as known data in the time period. The corresponding representative MCs in $\varGamma$ and the generated incremental MC$_{\text{new}}$ are used to jointly construct the next graph. The difference is that for the identified events, the distance between MCs is set to 0 as follows:

$$d_{ij} = \begin{cases} 0, \text{ if } x_i \in \varGamma_k \text{ and } y_j \in \varGamma_k \\ \min\limits_{\substack{mx_k \in x_i \\ my_l \in y_j}} d(mx_k, my_l), \text{ otherwise} \end{cases}, \tag{9.43}$$

where $k = 1, 2, \ldots, K$. Therefore, according to the BIC, the bipartite graph can be partitioned into existing events and new events.

There are still several challenging problems in hypergraph computation for sentiment analysis tasks that can be continued for more research. First, for the sentiment recognition task, the case of conflicting multi-modal information can be considered. Second, further consideration can be given to the information that may be hidden in broken posts and users for the detection task on real-time social events. These tasks take into account the positive or negative associations among multiple entities, where the hypergraph is suitable for modeling such correlations.

9.4 Emotion Recognition

Emotion recognition has gained wide recognition in neuroscience and psychology research [11], and artificial intelligence offers more reliable and accurate computational models for the identification and study of emotions. It has also been extensively applied in real life [12], especially in human–computer interaction, motor vehicle driving assistance training, emotion classification in movies, and other pertinent similar areas [13].

Emotion recognition has three main goals [14]: first, to enable the understanding, inference, and recognition of human emotions by intelligent systems; second, to make it possible for systems to make human-like expressions of emotion in response to stimuli (e.g., conversational agents or robots); and third, to make it possible for intelligent systems to actually perceive emotions. Over the past three decades, researchers from several disciplines have pursued these three goals in different ways, with the method of recognizing emotions as the central issue of research. Although it has been studied for many years, progress is still being made. The reality is that there are various ways for people to convey their emotions, including language, gestures, facial expressions, and physiological signs [15]. Finding a suitable method to identify and analyze human emotions may be a long-term problem. Human volition determines the first three modalities, and there are substantial individual variances [16]. Because of these, approaches based on these three modalities have limitations in terms of accuracy and reliability. In contrast, physiological signals cannot be readily blocked or concealed and are simultaneously governed by the body's neurological and hormonal systems. They are also often independent of human will. Therefore, physiological signals rather than visual or auditory cues may offer more accurate information about emotions [17]. A multitude of environmental and psychological elements, including interests and personality, can have an impact on human emotion, which is a highly subjective phenomenon.

Nonetheless, because of the following factors, recognizing emotions through physiological signals is still a work in progress:

- Existence of the emotional gap and ambiguity in the concept of emotions [18]
- Potential associations between modality and subject [19]

- Specificity of the stimulus response (SR) and individual response (IR) [20]
- Noise and incomplete data in the data [21]
- Multifactorial influences [22]

In this case, the hypergraph structure allows the establishment of complex correlations that can simultaneously take into account: (a) correlations between EEG, EOG, and EMG signals, which are signals from several modalities; (b) correlations between subjects; and (c) patterns of physiological signal changes in a single subject in response to various stimuli. Two methods are presented for emotion prediction using hypergraph computation, including multi-modal vertex-weighted hypergraph learning (MVHL) [7, 8] and multi-hypergraph neural networks (MHGNN) [9].

(1) Multi-Modal Vertex-Weighted Hypergraph Learning

Hypergraphs have been used to depict the link between physiological data and personality [7]. In this way, MVHL introduces a multi-modal vertex-weighted hypergraph learning method for personalized emotion recognition (PER) that takes into account vertex weights, hyperedge weights, and modal weights. Each vertex in this method is a composite tuple (subject, stimulus). A hypergraph structure is used to develop personality correlations between various subjects and physiological correlations between the corresponding stimuli. Each vertex and hyperedge, as well as the weights of the various hypergraphs, are automatically learned. Hyperedge weights are used to create the optimal representation, while vertex weights are used to describe the impact of various samples and patterns in the learning process. The calculated factors—known as sentiment relevance—are employed for sentiment identification and are learned on a multi-modal vertex-weighted hypergraph. The fact that the vertices are composite with incorporated data from various subjects allows MVHL to identify numerous subjects' emotions at once.

The framework of this model is shown as follows. First, a composite tuple of vertices (subjects, stimuli) is formed using the subjects and the stimuli used to elicit the subjects' emotions. Second, multi-modal hyperborders are constructed to form personality associations among different subjects and physiological associations among the corresponding stimuli. Finally, after joint learning of vertex-weighted multi-modal multi-task hypergraphs, PER results can be obtained.

Hypergraph Construction This model constructs the hypergraph structure by pairwise similarity between different samples. The pairwise similarity of u_i and u_j's personalities is measured by the cosine function:

$$s_{PER}(u_i, u_j) = \frac{< \mathbf{p}_i, \mathbf{p}_j >}{\|\mathbf{p}_i\| \cdot \|\mathbf{p}_j\|}, \tag{9.44}$$

where u_i's personality vector is denoted by $\mathbf{p}_i$. The centroid is determined by selecting one vertex at a time, and a hyperedge is built to link the centroid to its K nearest neighbors in the existing representation space. It should be noted that personified hyperedges are built using both intra- and inter-subject viewpoints. A hyperedge links all the vertices from the same subject together. Additionally, based on personality similarities, the closest K subjects for each subject are chosen, and all of their vertices are connected by creating another hyperedge.

Assume that the constructed hypergraphs are $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m, \mathbf{W}_m)$, where $\mathcal{V}_m$ and $\mathcal{E}_m$ denote the vertex set and hyperedge set, respectively, and $\mathbf{W}_m$ is the diagonal hyperedge weight matrix of the m-th hypergraph ($m = 1, 2, \ldots, M$). The incidence matrix $\mathbf{H}_m$ can be computed as

$$\mathbf{H}_m(v, e) = \begin{cases} 1, \text{ if } v \in e \\ 0, \text{ if } v \notin e \end{cases}. \tag{9.45}$$

The different weights of the vertices are learned to evaluate their value and contribution to the learning process. It is distinct from the classic hypergraph learning method, which simply views all the vertices equally. Assume $\mathbf{U}_m$ is the diagonal matrix of vertex weight. The vertex degree and the hyperedge degree are defined as $d_m(v) = \sum_{e \in \mathcal{E}_m} \mathbf{W}_m(e)\mathbf{H}_m(v, e)$ and $\delta(e) = \sum_{v \in \mathcal{V}_m} \mathbf{U}_m(e)\mathbf{H}_m(v, e)$. Accordingly, the two diagonal matrices are defined as $\mathbf{D}_m^v(i, i) = d_m(v_i)$ and $\mathbf{D}_m^e(i, i) = \delta_m(e_i)$.

Multi-Modal Vertex-Weighted Hypergraph Learning The goal is to simultaneously study the correlations among the included physiological signals and the personality relations across various subjects. The framework of the multi-modal vertex-weighted hypergraph learning is presented in Fig. 9.9. Given N subjects $u_1, \ldots, u_N$ and the involved stimuli $s_{ij}(j = 1, \ldots, n_i)$ for u_i, we assume that the c-th emotion category's compound vertices and associated labels are $\{(u_1, s_{1j})\}_{j=1}^{n_1}, \ldots, \{(u_N, s_{Nj})\}_{j=1}^{n_N}$ and $\mathbf{y}_{1c} = [y_{11}^c, \ldots, y_{1n_1}^c]^\top, \ldots, \mathbf{y}_{Nc} = [y_{N1}^c, \ldots, y_{Nn_N}^c]^\top$, where $c = 1, \ldots, n_e$.

The count of emotion categories is denoted as n_e. The estimated values of all stimuli associated to the specified users of the c-th emotion category, also

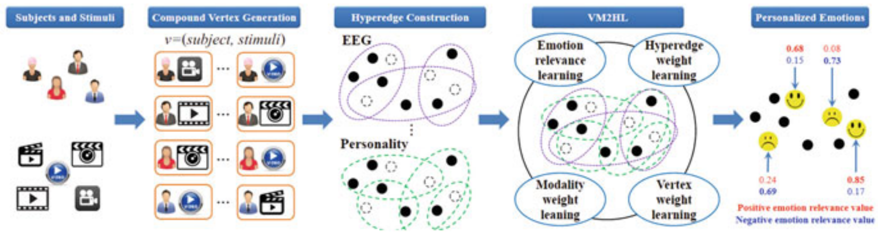


Fig. 9.9 Overall framework of the multi-modal vertex-weighted hypergraph learning. This figure is from [7]

known as emotion relevance, are given by $\mathbf{r}_{1c} = [r_{11}^c, \ldots, r_{1n_1}^c]^\top, \ldots, \mathbf{r}_{Nc} = [r_{N1}^c, \ldots, r_{Nn_N}^c]^\top$. $\mathbf{y}_c$, $\mathbf{r}_c$ are denoted by

$$\mathbf{y}_c = [\mathbf{y}_{1c}^\top, \ldots, \mathbf{y}_{Nc}^\top]^\top, \mathbf{r}_c = [\mathbf{r}_{1c}^\top, \ldots, \mathbf{r}_{Nc}^\top]^\top. \tag{9.46}$$

Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_c, \ldots, \mathbf{y}_{n_e}]$, $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_c, \ldots, \mathbf{r}_{n_e}]$, where the two trade-off parameters are λ and η. The hypergraph structure's regularizer is defined as follows:

$$\Psi(\mathbf{R}, \mathbf{W}, \mathbf{U}, \boldsymbol{\alpha}) = \sum_{c=1}^{n_e} \mathbf{r}_c^\top \sum_{m=1}^{M} \alpha_m (\mathbf{U}_m - \Theta_m) \mathbf{r}_c, \tag{9.47}$$

where $\Theta_m = (\mathbf{D}_m^v)^{-1/2} \mathbf{U}_m \mathbf{H}_m \mathbf{W}_m (\mathbf{D}_m^e)^{-1} \mathbf{H}_m^\top \mathbf{U}_m (\mathbf{D}_m^v)^{-1/2}$. Then, $\Delta = \sum_{m=1}^{M} \alpha_m (\mathbf{U}_m - \Theta_m)$ can be seen as the fused hypergraph Laplacian with vertex weighting.

(2) Multi-Hypergraph Neural Networks

Multi-hypergraph neural network (MHGNN) uses hypergraph to build complex correlations and identify emotions by physiological signals, which can take into account: (a) correlations between signals of various modalities, i.e., z EEG, EOG, and EMG; (b) relationships between subjects; and (c) patterns of physiological signal changes in a single person in response to various stimuli. This model groups each given subject and stimuli to a complex tuple, respectively. Assuming it is a vertex in the hypergraph, it would generate a hypergraph for each pattern with its corresponding physiological signal, making use of the term hyperedge to express the correlations among the physiological signals in response to various stimuli. The vertices are then categorized within the MHGNN framework in accordance with the intricate relationships in the data. As a result, the categorization of vertices in various hypergraphs can be equated to the recognition of emotions. Different hypergraph neural networks are combined using a fully connected network. The relative relevance of various multi-modal physiological signals is also taken into account of this network when classifying emotions. This framework's primary benefit is its ability to combine multi-modal data and to represent three intricate relationships of the data. Figure 9.10 shows the pipeline of the MHGNN framework.

Modeling of Multi-Hypergraph Subject correlation is formulated using a multi-hypergraph structure given a number of features from various physiological inputs. Each modality is represented by a separate hypergraph. The connections between the vertices of the hypergraph are constructed using hyperedges, and each vertex on the hypergraph represents a topic to be learned with a description of its corresponding stimuli. The k-NN method is used to generate hypergraphs, where k is a hyperparameter for assessing the connectivity. The hyperedges are created after all vertices have acted as the centroid. Each vertex gets chosen as a centroid once. We assume that $S = S_1, S_2, \ldots, S_n$ is defined as a training set with modality

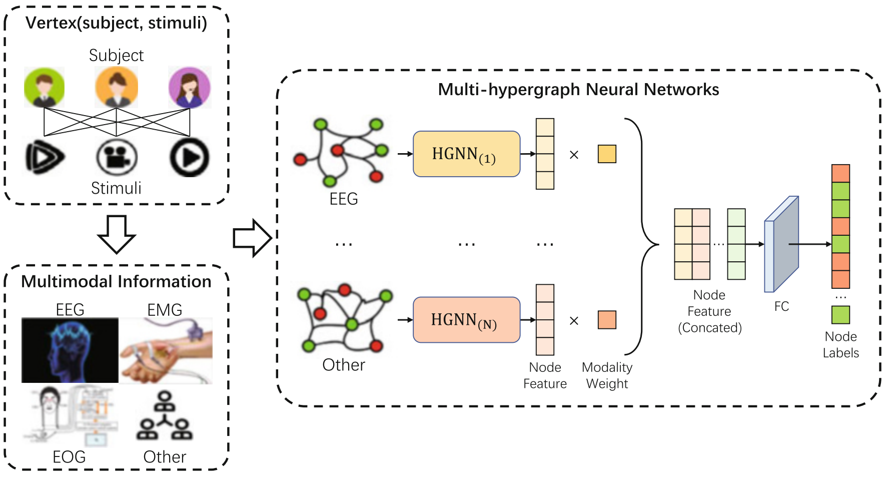


Fig. 9.10 The pipeline of multi-hypergraph neural networks

i's features $\mathbf{X}^{(i)} = \mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \ldots, \mathbf{x}_n^{(i)}$, where vector $\mathbf{x}_j^{(i)}$ is the feature of the j-th training sample from modality i and S_j denotes the j-th training sample. According to the KNN approach, the vertex v_p shares the hyperedge with the k nearest vertices above and around it. Hyperedge e_p is centered on the vertex v_p. The Euclidean distance between the corresponding feature vectors represents the separation between two vertices. The correlation between vertex p and vertex q is represented by the matrix element $h_{p,q}$. As an exponential representation of Euclidean distance, the correlation can be described as

$$h_{p,q}^{(i)} = \begin{cases} \exp(-\frac{d\left(\mathbf{x}_p^{(i)}, \mathbf{x}_q^{(i)}\right)^2}{d^2}), & q \in u_p \\ 0, & q \notin u_p \end{cases}, \tag{9.48}$$

where $d(\mathbf{x}_p^{(i)}, \mathbf{x}_q^{(i)})$ stands for the feature space Euclidean distance between samples p and q. The weight matrix $\mathbf{W}^{(i)}$ is set to be an identity matrix in our model because we lack prior knowledge regarding the significance of hyperedges. As a result, the incident matrix $\mathbf{H}^{(i)}$ contains all the data for the hypergraph.

An incidence matrix $\mathbf{H}(i)$ is generated for each modality. Finally, m incident matrices can be generated for m modalities.

Multi-Hypergraph Convolutional Networks The creation of subject representation and subsequent emotion classification are crucial steps in emotion recognition. Deep neural networks have made significant progress in the representation of data in the last few years. However, given the intricacy of data correlations, it is still work in progress. In order to represent data and recognize emotions, a multi-hypergraph convolutional network framework that can simultaneously take into account several physiological inputs from different people is developed.

In a hypergraph convolutional network, the spatial convolution is viewed from the perspective of graph spectral theory as a spectral matrix product, and the hypergraph Laplacian $\varDelta$ is leveraged to convert it from the spatial domain to the spectral domain. $\varDelta$ can be formulated as $\varDelta = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$, where $\mathbf{D}_e$ and $\mathbf{D}_v$ are the matrices of hyperedge degree and vertex degree, respectively. In this case, it is possible to formulate a hypergraph convolutional layer for each modality as

$$\mathbf{X}_{(l+1)}^{(i)} = \sigma\left(\mathbf{D}_v^{(i)-1/2}\mathbf{H}^{(i)}\mathbf{W}^{(i)}\mathbf{D}_e^{(i)-1}\mathbf{H}^{(i)\top}\mathbf{D}_v^{(i)-1/2}\mathbf{X}_{(l)}^{(i)}\boldsymbol{\Theta}_{(l)}^{(i)}\right), \tag{9.49}$$

where $\boldsymbol{\Theta}_{(l)}^{(i)}$ is the learnable parameter of the l-th layer in i-th hypergraph neural network (HGNN) and σ is the activation function. When using hypergraph convolution, the parameters for $\boldsymbol{\Theta}^{(i)}$ are updated by backpropagating the feature $\mathbf{X}^{(i)}$. Hypergraph structure-related parameters, such as $\mathbf{D}_v^{(i)-1/2}\mathbf{H}^{(i)}\mathbf{W}^{(i)}\mathbf{D}_e^{(i)-1}\mathbf{H}^{(i)\top}\mathbf{D}_v^{(i)-1/2}$, are pre-computed and are not trainable in this procedure. The symbol $\mathbf{A}_h^{(i)}$ is used to represent these parameters for simplification, and the hypergraph convolutional layer can be rewritten as

$$\mathbf{X}_{(l+1)}^{(i)} = \sigma\left(\mathbf{A}_h^{(i)}\mathbf{X}_{(l)}^{(i)}\boldsymbol{\Theta}_{(l)}^{(i)}\right). \tag{9.50}$$

It is important to note that the formulation of graph convolution and hypergraph convolution is similar. The graph convolution is shown as follows:

$$\mathbf{X}_{(l+1)}^{(i)} = \sigma\left(\mathbf{D}^{(i)-1/2}\mathbf{A}^{(i)}\mathbf{D}^{(i)-1/2}\mathbf{X}_{(l)}^{(i)}\boldsymbol{\Theta}_{(l)}^{(i)}\right). \tag{9.51}$$

Hyperedges built from characteristics of several modalities are concatenated in traditional models of single hypergraph neural networks. However, because of their distinct sizes and dimensions, hyperedges have been known of being inconsistent. Additionally, there could be some variations in the perspectives from which various modalities approach the work. Some could be crucial, while others might not be just as important. In a single hypergraph model with identical weights, such discrepancies are not possible to see. However, simply concatenating distinct hyperedges makes it difficult to specifically weight them. A multi-hypergraph neural network structure is introduced to integrate multiple hypergraph structures in order to address the issue.

To calculate intermediate representations for each modality, m hypergraph neural network models are built using m hypergraphs for m modalities. The K-layer i-th hypergraph neural network may be expressed as follows:

$$HGNN(\mathbf{H}^{(i)}, \mathbf{X}^{(i)}) = \sigma_K^{(i)}\left(\mathbf{A}_h^{(i)}(\cdots\sigma_1^{(i)}(\mathbf{A}_h^{(i)}\mathbf{X}^{(i)}\boldsymbol{\Theta}_1^{(i)})\cdots)\boldsymbol{\Theta}_K^{(i)}\right). \tag{9.52}$$

The final output is then generated using the m output of intermediate representations by a fully connected layer. As a fusion layer, the layer dynamically combines the outcomes of hypergraph convolutions and weights them corresponding to their contributions. A softmax layer serves as the classifier. In layers of networks with diverse hypergraph structures, modality characteristics of various sizes and dimensions are learned. Finally, they are weighted automatically and merged into the fusion layer.

$\mathbf{W}_f$ and $\mathbf{b}_f$ stand for the weights and bias of the fusion layer, respectively. The model can be expressed as follows:

$$\begin{aligned} MHGNN(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(m)}) =& softmax\Big(\mathbf{W}_f \mathbf{W}_m [HGNN(\mathbf{H}^{(1)}, \mathbf{X}^{(1)}), \\ & HGNN(\mathbf{H}^{(2)}, \mathbf{X}^{(2)}), \ldots, \\ & HGNN(\mathbf{H}^{(m)}, \mathbf{X}^{(m)})] + \mathbf{b}_f\Big), \end{aligned} \tag{9.53}$$

where the matrix of modality weights is denoted by $\mathbf{W}_m = Diag\left(\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(m)}\right)$.

The patterns were discovered to represent a pair of interconnected and mutually reinforcing interdisciplinary concerns by examining the data findings making use of the network structure of the hypergraph. Another intriguing occurrence in the experiments was the variations in each subject's physiological characteristics. Therefore, what should be considered is to: (a) collect data according to the requirements of real application scenarios; (b) pay attention to individual differences; (c) analyze correlations between subjects of training and test samples; and (d) add more information such as action recognition information. Hypergraphs are considered as a good tool to discover biological patterns among them.

9.5 Summary

In this chapter, to illustrate the paradigm of using hypergraph computation in social media analysis, we overview three applications, i.e., recommender system, sentiment analysis, and emotion recognition. In recommender system, we discuss two specific applications: collaborative filtering and attribute inference. Collaborative filtering only considers the raw user–item network, and hypergraph is used to model the inter- and intra-domain (user or item) correlations in behavior space. Attribute inference further takes the attribute information into consideration in addition to the historical interactions. Besides, context information such as time and location can also be integrated, which is left to explore. In sentiment analysis, sentiment prediction and social event detection are covered. The former task mainly concerns the sentiment conveyed by each multi-modal tweet, while the latter one focuses on exploring a group of postings that are closely connected and cover the same subjects.

Furthermore, recognizing the emotion of people through multi-modal physiological signals is also presented. There are still many social media analysis applications worth exploring with hypergraph computation. For example, heterogeneous correlations widely exist in the social media context. How to utilize the complementary information among these heterogeneous associations with hypergraph computation has become a key issue. Besides, social media data are always dynamic rather than static, and the newcoming data may have different distributions compared with the existing data. Under such circumstances, the static hypergraph computation method cannot be directly applied, and the dynamic hypergraph computation paradigm is deserved to be investigated to solve this complex issue.

References

1. S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **52**(1), 1–38 (2019)
2. L. Zhang, S. Wang, B. Liu, Deep learning for sentiment analysis: a survey. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **8**(4), 1–25 (2018)
3. L. Shu, J. Xie, M. Yang, Z. Li, Z. Li, D. Liao, X. Xu, X. Yang, A review of emotion recognition using physiological signals. Sensors **18**(7), 2074 (2018)
4. S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in *Proceesings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2020), pp. 2020–2029
5. R. Ji, F. Chen, L. Cao, Y. Gao, Cross-modality microblog sentiment prediction via bi-layer multimodal hypergraph learning. IEEE Trans. Multimedia. **21**(4), 1062–1075 (2019)
6. S. Zhao, Y. Gao, G. Ding, T.-S. Chua, Real-time multimedia social event detection in microblog. IEEE Trans. Cybern. **48**(11), 3218–3231 (2018)
7. S. Zhao, G. Ding, J. Han, Y. Gao, Personality-aware personalized emotion recognition from physiological signals, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), pp. 1660–1667
8. S. Zhao, A. Gholaminejad, G. Ding, Y. Gao, J. Han, K. Keutzer, Personalized emotion recognition by personality-aware high-order learning of physiological signals. ACM Trans. Multimedia Comput. Commun. Appl. **15**(1s), 1–18 (2019)
9. J. Zhu, Y. Wei, Y. Feng, X. Zhao, Y. Gao, Physiological signals-based emotion recognition via high-order correlation learning. ACM Trans. Multimedia Comput. Commun. Appl. **15**(3s), 1–18 (2019)
10. W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in *Proceedings of the Advances in Neural Information Processing Systems* (2017), pp. 1024–1034
11. E. Cambria, Affective computing and sentiment analysis. IEEE Intell. Syst. **31**(2), 102–107 (2016)
12. S. Zhao, X. Zhao, G. Ding, K. Keutzer, EmotionGAN: Unsupervised domain adaptation for learning discrete probability distributions of image emotions, in *Proceedings of the 26th ACM International Conference on Multimedia* (2018), pp.1319–1327.
13. S. Poria, E. Cambria, R. Bajpai, A. Hussain, A review of affective computing: from unimodal analysis to multimodal fusion. Inf. Fusion. **37**, 98–125 (2017)
14. R. Calvo, S. D'Mello, Affect detection: An interdisciplinary review of models, methods, and their applications. IEEE Trans. Affective Comput. **1**(1), 18–37 (2010)
15. S. D'mello, J. Kory, A review and meta-analysis of multimodal affect detection systems. ACM Comput. Surv. **47**(3), 1–36 (2015)

16. M. Soleymani, S. Asghari-Esfeden, Y. Fu, M. Pantic, Analysis of EEG signals and facial expressions for continuous emotion detection. IEEE Trans. Affect. Comput. **7**(1), 17–28 (2016)
17. Y. Shu, S. Wang, Emotion recognition through integrating EEG and peripheral signals, in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing* (2017), pp. 2871–2875
18. S. Zhao, C. Lin, P. Xu, S. Zhao, Y. Guo, R. Krishna, G. Ding, K. Keutzer, CycleEmotion-GAN: Emotional semantic consistency preserved cycleGAN for adapting image emotions, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 2620–2627
19. M. Soleymani, M. Pantic, T. Pun, Multimodal emotion recognition in response to videos. IEEE Trans. Affect. Comput. **3**(2), 211–223 (2012)
20. P. Ekman, R. Levenson, W. Friesen, Autonomic nervous system activity distinguishes among emotions. Science **221**(4616), 1208–1210 (1983)
21. R. Subramanian, J. Wache, M.K. Abadi, R.L. Vieriu, S. Winkler, N. Sebe, Ascertain: emotion and personality recognition using commercial sensors. IEEE Trans. Affect. Comput. **7**(1), 17–28 (2018)
22. E. Kehoe, J. Toomey, J. Balsters, A. Bokde, Personality modulates the effects of emotional arousal and valence on brain activation. Soc. Cognit. Affect. Neurosci. **7**(7), 858–870 (2012)

Chapter 10
Hypergraph Computation for Medical and Biological Applications

Abstract Hypergraph computation, with its superior capability in complex data modeling, is a powerful tool for many medical and biological applications. In this chapter, we introduce four typical examples of the use of hypergraph computation in medical and biological applications, i.e., computer-aided diagnosis, survival prediction with histopathological images, drug discovery, and medical image segmentation. In each application, we present how to construct the hypergraph structure with different kinds of medical and biological data and different hypergraph computation strategies for these tasks respectively. We can notice that hypergraph computation has shown advantages in these applications.

10.1 Introduction

In the past few decades, massive biological and medical data were generated owing to the rapid development of big data techniques. These data can be used for tasks of disease gene analysis, disease risk assessment, targeted drug discovery, etc. The data further contribute to disease prevention and early diagnosis and treatment of diseases. The biological and medical data are complex, heterogeneous, and multi-modal, with widespread inter- and intra-data correlations. For example, in early disease diagnosis, patients with similar medical image appearance may also share similar disease conditions; different modalities of the medical image of the same patient, such as MRI and CT, may also exhibit disease characteristics from different perspectives; the patches within gigapixel histopathological images may have implicit collaborative associations that reveal patients' potential health risks. Therefore, how to model such correlation behind these data is very important for medical and biological applications.

Hypergraphs, which own the flexible hyperedges, provide a possible solution for modeling such complex correlations within medical and biological data. Given the observed data, the hypergraph structure can be generated using the previously mentioned methods and naturally incorporate multi-modal or heterogeneous data by concatenation of hyperedge groups and thus can discriminatively utilize the complementary information of these data. The applications of hypergraph computation in medical and biological tasks can be typically summarized as follows: (1) modeling the medical image, the patches, or the biological entities such as vertices, and connecting them with hyperedges following their feature similarity or high-order topological links; (2) exploring the high-order correlations between data using hypergraph label propagation or hypergraph neural networks so as to enhance the vertex representations; and (3) deploying these representations on the medical and biological tasks, such as medical image retrieval, disease identification, cancer tissue classification, survival prediction, and medical image segmentation.

In this chapter, we discuss five typical applications of using hypergraph computation in medical and biological applications, i.e., computer-aided diagnosis, survival prediction with histopathological images, drug discovery, and medical image segmentation. In computer-aided diagnosis, three specific applications are included, i.e., the identification [1] and medical image retrieval of MCI [2], autism spectrum disorder identification using brain functional networks [3], as well as the identification of COVID-19 by CT imaging [4]. For survival prediction with histopathological images, two techniques targeting different cases are displayed, including ranking-based survival prediction [5] and multi-hypergraph modeling for survival prediction [6]. In drug discovery, a heterogeneous hypergraph-based drug–target interaction prediction technique [7] is presented. For medical image segmentation, we introduce the hierarchical hypergraph patch labeling method. Part of the work introduced in this chapter has been published in [1–8].

10.2 Computer-Aided Diagnosis

Computer-aided diagnosis has made clinical diagnosis incredibly convenient with the advancement of artificial intelligence and owing to the widespread use of medical imaging data, including MRI, CT scan, histopathological images, and so on. Its main goal is to pursue a preliminary examination of patients for clinicians in order to increase diagnostic accuracy, avoid missed illnesses, and improve work efficiency. Many challenges still exist in the field of computer-aided diagnosis despite great machine learning and deep learning research advancements. It involves improper uses of information shared among patients and different forms of medical images, the continued existence of noisy data (such as variations in varied CT manufactures and patients' movement during imaging), and the confusion of cases in the early stages of illness.

In traditional approaches, the relationships among patients are frequently ignored in favor of merely taking into account one patient. The illness information of patients with similar medical images assists to raise the likelihood of computer-aided diagnosis since it makes sense that if the MRI or CT features of patients are related, then their disease conditions should also be similar. Therefore, since hyperedges in hypergraphs, unlike in graphs, can connect two or more vertices, this presents a potential solution for the first challenge by allowing hypergraphs to represent high-order illness connections among multiple individuals.

Computer-aided diagnosis with medical images frequently consists of three main steps in order to be more effective. Pre-processing the image is the first step, which mostly consists of enhancing visual information, filtering out the background, and separating the region of interest from the blank to lessen interference of irrelevant areas. The next stage is to extract the region of interest's features. Imaging features including infection lesion count, mean lesion area, lesion density, and morphological aspects must be extracted from images since it is informative and contains task-independent information. The final step is to use machine learning, deep learning, or other statistical approaches to diagnose patients and then identify various types and lesion types with the features gathered in the previous steps.

The use of hypergraph computing techniques in computer-aided diagnosis is introduced in the subsections that follow. Four specific applications are covered, namely MCI identification using MRI [1], medical image retrieval [2], COVID-19 identification using CT imaging [4], and ASD identification using brain functional networks [3]. First, we present a strategy for creating a hypergraph for each MRI sequence and modeling the best correlation of patients by information shared by several MRI sequences. It then explains how to generate multi-graph combination weights to discover the association among query subjects and the existing subject classes. This enhances the precision of medical image retrieval. In the third part, the details of the uncertainty vertex-weighted hypergraph learning approach distinguishing COVID-19 from other types of pneumonia symptoms are described. Finally, we show the application of dynamic hypergraph learning methods to diagnose the autism of children using multi-modal functional connectivity.

10.2.1 MCI Identification Using MRI

Identifying the initial phase of Alzheimer's disease (AD) [i.e., mild cognitive impairment (MCI)] to support the diagnosis is a proper but challenging task since AD is a relatively regular dementia in seniors. Taking into consideration that research has demonstrated that combining data from various data modalities can improve the accuracy of diagnosing AD/MCI, clinically routine scans are to be used in the upcoming hypergraph computing approaches to diagnosing AD to capture multiple MR sequences of various aspects of brain structures or functions and attempt to combine them optimally.

The centralized hypergraph learning method (CHL) [1] integrates numerous imaging data in a semi-supervised manner to estimate correlations among various subjects to indicate the possibility that subjects belong to the same class. This improves the utilization of multi-modal data, of which the global illustration is shown in Fig. 10.1. In contrast to the usual graphs, hypergraphs propagate information by a group of hyperedges connecting two or more vertices concurrently. They can also capture higher-order relationships among various subjects by selecting the nearest neighbors in the feature space, i.e., whether a set of subjects in this task has common information, therefore allowing each subject to maximize the

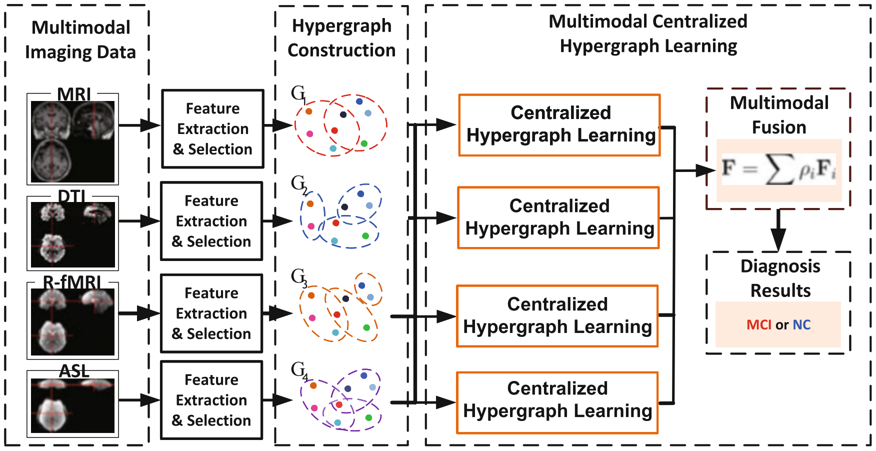


Fig. 10.1 A pipeline to classify MCI or NC from multi-modal imaging data using centralized hypergraph learning. This figure is from [1]

knowledge from MR sequences by optimizing concurrently the correlation and hyperedge weights among subjects. The entire process is sequentially presented in two stages, including the construction of a centralized hypergraph via processing data, and centralized hypergraph learning, to better introduce the details of using CHL in this chapter.

Different types of imaging data from patients with MCI and normal control (NC) need to be pre-processed as features before such data are used to construct the hypergraphs. Thereafter, a hypergraph $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i, \mathbf{W}_i \rangle$ is constructed for every sort of imaging data, where each subject is considered as a vertex, while the star expansion procedure is used to generate hyperedges. In particular, every vertex in each feature space is taken into account as the central vertex for generating a hyperedge, which consists of vertices located within distance $\varphi \bar{d}$ of the center vertex, where φ is a hyperparameter and $\bar{d}$ is the vertex's mean distance in feature space. The hypergraph incidence matrix $\mathbf{H}_i$ produced by the star expansion procedure is formalized as

$$\mathbf{H}_i(v, e) = \begin{cases} \exp\left(-\frac{d_i(v, v_c)}{0.1\bar{d}_i}\right) & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases}, \tag{10.1}$$

where $d_i(v, v_c)$ represents the length from the vertex v to the correlating center vertex v_c, and $\bar{d}_i$ is the vertex's mean distance in feature space of the i-th type imaging data. It should be noted that the hyperedge weights $\mathbf{W}_i$ start out with the same value, e.g., 1, when the hypergraph is generated.

For the MCI diagnostic work, which is regarded as a binary classification, various imaging data are employed to construct correlations among subjects using the centralized hypergraph learning method. Each step selects a hypergraph as the core hypergraph out of the four that were created from four types of data, with the others offering additional input for updating the hypergraphs. If hypergraph $\mathbf{H}_j$ is the core, we obtain the j-th centralized hypergraph, and to understand the relationship of the vertices, the optimization formula can be written as

$$\begin{aligned} \arg\min_{\mathbf{F}_j,\mathbf{W}_i} \left\{ \Omega_j^c(\mathbf{F}_j) + \lambda \mathscr{R}_{emp}(\mathbf{F}_j) + \mu \sum_i \sum_{e\in\mathscr{E}_i} \mathbf{W}_i(e)^2 \right\} \\ s.t.\ \mathbf{H}_i diag(\mathbf{W}_i) = diag(\mathbf{D}_i^v), diag(\mathbf{W}_i) \geq 0, \end{aligned} \quad (10.2)$$

where $\Omega_j^c(\mathbf{F}_j)$ is the regularizer to smooth out the correlations among vertices, $\mathscr{R}_{emp}$ represents the empirical loss, $\sum_i \sum_{e\in\mathscr{E}_i} \mathbf{W}_i(e)^2$ represents an l_2-norm regularizer, and $\mathbf{D}_i^v$ represents the degree matrix. By assigning different weights α_1, α_2 to core hypergraph and others, respectively, the regularizer term can be formulated as

$$\Omega_j^c(\mathbf{F}_j) = \alpha_1 \Omega_j(\mathbf{F}_j) + \sum_{i\neq j} \Omega_j(\mathbf{F}_i), \quad (10.3)$$

where $\Omega_j(\mathbf{F}_j)$ is equal to $\mathbf{F}_j^\top(\mathbf{I} - \Theta_i)\mathbf{F}_j$ with $\Theta_i = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$. Consequently, regularizer is rewritten as: $\Omega_j^c(\mathbf{F}_j) = \mathbf{F}_j^\top(\Delta_j^c)\mathbf{F}_j$ with $\Delta_j^c = \mathbf{I} - (\alpha_1\Theta_j + \alpha_2 \sum_{i\neq j} \Theta_i)$.

The optimization of Eq. (10.2) consists of two steps. In the following, we optimize the relevance matrix $\mathbf{F}_j$ with fixed $\mathbf{W}_i$ as

$$\arg\min_{\mathbf{F}_j} \left\{ \Omega_j^c(\mathbf{F}_j) + \lambda \mathscr{R}_{emp}(\mathbf{F}_j) \right\}, \quad (10.4)$$

which results in the closed-form answer for $\mathbf{F}_j = \frac{\lambda}{1+\lambda}(\mathbf{I} - \frac{1}{1+\lambda}(\alpha_1\Theta_j + \alpha_2 \sum_{i\neq j} \Theta_i))^{-1}\mathbf{Y}$. Following, we optimize the weight of hyperedges $\mathbf{W}_i$ with fixed $\mathbf{F}_j$ as

$$\begin{aligned} \arg\min_{\mathbf{W}_i} \left\{ \Omega_j^c(\mathbf{F}_j) + \mu \sum_i \sum_{e\in\mathscr{E}_i} \mathbf{W}_i(e)^2 \right\} \\ s.t.\ \mathbf{H}_i diag(\mathbf{W}_i) = diag(\mathbf{D}_i^v), diag(\mathbf{W}_i) \geq 0. \end{aligned} \quad (10.5)$$

which can be optimized by quadratic programming.

To best integrate data from various MRI, we generate the weights to every centralized hypergraph by minimizing the total hypergraph Laplacian, which is expressed as

$$\begin{aligned} &\arg\min_{\rho_i}\Big\{\sum \rho_i \Omega_i^c(\mathbf{F}_i) + \eta \sum \rho_i^2\Big\} \\ &s.t. \sum \rho_i = 1, \end{aligned}, \tag{10.6}$$

where ρ_i represents the weight of the i-th centralized hypergraph, and η represents the trade-off parameter of the Laplacian and l_2-norm regularizer. Determined by centralized hypergraph weights, the overall relevance matrix is $\mathbf{F} = \sum \rho_i \mathbf{F}_i$, of which the matching value can be used to categorize a subject.

In this subsection, we have introduced a centralized hypergraph learning method to model patient relationships for MCI identification. For each type of data, hypergraphs are constructed in the framework. In hypergraph learning, one hypergraph is chosen as the core hypergraph each time, and the remaining hypergraphs help the core hypergraph optimize the relevance matrix for prediction. The method not only takes into account the link among subjects, but it also makes use of a range of different types of data to increase the identification impact.

10.2.2 Medical Image Retrieval

Medical image retrieval is another crucial application of computer-aided diagnosis in Alzheimer's disease, along with the classification of patients with MCI or natural control introduced above. Its main goal is to offer clinicians with relevant MCI examples of visually comparable imaging data. Such data can also be provided to doctors in medical practice for instance thinking or scientific proof medicine.

Two primary stages help compensate for the MCI diagnosis-aided medical image retrieval technique [2], i.e., query about the class prediction for choosing candidates and ranking. The first stage involves finding the database's most relevant subjects based on the query subject. Such knowledge is then used to predict, under supervision, the query subject's category, i.e., the MCI patients or NC in this case. The graphs based on the pairwise object distance from various data modalities are combined into a multi-graph to predict the category of the query, after that every subject falling under the same category as the query is regarded as a potential subject. Second, the query subject and all of the candidate subjects are represented together in a new multi-graph. The learning process on the multi-graph reveals how related each candidate is to the query subject, allowing for ranking depending on the quality of similarity. The details of the two stages are shown in Fig. 10.2 [2] and explained below.

The query category is initially expected to use the subjects in the database given the query imaging data so that candidates can eventually be chosen based on the result. To analyze the similarity between the query subject and the training subjects chosen from the database, a graph $\mathscr{G}_i = \langle \mathscr{V}_i, \mathscr{E}_i, \mathbf{W}_i \rangle$ with $N + 1$ vertices is

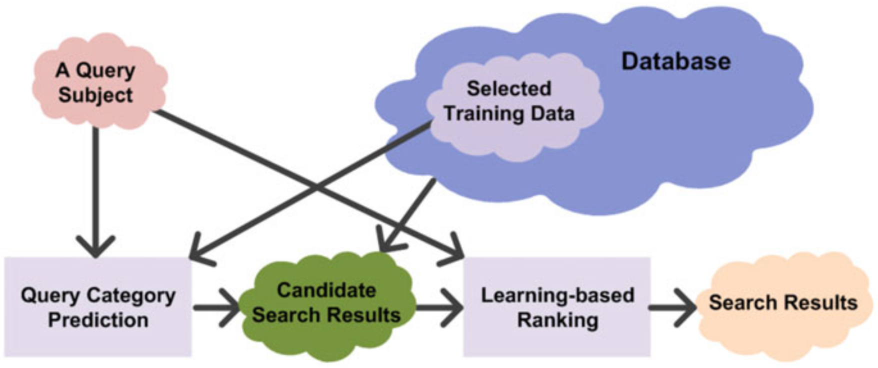


Fig. 10.2 The pipeline for medical image retrieval method. This figure is from [2]

generated for the imaging data of the i-th modality out of N_{mod} modalities. The weight $\mathbf{W}_i(v_s, v_t)$ of edge $\mathscr{E}_i(v_s, v_t)$, which connects the s-th and t-th vertices of the graph $\mathscr{G}_i$, is given by

$$\mathbf{W}_i(v_s, v_t) = \exp\left(\frac{d^2(v_s, v_t)}{\sigma_i^2}\right), \tag{10.7}$$

where $d(v_s, v_t)$ represents the Euclidean distance between vertices v_s and v_t in the feature space. Similar to the processing of identifying MCI, the optimization equation for the multi-graph learning task for query category prediction can be written as

$$\begin{aligned} &\arg\min_{\mathbf{F},\boldsymbol{\omega}} \left\{ \sum_{i=1}^{N_{mod}} \omega_i \Omega_i(\mathbf{F}) + \mu\mathscr{R}(\mathbf{F}) + \eta\|\boldsymbol{\omega}\|_2^2 \right\}, \\ &s.t. \sum_{i=1}^{N_{mod}} \omega_i = 1, \end{aligned} \tag{10.8}$$

where $\boldsymbol{\omega}$ and $\mathbf{F}$ represent the weighting parameters and the relevance matrix, respectively, μ, η represent the trade-off hyperparameters, $\mathscr{R}$ represents the empirical loss, and Ω_i represents the regularizer term defined as

$$\Omega_i = \frac{1}{2} \sum_{v_s, v_t} \mathbf{W}_i(v_s, v_t) \| \frac{\mathbf{F}(v_s, \cdot)}{\sqrt{\mathbf{D}_i(v_s, v_s)}} - \frac{\mathbf{F}(v_t, \cdot)}{\sqrt{\mathbf{D}_i(v_t, v_t)}} \|^2. \tag{10.9}$$

To solve the aforementioned optimization equation, **F** and $\boldsymbol{\omega}$ can be optimized alternatively. When $\boldsymbol{\omega}$ is fixed, the optimization equation for **F** is written as

$$\arg\min_{\mathbf{F}} \left\{ \sum_{i=1}^{N_{mod}} \omega_i \Omega_i(\mathbf{F}) + \mu \mathscr{R}(\mathbf{F}) \right\}, \tag{10.10}$$

which can be solved using the iterative process [9] formulated as

$$\mathbf{F}(t+1) = \frac{1}{\mu+1} \sum_{i=1}^{N_{mod}} \omega_i \Theta_i \mathbf{F}(t) + \frac{\mu}{\mu+1}\mathbf{Y}, \tag{10.11}$$

where $\mathbf{F}(t)$ is the t-th step of the iteration started out with $\mathbf{F}(0) = \mathbf{Y}$. When **F** is fixed, the optimization equation for $\boldsymbol{\omega}$ can be formulated as

$$\begin{aligned} &\arg\min_{\boldsymbol{\omega}} \left\{ \sum_{i=1}^{N_{mod}} \omega_i \Omega_i(\mathbf{F}) + \eta \|\boldsymbol{\omega}\|_2^2 \right\}, \\ &s.t. \sum_{i=1}^{N_{mod}} \omega_i = 1, \end{aligned} \tag{10.12}$$

which can be worked on by applying the Lagrangian method. All database subjects belonging to the same category are employed as candidate retrieval results based on the learned category of query subject.

Candidates are ranked for the retrieval of the most relevant subjects. Even though they are related to the same category of query subject, they may still differ from each other from the viewpoint of imaging appearance. Candidate subjects and query subjects construct graphs using each of N_{mod} modalities, where the i-th graph can be referred to $\hat{\mathscr{G}}_i$, in a manner similar to the previous classification step. Since the graph's weight $\boldsymbol{\omega}$ has been learned, the optimization equation can be written as

$$\arg\min_{\hat{\mathbf{f}}} \left\{ \sum_{i=1}^{N_{mod}} \omega_i \hat{\Omega}_i(\hat{\mathbf{f}}) + \hat{\lambda} \hat{\mathscr{R}}(\hat{\mathbf{f}}) \right\}, \tag{10.13}$$

where $\hat{\mathbf{f}}$ and $\hat{\Omega}$ represent the relevant vector and graph regularizer, respectively. $\hat{\mathscr{R}}$ is the empirical loss. The optimization task, such as Eq. (10.10), is handled using an iterative procedure, represented by

$$\hat{\mathbf{f}}(t+1) = \frac{1}{\hat{\lambda}+1} \sum_{i=1}^{N_{mod}} \omega_i \hat{\Theta}_i \hat{\mathbf{f}}(t) + \frac{\hat{\lambda}}{\hat{\lambda}+1}\hat{\mathbf{y}}. \tag{10.14}$$

The ranking of all candidates can be established by sorting based on the correlation given by $\hat{\mathbf{f}}$.

This subsection introduces the process of retrieving data relevant to the query subject from medical imaging datasets to support the diagnosis of MCI. The first stage selects the candidate set from the database, and the second stage computes the correlation between the query subject and all of the subjects in the candidate set and then ranks the retrieval based on the correlation. Both stages employ multi-graphs to describe the relationship between subjects, so as to facilitate retrieval tasks.

10.2.3 COVID-19 Identification Using CT Imaging

The COVID-19 pandemic, which has become the most widespread public health crisis since late 2019, is brought on by an extremely infectious virus and can induce multiple organ failures and server respiratory distress. Therefore, it is crucial to correctly distinguish COVID-19 from other forms of pneumonia to help correctly design pneumonia treatment programs. Nevertheless, the task is complex, as there are two main difficulties, namely noisy data resulting from the highly varied data gathered during crises, and confusing cases resulting from the similarity between COVID-19 and other types of pneumonia cases of the initial phases of symptoms.

Numerous investigations have demonstrated the usefulness of differentiating between COVID-19 and other types of pneumonia using CT, leading to the introduction of an uncertainty vertex-weighted hypergraph learning strategy to identify COVID-19 from other types of pneumonia using CT images [4]. It formulates data correlations among various instances to limit the interference by noisy data and confusing examples by employing an uncertainty rating quantification module and a vertex-weighted hypergraph structure. The framework introduction that follows is divided into three parts, namely pre-processing, measuring data uncertainty, and hypergraph construction and learning. Figure 10.3 depicts the overall illustration.

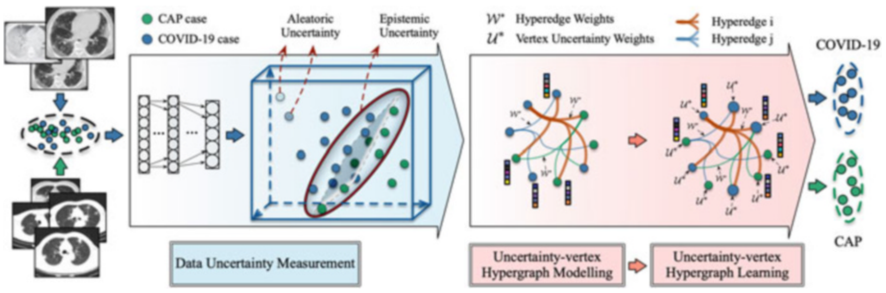

CAP case
COVID-19 case
Aleatoric Uncertainty
Epistemic Uncertainty
W* Hyperedge Weights
U* Vertex Uncertainty Weights
Hyperedge i
Hyperedge j
COVID-19
CAP
Data Uncertainty Measurement
Uncertainty-vertex Hypergraph Modelling
Uncertainty-vertex Hypergraph Learning

Fig. 10.3 An illustration of the uncertainty vertex-weighted hypergraph learning method for identifying COVID-19 among other types of pneumonia. This figure is from [4]

Regional features and radiomics features should be collected from the CT for every patient segregated using VB-Net [10] during the pre-processing stage. Regional features include the number of infected lesions and the surface area of the lesions, whereas textural features including the gray-level co-occurrence matrix are examples of radiomics features. The feature representation $\mathbf{X}$ of a patient's CT image is constructed by combining the two categories of features with information on age and gender.

Data uncertainty measurements are crucial in determining the dependability of various data throughout the learning process since noise can have an impact on data quality. The two types of uncertainty measurements are aleatoric and epistemic. The former one results from data abnormalities, noise, or other issues that lower the data quality, and the latter one is produced by the case's features being at the decision boundary. The goal of parameter estimation under aleatoric uncertainty is to minimize the KL divergence for both the actual and forecasted distributions, which can be represented by

$$\hat{\Theta} = \arg\min_{\Theta} \frac{1}{N} D_{KL}(P_D(\mathbf{X}_i) || P_\Theta(\mathbf{X}_i)), \tag{10.15}$$

where $P_D(\mathbf{X}_i)$, $P_\Theta(\mathbf{X}_i)$ represent the real distribution and the predicted distribution, respectively. By way of optimization, the loss function is expressed as

$$\mathscr{L}(\Theta) = \frac{1}{N} \sum_{i}^{N} \left(\frac{1}{2} \exp(-\alpha_\Theta(\mathbf{X}_i)) \mathbb{CE}\left(\mathbf{y}_i, f_\Theta(\mathbf{X}_i)\right) + \frac{1}{2} \alpha_\Theta(\mathbf{X}_i) \right), \tag{10.16}$$

where $\alpha_\Theta(\mathbf{X}_i)$ represents the log value of the estimated variance, and the aleatoric uncertainty defines as $A_\Theta(\mathbf{X}_i) = \exp(\alpha_\Theta(\mathbf{X}_i))$. Dropout can be used for inference to determine the epistemic uncertainty, which can be expressed as the model's inability to generate accurate predictions and is written as

$$\mathscr{E}(f_{\hat{\Theta}}(\mathbf{X}_i)) \approx \frac{1}{K} \sum_{k=1}^{K} f_{\hat{\Theta}(\omega^k)}(\mathbf{X}_i)^\top f_{\hat{\Theta}(\omega^k)}(\mathbf{X}_i) - \mathbf{E}(f_{\hat{\Theta}(\omega^k)}(\mathbf{X}_i))^\top \mathbf{E}(f_{\hat{\Theta}(\omega^k)}(\mathbf{X}_i)), \tag{10.17}$$

where ω represents the set of random variables and k represents the k-th test with dropout. Here, the overall uncertainty is $\mathscr{U}_{\hat{\Theta}}(\mathbf{X}_i) = A_{\hat{\Theta}}(\mathbf{X}_i) + \mathscr{E}(f_{\hat{\Theta}}(\mathbf{X}_i))$. With normalization, the final uncertainty can be formulated as

$$U_i = \sigma\left(\lambda \frac{\mathscr{U}_{\hat{\Theta}}(\mathbf{X}_i) - \mu_e}{\mathbf{s}_e}\right), \tag{10.18}$$

where μ_e and $\mathbf{s}_e$ represent the mean and the standard deviation of $\mathscr{U}$ and σ stands for the sigmoid function setting the output between 0 and 1.

Each instance is viewed as a vertex in the hypergraph that is constructed to mine high-order correlations among related patients for more precise prediction. Regional and radiomics features are used in the construction of hyperedges, respectively. In the regional features space, every vertex is regarded as a center vertex, and the nearest neighbor algorithm is used to link K nearest vertices to build a hyperedge. The similar method is applied to generate hyperedges using the radiomics feature. The uncertainty hypergraph, in contrast to the usual hypergraph, must take into account both the connection relationship and the vertex's uncertainty score, leading to a more comprehensive explanation of the incident matrix in uncertainty vertex hypergraph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{U} \rangle$ as

$$\mathbf{H}(v_j, e_i) = \begin{cases} U_j & \text{if } v_j \in e_i \\ 0 & \text{otherwise} \end{cases}. \tag{10.19}$$

The structure quantifies data uncertainty in comparison to conventional hypergraph learning strategies, and its optimization objective can be expressed as

$$\begin{cases} \mathcal{Q}_{\mathbf{U}}(\mathbf{F}) & = \arg\min_{\mathbf{F}}\{\Omega(\mathbf{F}) + \lambda \mathcal{R}_{emp}(\mathbf{F})\} \\ \Omega(\mathbf{F}, \mathcal{V}, \mathbf{U}, \mathcal{E}, \mathbf{W}) & = tr(\mathbf{F}^\top(\mathbf{U}^\top - \mathbf{U}^\top \Theta_{\mathbf{U}} \mathbf{U})\mathbf{F}) \\ \mathcal{R}_{emp}(\mathbf{F}, \mathbf{U}) & = \sum_{k=1}^{K} ||\mathbf{F}(:, k) - \mathbf{Y}(:, k)||^2 \end{cases}, \tag{10.20}$$

where $\Omega(\cdot)$ and $\mathcal{R}_{emp}(\cdot)$ represent the regular function and the empirical loss, respectively, and $\Theta_{\mathbf{U}}$ is equal to $\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$. It is reasonable to rewrite the empirical loss as

$$\mathcal{R}_{emp}(\mathbf{F}, \mathbf{U}) = tr(\mathbf{F}^\top\mathbf{U}^\top\mathbf{U}\mathbf{F} + \mathbf{Y}^\top\mathbf{U}^\top\mathbf{U}\mathbf{Y} - 2\mathbf{F}^\top\mathbf{U}^\top\mathbf{U}\mathbf{Y}). \tag{10.21}$$

The output matrix $\mathbf{F} \in \mathbb{R}^{n \times K}$ (K representing the number of classes, i.e., $K = 2$ in this case) is thus represented as

$$\mathbf{F} = \lambda(\mathbf{U}^\top - \mathbf{U}^\top\Theta_{\mathbf{U}}\mathbf{U} + \lambda\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{U}\mathbf{Y}. \tag{10.22}$$

New coming test cases can be classified as COVID-19 or other pneumonia types using the output label matrix established above.

10.2.4 ASD Identification Using Brain Functional Networks

Autism spectrum disorder (ASD) is a widespread developmental disorder that mostly affects children and has negative effects such as social communication impairments. Because of the rising cases, early identification and treatment of ASD are crucial in order to provide patients with new skills under clinical supervision. The diagnosis of ASD is mostly dependent on skilled specialists, and it is difficult

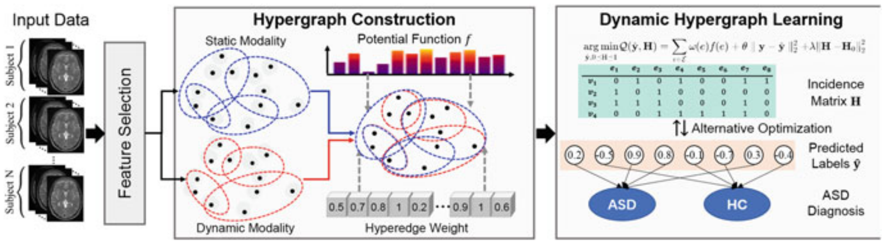


Fig. 10.4 A pipeline to classify ASD or healthy controls from brain functional networks data using dynamic hypergraph learning. This figure is from [3]

to identify ASD quickly due to the shortage of experts. The correlation of various functional connectivity (FC) pattern features in ASD patients can be used for rapid diagnosis.

The ASD identification method using brain functional networks [3] is divided into three stages, namely the selection of pre-processed features, hypergraph construction, and object identification using dynamic hypergraph learning. The overall process can be referred to Fig. 10.4. Static FC (sFC) and dynamic FC (dFC) are produced using a sliding window algorithm on the original functional magnetic resonance imaging time series in the first stage, and Lasso regression is then employed to accomplish the feature selection. The hypergraph construction stage creates a hypergraph based on the comparison of image features that represent data similarity in multiple modalities. Finally, ASD is identified using a multi-modal dynamic hypergraph learning technique that detects ASD and simultaneously improves the hypergraph structure.

The feature selection stage aims to discover valuable features in dFC and sFC sequences. The i-th subject's sFC sequence of τ time points is first separated into n sub-sequences, with the j-th sub-sequence of $\{j, n+j, 2n+j, \ldots\}$ time points. Defining $\bar{\mathbf{z}}_i^j$ as the dynamic FC feature of the j-th sub-sequence in subject i, the Lasso regression model, as the selection operator, can be expressed as

$$\arg\min_{\beta_0,\beta}\left(\frac{1}{2\tau'|\mathscr{P}|}\sum_{i\in\mathscr{P}}\sum_{j=1}^{\tau'}\left(y_i-\beta_0-\beta^\top\bar{\mathbf{z}}_i^j\right)^2+\mu|\beta|_1\right), \tag{10.23}$$

where $\tau' = \tau/n$ is the length of the sub-sequences, y_i represents the label of the subject, β is the regression coefficient, and μ stands for the trade-off hyperparameter. Features with zero coefficients are discarded, and the remaining are indicated as $\mathbf{z}_i^j$. Defining $\bar{\mathbf{x}}_i$ as the static FC feature of the i-th subject, the Lasso regression model is expressed as

$$\arg\min_{\gamma_0,\gamma}\left(\frac{1}{2|\mathscr{P}|}\sum_{i\in\mathscr{P}}\left(y_i-\gamma_0-\gamma^\top\bar{\mathbf{x}}_i\right)^2+\eta|\gamma|_1\right), \tag{10.24}$$

where y_i represents the label of the subject, γ is the regression coefficient, and η stands for the trade-off hyperparameter. Features with non-zero coefficients in the sFC selection operator represented as $\mathbf{x}_i$ are selected similarly to dFC.

The dFC sub-hypergraph $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and the sFC sub-hypergraph $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$, whose every vertex stands for a subject's sub-sequence, are combined to construct the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, i.e., $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. Since sFC features are subject level, the features of sFC sub-sequences inherit the subjects' static modality, i.e., $\mathbf{x}_i^j = \mathbf{x}_i$. Each vertex in each sub-hypergraph is regarded as a central vertex, and the nearest neighbor algorithm is employed to connect k neighbors ($k = 2n, 3n, \ldots, k_{max}n$) to create k_{max} hyperedges. When the two sub-hypergraphs are generated, the hypergraph is formed at the same time, and its incident matrix is expressed as

$$\mathbf{H}(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases}. \tag{10.25}$$

To enhance the structure of hypergraph and to help predict ASD, the potential equation of hyperedge can be defined as

$$f(e) = \sum_{u,v \in \mathcal{V}} \frac{\mathbf{H}(u, e)\mathbf{H}(v, e)g(u, v)}{(a + \alpha_1 + \alpha_2)\delta(e)}, \tag{10.26}$$

where

$$\begin{aligned} g(u, v) = \| &\frac{\hat{y}_u}{\sqrt{d(u)}} - \frac{\hat{y}_v}{\sqrt{d(v)}}\|_2^2 + \alpha_1 \|\frac{\mathbf{x}_u}{\sqrt{d(u)}} - \frac{\mathbf{x}_v}{\sqrt{d(v)}}\|_2^2 \\ &+ \alpha_2 \|\frac{\mathbf{z}_u}{\sqrt{d(u)}} - \frac{\mathbf{z}_v}{\sqrt{d(v)}}\|_2^2 \end{aligned}. \tag{10.27}$$

Here $\delta(e)$ represents the degree of hyperedge e, $\hat{y}_u$, $\hat{y}_v$ stand for to-be-learned labels of u, v, respectively, and α_1, α_2 are the trade-off hyperparameters. It is noted that the potential function determines the data distribution on the hyperedge jointly from sFC, dFC, and label space. The dynamic hypergraph learning cost function is formulated as

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{H}) = \sum_{e \in \mathcal{E}} \omega(e) f(e) + \theta \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 + \lambda \|\mathbf{H} - \mathbf{H}_0\|_2^2, \tag{10.28}$$

where $\omega(e)$ stands for the hyperedge's weight, $\mathbf{H}_0$ represents the initial hypergraph, and θ and λ are the trade-off hyperparameters, respectively. The objective function is shown to be divided into three terms: the first term is the loss function based on the hypergraph, and the following two terms are the empirical losses of $\hat{\mathbf{y}}$ and $\mathbf{H}$. The optimization of Eq. (10.28) consists of two stages. First, we optimize the to-be-learned labels $\hat{\mathbf{y}}$ with the fixed $\mathbf{H}$. The problem results in the closed-form solution as follows:

$$\hat{\mathbf{y}} = \left(\mathbf{I} + \frac{1}{\theta(1+\alpha_1+\alpha_2)\Delta}\right)^{-1}\mathbf{y}, \tag{10.29}$$

where $\Delta = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$. $\mathbf{I}$, $\mathbf{D}_v$, and $\mathbf{D}_e$ represent the identity matrix, vertex degree diagonal matrix, and hyperedge degree diagonal matrix, respectively. In the following, we optimize $\mathbf{H}$ with the fixed $\hat{\mathbf{y}}$ as

$$\mathscr{L}(\mathbf{H}) = \mathrm{tr}\Big((\mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2})\mathbf{K}\Big) + \lambda\|\mathbf{H}-\mathbf{H}_0\|_2^2, \tag{10.30}$$

where $\mathbf{K} = (\hat{\mathbf{y}}\hat{\mathbf{y}}^\top + \alpha_1\mathbf{X}\mathbf{X}^\top + \alpha_2\mathbf{Z}\mathbf{Z}^\top)/(1+\alpha_1+\alpha_2)$, which is optimized using the projected gradient method. Optimization can be done by the iterative procedure, formulated as

$$\begin{aligned}
\mathbf{H}_{k+1} &= \mathbf{P}[\mathbf{H}_k - h_k\nabla\mathscr{L}(\mathbf{H}_k)] \\
\nabla\mathscr{L}(\mathbf{H}) &= 2\lambda(\mathbf{H}-\mathbf{H}_0) + \mathbf{J}(\mathbf{I}\otimes\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{D}_v^{-1/2}\mathbf{H})\mathbf{W}\mathbf{D}_e^{-2} \\
&\quad + \mathbf{D}_v^{-3/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{J}\mathbf{W} \\
&\quad - 2\mathbf{D}_v^{-1/2}\mathbf{K}\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}
\end{aligned}, \tag{10.31}$$

where $\mathbf{J} = \mathbf{1}\mathbf{1}^\top$, h_k represents optimization step size of the k-th iteration, and $\mathbf{P}$ stands for the projection on the set $\{\mathbf{H}|0 \preceq \mathbf{H} \preceq 1\}$. When the iterative process converges, the labels of its sub-sequences are aggregated, and the result of prediction is the category with the highest score after aggregation.

In this section, we demonstrate the use of hypergraph-based approaches in four computer-aided diagnosis applications, namely MCI identification, medical image retrieval for MCI diagnostic assistance, COVID-19 identification, and ASD identification. Hypergraphs are employed in applications to represent high-order connections among subjects when mining complicated links among patients to gather knowledge than simply their images. In the future, it could be crucial to use hypergraphs to investigate few-shot learning approaches and transfer learning strategies in the domain of medical areas, such as MCI, COVID-19, and ASD.

10.3 Survival Prediction with Histopathological Image

Survival prediction is to model survival duration, which is the period that a patient is followed up on until a certain event, e.g., cancer recurrence or death. Survival prediction based on histopathological images is to predict the survival duration or survival risk to a satisfactory degree using only the patient's images, to estimate the severity, or to classify high and low risks, which guides the pathologist to evaluate the scenario. Since histopathological images typically include gigapixels, which are far more detailed than regular natural images, i.e., those in ImageNet [11] or MNIST [12], the main challenge of this work is how to reliably obtain the patient's feature representation for regression prediction analysis. Moreover, the relevant information for cells and tissues may not be readily extracted as it includes complex relationships and rich morphological structural content in histopathological images.

To overcome the challenge of the large number of pixels, there exists a technique [13] that randomly chooses patches in histopathological images with a variety of cells and without blank. It extracts patch features using a pre-trained CNN network and calculates survival risk using Lasso-Cox [14] regression. To enhance the patient representation, low-level patch features produced by a pre-trained CNN-based feature extractor are optimized by a graph convolutional neural network to construct the intricate relationship between patches [15]. The power of random patch selection to cover the details of the initial histopathological image and the lack of mutual information between patches limit the representation learning capabilities of the non-graph-based method, whereas the method that uses graph modeling applies pairwise correlations modeling to make up for the loss of structural information among cells with similar roles. Nevertheless, reducing complex high-order connections into pairwise relationships inevitably results in inaccurate modeling, losing data correlations among cells and tissues that are necessary to predict one's survival. Hence, the better solution is to model high-order data-associative representations employing hypergraph computational approaches to meet the challenges.

The following subsection explains how to use hypergraph computing in survival prediction based on histopathological images with two parts, namely ranking-based survival prediction [5] and phenotypic and topological hypergraphs-based survival prediction [6]. In the first part, a nearest-neighbor-based hypergraph modeling methodology is introduced, and optimization is achieved using a ranking-based method. In the second part, the hypergraphs are created in the image space and merged for prediction.

10.3.1 Ranking-Based Survival Prediction

This part describes the three stages required for executing the ranking-based survival prediction task via hypergraph representation [5], namely pre-processing before generating hypergraph, learning hypergraph representation, and survival ranking prediction, as illustrated in Fig. 10.5. It is worth noting that these three components are related to the framework of the graph-based survival prediction task in general, not just the rank-based survival hypergraph framework.

In the pre-processing stage, N patches are randomly chosen from each histopathological image, and each patch has the same size as a typical natural image (e.g., $224px \times 224px$). Directly choosing patches at random from the original image, however, likely picks up the noisy region as well (e.g., erosion and blank). Therefore, before randomization, the OTSU algorithm [16] is applied to

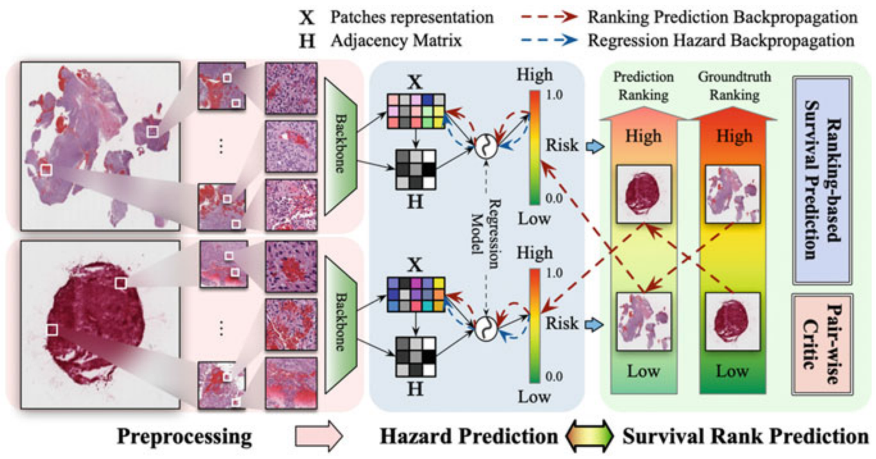


Fig. 10.5 A pipeline of ranking-based survival prediction utilizing hypergraph representation, including pre-processing, hazarding prediction via hypergraph representation, and ranking-based survival risk prediction. This figure is from [5]

segregate cell tissue samples with rich information. Next, the foremost patch-level image optical structure features $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times F}$ are extracted by a pre-trained deep neural network from ImageNet [11], where F represents the dimension of each patch feature. Image features, which are appropriate for the strata of complex tissue patterns, are included in the raw features that are retrieved from the pre-trained model and reflect the cells and tissues that are present in the patch.

Following pre-processing to extract feature information at the patch level, the hypergraph computing approach is used to produce the features representing the histopathological image level for the subsequent prediction of the survival risk score. Hypergraphs are created using the distance-based hypergraph generation method since intuitive cells and tissues with similar morphologies have comparable functionalities. Each patch is regarded as a vertex, and each vertex is considered as the center vertex to generate a hyperedge. This results in a total of N nodes and N hyperedges in the hypergraph reflecting the structural information of the histopathological image. We build hyperedges using the k nearest neighbor approach, which connects k vertices with the closest Euclidean distance between raw features from its center vertex. Therefore, the hypergraph incident matrix $\mathbf{H}$ is obtained. Beyond pairwise graph structures, hierarchical grouping patterns can be discovered using a hyperedge structure that creates a channel for the transfer and integration of information from the k nearest morphological patches. The information fusion among patch vertex is then accomplished using hypergraph convolutional layers, as shown below:

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\Theta^{(l)}\right), \tag{10.32}$$

where $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times C_l}$ is the l-th layer convolution input feature with N vertices and C_l dimensions, $\mathbf{X}^{(l+1)}$ is the l-th layer convolution output feature, σ stands for nonlinear activation function, and the l-th layer's learnable parameters are represented by $\Theta^{(l)}$. The output $\mathbf{X}^{(L+1)}$ of the last layer is used to forecast survival duration after L layers of convolution, where N hyperedges might reflect N patterns of causal variables. The predicted survival risk score is regressed using a fully connected neural network after $\mathbf{X}^{(L+1)}$ is squeezed into $\mathbf{X} \in \mathbb{R}^{1 \times C_{L+1}}$ via the pooling layer representing patient's representation. The patient's actual survival time t can be used to supervise the backpropagation process of the regression.

Ranking information, which can be used to infer the conditions of nearby patients, is also significant in regression tasks in addition to the specific survival duration of every single patient. Moreover, the ranking data accurately portray patients' ranks for high and low risks. The prediction of survival ranking is introduced at the final, most significant, and enlightening stage. Pairs of histopathological images (i.e., pairs of patients) should be taken into consideration since models are trained on a single image currently, and the inability to distinguish the relative risks of two similar instances is the most frequent reason for inaccurate patient risk comparisons. To fine-tune the model parameters and enhance the accuracy of the model's forecast ranking, a Bayesian-based method known as Bayesian Concordance Readjust (BCR) is presented. The BCR loss function, which is employed in pairwise training of histopathological images, embodies the Bayesian Concordance Readjust and can be formulated as follows:

$$\mathscr{L} = -\log\Big(\delta(\mathbb{W} \cdot (\mathbf{X}_i - \mathbf{X}_j))\Big), \tag{10.33}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ stand for the feature representation of patients i and j, respectively, and $\mathbb{W}$ represents the learnable parameters of regression.

In this subsection, we provide a ranking-based survival prediction method for predicting a patient's survival hazard score from a single WSI image. The method first extracts informative patches from WSI images and then applies a hypergraph to describe the correlations among patches to create overall features of WSI. Finally, the method considers relative ranking information among various patients and achieves greater prediction results.

10.3.2 Phenotypic and Topological Hypergraph Modeling

The hypergraph for mining high-order correlations in the data is essential for accurately generating feature representation of histopathological images. We can notice that the previously presented ranking-based survival prediction method only employs the nearest neighbor generation method when constructing a hypergraph. This method only fine-tunes image features among patches with similar features and mines high-order relationships from one single perspective, which tends to leave

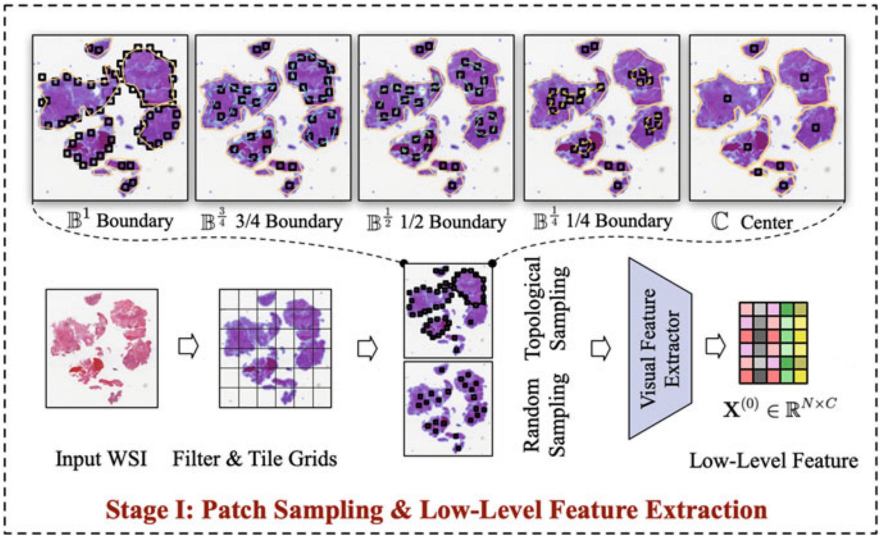

$\mathbb{B}^1$ Boundary
$\mathbb{B}^{\frac{3}{4}}$ 3/4 Boundary
$\mathbb{B}^{\frac{1}{2}}$ 1/2 Boundary
$\mathbb{B}^{\frac{1}{4}}$ 1/4 Boundary
$\mathbb{C}$ Center
Input WSI
Filter & Tile Grids
Topological Sampling
Random Sampling
Visual Feature Extractor
$\mathbf{X}^{(0)} \in \mathbb{R}^{N \times C}$
Low-Level Feature
Stage I: Patch Sampling & Low-Level Feature Extraction

Fig. 10.6 Patch sampling and low-level feature extraction. This figure is from [6]

other informative high-order relationships out. Therefore, here we describe a multi-hypergraph-based learning method for survival prediction [6], which efficiently achieves a high-order global representation of the histopathological image by using a variety of edges correlation modeling in several spaces and a basic hypergraph convolutional network.

The goal of multi-hypergraph modeling is to uncover topological linkages among patches in image space and high-order connections among patches in latent feature space. The random sampling approach previously employed cannot be used since it is essential to analyze the topological connections of the image space; instead, the sampling is carried out according to the position of the patch in the original image. Therefore, the sampling process uses a boundary-to-center strategy (shown in Fig. 10.6) after the OSTU algorithm [16] filters noisy regions to produce informative regions of interest. In addition to selecting the border $\mathbb{B}^1$ and the center $\mathbb{C}$ of regions of interest, patches are chosen based on various distance radios of $\frac{3}{4}$, $\frac{1}{2}$, and $\frac{1}{4}$, i.e., $\mathbb{B}^{\frac{3}{4}}$, $\mathbb{B}^{\frac{1}{2}}$, and $\mathbb{B}^{\frac{1}{4}}$ in Fig. 10.6 from boundary to the center. Patches with the same percentage of the distance from the border in the same region of interest and centers among regions can be taken up as correlating in the image space.

A multi-hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed by joining two sub-hypergraphs, namely a phenotypic sub-hypergraph $\mathcal{G}_{phe} = (\mathcal{V}, \mathcal{E}_{phe})$ created from the latent feature space and a topological sub-hypergraph $\mathcal{G}_{top} = (\mathcal{V}, \mathcal{E}_{top})$ generated from image space, i.e., $\mathcal{E} = \mathcal{E}_{phe} \cup \mathcal{E}_{top}$, as shown in Fig. 10.7. Based on the Euclidean distances between extracted patch visual features, as explained in the previous method, the incident matrix of the phenotypic sub-hypergraph $\mathbf{H}_{phe}$ is built using the k nearest neighbor method. In the incident matrix of the topological sub-

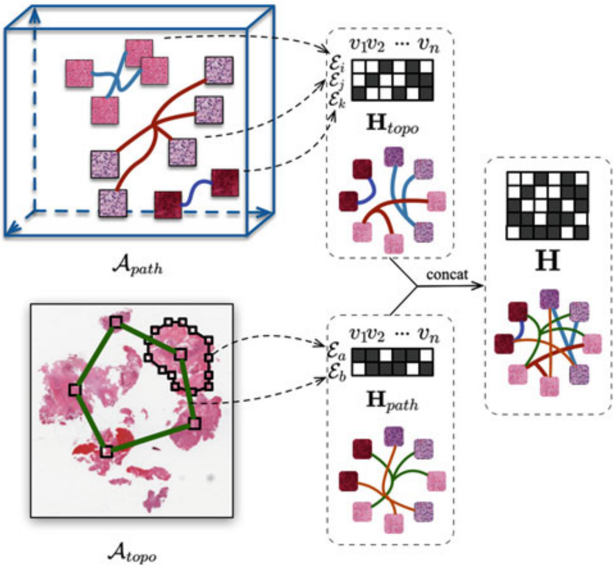


Fig. 10.7 Construction of multi-hypergraph, which contains a phenotypic sub-hypergraph and a topological sub-hypergraph. This figure is from [6]

hypergraph $\mathbf{H}_{top}$, each vertex is linked to its neighbors in the topological space, i.e., the centers of all regions of interest, $\mathbb{B}^{\frac{1}{4}}$, $\mathbb{B}^{\frac{1}{2}}$, $\mathbb{B}^{\frac{3}{4}}$, and the boundaries of each region of interest.

The standard hypergraph neural network is modified to the hypergraph max-mask convolution with an increased number of hyperedges, which can address the overfitting issue brought up by a lack of training data. Each layer's convolutional process consists of four steps, namely hyperedge feature gathering, max-mask operation, vertex feature aggregating, and vertex feature re-weighting.

The features of each hyperedge $\mathscr{F}_e^{(l)}$ are gathered during the first step from the vertices that are directly linked to it, which can be written as a product of $\mathbf{H}$ and $\mathbf{X}^{(l)}$. The hyperedge features $\mathscr{F}_e^{(l+1)}$ of the convolutional layer are then produced by performing a max-mask operation on the features excluding λ dominating hyperedges. In the final two steps, the output vertex features $\widetilde{\mathscr{F}}_v^{(l+1)}$ are obtained by aggregating the hyperedge features by multiplying matrix $\mathbf{H}^\top$ and re-weighting them using a learnable parameter $\Theta^{(l)}$, respectively. Therefore, the whole steps of each layer of the hypergraph neural network in the framework are formulated as

$$\begin{cases} \mathbf{X}^{(l+1)} &= \sigma\left[((\mathbf{I}-\mathbf{L})\mathbf{X}^{(l)} + \mathbf{H}^{-1}(\mathbf{I}-\mathbf{L})\mathbf{X}^{(\lambda)})\Theta^{(l)}\right] \\ \mathscr{F}_e^{(l+1)} &= \mathbf{H}^{-1}(\mathbf{I}-\mathbf{L})\mathbf{X}^{(l)} + \mathbf{X}^{(\lambda)} \end{cases}, \tag{10.34}$$

where $\mathbf{X}^{(\lambda)}$ stands for an offset matrix containing only the data from the dominant λ hyperedges, and $\mathbf{H}^{-1}(\mathbf{I}-\mathbf{L})\mathbf{X}^{(\lambda)}$ ensures the computing gradients and adjusting vertex features have no impact on the top λ hyperedges.

With two learnable weight vectors, the vertex feature matrix $\mathbf{X}^{(L+1)}$ and the hyperedge matrix $\mathscr{F}_e^{(L+1)}$ of the final layer are squeezed into feature vectors. The feature fusion module then merges the two vectors to establish a global feature representation that represents the entire hypergraph, i.e., the histopathological image for the regression task.

In this subsection, we introduce a general framework and a ranking-based optimization method for the task of survival prediction using histopathological images. The survival prediction challenges are then addressed by replacing a single nearest neighbor modeling algorithm with the multiple hypergraphs modeling method. The transformer network is a commonly used model of long-term sequential data, while histopathological images also include a significant quantity of sequential topological histopathological information, making it conceivable to incorporate transformer to the survival prediction task. Therefore, in future works, we can attempt to include transformer into the framework's feature extraction or the construction of hypergraphs component.

10.4 Drug Discovery

Predicting drug–target interactions (DTIs) is a critical step in the process of discovering new drugs to treat diseases. Nevertheless, the commonly used biochemical experimental methods in wet laboratories are always costly and tedious. The development of drug discovery computational methods, of which machine learning based methods are one of the most promising, has been prompted by the growing need for low-cost, effective, and efficient DTI prediction methods. The core idea of these methods is that similar targets may be linked with similar drugs, and for the drug the assumption is symmetric. This assumption *defacto* implies the potential high-order associations between drugs and targets, especially when considering the complex heterogeneous biological networks that contain different biological entities such as proteins.

In the DTI network, one single drug may interact with a group of targets, which can be generalized as a "one-to-many" pattern. When it comes to the aforementioned heterogeneous biological networks, the interactions between these biological entities become more complex, emerging as the "many-to-many" pattern. The hypergraph structure, which can naturally model high-order correlations owing to its flexible hyperedge, is suitable for modeling such a complex heterogeneous biological network. It can conveniently incorporate multiple complex interactions between different biological entities and further utilize the hypergraph computing technique to learn the correlations.

In this section, we present a heterogeneous hypergraph learning method for the DTI prediction (HHDTI) task [7]. The overall pipeline of the framework is illustrated in Fig. 10.8. It takes into consideration different types of interactions between biological entities (e.g., drug–target, drug–disease, and target–disease interactions) to facilitate DTI predictions.

(1) Heterogeneous Hypergraph Modeling
The overall procedure for modeling biological networks into a heterogeneous hypergraph is illustrated in Fig. 10.9. Given a heterogeneous biological network with different kinds of biological entities and interactions among these entities, the goal of hypergraph modeling is to characterize the heterogeneous biological network into a heterogeneous hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here $\mathcal{V} = \{\mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_o\}$ indicates the vertex set, and $\mathcal{E} = \{\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_r\}$ is the hyperedge set. o and r are the number of types for entities and interactions, respectively. Specifically, we have $\mathcal{V}_o = \{v_1, v_2, \ldots, v_{M_o}\}$ with M_o vertices and $\mathcal{E}_r = \{e_1, e_2, \ldots, e_{N_r}\}$ with N_r hyperedges.

In the heterogeneous biological network discussed here, the set of entity types O contains drug, target, and disease. The set of interaction types R includes dr–ta, ta–dr, dr–di, and ta–di interactions.[1] Therefore, o is equal to 3 and r is equal to 4.

Moreover, multiple sub-hypergraphs with one sub-hypergraph corresponding to one type of correlation on the basis of the overall heterogeneous hypergraph can be constructed. Therefore, four sub-hypergraphs are acquired in all, i.e., four incidence matrices, which are denoted as $\mathbf{H} \in \mathbb{R}^{M \times N_j}$, $j \in [1, r]$ and M is the number of two types of vertices corresponding to the correlation. Specifically, the four incidence matrices generated based on R are defined as $(\mathbf{H}_{dr-ta}, \mathbf{H}_{ta-dr}, \mathbf{H}_{dr-di}, \mathbf{H}_{ta-di})$. Figure 10.10 shows an example of a drug hypergraph.

(2) Drug and Target Embedding Learning
The same framework is used to create the overall embeddings for both drugs and targets. We now briefly introduce how this framework learns drug and target embeddings.

The overall embeddings are acquired by combining the main embeddings and the assisted embeddings. Particularly, the primarily vectorized representations for all drugs and targets are provided by the main embeddings, which are learned using direct DTIs. Contrarily, the assisted embeddings offer supplementary information discovered through disease-relevant data, such as dr–di and ta–di connections.

We first take a drug as an example to demonstrate the learning framework. The drug's main embeddings $\boldsymbol{\Phi}_d^k$ are learned from $\mathbf{H}_{dr-ta}$ using an unsupervised Bayesian deep generative model, i.e., hypergraph variational auto-encoder, while the drug assisted embeddings are generated from $\mathbf{H}_{dr-di}$ by leveraging the hypergraph neural networks (HGNN) [17]. For the main embeddings learning, given the DTI sub-hypergraph structure $\mathbf{H}_{dr-ta}$, the Bayesian deep generative model serves as a

[1] dr, ta, di are abbreviations of drug, target, and disease, respectively.

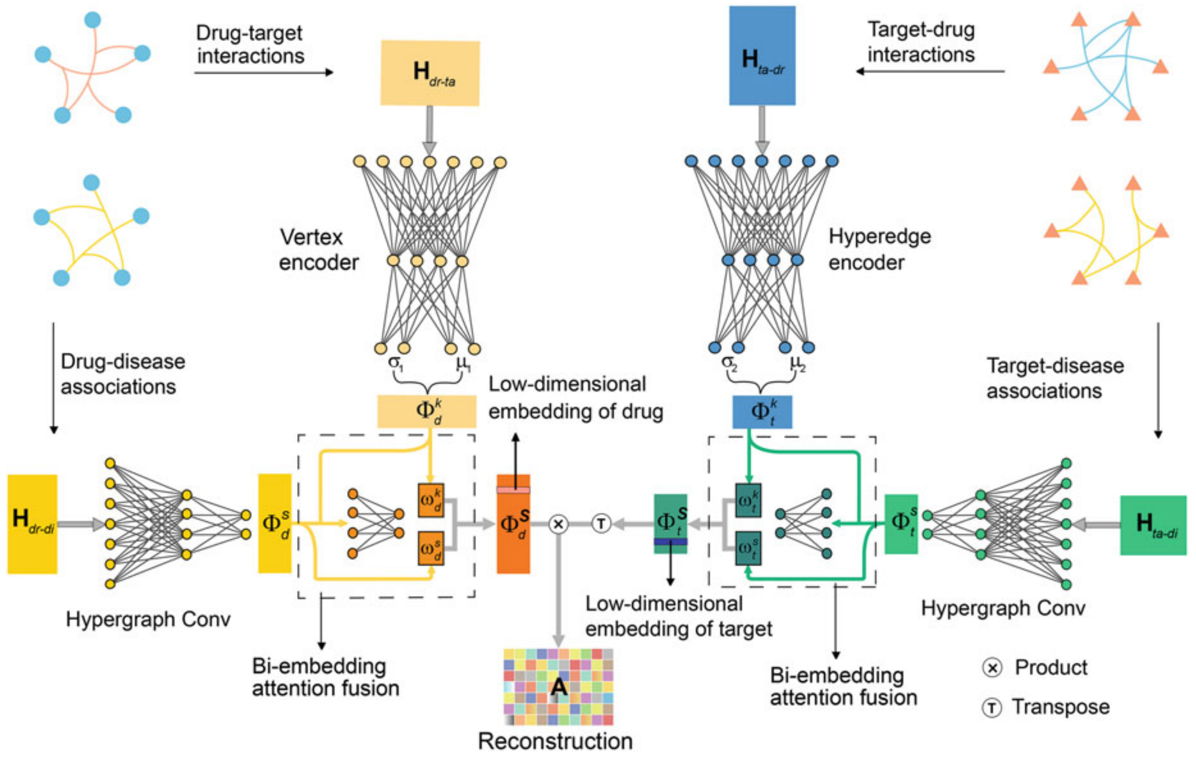


Fig. 10.8 An illustration of the HHDTI framework. This figure is from [7]

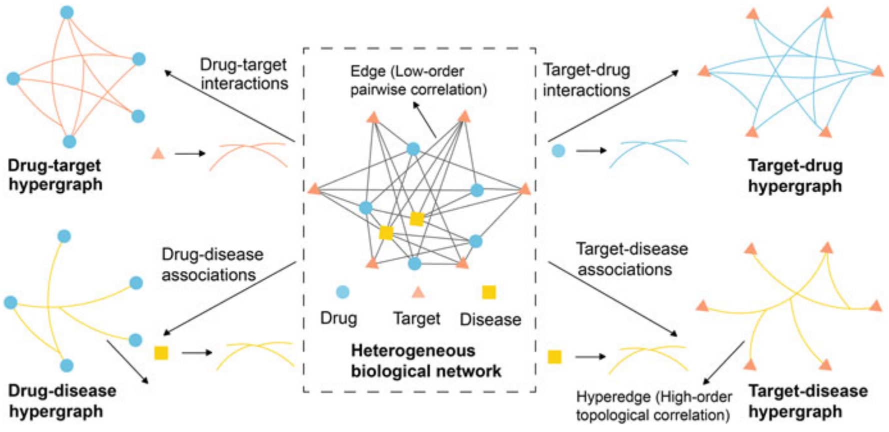

Drug-target interactions
Edge (Low-order pairwise correlation)
Target-drug interactions
Drug-target hypergraph
Target-drug hypergraph
Drug-disease associations
Target-disease associations
Drug
Target
Disease
Heterogeneous biological network
Drug-disease hypergraph
Hyperedge (High-order topological correlation)
Target-disease hypergraph

Fig. 10.9 The overall procedure for modeling biological networks into a heterogeneous hypergraph. This figure is from [7]

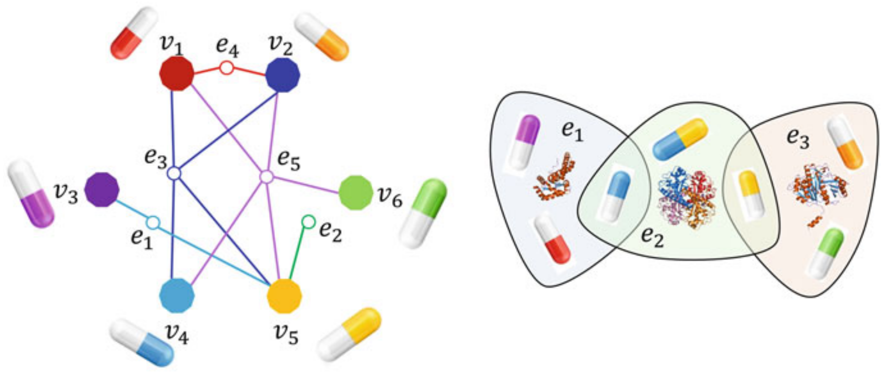


Fig. 10.10 An example of a drug hypergraph. Each vertex on the hypergraph represents a drug, and each hyperedge connects all the drugs that share the same target

vertex encoder [18] to explore the potential associations between drugs linked with one target. This method conducts a nonlinear mapping to transform the hypergraph structure $\mathbf{H}_{dr-ta}$ from the observed space into the shared space $\boldsymbol{\Phi}'_{dr-ta}$ as

$$\boldsymbol{\Phi}'_{dr-ta} = f\left(\mathbf{H}_{dr-ta}\mathbf{W}_{dr-ta} + \mathbf{b}_{dr-ta}\right), \tag{10.35}$$

where the activation function $f(\cdot)$ is nonlinear.

The hyperbolic tangent $tanh(x)(exp(x) - exp(-x)/exp(x) + exp(-x)$ is used here because of its analytic form and efficiency. Learnable weight and bias are represented by $\mathbf{W}_{dr-ta} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ and the $\mathbf{b}_{dr-ta} \in \mathbb{R}^{D_{\text{out}}}$. D_{in} and D_{out} are the corresponding dimensions of $\mathbf{H}_{dr-ta}$ and $\boldsymbol{\Phi}'_{dr-ta}$, respectively. Following the acquisition of the $\boldsymbol{\Phi}'_{dr-ta}$, two fully connected layers are used to estimate the mean and variance:

$$\boldsymbol{\mu}_{dr-ta} = f\left(\boldsymbol{\Phi}'_{dr-ta}\mathbf{W}^{\mu}_{dr-ta} + \mathbf{b}^{\mu}_{dr-ta}\right) \tag{10.36}$$

and

$$\boldsymbol{\sigma}_{dr-ta} = f\left(\boldsymbol{\Phi}'_{dr-ta}\mathbf{W}^{\sigma}_{dr-ta} + \mathbf{b}^{\sigma}_{dr-ta}\right), \tag{10.37}$$

where $\mathbf{W}^{\mu}_{dr-ta}, \mathbf{W}^{\sigma}_{dr-ta} \in \mathbb{R}^{D_{\text{out}} \times D}$ and $\mathbf{b}^{\mu}_{dr-ta}, \mathbf{b}^{\sigma}_{dr-ta} \in \mathbb{R}^{D}$ has been indicated before. The main embeddings $\boldsymbol{\Phi}^{k}_{d}$ are then sampled by

$$\boldsymbol{\Phi}^{k}_{d} = \boldsymbol{\mu}_{dr-ta} + \boldsymbol{\sigma}_{dr-ta} \odot \boldsymbol{\varepsilon}, \tag{10.38}$$

where $\odot$ is the Hadamard product and $\varepsilon \sim N(0, I)$.

In this way, the high-order structural correlations from the direct DTIs can be captured by the major embeddings. In addition to such straightforward interactions, other types of interactions can also contribute to DTI prediction, which has been validated by recent studies [19]. For instance, phenotypic side effects can be determined by how similar they are if these two drugs share a target [20, 21]. It has been verified in the literature that reported that targets can be used as a connection between drugs and illnesses [22]. Enlightened by these discoveries, auxiliary data are integrated into HHDTI, which can provide complementary information so as to improve prediction accuracy and treat extreme cases such as the cold-start problem (only a few DTIs can be fetched).

Specifically, the *dr–di* and *ta–di* correlations are considered here in HHDTI, and the embeddings learned from the corresponding dr–di incidence matrices $\mathbf{H}_{dr-di}$ are called drug assisted embeddings, which serve as the auxiliary representation for the drug's main embeddings. The drug assisted embeddings are learned by the HGNN model [17], with which the high-order correlations are encoded as

$$\text{Convh}(\mathbf{H}, \mathbf{X} \mid \mathbf{W}) = f\left(\left(\mathbf{D}^{v}\right)^{-1/2}\mathbf{H}\left(\mathbf{D}^{e}\right)^{-1}\mathbf{H}^{\top}\left(\mathbf{D}^{v}\right)^{-1/2}\mathbf{X}\mathbf{W}\right), \tag{10.39}$$

where $\mathbf{D}^{\boldsymbol{v}}$ and $\mathbf{D}^{\boldsymbol{e}}$ are the degree matrices of vertex and hyperedge, respectively. The corresponding degree of vertex and hyperedge are $\left(\mathbf{D}^{V}\right)_{k,k} = \sum_{j=1}^{L}\mathbf{H}^{k,j}$ and $(\mathbf{D}^{e})_{j,j} = \sum_{k=1}^{N}\mathbf{H}^{k,j}$, respectively. The matrix $\mathbf{W}$ is the learnable weight parameter, and $(\cdot)^{\top}$ is the transposition operator. Specifically, the convolutional layer used to learn the drug assisted embedding Φ^{s}_{d} can be formulated as

$$\boldsymbol{\Phi}^{s(l)}_{d} = \text{Convh}\left(\mathbf{H}_{dr-di}, \boldsymbol{\Phi}^{s(l-1)}_{d} \mid \mathbf{W}^{(I-1)}\right), \tag{10.40}$$

where $\Phi^{s(l-1)}_{d}$, $\Phi^{s(I)}_{d}$, and $\mathbf{W}^{(I-1)}$ represent the $(l-1)$-th layer's input, output, and trainable weight matrix, respectively. Here, the identity matrix is set as the initial value for $\mathbf{X}$. That is, we have $\boldsymbol{\Phi}_d^{s(0)} = \mathbf{X} = \mathbf{I}$. To create the overall embeddings, an attention module is used to combine the main embeddings and assisted embeddings into a single shared space. By determining the coefficients ω^i, the bi-embedding attention fusion process is specifically employed to give various weights to the main embeddings and assisted embeddings:

$$\omega^i = \frac{\exp\left(f\left(\boldsymbol{\Phi}^i \mathbf{W}^i + \mathbf{b}^i\right) \cdot \mathbf{P}^i\right)}{\sum_{j \in k,s} \exp\left(f\left(\boldsymbol{\Phi}^i \mathbf{W}^j + \mathbf{b}^i\right) \cdot \mathbf{P}^i\right)}, \tag{10.41}$$

where $\mathbf{W}^i \in \mathbb{R}^{D \times D'}$, $\mathbf{b}^i \in \mathbb{R}^{D'}$, and $\mathbf{P}^i \in \mathbb{R}^{D' \times 1}$ are trainable parameters. D and D' are the corresponding dimensions. The overall drug embeddings Φ^S can then be obtained by

$$\Phi_d^S = \omega^k \Phi_d^k + \omega^s \Phi_d^s. \tag{10.42}$$

The overall embeddings of targets Φ_d^S are generated similarly. The main difference lies in that here the $\mathbf{H}_{ta-dr}$ and $\mathbf{H}_{ta-di}$ are used as inputs. The target main embeddings Φ_t^k are learned using the same vertex encoder as that of drugs. The HGNN model is also adopted to yield the target assisted embeddings Φ_t^s from the target–disease association hypergraph. Finally, the embedding attention fusion is run to achieve the overall target embeddings Φ_t^S.

(3) Drug–Target Interactions Prediction

The likelihood of the drug and the target embeddings is calculated to create the reconstruction space $\boldsymbol{A}$, from which the DTI predictions are generated. That is, we have

$$\mathbf{A} = \text{Sigmoid}\left(\boldsymbol{\Phi}_d^S \left(\boldsymbol{\Phi}_t^S\right)^\top\right), \tag{10.43}$$

where $Sigmoid(\cdot)$ is the sigmoid function. We then give the variational lower bound $\mathscr{L}$, which is optimized by

$$\begin{aligned}\mathscr{L} = \mathbb{E}_q\left[\log p\left(\mathbf{A} \mid \boldsymbol{\Phi}_d^{\mathrm{s}}, \boldsymbol{\Phi}_t^{\mathrm{S}}\right)\right] - \beta\left(\mathrm{KL}\left(q\left(\boldsymbol{\Phi}_d^k \mid \mathbf{A}\right) \| p\left(\boldsymbol{\Phi}_d^k\right)\right)\right. \\ \left. + \mathrm{KL}\left[q\left(\boldsymbol{\Phi}_t^k \mid \mathbf{A}\right) \| p\left(\boldsymbol{\Phi}_t^k\right)\right]\right),\end{aligned} \tag{10.44}$$

where $\mathrm{KL}[q(\cdot)||p(\cdot)]$ is the metric from distribution $q(\cdot)$ to $p(\cdot)$ in Kullback–Leibler divergence space. Varying b provides different acquired representations by changing the amount of learning pressure provided during training. Inspired by the variational auto-encoder, Gaussian priors $p\left(\boldsymbol{\Phi}_d^k\right) = \prod_i p\left(\varphi_i^d\right) = \prod_i \mathcal{N}\left(\varphi_i^d \mid 0, \mathbf{I}\right)$ and $p\left(\boldsymbol{\Phi}_t^k\right) = \prod_j p\left(\varphi_j^t\right) = \prod_j \mathcal{N}\left(\varphi_j^t \mid 0, \mathbf{I}\right)$ can be taken into consideration. Here, $\mathbb{E}_q[\log p(\cdot \mid \cdot)]$ is the likelihood of reconstruction space $\boldsymbol{A}$.

In this part, we introduce a general hypergraph-based framework for DTI predictions. It is noted that the introduced framework introduced here is neither restricted to these types of complex interactions nor the DTI prediction task here; other types of interactions that may contribute to the DTI prediction task or even other projects containing complex correlations are also thinkable.

In real-world applications, the annotations for such biomedical data are computationally expensive and time-consuming. Therefore, self-supervised learning has received a lot of attention recently since it can mine useful information from the data in an unsupervised way. Under such circumstances, it is of great significance to further devise the self-supervised hypergraph computation for DTI predictions.

10.5 Medical Image Segmentation

In the field of medical imaging, hypergraph-based image segmentation methods also play a crucial role, where there are limitations of traditional multi-atlas segmentation (MAS) methods in segmenting anatomical structures with poor image contrast. The hypergraph can be used. The hypergraph can model complex subject-within and subject-to-atlas image voxel relationships and propagate label on atlas image to target subject images.

This method is named hierarchical hypergraph patch labeling (HHPL) [8], which characterizes higher-order associations between context features by constructing a hypergraph, and transforms hypergraph learning into a hierarchical model. At the same time, a dynamic label propagation strategy is used to augment reliably identified labels from subject images to help predict labels.

As shown in Fig. 10.11, pairwise relations and complex higher-order associations in hyperedges are compared when using the MAS method, where p_i is the subject image voxel, and $R_i(l)$ is defined as a 3-D cube of side length l centered on p_i. Image patches are extracted using the target object image at voxel p_i and the registration atlas image within the corresponding local neighborhood $R_{n,i}(l)$. Hyperedges can be constructed similarly between the atlas image voxels and target subject image voxels with the high-level context features from the label probability map.

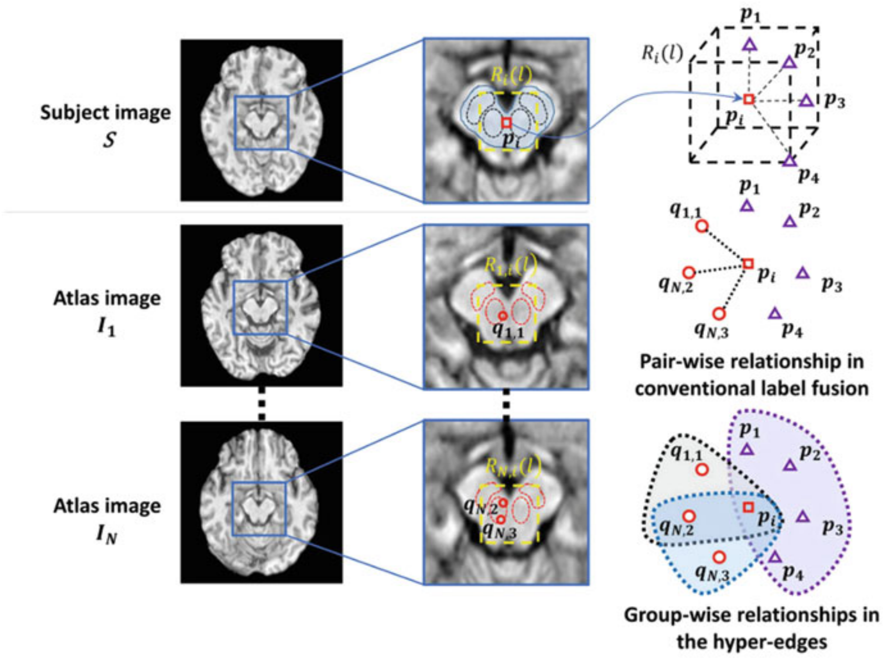


Fig. 10.11 Comparison of a simple pairwise relationship in the conventional MAS methods and the complex groupwise relationship in hyperedges (with much richer information). This figure is from [8]

In particular, the subject vertices under the label and the related atlas vertices with known labels affect the labels on the target topic vertex. The label propagation process follows two principles: (1) if vertices are grouped in the same hyperedge, they have the same anatomical label. (2) The label difference between vertices with known labels before and after label propagation is to be as small as possible. Therefore, the objective function of hypergraph learning is defined as follows:

$$\arg\min_{\mathbf{f}} \left\{ \|\mathbf{y}-\mathbf{f}\|_2^2 + \lambda \cdot \Phi\left(\mathbf{f}, \mathbf{H}, \mathbf{W}, \mathbf{D}_e, \mathbf{D}_0\right) \right\}. \tag{10.45}$$

The first term is the control to minimize the difference between the initialization label vector **y** and the prediction vector **f**. The second term is the graph balance term defined as

$$\Phi\left(\mathbf{f}, \mathbf{H}, \mathbf{W}, \mathbf{D}_e, \mathbf{D}_v\right) = \frac{1}{2} \sum_{e\in\varepsilon} \sum_{v,v'\subseteq e} \frac{w(e)h(v,e)h\left(v',e\right)}{\delta(e)} \left(\frac{f(v)}{\sqrt{d(v)}} - \frac{f\left(v'\right)}{\sqrt{d\left(v'\right)}} \right)^2. \tag{10.46}$$

We can determine the optimal $\hat{\mathbf{f}}$ by differentiating the objective function with respect to $\mathbf{f}$:

$$\hat{\mathbf{f}} = (\mathbf{I} + \lambda(\mathbf{I} - \boldsymbol{\Theta}))^{-1}\mathbf{y}. \tag{10.47}$$

Having obtained the optimized $\hat{\mathbf{f}}$, it is easy to obtain the anatomical labels on the subject image from the symbolic calculation target of the correlation value

$$\begin{cases} \text{foreground} & f_i > 0 \\ \text{background} & f_i < 0 \end{cases}, \quad i = 1, 2 \ldots |P| \, . \tag{10.48}$$

In other words, the segmentation can be repeatedly computed to improve the performance by: (1) hypergraph construction with high-level context features; (2) label propagation on hypergraph; and (3) the refinement of context features. The segmentation results can be found in Fig. 10.12.

10.6 Summary

In this chapter, we introduce three typical applications of hypergraph computation in medical and biological tasks. In computer-aided diagnosis, three specific applications are covered, i.e., the identification and medical image retrieval of MCI and the identification of COVID-19 by CT imaging. These examples show how to adopt hypergraph computation for the tasks of classification and retrieval in medical and biological fields. For the survival prediction with histopathological images, the demonstrated hypergraph computation techniques can also be expanded to similar regression tasks. The introduced paradigm may also be applied to other cases with complicated connections. In summary, these examples demonstrate the high-order correlation between medical and biological data, which are modeled and learned by hypergraph computation. These indeed can contribute to the corresponding study. In addition to the aforementioned examples, there are many medical and biological applications that have the potential to be explored with hypergraph computation, such as medical image enhancement and multi-modal fusion.

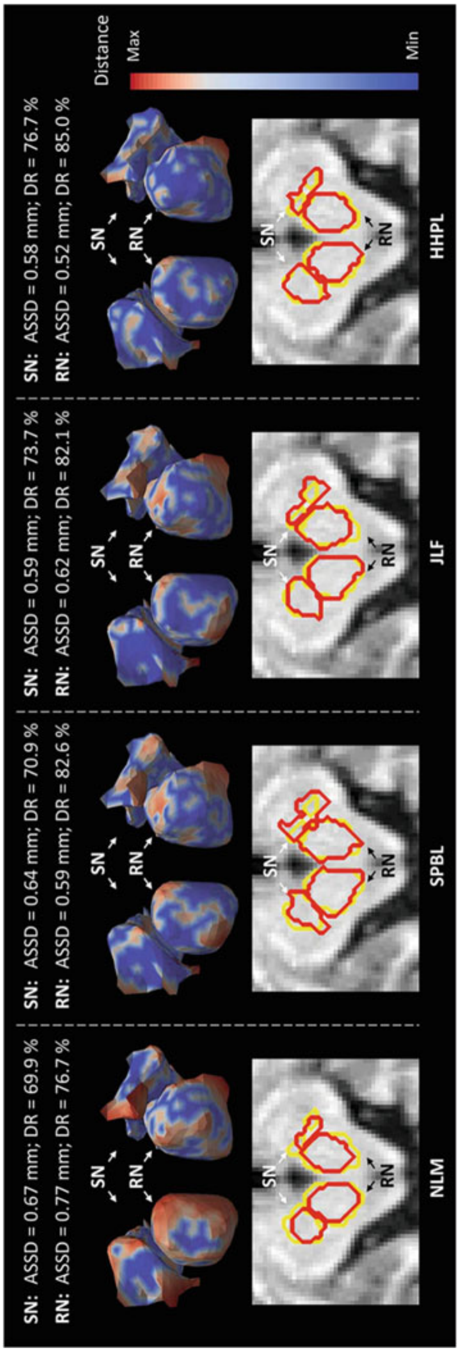

SN: ASSD = 0.67 mm; DR = 69.9 %
RN: ASSD = 0.77 mm; DR = 76.7 %
SN
RN
NLM
SN: ASSD = 0.64 mm; DR = 70.9 %
RN: ASSD = 0.59 mm; DR = 82.6 %
SN
RN
SPBL
SN: ASSD = 0.59 mm; DR = 73.7 %
RN: ASSD = 0.62 mm; DR = 82.1 %
SN
RN
JLF
SN: ASSD = 0.58 mm; DR = 76.7 %
RN: ASSD = 0.52 mm; DR = 85.0 %
SN
RN
HHPL
Distance
Max
Min

Fig. 10.12 Visual comparison of automatically segmented regions by four methods on a typical subject. This figure is from [8]

References

1. Y. Gao, C. Wee, M. Kim, P. Giannakopoulos, M. Montandon, S. Haller, D. Shen, MCI identification by joint learning on multiple MRI data, in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 78–85
2. Y. Gao, M. Kim, P. Giannakopoulos, S. Haller, D. Shen, Medical image retrieval using multi-graph learning for MCI diagnostic assistance, in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), pp. 86–93
3. Z. Zhang, J. Liu, B. Li, Y. Gao, Diagnosis of childhood autism using multi-modal functional connectivity via dynamic hypergraph learning, in *Proceedings of CAAI International Conference on Artificial Intelligence* (2021), pp. 123–135
4. D. Di, F. Shi, F. Yan, L. Xia, Z. Mo, Z. Ding, F. Shan, B. Song, S. Li, Y. Wei, Y. Shao, M. Han, Y. Gao, H. Sui, Y. Gao, D. Shen, Hypergraph learning for identification of COVID-19 with CT imaging. Med. Image Analy. **68**, 101910 (2021)
5. D. Di, S. Li, J. Zhang, Y. Gao, Ranking-based survival prediction on histopathological whole-slide images, in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (2020), pp. 428–438
6. D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, Y. Gao, Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction. IEEE Trans. Pattern Analy. Mach. Intell. 1–16 (2022). https://doi.org/10.1109/TPAMI.2022.3209652
7. D. Ruan, S. Ji, C. Yan, J. Zhu, X. Zhao, Y. Yang, Y. Gao, C. Zou, Q. Dai, Exploring complex and heterogeneous correlations on hypergraph for the prediction of drug-target interactions. Patterns **2**(12), 100390 (2021)
8. P. Dong, Y. Guo, Y. Gao, P. Liang, Y. Shi, G. Wu, Multi-Atlas segmentation of anatomical brain structures using hierarchical hypergraph learning. IEEE Trans. Neural Netw. Learn. Syst. **31**(8), 3061–3072 (2019)
9. D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency. in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 16 (2003)
10. F. Shan, Y. Gao, J. Wang, W. Shi, N. Shi, M. Han, Z. Xue, D. Shen, Y. Shi, Lung infection quantification of COVID-19 in CT images with deep learning (2020). Preprint arXiv:2003.04655
11. D. Jia, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255
12. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE. **86**, 2278–2324 (1998)
13. X. Zhu, J. Yao, F. Zhu, J. Huang, Wsisa: Making survival prediction from whole slide histopathological images, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 7234–7242
14. T. Robert, The lasso method for variable selection in the cox model. Statist. Med. **16**(4), 385–395 (1997)
15. R. Li, J. Yao, X. Zhu, Y. Li, J. Huang, Graph CNN for survival analysis on whole slide pathological images, in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018), pp. 174–182
16. N. Otsu, A threshold selection method from gray-level histograms. IEEE Trans. Syst Man Cybern. **9**(1), 62–66 (1979)
17. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
18. D. Kingma, M. Welling, Auto-Encoding variational bayes, in *Proceedings of International Conference on Learning Representations* (2014)

19. N.S. Madhukar, P.K. Khade, L. Huang, K. Gayvert, G. Galletti, M. Stogniew, J.E. Allen, P.Giannakakou, O. Elemento, A bayesian machine learning approach for drug target identification using diverse data types. Nat. Commun. **10**(1), 5221 (2019)
20. M. Campillos, M. Kuhn, A.C. Gavin, L.J. Jensen, P. Bork, Drug target identification using side-effect similarity. Science **321**(5886), 263–266 (2008)
21. M. Zhou, Y. Chen, R. Xu, A drug-side effect context-sensitive network approach for drug target prediction. Bioinformatics **35**(12), 2100–2107 (2019)
22. Q. Hu, Z. Deng, W. Tu, X. Yang, Z. Meng, Z. Deng, J. Liu, VNP: interactive visual network pharmacology of diseases, targets, and drugs. CPT Pharmacometrics Syst. Pharmacol. **3**(3), e105 (2014)

Chapter 11
Hypergraph Computation for Computer Vision

Abstract In this chapter, the applications of hypergraph computation in computer vision are introduced. Computer vision is one of the most widely used areas of hypergraph computation. The hypergraphs can be constructed by modeling the high-order relationship among inter- or intra-visual samples, and then computer vision tasks can be solved by hypergraph computation procedures. More specifically, four typical applications, including visual classification, 3D object retrieval, and tag-based social image retrieval, are provided, in which hypergraphs are used to model high-order relationship among samples and solve visual problems by hypergraph computation. For example, in social image retrieval, hypergraphs are used to model the high-order relationship among social images based on both visual and textual information, which is the high-order modeling of elements within samples.

11.1 Introduction

Hypergraphs have demonstrated excellent performance in modeling high-order relationship of data and have been applied in several fields. In computer vision, this property of hypergraphs is also promising for a wide range of works, and many researches focus on how to use hypergraph modeling to solve visual problems. On one hand, hypergraphs can be used to model high-order relationship of images within a class or different classes, and then to conduct the hypergraph-based label propagation procedures, which is useful for visual classification and retrieval. On the other hand, the relation can be modeled within the elements in a visual object to exploit the structural information.

In this chapter, we discuss four typical applications of hypergraph computation in computer vision, i.e., visual classification [1–6], 3D object retrieval [2, 7–12], and tag-based social image retrieval [13–17]. In these applications, the vertices represent the visual objects, and a hypergraph is constructed to formulate the high-order correlations among all the samples by some metric. In this hypergraph, some vertices are labeled. The prediction of other vertices can be obtained by the label propagation procedure. Visual classification and retrieval problems can be solved by this method. The elements within one sample, such as pixels in an image, can also be used to construct the hypergraphs. The properties with each element can be learned by conducting hypergraph computation, in which the semantic information can be learnt during this procedure. Part of the work introduced in this chapter has been published in [1, 2, 13].

11.2 Visual Classification

Visual classification is the most widely used area of hypergraph in computer vision. Since visual data have a strong clustering characteristic, i.e., visual objects under one label show a clustered distribution in the feature space, this property is fully consistent with the hypothesis of hypergraph-based semi-supervised learning, and therefore, hypergraph-based semi-supervised learning is theoretically well-suited for image classification. A large number of researches have demonstrated its good performance [1, 2]. While there are many applications of hypergraph computation for image classification, they almost follow the same process. It starts out with hypergraph modeling of visual data. After extracting features by some feature extractors, the hypergraph is modeled based on the nearest neighbor relationship of visual features in the Euclidean space, and then label propagation on the hypergraph is adopted to achieve classification. We use the example of multi-view 3D object classification to introduce the process in detail.

First, view-based 3D object classification needs to be introduced. Each 3D object can be represented by a set of views. Compared with the model representation method, the multi-view representation method is more flexible, with less computational overhead. It also has good representation capability. Classification of 3D objects is illustrated in Fig. 11.1. After obtaining the multi-view 3D object data, the first step is to extract the features. There are many feature extraction

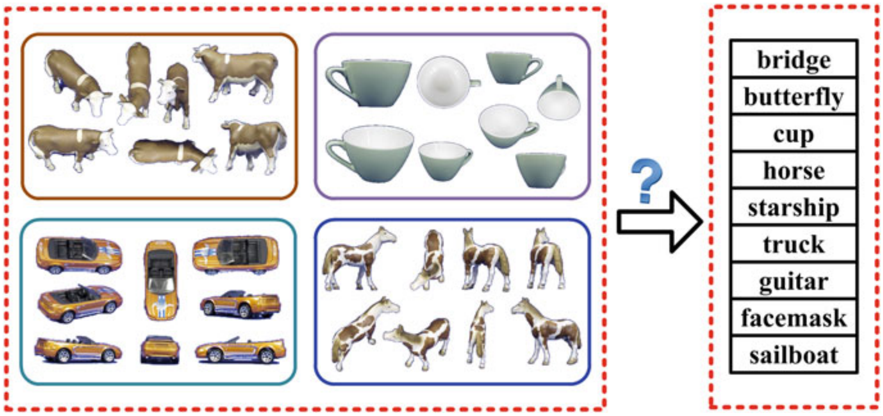


Fig. 11.1 An illustration of the view-based 3D object classification framework. This figure is from [1]

methods for multi-view 3D objects, such as MVCNN [18], Zernike moments, etc. After obtaining the features of each group of views and each image in them, hyperedges can be constructed by k-NN with Euclidean distance as the metric. In fact, if several different features are used, multiple hypergraphs can be constructed, i.e., each hypergraph is constructed based on one feature. If m features are used, m hypergraphs can be generated, denoted by $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathbf{W}_1), \mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathbf{W}_2), \ldots, \mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m, \mathbf{W}_m)$. After obtaining multiple hypergraphs, a weight $\omega_i, i = 1, \ldots, m$ is assigned to each hypergraph $\mathcal{G}_i$, which constitutes a weight vector ω. Up to this point, we obtain m hypergraphs with weights from the multi-view 3D dataset.

Transductive Hypergraph Computation

After getting multiple hypergraphs, we can get the label of each vertex by the formula of hypergraph-based semi-supervised learning. The pipeline is shown in Fig. 11.2a. Note that since we are using multi-modal data, the contribution of different modalities to the classification may be different, such that we also have

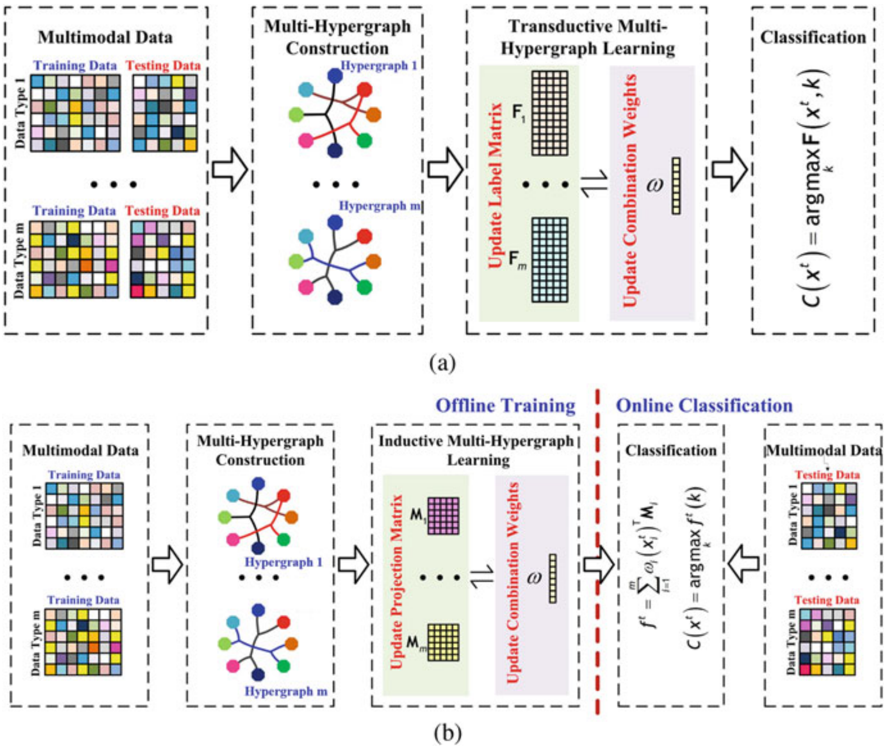


Fig. 11.2 The general frameworks of transductive and inductive multi-hypergraph computation algorithms. (**a**) tMHL: transductive multi-hypergraph computation. (**b**) iMHL: inductive multi-hypergraph computation. This figure is from [1]

to take into account the influence of different modal weights when calculating the classification results and updating the weights during the computing process. The method of weight updating is described in the next section, and the focus here is to establish the idea of hypergraph processing of multi-modal features.

Inductive Hypergraph Computation

In real-world visual classification endeavors, transductive hypergraph computation can only be updated globally, and the high time complexity can hardly meet efficiency requirements of visual classification. To help solve this problem, inductive hypergraph computation is introduced, which can learn both projections of data to labels and weight vectors of multiple hypergraphs. It can also achieve real-time inference performance for newly added data, as shown in Fig. 11.2b. It is described in the following.

In inductive hypergraph computation, a projection matrix $\mathbf{M}$ is learned, and the prediction for the unlabeled data is computed by $\mathbf{M}$.

The objective function for learning $\mathbf{M}$ is illustrated as

$$\arg\min_{\mathbf{M}} \left\{\Omega\left(\mathbf{M}\right) + \lambda \mathscr{R}_{emp}\left(\mathbf{M}\right) + \mu \Phi\left(\mathbf{M}\right)\right\}. \tag{11.1}$$

Under the assumption that it is more likely that the vertices connected with one or more hyperedges have the same label, the hypergraph Laplacian regularizer $\Omega(\mathbf{M})$ is defined as follows, and it is in quadratic form of $\mathbf{M}$:

$$\begin{aligned}\Omega\left(\mathbf{M}\right) =& \frac{1}{2}\sum_{k=1}^{c}\sum_{e\in\mathscr{E}}\sum_{u,v\in\mathscr{V}} \frac{\mathbf{W}\left(e\right)\mathbf{H}\left(u,e\right)\mathbf{H}\left(v,e\right)}{\delta(e)}\vartheta \\ =& \operatorname{tr}\left(\mathbf{M}^{\top}\mathbf{X}\Delta\mathbf{X}^{\top}\mathbf{M}\right),\end{aligned} \tag{11.2}$$

where $\vartheta = \left(\frac{\mathbf{X}^{\top}\mathbf{M}(u,k)}{\sqrt{d(u)}} - \frac{\mathbf{X}^{\top}\mathbf{M}(v,k)}{\sqrt{d(v)}}\right)^2$. It can be noted that $\Omega(\mathbf{M})$ is in quadratic form of $\mathbf{M}$. The empirical loss term $\mathscr{R}_{emp}(\mathbf{M})$ is defined as

$$\mathscr{R}_{emp}\left(\mathbf{M}\right) = ||\mathbf{X}^{\top}\mathbf{M} - \mathbf{Y}||^2. \tag{11.3}$$

$\Phi(\mathbf{M})$ is an $l_{2,1}$ norm regularizer. It is used to avoid overfitting for $\mathbf{M}$. Meanwhile, it makes the rows in the matrix more sparse to be informative. It is defined as

$$\Phi(\mathbf{M}) = ||\mathbf{M}||_{2,1}. \tag{11.4}$$

The objective function of inductive hypergraph computation task can be written as

$$\arg\min_{\mathbf{M}} \left\{\operatorname{tr}\left(\mathbf{M}^{\top}\mathbf{X}\Delta\mathbf{X}^{\top}\mathbf{M}\right) + \lambda||\mathbf{X}^{\top}\mathbf{M} - \mathbf{Y}||^2 + \mu||\mathbf{M}||_{2,1}\right\}. \tag{11.5}$$

Note that the regularizer $\Phi(\mathbf{M})$ is convex and non-smooth. Therefore, the objective function can be relaxed to the following:

$$\arg\min_{\mathbf{M},\mathbf{U}} \left\{\mathrm{tr}\left(\mathbf{M}^\top \mathbf{X}\Delta\mathbf{X}^\top\mathbf{M}\right)+\lambda||\mathbf{X}^\top\mathbf{M}-\mathbf{Y}||^2+\mu\mathrm{tr}\left(\mathbf{M}^\top\mathbf{U}\mathbf{M}\right)\right\}, \tag{11.6}$$

where $\mathbf{U}$ is a diagonal matrix, and its elements are defined as

$$\mathbf{U}_{i,i} = \frac{1}{2||\mathbf{M}\left(\mathbf{i},:\right)||_2^2}, \quad i = 1, \ldots, d. \tag{11.7}$$

To solve this optimization problem, $\mathbf{U}$ is set as an identity matrix first, and the iteratively reweighted least squares method is adopted. More specifically, each variable is updated alternately with the other fixed until convergence is achieved. First, $\mathbf{U}$ is fixed, and we derive objection with respect to $\mathbf{M}$. The closed-form solution is

$$\mathbf{M} = \lambda\left(\mathbf{X}\Delta\mathbf{X}^\top + \lambda\mathbf{X}\mathbf{X}^\top + \mu\mathbf{U}\right)^{-1}\mathbf{X}\mathbf{Y}. \tag{11.8}$$

Then $\mathbf{M}$ is fixed, while $\mathbf{U}$ is updated by Eq. (11.7). The procedure is repeated until both $\mathbf{U}$ and $\mathbf{M}$ converge.

Given a testing sample x^t, the prediction of x^t can be obtained by

$$C(x^t) = \arg\max_k \; {x^t}^\top\mathbf{M}. \tag{11.9}$$

Hypergraph computation can achieve good results in visual classification problems, where inductive hypergraph computation can achieve real-time online classification while maintaining good classification performance.

11.3 3D Object Retrieval

3D object retrieval targets on finding similar 3D objects in the database, given a 3D query. Usually, each 3D object can be described by several different types of data, such as multiple views, point clouds, mesh, or voxel. The main task of 3D object retrieval is to define an appropriate measure to calculate the similarity between each pair of 3D objects. Therefore, how to define such measures is the key for 3D object retrieval. Traditional methods mainly focus on either representation learning for each type of data or the distance metric for specific features. It is noted that the correlations among 3D objects are very complex, where the pair correlations and beyond-pair correlation both exist. To achieve better 3D object retrieval performance, it is important to take such high-order correlation among 3D objects into consideration. In this retrieval task, each vertex denotes a 3D object in

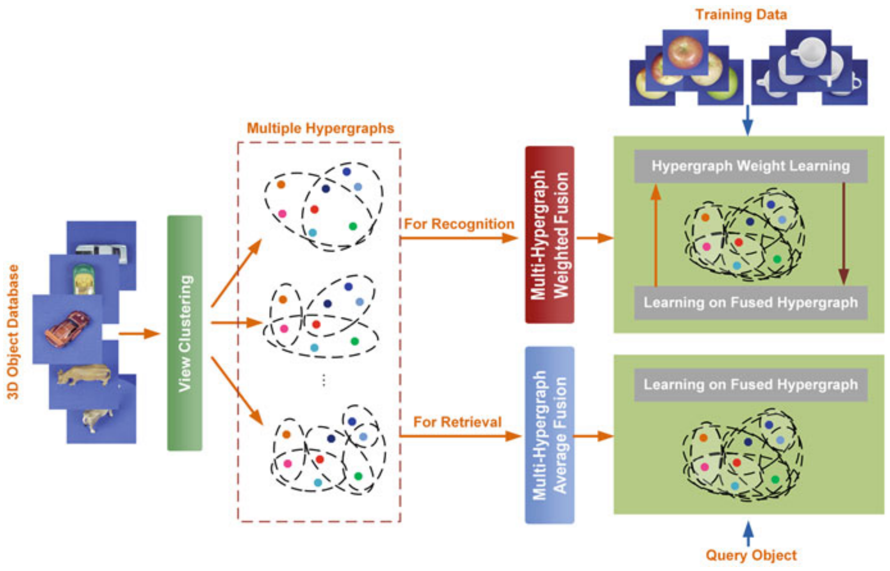

Training Data
Multiple Hypergraphs
3D Object Database
View Clustering
For Recognition
For Retrieval
Multi-Hypergraph Weighted Fusion
Multi-Hypergraph Average Fusion
Hypergraph Weight Learning
Learning on Fused Hypergraph
Learning on Fused Hypergraph
Query Object

Fig. 11.3 An illustration of the hypergraph computation method for 3D object retrieval using multiple views. This figure is from [2]

the database, and thus the number of vertices is equivalent to the number of objects in the database.

Hypergraph can be used for such correlation modeling in 3D object retrieval. We introduce the hypergraph computation method [2] for 3D object retrieval here, and the framework is shown in Fig. 11.3. First a group of hypergraphs can be generated, and the learning process is conducted for similarity measurement.

We take the multi-view representation as an example. All views of these 3D objects are first grouped into clusters. Objects with views in one cluster are then connected by hyperedges (note that a hyperedge can connect multiple vertices in a hypergraph). As a result, a hypergraph can be generated, in which vertices represent objects in a database. A hyperedge's weight is determined by the visual similarity between any two views in a cluster. Multiple hypergraphs can be generated by varying the number of clusters. These hypergraphs encode the relationships between objects at various granularities. When two 3D objects are connected by more and stronger hyperedges, they are with higher similarity. Then, these information can be used for 3D object retrieval.

To generate a 3D object hypergraph, each object is as a vertex in the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. The generated hypergraph has n vertices if there are n objects in a database. Each view for these 3D objects can be represented by pre-defined features, which can be different with respect to various of tasks. Given these features, the K-means clustering method can be used to group visual objects into clusters. Each object in a cluster has a corresponding hyperedge connecting them. There are two diagonal matrices $\mathbf{D}_v$ and $\mathbf{D}_e$ that represent the vertex and hyperedge degrees,

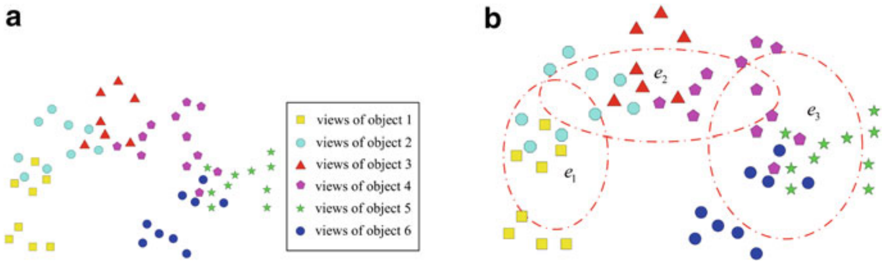


Fig. 11.4 An illustration of the hypergraph construction for 3D object hypergraph. (**a**) Views of different visual objects. (**b**) Hyperedges construction by view clusters. This figure is from [2]

respectively, and an incidence matrix $\mathbf{H}$ is generated. The weight of a hyperedge e can be measured by

$$w(e) = \sum_{x_a, x_b \in e} \exp\left(-\frac{d(x_a, x_b)^2}{\sigma^2}\right), \tag{11.10}$$

where $d(x_a, x_b)$ is the distance between x_a and x_b, which are two views in the same view cluster. $d(x_a, x_b)$ can be calculated using the Euclidean distance. The parameter σ is empirically set to the median distance between all pairs of these views. The hypergraph generation procedure is shown in Fig. 11.4.

Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathbf{W}_1)$, $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathbf{W}_2)$, $\cdots$, and $\mathcal{G}_{n_g} = (\mathcal{V}_{n_g}, \mathcal{E}_{n_g}, \mathbf{W}_{n_g})$ denote n_g hypergraphs, and $\{\mathbf{D}_{v_1}, \mathbf{D}_{v_2}, \ldots, \mathbf{D}_{vn_g}\}$, and $\{\mathbf{D}_{e_1}, \mathbf{D}_{e_2}, \ldots, \mathbf{D}_{en_g}\}$, and $\{\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_{n_g}\}$ be the vertex degree matrices, hyperedge degree matrices, and incidence matrices, respectively. The retrieval results are based on the fusion of these hypergraphs. The weight of the i-th hypergraph is denoted by α_i, where $\sum_{i=1}^{n_g} \alpha_i = 1$, and $\alpha_i \leq 0$.

It is possible to consider retrieval as a one-class classification problem [19]. As a result, we formulate the transductive inference in terms of a regularization problem: $\arg\min_{\mathbf{f}} \{\lambda R_{\text{emp}}(\mathbf{f})\} + \Omega(\mathbf{f})$, and the regularizer term $\Omega(\mathbf{f})$ is defined by

$$\frac{1}{2}\sum_{i=1}^{n_g} \alpha_i \sum_{e \in \mathcal{E}_i} \sum_{u,v \in \mathcal{V}_i} \frac{w_i(e)\mathbf{H}_i(u,e)\mathbf{H}_i(v,e)}{\sigma_i(e)} \times \left(\frac{\mathbf{f}(u)}{\sqrt{d_i(u)}} - \frac{\mathbf{f}(v)}{\sqrt{d_i(v)}}\right)^2, \tag{11.11}$$

where vector $\mathbf{f}$ represents the relevance score to be learned.

In this way, the similarity between each object and the query can be calculated based on the relevance score. It is noted that the feature used in this method can be selected based on the task itself, and multiple types of representations can also be used here. Given multiple features for the same data, or different features for multi-modal data, we can generate the hypergraph(s) using the method introduced in Chap. 4.

11.4 Tag-Based Social Image Retrieval

User-generated tags are widely associated with the social images, which describe the content of the images. These tags are useful for the social image retrieval tasks benefited from the rich contents. Figure 11.5 shows some examples of social images associated with tags.

The main challenge of applying such tags to social image retrieval is that too much noise makes it hard to mine the true relation among the tags and images, and the separation usage of the tags and images leads to a sub-optimal for image retrieval. In this section, we introduce a visual–textual joint relevance learning approach using hypergraph computation [13]. Figure 11.6 shows the illustration of the visual–textual joint relevance learning method on hypergraph for tag-based social image retrieval. In this method, the features for both the images and the tags are first extracted, and the hypergraph is constructed based on these features. Then, the hypergraph learning method is performed, and the learned semantic similarity can be used for tag-based social image retrieval.

In this example, the bag-of-visual-words feature is selected for image representation. For the i-th image, the visual content is represented by bag-of-visual-words f_i^{bow}, while for the corresponding tags, the bag-of-textual words representation f_i^{tag} is employed. Then, the visual-content-based hyperedges and the tag-based hyperedges are constructed, respectively. The visual-content-based hyperedges connect the images that have the same visual word, and the tag-based hyperedges connect the images that have the same tag word. Figure 11.7 provides the examples of hyperedge generation process using textual information and visual information, respectively. Therefore, the overall hypergraph has $n_e = n_c + n_t$ hyperedges, where n_c denotes the number of visual words, and n_t denotes the number of tag words. After the construction of the hypergraph, the images sharing more visual words or tags are connected by more hyperedges, which can be used for further processing. Figure 11.8 further shows the connections between two social images, based on the textual and the visual information, respectively.

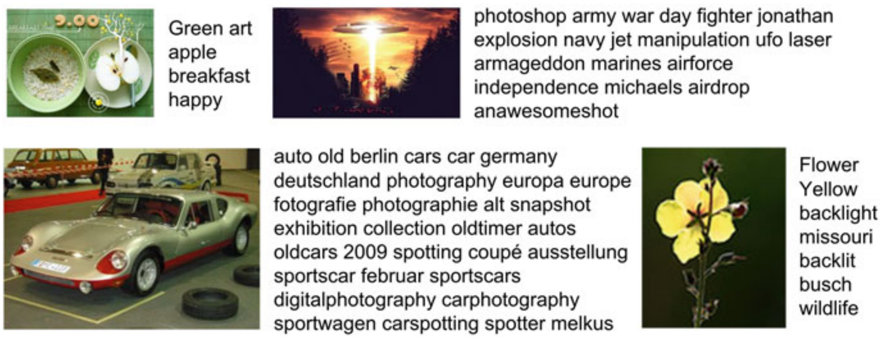

Green art apple breakfast happy
photoshop army war day fighter jonathan explosion navy jet manipulation ufo laser armageddon marines airforce independence michaels airdrop anawesomeshot
auto old berlin cars car germany deutschland photography europa europe fotografie photographie alt snapshot exhibition collection oldtimer autos oldcars 2009 spotting coupé ausstellung sportscar februar sportscars digitalphotography carphotography sportwagen carspotting spotter melkus
Flower Yellow backlight missouri backlit busch wildlife

Fig. 11.5 Some social image examples with associated with tags. This figure is from [13]

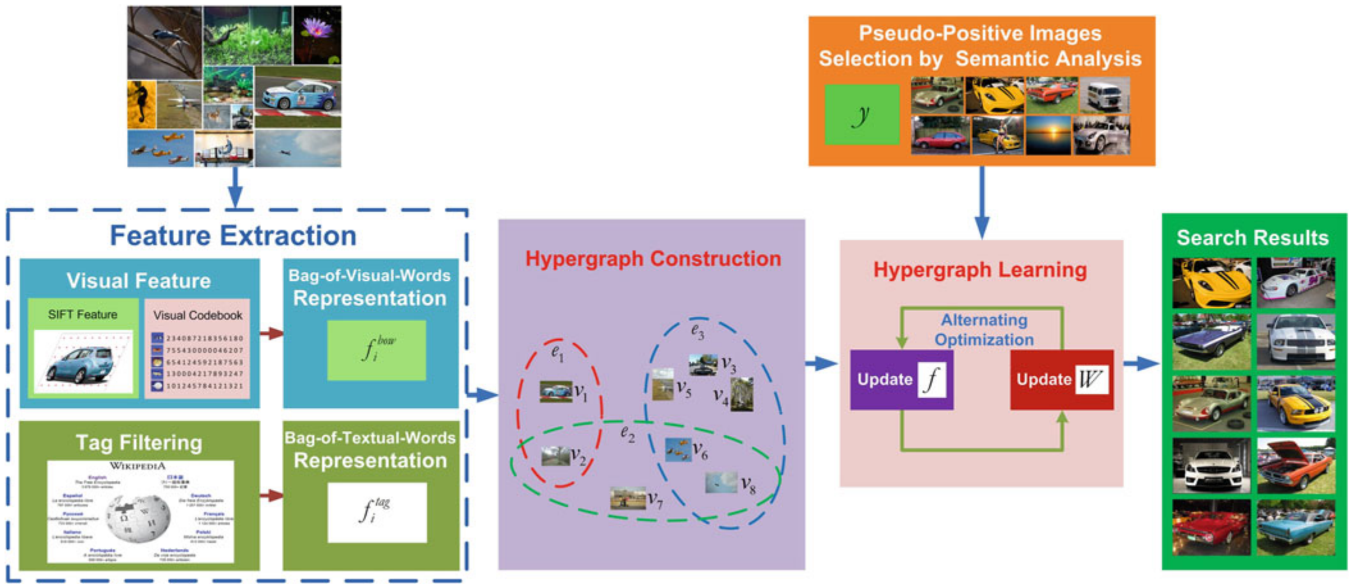


Fig. 11.6 The framework of the visual–textual joint relevance learning method on hypergraph for tag-based social image retrieval. This figure is from [13]

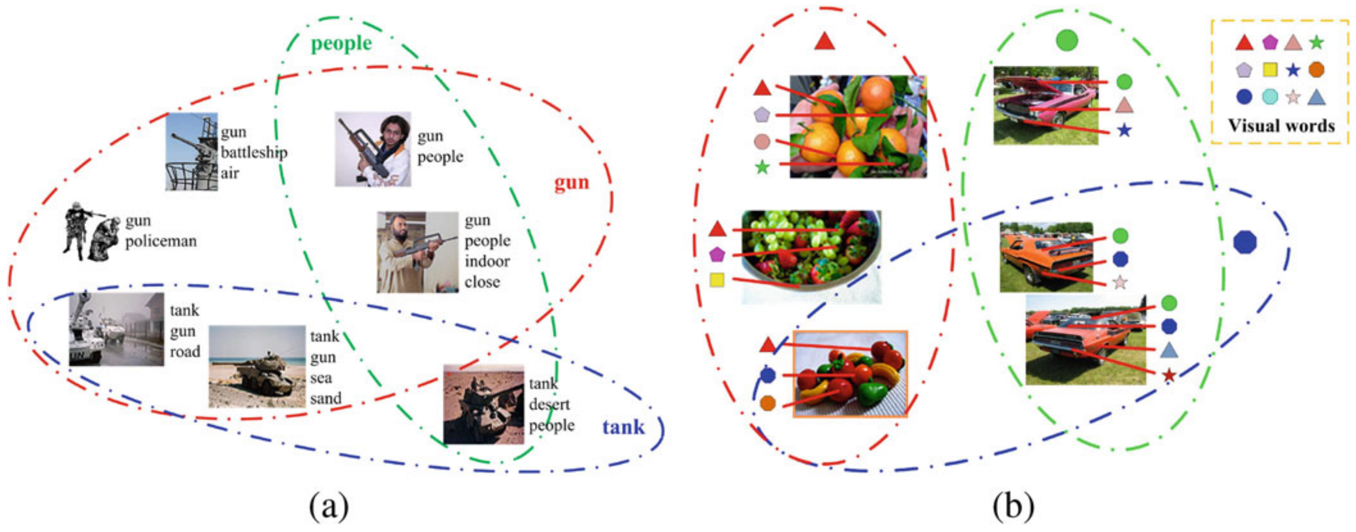


Fig. 11.7 Two examples of hyperedge generation. (**a**) shows hyperedges based on the textual information, in which the social images with the same textual words are connected by a hyperedge. (**b**) shows hyperedges based on the visual information, in which the social images with the same visual words are connected by a hyperedge. This figure is from [13]

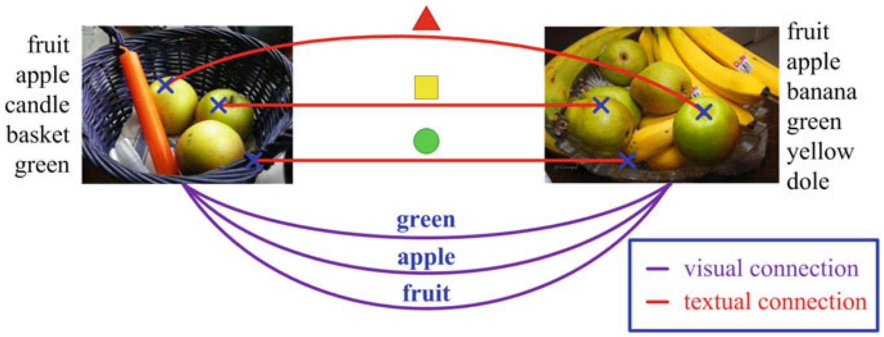


Fig. 11.8 An example of connections between two images from textual and visual directions. This figure is from [13]

Denoting **f** as the relevance score vector, **y** as the ground truth relevance, and **w** is the weight vector of hyperedges, the hypergraph computation can be formulated as

$$\arg\min_{\mathbf{f},\mathbf{w}} \Phi(\mathbf{f}) = \arg\min_{\mathbf{f}} \left\{ \mathbf{f}^\top \Delta \mathbf{f} + \lambda ||\mathbf{f}-\mathbf{y}||^2 + \mu \sum_{i=1}^{n_e} \mathbf{w}(i)^2 \right\}, \\ s.t. \quad \sum_{i=1}^{n_e} \mathbf{w}(i) = 1, \tag{11.12}$$

where λ and μ are the weighted parameters. The first term in Eq. (11.12) is the regularizer on the hypergraph structure, which is used to guarantee the smoothness over the hypergraph. The second term is the empirical loss between the relevance score vector and the ground truth. The last term represents the ℓ_2 norm of the hyperedge weights, which is used to learn better combination of different hyperedges. This optimization task can be easily solved using alternating optimization. First, **w** is fixed, and f is optimized by

$$\arg\min_{\mathbf{f}} \Phi(\mathbf{f}) = \arg\min_{\mathbf{f}} \left\{ \mathbf{f}^\top \Delta \mathbf{f} + \lambda ||\mathbf{f}-\mathbf{y}||^2 \right\}, \tag{11.13}$$

from which we can have

$$\mathbf{f} = \frac{1}{1-\xi}(\mathbf{I} - \xi\Theta)^{-1}\mathbf{y}, \tag{11.14}$$

where $\xi = \frac{1}{1+\lambda}$, $\Theta = \mathbf{I} - \Delta$.

Then, $\mathbf{f}$ is fixed, and $\mathbf{w}$ is optimized by

$$\arg\min_{\mathbf{w}} \Phi(\mathbf{f}) = \arg\min_{\mathbf{f}} \left\{ \mathbf{f}^\top \Delta \mathbf{f} + \mu \sum_{i=1}^{n_e} \mathbf{w}(i)^2 \right\}. \\ s.t. \sum_{i=1}^{n_e} \mathbf{w}(i) = 1, \mu > 0. \tag{11.15}$$

The Lagrangian can be applied here, and we have

$$\mathbf{w}(i) = \frac{1}{n_e} - \frac{\mathbf{f}^\top \Gamma \mathbf{D}_e^{-1} \Gamma^\top \mathbf{f}}{2 n_e \mu} + \frac{\mathbf{f}^\top \Gamma_i \mathbf{D}_e^{-1}(i,i) \Gamma_i^\top \mathbf{f}}{2\mu}, \tag{11.16}$$

where $\Gamma = \mathbf{D}_v^{-\frac{1}{2}} \mathbf{H}$ and Γ_i represents the i-th column of Γ.

The semantic relevance between an image x_i and the query tag t_q is estimated by

$$s(x_i, t_q) = \frac{1}{n_i} \sum_t s_{tag}(t_q, t), \tag{11.17}$$

which denotes the average similarity between t_q and all corresponding tags of x_i, and s_{tag} can be calculated as

$$s_{tag}(t_1, t_2) = e^{-FD(t_1,t_2)}, \tag{11.18}$$

where FD represents the Flickr distance [20].

Given these similarities between each image and the query tag, we can have the retrieval results accordingly. We also note that the features used in this application can be changed with respect to the requirement of different tasks.

11.5 Summary

In this chapter, we have introduced the applications of hypergraph computation on computer vision, including visual classification, 3D object retrieval, and tag-based social image retrieval. For classification and retrieval tasks, hypergraphs can be used to model the high-order relationships among samples in the feature space and solve the problem by hypergraph-based label propagation methods. The success of hypergraphs for computer vision is due to the fact that the feature correlations of visual data are more complex that are hard to be explored by pairwise correlation methods. Hypergraph computation can be further used in other computer vision tasks, such as visual registration, visual segmentation, gaze estimation, etc.

References

1. Z. Zhang, H. Lin, X. Zhao, R. Ji, Y. Gao, Inductive multi-hypergraph learning and its application on view-based 3D object classification. IEEE Trans. Image Process. **27**(12), 5957—5968 (2018)
2. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
3. J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification. IEEE Trans. Image Process. **21**(7), 3262–3272 (2012)
4. D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, Y. Gao, Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction. IEEE Trans. Pattern Analy. Mach. Intell. 1–16 (2022). https://doi.org/10.1109/TPAMI.2022.3209652
5. D. Di, S. Li, J. Zhang, Y. Gao, Ranking-based survival prediction on histopathological whole-slide images, in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, (2020), pp. 428–438
6. D. Di, J. Zhang, F. Lei, Q. Tian, Y. Gao, Big-hypergraph factorization neural network for survival prediction from whole slide image. IEEE Trans. Image Process. **31**, 1149–1160 (2022)
7. J. Bai, B. Gong, Y. Zhao, F. Lei, C. Yan, Y. Gao, Multi-scale representation learning on hypergraph for 3D shape retrieval and recognition. IEEE Trans. Image Process. **30**, 5327–5338 (2021)
8. G.Y. An, Y. Huo, S.E. Yoon, Hypergraph propagation and community selection for objects retrieval, in *Proceedings of the Advances in Neural Information Processing Systems*, (2021), pp. 3596–3608
9. D. Pedronette, L. Valem, J. Almeida, R. Torres, Multimedia retrieval through unsupervised hypergraph-based manifold ranking. IEEE Trans. Image Process. **28**(12), 5824–5838 (2019)
10. L. Nong, J. Wang, J. Lin, H. Qiu, L. Zheng, W. Zhang, Hypergraph wavelet neural networks for 3D object classification. Neurocomputing. **463**, 580–595 (2021)
11. S. Bai, X. Bai, Q. Tian, L.J. Latecki, Regularized diffusion process on bidirectional context for object retrieval. IEEE Trans. Pattern Analy. Mach. Intell. **41**(5), 1213–1226 (2019)
12. F. Chen, B. Li, L. Li, 3D object retrieval with graph-based collaborative feature learning. J. Visual Commun. Image Represen. **28**, 261–268 (2019)
13. Y. Gao, M. Wang, Z. Zha, J. Shen, X. Li, X. Wu, Visual-textual joint relevance learning for tag-based social image search. IEEE Trans. Image Process. **22**(1), 363–376 (2013)
14. Y. Wang, L. Zhu, X. Qian, J. Han, Joint hypergraph learning for tag-based image retrieval. IEEE Trans. Image Process. **27**(9), 4437–4451 (2018)
15. L. Chen, Y. Gao, Y. Zhang, S, Wang, B. Zheng, Scalable hypergraph-based image retrieval and tagging system, in *Proceedings of the 34th IEEE International Conference on Data Engineering* (2018), pp. 257–268
16. N. Bouhlel, G. Feki, C.B. Amar, Visual re-ranking via adaptive collaborative hypergraph learning for image retrieval, in *Proceedings of the Advances in Information Retrieval - 42nd European Conference on IR Research* (2020), pp. 511–526
17. Y. Chu, C. Feng, C. Guo, Social-guided representation learning for images via deep heterogeneous hypergraph embedding, in *Proceedings of the 2018 IEEE International Conference on Multimedia and Expo* (2018), pp. 1–6
18. H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 945–953
19. Y. Huang, Q. Liu, S. Zhang, D. Metaxas, Image retrieval via probabilistic hypergraph ranking, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 3376–3383
20. L. Wu, X. Hua, N. Yu, W. Ma, S. Li, Flickr distance: a relationship measure for visual concepts, IEEE Transa. Pattern Analy. Mach. Intell. **34**(5), 863–875 (2012)

Chapter 12
The DeepHypergraph Library

Abstract This chapter introduces the DeepHypergraph library, which bridges the hypergraph theory and hypergraph applications. This library provides the generation of multiple low-order structures (such as graph and directed graph), high-order structures (such as hypergraph and directed hypergraph), datasets, operations, learning methods, visualizations, etc. We first introduce the design motivation and the overall architecture of the library. Then, we introduce the "correlation structure" and "function library" of the Deephypergraph library, respectively.

12.1 Introduction

We have designed DeepHypergraph (DHG),[1] a deep learning library built upon PyTorch[2] for hypergraph computation. It is a general framework that supports both low-order and high-order message passing such as from vertex to vertex, from vertex in one domain to vertex in another domain, from vertex to hyperedge, from hyperedge to vertex, and from vertex set to vertex set. It supports the generation of a wide variety of structures such as low-order structures (graph, directed graph, bipartite graph, etc.) and high-order structures (hypergraph, etc.). Various spectral-based operations (such as Laplacian-based smoothing) and spatial-based operations (such as message passing from domain to domain) are integrated inside different structures. It also provides multiple common metrics for performance evaluation on different tasks. A group of state-of-the-art models has also been implemented and can be easily used for research. We also provide several visualization tools for demonstration of both low-order structures and high-order structures. Besides, the dhg.experiments module (that implements Auto-ML upon Optuna[3]) can automatically tune the hyperparameters of the models in training and return the model with

[1] deephypergraph.org.

[2] http://pytorch.org/.

[3] https://optuna.org/.

Q. Dai, Y. Gao, *Hypergraph Computation*, Artificial Intelligence: Foundations, Theory, and Algorithms, https://doi.org/10.1007/978-981-99-0185-2_12

the best performance. In this chapter, we first introduce the correlation structures in DHG and then introduce the function library in DHG.

12.2 The Correlation Structures in DHG

The core motivation of designing the DHG library is to attach the spectral-based and spatial-based operations to each specified structure. When a structure has been created, these related Laplacian matrices and message passing operations with different aggregation functions can be called and combined to manipulate different input features. Figure 12.1 illustrates the architecture of the "correlation structure" in DHG. Currently, the implemented correlation structures of DHG include graph, directed graph, bipartite graph, and hypergraph. For each correlation structure, DHG has developed the corresponding basic operations, such as construction and structure modification functions, related structure transformation functions, and learning functions.

The most computation process on those correlation structures (graph, hypergraph, etc.) can be divided into two categories: spectral-based convolution and spatial-based message passing. The spectral-based convolution methods, such as typical GCN [1] and HGNN [2], learn a Laplacian matrix for a given structure and perform vertex feature smoothing with the generated Laplacian matrix to embed low-order and high-order structures to vertex features. The spatial-based message passing methods, such as typical GraphSAGE [3], GAT [4], and HGNN+ [5],

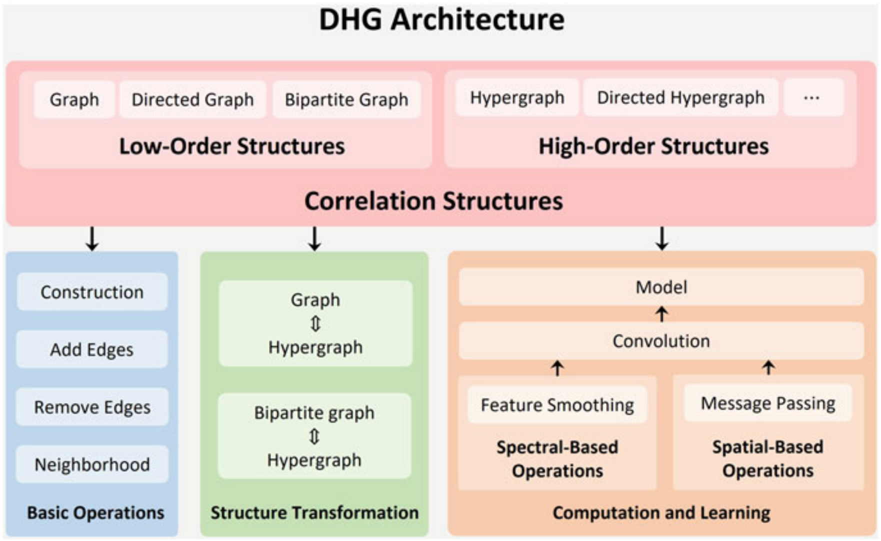


Fig. 12.1 The architecture of the "correlation structures" in DHG

perform vertex to vertex, vertex to hyperedge, hyperedge to vertex, and vertex set to vertex set message passing to embed the low-order and high-order structures to vertex features. The learned vertex features can also be pooled to generate the unified structure feature. Finally, the learned vertex features or structure features can be fed into many downstream tasks, such as classification, retrieval, regression, and link prediction, and applications including paper classification, movie recommender, drug exploitation, etc.

12.3 The Function Library in DHG

To facilitate the complex and repetition codes of learning on correlation structures, DHG further provides the function library. As shown in Fig. 12.2, the function library includes five parts: data module, metric module, visualization module, auto-ML module, and structure generators module.

In the data module, DHG integrates more than 20 public graph/bipartite graph/hypergraph datasets and some commonly used pre-process function such as File Loader and Normalization. By default, DHG can automatically download the integrated datasets and check the integrity of the downloaded files. You can also manually construct your own dataset of DHG style with the existing Datapipe functions in DHG.

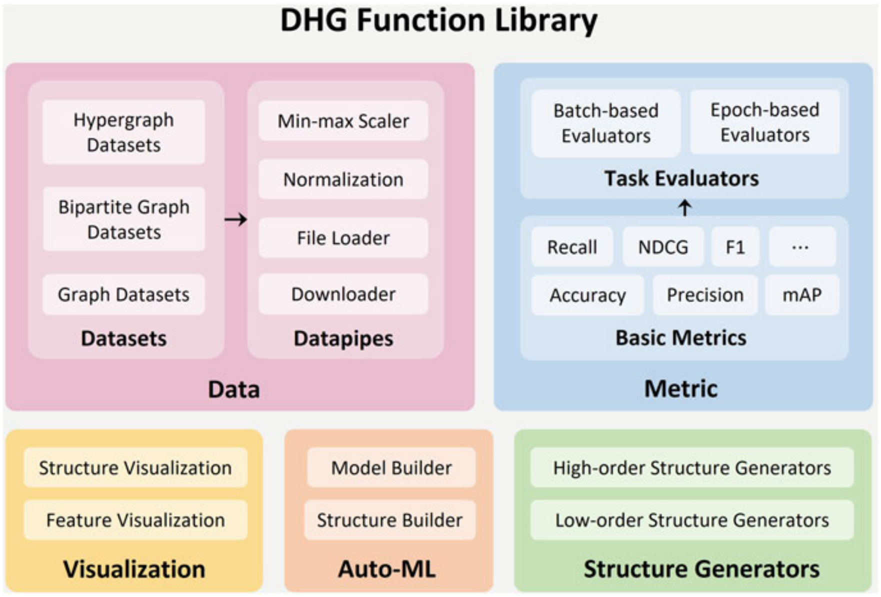


Fig. 12.2 The architecture of the "function library" in DHG

In the metric module, DHG has provided many widely used metrics such as *Accuracy*, *Recall*, and *mAP* for different tasks. Some encapsulation evaluators for different tasks such as classification, retrieval, and recommendation have also been implemented. Besides, DHG provides the structure and feature visualization functions, automatic hyperparameters search function, and random structure generation functions for different applications.

12.4 Summary

In this chapter, we introduce the DHG library for hypergraph computation. It simultaneously supports the generation and learning on low-order structures and high-order structures. Besides, many commonly used functions have also been integrated in the library.

References

1. T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in *Proceedings of the International Conference on Learning Representations* (2016)
2. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 3558–3565
3. W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 30 (2017)
4. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, (2017). Graph attention networks, in *Proceedings of the International Conference on Learning Representations* (2018)
5. Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN+: General hypergraph neural networks. IEEE Trans. Pattern Analy. Mach. Intell. **45**(3), 3181–3199 (2023)

Chapter 13
Conclusions and Future Work

13.1 Summary of This Book

Hypergraph computation has attracted much attention and shown apparent advantages in many application fields, such as computer vision, social networks, and biomedicine. In this book, we systematically introduce the basic knowledge, algorithms, and applications of hypergraph computation in three parts and discuss some recent progress in this direction.

In the first part, we mainly introduce the basic knowledge and main concepts of hypergraphs, including the definitions and symbols of common terms and the classification of hypergraphs. More importantly, we discuss the differences between hypergraphs and graphs from several aspects. Following, we introduce three hypergraph computation paradigms, namely, intra-hypergraph computation, inter-hypergraph computation, and hypergraph structure computation. In this part, we can have a general view of the different objectives in hypergraph computation.

In the second part, we specifically introduce a series of algorithms from hypergraph modeling to hypergraph neural networks. In hypergraph modeling sections, we show how to build a hypergraph structure from the collected data. As a typical and fundamental learning framework, label propagation on hypergraph describes how to derive the labels for unknown data from the labels for known data on the structure of a hypergraph. Other typical hypergraph computation tasks, including data clustering, cost-sensitive learning, and link prediction, are also introduced. Regarding the potential inaccurate hypergraph structure, we present the hypergraph structure evolution methods, which optimize the hypergraph structure on the basis of the initial structure. We further introduce the hypergraph neural network, which integrates the neural network framework into the hypergraph computation framework. The large scale hypergraph chapter discusses how to deal with large scale data for classification and clustering applications.

In the third part, we introduce practical examples of hypergraph computation in social media analysis, medical and biological applications, and computer vision, including specific tasks such as recommender system, sentiment analysis, computer-aided diagnosis, and image classification. In these examples, we show how to use hypergraph for high-order correlation modeling and select computation paradigms for different objectives. We further introduce the DeepHypergraph library for hypergraph computation.

13.2 Future Work

Although there have been many efforts to promote the development of hypergraph computation, there are still many open issues that need deep exploration, for instance, the mathematical foundations of hypergraph computation, the interpretability issues, and the temporal hypergraph modeling:

1. At present, the theory of hypergraph modeling and optimization is still far from completeness. As a flexible modeling method for high-order complex correlations, the hypergraph's main components, i.e., the number and degree of hyperedges on hypergraph, are not fixed, and how to measure the complexity of hypergraph structure is a problem worth further exploring. Previous investigation has shown superior performance of hypergraph computation in various applications, while the fundamental reason for this improvement and how much gain we can have from such high-order correlation modeling are still without a clear answer. In many tasks such as hypergraph matching, it is necessary to define the metrics in the hypergraph space. However, the problem is computationally expensive when the scale of hypergraph is very large. Therefore, efficient hypergraph matching and other algorithms are in immediate need. Existing hypergraph modeling methods still lack an evaluation of the quality of high-order correlation modeling and therefore lack credibility. It is needed to further explore the relationship between task complexity and structural complexity considering both the input data and the downstream tasks. It is expected that the hypergraphs can be generated according to the complexity of specific tasks and data, so as to achieve more reliable hypergraph modeling and optimization performance.
2. Interpretability is also an important research area of neural network models, and its purpose is to complete the explanation of black-box models through techniques such as feature masking and visualization. Since the hypergraph structure provides additional topological information, it brings out new opportunities to interpretability of hypergraph neural networks. Although there has been some work on the explanation problems of deep graph models in recent years, it is still in the infant stage. Interpretability of deep hypergraph model could be a potential road toward better deep neural network interpretability. There are two feasible paths to explanation techniques for hypergraph neural networks: instance-level explanation methods and model-level explanation methods. For example, it is possible to use different mask generation algorithms to obtain masks corresponding to vertices, edges, or the incidence matrix and then apply

the masks as disturbances to cover the original structure information to study the effects of different disturbances to the original structure. In addition, for hypergraphs in biochemistry, neurobiology, ecology, and engineering, of which the structure is highly correlated with their functions, how to combine domain knowledge to improve model interpretability is also an important issue. Finally, for text or image data, humans can easily understand the semantic information. However, it is difficult to intuitively understand the information of hypergraph structure. How to visualize the high-order complex correlations for intuitive understanding remains a challenge.

3. The combination of the temporal sequences and the hypergraph neural networks is also worth exploring. Recent research mainly focuses on static data and static hypergraph, where the data and the structure are kept fixed. However, in real-world applications, data may vary over time, which is called the temporal sequence, as well as the topology among the data. Therefore, the temporal information should be considered, and the temporal hypergraph neural networks aim to combine the temporal and spatial information. According to the variation of the data and the structure, there exist two main scenarios:

   - Time sequence data with static structure. This is a common scenario in the field of traffic forecasting, action recognition, and anomaly detection.
   - Time sequence data with evolving structure. This scenario mostly appears in the field of stock prediction and video relation detection.

   Under the above application circumstances, temporal hypergraph modeling is worth study. There are multiple challenges for the tasks mentioned above. For the sensor data, different types of the data are raised by different types of the sensors, while the typical hypergraph neural networks treat the data of the vertices equally. Both temporal and spatial high-order relationships vary over time, which makes the message passing procedure complex. New vertices/hyperedges emerge, and old vertices/hyperedges dissolve during the variation of the structure, which makes it complex to continuously model the varying correlation and aggregate the messages. The vertices/hyperedges may even be completely different at different time steps, which makes the representation-based method questionable. In order to model the temporal information, the vertex representations should be dynamic, and therefore, the representation should be learned on a functional space, rather than on the common vector space. The temporal information from both the vertex representation and the structure topology defies extraction. Considering these challenges, the temporal hypergraph still has a long way to go and needs further exploration.

Besides the above research directions, there are also several other interesting topics, such as big hypergraph model, hypergraph database, and distributed hypergraph, which have not been introduced in detail in this book and deserved further study.