Hypergraph Neural Networks

Yifan Feng,[1] **Haoxuan You,**[3] **Zizhao Zhang,**[3] **Rongrong Ji,**[1,2] **Yue Gao**[3*]
[1]Fujian Key Laboratory of Sensing and Computing for Smart City, Department of Congnitive Science
School of Information Science and Engineering, Xiamen University, 361005, China
[2]Peng Cheng Laboratory, China
[3]BNRist, KLISS, School of Software, Tsinghua University, 100084, China.
{evanfeng97, haoxuanyou}@gmail.com, rrji@xmu.edu.cn, {zz-z14,gaoyue}@tsinghua.edu.cn

Abstract

In this paper, we present a hypergraph neural networks (HGNN) framework for data representation learning, which can encode high-order data correlation in a hypergraph structure. Confronting the challenges of learning representation for complex data in real practice, we propose to incorporate such data structure in a hypergraph, which is more flexible on data modeling, especially when dealing with complex data. In this method, a hyperedge convolution operation is designed to handle the data correlation during representation learning. In this way, traditional hypergraph learning procedure can be conducted using hyperedge convolution operations efficiently. HGNN is able to learn the hidden layer representation considering the high-order data structure, which is a general framework considering the complex data correlations. We have conducted experiments on citation network classification and visual object recognition tasks and compared HGNN with graph convolutional networks and other traditional methods. Experimental results demonstrate that the proposed HGNN method outperforms recent state-of-the-art methods. We can also reveal from the results that the proposed HGNN is superior when dealing with multi-modal data compared with existing methods.

Introduction

Graph-based convolutional neural networks (Kipf and Welling 2017), (Defferrard, Bresson, and Vandergheynst 2016) have attracted much attention in recent years. Different from traditional convolutional neural networks, graph convolution is able to encode the graph structure of different input data using a neural network model and it can be used in the semi-supervised learning procedure. Graph convolutional neural networks have shown superiority on representation learning compared with traditional neural networks due to its ability of using data graph structure.

In traditional graph convolutional neural network methods, the pairwise connections among data are employed. It is noted that the data structure in real practice could be beyond pairwise connections and even far more complicated. Confronting the scenarios with multi-modal data, the situation for data correlation modelling could be more complex.

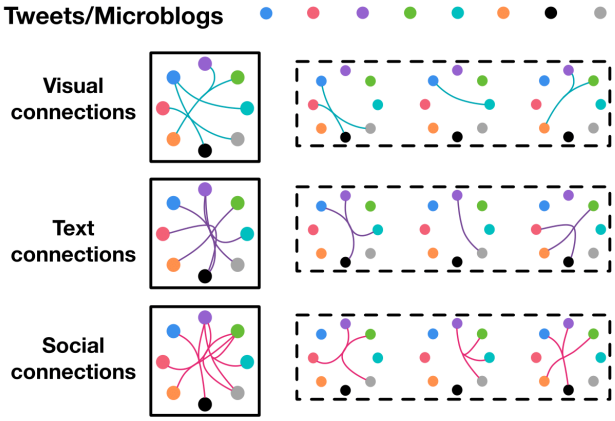

Figure 1: Examples of complex connections on social media data. Each color point represents a tweet or microblog, and there could be visual connections, text connections and social connections among them.

Figure 1 provides examples of complex connections on social media data. On one hand, the data correlation can be more complex than pairwise relationship, which is difficult to be modeled by a graph structure. On the other hand, the data representation tends to be multi-modal, such as the visual connections, text connections and social connections in this example. Under such circumstances, traditional graph structure has the limitation to formulate the data correlation, which limits the application of graph convolutional neural networks. Under such circumstance, it is important and urgent to further investigate better and more general data structure model to learn representation.

To tackle this challenging issue, in this paper, we propose a hypergraph neural networks (HGNN) framework, which uses the hypergraph structure for data modeling. Compared with simple graph, on which the degree for all edges is mandatory 2, a hypergraph can encode high-order data correlation (beyond pairwise connections) using its degree-free hyperedges, as shown in Figure 2. In Figure 2, the graph is represented using the adjacency matrix, in which each edge connects just two vertices. On the contrary, a hypergraph is easy to be expanded for multi-modal and heterogeneous

*Corresponding author. This work was finished when Yifan Feng visited Tsinghua University.

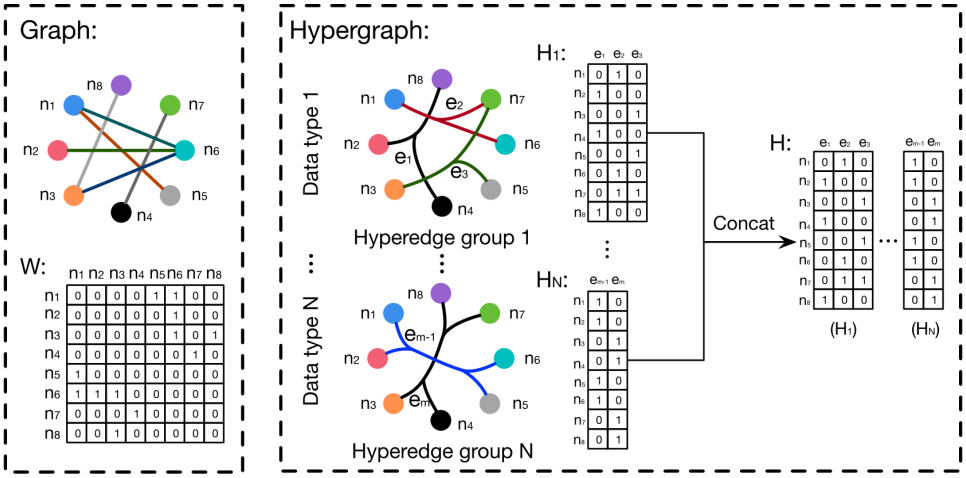

Figure 2: The comparison between graph and hypergraph.

data representation using its flexible hyperedges. For example, a hypergraph can jointly employ multi-modal data for hypergraph generation by combining the adjacency matrix, as illustrated in Figure 2. Therefore, hypergraph has been employed in many computer vision tasks such as classification and retrieval tasks (Gao et al. 2012). However, traditional hypergraph learning methods (Zhou, Huang, and Schölkopf 2007) suffer from their high computation complexity and storage cost, which limits the wide application of hypergraph learning methods.

In this paper, we propose a hypergraph neural networks framework (HGNN) for data representation learning. In this method, the complex data correlation is formulated in a hypergraph structure, and we design a hyperedge convolution operation to better exploit the high-order data correlation for representation learning. More specifically, HGNN is a general framework which can incorporate with multi-modal data and complicated data correlations. Traditional graph convolutional neural networks can be regarded as a special case of HGNN. To evaluate the performance of the proposed HGNN framework, we have conducted experiments on citation network classification and visual object recognition tasks. The experimental results on four datasets and comparisons with graph convolutional network (GCN) and other traditional methods have shown better performance of HGNN. These results indicate that the proposed HGNN method is more effective on learning data representation using high-order and complex correlations.

The main contributions of this paper are two-fold:

1. We propose a hypergraph neural networks framework, i.e., HGNN, for representation learning using hypergraph structure. HGNN is able to formulate complex and high-order data correlation through its hypergraph structure and can be also efficient using hyperedge convolution operations. It is effective on dealing with multi-modal data/features. Moreover, GCN (Kipf and Welling 2017) can be regarded as a special case of HGNN, for which the edges in simple graph can be regarded as 2-order hyperedges which connect just two vertices.
2. We have conducted extensive experiments on citation network classification and visual object classification tasks. Comparisons with state-of-the-art methods demonstrate the effectiveness of the proposed HGNN framework. Experiments also indicate the better performance of the proposed method when dealing with multi-modal data.

Related Work

In this section, we briefly review existing works of hypergraph learning and neural networks on graph.

Hypergraph learning

In many computer vision tasks, the hypergraph structure has been employed to model high-order correlation among data. Hypergraph learning is first introduced in (Zhou, Huang, and Schölkopf 2007), as a propagation process on hypergraph structure. The transductive inference on hypergraph aims to minimize the label difference among vertices with stronger connections on hypergraph. In (Huang, Liu, and Metaxas 2009), hypergraph learning is further employed in video object segmentation. (Huang et al. 2010) used the hypergraph structure to model image relationship and conducted transductive inference process for image ranking. To further improve the hypergraph structure, research attention has been attracted for leaning the weights of hyperedges, which have great influence on modeling the correlation of data. In (Gao et al. 2013), a l_2 regularize on the weights is introduced to learn optimal hyperedge weights. In (Hwang et al. 2008), the correlation among hyperedges is further explored by a assumption that highly correlated hyperedges should have similar weights. Regarding the multi-modal data, in (Gao et al. 2012), multi-hypergraph structure is introduced to assign weights for different sub-hypergraphs, which corresponds to different modalities.

Neural networks on graph

Since many irregular data that do not own a grid-like structure can only be represented in the form of graph, extending

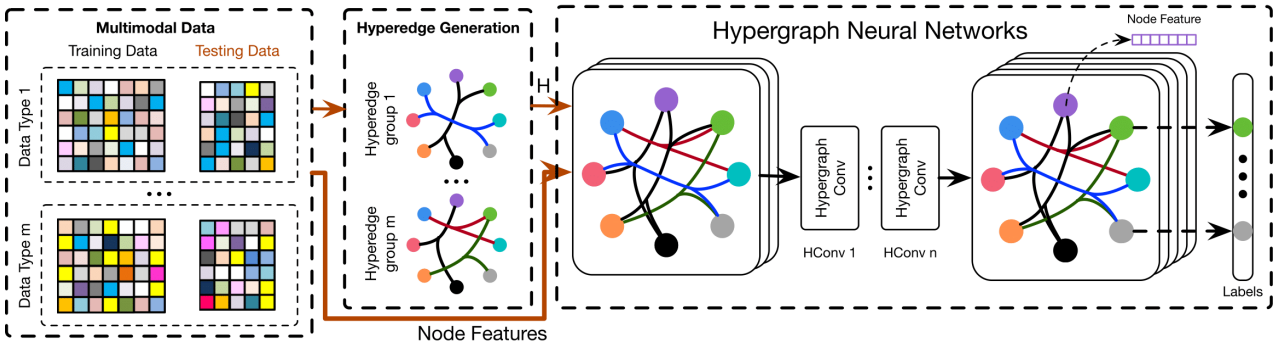

Figure 3: The proposed HGNN framework.

neural networks to graph structure has attracted great attention from researchers. In (Gori, Monfardini, and Scarselli 2005) and (Scarselli et al. 2009), the neural network on graph is first introduced to apply recurrent neural networks to deal with graphs. For generalizing convolution network to graph, the methods are divided into spectral and non-spectral approaches.

For spectral approaches, the convolution operation is formulated in spectral domain of graph. (Bruna et al. 2014) introduces the first graph CNN, which uses the graph Laplacian eigenbasis as an analogy of the Fourier transform. In (Henaff, Bruna, and LeCun 2015), the spectral filters can be parameterized with smooth coefficients to make them spatial-localized. In (Defferrard, Bresson, and Vandergheynst 2016), a Chebyshev expansion of the graph Laplacian is further used to approximate the spectral filters. Then, in (Kipf and Welling 2017), the chebyshev polynomials are simplified into 1-order polynomials to form an efficient layer-wise propagation model.

For spatial approaches, the convolution operation is defined in groups of spatial close nodes. In (Atwood and Towsley 2016), the powers of a transition matrix is employed to define the neighborhood of nodes. (Monti et al. 2017) uses the local path operators in the form of Gaussian mixture models to generalize convolution in spatial domain. In (Velickovic et al. 2018), the attention mechanisms is introduced into the graph to build attention-based architecture to perform the node classification task on graph.

Hypergraph Neural Networks

In this section, we introduce our proposed hypergraph neural networks (HGNN). We first briefly introduce hypergraph learning, and then the spectral convolution on hypergraph is provided. Following, we analyze the relations between HGNN and existing methods. In the last part of the section, some implementation details will be given.

Hypergraph learning statement

We first review the hypergraph analysis theory. Different from simple graph, a hyperedge in a hypergraph connects two or more vertices. A hypergraph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, which includes a vertex set $\mathcal{V}$, a hyperedge set $\mathcal{E}$. Each hyperedge is assigned with a weight by $\mathbf{W}$, a diagonal matrix of edge weights. The hypergraph $\mathcal{G}$ can be denoted by a $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix $\mathbf{H}$, with entries defined as

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

For a vertex $v \in \mathcal{V}$, its degree is defined as $d(v) = \sum_{e\in\mathcal{E}} \omega(e)h(v,e)$. For an edge $e \in \mathcal{E}$, its degree is defined as $\delta(e) = \sum_{v\in\mathcal{V}} h(v,e)$. Further, $\mathbf{D}_e$ and $\mathbf{D}_v$ denote the diagonal matrices of the edge degrees and the vertex degrees, respectively.

Here let us consider the node(vertex) classification problem on hypergraph, where the node labels should be smooth on the hypergraph structure. The task can be formulated as a regularization framework as introduced by (Zhou, Huang, and Schölkopf 2007):

$$\arg\min_f \{\mathcal{R}_{emp}(f) + \Omega(f)\}, \tag{2}$$

where $\Omega(f)$ is a regularize on hypergraph, $\mathcal{R}_{emp}(f)$ denotes the supervised empirical loss, $f(\cdot)$ is a classification function. The regularize $\Omega(f)$ is defined as:

$$\begin{aligned}\Omega(f) =& \frac{1}{2}\sum_{e\in\mathcal{E}}\sum_{\{u,v\}\in\mathcal{V}} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \\ & \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}}\right)^2,\end{aligned} \tag{3}$$

We let $\theta = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$ and $\mathbf{\Delta} = \mathbf{I} - \mathbf{\Theta}$. Then, the normalized $\Omega(f)$ can be written as

$$\Omega(f) = f^\top \mathbf{\Delta}, \tag{4}$$

where $\mathbf{\Delta}$ is positive semi-definite, and usually called the hypergraph Laplacian.

Spectral convolution on hypergraph

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{\Delta})$ with n vertices, since the hypergraph Laplacian $\mathbf{\Delta}$ is a $n \times n$ positive semi-definite matrix, the eigen decomposition $\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$ can be employed to get the orthonormal eigen vectors $\mathbf{\Phi} = \text{diag}(\phi_1, \ldots, \phi_n)$ and a diagonal matrix $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$ containing corresponding non-negative

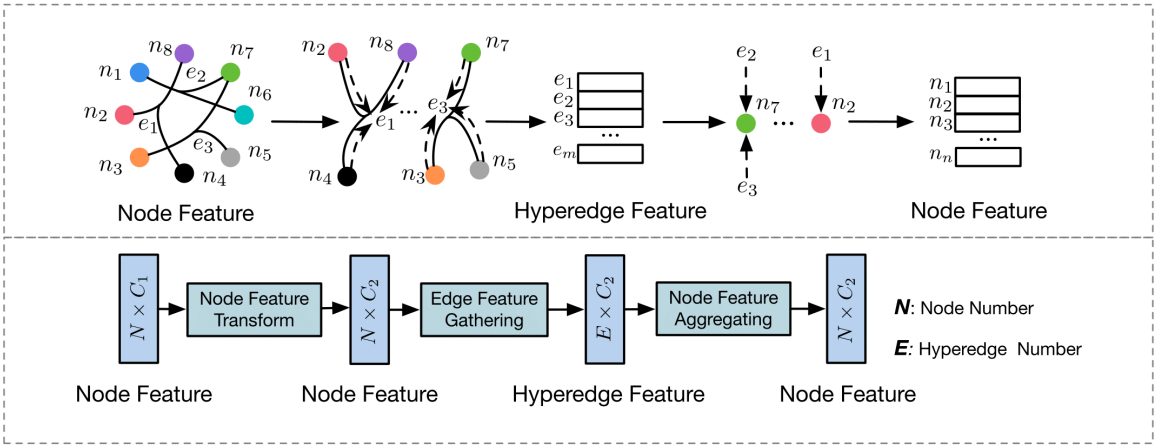

Figure 4: The illustration of the hyperedge convolution layer.

eigenvalues. Then, the Fourier transform for a signal $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ in hypergraph is defined as $\hat{\mathbf{x}} = \mathbf{\Phi}^\top \mathbf{x}$, where the eigen vectors are regarded as the Fourier bases and the eigenvalues are interpreted as frequencies. The spectral convolution of signal $\mathbf{x}$ and filter $\mathbf{g}$ can be denoted as

$$\mathbf{g} \star \mathbf{x} = \mathbf{\Phi}((\mathbf{\Phi}^\top \mathbf{g}) \odot (\mathbf{\Phi}^\top \mathbf{x})) = \mathbf{\Phi} g(\mathbf{\Lambda}) \mathbf{\Phi}^\top \mathbf{x}, \quad (5)$$

where $\odot$ denotes the element-wise Hadamard product and $g(\mathbf{\Lambda}) = \text{diag}(\mathbf{g}(\lambda_1), \ldots, \mathbf{g}(\lambda_n))$ is a function of the Fourier coefficients. However, the computation cost in forward and inverse Fourier transform is $\mathcal{O}(n^2)$. To solve the problem, we can follow (Defferrard, Bresson, and Vandergheynst 2016) to parametrize $g(\mathbf{\Lambda})$ with K order polynomials. Furthermore, we use the truncated Chebyshev expansion as one such polynomial. Chebyshv polynomials $T_k(x)$ is recursively computed by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. Thus, the $g(\mathbf{\Lambda})$ can be parametried as

$$\mathbf{g} \star \mathbf{x} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{\Delta}}) \mathbf{x}, \quad (6)$$

where $T_k(\tilde{\mathbf{\Delta}})$ is the Chebyshev polynomial of order k with scaled Laplacian $\tilde{\mathbf{\Delta}} = \frac{2}{\lambda_{max}} \mathbf{\Delta} - \mathbf{I}$. In Equation 6, the expansive computation of Laplacian Eigen vectors is excluded and only matrix powers, additions and multiplications are included, which brings further improvement in computation complexity. We can further let $K = 1$ to limit the order of convolution operation due to that the Laplacian in hypergraph can already well represent the high-order correlation between nodes. It is also suggested in (Kipf and Welling 2017) that $\lambda_{max} \approx 2$ because of the scale adaptability of neural networks. Then, the convolution operation can be further simplified to

$$\mathbf{g} \star \mathbf{x} \approx \theta_0 \mathbf{x} - \theta_1 \mathbf{D}^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{x}, \quad (7)$$

where θ_0 and θ_1 is parameters of filters over all nodes. We further use a single parameter θ to avoid the overfitting problem, which is defined as

$$\begin{cases} \theta_1 = -\frac{1}{2}\theta \\ \theta_0 = \frac{1}{2}\theta \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2}, \end{cases} \quad (8)$$

Then, the convolution operation can be simplified to the following expression

$$\begin{aligned} \mathbf{g} \star \mathbf{x} &\approx \frac{1}{2}\theta \mathbf{D}_v^{-1/2} \mathbf{H}(\mathbf{W} + \mathbf{I}) \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{x} \\ &\approx \theta \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{x}, \end{aligned} \quad (9)$$

where $(\mathbf{W} + \mathbf{I})$ can be regarded as the weight of the hyperedges. $\mathbf{W}$ is initialized as an identity matrix, which means equal weights for all hyperedges.

When we have a hypergraph signal $\mathbf{X} \in \mathbb{R}^{\mathbf{n} \times \mathbf{C_1}}$ with n nodes and C_1 dimensional features, our hyperedge convolution can be formulated by

$$\mathbf{Y} = \mathbf{D_v^{-1/2}} \mathbf{H} \mathbf{W} \mathbf{D_e^{-1}} \mathbf{H}^\top \mathbf{D_v^{-1/2}} \mathbf{X} \mathbf{\Theta}, \quad (10)$$

where $\mathbf{W} = \text{diag}(\mathbf{w_1}, \ldots, \mathbf{w_n})$. $\Theta \in \mathbb{R}^{C_1 \times C_2}$ is the parameter to be learned during the training process. The filter $\mathbf{\Theta}$ is applied over the nodes in hypergraph to extract features. After convolution, we can obtain $\mathbf{Y} \in \mathbb{R}^{n \times C_2}$, which can be used for classification.

Hypergraph neural networks analysis

Figure 3 illustrates the details of the hypergraph neural networks. Multi-modality datasets are divided into training data and testing data, and each data contains several nodes with features. Then multiple hyperedge structure groups are constructed from the complex correlation of the multi-modality datasets. We concatenate the hyperedge groups to generate the hypergraph adjacent matrix $\mathbf{H}$. The hypergraph adjacent matrix $\mathbf{H}$ and the node feature are fed into the HGNN to get the node output labels. As introduced in the above section, we can build a hyperedge convolutional layer $f(\mathbf{X}, \mathbf{W}, \mathbf{\Theta})$ in the following formulation

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{X}^{(l)} \mathbf{\Theta}^{(l)}), \quad (11)$$

where $\mathbf{X}^{(1)} \in \mathbb{R}^{N \times C}$ is the signal of hypergraph at l layer, $\mathbf{X}^{(0)} = \mathbf{X}$ and σ denotes the nonlinear activation function.

The HGNN model is based on the spectral convolution on the hypergraph. Here, we further investigate HGNN in the

property of exploiting high-order correlation among data. As is shown in Figure 4, the HGNN layer can perform node-edge-node transform, which can better refine the features using the hypergraph structure. More specifically, at first, the initial node feature $\mathbf{X}^{(1)}$ is processed by learnable filter matrix $\mathbf{\Theta}^{(1)}$ to extract C_2-dimensional feature. Then, the node feature is gathered according to the hyperedge to form the hyperedge feature $\mathbb{R}^{E\times C_2}$, which is implemented by the multiplication of $\mathbf{H}^\top \in \mathbb{R}^{\mathbf{E}\times\mathbf{N}}$. Finally the output node feature is obtained by aggregating their related hyperedge feature, which is achieved by multiplying matrix $\mathbf{H}$. Denote that $\mathbf{D_v}$ and $\mathbf{D_e}$ play a role of normalization in Equation 11. Thus, the HGNN layer can efficiently extract the high-order correlation on hypergraph by the node-edge-node transform.

Relations to existing methods When the hyperedges only connect two vertices, the hypergraph is simplified into a simple graph and the Laplacian $\mathbf{\Delta}$ is also coincident with the Laplacian of simple graph up to a factor of $\frac{1}{2}$. Compared with the existing graph convolution methods, our HGNN can naturally model high-order relationship among data, which is effectively exploited and encoded in forming feature extraction. Compared with the traditional hypergraph method, our model is highly efficient in computation without the inverse operation of Laplacian $\mathbf{\Delta}$. It should also be noted that our HGNN has great expansibility toward multi-modal feature with the flexibility of hyperedge generation.

Implementation

Hypergraph construction In our visual object classification task, the features of N visual object data can be represented as $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]^\top$. We build the hypergraph according to the distance between two features. More specifically, Euclidean distance is used to calculate $d(\mathbf{x}_i, \mathbf{x}_j)$. In the construction, each vertex represents one visual object, and each hyperedge is formed by connecting one vertex and its K nearest neighbors, which brings N hyperedges that links $K+1$ vertices. And thus, we get the incidence matrix $\mathbf{H} \in \mathbb{R}^{N\times N}$ with $N \times (K+1)$ entries equaling to 1 while others equaling to 0. In the citation network classification, where the data are organized in graph structure, each hyperedge is built by linking one vertex and their neighbors according to the adjacency relation on graph. So we also get N hyperedges and $\mathbf{H} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$.

Model for node classification In the problem of node classification, we build the HGNN model as in Figure 3. The dataset is divided into training data and test data. Then hypergraph is constructed as the section above, which generates the incidence matrix $\mathbf{H}$ and corresponding $\mathbf{D_e}$. We build a two-layer HGNN model to employ the powerful capacity of HGNN layer. And the softmax function is used to generate predicted labels. During training, the cross-entropy loss for the training data is back-propagated to update the parameters $\mathbf{\Theta}$ and in testing, the labels of test data is predicted for evaluating the performance. When there are multi-modal information incorporate them by the construction of hyperedge groups and then various hyperedges are fused together to model the complex relationship on data.

Experiments

In this section, we evaluate our proposed hypergraph neural networks on two task: citation network classification and visual object recognition. We also compare the proposed method with graph convolutional networks and other state-of-the-art methods.

Dataset	Cora	Pumbed
Nodes	2708	19717
Edges	5429	44338
Feature	1433	500
Training node	140	60
Validation node	500	500
Testing node	1000	1000
Classes	7	3

Table 1: Summary of the citation classification datasets.

Citation network classification

Datasets In this experiment, the task is to classify citation data. Here, two widely used citation network datasets, i.e., Cora and Pubmed (Sen et al. 2008) are employed. The experimental setup follows the settings in (Yang, Cohen, and Salakhutdinov 2016). In both of those two datasets, the feature for each data is the bag-of-words representation of documents. The data connection, i.e., the graph structure, indicates the citations among those data. To generate the hypergraph structure for HGNN, each time one vertex in the graph is selected as the centroid and its connected vertices are used to generate one hyperedge including the centroid itself. Through this we can obtain the same size incidence matrix compared with the original graph. It is noted that as there are no more information for data relationship, the generated hypergraph constructure is quite similar to the graph. The Cora dataset contains 2708 data and 5% are used as labeled data for training. The Pubmed dataset contains 19717 data, and only 0.3% are used for training. The detailed description for the two datasets listed in Table 1.

Experimental settings In this experiment, a two-layer HGNN is applied. The feature dimension of the hidden layer is set as 16 and the dropout (Srivastava et al. 2014) is employed to avoid overfitting with drop rate $p = 0.5$. We choose the ReLU as the nonlinear activation function. During the training process, we use Adam optimizer (Kingma and Ba 2014) to minimize our cross-entropy loss function with a learning rate of 0.001. We have also compared the proposed HGNN with recent methods in these experiments.

Results and discussion The results of the experimental results and comparisons on the citation network dataset are shown in Table 2. For our HGNN model, we report the average classification accuracy of 100 runs on Core and Pumbed,

which is 81.6% and 80.1%. As shown in the results, the proposed HGNN model can achieve the best or comparable performance compared with the state-of-the-art methods. Compared with GCN, the proposed HGNN method can achieve a slight improvement on the Cora dataset and 1.1% improvement on the Pubmed dataset. We note that the generated hypergraph structure is quite similar to the graph structure as there is neither extra nor more complex information in these data. Therefore, the gain obtained by HGNN is not very significant.

Method	Cora	Pubmed
DeepWalk (Perozzi, Al-Rfou, and Skiena 2014)	67.2%	65.3%
ICA (Lu and Getoor 2003)	75.1%	73.9%
Planetoid (Yang, Cohen, and Salakhutdinov 2016)	75.7%	77.2%
Chebyshev (Defferrard, Bresson, and Vandergheynst 2016)	81.2%	74.4%
GCN (Kipf and Welling 2017)	81.5%	79.0%
HGNN	**81.6**%	**80.1**%

Table 2: Classification results on the Cora and Pubmed datasets.

Visual object classification

Datasets and experimental settings In this experiment, the task is to classify visual objects. Two public benchmarks are employed here, including the Princeton ModelNet40 dataset (Wu et al. 2015) and the National Taiwan University (NTU) 3D model dataset (Chen et al. 2003), as shown in Table 3. The ModelNet40 dataset consists of 12,311 objects from 40 popular categories, and the same training/testing split is applied as introduced in (Wu et al. 2015), where 9,843 objects are used for training and 2,468 objects are used for testing. The NTU dataset is composed of 2,012 3D shapes from 67 categories, including car, chair, chess, chip, clock, cup, door, frame, pen, plant leaf and so on. In the NTU dataset, 80% data are used for training and the other 20% data are used for testing. In this experiment, each 3D object is represented by the extracted features. Here, two recent state-of-the-art shape representation methods are employed, including Multi-view Convolutional Neural Network (MVCNN) (Su et al. 2015) and Group-View Convolutional Neural Network (GVCNN) (Feng et al. 2018). These two methods are selected due to that they have shown satisfactory performance on 3D object representation. We follow the experimental settings of MVCNN and GVCNN to generate multiple views of each 3D object. Here, 12 virtual cameras are employed to capture views with a interval angle of 30 degree, and then both the MVCNN and the GVCNN features are extracted accordingly.

To compare with GCN method, it is noted that there is no available graph structure in the ModelNet40 dataset and the NTU dataset. Therefore, we construct a probability graph based on the distance of nodes. Given the features of data, the affinity matrix A is generated to represent the relationship among different vertices, and A_{ij} can be calculated by:

$$A_{ij} = \exp(-\frac{2Dij^2}{\Delta}) \tag{12}$$

where D_{ij} indicates the Euclidean distance between node i and node j. Δ is the average pairwise distance between nodes. For the GCN experiment with two features constructed simple graphs, we simply average the two modality adjacency matrices to get the fused graph structure for comparison.

Dataset	ModelNet40	NTU
Objects	12311	2012
MVCNN Feature	4096	4096
GVCNN Feature	2048	2048
Training node	9843	1639
Testing node	2468	373
Classes	40	67

Table 3: The detailed information of the ModelNet40 and the NTU datasets.

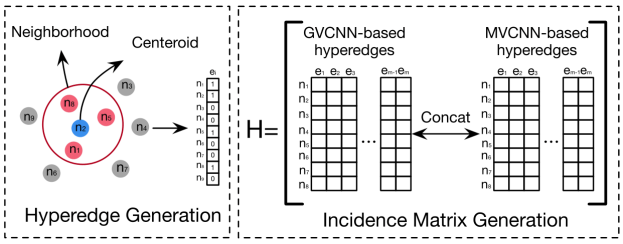

Figure 5: An example of hyperedge generation in the visual object classification task. Left: For each node we aggregate its N neighbor nodes by Euclidean distance to generate a hyperedge. Right: To generate the multi-modality hypergraph adjacent matrix we concatenate adjacent matrix of two modality.

Hypergraph structure construction on visual datasets In experiments on ModelNet40 and NTU datasets, two hypergraph construction methods are employed. The first one is based on single modality feature and the other one is based on multi-modality feature. In the first case, only one feature is used. Each time one object in the dataset is selected as the centroid, and its 10 nearest neighbors in the selected feature space are used to generate one hyperedge including the centroid itself, as shown in Figure 5. Then, a hypergraph $\mathcal{G}$ with N hyperedges can be constructed. In the second case, multiple features are used to generate a hypergraph $\mathcal{G}$ modeling complex multi-modality correlation. Here, for the i^{th} modality data, a hypergraph adjacent matrix $\mathbf{H_i}$ is constructed accordingly. After all the hypergraphs from different features have been generated, these adjacent matrices $\mathbf{H_i}$ can be concatenated to build the multi-modality hypergraph adjacent matrix $\mathbf{H}$. In this way, the hypergraphs using single modal feature and multi-modal features can be constructed.

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN (Feng et al. 2018)	91.8%	**92.6**%	91.5%	**91.8**%	92.8%	**96.6**%
MVCNN (Su et al. 2015)	92.5%	**92.9**%	86.7%	**91.0**%	92.3%	**96.6**%
GVCNN+MVCNN	-	-	-	-	94.4%	**96.7**%

Table 4: Comparison between GCN and HGNN on the ModelNet40 dataset.

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN ((Feng et al. 2018))	78.8%	**82.5**%	78.8%	**79.1**%	75.9%	**84.2**%
MVCNN ((Su et al. 2015))	74.0%	**77.2**%	71.3%	**75.6**%	73.2%	**83.6**%
GVCNN+MVCNN	–	–	–	–	76.1%	**84.2**%

Table 5: Comparison between GCN and HGNN on the NTU dataset.

Method	Classification Accuracy
PointNet (Qi et al. 2017a)	89.2%
PointNet++ (Qi et al. 2017b)	90.7%
PointCNN (Li et al. 2018)	91.8%
SO-Net (Li, Chen, and Lee 2018)	93.4%
HGNN	**96.7%**

Table 6: Experimental comparison among recent classification methods on ModelNet40 dataset.

Results and discussions Experiments and comparisons on the visual object recognition task are shown in Table 4 and Table 5, respectively. For the ModelNet40 dataset, we have compared the proposed method using two features with recent state-of-the-are methods in Table 6. As shown in the results, we can have the following observations:

1. The proposed HGNN method outperforms the state-of-the-art object recognition methods in the ModelNet40 dataset. More specifically, compared with PointCNN and SO-Net, the proposed HGNN method can achieve gains of 4.8% and 3.2%, respectively. These results demonstrate the superior performance of the proposed HGNN method on visual object recognition.

2. Compared with GCN, the proposed method achieves better performance in all experiments. As shown in Table 4 and Table 5, when only one feature is used for graph/hypergraph structure generation, HGNN can obtain slightly improvement. For example, when GVCNN is used as the object feature and MVCNN is used for graph/hypergraph structure generation, HGNN achieves gains of 0.3% and 2.0% compared with GCN on the ModelNet40 and the NTU datasets, respectively. When more features, i.e., both GVCNN and MVCNN, are used for graph/hypergraph structure generation, HGNN achieves much better performance compared with GCN. For example, HGNN achieves gains of 8.3%, 10.4% and 8.1% compared with GCN when GVCNN, MVCNN and GVCNN+MVCNN are used as the object features on the NTU dataset, respectively.

The better performance can be dedicated to the employed hypergraph structure. The hypergraph structure is able to convey complex and high-order correlations among data, which can better represent the underneath data relationship compared with graph structure or the methods without graph structure. Moreover, when multi-modal data/features are available, HGNN has the advantage of combining such multi-modal information in the same structure by its flexible hyperedges. Compared with traditional hypergraph learning methods, which may suffer from the high computational complexity and storage cost, the proposed HGNN framework is much more efficient through the hyperedge convolution operation.

Conclusion

In this paper, we propose a framework of hypergraph neural networks (HGNN). In this method, HGNN generalizes the convolution operation to the hypergraph learning process. The convolution on spectral domain is conducted with hypergraph Laplacian and further approximated by truncated chebyshev polynomials. HGNN is a more general framework which is able to handle the complex and high-order correlations through the hypergraph structure for representation learning compared with traditional graph. We have conducted experiments on citation network classification and visual object recognition tasks to evaluate the performance of the proposed HGNN method. Experimental results and comparisons with the state-of-the-art methods demonstrate better performance of the proposed HGNN model. HGNN is able to take complex data correlation into representation learning and thus lead to potential wide applications in many tasks, such as visual recognition, retrieval and data classification.

Acknowledgements

This work was supported by National Key R&D Program of China (Grant No. 2017YFC0113000, and No.2016YFB1001503), and National Natural Science Funds of China (No.U1705262, No.61772443, No.61572410, No.61671267), National Science and Technology Major Project (No. 2016ZX01038101), MIIT IT funds (Research and application of TCN key technologies) of China, and The National Key Technology R and D Program (No. 2015BAG14B01-02), Post Doctoral Innovative Talent Support Program under Grant BX201600094, China Post-Doctoral Science Foundation under Grant 2017M612134, Scientific Research Project of National Language Committee of China (Grant No. YB135-49), and Nature Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

References

Atwood, J., and Towsley, D. 2016. Diffusion-Convolutional Neural Networks. In *NIPS*, 1993–2001.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proc. ICLR*.

Chen, D.-Y.; Tian, X.-P.; Shen, Y.-T.; and Ouhyoung, M. 2003. On Visual Similarity Based 3D Model Retrieval. In *Computer Graphics Forum*, volume 22, 223–232. Wiley Online Library.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Gast Localized Spectral Filtering. In *NIPS*, 3844–3852.

Feng, Y.; Zhang, Z.; Zhao, X.; Ji, R.; and Gao, Y. 2018. Gvcnn: Group-View Convolutional Neural Networks for 3D Shape Recognition. In *Proc. CVPR*, 264–272.

Gao, Y.; Wang, M.; Tao, D.; Ji, R.; and Dai, Q. 2012. 3-D Object Retrieval and Recognition with Hypergraph Analysis. *IEEE Transactions on Image Processing* 21(9):4290–4303.

Gao, Y.; Wang, M.; Zha, Z.-J.; Shen, J.; Li, X.; and Wu, X. 2013. Visual-Textual Joint Relevance Learning for Tag-based Social Image Search. *IEEE Transactions on Image Processing* 22(1):363–376.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A New Model for Learning in Graph Domains. In *Proc. IJCNN*, volume 2, 729–734. IEEE.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep Convolutional Networks on Graph-Structured Data. *arXiv preprint arXiv:1506.05163*.

Huang, Y.; Liu, Q.; Zhang, S.; and Metaxas, D. N. 2010. Image Retrieval via Probabilistic Hypergraph Ranking. In *Proc. CVPR*, 3376–3383. IEEE.

Huang, Y.; Liu, Q.; and Metaxas, D. 2009.] Video Object Segmentation by Hypergraph Cut. In *Proc. CVPR*, 1738–1745. IEEE.

Hwang, T.; Tian, Z.; Kuangy, R.; and Kocher, J.-P. 2008. Learning on Weighted Hypergraphs to Integrate Protein Interactions and Gene Expressions for Cancer Outcome Prediction. In *Proc. ICDM*, 293–302. IEEE.

Kingma, D. P., and Ba, J. 2014. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*.

Kipf, T. N., and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. ICLR*.

Li, Y.; Bu, R.; Sun, M.; and Chen, B. 2018. PointCNN. In *NIPS*.

Li, J.; Chen, B. M.; and Lee, G. H. 2018. SO-Net: Self-Organizing Network for Point Cloud Analysis. In *Proc. CVPR*, 9397–9406.

Lu, Q., and Getoor, L. 2003. Link-based Classification. In *Proc. ICML*, 496–503.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *Proc. CVPR*, volume 1, 3.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online Learning of Social Representations. In *Proc. SIGKDD*, 701–710. ACM.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proc. CVPR* 1(2):4.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NIPS*, 5105–5114.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20(1):61–80.

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

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-View Convolutional Neural Networks for 3D Shape Recognition. In *Proc. ICCV*, 945–953.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention networks. In *Proc. ICLR*, volume 1.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proc. CVPR*, 1912–1920.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. *Proc. ICML*.

Zhou, D.; Huang, J.; and Schölkopf, B. 2007. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *NIPS*, 1601–1608.