Dynamic Hypergraph Neural Networks

Jianwen Jiang[1,2], **Yuxuan Wei**[1,2], **Yifan Feng**[3], **Jingxuan Cao**[1,2] and **Yue Gao**[1,2*]
[1]Beijing National Research Center for Information Science and Technology(BNRist)
[2]KLISS, School of Software, Tsinghua University, Beijing, China
[3]School of Information Science and Engineering, Xiamen University, Xiamen, China
{jjw17, weiyx15}@mails.tsinghua.edu.cn, evanfeng97@gmail.com, jingxuac@alumni.cmu.edu, gaoyue@tsinghua.edu.cn

Abstract

In recent years, graph/hypergraph-based deep learning methods have attracted much attention from researchers. These deep learning methods take graph/hypergraph structure as prior knowledge in the model. However, hidden and important relations are not directly represented in the inherent structure. To tackle this issue, we propose a dynamic hypergraph neural networks framework (DHGNN), which is composed of the stacked layers of two modules: dynamic hypergraph construction (DHG) and hypergrpah convolution (HGC). Considering initially constructed hypergraph is probably not a suitable representation for data, the DHG module dynamically updates hypergraph structure on each layer. Then hypergraph convolution is introduced to encode high-order data relations in a hypergraph structure. The HGC module includes two phases: vertex convolution and hyperedge convolution, which are designed to aggregate feature among vertices and hyperedges, respectively. We have evaluated our method on standard datasets, the Cora citation network and Microblog dataset. Our method outperforms state-of-the-art methods. More experiments are conducted to demonstrate the effectiveness and robustness of our method to diverse data distributions.

1 Introduction

Graphs are widely used to model pair-wise relations including paper citations, personal contacts and protein-protein interactions. However, besides pair-wise relations, there exists a large number of non-pair-wise relations that simple graphs are unable to model, for example, the communities in social networks and the clusters in feature embeddings. Hypergraph is a generalized structure for relation modeling. A hypergraph is composed of a vertex set and a hyperedge set, where a hyperedge contains a flexible number of vertices. Therefore, hyperedges are able to model non-pair-wise relations mentioned above. The number of vertices a hyperedge contains is defined as the degree of the hyperedge. Especially, if the degree of hyperedges is restricted to 2, a hypergraph is degenerated to a simple graph, indicating that simple graph is a subset of the hypergraph.

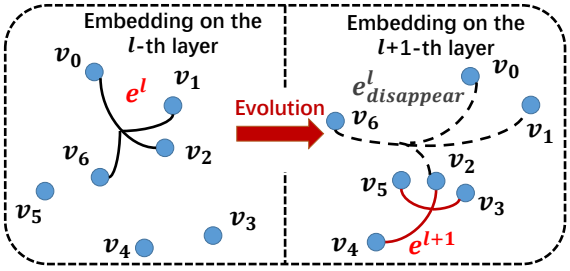

Figure 1: Dynamically constructed hyperedges. When embedding evolves from the l-th layer to the $l+1$-th layer, hyperedge $e^l = \{v_0, v_1, v_2, v_6\}$ disappears while hyperedge $e^{l+1} = \{v_2, v_3, v_4, v_5\}$ comes into existence.

Recently graph/hypergraph-based deep learning methods have received more and more attention from researchers. Inspired by convolutional neural network (CNN) [Krizhevsky *et al.*, 2012] in computer vision, researchers have designed graph-based neural networks for semi-supervised learning, like GCN [Kipf and Welling, 2017] and GAT [Veličković *et al.*, 2018]. Furthermore, HGNN [Feng *et al.*, 2018] is the first hypergraph neural network model. In a neural network model, feature embedding generated from deeper layer of the network carries higher-order relations that initial structure fails to capture. The major drawback of existing graph/hypergraph-based neural networks is that they only employ the initial graph/hypergraph structures while neglect the dynamic modifications of such structures from adjusted feature embedding.

Dynamic hypergraph structure learning (DHSL) [Zhang *et al.*, 2018] has been proposed to deal with this problem. DHSL uses raw input data to optimize hypergraph structure iteratively. Nonetheless, DHSL only updates the hypergraph structure on initial feature embedding, thus failing to exploit high-order relations among features. Also, the iterative optimization in DHSL suffers from expensive cost in time and space.

To tackle these issues, we propose a dynamic hypergraph neural networks (DHGNN) framework, which is stacked layers of dynamic hypergraph construction (DHG) module and

*Corresponding author.

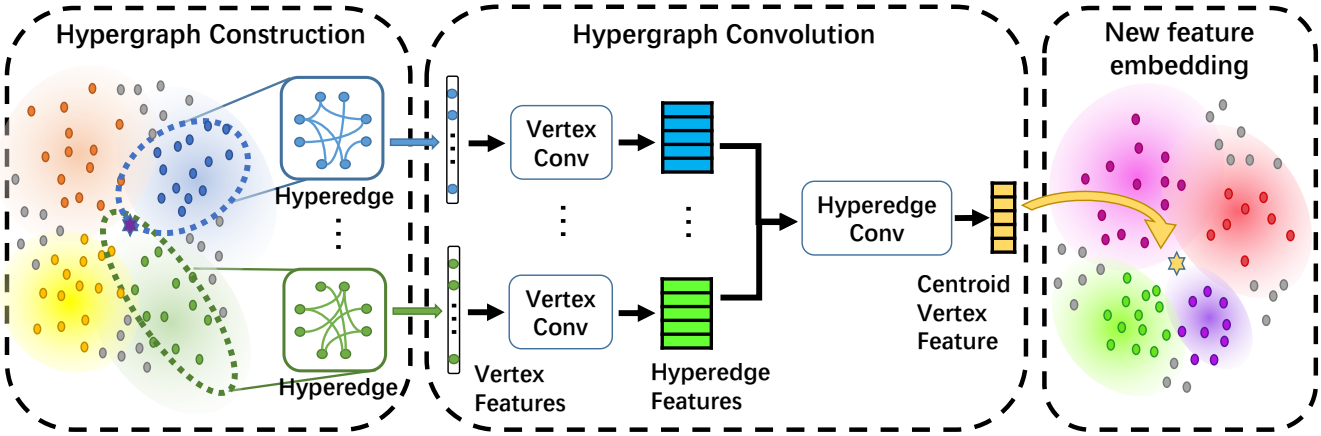

Figure 2: DHGNN framework. The first frame describes the hypergraph construction process on centroid vertex (the star) and its neighbors. For instance, two hyperedges are generated from two clusters (dashed ellipses). In the second frame, features of vertices contained in a hyperedge are aggregated to hyperedge feature through vertex convolution and features of adjacent hyperedges are aggregated to centroid vertex feature through hyperedge convolutoin. After performing such operations for all vertices on current layer feature embedding, we obtain the new feature embedding where new hypergraph structure will be constructed, as is shown in the third frame.

hypergraph convolution (HGC) module. In DHG module, we utilize k-NN method and k-means clustering method to update hypergraph structure based on local and global features respectively during a single inference process. Furthermore, we propose a hypergraph convolution method in HGC module by a stack of vertex convolution and hyperedge convolution. For vertex convolution, we use a transform matrix to permute and weight vertices in a hyperedge; for hyperedge convolution, we utilize attention mechanism to aggregate adjacent hyperedge features to the centroid vertex. Compared with hypergraph-based deep learning method HGNN, our convolution module better fuses information from local and global features provided by our DHG module.

We have applied our model to data with and without inherent graph structure. For data with inherent graph structure, we conducted an experiment on a citation network benchmark, the Cora dataset [Sen *et al.*, 2008], for the node classification task. In this experiment, we used DHGNN to jointly learn embeddings from given graph structure and a hypergraph structure from feature space. For data without inherent graph structure, an experiment was conducted on a social media dataset, the Microblog dataset [Ji *et al.*, 2019], for the sentiment prediction task. In this experiment, a multi-hypergrpah was constructed to model the complex relations among multimodal data.

Our contributions are summarized as follows:

1. We propose a dynamic hypergraph construction method, which adopts k-NN method to generate basic hyperedge and extends adjacent hyperedge set by clustering algorithm, i.e., k-means clustering. By dynamic hypergraph construction method, local and global relations will be extracted.
2. We conducted experiments on network-based classification and social media sentiment prediction. On the network-based task, our method outperforms state-of-the-art methods and shows higher robustness to different data distributions. On social media sentiment prediction, we observe performance improvement against state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 introduces related work in graph-based deep learning and hypergraph learning. Section 3 explains the proposed dynamic hypergraph neural networks method. Applications and experimental results are presented in Section 4. Finally, we draw conclusions in Section 5.

2 Related Work

In this section, we give a brief review on graph-based deep learning and hypergraph learning.

2.1 Graph-based Deep Learning

Semi-supervised learning on graphs has long been an active research field in deep learning. DeepWalk [Perozzi *et al.*, 2014] and Planetoid [Yang *et al.*, 2016] view sampled paths in graphs as random sequences and learn vector embedding from these sequences.

After the great success of convolutional neural networks [Krizhevsky *et al.*, 2012] in image processing, researchers have been devoted to designing convolutional methods for graph-based data. Existing graph neural network methods can be divided in two main categories: spectral methods and spatial methods.

Based on spectral graph theory, spectral graph convolutional methods use graph Laplacian eigenvectors as graph Fourier basis. After transforming features to spectral domain, a spectral convolution operation is conducted on spectral features. To overcome the expensive computation cost in Laplacian factorization, ChebyshevNet introduces Chebyshev polynomials to approximate Laplacian eigenvectors [Defferrard *et al.*, 2016]. GCN further simplifies the process and uses one-order polynomial on each layer [Kipf and Welling, 2017].

Different from spectral methods, spatial graph convolution methods leverage spatial sampler and aggregator to generate neighborhood feature embedding. MoNet defines a generic spatial convolution framework for deep learning on non-Euclidean domains [Monti *et al.*, 2017]. GraphSAGE defines sampler and aggregator in graph neural network and tries LSTM as neighborhood aggregator [Hamilton *et al.*, 2017]. GAT introduces self-attention mechanism and computes the attention coefficients between node pairs [Veličković *et al.*, 2018]. In the field of computer vision, DGCNN, a 3D point cloud learning method, also leverages the concept of spatial graph convolution in its model [Wang *et al.*, 2018].

2.2 Hypergraph Learning

Hypergraph learning is first introduced by [Zhou *et al.*, 2007] as a label propagation method for semi-supervised learning. This method aims to minimize the differences in labels of vertices that share the same hyperedge. [Huang *et al.*, 2009] discusses the construction methods of hypergraph, including k-NN method and search radius method. More recent works concentrate on the learning of hyperedge weight, intending to assign larger weight to hyperedges or sub-hypergraphs with higher importance [Gao *et al.*, 2012]. Besides learning label propagation on hypergraph, dynamic hypergraph structure learning proposes learning of hypergraph structure by a dual optimization process [Zhang *et al.*, 2018]. Like graph neural network, hypergraph neural network (HGNN) has been proposed as the first deep learning method on hypergraph structure, employing hypergraph Laplacian to represent hypergraph from spectral perspective [Feng *et al.*, 2018].

Hypergraph has many aspects of applications. In computer vision, hypergraph is used to describe relations among visual features for tasks like visual classification [Wang *et al.*, 2015], image retrieval [Huang *et al.*, 2010] and video object segmentation [Huang *et al.*, 2009]. There are also works using hypergraph structure for label propagation on 3D model classification [Zhang *et al.*, 2018]. In social media, MHG [Chen *et al.*, 2015] and Bi-MHG [Ji *et al.*, 2019] are proposed to deal with multimodal data.

3 Dynamic Hypergraph Neural Networks

In this section, we introduce the dynamic hypergraph neural networks (DHGNN) proposed in detail. As is illustrated in Figure 2, a DHGNN layer consists of two major part: dynamic hypergraph construction (DHG) and hypergraph convolution (HGC). We will first introduce these two parts in the following subsections and then discuss the implementation of dynamic hypergraph neural networks in the last subsection.

3.1 Dynamic Hypergraph Construction

Given feature embedding $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_n]$ where $\mathbf{x}_i (i = 1, 2, ..., n)$ denotes the feature of the i-th sample, we construct hypergraph $\mathcal{G}$. In hypergraph, vertex u denotes a sample and hyperedge e denotes a sample collection containing a flexible number of samples. Therefore, a hypergraph can be formulated as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the vertex set and $\mathcal{E}$ denotes the hyperedge set.

Algorithm 1 Hypergraph Construction

Input: Input embedding $\mathbf{X}$; hyperedge size k; adjacent hyperedge set size S
Output: Hyperedge set $\mathcal{G}$
Function: k-Means clustering $kMeans$; k-nearest neighborhood selection knn; distance function dis; $S-1$ smallest distance index selection $topK$

```
C = kMeans(X)
for u in range(len(X)) do
    e_b = knn(X[u], X, k)
    G[u].insert(e_b)
    D = dis(C.center, u)
    D = sort(D)
    ind = topK(D, S − 1)
    for i in ind do
        G[u].insert(C[i])
    end for
end for
```

We use symbol $Con(e)$ to denote the vertex set a hyperedge e contains and use symbol $Adj(u)$ to denote the hyperedge set composed of all hyperedge containing the vertex u, which is formulated as:

$$Con(e) = \{u_1, u_2, ..., u_{k_e}\} \tag{1}$$

$$Adj(u) = \{e_1, e_2, ..., e_{k_u}\} \tag{2}$$

where k_e and k_u is the number of vertices in hyperedge e and the number of hyperedges containing vertex u. Vertex u is defined as the *centroid* vertex of hyperedge set $Adj(u)$.

We combine k-NN methods and k-means clustering methods for dynamic hypergraph construction to exploit local and global structure. On one hand, we have computed the $k-1$ nearest neighbors for each vertex u. These neighborhood vertices, along with the vertex u, form a hyperedge in $Adj(u)$. On the other hand, we have conducted k-means algorithm on the whole feature map of each layer according to Euclidean distance. For each vertex, the nearest $S-1$ clusters will be assigned as the adjacent hyperedges of this vertex. The detailed procedure is described in Algorithm 1.

We perform such procedure on the feature embedding of each layer. Especially, we initialize hypergraph structure with the input feature embedding. Therefore, the hyperedge set is dynamically adjusted as the feature embedding evolves with network going deeper. In this way, we are able to obtain better hypergraph struture for high-order data relation modeling with deep neural network.

3.2 Hypergraph Convolution

Hypergraph convolution is composed of two submodules: vertex convolution submodule and hyperedge convolution submodule. Vertex convolution aggregates vertex features to hyperedge and then hyperedge convolution aggregates adjacent hyperedge features to centroid vertex by hyperedge convolution.

Vertex Convolution
Vertex convolution aggregates vertex features to the hyperedge containing these vertices. A simple solution is pooling

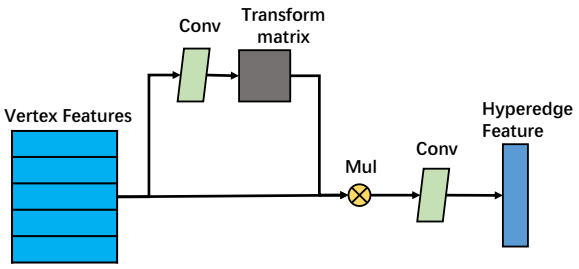

Figure 3: Vertex convolution module. For k vertices, a $k \times k$ transform matrix is computed by convolution. We multiply transform matrix and input vertex feature matrix to get permuted and weighted vertex feature matrix. Then a 1-dimension convolution is conducted for the 1-dimension hyperedge feature.

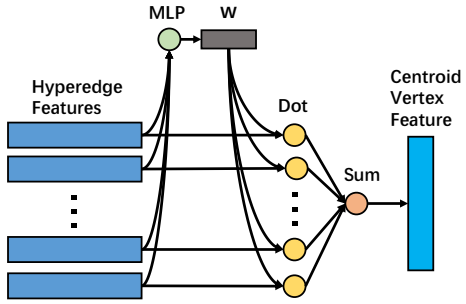

Figure 4: Hyperedge convolution module. We use self-attention mechanism to aggregate hyperedge features. Self-attention weight w is computed through multi-layer perception (MLP) from hyperedge features. Then weighted average of hyperedge features is calculated as centroid vertex feature.

method like max pooling and average pooling. State-of-the-art methods employ a fixed, pre-computed transform matrix generated from graph/hypergraph structure for vertex aggregation. However, such methods are unable to model discriminative information among vertices features well. To tackle this issue, we learn the transform matrix $\mathbf{T}$ from the vertex features for feature permutation and weighting, as is shown in Figure 3. The transform matrix enables both inter-vertex and inter-channel information flow. In implementation, we use multi-layer perception (MLP) to generate transform matrix $\mathbf{T}$ and use 1-d convolution to compact the transformed features, as is described by equation 3.

$$\mathbf{T} = MLP(\mathbf{X}_u) \tag{3a}$$
$$\mathbf{x}_e = conv(\mathbf{T} \cdot MLP(\mathbf{X}_u)) \tag{3b}$$

Hyperedge Convolution

Hyperedge convolution aggregates hyperedge features to centroid vertex feature, as is illustrated in Figure 4. Attension mechanism is employed in hyperedge convolution, using multi-layer perception (MLP) to generate weight score of each hyperedge. The output centroid vertex feature is computed as a weighted sum of input hyperedge features. The procedure can be formulated as follows:

$$w = softmax(\mathbf{x}_e\mathbf{W} + \mathbf{b}) \tag{4a}$$
$$\mathbf{x}_u = \sum_{i=0}^{|Adj(u)|} w^i \mathbf{x}_e^i \tag{4b}$$

$|Adj(u)|$ denotes the size of adjacent hyperedge set, $\mathbf{x}_e$ denotes adjacent hyperedge features and $\mathbf{x}_u$ denotes centroid vertex feature. $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters.

Algorithm 2 Hypergraph Convolution

Input: Input sample $\mathbf{x}_u$; hypergraph structure $\mathcal{G}$
Output: Output sample $\mathbf{y}_u$

```
xlist = Φ
for e in Adj(u) do
    X_v = VertexSample(X, G)
    x_e = VertexConv(X_v)
    xlist.insert(x_e)
end for
X_e = stack(xlist)
x_u = EdgeConv(X_e)
y_u = σ(x_u W + b)
```

3.3 Dynamic Hypergraph Neural Networks

By combining vertex convolution and hyperedge convolution, we describe a hypergraph convolutional layer as Algorithm 2. For each hyperedge e in $Adj(u)$, we first sample k vertices in e and obtain $\mathbf{X}_u \in \mathbb{R}^{k\times d}$, where d is input dimension of feature. Vertex Convolution $VertexConv$ transforms vertex features $\mathbf{X}_u$ to hyperedge feature $\mathbf{x}_e \in \mathbb{R}^d$. Aftering performing $|Adj(u)|$ vertex convolution, we stack $|Adj(u)|$ hyperedge features to get adjacent feature matrix $\mathbf{X}_e \in \mathbb{R}^{|Adj(u)|\times d}$. Then Hyperedge Convolution $EdgeConv$ aggregates adjacent hyperedge features to feature $\mathbf{x}_u$ of vertex u. At last, $\mathbf{x}_u$ is updated to $\mathbf{y}_u$ by a non-linear activation function σ following a linear layer.

A DHGNN model is composed of a stack of several layers of dynamic hypergraph construction construction module and hypergraph convolution module. As is shown in Figure 2, a hypergrpah convolutional layer updates vertex features for new feature embedding, based on which a new hypergraph structure will be constructed.

4 Applications and Experiments

In this section, we applied our dynamic hypergraph neural network to two types of data: citation network with inherent graph structure and social media multimodal data without inherent data structure. Especially, for data with inherent graph structure, we sample k vertices in 1-order neighborhood of u and these k vertices also form a hyperedge in $Adj(u)$.

4.1 Experiments on Citation Network

Citation network is a typical graph-structure dataset. Since graph is a special case of hypergraph, our model applies to graph-structure data as well. Furthermore, by constructing

dynamic hypergraph, we are able to deploy information from both citation relation and feature embedding relation in a uniform manner. To evaluate the performance of our method, series of experiments were conducted on the public benchmark citation network dataset, Cora dataset.

Cora dataset. Cora dataset is a benchmark dataset of citation network. In Cora dataset, there are 2,708 vertices denoting academic papers and 5,429 edges denoting citation relations between pairs of papers. Each vertex has a bag-of-word feature vector and a category label indicating the subject that the paper belongs to. There are 7 categories in total.

Experimental setup. We have conducted experiments on different splits of the Cora dataset including standard split described in [Yang *et al.*, 2016]. Because the standard split uses fixed training samples with 5.2% of dataset, it is possible for a method to be influenced by the fixed data distribution. Therefore, for further comparison, we randomly sampled different proportion of data as training set to demonstrate the effectiveness of our method. The proportion for training set is selected as 2%, 5.2%, 10%, 20%, 30% and 44%, respectively. We compared our method with recent representative methods like GCN [Kipf and Welling, 2017], HGNN [Feng *et al.*, 2018] and GAT [Veličković *et al.*, 2018]. 10 times average accuracy was reported in Table 1 for comparison.

We used 2 layer dynamic hypergraph neural network with a GCN-style input layer for feature dimension reduction. We used 400 cluster centers in k-means clustering method and chose 64 as the receptive field size. We added two dropout layers with the dropout rate of 0.5 before two convolutional layers.

Semi-supervised node classification. We compared DHGNN with most recent graph/hypergraph-based neural network methods on different dataset splits. Experimental results are listed as Table 1, showing that our method outperformed state-of-the-art by 1.5%, 0.5%, 0.1%, 0.1%, 0.5%, 0.4% when 2%, 5.2%, 10%, 20%, 30%, 44% randomly sampled data was used as training set, respectively. Moreover, we observed that hypergraph structure was relatively more competitive when training set was smaller. The reason is that graph convolution only uses 1-order adjacent relation while hypergraph convolution utilizes high-order relation, which is helpful to the label propagation process on a sparsely labelled graph.

Ablation experiments. To evaluate the effectiveness of proposed dynamic hypergraph construction (DHG) module and hypergraph convolution (HGC) module, we conducted two ablation experiments on the Cora dataset, where each module mentioned above was removed from the complete model. We compared the ablated models against the complete model and investigated the influence of hyperparameter k, which denotes the number of sampled vertices in a hyperedge. Results are shown in Figure 5. From the results, we observe that DHG module and HGC module always improve the performance of baseline with different k. As k increases, the gain from both modules increases, indicating the effectiveness of our method. It is noted that even when $k = 4$ (is much smaller than the max degree in the Cora dataset, 169), our method still obtains similar performance with other hypergraph-based learning method, i.e., HGNN and when $k = 128$, our method outperforms HGNN by 0.9%. This implies that our method is able to aggregate neighborhood features better.

lr	#train	GCN	HGNN	GAT	DHGNN
std	140	81.5%	81.6%	**83.0%**	82.5%
2%	54	69.6%	75.4%	74.8%	**76.9%**
5.2%	140	77.8%	79.7%	79.4%	**80.2%**
10%	270	79.9%	80.0%	81.5%	**81.6%**
20%	540	81.4%	80.1%	83.5%	**83.6%**
30%	812	81.9%	82.0%	84.5%	**85.0%**
44%	1200	82.0%	81.9%	85.2%	**85.6%**

Table 1: Performance comparisons on Cora with different splits. "lr" stands for label rate, "#train" stands for number of training samples and "std" stands for standard split. Standard split experiment and 5.2% split experiment share the same number of training samples. Different from the standard split setting, samples in 5.2% split is randomly selected. 44% is the largest possible size of training set with standard validation and test set.

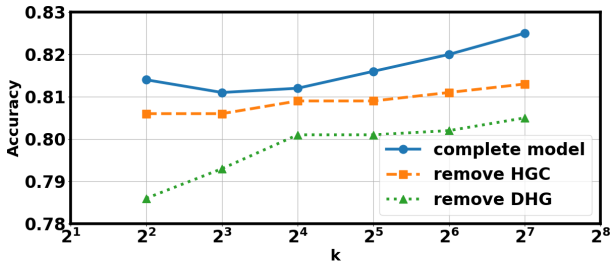

Figure 5: Ablation experiment on dynamic hypergraph(DHG) module and hypergraph convolution (HGC) module. For model without dynamic hypergraph, we used inherent graph structure on Cora for convolution operation. For model without hypergraph convolution, we used average pooling for substitution of vertex convolution and hyperedge convolution.

4.2 Microblog Sentiment Prediction

Apart from experiments on citation network, we also evaluated our model on a more complicated task, social media sentiment prediction. Multi-modality is an important feature of social media. We used hypergraph to model the high-order relations among different modalities. Specifically, vertices were used to denote a tweet. Hyperedge sets were constructed according to each modality feature and hyperedges from multi-modalities jointly represent the correlation between vertices. In our experiment, we used the Microblog dataset to evaluate our hypergraph model.

Microblog dataset. The Microblog dataset contains 5,550 tweets crawled from Sina Microblog platform [1] during Feb. 2014 to Apr. 2014. Each tweet has three modalities: text, image and emoticon. We generated 2547-dimension bag-of-words textual feature using Chinese auto-segmentation system ICTCLAS [Zhang *et al.*, 2003]. To generate visual feature, we used SentiBank [Borth *et al.*, 2013], a kind

[1] https://www.weibo.com

of ANP detector library pre-trained on Twitter images to transform Microblog images to 1553-dimension feature vector. For emoticons, we built a emoticons dictionary with 49 frequently-used emoticons and computed bag-of-emoticons features. Each tweet has a label indicating its emotional polarity (postive or negative). The task is to predict tweet emotional polarity by multimodal features. There are 4196 positive tweets and 1354 negative tweets in the dataset.

Experimental setup. We followed the experimental setup in [Ji et al.], where 4,650, 400, 500 tweets were randomly selected as training, validation and test set, respectively. 10-times average accuracy was reported for method evaluation. We used 2 layer dynamic hypergraph neural network with a multi-input fully-connected layer for feature dimension reduction. Dimensions of each modality feature were reduced to 32 before hypergraph convolution. We constructed three hyperedge sets for three modality respectively and merged these sets as one multimoal hyperedge set. For each modality, we use 400 cluster centers in k-means clustering method and the number of vertices contained is 8 in each clusters. We select 2 nearest clusters from k-means clusters and one k-NN cluster as the adjacent hyperedge set of each vertex. We use identical activation and dropout setting with Section 4.1. We compare our model with recent approaches for multimodal sentiment prediction, such as Multi-kernel SVM [Zhang *et al.*, 2011], Cross-media Bag-of-words Model [Wang *et al.*, 2014], Bi-layer Multimodal Hypergraph Learning [Ji *et al.*, 2019], etc. Experiments were conducted on a Nvidia GeForce GTX 1080 Ti GPU with 11G memory and 10.6 T-flops computing capacity.

Microblog sentiment prediction. In this experiment, we ran DHGNN to fuse features from mutliple modalities for sentiment label prediction. Experimental results are shown in Table 2, which can be summarized as:

1. In terms of prediction accuracy, DHGNN achieved higher performance with 1.8% accuracy gains in multimodal sentiment prediction task compared with current state-of-the-art method.
2. In terms of time expense, DHGNN remarkably shortened training time compared with current state-of-the-art methods (2300 times as fast as Bi-MHG and 1.4 times as fast as HGNN).

Experimental results indicate that our method outperformed the state-of-the-art method in both prediction accuracy and training speed. The experiments on Microblog dataset demonstrates the effectiveness of our method in modeling the high-order relations among multimodal data.

4.3 Discussion

Discussion on accuracy. Compared with statically initialized hypergraph structure, dynamic hypergraph structure can better represent data distribution in deeper layers. Compared with pooling and multi-layer perception, hypergraph convolution uses fixed-size, weight-shared learnable convolution kernel for feature extraction, thus being better for information aggregation. Ablation experiments demonstrate the effectiveness of the dynamic hypergraph construction and hypergraph convolution respectively. Despite of this, we also note that in the standard split of Cora dataset, GAT performs better than DHGNN. The main reason is that in standard split, training set contains fixed samples, thus suffering from larger randomness and bias. On the other settings, we have randomly sampled training set for 10 times and reported the average accuracy for comparison to suppress such randomness and bias.

Method	Acc	Train Time
CBM-NB [Wang *et al.*, 2014]	71.6%	-
CBM-LR [Wang *et al.*, 2014]	79.9%	-
CBM-SVM [Wang *et al.*, 2014]	81.6%	-
HGNN [Feng *et al.*, 2018]	86.8%	2m11s
MHG_noW [Chen *et al.*, 2015]	87.3%	-
MHG [Chen *et al.*, 2015]	88.6%	-
MMHG [Chen *et al.*, 2015]	88.7%	-
Bi-MHG [Ji *et al.*, 2019]	90.0%	58.5h
DHGNN (our method)	**91.8%**	1m32s

Table 2: Performance comparisons on the Microblog dataset

Discussion on time complexity. Traditional hypergraph learning models like Bi-MHG involve iterative optimization and matrix inversion, thus suffering from larger time cost than neural network model. When comparing HGNN and DHGNN, we find that parameter number of both models 0.133M, indicating that it takes roughly the same time to train a HGNN/DHGNN epoch. However, it takes 30 epochs on average for DHGNN to converge on Microblog sentiment dataset while it takes 200 epochs on average for HGNN to converge. Therefore, DHGNN runs faster on Microblog sentiment dataset.

5 Conclusions

In this paper, we propose a dynamic hypergraph neural networks framework to update hypergraph structure on each layer. The method consists of two important modules: dynamic hypergraph construction method and hypergraph convolution, where hypergraph convolution includes vertex convolution and hyperedge convolution for hypergraph neighborhood feature aggregation. We apply our model to citation network data and multimodal social media data for evaluation. Results demonstrate that our model achieves similar of better performance compared with state-of-the-art methods and is more robust to different data distributions. We also investigate the effectiveness of dynamic hypergraph construction module and hypergraph convolution module independently by ablation experiment. In our model, k-NN method and k-means clustering method are used in hypergraph dynamic construction. Future work can concentrate on better and more interpretable hypergraph construction methods.

Acknowledgments

This work was supported by National Natural Science Funds of China (U1701262, U1801263, 61671267).

References

[Borth *et al.*, 2013] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 223–232. ACM, 2013.

[Chen *et al.*, 2015] Fuhai Chen, Yue Gao, Donglin Cao, and Rongrong Ji. Multimodal hypergraph learning for microblog sentiment prediction. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[Feng *et al.*, 2018] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *AAAI 2019*, 2018.

[Gao *et al.*, 2012] Yue Gao, Meng Wang, Dacheng Tao, Rongrong Ji, and Qionghai Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9):4290–4303, 2012.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[Huang *et al.*, 2009] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas. Video object segmentation by hypergraph cut. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1738–1745. IEEE, 2009.

[Huang *et al.*, 2010] Yuchi Huang, Qingshan Liu, Shaoting Zhang, and Dimitris N Metaxas. Image retrieval via probabilistic hypergraph ranking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3376–3383. IEEE, 2010.

[Ji *et al.*, 2019] Rongrong Ji, Fuhai Chen, Liujuan Cao, and Yue Gao. Cross-modality microblog sentiment prediction via bi-layer multimodal hypergraph learning. *IEEE Transactions on Multimedia*, pages 1062–1075, 2019.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Monti *et al.*, 2017] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael 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*, pages 5115–5124, 2017.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

[Wang *et al.*, 2014] Min Wang, Donglin Cao, Lingxiao Li, Shaozi Li, and Rongrong Ji. Microblog sentiment analysis based on cross-media bag-of-words model. In *Proceedings of international conference on internet multimedia computing and service*, page 76. ACM, 2014.

[Wang *et al.*, 2015] Meng Wang, Xueliang Liu, and Xindong Wu. Visual classification by l_1-hypergraph modeling. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2564–2574, 2015.

[Wang *et al.*, 2018] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[Yang *et al.*, 2016] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ICML*, 2016.

[Zhang *et al.*, 2003] Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 184–187. Association for Computational Linguistics, 2003.

[Zhang *et al.*, 2011] Daoqiang Zhang, Yaping Wang, Luping Zhou, Hong Yuan, Dinggang Shen, Alzheimer's Disease Neuroimaging Initiative, et al. Multimodal classification of alzheimer's disease and mild cognitive impairment. *Neuroimage*, 55(3):856–867, 2011.

[Zhang *et al.*, 2018] Zizhao Zhang, Haojie Lin, Yue Gao, and KLISS BNRist. Dynamic hypergraph structure learning. In *IJCAI*, pages 3162–3169, 2018.

[Zhou *et al.*, 2007] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608, 2007.