HYPER-SAGNN: A SELF-ATTENTION BASED GRAPH NEURAL NETWORK FOR HYPERGRAPHS

(Machine Learning Term Project)

Ans Munir
Information Technology University
Lahore
msds20033@itu.edu.pk

Dr. Ali Ahmed
Information Technology University
Lahore
(Instructor)

Abstract

To extract the patterns from the hypergraph that has higher order interaction we used graph representation learning. Currently the previous models can not handle hypergraphs that has different sizes and types of hyperedges. In reality the models that can predict hyperedge on homogeneous and heterogeneous hypergraphs with variable number of nodes are not available. In this work we have showed our model, Hyper-SAGNN that is graph neural network based on self attention mechanism that can predict hyperedge that has different types and number of nodes. We have compared our Hyper-SAGNN model with other previously available best models and showed that our model performs better than those models.

1. Introduction

In order to show interaction between higher order complex objects we often use graph structure. These graphs structures consists of nodes that are connected together with edges as shown in figure 1.

The nodes also known as vertices represents the real world objects and the edges represents the interaction between them. Very popular example of Graph structure is a social media network. In social media network each person is represented by a node and interaction between two persons is represented by an edge. It is shown in the figure 2

1.1. Problem Statement

There are many existing techniques that are very effective and accurate in node prediction and the link prediction between these nodes. Most of these techniques of link prediction and node prediction are for pair-wise nodes. But in real life this interaction is higher order and not limited to pair-wise nodes. Lets take an example of social media network in which interaction of people mostly lies in higher order.

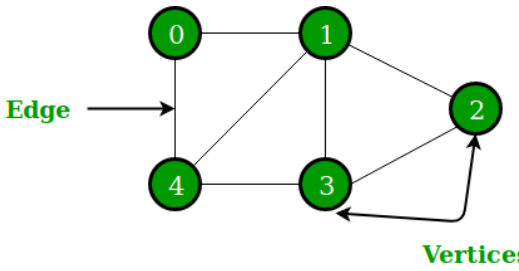

Figure 1. A graph consists of nodes (also known as vertices) and edges connected those vertices. Image Source: Internet

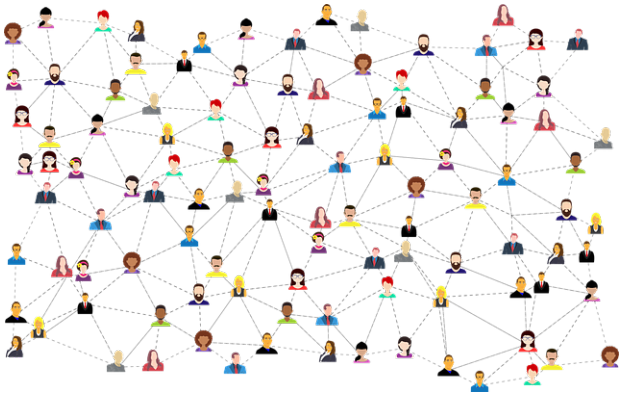

Figure 2. A graph represents the interaction between different people on social media. Image Source: Internet

Similarly, there are situations where there are different objects involved in a relationship like (human, location, activity). To represent this higher order interaction we use hypergraphs. Hypergraphs has hyperedge that represents all the objects interacting with each other.

A simple approach to analyze this high order interaction is to flatten the hyperedge into pair-wise interaction and do-

ing so we assume that this hyperedge is decomposable. One previous study [8] suggested that each hyperedge can not be decomposable.

Different Hyperedges in an hypergraph can consisits of different number and types of nodes. Therefore we need methods that can handle these hyperedges that has different numbers and different node types. But unfortunately, there is no method that can learn embeddings of an hypergraph and that can also predict hyperedges that has different types and sizes.

1.2. Objectives

Our main objective is to design a model that can predict an hyperedge that has different size and types of nodes. This model is known as Hyper-SAGNN, a Graph Neural Network based on self attention mechanism. Given a set of nodes our model can predict whether they will make an hyperedge or not.

1.3. Limitations and Scope

Currently our model uses the set of nodes given in the tuple to calculate static and dynamic embedding. Therefore it is dependent on the nodes that are present in a given tuple to predict whether those nodes will make a hyperedge or not.

2. Literature Review

Previously deep learning based methods were available to work with graph structures. Now there are also some deep learning methods available that can handle hypergraphs. The Hyperedge Based Embeddings (HEBE) method [5] learns the embeddings of each node by representing it as a hyperedge in a specific heterogeneous event. But [9] showed that HEBE does not show good results when hypergraph is sparse.

As stated above, previous methods divide the hyperedge into pair-wise relationship. There are two methods to divide the hyperedge into pair-wise relationship. One is explicit and other is implicit. For example, if there is a hyperedge (a_1, a_2, a_3) explicit approach would divide this hyperedge into following edges $(a_1, a_2), (a_2, a_3), (a_3, a_1)$. However implicit approach on the other-hand adds a hidden node z and divides the hyperedge as $(a_1, z), (a_2, z), (a_3, z)$.

The Deep Hypergraph Embedding (DHNE) directly model the hyperedge using Multi-layer perceptron (MLP). This DHNE model achieves better performance than other methods like Deepwalk [2], node2vec [4] and HEBE. However, DHNE uses MLP that takes input of fixed-size. Therefore, DHNE only works with s-uniform hypergraphs i.e. hyperedges that has same s nodes. If we want to apply DHNE on hyperedges that has differnt types of nodes or different number of nodes in different hyperedges then we need to train the DHNE model for each type of hyperedge. This will require a lot of computational power and resources.

Heterogeneous Hyper-network [6] Embedding also use Multi-layer perceptron to. Like DHNE model it also require training for each type of hyperedge. Another method, hyper2vec [7] was proposed to generate the node embeddings in the hyperedge. Hyper2vec surpass other embedding generation methods for node classification like HGE [3]. But hyper2vec can't able to predict links because hyper2vec generates the node embeddings only and do not learn a function that can then predict the hyperedge from the nodes. There are some other GNN based methods like [10], [15], [12] use convolutional operation and attention mechanism extended from graphs to hypergraphs. However, like hyper2vec these methods also do not predict hyperedge directly.

3. Methodology

Hypergraph We can define hypergraph as $G = (V, E)$ $where$ $V = \{v_1, ..., v_n\}$ represents hypergraph nodes and $E = \{e_i = (v_1^{(i)}, ..., v_k^{(i)})\}$ shows the set of hyperedge in the hypergraph.

If the hypergraph has same size of hyperedges i.e. each hyperedge contains same number of nodes in it lets say k nodes in each hyperedge then we will call it as k-uniform hypergraph. Even if a hypergraph has same number of nodes in each hyperedge still these hyperedges can have different node types. Therefore if a hypergraph has same type of nodes in each hyperedge then we call it as homogeneous hypergraph. However if a hypergraph has different types of nodes in a hyperedge we will call it as heterogeneous hypergraph.

Problem of Hypergraph Prediction We can represent the problem of hypergraph prediction as if we have been given a set of nodes $(v_1, v_2, ..., v_k)$ our task is to learn function to which when we give a set of nodes it gives a value. If that value is greater than specific threshold then we say that those set of nodes will form a hyperedge and if it will less than that particular threshold we say that those set of nodes will not form a hyperedge.

3.1. Hyper-SAGNN Structure

Here we want to learn a f that takes input of features of nodes $f(v_1, v_2, ..., v_k)$ and gives the probability that whether those features will form a hyperedge or not. In this model we do not assume that input should have homogeneous nodes or are of same size. These nodes can be of different types and given tuple can of different sizes.

GNN based methods like GraphSAGE [14] normally define a special computational graph for every node. This method allows GraphSAGE to aggregate information for nodes that has different node degrees. Graph Attention

Network (GAT) [11] aggregates the information using self attention mechanism. To learn the function f our model Hyper-SAGNN uses mechanism of self-attention inside every set of nodes.

For demonstration of mechanism of self-attention we have used the similar notations as used in [11], [1]. If we have given nodes set and matrices of weight then to show the pair-wise importance of nodes we first need to compute the attention coefficients.

$$e_{ij} = (W_Q^T x_i)^T (W_K^T x_j), \quad \forall 1 \leq i, j \leq k \tag{1}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{1 \leq l \leq k} exp(e_{il})} \tag{2}$$

Then we take transformed features and calculate their weighted sum. Then we apply activation function:

$$\overrightarrow{d_i} = \tanh\Big(\sum_{1 \leq l \leq k, i \neq j} \alpha_{ij} W_V^T x_j \Big) \tag{3}$$

In Graph Attention Network, we apply mechanism of self attention to each node along with all of the first-order neighbours of that node. In Hyper-SAGNN, for any particular node v_i we aggregate information along its neighbours that are given in any particular tuple that is given. The organization of Hyper-SAGNN is demonstrated in 3

We can represent the input to our model in the form of tuple like $(\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_k})$. Every input set will first pass from the feed forward neural network that is position-wise and produces static embeddings $(\overrightarrow{s_1}, \overrightarrow{s_2}, ..., \overrightarrow{s_k})$ $where$ $s_i = tanh(W_s^T \overrightarrow{x_i})$. We called s_i as static embedding because embedding of any particular node i remains the same irrespective of the given set. The input set of nodes also then passes through the layer of multi-head attention to make vectors of dynamic embeddings $(\overrightarrow{d_1}, \overrightarrow{d_2}, ..., \overrightarrow{d_k})$. These vectors are called dynamic due to the reason that they depends on all other features of node present inside the input set therefore dynamic embedding of a node varies from tuple to tuple.

Then hadamard-power of the difference in dynamic and static pairs has been calculated. Next they are passed through a neural network with 1 layer and its activation function is sigmoid. It will give the scores of probability p_i. At the end, we take the average of all the probabilities and it will give us the final probability p

$$o_i = W_o^T ((\overrightarrow{d_i} - \overrightarrow{s_i}))^{o2} + b \tag{4}$$

$$p = \frac{1}{K} \sum_{i=1}^{k} p_i = \frac{1}{K} \sum_{i=1}^{k} \sigma(o_i) \tag{5}$$

This network, Hyper-SAGNN tries to develop the correlation between the probability that given vertices will make

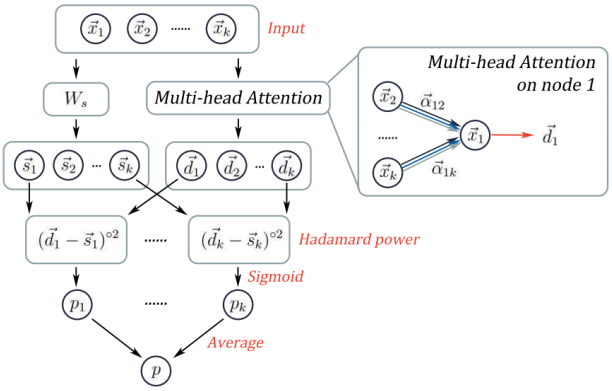

Figure 3. Neural Network structure used in Hyper-SAGNN. Image Source: Original Paper.

hyperedge or not and the avg distance that we calculated between static and dynamic embeddings . This distance shows inside each tuple how good the nodes neighbours features approximate static embedding of every node. Its design principal is similar to some extent to Continious Bag of word (CBOW) model [13]. The CBOW also takes conext into account and tries to predict target word.

3.2. Features generation approaches

To work with Hyper-SAGNN we need to generate x_i only through the graph. We have used 2 already available techniques to generate features x_i as shown in 4. First method is random walk based. For this a biased random walk scheme has been designed that is used to sample walks. Then features has been generated through skip-gram model that is trained for this purpose. The second method is encoder based. In encoder based method we used respective row of the adjacency matrix as features. These features are then passed through the structure that is like the auto-encoder. This will reduce its dimensionality along the output of hidden layer that is used as features.

4. Results

We will compare results of our model Hyper-SAGNN with DHNE model as it has showed very good performance over previous available methods like Deepwalk and HEBE

4.1. Performance Comparison with Existing Methods

We used the network reconstruction task to test the effectiveness of learned function and embedding vectors. We compare our model Hyper-SAGNN with both the random walk method and also the encoder-based method that has pretrained embeddings against the baseline model node2vec and a superior model DHNE. First of all we trained our model. Then in original network we used the learned embeddings in prediction of the hyperedge. We follow the

Methods	GPS		MOVIELENS		DRUG		WORDNET	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
node2vec-mean	0.572	0.188	0.557	0.197	0.668	0.246	0.613	0.215
node2vec-min	0.570	0.187	0.535	0.186	0.682	0.257	0.576	0.201
DHNE	0.959	0.836	0.974	0.878	0.952	0.873	0.989	0.953
Hyper-SAGNN-E	0.971	**0.877**	0.991	0.952	0.977	0.916	0.989	0.950
Hyper-SAGNN-W	**0.976**	0.857	**0.998**	**0.986**	**0.988**	**0.945**	**0.994**	**0.956**

Table 1. AUROC (Area Under the Receiver Operating Characteristic curve) and AUPR (Area under the Precision-Recall curve) values for the reconstruction of network. We represent the model as Hyper-SAGNN-W that has been trained with walk-based method and represented the model as Hyper-SAGNN-E that was trained by encoder-based method.

Methods	GPS		MOVIELENS		DRUG		WORDNET	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
node2vec-mean	0.563	0.191	0.562	0.197	0.670	0.246	0.608	0.213
node2vec-min	0.570	0.185	0.539	0.186	0.684	0.258	0.575	0.200
DHNE	0.910	0.668	0.877	0.668	0.925	0.859	0.816	0.459
Hyper-SAGNN-E	**0.952**	**0.798**	0.926	0.793	**0.961**	**0.895**	**0.890**	0.705
Hyper-SAGNN-W	0.922	0.722	**0.930**	**0.810**	0.955	0.892	0.880	**0.706**
Hyper-SAGNN-E (mix)	0.950	0.795	0.928	0.799	0.956	0.887	0.881	0.694
Hyper-SAGNN-W (mix)	0.920	0.720	0.929	0.811	0.950	0.889	0.884	0.684
Methods	GPS (2)		MOVIELENS (2)		DRUG (2)		WORDNET (2)	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
Hyper-SAGNN-E (mix)	0.921	0.899	0.971	0.967	0.981	0.973	0.891	0.897
Hyper-SAGNN-W (mix)	0.931	0.910	0.999	0.999	0.999	0.999	0.923	0.916

Table 2. Results of AUROC and AUPR showing performance of prediction for edge or hyperedge. We trained Hyper-SAGNN with both edges and hyperedeges and they are annotated as (mix) . Datasets that has "(2)" shows the performance of Hyper-SAGNN on pair-wise edge.

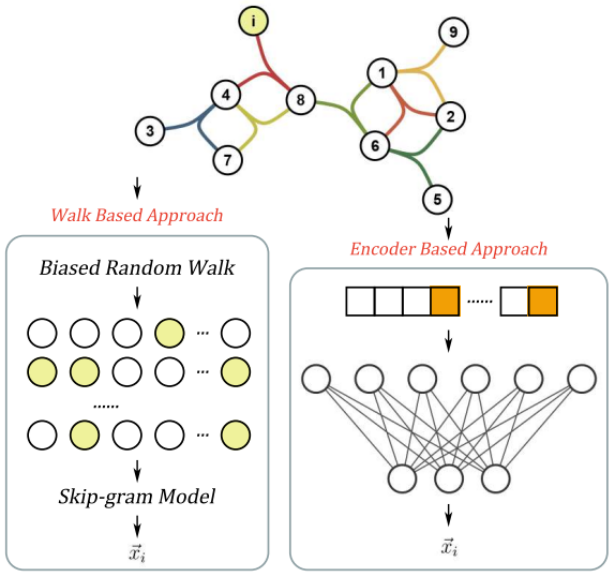

Figure 4. Illustration showing mechanism of generating node features. Image Source: Original Paper.

same setup as in DHNE. We tested the performance of all the models based on AUROC and AUPR. Table 1 shows the

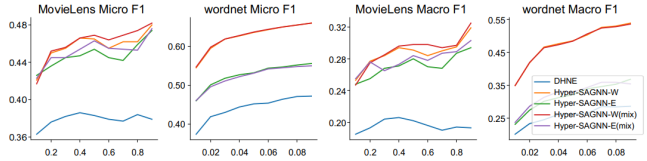

Figure 5. Results of wordnet and MovieLens datasets classification. Our model used random walk method for training is denoted by "W" and the model that used encoder based method during training is denoted by "E". Similarly the model that is used both edges as well as hyperedges for traininng are written as "(mix)". Source: Original Paper.

AUROC and AUPR values of different models. It can be shown from the table that our model Hyper-SAGNN shows the best results from all other previous models including DHNE with both encoder based approach and random walk based approach. But random walk method shows the best results from all other methods.

Next we tested Hyper-SAGNN's performance to predict the hyperedge. We divide the data set randomly in train and test datasets. The results are shown in Table 2

In addition to the hyperedge prediction task, we also

tested our model for node classification task. With MovieLens dataset we do the experiment of multi-label classification. And with wordnet dataset, we do experiment of multi-class classification. We used logistic regression classifier for these experiments. To evaluate their performance we used Micro-F1 and Macro-F1 as shown in Figure 5. It can be observed from the figure 5 that Hyper-SAGNN shows better performance from DHNE and random walk method used in Hyper-SAGNN model gives the best results.

4.2. Hyper-SAGNN results on Non-k-Uniform Hypergraph

Further we have tested our model with non-k-uniform heterogeneous hypergraph. We take our 4 datasets and divide every hyperedge into 3 pair-wise edges and then add all those pair-wise edges back into the original graphs. Then we use these datasets for link prediction tasks. We also used the same datasets for node classification using the same settings that was in prediction of link. Results of link prediction is shown in Table 2. The outcome of node classification experiment is shown in Figure 5

5. Conclusion

We developed a new GNN known as Hyper-SAGNN that learns hypergraph representation. Our model works with uniform and non-uniform hypergraphs as well as homogeneous and heterogeneous hypergraphs. Previous methods lack the ability to predict hyperedge on heterogeneous non-k-uniform hypergraphs but we have shown that our model showed very good performance on hyperedge prediction for non-k-uniform heterogeneous hypergraphs.

6. Important Note

I presented the research paper in the class as a term project for my machine learning course and also reproduced the results by running its code. I am not the author or co-author of this paper. The original paper and the authors can be found on this Link

References

[1] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Łukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 5998–6008, 2017. 3

[2] Rami Al-Rfou Bryan Perozzi 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, 2014. 2

[3] Tak-Shing Chan Chia-An Yu, Ching-Lun Tai and Yi-Hsuan Yang. Modeling multi-way relations with hypergraph embedding. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1707–1710, 2018. 2

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016. 2

[5] Fangbo Tao Meng Jiang Brandon Norick Huan Gui, Jialu Liu and Jiawei Han. Large-scale embedding learning in heterogeneous event data. In *IEEE 16th International Conference on Data Mining (ICDM)*, pages 907–912, 2016. 2

[6] Fei Wang Anil K Jain Inci M Baytas, Cao Xiao and Jiayu Zhou. Heterogeneous hyper-network embedding. In *IEEE International Conference on Data Mining (ICDM)*, pages 875–880, 2018. 2

[7] Fanghua Ye Jiajing Wu Zibin Zheng Jie Huang, Chuan Chen and Guohui Ling. Hyper2vec: Biased random walk for hyper-network embedding. In *International Conference on Database Systems for Advanced Applications*, pages 273–277, Springer, 2019. 2

[8] Xiao Wang Fei Wang Ke Tu, Peng Cui and Wenwu Zhu. Structural deep embedding for hypernetworks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[9] Xiao Wang Fei Wang Ke Tu, Peng Cui and Wenwu Zhu. Structural deep embedding for hypernetworks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[10] Prateek Yadav Anand Louis Naganand Yadati, Madhav Nimishakavi and Partha Talukdar. Hypergcn. Hypergraph convolutional networks for semi-supervised classification. In *arXiv preprint arXiv:1809.02589*, 2018. 2

[11] Arantxa Casanova Adriana Romero Pietro Lio Petar Velickovi ˇ c, Guillem Cucurull and Yoshua Bengio. Graph attention networks. In *arXiv preprint arXiv:1710.10903*, 2017. 3

[12] Feihu Zhang Song Bai and Philip HS Torr. Hypergraph convolution and hypergraph attention. In *arXiv preprint arXiv:1901.08150*, 2019. 2

[13] Kai Chen Greg S Corrado Tomas Mikolov, Ilya Sutskever and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), Advances in Neural Information Processing Systems*, page 3111–3119, Curran Associates, Inc., 2013. 3

[14] Zhitao Ying Will Hamilton and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017a. 2

[15] Zizhao Zhang Rongrong Ji Yifan Feng, Haoxuan You and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019. 2