Graph Structure Learning based on Mistakenly Predicted Edges from Reconstructed Graph Representation

Deogho Choi
Department of Semiconductor and Display Engineering
Sungkyunkwan University
Suwon, Republic of Korea
sailoer123@skku.edu

Jitae Shin*
Department of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
jtshin@skku.edu

***Abstract*—Recently, Graph Neural Networks (GNNs) have been focused because they are suitable for considering complex interactions. Such a real-world problem containing complex interactions can be represented by graph-structured data. However, this graph-structured data may contain task-irrelevant edges or missing edges due to human error. These noisy or incomplete edges can degrade the performance of trained GNNs. To tackle this problem, Graph Structure Learning (GSL), which focuses on finding a better graph representation of a given original graph based on the main learning objectives, has been studied. In this research, we propose a framework called Confusion-GSL that includes conventional graph neural network for node classification and edge selection for finding better graph representation. We focus on the fact that the probability matrix can be used to reconstruct graph representation like learned node embedding in Graph Auto Encoder (GAE). After training node classification, we calculate Gram matrix of the probability matrix and use this Gram matrix as an approximation of adjacency matrix. By applying Gumbel-Softmax to difference matrix between approximated adjacency matrix and original adjacency matrix, we select the connections likely to exist or not exist. After that we re-train GNN with this modified graph representation. We evaluate our proposed framework on two citation network datasets commonly used: Cora, Citeseer and Pubmed.**

Keywords—*graph neural network, graph auto encoder, graph structure learning, node classification*

I. INTRODUCTION

Most of the problems that arise in the real world can be analyzed in a more profound pattern when considering the relationship among data rather than judging them individually. Recently, Graph Neural Networks (GNNs) have been focused because they are suitable for considering complex interactions. The success of GNNs can be explained by message passing algorithm that enriches feature learning. However, if the obtained original graph includes task-irrelevant edges or missing edges due to human error, these small noises will be accumulated through propagation, and it will degrade the performance of GNNs. To tackle this problem, Graph Structure Learning (GSL), which focuses on finding a better representations of a given original graph based on the main learning objectives, has been studied. Most of previous papers about GSL try to find optimal graph structure with additional learning parameters. These additional parameters are usually used to learn the probability of being connected between each node pairs [1][2][3][4][5][6]. Unlike previous papers, this study made it possible to generate better graph representation without additional learning parameters. In this research, we propose Confusion-GSL inspired from Graph Auto Encoder (GAE) that is used for the link prediction task using Graph Convolutional Network (GCN) [7][8]. Similar to the method used in that paper, we focus on the fact that the probability matrix can be used to reconstruct graph representation like learned node embedding do in GAE. After training node classification task with cross entropy loss, we calculate Gram matrix of the probability matrix and used this Gram matrix as an approximation of adjacency matrix. And then we calculate difference between this approximation of adjacency matrix and original adjacency matrix. Positions that differ a lot in the matrix mean that the connection is likely to exist or not exist. Therefore, we applied Gumbel-Softmax to the difference matrix between approximated adjacency matrix and original adjacency matrix to select high probable edges for generating and deleting [9][10].

II. CONFUSION-GSL

A. Notations

We represent an input graph as $G = (V, A)$. The input graph G includes N nodes and all nodes contain L length of node feature. $V \in R^{N \times L}$ represents node feature matrix and $A \in R^{N \times N}$ represents adjacency matrix that is a binary matrix where $A(u, v) = 1$ if there is a connection between node u and node v. Furthermore, we use $Y \in R^{N \times C}$ which includes node label among C classes. Although we have label information about all the nodes, only the labels of training nodes are used during training for node classification task and labels of testing nodes are used for evaluation.

B. Framework

The framework of our proposed Confusion-GSL is described in Fig. 1. It consists of two main parts: conventional graph neural network for node classification (surrounded by blue dashed line) and edge selection for finding better graph representation (surrounded by red dashed line).

The first part for node classification task follows conventional training strategies. We employ GNN and feedforward input graph G which contains node feature matrix V and adjacency matrix A. It outputs the probability

978-1-6654-8559-3/22/$31.00 ©2022 IEEE

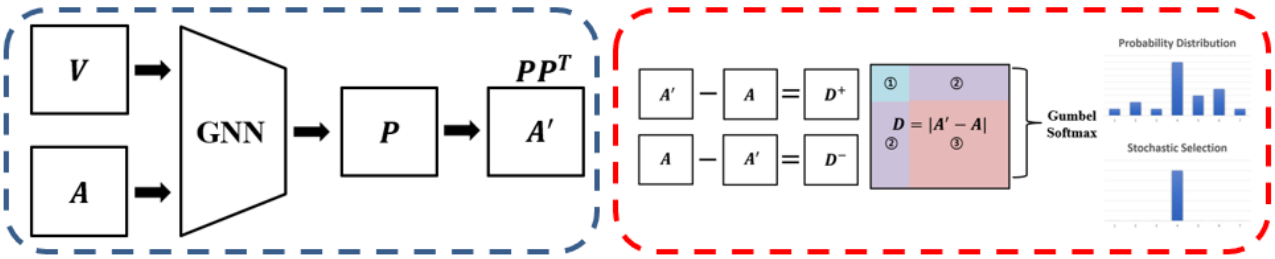

Fig. 1. Framework of Confusion-GSL

matrix P and we calculate cross entropy loss for predictions of training nodes. As GAE do for generating reconstructed adjacency matrix from node embedding, we calculate Gram matrix A' of the probability matrix P [7]. This Gram matrix A' works as approximation of adjacency matrix A because if two nodes assigned to same class, their inner product of probability distributions is close to 1. In the opposite case that two nodes assigned to another classes, their inner product is close to 0. Even though we don't use explicit reconstruction loss like GAE, but somewhat well-classified probability matrix is naturally sufficient for generating reconstructed adjacency matrix.

In the second part for edge selection, we calculate difference matrix D by subtracting approximated adjacency matrix A' and original adjacency matrix A. More specifically, positive difference matrix D^+ is acquired by subtracting original adjacency matrix A from approximated adjacency matrix A' and negative difference matrix D^- is acquired by subtracting approximated adjacency matrix A' from original adjacency matrix A. As described in Table 1, if there is a large difference between approximated adjacency matrix A' and original adjacency matrix A, that position can be interpreted that connection is likely to exist in case of positive difference matrix D^+ and not exist in case of negative difference matrix D^-. Therefore, we apply Gumbel-Softmax to positive difference matrix D^+ for selecting generated edges and to negative difference matrix D^- for selecting deleted edges. Furthermore, we can divide approximated adjacency matrix A' into three parts: training node & training node, training node & non-training node, and non-training node & non-training node. If we think about it from the perspective of learning, GNN knows labels of training nodes and these labels work as important role in correcting the training direction. But in case of non-training nodes, GNN doesn't know this information and they rely on propagation from trained training nodes. So, we only focus on training node & non-training node section for modifying graph representation because training node & training node section doesn't need for additional help from graph modification and non-training node & non-training node section is unstable due to the modified graph representation. We apply Gumbel-Softmax to difference matrix D to select generated edges and deleted edges. After that, we add these selected edges to original adjacency matrix A and retrain GNN using this modified graph structure for better performance.

TABLE I. CONFUSION MATRIX OF RECONSTRUCTION

A'	A	$\lvert A' - A \rvert$	*Operation*
$p_u p_v \sim 1$	1	Low	Maintain
$p_u p_v \sim 1$	0	High	Generate
$p_u p_v \sim 0$	1	Low	Maintain
$p_u p_v \sim 0$	0	High	Delete

III. EXPERIMENTS

A. Datasets

In this research, two citation network datasets were used, which are Cora, Citeseer and Pubmed dataset. The brief description about these datasets is shown in Table 2.

TABLE II. DESCRIPTION OF CITATION DATSETS

Dataset	*Cora*	*Citeseer*	*Pubmed*
Nodes	2,708	3,327	19,717
Edges	5,429	4,732	44,338
Features	1,433	3,703	500
Classes	7	6	3
Training Nodes	140	120	60
Validation Nodes	500	500	500
Testing Nodes	1,000	1,000	1,000

B. Evaluation

The proposed Confusion-GSL was implemented with Pytorch Geometric (PyG) [11]. We stacked two GCN layers and the number of units for hidden layer is 16. ReLU activation function was applied after graph convolution operation. Adam optimizer was used with learning rate 0.01 and L2 regularization was applied with weight decay hyperparameter $2 \cdot 10^{-3}$. We followed the transductive setting as same with other papers. Only the labels of training nodes were used for calculating classification loss and testing nodes were participated for calculating model accuracy. The performance of other methods for comparison was taken from the values reported in their papers. Table 3 shows the classification performance of Confusion-GSL and other state-of-the-art methods. The proposed method is shown to achieve the highest value on the Cora dataset. On the other hand, we didn't achieve the best performance on the Citeseer and Pubmed dataset, but we still achieved good performance similar to most other methods.

TABLE III. RESULT OF NODE CLASSIFICATION IN TERMS OF ACCURACY (%)

Model	***Cora***	***Citeseer***	***Pubmed***
GCN [8]	81.5	70.9	79.0
GLNN [1]	83.4	72.4	76.7
PTDNet [2]	82.8	72.7	79.8
GRCN [3]	84.2	73.6	79.0
LDS [4]	84.1	**75.0**	-
Neural Sparse [5]	83.7	74.1	-
IDGL [6]	84.5	74.1	**83.0**
Confusion-GSL (Proposed method)	**84.7**	72.5	80.3

IV. Conclusion

We propose novel framework called Confusion-GSL to find better graph representation. Unlike other methods, our proposed framework doesn't need additional learning parameters to learn the probability of being connected between each node pairs. We focus on the fact that the probability matrix can be used to reconstruct graph representation like learned node embedding in GAE. The Gram matrix of this probability matrix is used for calculating probability of being connected or disconnected. By applying the Gumbel-Softmax to this difference matrix, we can generate modified graph representation. The performance of GNN trained with this modified graph representation shows good performance on the Cora, Citeseer and Pubmed dataset.

Acknowledgment

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2022-2018-0-01798) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation)

References

[1] X. Gao, W. Hu, and Z. Guo, "Exploring Structure-Adaptive Graph Learning for Robust Semi-Supervised Classification", *IEEE International Conference on Multimedia and Expo*, IEEE, pp. 1-6, July 2020.

[2] D. Luo, et al, "Learning to Drop: Robust Graph Neural Network via Topological Denoising", *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ACM, pp. 779-787, March 2021.

[3] D. Yu, et al, "Graph-Revised Convolutional Network", *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 378-393, September 2020.

[4] L. Franceschi, et al, "Learning Discrete Structures for Graph Neural Networks", *International Conference on Machine Learning*, PMLR, pp. 1972-1982, May 2019.

[5] C. Zheng, et al, "Robust Graph Representation Learning via Neural Sparsification", *International Conference on Machine Learning*, PMLR, pp. 11458-11468, November 2020.

[6] Y. Chen, L. Wu, and M. Zaki, "Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings", *Advances in Neural Information Processing Systems 33*, NeurIPS, pp. 19314-19326, 2020.

[7] T. N. Kipf, and M. Welling, "Variational Graph Auto Encoders", *arXiv preprint*, arXiv, 1611.07308, 2016.

[8] T. N. Kipf, and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *arXiv preprint*, arXiv, 1609.02907, 2016.

[9] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax", *arXiv preprint*, arXiv, 1611.01144, 2016.

[10] C. J. Maddison, A. Mnih, and Y. W. Teh, "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables", *arXiv preprint*, arXiv, 1611.00712, 2016.

[11] M. Fey, and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric", *arXiv preprint*, arXiv, 1903.02428, 2019.