MESSAGE-PASSING SELECTION: TOWARDS INTERPRETABLE GNNS FOR GRAPH CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we strive to develop an interpretable GNNs' inference paradigm, termed *MSInterpreter*, which can serve as a plug-and-play scheme readily applicable to various GNNs' baselines. Unlike the most existing explanation methods, *MSInterpreter* provides a *M*essage-passing *S*election scheme (*MSScheme*) to select the critical paths for GNNs' message aggregations, which aims at reaching the self-explaination instead of post-hoc explanations. In detail, the elaborate *MSScheme* is designed to calculate weight factors of message aggregation paths by considering the vanilla structure and node embedding components, where the structure base aims at weight factors among node-induced substructures; on the other hand, the node embedding base focuses on weight factors via node embeddings obtained by one-layer GNN. Finally, we demonstrate the effectiveness of our approach on graph classification benchmarks.

1 INTRODUCTION

Recently, several advanced approaches (Ying et al., 2019; Luo et al., 2020; Schlichtkrull et al., 2021; Huang et al., 2022; Vu & Thai, 2020; Gui et al., 2022; Yuan et al., 2021; Schnake et al., 2021; Yuan et al., 2020; Yu & Gao, 2022) have been proposed to explain the predictions of graph neural networks (GNNs), and are divided into two categories (Yuan et al., 2022), i.e., instance- and model-level explanation. The instance-level aims at critical nodes or subgraphs; on the other hand, model-level focuses on a more high-level explainability. However, most of existing GNNs' explainers are post-hoc and lack the study regarding message-passing selection for GNNs' inference.

In this paper, we aim at selecting the critical message-passing paths for GNNs' aggregations, and thus develop an interpretable [1] GNNs' inference paradigm, i.e., *MSInterpreter*, for the task of graph classification. As shown in Figure 1, the main process is to build the *M*essage-passing *S*election scheme (*MSScheme*), and then plug it at the beginning of the existing GNNs' baselines to reach self-explainable inference. The contributions and details are summarized as follows:

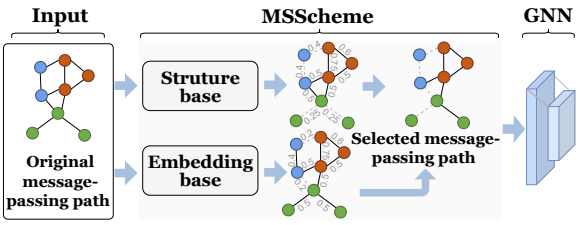

Figure 1: An illustration of the proposed *MSInterpreter* framework, i.e., plugging the *MSScheme* at the beginning of GNNs for an interpretable inference.

- We propose the *MSScheme* to calculate the weight factors among the connected nodes and then select the critical message-passing path for GNNs' aggregations.
- We plug the *MSScheme* at the beginning of an arbitrary GNN to build *MSInterpreter*, which is an end-to-end learnable framework to reach the self-explainable GNNs.
- We apply the *MSInterpreter* into graph classification task and provide the experimental analysis to support the claim that it can achieve significantly improved explanations.

[1] An interpretable model means that it can provide humanly understandable interpretations of its predictions, i.e., a model is no longer a black-box to some extent (Yuan et al., 2022).

2 METHOD

Notation. A graph consisting of n nodes can be represented as $G = (\boldsymbol{A}, \boldsymbol{X})$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ represent adjacency matrix and node feature matrix respectively. For a set of labeled graphs $\mathbb{S} = \{(G_1, y_1), .., (G_N, y_N)\}$, where $y_i \in \mathbb{Y}$ denotes the label of i-th graph $G_i \in \mathbb{G}$. The graph classification task is to learn a function $f : \mathbb{G} \rightarrow \mathbb{Y}$ that maps graphs to the labels' set. The message-passing path is initialized as $\boldsymbol{M} = \boldsymbol{A}$. $\mathcal{N}(v)$ denotes the neighbor set of vertex v. To efficiently obtain the critical message aggregation path for GNNs' inference with application to graph classification, we introduce the elaborate *MSScheme* as following.

MSScheme. The *MSScheme* aims at the critical message-passing paths for GNNs' inference, which is the essential module to reach the self-explaination. The *MSScheme* is formulated as:

$$\boldsymbol{M} = \mathcal{M}(\boldsymbol{A}, \texttt{Mask}) \in \mathbb{R}^{n\times n} \tag{1}$$

where $\mathcal{M}$ is the message-passing selection function using mask matrix $\texttt{Mask} \in \mathbb{R}^{n\times n}$. To obtain mask matrix, we firstly compute the edge weight factors. Then, we sort these weights, and set the high-weight edges as true, and vice versa. Specifically, the edge weights are obtained by considering two components: the vanilla structure base and the node embedding base. For the vanilla structure base, we compute the edge weight between nodes v_i and v_j using the intersection over union (IoU) of the number of $\mathcal{N}(v_i)$ and $\mathcal{N}(v_j)$, i.e., $\texttt{W}_{str}(v_i, v_j) = \psi(\frac{num(\mathcal{N}(v_i)\bigcap\mathcal{N}(v_j))}{num(\mathcal{N}(v_i)\bigcup\mathcal{N}(v_j))})$, where ψ is a norm operation in a whole graph. On the other hand, edge weight regarding node embedding is computed by $\texttt{W}_{emb}(v_i, v_j) = \varphi(h_{v_i}, h_{v_j})$, where h_{v_i} and h_{v_j} denote the node embeddings of v_i and v_j via one-layer GNN. φ is a mapping function like line or gaussian kernel function, etc. The resulting edge factor is:

$$\texttt{W}_{com}(v_i, v_j) = \texttt{W}_{str}(v_i, v_j) + \alpha \cdot \texttt{W}_{emb}(v_i, v_j), \tag{2}$$

where α is a hyperparameter to balance two weight factors.

MSInterpreter. To build the self-explainable GNNs, we plug the proposed *MSScheme* at the beginning of the GNNs predictions. As shown in the Figure 1, we use the *MSScheme* to select the critical message aggregation path to build the interpretable GNNs' inference, i.e., *MSInterpreter*, which is beneficial to build an end-to-end framework while training GNNs for the task of graph classification.

3 EXPERIMENT

We compare *MSInterpreter* with four popular explainable methods PGExplainer (Luo et al., 2020), GNNExplainer (Ying et al., 2019) ,SubgraphX (Yuan et al., 2021), and GStarX (Zhang et al., 2022). In Table 1, we list these methods' accuracy, recall, and F1 score, and our method achieves the competitive performances. More experimental details are provided in Appendix A.

Table 1: Evaluation of several explainable methods on two graph classification datasets (with annotated explanatory edges) using a three-layer GIN architecture.

Dataset	BA-2MOTIFS			$MUTAG_0$		
	Acc.	Rec.	F_1	Acc.	Rec.	F_1
GNN-Exp.	49.21 $_{\pm 0.39}$	52.65 $_{\pm 1.91}$	30.85 $_{\pm 0.96}$	50.48 $_{\pm 1.82}$	59.22 $_{\pm 0.70}$	48.00 $_{\pm 0.80}$
PGE-Exp.	36.05 $_{\pm 0.11}$	80.23 $_{\pm 0.57}$	35.13 $_{\pm 0.21}$	56.27 $_{\pm 0.59}$	83.60 $_{\pm 0.89}$	59.44 $_{\pm 0.56}$
SubgraphX	**71.15** $_{\pm 1.00}$	52.08 $_{\pm 0.82}$	40.42 $_{\pm 0.18}$	69.05 $_{\pm 0.91}$	38.35 $_{\pm 2.90}$	48.62 $_{\pm 2.68}$
GStarX	61.85 $_{\pm 0.01}$	86.86 $_{\pm 1.67}$	49.56 $_{\pm 0.45}$	46.96 $_{\pm 0.01}$	74.62 $_{\pm 1.85}$	58.88 $_{\pm 0.24}$
MSInterpreter	66.34 $_{\pm 0.37}$	**90.78** $_{\pm 0.87}$	**53.75** $_{\pm 0.53}$	**79.31** $_{\pm 0.95}$	**88.69** $_{\pm 1.63}$	**76.12** $_{\pm 1.04}$

4 DISCUSSION & CONCLUSIONS

In this paper, we introduce an novel framework to explain the GNNs' inference with application to graph classification. Firstly, we build a scheme *MSScheme* to analyze the weight factors of message-passing paths, which is essential to obtain the crucial message aggregations for GNNs' inference. Then, we plug the *MSScheme* at the begin of the GNNs' predictions to build an end-to-end interpretable paradigm, i.e., *MSInterpreter*, which aims at reaching the self-explaination GNNs for graph classification. Finally, we perform our proposed method in graph classification to demonstrate its superior performance.

REFERENCES

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Shurui Gui, Hao Yuan, Jie Wang, Qicheng Lao, Kang Li, and Shuiwang Ji. Flowx: Towards explainable graph neural networks via message flows. *arXiv preprint arXiv:2206.12987*, 2022.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, et al. Dig: A turnkey library for diving into graph deep learning research. *The Journal of Machine Learning Research*, 22(1):10873–10881, 2021.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 2020.

Kaspar Riesen, Horst Bunke, et al. Iam graph database repository for graph based pattern recognition and machine learning. In *SSPR/SPR*, volume 5342, pp. 287–297, 2008.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. In *International Conference on Learning Representations*, 2021.

Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021.

Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022*, pp. 1018–1027, 2022.

Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 2020.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 2019.

Zhaoning Yu and Hongyang Gao. Motifexplainer: a motif-based graph neural network explainer. *arXiv preprint arXiv:2202.00519*, 2022.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 430–438, 2020.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pp. 12241–12252, 2021.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. In *Advances in Neural Information Processing Systems*, 2022.

A APPENDIX

Dataset. We compare two datasets with edge interpretation labels: MUTAG_0 and BA2Motifs. The statistics for the BA2Motif (Luo et al., 2020) and MUTAG_0 (Tan et al., 2022) datasets are shown in Table 2.

- MUTAG_0 is a molecular dataset for graph classification tasks and consists of 4,337 molecular graphs. In this dataset, nodes represent different atoms, and edges represent chemical bonds. Each graph is assigned to one of the two categories according to its mutagenic effects (Riesen et al., 2008). We observe that carbon rings are present in both mutagenic and non-mutagenic graphs, but the carbon rings with the chemical groups NH_2 or NO_2 are mutagenic. Therefore, the carbon ring can be considered a shared base map, and the two groups NH_2 and NO_2 are the base sequence of the mutagenic map. For the non-mutagenic graphs, there is no explicit base sequence. For the mutagenic graphs, there are true edge masks which marks the edges of the mutagenic motifs.
- BA2Motifs is a synthetic graph motif detection dataset. BA2Motifs contains Barabasi-Albert (BA) base graphs of size 20 and each graph has five node patterns. The node features are 10-dimensional all-one vectors. Patterns can be house-like structures or cycles. The graphs are divided into two categories based on the topics they contain. Therefore the interpretation of this dataset is the edges that make up the two patterns.

Experiments Settings. We use three-layer GIN as the backbone network and set the hidden dimensions as 128. We divide the dataset into three random groups (80%/10%/10%) as training, validation, and test sets.

For the BA2Motifs dataset, we compute the four metrics only for correctly predicted data. For the MUTAG_0 dataset, we add one more restriction that the prediction category is mutagenic since only mutagenic data have interpreted edges. We uniformly set the batch size to 64 and the number of epochs to 100. For our method, we keep the sparsity at 0.5 and the hyperparameter α is set to 0.5.

Our implementation is based on Python 3.8.15, PyTorch 1.12.0, PyTorchGeometric 2.2.0 (Fey & Lenssen, 2019), and DIG (Liu et al., 2021). We adopt the GNN implementation and the provided implementation of the baseline explainer from the DIG library.

Graph Classification. We provide the accuracy about graph classification in table 3.Compared to the three post-hoc explainers, our method keeps the accuracy no less than the accuracy of backbone.

Table 2: Dataset Statistics.

Dataset	**Graphs**	**Edges(avg.)**	**Nodes(avg.)**	**Classes**
MUTAG_0	2301	32.54	31.74	2
BA2Motifs	1000	25.00	51.39	2

Table 3: Accuracy of graph classification for various interpretation methods

	GIN	GNN-Exp.	PGE-Exp.	SubgraphX	GSTtarX	MSInterpreter
MUTAG_0	0.995	0.826	0.995	0.904	0.934	1.0
BA2Motifs	1.0	0.993	1.0	0.931	0.964	1.0