Explanations for Graph Neural Networks via Layer Analysis

Qinfeng Li[1], Xinrui Kang[1], Wenyuan Li[1], and Dong Liang[2](✉)

[1] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China

[2] Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China

liangdong@bupt.edu.cn

Abstract. Like many deep learning models, graph neural networks (GNNs) are regarded as black boxes and lack interpretability. Therefore, it is difficult for GNNs to be fully trusted by humans to be applied to various life scenarios. Based on this problem, we propose a new interpretability method called LAExplainer, which is used to explain GNNs hierarchically at the model level. In particular, LAExplainer not only focuses on the overall interpretation of the model, but also analyzes the interpretation problems between layers. Our approach interprets the middle-level process of the model through layer-by-layer analysis, and uses it as a basis to guide the construction of sub-graphs to reduce the size of the sub-graph set, which effectively explain the overall model. In addition, the approach will analyze the importance of model features and produce an adjustable principal component selection mechanism. In terms of evaluation indicators, we propose to set hyperparameters so that the two results of Fidelity and Sparsity can be changed simultaneously by adjusting the hyperparameters during the interpretation of GNNs. Experimental results show that our proposed method is effective in synthetic data sets and real data sets, and the results of the visualized sub-graphs are more in line with human understanding.

Keywords: Graph Neural Networks · Explanations · Model-level · Generation

1 Introduction

Many machine-learning tasks require the analysis of non-Euclidean data, such as social networks, knowledge maps, recommendation systems, and data from the life sciences field [6, 9, 20, 44]. These data can be represented by graphs using node characteristics and node connections (edges) [3, 19, 34]. The outstanding ability of graph neural networks (GNNs) to model the dependency relationship between graph nodes has made a breakthrough in the research field related to graph analysis.

A GNN is an organic combination of connectionism and symbolism [1, 2]. It enables the application of the deep learning model to the non-Euclidean structure of a graph and endows the model with a certain causal reasoning abilities [11, 21].

Because of their recursive-messaging scheme, the GNNs can achieve the most advanced performance on various tasks [3, 37]. In this scheme, the information is encoded at the node and passed along the edge of the graph. Similar to the traditional deep learning framework [45], GNNs exhibit a complex function that is quite opaque to humans [10, 15].

It is necessary to explain the black box model [2, 22, 25]: the deep model cannot be fully trusted without understanding the underlying mechanism behind the prediction, which hinders the use of the deep model in applications related to fairness, privacy, and security [5]. To deploy the depth model safely and reliably, it is necessary to provide accurate prediction and human-understandable interpretation, especially for users in interdisciplinary fields [8, 16].

Compared with the fields of image and text [8, 24, 29], there is less research on the interpretability of graph models. However, this is the key to understanding deep GNNs. In recent years, several methods have been proposed to interpret GNN prediction [27, 41], such as XGNN [42], PGExplainer [23], and GraphSVX [4, 7]. These methods [12] provide different interpretations from different angles. However, there is still a lack of standard datasets and metrics to evaluate the interpretation results. Therefore, the existing methods of interpreting GNN show significant differences in the interpretation effects of different GNN models or datasets.

Using XGNN as an example, this method proposes explaining, this method proposes to explain the GNN through graph generation. Instead of directly optimising the input graph, it trains a graph generator to maximise the target graph prediction. The generated graph is then considered the interpretation of target prediction and is expected to include discriminant graph patterns. In XGNN, the expression of the graph generation is a reinforcement-learning problem. For each step, the generator predicts the addition of an edge to the current graph. Then, the generated graph is input into the trained GNN, and feedback is obtained through the policy gradient to train the generator. In addition, some graph rules are added to encourage interpretation to be both effective and understandable.

However, the set of partitioned subgraphs increases exponentially with the number of nodes and features [26, 28]. When the depth of the GNN model is deep or the graph structure is complex, it is difficult to select a limited set of subgraphs and evaluate them. Most existing methods explore the effects of changing the input on the model prediction and interpreting the model based on it, which cannot explain the GNN clearly. These methods cannot focus on the content and path of information transmission in each step of the information transmission of GNN.

Given these limitations, we propose a systematic interpretation framework that encapsulates the recently introduced GNN interpretations inspired by some methods of CNN interpretation. It provides different and common views on the functions of several existing works, which will make our interpretation framework more complete and inspire future work. In this study, we use this framework to define the LAExplainer interpreter. The LAExplainer interprets the middle layer process of the model through a layer-by-layer analysis. After obtaining the interpretation results, it will be used as a basis to guide the construction of subgraphs, reduce the size of the subgraph set, and effectively explain the whole model. Finally, we evaluated the LAExplainer's node and graph classification tasks on real-world and synthetic datasets. We show that it is superior to the existing baseline in interpretation accuracy and further verify its desirable aspects, such as robustness or certainty.

2 Related Work

2.1 Graph Neural Networks

Nodes and edges of connected nodes make up graphs. Popular in-depth learning-based GNNs are networks that process graph-type data. The goal of these networks is to learn the representation of each node. The representation of each node is calculated from the characteristics of the node, the edges connected to the node, and the representation and characteristics of its neighbour nodes [42]. Tasks that focus on nodes can be accomplished by using direct representations, while tasks that focus on the entire graph can obtain a global representation by pooling all the representations of nodes or other methods and then performing the corresponding tasks. GNNs are mainly divided into graph convolution networks (GCNs) [7, 17], attention-based update graph networks (GATs) [5, 18, 33, 35], gated update graph networks, and graph networks with jump edges [36, 40]. The GCN is currently the most important graph network [31]. It is a natural generalisation of vehicles in a graph structure. A convolution neural network is a method that uses a local perception area [43], shared weights, and downsampling in the spatial domain. It has stable and invariant characteristics relative to displacement, scaling, and distortion, and can extract the spatial features of images well.

2.2 Instance-Level Explanations

The design of the instance-level methods [30] is similar to feature engineering in that they find some of the features in the input data that most affect the prediction results and provide input-dependent explanations for each input graph. Given an input graph, the instance-level approach design explores the important features that affect model predictions to interpret depth models. The instance-level method is divided into four branches based on how they obtain signature importance scores:

Gradients [32] or eigenvalues usage represents the importance of different input features based on the gradient/feature method.

The perturbation-based method monitors the change in the predicted values under different input perturbations to learn the importance score of the input characteristics.

Based on the decomposition method, the prediction score, such as the prediction probability, is first decomposed into the neurons of the last hidden layer. The scores are then propagated back one by one to the input space, and the decomposition score is considered as the importance score.

The proxy-based method first extracts a sample of a dataset from the neighbours of a given example. Next, a simple and interpretable model, such as a decision tree [14], is fitted to the sampled dataset. The original forecast is explained by interpreting the proxy model.

2.3 Model-Level Explanation

Model-level methods directly interpret the model of a GNN without considering any specific input instances. This input-independent interpretation is high-level and can account for general behaviour. This direction is less explored than the instance-level approach. In recent years, the main methods that have been used include XGNN, PGExplainer, and GraphSVX. They are primarily graph-based, maximising the prediction probability of a class by generating graph patterns and interpreting this class using the graph patterns [13, 38].

XGNN trains a graph generator so that the generated graph patterns maximize a certain prediction of the model. Then formulate the graph generation as a reinforcement learning task, where for each step, the graph generator predicts how to add an edge into the current graph. The graph generator is trained via a policy gradient method based on information from the trained GNNs.

PGExplainer adopts a deep neural network to parameterize the generation process of explanations, which enables PGExplainer a natural approach to explaining multiple instances collectively.

GraphSVX is a decomposition technique that captures the "fair" contribution of each feature and node towards the explained prediction by constructing a surrogate model on a perturbed dataset. It extends to graphs and ultimately provides as explanation the Shapley Values from game theory.

Overall, these two methods explain the depth map model from different perspectives. Instance-level methods provide case-specific explanations, while model-level methods provide high-level insights and a general understanding of how depth map models work.

2.4 Desirable Properties of Explanations

Often overlooked when designing interpreters are the ideal attributes of interpretation that have become the focus of social sciences and the machine learning community. From a theoretical point of view, a good explanation is accurate (true) and reflects the proportional importance (meaningful) of a feature to the prediction. They are also stable and consistent (robust), which means that when changed to a similar model or similar instance, the variance is small. In addition, they reflect the certainty (decomposability) of the model and represent its (global) functionality as much as possible [39].

3 Method

3.1 Preliminary Concepts and Background

Graph Rules

When generating graphs to interpret GNNs, we often produce meaningless graphs or graphs that are not people-centric and inconvenient to understand. Therefore, we defined some easy-to-understand graph rules. First, only one edge is allowed between any two nodes. Second, the resulting graph cannot contain more nodes than a predefined maximum number of nodes. In addition, we combined dataset-specific rules to guide graphics generation. For example, in a chemical dataset, each node represents an atom; hence its degree cannot exceed the valency of the corresponding atom. At the same time, to be more explanatory, we need to minimise the number of subgraphs independent of other nodes in the generated graph such that the resulting graph is not too fragmented. It is important to generate as few edges as possible to find interpretable parts in GNNs.

Shapley Value

The Shapley value equitably distributes the benefits of cooperation by considering the contributions made by each participant. It describes the fair distribution of the total game revenue based on each participant's contribution, assuming they all cooperate. We obtain it by calculating the average marginal contribution of each participant to join any possible alliance of participants. This approach is based on the game theory. The complete set of participants is defined as $N = \{x_1, x_2, ..., x_n\}$ with n elements x_i And any number of participants can form a subset $S \subseteq N$. .. V (S) represents the value generated by the cooperation of the elements in the S subset. The value of the final allocation is expressed as φ_i(N, v).

When allocating the benefits, we must meet the following principles. Effectiveness, that is, all values are distributed $\sum_{i \in N} \varphi_i(N, v) = v(N)$. Symmetry, that is, if x$_i$ and x$_j$ are equivalent in status (and can be interchanged with each other), the benefits should be the same. Income from not being a contributor is 0. Additivity, that is, if the same person performs two tasks, then the benefits of the two tasks should be divided together as if their division was separate.

Previous studies have proved that the Shapley value is the only solution that satisfies the four conditions of appeal, and the formula is as follows:

$$\varphi_i(N, v) = \frac{1}{N!} \sum_{S \subseteq N \setminus \{i\}} |S|!(|N| - |S| - 1)![v(S \cup \{i\}) - v(S)] \tag{1}$$

Unified Framework for GNN

By identifying the relationship between graphic patterns and GNN predictions, we can better understand the model and verify that it works as expected. As mentioned previously, the existing GNN interpreters focus only on the impact of changes to the input on the predicted results, trying to explain the GNN black box in a single step. This leads to two problems: first, it is difficult to effectively explain the black box in the subgraph segmentation of the step; second, when the graph structure is too complex, the subgraph segmentation will be blind if the subgraph set is too large.

Therefore, we constructed a new GNN interpreter framework to solve these two problems. Intuitively, given a trained GNN model, its model-level interpretation should explain what graph patterns or subgraph patterns lead to certain predictions, while at each layer of GNN, the model focuses more on the graph patterns. A meaningful graph pattern may consist of several fixed modules. Different motifs can be found in graphs with different functions, which means that different motifs may be directly related to the graph function. For example, the function of a DNA strand is expressed through the sequence of bases to which it is anchored. In this framework, we refined each part of the algorithm to form the final interpreter, LAExplainer.

Formally, f(·) represents the trained GNN classification model, and $C = \{c_1, c_2, ...c_m\}$ represents the classification set, $l_n(\cdot)$ represents the output of the nth layer. In the layer-by-layer interpretation of the GNN model, our goal was to partition the subgraph that can maximise the output of this layer. It can be expressed as

$$g_n^* = \underset{g}{\operatorname{argmax}}\, l_n(g) \tag{2}$$

The g_n^* table is the target subgraph when we explain the nth layer of the GNN network. We adopt the gradient calculation method to find this subgraph, which will be explained later.

When a graph is identified by a given GNN (f(·)), the model provides a classification $c_i \in C$. In the overall interpretation of the GNN model, our goal was to divide the GNN model prediction into the maximum probability prediction category of the subgraph. This can be expressed as

$$G^* = \underset{G}{\operatorname{argmax}}\, P(f(G) = c_i) \tag{3}$$

where G^* represents the divided target subgraph. Popular methods to find this subgraph include perturbation-based methods and graph generator methods based on reinforcement learning. Their disadvantages have been described above. We adopted the approach of aggregating the small-scale subgraphs obtained through layer-by-layer analysis to generate the explanatory subgraphs of the GNN model. Before the aggregation, we analysed the Shapley value of the subgraph. We set the threshold to select the part that had the most significant influence on the prediction result, which was represented by SHP (·). At the same time, the part of the subgraph that conformed to the graph structure was selected through the graph rule judgment, represented by GraphR(·). This can be expressed as

$$G^* \approx G_{exp} = GraphR\left(\sum_{n=0}^{N} g_n \cdot Shp(g_n)\right) \tag{4}$$

This is an approximate calculation. In this process, we effectively reduced the algorithm complexity such that it can explain the deep GNN model and improve the robustness of the algorithm. We calculated the degree of similarity later to prove the feasibility of this approximation.

An overview of the proposed approach presentation is shown in Fig. 1. The GNN model was obtained by training a batch of graph datasets. These graphs, classified as the third category through human observation, contained the same characteristics: rectangular boxes composed of four nodes, red, yellow, green, and blue, which is the basis for classification. By maximizing the output layer by layer in the frame, we obtained the subgraph of each layer of the GNN, such as an edge of the rectangle. Then, by calculating the Shapley values of these subgraphs, we evaluated their importance and provided a basis for selecting aggregation. Finally, the aggregate subgraph was modified and verified by graph rules to obtain the total subgraph G, which maximised the prediction probability of the GNN model. Finally, the interpretation of GNN model was completed.

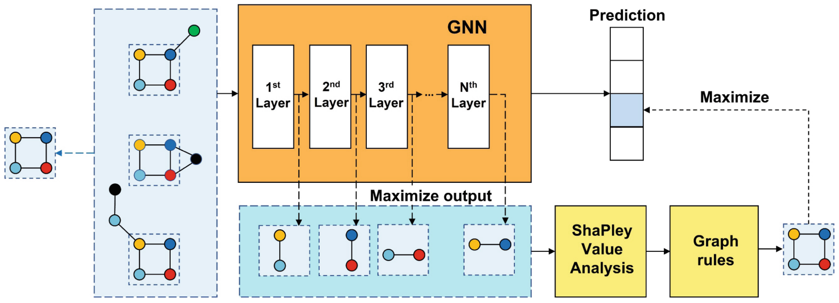


Fig. 1. Proposed GNN model interpretable framework through interlayer analysis.

3.2 Detailed Method

Interpretation of GNN at Each Level

In this section, we explain a separate layer of GNNs. Suppose the graph calling GNN has N nodes, and each node has F features. $l_n(\cdot)$ was used to represent the nth layer of the trained GNN. The inputs of this layer were the feature matrix $h_n \in \mathbb{R}^{N\times F}$ and the adjacency matrix $A \in \mathbb{R}^{N\times N}$ of the graph and the output was the feature matrix $h_{n+1} \in \mathbb{R}^{N\times F}$ of the next layer. This can be expressed as

$$h_{n+1} = l_n(h_n, A) \tag{5}$$

Our goal in this section is to find a suitable subgraph mask $M_{An} \in \mathbb{R}^{N\times N}$ to divide the input graph to explain this layer, as shown in Fig. 2. We need to input the subgraph as much as possible to the nth layer to obtain a larger output characteristic matrix h_{n+1}, and at the same time, we need to divide the subgraph as small as possible in structure, so the optimization objective can be expressed as follows:

$$M_{An} = \underset{M_{An}}{\operatorname{argmax}}(\lambda_1 \| l_n(h_n, A \odot M_{An}) \| - \lambda_2 \| A \odot M_{An} \|) \tag{6}$$

where $\odot$ represents the multiplication of two matrices by bits, and the element in the subgraph mask M_{An} is only 0 or 1. If it is 1, it indicates that the adjacency relationship is selected; otherwise, it is abandoned. Each row in the feature matrix h_n represents the feature vector of a node, and the adjacency matrix A is symmetric and represents the connection between two nodes. In the optimisation goal, it is evident that λ_1 and λ_2 are two hyperparameters. When the proportion of λ_1 is relatively high, a larger subgraph will be output as the interpretation of this layer of the model; otherwise, a smaller subgraph will be output as the interpretation of the model. In practice, because the depth of the GNN model is generally much larger than the number of nodes in the graph, we often use smaller subgraphs in the inter-layer interpretation. The final output subgraph is represented by $g_n = A \odot M_{An}$.

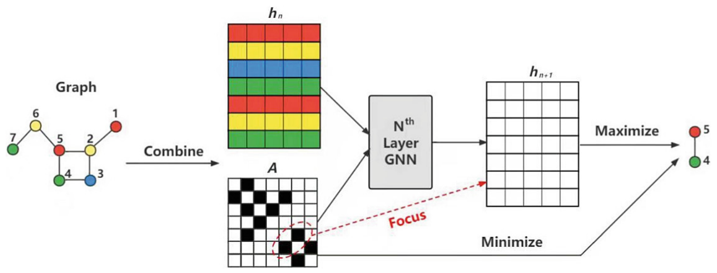


Fig. 2. Schematic diagram of GNNs interlayer interpretation subgraph by maximum output.

Explanation of the Overall GNN

In this section, we explain the GNN as a whole. During this period, we obtained many small subgraphs to explain the GNN at each layer, and our goal is to rationally aggregate these graphs to form a subgraph to explain the GNN model. In the overall framework, we mentioned that to screen the subgraph and splice the selected subgraph into a complete subgraph. Assume that the GNN has an M layer and use $G = \{g_1, g_2, g_3, \ldots ..g_M\}$ to represent the set of subgraphs generated in the previous step. We first calculate the Shapley value of each subgraph to measure the importance of each subgraph. To extend the Shapley value to the graph, the gain function is represented by GNN prediction probability, which is $v(g_n) = P(f(g_n) = c_i)$. Therefore, for the $i \in \{1, 2, 3 \ldots, M\}$ subgraph, its Shapley value is:

$$\varphi_i(G, P) = \frac{1}{M!} \sum_{S \subseteq G\{i\}} |S|!(|M| - |S| - 1)!\big[P(f(S \cup i) = c_i) - P(f(S) = c_i)\big] \tag{7}$$

From this, we can filter the subgraph for $Shp(g_i) = 1$ if $\varphi_i(G, P) > \varepsilon$, or $Shp(g_i) = 0$. There is a threshold ε where the selection of a subgraph is if its Shapley value is greater than this threshold. This is a hyper-parameter, and the subgraph used to explain GNNs increases when ε is smaller and smaller when ε is larger. After screening, to form a complete subgraph, the aggregation of the selected subgraph denoted is G'_{exp}.

Subsequently, to explain the overall GNNs, we generated subgraphs. However, this subgraph only seeks to maximise the prediction probability as much as possible without considering the meaning of generating subgraphs. In many cases, we sacrifice a certain degree of accuracy for the convenience of people to generate a human-centred subgraph as the output of the interpreter. Therefore, the generated sub-graph verification and modification is through graph rules, which expression is $G_{exp} = GraphR\big(G'_{exp}\big)$. Previously described are the specific graph rules.

Interpretation of Graph Node Features

We divided the graph structure and interpretation and the node feature interpretation into two independent parts and then summarised the results into GNNs interpretation. Similar to the screening of subgraphs, we can also sort and filter the contributions of node features using Shapley values. Its use is to represent the set of features, so we can also calculate the Shapley value of each feature and sort it. In this interpretation, we can choose some more important features.

4 Evaluation of Experiment

4.1 Datasets and Experimental Settings

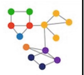
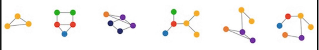
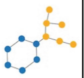
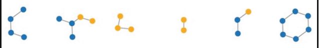
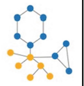
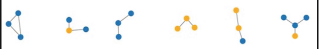
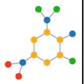
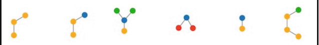
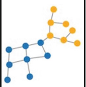
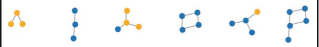
We conducted several experiments on different datasets and GNN models to prove the validity of our proposed method. We evaluated our LAExplainer with five datasets for graph classification and node classification tasks, including synthetic and biochemical data, The results are shown in the Table 1. We summarise these datasets as follows:

BA-community: This is a node classification dataset with eight different labels. By combining two Ba-shaped graphs with randomly added edges, we obtained for each graph. The membership of the Ba-shape diagram and its structural location determine the node label.
Tree-cycle: This is a node classification dataset with two different labels. Each graph consists of a base-equilibrium tree graph with a depth equal to eight and a 6-node periodic motif. The connection between these two parts was random. The labelling of nodes in the base diagram is 0; otherwise, 1.
BA-2motifs: This is a graph classification dataset with two different graph labels. There are 800 graphs, each obtained by attaching different motifs, such as house-like and five-node cycle motifs, to the base BA graph. The labelling of the different figures is according to the type of motif.
BBBP: This is a molecular dataset for the graph classification task. In these datasets, each graph represents a molecule, whereas nodes are atoms and edges are bonds. The chemical function of the molecule determines the label.
Tree-grids: This is a node classification dataset with two different tags. It is the same as the tree-cycle dataset, except that the tree-grids dataset uses 9-node grid motifs instead of periodic motifs.

In this experiment, we used an RTX 3090 graphics card with 24 G video memory, an Intel Xeon E5-2678 V3 24-core 32 G CPU, 2 TB hard disk, and the running environment was PyTorch - CUDa10-CUDNn7 1.4.0 and Python 3.6.

In this study, the LAExplainer method use is to explain the GNN model both as a whole and between layers. Therefore, in addition to the subgraphs explaining the model, the algorithm output includes many smaller subgraphs. As shown in Table 1, because the number of layers of each GNN model is different, there was a selection of six representative interlayer subgraphs for each data as illustrations. We can also intuitively observe that obtaining subgraphs of the overall interpretation of the GNN model is through overlapping and stitching these subgraphs. In terms of accuracy, LAExplainer can correctly identify basic graphic structures and outperform leading baselines in all but one task, in addition to providing higher theoretical assurance and humane interpretation. On the BA-Community dataset, owing to the relatively complex node types and structures involved in the graph features, accuracy improvement was not substantial.

Table 1. Visualization and precision comparison of experimental results.

	Visualization		**Accuracy**			
	Explainer of GNN	**Explainer of Layer**	**LAExplainer**	**GNNExplainer**	**XGNN**	**PGExplainer**
BA-Communit			*0.89*	*0.74*	***0.90***	*0.79*
Tree-Cycle			***0.97***	*0.85*	*0.93*	*0.95*
BA-2Motif			***0.95***	*0.67*	*0.90*	*0.83*
BBBP			***0.78***	*0.64*	*0.71*	*0.77*
Tree-Grid			***0.90***	*0.83*	*0.86*	*0.87*

4.2 Fidelity and Sparsity

The explanatory method performance analysis should be based on the model's prediction results. The interpretation should be faithful to the model, and the explanatory method should identify the input characteristics that are important to the model. There has been a proposal for fidelity metrics to assess this. The key idea is that if the important input features identified by the interpretation technique (node/edge/node features) are discriminant to the model, the model's prediction should significantly change when we remove these features. Therefore, fidelity definition is the difference in accuracy between the original forecast and a new forecast that hides important input characteristics.

By analysing the performance of explanatory methods in terms of input graph data, explanatory methods should be sparse, that is, to capture the most important input features

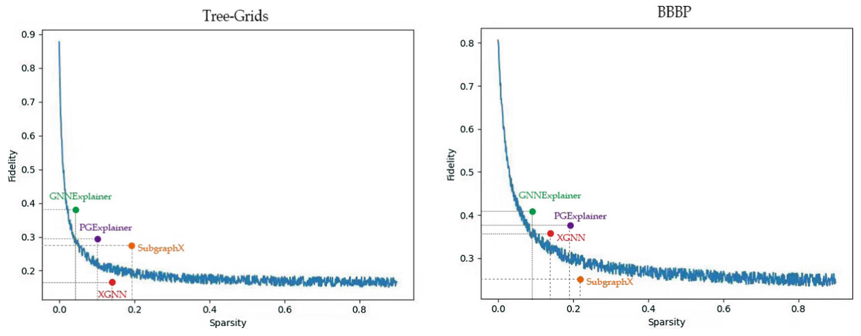


Fig. 3. Fidelity and sparsity adjustment relationship of LAExplainer algorithm.

and ignore irrelevant features; such features measurement can be using sparsity metrics. Specifically, it measures the score chosen by the interpretation method as an important feature. The higher the value, the sparser is the generated data.

In the LAExplainer method, because of the hyperparameters settings, fidelity and sparsity can change the results of the two parts synchronously in the GNNs interpretation process by adjusting the hyperparameters. We drew the relationship curves of fidelity and sparsity in BBBP and tree-grids data sets by adjusting parameters, and comparing them with GNNExplainer, PGExplainer, XGNN, and SubgraphX. The results are in Fig. 3.

Seen from the figure that the evaluation of fidelity and sparsity in the two data sets, LAExplainer based on the existing mainstream algorithm, it is worth noting that fidelity cannot decrease to zero at the end of the curve when the sparsity is large. The algorithm by design cannot generate a large subgraph, so it does not have sufficient guiding significance.

5 Conclusion

In recent years, graph neural networks have been extensively studied, and many model-level interpreters of GNNs have been produced. In this study, we first introduce a new INTERPRETATION framework of GNNs, which focuses on interpreting the model as a whole and analyses the interpretation problem between layers. The LAExplainer algorithm was developed in the process of refining the framework. The algorithm uses the characteristics of the above framework to interpret the layers of GNNs through the subgraph. It aggregates the subgraph into a large subgraph of model interpretation through Shapley values. In addition, the method also analyses the importance of model features and generates an adjustable principal component selection mechanism. Finally, we verified the validity of this method by using synthetic and real data sets and visualized the resulting partial-molecular diagrams. The results produced by observation were also consistent with human understanding.

References

1. Alon, U. Network Motifs: Theory and Experimental Approaches. Nat Rev Genet 2007, 8 (6), 450-461. https://doi.org/10.1038/nrg2102.
2. Baldassarre, F.; Azizpour, H. Explainability Techniques for Graph Convolutional Networks. arXiv:1905.13686 [cs, stat] 2019.
3. Chen, Z.; Li, X.; Bruna, J. Supervised Community Detection with Line Graph Neural Networks. arXiv:1705.08415 [stat] 2020.
4. Duval, A.; Malliaros, F. D. GraphSVX: Shapley Value Explanations for Graph Neural Networks. arXiv:2104.10482 [cs] 2021.
5. Gao, H.; Ji, S. Graph U-Nets. arXiv:1905.05178 [cs, stat] 2019.
6. Gao, H.; Ji, S. Graph Representation Learning via Hard and Channel-Wise Attention Networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; ACM: Anchorage AK USA, 2019; pp 741–749. https://doi.org/10.1145/3292500.3330897.
7. Gao, H.; Wang, Z.; Ji, S. Large-Scale Learnable Graph Convolutional Networks. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining 2018, 1416–1424. https://doi.org/10.1145/3219819.3219947.

8. Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N.; Peters, M.; Schmitz, M.; Zettlemoyer, L. AllenNLP: A Deep Semantic Natural Language Processing Platform. arXiv:1803.07640 [cs] 2018.
9. Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. arXiv:1704.01212 [cs] 2017.
10. Hagberg, A. A.; National, L. A.; Alamos, L.; Schult, D. A.; Swart, P. J. Exploring Network Structure, Dynamics, and Function Using NetworkX. 2008.
11. Hamilton, W. L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs, stat] 2018.
12. Huang, Q.; Yamada, M.; Tian, Y.; Singh, D.; Yin, D.; Chang, Y. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. arXiv:2001.06216 [cs, stat] 2020.
13. Jang, E.; Gu, S.; Poole, B. Categorical Reparameterization with Gumbel-Softmax. arXiv:1611.01144 [cs, stat] 2017.
14. Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. arXiv:1802.04364 [cs, stat] 2019.
15. Jin, W.; Barzilay, R.; Jaakkola, T. Multi-Objective Molecule Generation Using Interpretable Substructures. arXiv:2002.03244 [cs, stat] 2020.
16. Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] 2017.
17. Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs, stat] 2017.
18. Lee, J.; Lee, I.; Kang, J. Self-Attention Graph Pooling. arXiv:1904.08082 [cs, stat] 2019.
19. Lei, T.; Barzilay, R.; Jaakkola, T. Rationalizing Neural Predictions. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Austin, Texas, 2016; pp 107-117. https://doi.org/10.18653/v1/D16-1011.
20. Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. Learning Deep Generative Models of Graphs. arXiv:1803.03324 [cs, stat] 2018.
21. Liu, M.; Luo, Y.; Wang, L.; Xie, Y.; Yuan, H.; Gui, S.; Yu, H.; Xu, Z.; Zhang, J.; Liu, Y.; Yan, K.; Liu, H.; Fu, C.; Oztekin, B.; Zhang, X.; Ji, S. DIG: A Turnkey Library for Diving into Graph Deep Learning Research. arXiv:2103.12608 [cs] 2021.
22. Lundberg, S.; Lee, S.-I. A Unified Approach to Interpreting Model Predictions. arXiv:1705.07874 [cs, stat] 2017.
23. Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; Zhang, X. Parameterized Explainer for Graph Neural Network. arXiv:2011.04573 [cs] 2020.
24. K. Toutanova, K. Dan, C. Manning and Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: North American Chapter of the Association for Computational Linguistics, 2003. https://doi.org/10.3115/1073445.1073478
25. Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; Alon, U. Network Motifs: Simple Building Blocks of Complex Networks. Science 2002, 298 (5594), 824-827. https://doi.org/10.1126/science.298.5594.824.
26. Montavon, G.; Lapuschkin, S.; Binder, A.; Samek, W.; Mueller, K.-R. Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition. Pattern Recognition 2017, 65, 211-222. https://doi.org/10.1016/j.patcog.2016.11.008.
27. Pope, P. E.; Kolouri, S.; Rostami, M.; Martin, C. E.; Hoffmann, H. Explainability Methods for Graph Convolutional Neural Networks. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); IEEE: Long Beach, CA, USA, 2019; pp 10764-10773. https://doi.org/10.1109/CVPR.2019.01103.

28. Shen-Orr, S. S.; Milo, R.; Mangan, S.; Alon, U. Network Motifs in the Transcriptional Regulation Network of Escherichia Coli. Nat Genet 2002, 31 (1), 64-68. https://doi.org/10.1038/ng881.
29. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs] 2014.
30. Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. SmoothGrad: Removing Noise by Adding Noise. arXiv:1706.03825 [cs, stat] 2017.
31. Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; Riedmiller, M. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs] 2015.
32. Sutton, R. S.; McAllester, D.; Singh, S.; Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In IN ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 12; MIT Press, 2000; pp 1057-1063.
33. Thekumparampil, K. K.; Wang, C.; Oh, S.; Li, L.-J. Attention-Based Graph Neural Network for Semi-Supervised Learning. arXiv:1803.03735 [cs, stat] 2018.
34. Tremblay, J.-P.; Sorenson, P. G. An Introduction to Data Structures with Applications, 2nd ed.; McGraw-Hill computer science series; McGraw-Hill: New York, 1984.
35. Velikovi, P.; Cucurull, G.; Casanova, A.; Romero, A.; P Liò.; Bengio, Y. Graph Attention Networks. arXiv:1710.10903 [cs, stat] 2018.
36. Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Xie, X.; Guo, M. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. arXiv:1711.08267 [cs, stat] 2017.
37. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful Are Graph Neural Networks? arXiv:1810.00826 [cs, stat] 2019.
38. Ying, R.; Bourgeois, D.; You, J.; Zitnik, M.; Leskovec, J. GNNExplainer: Generating Explanations for Graph Neural Networks. arXiv:1903.03894 [cs, stat] 2019.
39. You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. arXiv:1806.02473 [cs, stat] 2019.
40. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. arXiv:1609.05473 [cs] 2017.
41. Yuan, H.; Ji, S. StructPool: Structured Graph Pooling via Conditional Random Fields. In International Conference on Learning Representations; 2020.
42. Yuan, H.; Tang, J.; Hu, X.; Ji, S. XGNN: Towards Model-Level Explanations of Graph Neural Networks. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining 2020, 430-438. https://doi.org/10.1145/3394486.3403085.
43. Zeiler, M. D.; Fergus, R. Visualizing and Understanding Convolutional Networks. arXiv:1311.2901 [cs] 2013.
44. Zhang, M.; Chen, Y. Link Prediction Based on Graph Neural Networks. arXiv:1802.09691 [cs, stat] 2018.
45. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. arXiv:1512.04150 [cs] 2015.