Evaluating Interpretability Methods for Graph Neural Networks

Ryoji Kubo
Department of Computer Science
New York University
New York, NY 10003
ryojikubo@nyu.edu

Abstract

Graph Neural Networks (GNNs) has emerged as a powerful tool for machine learning on Graphs. However, its lack of interpretability is still an ongoing challenge. Many interpretability methods have been proposed for GNNs, but it is difficult to evalaute the performance of such interpretability methods. We propose EVAL-XG, an evaluator for GNNs interepretability method, which is able to estimate the true data generation distribution given any subset of inputs. By using EVAL-XG, GNNs interpretability methods can be evaluated on any dataset. Experiments on synthetic dataset show that GNNExplainer, a popular GNNs interpretability method does not perform well evaluated by EVAL-XG, and that their model is not able to give explanation with regards to the true data generation distribution.

1 Introduction

Many real-world data can be naturally represented as graphs. These data can be seen in different domains such as biomedical, chemical, social, and information [Cho et al., 2011, Zitnik et al., 2018]. With the rise of big data, these graphs are becoming increasing complex. To learn interesting characteristics of different graphs and perform downstream tasks such as node classification and edge prediction, Graph Neural Networks (GNNs) have emerged as the state-of-art machine learning architecture on graphs [Hamilton et al., 2018].

Despite GNNs' strong ability to capture different features of a graph, it lacks interpretability similar to other deep models [Ying et al., 2019]. This becomes a problem, for example, if GNNs are employed in high-risk situations such as used for predicting drug-side effects, where domain experts may want to know why the model gives certain outputs [Zitnik et al., 2018].

To tackle this issue, different GNNs interpretability methods have been proposed. One of the popular branch of GNNs interpretability methods is perturbation-based methods. Intuitively, these methods learn explanation by masking supposedely important features (edges, node features) and looking at how the GNNs performance degrades [Yuan et al., 2021]. A particular method that is popular in this branch is GNNExplainer [Ying et al., 2019]. It returns a node feature mask and edge mask that characterizes the importance of the corresponding node features/edges for a particular prediction.

However, this method has a significant drawback. Such perturbation-based methods violates a key assumption in machine learning: the training and evaluation data come from the same distribution. The training data and the masked inputs will come from a different distribution, and thus it is unclear whether the performance degradation comes from distribution shift or due to the masked features being truly informative.[Hooker et al., 2019]

Another challenge in GNNs interpretability research is in the evaluation of interpretability methods. Currently, GNNs interpretability methods are mainly evaluated on synthetic datasets and assessing

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

how well the interpretability method performs in capturing the already-known important features [Yuan et al., 2021]. The drawback of this method is that when such interpretability methods are used on real-world graph dataset, often, we are not able to check its performance on the dataset as it is uncertain what are the supposedly important features in real-world graph datasets.

To address similar issues, in the general machine learning domain, EVAL-X has been proposed [Jethani et al., 2021]. EVAL-X learns to approximate the true data generation distribution of the target given subsets of input by training on samples taken from independently masked inputs. Once an interpretability method returns the subset of inputs that are important for the prediction, EVAL-X can be used to evaluate the interpretability method. This is possible by the predictive performance metrics returned by EVAL-X given the subset of inputs.

We propose EVAL-XG, an equivalent of EVAL-X but for GNNs. The main contribution is that we translate EVAL-X for graphs by taking in consideration both the edges and node features as the input space. By using EVAL-XG, we can address two problems in GNNs interpretability methods research: (i) we have an evaluation metric that can be used in any dataset, (ii) we can make sure that the interpretability method respects the true data generation distribution given subset of inputs. (i) can be achieved by the predictive performance metrics returned by EVAL-XG given the subset of inputs, and (ii) can be achieved by instead of using the model trained on the full feature set to learn the explanation, we use EVAL-XG to return a explanation for the prediction.

We conduct experiments to see how model trained only on the full feature set cannot learn the true data generation distribution. We also evaluate GNNExplainer, and see how the quality of explanation differs when using EVAL-XG instead of a model trained only on the full feature set.

The experiments were conducted on synthetic datasets for a node classification task for simplicity. (The code and datasets are available at https://github.com/ryoji-kubo/EVAL-XG)

1.1 Related Work

1.1.1 GNNExplainer

GNNExplainer is one of a perturbation-based interpretability method. Intuitively, it learns which node features and edges are important for a prediction by the performance degradation of the model when certain inputs are masked. The mask will entail information of which node features/edges are important. GNNExplainer uses soft masks which contains continuous values between $[0, 1]$, and since this mask will be combined with the node features, it has been pointed out that they suffer from "introduce evidence" problem, that any non-zero or non-one value in the mask may introduce new semantic meaning. The masks are optimized by maximizing the mutual information of the original prediction and the prediction based on the newly masked graph. [Ying et al., 2019, Yuan et al., 2021]

1.1.2 EVAL-X

Let features $\mathbf{x}$ be a random vector in $\mathbb{R}^D$, and the response $\mathbf{y} \in \{1, ..., K\}$. For a given positive integer $j \leq D$, let $\mathbf{x}_j$ be the jth component of $\mathbf{x}$ and $\mathbf{x}_\mathcal{S} := \{\mathbf{x}_j\}_{j \in \mathcal{S}}$ ve a subset of features, where $\mathcal{S} \subseteq \{1, ..., D\}$. F is a distribution over $(\mathbf{x}, \mathbf{y})$. For every instance $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \sim F(\mathbf{x}, \mathbf{y})$, *instance-wise feature selection* (IWFS) identified a minimal subset of features $\mathbf{x}^{(i)}_{\mathcal{S}^{(i)}}$ such that under the conditional distribution $F(\mathbf{y}|\cdot)$ [Yoon et al., 2019].

$$F(\mathbf{y}|\mathbf{x}_{\mathcal{S}^{(i)}} = \boldsymbol{x}^{(i)}_{\mathcal{S}^{(i)}}) = F(\mathbf{y}|\mathbf{x} = \boldsymbol{x}^{(i)}) \tag{1}$$

$F(\mathbf{y}|\mathbf{x})$ is either the population distribution from which the data is drawn or a trained model.

Evaluation of IWFS should be done on the true conditional distribution. Evaluating any potential selection of a subset of features $\mathcal{R}$ requires access to $F(\mathbf{y}|\mathbf{x}_\mathcal{R})$. This can be done by training on randomly masked $\mathbf{x}$. EVAL-X trains an evaluator model $q_{\text{eval-x}}$ to estimate $F(\mathbf{y}|\mathbf{x}_\mathcal{R})$ by maximizing

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim F}\mathbb{E}_{\boldsymbol{r}\sim\mathcal{B}(0.5)}[\log q_{\text{eval-x}}(\boldsymbol{y}|m(\boldsymbol{x}, \boldsymbol{r}); \eta)] \tag{2}$$

$\boldsymbol{r}$ is sampled randomly, independent of $\mathbf{x}$, from a Bernoulli distribution, mimicking any potential selection of the input. The masking function m replaces features with a mask token $\mathcal{M}$ using binary indicator $\boldsymbol{r}$. For example, whether to hide the jth feature $\boldsymbol{x}_j$ is chosen by selector variable $\boldsymbol{r}_j$:

$$m(\boldsymbol{x}^{(i)}, \boldsymbol{r}^{(i)})_j = \begin{cases} \boldsymbol{x}^{(i)}_j & \text{if } \boldsymbol{r}^{(i)}_j = 1 \\ \mathcal{M} & \text{if } \boldsymbol{r}^{(i)}_j = 0 \end{cases} \tag{3}$$

In this case, letting $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_D$ be a D-dimensional feature space, the mask token $\mathcal{M}$ is chosen such that $\mathcal{M}$ is not in any of the feature spaces $\mathcal{X}_1 \times ... \times \mathcal{X}_D$. At optimality, EVAL-X learns the true $F(\mathbf{y}|\mathbf{x}_\mathcal{R})$. Predictive performance metrics returned by EVAL-X should be used to quantitatively evaluate selections. [Jethani et al., 2021]

2 EVAL-XG: Evaluator for Graph Neural Networks

We now describe how we can build EVAL-XG, an equivalent of EVAL-X for GNNs. We first introduce some preliminaries we refer to throughout the paper.

Let G denote a graph on edges E and nodes V that are associated with D-dimensional node features $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^D$. $\mathbf{x}^l$ denotes the l th node feature. For simplicity, we consider the problem of explaining a node classification task. Let f denote a label function on nodes $f : V \mapsto \{1, ..., C\}$ that maps every node in V to one of C classes. The GNN model Φ is optimized on all nodes in the training set and is then used for prediction, i.e., to approximate f on new nodes.

Let us denote the computation graph for node v as $G_c(v)$, the associated adjacency matrix by $A_c(v) \in \{0, 1\}^{n \times n}$, and the associated feature set by $X_c(v) = \{\mathbf{x}_j | v_j \in G_c(v)\}$. The l th node feature set is denoted by $X_c(v)_l = \{\mathbf{x}_j^l | v_j \in G_c(v)\}$. Each instance of these are distributed as $(G_c(v)^{(i)}, X_c(v)^{(i)}, y^{(i)}) \sim F(G_c(v), X_c(v), Y)$, where Y is a random variable representing labels $\{1, ..., C\}$ indicating the probability of nodes belonging to each of C classes [Ying et al., 2019].

The GNN model Φ learns a conditional distribution $F(Y|G_c, X_c)$ which is equivalent to learning f.

Here, we consider the explanation $(G_\mathcal{S}(v), m(X_\mathcal{S}(v), r))$ given for each prediction $\hat{y}$ for v, where $G_\mathcal{S}(v)$ is a small subgraph of the computation graph. $X_\mathcal{S}(v)$ is the associated features of $G_\mathcal{S}(v)$, and $m(X_\mathcal{S}(v), r)$ is a small subset of node features masked out by the selector variable r. The masking function m takes as input $X_\mathcal{S}(v)$ and r and masks out the selected features by replacing the features with a mask token $\mathcal{M}$. For example, indicator r_l is used to control whether to mask the l th node feature for the node features in $X_\mathcal{S}(v)$:

$$m(X_S(v), r)_l = \begin{cases} X_\mathcal{S}(v)_l & \text{if } r_l = 1 \\ \mathcal{M} & \text{if } r_l = 0 \end{cases} \tag{4}$$

$\mathcal{M}$ is chosen again such that it is not in any of the D-dimensional feature space.

Following EVAL-X, we will need to evaluate the selection of features on the true data generation distribution, $F(Y|G_{\mathcal{R}_1}(V), m(X_{\mathcal{R}_1}(V), \mathcal{R}_2))$ where $\mathcal{R}_1$ is any potential selection of a subgraph of G_c and $\mathcal{R}_2$ is any potential selection of a subset of node features $X_{\mathcal{R}_1}$. This can similarly be estimated using $q_{\text{eval-xg}}$ by maximizing

$$\mathbb{E}_{G_c(v), y \sim F} \mathbb{E}_{G_\mathcal{S}(v) \sim B_1(G_c(v); 0.5)} \mathbb{E}_{r \sim B_2(X_\mathcal{S}(v); 0.5)} [\log q_{\text{eval-xg}}(y | G_S(v), m(X_S(v), r); \Phi)] \tag{5}$$

$B_1(G_C(v); 0.5)$ returns the adjacency matrix $A_\mathcal{S}(v) \in \{0, 1\}^{n \times n}$ for the subgraph $G_\mathcal{S}(v)$ by obtaining the selector variable s using independent Bernoulli distribution with probability 0.5, denoted $\mathcal{B}(0.5)$:

$$A_\mathcal{S}(v)[i, j] = \begin{cases} 1 & \text{if } A_c(v)[i, j] = 1 \text{ and } s[i, j] = 1 \\ 0 & \text{if } A_c(v)[i, j] = 0 \text{ or } s[i, j] = 0 \end{cases} \tag{6}$$

where $s \sim \mathcal{B}(0.5)$ Notice, by using this sampling method, the subgraph constraint of $A_\mathcal{S}[i, j] \leq A_c[i, j] \; \forall i, j$ is enforced.

$B_2(X_\mathcal{S}(v); 0.5)$ returns the selector variable r that indicates which node features to be masked from $X_\mathcal{S}(v)$, sampled from $\mathcal{B}(0.5)$.

Predictive performance metrics returned by EVAL-XG can then be used to quantitatively evaluate selections returned by interpretability methods.

3 Experiments

In the following experiments, we assess these two main questions.

Q1 Can EVAL-XG learn the true distribution $F(Y|G_{\mathcal{R}_1}(V), m(X_{\mathcal{R}_1}(V), \mathcal{R}_2))$?

Q2 How does the quality of explanation given by GNNExplainer differ when optimized using EVAL-XG?

We do this by comparing in total of four predictive performance metrics (**M1 - M4**) of a model trained only on the full feature set (full-model) to that of EVAL-XG. We perform these experiments on a synthetic dataset for simplicity.

First, full-model and EVAL-XG are trained on a task on the dataset. Afterwards, we employ GNNExplainer using full-model to give selections of node features/edges. We pass these selections to EVAL-XG and full-model and get the two predictive performance metrics. We repeat this using GNNExplainer optimized on EVAL-XG and get two similar metrics.

M1 Predictive performance metric of EVAL-XG using selections returned by GNNExplainer optimized on full-model

M2 Predictive performance metric of full-model using selections returned by GNNExplainer optimized on full-model

M3 Predictive performance metric of EVAL-XG using selections returned by GNNExplainer optimized on EVAL-XG

M4 Predictive performance metric of full-model using selections returned by GNNExplainer optimized on EVAL-XG

To answer **Q1**, we compare **M1** to **M2**, and **M3** to **M4**. The hypothesis is that if EVAL-XG is able to learn the true distribution $F(Y|G_{\mathcal{R}_1}(V), m(X_{\mathcal{R}_1}(V), \mathcal{R}_2))$, the performance metrics should be higher for that of EVAL-XG than full-model. (**M1** $>$ **M2** and **M3** $>$ **M4**)

To answer **Q2**, we compare **M1** to **M3**, and the hypothesis is that GNNExplainer optimized on EVAL-XG can return better explanation compared to that of full-model. (**M1** $<$ **M3**)

3.1 Synthetic Dataset

We construct a synthetic dataset of a cycle graph of length 2000. Each node v has the following independently sampled node features $\{x_i\}_{i=1}^{i=5}$ and binary label y_v.

$$\{x_i\}_{i=1}^{i=3} \sim \begin{cases} \mathcal{N}(2,1) & \text{with probability } 0.5 \\ \mathcal{N}(-2,1) & \text{with probability } 0.5 \end{cases} \tag{7}$$

$$\{x_i\}_{i=4}^{i=5} \sim \mathcal{N}(0,1) \tag{8}$$

$$y_v \sim \text{Bernoulli}\left(\frac{1}{1+g(\mathbf{x}_v, \mathbf{x}_{N(v)})}\right) \tag{9}$$

$$g(\mathbf{x}_v, \mathbf{x}_{N(v)}) = \text{self}(\mathbf{x}_v) + \text{neighbor}(\mathbf{x}_{N(v)}) \tag{10}$$

$$\text{self}(\mathbf{x}_v) = \exp(\Sigma_{i=1}^{i=3} x_{v,i} - 3) \tag{11}$$

$$\text{neighbor}(\mathbf{x}_{N(v)}) = \Sigma_{j \in N(v)} \exp(\Sigma_{i=1}^{i=3} x_{j,i} - 3) \tag{12}$$

With this synthetic dataset, we know that only the first three node features and the two incoming edges are important for predicting y_v. Thus, we only passed the top three node features and top two edges selected by GNNExplainer for the node classification task.

3.1.1 Model Training

We split the nodes into train/val/test set of $1400/300/300$ nodes. We used a model with an architecture of linear $\rightarrow$ GCN $\rightarrow$ linear $\rightarrow$ linear. Both EVAL-XG and full-model were trained for 500 epochs using Adam for optimization. No hyper-parameter tuning was done for any of the models. The exponential of node features were used such that the mask token $\mathcal{M} = 0$ was not in the node feature space. Although $\mathcal{B}(0.5)$ was used for sampling $A_{\mathcal{S}}(v)$, $\mathcal{B}(0.2)$ was used instead for sampling r. This modification still enables EVAL-XG estimating the true distribution $F(Y|G_{\mathcal{R}_1}(V), m(X_{\mathcal{R}_1}(V), \mathcal{R}_2))$

Table 1: Results on Synthetic Dataset

	M1	**M2**	**M3**	**M4**
Pair 1	63.3%	57.7%	63.3%	58.3%
Pair 2	58.7%	54.7%	56.7%	56.7%
Pair 3	59.0%	53.7%	59.3%	55.3%
Pair 4	56.3%	56.3%	62.3%	61.3%
Pair 5	58.0%	54.3%	58.0%	55.0%

since any feature subset can be obtained. This change was due to the fact that with $r \sim \mathcal{B}(0.5)$, the models were not receiving enough important features to learn the distribution. GNNExplainer was trained for 200 epochs for each prediction. Explanation was only derived for the nodes in the test set as well as the performance of whether they can predict node label given the explanation.

3.1.2 Results

5 different pairs of EVAL-XG and full-model were trained and their corresponding score (**M1 - M4**) were recorded. We used a metric of accuracy. We summarized our results on Table 1. First of all, we notice by looking at **M1** and **M3** that GNNExplainer does not perform well by the evaluation given by EVAL-XG. In terms of addressing **Q1**, we note that in general, **M1** $>$ **M2** and **M3** $>$ **M4** seems to hold. This supports the claim that EVAL-XG is able to estimate the true distribution better than full-model. However, for **Q2**, we do not see a general trend of **M1** $<$ **M3**. This could be because GNNExplainer does not perform well at giving the explanation, and that we are not able to see the improvement from using a model that estimates the true distribution.

4 Discussion

We proposed EVAL-XG, an evaluator for interpretability methods for Graph Neural Networks. EVAL-XG creates an evaluator model to approximate the true data distribution given any subset of inputs. We have conducted experiments to confirm that EVAL-XG is able to estimate the true data distribution better than a model only trained on the full feature set.

There remains certain challenges for EVAL-XG. One is that it is infeasible to check whether EVAL-XG estimates the true data distribution well. That would require training different models for each feature subset and comparing the predictive performance metric. Another challenge is in the difficulty of learning the true distribution. More research on how to sample the different feature subset efficiently would allow for a faster convergence of EVAL-XG.

References

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 1082–1090, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308137. doi: 10.1145/2020408.2020579. URL `https://doi.org/10.1145/2020408.2020579`.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks, 2019.

Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations, 2021.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks, 2019.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJg_roAcK7.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey, 2021.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 06 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty294. URL https://doi.org/10.1093/bioinformatics/bty294.