CX4071 Network Science

Project 2 Report - GNNExplainer

Name	Matriculation No.
Chandrasekhar Aditya	U1923951A
Vincent Yong Wei Jie	U1820986L

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

1. Introduction

Graphs serve as powerful data representations that models actors in a network and the relationships among them. However, they are challenging to work with due to the need to model the relational information as well as the node features. As such, Graph Neural Networks (GNNs), which capture both graph structure and node features, have been developed for a multitude of tasks. These tasks include node classification and link prediction. GNNs are connectionist models that help capture the dependence of graphs via message passing between connected nodes. They also retain a state that can represent information from its neighbourhood with arbitrary depth unlike the standard neural networks.

2. Problem

In deep learning, it is not uncommon to treat deep neural networks as 'black boxes' as it is believed that such networks have learnt the salient features to perform their tasks. However, this can lead to cases of unexplainable misclassifications or errors by the model which are only discovered after deployment.

Similarly, GNNs lack transparency as there is currently no easy way to decipher why the network has output certain results. Without such tools to account for and explain the model's output, it hinders the adoption of GNNs for important tasks. Lack of model explainability makes it hard to trust the model and also makes it more challenging to identify systematic patterns of mistakes the networks might have made in the training phase.

2.1. Current Approaches

Currently, there are 2 approaches for this problem of model explainability in GNNs.

The first method is to approximate models with simpler surrogate models, which are then probed for explanations. The key in this method is that by establishing simpler surrogate models, it retains how the model is producing its output while making it easier to analyse since it is smaller.

The second method is to identify important aspects of the computation. This means that important features are identified in the input which are then picked up by the model to make its prediction. This allows qualitative conclusions to be made about which set of features the model is looking for. However, this method does not work well on discrete inputs such as a graph's adjacency matrix, limiting its effectiveness on GNNs. Another way to identify such kinds of features is to use attention mechanisms where the attention values indicate which part of the graph the model is focussing on and is thus, more important. However, current attention mechanisms are unable to model the unique attention values that each edge has with respect to a node and instead has global values across all nodes.

3. Contributions

With the importance of model explainability to build trust in GNNs and the limitations of current approaches, GNNExplainer, an approach for explaining predictions made by GNN, have been proposed and developed

GNNExplainer tackles the problem of model explainability in node classification by returning an explanation in the form of a small subgraph of the input graph together with a small subset of node features that are most influential for the prediction. In the paper, the authors' make the observation that only the subset of the graph structure and the node features are required to explain the output. As such, the explanation only consists of these aspects.

GNNExplainer finds the subgraph and the subset of node features by maximising the Mutual Information (MI) which is the change in the probability of a certain prediction when the graph is constrained to the subgraph (G_s) and the subset of neighbouring node features (X_s). Intuitively, when there is an important node or node feature, the removal of it will lead to a large change in the probability of making that prediction. Hence, the large change is indicative that the particular node or node feature should be included in the explanation. The equation is shown in the image below.

$$\max_{G_S} MI\left(Y, (G_S, X_S)\right) = H(Y) - H(Y|G = G_S, X = X_S).$$

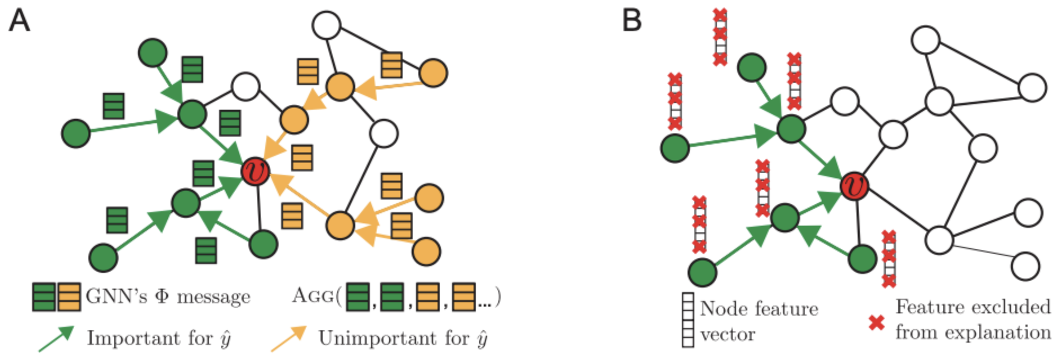


Figure 2: **A.** GNN computation graph G_c (green and orange) for making prediction $\hat{y}$ at node v. Some edges in G_c form important neural message-passing pathways (green), which allow useful node information to be propagated across G_c and aggregated at v for prediction, while other edges do not (orange). However, GNN needs to aggregate important as well as unimportant messages to form a prediction at node v, which can dilute the signal accumulated from v's neighborhood. The goal of GNNEXPLAINER is to identify a small set of important features and pathways (green) that are crucial for prediction. **B.** In addition to G_S (green), GNNEXPLAINER identifies what feature dimensions of G_S's nodes are important for prediction by learning a node feature mask.

3.1. Generic, model-agnostic

Modern GNNs are based on message passing architectures on the input graph which can be composed in many ways and it is possible to compose it in the way used by the paper. Thus GNNExplainer can be applied to many types of GNNs such as Gated Graph Sequence Neural Networks and Line-Graph Neural Networks. This makes it versatile, greatly increasing the number of areas in which this research can be applied to.

3.2. Works on multiple tasks

GNNExplainer can be used to explain predictions for other tasks such as link prediction and graph classification, with small tweaks to the method presented in the paper. For example, in the case of link prediction, instead of node masks to create subgraphs for the node classification task, edge masks are needed to create the subgraphs. This flexibility allows the research to be used in more areas of GNNs.

3.3. Incorporate relational information

Unlike previous approaches, GNNExplainer is able to incorporate relational information, which is the essence of graphs in its explanations. As mentioned above, GNNExplainer calculates the change in probability of outputting a certain prediction. The function, shown in the image above, to calculate the probability takes into account the subgraph structure and also the node features of neighbouring nodes around the node that we are trying to classify. As such, it can be easily seen that the important relational data is captured in both the subgraph structure and the neighbouring nodes' features.

3.4. Multi-instance explanations

Single-instance explanation accounts for the GNN outputting a certain label for a node. This capability has already been explained above. Multi-instance explanations expand on this by providing explanations for why multiple nodes have the same class predicted by the GNN. Multi-instance explanations are achieved by first generating single-instance explanations for each instance in the particular class being analysed. Pattern matching is then performed to identify common patterns across all the single-instance explanations. These common patterns serve as the explanation for all these nodes being classified as a particular class.

3.5. Computational Complexity

The number of parameters in GNNExplainer depends on the size of the computation graph G_c for a node v whose prediction we want to explain. The size of the mask M which is to be learned is equal to the size of the adjacency matrix of $G_c(v)$. Since the computation graphs are relatively small compared to the size of the exhaustive L-hop neighbourhood (Nodes at distance L from node v), GNNExplainer effectively generates explanations even when the input graphs are huge.

4. Connections with CX4071 Content

4.1. Synthetic Graph Generation

In this paper, the authors mentioned that to construct synthetic datasets for node classifications, they used the Barabasi-Alberta (BA) model for generating graphs. The BA model is used to generate a power-law degree distribution in graphs so that they're representative of real world graphs.

The BA model takes into account both growth and preferential attachment. In the BA model, we start with N nodes with links between which are chosen arbitrarily as long as each node has at least one link. As each step, we add a new node with M links that connects the new node to M nodes in the network. This connection is done by using preferential attachment. In preferential attachment, the probability that a node connects to a node with k links is proportional to k.

The graph networks created using this model are representative of 3 main quantitative properties that are seen in real world graphs:

1. Average Path Length - average distance between any 2 pair of nodes in the graph
2. Clustering Coefficient - It measure the fraction of a node's neighbourhood that are connected
3. Degree Distribution - probability that a randomly chosen node has degree k

This concept was used by the authors to create:

1. **BA-Shapes**
   In this, we start with a base BA graph of 300 nodes and a set of 80 five-node 'house'-structure network motifs, which are attached to randomly selected nodes of the base graph. The resulting graph is further perturbed by adding 0.1N random edges. Nodes are assigned to 4 classes based on their structural roles.

2. **BA-Community**
   This dataset is a union of 2 BA-Shapes graphs. Nodes have normally distributed feature vectors and are assigned to one of 8 classes based on their structure roles and their community memberships.

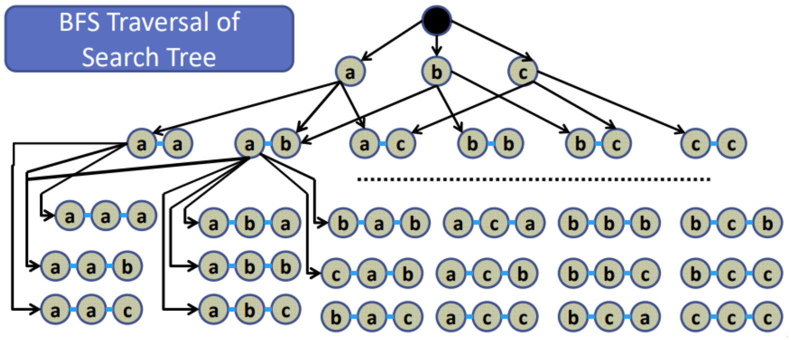
4.2. Subgraph Mining

In the cost function shown in Section 3, there is a need to mine subgraphs from the graph to calculate the mutual information values. From CX4071, we have learnt that mining subgraphs is a problem with exponential time complexity. The authors also support this statement by stating that "Direct optimization of GNNExplainer's objective is not tractable as G_c has exponentially many subgraphs".

In CX4071, we learnt the Apriori Approach and Pattern Growth Approach for mining subgraphs.

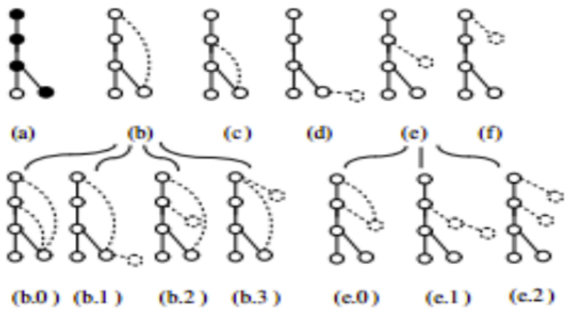
4.2.1. Frequent Subgraph Mining (FSG)

The FSG algorithm is an Apriori Approach that generates subgraphs of size K+1 edges by joining candidates of size K edges. Once these candidates are generated, we count the frequency of each element in the candidate set to find out whether they're frequent or not. However, this approach produces too many candidates at a high depth of the search tree and the removal of duplicates is expensive. More memory is also consumed since the search tree is traversed using the BFS algorithm.



4.2.2. gSpan Algorithm

The gSpan algorithm is a Pattern Growth Approach where we first sort all frequent 1-edge graphs into a list S based on their DFS lexicographical order. For each graph in S, we add one new edge to construct all its children. The new edge can only be added at nodes that lie on the rightmost path of DFS-tree. The child is pruned if it does not represent its minimum DFS code or if it is infrequent, reducing the search space size. This avoids patterns that have already been visited and makes sure no missing patterns that should have been visited.



In this research paper, the author uses a NxN adjacency matrix where N is the number of nodes to mask out certain parts of the graph to produce subgraphs. They then continue to mine subgraphs from these initial subgraphs which are potential candidates in explaining the GNN's predictions. This implies a similar approach as gSpan.

To determine the explanation subgraph, the importance weight of edges (gradients for GRAD baseline, attention weights for ATT baseline, and masked adjacency) is calculated. A threshold value is used to remove the low-weight edges and identify the explanation subgraph.

4.3. Pattern Matching

In this paper, the authors mentioned that this same method can be used to generate multi-instance explanations as well. The authors “propose a solution based on GNNExplainer to find common components of explanations for a set of 10 explanations”. This suggests that from the 10 sets of explanations, they will search for common subgraphs to generate a global explanation for the set of nodes that the GNN has predicted labels for. This is similar to the problem of pattern matching in CX4071.

In CX4071, we learnt Ullmann's Backtracking Algorithm and gIndex for Pattern Matching.

4.3.1. Ullmann’s Backtracking Algorithm

Given a query graph Q of size n and a data graph G of size m, we create a matrix M of dimensions (n,m). Matrix M is filled with 1’s where there’s a matching label between Q and G, and the rest of the values are 0. From this matrix M, we perform a DFS traversal to backtrack and find if the subgraph exists in the data graph.

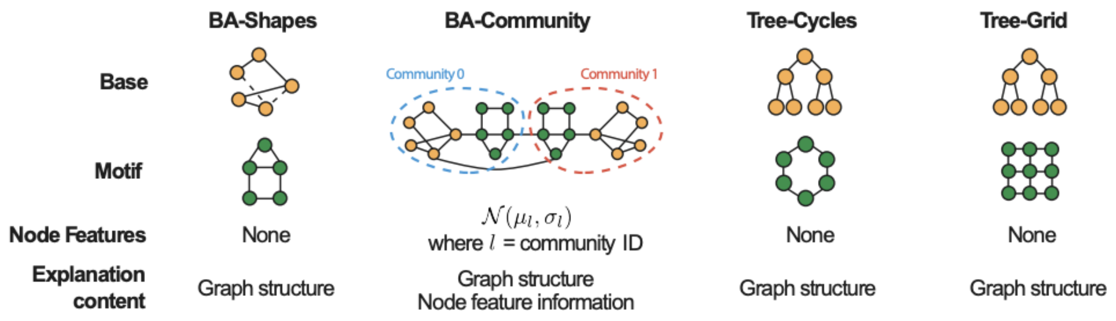
4.3.2 gIndex

In this algorithm, we index the graph structure by making use of frequent substructure as the basic indexing feature. Frequent substructure are ideal candidates since they explore the intrinsic characteristics of the data and are relatively stable to database updates. To reduce the size of index structure, 2 techniques, size-increasing support constraint and discriminative fragments, are used. When the database is queried for a pattern, the indexed data is used to find where the patterns are matched and the relevant graph structures are returned.

In the paper, the authors measure the overlap in the explanations using matrix operations of the subgraphs’s adjacency matrix which has been explained in Section 4.1. They term the overlap of these explanations as alignment, which is used as a measurement of how similar they are. The use of matrix operations to calculate alignment is more efficient as matrix operations can be done in parallel, making use of hardware like Graphic Processing Units (GPUs) to accelerate this process. In addition, the authors state that the single-instance explainer “produces concise subgraph” and hence, “a matching that is close to the best alignment can be efficiently computed”. As such, the methods used by the authors may be more efficient than what we have learnt in CX4071.

5. Discussion of experiments

In the paper proposed, the authors validate GNNExplainer with synthetic datasets as well as real-world datasets. In replicating their experiments, we only replicate the ones done on the synthetic datasets as the graphs are often of a smaller size and thus, more practical for us to replicate with limited compute resources. These different types of synthetic datasets and results are shown below.



	BA-Shapes (Base)	BA-Shapes (Motif)
Accuracy (From paper)	0.925	0.836
Accuracy (From replication)	0.971	0.911

In the source code, we have a function that helps to generate the synthetic BA-Shapes graphs known as gen_syn1 in gengraph.py. We try to investigate the impact of the graph size, in terms of the number of nodes, on the accuracy. Many domains of deep learning such as computer vision and natural language processing typically have greater accuracy as the dataset size increases. We thus expect to see similar patterns in graph neural networks. The experiment results are shown below.

Number of nodes	30	300	3000
Accuracy	0.965	0.971	0.999

From the results shown above, we see that there seems to be some correlation between the size of the graph and the test accuracy. As such, it might suggest that these GNNExplainer perform better with larger graphs. However, like many other deep learning applications, increasing the dataset size comes with drawbacks such as a linearly increasing training time. We observed this as we increased the size of the graph from 300 to 3000 nodes, which took approximately 20 times longer to train. Hence, we were unable to obtain results for larger graphs.

6. Limitations

6.1. Explanation size

In the paper, the authors limit the size of the subgraph to have at most K_M nodes. The authors do not explain how the threshold was established. However, as we have learnt from the gIndex algorithm, it is hard to find a good threshold level and a fixed threshold is not useful due to different characteristics of different graphs. Hence, gIndex introduces an adaptive threshold for minimum support based on the size of the graph. This could be one of the ways to build upon what the authors have created.

Another method would be to not limit the graph explanation size at all. Instead, we can normalise the MI function by the size of the subgraph and the subset of node features used. Intuitively, when we strip away more nodes and features to create the subgraph and the subset of node features, we would expect a greater change in probability. As such, we should normalise it with how much the original graph has changed to have a better measure for MI.

7. Conclusion

GnnExplainer is a novel method for explaining predictions of Graph Neural Network on any graph-based machine learning task without requiring modification of the underlying architecture of the model. GnnExplainer is able to leverage recursive neighborhood aggregation on graph neural networks to identify important graph pathways as well as highlight relevant node feature information that is passed along edges of these paths. From our analysis, we see the connections between the coursework of CX4071 and the paper in graph pattern matching and graph pattern mining. We also created multiple synthetic datasets and replicated the results as the research paper.