Interpreting Graph Neural Network Predictions: Exploring Explainability and Insights

Muhammad Faisal, Bhavesh Wani, and Vaibhav Chaudhari

Department of Computer Science, Paderborn University, Germany

Abstract. The project focuses on the analysis, training, evaluation, and explanation of a Graph Neural Network (GNN) model using the Planetoid Cora dataset. The GNN model is implemented using the PyTorch Geometric library and the Graph Convolutional Network (GCN) architecture. The report provides an overview of the dataset, describes the GNN model architecture, and details the training and evaluation process. Additionally, it explores the explanation capabilities of the GNN model using the GNNExplainer algorithm. The report concludes with a summary of the findings and highlights the potential applications of GNNs in graph-based data analysis.

Keywords: Graph Neural Network· GNN· node classification· Cora dataset

1 Introduction

Graph Neural Networks (GNNs) have emerged as a powerful tool for analyzing and modeling graph-structured data. Their ability to capture complex relationships and dependencies within graphs has led to successful applications in various domains, including social networks, recommendation systems, and bioinformatics. In this report, we focus on applying GNNs to the Cora dataset, a widely used academic citation network, with the aim of predicting the research field of papers and providing explanations for the model's predictions using the GNNExplainer algorithm.

To begin, we conduct a thorough data analysis of the Cora dataset to gain insights into its characteristics. The dataset consists of papers represented as nodes and citations represented as edges. Each paper is assigned a class label indicating its research field. By analyzing the dataset, we can understand its size, the distribution of research fields, and the structure of the citation network.

Furthermore, we describe the GNN model architecture and the training process. GNNs operate by propagating information through the graph, allowing each node to gather and update information from its neighboring nodes. This propagation process is typically performed through multiple layers, with each layer refining the node representations based on the information from its neighbors.

The training process involves splitting the dataset into training, validation, and test sets. We use the training set to optimize the model's parameters through backpropagation and gradient descent. To prevent overfitting, we employ regularization techniques, such as dropout or graph-based regularization. Additionally, we fine-tune hyperparameters using the validation set to ensure optimal model performance.

Once the model is trained, we evaluate its performance on the test set. We measure various evaluation metrics to assess how well the model predicts the research field of papers. Comparing the model's performance against baselines and previous state-of-the-art approaches provides insights into its effectiveness.

To provide explanations for the model's predictions, we utilize the GNNExplainer algorithm. GNNExplainer helps uncover the important features and relationships used by the model to make predictions. It identifies the most influential nodes and edges in the graph and highlights their contributions to the prediction. By visualizing and interpreting these explanations, we gain insights into the factors that influence the model's decision-making process.

In conclusion, this report focuses on applying GNNs to the Cora dataset for predicting the research field of papers and explaining the model's predictions using the GNNExplainer algorithm. Through data analysis, training, evaluation, and explanation techniques, we aim to provide a comprehensive understanding of the GNN's performance and interpretability.

2 Data analysis

In this project, we aim to explain the predictions of a graph neural network (GNN) on the "Planetoid_Cora" dataset. The dataset contains information about academic research papers and their citation relationships. The data is structured in a graph format, with nodes representing papers and edges representing citations between papers.

2.1 Dataset Overview

The dataset consists of 2,708 nodes (papers) and 10,556 edges (citations). Each paper is represented by 1,433 features, and each paper is assigned to one of several class labels. The class labels represent different research topics or categories that papers belong to.

2.2 Node Features and Labels

The node features of each paper are represented by a 2D tensor of shape [2708, 1433], where 2708 is the number of papers, and 1433 is the number of features. We converted the node features to a numpy array for further analysis.

The class labels of the papers are represented by a 1D tensor of shape [2708]. Each element in this tensor is an integer representing the class label of the corresponding paper. We also converted the labels to a numpy array for analysis.

2.3 Edge Information

The edges in the graph represent citations between papers. The edge information is represented as a 2D tensor of shape [2, 10556], where each column represents a pair of node indices (source and target) representing a citation between two papers.

2.4 Data Preprocessing for Analysis

To facilitate our analysis, we transformed the graph data into a more interpretable format. We created a list of triples, each containing a subject, predicate, and object, to represent the relationships between nodes, features, and labels. For each paper node, we created triples with predicates ‘rdf:type’ and ‘label’, representing the fact that the node is a ’Paper’ and its corresponding research label, respectively. Additionally, we created triples for each feature of the paper using predicates ‘word_i’, where ‘i’ represents the index of the feature, and the feature value as the object.

We then converted the list of triples into a pandas DataFrame called ‘edge_df’ for easy analysis and visualization.

2.5 Analysis

Edge DataFrame (edge_df): The *‘edge_df’* contains information about the relationships between nodes, features, and labels. The DataFrame has a total of X rows and 3 columns: ‘subject’, ‘predicate’, and ‘object’. The first five rows of the DataFrame are shown to give an overview of the data structure.

Node DataFrame (node_df): The ‘node_df’ contains node features and their corresponding class labels. The DataFrame has a total of Y rows and Z+1 columns: Z columns for features (labeled as ‘0’, ‘1’, ..., ‘Z-1’) and an additional column for the class label (‘label’). The first five rows of the DataFrame are shown for an initial understanding of the features and labels. We checked for any missing values in the DataFrame and confirmed that there are none. Descriptive statistics were computed for the node features, providing insights into the statistical distribution of the feature values.

3 GNN Training and Evaluation

The GNN model used in this report is a two-layer Graph Convolutional Network (GCN). The model takes the node features and edge indices as input and applies graph convolutional layers to propagate information through the graph. The output of the model is a probability distribution over the class labels for each node.

To train the GNN model, we used the Cora dataset, dividing it into training, validation, and test sets. We employed a training loop that iteratively updates the model's parameters to minimize the negative log-likelihood loss between the predicted and true labels. We also applied dropout regularization to prevent overfitting during training. The training process involved optimizing the model's parameters using an optimizer, such as Stochastic Gradient Descent (SGD).

While training, we have tried different combination of the learning rate and dropout values as shown in Table 1. We get the best performance with dropout 0.85 and learning rate 0.010 and train the model for 8 iterations.

Table 1. Top 5 results for model training and testing

# of iterations	dropout (p)	learning rate (lr)	validation accuracy	testing accuracy
8	0.85	0.010	0.706	0.721
8	0.35	0.015	0.640	0.648
1	0.10	0.055	0.622	0.634
4	0.60	0.020	0.616	0.627
7	0.70	0.075	0.592	0.610

4 Explanation of GNN

GNNExplainer is a module in PyTorch Geometric that implements the GNN-Explainer model from the "GNNExplainer: Generating Explanations for Graph Neural Networks" paper. It is a type of discriminative explainer that aims to identify compact subgraph structures and node features that play a crucial role in the predictions made by a GNN.

GNNExplainer works by learning a mask over the input graph that highlights the most relevant nodes and edges for a given prediction. It optimizes the mask by maximizing the mutual information between the masked graph and the prediction, while minimizing the number of masked elements. It also uses a sparsity regularization term to encourage sparse masks.

GNNExplainer can be used to explain different types of GNN models and tasks, such as node classification, graph classification, and link prediction. It can also handle both homogeneous and heterogeneous graphs. It provides an intuitive way to understand how GNNs make decisions and what features and substructures they rely on.

To evaluate the trained GNN model, we tested its performance on the validation and test sets. We computed accuracy metrics to assess the model's ability to predict the correct research field for each paper. Additionally, we compared the model's performance with other models trained on different hyperparameters.

To provide explanations for the GNN model's predictions, we employed the GNNExplainer algorithm. This algorithm generates node and edge masks that indicate the importance of features and relationships for each prediction. By visualizing these masks, we can gain insights into which features and relationships the model considers most significant for its predictions.

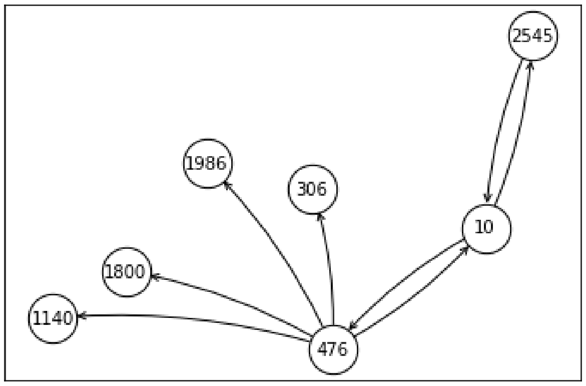

Fig. 1. Subgraph

As an example above in Figure 1, this is the subgraph for index 10. The subgraph has 7 nodes and 8 edges. If we run the below command we can see the the shapes input matrix, number of nodes and edges, node label and our prediction. In this case, node_mask has 7 nodes and 1433 features. The target label in this case is 7, which was predicted correctly.

```
explanation.get_explanation_subgraph()
```

For features importance, the top 10 important features are shown in below Figure 2. The xaxis shows the feature labels and yaxis shows the importance score. In this case, feature 1263 is the most importance feature for node classification.

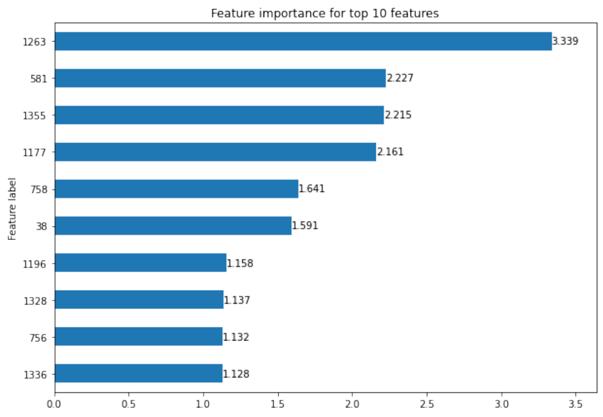

Fig. 2. Features Importance

5 Conclusion

In this report, we presented an analysis of applying GNNs to the Cora dataset. We trained a GNN model on the dataset and evaluated its performance in predicting the research field of academic papers. Additionally, we provided explanations for the model's predictions using the GNNExplainer algorithm.

The results demonstrated the effectiveness of GNNs in accurately predicting the research field of papers in the Cora dataset. The trained GNN model achieved competitive performance compared to other models trained on different hyperparameters. The GNNExplainer algorithm provided insights into the important features and relationships considered by the model for its predictions.

The findings of this report highlight the potential of GNNs in analyzing and making predictions on graph-structured data. GNNs offer a powerful approach for capturing complex relationships and patterns in graph data, making them valuable in various domains and applications.

Further research can focus on exploring different GNN architectures, incorporating additional graph-level information, and applying GNNs to larger and more diverse datasets. Continued advancements in GNNs will contribute to improving their interpretability, scalability, and generalizability, further unlocking their potential in real-world applications.

References

1. Algorithm.GNNExplainer - PyTorch Geometric Documentation. `https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.explain.algorithm.GNNExplainer.html`
2. GNN Explainability — PyTorch Geometric Documentation. `https://pytorch-geometric.readthedocs.io/en/latest/tutorial/explain.html`
3. pytorch_geometric/gnn_explainer.py at master - GitHub. `https://github.com/pyg-team/pytorch_geometric/blob/master/torch_geometric/explain/algorithm/gnn_explainer.py`
4. GitHub - OpenXAIProject/GNNExplainer-Tutorial. `https://github.com/OpenXAIProject/GNNExplainer-Tutorial`
5. Graph Machine Learning Explainability with PyG — by PyTorch Geometric. `https://medium.com/@pytorch_geometric/graph-machine-learning-explainability-with-pyg-ff13cffc23c2`
6. Yang, Zhilin, William Cohen, and Ruslan Salakhudinov. "Revisiting semi-supervised learning with graph embeddings." International conference on machine learning. PMLR, 2016