PGExplainer

1. Introduction

PGExplainer was first introduced in august 2020 in a scientific paper *"Parameterized Explainer for Graph Neural Network"*. Despite advancements in Graph Neural Networks (GNNs), explaining their predictions remains a complex challenge. Current methods focus on local explanations for individual instances, hindering global understanding and limiting generalizability. To address this, in the paper they introduced PGExplainer, a parameterized explainer for GNNs. Unlike existing approaches, PGExplainer leverages a deep neural network to collectively explain multiple instances, improving generalization and facilitating inductive settings. Experiments on synthetic and real datasets demonstrate its superior performance, achieving up to a 24.7% relative improvement in AUC for explaining graph classification over leading baselines.

2. Simple explanation

PGExplainer is a tool designed for explaining graph structures in Graph Neural Networks (GNNs). Unlike GNNExplainer, which explains both structure and features, PGExplainer focuses solely on graph structures. This is because explaining features in GNNs is similar to non-graph neural networks and has been extensively studied. PGExplainer is versatile, suitable for interpreting various types of GNNs.

3. Formulation[1]

To explain predictions made by a Graph Neural Network (GNN) model on real-life graphs with underlying structures, we decompose the original input graph G_0 into two components: G_s, representing the essential subgraph contributing significantly to GNN predictions, and ΔG, containing task-irrelevant edges. Following a methodology inspired by previous work, PGExplainer seeks G_s by maximizing the mutual information between GNN predictions (Y_o) and the underlying structure Gs.

$$\max_{G_s} \mathrm{MI}(Y_o, G_s) = H(Y_o) - H(Y_o | G = G_s),$$

This is formulated as a maximization problem where G_s is chosen to minimize the conditional entropy . However, directly optimizing this objective is impractical due to the exponential number of potential explanatory graphs. To address this, we introduce a relaxation, assuming G_s follows a Gilbert random graph model. This allows us to reformulate the objective as minimizing the conditional entropy under the distribution of the explanatory graph, parameterized by θ's. The optimization is approximated by sampling Gs from this distribution, denoted as q(Θ). The objective is to find the optimal Gs that best explains the GNN predictions while considering the conditional independence of selected edges in Gs.

4. Example

- **CORA Dataset (Multi-Class Classification):**

The model is a 2-layer Graph Convolutional Network (GCN) trained on the CORA dataset for node classification. PG Explainer is applied to elucidate the decision-making process of the GNN. The training loop includes optimizing the model's parameters and computing the loss for better prediction accuracy.

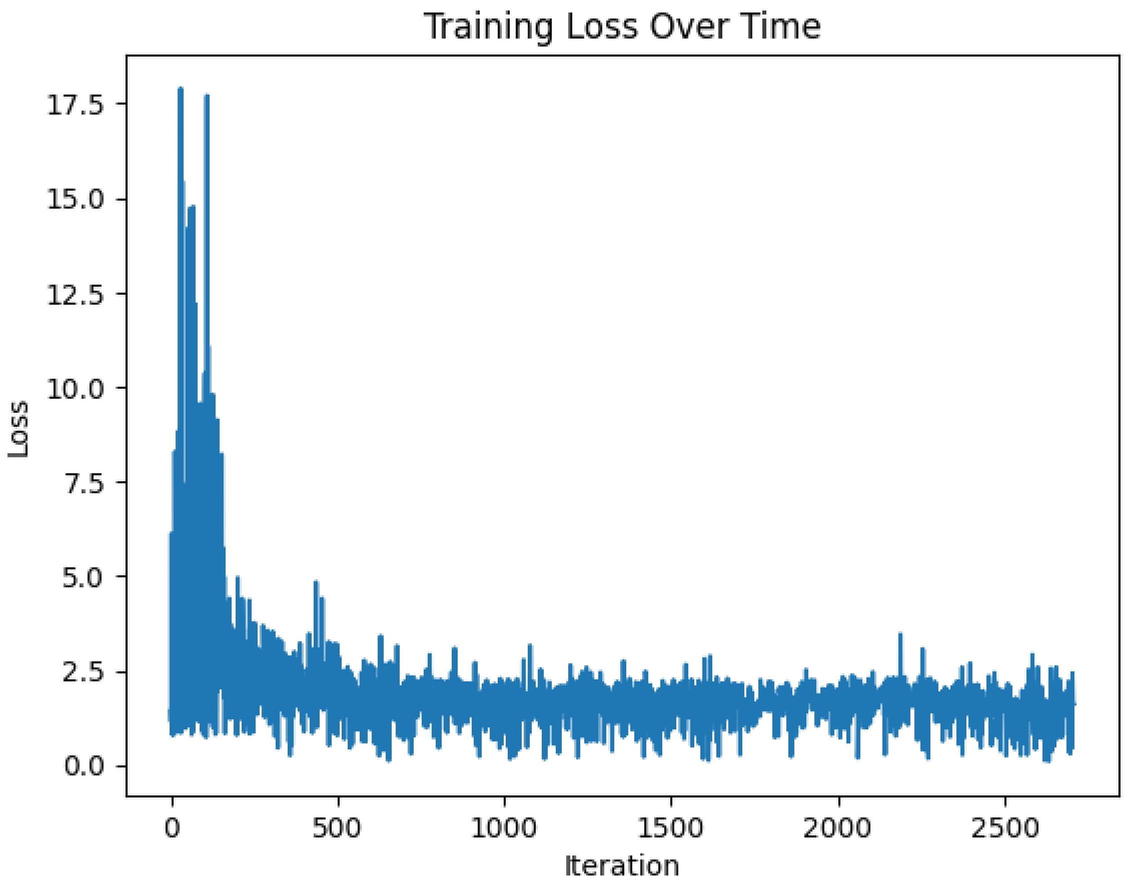

Fig 1. Loss parameter of the explainer during its training on CORA.

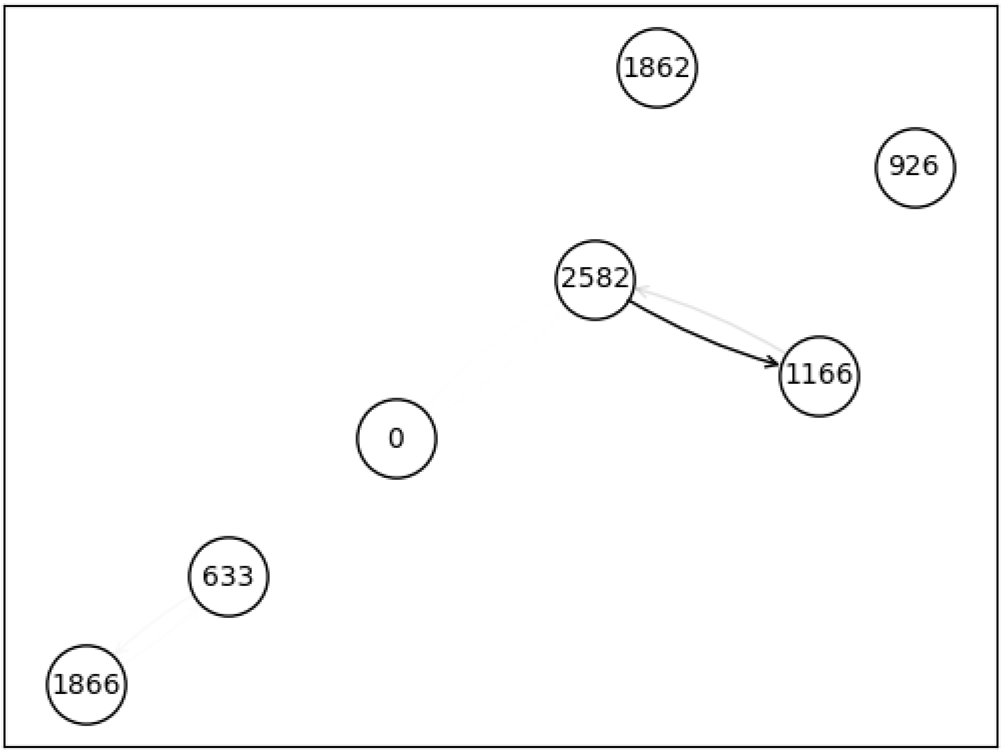

Fig 2. The output of the explainer in case of the prediction for node 0.

- **Minesweeper Dataset (Binary Classification):**

The model is a GCN designed for binary classification on the Minesweeper dataset, where nodes represent cells in a Minesweeper grid.
PG Explainer is employed to interpret the predictions by attributing importance to edges in the graph. The training loop involves optimizing the binary classification model with a focus on explaining individual node predictions.

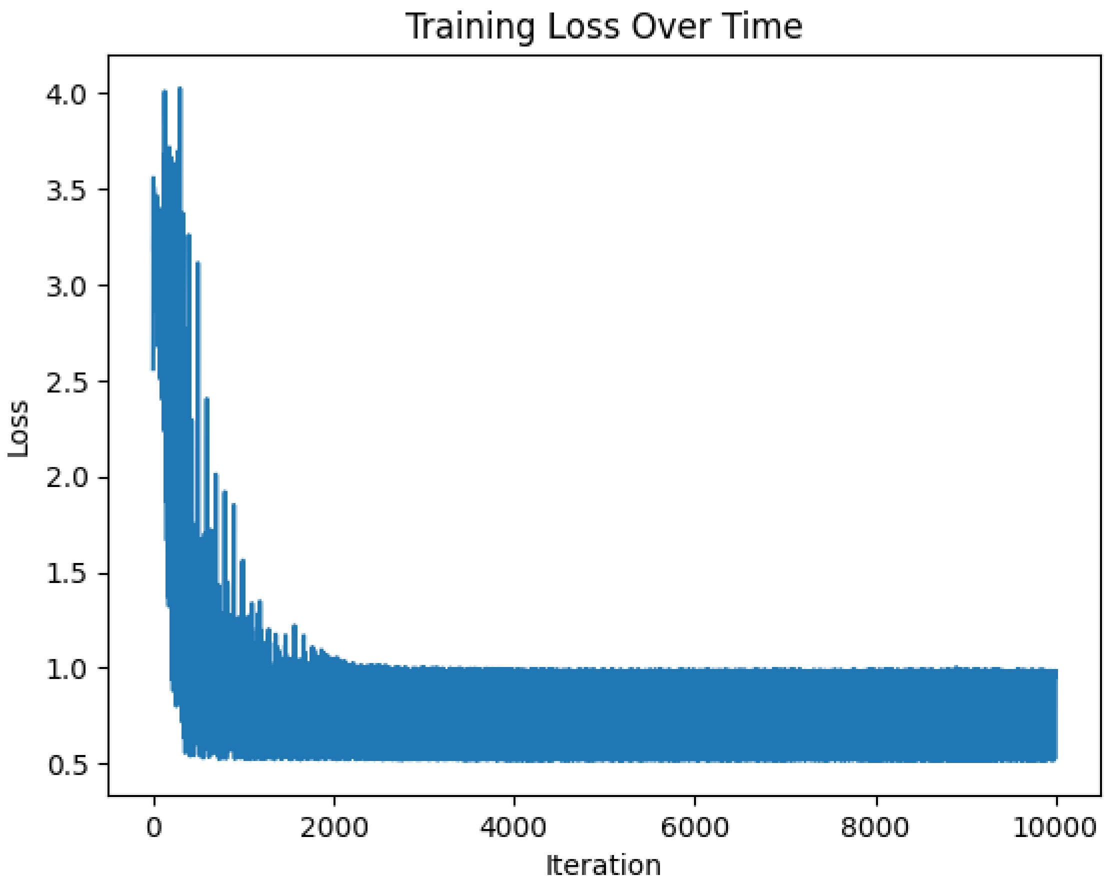

Fig 3. Loss parameter of the explainer during its training on minesweeper.

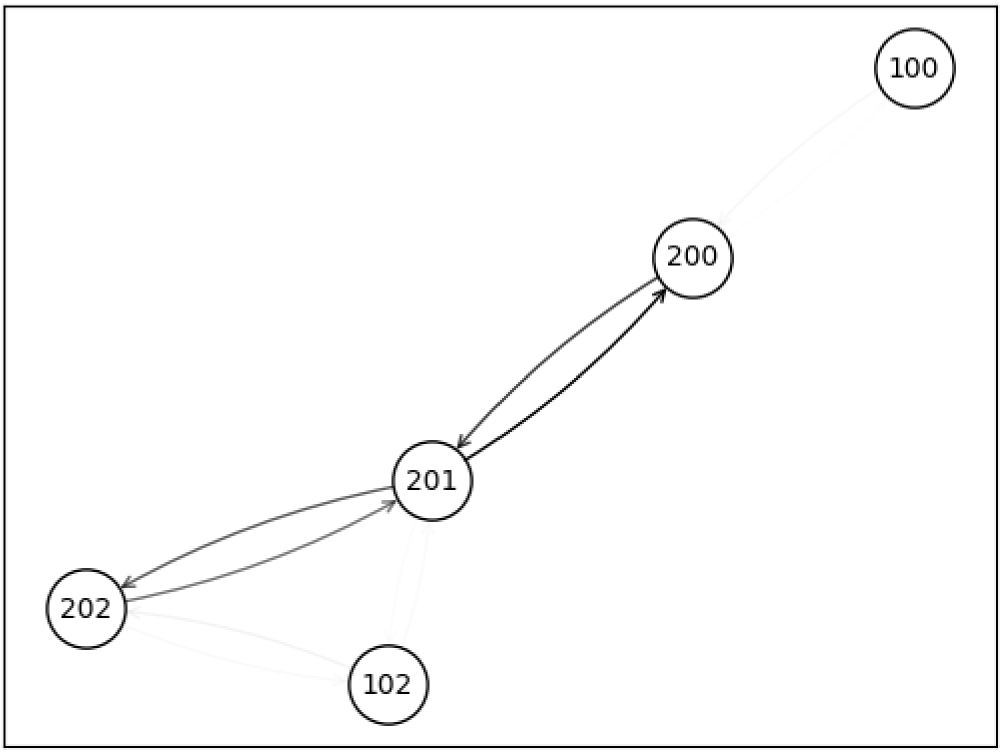

Fig 2. The output of the explainer in case of the prediction for node 0.

5. References

- [2011.04573] Parameterized Explainer for Graph Neural Network
- torch_geometric.explain.algorithm.PGExplainer — pytorch_geometric documentation