Explainability for GNN

Graph Neural Networks

Salah Eddine Azekour ___ Leopold Stevens

CentraleSupelec

Graph Neural Network

- Increasingly popular to represent real-world data: social networks, chemical molecules, financial data
- Different purposes of GNN: node classification, graph classification, link prediction
- Many complex GNN operations: graph conv, graph attention, graph pooling
- Explainability of graph models is less explored compared to image and text domains.

Challenges

- Graphs are not grid-like data
- Adjacency matrices in graphs contain discrete values
- Graph nodes and edges contribute together to the final predictions of GNNs
- Important to study structural information of graph
- Graph data are less intuitive than images and texts
- Need for standard datasets and evaluation metrics for explanation tasks

XAI Approaches

Instance Level Approaches:

- Gradients/features
- Perturbations
- Decomposition
- Surrogate

Model Level Approach:

- Generation

Instance level: Gradient/features

- Straightforward solution for explaining GNN models.
- Gradients or hidden feature maps are used as approximations of input importance.
- Larger gradients or feature values indicate higher importance.

SA

- Computes importance scores based on the squared values of gradients.
- Advantages: Simple and efficient.
- Disadvantages: Only reflects sensitivity between input and output, cannot accurately show importance; suffers from saturation problems.

Guided BP

- Back-propagates only positive gradients and clips negative gradients to zero to compute importance scores.
- Advantages: Eliminates negative gradients and their potential confusing explanations.
- Disadvantages: same limitations as SA.

CAM

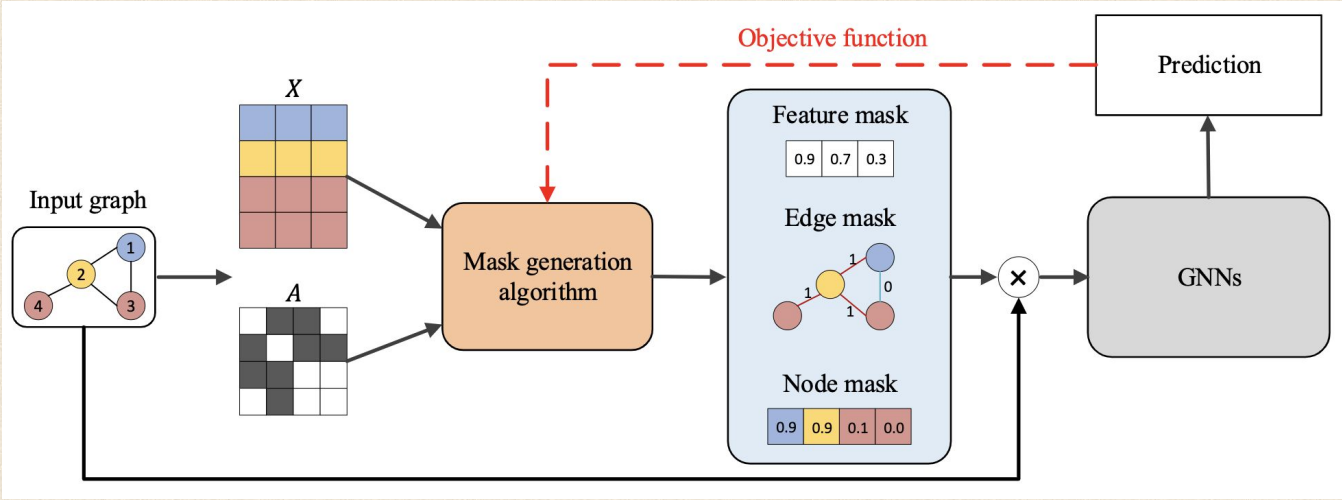
- Maps node features to input space and combines feature maps to obtain importance scores for input nodes.
- Advantages: Simple and directly identifies important nodes.
- Disadvantages: Requires specific GNN structure; cannot explain node classification tasks.

Grad-Cam

- Computes importance scores by using gradients to combine different feature maps.
- Advantages: Extends CAM to general graph classification models
- Disadvantages: Cannot explain node classification tasks

Instance level: Perturbation

- Capture important input features by studying output variations with different input perturbations
- Generate masks for node, edge, and features to indicate important input features
- Soft masks have continuous values and can be updated by back-propagation, but suffer from the "introduced evidence" problem (i.e., non-zero or non-one values may introduce noise to the input graph)
- Discrete masks only have values of 0 or 1, but involve non-differentiable operations such as sampling



Instance level: Perturbation

GNNExplainer

- Learns soft masks for edges and node features to explain the predictions via mask optimization
- Advantage: Can optimize masks for each input graph individually
- Disadvantage: Obtained masks are soft masks, which suffer from the "introduced evidence" problem

PGExplainer

- Learns approximated discrete masks for edges to explain the predictions
- Advantage: Obtained masks can solve the "introduced evidence" problem and provide a global understanding of the trained GNNs
- Disadvantage: Involves non-differentiable operations (sampling)

GraphMask

- Trains a classifier to predict whether an edge can be dropped without affecting the original predictions
- Advantage: same as PGExplainer
- Disadvantage: Dropped edges are replaced by learnable baseline connections, which may change graph structures

ZORRO

- Uses discrete masks to identify important input nodes and node features using a greedy algorithm
- Advantage: Using hard masks avoids the "introduced evidence" problem
- Disadvantage: Greedy mask selection algorithm may lead to local optimal explanations

Causal Screening

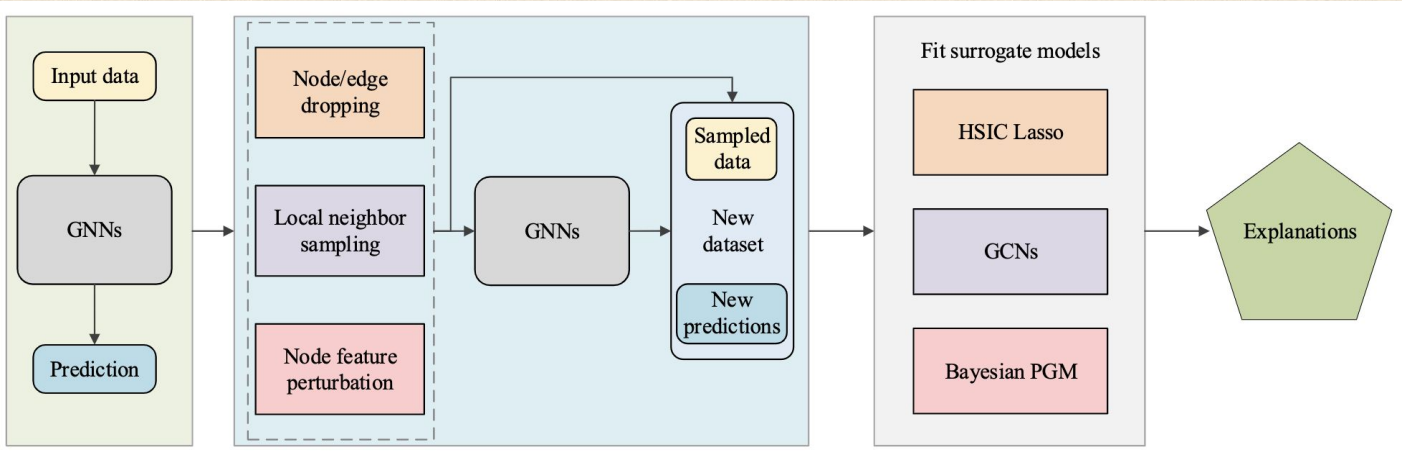
- Identifies edge mask based on causal attribution of edges in input graph.
- Advantage: Same as Zorro
- Disadvantage: May lack a global understanding and stuck in local optimal explanations

SubGraphX

- Explores subgraph-level explanations using Monte Carlo Tree Search
- Advantage: Obtained subgraphs are more human-intelligible and suitable for graph data
- Disadvantage: Computational cost is more expensive since it needs to explore different subgraphs.

Instance level: Surrogate

- Simple and interpretable models to approximate complex deep models
- The relationships in neighboring areas of the input are less complex and can be well captured by simpler models.
- General pipeline:
  - Obtain a local dataset
  - Fitting an interpretable model
  - Use the explanations from the model as explanations for the original model



Instance level: Surrogate

Graph Lime

- Considers neighboring nodes and their predictions as its local dataset and fitting it to a kernel-based feature selection algorithm
- Advantage: Provides explanations for node features.
- Disadvantage: Ignores graph structures, such as nodes and edges, which are important for graph data.

Rel Ex

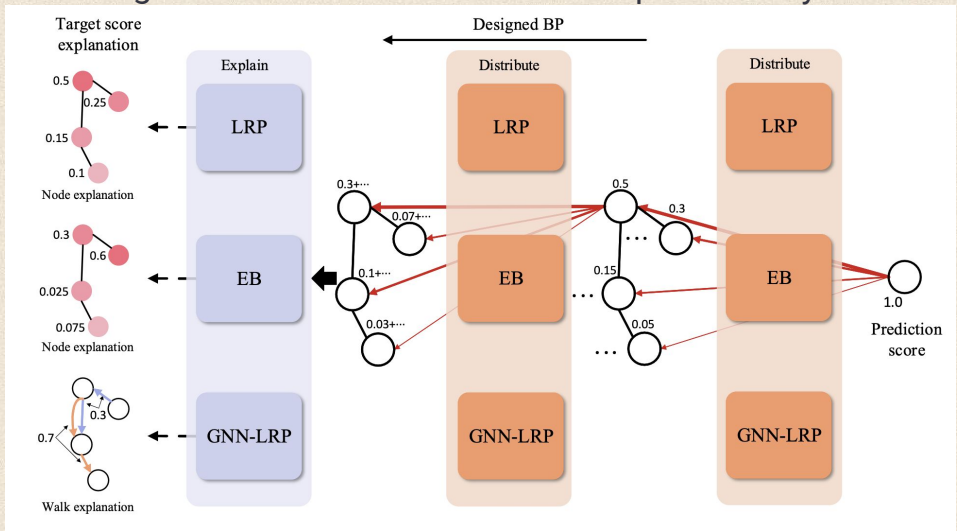
- Randomly samples connected subgraphs from computational graph, feeds them to trained GNNs, and uses a GCN model as surrogate model to fit local datasets.
- Advantage: Can provide explanations regarding important nodes.
- Disadvantage: Contains multiple steps of approximations, making the explanations less convincing and trustable.

PGM Explainer

- Randomly perturbs node features of random nodes in computational graph, records perturbation influences on GNN predictions
- Advantage: Can be used to explain both node classification and graph classification tasks.
- Disadvantage: Ignores graph edges, which contain important graph topology information.

Instance level: Decomposition

- Measure the importance of input features by decomposing the original model predictions into several terms.
- Study the model parameters to reveal the relationships between the features in the input space and the output predictions.
- General pipeline:
  - Backpropagate the prediction score layer by layer
  - Treat the model's prediction as the initial target score
  - Decompose and distributing the score to the neurons in the previous layer following the decomposition rules.



Instance level: Decomposition

LRP

- Decomposes output prediction score to different node importance scores based on hidden features and weights.
- Advantages: Trustable explanation results due to direct development based on model parameters.
- Disadvantages: Cannot study importance of graph structures, requires comprehensive understanding of model structures, limiting its application for non-expert users.

Excitation BP

- Defines probability of a neuron in current layer as equal to the total probabilities it outputs to all connected neurons in next layer.
- Advantages: Shares same advantages as LRP algorithm.
- Disadvantages: Shares same limitations as LRP algorithm.

GNN LRP

- Studies importance of different graph walks by providing view of high-order Taylor decomposition to develop score decomposition rule.
- Advantages: Solid theoretical background.
- Disadvantages: Approximations in computations may not be accurate, high computational complexity, challenging for non-experts to use

Model level: Generation

- Generate synthetic input-output pairs to represent the behavior of the original model.
- Capture the behavior of the original model across the entire input space, rather than just a local region.
- Generating high-quality data can be challenging and requires a good understanding of the original model's behavior

XGNN

- Train a graph generator via reinforcement learning to maximize a target graph prediction
- The generator predicts how to add an edge to the current graph at each step.

- Advantages: provides global understanding of the trained GNNs, and explanations are general.

Disadvantages: only demonstrated effectiveness in explaining graph classification models

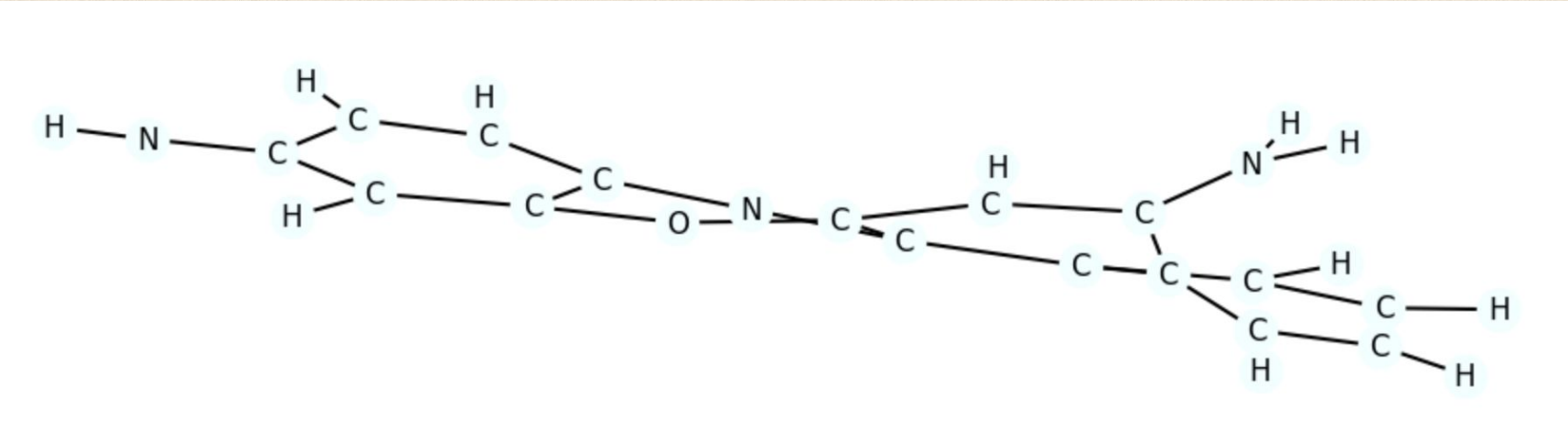
Evaluation metrics

- **Fidelity+** studies the change of prediction accuracy after masking out important input features.
- **Fidelity-** studies the prediction change by keeping important input features and removing unimportant features.
- **Sparsity** measures the fraction of features selected as important by explanation methods.
- **Stability** measures whether an explanation method is stable by comparing the difference between the original and perturbed graph.
- **Accuracy** used for synthetic datasets by comparing the explanations with ground truths.
- Different metrics should be combined to evaluate explanation results.

Dataset:

- The Mutagenicity dataset is a binary classification problem where the task is to predict whether a given chemical compound is mutagenic or non-mutagenic.
- The dataset contains 4,045 chemical compounds, where each compound is represented by 1,027 binary features. These features represent the presence or absence of certain substructures or fragments in the compound.
- The positive class contains 1,058 compounds and the negative class contains 2,987 compounds, resulting in a class imbalance.
- The Mutagenicity dataset is commonly used in the field of molecular biology to study the relationship between the structure of chemical compounds and their mutagenic potential.
- The Mutagenicity dataset is widely used as a benchmark dataset for evaluating the performance of machine learning models for predicting mutagenicity. Many studies have reported the results of their models on this dataset, making it a well-established benchmark.

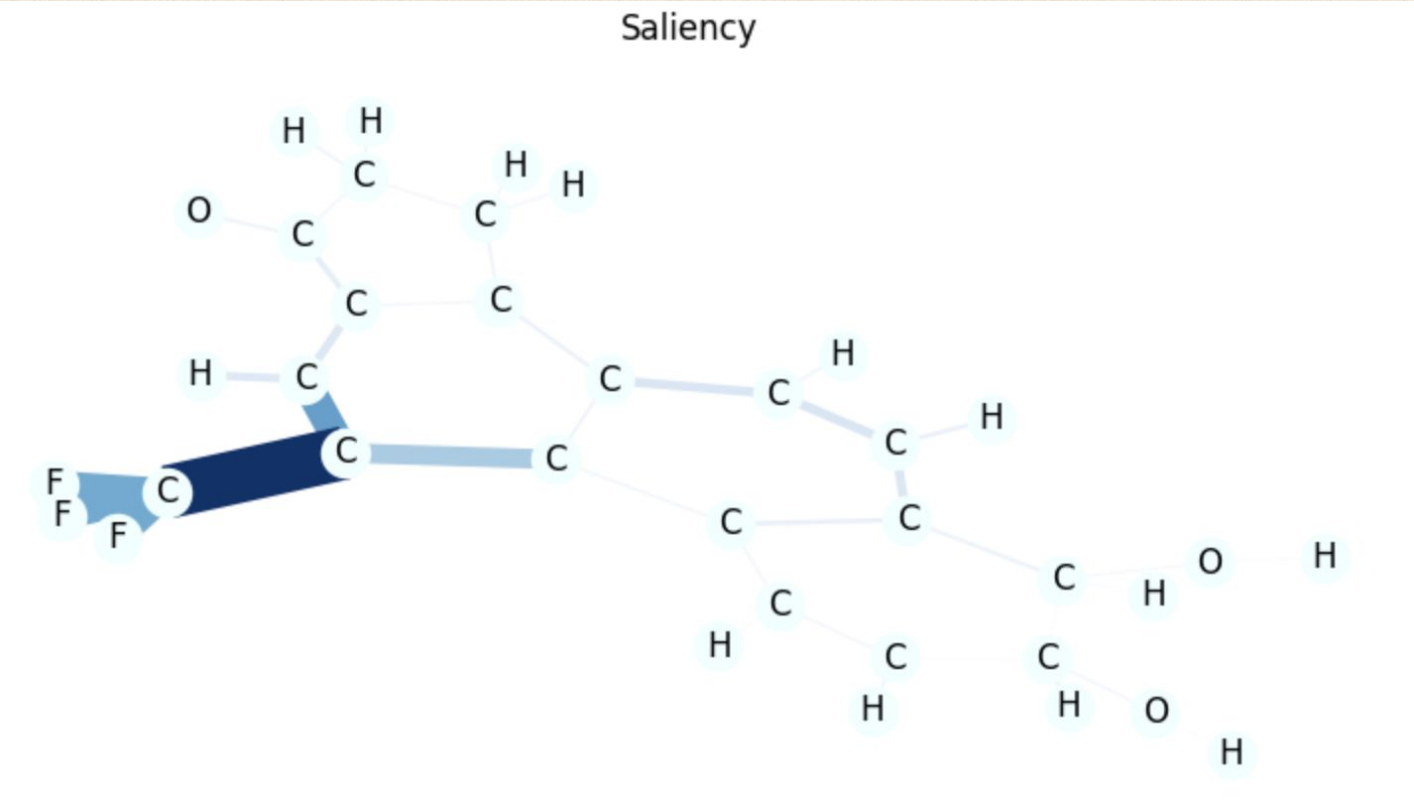
Dataset: example



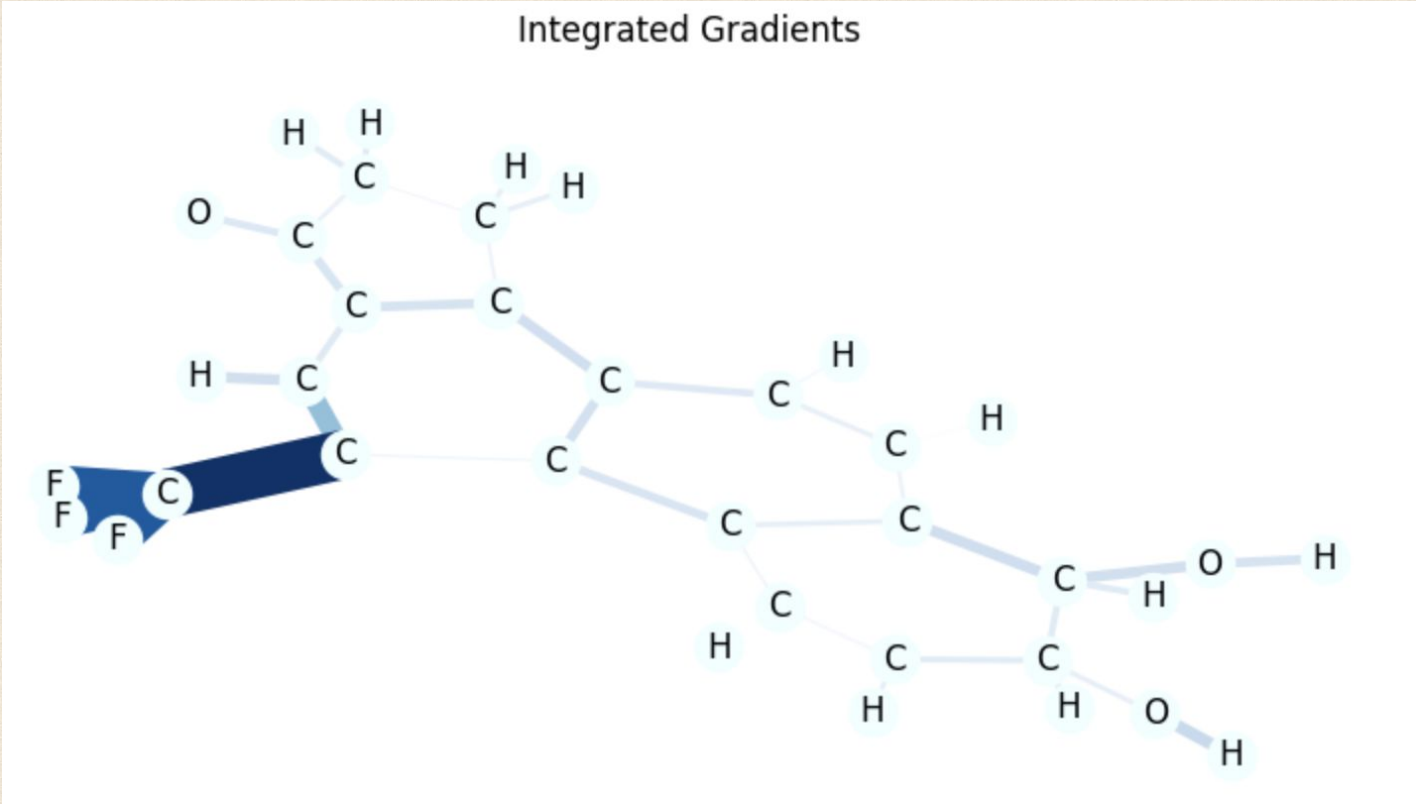
Model Architecture

- This model consists of 5 graph convolutional neural network (GCN) layers followed by 2 linear layers and a log softmax activation function for binary classification
- A ReLU activation is applied after each GCN layer
- After the 5 GCN layers global pooling is applied to aggregate the node features into a single vector for each graph.
- Dropout is used to apply dropout regularization to the output of the first linear layer with a dropout probability of 0.5.
- The negative log likelihood is used as a loss function for both training and testing
- Overall it takes in a graph as input and produces a probability distribution over the classes of the input graph.

Saliency Method



Integrated Gradients



Comparison

- Integrated gradients tend to be more interpretable than saliency methods since they provide a clearer visualization of the contribution of each feature to the model's output.
- Saliency methods may be more sensitive in some cases since they can identify features that have a small but significant impact on the model's output.

Any Questions ?