Classifying and Understanding Financial Data Using Graph Neural Network

Xiaoxiao Li[1*] **Joao Saude** [2] **Prashant Reddy** [2] **Manuela Veloso**[2]
[1]Yale University
[2]J.P.Morgan AI Research

Abstract

Real data collected from different applications usually do not have a pre-defined data model or is not organized in a pre-defined manner. Analyzing messy, unstructured data and extracting useful data information is difficult. For the data collected in financial institutions, usually, it has additional topological structures and is amenable to be represented as a graph. For example, social networks, communication networks, financial systems, and payments networks. The graph structure can be built from the connection of each entity, such as financial institutions, customers, or computing centers. We consider how the different entity influences each other's label through a label prediction problem based on the graph structure. Given the structured data, Graph Neural Networks (GNNs) is a powerful tool that can mimic experts' decision on node labeling. GNNs combine node features through graph structure by using a neural network to embed node information and pass it through edges in the graph. We want to identify the informative interaction in the input data used by the GNN model to classify the node in the graph and examine if the model works as we desire. However, due to the complex data representation and non-linear transformations, explaining decisions made by GNNs is challenging. In this work, we propose graph representation methods for finical transaction data and new graph features' explanation methods to identify the informative graph topology. We use four datasets (one synthetic and three reals) to validate our methods. Our results demonstrate that graph-structured representation help to analyze financial transaction data, and our explanation approach can mimic patterns in human interpretation and disentangle different features in the graphs.

1 Introduction

In recent years, with the rapid development of new technologies such as big data, cloud computing, and artificial intelligence, these new technologies are deeply integrated with financial services, releasing financial innovation vitality and application potential, which has greatly promoted the financial industry. In this development process, big data technology is the most mature and widely used. However, faced with such a vast information ocean, especially unstructured data information, how to store, query, analyze, mine, and utilize these massive information resources is particularly critical. Traditional relational databases are mainly oriented to transaction processing and data analysis applications. They are good at solving structured data management problems. There are some inherent shortcomings in managing unstructured data, especially when dealing with massive unstructured information. In response to the challenges of unstructured data analysis, one strategy is transforming unstructured data to structured data, which will help similar information labeling, retrieving, searching, and clustering, and it will help finance better serve the real economy and effectively promote the overall development of the financial industry.

Our contemporary society relies heavily in interpersonal/cultural relations (social networks), our economy is densely connected and structured (commercial relations, financial transfers, supply/distribution chains), the geopolitical relations are also very structured (commercial and political unions), we also rely on transportation networks (roads, railroads, maritime and flight connections), and our cybersystems are also structurally connected (computers networks, internet). Moreover, those complex network structures also appear in nature, on biological systems, like the brain, vascular and nervous systems, and also on chemical systems, for instances atoms connections on molecules. Nowadays, with modern technologies we collect data from all the above systems and their relations since this data is hugely structured and depends heavily on the relations within the networks, it makes sense to represent the data as a graph, where nodes represent entities and the edges the connections between them.

Artificial intelligence is becoming a new direction for financial big data applications. Graph Neural Networks (GNNs) such as GCN (Kipf and Welling 2016), GraphSage (Hamilton, Ying, and Leskovec 2017), is a kind of deep learning architectures that can handle graph-structured data by preserving the information structure of graphs. Our primary focus is the node labeling problem, such as fraud detection, credit issuing, customer targeting, social network user classification, which can mimic experts' decisions on

[*]This work was done at J.P.Morgan AI Research

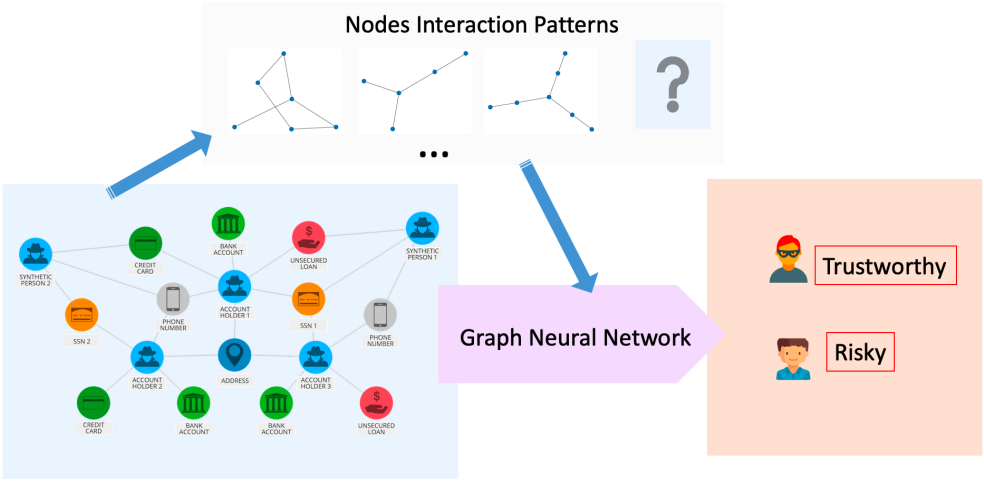

Figure 1: We can build graph structure for transaction system and try to understand what interaction patterns are informative for fraud detection using GNN and explaination methods.

node labeling. GNNs are able to combine node features, connection patterns, and graph structure by using a neural network to embed the node information and pass it through edges in the graph. However, due to the complex data representation and non-linear transformations performed on the data, explaining decisions made by GNNs is a challenging problem. One example about understanding fraud detection from explaining GNN node classification decision is shown in Figure. 1. Therefore we want to identify the patterns in the input data that were used by the GNN model to make a decision and examine if the model works as we desire.

Although deep learning model visualization techniques have been developed in convolution neural network (CNN), those methods are not directly applicable to explain weighted graphs with node features for the classification task. A few work have been down on explaining GNN ((Pope et al. 2019; Baldassarre and Azizpour 2019; Ying et al. 2019; Yang et al. 2019)). However, to our best knowledge, no work has been done on explaining comprehensive features (namely node feature, edge feature, and connecting patterns) in weighted graph, especially for node classification problem. Here we propose a few graph feature explanation methods to formulate financial data in based on graph structure. We use three datasets (one synthetic and two real data) to validate our methods. Our results demonstrate that by using explanation approach, we can discover the data patterns used for node classification correspond to human interpretation and those explanation methods can be used for understanding data, debugging GNN model and examine model decisions, and in other tasks.

Our contribution is summarized as follows:

1. We propose to transfer financial transaction data to weighted graph representation for further analysis and data information understanding.
2. We propose to use GNN to analyze financial transaction data, including fraud detection and account matching.
3. We provide the tools to interpret informative interactions between entities for entity labeling in the structured-graph format.

Paper structure: In section 2, we introduce Graph representation. Then in section 3, we walk through the operations in GNN. In section 4, the formula of graph explanation is described, and the corresponding methods are introduced. In section 5, we propose the evaluation metrics and methods. The experiments and results are presented in section 6. We conclude the paper in section 7.

2 Data Representation – Weighted Graph

For the financial data, which contains user and interaction information, we can model each entity in the as a node and build the underlying connection between them using based on their interaction. In this section, we introduce the necessary notation and definitions. We denote a graph by $G = (V, \mathcal{E})$ where V is the set of nodes, $\mathcal{E}$ the set of edges linking the nodes and X the set of nodes' features. For every pair of connected nodes, $u, v \in V$, we denote by $e_{vu} \in \mathbb{R}$ the weight of the edge $(v, u) \in \mathcal{E}$ linking them. We denote $E[v, u] = e_{vu}$, where $E \in \mathbb{R}^{|\mathcal{E}|}$. For each node, u, we associate a d-dimensional vector of features, $X_u \in \mathbb{R}^d$ and denote the set of all features as $X = \{X_u : u \in V\} \in (\mathbb{R}^d)^{|V|}$.

Edge features contain important information about graphs. For instances, the graph G may represent a banking system, where the nodes V represents different banks, and the edges E are the transaction between them; or graph G may represent a social network, where the nodes V represent different users, and the edges E is the contacting frequencies between the users. We consider a node classification task, where each node u is assigned a label $y_u \in I_C = \{0, \ldots, C-1\}$. In the financial application, the node classification problem can be fraud detection, new costumer discovery, account matching, and so on.

3 GNN Utilizing Edge Weight

Different from the state of art GNN architecture, *i.e.* graph convolution networks (GCN) (Kipf and Welling 2016) and graph attention networks (GAT) (Veličković and others 2018), some GNNs can exploit the edge information on graph (Gong and Cheng 2019; Shang et al. 2018; Yang et al. 2019). Here, we consider weighted and directed graphs, and develop the graph neural network that uses both nodes and edges weights, where edge weights affect message aggregation. Not only our approach can handle directed and weighted graphs but also preserves edge information in the propagation of GNNs. Preserving and using edges information is important in many real-world graphs such as banking payment network, recommendation systems (that use social network), and other systems that heavily rely on the topology of the connections. Since, apart from node (atomic) features also attributes of edges (bonds) are important for predicting local and global properties of graphs. Generally speaking, GNNs inductively learn a node representation by recursively aggregating and transforming the feature vectors of its neighboring nodes. Following (Battaglia et al. 2018; Zhang, Cui, and Zhu 2018; Zhou et al. 2018), a per-layer update of the GNN in our setting involves these three computations, message passing Eq. (1), message aggregation Eq. (2), and updating node representation Eq. (3), which can be expressed as:

$$\mathbf{m}_{vu}^{(l)} = \text{MSG}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}, e_{vu}) \quad (1)$$

$$\mathbf{M}_i^{(l)} = \text{AGG}(\{\mathbf{m}_{vu}^{(l)}, e_{vu}\} \mid v \in \mathcal{N}(u)\}) \quad (2)$$

$$\mathbf{h}_u^{(l)} = \text{UPDATE}(M_u^{(l)}, \mathbf{h}_u^{(l-1)}) \quad (3)$$

where $\mathbf{h}_u^{(l)}$ is the embedded representation of node u on the layer l; e_{vu} is the weighted edge pointing from v to u; $\mathcal{N}(u)$ is u's neighborhood from where it collects information to update its aggregated message M_i. Specifically, $\mathbf{h}_u^{(0)} = \mathbf{x}_u$ as initial, and $\mathbf{h}_u^{(L)}$ is the final embedding for node u of an L-layer GNN node classifier.

Here, following (Schlichtkrull et al. 2018), the GNN layer using edge weight for filtering can be formed as the following steps:

$$\mathbf{m}_{vu}^{(l)} = W_1^{(l-1)}\mathbf{h}_v^{(l-1)} \quad \text{(message)} \quad (4)$$

$$\mathbf{M}_u^{(l)} = \sum_{v \in \mathcal{N}(u)} g(\mathbf{m}_{vu}^{(l)}, \mathbf{h}_u^{(l-1)}, e_{vu}) \quad \text{(aggregate)} \quad (5)$$

$$\mathbf{h}_u^{(l)} = \sigma(W_0^{(l-1)}\mathbf{h}_u^{(l-1)} + \mathbf{M}_u^{(l)}) \quad \text{(update)} \quad (6)$$

where $\mathcal{N}(u)$ denotes the set of neighbors of node u and e_{vu} denotes the directed edge from v to u, W denotes the model's parameters to be learned, and ϕ is any linear/nonlinear function that can be applied on neighbour nodes' feature embedding. We set $h^{(l)} \in \mathbb{R}^{d^{(l)}}$ and $d^{(l)}$ is the dimension of the l^{th} layer representation.

As the graph convolution operations in (Gong and Cheng 2019), the edge feature matrices will be used as filters to multiply the node feature matrix. To avoid increasing the scale of output features by multiplication, the edge features need to be normalized, as in GAT (Veličković and others 2018) and GCN (Kipf and Welling 2016). Due to the aggregation mechanism, we normalize the weights by in-degree $\bar{e}_{vu} = e_{vu} / \sum_{v \in \mathcal{N}(u)} e_{vu}$. Our method can deal with negative edges-weighted by re-normalizing them to a positive interval, for instances $[0, 1]$, therefore in the following we use only positive weighted edges and the edge weights are used as message filtering ratio. Depending on the the problem:

- g can simply defined as: $g = \bar{e}_{vu}\mathbf{m}_{vu}^{(l)}$; or
- g can be a gate function, such as a rnn-type block of $\mathbf{m}_{vu}^{(l)}$, i.e. $g = GRU(\bar{e}_{vu}\mathbf{m}_{vu}^{(l)}, \mathbf{h}_u^{(l-1)})$.

4 Explaining Informative Component of Graph Structures Data

Relational structures in graphs often contain crucial information for node classification, such as graph's topology and information flow (i.e., direction and amplitude). Therefore, knowing which edges contribute the most to the information flow towards or from a node is essential to understand and interpret the node classification evidence.

We tackle the weighted graph feature explanation problem as a two-stage pipeline. First, we train a node classification function, in this case, a GNN. The GNN inputs are a graph $G = (V, \mathcal{E})$, its associated nodes' features, X, and its true nodes' labels Y. We represent this classifier as $\Phi : G \mapsto (u \mapsto y_u)$, where $y_u \in I_C$. One advantage of GNNs is the preservation of the information flow across nodes as well as the data structure. Furthermore, it is invariant to permutations on the ordering. Hence it keeps the relational inductive biases of the input data (see (Battaglia et al. 2018)). Second, given the node classification model and node's true label, the explanation part provides a subgraph and a subset of features retrieved from the k-hop neighborhood of each node u, for $k \in \mathbb{N}$ and $u \in V$. The subgraph, along with the subset of features, is the minimal set of information and information flow across neighbor nodes of u, that the GNN uses to compute the node's label. We define $G_S = (V_S, \mathcal{E}_S)$ to be a subgraph of G, where $G_S \subseteq G$, if $V_S \subseteq V$ and $\mathcal{E}_S \subseteq \mathcal{E}$. Consider the classification $y_u \in I_C$ of node u, then **Informative Components Detection** computes a subgraph, G_S, containing u, that aims to explain the classification task by looking at the edge connectivity patterns $\mathcal{E}_S$ and their connecting nodes V_S. This provides insights on the characteristics of the graph that contribute to the node's label.

4.1 Maximal Mutual Information (MMI) Mask

Due to the properties of the GNN, (5), we only need to consider the graph structure used in aggregation, *i.e.*, the *computational graph* w.r.t. node u is defined as $G_c(u)$ containing N' nodes, where $N' \leq N$. The node feature set associated with the $G_c(u)$ is $X_c(u) = \{x_v | v \in V_c(u)\}$. For node u, the label prediction of GNN Φ is given by $\hat{y}_u = \Phi(G_c(u), X_c(u))$, which can be interpreted as a distribution $P_\Phi(Y|G_c, X_c)$ mapping by GNN. Our goal is to identity a subgraph $G_S \subseteq G_c(u)$ (and its associated features $X_S = \{\mathbf{x}_w | w \in V_S\}$, or a subset of them) which the GNN uses to predict u's label.

Using ideas from Information theory (Cover and Thomas 2012) and following GNNExplainer (Ying et al. 2019), the informative explainable subgraph and nodes features subset are chosen to maximize the mutual information (MI):

$$\max_{G_S} I(Y,(G_S,X_S)) = H(Y|G,X) - H(Y|G_S,X_S). \quad (7)$$

Since the trained GNN node classifier Φ is fixed, the $H(Y)$ term of Eq.(7) is constant. As a result, it is equivalent to minimize the conditional entropy $H(Y|G_S,X_S)$:

$$-\mathbb{E}_{Y|G_S,X_S}[\log P_\Phi(Y|G_S,X_S)]. \quad (8)$$

Therefore, the explanation to the graph components with prediction power w.r.t node u's prediction $\hat{y}_u$ is a subgraph G_S and its associated feature set X_S, that minimize (8). Thus, the objective of the explanation is to pick the top informative edges and its connecting neighbours, which form a subgraph, for predicting u's label. Because, some edges in u's computational graph $G_c(u)$ might form important message-passing (5) pathways, which allow useful node information to be propagated across $G_c(u)$ and aggregated at u for prediction; while some edges in $G_c(u)$ might not be informative for prediction. Instead of directly optimize G_S in (8), since there are exponentially many discrete structures $G_S \subseteq G_c(u)$ containing N' nodes, the GNNExplainer (Ying et al. 2019) optimizes a mask $\mathcal{M}_{sym}^{N'\times N'}[0,1]$ on the binary adjacent matrix, which allows gradient descent to be performed on G_S.

If we use the edge weights for node embedding, the connection can be treated as binary and fit into the original GNNExplainer. However, if we use edge weights as filtering, the mask should affect filtering and normalization. We extend the original GNNExplainer method by considering edge weights and improving the method by adding extra regularization. Unlike GNNExplainer, where there are no constraints on the mask value, we add constraints to the value learned by the mask

$$\sum_w \mathcal{M}_{vw} e_{vw} = 1, \quad \mathcal{M}_{vw} \geq 0, \text{ for } (v,w) \in \mathcal{E}_c(u), \quad (9)$$

and perform a projected gradient decent optimization. Therefore, rather than optimizing a relaxed adjacency matrix in GNNExplainer, we optimize a mask $\mathcal{M} \in [0,1]^Q$ on weighted edges, supposing there are Q edges in $G_c(u)$. Then $E_c^{\mathcal{M}} = E_c \odot \mathcal{M}$, where $\odot$ is element-wise multiplication of two matrix. The masked edge $E_c^{\mathcal{M}}$ is subject to the constraint that $E_c^{\mathcal{M}}[v,w] \leq E_c[v,w], \forall (v,w \in \mathcal{E}_c(u)$. Then the objective function can be written as:

$$\min_M -\sum_{c=1}^{C} \mathbb{I}[y=c] \log P_\Phi(Y|G_c = (V_c, E_c \odot \mathcal{M}), X_c). \quad (10)$$

In GNNExplainer, the top k edges may not form a connected component including the node (say u) under prediction i. Hence, we added the entropy of the $(E_c \odot \mathcal{M})_{vu}$ for every node v pointing to node u' as a regularization term,

Algorithm 1 Optimize mask for weighted graph

Input: 1. $G_c(u)$, computation graph of node u; 2. Pretrained GNN model Φ; 3. y_u, node u's real label; 4. $\mathcal{M}$, learn-able mask; 5. K, number of optimization iterations; 6. L, number of layers of GNN.

$\mathcal{M} \leftarrow$ randomize parameters ▷ initialize, $\mathcal{M} \in [0,1]^Q$
$\mathbf{h}_v^{(0)} \leftarrow \mathbf{x}_v$, for $v \in G_c(u)$
for $k = 1$ to K **do**
  $\mathcal{M}_{vw} \leftarrow \frac{exp(\mathcal{M}_{vw} e_{vw})}{\sum_v exp(\mathcal{M}_{vw} e_{vw})}$ ▷ renormalize mask
  for $l = 1$ to L **do**
    $\mathbf{m}_{vu}^{(l)} \leftarrow W_1^{(l-1)} \mathbf{h}_v^{(l-1)}$ ▷ message
    $M_u^{(l)} \leftarrow \sum_v g(\mathcal{M}_{vu} \mathbf{m}_{vu}^{(l)}, \mathbf{h}_u^{(l-1)})$ ▷ aggregate
    $\mathbf{h}_u^{(l)} \leftarrow \sigma(W_0 \mathbf{h}_u^{(l-1)} + M_u^{(l)})$ ▷ update
  end for
  $\hat{\mathbf{y}}_u \leftarrow$ *softmax*$(\mathbf{h}_u^{(L)})$ ▷ predict on masked graph
  $loss \leftarrow$ *crossentropy*$(\mathbf{y}_u, \hat{\mathbf{y}}_u)$ + *regularizations*
  $\mathcal{M} \leftarrow$ *optimizer*$(loss, \mathcal{M})$ ▷ update mask
end for
Return: $\mathcal{M}$

to ensure that at least one edge connected to node u is selected. After mask $\mathcal{M}$ is learned, we use a threshold to remove small $E_c \odot \mathcal{M}$ and isolated nodes. Our proposed optimization methods to optimize $\mathcal{M}$ maximizing mutual information (equation (7)) under above constrains is shown in Algorithm 1.

4.2 Guided Gradient (GGD) Salience

Guided gradient-based explanation methods(Simonyan, Vedaldi, and Zisserman 2013) is perhaps the most straight forward and easiest approach. By simply calculate the differentiate of the output with respect to the model input then apply norm, a score can be obtained. The gradient-based score can be used to indicate the relative importance of the input feature, since it represent the change in input space which corresponds to the maximizing positive rate of change in the model output. Since egde weights have performed as filtering in GNN, we can obtain the edge mask as

$$g_{vu}^E = ReLU(\frac{\partial \hat{y}_u^c}{\partial e_{vu}}) \quad (11)$$

where $c \in \{1,\ldots,C\}$ is the correct class of node u, and y_u^u is the score for class c before softmax layer. where $\mathbf{x}_v$ is node v's feature. Here, we select the edges whose g^E is in the top k largest ones and their connecting nodes. The advantage of contrastive gradient salience method is easy to compute. However, it was argued that recently that it generally perform worse than newer techniques ((Zhang et al. 2018; Selvaraju et al. 2017)).

4.3 Edge Weighted Graph Attention (E-GAT)

The Graph Attention Layer takes a set of node features $\mathbf{H^{(l-1)}} = \{\mathbf{h}_1^{(l-1)}, \mathbf{h}_2^{(l-1)}, \cdots, \mathbf{h}_N^{(l-1)}\}$, $\mathbf{x}_i \in \mathbb{R}^F$ as input, and maps them to $\mathbf{H^{(l)}} = \{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \cdots, \mathbf{h}_N^{(l)}\}$, $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$. The idea is to compute an embedded representation

of each node $v \in V$, by aggregating its 1-hop neighborhood nodes $\{\mathbf{h}_v^{(l-1)}, \forall v \in \mathcal{N}(u)\}$ following a self-attention mechanism *Att*: $\mathbb{R}^{d^{(l)}} \times \mathbb{R}^{d^{(l)}} \rightarrow \mathbb{R}$ (Veličković and others 2018). Different from the original (Veličković and others 2018), we leverage the edge weights of the underlying graph. The modified **attention** $\alpha_{vu} \in \mathbb{R}$ can be expressed as a single feed-forward layer of $\mathbf{x}_v$ and $\mathbf{x}_w$ with *edge weight* e_{vu}:

$$\alpha_{wv}^{(l-1)} = Att(W_a\mathbf{h}_w^{(l-1)}, W_a\mathbf{h}_v^{(l-1)}) \tag{12}$$

$$= LeakyReLU((\mathbf{a})^\intercal[W_a\mathbf{h}_w^{(l-1)} \| W_a\mathbf{h}_v^{(l-1)}])e_{wv}, \tag{13}$$

where α is the attention weight on $v \rightarrow u$ and indicates the importance of node j's features to node i. It allows every node to attend all the other nodes on the graph based on their node features, weighted by the underlying connectivity. The $W_a \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is a learnable linear transformation that maps each node's feature vector from dimension $d^{(l-1)}$ to the embedded dimension $d^{(l)}$. The attention mechanism *Att* is implemented by a **nodal attributes learning vector** $\mathbf{a} \in \mathbb{R}^{2d^{(l)}}$ and *LeakyRelu* with input slope = 0.2. For explanation purpose, in order to make coefficients comparable cross different edges, we normalized the weights across the source nodes:

$$\tilde{\alpha}_{vw} = \frac{\alpha_{vw}}{\sum_{w \in N(v)} \alpha_{vw}} \tag{14}$$

where $T(v)$ is the target nodes set where v points to. Then, there will be an attention embedding layer before graph convolutional layer:

$$\mathbf{h}_v^{(l)} = \sum_{w \in \mathcal{N}(v)} \tilde{\alpha}_{vw}^{(l-1)} W_a \mathbf{h}_w^{(l-1)}. \tag{15}$$

Then we average the attention over the layers.

4.4 Culturing Node Class Sensitivity

For each node u, we computed the sensitivity of labeling it as class $i \in \{0, \ldots, C-1\}$ with respect to all nodes in the computational graph $v, w \in V_c(u) \setminus u$

$$\begin{bmatrix} ReLU(\|\frac{\partial \hat{y}_u^1}{\partial \mathbf{x}_v}\|) & \ldots & ReLU(\|\frac{\partial \hat{y}_u^C}{\partial \mathbf{x}_v}\|) \\ \vdots & \ddots & \vdots \\ Relu(\|\frac{\partial \hat{y}_u^1}{\partial \mathbf{x}_w}\|) & \ldots & ReLU(\|\frac{\partial \hat{y}_u^C}{\partial \mathbf{x}_w}\|) \end{bmatrix}, \tag{16}$$

then we clustered each row vector of the previous matrix, to obtain the set of neighbour nodes that have same contribution pattern to classify node u to each of class $i \in I_C$.

This method can be used to validate if the nodes on informative subgraph have the same node feature sensitivity. Also, it can show the similarity between the neighbors in G_c.

5 Evaluation Metrics and Methods

For synthetic data, we can compare explanation with data generation rules. However, for real data, we do not have ground truth for the explanation. In order to evaluate the results, we propose the evaluation metrics for quantitatively measuring the explanation results and propose the correlation methods to validate if edge connection patter or node feature is the crucial factor for classification. We define metrics *consistency*, *contrastivity* and sparsity (Here, definition of *contrastivity* andsparsity are different from the ones in(Pope et al. 2019)) to measure informative component detection results. Firstly, To measure the similarity between graphs, we introduce graph edit distance (GED) (Abu-Aisheh et al. 2015), which is a graph similarity measure analogous to Levenshtein distance for strings. It is defined as minimum cost of edit path (sequence of node and edge edit operations) transforming graph G1 to graph isomorphic to G2. In case the structure is isomorphic but edge weights are different. If GED=0, Jensen-Shannon Divergence (JSD) (Nielsen 2010), is added on GED to further compare the two isomorphic subgraphs. Specifically, we design consistency as the GED between the informative subgraphs of the node in the same class, as whether the informative components detected for the node in the same class are consist; and design contrastivity as the GED across the informative subgraphs of the node in the same class, as and whether the informative components detected for the node in the different class are contrastive; Sparsity is defined as the density of mask $\sum_{e_{vw} \in G_c(u)} \Upsilon_{vw}/Q, \Upsilon \in \{\mathcal{M}, g^E\}$, as the density of component edge importance weights.

6 Experiments

Note that, the color codes for all the figures below follow the on denoted in Figure 2. The red node is the node we try to classify and explain.

6.1 Synthetic Data

Data Following (Ying et al. 2019), we generated a Barabási–Albert (BA) graph with 15 nodes and attached 10 five-node house-structure graph motifs are attached to random nodes, ended with 65 nodes in Figure 2. We created a small graph for visualization purpose. However, the experiment results held for large graphs. Several natural and human-made systems, including the Internet, citation networks, social networks, and banking payment system can be

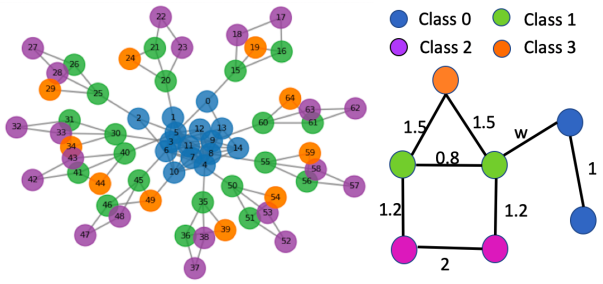

Figure 2: Synthetic BA-house graph data and corresponding edge weights, each BA node belongs to class "0," and each "house" shape node belongs labeled "1-3" based on its motif. The node orders are denoted.

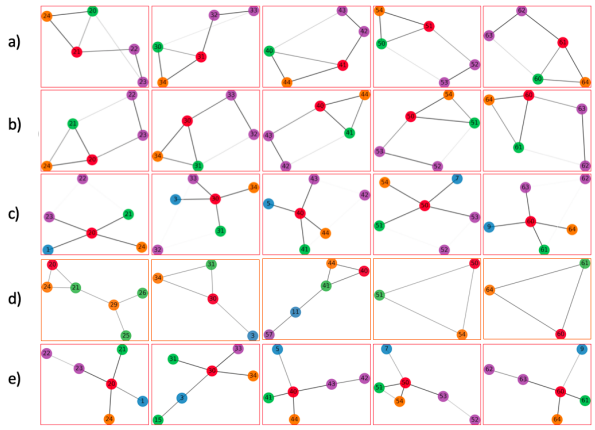

Figure 3: Informative Components. Row a)-c), $w = 0.1$. Row a) is for the node in class one not connecting to class 0 nodes using MMI mask. Row b) is for the node in class one connecting to class 0 nodes using MMI mask. Row c) is for the node in class one connecting to class 0 nodes using GGD. Row d) is for the node in class one connecting to class 0 nodes using E-GAT. Row e) is for the node in class one connecting to class 0 nodes using MMI mask, but $w = 2$.

Table 1: Saliency component compared with 'house' shape (*Measuring on all the nodes in class 1 with* $w = 0.1$)

Method	MMI mask	GGD	E-GAT
AUC	0.932	0.899	0.667

thought to be approximately a BA graph, which certainly contains few nodes (hubs) with unusually high degree and a big number of nodes poorly connected. The edges connecting with different node pairs were assigned different weights denoted in Figure 2 as well, where w was an edge weight we will discuss later. Then, we added noise to synthetic data by uniformly randomly adding $0.1N$ edges, where N was the number of nodes in the graph. In order to constrain the node label is determined by motif only, all the node feature $\mathbf{x}_i$ was designed the 2-D node attributes with the same constant.

GNN Training We use $g = \bar{e}_{vu}\mathbf{m}_{vu}^{(l)}$ in Eq. (4). The parameters setting are *input_dim = 2, hidden_dim = 8, num_layers = 3 and epoch =300.* We randomly split 60% of the nodes for training and the rest for testing.

Results GNN achieved 100% and 96.7% accuracy on training and testing dataset correspondingly. We performed informative component detection (kept top 6 edges) and compare them with human interpretation – the 'house shape,' which can be used as a reality check (Table 1). In Figure 3, we showed the explanation results of the node in the same place but has different topology structure (row a & b) and compared how eight weights affected the results (row a & e). We also showed the results generated by different methods (row a & c & d).

We showed a clustering results of node 20 on SynComp in Figure 4. Node $21 - 24$ were clustered together (visualized on 2-D space by T-SNE (Maaten and Hinton 2008) in Figure 4 (a)), since they were most sensitive to predicting node 20 to class 1, which matched the informative components (4 (b)) shown in main paper. The other clusters grouped the nodes in $G_c(u)$ by their saliency sensitive to a certain class in $\{0, 2, 3\}$.

This method can be used to validate if the nodes on informative subgraph have the same node feature sensitivity. Also, it can show the similarity between the neighbors in G_c.

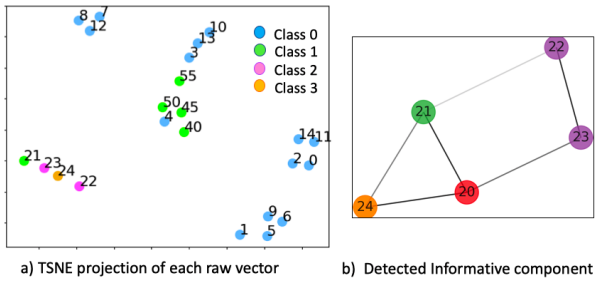

Figure 4: Node class sensitivities clustering (a) and comparing with informative subgraph (b). Node orders are denoted as numbers and node labels are denoted as colors.

6.2 Bitcoin OTC Data

Bitcoin is a cryptocurrency that is used for trading anonymously. There is counterparty risk due to anonymity. We use Bitcoin dataset ((Kumar et al. 2018)) collecting in one month, where Bitcoin users rate the level of trust to the users they made transactions to. The rating scales are from -10 to +10 (except for 0). According to OTC's guideline, the higher the rating, the more trustworthy. We labeled the users whose rating score had at list one negative score as risky; the users whose more than half received ratings were greater than one as trustworthy users; the users who did not receive any rating scores as an unknown group; and the rest of the users were assigned to the neural group. We chose the rating network data at a time point, which contained 1447 users, 5739 rating records. We renormalized the edge weights to $[0, 1]$ by $\tilde{e}_{ij} = e_{ij}/20 + 1/2$. Then we trained a GNN on 90% unknown, neutral and trustworthy node, 20% risky node, those nodes only, and perform classification on the rest of the nodes. We chose g as a GRU gate and the other settings are setting are *hidden_dim = 32, num_layers = 3 and*

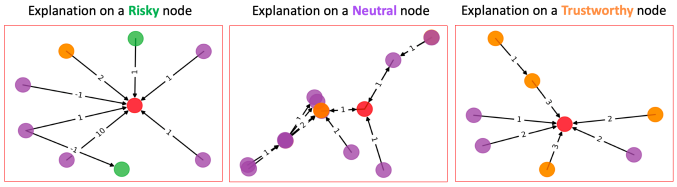

Figure 5: Informative subgraph detected by MMI mask (showing the original rating scores on the edges).

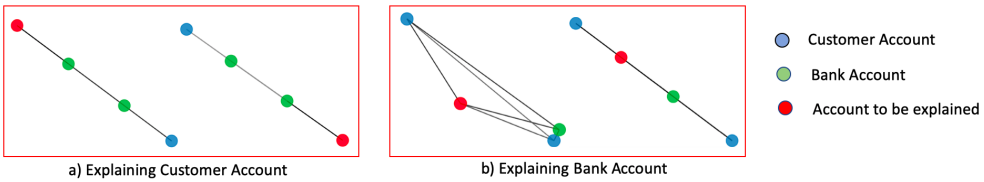

Figure 6: Examples of informative components for account matching.

epoch =1000. Learning rate was initialized as 0.1, and decreased half per 100 epochs. We achieved accuracy 0.730 on the training dataset and 0.632 on the testing dataset. Finally, we showed the explanation result using MMI mask since it is more interpretable (see Figure 5) and compared them with possible human reasoning ones. The pattern of the informative component of the risky node contains negative rating; the major ratings to a trustworthy node are greater than 1; and for the neutral node, it received lots of rating score 1. The informative components match the rules of how we label the nodes.

6.3 Bank Transaction Data - Account Matching

We use a database of 1000 international payment transaction records, that involve four parties, the originating (ORG) account, the sending (SND) bank, the receiving (RCV) bank and the beneficiary (BEN) account. Each party plays a role in the transaction, and the set of possible roles is $I_4 = \{\text{ORG, SND, RCV, BEN}\}$. The task is to classify each node as being either an account (ORG or BEN) or a bank (SNF or RCV), since the input graph data is noisy. For this, we build a transaction graph using the payment data, nodes are accounts of banks; the transactions between the nodes are the directed edges in the graph; and transaction amounts are the edge features. Furthermore, each node is associated with categorical features ('party ID type' and 'country'). We used one hot encoding to convert the node features to 10×1 vectors, and edge features were normalized to $[0, 1]$. We labeled 10% of the data and trained a GNN to classify each account as a bank and costumer account. We use the same GNN architecture as described in synthetic data experiments, and use Adam optimization method with fixed learning rate 0.01 and trained the model for 100 epochs until convergence. The accuracy of node classification task achieved is 100%. Using our explanation algorithm, we present a visualization of the informative components detected for each account type - costumer account or bank account in Figure 6.

For the above two real datasets, we measured consistency, contrastivity, and sparsity by selecting the top 4 edges. Since when attention layer was added in GNN, we could not achieve good classification results in the real datasets, we only applied MMI mask and GGD methods for informative components detection in the real data. The measurement of MMI mask and GGD methods are listed in Table 2. The higher contrastility values compared with consistency value, shows GNN replied on data topolgy information for node classification and nodes in the same class have similar topology structure. Low spasity values indicate the informative components have high information entropy, showing the potential of the explanation methods to extract informative patterns from the financial data.

Table 2: Evaluate informative components
(*Average on all the correctly classified nodes*)

	Dataset	Consistency	Contrastivity	Sparsity
MMI	BitCoin	1.79	3.23	0.126
	Account	1.25	1.93	0.021
GGD	Bitcoin	2.17	2.90	0.124
	Account	1.44	1.89	0.095

7 Conclusion

In this work, we formulate the transaction data in financial system as a weighted directed graph. We apply explanation methods on weighted graph in GNN node classification task, which can provide subjective and comprehensive explanations of data interaction patterns used in GNN. We also propose evaluation metrics and methods to validate the explanation results. The explanations may benefit debugging, feature engineering, informing human decision-making, building trust, etc. Our future work will include extending the explanation to graphs with multi-dimensional edge features and explaining different graph learning tasks, such as link prediction and graph classification.

References

Abu-Aisheh, Z.; Raveaux, R.; Ramel, J.-Y.; and Martineau, P. 2015. An exact graph edit distance algorithm for solving pattern recognition problems.

Baldassarre, F., and Azizpour, H. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Cover, T. M., and Thomas, J. A. 2012. *Elements of information theory*. John Wiley & Sons.

Gong, L., and Cheng, Q. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9211–9219.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kumar, S.; Hooi, B.; Makhija, D.; Kumar, M.; Faloutsos, C.; and Subrahmanian, V. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 333–341. ACM.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Nielsen, F. 2010. A family of statistical symmetric divergences based on jensen's inequality. *arXiv preprint arXiv:1009.4004*.

Pope, P. E.; Kolouri, S.; Rostami, M.; Martin, C. E.; and Hoffmann, H. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10772–10781.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

Shang, C.; Liu, Q.; Chen, K.-S.; Sun, J.; Lu, J.; Yi, J.; and Bi, J. 2018. Edge attention-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1802.04944*.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps.

Veličković, P., et al. 2018. Graph attention networks. In *ICLR*.

Yang, H.; Li, X.; Wu, Y.; Li, S.; Lu, S.; Duncan, J. S.; Gee, J. C.; and Gu, S. 2019. Interpretable multimodality embedding of cerebral cortex using attention graph network for identifying bipolar disorder. *MICCAI* 671339.

Ying, R.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*.

Zhang, J.; Bargal, S. A.; Lin, Z.; Brandt, J.; Shen, X.; and Sclaroff, S. 2018. Top-down neural attention by excitation backprop. *International Journal of Computer Vision* 126(10):1084–1102.

Zhang, Z.; Cui, P.; and Zhu, W. 2018. Deep learning on graphs: A survey.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2018. Graph neural networks: A review of methods and applications.