Problem Description

In this study we tackle the problem of money laundering detection in large-scale financial networks. We generate synthetic graph-structured data emulating a financial system with embedded money laundering topologies. We employ various Graph Deep Learning techniques and compare their effectiveness in detecting fraudulent accounts.

Synthetic Dataset Generation

Graph \( G(V, E) \) directed-multigraph with \( n \) vertices and \( m \) edges:
- Vertices: Bank accounts
- Edges: Transactions between accounts
- Financial graph-structured dataset generated using AMLEx. We specify number of co-mail and anomalous accounts, types of money laundering topologies, and the duration of simulation.
- Post-processing of generated dataset. Using edge-level features (transaction amount as weights of the edges), and node connectivity metrics, we compute node-level features used for training GNN models.

Money Laundering Topologies

Feature Generation

A set of node-level features are generated for each account:
- \( GAV\): geometric average of weights. For weight \( w_{ij} \) of neighbors of node \( i \) of total degree \( d_i \):
  \[
  GAV = \left( \frac{w_{ij}}{d_i} \right)^{1/d_i} \]
- \( GAV10 \) and \( GAV20 \): The GAV of each node computed for the largest 10% and 20% of its connected edge weights respectively.
- Standardized Node Degree: Where \( d_i \) is a vector of all node total degrees in graph \( G \):
  \[
  SND = \frac{d_i - \mu(d)}{\sigma(d)}
  \]
- Node Degree (IN/OUT) and Unique Node Degree (IN/OUT) when Graph \( G \) is reduced to Digraph and only one edge between nodes is permitted.
- Degree Frequency: Node's Total Unique Degree divided by Nodes Total Degree.
  \[
  DF(d) = \frac{d_i}{\sum d_i}
  \]
- Node's Community Edge Density: Communities \( C \) of a node \( i \) are computed using the Louvain Algorithm [Blondel et al., 2008]. Nodes are given their respective communities' edge density.
- Scale Node's Community Edge Density divided by Community Size.
  \[
  ScED = \frac{SED}{|C|}
  \]
- A node's (min/max/mean/std/total) transaction amount in.
- A node's (min/max/mean/std/total) transaction amount out.

Graph Neural Network Models And Datasets

Message Passing Neural Network (MPNN) [Gilmer et al., 2017]
Where \( [\alpha] \) is the \( j \)th convolutional layer for node \( i \), \( \alpha_{ij} \) and \( \alpha_{ji} \) are learnable parameters, \( \alpha_{ij} \) is an element of the graph shift operator for nodes \( i \) and \( j \), and \( \alpha_{ij} \) is the aggregator function for neighborhood of node \( i \).

GraphSAGE [Hamilton et al., 2017]
Convolutional Network (GCN) [Kipf and Welling, 2017]
Graph Attention Network (GAT) [Ying et al., 2018]
Graph Isomorphism Network (GIN) [Xu et al., 2019]

Datasets

Four datasets generated with varying number of anomalous nodes \( V_a \), number of normal nodes \( V_n \), and ratios of anomalous to normal nodes. Number of nodes remains constant across all datasets. Only the number of anomalous accounts and number of edges changes.

Data Processing and Model Training Pipeline

Algorithm 1 Node and Edge Feature Generation

Input: \( G(V, E) \) \( \Rightarrow \) generated dataset from AMLEx
Output: \( A_v, A_e, v, y \) graph features, edge attributes, edge list, target
1. \( A_v = \text{empty}(V) \)
2. \( A_e = \text{empty}(E) \)
3. \( A_v = \text{computeCommunity}(G(V, E)) \)
4. \( A_e = \text{computeCommunity}(G(V, E)) \)
5. \( \text{total} = \text{total} + \text{count} \)
6. \( \text{select} = \text{edge list} \)
7. \( y = \text{return node class} \)
8. \( \text{return} A_v, A_e, v, y \)

Algorithm 2 GNN Model Training Pipeline

Input: \( A_v, A_e, v, y \)
Output: model
1. \( \text{hidden} = \text{model.num_features} \)
2. \( \text{train}, \text{valid}, \text{test} = \text{split_data}(A_v, A_e, v, y) \)
3. \( \text{model} = \text{GINNModel}(\text{hidden} = \text{hidden}) \)
4. \( \text{train}, \text{valid}, \text{test} = \text{run_train}(\text{model}, \text{train}, \text{valid}, \text{test}) \)
5. \( \text{print} \)
6. \( \text{model.save} \)
7. \( \text{model.eval} \)
8. \( \text{return} \)
9. \( \text{return model} \)

Numerical Results

Performance of GNN Models Across Datasets

Model Performances on Topology Types Across Dataset Balances

GraphSAGE Node Classification Results for Dataset 4 (24% anomalous, 98% normal)

Model Training Times