Graph Algorithms and Machine Learning

Anton Sema, Julia Korol

July 2023

1 Introduction

Fraud detection refers to a collection of processes and analyses employed by businesses to identify and prevent unauthorized financial activities [2]. Such activities occur when individuals unlawfully obtain money or assets from other customers or companies. Fraud is not limited to banking systems alone but can also be found in areas such as insurance, healthcare, and government sectors.

Examples of fraudulent activities include stealing data to create false identities, money laundering, credit card fraud, cyber hacking, and mobile fraud.

Effective fraud detection procedures are essential for ensuring the safety of customers and businesses. However, detecting fraud is challenging for several reasons. Firstly, criminals continuously devise new methods to commit crimes, making it difficult to automate detection. Secondly, the process of verifying transactions as legitimate often takes time, which is typically slower than the rate at which illegal activities occur. Lastly, fraudulent transactions constitute only a small portion of all transactions, making it challenging to identify patterns of behavior that lead to criminal activity. This further complicates the use of traditional machine learning approaches.

In this context, machine learning algorithms that operate on graphs can prove to be valuable. Representing transactions as graphs provides additional information about the system, enabling the consideration of new factors and enhancing effectiveness.

In our project, we implemented two graph-based machine learning algorithms using an open-source dataset of Bitcoin transactions [4]. We compared the work of two standard models - Graph Convolutional Network and Graph Attention Network. Since the second one performed better, we further improved it with the voting method.

2 Methods

Graph Neural Networks (GNNs) are machine learning models designed to operate on graph data. In these models, we aim to create embeddings that capture the structural information of the network by considering the vertices and their features. A common approach [1] is to define a function $\mathbf{m}_{\mathcal{N}(u)}$ that aggregates information encoded by the embedding $\mathbf{h}_v^{(k)}$ from neighboring vertices of u (denoted as $\mathcal{N}(u)$) in an iteration k, as depicted below:

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})$$

In a GNN model, as described in [1], a typical step involves the following operation:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v,$$

$$\text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}(u)}\right)$$

Here, $\mathbf{W}_{\text{self}}^{(k)}$ and $\mathbf{W}_{\text{neigh}}^{(k)} \in \mathcal{R}^{d^{(k)} \times d^{(k-1)}}$ are trainable parameter matrices, and σ represents an activation function such as ReLU or sigmoid.

2.1 GCN

One of the models we utilized is the Graph Convolutional Network (GCN), which is a specific type of GNN that employs the following embedding technique [1]:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

where $\mathbf{W}$ denotes the trainable parameter matrix, that include self-loops. In other words, GCN is a GNN that normalizes the information by the degree of the source of the information and a degree of the node itself.

2.2 GAT

The second model utilized in the project is the Graph Attention Network (GAT). In this approach, the objective is to assign importance to each neighbor of a vertex during the aggregation process. The aggregate function is defined as:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$

In essence, we introduce an additional function α that determines the significance of the relationship between any two neighbors. A simple example of such an α function is a bilinear attention mode:

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{h}_u^\top \mathbf{W} \mathbf{h}_v\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{h}_u^\top \mathbf{W} \mathbf{h}_{v'}\right)}$$

However, in `pytorch` library a function GATConv, the classical definition from the paper "Graph Attention Networks" [3] is employed.

2.3 The voting method

As mentioned in the results, the GAT method exhibited superior performance on the dataset. To further enhance the results, we implemented the GAT method once again, but this time incorporating the voting technique. This method allows to combine the outputs of multiple models and determine the final result through voting.

There are two types of voting models that can be distinguished:

- **Hard voting**: In this approach, each classification result contributes to the final decision, and the outcome is determined by the class that receives the highest number of votes.
- **Soft voting**: Here, the final result is obtained by averaging all the individual classification results.

In our implementation, we opted to use the hard voting method to combine the outputs of the GAT models.

3 Implementation

In this section we briefly discuss our implementation of the above mentioned methods on Eliptic Data Set. The code can be found on the `Github` repository.

3.1 Elliptic Data Set

The Elliptic Data Set [4] contains the record of Bitcoin Blockchain transactions. Each node represents a particular transaction, and an edge can be considered as a flow of Bitcoins between two transactions. There are 166 node features. Additionally, nodes are labeled as "licit" (21% of the nodes), "illicit" (2%) or

"unknown". Due to simplicity, we used only the subgraph spanned on the labeled nodes, which leaves us with 42,019 "licit" nodes and 4545 "illicit" ones.

3.2 Code

The programs were implemented in Python, with the use of the libraries `numpy`, `pandas` and `pytorch`. The data was splitted using the `train_test_split` function, with the 15% test size.

3.3 Results

Method	GCN	GAT	GAT-vote
time (s)	60.93	38.9	7831.12
# epochs	15	40	40
licit accuracy (%)	83.145	91.353	
illicit accuracy (%)	72.536	82.043	5415
overall accuracy (%)	545	778	7507

4 Summary

Our project focused on implementing basic graph ML algorithms on the data set containing fraud bank transactions. It main point was to use the GNNs and GATs algorithms to detect the illicit ones.

The project was an opportunity to broaden the knowledge about those algorithms and face some practical problems connected with them. A notable challenge we encountered was working with training and validation data. Unlike conventional datasets for CNNs, RNNs, and LSTMs, handling GNN data required special attention. Parameter tuning also played a significant role in our approach.

We implemented message passing, a unique feature of GNNs, for training purposes. We compared this results with the work of GATs algorithms, a combination of attention mechanisms commonly used in NLP transformers and with classical GNNs. Our findings indicated that GAT consistently outperformed traditional GNNs, leading to improved results.

To further enhance the project's outcomes, we implemented the voting method, resulting in significant performance improvements. Overall, our project shed light on the promising potential of GNNs, offering valuable insights for future research and applications in diverse domains.

References

[1] W. L. Hamilton. *Graph Representation Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning*, volume 14. McGill University, 2020.

[2] A. Taing. Fraud detection with graph attention networks, 2022. URL `https://medium.com/stanford-cs224w/fraud-detection-with-gat-edac49bda1a0`. Stanford CS224W GraphML Tutorials.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018.

[4] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, August 2019. URL `https://www.kaggle.com/datasets/ellipticco/elliptic-data-set`. KDD '19 Workshop on Anomaly Detection in Finance.