RICE ENGINEERING
Computer Science

Fraud Detection on Credit Card Transactions

COMP-559
Machine Learning with Graphs

Students:

Fernando Raverta (fr15)
Sathwik Yarlagadda (sy81)
Minh Nguyen (mn78)

1 Introduction

Credit card fraud detection (CCFD) encompass detecting scenarios where cards are forged, stolen, or otherwise compromised, allowing fraudsters to make purchases under the guise of the legitimate cardholder. As most of the transaction are legitimate made by the card owners, the problem of detecting those fraudulent observations can be seen as anomaly detection problem.

However, the CCFD problem has specific challenges that the solution methods should tackle. Some of the most important ones are as follows:

- **Training data availability**: training data for fraud detection is very scarse due to its sensitive nature. Many datasets are either fully synthetic or the data has been altered so to not reveal sensitive information about credit card users.
- **Dynamic fraudster behavior**: Fraudsters are not static entities, they learn and adapt their behavior so to be able to bypass security measures. Also, they engage in camouflage to prevent systems from detecting their scams.
- **Class imbalance**: The rarity of fraud records compared to legitimate transactions creates a skewed dataset, complicating the training of models to detect fraud effectively.

These specific challenges highlights the need for developing specific solutions for the CCFD problem. In this context, Graph Machine Learning which can leverage information from the relationship among entities appears as a promising approach.

1.1 Project goal

The aim of this project is to explore the the CCFD problem using Graph Machine Learning approaches. Particularly, we will implement GraphSAGE algorithm ([1]) and then, a modification proposed by [2], explicitly designed for the CCFD problem. During our review of the study [3], we detected the following weakness:

- The approach proposed was tested on a private dataset from a financial company on China, and as a consequence, the results are not repeatable by readers. Moreover, in an extension of this former paper ([3]), the authors claim they tested the approach in 2 other datasets that were supposed to be publicly exposed. However, we were unable to find one of the them and the other was about Mobile Advertising frauds that apparently did not involved any credit card transactions.
- We tested the proposed approach on predicting "Card-not-present" frauds, which [4] defines as frauds conducted remotely via the internet, mail, or phone, using card information without the physical card itself. However, the original authors did not tailor their proposal specifically for these scenarios. Therefore, we extend the evaluation to include "Card-present" frauds, a different type of fraud where a fraudster successfully executes a transaction using a physical payment card at an ATM or a point of sale (POS).

To assess the proposal, we will use the state-of-the-art Credit Card Fraud Transaction simulator proposed at [5]. This project presents several challenges including modeling the transactions data into the a Graph representation proposed by [2]. Moreover, we will also implement GraphSAGE, a popular GNN approach, and we will use XGBoost to set a baseline for the results.

This report is organized in the following manner: Initially, we provide a detailed description of the simulated dataset. Subsequently, we introduce the concept of the Transition Graph (TG), the graph abstraction that underpins the solution we implemented, and explain our modeling approach for this problem. Finally, we present and discuss the results of our analysis.

2 Methodology

2.1 Credit card fraud transaction scenario simulator and simulated dataset

A simulated dataset is a set of data artificially generated to model and mirror the characteristics of real-word data, while being created under controlled conditions based on specific rules or algorithms. These types of datasets are particularly useful in scenarios where the real-world data contains sensitive information, such as in our case of fraud detection: credit card numbers or other personal financial information.

For our project, we used a transaction data simulator [6] to create a simulated dataset [5] of legitimate and fraudulent transactions. The simulated transaction dataset adopts a simplified design with 6 main features:

- The transaction ID ($TRANSACTION\_ID$): A unique identifier for the transaction
- The date and time ($TX\_DATETIME$): Date and time at which the transaction occurs
- The customer ID ($CUSTOMER\_ID$): The identifier for the customer. Each customer has a unique identifier
- The terminal ID ($TERMINAL\_ID$): The identifier for the merchant (or more precisely the terminal). Each terminal has a unique identifier
- The transaction amount (TX_AMOUNT): The amount of the transaction.
- The fraud label (TX_FRAUD): A binary variable, with the value for a legitimate transaction, or the value for a fraudulent transaction.

Despite its simplicity, the simulated dataset also captures many challenges encountered with real transaction data. One of which is class imbalance. The dataset is able to mimic realistic scenario where fraudulent are extremely rare in real life.

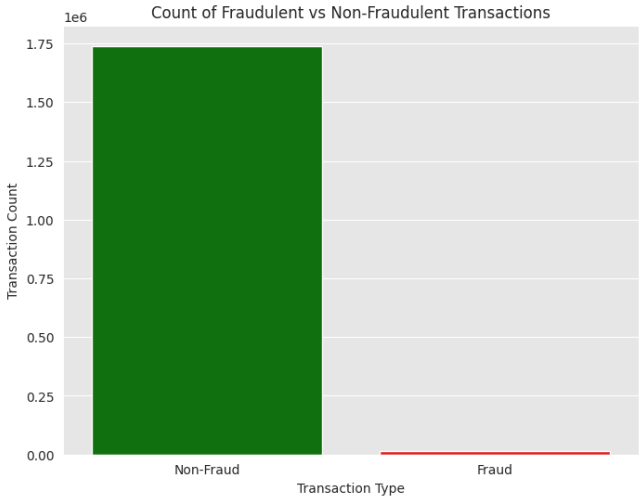

Figure 2: Comparison of Fraudulent and Non-Fraudulent Transactions

From our analysis of the dataset before data pre-processing, we noticed that our the number of fraudulent cases is less than 1% total of the transaction simulated as demonstrated in Figure 2.

2.2 Implementation

In this project, we have implemented the methodology as proposed by Liu, et al. [2], which outlines a novel approach for solving the problem of CCFD using Graph Neural Network. Our implementation closely follows the guidelines and algorithms described in their work.

2.2.1 Transaction Graph (TG)

The authors of the paper discuss the development of a credit card TG, which is a weight multi-graph designed to thoroughly assess the relationships and the original attributes of transactions, as well as their dynamic aspects.

Definition 1. *A transaction record, denoted as $r = (a_1, a_2, \ldots, a_m)$, is defined as a vector of m attribute values. Each attribute a_i is included in a set A_i, which defines the permissible range of values for the i-th attribute.*

The construction of a TG involves defining a set of logical propositions $P = \{p_1, p_2, \ldots, p_t\}$, which determine the connections between transaction records based on attribute values. In our implementation, the proposition $p_1(x, y)$ would connects nodes x and y if they share the same customer id and their transaction occur are within 14 days of each other. The graph then utilizes a weighting function $Weight : P \to \mathbb{N}$ to assign importance to each proposition, influencing the strength of the connections in the TG. We picked the weight for p_1 to be 1 or $Weight(p_1) = 1$. Here is a formal definition of $p_1(x, y)$ in our current implementation.

$$p_1(x, y) = \begin{cases} \text{true} & \text{if } x.\text{CUSTOMER\_ID} = y.\text{CUSTOMER\_ID} \wedge \\ & |x.\text{TX\_DATETIME} - y.\text{TX\_DATETIME}| \leq 14 \text{ days} \\ \text{false} & \text{otherwise} \end{cases}$$

Similarly, we have p_2 as transaction records x and y have the same terminal id and take place within 28 days with a weight of 1. The proposition logics are specifically chosen based on some

potential fraud scenarios as described in [7]. p_2 with a weight of 2 is selected to efficiently track and analyze extended fraudulent activities on terminals over 28 days. In scenario 3, p_1 is designed to detect significant alterations in transaction patterns due to a fraudulent payment terminal or point-of-sale (POS), focusing on transactions within a 14-day period, thus adapting to the dynamic nature of fraud.

With the proposition logic properly defined, we will have a formal definition for TG below.

Definition 2. *Given a set of transaction records R, its Transaction Graph (TG) is a weighted multi-graph $G = (V, E, P, Weight)$ where:*

1. $V = \{v_1, v_2, \ldots, v_{|R|}\}$ *represent* $|R|$ *nodes corresponding to the records in* R*;*
2. $P = \{p_1, p_2, \ldots, p_t\}$ *is the set of* l *logic propositions and* $l \geq 1$*;*
3. $Weight : P \to \mathbb{N}$ *is the weight function;*
4. $E = \bigcup_{i=1}^{l} \{(x, y) | x \in V \wedge y \in V \wedge x \neq y \wedge p_i(x, y) = true \wedge w = Weight(p_i)\}$.

2.2.2 Modified GraphSAGE algorithm

With the TG constructed, it will be employed by the GraphSAGE algorithm to enhance fraud detection, involving two primary steps. Initially, a fixed number t of neighbors $N'(v)$ is randomly sampled from each node v's neighborhood $N(v)$. Node embeddings are then updated through several aggregation functions AGGREGATE_k for $k = 1, \ldots, K$, utilizing weight matrices W^k to integrate and propagate neighbor information across the network. Each node's representation h^k is refined by concatenating its previous state with aggregated inputs, followed by a nonlinear activation function σ, preparing it for subsequent processing or downstream applications. However, due the complex topological relationship of the TG, the paper [2] proposed a new method of sampling and aggregation.

1. Sampling

   The TG needs a more advanced sampling strategy that considers attribute similarities, measured by cosine similarity, and the strength of connections, defined by edge weights, to enhance node selection. This methodology aims to improve information aggregation by selecting nodes that are both attribute-similar and strongly connected, facilitating more effective differentiation between fraudulent and normal transactions. The distance between nodes based on attribute vectors is calculated as:

   $$\text{dis}(v, u) = \exp(r_v \cdot r_u)$$

   The selection probability for a pair of nodes is determined using:

   $$\text{Pro}(v, u) = \frac{w(v, u) \cdot \text{dis}(v, u)}{\sum_{u \in N'(v), v \neq u} w(v, u) \cdot \text{dis}(v, u)}$$

2. Attention-based Aggregating Process

   The attention-based aggregation process in the TTG utilizes GraphSAGE with mean-pooling initially to derive node representations. As fraud detection can be sensitive to the surrounding node context, an attention mechanism modifies the aggregation by weighting neighbor contributions based on their relevance, defined by both the attribute similarity and transaction timing. This approach adjusts node representations dynamically, focusing on more relevant transaction patterns for effective fraud detection.

The modified aggregation function incorporating attention is given by:

$$h^k_{N'(v)} = \alpha_{v,u} \cdot \text{AGGREGATE}_k(h^{k-1}_u, \forall u \in N'(v))$$

where $\alpha_{v,u}$ is the attention score between nodes v and u, defined as:

$$\alpha^k_{u,v} = \delta_{t,v,u} \cdot \exp(\sigma(\mathbf{d}^T_k \cdot \text{concat}(W^k h^k_v, W^k h^k_u)))$$

where $\mathbf{d}_k$ are the weight parameters of the k-th attention layer, and $\delta_{t,v,u}$ represents the normalized transaction time interval adjustment between nodes.

Based on these new proposed methods for the 2 main steps in GraphSAGE, below is the new algorithm that we are implementing:

Algorithm 1 Graph algorithm based on similarity sampling and attention mechanism

Input: TG $G = (V, E, P, \text{Weight})$, record features r_v of each record node, level number k, non-linear activation function σ, neighborhood sample size *nums*, random sampling threshold β, and edge weight $\{\text{Weight}_1, \ldots, \text{Weight}_n\}$.
Output: embedding representations z_v for every node in graph
Choose level to sample according to neighborhood sample size *nums*.
for $i = 1$ to *nums* **do**
    $N'(v)^i \leftarrow$ sampling based on $\text{Pro}(v, u)$ from $N(v)$;
end for
$h^0_v \leftarrow r_v, \forall v \in V$;
for $k = 1$ to K **do**
    for $v \in V$ **do**
        $h^k_{N'(v)} \leftarrow \alpha_{v,u} \cdot \text{AGGREGATE}_k(h^{k-1}_u, \forall u \in N'(v))$;
        $\alpha^k_{u,v} \leftarrow \delta_{t,v,u} \cdot \exp(\sigma(\mathbf{d}^T_k \cdot \text{concat}(W^k h^k_v, W^k h^k_u)))$;
        $h^k_v \leftarrow \sigma(W^k \cdot \text{concat}(h^{k-1}_v, h^k_{N'(v)}))$;
    end for
    Normalize $h^k_v \leftarrow \frac{h^k_v}{\|h^k_v\|_2}, \forall v \in V$;
end for

2.2.3 Graph Neural Network for Credit Card Fraud Detection (GNNCCFD) model

Our model leverages the GraphSAGE algorithm above to effectively process TG. It operates in two main layers ($K = 2$), using the mean-pooling aggregation strategy to integrate information from neighbor nodes. Sampling is conducted on the neighborhood level with a size of 20 neighbors based on their importance, determined through attention scores calculated by the cosine similarity and time intervals.

The first convolution layer transforms input features into a hidden representation, followed by a normalization step using the L2 norm and the ReLU activation function. Dropout is applied before the second convolution layer processes the hidden representation into output features. The model outputs logits for binary classification tasks. The learning process is governed by the Adam optimizer with a learning rate of 0.01, and loss is computed using the binary cross-entropy with logits (BCEWithLogitsLoss).

Key Hyperparameters:

- Two graph convolution layers (`SAGEConv`) are used, with 64 hidden layers.
- The neighborhood sampling size is fixed at 20 (`num_samples=20`).

- Training involves 100 epochs, indicating the duration over which model parameters are adjusted.
- A threshold (0.2) is used to classify outputs from the sigmoid-activated layer during testing, balancing sensitivity and specificity.

3 Experiment Results

We performed an study over the credit card simulated dataset. We removed the fraud transaction belonging to the fraud 1 because we wanted to focus our attention on frauds that can be learn from the transaction relationships (links). We select XGBoost (50 trees, max_depth=6) as baseline model because it was used in the paper we read. Then, we tested 2 machine learning over graph solutions: Graph-SAGE -a general purpose model for extracting embedding from graphs- and GNNCCFD model which claim to be an improvement of Graph-SAGE for the CCFD porblem. For the sake of truth, in the results using we did not leverage the attention mechanism proposed in GNNCCFD but only its neighbors sampling designed to deal with the particularities of CCFD problem. The parameters used for each model are summarized in table 1.

Parameter	**XGBoost**	**Graph-SAGE**	**GNNCCFD**
n_estimators	50	—	—
max_depth	6	—	—
Random state	42	—	—
dim_in	—	4	4
dim_h	—	64	64
dim_out	—	2	2
num_of_samples	—	—	20
edge_time_intervals	—	—	included
Epochs	—	250	250

Table 1: Model Parameters for XGBoost, Graph-SAGE, and GNNCCFD

The models were compared varying the fraudulent to legitimate transactions ratio. For achieving the ratio selected we randomly sub-sample legitimate transactions. This is a common practice on imbalanced datasets and was indeed used in [2]. Results in terms of F1-score, recall and AUC are shown on Fig. 3. GNNCCFD, a model designed for CCFD problem, outperformed the plane machine learning solution (XGBoost) and the general Graph-SAGE approach in the 3 metrics and for every fraudulent to legitimate transactions ratio.

However, there is significant room for improvement since the performance metrics are still far from ideal. By fine-tuning the models and fully implementing the attention-based aggregation process, GNNCCFD should be able to excel in this setting.

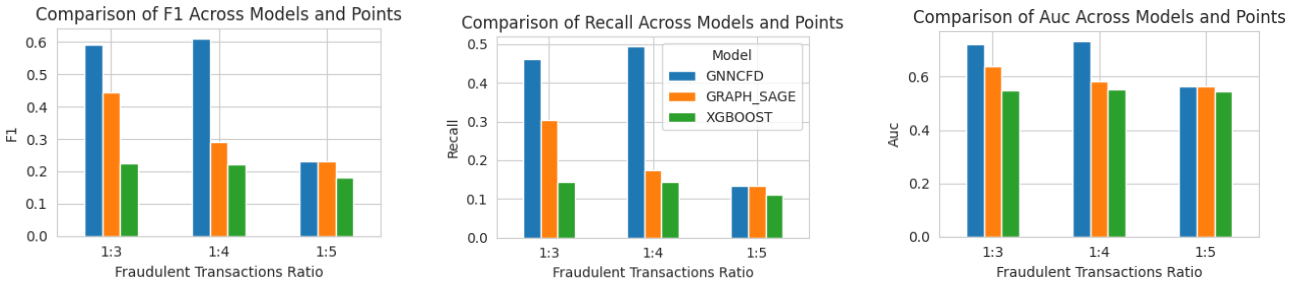

Figure 3: Comparison of XGBoost, Graph-SAGE and GNNCCFD varying fraudulent transaction ratio

4 Challenges

In the context of credit card fraud detection (CCFD), class imbalance is a significant hurdle. Typically, datasets are skewed with a higher number of legitimate transactions compared to fraudulent ones. This disparity can compromise machine learning models by predisposing them toward the majority class, leading to many fraudulent transactions being incorrectly classified as legitimate. Most conventional algorithms are optimized for overall accuracy, which often means predicting the majority class for all data points. This method effectively identifies most legitimate transactions but fails to detect fraud. The consequences of such oversight are critical, as undetected fraud can result in substantial financial losses and erode trust in financial systems.

5 Conclusion

We applied our knowledge from COMP-559 to set up this project, in which we implemented and tested the approach proposed by [2]. Translating the idea into this specific setting presented significant challenges, requiring us to develop a modeling approach and implement the algorithms effectively. Undoubtedly, this work has deepened our understanding of the course concepts and laid a solid foundation for us to apply the Graph Machine Learning toolset in professional contexts.

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[2] GuanJun Liu, Jing Tang, Yue Tian, and Jiacun Wang. Graph neural network for credit card fraud detection. In *2021 International Conference on Cyber-Physical Social Intelligence (ICCSI)*, pages 1–6, 2021. doi: 10.1109/ICCSI53130.2021.9736204.

[3] Yue Tian et al. Asa-gnn: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection. In *Proceedings of the IEEE International Conference on Neural Networks and Learning Systems*, 2023.

[4] Machine Learning Group, Université Libre de Bruxelles (ULB). Reproducible machine learning for credit card fraud detection - practical handbook, section 2.1: Transaction data simulator. `https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_2_Background/CreditCardFraud.html#card-present-vs-card-not-present-frauds`, 2023.

[5] Machine Learning Group, Université Libre de Bruxelles (ULB). Simulated data transformed - github repository. `https://github.com/Fraud-Detection-Handbook/simulated-data-transformed`, 2023.

[6] Machine Learning Group, Université Libre de Bruxelles (ULB). Reproducible machine learning for credit card fraud detection - practical handbook, section 3.1: Transaction data simulator. `https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/SimulatedDataset.html`, 2023.

[7] Machine Learning Group, Université Libre de Bruxelles (ULB). Reproducible machine learning for credit card fraud detection - practical handbook, section 3.2.5: Fraud scenarios generation. `https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/SimulatedDataset.html#fraud-scenarios-generation`, 2023.