Integer Formulation for Computing Transaction Aggregation to Detect Credit Card Fraud

Mauro Escobar[1], Claudia D'Ambrosio[1], Leo Liberti[1], Sonia Vanier[2]

[1] LIX CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
{escobar,dambrosio,liberti}@lix.polytechnique.fr
[2] Université Paris 1 Panthéon-Sorbonne, Paris, France
Sonia.Vanier@univ-paris1.fr

Abstract

Banks assume great costs caused by fraudulent credit card transactions. With the development of new means of payment and the virtualization of banking services, fraud is becoming more and more difficult to detect and its frequency is growing at a dizzying rate. Currently available defense tools are not enough to counter fraud in continuous evolution. In this work, we present an integer formulation to compute functions of transactions data that turn to be useful for detecting behavioral patterns present in past fraudulent events. Our formulation allows the aggregation of transactions over any set of features, in addition to filtering transactions that verify structured requirements. We test our model with real-world data from a French bank, using SAT solvers. Numerical results obtained on several instances show the effectiveness of our approach.

Keywords : *Credit card, Fraud detection, Transaction aggregation.*

1 Introduction

Fraud on credit cards transactions is a great concern for banks since it represents a non-negligible loss for them. In 2018, 92.4% of fraud events in the French bank system were associated to credit card payments and withdrawals, corresponding to 439 million euros and 42% of the total fraud losses (38.4% for payments and 3.6% for withdrawals). Note that the fraud rate on card transactions was only 0.062% [2].

There are different reasons that explain the source of the credit card fraud. Fraud can occur when a card has been lost or stolen, and it is subsequently used by someone else. A common card theft method, for example, is the interception of credit cards in the mail system before its arrival to the client's home address. Another type of fraud happens when the information of the magnetic strip on the card is stolen and used to produce a counterfeit card. This can be performed by placing additional card readers in ATMs or payment terminals. Malicious software is also used to participate in fraudulent transactions: some of these algorithms randomly create credit card numbers and test them over the internet in order to verify if the number is accepted by online merchants; there are also false merchant websites that capture credit card numbers from clients that believe in the security of these interfaces. We finalize this summary of types of fraud cases by the abusive clients, which are clients who make purchases but then declare that these payments were not done by them. See [1] for a more detailed description of fraud methods.

In this work, we analyze data features useful to predict whether a payment transaction can be fraudulent or not. Dozens of features come together with the payment information of each purchase, while many other can be constructed from the purchase history of the bank clients. The former, usually called *primary attributes* [4], include among others: date and time of the transaction, credit card number, transaction amount, currency, type of transaction (physical or long distance), merchant category code, merchant country, merchant identification, acquirer country, and acquirer identifier. The latter, the *derived attributes*, are computed by analyzing the behavior of the clients, that is, these features are functions that have as arguments the set of transactions that happened in the recent past. As an example, the amount spent in the last 24 hours by the card associated to a transaction is a derived attribute of this transaction (taking into account the 24 hours before the transaction happens).

In [4, 5], the authors discuss a transaction aggregation strategy to compute derived attributes from primary variables, where the aggregation occurs for each transaction over transactions performed by the same client. Fourteen new derived attributes are presented in [4]. Some examples are: “transaction amount over month” (average spending per transaction over a 30-day period on all transaction till this transaction), “number same merchant” (total number of transactions with the same merchant during the last month), “amount merchant type over 3 months” (average weekly spending on a merchant type during past 3 months before a given transaction).

Our contribution is an integer programming formulation that computes a *generalized derived attribute* by allowing any primary attribute as aggregation field. For example, aggregation by merchant could help to detect fraudulent points of sale.

2 Preliminaries

Consider a database $\mathcal{D} = \{T_1, \dots, T_N\}$ of N transactions. Let $S = \{1, \dots, N\}$ be the set of indices of transactions in $\mathcal{D}$. A transaction with index $s \in S$ is described by a vector $T_s = (\mathtt{t}_s, \mathtt{c}_s, \mathtt{a}_s, \mathtt{r}_s, \mathtt{m}_s, \mathtt{p}_s, \mathtt{q}_s, \mathtt{b}_s, \mathtt{d}_s)$ of primary attributes each of them summarized in Table 1.

Attribute	Type	Description
$\mathtt{t}_s$	numerical[1]	date and time
$\mathtt{c}_s$	categorical	credit card number (client identifier)
$\mathtt{a}_s$	numerical	amount of the transaction (in euros)
$\mathtt{r}_s$	categorical	original currency of the transaction
$\mathtt{m}_s$	categorical	merchant category code (MCC)
$\mathtt{p}_s$	categorical	merchant country
$\mathtt{q}_s$	categorical	merchant identifier
$\mathtt{b}_s$	categorical	acquirer (merchant's bank) country
$\mathtt{d}_s$	categorical	acquirer identifier

TAB. 1: Primary attributes of transaction T_s.

We denote by the corresponding uppercase letter $\mathtt{C} = \{\mathtt{c}_s : s \in S\}$ the set of all credit card numbers in the database. Similarly, we denote with $\mathtt{R}, \mathtt{M}, \mathtt{P}, \mathtt{Q}, \mathtt{B}, \mathtt{D}$ the set of values of the remaining categorical primary attributes. We denote by the symbols $\mathbb{f}$ or $\mathbb{g}$ any generic attribute in the set $\{\mathtt{c}, \mathtt{r}, \mathtt{m}, \mathtt{p}, \mathtt{q}, \mathtt{b}, \mathtt{d}\}$, and by the uppercase letters $\mathbb{F}$ or $\mathbb{G}$ the corresponding set of values $\mathtt{C}$, $\mathtt{R}$, $\mathtt{M}$, $\mathtt{P}$, $\mathtt{Q}$, $\mathtt{B}$, or $\mathtt{D}$ that these attributes take in the database. From here onwards, we will use interchangeably the terms attribute and field (which were also referred as “features” in the introduction). We assume that values in the categorical fields are injected to integer values.

3 Formulation

We describe an integer formulation for computing a derived attribute. Such derived attributes are a generalization of the ones computed in [4, 5], since they allow any of the primary attributes

[1]We can consider $\mathtt{t}_s$ as a numerical attribute by considering the length of the time interval between a fixed initial time (before any transaction in $\mathcal{D}$) and the time of T_s.

to be the aggregation field. They also consider conditions that filter the transactions involved in the aggregation.

In order to compute the new attribute, we consider:

- $\tau > 0$: a length of time, to consider past transactions that recently happened before a specific transaction,
- $\mathbb{f}^1, \ldots, \mathbb{f}^k \in \{\mathtt{c}, \mathtt{r}, \mathtt{m}, \mathtt{p}, \mathtt{q}, \mathtt{b}, \mathtt{d}\}$: k different aggregation fields,
- $\mathbb{g}^1, \ldots, \mathbb{g}^\ell \in \{\mathtt{c}, \mathtt{r}, \mathtt{m}, \mathtt{p}, \mathtt{q}, \mathtt{b}, \mathtt{d}\}$: ℓ different condition fields,
- $g^1, \ldots, g^\ell$: ℓ values for the corresponding condition fields, that is, $g^j \in \mathbb{G}^j$ for each $j \leq \ell$,
- $M > 0$: a large constant.

Using these symbols, for each $s \in S$, we compute a derived attribute $\mathtt{x}_s$ that counts transactions having the following characteristics. This new attribute considers only the transactions that happened at most τ units of time before T_s. Moreover, these transactions are aggregated according to fields $\{\mathbb{f}^i\}_{i=1}^k$, that is, the considered transactions are the ones having the same categorical value as T_s on these fields. Lastly, this count only includes transactions that have value g^j on field $\mathbb{g}^j$, for $j \in \{1, \ldots, \ell\}$. The attribute $\mathtt{x}_s$ can be expressed as follows:

$$\begin{aligned}
\forall s \in S: &\quad & \mathtt{x}_s &= \sum_{u=1}^{N} y_{s,u} & (1)\\
\forall s,u \in S: &\quad & y_{s,u} &= \prod_{i=1}^{k+2} w_{s,u,i} \cdot \prod_{j=1}^{\ell} z_{u,j} & (2)\\
\forall s,u \in S, \forall i \leq k: &\quad & 1 - w_{s,u,i} &\leq |\mathbb{f}^i_s - \mathbb{f}^i_u| \leq M(1 - w_{s,u,i}) & (3)\\
\forall s,u \in S: &\quad & (\mathtt{t}_u + 1) w_{s,u,k+1} &\leq \mathtt{t}_s \leq \mathtt{t}_u(1 - w_{s,u,k+1}) + M w_{s,u,k+1} & (4)\\
\forall s,u \in S: &\quad & (\mathtt{t}_s - \mathtt{t}_u) w_{s,u,k+2} &\leq \tau \leq (\mathtt{t}_s - \mathtt{t}_u - 1)(1 - w_{s,u,k+2}) + M w_{s,u,k+2} & (5)\\
\forall u \in S, \forall j \leq \ell: &\quad & 1 - z_{u,j} &\leq |g^j - \mathbb{g}^j_u| \leq M(1 - z_{u,j}) & (6)\\
\forall s,u \in S, \forall i \leq k+2, \forall j \leq \ell: &\quad & y_{s,u}, w_{s,u,i}, z_{u,j} &\in \{0,1\}. & (7)
\end{aligned}$$

Eq. (1)-(2) consider counting transaction T_u into $\mathtt{x}_s$ when $y_{s,u} = 1$, *i.e.* when all the binary variables $\{w_{s,u,i}\}_{i=1}^{k+2}, \{z_{u,j}\}_{j=1}^{\ell}$ are equal to 1. Note that $w_{s,u,i}$ depends on the attributes of transactions T_s and T_u, whereas $z_{u,j}$ depend only on T_u.

We will assume that M is larger than τ and than any of the possible values of $\mathtt{t}_s$, $|\mathbb{f}^i_s - \mathbb{f}^i_u|$ and $|g^j - \mathbb{g}^j_u|$. Note that we can strengthen this formulation by considering a different constant M for each of the Eq. (3)-(6), we omit this fact in order to simplify the notation. It is not difficult to verify that Eq. (3) implies that $w_{s,u,i} = 1$ if and only if $\mathbb{f}^i_s = \mathbb{f}^i_u$. Similarly, Eq. (6) implies that $z_{u,j} = 1$ if and only if $\mathbb{g}^j_u = g^j$.

The following result clarifies why Eq. (4)-(5) characterize that T_u happens shortly before T_s.

Proposition 1 *Let a, b, c be positive integers such that $b \leq c$ and let w be a binary variable. Then, $(a+1)w \leq b \leq a(1-w) + cw$ implies the equivalence "$w = 1$ if and only if $a < b$."*

Proof : Assume that the inequalities $(a+1)w \leq b \leq a(1-w) + cw$ hold. If $w = 1$, then $(a+1) \leq b \leq c$ which leads to $a < b$. If $w = 0$, then $0 \leq b \leq a$. $\square$

Therefore, by Eq. (4)-(5), the variables $w_{s,u,k+1}$ and $w_{s,u,k+2}$ are equal to 1 if and only if $\mathtt{t}_u < \mathtt{t}_s$ and $\mathtt{t}_s - \mathtt{t}_u \leq \tau$, respectively. That is, T_u happened before T_s and the time in between both transactions is less or equal than τ.

Finally, we can linearize Eq. (2) with the following set of equations:

$$\forall s, u \in S, \forall i \leq k+2: \quad y_{s,u} \leq w_{s,u,i} \tag{8}$$

$$\forall s, u \in S, \forall j \leq \ell: \quad y_{s,u} \leq z_{u,j} \tag{9}$$

$$\forall s, u \in S: \quad y_{s,u} + k + \ell + 1 \geq \sum_{i=1}^{k+2} w_{s,u,i} + \sum_{j=1}^{\ell} z_{u,j}. \tag{10}$$

Note that, on one hand, if one of the w or z variables is zero, then the right hand side of Eq. (10) is at most $k + \ell + 1$ and, by Eq. (8)-(9), $y_{s,u} = 0$. On the other hand, if all w and z variables are equal to 1, then necessarily, by Eq. (10), $y_{s,u}$ must be 1.

We can also compute the amount of money spent using the same parameters by multiplying $y_{s,u}$ by the amount of each transaction T_u:

$$\forall s \in S: \quad \mathtt{y}_s = \sum_{u=1}^{N} \mathtt{a}_u y_{s,u}, \qquad \text{Constraints (3)-(10).}$$

4 Conclusions and Future Work

In this work we propose an innovative and efficient approach for bank fraud detection, as a first stage of an algorithm that finds new derived attributes that have positive correlation with historical fraudulent events. This problem is of major importance in modern societies both for banks and for all their industrial and individual customers.

We have implemented a first version of this model by restricting the aggregation to the card number field and conditions that filter transactions to one merchant country and one merchant category code. We transform the integer model into a satisfiability problem using the Python package PySAT [3]. Using a database of 2,400 transactions, in less than 5 minutes, the solver finds parameters for which the correlation between the derived attribute and fraud is large enough to be consider as possible predictor of fraud. (We consider that a derived attribute that have a correlation higher than 10% with fraud is useful as a predictor, since, in practice, no derived attribute have a correlation higher than 20%.)

It is not difficult to extend the described model to allow conditions that filter transactions over a set of values instead of single values, that is, extending the condition $\mathbb{g}_u^j = g^j$ to $\mathbb{g}_u^j \in \{g^{j,1}, \ldots, g^{j,n_j}\} \subseteq \mathbb{G}^j$. With this, we will implement an optimization program that finds the parameters that build derived attributes with high fraud correlation, given a fixed length of aggregation τ. By repeating this procedure, with different lengths of aggregation, we can construct a set of derived attributes that, by combining them through logical expression, would detect specific buying behaviors that are present when fraud has been committed.

References

[1] R.J. Bolton and D.J. Hand. Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235–249, 2002.

[2] Banque de France. Rapport annuel 2018 de l'Observatoire de la sécurité des moyens de paiement, 2019.

[3] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[4] S. Jha, M. Guillen, and J.C. Westland. Employing transaction aggregation strategy to detect credit card fraud. *Expert Systems with Applications*, 39(16):12650–12657, 2012.

[5] C. Whitrow, D.J. Hand, P. Juszczak, D. Weston, and N.M. Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009.