FraudMemory: Explainable Memory-Enhanced Sequential Neural Networks for Financial Fraud Detection

Kunlin Yang
Renmin University of China
kunliny@ruc.edu.cn

Wei Xu
Renmin University of China
weixu@ruc.edu.cn

Abstract

The rapid development of electronic financial services brings significant convenience to our daily life. However, it also offers criminals the opportunity to exploit financial systems to do fraudulent transactions. Previous studies on fraud detection only deal with single type transactions and cannot adapt well to evolving environment in reality. In addition, their black box models pay less attention on the interpretability of fraud detection results. Here we propose a novel fraud detection algorithm called FraudMemory. It adopts state-of-art feature representation methods to better depict users and logs with multiple types in financial systems. Our model innovatively uses sequential model to capture the sequential patterns of each transaction and leverages memory networks to improve both the performance and interpretability. Also, with the incorporation of memory components, FraudMemory possesses high adaptability to the existence of concept drift. The empirical study proves that our model is a potential tool for financial fraud detection.

1. Introduction

With migration of traditional business to the Internet and the explosive expansion of e-commerce, electronic financial transactions grow rapidly both in volume and variety [1]. In payment industry, cashless transacting functions become a key field for nowadays banking systems and many new products, e.g. credit cards, online banking, stored-value facilities, come into popularity, bringing convenience to our life.

At the same time, electronic transaction fraud emerges as a serious issue under this background. Over 160 related companies report that annual online fraud is 12 times larger than offline frauds in number [2], resulting severe financial loss to the global economy. With the financial systems becoming more complex, fraudulent actions also turn into more sophisticated forms. It has become a challenging task and an increasing concern to all financial institutions [3]. Consequently, payment service provider usually implement a transaction classification system to alert on suspicious transactions [4], i.e. fraud detection system. In this work, we focus on building a fraud detection model for general-purpose financial systems with multiple operation types, like online banking transaction systems.

Fraud detection is a hot topic among data mining communities [5, 6]. Specific to financial transaction area, credit card fraud detection receives the most scholarly attention [3, 4]. Generally, fraud detection methods can be divided into two aspects, expert-driven or data-driven methods [1]. Expert-driven methods usually apply to expert systems and identify fraudulent transaction by predefined rules under certain scenarios [8, 9]. The rules are concluded by experts and labor-consuming with lack of adaptability to new fraud transaction pattern. Data-driven methods are based on statistical methods or other machine learning algorithms, requiring a large amount of high-quality data to train predictive models. They learn genuine and fraudulent patterns automatically from given training dataset and predict future transaction labels by past patterns. Both supervised and unsupervised learning algorithms are applicable in this field and a lot of methods have been studies, including support vector machine, random forests, hidden Markov model and so on [10, 11, 12]. Among those methods, artificial neural networks and ensemble methods like random forests empirically achieve best performance [13, 14]. However, some accurate data-driven methods tend to be locked in black box, especially neural networks, with insufficient interpretability [15].

Fraudulent transactions usually come with sophisticated characteristics. They are very rare among millions of transactions processed a day and manipulators behind them are carefully organized and well considered [15]. They would study on the principles of target fraud detection systems, especially expert-driven systems, and exploit vulnerabilities to

URI: https://hdl.handle.net/10125/59542
ISBN: 978-0-9981331-2-6

swindle victims. In addition, fraudulent transactions usually finish in a short time, requiring the efficiency of real-time fraud detection systems [3].

Even though recent researches have endeavored to design accurate fraud detection models, there still remain many unresolved problems. First, few studies work on the interpretability of fraud detection systems while this is an important system quality measure [16]. Explanations in results may help improve the transparency of information systems and thus increase users' confidence and trust in the system [17]. Second, patterns of users, whether legitimate or not, are dynamic and evolve with time, which is called concept drift [1]. The transactions patterns are affected by many factors, including consumption and seasonality, while fraudulent manipulators have to frequently change their behaviors to continue on swindling without being spotted [15]. Third, fraud transactions highly depend on the context and fraud detection algorithms are intrinsically based on sequential environment [1]. However, sequential fraud detection is a still little explored area [1, 18]. Last but not least, logs of financial systems are usually multimodal with different attributes set for each operation mode, which contain many discrete data types. Traditional one-hot encoding method is not suitable to handle feature space with high dimensions and sparsity.

In this study, we propose a novel fraud detection model named FraudMemory. It integrate sequential neural networks and memory networks to achieve both high performance and high interpretability. We also bring together traditional feature engineering method in information systems with state-of-the-art machine learning algorithms to construct continuous vector representations of transaction data and user profile. Our proposed method is tested in a real-world dataset with long span of time and shows the superiority in both effectiveness and interpretability.

2. Literature Review

Fraud detection is a field that unifies many essential topics in information systems and data mining. Based on current research status, we first introduce concept drift phenomena, the unsolved problem in fraud detection, and its possible solutions. Sequential learning serves as the core of our model and we then review its recent progress and its application in this field. Finally, we discuss the advantage and methods of continuous feature representations.

2.1. Concept drift

When training a model from data streams, it is often assumed that the data distribution and patterns remains static with time passing. In other words, if a hidden function f_t is used to define the data generator at time t, f_t should not change with time t [19]. Concept drift occurs when f_t changes over time. Because trained model make no attempt to adapt to significant change on data, the existence of concept drift fails to support the model assumption and often results in the decrease in performance [20].

In financial fraud detection, users' behaviors may change owing to a variety of reasons, including income, lifestyle and holiday seasons [21]. Besides, class distribution and volume of data fluctuate with time and new fraudulent pattern might appear. Generally speaking, there are three main ideas to solve concept drift problems, using adaptive learners, constructing newest training data and building ensemble learners [19]. Firstly, adaptive learners use heuristic algorithms to detect contradicting new data and make adaptation based on employed learner [19]. Secondly, newest data construction is widely used by importing a fixed-size time window or time-attenuation weights in order to ensure learners updated [4, 15]. Nevertheless, past information about users is critical to fraud detection systems and only keep recent instances is not a perfect way [22]. Finally, ensemble learners combine learners trained from different batch of transactions on different times to predict the future. This method has been doubted because using history learners indiscriminately is only helpful under constant concept [19].

2.2. Sequential modeling

Sequential modeling is a hot topic in deep learning with a wide application. It aims at extracting sequential relationship and features. Sequential modeling usually adapt a hidden state architecture and has two main types, probabilistic models and non-probabilistic models. Hidden Markov models (HMM) and recurrent neural networks (RNN) are examples of them respectively. Among the methods, RNN and its variants, long short-term memory networks (LSTM) and gated recurrent unit networks (GRU) are proved the state of the art sequential modeling approaches [23].

One of the advantages of sequential models like RNN is that they possess a certain length of memory to help them make predictions. However, their memory size tend to be too small and memorized transactions are not clearly compartmentalized, which make it hard for them to check the memory from the history. In order to incorporate long-term memory into sequential modeling, memory networks are proposed to rectify this problem [24]. Any traditional learner can use memory

networks as its memory component to utilize history data.

Transactional fraud detection is a sequential modeling problem but sequential methods are scarce in this field [1]. HMM was used in credit card fraud detection task [25], while RNN is first introduced in [1]. Sequential modeling is still an unexplored field in fraud detection research.

2.3. Feature representations

Feature engineering has been a long discussed topic in both information systems and machine learning communities. Traditional feature engineering methods in fraud detection usually manually extracted and aggregated features [4, 15]. They achieved better performance than row data, but it is hard to say they were the best feature representations of transactions. On the other hand, machine learning community tends to adopt latent feature representations (i.e. embedding vectors) as input for training. We also focus on this method.

Autoencoders are a popular way to extract latent feature representations [26]. They leveraged multilayer neural networks to learn underlying data distribution, with the object of dimensionality reduction or data compression. Extracted feature representations could help classifier train a better model with less time [27]. Similar to the idea of general autoencoders, word embeddings made a step forward by learning the context of a given feature due to the context dependence of natural language. Two state of the art algorithms were proposed in [28], namely, continuous bag-of-words model (CBOW) and continuous skip-gram model (Skip-gram). CBOW aims at predicting the current word under given context while Skip-gram uses a word to predict the context.

For graph based data, especially for data from knowledge graph or with multiple relations, it is difficult to aggregate global representation of each node while there exist many efficient translation-based embedding methods, like TransE[29], TransR[30]. Those methods follow the same principle $h + r \approx t$, where h and t are the head and tail nodes or entities respectively and r is the relation between them. They help to embed a complex graph into continuous vector feature space with preservation of certain information from the original graph [29]. Translation-based embedding is a powerful tool for preprocess complex graphs and its outcome serve a wide range of applications.

3. Problem Definition

The financial fraud detection problem is defined as follows. In a typical financial information systems with transactional operation, let $\mathcal{U} = \{u_1, u_2 \cdots\}$ and $\mathcal{L} = \{l_1, l_2 \cdots\}$ denote the set of users and their operation logs respectively. For each user $u_i \in \mathcal{U}$, the logs $\mathcal{L}^i \in \mathcal{L}$ are formed up by sessions $\mathcal{S}^i = \{S^i_{t_1}, S^i_{t_2} \cdots\}$, where t_j denotes a timestamp. In any session $S^i_{t_j}$ of u_i, there is a log sequence with at least one log $S^i_{t_j} = \{l^i_{1t_j}, l^i_{2t_j} \cdots\}$ and note that $l^i_{kt_j} \in \mathcal{L}$ and k is the order of the log in a sequence. Each log has the same attributes $\mathcal{A}$ and each attribute has its possible value set $\mathcal{A}_i$. Different logs might have different operation types, like logging in, changing password, transferring etc., and specific operation types depend on financial transactional information systems.

Fraud detection systems accept a stream of sessions, and each session contains a sequence of logs. The goal of it is to use a function $\mathcal{F}$ to label each log as fraud or not fraud, given by

$$\mathcal{F}(l^i_{kt_j}) = \begin{cases} 1 & \text{if } l^i_{kt_j} \text{ is a fraudulent log} \\ 0 & \text{if } l^i_{kt_j} \text{ is a genuine log} \end{cases} \tag{1}$$

From transactional logs, we can easily extract several users transaction graphs $\mathcal{G}_r$, which consists of massive user-relation-user triples (u_i, r, u_j). Here $u_i, u_j \in \mathcal{U}$ and r denote payer, payee and a certain attribute in related transaction log respectively. In particular, transaction graphs are only made up of logs in transaction operation and the structures of each graph is the same except the relation definition.

4. Methodology

This fraud detection methodology is designed to detect transactional fraud in common financial systems. Our proposed algorithm, FraudMemory, is a general fraud detection algorithm and can be easily incorporated to financial information systems through customized interface. We first introduce the framework of FraudMemory and then discuss the detailed implementation of feature representation and fraud detection algorithm in the following sections.

4.1. Framework

Figure 4.1 illustrates the process of training and application of our proposed method. Time flows from left to right, from up to down. During training time, transactional logs are extracted from logs database for

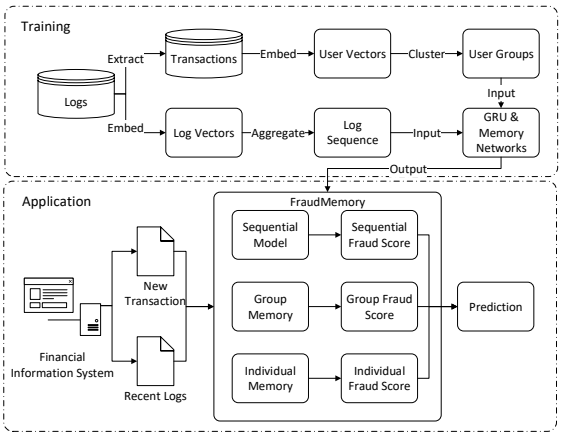

Figure 1. Training and application of FraudMemory

construction of transaction graphs in order to embed user representation vectors. Users are then clustered into groups based on user vectors. On the other hand, each log in logs database is used to embed log vectors at the attribute level. We aggregate log vectors to form equal-length sequences to be fed into the learner composed of GRU and memory networks along with user groups.

In application, the financial information systems input new transactions into FraudMemory through predefined interface. At the same time, recent logs of the same user are automatically extracted from logs database and transformed through embedding models to form a sequence with new transaction. FraudMemory process this sequence via its sequential model and memory components. It analyzes the sequential pattern to generate a sequential fraud score and uses the sequence to access group and individual memories with outcome as group and individual fraud scores. The final prediction is made based on the three scores.

4.2. Feature Representation

4.2.1. User profile representation In light of the RFM model - Recency, Frequency, Money - described in [15],we use the three attributes to derive our transactional graphs based. We totally construct three graphs, recency graph $\mathcal{G}_R$, frequency graph $\mathcal{G}_F$ and money graph $\mathcal{G}_M$, and their weights are defined by the average time interval between transactions, the frequency of transactions and the average monetary value in each transaction respectively. All the nodes and edges in the three graphs are the same but weights of edges vary from each other.

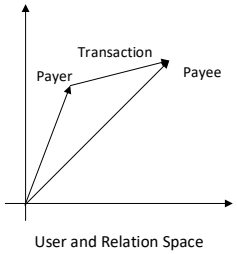

Figure 2. Illustration of TransE

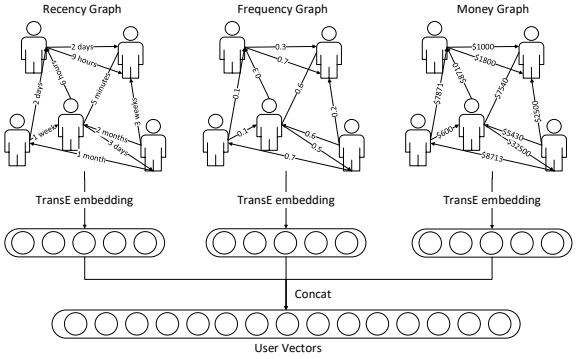

Figure 3. Illustration of user vector construction

We borrow the idea from knowledge graph embedding methods to extract latent vectors in our graphs and TransE is adopted as the final approach [29]. Given the fact that transactional graphs do not possess complex relations as knowledge graph, we manually divide weights into several levels and take weight levels as the relations. For example, monetary relation can be set by the power to 10 (e.g. \$124 is relation 2 and \$32123 is relation 5). Thus, the transacional graphs are similar to knowledge graphs in structure.

In TransE model, users and relations are assumed as vectors under the same place, defined by $\mathbb{R}^d$, where d is the dimension. The target is, for each transaction (u_i, r, u_j), the formulation $u_i + r \approx u_j$ holds. Figure 4.2.1 offers a simple illustration and the score function of TransE is

$$f_r(u_i, u_j) = \|\mathbf{u_i} + \mathbf{r} - \mathbf{u_j}\|_\mathbf{2}^\mathbf{2} \tag{2}$$

From each graph, we derive a latent vector with dim d for each user following [29]. In order to describe user profile in details, we concatenate the three vectors into one user vector with dim $3d$ in represent for the RFM attribute of users. Till now, we have user vectors of all users and then perform a clustering algorithm (e.g. K-Nearest Neighbors) to group the users. Each user is

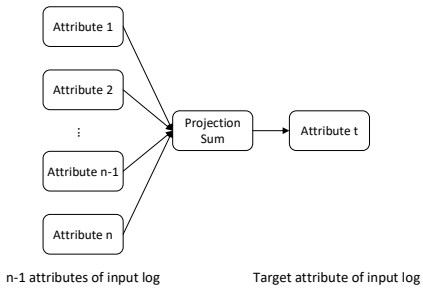

Figure 4. CBOA architecture

now in one group, and under the same groups, users behave similarly to each other.

4.2.2. Log representation Out of the consideration of fine-grained feature representation, we want the log vector not only contains log latent information, but also can be presented in attribute-level as in real life. Traditional autoencoders is unsuitable for this task since they mix up all the information. Finally, we modify the continuous bag-of-words model (CBOW) [28] and design our log representation learner called continuous bag-of-attributes model (CBOA). CBOW originates in natural language processing, while in CBOA, attributes serve as the words in CBOW. The architecture of CBOA is given by Figure 4.2.2.

Unlike the length of sentences in CBOW, the length of sequences in CBOA is fixed, which is equal to the attributes of each log (e.g. operation, time, value, location, channel). Because each operation has different meaning, the corresponding log attributes might vary. For example, for logging in operation, the value attribute is always zero. For categorical attributes, we directly use the value while we manually convert continuous attributes to categorical attributes by dividing the value to different levels.

For an incoming log, we convert it to a sequence of a_{ij}, where $i \in \mathcal{A}$ and $j \in \mathcal{A}_i$ denote the attribute and its value respectively. Note that the order of a_{ij} in the sequence does not matter and for any attribute a_{ij}, all the other a_{mn} $(m \in \mathcal{A}\, and\, m \neq i)$ serve as the context for a_{ij}. The goal of CBOA is using the context attributes to predict the target attribute, namely, maximizing the log probability

$$\log p(a_{ij}|\{a_{mn}|m \in \mathcal{A} \text{ and } m \neq i \text{ and } n \in \mathcal{A}_m\}) \quad (3)$$

Thus, by following [31], we not only obtain the log vectors, but also derive attribute vectors.

4.3. Fraud Detection Algorithm

4.3.1. Sequential fraud detection RNN is a powerful tool to learn sequential pattern, but when dealing with long sequence, it suffers from "vanishing gradients" problems, resulting in poor performance. Two variants of RNN are thus invented, Long Short-Term Memory (LSTM) and Gated Recurrent Unit networks (GRU), while we choose GRU as our sequential model since it has less parameters with similar performance comparing to LSTM for efficiency reasons [32].

As defined before, the log sequences are formed up by session. Since the length of sessions is not necessarily the same, we set the sequence length of GRU to t, and directly select the most recent t logs, which might all come from the same session or adjacent sessions.

Given the log sequence $\{l_1^u, l_2^u \cdots l_t^u\}$ of user u, where l_t^u is the one to be classified, GRU derive the current hidden state vector h_t^u based on previous hidden state vector h_{t-1}^u by

$$h_t^u = GRU(h_{t-1}^u, l_i^u; \Theta) \quad (4)$$

where Θ denotes the GRU parameters to learn and the computation of Θ follows the method in [32]. It generates the sequential fraudulent score $Score_{sequence}$ based on the last sequence representation h_t^u by using a multiple-layer perception (MLP) to perform nonlinear transformation

$$Score_{sequence} = MLP(h_t^u) \quad (5)$$

The fraudulent score is used to classify transactions.

4.3.2. Memory-enhanced fraud detection To leverage the history logs in our framework, we use memory networks to grant our framework with "memory" to increase performance and solve the concept drift problem. Memory networks have many slots to store past information, which highly enlarge the memory of traditional classifier including RNN and enable classifier to accurately remember information from the past. Slightly different from [24], our memory-enhanced networks model have two operations, $Read$ and $Update$, representing reading from existing memory and updating existing memory with new logs. It accept the sequential hidden state vector as an input and output a memory fraudulent score.

$Read$ operation regards the input vector as a query and use the query to ask memory slots M_u in order to

find out if this query can jog similar past memory or at least not contradict to past behaviors. The abstractive form of read operation is given by

$$m_t^u = Read(\{m_1^u, m_2^u \cdots\}, \tilde{h_t^u}) \tag{6}$$

where $m_1^u, m_2^u \cdots \in M^u$, m_t^u is the output latent memory vector recalled by memory networks. The interaction between query vector and memory slots is carried out by using Euclidean distance $Distance$ function to calculate the dissimilarity of them. Similar to human thinking process, the result of brain does not source from only one piece of memory but a mixture of multiple factors. Consequently, we adopt the attentive combination of memory networks

$$m_t^u = \sum_{i=1}^{M^u} \omega_i^u \cdot m_i^u \tag{7}$$

where ω_i^u is the attention weight of memory m_i^u. We expect the more similar the query and the memory, the higher the weight is, so we use the $Softmax$ function to compute the weights:

$$\begin{aligned} \omega_i^u &= Softmax(-Distance(\tilde{h_t^u}, m_i^u)) \\ &= \frac{e^{-Distance(\tilde{h_t^u}, m_i^u)}}{\sum_j^{M^u} e^{-Distance(\tilde{h_t^u}, m_j^u)}} \end{aligned} \tag{8}$$

It should be noted that the $Distance$ function does not compute the distance directly by the whole vector, because the memory vector consists of multiple attributes. Instead, in order to discriminate the influence of different attributes, $Distance$ function calculate the distance by attributes, given by

$$Distance(\tilde{h_t^u}, m_i^u) = \sum_{a=1} \|\tilde{h_{t,a}^u} - m_{i,a}^u\|_2^2 \tag{9}$$

where a denotes the ranking of attributes and $\tilde{h_{t,a}^u}$ and $m_{i,a}^u$ denote the elements of a in $\tilde{h_t^u}$ and m_i^u.

For $Update$ operation, it is a cognitive process that new information affects existing memory, so each memory slot will possibly be influenced by incoming query. In order to reduce computation cost, we heuristically choose to not update the memory if the smallest distance between the query and slots is smaller than a threshold θ. When updating, we can either update the slots using the query or fill a new memory slot with it if there are empty slots. The updating strategy is given by

$$\{m_1^u, m_2^u \cdots\}^{updated} = Update(\{m_1^u, m_2^u \cdots\}, \tilde{h_t^u}) \tag{10}$$

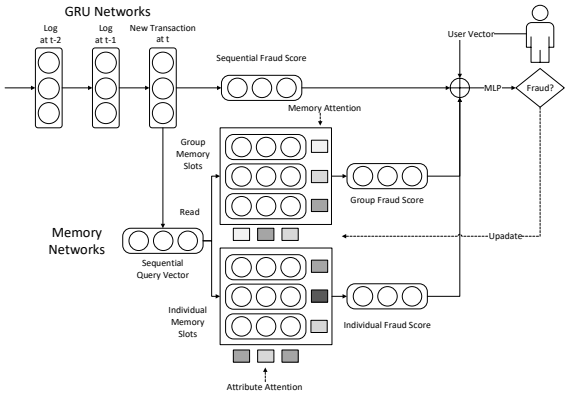

Figure 5. Illustration of FraudMemory process

Out of the same consideration of $Read$, we use $Distance$ to derive how much to update for each slot i by using a gate wight z_i, computed by

$$z_i = Sigmoid(Distance(\tilde{h_t^u}, m_i^u)) \tag{11}$$

where $Sigmoid$ function is in the form

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

Thus, each memory slot i is updated as follows

$$m_i^u = z_i \cdot m_i^u + (1 - z_i) \cdot \tilde{h_t^u} \tag{13}$$

As long as the new transaction is "impressive" enough, it would be stored into the memory of our model and might have potential influence to future fraud detection process. Particularly, the incoming transaction might be genuine or fraudulent, so the genuine and fraudulent memory are stored separately in memory slots. Thus, no matter whether the fraud patterns are cyclic or evolving, there are always clues in the memory. In practice, each user possesses two different sets of memory, individual and group memory, and details would be discussed in the following subsection.

4.3.3. Complete fraud detection information system The complete FraudMemory is a hybrid of sequential and memory-enhanced fraud detection methods. The process is illustrated in Figure 4.3.3. Given a log sequence $\{l_1^u, l_2^u \cdots l_t^u\}$ of user u, the GRU networks are fed the latent log vectors of the sequence and output a sequential hidden vector of the last transaction $\tilde{h_t^u}$. This sequential vector is first used as the query vector in memory networks. Considering

that a user memory may possess large variation for relatively small amount of logs, we adopt two different memory sets, individual and group memory. Each group has a set of group memory and each user has a set of individual memory. Note that the group memory are shared by all the users under the group (i.e. group members have the reading and updating authority to the group memory) while individual memory is private. Except that, the two memory networks are functionally the same.

The GRU networks and memory networks totally output three fraudulent vectors, $\tilde{h_t^u}$, m_t^u and m_t^g from GRU networks, group memory and individual memory respectively. We concatenate the three vectors and user vector u together and use a non-linear transformation MLP to derive the final fraudulent score.

$$Score = MLP(Concat(\tilde{h_t^u}, m_t^u, m_t^g, u)) \quad (14)$$

In practice, we manually assign each memory slot and each attribute in one memory slot attention weights, which is shown as colored square in Figure 4.3.3 and darker color means higher attention. The memory attention and attribute attention are computed in Equation (8) and (9) and the memory slot with highest attention weights would possess attribute attention. That is to say every memory slot has a memory attention weight but only one slot has attribute attention weight. Those attention weights are extremely useful for interpretation on predicting results. The weight value is in proportion to the suspicious degree and higher attribute attention weight value contributes more to the fraudulent score. Thus, the result of FraudMemory is highly explainable with the help of those attention weights.

5. Empirical Study

In this section, we introduce the setup of our experiments firstly, and then present our empirical results. Detailed discussion is given based on the results.

5.1. Experimental setup

5.1.1. Dataset description We obtain the labelled dataset from a collaborating Chinese online banking system. It consists of complete usage history of 35,766 core users from March, 2013 to November, 2013 with around 8.9 million logs, with 4.2 million transactional logs and 13,827 fraud transactions. There are 54 types of operation, 12 of which are related to transaction including inter-bank transfe and mobile bank transfer etc and 26 of which involve sensitive information like password change and linked mobile phone change etc. Those transactions are labeled by both the existing expert system and risk management departments.

The original dataset is too large to build a model so we first eliminate users who registered during the dataset time range and from the remaining users, then extract those who suffered from or involved in fraud as seeds to find out all the users that have transactions with them as a subset. We also add users who actively participate in the transactions of the subset. Logs in unimportant sessions, like those that only contain logging in or have payment to trusted account (e.g. payment for electricity bill), are removed to reduce the data amount. Till now, there are 21,032 core users and 3.8 million logs and we only randomly select logs in a certain time window for training and testing. Given the fact that transactional dataset and our model is time-aware (i.e. future data cannot be used to predict the past), cross validation is unsuitable. Instead, we divide the time span into several parts, and use the current time window to predict the following time window. In our experiment, we set the time span to one month and there are 8 times validation.

5.1.2. Evaluation metrics We adopt the following three metrics: recall, precision and area under ROC curve (AUC). Recall is an important metric in fraud detection and represents the percentage of fraudulent transactions detected to all fraudulent transactions. Precision measure the percentage of truly-fraudulent transactions to all transactions classified as fraud. AUC is considered the measure of overall performance.

5.1.3. Baselines We use three traditional classifier as our baselines, support vector machine (SVM), deep neural networks(DNN) and random forests (RF). They are traditional machine learning algorithms widely used in financial risk detection fields and among them, RF is considered as the state-of-art fraud detection algorithm [15, 1]. However, since they are not sequential model, the input of them is the concatenation of user and log vectors. We also adopt pure GRU sequential fraud detection model (GRU) and FraudMemory with only group memory (GRU+Mem) as our baselines. Our model is denoted by FraudMemory.

5.1.4. Parameter setting For user profile representation, we set the dimension of latent vectors in each graph to 30 and thus the dimension of user vectors is 90. The number of groups is 65. For log representation, we set the dimension of each attribute to 6 and the number of attributes in each log is 12,

Table 1. Results comparison of different methods with default cut-off and cut-off that make the maximum false positive rate at 1%. "†" indicates that the 8 times validation results of FraudMemory significantly surpass the baseline at the significance level of 0.01 (Only applicable for default).

Methods	Default			At 1% maximum positive rate		
	Recall	Precision	AUC	Recall	Precision	AUC
SVM	0.507†	0.801†	0.691†	0.162†	0.922†	0.577†
DNN	0.671†	0.873†	0.810†	0.571†	0.941†	0.744†
RF	0.619†	0.834†	0.737†	0.415†	0.938†	0.672†
GRU	0.788†	0.918†	0.900†	0.709†	0.957†	0.853†
GRU+Mem	0.853†	0.938†	0.942†	0.785†	0.982	0.891†
FraudMemory	**0.874**	**0.968**	**0.969**	**0.843**	**0.984**	**0.917**

including operation type, amount, channel, location, hour, day in a week, month etc. So the dimension of log vector is 72. As for the GRU model, we adopt one-layer GRU network with hidden layer size of 72. The number of slots in individual memory network is 8 and that in group memory network is 16.

5.2. Results and analysis

The results of our model and other different baselines are given in Table 1. Except for the basic fraud detection experiment, we did an additional experiment to study the case when the false positive rate is at most 1% following [15]. Because in practice, not only high recall and accuracy are pursued by risk management department, low false positive rate is also an important metrics. A false positive case indicates that a genuine transaction is classified as fraud and financial system would automatically reject the transaction, adding unnecessary trouble to customers and reducing customers satisfactory. Out of this consideration, we set the false positive rate to at most 1% by tuning cut-off of models, which is an acceptable number in practice.

From the results, it can be observed that:

(1) The results show a wide variation between traditional classifiers and sequential models. Among traditional classifiers, SVM performs the worst while DNN outperforms RF. This might because the latent feature representations in our experiments are more suitable for neural networks model rather than RF. However, there exist a significant gap between best traditional classifier model DNN and the worst sequential model GRU. This partly because we don't explicitly aggregate history transactions in our dataset, even though some history information is already embedded into user vectors. On the other hand, the results also indicate that the superiority of sequential models under fraud detection context.

(2) Among the sequential models, models with memory all outperform the single sequential model GRU. The usage of memory networks augment the

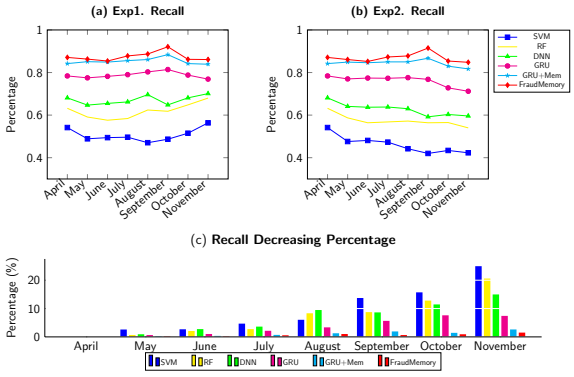

Figure 6. Analysis on concept drift

performance significantly. In addition, we observe that the use of individual memory (FraudMemory) only slightly improves the result than that with only group memory (GRU+Mem), but it is still significantly better. This can be explained by the fact that group memory are more complete and have contained most of the important information. Individual memory, however, is a specific subset of the group memory and it makes sense that its improvement is not that large.

(3) Speaking of false positive rates, we find that the changes in traditional classifier are more marked than those in sequential classifier. It can also be inferred that models with memory vary less when changing the false positive rate, and more memory indicates less variation. It helps prove the robustness of our proposed method.

(4) Finally, it is obviously that the proposed model FraudMemory performs consistently better than other baselines by a large margin. FraudMemory is the combination of GRU and memory networks and it not only captures the pattern in recent logs, but also finely balance the prediction based on group and individual memory. Another two potential merits of FraudMemory is its robustness to concept drift and high interpretability, which are discussed in the following two subsections.

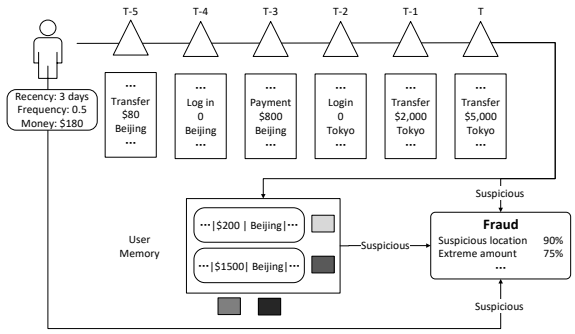

Figure 7. A fraud detection example

5.3. Detailed analysis on concept drift

In order to find out the adaptability of our proposed model and baselines to concept drift problem, we design a comparison experiment composed of two sub-experiment. The first (Exp1) is the same with the one in the previous section, using logs in current time window to predict transactions in the following time window. In the second one (Exp2), models are only trained by logs in the first time window and transactions in all the following time windows are used for testing, in order to evaluate the performance changes with time evolving.

We use evaluation metric recall as example and the results are shown in 5.2. 5.2.(a) is the result for Exp1, where models are tested on data with little concept drift (we assume transactions in adjacent months are similar). 5.2.(b) is the result for Exp2, where models are tested on data with concept drift and the degree of concept drift increases with time. It can be observed from the line graph that as time evolves, the performance of models declines in various degree and this phenomena is extremely significant in the last month.

Furthermore, to quantitatively study the effect of concept drift, we draw a bar graph in 5.2.(c), showing how much performance decrease. We can clearly find that at first, the concept drift is not significant and models in Exp2 perform similar as they do in Exp1. However, at the last month, which is only eight month after, the most affected model has around 25% decrease in performance. Other models without memory also have at least 8% decrease. On the other hand, for models with memory, they are affected less by concept drift. FraudMemory, in particular, the decrease is even less than 2%, which demonstrate the high adaptability of our model.

5.4. Detailed analysis on interpretability

We have mentioned that FraudMemory has high interpretability and we present a visualization in Figure 5.3. Since the latent embedding is too abstractive to show, here we use the original attribute instead of embedding vectors to be more intuitive.

The user usually always use the account in Beijing and all his memory is within Beijing. However, this session, he suddenly transfer large amount of money twice in Tokyo. The sequential model deem it as suspicious. At the same time, this transaction does not ”recall” anything in the memory, and the location and amount high distracted from his behavior (the location get the highest weight and the amount get the second). In addition, this transaction does not conform to the user vector, which reflect the RFM attributes of the user. Here we omit group memory for simplicity. Consequently, the system gives fraud alert and relative reasons.

Note that this is a highly simplified example and the actual mechanism is much more complex. But it does not affect its ability to offer possible justification.

6. Conclusion

In this paper, we proposed a memory-enhanced sequential neural model with high interpretability to detect financial fraud. Our model is suitable for financial systems with complex operation types, in which users and logs are objectively and accurately depicted. Being endowed with the benefits of sequential and memory components, our model possess high performance with low false positive rate and adaptability to concept drift. Unlike other black-box model, every part of our model is explainable, whether in feature embedding or prediction results. Nonetheless, our model still has some drawbacks like the cold start problem since our model required detailed description of users and that will be our future improvements.

References

[1] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, “Sequence classification for credit-card fraud detection,” *Expert Systems with Applications*, vol. 100, pp. 234–245, 2018.

[2] L. Zheng, G. Liu, W. Luan, Z. Li, Y. Zhang, C. Yan, and C. Jiang, “A new credit card fraud detecting method based on behavior certificate,” in *Networking, Sensing and Control (ICNSC), 2018 IEEE 15th International Conference on*, pp. 1–6, IEEE, 2018.

[3] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, “Effective detection of sophisticated online banking fraud on extremely imbalanced data,” *World Wide Web*, vol. 16, no. 4, pp. 449–475, 2013.

[4] 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*, vol. 18, no. 1, pp. 30–55, 2009.

[5] J. L. L. Herrera, H. V. R. Figueroa, and E. J. R. Ramírez, "Deep fraud. a fraud intention recognition framework in public transport context using a deep-learning approach," in *Electronics, Communications and Computers (CONIELECOMP), 2018 International Conference on*, pp. 118–125, IEEE, 2018.

[6] N. Carneiro, G. Figueira, and M. Costa, "A data mining based system for credit-card fraud detection in e-tail," *Decision Support Systems*, vol. 95, pp. 91–101, 2017.

[7] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, "Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization," *International Journal of Data Science and Analytics*, pp. 1–16, 2018.

[8] D. Sánchez, M. Vila, L. Cerda, and J.-M. Serrano, "Association rules applied to credit card fraud detection," *Expert systems with applications*, vol. 36, no. 2, pp. 3630–3640, 2009.

[9] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, "Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning," *Information Fusion*, vol. 10, no. 4, pp. 354–363, 2009.

[10] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[11] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, 2016.

[12] E. Ngai, Y. Hu, Y. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Systems*, vol. 50, no. 3, pp. 559–569, 2011.

[13] J. R. Dorronsoro, F. Ginel, C. Sgnchez, and C. Cruz, "Neural fraud detection in credit card operations," *IEEE transactions on neural networks*, vol. 8, no. 4, pp. 827–834, 1997.

[14] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," *Expert systems with applications*, vol. 41, no. 10, pp. 4915–4928, 2014.

[15] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015.

[16] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang, "Aimq: a methodology for information quality assessment," *Information & management*, vol. 40, no. 2, pp. 133–146, 2002.

[17] N. Tintarev and J. Masthoff, "A survey of explanations in recommender systems," in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pp. 801–810, IEEE, 2007.

[18] B. Wiese and C. Omlin, "Credit card transactions, fraud detection, and machine learning: Modelling time with lstm recurrent neural networks," in *Innovations in neural information paradigms and applications*, pp. 231–268, Springer, 2009.

[19] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: an overview," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 89–101, 2012.

[20] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[21] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection and concept-drift adaptation with delayed supervised information," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, pp. 1–8, IEEE, 2015.

[22] D. Malekian and M. R. Hashemi, "An adaptive profile based fraud detection framework for handling concept drift," in *Information Security and Cryptology (ISCISC), 2013 10th International ISC Conference on*, pp. 1–6, IEEE, 2013.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

[24] S. Sukhbaatar, J. Weston, R. Fergus, *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, pp. 2440–2448, 2015.

[25] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, "Credit card fraud detection using hidden markov model," *IEEE Transactions on dependable and secure computing*, vol. 5, no. 1, pp. 37–48, 2008.

[26] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, pp. 153–160, 2007.

[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *International Conference on Neural Information Processing Systems*, pp. 2787–2795, 2013.

[30] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion.," in *AAAI*, vol. 15, pp. 2181–2187, 2015.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[32] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.