LAW: Learning Automatic Windows for Online Payment Fraud Detection

Cheng Wang, *Senior Member, IEEE*, Changqi Wang, Hangyu Zhu, and Jipeng Cui

Abstract—The rapid development of internet finance has caused increasing concern in online payment fraud due to its great threat. It is typical to employ rule systems or machine learning-based techniques to detect frauds. For the most significant features of such fraudulent transactions are exhibited in a sequential form, the sliding time window is a widely-recognized effective tool for this problem. With a sliding time window, features about the transaction characteristics can be extracted, and the latent patterns hidden in transaction records can be captured. However, the adaptive setting of sliding time window is really a big challenge, since the transaction patterns in real-life application scenarios are often too elusive to be captured. As a matter of fact, the practical setting usually needs to be updated and refined with manual intervention regularly. This is time-consuming indeed. In this work, we pursue an adaptive learning approach to detect fraudulent online payment transactions with automatic sliding time windows. Accordingly, we make efforts on optimizing the setting of windows and improving the adaptability. We design an intelligent window, called *learning automatic window* (*LAW*). It utilizes the learning automata to learn the proper parameters of time windows and adjust them dynamically and regularly according to the variation and oscillation of fraudulent transaction patterns. By the experiments over a real-world dataset of the online payment service from a commercial bank, we validate the gain of *LAW* in terms of detection effectiveness and robustness. To the best of our knowledge, this is the first work to make a sliding time window for fraud detection capable of learning its proper size in changing situations.

Index Terms—Fraud Detection, Online Payment, Learning Automata, Sliding Time Window

1 Introduction

With the emergence of fast and convenient e-commerce systems, online payment has become a hot topic and people tend to shop and trade online more than ever before. This popularity results in a large amount of electronic transaction data, accompanied with a rapid increase in the number of online payment frauds. Obtaining a person's account information or actual credit card is a common occurrence under the era background of information explosion and disclosure [1]. Actually, identity theft [2] and account takeover (ATO) [3] are the most popular cybercrime activities in online payment services, and may somehow explain the continuous occurrences of fraudulent transactions. This is not only a threat to thousands of customers [4], but also a challenge to the security of the entire e-commerce system [5], [6]. More efficient and comprehensive online payment fraud detection systems are therefore in dire need.

In the online payment scenario, the occurrence of a transaction produces a transaction record, which contains amounts of information. The objective of fraud detection is to make a distinction between the legitimate and fraudulent transactions with these scattered transaction information fields. For example, if a customer buys something or makes a transaction that exceeds his/her trading amount limit at an unusual time, this trade operation could be detected and intercepted as an exception. Afterwards, the customer will be alerted that his/her account is under risk. To this end, it is useful to figure out the latent anomaly patterns hidden in the user's recent transaction records.

In the real world, fraudulent transactions will appear in clusters [7], [8]. In other words, there are repeated occurrences of similar fraudulent transactions within a short range of time. It gives some hints that normal and stolen accounts have different trading behavior patterns. To capture and represent the behavior patterns, we propose a concept of the sliding time window for each customer. Through the statistical analysis of trading characteristics within a time window, such as the number of transactions, the cumulative value of transaction amount, the mean and variance of the transaction amount difference, and the time interval, we can generate new window-relevant features to represent the behavior patterns. Then, we can use them for fraud detection. The different sizes of a time window might generate different features and then result in totally different performance of models. How to determine a proper size of the time window is an intractable problem to solve. It is straightforward to exhaustively test each possible time window size, but this method is quite labor intensive and time-consuming [9]. We innovatively propose to utilize learning automata [10] to solve this problem. With learning automata, our method is capable of selecting the most suitable time window size automatically in a limited time.

In the financial industry, machine learning models used in risk control systems are required to be sufficiently robust [11]. Meanwhile, this is a typical limitation of most rule-based systems [12] and other traditional

- *Cheng Wang (corresponding author), Changqi Wang, Hangyu Zhu, and Jipeng Cui are with the Department of Computer Science and Engineering, Tongji University, and with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, China. (E-mail: {cwang, wcq1051068488, 1830823, 1410495}@tongji.edu.cn)*

machine learning-based techniques [13], [14], [15]. Once new fraud attack methods turn up, the performance of current models tend to decline, as the previous model variables and features portraying the fraud behaviour patterns already become obsolete. It is difficult for the fixed time window size to adapt to the variability of fraud attack in this situation. Therefore, with the regular verification and feedback of fraudulent labels, the changing patterns of fraud attacks should be learned regularly to identify the risks more accurately and enhance the robustness of models. Another factor that affects the performance of fixed time window is the so-called *concept drift* [16], [17], [18]. In the dynamic online payment environments, data distribution changes over time. It yields the phenomenon of concept drift that has little association with fraud [19]. When the statistical properties of target variables change over time in an unpredictable way, the prediction accuracy of machine learning models will decrease correspondingly. For example, customers' seasonally changed shopping behaviors will bring down the performance of a risk prediction model built on the fixed time window, making it work only intermittently. So instead of a fixed time window, we need a dynamic one, which can be adapted to the environments regularly and thus brings the effectiveness and robustness to risk prediction models.

In this paper, we propose an online payment fraud detection method based on the devised *Learning Automatic Window*. We call our method *LAW* for simplicity. By utilizing learning automata, we can optimize the size of the sliding time window automatically. We extract 14 new features from 8 raw fields of transaction data. Among these new features, 8 of them are related to the sliding time window, hence are called window-dependent features. With all generated features, we adopt a selected classifier to implement the risk prediction model. The extensive experiments over real-life data validate the performance gain of our *LAW* in terms of detection efficiency and effectiveness. Particularly, with the help of a regular online updating scheme, the sliding time window in our method can be adapted to the dynamic environment itself regularly. This effectively ensures the good robustness of *LAW*.

The rest of this paper is organized as follows. Section 2 gives a review of the relevant studies. Section 3 presents our proposed method. The experiments and result analysis are given in Section 4. Finally, we conclude our work in Section 5.

2 RELATED WORK

Fraud detection is highly related to anomaly detection. Traditional methods involve the extensive use of auditing where fraudulent behaviors are manually observed and reported [20]. This paradigm is not only time consuming, expensive and inaccurate, but also impractical in the big data era. Not surprisingly, financial institutions have turned to automatic detection techniques based on statistical and computational methods. Here, we review recent and fruitful studies on fraud detection, especially on online payment fraud.

Traditional rule-based fraud detection systems usually have a single function and limited detection effect. Recently, many advanced association rule-based fraud detection systems [21], [22], [23] have shown great results. Supervised learning-based methods are popular for credit card fraud detection purposes, such as Logistic Regression [24], [25], Bayesian-based model [26], [27], Random Forest [28], [29], [30] and Neural Networks [16], [31], [32], [33]. Some semi-supervised and graph-based fraud detection methods [34], [35], [36] are effective in the context of weak labels. They mined the relationships of users through the transaction graph to detect fraud. Meanwhile, some unsupervised methods [37], [38] were exploited for fraud detection. They intended to identify hidden structures in unlabeled transaction data. The behavioral models [39], [40], [41], [42] are also increasingly applied to solve the issue of fraud detection, which can dig the user's trading behavior patterns and identify the abnormal trading patterns.

Next, we review some fraud detection methods based on either the model integration and optimization or feature selection. Utkarsh et al. [43] assigned a consistency score to each data point, and they used an ensemble of clustering methods to detect outliers in large datasets. Sahil et al. [44] employed different supervised machine learning algorithms to detect fraudulent transactions of the credit card on a real-world dataset and used these algorithms to implement a super classifier by ensemble learning. Sohony et al. [45] presented an ensemble method based on a combination of Random Forest and Neural Network. Due to the advantages of ensemble learning, their method achieved high accuracy and confidence for label prediction of new samples. Alejandro et al. [28] expanded a transaction aggregation strategy, and proposed to create a new set of features based on the analysis of periodic behaviors of a transaction using the von Mises distribution.

Our work focuses on a unique aspect of feature optimization, that is, generating the optimal feature sets by optimizing the size of a sliding time window. There have been enlightening studies on the issue of window optimization in many other fields besides. Liono et al. [46] pursued the optimal window size by optimizing multi-objective function to balance between the minimising impurity and maximising class separability in temporal segments. It was utilized to recognize multiple activities from heterogeneous sensor streams. Inspired by them, our method selects the window size by optimizing a different objective function. Shibli et al. [47] proposed a novel empirical model that can adaptively adjust the window parameters for a narrow band-signal using spectrum sensing technique. The appropriate window size was selected where two closest sinusoids can be distinguished using a specific formula. It can not only improve the spectrogram visualization but also effec-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing

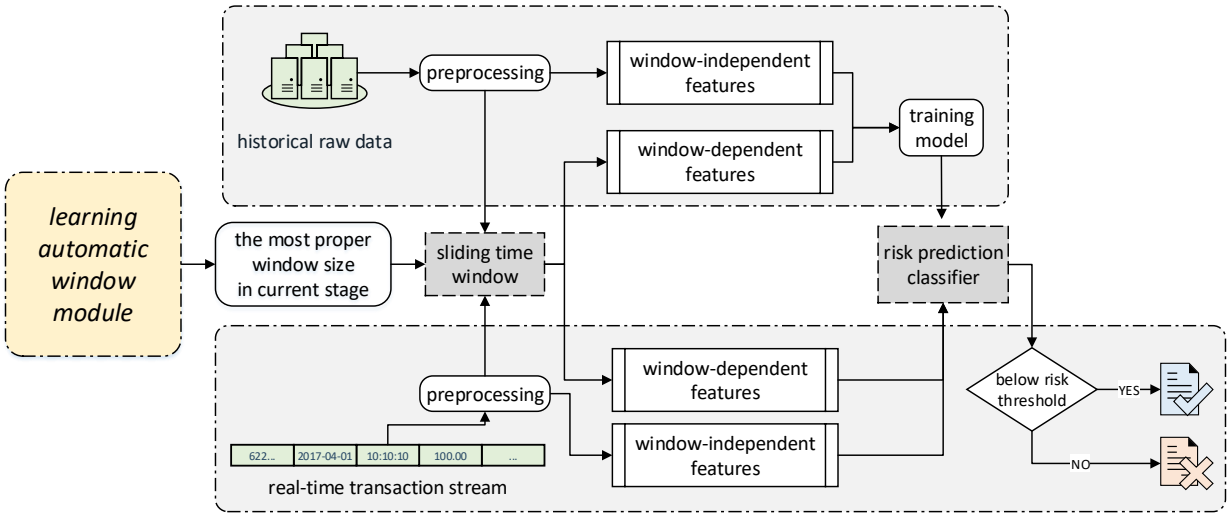


Fig. 1. Workflow of the fraud detection system. It contains three main modules: the automatic window size generation module, the risk prediction model training module, and the real-time streaming data processing module.

tively reduce the computation cost. Although both of them were devoted to determining the optimal window size, they are hard to directly apply for our work since the inconsistency of optimization objective in different application scenarios.

Different from these window optimization methods introduced above, we obtain the proper time window size with the help of learning automata (LA). Learning automata intends to learn the optimal action from a set of allowable actions. It operates through maximizing the probability of being rewarded based on interaction with a random environment. We now review the latest researches on LA and its applications. Seyyed et al. [48] proposed to combine wrapper and filter ideas, and used estimator learning automata to efficiently determine a feature subset. The subset was chosen to satisfy a desirable tradeoff between the accuracy and efficiency of the learning algorithm. Liu et al. [49] designed a method that combined the firefly algorithm and LA to select optimal features for motor imagery EEG. Erik et al. [50] explored the use of learning automata algorithm to compute threshold selection for image segmentation. And the algorithm possessed the ability to perform the automatic multi-threshold selection. Zhang et al. [51] proposed a reverse philosophy called last-position elimination-based learning automata. The action graded last in terms of the estimated performance was penalized by decreasing its state probability and was eliminated when its state probability became zero. The proposed schemes can achieve significantly faster convergence and higher accuracy than the classical ones. Inspired by these LA applications, we utilize LA to optimize the time window size for generating the set of suitable features.

3 THE PROPOSED METHOD

In this section, we present our method for online payment transaction fraud detection in the real-life online

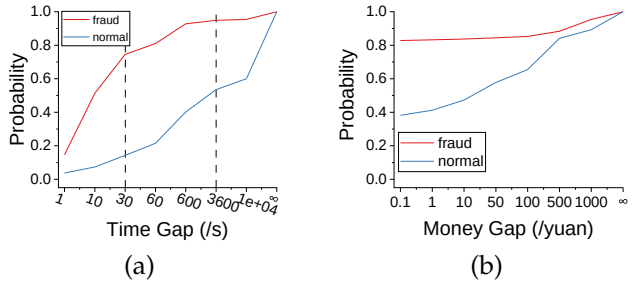


Fig. 2. The respective cumulative distribution functions of time gap and money gap between two adjacent transactions.

payment scenarios. The whole workflow of the fraud detection system is depicted in Fig. 1. We first elaborate on the features that are used to distinguish legitimate and fraudulent transactions. Then, we introduce the design of sliding time window. Finally, we describe how to use learning automata to generate a dynamic window adaptively. The devised *Learning Automatic Window (LAW)* is the core of our method. This is the reason why we call the proposed method *LAW*.

3.1 Feature Preprocessing

First of all, we need to preprocess the original transaction data. Upon doing so, we can obtain powerful features for learning models. Many of the original data fields do not significantly contribute to fraud detection, such as the *media serial number*, *CPU number*, *operating system version*, *registration time*, and so on. Therefore, we select and process some raw fields with high correlation with fraud detection. The selected raw fields of data are summarized in Table 1.

We analyze the transaction dataset, which will be introduced in Section 4.1 later, by counting the time gap (in units of second) and money gap (in units of Chi-

Table 1: Descriptions of the raw data fields

Feature	Data Type	Description
User_ID	String	Transaction card account of customers, one account represents one User_ID
Transaction_Time	String	The time that transaction happens, starting from the year, accurate to second
Transaction_Amount	Float	The amount of money in a transaction, in units of yuan, the basic unit of Chinese currency (RMB).
Pre-trade_Account_Balance	Float	Balance of the account before the transaction happens, in units of yuan
Daily_Limit	Float	Maximum amount limit for daily transactions, in units of yuan
Single_Limit	Float	Maximum amount limit for a single transaction, in units of yuan
Check	String	The verification tool at the process of the transaction, like U-shield, electronic cipher, etc.
Frequent_IP	Boolean	A status bit indicating whether the current IP is consistent with the IP frequently used by customers

nese yuan) between two adjacent fraudulent transactions and two adjacent legitimate transactions, respectively. As shown in Fig. 2, we get their distributions, where the cumulative distribution represents the proportion of quantity when the gap is less than a certain value. We find that when the time and money gap between two adjacent transactions are small, fraudulent transactions account for the majority. It shows that the fraudulent transactions tend to appear in clusters. They often appear sequentially and similarly in a short time. This is reasonable as it reflects the psychology that fraudsters want to transfer money in a compromised account as soon as possible. We can rely on the paradigm of sliding time window to generate some specific features, called *window-dependent features*, which could be used to describe this kind of behavior pattern in a certain period. Meanwhile, we can generate some features that do not depend on sliding time windows, and term them as *window-independent features*.

Next, we will discuss the generation of the two kinds of features separately.

3.1.1 Window-Independent Features

We extract six window-independent features using the selected raw fields. Two of these features, *Check* and *Frequent_IP*, are obtained directly from the original fields without any changes. The others are either transformed through one of the fields directly or obtained through statistical calculations between different fields. A brief description of these features and their data types is presented in Table 2.

3.1.2 Window-Dependent Features

As mentioned above, the analysis of customers' behavior patterns within a period of time contributes to transaction fraud detection. For this purpose, firstly, we define the concept of a transaction's sliding time window, which is referred to a period of time before it happens; then we extract eight new features, namely *window-dependent features*, by statistic analysis or transformation of raw transaction data within a sliding time window. The window-dependent features are also summarized in Table 2.

3.2 Sliding Time Window Design

As has been discussed, a user's normal and abnormal transaction behavior patterns often change over time. In order to capture this kind of behavior patterns and discriminate them, we can establish a sliding time window. When a user generates a new transaction, the window will slide one step forward to update the transactions within it. Due to the fluctuations of users' trading behavior patterns, the sliding time window should be elastic to adapt to new environments and enhance the robustness of model. In other words, the size of time window needs to be flexible and adaptable as time changes and transaction progresses.

Next, we describe the specific design of sliding time window. In our method, we maintain a transaction list for each user respectively. We store the transaction list as a linked list structure, which means that the head and tail operation of data is very fast regardless of the data amount. For each user, we build a transaction list to store his/her transaction data within a time window. When a user generates a new transaction, the transaction list will be updated by adding the transaction to its top. Once the time range of a transaction list exceeds that of the time window, the old transactions beyond the time window should be removed from the list.

In this work, we choose the REDIS [52] database for an implementation of sliding time window, which can quickly perform streaming calculations. Note that the REDIS database is not necessary and can be replaced by other efficient databases in our method. An implementation of the transaction list and time window is illustrated in Fig. 3.

From Fig. 2(a), we get that when the time gap between two adjacent transactions varies from 30 seconds to an hour, there is a large degree of distinction between the cumulated distributions of normal and fraudulent transactions. So we set the candidate time window size to range from 30 seconds to an hour. For calculation convenience, the step size is set to 30 seconds.

3.3 Window Size Optimization Using LA

We aim at devising an automatic selection scheme for the size of sliding time window. On the one hand, it is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing

Table 2: Descriptions of the newly generated features

	Feature	Data Type	Description
window independent	Check	INT	Please refer to Table 1
	Frequent_IP	Boolean	Please refer to Table 1
	Overpaid_Limit	Boolean	Whether the single transaction amount exceeds the Single_Limit, whether the daily transaction amount exceeds the Daily_Limit
	Overpaid_Left	Boolean	Whether the transaction amount exceeds the balance of the account
	Time_Gap	Float	Time difference between two adjacent transactions
	Money_Gap	Float	Amount difference between two adjacent transactions
window dependent	Times_Window	INT	The number of transactions appear in the time window
	Avg_amt_window	Float	The average amount of all the transactions in the time window
	Var_amt_window	Float	The variance of the amount of all the transactions in the time window
	Acc_amt_window	Float	The cumulative amount of all the transactions in the time window
	Avg_amt_gap_window	Float	The average amount difference between any two adjacent transactions in the time window
	Avg_time_gap_window	Float	The average time difference between any two adjacent transactions in the time window
	Var_amt_gap_window	Float	The variance of amount difference between any two adjacent transactions in the time window
	Var_time_gap_window	Float	The variance of time difference between any two adjacent transactions in the time window

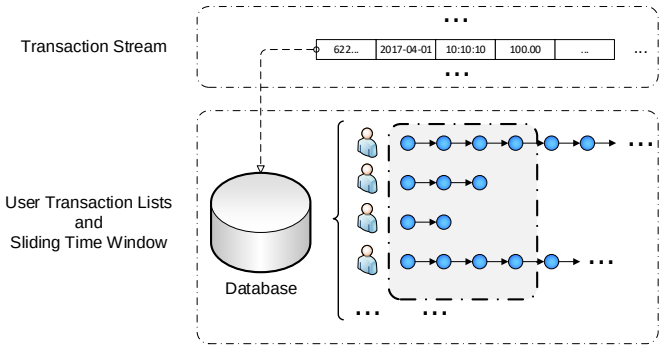


Fig. 3. The architecture of sliding time window.

attributed to the fact that different window sizes will result in fluctuation of model performance. We should choose a proper time window size from a large number of options by optimization. Every time we make a choice of time window size, we will construct new window-dependent features, and feed them together with those window-independent features to a classifier, and evaluate its predictive performance. Obviously, the optimization process is quite time-consuming and labor intensive, given a large number of options for time window size. So we use learning automata (LA) to automatically select the size of time window. On the other hand, in order to maintain the robustness of the predictive model, the time window should be flexible and adaptable to the dynamic environments. Since for the online payment fraud detection systems, the feedback signal, which indicates whether a payment transaction is fraudulent or not, may have a delay of one month or more. Thus we can add a dynamic online updating scheme, that is to say, adjusting the time window size periodically. We call the above method *LAW*.

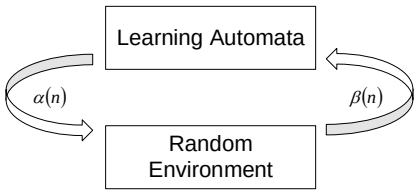


Fig. 4. The framework of Learning Automata.

To make this paper self-contained, we introduce the basic concept of learning automata (LA). Fig. 4 shows the typical system architecture of LA. At first, LA selects an action $\alpha(n)$ via a stochastic process, then it applies the selected action to the environment and receives a reward signal $\beta(n)$ from the performance evaluation function. An action that achieves desirable performance is rewarded by an increase in its probability density. Meanwhile, those underperformed actions are penalized or left unchanged depending on different reward strategies. So during the learning process, the internal probability density of LA will be updated continuously. When the probability density gets convergence, the action with the highest probabilities corresponds to the optimal action.

Next, we discuss how to select the proper time window size automatically with LA. LA interacts with the environment continuously in order to find out the optimal action for the current state. In our study, the environment corresponds to the whole transaction dataset and the chosen classifier, and an action corresponds to a selection of time window size. In Table 3, we list the main components of LA and their corresponding notations.

In Table 3, the size of time window x_t acts as both the output of LA and the input of the environment. Conversely, the reward value β_t corresponds to the output of the environment and the input of LA. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing

Table 3: Notations and parameters in LA

Notation/Parameter	Definition/Meaning
x_t	The size of time window selected at iteration t
β_t	The reward obtained by LA at iteration t
tpr_t	The weighted true positive rate of the classifier prediction under the current time window size at iteration t
TPR_t^m	The list of m weighted true positive rates in the previous m iterations at iteration t
$f_t(x)$	The probability density function of time window size at iteration t

definition of the reward value $\beta_t \in [0, 1]$ is critical, as it not only determines the utility of the currently selected time window size, but also serves as a basis to update the probability density function of time window size during LA training. As interpreted in Table 3, tpr_t plays an important role in the calculation of reward values. It is a performance metric that is set manually. Actually, it can be calculated as a weighted true positive rate, given that the false positive rate is below 0.05%, 0.1%, 0.5% and 1%, respectively. That is,

$$\begin{aligned} tpr_t = {} & 0.4 \cdot (tpr_t | fpr_t = 0.0005) + 0.3 \cdot (tpr_t | fpr_t = 0.001) \\ & + 0.3 \cdot (tpr_t | fpr_t = 0.001) + 0.2 \cdot (tpr_t | fpr_t = 0.005) \\ & + 0.1 \cdot (tpr_t | fpr_t = 0.01). \end{aligned}$$

We choose these false positive rate values because they are the ones to which the industry usually pays more attention [36], [53], [54]. When the false positive rate is smaller, its weight should be bigger, as the goal of optimization is to achieve a larger true positive rate and a smaller false positive rate at the same time. The setting of the weights 0.4, 0.3, 0.2, and 0.1 follows this principle and their sum is 1 which restricts $tpr_t \in [0, 1]$. In addition to the list $TPR_t^m = \{tpr_t, tpr_{t-1}, ..., tpr_{t-m+1}\}$, we also define TPR_{med} and $TPR_{\max}$ as the median and maximum values in TPR_t^m. In our method, LA adopts the reward policy of reward/inaction [55], that is to say, this policy rewards the actions with good performance and does not reward or punish those with bad performance. Following the above principle, when the current weighed true positive rate tpr_t is larger than the median true positive rate TPR_{med} in the list TPR_t^m, we can conclude that the current selected time window size x_t is of good performance, while the observation that tpr_t is smaller than or equal to TPR_{med} means a bad choice of time window size. Based on the above settings, we can get a specific definition of the reward value β_t by:

$$\beta_t = \max\left\{0, \ \frac{tpr_t - TPR_{med}}{TPR_{\max} - TPR_{med}}\right\}. \tag{1}$$

Obviously, β_t is always greater than 0 and its value is restricted between 0 and 1.

During the training process of LA, an action will be selected based on the probability density function $f_t(x)$ of time window size at each iteration, where $f_t(x) \in [0, 1]$ and $x \in [\alpha_{\min}, \alpha_{\max}]$. It holds that those time window sizes with greater probabilities will have more opportunity to be selected. Next, the actions with good performance will receive an increase in the probability via a neighborhood function $N(x, x_t)$ (a probability distribution function). Then, given the reward value β_t, the probability density function $f_t(x)$ is updated according to Eq. (2),

$$f_{t+1}(x) = \alpha \cdot [f_t(x) + \beta_t \cdot N(x, x_t)], \tag{2}$$

where α is used to re-normalize the probability distribution such that

$$\int_{x_{\min}}^{x_{\max}} f_{t+1}(x)dx = 1.$$

The new probability density function determines the time window size to be selected at the next iteration.

Note that a proper neighborhood function $N(x, x_t)$ is usually determined according to the characteristics of specific applications. Taking into account the performance distribution of window sizes, we adopt a Gaussian distribution function as the neighborhood function. The reason is that the Gaussian distribution is roughly consistent with the fact that the window size around the superior size may also be excellent here, i.e., its probability density is augmented in the vicinity of superior action, and the further away other actions are from the superior action, the less their probability density increase. More specifically, the adopted Gaussian neighborhood function, say $G(x, x_t)$, is a symmetric Gaussian function that centers on the current time window size x_t, and is defined in Eq. (3),

$$N(x, x_t) := G(x, x_t) = \lambda \cdot \frac{1}{\sqrt{2\pi}\sigma_1} \cdot e^{-\frac{(x-x_t)^2}{2\sigma_1^2}}, \tag{3}$$

where the hyper-parameters λ and σ_1 control the updating speed of the probability density function $f_t(x)$.

Based on the above discussion about the settings of LA, we introduce the process of our method *LAW* in details. In Fig. 5, we depict the workflow of selecting the most proper time window size, which corresponds to the learning automatic window module in Fig. 1. At the very beginning, the probability density function $f_t(x)$ is initialized such that all the window sizes have the same probability density, and their total probability of accumulation is 1. At each iteration, LA chooses a time window size x_t based on the probability density function. If x_t has been selected in the previous iteration, then there is no need for subsequent feature construction and model training steps, since we have done all these work before. Here we can directly call the previously trained model and get into the model prediction performance evaluation step. If not like that, we should first construct the window-dependent features based on the window of the chosen size. Afterwards, we use

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing

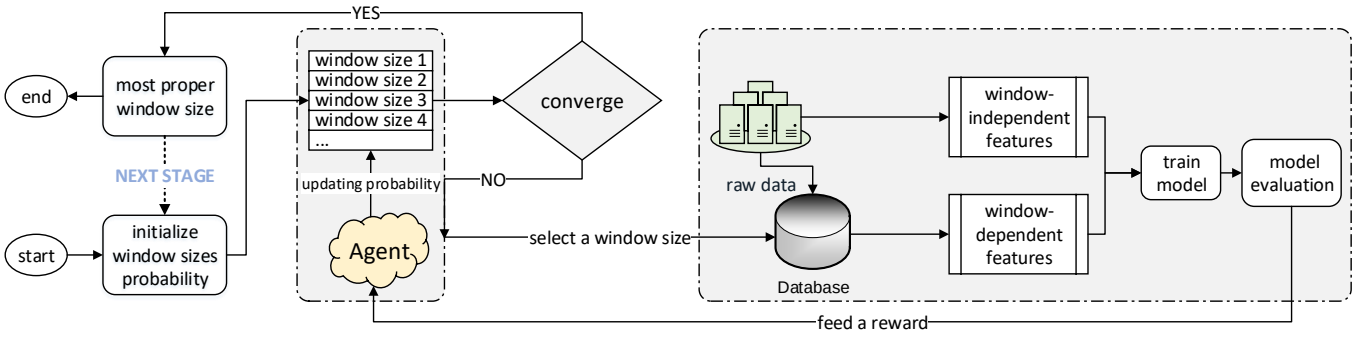


Fig. 5. Workflow of our *LAW* to select the most proper time window size.

them together with the window-independent features to train a model, and evaluate its predictive performance, tpr_t, on an off-line testing dataset. Thus, LA can update the recent m predictive evaluations in TPR_t^m and get the reward value β_t according to Eq. (1). After that, it updates and re-normalizes the probability density function $f_t(x)$ according to Eq. (2). As LA gradually converges towards those more worthy regions, these selected time window sizes should be evaluated more frequently. The probabilities of these regions are higher than others after a number of iterations, which results in better discriminations between the selected actions and unselected ones.

To achieve significantly faster convergence and higher accuracy than the classical pursuit schemes, we learn from the Last-Position Elimination-Based Learning Automata [51]. Before the re-normalization of the probability density function during each iteration, we set a threshold and eliminate the probability below the threshold. It makes the distinction between the worthy areas and other areas more obvious, thus it can help accelerate the speed of convergence.

For the training of LA, we can either set an iteration upper bound or wait until it converges. Here, we choose the latter, thus the number of iterations for LA getting convergence is not fixed. When LA completely converges, we select the window size with the largest probability as the most suitable window size for the current stage.

The most proper time window size selected above is applicable for a period of time. If we set one month as the length of time stage, when entering the next time stage, that is, one month later, the previous time window size will no longer work well and therefore lead to a decline in the performance of model. So we need to adjust and modify the most proper time window size dynamically and regularly. For the purpose of achieving the time window with a flexible size to enhance the robustness of the whole model, our *LAW* also contains an online updating scheme. Now we present how to adjust the size of sliding time window automatically with the updating scheme. We utilize the converged probability density of time window size in the previous stage to initialize LA in the current stage, as shown in Fig. 5. Here when entering the next stage, the new probability density function of LA is initialized as a Gaussian distribution, centering on the most suitable time window size selected in the previous stage. Upon doing so, we aim to make a trade-off between the explore and exploit schemes of LA. On the one hand, we need to exploit the converged probability density in the previous stage, since the probability density of the current LA has a great chance to converge around it. On the other hand, it is necessary to explore the areas far away from the previous proper time window size in case that they deserve good performance. The newly initialized probability density function $f_{Initial}(x, x_l)$ is defined as:

$$f_{Initial}(x, x_l) = \frac{1}{\sqrt{2\pi}\sigma_2} \cdot e^{-\frac{(x-x_l)^2}{2{\sigma_2}^2}}, \tag{4}$$

where x_l represents the most suitable window size in the last stage and σ_2 controls the weight of x_l. The remaining iteration procedure has no difference from that in the previous stage. Again we choose the window size with the largest probability when LA converges. We provide the whole procedure of *LAW* in Algorithm 1.

4 EXPERIMENTAL EVALUATION

In this section, we investigate the performance of the proposed method *LAW* for fraud detection in a real-life online payment scenario. For this purpose, we evaluate the performance and then analyze the gain of using time window and LA, respectively.

4.1 Dataset Description

Our experiments are conducted on a dataset of real-life online payment records from a commercial bank. The records were collected within three months from April 1, 2017, to June 30, 2017. In this work, the dataset is partitioned into two parts, with the records from April and May as the training data and those from June as the testing data. The total number of transaction records used for training is $2,459,334$ while the number for testing is $1,042,714$.

All the transactions are labelled with integers ranging from 0 to 10 by the commercial bank manually. The meanings of labels are presented in Table 4. Among the

Algorithm 1 The Procedure of *LAW*

Require:
The number of the candidate window sizes, N;
Hyper-parameters of Gaussian neighborhood function, λ and σ_1;
Hyper-parameter of the newly initialed probability density function, σ_2;
The size of recent weighted true positive rate list TPR_t^m, m;
The threshold for last-position elimination, t;
Ensure:
if There is no previous stage before **then**
  Initial the probability density function: $f_t(x) = \frac{1}{N}$;
else
  Initial $f_t(x)$ according to Eq. (4);
end if
while $f_t(x)$ does not get convergence **do**
  Choose a size of the time window, x_t, based on the probability density function;
  if x_t is a repeated time window size chosen before **then**
    Obtain predictive evaluation tpr_t based on previous model;
  else
    Construct the new window-dependent features, train a model based on all features and obtain tpr_t;
  end if
  Update the recent weighted true positive rate list TPR_t^m;
  Compute the reward value of learning automata β_t according to Eq. (1);
  Update $f_t(x)$ according to Eq. (2);
  Eliminate the probability below the threshold t and re-normalize $f_t(x)$;
end while
Select the most suitable window size $x_{optimal}$ according to the final probability density function;
return $x_{optimal}$;

labels, 0 and 1 represent the normal transactions called *negative* samples, $2 \sim 6$ represent uncertain transactions, and $7 \sim 10$ represent fraudulent transactions which we call *positive* samples. We exclude the uncertain transactions, and then the remaining transactions form the actually adopted dataset for training and testing. We further draw a number of transaction records from the last half month of the training data, and use them as the off-line testing data. In order to simulate different time periods, we divide the testing data into two equal parts chronologically, resulting in two online-testing datasets.

The statistics of positive and negative transaction records in each dataset is summarized in Table 5.

4.2 Setup and Metrics

Our experiments are conducted on a server with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz and 128 GB RAM.

Table 4: Meanings of labels

Label	Meaning
0	Normal transactions released by system
1	Normal transactions after phone confirmed but intercepted by system before
2	Unmarked transactions
3	Customer call is not answered
4	Invalid customer phone number
5	Customer unsure
6	Customer reject to answer
7	Other kinds of fraud
8	Trojan virus
9	Telecommunications fraud
10	Phishing website

Table 5: Statistics of transaction records

Dataset	Label (Property)	Quantity
training	**0-1 (negative)**	**1,806,720**
	2-6 (discarded)	20,206
	7-10 (positive)	**32,788**
off-line testing	**0-1 (negative)**	**587,097**
	2-6 (discarded)	4,918
	7-10 (positive)	**7,605**
online testing Part 1	**0-1 (negative)**	**534,033**
	2-6 (discarded)	5,919
	7-10 (positive)	**10,737**
online testing Part 2	**0-1 (negative)**	**469,506**
	2-6 (discarded)	8,358
	7-10 (positive)	**14,163**

We conduct experiments to verify our method *LAW* on the testing dataset Part 1, and make a comparison between *LAW* without an online updating scheme and *LAW* containing an online updating scheme on the testing dataset Part 2.

In order to evaluate the effectiveness of our method, we utilize the True Positive Rate (Sensitive or Recall), False Positive Rate (1-Specificity or Disturb), Precision, F1-score, and ROC (Receiver Operating Characteristic Curve) as the metrics to quantify the detection performance over the testing datasets.

4.3 Feature Generation Using Time Window

The generation of window-independent features is easy and once for all, but the generation of window-dependent features is affected by the specific selection of time window size. Intuitively, the feature generation time is positively related to the time window size. Here, we analyze the efficiency of feature generation under each time window size on the testing dataset Part 1. From Fig. 6, we can observe that the size of time window and the feature generation time approximately depend linearly on each other, where the generation time is calculated by using more than $500,000$ transactions totally.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing

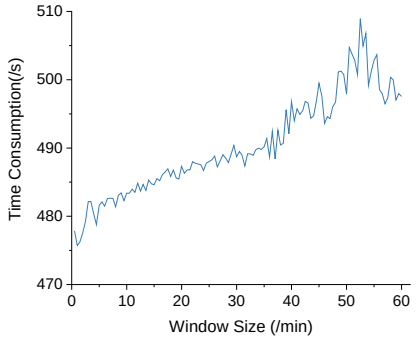


Fig. 6. Time consumption of feature generation on the testing dataset Part 1.

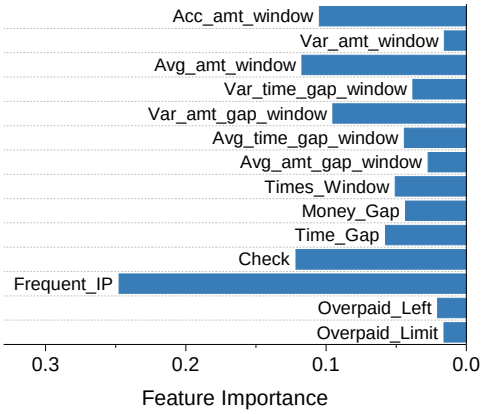


Fig. 7. The feature importances.

If we consider a single transaction record, the average feature generation time is about 1.1 milliseconds. Meanwhile, when selecting different time window size, the time spent in feature generation for each transaction fluctuates within a range of no more than 0.05 milliseconds, which is negligible compared with model training time. So we can conclude that the selection of time window size has little impact on the whole learning time.

To manifest the importance of window-dependent features, we utilize the Gini Index [56] to conduct an assessment. We choose 30 seconds as the time window size, the importance of all features can be found in Fig. 7. We observe that, in addition to *Frequent_IP* and *Check*, most window-dependent features are more important than window-independent features in improving model performance. The result means that the generated features by using time window really play an important role in fraud detection. It is noteworthy that *Frequent_IP* and *Check* are of high importance since their meaning is closely related to fraud. Specially speaking, *Frequent_IP* indicates that whether a customer makes a transaction in a high-risk environment, such as an infrequently used IP. There is a reason to suspect that the customer's account may be stolen by the attacker in a high-risk environment. The high importance of *Check* lies in the fact that different verification tools are vulnerable to targeted attacks by fraudsters due to their vulnerabilities.

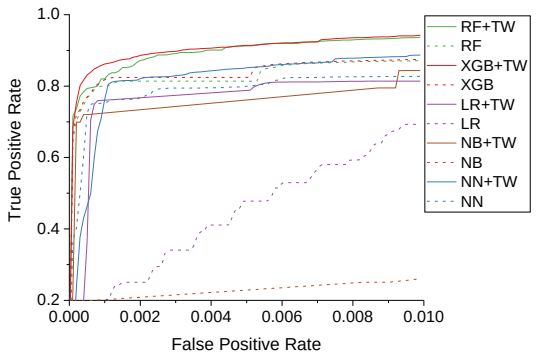
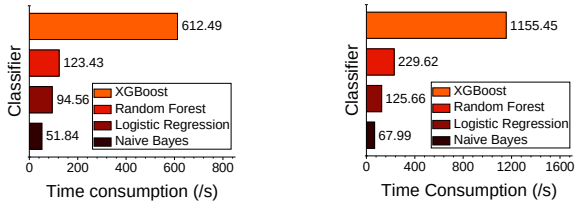


Fig. 8. ROC curves of predictive models using different classifiers on the testing dataset Part 1.



(a) Without TW (b) With TW

Fig. 9. The average training time consumptions of different classifiers.

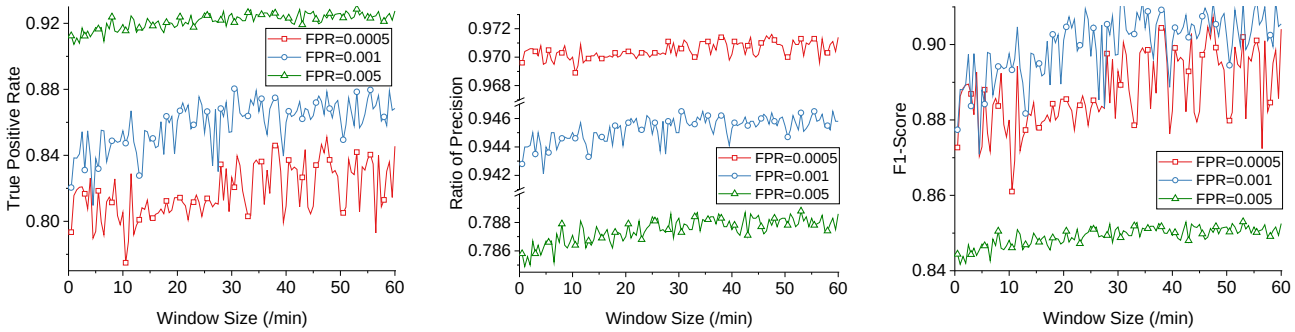
4.4 Performance Evaluation of Time Window Models

In this section, we investigate how time window and its size affect fraud detection.

Firstly, we compare the performance of different popular classifiers using or not using window-dependent features on the testing dataset Part 1. We set all their window sizes as 30 seconds. To this end, we make an analysis of the training efficiency for each classifier. These models are all implemented on open source packages[1] in the Python language environment, on which all these models except XGBoost are implemented through the machine learning tool 'scikit-learn' and XGBoost model is implemented by importing the 'xgboost' open source package. Note that, the Neural Network is built by a multi-layer perception classifier (MLPClassifier) with two hidden layers. In general, the k-fold cross validation is a widely-used parameter selection method. However, in practical anti-fraud applications, it might cause the so-called *time crossing* problem where some future information is utilized for checking transactions. Therefore, we select the parameters through *grid search* on the offline testing dataset, and then evaluate the performance of our model on the online testing Part 1 dataset. The *grid search* is a special exhaustive search method in 'scikit-learn', which verifies the model performance under each combination in the parameter combination list and takes the combination of parameters with the highest per-

1. The open source packages used in this work can be obtained via https://scikit-learn.org/stable/supervised_learning and https://xgboost.readthedocs.io/en/latest.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing



(a) True Positive Rate (b) Precision (c) F1-Score

Fig. 10. The True Positive Rate, Precision and F1-Score when choosing different sizes of time window.

formance as the optimal parameter. Finally, 500 trees with a depth of 6 are selected for the Random Forest and XGBoost models, the number of minimum sample partitions is selected to be 100 in RF, and 80% features and samples are randomly selected to establish a decision tree. The Logistic Regression selects the 'sag' solver and 'L2' penalty and conducts 100 iteration training. The rest of the parameters are set by default. The ROC curves are depicted in Fig. 8. We can observe that with time windows, all involved methods, including Random Forest ($RF + TW$), XGBoost ($XGB + TW$), Logistic Regression ($LR + TW$), Naive Bayes ($NB + TW$) and Neural Network with two hidden layers ($NN + TW$), outperform their counterparts without time windows.

Now we already have a preliminary understanding of the approximate performance of all candidate classifiers. Next, we need to choose one of them for future use in our method *LAW*. As LA requires multiple learning iterations to converge, the chosen classifier should not only be good at performance, but also be economical in terms of training time. Fig. 9 depicts the average time consumptions of training the model using different classifiers except for Neural Network whose time consumption is far greater than that of other classifiers even considering a pre-trained model. From Fig. 8, we can see that both $RF+TW$ and $XGB+TW$ have almost comparably high performance, but Fig. 9 suggests that training $RF + TW$ costs less time than $XGB + TW$. Based on the above facts, we choose Random Forest as the classifier for our method *LAW* in the following experiments.

We then analyze the fluctuations of true positive rate, precision and F1-score with different choices of time window size on off-line testing data. In this test, the values of false positive rate are assigned as 0.0005, 0.001 and 0.005, respectively. We choose the time windows with sizes ranging from half a minute to 60 minutes to make the analysis. When selecting different time window sizes, we keep both the training samples and the setting of Random Forest classifier unchanged. We can then obtain the statistical information of the listed metrics in Fig. 10. It reflects how time window size affects model performance. By the performance comparison under different false positive rates, we observe that the change in time window size is not synchronized with those in the True Positive Rate, Precision and F1-score. As the performance fluctuates a lot when the window size changes, selecting the most suitable window size is of great significance.

4.5 Performance Evaluation of LAW

As mentioned above, different choices of time window size result in different performance. In this section, we elaborate on the performance obtained by using *LAW*.

We set the input parameters in Algorithm 1 according to the following procedure. We set the number of the candidate window sizes N as 120 by searching all time window sizes between 0.5 and 60 minutes at the interval of 30 seconds. The size of recent weighted true positive rate list, m, is set to be 10 based on the experience. According to the principle of last-position elimination that is used during each iteration for faster convergence, the elimination threshold, t, is set to be a quarter of the initial probability density, i.e., $1/4N$.

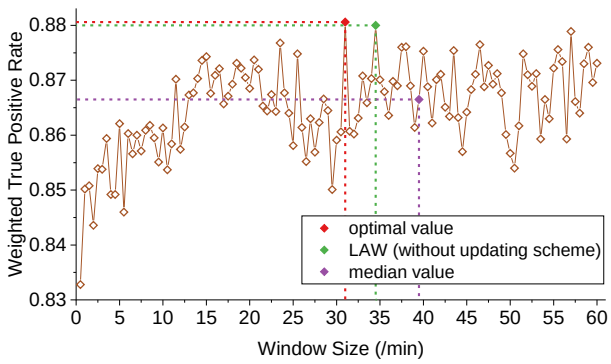
To demonstrate the performance of proposed method better, we set a new metric that is shown in Eq. (5):

$$M_{relative} = \frac{TPR_{weighted} - TPR^{median}_{weighted}}{TPR^{optimal}_{weighted} - TPR^{median}_{weighted}}, \tag{5}$$

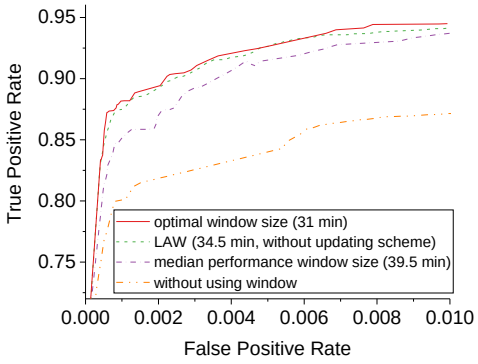
where $TPR_{weighted}$ is the weighted true positive rate under our method, $TPR^{optimal}_{weighted}$ means the optimal weighted true positive rate under the exhaustive method, and $TPR^{median}_{weighted}$ means the median weighted true positive rate under the exhaustive method. This relative performance metric changes its value in the range of $[0, 1]$. When it is closer to 1, the relative performance is higher, that is to say, the closer to the best performance under the exhaustive method. Furthermore, we select the methods with two fixed time window sizes as comparative methods (i.e. the optimal window size and the median performance window size, respectively). In addition, we compare *LAW* with the method without using time window.

We use the testing dataset Part 1 to evaluate method *LAW* without the updating scheme. In Fig. 11(a), we

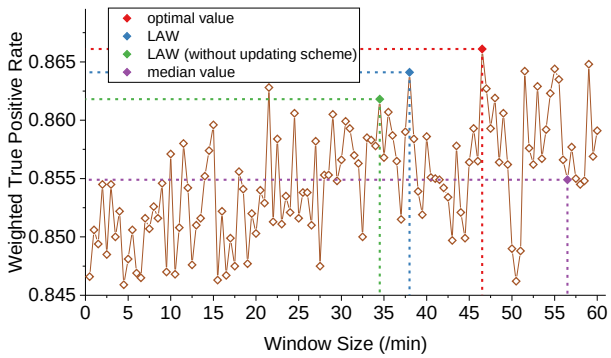
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing



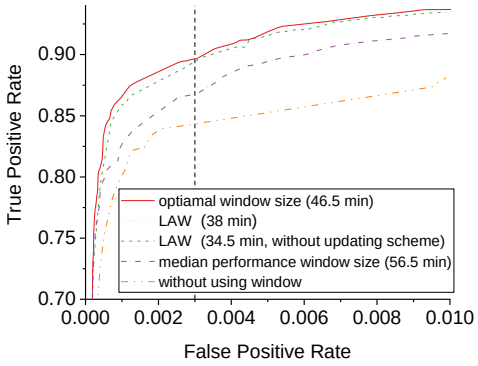
(a) Performance under exhaustive method on the testing dataset Part 1



(b) ROC curves of predictive models with different methods on the testing dataset Part 1



(c) Performance under exhaustive method on the testing dataset Part 2



(d) ROC curves of predictive models with different methods on the testing dataset Part 2

Fig. 11. Performance (weighted true positive rate) under the exhaustive method and ROC curves of predictive models with different methods.

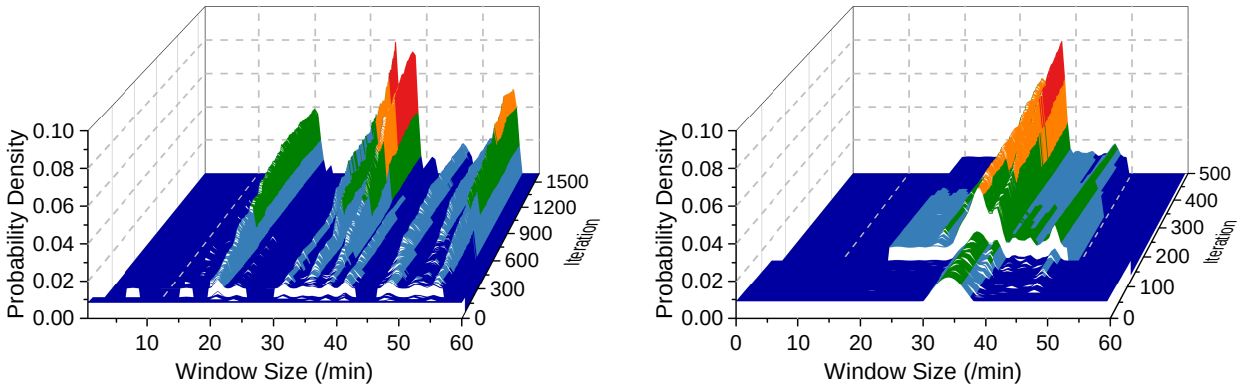
show the performance of the exhaustive method on the testing dataset Part 1. From this, we select the optimal window size (31 minutes) and the median performance window size (39.5 minutes, ranks 60th in performance under all 120 candidate window sizes) to make a comparison. The performance of the window size (34.5 minutes) selected by our methods ranks 2/120 (top 2%) among all candidate window sizes on the testing dataset Part 1. According to the statistics of performance under the exhaustive method, the relative performance value $M_{relative}$ is 0.9574 on the testing dataset Part 1.

From Fig. 11(b), we can get the conclusion that our method *LAW* achieves better detection performance than other methods except the optimal window size obtained under the exhaustive method. The performance of our proposed approach is very close to the best performance under the exhaustive method, it does not cause a significant drop in accuracy (weighted true positive rate) as compared to the exhaustive method.

In a similar way, we evaluate our method *LAW* without updating scheme on the testing dataset Part 2. Further, we use the testing dataset Part 1 as the offline testing set, the following procedure is almost the same with the previous learning stage, except that we need to re-initialize the probability density function at the beginning. Then we evaluate *LAW* on the testing dataset Part 2.

In Fig. 11(c), we represent the performance of the exhaustive method on the testing dataset Part 2. We also select the optimal window size (46.5 minutes) and the median performance window size (56.5 minutes). The performance of the window size (38 minutes) selected by our methods ranks 4/120 (top 4%) among all candidate window sizes on the testing dataset Part 2. The relative performance value $M_{relative}$ is 0.8214 on the testing dataset Part 2. In Fig. 11(d), our method *LAW* outperforms the methods using median performance window size under the exhaustive method or without using time window. Even the *LAW* without the updating scheme shows a better performance than the previous two. In addition, our *LAW* or *LAW* without the updating scheme both have a close performance compared with the optimal window size under exhaustive method. Although it seems that *LAW* without updating scheme has comparable performance as *LAW*, when we require very small false positive rate, say, below 0.003, the latter shows nonnegligible advantages. This validates the sensitivity of our method *LAW* to changing transaction patterns, by which it can still achieve performance gain over the exhaustive search when encountering fluctuations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing



(a) on the off-line testing dataset (at the very beginning stage)

(b) on the testing dataset Part 1 (utilizing the most suitable window size selected in previous stage)

Fig. 12. Convergence process of probability density function of time window.

Table 6: The efficiency of the convergence using different hyper-parameters

λ	σ_1	Iteration times	Converge
1/240	0.5	763	False
	1	1527	True
	1.5	2355	True
1/120	1	874	True
1/60		173	False

4.6 Advantage analysis of LAW

In this section, we further analyze the advantages of *LAW* based on the experiments in Section 4.5.

We first analyze the parameters λ and σ_1 in Eq. (3), which control the updating speed of the probability density function. We conduct groups of comparative experiments by varying the values of these two hyper-parameters. Upon doing so, we can make a comparison of the convergence efficiency, and find out how they work. From Table 6, we get that λ decreases linearly with the iteration time while σ_1 increases linearly with the iteration time.

Meanwhile, when λ is large enough or σ_1 is small enough, LA cannot converge and the most suitable window size cannot be determined. In this work, we choose $\lambda = 1/240$ and $\sigma_1 = 1$. It takes 1527 iterations for LA to achieve convergence and the most suitable window size is 34.5 minutes (2070 seconds). As the number of iterations increases, the probability density function converges to the most suitable window size. In other words, the probability of selecting the area around the most suitable time window size becomes larger than others. We can get the detailed information of the convergence process of *LAW* without updating scheme in Fig. 12(a).

With the help of updating scheme, LA significantly reduces the number of iterations when convergence is achieved. The detailed convergence information of *LAW* on the testing dataset Part 1 can be observed in Fig. 12(b). It utilizes the most suitable window size selected in the previous learning stage as the premise to re-initialization. More specifically, a Gaussian distribution with $\sigma_2 = 6$ is initialized as the new probability density by centering on the most suitable time window size (34.5 minutes) selected in the previous learning stage. Fig. 13 shows the comparison between the final converged probability density in the previous stage and the initialized probability density function. The new most suitable window size is different from the size in the previous learning stage, it becomes a little larger, changing from 34.5 minutes (2070 seconds) to 38 minutes (2280 seconds). It takes fewer iterations (about 500) for LA to get convergence, this can be mainly explained by the fact that the newly initialized probability density is based on the historical experience. This verifies that the online updating scheme helps reduce the iterations of convergence to find the most proper window size.

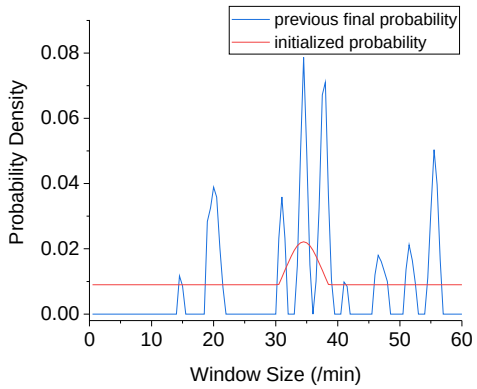
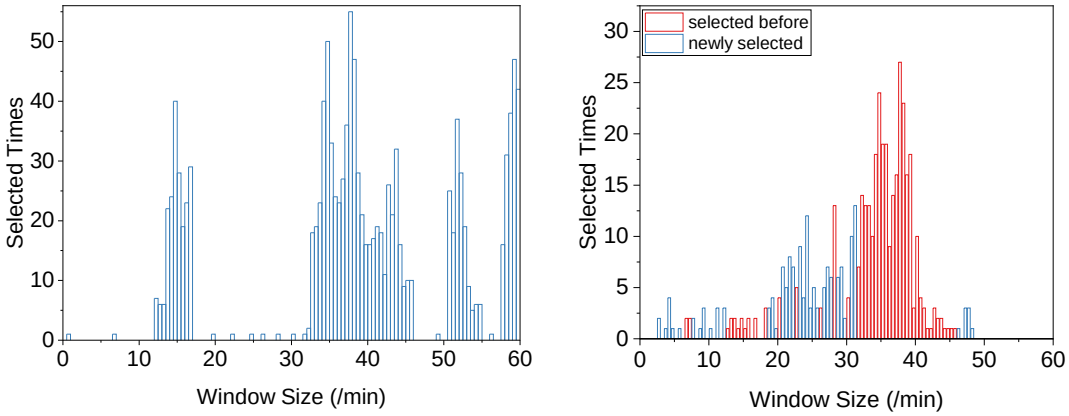


Fig. 13. Comparison of the previous final probability and the initialized probability.

Next, we analyze the number of times that each time window size is selected during the convergence process of LA. Due to the fact that most of the iteration time is occupied by generating new features and manipulating classification models, higher efficiency can be achieved

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3037784, IEEE Transactions on Dependable and Secure Computing



(a) on the off-line testing dataset (b) on the testing dataset Part 1

Fig. 14. Times of each window size selected in the whole convergence process.

when LA selects more repeating values of time window size during iteration. As shown in Fig. 14, only parts of time window sizes are selected actually. So the actually consumed time will be greatly reduced with the help of LA, and it is just part of the time that is calculated under all time window sizes using the exhaustive method. The ratio T_{ratio} between them can be defined as:

$$T_{ratio} = \frac{N_{non-repeating}}{N_{total}}, \tag{6}$$

where $N_{non-repeating}$ represents the number of the time window sizes that are not repeatedly picked by LA and N_{total} represents the total number of time window sizes. In Fig. 14(a), 63 non-repeating time window sizes are selected by LA during the whole convergence process on the off-line testing dataset, so it holds that $T_{ratio} = 63/120$. It means that the time spent in manipulating the classification models is only $63/120$ of the time used by the exhaustive methods. These are the benefits that LA brings. Fig. 14(b) shows the times that each time window size was selected on the testing dataset Part 1, where we get the $T_{ratio} = 81/120$. It saves a third of the time as compared with the exhaustive method. Actually, we do not need further time consumption when a chosen window size has appeared in the previous learning stage. Just as shown in Fig. 14(b), since the window sizes within red parts have been picked in the previous learning stage, they need not to be calculated again. So the real ratio holds that $T_{ratio} = 36/120$, which is considerably time-saving.

To show the time saving of our method more clearly, we quantify the full end-to-end running time to demonstrate the superiority of our method in terms of time consumption, where most of the iteration time is occupied by generating new features and manipulating classification models and the running time of the *LAW* is indeed infinitesimal to the former. As shown in Fig. 15, our two experiments show that nearly more than half the time has been saved compared to the exhaustive method.

In summary, we can conclude that the most suitable time window size selected by LA does improve the performance of predictive models. Especially, the dedicated online updating scheme in our method *LAW* is able to ensure good robustness of the whole model in dynamic environments. Moreover, compared to the exhaustive methods, ours is very time-saving in both selecting and adjusting the sizes of time windows.

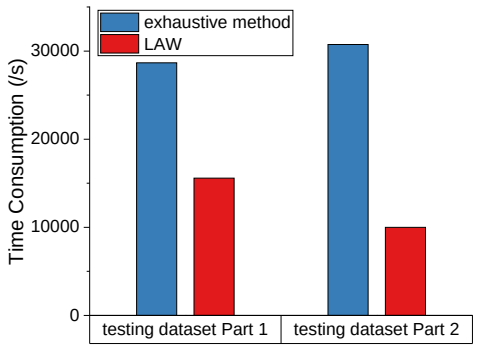


Fig. 15. Comparison of the full end-to-end running time.

5 CONCLUSION

In this work, we design a method called *LAW* to perform online payment fraud detection. It searches for the appropriate window-dependent features by automatically learning the most suitable time window size. These features can represent customers' behavior patterns in a certain period. Our method achieves a more economical and time-saving performance for window size selection compared with the brute-force methods. Furthermore, *LAW* enables the window size to be adaptive to dynamic environments by implementing an updating scheme. By experiments on the real-life online payment dataset from a commercial bank, our method has been proven to have a better detection efficiency than those using preset fixed window size or without using time window. In addition, with the dynamic updating scheme, our *LAW* can ensure the robustness of model better.

There are some useful and interesting issues left to study:

(1) In this paper, the candidate values of the time window sizes are still limited. One of our future works is to expand the range of values and search space to enhance usability of the proposed method.

(2) Moreover, for the purpose of allocating iteration times more rationally and saving more time, we will exploit another paradigm, called the *optimal computing budget allocation* (OCBA), to maximize the probabilities of selecting those true optimal actions in the future.

(3) Another future work is that we could utilize different window sizes for different features, for example, a model with two sets of window-dependent features extracted from a small and a large window. We are looking forward to exploring this optimization issue under multiple windows.

(4) Finally, we are pushing for the opportunity to cooperate with more banks and third-party fintech companies in the future, so that we can obtain more and larger (including time span and data scale) data samples to verify our proposed method.

REFERENCES

[1] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein, "Data breaches, phishing, or malware?: Understanding the risks of stolen credentials," in *Proc. ACM SIGSAC 2017*, pp. 1421–1434.

[2] P. Hille, G. Walsh, and M. Cleveland, "Consumer fear of online identity theft: Scale development and validation," *Journal of Interactive Marketing*, vol. 30, pp. 1–19, 2015.

[3] L. Coppolino, S. D'Antonio, V. Formicola, C. Massei, and L. Romano, "Use of the dempster-shafer theory to detect account takeovers in mobile money transfer services," *J. Ambient Intelligence and Humanized Computing*, vol. 6, no. 6, pp. 753–762, 2015.

[4] K. Thomas, F. Li, C. Grier, and V. Paxson, "Consequences of connectivity: Characterizing account hijacking on twitter," in *Proc. ACM SIGSAC 2014*, pp. 489–500.

[5] N. F. Ryman-Tubb, P. Krause, and W. Garn, "How artificial intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark," *Eng. Appl. of AI*, vol. 76, pp. 130–157, 2018.

[6] S. Xie and P. S. Yu, "Next generation trustworthy fraud detection," in *Proc. The 4th IEEE International Conference on Collaboration and Internet Computing*, 2018, pp. 279–282.

[7] Y. Ban, X. Liu, L. Huang, Y. Duan, X. Liu, and W. Xu, "No place to hide: Catching fraudulent entities in tensors," in *Proc. WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 83–93.

[8] C. Jing, C. Wang, and C. Yan, "Thinking like a fraudster: Detecting fraudulent transactions via statistical sequential features," in *Proc. FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, pp. 588–604.

[9] ——, "Thinking like a fraudster: Detecting fraudulent transactions via statistical sequential features," *Proc. International Conference on Financial Cryptography and Data Security*, pp. 140–155, 2019.

[10] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "Guest editorial learning automata: theory, paradigms, and applications," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 32, no. 6, pp. 706–709, 2002.

[11] R. Barskar, A. J. Deen, J. Bharti, and G. F. Ahmed, "The algorithm analysis of e-commerce security issues for online payment transaction system in banking technology," *CoRR*, vol. abs/1005.4266, 2010. [Online]. Available: http://arxiv.org/abs/1005.4266

[12] C. Whitrow, D. J. Hand, P. Juszczak, D. J. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," *Data Min. Knowl. Discov.*, vol. 18, no. 1, pp. 30–55, 2009.

[13] T. Fawcett and F. J. Provost, "Adaptive fraud detection," *Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 291–316, 1997.

[14] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235–249, 2002.

[15] C. Wang and H. Zhu, "Representing fine-grained co-occurrences for behavior-based fraud detection in online payment services," *IEEE Trans. Dependable and Secure Computing*. [Online]. Available: https://doi.org/10.1109/TDSC.2020.2991872

[16] A. D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 29, no. 8, pp. 3784–3797, 2018.

[17] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[18] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[19] D. Malekian and M. R. Hashemi, "An adaptive profile based fraud detection framework for handling concept drift," in *Proc. International ISC Conference on Information Security and Cryptology*, 2014, pp. 1–6.

[20] J. West, M. Bhattacharya, and M. R. Islam, "Intelligent financial fraud detection practices: An investigation," in *Proc. International Conference on Security and Privacy in Communication Networks*, 2014, pp. 186–203.

[21] A. Jarovsky, T. Milo, S. Novgorodov, and W. Tan, "Rule sharing for fraud detection via adaptation," in *Proc. IEEE ICDE 2018*, pp. 125–136.

[22] ——, "GOLDRUSH: rule sharing system for fraud detection," *The Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1998–2001, 2018.

[23] T. Milo, S. Novgorodov, and W. Tan, "Interactive rule refinement for fraud detection," in *Proc. The 21th International Conference on Extending Database Technology*, 2018, pp. 265–276.

[24] S. Jha, M. Guillen, and J. C. Westland, "Employing transaction aggregation strategy to detect credit card fraud," *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12 650–12 657, 2012.

[25] Q. Yang, X. Hu, Z. Cheng, K. Miao, and X. Zheng, "Based big data analysis of fraud detection for online transaction orders," in *Proc. The 5th International Conference on Cloud Computing*, 2014, pp. 98–106.

[26] A. G. C. de Sá, A. C. M. Pereira, and G. L. Pappa, "A customized classification algorithm for credit card fraud detection," *Eng. Appl. of AI*, vol. 72, pp. 21–29, 2018.

[27] S. Akila and U. S. Reddy, "Cost-sensitive risk induced bayesian inference bagging (RIBIB) for credit card fraud detection," *J. Comput. Science*, vol. 27, pp. 247–254, 2018.

[28] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. E. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Syst. Appl.*, vol. 51, pp. 134–142, 2016.

[29] 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.

[30] S. Nami and M. Shajari, "Cost-sensitive payment card fraud detection based on dynamic random forest and *k*-nearest neighbors," *Expert Syst. Appl.*, vol. 110, pp. 381–392, 2018.

[31] Y. Wang, S. Adams, P. A. Beling, S. Greenspan, S. Rajagopalan, M. C. Velez-Rojas, S. Mankovski, S. M. Boker, and D. E. Brown, "Privacy preserving distributed deep learning and its application in credit card fraud detection," in *Proc. The 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, 2018, pp. 1070–1078.

[32] S. Yuan, X. Wu, J. Li, and A. Lu, "Spectrum-based deep neural networks for fraud detection," in *Proc. ACM CIKM 2017*, pp. 2419–2422.

[33] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit card fraud detection using convolutional neural networks," in *Proc. NeurIPS 2016*, pp. 483–490.

[34] J. Qian, X. Li, C. Zhang, L. Chen, T. Jung, and J. Han, "Social network de-anonymization and privacy inference with knowledge graph model," *IEEE Trans. Dependable and Secure Computing*, vol. 16, no. 4, pp. 679–692, 2019.

[35] B. Hooi, K. Shin, H. A. Song, A. Beutel, N. Shah, and C. Faloutsos, "Graph-based fraud detection in the face of camouflage," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 4, pp. 44:1–44:26, 2017.

[36] J. J. Ying, J. Zhang, C. Huang, K. Chen, and V. S. Tseng, "*FrauDetector*$^+$: An incremental graph-mining approach for efficient fraudulent phone call detection," *ACM Trans. Knowledge Discovery from Data*, vol. 12, no. 6, pp. 68:1–68:35, 2018.

[37] D. Olszewski, "Fraud detection using self-organizing map visualizing the user profiles," *Knowl.-Based Syst.*, vol. 70, pp. 324–334, 2014.

[38] D. de Roux, B. Perez, A. Moreno, M. Villamil, and C. Figueroa, "Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach," in *Proc. ACM SIGKDD 2018*, pp. 215–222.

[39] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P. Portier, L. He-Guelton, and O. Caelen, "Sequence classification for credit-card fraud detection," *Expert Syst. Appl.*, vol. 100, pp. 234–245, 2018.

[40] T. Wüchner, A. Cislak, M. Ochoa, and A. Pretschner, "Leveraging compression-based graph mining for behavior-based malware detection," *IEEE Trans. Dependable Secure Computing*, vol. 16, no. 1, pp. 99–112, 2019.

[41] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: effective and efficient behavior-based android malware detection and prevention," *IEEE Trans. Dependable Secure Computing*, vol. 15, no. 1, pp. 83–97, 2018.

[42] M. U. K. Khan, H. S. Park, and C. Kyung, "Rejecting motion outliers for efficient crowd anomaly detection," *IEEE Trans. Information Forensics and Security*, vol. 14, no. 2, pp. 541–556, 2019.

[43] U. Porwal and S. Mukund, "Credit card fraud detection in e-commerce: An outlier detection approach," *CoRR*, vol. abs/1811.02196, 2018. [Online]. Available: http://arxiv.org/abs/1811.02196

[44] S. Dhankhad, E. A. Mohammed, and B. Far, "Supervised machine learning algorithms for credit card fraudulent transaction detection: A comparative study," in *Proc. The 19thIEEE International Conference on Information Reuse and Integration,*, 2018, pp. 122–125.

[45] I. Sohony, R. Pratap, and U. Nambiar, "Ensemble learning for credit card fraud detection," in *Proc. ACM India Joint International Conference on Data Science and Management of Data*, 2018, pp. 289–294.

[46] J. Liono, A. K. Qin, and F. D. Salim, "Optimal time window for temporal segmentation of sensor streams in multi-activity recognition," in *Proc. The 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2016, pp. 10–19.

[47] S. Nisar, O. U. Khan, and M. Tariq, "An efficient adaptive window size selection method for improving spectrogram visualization," *Comp. Int. and Neurosc.*, vol. 2016, pp. 6 172 453:1–6 172 453:13, 2016.

[48] S. H. Seyyedi and B. Minaei-Bidgoli, "Estimator learning automata for feature subset selection in high-dimensional spaces, case study: Email spam detection," *Int. J. Communication Systems*, vol. 31, no. 8, 2018.

[49] A. Liu, K. Chen, Q. Liu, Q. Ai, Y. Xie, and A. Chen, "Feature selection for motor imagery EEG classification based on firefly algorithm and learning automata," *Sensors*, vol. 17, no. 11, p. 2576, 2017.

[50] E. Cuevas, D. Zaldivar, and M. A. P. Cisneros, "Seeking multi-thresholds for image segmentation with learning automata," *CoRR*, vol. abs/1405.7361, 2014. [Online]. Available: http://arxiv.org/abs/1405.7361

[51] J. Zhang, C. Wang, and M. Zhou, "Last-position elimination-based learning automata," *IEEE Trans. Cybernetics*, vol. 44, no. 12, pp. 2484–2492, 2014.

[52] R. M.Lerner, "Redis," *Linux journal*, no. Sep. TN.197, pp. 20–23, 12 2010.

[53] Y. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, Y. A. Qi, and Z. Zhou, "Distributed deep forest and its application to automatic detection of cash-out fraud," *ACM Trans. Intelligent Systems and Technology*, vol. 10, no. 5, pp. 55:1–55:19, 2019.

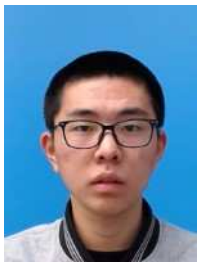
[54] Y. Zhu, D. Xi, B. Song, F. Zhuang, S. Chen, X. Gu, and Q. He, "Modeling users' behavior sequences with hierarchical explainable network for cross-domain fraud detection," in *Proc. WWW 2020, Taipei, Taiwan, April 20-24, 2020*, pp. 928–938.

[55] B. J. Oommen and E. R. Hansen, "The asymptotic optimality of discretized linear reward-inaction learning automata," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 14, no. 3, pp. 542–545, 1984.

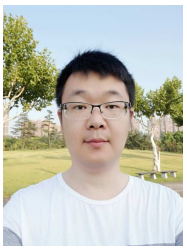
[56] D. Liu, Z. Li, K. Du, H. Wang, B. Liu, and H. Duan, "Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains," in *Proc. ACM CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 537–552.

Cheng Wang received MS degree at Department of Applied Mathematics from Tongji University in 2006 and the PhD degree in Department of Computer Science at Tongji University in 2011. He is currently a professor in the Department of Computer Science at Tongji University. His research interests include cyberspace security and intelligent information services.

Changqi Wang received his bachelor degree of engineering at School of Computer Science and Technology from Central South University in June, 2017. He is now a master student at Department of Computer Science at Tongji University in Shanghai, China. His research interests include data mining, autoML, risk control and fraud detection.



Hangyu Zhu received his bachelor degree in Computer Science from Shanghai Maritime University in Shanghai of China in June, 2018. He is now a master student at Department of Computer Science at Tongji University. His research interests include network embedding and online fraud detection.



Jipeng Cui is now a phD student at Department of Computer Science at Tongji University. His research interests include fraud detection, identity theft detection, and recommender system.