Expert Systems With Applications 91 (2018) 235–251

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Sequential fraud detection for prepaid cards using hidden Markov model divergence

William N. Robinson*, Andrea Aria

Computer Information Systems, Georgia State Unviversity, Atlanta, GA, USA

ARTICLE INFO

Article history:
Received 14 March 2017
Revised 23 August 2017
Accepted 24 August 2017
Available online 12 September 2017

Keywords:
Stored value cards
Transaction processing
Fraud detection
Hidden Markov model
KL divergence
Security

ABSTRACT

Stored-value cards, or prepaid cards, are increasingly popular. Like credit cards, their use is vulnerable to fraud, costing merchants and card processors millions of dollars. Prior techniques to automate fraud detection rely on a priori rules or specialized learned models associated with the customer. Mostly, these techniques do not consider fraud sequences or changing behavior, which can lead to false alarms. This study demonstrates how a transaction model can be dynamically created and updated, and fraud can be automatically detected for prepaid cards. A card processing company creates models of the store terminals rather than the customers, in part, because of the anonymous nature of prepaid cards. The technique automatically creates, updates, and compares hidden Markov models (HMM) of merchant terminals. We present fraud detection and experiments on real transactional data, showing the efficiency and effectiveness of the approach. In the fraud test cases, derived from known fraud cases, the technique has a good F-score. The technique can detect fraud in real-time for merchants, as card transactions are processed by a modern transaction processing system.

1. Introduction

This work presents a store-centric approach to fraud detection. Sequential anomalies are detected using hidden Markov model analysis over a merchant's stream of financial card transactions. This approach detects fraud that would not be found using the more common single-card transaction analysis.

1.1. Cash-card transactions

This research began with a real-world problem. A prepaid-card transaction processor was increasingly experiencing fraud that was not detected by their transaction monitoring system (TMS). A prepaid card is also known as a stored-value card or cash-card. It is a branded product, like Starbucks, AT&T, or Visa, that has the equivalent of cash stored on the card.

It important to observe that such prepaid cards are not associated with a person, commonly. Moreover, cards are rarely reloaded with money. Thus, the lifetime of a card is relatively short—from months to a year, for example. Therefore, there is little information from which to create a card model.

The company experiencing fraud is the international card processor, Card Communications International (CardCom[1]). CardCom has a distribution network of more than 75,000 retail locations. It was the first national point-of-service-activation (POSA) and distribution partner for all major wireless telephone carriers in the USA. CardCom provides many services and products.

CardCom's TMS was typical system in that transaction attributes were checked against threshold values—values outside of the specified ranges are considered potential fraud. Consider this illustrative rule: *if a terminal sold more than $3000 of a specific product* (e.g., *a $25 AT&T card) then fraud may have occurred* (and thus, any subsequent transactions from the terminal shall be voided). Initially, this rule-based approach worked well. However, as the ruleset grew, the effort of maintaining the TMS grew too, and consequently the accuracy to detect fraud fell.

The company's rule-based TMS was based on detailed analysis of the transaction histories of each merchant, store, terminal, and product, as well as season (e.g., holiday) and sales specials. The result is hundreds of rules for the different combinations. Moreover, the thresholds used in the rules are derived from transaction histories, and thus require constant update to be current. Specifying, updating, and monitoring the rules themselves is a complex and

* Corresponding author.
E-mail addresses: wrobinson@gsu.edu (W.N. Robinson), aaria@gsu.edu (A. Aria).

[1] CardCom is a pseudonym for the real, international card processor based in Georgia, USA.

http://dx.doi.org/10.1016/j.eswa.2017.08.043
0957-4174/

burdensome chore. Thus, while the rule-based attribute threshold technique is simple in theory, its use in practice is overly complex.

CardCom increasingly faces a new kind of fraud, where individual transaction attributes are within normal ranges, but a sequence of many small transactions leads to big fraud. Consider a fraudster who takes a bundle of 50 cards and activates them in sequence. Such real cases have occurred, for example when an employee is engaged in fraud. Each activation fails to trigger a fraud detection rule. Nevertheless, the sequence of 50 cards is an anomaly, and should be considered as potential fraud. Sequence analysis is needed to find such fraud. CardCom needs to improve fraud detection, using an approach that requires little maintenance and address fraud sequences.

1.2. Approaches to transaction fraud

In the prepaid card context, the fraud detection system (FDsing a known level of detection, albeit faulty, the new TMS component should ideally work alongside the existing TMS.

A review of the literature reveals two general kinds of fraud detection techniques for transactions: supervised and unsupervised methods. In supervised methods, samples of both fraudulent and non-fraudulent records are used to construct models, which classify observations as either fraudulent or non-fraudulent (Bahnsen, Aouada, Stojanovic, & Ottersten, 2016). Unsupervised methods require little or no prior classifications to identify anomalies, which are subject to subsequent review for the determination of fraud. Rule-based models illustrate the supervised approach. The rules can be automatically derived from training data or directly specified by experts, often to counteract a recent unidentified fraud case (Sánchez, Vila, Cerda, & Serrano, 2009). Other supervised approaches include neural networks, Bayesian models, genetic algorithms, etc., which use machine learning to derive a model of common transaction entities. Unsupervised methods include monitoring unusual transaction characteristics, such as unusually expensive purchases or unusual locations or merchants. A single credit card typically has a near linear slope of cumulative credit spending. Thus, a significant rise above mean slope triggers the threshold for fraud consideration (Hand & Blunt, 2001).

These prior approaches do not meet the four prepaid card criteria. As noted previously, the rule-based method is overly complex for the many prepaid card contexts. It may be improved through automated rule generation; however, this requires a transaction history. Transaction history trained models can be adapted to the prepaid card context, but must address two issues. First, note that card history is central to most of these techniques. Unfortunately, prepaid cards have little history, as they are often discarded after their value is used. Additionally, fraud often occurs near the time of purchase, when the card has no consumer transactions. Alternatively, this approach may be adapted to model the terminal, store, and merchant histories, although we are not aware of any examples. We call this the *store-centric approach*. A second issue is concept drift, which occurs when the real behavior moves away from the modeled behavior. For example, past transactions can be classified as high-, medium-, or low-cost based on the user's transaction history (Srivastava, Kundu, Sural, & Majumdar, 2008). As the user's wealth or inflation increases over time, fewer transactions will be labeled low and medium, while more will be labeled high, and thus the labels are less informative. This drift can be addressed by generating a new set of classified labels whenever the distribution of current labels fits the data poorly. Some approaches consider such concept drift.

Sequential fraud is yet another concern. Many sequence-based analyses consider a valid prefix sequence and then identify a single new transaction as fraudulent. In contrast, a sequence of new transactions may be anomalous, while any single transaction within is valid. Rules can specify sequence characteristics, such as *if a store sells more than 50 of a product in sequence, then fraud may have occurred*. The variety of sequential frauds makes such rules numerous and therefore complex to maintain. Moreover, such rules depend on data windows that are preprocessed to present the necessary metrics. Fraud models may be trained, but again they require sequence analysis, such as provided by Markov models.

1.3. Finding fraud with little history

The store-centric approach introduced in this article uses a little transaction history to detect fraud. It satisfies the four prepaid card context issues: (1) automated model maintenance, (2) little transaction history, (3) automated customization to context, and (4) high detection accuracy for sequentially fraudulent transactions. It fills this prepaid card fraud-detection gap. The approach relies on sequence analysis provided by hidden Markov models (HMM), which are automatically created and compared around the most recent store transactions. HMM software is commonplace; for example, there are open-source versions for Java and R. The idea is to use an HMM modeler for the data. Then, the HMM divergences will raise alerts whenever a threshold is reached. The HMM divergence method provides a baseline of sequence analysis that can easily be added to existing fraud detection systems. The approach is demonstrated using actual prepaid card data and associated real fraud cases. Experiments using the dataset explore method parameters, including number of fraud incidents, fraud threshold, and window size. The findings indicate that the method works well under a variety of conditions. However, it is not intended to replace all fraud techniques. Instead, it provides an effective baseline for the construction of a more comprehensive TMS.

1.4. Article overview

This article continues with a summary of related fraud detection systems. Next, we introduce a well-known approach, HMM's for card histories, for comparison with our store-centric, windowed-history approach. The following sections present our hypotheses, data collection, experiments, and findings. The article ends with a discussion of the approach and conclusions.

2. Background on finding transaction fraud

2.1. Related work on fraud detection systems

A fraud detection system (FDS) reviews behaviors involving a financial instrument, including its transactions, to identify unusual behavior and classify it as fraudulent (Phua, Smith-Miles, Lee, & Gayler, 2007). There are a number of literature surveys on these systems (Bolton & Hand, 2002; Kou, Lu, Sirwongwattana, & Huang, 2004). Here, we highlight a few approaches to a FDS.

Dempster–Shafer theory is use to combined information sources to classify transactions (Singh, Shukla, Rakesh, & Tyagi, 2011). For example, a cardholder activity profile characterizes transaction and shopping behavior. A rule-based component measures the degree of fraud in a transaction. Then, the Dempster–Shafer theory is applied to combine several such information sources to derive an overall belief (Panigrahi, Kundu, Sural, & Majumdar, 2009). Finally, a Bayesian learner can be applied to weaken or strengthen this belief using labeled genuine and fraudulent transactions. Similarly, Krivko (2010) combines expert rules with unsupervised individual profile models to build a hybrid detection model. The general idea is to improve detection by combining models, with the preceding combining unsupervised with supervised models, while Louzada and Ara (2012) combines multiple supervised classifiers.

Table 1
Issues in fraud detection techniques.

	Dempster-Shafer theory	Neural Network	Bayesian Network	Genetic Algorithms	Artificial Immune System	K-nearest neighbour Algorithm	Support Vector Machine	Decision Tree	HMM	KL Divergence
Requires training	High	High	High	High	Low	High	High	High	High	Low
Compute-intensive	High	High	High	High	Medium	Low	Medium	Medium	Medium	Medium
Requires data pre-processing and pre-analysing	High	Medium	Medium	Medium	Low	Low	Medium	Medium	High	Low
Concept drift addressed	Low	Medium	Medium	Medium	Medium	Low	Medium	Medium	Medium	High

An approached using neural networks and data mining was developed by Brause, Langsdorf, and Hepp (1999) for fraud detection purposes. This approach has the advantage that it does not require supervised learning to update. However, neural networks are computing intensive, which is a drawback (Raj & Portia, 2011; Zareapoor, Seeja, & Alam, 2012). SODRNN is another unsupervised approach. It uses the reverse *k* nearest neighbor algorithm to detect the outliers for fraud detection. The algorithm uses a data streaming technique to scan the data only once instead of several times, which is more common (Ganji & Mannem, 2012). A genetic algorithm can be used for minimizing the transaction classification error. The approach requires several cycles tries to optimize a solution, which can cause some latency issues in FDS (Duman & Ozcelik, 2011).

Decision trees are used in some FDS (Raj & Portia, 2011). During training, a decision tree is created using training data, which includes normal and fraudulent transactions. During detection, new transactions will be matched with the tree to be classified as normal or fraudulent. A SVM (Support Vector Machine) is another supervised learning method; it uses an optimized hyperplane that classifies transactions as normal or fraudulent (Kim, Pang, Je, Kim, & Bang, 2003).

Table 1 presents a summary of important issues addressed by various fraud detection techniques. Most techniques require some form of supervised training before the system can be applied. Exceptions include the artificial immune system (Timmis, Neal, & King, 1999) and our use of using hidden Markov model divergence (KL Divergence). This is important because of *concept drift* (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014), which requires updating of the fraud model.

Over time, the characterization of fraud can change. A classic case is the credit user who never travels overseas. Then, the user makes several overseas trips. Previously, the overseas purchases would be identified correctly as fraud. Whereas, once the real trip begins, the purchases should be considered legitimate. To address this for a technique that requires training, the model must be retrained (during the trip) to address the legitimately changing context of the user. Table 1 shows that most techniques do not directly address concept drift. Taken together, required training and unaddressed concept drift leads to intermittent updating of the fraud detection system, the delay of which can result in episodes of poor quality fraud detection.

Table 1 summarizes the need to pre-process or pre-analyze data prior to constructing the fraud model. For example, to build a HMM, one technique first clusters the data to find purchase price categories (Srivastava et al., 2008). These categories model a user's purchase. As the user's purchases change over time, the modeled categories will need to be updated—if not, then the user's behavior will drift away from the modeled categories. Thus, such trained models may need to be updated to accommodate concept drift, where present.

Finally, the techniques of Table 1 differ in their computational complexity. Some, like neural networks and Bayesian networks are intrinsically computationally intensive. Our use of HMM is roughly in the middle; however, in practice, computation time is very low, and can be further improved through improved HMM algorithms (Sahraeian & Yoon, 2011).

There are a variety of other techniques used for fraud detection in general. Quah and Sriganesh (2008) mined spending patterns, and applied the resulting model to detect fraudulent transactions. Jha, Guillen, and Westland, (2012) used transaction aggregation strategy to capture the buying behavior of credit card holder and used these aggregations to identify fraudulent transactions. Sánchez et al. (2009) used association rules for fraud discovery in credit card transactions. Duman and Ozcelik (2011) use genetic algorithms and the scatter search to find fraudulent credit card transactions.

There are a number of studies that indirectly address fraud, by analyzing financial transactions. Yeh and Lien (2009) consider six different data mining techniques to predict the payment default of a credit card holders. Xiong, Wang, Mayers, and Monga (2013) used sequence patterns, obtained by sequence mining, to predict consumer bankruptcy. Nie, Rowe, Zhang, Tian, and Shi (2011) used decision tree and logistic regression techniques predict credit card churn.

These approaches to fraud focus on the individual and his or her transactions. Another, complementary approach is to review the context of the transactions—in particular, the merchant where the fraud occurs. That is the approach taken in this study. A review of the preceding studies finds that, although rarely used, analysis of the sequential patterns has been effective in modeling financial transactions. This study takes the sequence analysis approach, using HMM's to model sequences of merchant transactions.

The approach described herein fills a gap in the fraud techniques in that it does not require training or preprocessing, it addresses fraud concept drift, and it analyzes fraud sequences. Additionally, the store-centric approach uniquely addresses the anonymous, short-lived card context that is common to prepaid cards by modeling merchant locations rather than customer cards. Before we describe the technique in detail, we next summarize HMM techniques with a HMM-based FDS.

2.2. HMM background and previous work

Hidden Markov Models (HMM) have been used in applications such as voice recognition, bioinformatics, and genomics. HMM's have been used for anomaly detection, categorizing TCP packets as normal or attack (Joshi & Phoha, 2005), and categorizing system software calls (Hoang, Hu, & Bertok, 2003; Ourston, Matzner, Stump, & Hopkins, 2003). The key concept is to model behavior using an HMM. When the observed behavior varies from the modeled behavior, then is it considered an abnormal sequence. This variance-from-model concept also applies to fraud detection, where fraud is considered abnormal behavior when compared to the legitimate, modeled behavior (Srivastava et al., 2008).

An HMM is a stochastic signal model (Rabiner, 1989). In our application, the signals are sequences of discrete typed events. A HMM provides algorithms to solve three important problems:

1. *Compute the probability* that an observed sequence, O, is represented by a HMM, λ (using the Forward-Backward Procedure (Baum & Eagon, 1967)).
2. Adjust the parameters of a HMM, λ, to *maximize the fit* to an observed sequence, O (using the Baum-Welch algorithm (Baum, Petrie, Soules, & Weiss, 1970)).
3. Compute the optimal HMM state sequence that best explains an observed sequence, O (using the Viterbi Algorithm (Forney, 1973)).

In this work, we show how applying the first two algorithms help to identify abnormal behaviors from an event stream.

A HMM can be characterized as follows:

N is the number of states in the model, $S = \{S_1, S_2, \ldots S_N\}$, where S_i is a state. Time t denoted by q_t. (1)

M is the number of distinct observation symbols per state; the set is denoted as $V = \{V_1, V_2, \ldots V_M\}$. (2)

The state transition probability matrix $A = \{a_{i,j}\}$, where $a_{i,j} = \mathrm{P}(q_{t+1} = S_j | q_t = S_i)$, $1 \le i \le N$, $1 \le j \le N$; $t = 1, 2, \ldots$ Where any state can reach another state in a single step, we have $a_{i,j} > 0$ for all i, j. Additionally, $\sum_{j=1}^{N} a_{i,j} = 1, \ 1 \le i \le N.$ (3)

The emission (aka observation symbol) probabilities matrix $B = [b_j(k)]$, where $b_j(k) = P(V_k, S_j)$, $1 \le j \le N$, $1 \le k \le M$ and $\sum_{k=1}^{M} b_j(\mathrm{k}) = 1, \quad 1 \le j \le N$ (4)

The initial state probability vector $\pi = [\pi_i]$, where $\pi_1 = \mathrm{P}(q_1 = S_1)$, $1 \le i \le N$, such that $\sum_{j=1}^{N} \pi_i = 1.$ (5)

The observation sequence $\mathrm{O} = \mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \ldots \mathrm{O}_R$ where each observation O_t is from V, and R is the number of observations (6)

We use the common notation $\lambda = (A, B, \pi)$ to indicate the complete set of parameters of the model, where A, B implicitly include N and M.

To illustrate the use of HMM, consider the (Srivastava et al., 2008) study, which uses the first two algorithms to detect abnormal behavior of credit card transactions, where deviations are flagged as fraud. The approach first applies a clustering algorithm to a customer's credit-card transaction history. Clustering splits the transactions into clusters, each representing a certain price-range, or category; the clusters are low, medium, and high. The cardholder generates transaction sequences, which are analyzed as sequences of categorized purchase prices. For example, a cardholder may buy electronics in one store and groceries in another. The bank issuing the credit card has the transaction date, location, and amount, but may have limited or no information about the purchased products (depending on the information the merchant provides). Moreover, it can be difficult to infer the products, because a merchant like Walmart can sell a variety of products, such as electronics, grocery, auto parts etc. Therefore, one can model the product type as a hidden state in the HMM, whereas the purchase amount of each transaction is an HMM observation.

Fig. 1 illustrates an HMM from Srivastava et al. (2008). In the figure, the hidden states are categorized as Grocery (Gr), Electronic (El) and Miscellaneous (Mi) and the observations are categorized as low price (l), medium price (m) and high price (h). At each hidden state, there is a probability that a purchased product will fall into one of the price ranges (emission probabilities). The probability of one purchase category following another is the HMM state transition. Finally, there are initial (state) probabilities associated with the purchase categories. To create a customer's HMM with this approach, a customer transaction history is analyzed to (1) create the product price-ranges through clustering, and (2) the HMM probabilities are determined through training using the Baum–Welch algorithm (Baum et al., 1970). Once a customer's HMM is so created, then it can be used to find fraud.

The trained HMM models the cardholder purchase behavior, as transaction sequences. The probability of an observed sequence, O, is represented by:

$$\alpha_1 = \mathrm{P}(\mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \ldots \mathrm{O}_\mathrm{R} | \lambda) \quad (7)$$

With the next transaction generated by the cardholder $\mathrm{O}_{\mathrm{R}+1}$, the O_1 observation is dropped and the new observation is added to the sequence. (The sequence, O, is a sliding window of transactions). The probability of this new observed sequence, O', is represented by the same HMM.

$$\alpha_2 = \mathrm{P}(\mathrm{O}_2, \mathrm{O}_3, \ldots \mathrm{O}_{\mathrm{R}+1} | \lambda) \quad (8)$$

Thereafter, probability differences can be calculated as follows:

$$\Delta\alpha = \alpha 1 - \alpha 2 \quad (9)$$

For Srivastava et al. (2008), if the $\Delta\alpha$ is positive, it means that the second sequence probability accepting by the HMM is lower than previous sequence and thus the newest transaction is a potentially fraudulent transaction. Conversely, if $\Delta\alpha$ is negative, it means that the second sequence acceptance probability is higher and can be considered as a genuine cardholder transaction. In this case, the newly observed transaction, $\mathrm{O}_{\mathrm{R}+1}$, is appended to O, thereby updating the modeled sequence to reflect the valid customer behavior. This method assumes that single transactions may contain fraud, and when found, they are excluded from HMM so as to not model fraud. Thus, the method does not identify fraud sequences, but rather it identifies a fraudulent transaction added to a valid sequence. Experiments consider the length of O between 5 and 25 transactions, with 15 being the best, which resulted in about 80% accuracy.

This approach to fraud detection, by Srivastava et al. (2008), is card-centric. One HMM models one card for one cardholder. After each transaction, either fraud is identified or the HMM is updated. This approach can identify an individual transaction as fraudulent based on anomalies in a single-card transaction history. The performance overhead is confined to the three HMM algorithms (summarized above), in which the acceptance probably is computed (after every transaction) and the HMM is updated (after every non-fraudulent transaction). Occasional updates are required to update the customer profile of clustered prices and reconsider the number of product categories, which determine the number of hidden states.

There are two variations on this approach, which we consider next. First, anomalies can be detected by comparing HMMs directly

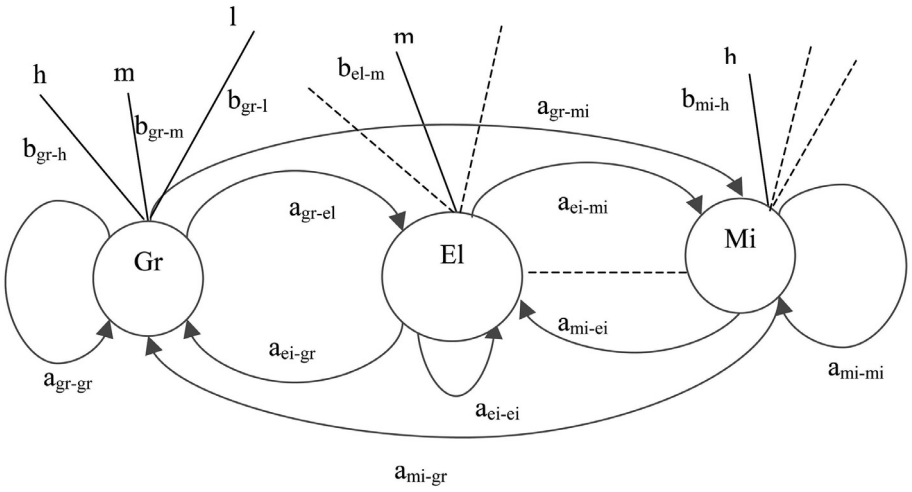

Fig. 1. HMM for credit card fraud detection (from (Chudova & Smyth, 2002)).

rather than comparing acceptance probabilities (Eq. (9) above). Second, we can take a store-centric approach to detect fraud based on behaviors in the stores. We consider these two ideas next.

2.3. Anomalies in HMMs

Anomaly identification detects significant changes in modeled events between consecutive windows of event data. The model comparison approach identifies fraud in a data window. In the preceding approach (Srivastava et al., 2008), the window size is 15 and comparison occurs with each new transaction. Each potential fraud sequence is of length one, and thus sequential fraud is not addressed (Srivastava et al., 2008). However, it is possible to have larger window sizes, accumulate many new transactions before comparing, and compare the HMM's rather than the acceptance probabilities. An advantage of this latter approach is that there are fewer calculations. Another advantage is that accumulating more than one new transaction allows for *fraud sequences* to be considered as a whole, rather than considering just one transaction, which may be part of a larger pattern. A window that is identified as potentially having fraud can be subsequently analyzed, in detail, with the data window as context, which can increase true positives and reduce false negatives. The disadvantage is that the window, not a transaction is analyzed for fraud. Thus, after a fraudulent transaction, 99 more transactions can occur before it is detected with a 100-length window. For a credit card holder, 100 transactions may be a couple of weeks; however, for a large merchant store, it's about 15 s.

HMMs can be used to identify anomalies by: (1) *comparing consecutive HMMs* generated from the observation sequences, or (2) *comparing consecutive acceptance probabilities*. We build-up to these two techniques through a series of equations, presented next.

Consider how a HMM may be applied to an observation sequence to compute an acceptance probability. The acceptance probability α_1, is defined as follows: $\alpha_1 = P(O_1, O_2, O_3, \ldots O_R|\ \lambda)$. The acceptance probability of the next overlapping sequence of length R is $\alpha_2 = P(O_2, O_3, O_4, \ldots O_{R+1}|\ \lambda)$. Now, consider comparing the two acceptance probabilities:

$$\Delta\alpha = \alpha_2 - \alpha_1 \tag{10}$$

$$\%\Delta\alpha \underline{\text{def}}\ \Delta\alpha/\alpha_1 \tag{11}$$

We can also consider non-overlapping sequences (aka data windows). Let the probability $\beta_2 = P(O_{R+1}, O_{R+2}, \ldots O_{R*2}|\ \lambda)$.

$$\Delta\beta = \beta_2 - \alpha_1 \tag{12}$$

$$\%\Delta\beta \underline{\text{def}}\ \Delta\beta/\alpha_1 \tag{13}$$

The preceding Eq. (13), which we call windowed HMM acceptance, compares acceptance probabilities of consecutive data windows. It is a generalization of Eq. (11), which is the approach taken by Srivastava et al. (2008). The difference is that the data may include non-overlapping sequences. The windowed HMM acceptance applies as follows:

1. Observation sequence O^1 is observed, $\alpha_1 = P(O^1|\ \lambda_1)$
2. Observation sequence O^2 is observed, $\beta_2 = P(O^2|\ \lambda_1)$
3. $\%\Delta\beta$ is computed
4. If $\%\Delta\beta \geq \text{Threshold}_\beta$, then a transition is recognized

Windowed KL divergence further extends windowed HMM acceptance with more precision. Windowed KL divergence (D_{kl}) compares consecutive HMMs as follows:

$$\Delta\lambda_{2,1} = D_{kl}(\lambda_2, \lambda_1) \tag{14}$$

To find the divergence between two HMMs, we apply the Kullback–Leibler algorithm Kullback, 1997), which compares two probability distributions and is often shortened as *KL divergence.* The acceptance calculation ((11) compares two acceptance probabilities, whereas KL divergence directly compares the HMM's. The acceptance difference calculation is limited to comparing two numbers (acceptance probabilities), each of whose calculation may involve only a subset of the entire HMM (Baum & Eagon, 1967). In contrast, the KL divergence compares both HMM's in their entirety. It measures the information divergence, in bits, between the two probably distributions representing the sequential data. Thus, KL divergence provides more accurate change information than acceptance probability differences (Sahraeian & Yoon, 2011). Moreover, KL divergence is not a symmetric, in that $D_{kl}(\lambda_2, \lambda_1) \neq D_{kl}(\lambda_1, \lambda_2)$. Thus, transition from normal to fraud has a different D_{kl} than a transition from fraud to normal. Thus, the precision of the KL divergence helps to reduce the number of false positives.

Given an observation sequence, O, the emission probabilities matrix, B, is generated from the observed distribution of the symbols from V found in O. In our application, this is done by counting the purchase events, V, to determine their distribution in O and then adjusting the probabilities according to the state transition probability matrix A. Thus, each observation sequence, O^i, is modeled by an HMM $\lambda_i = (A, B, \pi)$, where A and π are given, and B models the probabilities of O^i. Each HMM is improved through optimization, which updates A, B, and π for a better fit with O^i (Baum et al., 1970).

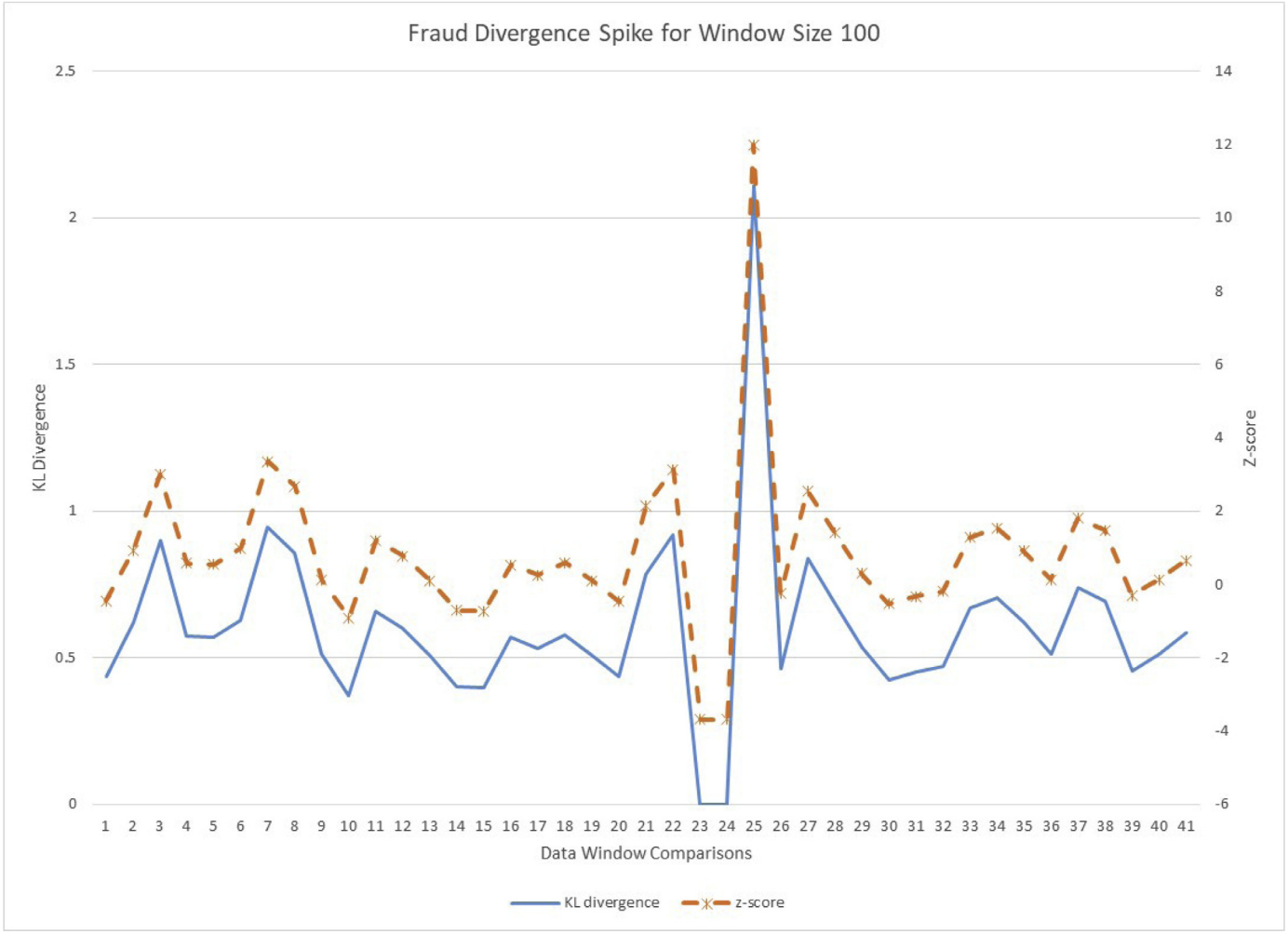

Fig. 2. An illustration of fraud spike for window size 100, one injection of length 300. KL divergence values are scaled at the left, while their z-score values are scaled at the right.

The KL divergence approach applies as follows:

1. Observation sequence O^1 is observed, $\bar{\lambda}_1 = \max P(O^1|\lambda_1)$
2. Observation sequence O^2 is observed, $\bar{\lambda}_2 = \max P(O^2|\lambda_2)$
3. $\Delta\lambda_{2,1} = D_{kl}(\lambda_2, \lambda_1)$ is computed (Kullback, 1997)
4. If $\Delta\lambda \geq \text{Threshold}_\lambda$, then an anomaly is recognized

In this study, we apply the KL divergence approach to merchant terminal data. Fig. 2 illustrates how the approach identifies fraud using a portion of real merchant data. The graph plots KL divergence for a series of data windows; each point represents a divergence between two data windows. Both the KL divergences and their z-scores are graphed. The latter standard score indicates the number of standard deviations a point is from the mean. The fraud begins at point 22 with z-score of 3.14. The next two window divergences are below the mean at −3.7 because the fraud transactions are identical and thus have little divergence. Finally, point 25 has the high z-score of 12, indicating the extreme change from consistent fraud transitions back to normal transitions. Our method identifies this fraud end. Fig. 3 illustrates finding the same fraud using a window of length 300. Notice that the points are attenuated, but the fraud spike is higher. In the following experiments, the true/false positives and true/false negatives are calculated based on finding the fraud end point. Once the end is found, then the software finds the fraud start by looking back a limited number of points for a KL divergence above a threshold.

Fraud threshold affects fraud detection. The threshold specifies the value at which the KL divergence z-score is considered anomalous, and therefore a potentially fraudulent window. Fig. 2 shows KL divergence spikes found in the analysis stream. In the figure, a z-score threshold of 3.4 would identify only the fraud, whereas a value of 3.3 would identify the fraud and one false positive (at point 7). Thus, the lower the threshold the more windows will be considered as potentially fraudulent.

3. Method

Having introduced the prepaid card context and fraud detection with KL divergence, we how turn to the store-model divergence method (SMDM). The method generally applies as follows (with parameters described in Section 3.3):

1. For each store, construct and optimize an HMM model, λ_1, from a sequence of prior transactions, O^i.
2. As new transactions occur, add them to the next sequence, O^{i+1}.
3. When the specified window size is reached, create the next HMM, λ_2, from O^{i+1}.
4. Now, compare the HMM's, $\Delta\lambda_{2,1} = D_{kl}(\lambda_2, \lambda_1)$.
5. When the divergence, $\Delta\lambda$ is greater than the specified threshold, then raise a fraud alert.
6. Finally, update the current HMM to be the most recent HMN, λ_2. Thus, the HMM model is updated with each new window.

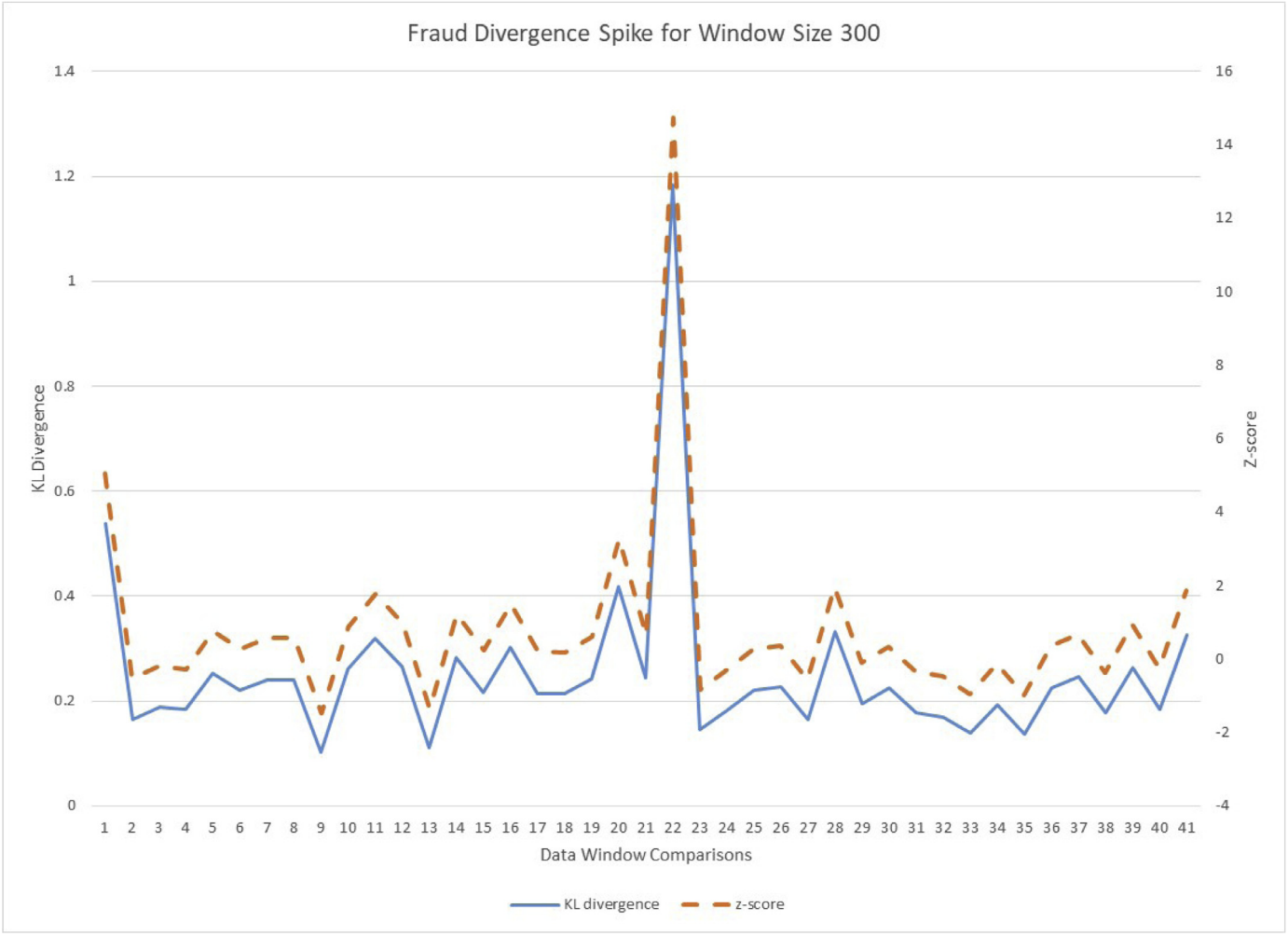

Fig. 3. An illustration of fraud spike for window size 300, one injection of length 300. KL divergence values are scaled at the left, while their z-score values are scaled at the right.

The use of HMM analysis is similar to that of Srivastava et al. (2008) introduced in Section 2.2; however, significant differences are that: (i) KL divergence is applied rather than acceptance differences, (ii) stores rather than credit cards are modeled, (iii) transactions amounts are not clustered and thus do not require maintenance, (iv) data is windowed and may be overlapping, and (v) fraud is assumed to consist of a transaction sequence, rather than an individual transaction.

3.1. Hypotheses

Given the SMDM, we now specify the hypotheses for the prepaid card context:

- **H1:** The method meets the four criteria of the prepaid card context: (1) automated model maintenance, (2) little transaction history, (3) automated customization to context, and (4) high detection accuracy for the sequentially fraudulent transactions.
- **H2:** The method finds fraud cases that were missed by CardCom's TMS.
- **H3:** The method is reasonably efficient, such that it can be applied by companies, like CardCom, using today's common computer systems.

Next, we present the experiment design that tests these hypotheses.

3.2. Model encoding

To model the data stream of product actions requires encoding for HMM's. The observation sequence, O^i, contains actions on products. The stream of product actions are converted to a set of observed symbols. These observed symbols are $V = \{product\text{-}action_1, product\text{-}action_2, \ldots product\text{-}action_n\}$, where *product-action* is the encoding of an action on a product. For example, purchase *Apple iTunes $25*, balance lookup *Apple iTunes $50*, add cash to *AT&T $25*, etc. Note that most products have the price embedded in the product type. Additionally, other actions are possible. However, with the focus on fraud, herein, we only present an analysis of the purchase related actions.

HMM's have a hidden state, which partitions the transition probabilities. In the SMDM, the hidden states model the sales context: normal, weekend, and holidays. The transition matrix, *A*, begins with equality probabilities among three different states of sales: normal, weekend and holidays. The initial state probabilities, π, are also equal. The emission probability matrix, *B*, is defined by the sales states x the product types. Thus, this is {normal, weekend, holiday x product types}, where each value represents the probability of selling one of a product while in one of the three sales states. Thus, a store's HMM $\lambda_s = (A, B, \pi)$, where the emission probabilities of *B* are derived from the transaction history and then optimized with the Viterbi Algorithm (Forney, 1973).

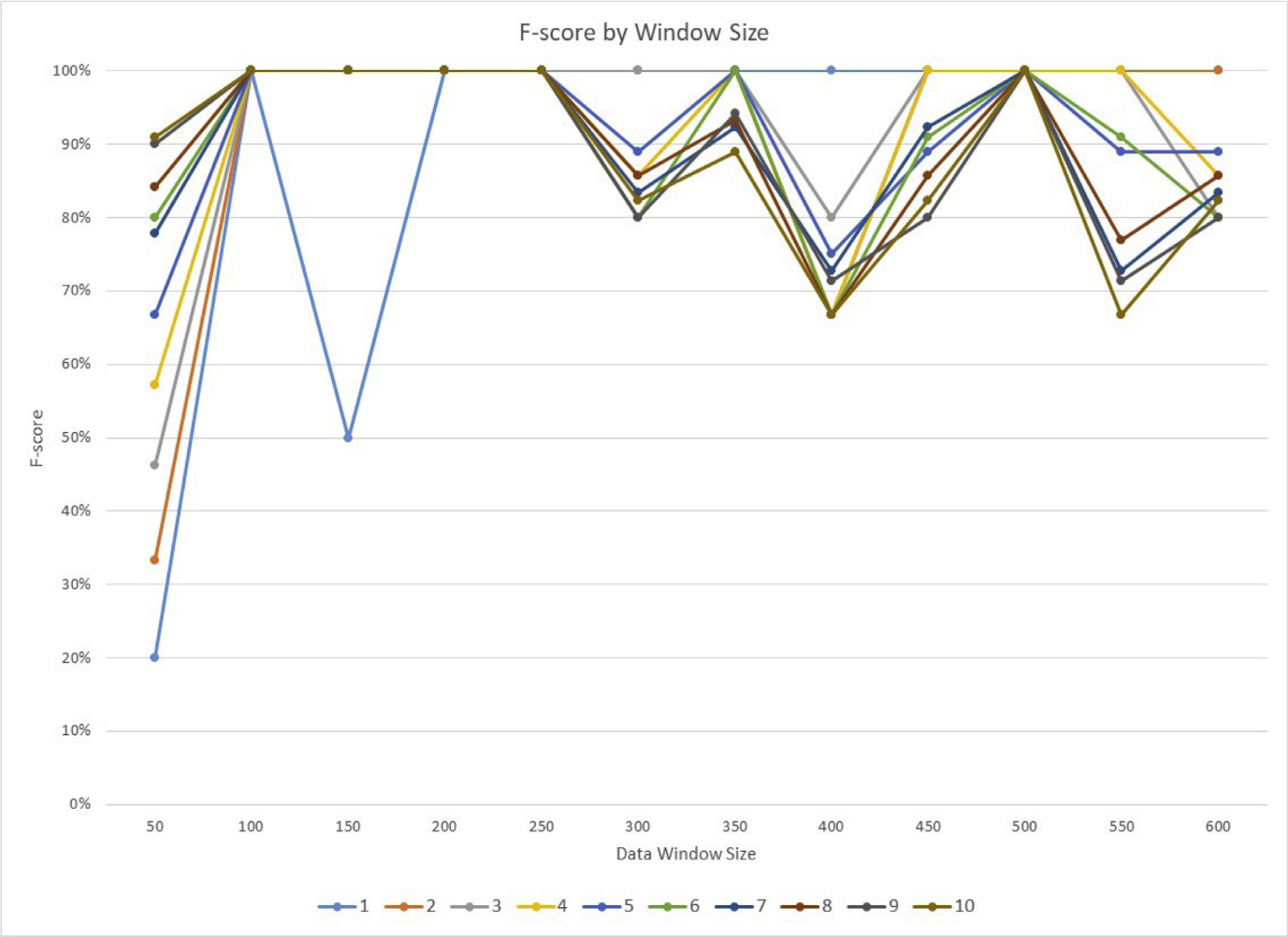

Fig. 4. F-scores for fraud case 1 with number of injections varying from 1 to 10; x-axis is window size.

Table 2
SMDM experiment parameters.

Parameter	Description	Range of values
Number of products	The number of distinct products in the purchase log.	15–30, by 1′s → 15
Number of incidents	The number of incidents randomly injected in the experiment data.	1–10, by 1′s → 10
Window size	The number of records in each data window.	100–600, by 50′s → 12
Threshold	The z-score value above which change is considered fraud.	3–8, by 1′s → 6
Overlap	The percentage a data window overlaps with the next data window.	0 and 50% → 2
Diffusion	The number of transactions interleaved with the fraud sequence.	0–5, by 1′s → 6

3.3. Experiment design

The SMDM experiments vary parameters presented in Table 2. Three actual CardCom fraud cases serve as the basis for generating the experiments. Excluding window overlap, number of products, and diffusion parameters (which were ran less extensively) there are 720 experiments for each of the three cases. In total, herein, we report on over 2000 experiments. Evaluation criteria include fraud detection qualities and computing performance. If SMDM detects fraud in the experiments, then hypotheses H2 is confirmed. If the processing time is acceptable, then hypotheses H3 is confirmed. Finally, if SMDM detects fraud in a variety of conditions with no maintenance, then hypotheses H1 is confirmed.

To prepare for the experiments, a review of CardCom fraud was conducted. First, interviews with the designers of the CardCom TMS were conducted to establish the kinds of undetected fraud. Three fraud cases, derived from real incidents, were described; for each, a fraudulent data sequence was obtained for use in experimentation. Second, data was reviewed to establish normal and fraudulent data patterns. This resulted in a selection of three million records for experimentation. Together, the sampled data and the fraudulent data sequences were used for experimentation. The collected data is described next.

4. Data collection

To illustrate the data, we briefly explain a common business scenario for one type of cash card. The nuances of this real scenario will reveal how certain kinds of fraud have occurred. This in turn justifies our focus on store-centric fraud analysis. Moreover, although the potential fraud described herein applies to CardCom services, such analysis applies to other card processors.

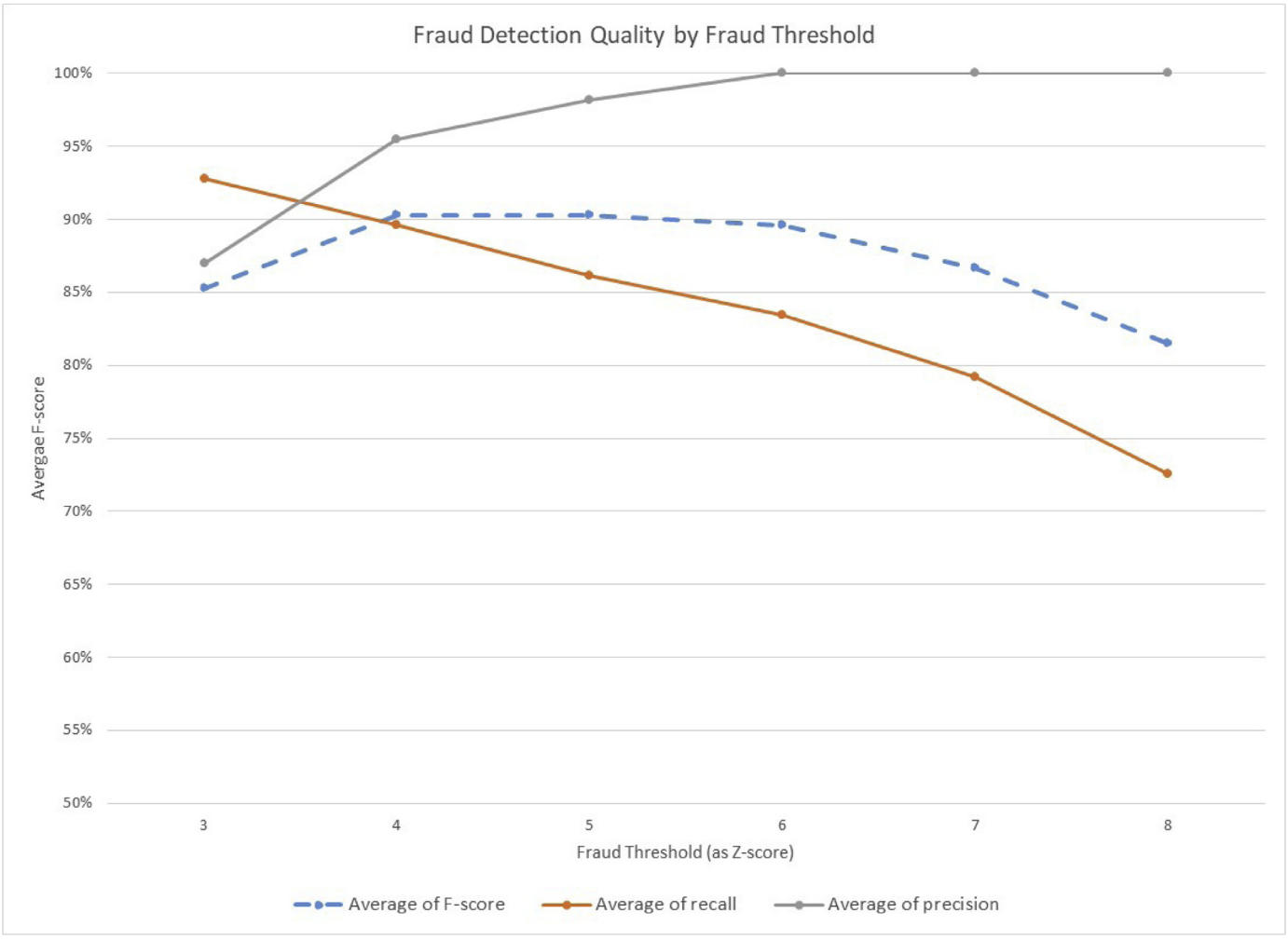

Fig. 5. Thresholds and the mean F-score, precision, and recall for the case 1 experiments. Each point represents 10 different fraud experiments over 12 different window sizes.

CardCom cards can be sold at any merchant having access to internet or dedicated data transfer lines. CardCom cards include branded financial cards like Visa or Master Card. To purchase, the customer selects a card at a merchant, takes the card to cashier, and provides funds (e.g., cash) to be associated with the card. The cashier scans the bar code, accepts the cash, and transaction information is exchanged between the merchant and CardCom. At CardCom, some business logic is executed and transaction information (e.g., card denomination) is transferred to the financial institution, which will update the account (with funds). This completes the sale. Now, the customer can open the package to access the card. Henceforth, the customer can use this card, until there are no funds remaining on the card.

All transactions are logged at CardCom. As part of logging, the following information is retained:

- TransactionId: The unique transaction Id.
- VAN16: The Vendor Account Number, which is the card number that is encoded in magnetic stripe or in the scanning bar code.
- Date & Time: The transaction date and time.
- OpCode: Represents the action taken on the card, such as activation or recharge, etc.
- NodeType: Indicates that where transaction originated.
- RootNode: A number assigned to the transaction source; for example, the terminal number within a store.
- Amount: The amount of the transaction.

Table 3
Descriptive statistics of the experiment data, containing 277,721 rows.

	Min	Max	Mean	Std. deviation
Amount	10	500	23.993	16.723
Product id	1	30	11.302	8.404
Seconds between transactions	0	28	0.153	0.494

- SerialNumber: The card serial number.
- CardId: The ID, which can be joined with product related tables, to provide the product information.

Among all the information logged, this information provides sufficient context our discussion. Notice that cards are processed on the CardCom network. Thus, CardCom can review all card operations, such as purchase, knowing the merchant, processing location, and other potentially useful information, which can be modeled for fraud detection.

The experiment data consists of logged transitions, with attributes as described above. The collected data, from 2012, includes about three million transactions. These were filtered to include only one merchant, the largest national retail chain in the USA. These were further filtered to include one store. Finally, the products were filtered to include only the top-30, because the remaining products occupy less than 0.8% of the transactions and histori-

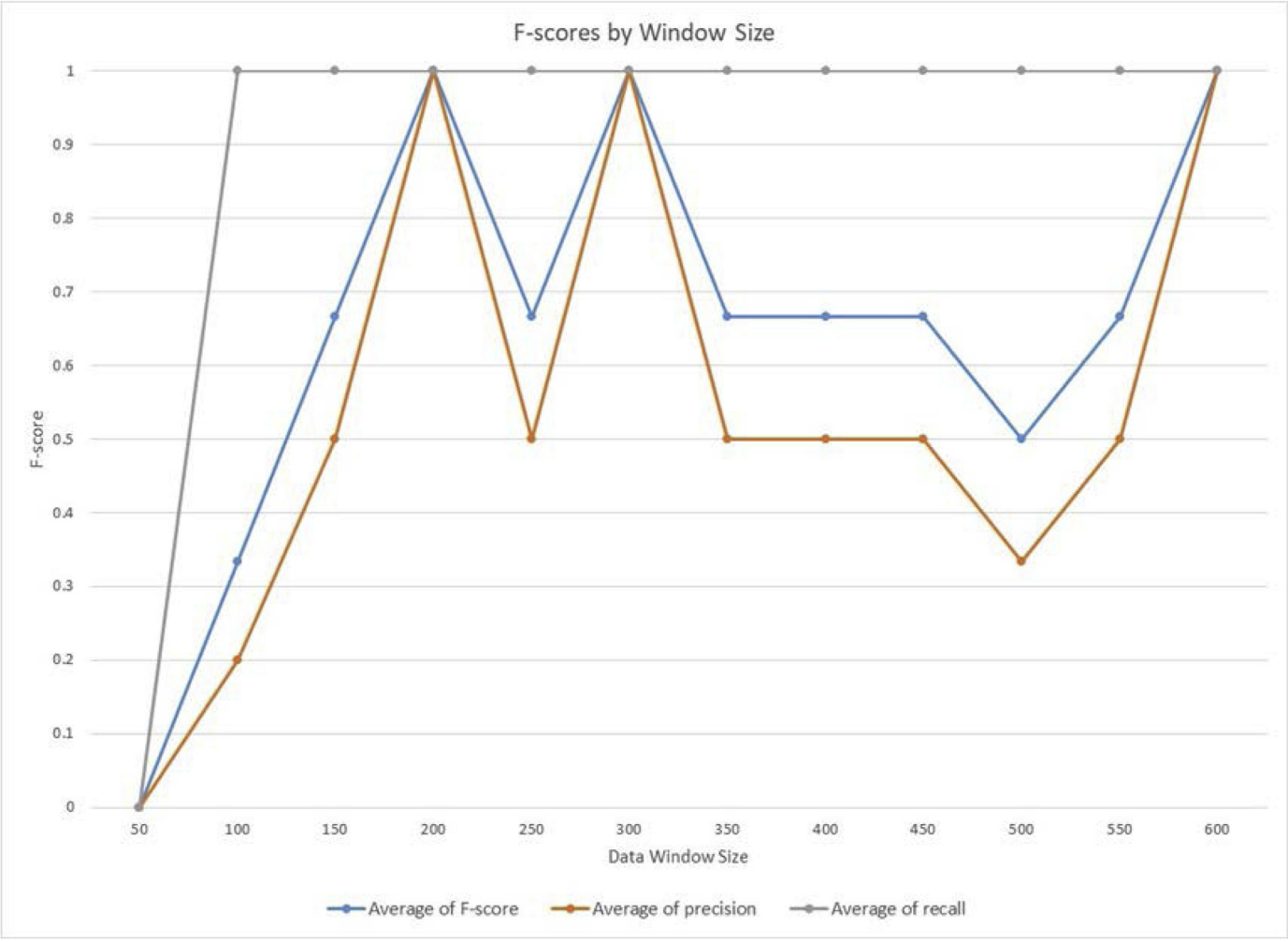

Fig. 6. Offset of 175 transactions showing F-score, precision, and recall for one fraud incident, case 1.

cally have no fraud. Table 3 summarizes the descriptive data statistics of the experiment dataset.

5. Experiments

Each experiment follows a common form: (1) the experiment parameters are set, (2) the experiment is run, and (3) the evaluation is collected. Two parameters impose constraints on the data: (1) number of products, and (2) number of incidents. Rather than review all the data looking for segments that met these requirements, we instead used the same baseline data for all experiments and manipulated these parameters. Thus, we filtered the data to meet the specified number of products and injected real fraud signatures to meet the specified number of incidents. This is a common approach in fraud analysis (Lundin, Kvarnström, & Jonsson, 2002; Natella, Cotroneo, Duraes, & Madeira, 2013). Also, by injecting fraud we know where the fraud is so that we can precisely calculate the quality measures, such as true and false positives.

The thousands of experiments derived from variations on the three fraud cases demonstrate an F-score above 0.9 for window sizes above 200. Experimenting with data stream samples provides guidance on selecting appropriate threshold and window sizes. After focusing on window sizes of 200 and 500, with 50% overlap and a z-score threshold of 5, the mean F-score improved to near 1.0. Next, we introduce the fraud cases and the experiments.

5.1. Three fraud cases

These real fraud cases have financial implications for the stores and CardCom. Fraud originating in the store terminals ultimately results in funds lost by the store, whereas fraud originating in the CardCom network results in funds lost by CardCom. The following three fraud cases ensures that both store terminals and the CardCom network are tested.

- Case 1: Individual Store Terminal Access

In this scenario, an intruder accesses a store terminal (after store hours) and begins swiping many CardCom cards using the terminal (for activation and fraudulent purchase). The fraud test data consists of a single store terminal having an uninterrupted sequence of *sp* single-product cards purchases.

- Case 2: Group Store Terminal Access

Test scenario two consists of *ni* intruders infiltrating a store; each one picks a bundles of products (CardCom cards) and starts swiping them on the store terminals. Each processed *gp* cards, uninterrupted. The products differ, so the sequence of product purchases are interleaved in the merchant log.

- Case 3: Networking Hacking
  Test scenario three consists of one hacker breaking into the CardCom network and using a computer program to activate

Fig. 7. F-scores for fraud case 1 with number of injections varying from 1 to 10 (x-axis), rolling windows (depth axis) have 50% overlap.

(purchase) *np* products of *nt* different product types, in sequence.

The following three sections summarize hundreds of experiments with each of the fraud cases as a basis.

5.2. Case 1: individual store terminal access

This fraud sequence consists of 300 single-product cards purchases. In Fig. 4, each plotted point is the F-score for one of 120 experiments with threshold of 5.0 and varying window sizes and number of fraud incidents. The mean F-score of all windows and number of incidents in Fig. 4 is 0.9.

The fraud task is to label correctly items as fraud (aka true positive, TP). Mistakes occur when non-fraud items are labeled incorrectly as fraud (aka false positives FP), or vice versa. With these terms, we can apply common quality measures, including *precision* (TP/TP + FP), *recall* (TP/TP + FN), and *F-score* (2 × (precision × recall) / (precision + recall)). These are common in the evaluation of FDS (Stolfo, Fan, Lee, Prodromidis, & Chan, 2000).

Fig. 4 shows downward drops in F-score. The line labeled 1, denoting one fraud incident in the 277,721 transactions, has a drop with window size 150. Such drops coincide with increases in the number of false positives, which is reflected in decreases in precision. Many windows have a similar pattern, where the false positives bring down the F-score.

Fig. 5 graphs the mean F-score, precision, and recall for all case 1 experiments with varying thresholds. The graph illustrates the common inverse relationship between precision and recall, as well as the maximum F-score between 3 and 4.

Fig. 4 also shows that window size plays a role in F-score, with larger windows generally associated with higher F-scores. Optimizing the window size is difficult because it depends on the sequential fraud length, which is unknown and will vary. Recall that Figs. 2 and 3 show how fraud is detected with window sizes 100 and 300, with the larger windows showing a larger spike. A smaller window discovers fraud sooner, but even a large window of size 600 transactions ends after an average of 92 s in the case of our large merchant.

Window offset also plays a role in fraud discovery. For example, if the fraud begins exactly at the beginning of a fraud window, then the KL divergence between the two data windows will be larger than if the fraud begins in the middle of a window. This can be addressed by starting data windows at different transactions. Beginning with the first transaction has transaction offset zero, while beginning after *n* transactions has transaction offset *n*. Compare the precision drops of Fig. 4 with those of Fig. 6, which has an offset of 175 transactions. Notice that the drops are shifted right, with the larger window sizes.

To ensure fraud is detected, multiple data windows can be ran by concurrent processes, each process running a different window size. Another approach is to use a rolling, overlapping window. For example, given a window size of 100 with 50% overlap, the windows would be indexed as 1–100, 50–150, 101–200, etc.

After a reviewing the results of window sizes, offsets, and thresholds, we settled on window sizes of 200 and 500 with 50%

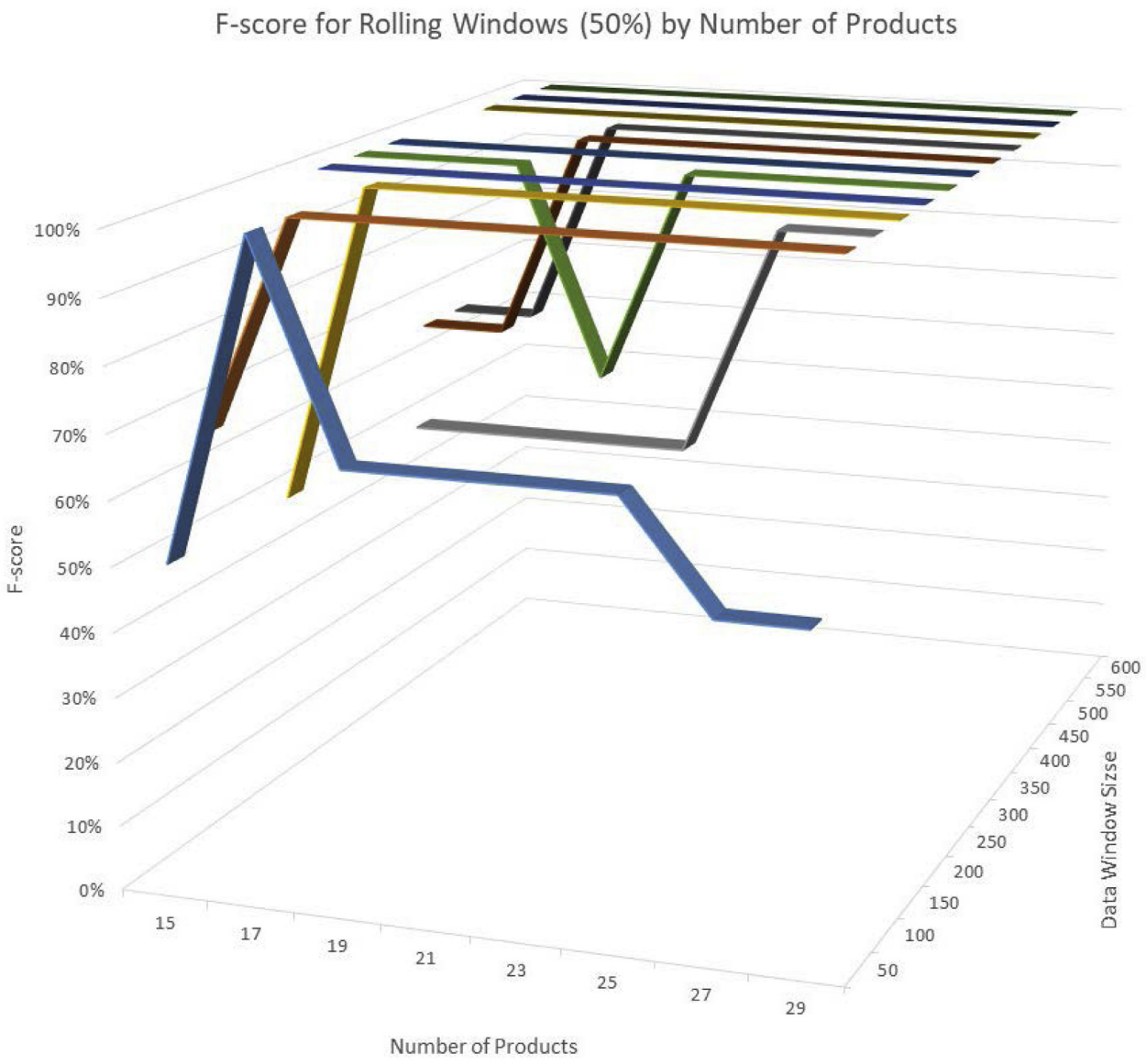

Fig. 8. F-scores for fraud case 1 with 1 injection and products varying from 15 to 29; depth-axis is window size.

overlap. These were chosen because they: (1) have good F-scores for the variations of fraud case 1; (2) have recall scores of mostly 1.0; (3) have sufficient data to model both small and large frauds, which can be completed within a reasonable time, of 92 s on average; and (3) have varied factorizations of the data—the initial data indices are 1–200, 100–300, 1–500, and 250–750. Fig. 7 plots the F-scores window sizes from 100 to 600, with 50% overlap, with the number of fraud injections ranging from 1 to 10. As an illustration, the point at window 100 and fraud 1 represents the experiment for window size 100 with 50% overlap and one fraud injection. The mean F-score of the experiments is 0.99.

The number of products in the data stream is the last parameter we will consider here. All prior experiments considered 30 products. We experimented with reducing the number of products. Fig. 8 shows the results of decreasing the number of products, from 29 to 15, using the same threshold as Fig. 4. The least frequent products are removed first. The pattern is similar to the one-fraud injection plot of Fig. 4, where the small windows score poorly, followed by better F-scores for the larger windows. However, Fig. 8 shows that the fewer the products the lower the F-score, except with windows sizes above 500. Together the products, product sequences, and window sizes determine the distributions that are compared using KL divergence. Thus, reducing the products reduces the transitions, simplifying the distributions, and thus making it more difficult to detect their divergence. Conversely, as the number of products increases the processing requirements increases.

One last consideration is sequence diffusion. Now, we consider the case where a fraud sequence is interspersed among valid transactions. Recall, in this study all transactions are valid—it is the unusual sequencing that provides the fraud signature, rather than the transaction attributes. Table 4 shows how diffusion reduces the ability to detect fraud, with F-score falling to an average of 0.83 with one transaction interspersed for every fraud transaction and 0.33 with five transactions interspersed between every fraud transaction. To make it comparable with the prior figures, the fraud

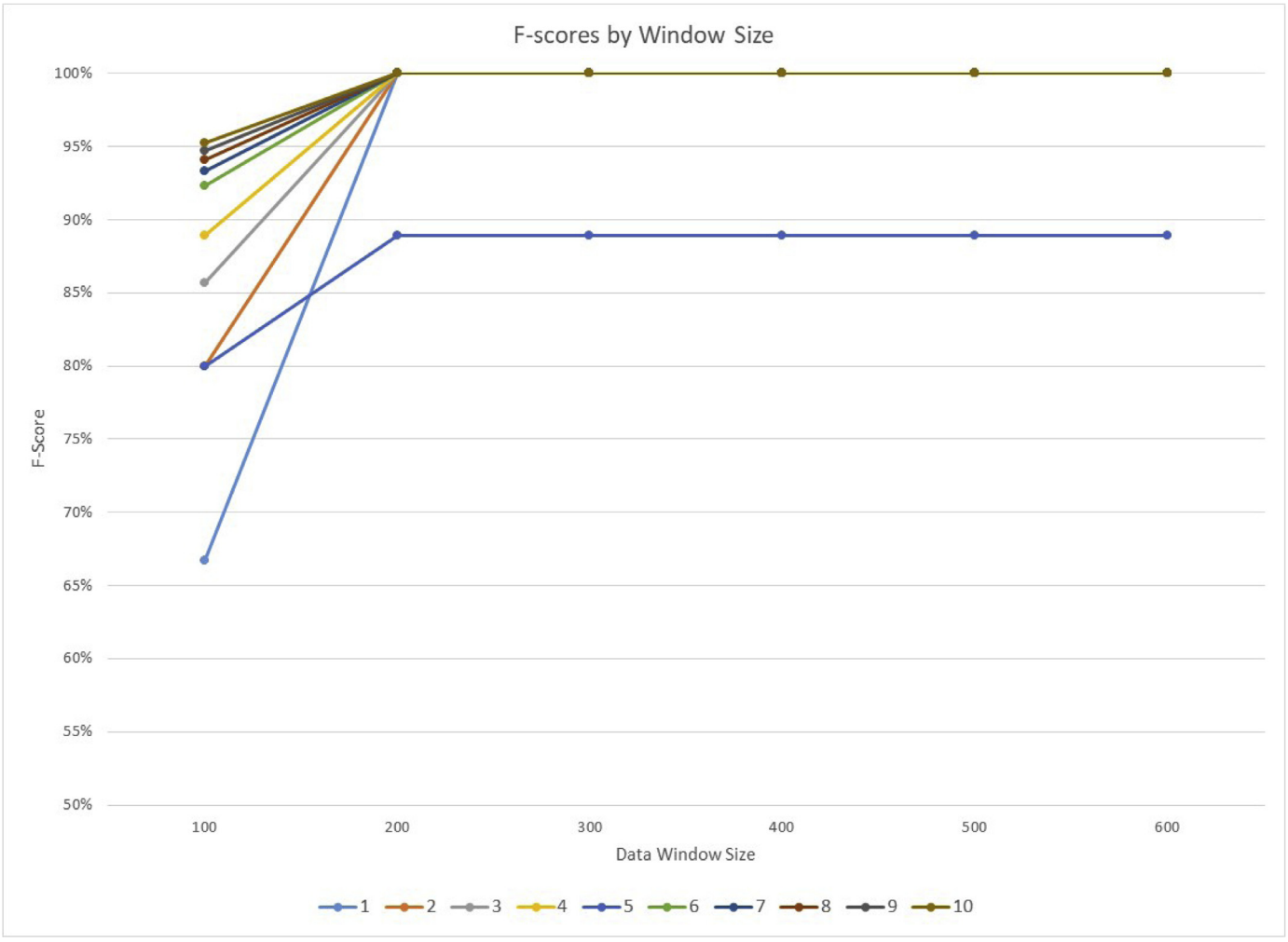

Fig. 9. F-scores for fraud case 2 with number of injections varying from 1 to 10 (x-axis).

Table 4
F-scores for fraud case 1 with 1 injection and intervening transactions varying from 1 to 5.

Window size	1	2	3	4	5	Average
100	0.67	0.67	–	–	–	0.27
200	0.83	0.67	1.00			0.83
300	1.00	1.00	0.67	–	–	0.53
400	0.83	0.67	1.00		1.00	0.88
500	0.67	0.67	1.00	1.00		0.83
600	0.83	0.67	1.00	1.00		0.88
Average	**0.81**	**0.72**	**0.78**	**0.50**	**0.33**	**0.70**

threshold remains at 5.0; dropping the threshold to 4.0 improves the results only slightly (e.g., 0.36 for five transactions). With interleaved transactions varying from one to five, a window size of 600 averages a 0.88 F-score, which is good when compared with similar results in this domain.

5.3. Case 2: group store terminal access

This second fraud case two consists of two intruders infiltrating a store. Each intruder picks bundles of products (packs of 25 or 50 CardCom cards) totaling 150 cards each, for a grand total fraud length of 300. The intruders swipe the cards on the store terminals (for activation and fraudulent purchase). Each processes their cards, uninterrupted; however, the two products differ and the sequence of product purchases becomes interleaved in the merchant log.

Experiments for fraud case two are shown in Fig. 9 which plots the F-scores for window sizes with 50% overlap and a threshold of 5.0. Window sizes smaller than 200 have increased false positive causing their lower F-scores. Windows of size 200 and more are perfect, except for the case of five fraud injections, which has an F-score of 0.89, because of one false negative per window.

5.4. Case 3: networking hacking

This third fraud case consists of one hacker breaking into the CardCom network and using a computer program to activate (purchase) five products of five different product types, in sequence. The total fraud length is 300.

Experiments for fraud case three are shown in Fig. 10, which plots the F-scores for window sizes with 50% overlap and a threshold of 5.0. The F-scores are 1.0, except for the small window sizes and the cases of 8–10 fraud injections with window size 600, which are 75%, 80%, and 86%.

5.5. Performance analysis

Next, we present experiments showing how performance is affected by the parameters of window size, number of products, and number of fraud cases. The time values should be considered rel-

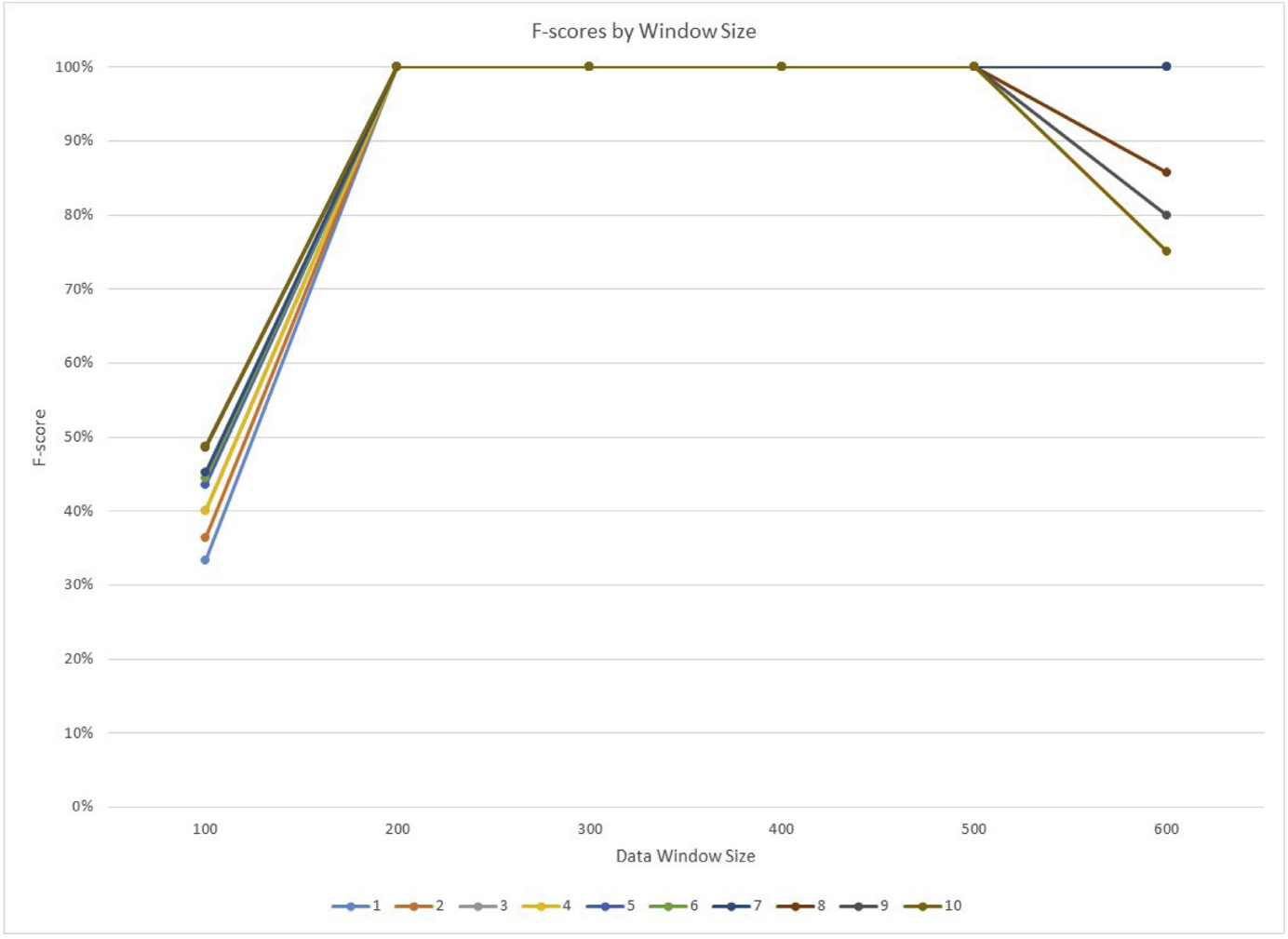

Fig. 10. F-scores for fraud case 3 with number of injections varying from 1 to 10; x-axis is window size.

ative values, as these experiments were conducted on a desktop computer.[2]

Fig. 11 shows the effect of window size on analysis time, using data from the Fraud Case 1. The total calculation is comprised of the HMM model construction time (mean of 7.7 ms) and the KL divergence time (mean of 87.6 ms). Analysis time is nearly constant for both. The chart is a bit misleading with respect to the total processing time. A small window size will have more calculations (for the windows and their comparisons) than a larger window size for the same data. Thus, 200 transactions processed by windows of size 25 is processed in $(8 \times 7)+(7 \times 89)$ ms, whereas a window of size 100 is processed in $(2 \times 7)+(1 \times 87)$ ms.

Theoretically, processing time increases because of the encoding of the emission probability matrix, *B,* which is defined by the sales states x the product types—larger matrices take more memory and time to process. However, our Java implementation allocates memory based on the specified matrix size, which explains the near linear performance.

6. Experiment findings

We now return to the three hypotheses of Section 3.1. We begin, in reverse order, with hypothesis H3. The average time to process two windows is 103 ms (using a desktop computer). Assuming a window size of 500 for the large merchant where a transaction occurs every 0.153 s, the total time to obtain and process two data windows is 153 s + 103 ms. Thus, the fraud detection time is less than processing a single transaction. This is reasonably efficiently using today's common transaction systems.

Hypothesis 2 asserts that the method finds fraud cases that were missed by CardCom's TMS. The preceding experiments show that this is true under a variety of parameters and fraud scenarios. In particular, recall is nearly 100% when the threshold is low, which does increase false positives. Nevertheless, an overall F-score near 1.0 is good in this context.

Finally, Hypothesis 3 asserts that the method meets the four criteria of the prepaid card context. Once configured, the method uses windowing to adapt to changing purchase patterns. The good F-score over a variety of circumstances demonstrates its effectiveness at finding sequential fraud occurrences.

7. Experiment limitations

Potential limitations of the experiments include the data sample and the fraud sample. In particular, different samples may produce different results. The data sample of 277,721 records was drawn from years 2010–2012. Months were selected based on their higher volume, and then data was randomly drawn for one store of a large merchant. This sample was guided by the CardCom fraud experts, who indicated that more fraud occurs at large merchants during times peak transactions. Nevertheless, the method

[2] Windows 10 64-bit, W3680 @ 3.33 GHZ with 12GB memory.

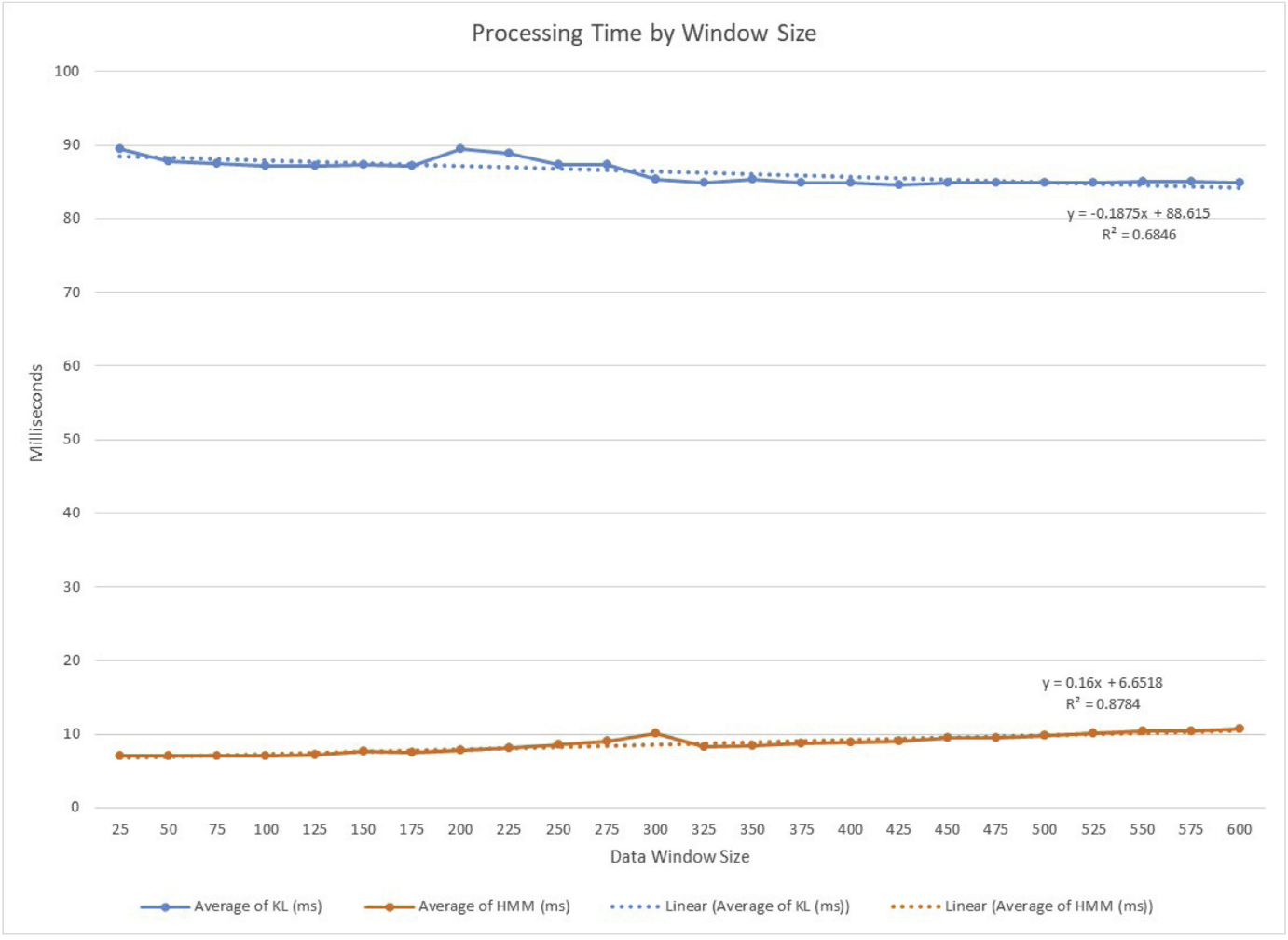

Fig. 11. Average time (milliseconds) to model and difference HMM's by window size.

has shown similar results when applied to smaller sample sizes randomly selected and then injected with fraud.

The fraud cases were drawn from real, troublesome occurrences. Their variations included multiple instances (up to 10) and interleaving a fraud sequence with legitimate transactions. In all cases, the KL divergence was able to identify the anomalous from the normal transactions. However, it's likely that a small number of fraudulent transactions widely distributed over the data stream would go unnoticed. Moreover, it's likely that few, if any methods, would find such fraud because they appear to be legitimate.

8. Discussion

CardCom developed an FDS to combat known cases of fraud that have occurred within the CardCom network. Many of their fraud instances have be determined through transactional log analysis, retrospectively. CarCom's Transaction Monitoring System (TMS) codifies these instances as rules.

The rules are of two forms: (i) fraud patterns based on prior fraud cases and (ii) transaction statistics of sales volume (dollar amount of transactions) and transaction volume (number of transactions) for a particular merchant location in real time, as well as daily-sales transactions averages. When a fraud alert is triggered, actions vary from a warning to a shutdown of transactions for a product or product category at a location.

The rules rely on a running window of data, in which the measures are computed. Thus, an individual transaction is not identified as fraud, but instead a data window is identified as containing potential fraud. Moreover, the rules do generate false positives, especially during special sales events or holidays.

The store-model divergence method (SMDM) introduced herein provides a supplemental technique to combat fraud sequences. The approach is simple to apply: (1) download open-source HMM software, (2) specify transaction encoding, (3) optimize the parameters for the data, and then (4) enable it. The method provides a maintenance-free baseline for sequential fraud identification and requires little in computational resources.

The SMDM approach works well with a variety of parameter settings. For CardCom, the default setting is a 50% rolling window size of 500 and a z-score threshold of 5, resulting in F-score averages near 0.94, which is an improvement over prior methods that identify a fraudulent transaction added to a valid sequence. Moreover, the SMDM approach identifies fraud sequences. Nevertheless, the parameters should be customized to the data context through experimentation.

CarCom's rule-based FDS requires constant, complex maintenance. Many approaches require model updates as the stream of valid transactions drifts away from the fixed model. As the drift occurs, more fraud detection mistakes will arise. For example, Srivastava et al. (2008) relies on analyzing and clustering cardholder transactions to create observation labels for the HMM. As the user's behavior changes, the transaction labels will become invalid, and with it the transaction model. In contrast, the SMDM

approach builds new HMM's with each window to remain current. This approach works for store models, where the volume is higher than an individual's credit card. Generally, the technique is most applicable to ephemeral financial instruments, such as stored-value cards, where there is sequential fraud and building a model of the customer has limited value because of its anonymous usage or its limited lifetime. The approach is appealing to prepaid cards, where thousands of cards are activated, used, and deactivated creating the potential for significant sequential fraud.

9. Conclusion

Electronic transactions, especially involving consumer financial cards, are prone to fraud. Various techniques have been applied in Fraud Detection Systems. Many commercial systems rely on relatively fixed rules about aggregated data. Such systems require complex maintenance, given the dynamics of the merchants, products, and fraud schemes. In the prepaid card context, four issues must be addressed: (1) automated model maintenance, (2) little transaction history, (3) automated customization to context, and (4) high detection accuracy for sequential fraudulent transactions. The SMDM technique applies sequence analysis to localized histories of sequential typed transactions to discover anomalies as potential fraud. The approach is effective and efficient.

The store-centric divergence method is most appropriate where: (1) transactions can be meaningfully aggregated, such as by merchant, (2) there are many, continuous, typed transactions, (3) there are anonymous or short-lived financial instruments, and (3) the relatively short delay of small, window-based processing is acceptable. These attributes are common to the prepaid card context. However, one can imagine other contexts, such as auctions or stock exchanges. In such cases, the information contained within transaction sequences, and their histories, provides for an effective and efficient means to discover potential fraud.

Based on our experience, we would advocate the use of the approach in cases where individual transactions appear valid, but their sequence may be fraudulent. The method is simple, efficient, and effective. As such, it should be considered another tool in the kit to combat fraud.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. IIS 1217552.

References

Bahnsen, A. C., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications, 51*, 134–142.

Baum, L. E., & Eagon, J. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society, 73*(3), 360–363.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics, 41*(1), 164–171.

Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, 235–249.

Brause, R., Langsdorf, T., & Hepp, M. (1999). Neural data mining for credit card fraud detection. *Paper presented at the tools with artificial intelligence, 1999. proceedings. 11th IEEE international conference on*.

Chudova, D., & Smyth, P. (2002). *Pattern discovery in sequences under a Markov assumption*.

Duman, E., & Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications, 38*(10), 13057–13063.

Forney, G. D., Jr. (1973). The viterbi algorithm. *Proceedings of the IEEE, 61*(3), 268–278.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 44.

Ganji, V. R., & Mannem, S. N. P. (2012). Credit card fraud detection using anti-k nearest neighbor algorithm. *International Journal on Computer Science and Engineering, 4*(6), 1035.

Hand, D. J., & Blunt, G. (2001). Prospecting for gems in credit card data. *IMA Journal of management Mathematics, 12*(2), 173–200.

Hoang, X. D., Hu, J., & Bertok, P. (2003). A multi-layer model for anomaly intrusion detection using program sequences of system calls. *Paper presented at the proc. 11th IEEE int'l. conf.*

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

Joshi, S. S., & Phoha, V. V. (2005). Investigating hidden Markov models capabilities in anomaly detection. *Paper presented at the proceedings of the 43rd annual southeast regional conference-volume 1*.

Kim, H.-C., Pang, S., Je, H.-M., Kim, D., & Bang, S. Y. (2003). Constructing support vector machine ensemble. *Pattern recognition, 36*(12), 2757–2767.

Kou, Y., Lu, C.-T., Sirwongwattana, S., & Huang, Y.-P. (2004). Survey of fraud detection techniques. *Paper presented at the networking, sensing and control, 2004 IEEE international conference on*.

Krivko, M. (2010). A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications, 37*(8), 6070–6076.

Kullback, S. (1997). *Information theory and statistics*. Dover Pubns.

Louzada, F., & Ara, A. (2012). Bagging k-dependence probabilistic networks: An alternative powerful fraud detection tool. *Expert Systems with Applications, 39*(14), 11583–11592.

Lundin, E., Kvarnström, H., & Jonsson, E. (2002). A synthetic fraud data generation methodology. *Paper presented at the international conference on information and communications security*.

Natella, R., Cotroneo, D., Duraes, J. A., & Madeira, H. S. (2013). On fault representativeness of software fault injection. *IEEE Transactions on Software Engineering, 39*(1), 80–96.

Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications, 38*(12), 15273–15285.

Ourston, D., Matzner, S., Stump, W., & Hopkins, B. (2003). Applications of hidden markov models to detecting multi-stage network attacks. *Paper presented at the system sciences, 2003. proceedings of the 36th annual Hawaii international conference on*.

Panigrahi, S., Kundu, A., Sural, S., & Majumdar, A. K. (2009). Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning. *Information Fusion, 10*(4), 354–363.

Phua, C., Smith-Miles, K., Lee, V., & Gayler, R. (2007). *Adaptive spike detection for resilient data stream mining*.

Quah, J. T., & Sriganesh, M. (2008). Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications, 35*(4), 1721–1732.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257–286.

Raj, S. B. E., & Portia, A. A. (2011). Analysis on credit card fraud detection methods. *Paper presented at the computer, communication and electrical technology (ICCCET), 2011 international conference on*.

Sahraeian, S. M. E., & Yoon, B.-J. (2011). A novel low-complexity HMM similarity measure. *IEEE Signal Processing Letters, 18*(2), 87–90.

Sánchez, D., Vila, M., Cerda, L., & Serrano, J.-M. (2009). Association rules applied to credit card fraud detection. *Expert Systems with Applications, 36*(2), 3630–3640.

Singh, S. P., Shukla, S. S. P., Rakesh, N., & Tyagi, V. (2011). Problem reduction in online payment system using hybrid model. arXiv preprint arXiv:1109.0689.

Srivastava, A., Kundu, A., Sural, S., & Majumdar, A. K. (2008). Credit card fraud detection using hidden Markov model. *Dependable and Secure Computing, IEEE Transactions on, 5*(1), 37–48.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. *Paper presented at the DARPA information survivability conference and exposition, 2000. DISCEX'00. proceedings*.

Timmis, J., Neal, D., & King, C. (1999). The development of an artificial immune system for real world applications. *Applications of Artificial Immune Systems, D. Dasgupta, 1*, 157–186.

Xiong, T., Wang, S., Mayers, A., & Monga, E. (2013). Personal bankruptcy prediction by mining credit card data. *Expert Systems with Applications, 40*(2), 665–676.

Yeh, I.-C., & Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications, 36*(2), 2473–2480.

Zareapoor, M., Seeja, K., & Alam, M. A. (2012). Analysis on credit card fraud detection techniques: Based on certain design criteria. *International Journal of Computer Applications, 52*(3).

William N. Robinson Received a computer science B.S. from Oregon State University, and computer science M.S. and Ph.D. from the University of Oregon. He was visiting professor at Oregon State University and currently is associate professor at Georgia State University. He was program chair (1999) and general chair (2009) of IEEE Requirement Engineering Conference. Memberships includes IFIP 2.9, IEEE, and AIS.

Andrea Aria Received a B.S. in Civil Engineering, Sharif University of Technology; M.S. in Earthquake Engineering, University of Tehran; M.S. in Computer Science, Clark Atlanta University; and computer science Ph.D. from Georgia State University, where he is a clinical professor.