Credit Card Fraud Detection Using Hidden Markov Model

SIVA PARVATHI.NELLURI[1], SHAIK. NAGUL[2], Dr.M.KISHOREKUMAR[3]
1. MTech IIYear student, 2. Guide, M.Tech, Assistant professor, 3.H.O.D of NIST College, M.Tech, PhD.

Abstract—Due to a rapid advancement in the electronic commerce technology, the use of credit cards has dramatically increased. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. In this paper, we model the sequence of operations in credit card transaction processing using a Hidden Markov Model (HMM) and show how it can be used for the detection of frauds. An HMM is initially trained with the normal behavior of a cardholder. If an incoming credit card transaction is not accepted by the trained HMM with sufficiently high probability, it is considered to be fraudulent. At the same time, we try to ensure that genuine transactions are not rejected. We present detailed experimental results to show the effectiveness of our approach and compare it with other techniques available in the literature.

Index Terms—Internet, online shopping, credit card, e-commerce security, fraud detection, Hidden Markov Model.

Ç

1 Introduction

The popularity of online shopping is growing day by day. According to an ACNielsen study conducted in 2005, one-tenth of the world's population is shopping online [1]. Germany and Great Britain have the largest number of online shoppers, and credit card is the most popular mode of payment (59 percent). About 350 million transactions per year were reportedly carried out by Barclaycard, the largest credit card company in the United Kingdom, toward the end of the last century [2]. Retailers like Wal-Mart typically handle much larger number of credit card transactions including online and regular purchases. As the number of credit card users rises world-wide, the opportunities for attackers to steal credit card details and, subsequently, commit fraud are also increasing. The total credit card fraud in the United States itself is reported to be $2.7 billion in 2005 and estimated to be $3.0 billion in 2006, out of which $1.6 billion and $1.7 billion, respectively, are the estimates of online fraud [3].

Credit-card-based purchases can be categorized into two types: 1) physical card and 2) virtual card. In a physical-card-based purchase, the cardholder presents his card physically to a merchant for making a payment. To carry out fraudulent transactions in this kind of purchase, an attacker has to steal the credit card. If the cardholder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. In the second kind of purchase, only some important information about a card (card number, expiration date, secure code) is required to make the payment. Such purchases are normally done on the Internet or over the telephone. To commit fraud in these types of purchases, a fraudster simply needs to know the card details. Most of the time, the genuine cardholder is not aware that someone else has seen or stolen his card information. The only way to detect this kind of fraud is to analyze the spending patterns on every card and to figure out any inconsistency with respect to the "usual" spending patterns. Fraud detection based on the analysis of existing purchase data of cardholder is a promising way to reduce the rate of successful credit card frauds. Since humans tend to exhibit specific behavioristic profiles, every cardholder can be represented by a set of patterns containing information about the typical purchase category, the time since the last purchase, the amount of money spent, etc. Deviation from such patterns is a potential threat to the system.

Several techniques for the detection of credit card fraud have been proposed in the last few years. We briefly review some of them in Section 2.

- The authors are with the School of Information Technology, Indian Institute of Technology, Kharagpur, 721302 India.

2 Related Work on Credit Card Fraud Detection

Credit card fraud detection has drawn a lot of research interest and a number of techniques, with special emphasis on data mining and neural networks, have been suggested. Ghosh and Reilly [4] have proposed credit card fraud detection with a neural network. They have built a detection system, which is trained on a large sample of labeled credit card account transactions. These transactions contain example fraud cases due to lost cards, stolen cards, application fraud, counterfeit fraud, mail-order fraud, and nonreceived issue (NRI) fraud. Recently, Syeda et al. [5] have used parallel granular neural networks (PGNNs) for improving the speed of data mining and knowledge discovery process in credit card fraud detection. A complete system has been implemented for this purpose. Stolfo et al. [6] suggest a credit card fraud detection system (FDS) using metalearning techniques to learn models of fraudulent credit card transactions. Metalearning is a general strategy that provides a means for combining and integrating a number of separately built classifiers or models. A metaclassifier is thus trained on the

correlation of the predictions of the base classifiers. The same group has also worked on a cost-based model for fraud and intrusion detection [7]. They use Java agents for Metalearning (JAM), which is a distributed data mining system for credit card fraud detection. A number of important performance metrics like True Positive—False Positive (TP-FP) spread and accuracy have been defined by them.

Aleskerov et al. [8] present CARDWATCH, a database mining system used for credit card fraud detection. The system, based on a neural learning module, provides an interface to a variety of commercial databases. Kim and Kim have identified skewed distribution of data and mix of legitimate and fraudulent transactions as the two main reasons for the complexity of credit card fraud detection [9]. Based on this observation, they use fraud density of real transaction data as a confidence value and generate the weighted fraud score to reduce the number of misdetections.

Fan et al. [10] suggest the application of distributed data mining in credit card fraud detection. Brause et al. [11] have developed an approach that involves advanced data mining techniques and neural network algorithms to obtain high fraud coverage. Chiu and Tsai [12] have proposed Web services and data mining techniques to establish a collaborative scheme for fraud detection in the banking industry. With this scheme, participating banks share knowledge about the fraud patterns in a heterogeneous and distributed environment. To establish a smooth channel of data exchange, Web services techniques such as XML, SOAP, and WSDL are used. Phua et al. [13] have done an extensive survey of existing data-mining-based FDSs and published a comprehensive report. Prodromidis and Stolfo [14] use an agent-based approach with distributed learning for detecting frauds in credit card transactions. It is based on artificial intelligence and combines inductive learning algorithms and metalearning methods for achieving higher accuracy.

Phua et al. [15] suggest the use of metaclassifier similar to [6] in fraud detection problems. They consider naive Bayesian, C4.5, and Back Propagation neural networks as the base classifiers. A metaclassifier is used to determine which classifier should be considered based on skewness of data. Although they do not directly use credit card fraud detection as the target application, their approach is quite generic. Vatsa et al. [16] have recently proposed a game-theoretic approach to credit card fraud detection. They model the interaction between an attacker and an FDS as a multistage game between two players, each trying to maximize his payoff.

The problem with most of the abovementioned approaches is that they require labeled data for both genuine, as well as fraudulent transactions, to train the classifiers. Getting real-world fraud data is one of the biggest problems associated with credit card fraud detection. Also, these approaches cannot detect new kinds of frauds for which labeled data is not available. In contrast, we present a Hidden Markov Model (HMM)-based credit card FDS, which does not require fraud signatures and yet is able to detect frauds by considering a cardholder's spending habit. We model a credit card transaction processing sequence by the stochastic process of an HMM. The details of items purchased in individual transactions are usually not known to an FDS running at the bank that issues credit cards to the cardholders. This can be represented as the underlying finite Markov chain, which is not observable. The transactions can only be observed through the other stochastic process that produces the sequence of the amount of money spent in each transaction. Hence, we feel that HMM is an ideal choice for addressing this problem. Another important advantage of the HMM-based approach is a drastic reduction in the number of False Positives (FPs)—transactions identified as malicious by an FDS although they are actually genuine. Since the number of genuine transactions is a few orders of magnitude higher than the number of malicious transactions, an FDS should be designed in such a way that the number of FPs is as low as possible. Otherwise, due to the "base rate fallacy" effect [17], bank administrators may tend to ignore the alarms. To the best of our knowledge, there is no other published literature on the application of HMM for credit card fraud detection.

The rest of the paper is organized as follows: In Section 3, we first briefly explain the working principle of an HMM. We then show how to model credit card transaction processing using HMM in Section 4. We also describe the complete process flow of the proposed FDS in this section. Detailed experimental result is presented in Section 5. Finally, we conclude the paper with some discussions.

3 HMM Background

An HMM is a double embedded stochastic process with two hierarchy levels. It can be used to model much more complicated stochastic processes as compared to a traditional Markov model. An HMM has a finite set of states governed by a set of transition probabilities. In a particular state, an outcome or observation can be generated according to an associated probability distribution. It is only the outcome and not the state that is visible to an external observer [18].

HMM-based applications are common in various areas such as speech recognition, bioinformatics, and genomics. In recent years, Joshi and Phoba [19] have investigated the capabilities of HMM in anomaly detection. They classify TCP network traffic as an attack or normal using HMM. Cho and Park [20] suggest an HMM-based intrusion detection system that improves the modeling time and performance by considering only the privilege transition flows based on the domain knowledge of attacks. Ourston et al. [21] have proposed the application of HMM in detecting multistage network attacks. Hoang et al. [22] present a new method to process sequences of system calls for anomaly detection using HMM. The key idea is to build a multilayer model of program behaviors based on both HMMs and enumerating methods for anomaly detection. Lane [23] has used HMM to model human behavior. Once human behavior is correctly modeled, any detected deviation is a cause for concern since an attacker is not expected to have a behavior similar to the genuine user. Hence, an alarm is raised in case of any deviation.

An HMM can be characterized by the following [18]:

1. N is the number of states in the model. We denote the set of states $S = \{S_1, S_2, \ldots S_N\}$, where S_i, $i = 1,2,\ldots,N$ is an individual state. The state at time instant t is denoted by q_t.
2. M is the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modeled. We denote the set of symbols $V = \{V_1, V_2, \ldots V_M\}$, where V_i, $i = 1,2,\ldots,M$ is an individual symbol.

3. The state transition probability matrix $A = [a_{ij}]$, where

$$a_{ij} = P(q_{t+1} = S_j / q_t = S_i), 1 \le i \le N, 1 \le j \le N; t = 1,2,\ldots : \quad (1)$$

For the general case where any state j can be reached from any other state i in a single step, we have $a_{ij} > 0$ for all i, j. Also

$$\sum_{j=1}^{N} a_{ij} = 1,\ 1 \le i \le N.$$

4. The observation symbol probability matrix $B = [b_j(k)]$, where

$$b_j(k) = P(V_k \backslash S_j), 1 \le j \le N; 1 \le k \le M \text{ and}$$

$$\sum_{K=1} b_j(k) = 1; 1 \le j \le N \quad (2)$$

5. The initial state probability vector $\pi = [\pi_i]$, where

$$\pi_i = P(q_1 = S_i), 1 \le i \le N; \text{ such that } \sum_{i=1}^{N} \pi_i = 1 \quad (3)$$

6. The observation sequence $O = O_1, O_2, O_3, \ldots O_R$, where each observation O_t is one of the symbols from V, and R is the number of observations in the sequence.

It is evident that a complete specification of an HMM requires the estimation of two model parameters, N and M, and three probability distributions A, B, and π. We use the notation $\lambda = (A, B, \pi)$ to indicate the complete set of parameters of the model, where A, B implicitly include N and M.

An observation sequence O, as mentioned above, can be generated by many possible state sequences. Consider one such particular sequence

$$Q = q_1, q_2, \ldots q_R \quad \textbf{(4)}$$

where q_1 is the initial state.

The probability that O is generated from this state sequence is given by

$$P(O/Q, \lambda) = \prod_{t=1}^{R} p\left(\frac{o_t}{q_t}, \lambda\right) \quad (5)$$

where statistical independence of observations is assumed. Equation (5) can be expanded as

$$P(O/Q,\lambda) = b_{q1}(o_1), b_{q2}(o_2) \text{------} b_{qR}(o_R). \quad (6)$$

The probability of the state sequence Q is given as

$$P(Q/\lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \ldots a_{qR-1qR} \quad (7)$$

Thus, the probability of generation of the observation sequence O by the HMM specified by can be written as follows:

$$P(O/\lambda) = \sum_{all\ Q} p(O \backslash Q\lambda), p(Q \backslash \lambda) \quad (8)$$

Deriving the value of $P(O\backslash\lambda)$ using the direct definition of (8) iscomputationally intensive. Hence, a procedure named as Forward-Backwardprocedure[18] isusedtocompute $P(O\backslash\lambda)$.

TABLE 1
Notations and Acronyms

Notation	Meaning
M	Number of observation symbols
N	Number of hidden states
V_k k=1..M	Observation symbols
l, m, h	Price ranges – low, medium and high
$a_{x\text{-}y}$	Probability of transition from the Hidden Markov Model state x to state y
K	Number of clusters
c_i	Centroid of cluster i
R	Sequence length
α	Probability of acceptance of a sequence by Hidden Markov Model
Acronym	**Expanded Form**
FDS	Fraud Detection System
HMM	Hidden Markov Model
SP	Spending Profile
hs, ms, ls	High spending, Medium spending, Low spending group
TP, FP	True Positive, False Positive

4 Use of HMM for Credit Card Fraud Detection

An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification. FDS receives the card details and the value of purchase to verify whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find any anomaly in the transaction based on the spending profile of the cardholder, shipping address, and billing address, etc. If the FDS confirms the transaction to be malicious, it raises an alarm, and the issuing bank declines the transaction. The concerned cardholder may then be contacted and alerted about the possibility that the card is compromised. In this section, we explain how HMM can be used for credit card fraud detection.

A set of notations and acronyms used in the paper is given in Table 1.

4.1 HMM Model for Credit Card Transaction Processing

To map the credit card transaction processing operation in terms of an HMM, we start by first deciding the observation symbols in our model. We quantize the purchase values x into M price ranges $V_1; V_2; \ldots V_M$, forming the observation symbols at the issuing bank. The actual price range for each symbol is configurable based on the spending habit of individual cardholders. These price ranges can be determined dynamically by applying a clustering algorithm on the values of each cardholder's transactions, as shown in Section 5.2. We use V_k, k ¼ 1; 2; ... M, to represent both the observation symbol, as well as the corresponding price range.

In this work, we consider only three price ranges, namely, Low(l),medium(m),and high(h). our set of observation symbols is, therefore, V={l,,m,,h} making M=3. For example, let l = (0, $100), m = ($100, $500), and h =($500; credit card limit] . If a cardholder performs a transaction of $190, then the corresponding observation symbol is m.

A credit cardholder makes different kinds of purchases of different amounts over a period of time. One possibility is to

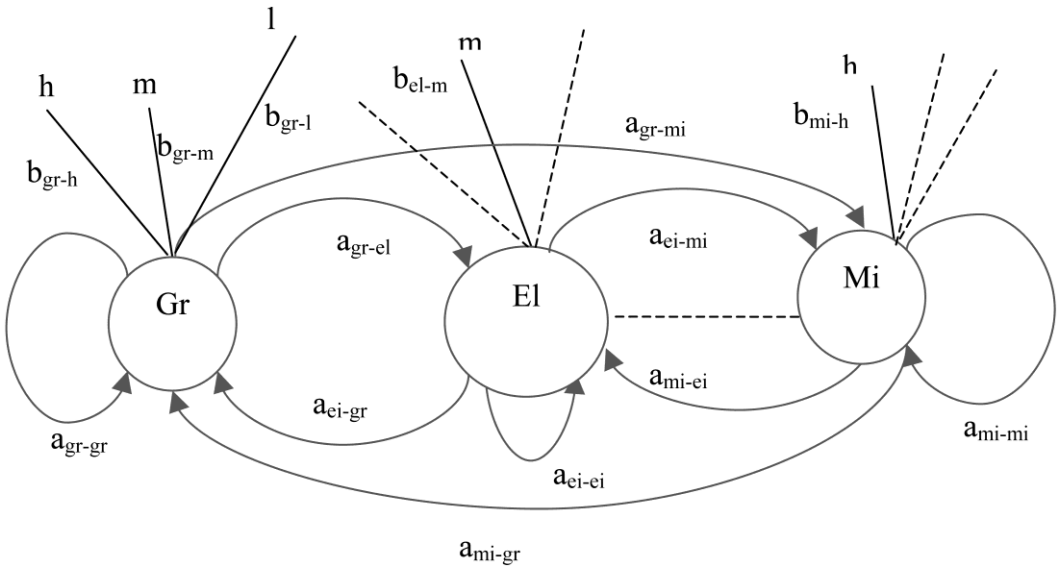


Fig. 1. HMM for credit card fraud detection.

consider the sequence of transaction amounts and look for deviations in them. However, the sequence of types of purchase is more stable compared to the sequence of transaction amounts. The reason is that, a cardholder makes purchases depending on his need for procuring different types of items over a period of time. This, in turn, generates a sequence of transaction amounts. Each individual transaction amount usually depends on the corresponding type of purchase. Hence, we consider the transition in the type of purchase as state transition in our model. The type of each purchase is linked to the line of business of the corresponding merchant. This information about the merchant's line of business is not known to the issuing bank running the FDS. Thus, the type of purchase of the cardholder is hidden from the FDS. The set of all possible types of purchase and, equivalently, the set of all possible lines of business of merchants forms the set of hidden states of the HMM. It should be noted at this stage that the line of business of the merchant is known to the acquiring bank, since this information is furnished at the time of registration of a merchant. Also, some merchants may be dealing in various types of commodities (For example, Wal-Mart, K-Mart, or Target sells tens of thousands of different items). Such types of line of business are considered as Miscellaneous, and we do not attempt to determine the actual types of items purchased in these transactions. Any assumption about availability of this information with the issuing bank and, hence, with the FDS, is not practical and, therefore, would not have been valid. In Section 5, we show the effect of the choice of the number of states on the system performance.

After deciding the state and symbol representations, the next step is to determine the probability matrices A, B, and so that representation of the HMM is complete. These three model parameters are determined in a training phase using the Baum-Welch algorithm [18]. The initial choice of parameters affects the performance of this algorithm and, hence, they should be chosen carefully.

We consider the special case of fully connected HMM in which every state of the model can be reached in a single step from every other state, as shown in Fig. 1. Gr, El, Mi, etc., are names given to the states to denote purchase types like Groceries, Electronic items, and Miscellaneous purchases. Spending profiles of the individual cardholders are used to obtain an initial estimate for probability matrix B of (2). In Section 4.2, we explain how to determine observation symbols dynamically from a cardholder's transactions.

4.2 Dynamic Generation of Observation Symbols

For each cardholder, we train and maintain an HMM. To find the observation symbols corresponding to individual cardholder's transactions dynamically, we run a clustering algorithm on his past transactions. Normally, the transactions that are stored in the issuing bank's database contain many attributes. For our work, we consider only the amount that the cardholder spent in his transactions. Although various clustering techniques could be used, we use K-means clustering algorithm [24] to determine the clusters. K-means is an unsupervised learning algorithm for grouping a given set of data based on the similarity in their attribute (often called feature) values. Each group formed in the process is called a cluster. The number of clusters K is fixed a priori. The grouping is performed by minimizing the sum of squares of distances between each data point and the centroid of the cluster to which it belongs.

In our work, K is the same as the number of observation symbols M. Let $c_1, c_2, \ldots c_M$ be the centroids of the generated clusters. These centroids or mean values are used to decide the observation symbols when a new transaction comes in. Let x be the amount spent by the cardholder u in transaction T. FDS generates the observation symbol for x (denoted by O_x) as follows:

$$O_x = V_{\arg\min_i |x - c_i|} \quad (9)$$

As mentioned before, the number of symbols is 3 in our system. Considering M=3, if we execute K-means algorithm on the example transactions in Table 2, we get the clusters, as shown in Table 3, with c_l, c_m, and c_h as the respective centroids. It may be noted that the dollar amounts 5, 10, and 10 have been clustered together as c_l resulting in a centroid of 8.3. The percentage (p) of total number of transactions in this cluster is thus 30 percent. Similarly,

TABLE 2
Example Transactions with the Dollar Amount Spent in Each Transaction

Transaction no.	1	2	3	4	5	6	7	8	9	10
Dollar Amount	40	25	15	5	10	25	15	20	10	80

TABLE 3
Output of K-Means Clustering Algorithm

Cluster mean/centroid name	c_l	c_m	c_h
Observation symbol	$V_1 = l$	$V_2 = m$	$V_3 = h$
Mean value (Centroid)	8.3	20	60
Percentage of total transactions (p)	30	50	20

dollar amounts 15, 15, 20, 25, and 25 have been grouped in the cluster c_m with centroid 20, whereas amounts 40 and 80 have been grouped together in cluster c_h. c_m and c_h, thus, contain 50 percent and 20 percent of the total number of transactions. When the FDS receives a transaction T for this cardholder, it measures the distance of the purchase amount x with respect to the means c_l, c_m, and c_h to decide (using (9)) the cluster to which T belongs and, hence, the corresponding observation symbol. As an example, if x = \$10, then in Table 3 using (9), the observation symbol is V_1 = l.

4.3 Spending Profile of Cardholders

The spending profile of a cardholder suggests his normal spending behavior. Cardholders can be broadly categorized into three groups based on their spending habits, namely, high-spending (hs) group, medium-spending (ms) group, and low-spending (ls) group. Cardholders who belong to the hs group, normally use their credit cards for buying high-priced items. Similar definition applies to the other two categories also.

Spending profiles of cardholders are determined at the end of the clustering step. Let p_i be the percentage of total number of transactions of the cardholder that belong to cluster with mean c_i. Then, the spending profile (SP) of the cardholder u is determined as follows:

$$SP(u) = \arg \max_i (p_i)\text{:} \quad (10)$$

Thus, spending profile denotes the cluster number to which most of the transactions of the cardholder belong. In the example in Table 3, the spending profile of the cardholder is 2, that is, m and, hence, the cardholder belongs to the ms group.

4.4 Model Parameter Estimation and Training

We use Baum-Welch algorithm to estimate the HMM parameters for each cardholder. The algorithm starts with an initial estimate of HMM parameters A, B, and and converges to the nearest local maximum of the likelihood function. Initial state probability distribution is considered to be uniform, that is, if there are N states, then the initial probability of each state is 1/N. Initial guess of transition and observation probability distributions can also be considered to be uniform. However, to make the initial guess of observation symbol probabilities more accurate, spending profile of the cardholder, as determined in Section 4.3, is taken into account. We make three sets of initial probability for observation symbol generation for three spending groups—ls, ms, and hs. Based on the cardholder's spending profile, we choose the corresponding set of initial observation probabilities. The initial estimate of symbol generation probabilities using this method leads to accurate learning of the model. Since there is no a priori knowledge about the state transition probabilities, we consider the initial guesses to be uniform. In case of a collaborative work between an acquiring bank and an issuing bank, we can have better initial guess about state transition probabilities as well.

We now start training the HMM. The training algorithm has the following steps: 1) initialization of HMM parameters, 2) forward procedure, and 3) backward procedure. Details of these steps can be found in [18]. For training the HMM, we convert the cardholder's transaction amount into observation symbols and form sequences out of them. At the end of the training phase, we get an HMM corresponding to each cardholder. Since this step is done offline, it does not affect the credit card transaction processing performance, which needs online response.

4.5 Fraud Detection

After the HMM parameters are learned, we take the symbols from a cardholder's training data and form an initial sequence of symbols. Let $O_1, O_2, \ldots O_R$ be one such sequence of length R. This recorded sequence is formed from the cardholder's transactions up to time t. We input this sequence to the HMM and compute the probability of acceptance by the HMM. Let the probability be α_1, which can be written as follows:

$$\alpha 1 = P(O_1, O_2, O_3, \ldots O_R \backslash \lambda) \quad (11)$$

Let O_{R+1} be the symbol generated by a new transaction at time t+1. To form another sequence of length R, we drop O_1 and append O_{R+1} in that sequence, generating $O_2, O_3, \ldots O_R, O_{R+1}$ as the new sequence. We input this new sequence to the HMM and calculate the probability of acceptance by the HMM. Let the new probability be α_2

$$\alpha_2 = P(O_2, O_3, O_4, \ldots O_{R+1} \backslash \lambda) \quad (12)$$

$$\text{Let } \Delta\alpha = \alpha 1 - \alpha 2 \quad (13)$$

If $\Delta\alpha > 0$, it means that the new sequence is accepted by the HMM with low probability, and it could be a fraud. The newly added transaction is determined to be fraudulent if the percentage change in the probability is above a threshold, that is,

$$\Delta\alpha/\alpha 1 \geq \text{Threshold:} \quad (14)$$

The threshold value can be learned empirically, as will be discussed in Section 5. If O_{R+1} is malicious, the issuing bank does not approve the transaction, and the FDS discards the symbol. Otherwise, O_{R+1} is added in the sequence permanently, and the new sequence is used as the base sequence for determining the validity of the next transaction. The reason for including new nonmalicious symbols in the sequence is to capture the changing spending behavior of a cardholder. Fig. 2 shows the complete process flow of the proposed FDS. As shown in the figure, the FDS is divided into two parts—one is the training module, and the other is detection.

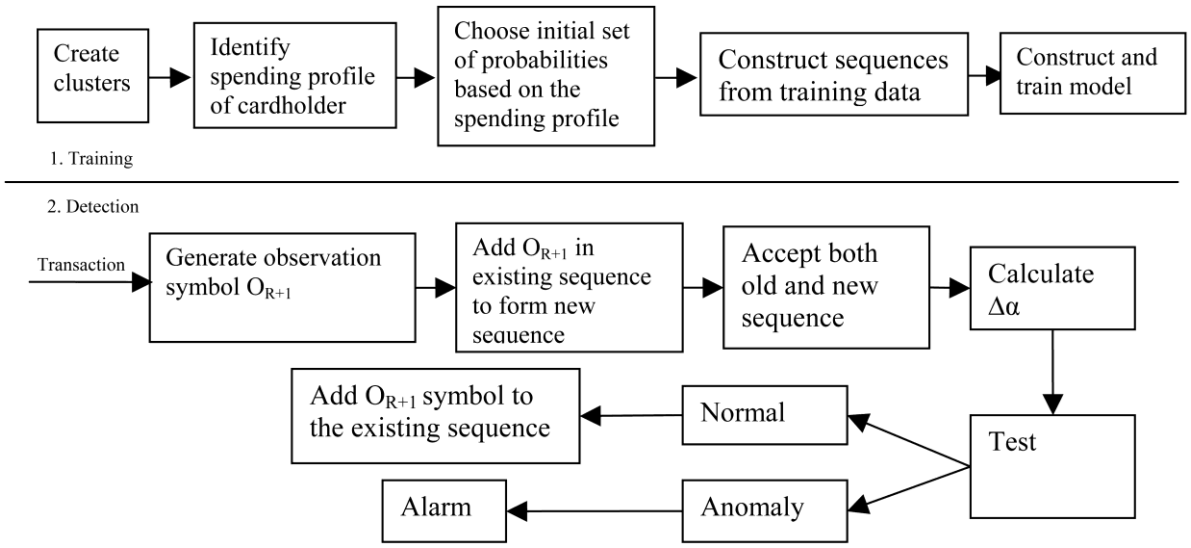


Fig. 2. Process flow of the proposed FDS.

Training phase is performed offline, whereas detection is an online process.

5 Results

Testing credit card FDSs using real data set is a difficult task. Banks do not, in general, agree to share their data with researchers. There is also no benchmark data set available for experimentation. We have, therefore, performed large-scale simulation studies to test the efficacy of the system. A simulator is used to generate a mix of genuine and fraudulent transactions. The number of fraudulent transactions in a given length of mixed transactions is normally distributed with a user specified μ(mean) and σ (standard deviation), taking cardholder's spending behavior into account.μ specifies the average number of fraudulent transactions in a given transaction mix. In a typical scenario, an issuing bank, and hence, its FDS receives a large number of genuine transactions sparingly intermixed with fraudulent transactions. The genuine transactions are generated according to the cardholders' profiles. The cardholders are classified into three categories as mentioned before—the low, medium, and hs groups. We have studied the effects of spending group and the percentage of transactions that belong to the low-, medium-, and high-price-range clusters. We use standard metrics—True Positive (TP) and FP, as well as TP-FP spread and Accuracy metrics, as proposed in [7] to measure the effectiveness of the system. TP represents the fraction of fraudulent transactions correctly identified as fraudulent, whereas FP is the fraction of genuine transactions identified as fraudulent. Most of the design choices for a FDS that result in higher values of TP, also cause FP to increase. To meaningfully capture the performance of such a system, the difference between TP and FP, often called the TP-FP spread, is used as a metric. Accuracy represents the fraction of total number of transactions (both genuine and fraudulent) that have been detected correctly. It can be expressed as follows:

$$\text{Accuracy} = \frac{\text{No.of good trans.detected as good} + \text{No.of bad transected as bad}}{\text{total no of transactions}} \quad (15)$$

We first carried out a set of experiments to determine the correct combination of HMM design parameters, namely, the number of states, the sequence length, and the threshold value. Once these parameters were decided, we performed comparative study with another FDS.

5.1 Choice of Design Parameters

Since there are three parameters in an HMM, we need to vary one at a time keeping the other two fixed, thus generating a large number of possible combinations. For choosing the design parameters, we generate transaction sequences using 95 percent low value, 3 percent medium value, and 2 percent high value transactions. The reason for using this mix is that it represents a profile that strongly resembles a ls customer profile. We also consider the and μ and σ values to be 1.0 and 0.5, respectively. This is chosen so that, on the average, there will be 1 fraudulent transaction in any incoming sequence with some scope for variation. After the parameter values are fixed, we will see in Section 5.2, how the system performs as we vary the profile and the mix of fraudulent transactions.

For parameter selection, the sequence length is varied from 5 to 25 in steps of 5. The threshold values considered are 30 percent, 50 percent, 70 percent, and 90 percent. The number of states is varied from 5 to 10 in steps of 1. We consider both TP and FP for deciding the optimum parameter values. Thus, there are a total of 120 (5 × 4× 6) possible combinations of parameters. The number of simulation runs required for obtaining results with a given confidence interval (CI) was derived as follows [25], [26].

An initial set of five simulation runs, each with 100 samples, was carried out to estimate the mean and standard deviation of both TP and FP for a fixed sequence length, number of states, and threshold value. Mean TP was found to be an order of magnitude higher than mean FP. Standard deviation of TP was 0.1 and that for FP was 0.005. We set the target 95 percent CI for TP and FP, respectively, as þ= 2.5 percent and þ= 0.25 percent around their mean values. Using Student's t-distribution, the minimum number of simulation runs required for obtaining desired CI for TP

TABLE 4
Variation of TP and FP with Different Sequence Lengths

Threshold (%)	TP averaged over all the 6 states for different sequence lengths					FP averaged over all the 6 states for different sequence lengths				
	5	10	15	20	25	5	10	15	20	25
30	0.52	0.56	**0.64**	0.58	0.6	0.05	0.05	**0.05**	0.05	0.05
50	0.54	0.54	**0.63**	0.57	0.6	**0.03**	0.05	0.04	0.05	0.05
70	0.50	0.60	0.60	**0.61**	0.59	**0.04**	**0.04**	0.05	0.05	0.05
90	0.42	0.52	**0.59**	0.58	0.57	**0.02**	0.04	0.05	00.05	0.05

TABLE 5
Variation of TP and FP with Different Number of States

Threshold (%)	TP averaged over all the 5 sequence lengths for different no. of states						FP averaged over all the 5 sequence lengths for different no. of states					
	5	6	7	8	9	10	5	6	7	8	9	10
30	0.56	0.59	0.55	0.60	**0.61**	0.55	0.06	**0.04**	0.05	0.05	**0.04**	**0.04**
50	0.57	**0.60**	0.52	0.56	0.59	0.59	0.05	**0.04**	0.05	0.05	0.05	**0.04**
70	0.57	0.59	0.56	0.58	0.56	**0.62**	**0.04**	0.05	**0.04**	0.05	**0.04**	0.05
90	0.56	0.51	**0.60**	0.53	0.49	0.55	**0.03**	0.04	**0.03**	0.04	0.04	0.04

was derived as 83 and that for FP as 23. Based on these observations, we set the number of simulation runs for all the experiments to be 100. The results obtained were within the desired CI, as mentioned above.

Since it is not convenient to present the detailed results for each of the 120 combinations, we show summarized results. In Table 4, we show the results for each value of sequence length averaged over all the six states. Similarly, we present results for each value of the number of states averaged over all the five sequence lengths in Table 5.

In Tables 4 and 5, the highest value of TP, as well as the lowest value of FP, has been highlighted for each row. It is seen from the two tables that FP shows a clear trend of decreasing with higher threshold and smaller sequence lengths. However, the number of states does not have a strong influence either on TP or on FP. In Table 4, it is seen that TP is high for sequence length 15 in 75 percent of the cases. Also, fraud detection time increases linearly with the sequence length, as shown in Fig. 3. The results have been plotted for a Java implementation on a 1.8 GHz Pentium IV processor machine. Hence, we choose 15 as the length of observation symbol sequence for optimum performance. Once sequence length is decided, it is seen in Table 4 that the threshold could be set to either 30 percent or 50 percent. Although TP is higher for threshold =30%, FP is also higher. To minimize FP, we choose threshold = 50%. After choosing sequence length and threshold, we have to choose the number of states. Since there is no clear indication from the above summary information, as presented in Tables 4 and 5, we take a look at the detailed data for TP when threshold ¼ 50%, as shown in Table 6.

It is seen that for sequence length ¼ 15, the highest value of TP occurs for no: of states ¼ 10 and, hence, it would be a good choice for our design.

We have also analyzed the time taken by the training phase, which is performed offline for each cardholder's HMM. Fig. 4 shows the plot of model learning time against the number of sequences in the training data. As the size of training data increases, learning time increases, especially beyond 100. We therefore, use 100 sequences for training the HMM. Although done offline, the model learning time has a strong impact on the scalability of the system. Since an HMM

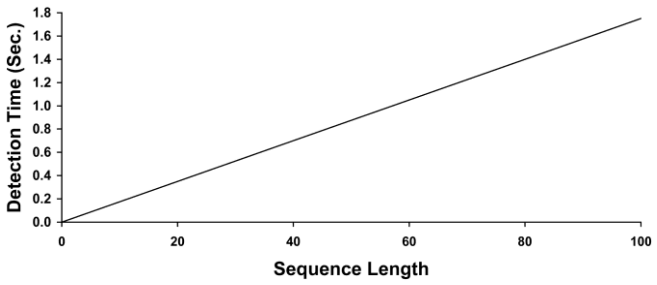


Fig. 3. Detection time versus length of sequence.

TABLE 6
Detailed Result of TP for threshold ¼ 50%

No. of States	Sequence Length				
	5	10	15	20	25
5	0.53	0.54	0.53	0.67	0.59
6	0.50	0.55	0.67	0.43	0.56
7	0.54	0.42	0.67	0.43	0.56
8	0.58	0.46	0.54	0.60	0.63
9	0.50	0.61	0.60	0.63	0.63
10	0.57	0.65	**0.75**	0.49	0.5

is trained for each cardholder, it is imperative that the training time is kept as low as possible especially when an issuing bank is meant to handle millions of cardholders with many new cards being issued everyday. The online processing time of about 200 ms on a 1.8 GHz Pentium IV machine also shows that the system will be able to handle a large number of concurrent operations and, hence, is scalable.

Thus, our design parameter setting is given as follows:

1. number of hidden states N =10,
2. length of observation sequence R = 15,
3. Threshold value =50%, and
4. number of sequences for training = 100.

With this design parameter setting, we next proceed to study the performance of the system under various combinations of input data.

5.2 Comparative Performance

In this section, we show performance of the proposed system as we vary the number of fraudulent transactions and also the spending profile of the cardholder. Our design parameter setting is as obtained in the previous section. We compare performance of our approach (denoted by OA below) with the credit card fraud detection technique proposed by Stolfo et al. [6] (denoted by ST below). For comparison, we consider the metrics TP and FP, as well as TP-FP and Accuracy [7].

We carried out experiments by varying both the transaction amount mix, as well as the number of fraudulent transactions intermixed with a sequence of genuine transactions. Transaction amount mix is captured by the cardholder's profile. We consider four profiles. One of them is the mixed profile, which means that spending profile is not considered at all by our approach, as explained in Section 4.3. The other profiles considered are (55 35 10), (70 20 10), and (95 3 2). Here, ða b cÞ profile represents a ls profile cardholder who has been found to carry out a percent of his transactions in the low, b percent in medium, and c percent in the high range. Thus, our attempt is to see how the system performs in the presence of different mixes of transaction amount ranges in the transactions. It may be noted that for cardholders in the other two groups, namely, hs and ms, will show similar

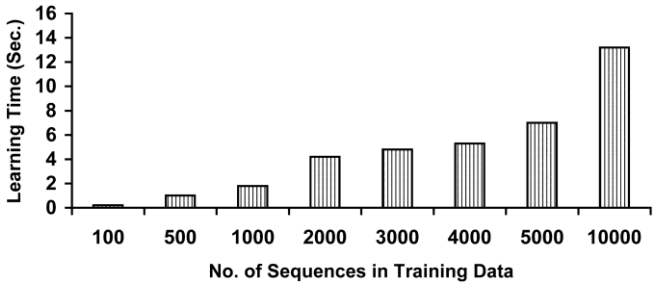


Fig. 4. Model learning time versus number of sequences in training data.

performance as only the relative ordering of a, b, and c will change. We also vary the mean value of malicious transactions from 0.5 to 4.0 in steps of 0.5. The value is kept fixed at 0.5 for all the experiments. Thus, every sequence of transaction that we use for testing is a mixed sequence containing both genuine, as well as malicious, transactions. For each combination of spending profile and malicious transaction distribution, we carried out 100 runs and report the average result. The same set of data was used to determine the performance of both OA and ST.

Fig. 5a shows the variation of TP and FP for the two approaches using the spending profile (95 3 2). Variation of TP-FP and Accuracy is shown in Fig. 5b.

It is seen from the figures that TP of the proposed approach is very close to that of Stolfo et al's approach. Also, both the approaches have almost similar values of FP. As a result, the two systems have comparable accuracies and average TP-FP spread. Further, the two approaches exhibit similar trend with variation in . Next, we show how the two systems behave as we mix the transaction amounts. Percentages of low value, medium value, and high value transactions are changed from (95 3 2), as shown above to (70 20 10) in Figs. 6a and 6b and (55 35 10) in Figs. 7a and 7b.

There are a number of interesting observations from the above two sets of figures. The first observation is that, TP falls and FP rises for both the approaches, as transactions no longer remain strictly ls in nature. However, the FP rate of ST rises sharply, although the TP rate does not degrade to a great extent. On the other hand, for our approach, FP rate remains low while there is a graceful degradation of the TP rate. As a result, the Accuracy of the proposed system remains close to 80 percent for all the above settings. Accuracy by Stolfo et al's approach falls drastically to about 60 percent for (55 35 10). Thus, our approach has 15-20 percent better Accuracy. Although TP-FP values are close for the profile (70 20 10), there is more than 15 percent difference in TP-FP for the profile (55 35 10) with our approach performing better.

In Figs. 8a and 8b, we show the performance of the two approaches when the profile is mixed, which means that all the three ranges of transactions are equiprobable. It is seen that the FP of the ST method has gone up sharply. In fact, FP is even higher than TP. As a result, the mean accuracy has dropped below 40 percent. Our approach shows a fall in TP, but the FP has not degraded a lot. As a result, the Accuracy of our system still remains around 80 percent. The TP-FP value for the ST approach has become negative even for ¼ 0:5. For our approach, the TP-FP value became negative only after ¼ 2:5.

From the above results, it can be concluded that the proposed system has an overall Accuracy of 80 percent even under large input condition variations, which is much higher

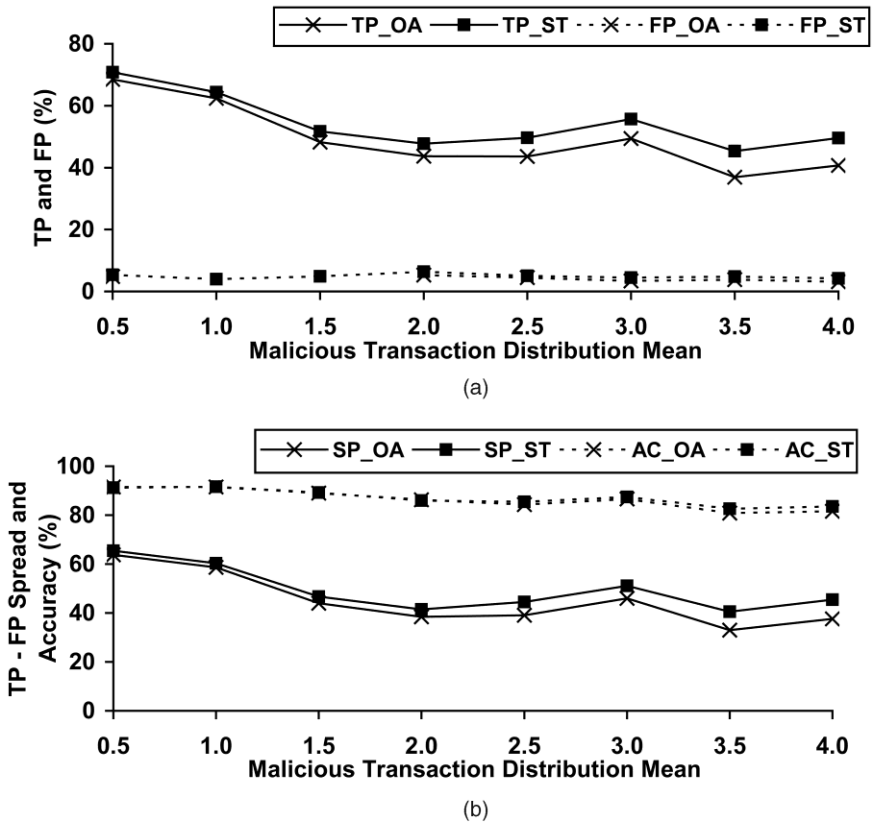


Fig. 5. Performance variation of the two systems (OA and ST) with the mean ð Þ of malicious transaction distribution for the spending profile (95 3 2). (a) TP and FP. (b) TP-FP spread (SP) and Accuracy (AC).

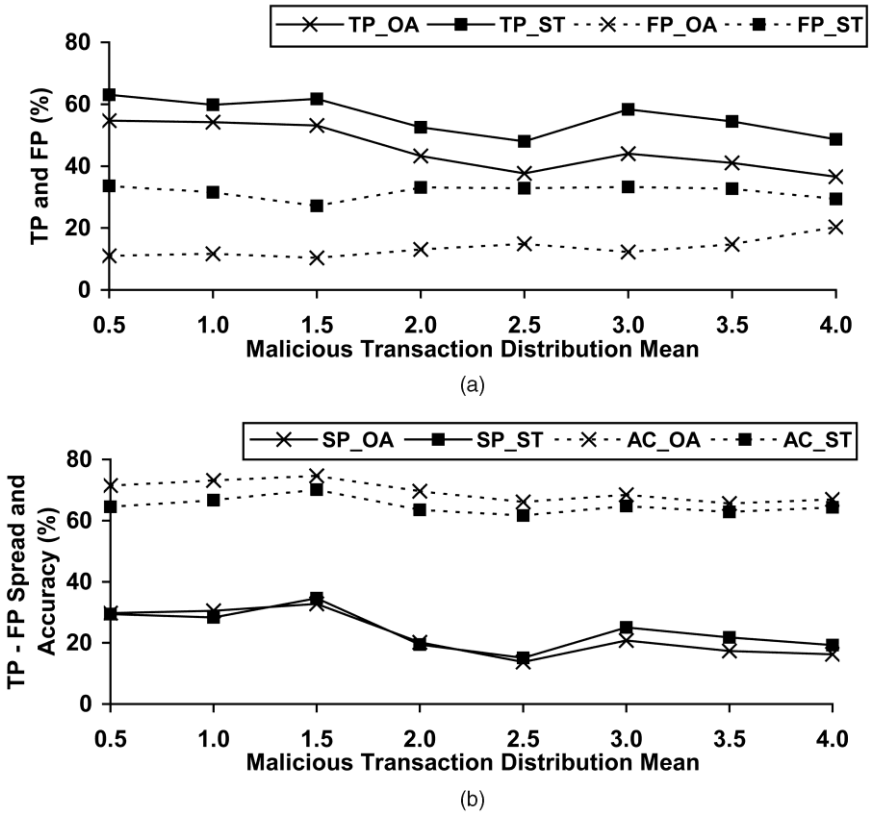
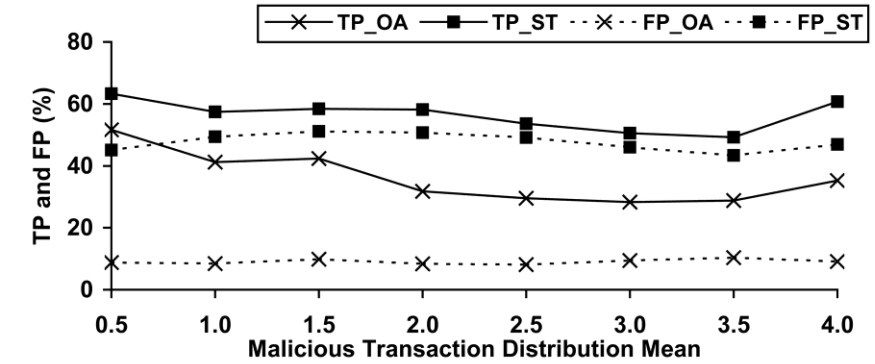
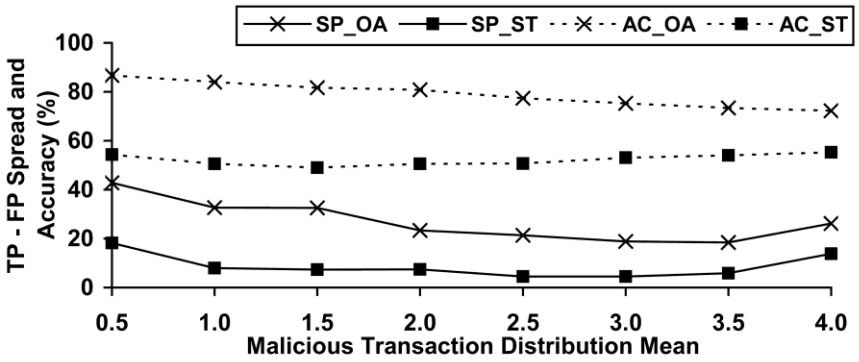


Fig. 6. Performance variation of the two systems (OA and ST) within the mean ð Þ of malicious transaction distribution for the spending profile (70 20 10). (a) TP and FP. (b) TP-FP spread (SP) and Accuracy (AC).

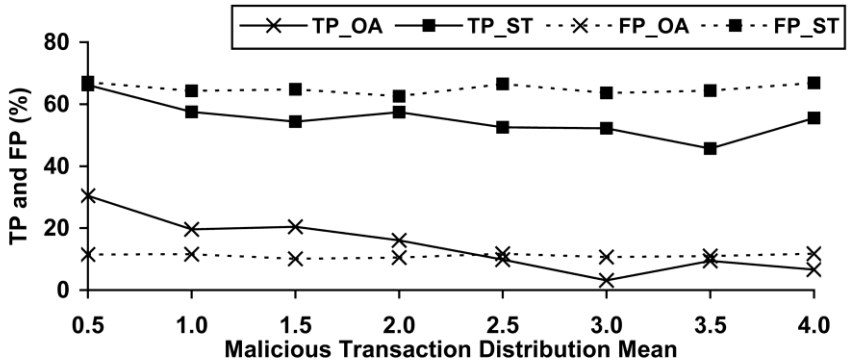


(a)

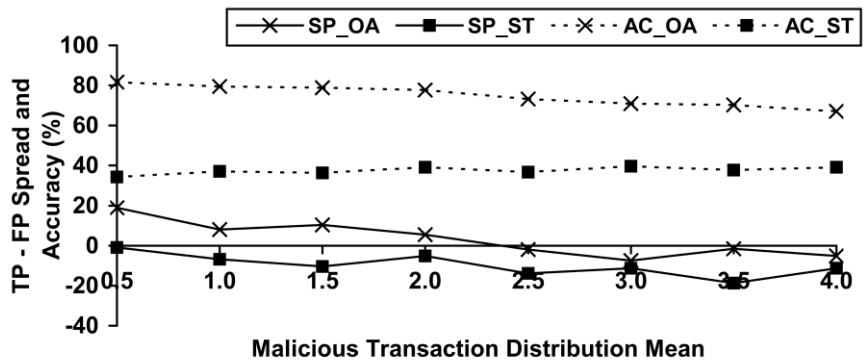


(b)

Fig. 7. Performance variation of the two systems (OA and ST) with the mean ð Þ of malicious transaction distribution for the spending profile (55 35 10). (a) TP and FP. (b) TP-FP spread (SP) and Accuracy (AC).



(a)



(b)

Fig. 8. Performance variation of the two systems (OA and ST) with mean ð Þ of malicious transaction distribution for mixed profile. (a) TP and FP. (b) TP-FP spread (SP) and Accuracy (AC).

than the overall Accuracy of the method proposed by Stolfo et al. [6]. Our system can, therefore, correctly detect most of the transactions. However, when there is no profile information at all, the system shows some performance degradation in terms of TP-FP. This observation highlights the importance of profile selection as explained in Section 4. Also, when there is little difference between genuine transactions and malicious transactions, most of the credit card FDSs suffer performance degradation, either due to a fall in the number of TPs or a rise in the number of FPs.

6 Conclusions and Discussions

In this paper, we have proposed an application of HMM in credit card fraud detection. The different steps in credit card transaction processing are represented as the underlying stochastic process of an HMM. We have used the ranges of transaction amount as the observation symbols, whereas the types of item have been considered to be states of the HMM. We have suggested a method for finding the spending profile of cardholders, as well as application of this knowledge in deciding the value of observation symbols and initial estimate of the model parameters. It has also been explained how the HMM can detect whether an incoming transaction is fraudulent or not. Experimental results show the performance and effectiveness of our system and demonstrate the usefulness of learning the spending profile of the cardholders. Comparative studies reveal that the Accuracy of the system is close to 80 percent over a wide variation in the input data. The system is also scalable for handling large volumes of transactions.

Acknowledgments

The authors would like to the anonymous reviewers for their constructive and useful comments. This work is partially supported by a research grant from the Department of Information Technology, Ministry of Communication and Information Technology, Government of India, under Grant 12(34)/04-IRSD dated 07/12/2004.

References

[1] "Global Consumer Attitude Towards On-Line Shopping," http://www2.acnielsen.com/reports/documents/2005_cc_onlineshopping.pdf, Mar. 2007.
[2] D.J. Hand, G. Blunt, M.G. Kelly, and N.M. Adams, "Data Mining for Fun and Profit," Statistical Science, vol. 15, no. 2, pp. 111-131, 2000.
[3] "Statistics for General and On-Line Card Fraud," http://www.epaynews.com/statistics/fraud.html, Mar. 2007.
[4] S. Ghosh and D.L. Reilly, "Credit Card Fraud Detection with a Neural-Network," Proc. 27th Hawaii Int'l Conf. System Sciences: Information Systems: Decision Support and Knowledge-Based Systems, vol. 3, pp. 621-630, 1994.
[5] M. Syeda, Y.Q. Zhang, and Y. Pan, "Parallel Granular Networks for Fast Credit Card Fraud Detection," Proc. IEEE Int'l Conf. Fuzzy Systems, pp. 572-577, 2002.
[6] S.J. Stolfo, D.W. Fan, W. Lee, A.L. Prodromidis, and P.K. Chan, "Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results," Proc. AAAI Workshop AI Methods in Fraud and Risk Management, pp. 83-90, 1997.
[7] S.J. Stolfo, D.W. Fan, W. Lee, A. Prodromidis, and P.K. Chan, "Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project," Proc. DARPA Information Survivability Conf. and Exposition, vol. 2, pp. 130-144, 2000.
[8] E. Aleskerov, B. Freisleben, and B. Rao, "CARDWATCH: A Neural Network Based Database Mining System for Credit Card Fraud Detection," Proc. IEEE/IAFE: Computational Intelligence for Financial Eng., pp. 220-226, 1997.
[9] M.J. Kim and T.S. Kim, "A Neural Classifier with Fraud Density Map for Effective Credit Card Fraud Detection," Proc. Int'l Conf. Intelligent Data Eng. and Automated Learning, pp. 378-383, 2002.
[10] W. Fan, A.L. Prodromidis, and S.J. Stolfo, "Distributed Data Mining in Credit Card Fraud Detection," IEEE Intelligent Systems, vol. 14, no. 6, pp. 67-74, 1999.
[11] R. Brause, T. Langsdorf, and M. Hepp, "Neural Data Mining for Credit Card Fraud Detection," Proc. IEEE Int'l Conf. Tools with Artificial Intelligence, pp. 103-106, 1999.
[12] C. Chiu and C. Tsai, "A Web Services-Based Collaborative Scheme for Credit Card Fraud Detection," Proc. IEEE Int'l Conf. e-Technology, e-Commerce and e-Service, pp. 177-181, 2004.
[13] C. Phua, V. Lee, K. Smith, and R. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection Research," http://www.bsys.monash.edu.au/people/cphua/, Mar. 2007.
[14] S. Stolfo and A.L. Prodromidis, "Agent-Based Distributed Learning Applied to Fraud Detection," Technical Report CUCS-014-99, Columbia Univ., 1999.
[15] C. Phua, D. Alahakoon, and V. Lee, "Minority Report in Fraud Detection: Classification of Skewed Data," ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 50-59, 2004.
[16] V. Vatsa, S. Sural, and A.K. Majumdar, "A Game-theoretic Approach to Credit Card Fraud Detection," Proc. First Int'l Conf. Information Systems Security, pp. 263-276, 2005.
[17] S. Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," ACM Trans. Information and System Security, vol. 3, no. 3, pp. 186-205, 2000.
[18] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. IEEE, vol. 77, no. 2, pp. 257-286, 1989.
[19] S.S. Joshi and V.V. Phoha, "Investigating Hidden Markov Models Capabilities in Anomaly Detection," Proc. 43rd ACM Ann. Southeast Regional Conf., vol. 1, pp. 98-103, 2005.
[20] S.B. Cho and H.J. Park, "Efficient Anomaly Detection by Modeling Privilege Flows Using Hidden Markov Model," Computer and Security, vol. 22, no. 1, pp. 45-55, 2003.
[21] D. Ourston, S. Matzner, W. Stump, and B. Hopkins, "Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks," Proc. 36th Ann. Hawaii Int'l Conf. System Sciences, vol. 9, pp. 334-344, 2003.
[22] X.D. Hoang, J. Hu, and P. Bertok, "A Multi-Layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls," Proc. 11th IEEE Int'l Conf. Networks, pp. 531-536, 2003.
[23] T. Lane, "Hidden Markov Models for Human/Computer Interface Modeling," Proc. Int'l Joint Conf. Artificial Intelligence, Workshop Learning about Users, pp. 35-44, 1999.
[24] L. Kaufman and P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley Series in Probability and Math. Statistics, 1990.
[25] K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, second ed. John Wiley & Sons, 2001.
[26] J. Banks, J.S. Carson II, B.L. Nelson, and D.M. Nicol, Discrete-Event System Simulation, fourth ed. Prentice Hall, 2004.