Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 35 (2008) 1721–1732

Expert Systems with Applications

www.elsevier.com/locate/eswa

Real-time credit card fraud detection using computational intelligence

Jon T.S. Quah [*], M. Sriganesh

School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Republic of Singapore

Abstract

Online banking and e-commerce have been experiencing rapid growth over the past few years and show tremendous promise of growth even in the future. This has made it easier for fraudsters to indulge in new and abstruse ways of committing credit card fraud over the Internet. This paper focuses on real-time fraud detection and presents a new and innovative approach in understanding spending patterns to decipher potential fraud cases. It makes use of self-organization map to decipher, filter and analyze customer behavior for detection of fraud.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Self-organizing map; Unsupervised learning; Risk scoring; Transaction

1. Introduction

The fast and wide reach of the Internet has made it one of the major selling channels for the retail sector. In the last few years, there has been a rapid increase in the number of card issuers, card users and online merchants, giving very little time for technology to catch-up and prevent online fraud completely. Statistics shows that on-line banking has been the fastest growing Internet activity with nearly 44% of the population in the US actively participating in it (Fighting Fraud on the Internet, 1999). As overall e-commerce volumes continued to grow over the past few years, the figure of losses to Internet merchants was projected to be between $5 and $15 billion in the year 2005. Recent statistics by Garner group place online fraud rate between 0.8% and 0.9%, with auction fraud accounting to nearly half of the total incidents of fraud on the Internet (Online Banking, 2005). Considering the current trends of e-commerce volumes, the projected loss is $8.2 billion in the year 2006, with $3.0 billion in the US alone (Statistics for General & Online Fraud, 2007).

1.1. How and where does fraud begin

In order to understand the severity of credit card fraud, let us briefly look into the mechanisms adopted by fraudsters to commit fraud. Credit card fraud involves illegal use of card or card information without the knowledge of the owner and hence is an act of criminal deception. Fraudsters usually get hold of card information in a variety of ways: Intercepting of mails containing newly issued cards, copying and replicating of card information through skimmers or gathering sensitive information through *phishing* (cloned websites) or from unethical employees of credit card companies.

Phishing involves acquiring of sensitive information like card numbers and passwords by masquerading as a trustworthy person or business in an electronic communication such as e-mail (Schneck, 2007). Fraudsters may also resort to generation of credit card numbers using BIN (Bank Identification Numbers) of banks. A recent scheme of *Triangulation* takes fraud fighters many days to realize and investigate (Bhatla, Prabhu, & Dua, 2003). In this method, the fraudster operates through an authentic-looking website, where he advertises and sells goods at highly discounted prices. The unaware buyer submits his card information and buys goods. The fraudster then places an order with a genuine merchant using the stolen card information. He then uses

[*] Corresponding author. Tel.: +65 67905871; fax: +65 62701556.
E-mail address: itsquah@ntu.edu.sg (J.T.S. Quah).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.08.093

the stolen card to purchase other goods or route funds into intractable accounts. Its only after several days that the merchant and card owners realize about the fraud. This type of fraud causes initial confusion that provides camouflage for the fraudster to carry out their operations.

1.2. Impact of fraud

It is interesting to note that credit card fraud affects card owners the least because their liability is limited to the transactions made. The existing legislations and cardholder protection policies as well as insurance schemes in most countries protect the interests of the cardholders. However, the most affected are the merchants, who, in most situations, do not have any evidence (eg. Digital signature) to dispute the cardholders' claim of misused card information. Merchants end up bearing all the loses due to chargeback, shipping cost of goods, card issuer fees and charges as well as their own administrative costs. Excessive fraudulent cases involving the same merchant can drive away customers, cause card issuer banks to withdraw service and also result in loss of reputation and goodwill (Yu & Singh, 2002). Card issuer banks have to bear the administrative cost of investigations into fraud cases as well as infrastructure costs of setting up the required software and hardware facilities to combat fraud. They also incur indirect costs through transaction delays. Studies show that the average time lag between the fraudulent transaction date and chargeback notification can be as high as 72 days, thereby giving fraudsters sufficient time to cause severe damage (Bhatla et al., 2003).

1.3. Fraud detection and prevention

The negative impacts of fraud make it very clear and necessary to put in place an effective and economical fraud detection system. Recent technological advancements to combat fraud have contributed number of solutions in this area (Bhatla et al., 2003; All points protection, 2007). Fraud detection techniques involving sophisticated screening of transactions to tracking customer behaviour and spending patterns are now being developed and employed by both merchants as well as card issuer banks (Tan & Thoen, 2000). Some of the recently employed techniques include transaction screening through Address Verification Systems (AVS), Card Verification Method (CVM), Personal Identification Number (PIN) and Biometrics. AVS involves verification of address with zip code of the customer while CVM and PIN involve checking of numeric code that is keyed in by the customer. Biometrics might involve signature or fingerprint verification. Rule-based methods and maintaining of positive and negative lists of customers and geographical regions are also used in practice. Data mining and credit scoring methods focus on statistical analyses and deciphering of customer behaviour and spending patterns to detect frauds (Huang, Chen, & Wang, 2007). Neural networks are capable of deriving patterns out of databases containing historical transactions of customers. These neural networks can be 'trained' and are 'adaptive' to the emerging new forms of frauds.

Deployment of sophisticated techniques and screening of every transaction alone will not reduce losses. It is necessary to employ an *effective and economical solution* to combat fraud (Turney, 1995). Such a solution should not only detect fraud cases efficiently but also turn out to be cost-effective. The idea is to strike a balance between the cost involved in transaction screening and review and the losses due to fraudulent cases. Analyses show that review of only 2.0% of transactions can result in reducing fraud losses accounting to 1.0% of total value of transactions. While a review of as high as 30% of transactions can reduce the fraud loses drastically to 0.06%, but that increases review costs exorbitantly (Bhatla et al., 2003). The estimated cost of not using anti-fraud software was about $60 billion in 2005 (Statistics for General & Online Fraud, 2007).

The key to minimize total costs is to categorize transactions and review only the potentially fraudulent cases. This should involve deployment of a step-by-step screening, filtering and review mechanism. A typical deployment can involve initial authentication of transactions through PIN, expiry date on card, AVS and CVM. A second level of screening can involve comparing with positive and negative lists as well as rules based on customers, geographical regions, IP addresses and policies. Risk and credit scoring with pattern and behaviour analyses can come next, followed by manual review. This classifies and filters out transactions as genuine or fraudulent in every step and as a result only a few transactions would require further manual review. Such a solution reduces the overall processing delay as well as total costs involved in manpower and administration.

The focus of this paper will now shift to risk scoring and behavioral pattern detection using neural networks.

2. Neural networks in fraud detection – literature review

Neural Networks have been extensively put to use in the areas of banking, finance and insurance. They have been successfully applied into credit scoring of customers, bankruptcy or business failure prediction, stock price forecasting, bond rating, currency prediction and many more areas (Stock Analysis, 2007; Quah & Srinivasan, 1999). In the area of fraud detection and prevention, neural networks like feed-forward networks with back-propagation have found immense applications (Fraud Brief – AVS & CVM, ClearCommerce Corporation, 2003; The Evolving Threat of Card Skimming, FairIsaac, 2007; ClearCommerce Fraud Prevention Guide, 2002). Usually such applications of neural networks systems involve knowing about the previous cases of fraud, to make systems learn the various trends. Fraud cases are statistically analyzed to derive out relationships among input data and values for certain key parameters in order to understand the various patterns of fraud. This knowledge of fraud trends is then iteratively taught to feed-forward neural networks, which can successfully identify similar fraud cases occurring in the future.

However, banks and merchant companies do not always know about the prior trends of genuine and fraudulent cases or there might not have been much effort done in the past to keep track of fraud cases. Also, many software products would involve complex statistical analyses of data and calculation of various parameters before making use of neural networks.

More recently, some research has begun to employ the *clustering* capabilities of neural networks in fraud detection (Zaslavsky & Strizhak, 2006). Self-organizing maps or Kohonen maps, a family of neural networks, are being suggested for forming customer profiles and analyzing fraud patterns. In this research, all transactions in the payment system (PS) are classified into *genuine and fraudulent sets*. Two hypotheses have been proposed:

- If a new incoming transaction on a card is similar to all previous transactions from the genuine set, then it is considered genuine.
- If a new incoming transaction on a card is similar to all previous transactions from the fraudulent set, then it is considered fraudulent.

Hence, the main idea involves forming patterns of 'legal cardholder' and 'fraudster' through neural network learning and to develop rules that govern the behaviors.

3. Self-organizing maps (SOM)

We will now briefly understand the concept of SOM and its applications into such type of classification. SOM is a neural network that employs *unsupervised learning* to configure its neurons according to the topological structure of the input data. This process, called as self-organization, is an iterative tuning of the weights of neurons so as to approximate the input data. The result is the clustering and profiling of input data (Kohonen, 1990; Zaslavsky & Strizhak, 2006; Kiang, Kulkarni, & Tam, 1995).

In the process of self-organization, the historical data is first identified and pre-processed. Symbolic data is converted to numerical data and multiple sets of input vectors are formed. These input vectors are multi-dimensional and may include parameters like transaction amount, transaction time, terminal number, etc. Cumulative quantities like total amount of transactions during the last D hours, number of terminals used by cardholder during the last D hours, etc. are also included (Zaslavsky & Strizhak, 2006). These input vectors are fed into the SOM and weights of the neurons are adjusted iteratively. Various distance measures can be employed during the iteration – Euclidean distance, Manhattan distance, Chebychev Distance, etc. (Zaslavsky & Strizhak, 2006). At the end of the training, historical data present in the database gets classified into genuine and fraudulent sets through the process of self-organization. Thereafter, any new incoming transaction is also pre-processed in the same manner and fed to the SOM. Based on a certain threshold value, its degree of similarity with the genuine or fraudulent sets is identified, and the final decision is taken on the type of this transaction.

3.1. Our approach

Our approach towards real-time fraud detection is best modeled by the prototype shown in Fig. 1. It is a multi-layered approach consisting of:

- The initial authentication and screening layers.
- The risk scoring and behavior analysis layer (core layer) (Fig. 2).
- A layer of further review and decision-making.

The initial authentication and screening layers, as stated before, may consist of verification mechanisms for PIN, address and expiry date, screening based on positive and negative lists and some maintained rules. The final review layer may consist of manual review or review through other mechanisms.

The purpose of the core layer is for risk scoring and customer behavioral analyses. The uniqueness in our approach lies here. It consists of two sub-layers – a layer of SOM followed by either a feed-forward neural network or rule-based risk scoring system.

The layer of SOM has multiple purposes:

- To *classify and cluster* input data.
- To *detect and derive hidden patterns* in input data.
- To act as a *filtering mechanism* for further layers.

The clustering and classification capabilities of SOM make it the appropriate neural network for realizing these

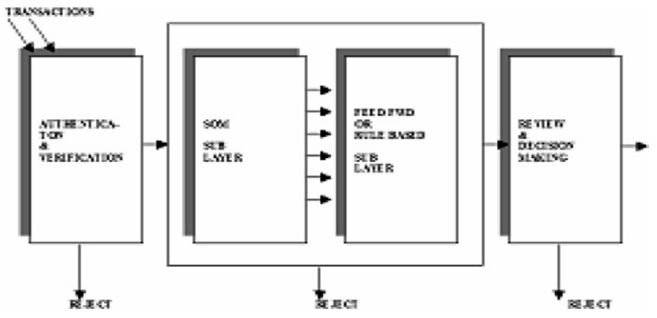


Fig. 1. Prototype structure.

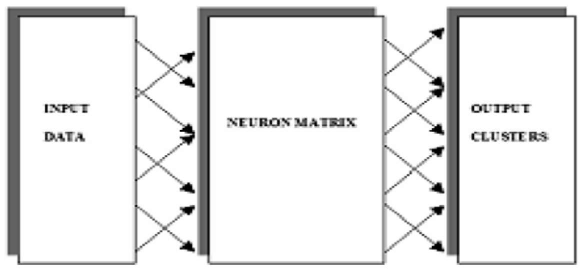


Fig. 2. SOM clustering sub-layer.

functionalities. As stated in the earlier part of this paper, banks and merchant companies do not always have clear knowledge of previous fraud cases and patterns. When historical data is fed into the SOM, many clusters and hidden interesting patterns can emerge without any external supervision. SOM can map complex multi-dimensional input data consisting of non-linear relationships to easily understandable one or two-dimensional data, which traditional linear statistical methods cannot do. SOM can analyze data similar to independent component analysis and come out with hidden interesting patterns. The extensive literature on SOM has ample proof of these capabilities (Kohonen, 1990; Vesanto & Alhoniemi, 2000; Serrano-Cinca, 1996). Many new fraud patterns, which otherwise can neither be derived nor imagined, can be unearthed through SOM.

Another interesting feature to note is the *filtering* capabilities of SOM. SOM need not necessarily be used to classify input data into genuine and fraudulent sets or for decision-making and risk scoring. We propose that random dense and sparse clusters of input data can be formed through SOM, which can then be used to judge the spending patterns of customers. A dense cluster for a customer might mean that many of his transactions are of one particular type, while a sparse cluster can mean that very few of his transactions are of another particular type. This can be used to filter out transactions for further processing. For instance, any transaction falling into a dense cluster within the range of a certain threshold value can mean that it is a commonly executed transaction for the customer and hence does not require any further review. Such transactions can be sent for authorization directly. It aids to identify certain 'suspicious' transactions that require further review. Such an filtering approach has many advantages – it reduces the overall time for processing a transaction, reduces the complexity involved in processing a transaction and also reduces the cost of review, which is an important aspect in fraud detection.

Usually, while pre-processing data for further analysis by neural networks or other risk scoring mechanisms, systems need to perform many complex computations and costly database queries. For instance, determining the number of transactions of more than $X performed in the last Y hours by a certain customer might require querying of databases and performing cumulative operations like summations and comparisons. Such operations might turn out time consuming and costly, adding up to the overall processing time and cost, especially when performed in real-time. Hence, such operations should be performed as minimal number of times and on as minimal number of transactions as possible. Most of the processing should be done directly on raw available data, which does not include cumulative values. SOM can directly work on raw data, and filter out those transactions that might require processing based on cumulative values.

Apart from these advantages, SOM can also be used to classify input transactions into predetermined sets like inter-bank transactions, bill payments, account transfers, etc. Two or more of such sets can be combined to obtain various transaction categories like 'common but fraud-prone', 'rare but safe', etc. Hence, SOM can be used for more than just dividing transactions into genuine and fraudulent sets.

The layer of feed-forward network or rule-based system can process and analyze the output of earlier SOM layer. Only 'suspicious' or potentially fraudulent transactions are passed on to this layer, which can analyze the transactions based on specific fraud patterns or trends. The feed-forward network with back-propagation can be made to learn specific fraud trends while the rule-based system can consist of specific inbuilt rules. This layer can project a numerical risk score based on the characteristics of each transaction. Such a single score can provide a good overall picture of the transaction and can aid in decision-making. Thresholds can be set such that only transactions having scores exceeding a certain value be sent for further review or rejected.

The following sections discuss the processing mechanism of our fraud detection system.

4. System design and architecture

The SOM forms the main component of our design. The interesting properties of the SOM and its potential applications to fraud detection have already been discussed. Here, we shall provide a detailed prototype as well as some practical aspects in the realization of SOM in fraud detection.

4.1. Input vectors

Our SOM sub-layer consists of a set of 'input vectors' fed to a neuron matrix, which can form clusters or patterns. Input vectors are usually high-dimensional vectors consisting of real-world quantities, which can be broadly classified as: Customer-related, Account-related and Transaction-related. Most of the customer-related and account-related quantities are static and are stored in the databases of banking systems. Transaction-related quantities are dynamic and will depend on the incoming transactions.

Customer-related quantities include customer number, branch of the customer, customer type (e.g. high profile, risk-prone), etc. Account-related quantities can include account number, currency of the account, account opening date, last date of credit/debit, available balance, expiry date, etc. Transaction-related quantities can include transaction reference number, account number, debit or credit, transaction currency, transaction timestamp, value-date (date on which the actual funds are transferred), PoS (Point of Sale) or terminal reference number, etc. During the training phase, cumulative quantities like number of transactions performed during the past X hours, number of transactions beyond $X, etc. may also be included. However, these are not included during real-time processing.

Such sort of classifications facilitates working on smaller data sets at any time and is also useful in differentiating data based on customers or geographical regions or volumes. This also enables the creation of individual customer profiles.

We represent the input vectors as $I_k = \{i_1, i_2, i_3, \ldots, i_n\}$.

4.2. Preprocessing of input vectors

It is important to note that many of the input quantities discussed above are *symbolic* quantities. Hence, some sort of preprocessing is required before they are fed to the SOM. One way of preprocessing is to convert them into numerical quantities and then normalize them so that they fall into a specific range. The conversion to numeric quantities can be done based on their frequency of occurrences in the overall input data. For instance, the account numbers of a customer can be CUSTACC1, CUSTACC2 and CUSTACC3. He might wish to perform transactions on CUSTACC1 more frequently than the other two. By determining the frequency or number of transactions for each of these accounts in a given set of transactions, and by normalizing the data using $Z = (N_i - M_i)/S$, where N_i is the frequency of occurrence of a particular account, M is the mean and S is the standard deviation, we can arrive at normalized numerical quantities.

4.3. Neuron matrix

The neuron matrix is like an *elastic blanket* of neurons that can adapt to patterns present in the input vectors. This is usually a two-dimensional array of neurons, which tends to 'learn' from each input vector. Also, the topographically close neurons activate each other and create a smoothing effect leading to global ordering. The commonly used similarity metric is the Euclidean distance given by:

$$C = \min\{\|x - m_i\|\},$$

which means the minimum of the distance of each neuron from the current input vector x. Other distance measures can also be used. The widely accepted smoothing function is the Gaussian function, which tends to have 'large impact' at the center and gradually weakening effect as the distance increases from the center. The 'modifications' on the neurons during the iterations also follow a gradually decreasing function such as $0.9 * (1 - t/\text{steps})$, where t is the current iteration number and *steps* is the total number of iterations to be performed. SOM tends to converge quite fast without many complex calculations.

4.4. Output vectors

The main output of SOM is the patterns and clusters it has deciphered out of the input vectors. These clusters may be dense or sparse and may be spread out in various ways across the neuron matrix. The output vectors can be of the form of $\{C_j, I_k, g\}$, where C_j can be a particular cluster number into which input vector I_k has fallen, while g can be a measure of similarity with the cluster. Based on the value of g and some preset threshold values, we can determine the nature of the transaction. These clusters sketch out the profiles of each customer and derive out spending and behavioral patterns. Separate customer profiles can be created for each customer.

4.5. Interpretation of the output of SOM

The output can be interpreted and used from various angles so as to arrive at important results:

- The clustering of input data into dense and sparse clusters shows the categories of transactions performed more frequently as well as rarely by each customer.
- Since the SOM brings out relationships between the various attributes present in input vectors, new hidden patterns can be unearthed. For instance, patterns such as a customer spending large amounts from only one of his accounts only when transacting from his workplace terminal and also only during month-ends, perhaps for paying off monthly bills or purchasing monthly commodities or purchasing goods once his monthly salary gets credited to his account. He breaches the pattern only when he is on leave or is traveling. If the extent of breach or 'suspicious activity' exceeds a certain threshold, then those transactions can be sent for review. In this way SOM can determine *adhoc rules* specific to customers, which neither can be generalized nor can be known beforehand to train feed-forward neural networks.
- SOM can filter out the number of transactions that need to be sent for review, thereby reducing overall processing time, cost and complexity.

As stated before, the output of the SOM sub-layer can form the input of a feed-forward neural network or any other risk scoring model.

4.6. Risk scoring sub-layer

This sub-layer can be a feed-forward neural network, any rule-based system or other risk scoring model. It would process only the suspicious transactions detected by the SOM sub-layer. Common trends in frauds can be identified and neural networks can be trained to detect similar cases of fraud. This needs to be a continuous process of learning and adaptation as new trends of frauds emerge. Based on the type of fraud, a certain risk score can be determined. Similarly, a rule-based or any other risk scoring method can analyze and search for specific fraud patterns from prior knowledge. For instance, repeated trials with the same card number but with invalid expiry dates can mean a high chance of potential fraud. Such a case can be assigned a high risk score. It is appropriate here to state some observed potential fraud patterns:

- The customer has changed personal information (including password) before performing a high volume transaction.
- Invalid logins before a transaction.
- Idle account has suddenly become active.
- Transactions have happened in the past D hours during which the customer has applied for card replacement.
- Transactions with unusual timestamps or from 'black-listed' geographical locations.
- Inter-country or inter-currency transactions of high amounts.
- New or unknown payee as per historical records.

Individual or combination of such cases can be identified and risk scores can be assigned to each based on the severity of the case.

4.7. Post training

Once the SOM and risk scoring sub-layers are trained based on historical data, they can be used to scan new incoming transactions real-time. In order to reduce processing time, the cumulative quantities may not be included in the input to the SOM. The SOM can directly work on the details available from the incoming transaction and place it into appropriate clusters. While the next sub-layer may calculate cumulative quantities and perform more complex calculations, if found suspicious by the SOM sub-layer. This reduces the number of transactions undergoing complex verifications making it feasible for real-time processing.

4.8. Integration into existing banking systems

Many existing banking systems have databases of banking data. Such data may be maintained in the form of tables of branches, customers, accounts, and transactions. Hence the customer-related, account-related and historical transaction-related information can be directly obtained from databases, to be fed into the Core sub-layers. Many banking systems even maintain cumulative quantities like number of invalid logins, number of transactions performed during the day, etc (Combating Money Laundering, 2007). Also, information like customer type, last date of credit or debit activity on accounts, date of account modification, daily opening and closing balances, value-dated balances for each day, etc. are maintained. Transactional details like transaction reference number, amount, currency, timestamp, value-date, etc. are directly stored in tables. Hence, most of the data required for profiling of customers and for detecting historical fraud patterns are available directly. In this sense, the fraud detection Core sub-layers can sit on top of the existing banking systems.

Typically, all new incoming transactions for customers are first authenticated and verified at a basic level and then stored in temporary tables. Such transactions are then authorized, moved to permanent tables and financials are 'posted' to accounts based on their value-dates. The fraud detection sub-layers can functionally be position in between the two processes, as shown in Fig. 3. The sub-layers would scan incoming transactions real-time and provide a feedback to the banking system. Based on their feedback, which can be in the form of a risk score, the banking system can either authorize or reject the transaction. Some transactions can even be sent for further review before authorization.

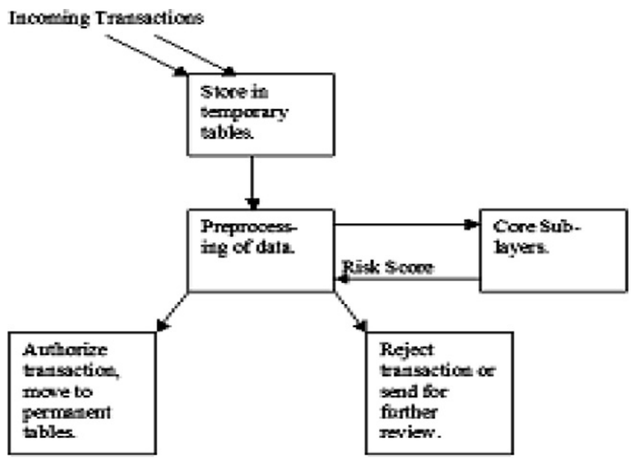

Incoming Transactions
Store in temporary tables.
Preprocessing of data.
Core Sub-layers.
Risk Score
Authorize transaction, move to permanent tables.
Reject transaction or send for further review.

Fig. 3. Integration with existing banking system.

4.9. Illustrative example

In this section, we shall demonstrate and describe our approach based on experimental data. The data used is from the test database (an extraction from the actual banking database) of a well-known bank[1]. We created tables with appropriate records for customers and their accounts. We then considered their transactional data for which the input quantities were variables such as:

- branch_code – branch code of the transaction.
- cust_ac_no – account number.
- ccy – debit currency in which the transaction is done.
- drcr_ind – debit or credit.
- cpty_ac_no – payee account number.
- locn_ref_no – terminal or PoS reference.
- txn_amt – transaction amount.

These variables were chosen mainly because they distinctively depict the features of any particular transaction.

Around 100 transactions per customer over a period of one month were considered. This is the usual practical scenario during active spending.

For each customer, the symbolic variables (Table 1) were converted to numeric values and then normalized

[1] A prominent bank with over 200 million customers in more than 100 countries. The data used in this study is taken from its branch in Singapore.

Table 1
Symbolic variables

	Br_Code	Ac_No	Ccy	DrCr_ind	Cpty_Ac_I	Locn	Txn amt
1	PL0	PL0123	SGD	D	CPTY01	LCN1	1000
2	PL1	PL1123	SGD	D	CPTY02	LCN2	1200
3	PL0	PL0123	SGD	D	CPTY03	LCN2	2300
4	PL1	PL1123	SGD	D	CPTY03	LCN2	1200
5	PL0	PL0123	SGD	D	CPTY01	LCN1	1000
6	PL0	PL0123	SGD	D	CPTY03	LCN2	1200
7	PL0	PL0123	SGD	D	CPTY02	LCN2	1200
8	PL1	PL1123	SGD	D	CPTY03	LCN1	1000
9	PL0	PL0123	SGD	D	CPTY01	LCN2	650
10	PL1	PL1123	SGD	D	CPTY03	LCN2	650
11	PL0	PL0123	SGD	D	CPTY01	LCN1	2300
12	PL1	PL1123	SGD	D	CPTY03	LCN2	650
13	PL2	PL2123	USD	C	SELF01	LUS01	2000
14	PL0	PL0123	SGD	D	CPTY03	LCN2	2300
15	PL2	PL2123	USD	C	SELF01	LUS01	1800

(Table 2). All these numeric transactional values were stored in tables and then fed to the SOM.

A neuron matrix of 70×70 nodes was used. Each neuron was addressed by indexes (x, y) in the matrix. Also, each neuron had a neuron vector associated with it. These neuron vectors had similar dimensions and quantities as the input vectors. They were initialized to random values.

The Gaussian function was used as the smoothing function. A monotonically time decreasing function $0.9 * (1 - t/\text{steps})$, where t is the variable iteration number, was used along with it for creating a gradual smoothing effect. For each customer, 200 steps of iterations were performed.

The whole experiment was done with two measures of similarity:

- Euclidean-distance function, as described earlier.
- *Gravity* function – This is another novel feature of our system. The gravity function is of the form $F = m_1 \cdot m_2/r^2$. Here, m_1 is the portion of input vector consisting of certain chosen scalar quantities, while m_2 is the corresponding portion of neuron vector. $m_1 \cdot m_2$ is the dot product of the two. r^2 is the square of the Euclidean distance between either the entire vectors or remaining portion of vectors.

For instance, our multi-dimensional vectors contained numeric values of {branch_code, cust_ac_no, ccy, drcr_ind, cpty_ac_no, locn_ref_no, txn_amt}. We considered m_1 and m_2 as txn_amt, while r was the Euclidean distance between the vectors excluding the txn_amt. Such an approach tends to give 'weightage' or 'importance' to certain quantities; txn_amt in this example. Also, the convergence tends to be faster.

The outputs in both cases were distinct clusters for each customer.

After this clustering step, example transactions were posted to simulate real-time processing. These transactions were first stored in temporary tables as unauthorized. They were then fed to the SOM to determine which cluster they would fall into and by what degree of 'nearness'. Based on whether the degrees were 'acceptable' or not, these transactions were either authorized or rejected by the system. The authorized

Table 2
Variables converted to numeric values and normalized

	Br_Code	Ac No	Ccy	DrCr_ind	Cpty_Ac_I	Locn	Txn amt
1	0.674453	0.674453	0.707107	0.707107	0.037635	−0.73363	1000
2	−0.73577	−0.73577	0.707107	0.707107	−0.15054	0.677199	1200
3	0.674453	0.674453	0.707107	0.707107	0.301084	0.677199	2300
4	−0.73577	−0.73577	0.707107	0.707107	0.301084	0.677199	1200
5	0.674453	0.674453	0.707107	0.707107	0.037635	−0.73363	1000
6	0.674453	0.674453	0.707107	0.707107	0.301084	0.677199	1200
9	0.674453	0.674453	0.707107	0.707107	−0.15054	0.677199	1200
3	−0.73577	−0.73577	0.707107	0.707107	0.301084	−0.73363	1000
9	0.674453	0.674453	0.707107	0.707107	0.037635	0.677199	650
10	−0.73577	−0.73577	0.707107	0.707107	0.301084	0.677199	650
11	0.674453	0.674453	0.707107	0.707107	0.037635	−0.73363	2300
12	−0.73577	−0.73577	0.707107	0.707107	0.301084	0.677199	650
13	0.061314	0.061314	−0.70711	−0.70711	0.564532	0.056433	2000
14	0.674453	0.674453	0.707107	0.707107	0.301084	0.677199	2300
15	0.061314	0.061314	−0.70711	−0.70711	0.564532	0.056433	1800

transactions were moved to permanent tables, their financials were posted and balances of accounts were updated. The system was implemented using PL/SQL, Oracle and C++, the details of which are given in the following sections.

4.10. Results

Figs. 4 and 5 show the plots of the output of SOM sub-layer. In our experiment, the co-ordinates (x,y) of the plots were in the range [0, 69]. As can be seen, there are distinct separate clusters formed, which are spread out across the matrix area. To identify the clusters formed, we selected cluster centers C_c and cluster radius C_r. Out of the neurons that have been activated by the input vectors, all those falling within the radius C_r from a C_c were considered as one cluster.

During real-time processing of new incoming transactions, we designated one more parameter called threshold radius R_t. If a transaction either fell within the cluster radius from a cluster center or within the threshold radius from a border neuron of a cluster, it was considered to have the same properties as that cluster. All distances were calculated using the Euclidean distance formula. We assumed $C_r = 1.0$ and $R_t = 0.05$. By varying the values of C_r and R_t, we arrived at various levels of tolerance in accepting transactions (Tables 3 and 4). Also, the measure of how near the transaction was from the cluster center acted as the input to the next sub-layer for calculating the risk score. The achieved accuracy was constant.

Tables 5 and 6 depict some sample results for a customer. A few points can be noted from these figures:

- The customer uses terminal LCN1 to transact with payee CPTY1. He uses terminal LCN2 to transact with payee CPTY2.

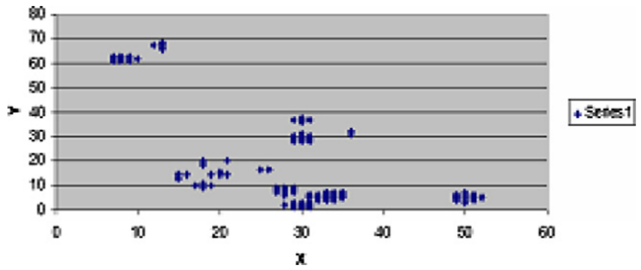


Fig. 4. Clustering using Euclidean distance.

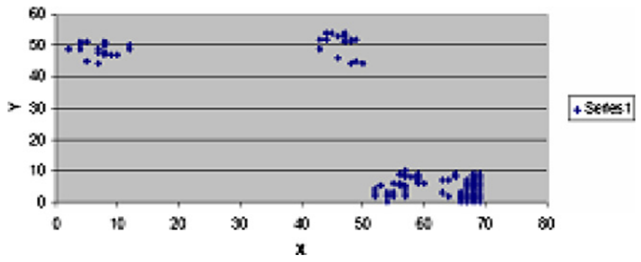


Fig. 5. Clustering using gravity.

Table 3
Clustering with varying cluster radius

No. of txns	Cr	Rt	No. of clusters
105.00	1.00	0.05	12
105.00	2.00	0.05	5-6
105.00	1.50	0.05	8-9

Table 4
Clustering with varying threshold radius

Cr	Rt	%Change in txn param	%Change in dist from cluster	Below threshold [20%]?
1	0.05	0	0	Y
1	0.05	20	11.80339887	Y
1	0.05	40	22.47448714	N
1	0.1	20	11.80339887	Y

- This may be considered something like he transacts with one particular payee from his home computer and with another payee from his office computer.
- The co-ordinates of the neurons that have adapted to similar transactions are nearby each other. For instance, transaction numbers 1 and 5 (Table 5) have co-ordinates (30,29) and (30,30) respectively, which are nearby each other.
- Similarly, transaction numbers 21 and 34 (Table 6) are nearby each other, since they have almost all parameters similar.

As we changed the input quantities, similar new clusters were formed. By adjusting the radii, we could vary the number as well as density of the clusters giving rise to various levels of sensitivity. On the simulated practical environment with around 100 transactions, 70X70 neurons and 200 steps of iterations per customer, the training time was less than 1 minute. The real-time processing was very fast and the system quickly authorized or rejected the transactions based on the output of the Core sub-layers.

5. Implementation details

5.1. Architecture

The working implementation of our model is made of the following layers:

- Bank application layer
- Fraud detection layer
- Backend database layer

The application layer has a hierarchical design consisting of the following levels:

- Bank
- Branch

Table 5
An example of similar transactions

	Br_code	Ac_No	Ccy	DrCr_ind	Cpty_Ac_I	Locn	Txn amt	*X*	*Y*
1	PL0	PL0123	SGD	D	CPTY01	LCN1	1000	30	29
5	PL0	PL0123	SGD	D	CPTY01	LCN1	1000	30	30
11	PL0	PL0123	SGD	D	CPTY01	LCN1	2300	29	29
16	PL0	PL0123	SGD	D	CPTY01	LCN1	650	30	28
29	PL0	PL0123	SGD	D	CPTY01	LCN1	800	31	29
46	PL0	PL0123	SGD	D	CPTY01	LCN1	2300	31	30
66	PL0	PL0123	SGD	D	CPTY01	LCN1	1200	29	30
69	PL0	PL0123	SGD	D	CPTY01	LCN1	2300	29	28
90	PL0	PL0123	SGD	D	CPTY01	LCN1	800	31	28
97	PL0	PL0123	SGD	D	CPTY01	LCN1	1000	30	31

Table 6
Another example of similar transactions

	Br_code	Ac_No	Ccy	DrCr_ind	Cpty_Ac_I	Locn	Txn amt	*X*	*Y*
7	PL0	PL0123	SGD	D	CPTY02	LCN2	1200	13	67
21	PL0	PL0123	SGD	D	CPTV02	LCN1	1000	30	37
26	PL0	PL0123	SGD	D	CPTV02	LCN2	800	12	67
31	PL0	PL0123	SGD	D	CPTV02	LCN2	1800	13	66
34	PL0	PL0123	SGD	D	CPTY02	LCN1	1800	30	36
51	PL0	PL0123	SGD	D	CPTY02	LCN1	800	29	37
81	PL0	PL0123	SGD	D	CPTY02	LCN2	1000	13	68
87	PL0	PL0123	SGD	D	CPTY02	LCN1	1000	30	38
92	PL0	PL0123	SGD	D	CPTY02	LCN1	1200	31	37

- Customer
- Account
- Authentication
- Contract
- Accounting

At the bank and branch levels, the basic data corresponding to the overall bank and branch are maintained. This includes bank name, bank code, branch code, branch local currency, etc.

The customer and account levels have customer details such as customer number, name, whether preferred customer, customer type, etc., and account details like branch code of account, account number, customer, currency of account, account opening date, available balance, dates of activity, card number, account status (dormant, deceased, blocked), whether inter-bank or inter-country transactions are allowed, etc.

The authentication level holds basic data for first level of authentication, such as card number, expiry date of card, number of invalid/successive logins, etc.

The contract level is for executing transactions. Various kinds of 'contracts' like funds transfer across accounts, bill payments, direct debit, etc, are first 'booked' at this level. Each contact is given a reference number for identification. It also carries debit/credit account numbers, contract date, currency, amount, PoS reference number, etc.

The accounting level contains of all accounting functionalities related to posting of financials and updating of account balances.

The fraud detection layer involves the SOM sub-layer and a rule-based layer for processing transactions that fail to pass through screening at SOM sub-layer.

The backend database layer consists of the tables for holding actual data. This is implemented on Oracle and forms the skeletal structure for data access and manipulation.

5.2. Mechanism

The authentication level ensures that basic level of verification while logging into the system is performed before allowing the customer to perform transactions. All transactions are booked as contracts into the system. These contracts are broken down into debit and credit legs. These transactions are then stored in temporary tables. These are not yet authorized and have status as 'U'.

The fraud detection layer is trained from historical transactions as discussed in Section 4.8.

The transactions from temporary tables are sent to the fraud detection layer for feedback on the degree of suspicion. Based on the feedback, these transactions are either authorized or rejected. Authorized transactions are marked as 'A', their accounting entries are passed and the financials are updated. The corresponding dated account balances are updated. The rejected transactions are marked as 'R' and the financials are left untouched. All authorized transactions are moved to permanent tables. The whole functionality is implemented in PL/SQL and C++.

6. Discussion

The uniqueness of our approach, as already discussed, lies in using the clustering and filtering capabilities of SOM for fraud detection. Clustering helps in identifying new hidden patterns in input data, which otherwise cannot be identified through traditional statistical methods. No *a priori* information regarding the spending patterns needs to be known. Also, the filtering of transactions for further review reduces the overall cost as well as processing time. With appropriate number of neurons and iteration steps, SOM tend to converge quite fast. *The layered design enables only the Core sub-layers to be separated and integrated into existing banking systems easily.* The sub-layers can be trained periodically so as to adapt to new trends of frauds.

The proposed approach is at early stage of a research aiming at using the properties of SOM to achieve efficient and cost-effective real-time fraud detection.

We would also like to point out some potential weaknesses in the approach, ways to overcome them and also some special observations we came across during our experiments. The clustering as well as measure of similarity depends on the cluster centers. Hence, choosing of appropriate cluster centers is very important. This can be done accurately by analyzing the plotted points as well as the properties of the transactions that have 'fallen' in the vicinity. Also, since the choices of cluster and tolerance radii determine the level of sensitivity, they too become important. There might be situations where an input transaction might fall into two different clusters. Hence, the similarity metric, number of neurons and smoothing function should be appropriately chosen so that distinct clusters are formed which are separated by desirable distances and spread across the whole matrix blanket. The input quantities should be appropriately chosen so that uninteresting patterns do not subdue interesting and important ones.

The use of gravity function is a proposed new approach and there is lot of scope for improvement in this. Choosing of appropriate quantities to represent the 'm-values' is important. Sometimes, the use of this function tends to 'attract' many unrelated input vectors into the same cluster. An appropriate scaling down factor (of the order of 0.25 or 0.5) can be used to reduce the 'force of attraction'. One might also use 'repulsion' along with 'attraction' to ensure that all input vectors do not cluster onto one region in the matrix.

7. Future work

As stated before, this approach is still in the early stages of research and hence there is ample scope for further research and extension of ideas.

The Internet platform is by itself a huge marketplace with lot of players – merchants, customers, banks and other financial institutions. Hence, the retail e-commerce system as a whole is very dynamic and voluminous. In such a system, the behavior of a customer is no longer just dependent on him alone. So, when it comes to profiling of customers and judging their spending patterns, a lot of direct and indirect factors come into picture. Let us understand this with a few instances. A person who has got a new high-paying job can suddenly start spending lavishly. If he has shifted his house farther away from his workplace, he might buy a new car and spend on everyday fuel. Hence, changes in lifestyle can affect spending patterns. Merchants may come out with new schemes, discounts and offers to affect spending patterns of customers. Changes in government policies (like reduction of taxes on goods) and bank schemes (like 5% cash-back on every $100 spent on card) can also affect spending patterns. On the other hand, customers may also reduce online spending if the media projects too many fraud cases! As all these happen, fraudsters are coming out with newer ways to commit fraud!

Hence, the key to accurate fraud detection and customer profiling lies in developing a dynamic system that can model and adapt to changing patterns in the e-marketplace and to converge more and more information from all fronts for decision-making. Other than customer profiles, merchant profiles, their selling patterns, rules and policies in the market, etc. need to be considered for accurate fraud detection.

Appendix A. Sample experiment results of fraud detection SOM sub-layer

In this appendix, transaction data of five randomly selected customers were fed to SOM and the outputs were plotted.

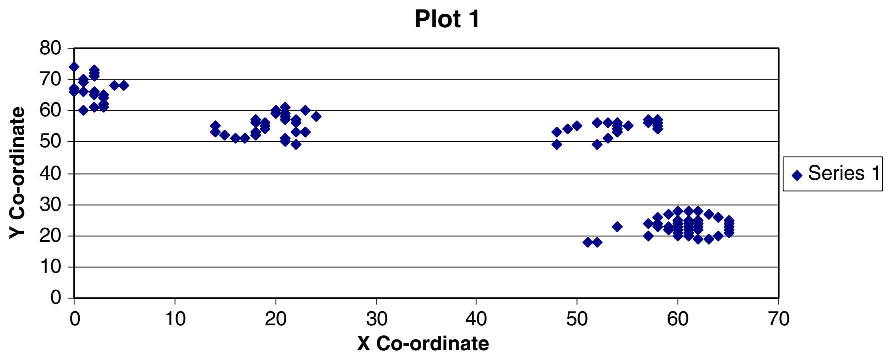
Number of transaction per customer: 105
Number of customers selected: 5
Number of nodes in SOM: 70×70
Number of steps of iteration: 200
Number of parameters considered: 7
Parameters considered:

- Branch Code of Transaction
- Account Number
- Currency
- Debit/Credit Indicator
- Counterparty (Payee) Account Number
- Location Reference Number
- Transaction Amount

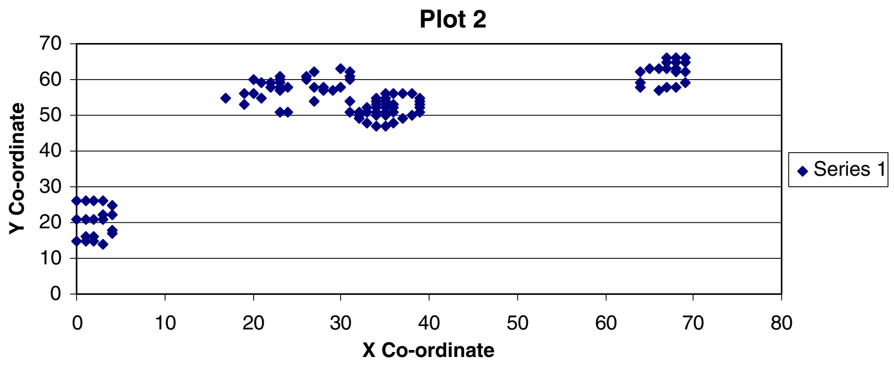
Similarity metric: Euclidean distance

The following are the plots:

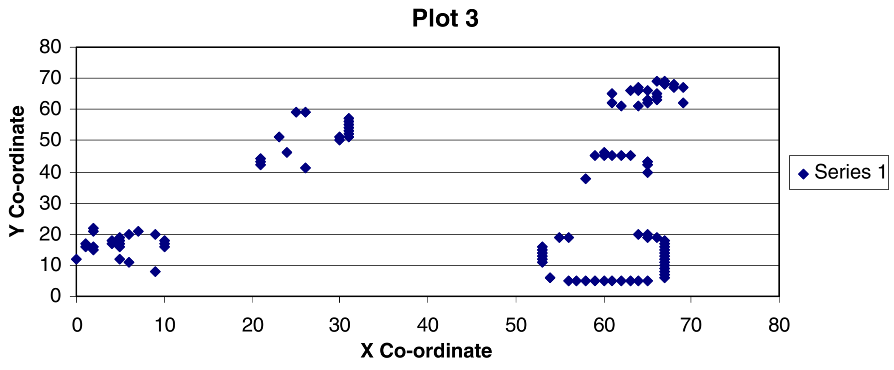
Customer 1



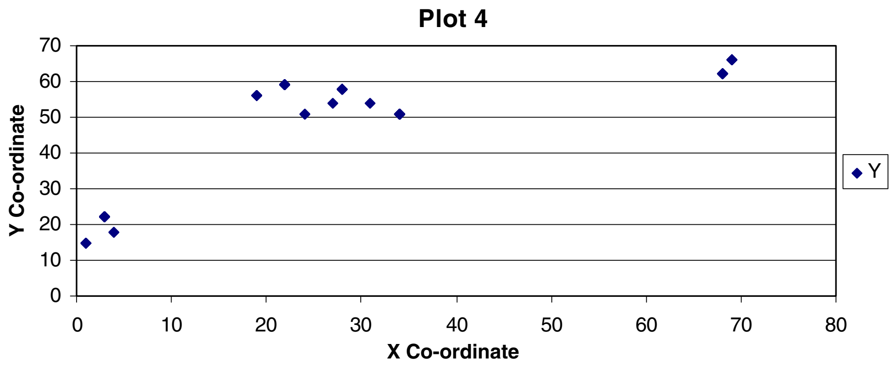
Customer 2



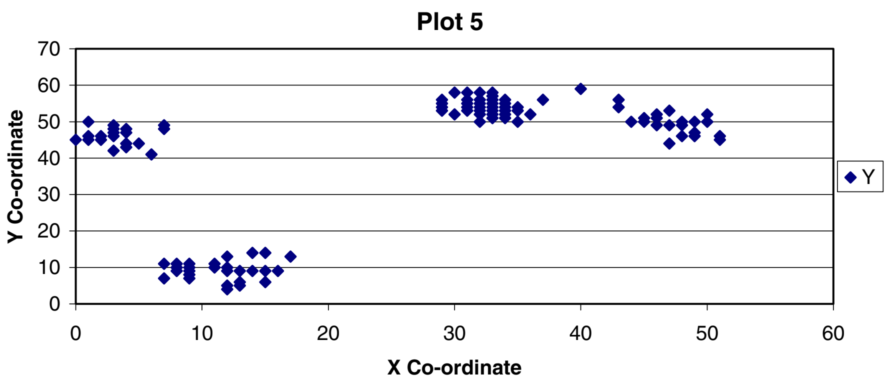
Customer 3



Customer 4



Customer 5



References

All points protection: One sure strategy to control fraud, FairIsaac, http://www.fairisaac.com, January 2007.

Bhatla, T.P., Prabhu, V., & Dua, A. (2003). Understanding credit card frauds, Cards business review.

Serrano-Cinca, C. (1996). Self-organizing neural networks for financial diagnosis. *Decision Support Systems, 17*(July), 227–238.

ClearCommerce fraud prevention guide, ClearCommerce corporation, 2002, http://www.clearcommerce.com.

Combating money laundering, iFlex Consulting, iFlex Solutions Ltd., http://www.iflexsolutions.com, Jan 2007.

Fighting fraud on the internet: An advanced approach, Meridian Research, September 1999.

Fraud Brief – AVS and CVM, ClearCommerce Corporation, 2003, http://www.clearcommerce.com.

Huang, C.-L., Chen, M.-C., & Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*(Aug).

Kohonen, T., 1990. The self-organizing map, In *Proceedings of the IEEE 78* (9), 1464–1480 (September).

Online banking, 2005: A Pew Internet Project Data Memo, http://www.pewinternet.org/PPF/r/149/report_display.asp.

Quah, T.-S., & Srinivasan, B. (1999). Improving returns on stock investment through neural network selection. *Expert Systems with Applications, 17*(4).

Schneck, P. A., 2007. The state of messaging security: From Email to VoIP, Strategic Development, CipherTrust Inc., http://www.ciphertrust.com, Jan.

Statistics for general and online fraud, ePaynews.com, http://www.epaynews.com/statistics/fraud.html#2, Jan 2007.

Stock analysis, http://www.financialtimingmodels.com, Jan 2007.

Tan, Y.-H., & Thoen, W. (2000). An outline of a trust model for electronic commerce. *Applied Artificial Intelligence, 14*, 849–862.

The evolving threat of card skimming, FairIsaac, http://www.fairisaac.com, Jan 2007.

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artifical Intelligence Research, 2*, 369–409.

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks, 11*(May), 586–600.

Yu, B., & Singh, M. P. (2002). Distributed reputation management for electronic commerce, Computational Intelligence 18, 535–549.

Zaslavsky, V., & Strizhak, A. (2006). Credit card fraud detection using self-organizing maps. *Information and Security, 18*, 48–63.