Adaptive Communal Detection in Search of Adversarial Identity Crime

Clifton Phua, Vincent Lee
Monash University
Clayton School of Information Technology, Wellington Road, Clayton, Victoria 3800, Australia
+61 411 883 108, +61 3 9905 2360
clifton.phua@infotech.monash.edu.au, vincent.lee@infotech.monash.edu.au

Kate Smith-Miles
Deakin University
School of Engineering and Information Technology, 221 Burwood Highway, Burwood, Victoria 3125, Australia
+61 3 9244 6320
katesm@deakin.edu

Ross Gayler
Baycorp Advantage
Level 12, 628 Bourke Street, Melbourne, VIC 3000, Australia
+61 3 8629 1687
ross.gayler@baycorpadvantage.com

"In conflict, straightforward actions generally lead to engagement; ***surprising actions*** *generally lead to victory."*

- Sun Tzu, ~500 BC, "The Art of War"

ABSTRACT

This paper is on adaptive real-time searching of credit application data streams for identity crime with many search parameters. Specifically, we concentrated on handling our domain-specific adversarial activity problem with the adaptive Communal Analysis Suspicion Scoring (CASS) algorithm. CASS's main novel theoretical contribution is in the formulation of State-of-Alert (SoA) which sets the condition of reduced, same, or heightened watchfulness; and Parameter-of-Change (PoC) which improves detection ability with pre-defined parameter values for each SoA. With pre-configured SoA policy and PoC strategy, CASS determines when, what, and how much to adapt its search parameters to ongoing adversarial activity. The above approach is validated with three sets of experiments, where each experiment is conducted on several million real credit applications and measured with three appropriate performance metrics. Significant improvements are achieved over previous work, with the discovery of some practical insights of adaptivity into our domain.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *Data mining*; I.2.6 [**Artificial Intelligence**]: Learning – *Parameter learning*; J.1 [**Administrative Data Processing**]: *Business*

General Terms

Algorithms, Experimentation, Security

Keywords

Adaptive fraud/communal detection, adversarial identity crime, automated credit application stream processing system

2007 ACM SIGKDD Workshop on Domain Driven Data Mining (DDDM2007), August 12, 2007, San Jose, California, USA.

1. INTRODUCTION

Our work foils fraudsters' attempts to commit identity crime with credit applications with our unique type of anomaly detection strategy. Here, we recommend an adaptive Communal Analysis Suspicion Scoring (CASS) algorithm [9] to examine application streams to detect the changing attack patterns (new false negatives) which are in direct response to our existing search parameters. This is necessary because committing this type of fraud tends to be very creative and adaptive because identity fraudsters find new ways of committing their crimes when established avenues are either cut off or blocked. Therefore, we have dynamic detection models that continually learn from examples seen and keep automatically adjusting their parameters and their values to match the latest patterns of these criminals.

Detection of identity crime in credit applications (credit cards or personal loans) is beneficial to both financial institutions and the general public. Our previous work on CASS addressed this problem by recommending real-time, stream analysis of similarity between the incoming applications and recent prior applications from a credit bureau point-of-view. Our approach is rapid as it is not completely dependent on flagging of known fraudulent applications. Within a short time frame, fraudsters generate many more similar applications on average than legitimate applicants. In [9], we proved that high similarity, after filtering out legitimate relationships and data errors, indicates high suspicion.

Yet, in general, constant fraudster/criminal innovation tends to triumph over existing real-time detection techniques. We term this phenomenon as the adversarial activity problem. Specific to our credit application domain, identity fraudsters have no cost associated with changing the false details on their applications and intentionally craft their applications to look creditworthy. As identity fraudsters continuously morph their style(s) to avoid detection, it is one of the most attractive domains to find and study adaptive behaviour. Our key research question here is to determine if dynamic search parameters can be more useful than static ones to hinder adversarial activity.

In addition to one-scan of data, real-time responsiveness, and incremental maintenance of knowledge, CASS reduces false positives by accounting for real-world, genuine communal relationships/links between people (family, friend, housemate, colleague, or neighbour) using continually updated whitelists and improves data quality. Its design is to calculate attribute vector,

approximate communal score, average previous score, and overall suspicion score [9].

In this paper, we extend previous work by incorporating *adaptivity* into our stream detection system so that it has the strategic ability to automatically decide:

- *When* adversarial counter-measures and new attacks, volume drift, concept drift, and population drift happen and select *state-of-alert*; and,
- *What* and *how much* to change/tradeoff the current system's parameter settings with *parameters-of-change*. This is to *ensure* adversaries do not have perfect knowledge of current system settings and to *sustain* optimum performance.

In an automated stream processing system, it is very hard to ascertain the cause of changes in data immediately. It could be deliberate adversarial influence and/or other naturally occurring external influences. The latter include volume drift where overall demand fluctuates from marketing campaigns of individual subscribers or entry/exit of subscribers; concept drift where the legitimate applicant demographics change over time; and population drift where volume of both illegitimate and legitimate classes fluctuate independently of each other.

To complicate things, these different influences can possibly happen simultaneously. For example, a sudden and significant increase in similarity/density could either be an adversarial offensive move and/or a large group of family members submitting multiple legitimate applications (output-based) and/or an increase in credit demand (input-based).

Indeed, the real reason can only be understood when looking back in retrospect (although understanding is completely dependent on accurate, complete, and timely positive class labels). This is too slow. Therefore, our adaptive version of the CASS has to account for all possible influences, not just adversarial operations, on streams.

Section 2 summarises the research problems and solutions of related work on adaptive detection. Section 3 discusses the business problem and algorithm's current parameters at the mini-discrete data stream; and section 4 focuses on the adaptive parameters at the micro-discrete data stream. Section 5 gives a brief description of the challenges in identity data and section 6 argues for the use of three performance measures. Section 7 presents the results, section 8 discusses the results and their limitations, and section 9 concludes the paper with future work.

2. RELATED WORK

Selected works and their relevant ideas on adaptivity from spam, intrusion, and duplicate detection are highlighted here.

2.1 Spam Detection

Game theory is applied to a spam detection scenario where the only current solution is to regularly re-train classifier by hand [6]. The authors propose simplified modelling of the cost-sensitive strategic interaction between a naïve Bayes classifier and an adversary to automatically re-train the classifier. They assumed perfect knowledge between the two opponents, tested under different false positive costs, and argue that *this can keep system up-to-date without manual intervention for longer times*. In practice, [8] highlight that adversaries learn how to generate more false negatives from prior knowledge, observation, and experimentation.

The email server is used to detect spam [13]. Emails are first encoded as hash-based text and large volumes of similar emails are found with document space density. *By searching for similarity/density, no class labels are required and detection time is fast*. Also, on a small sample of labelled emails, support vector machines (SVM) algorithm is the best supervised algorithm but the detection time is too long for an event-driven system.

2.2 Intrusion/ Duplicate Detection

Buffer overflow attacks are detected in programs and are shut down, and the detection system is upgraded to higher security mode [5]. Programs are compiled with a two-mode system which either detects changes to return address (high speed, low security) or by preventing any change (low speed, high security). StackGuard adaptively changes between these two modes and concludes that it is not a matter of which mode is better but how to *balance the tradeoffs of each mode*.

Approximate duplicates are identified for data cleaning in databases which can be adapted to any specific record linkage domain [1]. *A small training set of labelled duplicate and non-duplicate pairs is used to learn similarity metrics threshold for each attribute.* The thresholds are then applied during record linkage, and the resultant labelled numerical feature vectors are used as the training set to train a binary SVM classifier.

3. COMMUNAL DETECTION OVERVIEW

This section defines the business aspect of the problem followed by the newer version of the non-adaptive CASS (Communal Analysis Suspicion Scoring) algorithm [9], simplified discussions on the different types of static input parameters, and the output measurements at the mini-discrete data stream.

3.1 Business Viewpoint

Stage 1:
Diverse Applicants
Stage 2:
Multiple Subscribers
Stage 3:
Decision Making
Stage 4:
Central Processes

Stage 5:
Central Databases

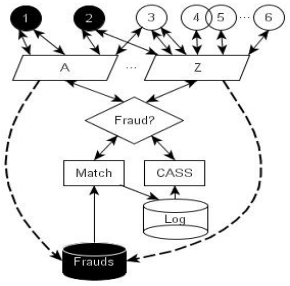

Figure 1: Overall view of important stages in a real-time, stream processing identity crime detection system

3.1.1 Stage 1

With reference to Figure 1, fraudster(s) may possibly submit some applications to gain illegal access to credit. At the same time,

many normal people will submit applications through the Internet or paper-based forms. When compared to each other, the concept of *implicit-ness* applies as some of these identity examples will match exactly or approximately.

A particular fraudster ❶ can submit similar applications to the same subscriber; another fraudster ❷ can submit similar applications to different subscribers (organisations which pay a fee for the fraud detection service). Fraudster(s) will either commit multiple identity deception through true identity theft (misuse someone else's identity information), synthetic identity fraud (create a completely fictitious identity), or a combination of both.

From a system's point-of-view there are many real identity examples and relatively few fake ones. For legitimate reasons, the same person ③ can apply for multiple credit products with the same or different subscribers, and different people ④⑤ usually have communal relationships (family, friend, housemate, colleague, or neighbour) to each other. *Sparsity* is a key characteristic of the data as majority of people ⑥ have identity details which are distinctly unique from others.

3.1.2 Stage 2

When individual subscribers receive the applications, they may have their own internal fraud/credit checks before querying the identity examples against the external databases. As a result, the credit bureau receives a single, overall, braided, and episodic *stream* of identity examples [7] from multiple subscribers.

Subscribers A to Z may have different stream arrival rates and sudden behavioural changes, and contribute different number and types of identity attributes. Yet, monitoring of many subscribers simultaneously has shared benefits because it is common for more than one subscriber to be attacked with the same identity attribute values.

3.1.3 Stages 3, 4, and 5

For checking each incoming application to known frauds, a similarity score is given to the individual subscriber to accept/reject application. Our CASS approach is complementary to known fraud matching but requires *evaluation* from the overall detection system's point-of-view using appropriate performance metrics. Matching is done against the periodically updated known frauds database and is a subset of the fraud enquiry log; and CASS processes incoming applications against the continually updated fraud enquiry log.

With reference to figure 1, "Fraud?" yields a binary decision: for the individual subscriber to decide whether the application is fraud or not. "CASS" and "Match" are processes: "CASS" is our proposed stream-based fraud detection algorithm; "Match" is an existing known fraud matching routine.

3.2 Current CASS Algorithm

Another way of understanding CASS is to visualise it as a directed acyclic graph with all edges pointing backwards in time. There are currently eighteen static parameters and seven output measurements in CASS. Each parameter is an adjustable quantity that governs some aspect of the system's performance and applicants' characteristics. They are non-adaptive as the same parameter setting is applied to the entire continuous data stream.

Table 1: Newer communal detection framework overview for the mini-discrete data stream adapted from [9] "_" denotes modified or new

Input	Process	Output
g_x, $\underline{u}_{x,y}$, $\underline{W}$, s, $\underline{c}$ $T_{similarity}$, $T_{attribute}$, T_{EI}, α, *Cross-match*, *Fuzzy-match* N, w_{normal}, w_{Rx}, ς, β, η, $\underline{\lambda}$	Process each current example against previous examples within an example-based window 1) Calculate single-link attribute vector and compare to whitelist and improve data quality 2) Calculate single-link communal score 3) Calculate single-link average previous suspicion score 4) Calculate multiple-links suspicion score for each current example by adding and smoothing current communal score and average previous suspicion score of all single links	$S(v_i)$, $E_I(v_i)$, $E_O(v_i)$ G_x, R_x, Ω_x, δ_x

In Table 1's "Input" column, the first five parameters are for data extraction. The second six parameters are to compare identity examples. The third three parameters are to set up a whitelist (with three attributes for each whitelist item: link-type attributes, volume, and weight) for each mini-discrete data stream. And the last four parameters are to trade-off speed and security, and improve data quality. The four key parameters are explained in Section 3.3 and the other thirteen parameters are listed in the appendix.

The "Process" or algorithm has four main calculation steps to generate a suspicion score for each identity example. The "Output" column shows that suspicion scores/inlinks/outlinks, graph, whitelist, classifier, average score and links are created as mini-discrete data stream results; and can be segmented into months and subscribers.

3.3 Key Parameters

Out of the four key parameters, the two most interesting ones to experiment with are W and $T_{attribute}$ because they have strongest influence on results [9]. To explain further, W determines the search space while $T_{attribute}$ determines the minimum requirement of attribute matches for a pair-wise example match. Each of these two parameters' values has inherent tradeoffs between speed (system resource constraints) and security (high true positive and low false positive rates). For example, a low W which searches fewer examples is faster but poorer than a higher W. Another example is a low $T_{attribute}$ which will result more examples as matches is slower but can be better than a higher $T_{attribute}$.

The data stream extraction parameter W is an example-based window which indicates how many previous examples v_j to compare the current example v_i against for a numeric suspicion score. In previous work, v_i is originally matched against W number of subsequent v_j within a large $u_{x,y}$, and a set of *historically* scored (more suspicious) examples u_s. In current work, since $u_{x,y}$ is time-based and small and to simulate on-demand query and analysis of data stream, v_i is matched against W number of a subsequent set of *recently* scored (less suspicious) examples u_λ and to u_s. W is split between recent examples u_λ and historical examples u_s, as described below. As a guideline, the initial W should exceed 10,000. The critical pair-wise example comparison parameter $T_{attribute}$ is the attribute threshold for the minimum number of matched attributes required to connect v_i to v_j for a communal score.

The other two important parameters which are specific to our domain include c and λ (lambda). The data stream extraction parameter c is the new recency factor to determine the proportion of previously scored examples in u_λ and u_s with default of 0.5, where $W = [c * u_\lambda] + [(1 - c) * u_s]$. The data quality improvement parameter λ is the approximate duplicate factor from the same subscriber and λ should not be too large.

3.4 Mini-Discrete Data Stream Output Measurements

For each current/incoming example, $S(v_i)$ is the suspicion score. $E_I(v_i)$ is the number of inlinks, with a default of 0; and $E_O(v_i)$ is the number of outlinks.

After each current mini-discrete data stream, G_x represents the simple direct graph with vertices and edges and R_x represents the whitelist. Ω_x (omega) represents the average suspicion score per example and δ_x (delta) represents the average links per example.

3.5 Suspicion Score

$$S(v_i) = \sum_{v_j \in M(v_i)} \left[w_{i,j} + S(v_j) / E_O(v_j) \right] \quad (1)$$

where $S(v_i)$ is the total suspicion score of the current credit application v_i with k-pairs (multiple-pairs) of linked v_j and $S(v_i) \geq 0$. $M(v_i)$ is the set of k examples which v_i links to. Therefore, a high suspicion score $S(v_i)$ indicates three possibilities: strong connection between v_i and v_j (high link communal weight - $w_{i,j}$) which is dependent on number of non-missing attribute values; and/or the high quality of v_j (high average suspicion scores - $S(v_j)/E_O(v_j)$); and/or the large quantity of v_j (many connected examples - k).

$$S(v_i) = \sum_{v_j \in M(v_i)} \left[(1-\alpha) * w_{i,j} + \alpha * S(v_j) / E_O(v_j) \right] \quad (2)$$

where α is exponential smoothing factor incorporated into (1) and needed to gradually discount the effects of previous suspicion scores and $0 \leq \alpha \leq 1$ [9].

4. ADAPTIVE PARAMETERS

This section introduces the adversary strategic attack cycle. This is followed by an overview adaptive CASS algorithm, a non-trivial extension of the static version from the previous section. Then, the two adaptive parameters - dynamic cause and dynamic effect - which decides the *when*, *what*, and *how much* to change the parameters are discussed.

4.1 Adversary Strategic Attack Cycle

Central to most adversarial detection domains such as spam, intrusion, and fraud is the presence of determined/professional adversaries and their three steps in the strategic attack cycle. Below is our domain-specific identity crime or credit application fraud version of the cycle:

4.1.1 Step 1

To probe for vulnerabilities, adversaries will flood the system with a volume of new diverse attack styles. Each style is a group of similar identity examples which can be real (harder to obtain but easier to succeed) or synthetic (effortless to create but more difficult to get approved).

4.1.2 Step 2

In our domain, some of these styles will be rejected by the system because they are associated with known frauds (previous real or synthetic styles) or not associated with any credit history (new synthetic styles). These rejected styles are not likely to be used by fraudsters again.

4.1.3 Step 3

Some styles will be accepted by various subscribers (depending on their willingness to extend credit without physical identity verification and to higher risk applicants). Subsequently, adversaries will re-use/duplicate these styles and submit them to increase the system's false negatives until their *modus operandi* is discovered. Step 1 is then repeated, as with newly gained knowledge about the parameters of the detection system, adversaries take new counter-actions to mitigate the undesirable effects on them.

4.2 Adaptive CASS Algorithm

We learnt that the emerging patterns (EP) algorithm [11] has the potential for solving the adversarial activity problem. In our system, the EP algorithm basically searches for multi-attribute labelled links which belong to most fraudulent identity examples but none or few to legitimate identity examples, and scores new identity examples accordingly. However, in EP experiments which are not described in this paper, we only found six statistically significant emerging fraud link patterns in several hundred unique and distinct fraud link patterns.

As our alternative, the key idea here is to be sensitive to any significant changes and constantly adapt appropriate parameter settings before processing every new micro-discrete data stream. By changes, we mean assigning a state-of-alert (SoA) by detecting deviation from an established baseline. Applied to the entire continuous data stream, this adaptive version tunes selected parameters-of-change (PoC) based on the SoA policy. Although there is work on a middleware system [3] which explores adaptive parameters within a stream environment, this is the first intelligence and security informatics paper to safeguard against the adversarial activity problem. The manual configurations of state-of-alert policy and parameters-of-change strategy are described in sub-sections 4.2 and 4.3 respectively.

There is a need to adapt to single or multiple adversaries operating simultaneously and to their *modus operandi*'s temporal characteristics: once-off, occasional, seasonal, and frequent styles/link-types. There are already frequent and obvious fraud styles of related identities which have been discovered and filtered out of the whitelist.

In Table 2's "Input" column, original parameter setting refers to the initial configuration of eighteen static parameters at the start of the algorithm. The next four measurements (not parameters) are long-term (baseline) and short-term indicators of previous average suspicion score and previous average links per example. The two adaptive/dynamic parameters, dynamic cause *DC* and dynamic effect *DE,* are the most important because they are executed at run-time to tune parameters.

DC and *DE* determine first two operations and last two operations respectively in the "Process" column. *State-of-alert* (SoA) is the condition of reduced, same, or heightened watchfulness. In order to determine the state-of-alert, input size and output interestingness are measured and compared between mini- and micro-discrete data streams.

Parameter-of-change (PoC) is a pre-set parameter which has potential to significantly improve the system's detection ability. It has pre-defined parameter values for each state-of-alert. The "Output" column gives the same or new parameter settings as well as the current short-term average suspicion score and average links per example.

Table 2: Adaptive communal detection overview at the micro-discrete data stream

Input	Process	Output
Original parameter setting with the measurements Ω_{x-1}, δ_{x-1}, $\Omega_{x,y-1}$, $\delta_{x,y-1}$ *DC, DE*	Before processing any micro-discrete data	*Same or New parameter setting with the measurements* $\Omega_{x,y}$, $\delta_{x,y}$
	1) Calculate input size of current micro-discrete data stream	
	2) Determine the *state-of-alert* between mini- and micro-discrete data streams based on policy	
	3) Select *parameter-of-change* based on strategy	
	4) Apply parameter value(s) to suit state-of-alert	

4.3 Dynamic Cause *DC*

For concreteness, the following are three recommended policies to determine state-of-alert (SoA) to trade-off efficiency (input size) and effectiveness (output suspiciousness):

4.3.1 Input SoA policy

Using only input size with an input threshold T_{input} as baseline dependent on $u_{x,y}$'s time representation:

$$\text{SoA} = \begin{cases} \text{low} & \text{if } u_{x,y} \geq T_{input} \\ \text{medium} & \text{else} \end{cases} \quad (3)$$

4.3.2 Output SoA policy

Using only output suspiciousness with Ω_{x-1} and δ_{x-1} as baseline dependent on g_x's time representation:

$$\text{SoA} = \begin{cases} \text{medium} & \text{if } \Omega_{x-1} \leq \Omega_{x,y} \text{ and } \delta_{x-1} \leq \delta_{x,y} \\ \text{low} & \text{else} \end{cases} \quad (4)$$

4.3.3 Both SoA policies

Combining both (3) and (4):

$$\text{SoA} = \begin{cases} \text{low} & \text{if input SoA} = \text{low} \\ & \quad \text{and output SoA} = \text{low} \\ \text{high} & \text{if input SoA} = \text{medium} \\ & \quad \text{and output SoA} = \text{medium} \\ \text{medium} & \text{else} \end{cases} \quad (5)$$

The SoA policies can have more than three states. For example, it is possible to have five states where there are two additional states - very low SoA and very high SoA.

4.4 Dynamic Effect *DE*

There are eighteen static parameters to choose from (see sections 3.2 to 3.5) as parameters-of-change. To achieve the effect of "surprising actions", there must be a sufficient number of parameters-of-change where the system can randomly select to ensure adversaries do not have perfect knowledge of current system settings. At the same time, it should be able to adapt to other drifts.

For concreteness, there are two recommended strategies to use the selected parameters-of-change (PoC) to suit efficiency (low SoA), no change (medium SoA) and effectiveness (high SoA).

4.4.1 Single PoC strategy

This involves systematic choice of parameter values. For example, if PoC is W and initial W = 10000, then depending on the SoA, the PoC value will either be 5000, 10000, or 20000 only.

$$\text{PoC values} = \begin{cases} W/2 & \text{if SoA = low} \\ W & \text{if SoA = medium} \\ 2W & \text{if SoA = high} \end{cases} \quad (6)$$

Another example, if PoC is $T_{attribute}$ and initial $T_{attribute}$ = 3, then depending on the SoA, the PoC value will either be 4, 3, or 2 only.

$$\text{PoC values} = \begin{cases} T_{attribute} + 1 & \text{if SoA = low} \\ T_{attribute} & \text{if SoA = medium} \\ T_{attribute} - 1 & \text{if SoA = high} \end{cases} \quad (7)$$

One SoA policy and one PoC are paired. For example using (6), there can be an "input SoA policy" with *W* as the "Single PoC" and this is denoted as *input/W*. Another example using (7), there can be "Both SoA policies" with $T_{attribute}$ as the "Single PoC" and this is denoted as *both/*$T_{attribute}$. Given three SoA policies and five PoCs, there are 3 x 5 = 15 different SoA-policy/PoC combinations, one of which the system can set as an "action".

4.4.2 Multiple PoCs strategy

This entails random choice of parameter, on top of systematic choice of parameter value. ξ (xi) is the random function with at least two SoA-policy/PoC pairs as inputs which are all equally likely to happen. Each SoA-policy/PoC pair decides *when* change is necessary, and *what* and *how much* to change. For example, the adaptive CASS algorithm randomly selects *input/W* or *both/*$T_{attribute}$ and this is denoted by ξ*(input/W*, *both/*$T_{attribute}$*)*. In the same random function, it is possible to have different SoA policy for the same PoC, and vice versa. Given a random function with *n* SoA-policy/PoC input pairs, there will be *n*-1 possible "surprising actions". In other words, the multiple PoCs strategy is a systematic plan, continuously monitored and adapted, to maintain or improve the algorithm's performance.

5. IDENTITY CRIME DATA

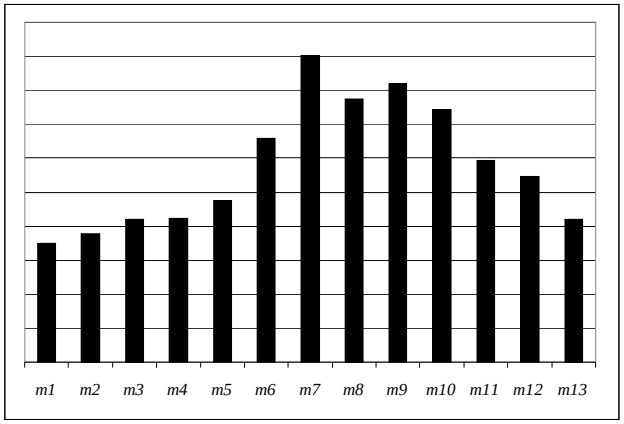

Figure 2: Monthly proportion of known frauds in applications

Due to privacy and confidentiality reasons, exact numbers on data volume, all the actual attributes, and costs are not published here. Instead, we use broad descriptions of these numbers. Figure 2 highlights the fact that months *m7* to *m10* (we treat each month as a mini-discrete data stream in this paper) have the highest rate of fraud but this is most likely false. For this study, the real and recent data extracted from the fraud enquiry log spans a total of 13 consecutive months (*m1* to *m13*), but only linked to 8 months of known frauds data (*m7* to *m14*). Ideally, fraud from *m10* to *m13* should be the same or higher than *m9* because the volume of applications increased after *m9*. In other words, the known fraud rate and its derived CASS true positive rates have been understated in this paper, especially the earliest and latest months. However, even if we take the highest fraud rate *m7* as the average of all months, the overall known fraud rate will still be less than 1%.

In addition, this is a highly technical task because of the extreme class imbalance (much lesser than 1% known frauds in all data), large scale (several million structured examples), and sparse identity attributes (several dozen of them such as names, addresses, and date-of-birth). To complicate matters, when the data set is modelled as a data stream, there are few significant indicators of fraud in temporal (monthly, daily, and hourly), spatial (country, state, and suburb), and sub-stream (a few dozen subscribers) patterns.

6. PERFORMANCE METRICS

Our analysis has to recognise two distinctive characteristics in the CASS algorithm scores (they are different from classifier scores): First, they can exceed one since each identity can be similar to many others. Second, most of them are zeros since identity examples are usually sparse. For our assessment purposes, *all the zero scores have been removed* since they will not be investigated, and metrics which use true negatives such as accuracy and receiver operating characteristic curves are avoided since false positive rates are likely to be understated [4]. The metrics with a quote appended have omitted zero scores for assessment. The following three most complementary and appropriate quantitative assessments [2] for the CASS algorithm on this fraud phenomenon based on our practical experience.

6.1.1 F-measure curve FME'

- A single curve consists of multiple values under different thresholds.
- Each value depicts a trade-off between precision *PRE* and recall *REC*.
- At each threshold increments of 0.1 from 0 to 1:

$$FME' = 2*(PRE*REC)/(PRE+REC) \quad \text{where} \quad PRE = TP/(TP+FP) \text{ and } REC = TP/(TP+FN) \quad (8)$$

6.1.2 Percentage of detected from all frauds PDF'

- Single-valued and easy to interpret.
- Acts as a check and balance to *FME'* because there can be good *FME'* results but with very low *PDF'*.
- Same as *REC* with threshold of 0.001.

6.1.3 Percentage of detected frauds in riskiest k percent of examples NTOP-k

- Single-valued and easy to interpret.
- Concerned with most suspicious by ranking scores in descending order, and in our case, investigating only top 1% of non-zero examples with *NTOP1*.
- Similar to *FME'* and *PRE* with highest possible threshold.

7. EXPERIMENTS

7.1 Design

Due to page limits, only the 18 representative experiments are reported (see Table 3) to illustrate our adaptive concepts. Each experiment is validated against 13 months worth of several million identity examples. They are sub-divided into experimental sets of 6 in *d*, 6 in *e*, and 6 in *f*, where each set is a similar group of experiments. Experimental set *d* is to verify the results from the previous work, *e* and *f* are to address the adversarial activity problem specific to our identity crime application domain.

The baseline experiment *d2* uses the following parameters based on our previous experience and current domain knowledge:

- g_x = "month", $u_{x,y}$ = "hour", $W = w$, $s = 0.001$, $c = 0.5$
- $T_{similarity} = 0.8$, $T_{attribute} = 3$, $T_{EI} = 5$, $\alpha = 0.5$, *Cross-match* = TRUE , *Fuzzy-match* = TRUE
- $N = 100$, w_{normal}, $w_{Rx} = 0.01$
- $\varsigma = 0.9$, β = TRUE, $\eta = 120$, $\lambda = 1$
- *DC* = "", *DE* = ""

In previous work, we listed and discussed another 54 experiments with sets *a* to *c* of 18 each (these 3 sets have different *W*). There are significant differences between the current and previous work. Generally, the current experiments are more realistic and more diverse. For example, instead of using separate data sets of "month", now the data sets are of yearly "quarter". Here *d2* is compared to *a1* (both are baseline experiments):

- Use of entire data set versus half; *W* in *d2* is the size of *W* in *a1*; $u_{x,y}$ = "hour" instead of "day" or fixed size example-based extraction.
- Reduction of T_{EI} from 10 to 5; of α from 0.8 to 0.5.
- Introduction of *c*, λ, *DC*, and *DE*.

The purpose of the experiments and rationale for parameter values are explained below. In Table 3, the columns list the experiment, the parameter to be changed in the baseline experiment *d2*, and followed by the corresponding new parameter value. For example, *d1* uses $W = w/2$, *e3* uses *DC* = *both/W*, and *f6* uses *DE* = ξ*(both/W, input/c, output/*λ*)*.

The *d* experiments test three different parameter values in each of the two parameters. This is to ensure that we choose the appropriate PoC values for *e* experiments. Actual *W* values are not listed due to confidentiality reasons. The *e* experiments test three SoA policies using single PoC strategy on the two parameters. This is to determine the appropriate single PoC strategies for *f* experiments. All the PoC values are shown (see Table 4). The *f* experiments test six different multiple PoCs strategies. Each multiple PoCs strategy has two or three single PoC strategies with the most common being *both/W*.

Table 3: 13 months data with experimental sets of *d* to *f*

***d* Exp. No(s).**	**Parameter Value**	**Value 1**	**Value 2**	**Value 3**
d1, d2, d3	*W*	*w/2*	*w*	*2w*
d4, d5, d6	$T_{attribute}$	*4*	*5*	*6*
***e* Exp. No(s).**	**Single PoC Strategy**	**Input**	**Output**	**Both**
e1, e2, e3	*DC*	*input/W*	*output/W*	*both/W*
e4, e5, e6	*DC*	*input/*$T_{attribute}$	*output/*$T_{attribute}$	*both/*$T_{attribute}$
***f* Exp. No.**	**Multiple PoCs Strategy**	**Single PoC Strategy 1**	**Single PoC Strategy 2**	**Single PoC Strategy 3**
f1	*DE*	*both/W*	*both/c*	-
f2	*DE*	*both/W*	*both/*λ	*both/c*
f3	*DE*	*both/W*	*both/*λ	-
f4	*DE*	*both/W*	*input/c*	-
f5	*DE*	*both/W*	*output/*λ	-
f6	*DE*	*both/W*	*input/c*	*output/*λ

In Table 4, the columns list the experiments from set *e*, the parameter to be changed, and the corresponding PoC value for each SoA (independent of SoA policy). For example, *e1*, *e2*, and *e3* all use $W = w/2$ if SoA = low, $W = w$ if SoA = medium, and $W = 2w$ if SoA = high.

Table 4: PoC value for each SoA

***e* Exp. No(s).**	**Parameter Value =**	**Low**	**Medium**	**High**
e1, e2, e3	*W*	*w/2*	*w*	*2w*
e4, e5, e6	$T_{attribute}$	*4*	*3*	*2*

In the following experiments, we are particularly interested in finding out which are the better static or dynamic/adaptive experiments at threshold 1 than baseline experiment *d2* (since only the most suspicious examples are investigated).

7.2 Results

To simplify description of results, we highlight the best static and dynamic experiment of each PoC compared to the baseline experiment *d2*. Interesting results are also highlighted.

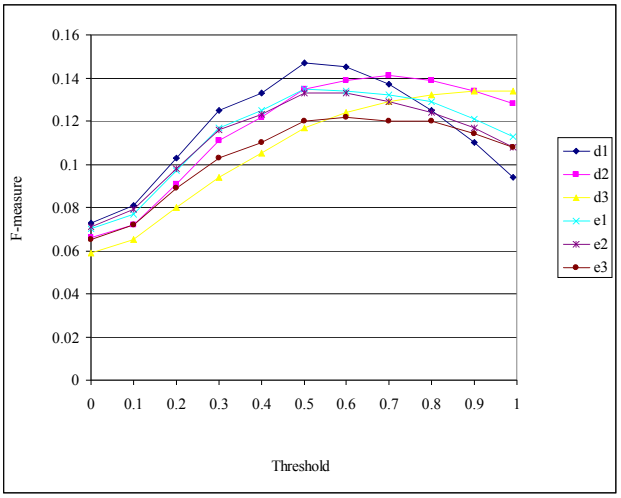

Figure 3: *FME'* of static versus dynamic *W*

We consider threshold of 1 to be most important. Therefore even though *d1* is generally higher for the lower thresholds (*d1* performs best at threshold 0.5, but is worst at threshold 1), the best static experiment is *d3* (better than baseline of 0.128) – the larger *W*, the better the *FME'*. The best dynamic experiment is *e1* (poorer than baseline) - with *W* as PoC, the input SoA policy works well.

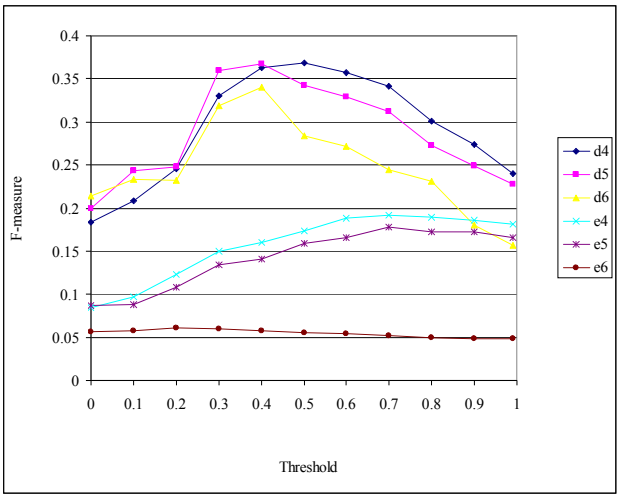

Figure 4: *FME'* of static versus dynamic $T_{attribute}$

The best static experiment is *d4* (better than baseline) - this can be deceiving because the increase in $T_{attribute}$ can lead to a corresponding increase in *FME'* but a likely decrease in *PDF'*. The best dynamic experiment is *e4* (better than baseline). The dynamic experiments have low *FMEs'* because of high false positive rates from using smaller $T_{attribute}$.

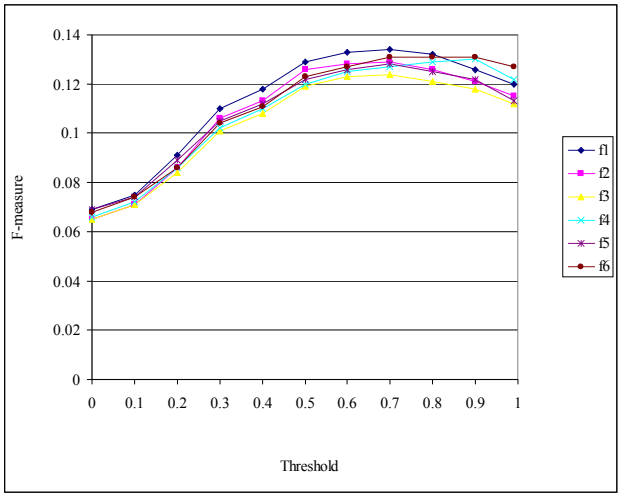

Figure 5: *FME'* of dynamic effect with multiple PoCs strategy

All experiments here have very similar *FME'* results. *f6* (similar to baseline), *f4*, and *f1* are slightly better than the rest.

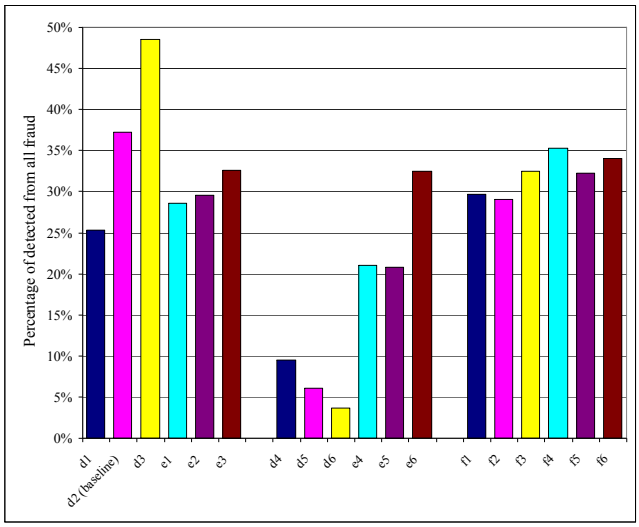

Figure 6: *PDF'* of experimental sets *d*, *e*, and *f*

Amongst the single PoC strategies, static *d3* (close to 50% of all frauds detected and is higher than baseline of 37%) and dynamic *e6* have the highest *PDF'*. Amongst the multiple PoC strategies, *f4* has the highest *PDF'* (marginally lower than baseline). There are a few interesting observations with regard to *PDF'*:

- "Both SoA policies" tend to detect more frauds than "Input and Output SoA policies".
- All single PoC strategies and multiple PoCs strategies detect lesser frauds than baseline.
- High $T_{attribute}$ detect very little frauds.

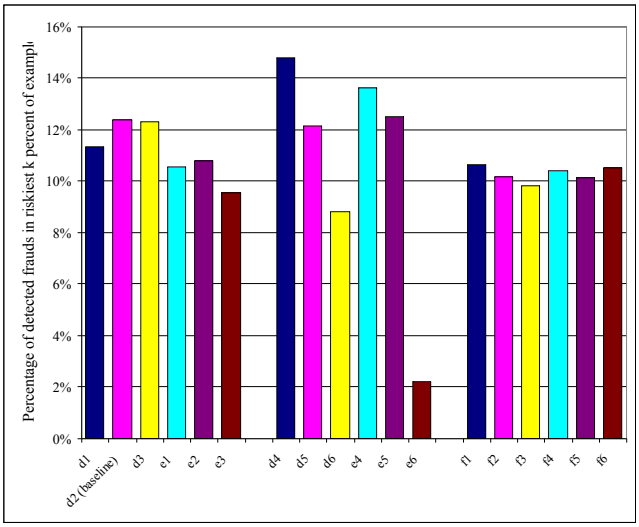

Figure 7: ***NTOP1*** of experimental sets *d*, *e*, and *f*

The *NTOP1* results are like a summarised form of *FMEs'* at threshold 1. But there are slight differences in results. The *NTOP1* sample is smaller than the *FMEs'* (at threshold 1) sample. Basically, it shows that:

- "Both SoA policies" has lower *NTOP1* than "Input and Output SoA policies".
- $T_{attribute}$ has the highest and lowest *NTOP1*.

8. DISCUSSION AND LIMITATIONS

From current experiments *d*, there are significant improvements from previous experiments *a* [9]: Current *FME'* peaks at around threshold 0.5 instead of 0.2. On average, current *FME'* is 0.02 higher, *PDF'* is at least twice more, and *NTOP1* is about 3% higher than previous work.

In all experiments, the processing time per example is less than 1 second using code written in Visual Basic .NET. Roughly, it takes around 1 week for each experiment on a single Windows XP workstation (3.0 GHz CPU with 1 Gb RAM). If the CASS algorithm were to be implemented as a real-world detection system on the same machine, it can search a much larger *W*, probably ten- to fifty-fold more identity examples.

In experiments, almost all *FME*s' (>0.2) here are very low but results are deceivingly poor. **In practice**, this will be much higher for a few good reasons. For privacy and confidentiality reasons, some important attributes have been anonymised from identity strings to un-identifiable numbers. This means no fuzzy matching can be performed on them and fraudsters are known to make slight string variations to these de-identified attributes. Second, many of the class labels were still not known at the time when this data set was constructed. Hence, our performance measures evaluated predictions based on significantly smaller numbers of positive class labels. Third, there is extreme class imbalance which will lead to a large number of false positives. Fourth, several million examples is small relative to the hundreds of millions across multiple databases.

The following questions can help clarify how adaptive aspects can help to improve communal detection:

With more parameters and frequent parameter adjustment, is overfitting a possibility? No, because adaptive communal detection is not a statistical model. It searches different identity crime profiles with two additional new parameters *DC* and *DE*.

Were training and evaluation data properly segregated? No, each example is constantly "trained" by comparing against an example-based window and evaluated with the class label.

What is the role of the adaptive component? At the end of step 3 of the fraudsters' strategic attack cycle, fraudsters do learn about the detection system's parameter settings and will apply new styles at step 1 accordingly. Here, by regularly tuning some PoCs during higher SoA cancels out the fraudsters' advantageous knowledge. The systematic change in parameter settings can happen as fraud styles are at various stages of the cycle.

What is the effect on the fraudsters' strategic attack cycle? Step 1 is more difficult as adaptive parameters will increase the fraudsters' effort/cost on synthetic identity fraud examples: new styles must now have higher volume and be more creative to probe the new PoC values. It is now harder to determine our SoA policies and PoCs, and gauge our T_{input}, Ωs, and δs. Also, step 3 can be cut short as adaptive parameters can decrease their success rate: accepted styles can be detected more rapidly with the new PoC values. **In theory**, with the assumption of persistent adversaries, we argue that dynamic parameters with whitelists have more durability of predictive power than static ones if PoCs are chosen carefully. In this paper, we showed parameter *W* to be a better PoC than $T_{attribute}$ as part of the multiple PoCs strategy.

Did multiple performance metrics show that dynamic f experiments yield better results than static d experiments? Probably yes, in across all evaluation metrics, the multiple PoCs strategy *f* experimental results are more stable and are not significantly poorer than baseline. **In experiments**, we tested our concepts with no adversaries to actually respond to our counter-measures. **In practice**, however, most real-world detection systems have slower manual intervention to tune parameters and yet adversaries constantly try to know and respond to these parameters. We conclude that with automatic "surprising actions", our multiple PoCs strategy will have better results than static parameters with fully automated detection systems.

9. CONCLUSION AND FUTURE WORK

As the content of identity crime in credit applications perpetually counters the detection system to produce false negatives, the main technical contribution here is to present an adaptive version of the CASS algorithm. At pre-defined time intervals and by measuring current input size and previous output suspiciousness, CASS adaptively changes the appropriate parameter setting to trade-off efficiency/speed and effectiveness/security. Each parameter setting consists of many parameters to search for similar identities, and each chosen dynamic parameter will have at least two possible parameter values to fine-tune that search. In other words, the algorithm changes over time to improve performance.

Short-term future work will involve deeper analysis into class label and class imbalance issues of the important experiments detailed in this paper. Longer-term future work will entail improvements on the attribute-oriented Spike Analysis Suspicion

Scoring (SASS) algorithm [10] which is particularly useful in the scenario where adversaries re-use only 1 or 2 important attributes and is not being detected by example-oriented CASS.

10. ACKNOWLEDGMENTS

Thanks to ARC's financial support under Linkage Grant Number LP0454077; tutorial speakers at EII06 School, Peter Christen, and anonymous reviewers for helpful comments.

11. REFERENCES

[1] M. Bilenko and R. Mooney. "Adaptive Duplicate Detection using Learnable String Similarity Measures", *Proc. of SIGKDD03*, 2003, pp39-48.

[2] R. Caruana and A. Niculescu-Mizil, "Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria", *Proc. of SIGKDD04*, 2004, pp69-78.

[3] L. Chen and G. Agrawal, "Supporting Self-Adaptation in Streaming Data Mining Applications", *Proc. of IPDPS05*, 2006, pp1-10.

[4] P. Christen and K. Goiser, "Quality and Complexity Measures for Data Linkage and Deduplication", In F. Guillet and H. Hamilton (eds.), *Quality Measures in Data Mining, Studies in Computational Intelligence*, Springer, 2006.

[5] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, Q. Zhang P. Wagle, and H. Hinton, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks", *Proc. of the 7th USENIX*, 1998, pp63-78.

[6] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial Classification", *Proc. of SIGKDD04*, 2004, pp99-108.

[7] J. Kleinberg, "Temporal Dynamics of On-Line Information Streams", In M. Garofalakis, J. Gehrke, and R. Rastogi (eds.), *Data Stream Management: Processing High-Speed Data Streams*, Springer, 2005.

[8] D. Lowd and C. Meek, "Adversarial Learning", *Proc. of SIGKDD05*, 2005, pp641-647.

[9] C. Phua, R. Gayler, K. Smith-Miles, and V. Lee, "Communal Detection of Implicit Personal Identity Streams", *Proc. of ICDM06 Workshop on Mining Evolving and Streaming Data*, 2006.

[10] C. Phua, V. Lee, R. Gayler, and K. Smith-Miles, "Temporal Representation in Spike Detection of Sparse Personal Identity Streams", *Proc. of PAKDD06 Workshop on Intelligence and Security Informatics*, 2006.

[11] K. Ramamohanarao and J. Bailey, "Discovery of Emerging Patterns and Their Use in Classification" *Proc. of Australian Conference on Artificial Intelligence*, 2003, pp1-12.

[12] W. Winkler, "The State of Record Linkage and Current Research Problems", *Statistics of Income Division, Internal Revenue Service Publication* R99/04, 1999.

[13] K. Yoshida, F. Adachi, T. Washio, H. Motoda, T. Homma, A. Nakashima, H. Fujikawa, and K.Yamazaki, "Density-Based Spam Detector", *Proc. of SIGKDD04*, 2004, pp486-493.

APPENDIX

A. Data Stream Extraction Parameters

g_x is a current mini-discrete data stream being processed and contains a number of micro-discrete streams u. It is representative of a single month's, fortnight's, or week's data.

$u_{x,y}$ is a current micro-discrete data stream being processed and contains a number of identity examples v. It was originally example-based but has since been extended to the more realistic time-based data extraction. It is representative of a single day's, hour's, minute's, or second's data.

s is the suspicion threshold used to filter examples in u_s with a default of 0.

B. Pair-wise Example Comparison Parameters

$T_{similarity}$ is the similarity threshold for fuzzy matching of identity examples, using *Jaro-Winkler* [12], with defaults of either 0.8 (80% similar match) or 1 (exact match) on all attributes.

T_{EI} is the incoming link threshold for the acceptable number of incoming links to v_j, with a default of 5.

α (alpha) is the exponential smoothing factor needed to gradually discount the effects of average previous suspicion scores, with a default of 0.5.

Cross-match involves matching across selected identity attributes with $T_{similarity} = 1$.

Fuzzy-match involves matching within selected identity attributes with $T_{similarity} = 0.8$.

C. Whitelist Construction Parameters

N is the number of most common link-types/relationship-types ranked in descending order by link volume.

w_{normal} is the weight assignment for each link-type and weights are sorted in ascending order. A higher link-type rank means a smaller weight to reduce the single-link communal score.

w_{R1} is the starting weight for the highest ranked link-type.

D. Data Quality Improvement Parameters

ς (sigma) is the sampling factor for keeping all the minority class and randomly sampling the majority class, with a default of 0.9 (remove 90% of the majority class).

β (beta) is the Boolean blocking factor for filtering v_j immediately when the blocking key(s) (attribute or part-of-an-attribute to filter) is not the same as v_i, with a default of TRUE.

η (eta) is the exact duplicate factor for removing links of exact duplicates (where all available attributes match) from the same subscriber within a time difference of minutes, with a default of 120.