Temporal Representation in Spike Detection of Sparse Personal Identity Streams

Clifton Phua[1], Vincent Lee[1], Ross Gayler[2], and Kate Smith[1]

[1] Monash University, Clayton School of Information Technology, Melbourne
{clifton.phua, vincent.lee, kate.smith}@infotech.monash.edu.au
[2] Baycorp Advantage, Melbourne
r.gayler@mbox.com.au

Abstract. Identity crime has increased enormously over the recent years. Spike detection is important because it highlights sudden and sharp rises in intensity relative to the current identity attribute value (which can be indicative of abuse). This paper proposes the new spike analysis framework for monitoring sparse personal identity streams. For each identity example, it detects spikes in single attribute values and integrates multiple spikes from different attributes to produce a numeric suspicion score. Although only temporal representation is examined here, experimental results on synthetic and real credit applications reveal some conditions on which the framework will perform well.

1 Introduction

In security-related informatics, personal identity examples with sparse attributes arrive in streams. Sparse attributes usually consist of string occurrences and identifiers with an enormous number of possible values which can occur at widely spaced intervals (such as personal names and telephone numbers); in contrast with dense attributes which have a finite number of attribute values, and therefore occur more frequently (such as street numbers and postcodes). Each structured example is made up of a mixture of both sparse and dense attributes, and arrives rapidly and continuously into the data repository. To detect events of interest in streams, stream mining systems should be highly automated, scalable, and timely [6].

Some related work for analysing sparse attributes can be found in document and bio-surveillance streams. Kleinberg [10; 9] surveys threshold-, state-, and trend-based stream mining techniques used in topic detection and tracking. [16] surveys techniques for finding anomalies in time for disease outbreaks, and [7] use time series analysis techniques for a simulated bio-terrorism attack. Two significant stream processing systems are STREAM [1] which details the Continuous Query Language (CQL) and AURORA [2] which has been applied to financial services, highway toll billing, battalion and environmental monitoring.

This paper's main technical contributions are in the integration of spike detection and stream mining within a framework (Section 2.1): representation of time with respect to each attribute value (Section 2.2), use of EWMA (Exponentially Weighted Moving Average) in single attribute value spikes (Section 2.3), and integration of multiple attribute value spikes to score each example (Section 2.4). This paper

H. Chen et al. (Eds.): WISI 2006, LNCS 3917, pp. 115 – 126, 2006.

is novel within the credit application fraud detection context because it finds denser identity attribute values (which can be highly indicative of identity crime (Section 3.1) in a principled fashion. It describes the synthetic and real credit application data sets (Section 3.2), experimental design and evaluation measures (Section 3.3). Section 3.4 discusses the insights of using different temporal representations. Section 4 concludes and highlights likely future work.

2 Spike Analysis Framework

2.1 Overview

In Figure 1(a), the input data is extracted from the entire database and consists of examples sorted in descending arrival date and time order (a more recent example is placed above an older one); and these examples are either previously scored (exists within the window), the ones which has to be scored in arrival order (the data stream),

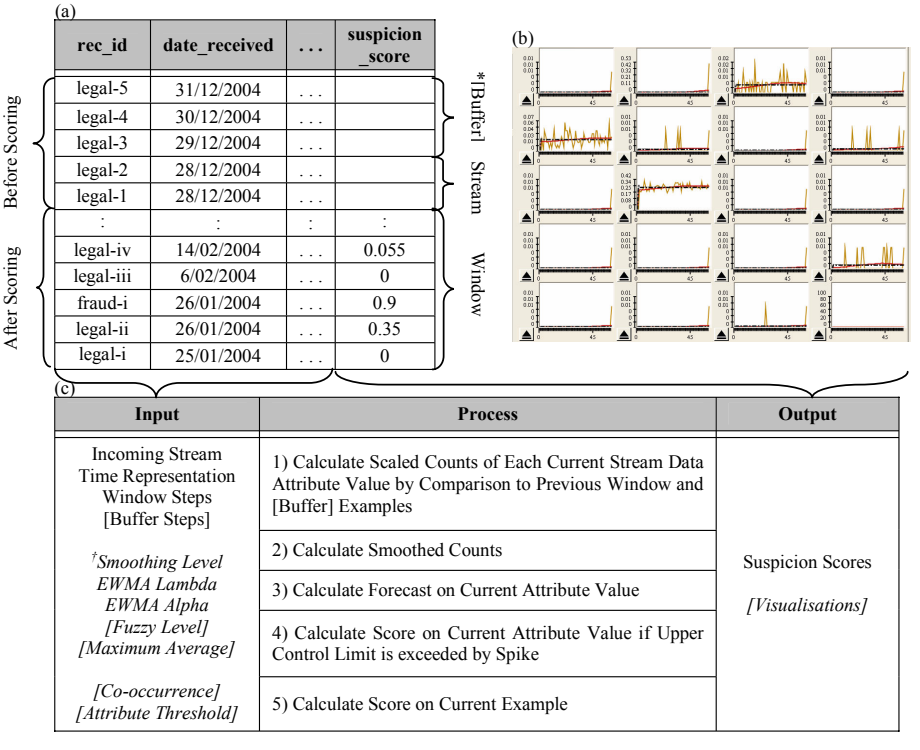

Fig. 1. Overview of input, process, and output sections of the spike analysis framework * square brackets indicate that the enclosed parameter is optional † *italic-typed parameters are not tested in this paper's experiments*

or the ones to be scored later (exists within the buffer). The buffer examples are always more recent than the stream examples, which are in turn, more recent than the window examples. For illustration, suppose the temporal representation is daily rate, window steps is one month and with no buffer, the two stream examples "legal-1" and "legal-2" are scored using prior examples which arrive between 29/11/2004 and 28/12/2004. The last column in Figure 1(a) contains the suspicion score output and the multiple time series plots in Figure 1(b) are the visual outputs for an example.

In Figure 1(c)'s "Input" column, the first four parameters determine how much input data to continuously extract using Structured Query Language (SQL) queries from the database to process an incoming stream; the next seven parameters can be used to fine-tune the processing capabilities of the spike detection system. For brevity, this paper presents only the results of varying the first four parameters (incoming stream, time representation, window and buffer steps) by holding the next seven parameters constant. Similarly for the "Output" column, this paper evaluates the suspicion scores without the assessment of visualisations. The "Process" column has five main calculation/algorithmic steps to generate a suspicion score for each example.

For concreteness, let Y represent a continuous data stream with an ordered set of $\{...,y_{j-2},y_{j-1},y_j,y_{j+1},y_{j+2},...\}$ discrete streams. Let y_j represent a discrete data stream with an ordered set of $\{d_{j,n},d_{j,n+1},d_{j,n+2},...,d_{j,n+p}\}$ examples to be processed online. Processing is carried out on p examples and from the viewpoint of the most recent stream y_j's oldest unscored example $d_{j,n}$ to the newest unscored example $d_{j,n+p}$. For simplicity, the subscript j is omitted to concentrate on processing one discrete stream and one example at a time. Let each d_n contain M number of attributes, where $d_n = \{a_{n,1},a_{n,2},...,a_{n,M}\}$. Each current attribute value $a_{n,i}$ is compared to previous ones $a_{n-1,i},a_{n-2,i},...,a_{n-W,i}$ in search of the same value ($a_{x,i} = a_{n,i}$) or approximate value ($a_{x,i} \approx a_{n,i}$), where W is the window size of extracted previous examples in the user-specified temporal representation. *Fuzzy level* is an optional parameter which utilises Levenshtein edit distance where at both extremes, 1 is an exact match and 0 is a complete mismatch. By default, *fuzzy level* is set to 1. W is needed to score p and has to be sufficiently large to maximise the detection capability of processing d_n.

In addition to comparing each $a_{n,i}$ to previous attribute values, $a_{n,i}$ can also be compared to subsequent ones $a_{n+1,i},a_{n+2,i},...,a_{n+B,i}$, where B is the buffer size of extracted later examples in the user-specified temporal representation. B has the potential to reduce the number of false positives [8] and has to be small to minimise the wait-time delay of processing d_n. Therefore, the task is to examine the rate where past and future examples had the same or approximate attribute value as the present one as a function of relative time.

2.2 Temporal Representation

Temporal representation is the choice of the x-axis in the time series for $a_{n,i}$. $W = t * k$ and $B = 0.1t * k$ where W and B are split into user-defined number of steps t and into step size(s) k. Let $W_{user-defined-rate}$, $B_{user-defined-rate}$ represent the case where the constant k is user-defined; and $W_{stream-rate}$, $B_{stream-rate}$ where the constant k is automatically determined by the overall daily stream rate. k can also be automatically represented by finer-grained temporal measurements (such as seconds, minutes, and hours) or coarser-grained ones (such as days, weeks, months, and years). Let $W_{daily-rate}$ and $W_{weekly-rate}$ (each week starts on a Sunday) represent the case where the variable k is the number of examples in a specific day and week respectively. For clarity, this paper omits results from other useful temporal representations such as day-of-week, week-of-month, season, school holidays, and public holidays.

2.3 Spike Detection of Single Attribute Value

Sparsity. As a rule-of-thumb, our definition of sparsity in an attribute is: if more than 80% of its values are not missing, and there are more than 100 distinct values, and more than 50% of these distinct ones are unique (occur only once). In this spike

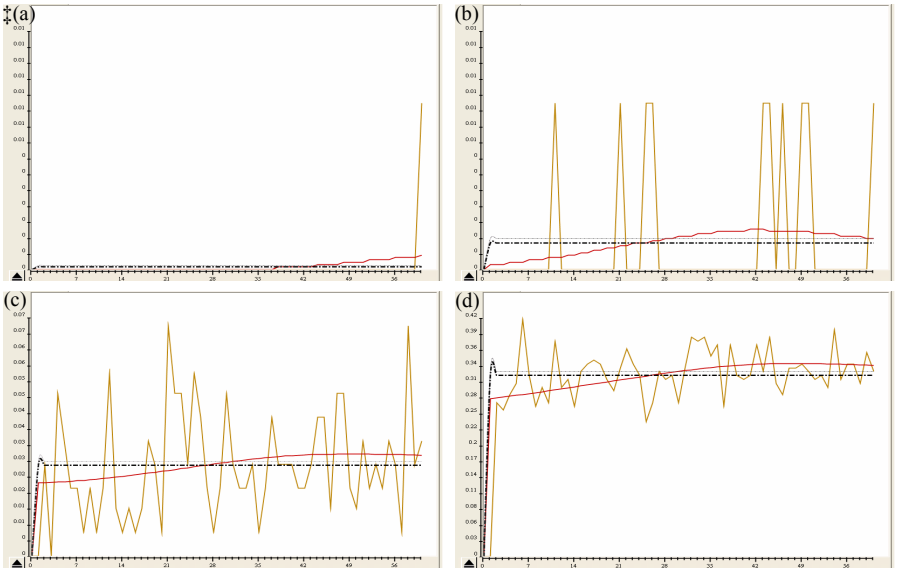

Fig. 2. Monitoring of $a_{n,i}$ in d_n with $W_{stream-rate} = t * k = 60 * 147 = 8820$ and $B_{stream-rate} = 0$: (a) Very sparse, (b) Sparse, (c) Dense, (d) Very dense ‡ y-axis is not able to display more than 2 decimal places

detection framework, assuming that $a_{n,i}$ is not a missing value, it is sparse if it occurs less than $0.01(W+B)$ times. For example in Figure 2, number of occurrences of "pan" in *last name* 2(a) - once in d_n, of "business systems" in *employer name* 2(b) - less than a hundred times, of "3" in *street number* 2(c) - a few hundred times, and of "vic" in *state* 2(d) - a few thousand times.

Maximum average is an optional parameter which is used to restrict dense attribute values where at both extremes, 1 includes and 0 excludes all attribute values. By default, *maximum average* is set to 1.

Scaling. It is needed to reduce variability of weekly and seasonal effects in raw counts, as well as to compare across different temporal representations. Rates, or the proportion of all attribute values in W and B having the specified attribute value $a_{n,i}$, are less sensitive to overall volume effects. For example in Figure 2, the orange (light solid) line signifies the scaled counts.

Smoothing. It is required to remove rare and chaotic effects in order to compare the current rate to the assumed constant background rate. Discrete Wavelet Transform (DWT) allows the decomposition of a time series into different resolutions using a mother wavelet. DWT has three main advantages over its other alternatives: efficiency, suitable for use with non-stationary data, and quantifies location in time and frequency [12]. For example in Figure 2, the red (dark solid) line is the background rate, and it is derived from applying the highest DWT *smoothing level* onto the scaled counts to get the least noisy curve.

Spike Detection. Linear forecasting on the smoothed time series is done by Exponentially Weighted Moving Average (EWMA) [15]. EWMA is a statistical quality control technique which specifies how much the current prediction is influenced by past observations. Given a measurement X_t at time step t, the definition of EWMA statistic Z_t is:

$$Z_t = \lambda X_t + (1-\lambda) Z_{t-1} \text{ for } t = 1,2,...,n \tag{1}$$

where Z_0 is the desired process mean μ_0. n is the number of observations to be monitored including μ_0. $0 < \lambda \leq 1$ is the weighting factor where higher values give more weight to current observation, with $\lambda = 1$ referring to the Shewhart control chart.

The estimated variance of Z_t for a large t is approximately:

$$\sigma_{z_i}^2 = \left(\frac{\lambda}{2-\lambda}\right)\sigma^2 \tag{2}$$

With the variance, the UCL applied from (2) is:

$$UCL = \mu_0 + \alpha\sigma_{z_i} \tag{3}$$

Advantages of using EWMA compared to many other time surveillance techniques include low implementation and computational complexity [16], and EWMA is a non-parametric monitoring algorithm as it makes no assumption about the distribution that X_t is drawn from [11]. For example, the dark dotted line is the average and the light dotted line is the UCL, and is obtained from setting $\lambda = 0.8$ and $\alpha = 0.0027$ (3σ).

If buffering is not implemented ($B = 0$), the score for each current attribute value $a_{n,i}$ is determined by how much the red line exceeds the light dotted line:

$$S(a_{n,i}) = \begin{cases} w_t(a_{n,i}) - UCL & \text{if } W_t(a_{n,i}) > UCL \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $W_t(a_{n,i})$ is the actual smoothed count of the current time step t. For example in Figure 2 (b), the dark dotted line $W_t(a_{n,i})$ did not exceed the light dotted line UCL, therefore $S(a_{n,i}) = 0$.

If buffering is implemented ($B > 0$), the score for each $a_{n,i}$ is:

$$S(a_{n,i}) = \begin{cases} B_t(a_{n,i}) - UCL & \text{if } B_t(a_{n,i}) > W_t(a_{n,i}) > UCL \\ W_t(a_{n,i}) - UCL & \text{if } W_t(a_{n,i}) > B_t(a_{n,i}) > UCL \\ 2\sqrt{(W_t(a_{n,i}) - UCL)} & \text{if } W_t(a_{n,i}) > UCL > B_t(a_{n,i}) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $B_t(a_{n,i})$ is the averaged smoothed count of all time steps in the buffer.

2.4 Integration of Spikes from Multiple Attributes

The normalised score for each current example $d_n \in y_j$ is:

$$S(d_n) = \begin{cases} \dfrac{1}{\max\left[\sum_{i=1}^{M} S(a_{x,i})\right]} \left[\sum_{i=1}^{M} S(a_{n,i})\right] & \text{if } 1 > S(d_n) \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

where $a_{x,i} \in d_x \in y_{j-1}$ and $0 \le S(d_n) \le 1$.

Co-occurrence is an optional parameter which allows for the searching of 2 or 3 combinations of attribute values. By default, *co-occurrence* is set to 1. *Attribute threshold* is another optional parameter which restricts the scoring of an example when the number of spiking attribute values exceeds it. By default, *attribute threshold* is set to number of attributes.

In summary, the basic idea involves monitoring attribute value frequency over time, in particular if any attribute value spikes. Spiking refers to any attribute value which rises sharply in intensity within a short and more recent time period relative to the current example. Based on the number and intensity of the attribute value spikes, each current example is then given a numeric suspicion score between 0 and 1 relative to other examples in the previous stream.

3 Application Domain

3.1 Identity Crime

Identity crime, consisting of identity fraud and theft, is about taking advantage of fictitious identities and/or real unsuspecting individuals' personal information to hide the perpetrators' real identities for their illegal goals. It features in a myriad of other crimes such as terrorism and financial crime [13]. It is well-known that identity crime is the fasting growing crime in the developed world and its monetary cost is often estimated to be in billions of dollars annually.

Credit bureaus collect millions of enquiries relating to credit applications annually. In Australia, there are currently around half a million enquiries made per month. Each enquiry contains many identity attributes and it is easy for criminals to tamper the attributes with seemingly valid values. [5] explains that the visible patterns of organised identity crime rings constantly change and it is difficult to detect with standard identity verification tools. However, these organised fraudsters exhibit detectable link and temporal patterns throughout the scams. In particular, they seem to target attacks on specific organisations at specific times for short durations.

3.2 Data Description

Synthetic Data Set. It has around 20 attributes such as primary key label, date received, personal names, addresses, phone numbers, personal and employer details; and has 52,750 applications (9% are fraudulent) which span the entire year 2004. This data set has been specifically created with the assumption that fraudsters will purposely use these fraudulent attribute values regularly, occasionally, seasonally, and/or once-off. Also, identity attribute values are reused within a very short period of time. It has very few missing attribute values. The justification, generation, and details of this synthetic data can be found in [14; 4]. Experiments on this data set can reveal if the spike analysis framework can perform well on sparse identity streams evaluated by weeks and imbalanced classes. On average in Table 1, each weekly

Table 1. Synthetic data streams for experiments

Stream number	Week	Total applications	Percentage fraudulent
1	10/03/2004 to 16/03/2004	990	13%
2	17/03/2004 to 23/03/2004	1070	14%
3	24/03/2004 to 30/03/2004	1032	11%
4	31/03/2004 to 06/04/2004	1117	16%

stream consists of slightly more than 1,000 applications to be scored in March and April 2004, and percentage fraudulent is higher than the average due to injected seasonal frauds.

Real. In comparison, the real data set has very similar number and type of attributes to the synthetic one except that some attributes have been hashed for privacy and confidentiality reasons. In contrast, the real one has 27,546 applications (50% are fraudulent) which start from July 2004 to July 2005 (13 months). The fraud percentage is very high because the legitimate applications have been randomly under-sampled by selecting 1 in a few hundred applications which has not been flagged as fraudulent. It has a significant amount of missing attribute values. Experiments on this data set offer a rare glimpse into the temporal behaviour of identity fraudsters. It also can determine if the spike analysis framework can perform well with sparse identity streams evaluated by weeks and missing attribute values. On average in Table 2, each weekly stream consists of around 600 applications to be scored in January 2005.

Table 2. Real data streams for experiments

Stream number	Week	Total applications	Percentage fraudulent
1	06/01/2005 to 12/01/2005	730	50%
2	13/01/2005 to 19/01/2005	698	50%
3	20/01/2005 to 26/01/2005	584	50%
4	27/01/2005 to 02/02/2005	392	50%

3.3 Experimental Design and Evaluation Measures

The first step is to simulate a streaming environment where incoming credit applications can be continuously scanned for significant spikes in its attribute values. Applications in the window and buffer are extracted to process the incoming application stream. An alarm is raised on an application if the suspicion score is ranked high amongst the applications before it. The second step is to select only sparse attributes to monitor. In both the synthetic and real data, only the sparsest attributes have been chosen for the experiments (the actual attributes used in experiments cannot be revealed).

In Table 3, $W_{user-defined-rate}\ (t;k)$ is used to determine the effects of the same W but different t and k. $W_{stream-rate}$ is used to examine the outcomes of the constant overall daily stream rate k (147 for synthetic and 70 for real data) but steadily increasing t. $W_{daily-rate}$ and $W_{weekly-rate}$ are used to study the results of the variable k but steadily increasing t. $B_{user-defined-rate}$ and $B_{stream-rate}$ are used to learn about the consequences of buffering $0.1t$ with a user-defined k.

Table 3. Different temporal representations for experiments

$W_{user\text{-}defined\text{-}rate}$	$W_{stream\text{-}rate}$	$W_{daily\text{-}rate}$	$W_{weekly\text{-}rate}$	$B_{user\text{-}defined\text{-}rate}$	$B_{stream\text{-}rate}$
20;500	10	10	7	2;500	1;500
40;250	20	20	14	4;250	2;250
50;200	30	30	21	5;200	3;200
80;125	40	40	28	8;125	4;125
100;100	50	50	35	10;100	5;100
200;50	60	60	42	20;50	6;50

Other parameters are held constant: *Smoothing level* = 5 (maximum), *EWMA lambda* = 0.8, *EWMA alpha* = 0.0027 (3σ), *Fuzzy level* = 1, *Maximum average* = 1, *Co-occurrence* = 1, and *Attribute threshold* = 20 (number of attributes).

The non-threshold-based, diverse measures of performance implemented in [3] are:

- **NTOP10** is an ordering metric which should be **maximised**. What is the percentage of positives in the top 10% of ranked examples?
- **AUC** (Area Under the ROC Curve) is another ordering metric which should be **maximised**. What is the performance under all possible thresholds?
- **RMS** (Root Mean Squared Error) is a probability metric which should be **minimised**. How much is the deviation of predictions from true class labels?

3.4 Results and Discussion

- **3(a) and 3(b) have the highest and most similar results but 3(b) uses a much smaller *W*.** In contrast, 3(c) performed poorly and 3(d) very poorly.
- **Stream 4 has the best and stream 3 has the worst NTOP10 and AUC**; except in 3(d) where Stream 1 has the best and stream 4 has the poorest NTOP10 and AUC.

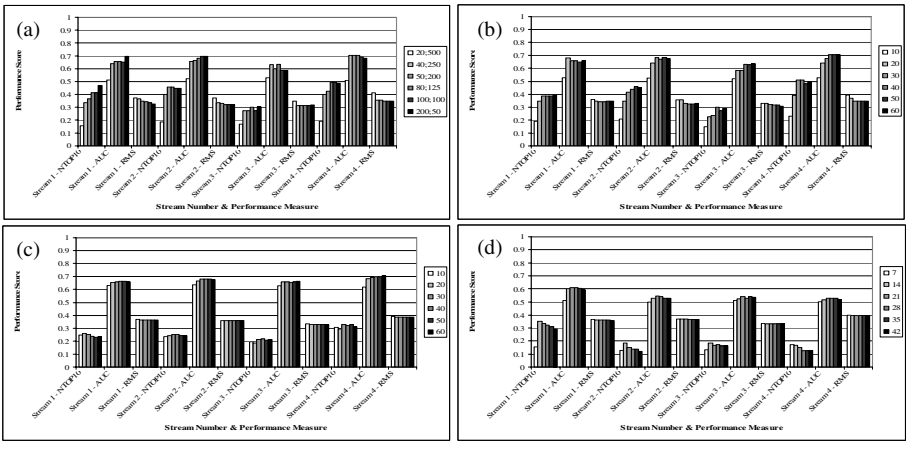

Fig. 3. Results on **synthetic data with no buffer** (a) user-defined rate, (b) stream rate, (c) daily rate, (d) weekly rate

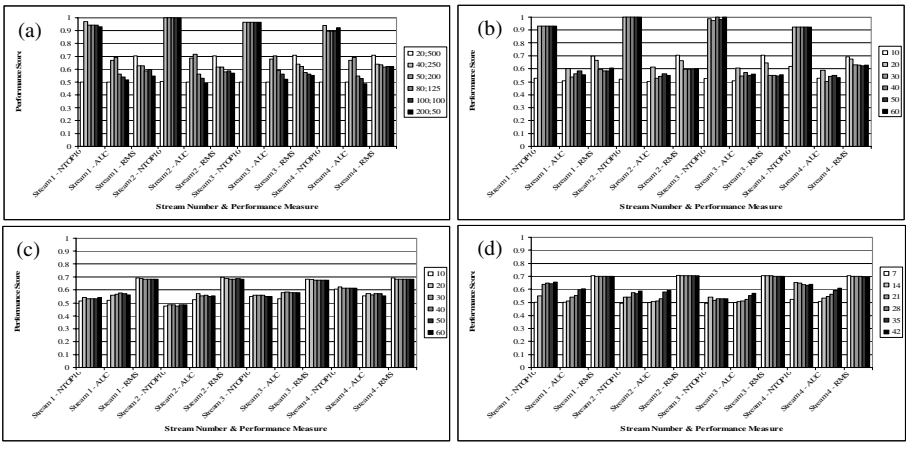

Fig. 4. Results on real data with no buffer (a) user-defined rate, (b) stream rate, (c) daily rate, (d) weekly rate

- **4(a) and 4(b) have the best results although 4(b) uses much smaller *W*. 4(a) is better than 4(b) in NTOP10 and AUC.** In contrast, both 4(c) and 4(d) perform significantly poorer in NTOP10.
- **Stream 2 has the best and stream 4 has the worst NTOP10 and AUC in 4(a) and 4(b).** Stream 4 has best and 2 have poorest NTOP10 and AUC in 4(c). Stream 1 and stream 4 has best; and stream 3 have poorest NTOP10 and AUC in 4(d).

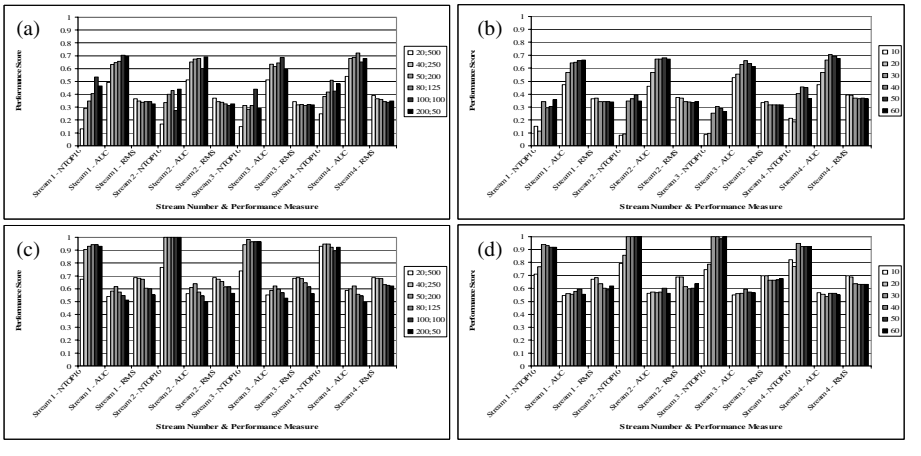

Fig. 5. Results on synthetic data with buffer (a) user-defined rate (b) stream rate, and on **real** data with buffer (c) user-defined rate (d) stream rate

- **5(a) has significantly better NTOP10s than 5(b), while AUC is basically similar. 5(c) is generally the same as 5(d), except *t*=20 of 5(d) is much lower than 5(c) for NTOP10 (Different from without buffer).**

- **Stream 4 has the best and stream 3 has the worst NTOP10 and AUC in 5(a) and 5(b). Stream 2 has the best and stream 4 has the poorest NTOP10 and AUC in 5(c) and 5(d) (Same as without buffer).**

Lessons learnt from experimenting with temporal representations are listed below:

- There is no optimal t, but the **optimal k** measured by AUC and RMS, is approximately the **overall daily stream rate** for all synthetic and real data streams.
- Overall, **as t gets larger, RMS decreases correspondingly**.
- The similar results of user-defined and stream rate windows show that **window size W does not have to be large**.
- Intuitively, user-defined and stream rate window steps produce much better in all 3 measures than daily and weekly window steps because the former two always process d_n on prior examples in a fixed step size k. On the other hand, **daily and weekly window steps can be overly sensitive** when processing d_n on few examples which have the same attribute values early in the start of day and week.
- In comparison with **synthetic data**, **NTOP10 for real data is significantly higher** (due to balanced binary classes), but **AUC for real data is lower** (most likely due to missing values in real data).
- **Buffers** do cause more irregular variations in results as t gets larger and k gets smaller, but on the whole, it **makes no overall improvement**. There are more significant differences between the results of synthetic data with buffer and without buffer (due to imbalanced classes).
- The **more sparse the attribute(s), the better the results**.
- On average, **fraudulent applications do exhibit more spiky behaviour than legitimate ones.** Measured by NTOP10 (top 10% of ranked examples), the spike detection framework results in about **two-fold increase in detection rate** for both synthetic data (>0.3) and real data (>0.9), compared to random selection of synthetic data (0.14) and real data (0.5) respectively.
- The **computation time is small** - about 1 minute for 1 attribute to score 1000 streaming applications on a normal workstation.

4 Conclusion

This paper has reported on a stream-in and score-out system for sparse personal identifiers. It has described the spike analysis framework and taken a step toward building a highly automated, scalable, and timely detection tool on credit application data. Using synthetic and real data, experiments with different temporal representations showed that window step size is best determined by the overall daily stream rate, window size does not have to be very large, buffers have not been proven to be useful yet, and the best detection rates are obtained with the sparsest attributes. Future extensions on this work will include testing out the other seven parameters on a few million real credit applications.

Acknowledgements

This research is financially supported by the Australian Research Council under Linkage Grant Number LP0454077. Ethics approval is granted by Monash SCERH under Project Number 2005/694ED. The real credit application data is provided by Baycorp Advantage. Special thanks to developers of FEBRL data set generator, Ironic Chart Plotter, and PERF evaluation software.

References

1. Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J. and Widom, J.: STREAM: The Stanford Stream Data Manager Demonstration description - short overview of system status and plans. Proceedings of SIGMOD03, 2003.
2. Balakrishnan, H., Balazinska, M., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Galvez, E., Salz, J., Stonebraker, M., Tatbul, N., Tibbetts, R., Zdonik, S.: Retrospective on Aurora. VLDB Journal, 13(**4**), 2004, pp370-383.
3. Caruana, R. and Niculescu-Mizil, A.: Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria. Proceedings of SIGKDD04, 2004, pp69-78.
4. Christen, P.: Probabilistic Data Generation for Deduplication and Data Linkage. Proceedings of IDEAL05, 2005.
5. Cook, M.: Fraud and ID Theft – The Lowdown on Fraud Rings. In Collections and Credit Risk **10**, 2005.
6. Fawcett, T. and Provost, F.: Activity Monitoring: Noticing Interesting Changes in Behaviour. Proceedings of SIGKDD99, 1999, pp53-62.
7. Goldenberg, A., Shmueli, G. and Caruana, R.: Using Grocery Sales Data for the Detection of Bio-Terrorist Attacks. In Statistical Medicine, Submitted, 2002.
8. Keogh, E., Chu, S., Hart, D. and Pazzani, M.: Segmenting Time Series: A Survey and Novel Approach. In Last, M., Kandel, A. and Horst, B. (eds.): Data Mining in Time Series Databases, World Scientific, 2004.
9. Kleinberg, J.: Bursty and Hierarchical Structure in Streams. Proceedings of SIGKDD02, 2002, pp91-101.
10. Kleinberg, J.: Temporal Dynamics of On-Line Information Streams. In Garofalakis, M., . Gehrke, J. and Rastogi, R. (eds.): Data Stream Management: Processing High-Speed Data Streams, Springer, 2005.
11. Montgomery, D.: Introduction to Statistical Quality Control. John Wiley and Sons Inc, 4th edition.
12. Percival, D. and Walden, A.: Wavelet Methods for Time Series Analysis (WMTSA). Cambridge University Press, 2000.
13. Phua, C., Lee, V., Gayler, R. and Smith, K.: A Comprehensive Survey of Data Mining-based Fraud Detection Research. Artificial Intelligence Review, submitted.
14. Phua, C., Gayler, R., Lee, V. and Smith, K.: On the Approximate Communal Fraud Scoring of Credit Applications. Proceedings of Credit Scoring and Credit Control, 2005.
15. Roberts, S.: Control-Charts-Tests based on Geometric Moving Averages. In Technometrics, **1**, pp239-250.
16. Wong, W.: Data Mining for Early Disease Outbreak Detection. PhD Thesis, Carnegie Mellon University, 2004.