Data mining a high-speed bursty stream
On a limited buffer in pseudo-stationary states
(IDDAC’S’2003)

Sam Sengupta¹, Bruno Andriamanalimanana¹, Stuart W. Card²
Kaustav Das¹, Rachita Sharma¹, Anitha Gunasekaran¹

¹State University of NewYork Institute of Technology
Utica NY 13504-3050
Email: {sengupta, fbra, dask, sharmar, gunaseka}@sunyit.edu
²Critical Technologies Inc
Suite 400 Technology Center, 4th Floor 1001 Broad St, Utica NY 13501
Email: stu@critical.com

Abstract: Mining a high speed bursty data stream is always a challenge on a limited size buffer. Often a relatively cheaper AMS (Anytime Mining Solution) approach may be the only plausible scheme one could rely on at times. Mining task becomes enormously complicated when the first-level buffer has to host several dependent streams. This becomes worse when incoming data streams take time to settle down in their respective steady states. A buffer sharing and capture models are indicated for some simple situations involving multiple streams. These models could be extended to generalize a linear buffer model to a hierarchical model.

Keywords: Data mining, Multiple stream mining, Buffer sharing, Buffer acquisition, Anytime states

1. Introduction

Online learning, by now, is more or less an established frontier in knowledge engineering realm. In the usual mode, captured incoming data at a buffer is examined on the fly to discover appropriate relations and as these are realized they are indexed and stored in a database for a future use. In this arrangement, the usual problems are time and space: Time to identify relations from the arrested sampled data before they are flashed out by the incoming data stream itself, and the limited buffer space that we’d use to capture sampling data in the first place.

In this arrangement, it is assumed that the streaming data S is in its stationary state and will remain in it for a foreseeable future – at least until we are cognizant of its movement. Normally we are interested in a system while it is in a stationary state during a period of observation; and therefore, if x and y are states of interest of the system S (the stream) inducing a relation R like

\[ R_\rho(x, y, t) \subseteq X \times Y \times T \]  

with a degree of belief or confidence pegged at some \( \rho \geq \rho_{\text{threshold}} \) we assume that in some sense the relation would converge asymptotically to a time-independent relationship like

\[ R_\rho(x, y, t) \rightarrow R_\rho(x_\infty, y_\infty) \]  

if we observe it for a sufficiently long enough time to deduce the system in this manner. Unfortunately, interesting systems are often anything but this straightforward. Usually a system may have more than one set of equilibrium points, and a system, under some situations, may often migrate from one stable state to another if sufficiently induced to do so. Therefore, it is quite
possible that during an observation period a data stream may not always manifest residency in a
stable phase point; if it moves away from its stable point the buffer must be flashed out to ensure
that the currently captured sample set doesn’t contain any data pertaining to previous stable set.
Failing this would contaminate the current state reading to the extent the buffer contains residue
from the previous state.

Also, if one can wait sufficiently long enough with a system approaching its asymptotic
equilibrium one obtains its relations (if any) arbitrarily close to what it ought to be at that state.
However, one may not get the time. The residence time of a system at a specific equilibrium point
may not be long enough; it might, after a while, leave the current state to migrate to another stable
state. Note that, in this case, we are deliberately moving away from the standard notion of
equilibrium. We contend that in a realistic situation data streams entering into a router buffer from
the Internet, for example, may jump from states to states without leaving much time to compute its
states in the limit \( t \to \infty \). Even though the system would not be pinned down at any particular
state for a sufficiently long time, we could, for many practical purposes, compute its approximate
states and use them when needed. Therefore, an anytime solution \( x(t) \) may be the only avenue left
for us when time to compute the stable state is not there or not enough samples are available in
support of the current state. In this case, perfect solutions are not sought (a) either they aren’t
there, or (b) the approximate solutions are sufficiently robust and reasonable in terms of action
they would invite.

Given this, a data mining process could be articulated in the following minimalist way. Given a
buffer to capture an incoming data stream \( S \), store only its relevant states and their statistics (e.g.
\( x \in S, \mu, \sigma, ... \)) so long the system isn’t settled in its approaching equilibrium (i.e. \( \frac{dx}{dt} \neq 0 \)).

Once the system is found approaching an equilibrium, the process could doc the anytime states at
a second level of buffer, flush the collecting buffer in order to use it to collect new states. For a
system settled in a stable equilibrium state, buffer requirement for incoming data stream is often
minimum. In this case, using the previous example, one might keep only appropriate cumulative
counters and recently received data so that on demand one can compute the state aggregates like
\( (\mu, \sigma) \) etc. Note that buffer would be required during the time when the system is still roaming
around or for a steady state. So, theoretically, a finite buffer could be logically shared by multiple
streams while they move in and out of stable states as long as we take advantage of their residency
in stable states. This is possible when incoming streams have bursty profiles. Sharing a finite
buffer among a multiple set of streams for data mining under different situations is highlighted.

2. Buffer Regeneration Policy

Consider a stream \( S \) at a steady state. If \( \zeta \) is a target state monitored at system equilibrium, i.e.
\( \exists \zeta \in S, \dot{\zeta} = 0 \) during its observation period \( \Delta T \) we note that \( \zeta = constant \) is a valid proposition while
the observation continues in this state. If \( \mu_{\zeta} \) is the sampling average over all \( \zeta \) sampled then one
choice for the state is its aggregate \( \mu_{\zeta} \), i.e.

\[
\zeta - \mu_{\zeta} = S_i(0, \sigma_{\zeta}) \quad \text{with} \quad S_i \to N, \text{the normal distribution} \quad (3)
\]

where \( S_i \) is the sampling distribution with a zero mean and a variance \( \sigma_{\zeta}^2 \) on the observable \( \zeta \) over
the sampled set. Accordingly, the sampled data in the buffer will fall within the \( \pm k\sigma_{\zeta} \) band
roughly with a probability of 0.95 for some value of \( k \). In general, one could choose the band size
arbitrarily. Note that the local distribution \( S_i \) tends to approach the Gaussian distribution in large
sample limit asymptotically. The collecting buffer could be in one of two states: (A) it would be
collecting pure stable state samples, or (B) it would be collecting samples from a roaming stream
(a transient stream out of a stable state) which has yet to settle down into a stable state. We observe

For buffer in state A: \(|\xi - \mu| \leq \theta \sigma_\xi \Rightarrow \text{stable } \xi\) (buffer to be used minimally)
For buffer in state B: \(|\xi - \mu| > \theta \sigma_\xi \Rightarrow \text{roaming stream. Buffer must be flushed of all residues.}

**Buffer management policy 1:**

Maintain only system tuples \(<\xi, \mu, \sigma\ldots(\forall \xi \in S) \land (\xi = 0)>\) through appropriate aggregate totals and approximate computation of other relevant higher moments if the system obeys A. \(\mu\) is the anytime solution of the system state currently realized near some equilibrium state \(\xi\). If the system is found obeying B, erase buffer and start preparing it to accept new states \(\xi\). Note that in this case, the system states are considered independent.

**Buffer management policy 2:**

In the event of dependent and correlated incoming data, the stream must be found stable on a hyperplane at some point \(\langle \xi_1, \xi_2 \ldots, \xi_k \rangle\) where the stability condition should now be structured as

\[
\prod_{m} \left| \frac{\sigma_m - \mu_m}{\sigma_m} \right| \leq \eta
\]  

(5)

What if the above is true only for some selected states \(\xi_i\) but fails when we include other states \(\xi_j\) into the inequality? In such situations the system would be bound on some stable hyperplane spanned by some stable states and yet would be found roaming on another hyperplane spanned by other system states. Note that in this condition we introduce the notion of multiplicative stability.

In this approach, one need not and doesn’t have to know the precise distribution function and the requisite sample size at the minimum that ought to be considered to be on the safe side. Even if the distribution function remains unknown (though its parameters are known), one could rely on the acceptance/rejection strategy as articulated above in a parametric sense.

**3. Buffer Sharing Model**

Our model hinges upon the fact that in many situations the incoming data stream is not always continuous but bursty. While this might be a difficult problem in general, in our case it accords us an opportunity precisely through its bursty-ness. A bursty stream X as shown in Fig. 1 does not need the buffer always; the buffer, for instance, remains unused between points A and B.
What if, during such quiescent periods, other streams are also accepted to use the idle buffer? If the streams are synchronized, the buffer could be fully utilized actively. Notice during the time the data is present, the buffer is mostly unused. The buffer would be needed during stream roaming time; once the stream is clearly identified in a stationary state, data mining problem moves away from buffer-residency problem to anytime state computation problem.

Consider a number of bursty streams in contention for buffer residency. Assume the streams to be similar to each other in their basic features, namely in their average quiescent period $w_{avg}$ and in their average burst length $L_{avg}$. Suppose any time an arriving data burst wants to be accommodated into the buffer it must first acquire a token from the buffer. If a token is available, it captures the token first during the rise of the burst signal (shown by the upward arrow in Fig. 2), and after the burst completes, it returns the token back to the buffer during the fall of the burst (shown by a downward arrow). This way, during a dead-signal period of one stream, another live-burst from another stream can be accommodated. This could be extended to a more involved token acquiring and releasing policy to synchronously engage multiple incoming streams.

### 4. Buffer Capture Model

In this section, we attempt to obtain a functional profile of a first-level capture buffer and make a rough estimate of its size under the assumption that it is to capture a single bursty data stream $S$ for data mining. For modeling purpose, we assume the incoming data stream is a sequence of rectangular pulses followed by a gap (dead-signal interval). The average burst width is $L$ and the average gap width is $W$. 

![Buffer Capture Model Diagram](image-url)
A complete pulse cycle is indicated in Fig. 3 along with the buffer occupancy profile such a pulse is assumed to induce. We note that the buffer in our model is not depleted during most of the burst period $L$; the buffer is used up while the signal is roaming between two stationary states. We assume the buffer depletion rate during a roaming period is going to be some $\mu$ units/time on the average. Furthermore, buffer acquisition period is going to be somewhat larger than $W$ since buffer-capture phase continues so long one is never sure with reasonable certainty that system is truly into a steady state phase. Therefore, the buffer capture phase is assumed to be on the average $W+\alpha L$, while the buffer is in quiescent mode for $(1-\alpha)L$ units of time. Note that the cycle time for a buffer is exactly the same as that of our signal, namely $W+L$.

Let

- $K =$ buffer flushing cost per cycle
- $\mu =$ buffer depletion rate during depletion part
- $h =$ the unit buffer cost per unit time
- $B =$ the peak buffer volume at the end of a refresh
- $T = W+L =$ buffer cycle time

Total relevant cost for Buffer management per unit time, $C$, is

$$ C = \frac{K\mu}{B + \mu L(1 - \alpha)} + h\frac{(1 - \alpha)LB + (W + \alpha L)/2}{W + L} $$

(6)

Therefore, at an optimum buffer volume $B'$ the relevant cost per unit time $C$ must be minimum and this yields

$$ \frac{dC}{dB} \bigg|_{B'} = 0 $$

and the optimum buffer capture volume in this case comes out to be

$$ B' = \frac{K\mu(W+L)}{h(W+L(2 - \alpha))} - \mu L(1 - \alpha) $$

(7)

The final version of this paper was presented at Proceedings of International Conference on Communications, Devices and Intelligent Systems (CODIS 2004), Kolkata, India.