Towards Burst Detection for Non-Stationary Stream Data

[work in progress]

Daniel Klan, Marcel Karnstedt, Christian Pölitz, Kai-Uwe Sattler
Department of Computer Science & Automation
TU Ilmenau, Germany
{*first.last*}@tu-ilmenau.de

Abstract

Detecting bursts in data streams is an important and challenging task, especially in stock market, traffic control or sensor network streams. Burst detection means the identification of non regular behavior within data streams. A specifically crucial challenge on burst detection is to identify bursts in the case of non-stationary data. One approach is to apply thresholds to discover such bursts. In this paper, we propose a new approach to dynamically identify suitable thresholds using techniques known from time series forecasting. We present fundamentals and discuss requirements for threshold-based burst detection on stream data containing arbitrary trends and periods.

1 Introduction

Data Stream Management Systems (DSMS) are built for processing and reacting to continuous input data. Depending on the kind of application, this also involves the processing of certain data mining tasks. One important task is burst detection. A burst is defined as a large number of occurring events. In data streams, these events are usually related to one or more time windows moving over the data stream.

For instance, if a DSMS is used for monitoring facility sensor data, detecting bursts is crucial for identifying abnormal situations like fire or heat-ups, where human interaction is needed and/or alarms should be triggered. In this scenario, detecting bursts can also help triggering the right actuations, like starting a sprinkler system, turn off heating systems or lower blinds. If one imagines heat sensors, a burst is detected by summing up the single temperatures from the time window of interest – any resulting sum that is above a given threshold is regarded as a burst and should result in according reactions. Beside facility management, burst detection is a noteworthy task in for instance stock trading systems, astrophysics and traffic control systems. In all cases we have to handle continuously streaming data, which demands for incremental one-pass algorithms. Additionally, all the mentioned scenarios usually claim for detecting bursts over a variety of dynamically chosen windows sizes, rather than above predefined ones. This is called *elastic burst detection* [Shasha and Zhu, 2004].

The naive approach of inspecting all interesting window sizes in sequence is not scalable and not suitable for data streams. There are several proposals dealing with this and similar problems. [Kleinberg, 2003; Wang *et al.*, 2002] focus on modeling bursty behavior in data streams, but not on an efficient algorithm for detecting them. The authors of [Vlachos *et al.*, 2004] mine for bursts in web search query logs. A closely related problem is the detection of outliers and anomalies [Shahabi *et al.*, 2000; Keogh *et al.*, 2002]. [Zhang and Shasha, 2006] suggests the utilization of *shifted aggregation trees* for elastic burst detection in acceptable time and space. Beside the fact that this approach is performing very well, the implementation is easy and light-weight. The idea is to build a synopsis structure containing all aggregates over all windows of all sizes of the input stream – of the largest sliding window, respectively. The authors propose the data structure needed, how to build and optimize it, as well as how to mine for bursts over the different levels of the synopsis. But, their approach in its original form is only applicable for stationary data. The problem is, the threshold for identifying bursts is only adapted very slowly with evolving input data. This does not perform satisfyingly in the presence of trends or periods. Usually, the amount of false positives and wrong negatives is too high in this case. Thus, the main idea of our work is to combine their approach with time series forecasting. To the best of our knowledge this is novel for burst detection, although forecasting methods have been used for solving similar problems. For instance, [Yamanishi and Takeuchi, 2002] utilizes auto regression models to detect outliers and change points in non-stationary data.

To illustrate the problem, again imagine heat sensors, located in a single living room. First, we will observe periods, because temperatures are changing with day and night hours. Second, especially on hot summer days, there will be a continuous heating up of the room, particularly in the midday hours. These facts should be involved in triggering actuations like lowering heating systems or blinds, but this should not result in an alarm just because the system thinks it detected a burst. Thus, the threshold for bursts must be adapted with occurring trends and periods.

There exists a lot of work concerning time series forecasting and analyzing trends and periods [Ergün *et al.*, 2004; Papadimitriou *et al.*, 2003; Hinneburg *et al.*, 2006]. For now, we use exponential smoothing following the Holt and Holt-Winters methodologies [Chatfield and Yar, 1988]. Like this, we are able to predict the next points of the input stream satisfyingly accurate. Unfortunately, forecasting methods tend to become the more inaccurate the more points we predict in the future. This is why we have to recompute the forecast periodically, ending in only short-term predictions. Thus, exponential smoothing is suitable, although we are going to investigate other methods as well.

A second main idea is to implement both – burst detection and time series forecasting – in two absolutely independent operators. Like this, they can be arbitrarily combined with other operators in a multi-functional DSMS.

The remainder of this paper is structured as follows. In Sections 2.1 and 2.2 we first briefly introduce the data structures proposed by [Zhang and Shasha, 2006] and discuss how they are applied in our context. Second, in Section 2.3 we present how exponential smoothing is used in order to adapt the threshold to trends and periods. Afterwards, we sketch the main algorithm in Section 2.4. Finally, we present motivating results from preliminary tests in Section 2.5, discuss open issues and future work in Section 3 and conclude in Section 4.

2 Non-Stationary Burst Detection

2.1 Aggregation Pyramid

Elastic burst detection is the problem of finding all windows w_i, $w_i \in \{w_1, ..., w_n\}$ at a time point t, such that for a monotonic aggregation function A the aggregate $A(w_i)$ exceeds a window specific threshold $f(w_i)$.

A simple approach to detect bursts within a data stream is to check each relevant window size starting at time point t for a burst. Consequently, for a burst detection over k different window sizes $O(k \cdot N)$ time is necessary, where N denotes the length of the explored sequence. This is not practicable for online data stream mining.

Another approach is presented by Zhang and Shasha in [Zhang and Shasha, 2006]. The authors introduce a data structure called *aggregation pyramid*. An aggregation pyramid is a triangular data structure over N stream elements with N levels. N corresponds to the length of the analyzed sequence (the size of the regarded time window). Level 0 has N cells containing the original data elements $y_1, ..., y_N$. Level 1 has $N-1$ cells and stores aggregates over two-element sets from the original data stream. The construction of the aggregation pyramid is recursive, that means level h has $N-h$ cells and each cell $c(h,t)$ contains the aggregates from a window of length $h+1$ starting at element t. The aggregate of cell $c(h,t)$ is computed by $c(h,t) = A(c(h-1,t), c(0,t+h))$. Consequently, level N stores the aggregate over the complete sequence. Figure 1 shows an aggregation pyramid over a windows of size 8.

The aggregation pyramid has some interesting properties. For instance, each cell $c(h,t)$ of the aggregation pyramid spans to a sub pyramid. The window starting at time t composed of the following $h+1$ cells on level 0 is called the shadow of the cell $c(h,t)$. Now it is straight forward to identify a burst. If a cell $c(h,t)$ exceeds the threshold for its shadow, a burst starting at time t is discovered.

2.2 Aggregation Tree

In order to efficiently identify the exact length and time of the burst Zhang et. al additionally introduce a data structure called shifted aggregation tree. The shifted aggregation tree is a hierarchical tree structure, whose nodes are a proper subset of the cells of the aggregation pyramid. Each level of the aggregation tree consists of a number nodes. The leaf nodes of the aggregation tree correspond to the level 0 nodes of the aggregation pyramid. A node on level i of the aggregation tree represents the aggregate of its child nodes on levels $\leq i-1$. Figure 1 also shows a three level aggregation tree (shaded cells). Each node of the aggregation tree corresponds to one node of the aggregation pyramid.

A central property for burst detection with aggregation trees is that the aggregate of each subsequence of length l,

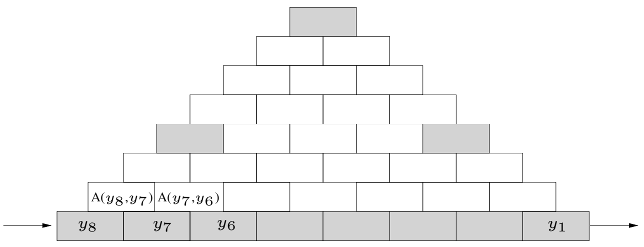


Figure 1: An aggregation pyramid over a window of size 8 and the corresponding three level aggregation tree (shaded cells)

with $l \leq h_i - s_i + 1$, is covered by the shadow of a node at level i. h_i denotes the size of the window at level i and s_i is the shift at level i. That means, each window of a size $\leq h_i - s_i$ is covered by at least one node on level i and consequently, each burst occurring in $h_i - s_i$ can be discovered.

For the detection of a burst within the aggregation tree it is sufficient to check if one aggregate of a subwindow between $h_{i-1} - s_{i-1} + 2$ and $h_i - s_i + 1$ is exceeded. If a burst is discovered a detailed search on the shadow of the node which exceeded the threshold is necessary.

2.3 Threshold Detection

A main problem of the burst detection is the definition of a suitable threshold. If the threshold is set too high, bursts are potentially not detected. The other case is that the threshold is set too low. This results in a high number of (maybe wrong) alerts. Moreover, this also results in a higher workload, because each time an alert is triggered all cells within the shadow of the concerned cell must be investigated.

[Zhang and Shasha, 2006] suggests a heuristic to identify a threshold dependent on the data distribution. The presented approach has the disadvantage that it is only appropriate for stationary data with well known distribution. Consequently, an algorithm using the presented threshold cannot react correctly to trends or periods. In the following, we present an approach using exponential smoothing for continuously forecasting the stream in order to adapt the threshold.

Exponential smoothing is a popular technique to produce smoothed time series. In single (also called simple) exponential smoothing the forecast is the linear combination of the weighted average of the current observation and the previous smoothed observations. With this, recent observations are more relevant for forecasting than older observations. The single exponential smoothing is defined by

$$S_t = \alpha \cdot y_t + (1-\alpha) \cdot S_{t-1} \tag{1}$$

S_t denotes the forecast value and α, $0 < \alpha < 1$, weights the influence of new values. By direct substitution of the equation back into itself S_t becomes the weighted average of a number of past observations

$$\begin{aligned} S_t &= \alpha \cdot (y_t + (1-\alpha) \cdot y_{t-1} + (1-\alpha)^2 \cdot y_{t-2} \\ &\quad + (1-\alpha)^3 \cdot y_{t-3} + \cdots) + (1-\alpha)^t \cdot y_0 \end{aligned} \tag{2}$$

Single smoothing is sufficient for short-range forecasting and static time series.

For non-static time series showing a trend the double exponential smoothing (also known as Holt exponential smoothing) is appropriate. An example is a room continuously warming up due to a running radiator. In this case an alert is undesirable. Double exponential smoothing is defined as

$$S_t = \alpha \cdot y_t + (1-\alpha)(S_{t-1} + b_{t-1}) \quad (3)$$
$$b_t = \gamma \cdot (S_t - S_{t-1}) + (1-\gamma)(b_{t-1}) \quad (4)$$

where α and γ, with $0 < \alpha, \gamma < 1$, are predefined weights. They can be achieved via non-linear optimization techniques, such as the Marquardt algorithm [Marquardt, 1963]. The first equation adjusts S_t for the trend b_{t-1} depending on S_{t-1}. The second equation updates the trend, which can be expressed as the difference between S_t and S_{t-1}.

If the time series contains both, a trend and seasonality, we can use the third exponential smoothing (also known as Holt-Winters exponential smoothing). Seasonality means that the data distribution contains periods of constant length. For instance, the temperature of a room is changing periodically with the time of the day and the outside temperature. Or, the occupants may heat up the room for the same period of time each day. The Holt-Winters exponential smoothing is defined as follows:

$$S_t = \alpha \cdot \frac{y_t}{I_{t-L}} + (1-\alpha)(S_{t-1} + b_{t-1}) \quad (5)$$
$$b_t = \gamma \cdot (S_t - S_{t-1}) + (1-\gamma) \cdot b_{t-1} \quad (6)$$
$$I_t = \beta \cdot \frac{y_t}{S_t} + (1-\beta) \cdot I_{t-L} \quad (7)$$

where α, β and γ, with $0 < \alpha, \beta, \gamma < 1$, are also predefined weights. b_t denotes the trend factor, I is the seasonal index and S_t is our overall smoothing. L is the length of a period, which must be known a-priori.

2.4 Non-Stationary Burst Detection Algorithm

In the following we describe the general burst detection algorithm (Algorithm 1) using the exponential smoothing. As presented here, the algorithm can handle linear trends.

We start with an initial learning phase in order to get first parameters S and b from the exponential smoothing (implemented in method $expsmoothing$). The size cnt of the window used for this should depend on the input data and the actual application scenario. This is one of the open issues we currently investigate. In parallel, we build the aggregation tree (included in method $bdet$). After the initial learning, we can start burst detection. This is also done by the method $bdet$, which runs the detection algorithm from [Zhang and Shasha, 2006].

The important point is the computation of the current threshold f. For this, we use the parameters from the exponential smoothing in order to predict the value μ of the current stream element. The standard deviation σ^2 used for computing f is determined from the N elements in the regarded time window.

In order to base the threshold computation on only short-term predictions and to identify changes in the stream, we have to recompute the exponential smoothing in regular intervals. Currently, we use a fixed interval of time cnt. We are investigating if this should be replaced by a more dynamic and flexible approach. Note that if a burst occurs, this will be included into smoothing calculation and we predict a wrong trend! Thus, we only run exponential smoothing on elements from non-bursty periods.

2.5 Motivating Observations

Figure 2(a) and Figure 2(b) show first observations for synthetic data sets. We compare the approach using thresholds

```
Input: new stream element $y_t$, time $t$, number $cnt$
Output: $true$ if burst is detected
/* collect first cnt points */
if $t = 0$ then
    $Y = \emptyset$;
end
if $t \leq cnt$ then
    $Y = Y \cup \{y_t\}$;
    /* only update tree */
    $bdet(-1, y_t)$;
    if $t < cnt$ then
        return $false$;
    end
    /* initial exponential smoothing on first points */
    $S, b = expsmoothing(Y)$;
    $Y = \emptyset$;
    $k = 0$;
end
/* collect next cnt points */
$Y = Y \cup \{y_t\}$;
/* predict current value ($k$ time steps since computing $S$) */
$\mu = S + k \cdot b$;
$f = \mu + 2 \cdot \sqrt{\sigma^2}$;
$k = k + 1$;
/* update tree and check for bursts */
$burst = bdet(f, y_t)$;
/* if a burst occurs, smoothing should be delayed */
if $burst = true$ then
    $Y = \emptyset$;
end
/* recompute exponential smoothing */
if $|Y| \geq cnt \wedge burst = false$ then
    $S, b = expsmoothing(Y)$;
    $Y = \emptyset$;
    $k = 0$;
end
return $burst$;
```

Algorithm 1: Basic algorithm

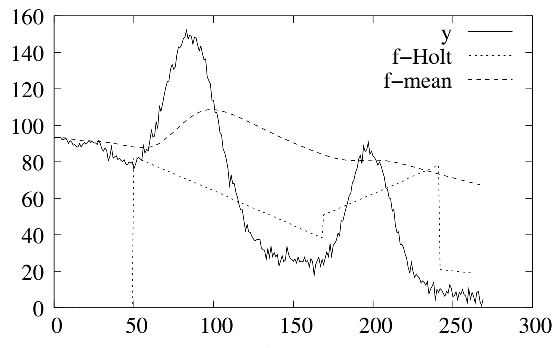
based on the Holt exponential smoothing with that using thresholds based on the mean (as [Zhang and Shasha, 2006] suggests) on normally distributed data.

The stream values (y) in Figure 2(a) show two well-separated Gaussians on a decaying exponential baseline plus normally distributed zero-mean noise with variance 6.25[1]. Using a threshold as suggested by Zhang (f-mean) both bursts are correctly detected. But, Figure 2(a) shows that the first burst increases the threshold in a way that almost prevents from detecting the second burst. Obviously, it is detected rather late. A detailed search on the corresponding part of the aggregation tree finds only a small part of the burst. In opposition, the algorithm using the Holt strategy (cnt is set to 16) adapts the threshold (f-Holt) in a way that the second burst is detected satisfyingly.

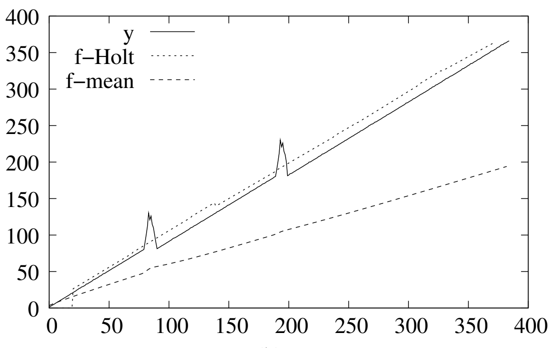
For our second experiment we combined a continuous trend with two small bursts. This simple experiment illustrates the general problem of the Zhang approach. The trend is not detected. Rather, the trend is interpreted as a burst of undefined length. In contrast, the Holt approach detects the bursts correctly by adapting the threshold to the trend.

These first observations show that the presented approach using a forecasting strategy in order to adapt the threshold for burst detection is very promising.

[1]http://www.nlreg.com/gauss1.htm



(a)



(b)

Figure 2: First observations on synthetic data

3 Ongoing Work

There are several open issues concerning our approach. The majority of them is part of our ongoing work. First, we need to evaluate and tune the parameters of our approach, e.g., the number of time steps we predict the stream in the future (*cnt*). This goes along with tuning the forecasting technique itself, e.g., find suitable smoothing parameters. Different forecasting techniques provide different qualities depending on the input data and the time we predict. Thus, we also plan to test forecasting methods other than the exponential smoothing technique.

A second field of work is the used aggregation tree itself. Here, we mainly investigate how to efficiently maintain the structure. This involves a comparison of batch-like versus incremental one-element updates. An interesting idea is to dynamically adapt the tree, i.e., let it grow or shrink.

After dealing with these rather theoretical issues we are going to implement the burst detection in a DSMS. For this, we will implement separated mining operators that can be used in the system. Based on this, we will elaborately evaluate the approach. This will be done on synthetic data as well as real data from facility sensors. Further, we are going to investigate the impact of different window sizes and different kinds of bursts. A formal definition of bursts and an according classification will be very helpful. Also, we need to investigate solutions for dealing with non-linear trends, like transforming the data accordingly.

4 Conclusion

Burst detection is a very important task, not at last in a facility management scenario. But, it reveals several challenges when applied on data streams. In this paper, we introduced a novel approach of detecting bursts in the presence of trends and seasonalities. To achieve this, we combined the idea of shifted aggregation trees as the basic synopsis structure with forecasting techniques. We showed that this simple approach is suited for handling linear trends. However, this is only a first step towards an all-encompassing solution. We will continue to develop the approach and evaluate it exhaustively. After all, this will be a major milestone on the way to an automated and intelligent facility management system.

References

[Chatfield and Yar, 1988] Chris Chatfield and Mohammad Yar. Holt-Winters Forecasting: Some Practical Issues. *The Statistician, Special Issue: Statistical Forecasting and Decision-Making*, 37(2):129–140, 1988.

[Ergün *et al.*, 2004] F. Ergün, S. Muthukrishnan, and S. C. Sahinalp. Sublinear methods for detecting periodic trends in data streams. In *LATIN*, 2004.

[Hinneburg *et al.*, 2006] A. Hinneburg, D. Habich, and M. Karnstedt. Analyzing Data Streams by Online DFT. In *ECML/PKDD-2006 Int. Workshop on Knowledge Discovery from Data Streams (IWKDDS-2006)*, pages 67–76, 2006.

[Keogh *et al.*, 2002] Eamonn Keogh, Stefano Lonardi, and Bill 'Yuan chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD'02*, pages 550–556, 2002.

[Kleinberg, 2003] J. M. Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7(4):373–397, 2003.

[Marquardt, 1963] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[Papadimitriou *et al.*, 2003] S. Papadimitriou, A. Brockwell, and Ch. Faloutsos. AWSOM: Adaptive, Hands-Off Stream Mining. In *VLDB 2003*, pages 560–571, 2003.

[Shahabi *et al.*, 2000] Cyrus Shahabi, Xiaoming Tian, and Wugang Zhao. Tsa-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In *SSDBM'00*, page 55, 2000.

[Shasha and Zhu, 2004] Dennis Shasha and Yunyue Zhu. *High Performance Discovery in Time Series: Techniques and Case Studies*. Springer, 2004.

[Vlachos *et al.*, 2004] M. Vlachos, Ch. Meek, Z. Vagena, and D. Gunopulos. Identifying Similarities, Periodicities and Bursts for Online Search Queries. In *SIGMOD 2004*, pages 131–142, 2004.

[Wang *et al.*, 2002] Mengzhi Wang, Tara M. Madhyastha, Ngai Hang Chan, Spiros Papadimitriou, and Christos Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *ICDE*, 2002.

[Yamanishi and Takeuchi, 2002] K. Yamanishi and J.-I. Takeuchi. A Unifying Framework for Detecting Outliers and Change Points from Non-stationary Time Series Data. In *SIGKDD 2002*, pages 676–681, 2002.

[Zhang and Shasha, 2006] Xin Zhang and Dennis Shasha. Better burst detection. In *ICDE '06*, pages 146–149, 2006.