IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

One Approach of Approximation for Incoming Data Stream in IoT based Monitoring System

Vladyslav Alieksieiev
Department of Applied Mathematics
Lviv Polytechnic National University
Lviv, Ukraine
vladyslav.i.alieksieiev@lpnu.ua

***Abstract*— IoT devices and platforms are a fast growing market. One can mention a number of businesses relying on easy opportunity to build real-time monitoring systems using modern software and IoT hardware solutions. However, the growth has revealed a number of complex problems. Many problems are in area of data processing and storing huge volumes of information. Due to wide use of different kinds of sensors, and even a sets of sensors within each single device, on one hand, practitioners discover unpleasant effects of data losses caused by data packages losses or delays while its transition from sensor to server. On the other hand, huge volumes of data require to use some big data approaches and many startup projects feel the problem of lack of resources. Many of them feel lack of data storage facilities or become unable to support huge data sets due to lack of finance. The paper is focused to research the problem approximation for incoming data stream to make it smaller the volume of data to be stored but to keep it possible to be used. A few approaches to use such data compression via its approximation are discussed with application to IoT based real-time monitoring system.**

Keywords*— *data compression, approximation algorithm, data stream processing, IoT platform, big data

I. INTRODUCTION

A contemporary IT industry trends revealed many new applications of data storages and an IoT became one of the primary trends [1]. Both huge enterprises and small businesses are now dependent from quality of data and quality of data analytics [2]. Some new approaches to understand data and the value of the data had appeared. The industry stepped behind the relational databases and time series has determined new approaches to store and process data [3]. IoT technologies gave an opportunity to design some new platforms for supporting business both with surveillance tools and analytics software applications. This research paper presents some discussion related to IoT bases "real-time" monitoring systems. Common architecture of such platforms are [4]:

- *Device* – a remote computer like Raspberry Pi or any of its alternatives or some custom hardware device that may include a set of sensors (business solution may consist of a whole network of such devices).
- *Internet* – any kind of Internet wireless connection via Wi-Fi, GPRS, 3G/4G or anything else supplied with mobile network (business solution may combine different types of Internet service providers to establish connection).
- *Cloud* – any popular IoT platform to store and process big data.

Adding some software solution to provide data analytics to that IoT platform than it becomes a powerful business tool supporting real-time monitoring. There is a number of companies developing their own platforms or exploiting powerful cloud services to provide their client with such platform as a reliable business solution. Many researchers and practitioners in area of data analysis declare statements similar to [2]: "data accumulation can enable deeper insights and help us to gain more experience and wisdom". There are many evidences of great performance of time series analysis already and there is a number of solutions for time series databases [3].

However, practical use of such system reveals some serious problems. Many of these problems were predicted earlier. For example, the problem of data uncertainty was known, described and even has some categorization [5]. The era of Big Data has just revealed the complexity of problems, which came with those volume, variety and velocity of big data. Recent researches denoted an importance of data losses problem in monitoring systems [4] and problems of storing big volumes of obsolete data in such systems [6]. One may find some techniques to solve these problems in an analytic manner [7-9]. Some techniques of data clearing allows to aggregate data simultaneously [10], [11].

Unfortunately, there are no common recipes yet in data science to manage with big data sources of any kind. There are some approaches and some of these approaches are well developed, but in some particular cases, there appears the specifics making it difficult or impossible to implement a common solution. Current research can be considered as an alternate or a supplement to those discussions presented in [6]. Unlike to [6] it is offered here not to rely on clustering or quantization, which is appropriate to process obsolete data, but to use approximation as a reliable and a well-developed technique of mathematics.

Generally, we still have the same problem of a large number of devices each with a set of sensors generating huge volume of data. The data from all the devices and all the sensors come to a server (non-relational database). Using these data for online monitoring system allows making some assumptions to ease the solution of the problems.

II. PREREQUISITES AND MEANS FOR SOLVING THE PROBLEM

Let's define the primary task for the problem solution. First, the aim of the research is to find a way to reduce the number of values in the incoming data flow from the sensors and not to lose the quality of understanding the scope. Second, we'd like to keep some quantitative scope if

possible or to have it in some approximation. Third, we can discuss the ability to recover original data in cases it is necessary. Nevertheless, the last option only remains the option yet.

The matter is, if we consider a real-time monitoring system having a primary target to notify about some critical issues, then it is of less interest to see what exactly normal conditions was surveyed. This allows us to ignore many aspects of value changes within some normal boundaries.

Two sub-problems can be solved simultaneously: 1) an approximation to store values in a database, and 2) an approximation to reduce data volume at the node (peripheral device) or an approximation "on-the-fly" to form smaller packages to be sent to a server. The first problem is rather simple, if to consider only the task to remove the excessive data (to remove less informative values). The second problem is more difficult and requires some actions at the node. The difficulty is that the node will decide about necessity of the values gathered from sensors. This should be made very carefully not to lose an important data. Therefore, the circumstances can be very important to understand whether to implement the approximation at the node. While we have a monitoring system with less analytics purpose, we can assume each node (peripheral device) to gather the same data with respect to its location. This means both problems can be solved successfully.

Now, we can define two key requirements for constructing an appropriate algorithm:

- Simplicity – the algorithm should be easy to implement and fast enough to be used "on-the-fly".
- Reliability – the algorithm should give a reliable approximation for a data set and hold the information about any abnormal values.

A. *Incoming data flow (stream)*

Let's assume the incoming sequence of values to be $f(t)$ with t as a time. Measurements are made with an equal time lapse $\Delta t = const$. This means each next value $f_n := f(t_n)$ is obtained after a fixed period $t_{n+1} = t_n + \Delta t$ and $n = 0 \ldots \infty$. This allows considering values as a discrete (Fig. 1). For the purpose of determinateness, the renumbered values presented at Fig. 2. The curve of the input sequence of values presented at Fig. 3. This curve is similar to a signal and one can offer to use some techniques of signal processing. There are many known methods of signal processing and data compression applicable to signals [12]. Unfortunately, we have no evidences of any periodic behavior or repeatable oscillations to be confident to implement those techniques of signal processing. For example, many techniques rely on assumption about definite periodicity in a signal and one of the hardest situations is to use these techniques to process "white noise". Our "signal" is similar to "white noise". There are regular appearance of some unpredictable values from sensors. Meanwhile each sensor has some "normal" range of values. This means, there could be a senseless part of values within that range, and some significant values outside the range can be cut as an outliers.

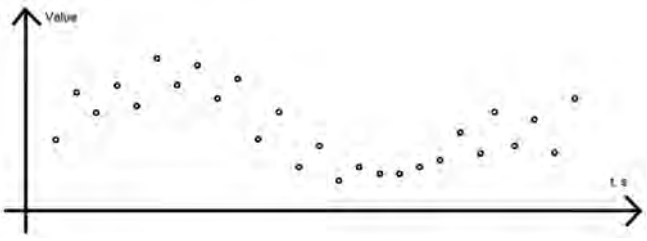

Fig. 1. Input sequence of discrete values (incoming data stream) – measurements with equal intervals of time Δt

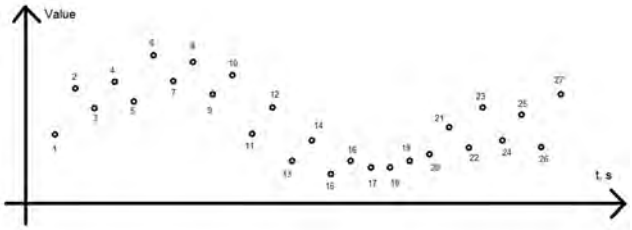

Fig. 2. Number the sequence of values for certainty

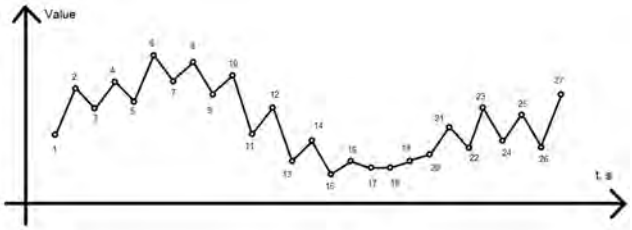

Fig. 3. The curve of the input sequence of values (incoming "signal")

B. *Idea of the algorithm*

All values in a data set can be divided into two sub-sets: 1) "maximums" and 2) "minimums". The "maximums" are the values greater than the previous ones. The "minimums", otherwise, are the values less than the previous ones. Both the first and the last values in the data set can be marked simultaneously as a "maximum" and a "minimum". The same simultaneous marking is possible with the consequently equal values. However, this can be an option for the case of equal values to have adequate presentation in resulting approximation. If there were an oscillation character observed, then it would be a rare situation. Thus, the decision about marking the equal values to be made according to necessity. The marking procedure result presented at Fig. 4.

Next, the local extremums can be found within each sub-set as shown at Fig. 5. Values number 1, 6, 18, 23, 25, and 27 are the local extremums among "maximums". Values number 1, 7, 15, 24, 26, and 27 are the local extremums among "minimums". Note, the first and the last values are both marked as local extremums.

There can be two strategies to select local extremums: 1) a use of all extremums, 2) a use of selected extremums only.

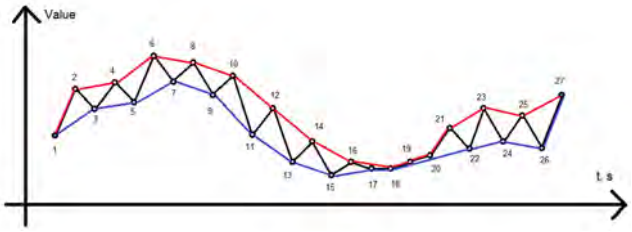

Fig. 4. Split the whole set of values into 2 groups: “maximums” (red, top) and “minimums” (blue, bottom)

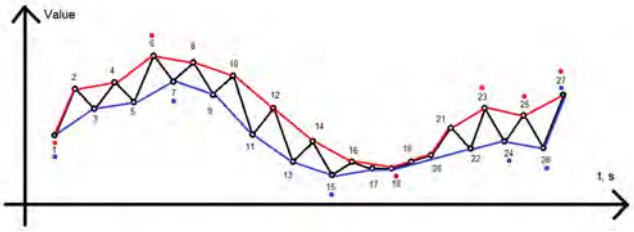

Fig. 5. Extremum selection within each of 2 groups: “maximums” (red, top) and “minimums” (blue, bottom)

According to a strategy of selection of local extremums, one can receive different kind of approximation. Fig. 6 and Fig. 7 present respectively the implementation of first and second strategy. The first one looks to be more accurate, compared to the second one. Nevertheless, each strategy has some advantages and disadvantages, while both give a rough view to data set and a good “compression” or “consolidation”. As it was asserted in [13] the approximation via extremums allows to recover signal using “bell-shaped impulse approximation”.

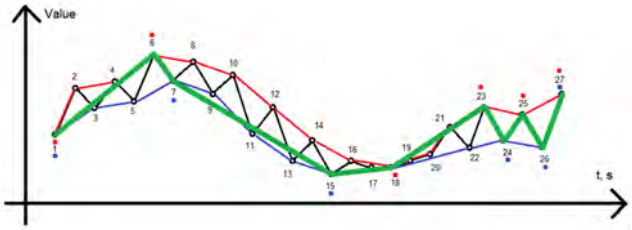

Fig. 6. Approximation for all extremums: “maximums” (red, top) and “minimums” (blue, bottom), approximation (green)

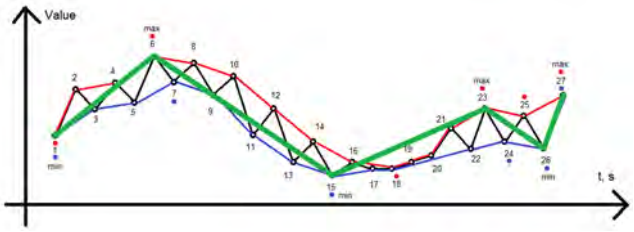

Fig. 7. Approximation for selected extremums – “maximums of maximums” and “minimums of minimums”: “maximums” (red, top) and “minimums” (blue, bottom), approximation (green)

In cases of necessity of further recover of original profile, the algorithm can be replaced to that in [13]. However, in circumstances described above for the purpose of a monitoring system, one may gain great benefits from data consolidation.

III. Solution of the Examined Problem

Hence, for certainty, let’s choose an approximation by extremum in the form of a selected extremums (this mean to use “maximum of maximums and minimums of minimums” according to those shown in Fig. 7). Now, we can consider how this algorithm is formally defined and what are the cases it can be used (implemented).

A. Peculiarities of implementation of approximation with extremums

There are two cases of possible application for consolidation of data: 1) aggregation of data already stored in the database, and 2) data aggregation “on-the-fly”, which can be performed at the node. Both are equivalent to the problems we established initially.

One should note that the offered approach for the algorithm is quite convenient. It requires simultaneously only a few values of data for calculations. This yield the calculations “on-the-fly” with just one previous f_{n-1} value and one current f_n value. It is also necessary to have a three previous values to fix “maximums” ($f_2^{max}, f_1^{max}, f_0^{max}$) and a three previous values to fix “minimums” ($f_2^{min}, f_1^{min}, f_0^{min}$), having indexes 0, 1, and 2 to stand for last, previous to the last and second previous values.

Another aspect is to have actual pair of values presenting both the value and a timestamp. Sure, in case of fixed time intervals $\Delta t = const$ for the values measurement, there is no need to fix the timestamp. Even in case of a time interval or a “window”, just the initial time t_0 supplemented with value order number is enough. However, when we disorder the time series, we need to fix a timestamp for each value. Anyway it is now possible to fix (to store) in the database only the extremums and not the whole sequence of values.

B. The formal algorithm

Now it is possible to describe a formal algorithm with the steps starting from new value f_n has come.

1. A **new value f_n is obtained** (if not then go to Step 3 – the end of the algorithm), and make a data values “shift” accordingly.

   1.1. If the new value is greater than the previous $f_n > f_{n-1}$ then the value should be marked as “**maximum**”:
   $f_2^{max} = f_1^{max}$, $f_1^{max} = f_0^{max}$ and $f_0^{max} = f_n$, and fix the time $t_1^{max} = t_0^{max}$, $t_0^{max} = t_n$ – the time is needed further to fix the extremum (local maximum).

      1.1.1. If $f_2^{max} < f_1^{max}$ and $f_1^{max} > f_0^{max}$ then the value $f_1^{max}(t_1^{max})$ is an extremum: $f_k^{extr} = [t_1^{max}, f_1^{max}(t_1^{max})]$, $k = k + 1$.

   1.2. If the new value is less than the previous $f_n < f_{n-1}$ then the value should be marked as “**minimum**”:
   $f_2^{min} = f_1^{min}$, $f_1^{min} = f_0^{min}$ and $f_0^{min} = f_n$, and fix the time $t_1^{min} = t_0^{min}$, $t_0^{min} = t_n$ – the time is needed further to fix the extremum (local minimum).

      1.2.1. If $f_2^{min} > f_1^{min}$ and $f_1^{min} < f_0^{min}$ then the value $f_1^{min}(t_1^{min})$ is an extremum: $f_k^{extr} = [t_1^{min}, f_1^{min}(t_1^{min})]$, $k = k + 1$.

   1.3. If the new value is equal to the previous $f_n = f_{n-1}$ then the value should be marked **both** as “**maximum**” and “**minimum**”:

1.3.1. Execute actions of Step 1.1.

1.3.2. Execute actions of Step 1.2.

2. If $k = K$ then the value is a limit (maximum allowed value) and Step 2.1 to be execute, otherwise – Step 2.2.:

   2.1. Fix the array of values f_K^{extr} (this means to save in database or send the package to server). Note: it is actually assumed here to have an array of pairs "time–value" $[t_k, f_k]$.

   2.2. Go back to Step 1.

3. Execute Step 2.1 over the array of values f_K^{extr} (means to process the rest of the values not fixed yet) and finish the algorithm execution.

IV. Results and discussion

The algorithm of approximation by extremums is very easy to understand, easy to implement, and easy to support. There are some problems, and the primary problem is, that the algorithm represents some kind of compression with losses. Nevertheless, those conditions of its application for a monitoring system match the key demand to keep the outliers and not to consider the inner normal range of values (due to its senseless in a common way).

However, there are some approaches to supply ability of approximate signal recovery in case of application of some special techniques. This can a very promising approach for many data science purposes, for IoT based platforms. Due to serious reduce of necessary volume for data storage, one can find it possible to use traditional RDBMS in some areas instead of big data sources.

The practical implementation of the algorithm to real data set at temperature surveillance system gives the average compression up to 10 times compared to initial volume. On one hand, this result can be considered a very particular case of a particular system, but, on the other hand, it relies on a strong mathematics, so it is rather consistent.

V. Conclusion

The algorithm in general is quite simple to implement both in case of aggregation of existing data from the database, and in case of processing data "on-the-fly" at the node (peripheral device). Parameter *K* allows to set some value analysis "window", so that one can adjust the accepted volumes of data to be "fixed" (at the database server, or to send the packet from node to server). This ease of use gives a great opportunity to make a software solution even with a "weak" hardware. Meantime, any of preferred task (at the peripheral device or at server) can be solved successfully.

References

[1] A. Oram. Scaling Data Science for the Industrial Internet of Things. O'Reily, 2017.

[2] Y. Lin, and W. Xiao. Implementing a Smart Data Platform: How Enterprises Survive in the Era of Smart Data. O'Reily, 2017.

[3] T. Dunning, and E. Friedman. Time Series Databases: New Ways to Store and Access Data. O'Reily, 2015.

[4] V. Alieksieiev, and O. Gaiduchok, "About the problem of data losses in real-time IoT based monitoring systems," Proceedings of International Scientific Conference "Mathematical Modeling" (Borovets, Bulgaria, December 13–16, 2017), STUME "Industry 4.0", Sofia, Bulgaria, Year I, vol. 1/1, pp.10–11, 2017

[5] C. J. Date. Database in Depth: Relational Theory for Practitioners. O'Reilly, CA, 2005.

[6] V. Alieksieiev, G. Ivasyk, V. Pabyrivskyi, and N. Pabyrivska, "Big data aggregation algorithm for storing obsolete data," Proceedings of International Scientific Conference "High Technologies. Business. Society 2018" (Borovets, Bulgaria, March 12–15, 2018), STUME "Industry 4.0", Sofia, Bulgaria, Year II, iss. 1 (3), vol. I "High Technologies", pp.113–115, 2018.

[7] P. Bruce, A. Bruce, Practical Statistics for Data Scientists. O'Reily, 2017.

[8] M. Milton. Head First Data Analysis. O'Reily, 2009..

[9] A. B. Downey. Think Stats. O'Reily, 2015.

[10] A. Jain, M. Murty, and P. Flynn, "Data Clustering: A Review," ACM Computing Surveys, vol. 31, no. 3, pp. 264-323, 1999.

[11] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," ArXiv.org, 2011. – https://arxiv.org/pdf/1109.2378.pdf

[12] S. W. Smith. The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, 1997.

[13] N. V. Myasnikova, M. P. Beresten, and M. P. Stroganov, "Approximation of multi extremum functions and its applications to technical systems," Herald of higher education institutions. Volga region. Engineering sciences, no. 2 (18), pp.113–119, 2011. [In Russian]