Noname manuscript No.
(will be inserted by the editor)

Constructing Fading Histograms from Data Streams

Raquel Sebastião · João Gama · Teresa Mendonça

Received: date / Accepted: date

Abstract The ability to collect data is changing drastically. Nowadays, data is gathered in the form of transient and finite data streams. Memory restrictions preclude keeping all received data in memory. When dealing with massive data streams, it is mandatory to create compact representations of data, also known as synopses structures or summaries. Reducing memory occupancy is of utmost importance when handling a huge amount of data. This paper addresses the problem of constructing histograms from data streams under error constraints. When constructing online histograms from data streams there are two main characteristics to embrace: the updating facility and the error of the histogram. Moreover, in dynamic environments, besides the need of compact summaries to capture the most important properties of data, it is also essential to forget old data. Therefore, this paper presents sliding histograms and fading histograms, an abrupt and a smooth strategies to forget outdated data.

Keywords Data Streams · Online Histograms · Error Constraints · Fading Histograms

Raquel Sebastião
LIAAD - INESC TEC, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200 - 465 Porto, Portugal
& Dep. Matemática, Fac. Ciências da Universidade do Porto (FCUP)
Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal
E-mail: raquel.sebastiao@inescporto.pt

João Gama
LIAAD - INESC TEC
& Fac. Economia da Universidade do Porto (FEP), Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
E-mail: jgama@fep.up.pt

Teresa Mendonça
Dep. Matemática, Fac. Ciências da Universidade do Porto (FCUP)
& Center for Research & Developments in Mathematics and Applications (CIDMA)
Dep. de Matemática, Campos Universitário de Santiago, 3810-193 Aveiro, Portugal
E-mail: tmendo@fc.up.pt

1 Introduction

When very large volumes of data arrive at a high-speed rate, it is impractical to accumulate and archive in memory all observations for later use. Nowadays, the scenario of finite stored data sets is no longer appropriate because information is gathered assuming the form of transient and infinite data streams, and may not even be stored permanently. Therefore, it is unreasonable to assume that machine learning systems have sufficient memory capacity to store the complete history of the stream.

In the data stream context "you only get one look". Processing time, memory and sample size are the crucial constraints in knowledge discovery systems [2] that handle this complex and interactive type of data. These restrictions impose that in the data stream systems, the data elements are quickly and continuously received, promptly processed and discarded immediately. Since data elements are not stored after being processed it is of utmost importance to create compact summaries of data, keeping only a small and finite representation of the received information.

Data synopses are helpful and critical in these circumstances: they provide a compact representation of data and their size is small compared to the original size of the data under analysis. Besides providing approximated representations of the data that is being processed, the data synopses will allow fast and relative approximations to be obtained in a wide range of problems, such as: range queries, selectivity estimation, similarity searching and database applications, classification tasks, change detection and concept drift.

The data stream context under which these synopses are used also imposes that their construction algorithms must be single pass, time efficient and have, at most, space complexity linear in relation to the size of the stream. Moreover, in most cases, data is not static and evolves over time. Synopses construction algorithms must allow online updates on the synopses structures to keep up with the current state of the stream.

Indeed, in dynamic environments, the useless of old data stresses out the need of forgetting the same. While constructing compact representations of data, the forget of outdated data can be abrupt or gradual: depending on the use of sliding histograms or fading histograms. The former approach is obtained when constructing a histogram over a sliding window and the second constructing fading histograms over the stream hitherto.

This paper is organized as follows. It starts by introducing synopses structures. The problem of constructing histograms is also addressed and an overview of some of the existing techniques to address this problem is presented. Section 3 proposes an approach to construct online histograms, under error constraints, from open-ended data streams. Section 4 advances two strategies, abruptly and smoothly, to forget outdated data and Section 5 compares both strategies when applied to an artificial data set, a public data set and a real word problem. Finally, Section 6 presents conclusions on the proposed approaches.

2 Synopses Structures

The massive amount of information that is produced at a high-speed rate demands the use of compact representations of data. The development of methods to deal with these problems is an issue of central interest to data mining community and has attracted a lot of research attention.

Synopses structures for massive data sets are discussed in [9]. In order to provide approximated answers to different queries, different kinds of summarization techniques are considered and the online update of such structures in a dynamic scenario is also discussed. Sampling [24], hot lists [5, 18], wavelets [3, 10, 16], sketches [4] and histograms [11, 13, 15] are examples of synopses methods to obtain fast and approximated answers.

2.1 Sampling

The *sampling* technique basically consists on selecting a representative subset of the data. However, simple random sampling may not be able to capture important properties of data (such as: skewness, unbalanced classes, among others). To overcome this main drawback, adaptive sampling methods have been proposed. For instance, a stratified sampling approach generate a sample with approximately the same percentage of each class as the data stream.

The work in [24] presents an one-pass algorithm for selecting without replacement a random sample of size n from set of size N, in which the value of N is unknown beforehand. The experiments show that the proposed reservoir algorithm runs in optimum time, outperforming similar methods.

2.2 Hot Lists

This summarization technique keeps the k items that appear more times in a data stream (most frequently or more than a threshold), maintaining an ordered set of $k < value, count >$ pairs. It is mostly used to compute aggregate statistics, for answering hot lists (such as best sellers lists or most popular itemsets), related queries (such as selectivity estimations in query optimization) and association rules.

In [18] an algorithm for approximate counting over data stream is introduced. This (deterministic) algorithm finds k items, each occurring at least $N/(k+1)$ times. It takes as input a parameter k (the number of counters) and processes the stream, retaining up to k pairs (the most observed tokens and their respective counters).

The technique presented in [5] relies on random sampling to construct sets of items and further these sets are grouped deterministically into a small number of subgroups. The method proposed is able to deal with deletion operations as well as inserts. Experiments show that this algorithm is remarkably accurate in dynamically tracking the hot items independent of the rate of insertions and deletions. However they ignore the fact that recent elements of a stream are more important than those arrived a long time ago.

2.3 Wavelets

Widely used in signal and image processing applications, a *Wavelet Decomposition* (like Discrete Cosine and Fourier decompositions) is a special mathematical transform that attempts to capture the trends through a hierarchical decomposition of numerical functions, commonly named wavelet coefficients. Using all of the wavelet coefficients the initial signal can be fully recovered. However, often very few of the wavelet coefficients are significant and most of them are small or insignificant. In practice, a small number of significant coefficients (those that carry the highest information) is needed to capture the trends of a signal providing a suitable approximation. Approximate query estimation is one example where the application of wavelets is feasible.

In the data stream domain, wavelets have traditionally been used in query processing applications. In this context, and due to the evolving nature of data streams, the dynamic maintenance of the dominant coefficients of the wavelet representation requires some novel algorithmic techniques. Consider a data stream as an unbounded sequence of numerical elements $x_1, x_2, ..., x_n$ and X a random variable with probability density function f. Since f is unknown, the sample $x_1, x_2, ..., x_n$ must be used for estimating the coefficients. Since n increases continuously over time, the resulting number of coefficients may be infinite. Due to memory constraints, it is not feasible to keep all of them. Thus, the wavelet decomposition of f has to be truncated.

The pioneer hierarchical synopsis was the Harr wavelet transform. Based on this compact representation technique, the Haar tree summarizes the data by successive layers of detail. The root node and the leaves contain the different levels of coefficients allowing to reconstruct the original data in terms of the coefficients in the root-to-leaf path.

Multi-dimensional wavelets are proposed in [3] as a summarization technique for approximate query processing in high-dimensional applications. The proposed approach starts employing multi-dimensional Haar wavelets to efficiently obtain effective, compact synopses of general relational tables. After the query execution is entirely processed in the wavelet coefficient domain, operating directly on the wavelet-coefficient synopses of relational tables, guarantees extremely fast response times. Experimental results with synthetic and real data sets support the effectiveness of the proposed approach.

In [10] is presented hierarchical one-pass wavelet-based approximations for signal summarization in small space. These techniques provide accurate representation of data and allow answering aggregates queries on the signal with reasonable accuracy. The proposed methods were evaluated under different types of aggregate queries in the context of telecommunications data.

The work in [16] advances a wavelet-inspired hierarchical synopsis with optimal error guarantees, superseding previous hierarchical summarization techniques, such as classical Haar wavelet synopses. The proposed one-pass Haar+ synopsis is constructed in time linear with the size of the data set for any monotonic distributive error metric.

2.4 Sketches

Sketch-based approaches are derived from wavelet methods. A *Sketch* is, essentially, a randomized linear projection of the underlying data vector and can be considered as a randomized version of wavelet techniques. Considering the space bounds limitations and the desirable accuracy, a sketch occupies very little space and allows quick and relative answers to queries. Furthermore, maintaining it as data is received is easy, allowing online updates and in one pass. This summarization method can be applied to approximate query estimation and to compute aggregate statistics.

The *Count-Min Sketch* for data stream summarization is introduced in [4]. This sublinear space data structure allows quick and accurate approximated answers for point, range and inner product queries. Moreover, it is also suitable to solve more elaborate data mining problems such as finding quantiles, frequent items and heavy hitters. A fundamental characteristic of such kind of data summarization is the improvements in time and space bounds when solving the problems mentioned above. The proposed summarization structure can be extended to solve problems on multi-dimensional streams.

2.5 Histograms

A histogram is a synopsis structure that allows accurate approximations of the underlying data distribution and provides a graphical representation of data. Histograms are widely applied to compute aggregate statistics, to approximate query answering, query optimization and selectivity estimation [13]. Besides, histograms are useful in a large variety of data mining applications, such as change detection and classification tasks. Moreover, for multidimensional problems it is feasible to construct histograms over multiple attributes capturing the joint frequency distributions accurately.

Consisting of a set of k non-overlapping intervals (also known as buckets or bins), a histogram is visualized as a bar graph that shows frequency data. The height of each bar drawn on each interval is proportional to the number of observed values within that interval.

There is a broad number of works proposing histograms as a feasible data summarization technique. The most important developments in this synopsis structure are discussed in [13]. Also [19] presents a study on histograms, proposing a taxonomy to capture existing histograms types and deriving new ones.

Often, literature identifies several classes of histograms:

- In **equi-width** histograms, the range of the variable is divided into k intervals of equal length (the range of the variable is equalized).
- **Equi-depth** or **equi-height** histograms are constructed so that the range of the observed variable is divided into k buckets such that the frequency of the values in all buckets should be equal.
- **V-optimal** histograms are defined as those that minimize the variance of the difference between the observed values and the approximations given by the corresponding assigned bucket. A V-optimal histogram is the histogram with the least variance of all histograms with the same number of buckets.

- In **maxDiff** histograms, after sorting the data, a bucket boundary is placed between two adjacent values, if the difference between them is one of the $k-1$ largest.
- **Compressed** histograms begin by storing in l ($l < k$) buckets the most frequent l source values. The remaining are divided into $k-l$ buckets as proposed by the equi-depth method.

To construct histograms in the stream mining context, there are some requirements that need to be fulfilled: the algorithms must be one-pass, supporting incremental maintenance of the histograms, and must be efficient in time and space [11, 12].

Therefore, when constructing online histograms from data streams there are two main characteristics to embrace:

- The updating facility.
- The error of the histogram.

MaxDiff and compressed histograms, usually, present small errors. In contrast, the equi-depth method could produce a histogram with high variance within intervals. Due to having the smallest variance possible among all the intervals, V-optimal histograms provide the smallest errors [19]. Although presenting the most accurate estimation of data, V-optimal histograms are very difficult to update as new data arrives. In fact, keeping a V-optimal histogram along with the data could imply reconstructing the histogram entirely instead of updating the existing one. Regarding practical issues, the work in [14] devoted efforts to improve the online maintenance of V-optimal histograms.

The construction of a maxDiff histogram requires a complete sort of the data. However, in practice, that could be impossible. Furthermore, the maintenance of such histogram is not straightforward. Taking into account the construction design, the update of a compressed histogram could be challenging and computationally demanding. Likewise, equi-depth histograms are also very complex to maintain. On the other hand, although can produce histograms with high variance, data updates in the equi-width strategy are effortless.

3 How to Remember?

To construct compact representations of data, histograms were selected among the other synopses structures. The understandable graphic representation, the clearness in tracking trends and in comparing data and the estimation of distribution of a random variable are the reasoning to select histograms. Furthermore, this paper proposes online equi-width histograms, in which the number of buckets is chosen under error constraints, to remember data that is being discarded. Despite not presenting the smallest error, the equi-width histograms were chosen based on the following reasons:

- The construction is effortless: it simply divides the range of the random variable into k non-overlapping intervals with equal width.
- The updating process is easy: each time a new data observation arrives, it just identifies the interval where it belongs and increments the count of that interval.

– Information visualization is simple: the value axis is divided into buckets of equal width.

3.1 Online Histograms under Error Constraints

A histogram provides a data summarization showing a graphical representation of the distribution of a random variable: the values of the random variable are placed into non-overlapping intervals and the height of the bar drawn on each interval is proportional to the number of observed values within that interval.

Let i be the current number of observations of a given variable X from which a histogram is being constructed. A histogram H_k is defined by a set of k buckets $B_1,\ldots,B_k$ in the range of the random variable and a set of frequency counts $F_1(i),\ldots,F_k(i)$.

Definition 1 Let k be the number of non-overlapping intervals of a histogram. For each observation x_i of a given variable X, the histogram counters are defined as:

$$C_j(i) = \begin{cases} 1, & x_i \in B_j \\ 0, & x_i \notin B_j \end{cases}, \quad \forall j = 1,\ldots,k \tag{1}$$

Along with the definition of the histogram counters, come the definitions of the histogram counts and histogram frequencies.

Definition 2 Let k be the number of non-overlapping intervals of a histogram. For each observation x_i of a given variable X, the histogram counts are defined as:

$$Ct_j(i) = \sum_{l=1}^{i} C_j(l), \quad \forall j = 1,\ldots,k \tag{2}$$

Definition 3 Let k be the number of non-overlapping intervals of a histogram. For each observation x_i of a given variable X, the histogram frequencies are defined as:

$$F_j(i) = \frac{Ct_j(i)}{i}, \quad \forall j = 1,\ldots,k \tag{3}$$

The set of k buckets of a histogram are defined by a set of break points $bp_1,\ldots,bp_{k+1}$, which divides the range of the variable, and by a set of middle break points $m_1,\ldots,m_k$.

Definition 4 Let k be the number of non-overlapping intervals of a histogram. The histogram buckets are defined as:

$$B_j = \left\{bp_j, bp_{j+1}, m_j\right\}, \quad \forall j = 1,\ldots,k \tag{4}$$

One of the main problems in using histograms is the definition of the number of buckets. A histogram with too many buckets is over detailed. On the other hand, if the histogram is constructed with too few buckets, important information may not be represented. Either way, a wrong number of the histogram buckets do not allow the underlying distribution of the data to be perceived.

Several rules exist to decide the number of buckets in a histogram. A rule that has been widely used is the Sturges's rule [23]: $k = 1 + log_2 n$, where k is the number of

intervals and n is the number of observed data points. This rule has been criticized because it implicitly uses a binomial distribution to approximate an underlying normal distribution. Sturges's rule has probably survived because, for moderate values of n (less than 200) produces reasonable histograms. Although, it does not work for a large number of observations. Alternative rules for constructing histograms include Scott's rule [21] for the class width: $h = 3.5sn^{-1/3}$ and Freedman and Diaconis's rule [7] for the class width: $h = 2(IQ)n^{-1/3}$ where s is the sample standard deviation and IQ is the sample interquartile range. However, these rules are not suitable to use in the context of open-ended data streams. In this context, all values of the variable are never observed, and therefore the total number of observations is unknown. Overcoming this drawback, the number of buckets in the online equi-width histograms proposed is defined establishing a bound on the mean square error of the histogram.

While estimating the probability distribution of a random variable, a histogram presents an error due to the reason that all the values of the variable within an interval are represented by the corresponding middle point. In each bucket, all the contained observations x_i are estimated by the corresponding middle point, which means that this approximation error is bounded by half of the width (W) of the interval:

$$x_i - m_j \leq \frac{W}{2}, \quad bp_j \leq x_i < bp_{j+1} \text{ and } \forall j = 1,..,k.$$

Considering an equi-width histogram, the range of the random variable, R, is divided into k non-overlapping intervals with equal width ($W = \frac{R}{k}$). For each observation x_i, the bound for the approximation error is:

$$x_i - m_j \leq \frac{R}{2k}, \quad bp_j \leq x_i < bp_{j+1}, \quad \forall j = 1,..,k. \tag{5}$$

Therefore, the error of representing a random variable by an equi-width histogram is defined as a function of this approximation error.

Definition 5 The square error[1] of the bucket B_j, defined by the interval $\{bp_j, bp_{j+1}\}$ and the middle break point m_j is given by:

$$SE(B_j) = \sum_{bp_j < x_l \leq bp_{j+1}} (x_l - m_j)^2, \quad \forall j = 1,\ldots,k. \tag{6}$$

Moreover, since a histogram is defined by a set of non-overlapping buckets and each data observation belongs only to one bucket, the total error of a histogram can be expressed as a sum of the bucket errors.

Definition 6 The mean square error of a histogram H_k with buckets $B_1,\ldots,B_k$, and a total of n observations, is the mean of the sum of the square errors along all the buckets:

$$MSE(H_k) = \frac{\sum_{j=1}^{k} SE(B_j)}{n} \tag{7}$$

[1] The square error is one of the most used error measures in histogram construction. It is also known as the V-Optimal measure and was introduced by [14].

Using equation 5, the square error of each bucket B_j is at most $C_j \frac{R^2}{4k^2}$:

$$SE(B_j) = \sum_{bp_j < x_l \leq bp_{j+1}} (x_l - m_j)^2 \leq \sum_{bp_j < x_l \leq bp_{j+1}} \left(\frac{R}{2k}\right)^2 = C_j \frac{R^2}{4k^2}, \forall j = 1, \ldots, k \quad (8)$$

where C_j is the count of bucket B_j, $\forall j = 1, \ldots, k$.

After simple algebraic manipulation, the mean square error of an equi-width histogram H_k is bounded, in the worst case, by $\frac{R^2}{4k^2}$:

$$MSE(H_k) = \frac{\sum_{j=1}^{k} SE(B_j)}{n} \leq \frac{\sum_{j=1}^{k} C_j \frac{R^2}{4k^2}}{n} = \frac{R^2}{4k^2} \quad (9)$$

Let ε be the admissible error for the mean square error of a histogram H_k with k buckets. Then, from equation 9, it turns out that $\frac{R^2}{4k^2} \leq \varepsilon$. And therefore, this guarantees that the mean square error of any equi-width histogram with at least $\frac{R}{2\sqrt{\varepsilon}}$ buckets is, at most, ε.

It must be stressed out that the constraint $k \geq \frac{R}{2\sqrt{\varepsilon}}$ does not guarantee that the equi-width histogram H_k is optimal, neither that, concerning the mean square error, k is the optimal number of buckets.

Figure 1 shows that the number of buckets linearly increases with the variable range (R) and when the admissible mean square error (ε) of the histogram decreases. Figure 1 (top) represents the number of buckets in function of ε and of R. The bottom figures give a projection of the number of buckets according to the variables ε and R (respectively).

Therefore, the input for the online equi-width histograms construction is the error parameter ε and the range of the variable. The range of the variable is only indicative and is used to define the non-overlapping intervals using an equi-width strategy and to establish the number of buckets in the histogram. Each time a new value of the variable is observed, the histogram is updated. The updating process determines the bucket corresponding to the observed value and increments the counts. These updates are processed online, performing a single scan over the data stream. It can process infinite sequences of data, processing each example in constant time and space. Keeping the histogram up with the incoming data only requires the set of buckets, the set of counts and the current number of observations seen so far to be stored in memory.

4 How to Forget?

In a dynamic environment, as new data is available, older observations are less useful. This stresses the need to forget outdated data, which is not describing the current state of the nature.

In this context, the forgetting approach is deeply related with the evolution of the process generating data. The forgetting process can be done abruptly or smoothly. In the former, the data seen so far is thrown away, which is known by a catastrophic forgetting of old data. This approach can be applied either after a fixed period of time

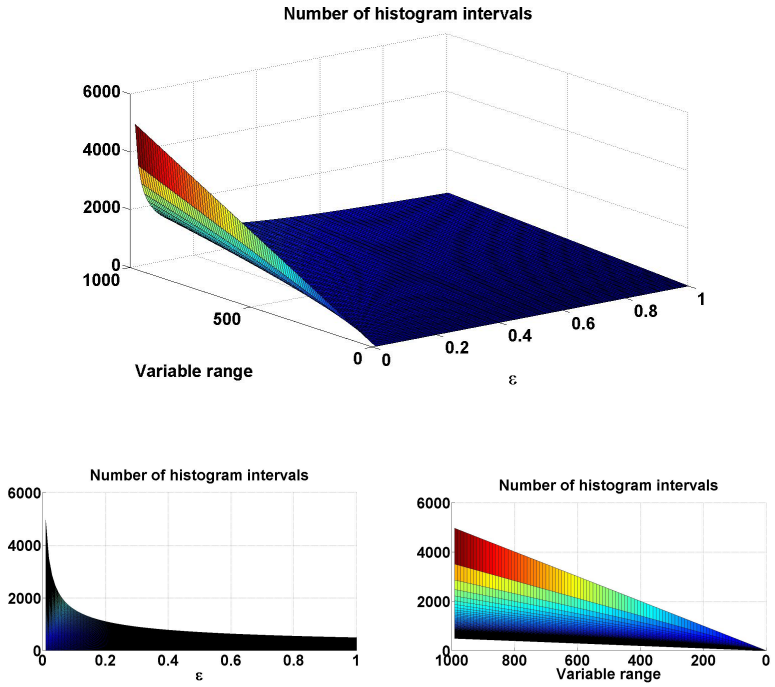

Fig. 1 Representation of the number of non-overlapping intervals. The top figure shows the dependency from the admissible error ε and the variable range R. Bottom figures show it according to only one variable.

or after the occurrence of a change. The second case, considers that although recent data is the most important, old data is still slightly related with the current state of nature. Within this approach, old data is forgotten gradually, using fading factors.

Evolving data streams are summarized using online histograms. With the construction of online histograms over a sliding window, a catastrophic forgetting of outdated data is achieved. On the other hand, a smooth forgetting mechanism is performed using fading histograms constructed over the entire stream.

4.1 Histograms over Sliding Windows

The most common approach to forgetting outdated data is sliding windows, where an algorithm is applied only to a small contiguous portion of the data set (window) which is moved (slide) along the data examples as they arrive. In a sliding windows approach, at each time step, the data distribution is approximated by a histogram constructed only from the observations that are included in the window. At every time i, a data record arrives and expires at time $i+w$, where w is the length of the window.

Histograms that are constructed over a sliding window, are, henceforth, referred to as sliding histograms. Sliding histograms allow attention to be focused solely on

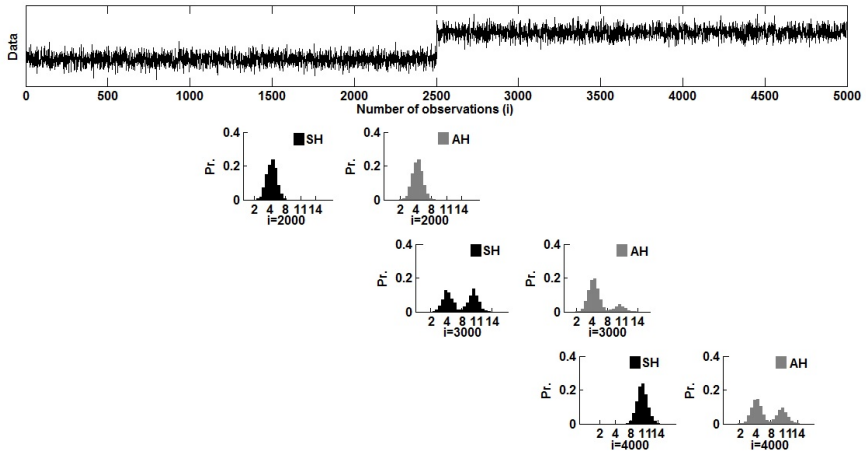

Fig. 2 Comparison of histograms constructed over a sliding window of length 1000 (SH) and over the entire stream (AH).

the most recent data, which captures the current state of nature and therefore the current properties of the process generating data.

Definition 7 Consider a sliding window (SW) of length w at observation x_i:

$$SW = \{x_l : l = i-w+1,\ldots,i\}.$$

The frequency counts of a sliding histogram (with k buckets) constructed at observation x_i over this window are defined by:

$$F_{w,j}(i) = \frac{\sum_{l=i-w+1}^{i} C_j(l)}{w}, \forall j = 1,\ldots,k, \tag{10}$$

To exemplify the forgetting ability of sliding histograms with respect to histograms constructed over the entire stream, artificial data was generated from two normal distributions. The initial 2500 observations follow a normal distribution with mean 5 and standard deviation 1 and the remaining 2500 observations follow a normal distribution with mean 10 and the same standard deviation.

For illustrative purposes, the number of buckets in each histogram was set to 20 (considering an admissible error $\varepsilon = 0.1$ for the mean square error of the histograms and using equation 9). With respect to the sliding histograms, a window of length 1000 was used.

Figure 2 (top) shows the artificial data with a change at observation 2500. The remaining plots display sliding histograms (constructed over a window of length 1000) and over the entire stream, at different observations: 2000, 3000 and 4000.

From the first representations, while in the presence of a stationary distribution, it turns out that both histograms produce similar representations of the data distribution. The size of 1000 examples of the sliding window is enough to capture the

behavior in this initial stage. The second and the third representations present a different scenario. At observations 3000, after the change occurred at observation 2500, the representations provided by both histograms strategies become quite different. It can be observed that in the sliding histogram representation, the buckets for the second distributions are reinforced, which does not occur on the histogram constructed over the entire stream. Furthermore, at observation 4000, in the histogram constructed over the entire stream, the buckets for the first distribution still filled, which is not in accordance with the current state of nature. While in the sliding histograms, those buckets are empty as a result of the fact that the sliding histogram is constructed only with the observations from a window on the current distribution.

However, to construct sliding histograms it is necessary to maintain in memory all the observations within the window, which could be costly depending on the applications and on the length of the window. Nevertheless, approaches based on sliding windows have been widely applied in summarization problems [1, 17]. The next section advances a memoryless approach to forgetting outdated data, which avoids keeping recent observations in the memory.

4.2 Fading Histograms

As stated before, with the previous approach a catastrophic forgetting is obtained, which in some applications could turn out to be excessive. As an alternative, using fading factors, data is forgotten gradually. In dynamic data streams, recent data is usually more important than old data. Therefore, the most feasible way to attribute different weights to data observations is an exponential approach, where the weight of data observations decreases exponentially with time. Exponential fading factors have been applied successfully in data stream evaluation [8].

Figure 3 illustrates the weight of examples according to their age, considering an exponential approach.

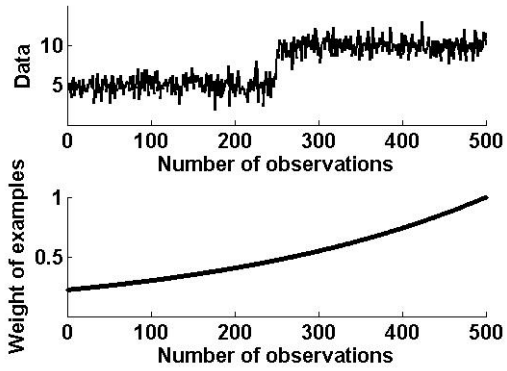

Fig. 3 The weight of examples as a function of age, in an exponential approach.

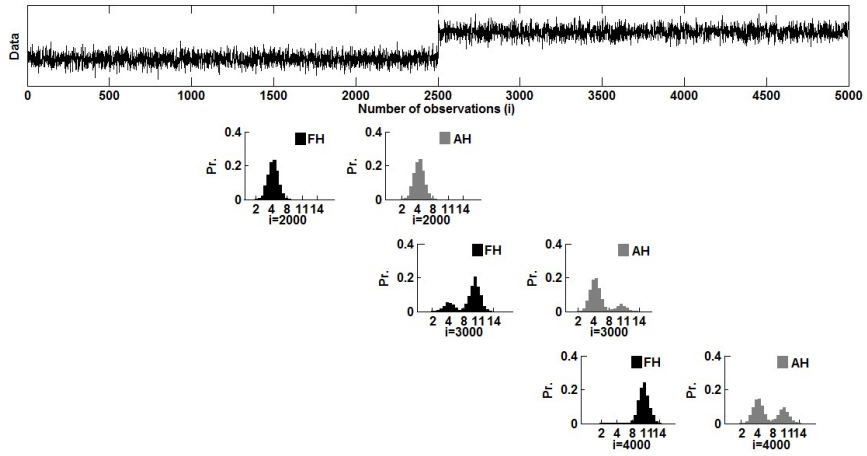

Fig. 4 Comparison of histograms computed with a fading factor of $\alpha = 0.997$ (FH) and histograms constructed over the entire stream (AH).

Following an exponential forgetting, histograms can be computed using fading factors, henceforth referred to as fading histograms. In this sense, data observations with high weight (the recent ones) contribute more to the fading histogram than observations with low weight (the old ones).

Definition 8 Let k be the number of buckets of a fading histogram. For each observation x_i of a given variable X, the α-frequencies are defined as:

$$F_{\alpha,j}(i) = \frac{\sum_{l=1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)}, \forall j = 1, \ldots, k, \tag{11}$$

where α is an exponential fading factor, such that $0 \ll \alpha < 1$.

According to this definition, old data is forgotten gradually, since it contributes less than recent data. Moreover, the recursive form enables the construction of fading histograms in the flow.

The forgetting ability of fading histograms with respect to histograms constructed over the entire stream is illustrated using the same artificial data as in figure 2. The number of buckets and the value of the fading factor for the fading histograms were set to 20 (the same settings as in Section 4.1) and $\alpha = 0.997$, respectively.

The top Figure 4 shows the artificial data with a change at observation 2500. The remaining plots display fading histograms (with a fading factor of $\alpha = 0.997$) and histograms constructed over the entire stream, at different observations: 2000, 3000 and 4000. The number of buckets in each histogram was set to 20 (the same settings as in Section 4.1).

These histograms are similar to those shown in Figure 2, and therefore the same conclusions can be taken for the fading histograms as those for the sliding histograms. Contrary to the histograms constructed over the entire stream, fading histograms capture the change better (as shown in the histograms produced at observation 3000), enhancing the fulfillment of the buckets for the second distribution. At observation 4000, it can be seen that the fading histogram produces a representation that keeps up with the current state of nature, forgetting outdated data (it must be pointed out that although they appear to be empty, the buckets for the first distribution present very low frequencies).

The fading factors are memoryless, an important property in streaming scenarios [8]. Hence, it turns out that fading histograms present a clear advantage over sliding histograms, which require all the observations inside the window to be maintained in memory. Nevertheless, fading histograms are, in fact, approximations of histograms constructed with data observations from a sliding window (for further details, see the Appendix 7). Moreover, there is a relation between the length of the sliding window and the value of the fading factor that should be used to approximate that window [20].

4.3 Representation Comparison

In order to compare sliding histograms with fading histograms, the same artificial data as in Figure 2 was used. With respect to the sliding histograms, a sliding window of length 1000 was used and to compute fading histograms, a fading factor of $\alpha = 0.997$ was used (from [20] it turns out that to approximate an estimate on a sliding window of length w with a fading estimate, the α must be set to $\alpha = \varepsilon^{\frac{1}{w}}$, where ε is the admissible approximation error). The number of buckets in each histogram was set to 20 (the same settings as in Section 4.1).

Figure 5 (top) shows artificial data with a change at observation 2500. The remaining plots display sliding histograms (constructed over a sliding window of length 1000) and fading histograms (with $\alpha = 0.997$) at different observations: 2000, 3000 and 4000.

From these representations, it is straightforward that both approaches to constructing histograms with a forgetting ability lead to similar results. Moreover, both histograms capture the change at observation 2500, as shown in the representations at observation 3000. However, it can be observed that the ability to forget outdated data is reinforced in the fading histogram, since the buckets from the initial distribution presented smaller values than the corresponding ones in the sliding window, while the buckets from the second distribution have higher values.

5 Experimental Comparison

This section provides an experimental comparison of the two proposed histograms to forget outdated data with online histograms constructed over the entire data, when applied to an artificial data set, a public data set and a real word problem.

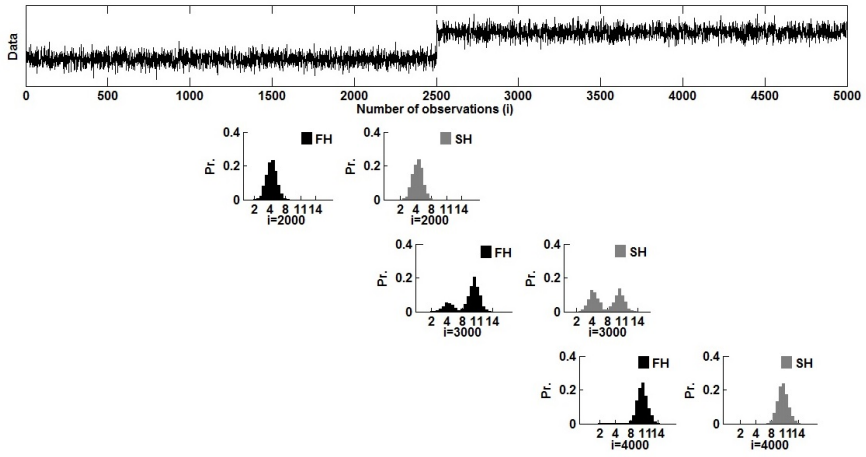

Fig. 5 Comparison of histograms constructed over a sliding window (of size 1000) and with fading factors ($\alpha = 0.997$).

5.1 Artificial Data

A comparison between the three representation methods was performed with the artificial data used in Section 4. Figure 6 (top) shows this artificial data, with a change at observation 2500. The bottom plot presents the average of histogram counts from a histogram constructed over the entire stream of data (solid thick line), from a sliding histogram (long dashed thick line) and from a fading histogram (dashed thick line).

In the presence of a change in data distributions, the advantage of using sliding or fading histograms instead of a histogram constructed with all the examples is obvious: both adapt their data representation to the occurred change, keeping a representation which is more faithful to the new state of the data distribution. Indeed, the histogram constructed over the entire stream of data, is not able to overcome the contributions of the initial state of data distribution, and therefore the representation of new data provided is corrupted by the initial examples. In this sense, it is quite difficult to perceive from this histogram representation the existence of a change in data distribution. In contrast, the representation provided by fading histograms make the change occurred in data distribution clear. From this example, it is easy to observe that the representation from fading histograms is able to reproduce the new state of data earlier than the representation constructed over sliding windows. Therefore, fading histograms provide representations of data more up-to-date than any of the others.

The advantages of fading histograms over sliding histograms and over histograms constructed with all the data examples are straightforward: they are able to adapt better to a new data distribution and they are memoryless with respect to sliding histograms, which is an important property in data stream context [8].

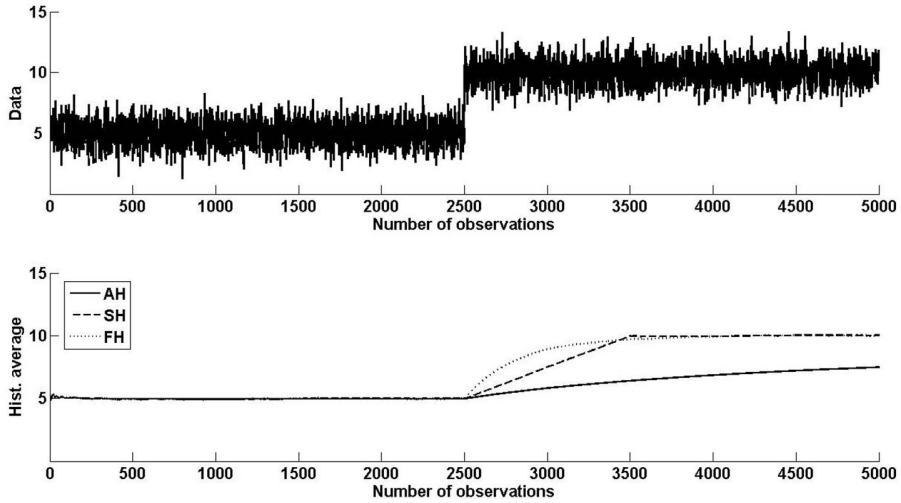

Fig. 6 Comparison of the average counts of different kinds of histograms applied to an artificial data set: a histogram constructed over the entire stream of data (solid thick line), a sliding histogram constructed over a sliding window of size 1000 (long dashed thick line) and a fading histogram computed with fading factor $\alpha = 0.997$ (dashed thick line).

5.2 SEA Concepts Data Set

In the previous experiment, the data set did not allow to compare the representations provided by the different histograms in large problems, which is important since concept drift mostly occurs in huge amounts of data arriving in the form of streams. To overcome this drawback, a comparison of the histograms proposed was performed using the SEA concepts data set [22], a benchmark problem for concept drift. Figure 7 (top) shows the error rate on this data set (computed using a naive-Bayes classifier), which presents three drifts (at examples 15k, 30k and 45k) that are represented by dashed lines.

The bottom plot presents the average of histogram counts from a histogram constructed over the entire stream of data (solid thick line), from a sliding histogram (long dashed thick line) and from a fading histogram (dashed thick line). The sliding histogram was constructed over a sliding window of length $5k$ and the fading histogram was computed using a fading factor of value 0.9994.

As it can be observed in the bottom Figure 7, with the histogram constructed over the entire data the drifts are not distinguishable. In opposite, both forgetting histograms are able to exhibit the drifts. For the three drifts, is notable that the average of the fading histogram is higher than the average of the sliding histogram, which indicates that the fading histogram represents the data better.

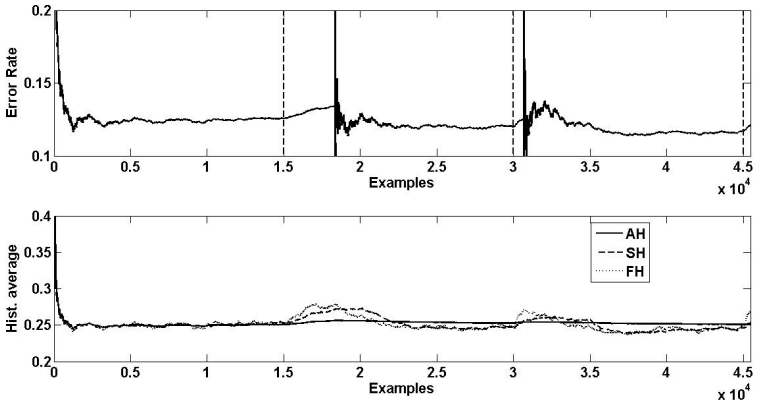

Fig. 7 Evolution of the error rate in the SEA concepts data set and the average counts of different kinds of histograms: a histogram constructed over the entire stream of data (solid thick line), a sliding histogram constructed over a sliding window of size 5000 (long dashed thick line) and a fading histogram computed with fading factor $\alpha = 0.9994$ (dashed thick line).

5.3 Industrial Data Set

This section presents the comparison of histograms representations from a real word problem. This industrial data set was obtained within the scope of the work presented in [6], with the objective of designing different machine learning classification methods for predicting surface roughness in high-speed machining. Data was obtained by performing tests in a Kondia HS1000 machining center equipped with a Siemens 840D open-architecture CNC. The blank material used for the tests was 170 $\times$ 100 $\times$ 25 aluminum samples with hardness ranging from 65 to 152 Brinell, which is a material commonly used in automotive and aeronautical applications. These tests were done with different cutting parameters, using sensors for registry vibration and cutting forces. A multi-component dynamometer with an upper plate was used to measure the in-process cutting forces and piezoelectric accelerometers in the X and Y axis for vibrations measures. Each record includes information on several variables used in a cutting process and the measurements for each test were saved individually.

For change detection purposes, the measurements of the cutting speed on X axes from 5 tests were joined sequentially in order to have only one data set with 4 changes with different magnitudes and sudden and low rates, at examples 45k, 90k, 210k and 255k.

Figure 8 (top) shows this data set. The bottom figure shows the average of histogram counts from a histogram constructed over the entire stream of data (solid thick line), from a sliding histogram (long dashed thick line) and from a fading histogram (dashed thick line). The sliding histogram was constructed over a sliding window of length $5k$ and the fading histogram was computed using a fading factor of value 0.9994.

It can be observed that with the histogram constructed over the entire data, only the first change is noticeable. However, either sliding or fading histograms, are able to represent the four changes. This indicates that both produce more appropriate representations of the data under analysis than the histogram constructed with all the data. Even in the third change, which is a change with a small magnitude rate, the histograms with the forgetting ability are able to assigned it. In conclusion, sliding and fading histograms provide more faithful representations of the data.

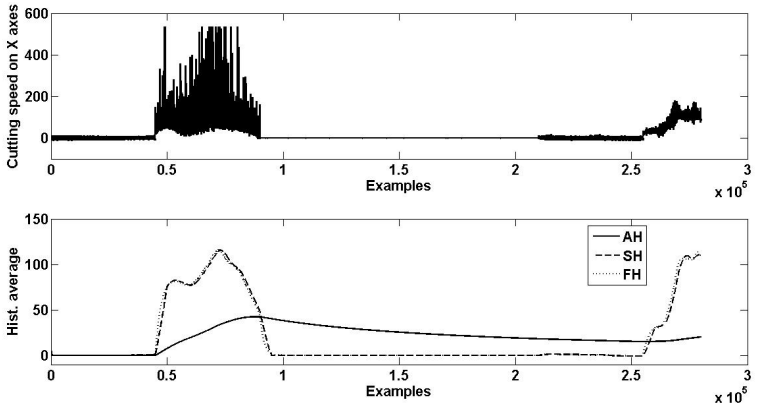

Fig. 8 Industrial data set and comparison of the average counts of different kinds of histograms: a histogram constructed over the entire stream of data (solid thick line), a sliding histogram constructed over a sliding window of size 5000 (long dashed thick line) and a fading histogram computed with fading factor $\alpha = 0.9994$ (dashed thick line).

6 Conclusions

This paper discusses the problem of data summarization from open-ended data streams, proposing approaches to remember important properties of data streams and to forget outdated data.

Online histograms under mean square error constraints are presented to remember properties of data. The advantage of this summarization structure is bi-fold: it constructs a compact representation of data and provides a visual interpretation of the underlying distribution.

Nowadays, it is not feasible to consider that data is collected in a static scenario. Under dynamic environments, a compact representation of data must forget outdated data in order to keep up with the actual state of evolving nature. To deal with non-stationary data streams, besides remembering data that is being discarded after it is processed, the online histograms must also forget data that is not describing the actual state of nature. To accomplish this target, online histograms are constructed over sliding windows or using fading factors. The former strategy enforces an abrupt

forgetting while the second approach, which has the advantage of being memoryless, lets outdated data slip out gradually.

Acknowledgements The work of Raquel Sebastião was supported by FCT (Portuguese Foundation for Science and Technology) under the PhD Grant SFRH/BD/41569/2007.

The authors acknowledge the support of the European Commission through the project MAESTRA (Grant number ICT-2013-612944).

This work was also funded by the European Regional Development Fund through the COMPETE Program, by the Portuguese Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701, and by the Projects NORTE-07-0124-FEDER-000056/000059 which is financed by the North Portugal Regional Operational Program (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF).

7 Appendix: Fading Histograms

This appendix presents some computations on the error of approximating the fading sliding histogram with the fading histogram. Considering the definition of histogram frequencies (1), the frequencies of a sliding histogram (with k buckets) constructed over a sliding window of length w and computed at observation i with an exponential fading factor α ($0 \ll \alpha < 1$), can be defined as follow:

$$F_{\alpha,w,j}(i) = \frac{\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)}, \forall j = 1,\ldots,k, \tag{12}$$

To approximate a fading sliding histogram by a fading histogram, the older data than that within the most recent window $W = \{x_l : l = i-w+1,\ldots,i\}$ must be taken into consideration. Therefore, for each bucket $j = 1,\ldots,k$, the proportion of weight given to old observations (with respect to W) in the computation of the fading histogram is defined as the bucket ballast weight:

$$B_{\alpha,w,j}(i) = \frac{\sum_{l=w}^{i-1} \alpha^{l}}{N_\alpha(i)}, \forall j = 1,\ldots,k, \tag{13}$$

where $N_\alpha(i)$ is the fading increment defined as $N_\alpha(i) = \sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)$.

As with the old observations, for each bucket $j = 1,\ldots,k$, the proportion of weight given to observations within the most recent window W is defined by:

$$B'_{\alpha,w,j}(i) = 1 - B_{\alpha,w,j}(i) = \frac{\sum_{l=0}^{w-1} \alpha^{l}}{N_\alpha(i)}, \forall j = 1,\ldots,k. \tag{14}$$

Hence, the error of approximating the fading sliding histogram with the fading histogram, both with k buckets, can be defined as:

$$\Delta_{\alpha,w}(i) = \sum_{j=1}^{k} \Delta_{\alpha,w,j}(i) = \sum_{j=1}^{k} \left\| F_{\alpha,w,j}(i) - F_{\alpha,j}(i) \right\|. \tag{15}$$

Theorem 1 *Let $\varepsilon < 1$ be an admissible ballast weight for the fading histogram. Then, $\Delta_{\alpha,w}(i) \leq 2\varepsilon$.*

Proof From the respective histogram frequencies definitions comes that the approximation error in each bucket is:

$$\Delta_{\alpha,w,j}(i) = \left\| \frac{\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)} - \frac{\sum_{l=1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)} \right\|, \forall j = 1, \ldots, k$$

Splitting each of these errors considering the frequencies inside and outside the most recent window of size w:

$$\Delta_{\alpha,w,j}(i) = \left\| \Delta_{\alpha,w,j}(i)^{in} - \Delta_{\alpha,w,j}(i)^{out} \right\|,$$

where

$$\Delta_{\alpha,w,j}(i)^{in} = \frac{\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)} - \frac{\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)},$$

and

$$\Delta_{\alpha,w,j}(i)^{out}(i) = \frac{\sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)}.$$

Looking for an upper bound on the error, the worst case scenario is that these two sources of error do not cancel out, rather adding up their effect:

$$\Delta_{\alpha,w,j}(i) \leq \left\| \Delta_{\alpha,w,j}(i)^{in} \right\| + \left\| \Delta_{\alpha,w,j}(i)^{out} \right\|.$$

Hence

$$\Delta_{\alpha,w,j}(i) \leq \left\| \frac{\left(\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)\right)\left(\sum_{j=1}^{k} \sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)\right) - \left(\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)\right)\left(\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)\right)}{\left(\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)\right)\left(\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)\right)} \right\| + \left\| \frac{\sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)} \right\| \Leftrightarrow$$

$$\Leftrightarrow \Delta_{\alpha,w,j}(i) \leq \left\| \frac{\left(\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)\right)\left(\sum_{j=1}^{k} \sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)\right)}{\left(\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l)\right)\left(\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)\right)} \right\| + \left\| \frac{\sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)} \right\| \Leftrightarrow$$

$$\Leftrightarrow \Delta_{\alpha,w,j}(i) \leq \left\| \frac{\left(\sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l) \right) \left(\sum_{j=1}^{k} \sum_{l=1}^{i-w} \alpha^{i-l} C_j(l) \right)}{\left(\sum_{j=1}^{k} \sum_{l=i-w+1}^{i} \alpha^{i-l} C_j(l) \right) N_\alpha(i)} \right\| + \left\| \frac{\sum_{l=1}^{i-w} \alpha^{i-l} C_j(l)}{N_\alpha(i)} \right\|$$

The upper bound on the error is given by considering all $C_j(l) = 1$:

$$\Delta_{\alpha,w,j}(i) \leq \left\| \frac{\left(\sum_{l=i-w+1}^{i} \alpha^{i-l} \right) \left(k \sum_{l=1}^{i-w} \alpha^{i-l} \right)}{\left(k \sum_{l=i-w+1}^{i} \alpha^{i-l} \right) N_\alpha(i)} \right\| + \left\| \frac{\sum_{l=1}^{i-w} \alpha^{i-l}}{N_\alpha(i)} \right\| = 2 \left\| \frac{\sum_{l=1}^{i-w} \alpha^{i-l}}{N_\alpha(i)} \right\|$$

Then, from bucket ballast weight definition comes that:

$$\Delta_{\alpha,w,j}(i) \leq 2 \left\| B_{\alpha,w,j}(i) \right\|$$

Considering in each bucket $j = 1, \ldots, k$ an admissible ballast weight, at most, of ε/k comes that:

$$\Delta_{\alpha,w}(i) = \sum_{j=1}^{k} \Delta_{\alpha,w,j}(i) \leq \sum_{j=1}^{k} 2\varepsilon/k = 2\varepsilon.$$

References

1. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '02, pp. 1–16. ACM, New York, NY, USA (2002). DOI 10.1145/543613.543615. URL http://doi.acm.org/10.1145/543613.543615
2. Barbará, D.: Requirements for clustering data streams. SIGKDD Explor. Newsl. **3**(2), 23–27 (2002). DOI 10.1145/507515.507519. URL http://doi.acm.org/10.1145/507515.507519
3. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate query processing using wavelets. In: A.E. Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, K.Y. Whang (eds.) VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, pp. 111–122. Morgan Kaufmann (2000)
4. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms **55**(1), 58–75 (2005). DOI 10.1016/j.jalgor.2003.12.001. URL http://dx.doi.org/10.1016/j.jalgor.2003.12.001
5. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. ACM Trans. Database Syst. **30**(1), 249–278 (2005). DOI 10.1145/1061318.1061325. URL http://doi.acm.org/10.1145/1061318.1061325
6. Correa, M., Bielza, C., Pamies-Teixeira, J.: Comparison of bayesian networks and artificial neural networks for quality detection in a machining process. Expert Syst. Appl. **36**(3), 7270–7279 (2009). URL http://dblp.uni-trier.de/db/journals/eswa/eswa36.html#CorreaBP09
7. Freedman, D., Diaconis, P.: On the histogram as a density estimator:L2 theory. Probability Theory and Related Fields **57**(4), 453–476 (1981). DOI 10.1007/BF01025868. URL http://dx.doi.org/10.1007/BF01025868

8. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3), 317–346 (2013)
9. Gibbons, P.B., Matias, Y.: Synopsis data structures for massive data sets. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 909–910 (1999). DOI 10.1145/314500.315083
10. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: One-pass wavelet decompositions of data streams. IEEE Trans. on Knowl. and Data Eng. **15**(3), 541–554 (2003). DOI 10.1109/TKDE.2003.1198389. URL http://dx.doi.org/10.1109/TKDE.2003.1198389
11. Guha, S., Koudas, N., Shim, K.: Approximation and streaming algorithms for histogram construction problems. ACM Trans. Database Syst. **31**(1), 396–438 (2006). DOI 10.1145/1132863.1132873. URL http://doi.acm.org/10.1145/1132863.1132873
12. Guha, S., Shim, K., Woo, J.: Rehist: Relative error histogram construction algorithms. In: In proceedings of the 30th International Conference on. Very Large Data Bases, pp. 300–311 (2004)
13. Ioannidis, Y.: The history of histograms (abridged). In: Proceedings of the 29th international conference on Very large data bases - Volume 29, VLDB '03, pp. 19–30. VLDB Endowment (2003). URL http://dl.acm.org/citation.cfm?id=1315451.1315455
14. Ioannidis, Y.E., Poosala, V.: Balancing histogram optimality and practicality for query result size estimation. In: M.J. Carey, D.A. Schneider (eds.) Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995, pp. 233–244. ACM Press (1995)
15. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. In: Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB '98, pp. 275–286. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998). URL http://dl.acm.org/citation.cfm?id=645924.671191
16. Karras, P., Mamoulis, N.: Hierarchical synopses with optimal error guarantees. ACM Transactions on Database Systems **33**, 1–53 (2008). DOI 10.1145/1386118.1386124
17. Lin, M.Y., Hsueh, S.C., Hwang, S.K.: Interactive mining of frequent itemsets over arbitrary time intervals in a data stream. In: Proceedings of the nineteenth conference on Australasian database - Volume 75, ADC '08, pp. 15–21. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2007). URL http://dl.acm.org/citation.cfm?id=1378307.1378315
18. Misra, J., Gries, D.: Finding Repeated Elements. Science of Computer Programming **2**, 143–152 (1982). DOI 10.1016/0167-6423(82)90012-0
19. Poosala, V., Ioannidis, Y.E., Haas, P.J., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. In: SIGMOD Conference, pp. 294–305 (1996)
20. Rodrigues, P., Gama, J., Sebastião, R.: Memoryless fading windows in ubiquitous settings. In: Proceedings of Ubiquitous Data Mining (UDM) Workshop, in conjunction with the 19th European Conference on Artificial Intelligence - ECAI 2010, pp. 27–32 (2010)
21. Scott, D.W.: On optimal and data-based histograms. Biometrika **66**(3), 605–610 (1979). DOI 10.1093/biomet/66.3.605. URL http://dx.doi.org/10.1093/biomet/66.3.605
22. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382. ACM Press (2001)
23. Sturges, H.A.: The choice of a class interval. American Statistical Association **21**, 65–66 (1926)
24. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. **11**(1), 37–57 (1985). DOI 10.1145/3147.3165. URL http://doi.acm.org/10.1145/3147.3165