Monitoring Incremental Histogram Distribution for Change Detection in Data Streams

Raquel Sebastião[1,2], João Gama[1,3]
Pedro Pereira Rodrigues[1,2,4], and João Bernardes[4,5]

[1] LIAAD - INESC Porto, L.A. Rua de Ceuta, 118, 6
4050-190 Porto, Portugal
[2] Faculty of Science, University of Porto
[3] Faculty of Economics, University of Porto
[4] Faculty of Medicine, University of Porto
[5] INEB, Porto
{raquel,jgama}@liaad.up.pt
{pprodrigues,joaobernardes}@med.up.pt

Abstract. Histograms are a common technique for density estimation and they have been widely used as a tool in exploratory data analysis. Learning histograms from static and stationary data is a well known topic. Nevertheless, very few works discuss this problem when we have a continuous flow of data generated from dynamic environments.

The scope of this paper is to detect changes from high-speed time-changing data streams. To address this problem, we construct histograms able to process examples once at the rate they arrive. The main goal of this work is continuously maintain a histogram consistent with the current status of the nature. We study strategies to detect changes in the distribution generating examples, and adapt the histogram to the most recent data by forgetting outdated data. We use the Partition Incremental Discretization algorithm that was designed to learn histograms from high-speed data streams.

We present a method to detect whenever a change in the distribution generating examples occurs. The base idea consists of monitoring distributions from two different time windows: the reference window, reflecting the distribution observed in the past; and the current window which receives the most recent data. The current window is cumulative and can have a fixed or an adaptive step depending on the distance between distributions. We compared both distributions using Kullback-Leibler divergence, defining a threshold for change detection decision based on the asymmetry of this measure.

We evaluated our algorithm with controlled artificial data sets and compare the proposed approach with nonparametric tests. We also present results with real word data sets from industrial and medical domains. Those results suggest that an adaptive window's step exhibit high probability in change detection and faster detection rates, with few false positives alarms.

Key words: Change detection, Data streams, Machine learning, Learning histograms, Monitoring data distribution, Adaptive Cumulative Windows

1 Introduction

Nowadays, the scenario of finite stored data sets is no longer appropriated because information is gathered assuming the form of transient and infinite data streams. As a large massive amount of information is produced at a high-speed rate it is no longer possible to use algorithms which require to store, in the main memory, the full historic data. In Data Streams the data elements are continuously received, treated and discarded. In this context processing time, memory and sample size are the crucial constraints in knowledge discovery systems [3]. Due to the exploratory nature of data and to time restrictions, a user may prefer a fast but approximate answer to an exact but slow answer. Methods to deal with these issues consist of applying synopsis techniques, such as histograms [12, 16, 19, 26], sketches [8] and wavelets [7, 15]. Histograms are one of the techniques used in data stream management systems to speed up range queries and selectivity estimation (the proportion of tuples that satisfy a query), two illustrative examples where fast but approximate answers are more useful than slow and exact ones.

In the context of open-ended data streams, as we never observe all values of the random variable, it is not appropriate to use the traditional histograms to construct a graphical representation of continuous data, because they require the knowledge of all data. Thus, there is still missing algorithms to address conveniently this issue. The Partition Incremental Discretization [12, 26] and the V-Optimal Histograms [14, 16, 18] are two examples. A key characteristic of a data stream is its dynamic nature. The process generating data is not strictly stationary and evolves over time. The target concept may gradually change over time. Moreover, when data is collected over time, at least for large periods of time, it is not acceptable to assume that the observations are generated at random according to a stationary probability distribution. Several methods in machine learning have been proposed to deal with concept drift [11, 17, 21, 23, 26, 28, 29]. Drifting concepts are often handled by time windows or weighted examples according to their age or utility. Another approach to detect drift concepts is monitoring distributions on two different time windows, monitoring the evolution of a statistical function between two distributions: from past data in a reference window and in a current window with the most recent data points [20, 27].

1.1 Previous Work

In a previous work [27], we presented a method to detect changes in data streams. In that work, we constructed histograms using the two layer structure of the Partition Incremental Discretization (PiD) algorithm and addressed the detection

problem by monitoring distributions using a fixed window model. In this work, we propose a new definition of the number of histogram's bins and the use of an adaptive-cumulative window model to detect changes. We also perform studies on the distance measures and advance a discrepancy measure based on the asymmetry of the Kullback-Leibler Divergence (KLD). We support this decision in previous results. The results of [27] suggest that the KLD achieve faster detection rates than the other tested distances measures (a measure based on entropy and the cosine distance).

1.2 Motivation, Challenges and Paper Outline

The motivation for studying time-changing high-speed data streams comes from the emergence of temporal applications such as communications networks, web searches, financial applications, and sensor data, which produces massive streams of data. Since it is impractical to store completely in memory all data, new algorithms are needed to process data online at the rate it is available. Another challenge is to create compact summaries of data streams. Histograms are in fact compact representations for continuous data. They can be used as a component in more sophisticated data mining algorithms, like decision trees [17].

As the distribution underlying the data elements may change over time, the development of methods to detect when and how the process generating the stream is evolving, is the main challenge of this study. The main contribution of this paper is a new method to detect changes when learning histograms using adaptive windows. We are able to detect a time window where change has occurred. Another contribution is an improved technique to initialize histograms satisfying user constraint on the admissible relative error.

The proposed method has potential use in medical and industrial domains, namely, in monitoring biomedical signals and production processes (respectively).

The paper is organized as follows. The next section presents an algorithm to continuously maintain histograms over a data stream. In section 3 we extend the algorithm for change detection. Section 4 presents preliminary evaluation of the algorithm in benchmark datasets and real-world problems. Last section concludes the paper and presents some future research lines.

2 Histograms

Histograms are one of the most used tools in exploratory data analysis. They present a graphical representation of data, providing useful information about the distribution of a random variable. A histogram is visualized as a bar graph that shows frequency data. The basic algorithm to construct histograms consists of sorting the values of the random variable and places them into *bins*. Next counts the number of data samples in each bin. The height of the bar drawn on the top of each bin is proportional to the number of observed values in that bin.

A histogram is defined by a set of k non-overlapping intervals and each interval is defined by its boundaries and a frequency count. The most used histograms

are either *equal width*, where the range of observed values is divided into k intervals of equal length $(\forall i, j : (b_i - b_{i-1}) = (b_j - b_{j-1}))$, or *equal frequency*, where the range of observed values is divided into k bins such that the counts in all bins are equal $(\forall i, j : (f_i = f_j))$.

When all the data is available, there are exact algorithms to construct histograms [25]. All these algorithms require a user defined parameter k, the number of bins. Suppose we know the range of the random variable (domain information) and the desired number of intervals k. The algorithm to construct equal width histograms traverses the data once; whereas in the case of equal frequency histograms a sort operation is required.

One of the main problems of using histograms is the definition of the number of intervals. A rule that has been used is the Sturges' rule: $k = 1 + log_2 n$, where k is the number of intervals and n is the number of observed data points. This rule has been criticized because it is implicitly using a binomial distribution to approximate an underlying normal distribution [6]. Sturges rule has probably survived because, for moderate values of n (less than 200) produces reasonable histograms. However, it does not work for large n. Scott gave a formula for the optimal histogram bin width which asymptotically minimizes the integrated mean square error. Since the underlying density is usually unknown, he suggested using the Gaussian density as a reference standard, which leads to the data-based choice for the bin width of $a \times s \times n^{-1/3}$, where $a = 3.49$ and s is the estimate of the standard deviation.

In exploratory data analysis, histograms are used iteratively. The user tries several histograms using different values of k (the number of intervals), and chooses the one that better fits his purposes.

2.1 The Partition Incremental Discretization (PID)

The *Partition Incremental Discretization* algorithm (*PiD* for short) that designed to provide a histogram representation of high-speed data streams. It learns histograms using an architecture composed by two layers. The first simplifies and summarizes the data, the algorithm transverses the data once and incrementally maintains an equal-width discretization; the second layer constructs the final histogram using only the discretization of the first phase. The first layer is initialized without seeing any data. As described in [12], the input for the initialization phase is the number of intervals (that should be much larger than the desired final number of intervals) and the range of the variable.

Consider a sample $x_1, x_2, ...$ of an open-ended random variable with range R. In this context, and allowing to consider extreme values and outliers, the histogram is defined as a set of break points $b_1, ..., b_{k-1}$ and a set of frequency

[6] Alternative rules for constructing histograms include Scott's (1979) rule for the class width: $k = 3.5sn^{-1/3}$ and Freedman and Diaconis's (1981) rule for the class width: $k = 2(IQ)n^{-1/3}$ where s is the sample standard deviation and IQ is the sample interquartile range.

counts $f_1, ..., f_{k-1}, f_k$ that define k intervals in the range of the random variable:

$$]-\infty, b_1],]b_1, b_2], ...,]b_{k-2}, b_{k-1}],]b_{k-1}, \infty[. \quad (1)$$

In a histogram, all x_i in a bin is represented by the correspondent middle point, which means that this approximation error is bounded by half of the length (L) of the bin. As the first layer is composed by equal-width histogram, we obtain:

$$x_i - m_j \leq \frac{L}{2} = \frac{R}{2k}, b_j \leq x_i < b_{j+1}, \forall j = 1, .., k. \quad (2)$$

Considering the set of middle-break points $m_1, ..., m_k$ of the histogram, we define the mean square error in each bin as the sum of the square differences between each point in that bin and their correspondent middle-break points: $\sum_i (x_i - m_j)^2 \leq n_j R^2/4k^2$, $b_j \leq x_i < b_{j+1}$, $\forall j = 1, .., k$ and n_j is the number of data points in each bin. The quadratic error (QE) is defined as the sum of this error along all bins:

$$QE(k) = \sum_{j}^{k} \sum_{i} (x_i - m_j)^2. \quad (3)$$

From the above equations it follows that the quadratic error is bounded, in the worst case, by: $nR^2/4k^2$, where n denotes the number of observed variables.

The definition of the number of intervals is one of the main problems of using histograms. The number of bins is directly related with the quadratic error. How different would the quadratic error be if we consider just one more bin? To study the evaluation of the quadratic error of a histogram with the number of bins, we compute the following ratio, which we refer to as the relative error: $\epsilon = \frac{QE(k)}{QE(k+1)}$.

In order to bound the decrease of the quadratic error, we define the number of bins of the first layer as dependent on the upper bound of the relative error (ϵ) and on the fail probability (δ):

$$N_1 = O(\frac{1}{\epsilon} ln \frac{1}{\delta}). \quad (4)$$

Establishing a bound for relative error, this definition of the number of bins ensures that the fail probability will converge to zero when N_1 increases. So, setting ϵ and δ and using this definition we control the decrease of the quadratic error. Figure 1 shows that the number of bins increases when the error decreases and the confidence increases. Figure 1 (top) represents the number of bins of $layer_1$ in function of ϵ and δ. The bottom figures give a projection of the number of bins according with the variables ϵ and δ (respectively).

So, differing from [12] the input for the initialization phase is a pair of parameters (that will be used to express accuracy guarantees) and the range of the variable:

- The upper bound on relative error ϵ.
- The desirable confidence level $1 - \delta$.
- The range of the variable.

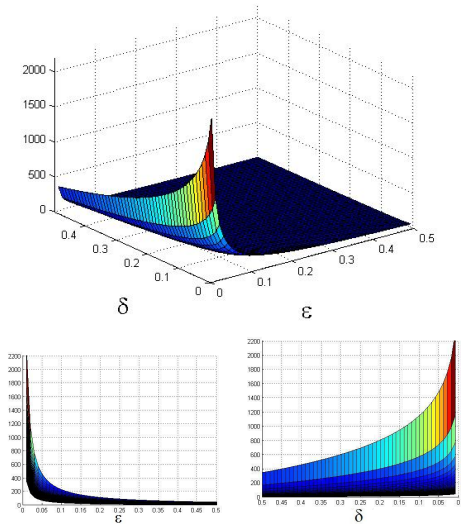


Fig. 1. Representation of the number of bins of $layer_1$. The top figure shows the dependency from ϵ and δ and bottom figures show it according to only one variable.

The range of the variable is only indicative. It is used to initialize the set of breaks using an equal-width strategy. Each time we observe a value of the random variable, we update $layer_1$. The update process determines the interval corresponding to the observed value, and increments the counter of this interval. The process of updating $layer_1$ works online, performing a single scan over the data stream. It can process infinite sequences of data, processing each example in constant time and space. The second layer merges the set of intervals defined by the first layer. The input for the second layer is the breaks and counters of $layer_1$, the type of histogram (equal-width or equal-frequency) and the desirable final number of intervals. The algorithm for the $layer_2$ is very simple. For equal-width histograms, it first computes the breaks of the final histogram, from the actual range of the variable (estimated in $layer_1$). The algorithm traverses the vector of breaks once, adding the counters corresponding to two consecutive breaks. For equal-frequency histograms, we first compute the exact number F of points that should be in each final interval (from the total number of points and the number of desired intervals). The algorithm traverses the vector of counters of $layer_1$ adding the counts of consecutive intervals till F. The computational costs of this phase can be ignored: it traverses once the discretization obtained in the first phase. We can construct several histograms using different number of intervals and different strategies: equal-width or equal-frequency. This is the main advantage of PiD in exploratory data analysis. We use PiD algorithm to create compact summaries of data, and along with the improvement of the number of bins definition, we also accomplished it with a change detection technique.

3 Change Detection

The algorithm described in the previous section assumes that the observations come from a stationary distribution. When data flows over time, and at least for large periods of time, it is not acceptable to assume that the observations are generated at random according to a stationary probability distribution. At least in complex systems and for large time periods, we should expect changes in the distribution of the data.

3.1 Related Work

When monitoring a stream is fundamental to know if the received data comes from the distribution observed so far. It is necessary to perform tests in order to determine if there is a change in the underlying distribution. The null hypothesis is that the previously seen values and the current observed values come from the same distribution. The alternative hypothesis is that they are generated from different continuous distributions.

There are several methods in machine learning to deal with changing concepts [21–23, 29]. In general, approaches to cope with concept drift can be classified into two categories: *i*) approaches that adapt a learner at regular intervals without considering whether changes have really occurred; *ii*) approaches that first detect concept changes, and next, the learner is adapted to these changes. Examples of the former approaches are *weighted examples* and *time windows* of fixed size. Weighted examples are based on the simple idea that the importance of an example should decrease with time (references about this approach can be found in [22, 24, 29]). When a time window is used, at each time step the learner is induced only from the examples that are included in the window. Here, the key difficulty is how to select the appropriate window's size: a small window can assure a fast adaptability in phases with concept changes but in more stable phases it can affect the learner performance, while a large window would produce good and stable learning results in stable phases but can not react quickly to concept changes.

In the latter approaches, with the aim of detecting concept changes, some indicators (e.g. performance measures, properties of the data, etc.) are monitored over time (see [21] for a good classification of these indicators). If during the monitoring process a concept drift is detected, some actions to adapt the learner to these changes can be taken. When a time window of adaptive size is used these actions usually lead to adjusting the window's size according to the extent of concept drift [21]. As a general rule, if a concept drift is detected the window's size decreases; otherwise the window's size increases.

Windows Models Most of the methods in this approach monitor the evolution of a distance function between two distributions: from past data in a *reference window* and in a *current window* of the most recent data points. An example of this approach, in the context of learning from Data Streams, has

been present by [20]. The author proposes algorithms (statistical tests based on Chernoff bound) that examine samples drawn from two probability distributions and decide whether these distributions are different.

In this work, we monitor the distance between the distributions in two time windows: a reference window that has a fixed size and refers to past observations and an adaptive-cumulative window that receives the actual observations and could have a fixed or an adaptive step depending on the distance between distributions. For both windows, we compute the relative frequencies: a set of empirical probabilities $p(i)$ for the reference window and $q(i)$ for the adaptive-cumulative window.

Adaptive-Cumulative Window Model In a previous work [27] we defined the windows sizes as dependent on the number of intervals of the $layer_1$, being half of these ones: $\frac{N_1}{2}$. In this work, in order to evaluate the influence of the number of examples required to detect a change, we defined the cumulative window (the current one) using an adaptive increasing step that depends on the distance between data distributions. Starting with a size of $\frac{N_1}{2}$, the step is incremented if the distance between data distributions increases and is decremented otherwise, according to the following relation:

$$WindowStep = \frac{N_1}{2}(1 - \frac{1}{\alpha} * |KLD(p||q) - KLD(q||p)|),^{7} \quad (5)$$

where α is related with the change detection threshold (introduced further in this paper).

Figure 2 shows the dependency of the window's step on distributions' distance.

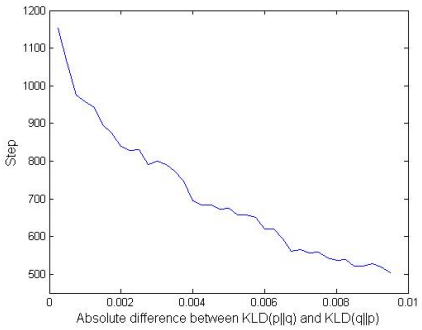


Fig. 2. Representation of the windows' step with respect to the absolute difference between $KLD(p||q)$ and $KLD(q||p)$. An illustrative example showing that the window's step decreases when the absolute difference between distances increases.

[7] KLD stands for Kullback-Leibler Divergence. This measure is introduced in the next subsection.

3.2 Distance Between Distributions - Kullback-Leibler Divergence

Assuming that sample in the *reference window* has distribution p and that data in the *current window* has distribution q, we use as a measure to detect whether has occurred a change in the distribution the Kullback-Leibler Divergence (KLD)

From information theory [4], the Relative Entropy is one of the most general ways of representing the distance between two distributions [10]. Contrary to the Mutual Information this measure assesses the dissimilarity between two variables. Also known as the Kullback-Leibler divergence, it measures the distance between two probability distributions and so it can be used to test for change.

Considering two discrete distributions with empirical probabilities $p(i)$ and $q(i)$, the relative entropy of p with respect to q is defined by:

$$KLD(p||q) = \sum_i p(i) log_2 p(i)/q(i). \tag{6}$$

The KLD is not a real metric since is asymmetric $KLD(p||q) \neq KLD(q||p)$. Nevertheless, it satisfies many important mathematical properties: is a nonnegative measure, it is a convex function of $p(i)$ and equals to zero only if $p(i) = q(i)$.

Given a reference window with empirical probabilities $p(i)$, and a sliding window with probabilities $q(i)$: lower values of $KLD(p||q)$, corresponds to smaller dispersion between the distributions of the two variables, meaning that them are closer. A higher value of the distance represents distributions that are further apart. Due to the asymmetric property, if the distributions are similar, the difference between $KLD(p||q)$ and $KLD(q||p)$ is small.

3.3 Decision Rule

According to the measure above, we define that had occurred a change in the distribution of the current window relatively to the reference distribution using a high quantile, the 99th percentile (or the 95th percentile, depending on the data length), as a boundary. If the absolute difference of the KLD between the distributions of the reference and the current windows and the KLD between the distributions of the current and the reference windows is greater than 1% (or 5%) we assign a change. If no change occurs, we maintain the reference distribution and consider more data points in the current window, and start a new comparison. If we detect any anomalies and/or deviations from what is expected, we can trigger an alert alarm and we clean the reference data set and initialize the process of search for changes. This decision rule is directly related with the window's step presented previously. From Eq. 5 it is clear that the closer the distributions are to the considered significance level, the smaller the step will be.

4 Experimental Evaluation

In this section we evaluate our proposed technique against statistics for non-parametric change detection. In order to compare results we used artificial data,

where we can control the changes in generating distributions. We also present detection results obtained with our method in real world datasets to reveal its applicability and usage.

For both kinds of datasets, the data is received at any time producing an equal-with histogram. The number of bins is defined according to Eq. 4, setting both the variables ϵ and δ as 1% (or 5%, depending on the data length). We considered that the initial data points should be used as a representation of data and that the number of initial points should be chosen according to the number of intervals of the $layer_1$. So we decided that the first $10 * N_1$ data points are part of a stabilization process and that no change occurs in this range. For the decision rule, we established that if the absolute difference of the KLD between the distributions of the reference and the current windows and the KLD between the distributions of the current and the reference windows is greater than 1% (or 5%) we assign a change.

We estimate the delay time using the number of examples between the real change point and the detected point.

4.1 Controlled Experiments with Artificial Data

We compare the presented method, using a Fixed-Cumulative Window Model (FCWM) and an Adaptive-Cumulative Window Model (ACWM), with statistics for nonparametric change detection, the two-sample Kolmogorov-Smirnov Test (KST) and the Wilcoxon Rank Sum Test (WRST).

We perform tests using data underlying Log Normal distributions with different parameters, we simulated changes in mean and in standard deviation (StD). Distribution changes are created as follows: we generated 10 streams with 60K points each, the first and second 30K points of each stream are generated from $P_0 = LogN(0, 1)$ and $P1$, respectively. For changes in the mean parameter $P1 = LogN(\Delta p, 1)$ and for changes in the standard deviation parameter $P1 = LogN(0, 1 + \Delta p)$, with $\Delta p = 0.1, .., 1$. The goals of these experiments are:

1. Ability to Detect and React to drift.
2. Resilience to False Alarms when there is no drift, which is not detect drift when there is no change in the target concept.
3. The number of examples required to detect a change after the occurrence of a change.

Figure 3 shows the delay time of the change detection tests using the described artificial data, as a function of Δp. For datasets with changes in the mean parameter it can be observed that the ACWM achieve, for all values of Δp, better results than the FCWM. We can also conclude that, except for small values of Δp, the nonparametric tests outperforms the ACWM. However, the KST and WRST, have miss detections for some datasets.

The WRST tests if two datasets are independent samples from identical continuous distributions with equal medians against the alternative hypothesis that they do not have equal medians. The median of median of a LogNormal distribution is defined as e^{μ}. In the datasets with change in the standard deviation

parameter the mean parameter remains the same ($\mu = 0$), meaning that the median is the same in all stream, so it does not make sense to use the WRST in this kind of datasets. For the other three tests we can observed that ACFW is the method that presents better results, achieving them with less delay time and without false alarms (the KST had 29 miss detections).

For both kinds of changes in distribution parameters, as Δp increases, one can observes that the delay time decreases, which is consistent with the design of experiments because as greater the Δp the more abrupt is the created change between distributions.

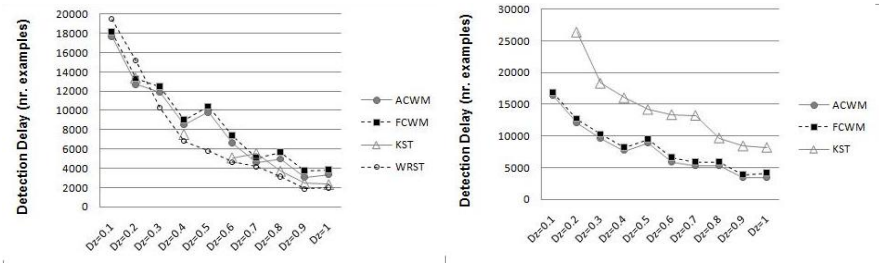


Fig. 3. Detection delay (nr. of examples) for datasets with changes in mean and in StD parameters as a function of Δp.

To evaluate the performance of the four detection tests we also use quality metrics such as *Precision* and *Recall*. The *Precision* gives a ratio between the correct detected changes and all the detected changes and *Recall* is defined as a ratio between the correct detected changes and all the occurred changes:

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN}.$$

For both quality metrics, the closest to one, the better are the results. For each detection test, we considered the sum of false alarms and missed detections (FP and FN, respectively) and the sum of correct detections (TP). For the two kinds of changes, the precision achieved by the change detection tests were equal to one. Table 1 shows the recall achieved by the four detection tests, for all the datasets with changes in the mean and standard deviation parameters, respectively.

Besides the recall values, that suggest the use of a window model to detect changes, we should also point out that both nonparametric tests compare distribution of all observed values in two different datasets. In the context of high-speed streams, data manipulations tend to become more laborious. Also for nonparametric tests, the critical values must be calculated for each distribution and these values may not always be generated by computer software. These are two reasons why nonparametric tests work only for low-dimensional data.

Table 1. Recall, for changes in mean and standard deviation parameters, of the four change detection tests.

Method	Change in mean	Change in StD
ACWM	1.00	1.00
FCWM	1.00	1.00
KST	0.86	0.71
WRST	0.90	-

Comparing obtained results with ACWM and FCWM, the advantage of using a window's step depending on the distributions' distance can be easily observed. For all datasets, the number of examples required to detect a change decreased, allowing a faster answer and a quicker adaptation of the algorithm in drift context. The results obtained with those datasets were very consistent and precise, supporting the use of a window's step depending on the distributions' distance improves the accuracy of the change detection algorithm.

4.2 Experiments with Real World Datasets

In order to evaluate our algorithm in real-world problems, we considered a dataset from an industrial environment and data sets from two different medical domains.

Industrial Dataset To obtain data, tests were carried out in a Kondia HS1000 machining centre equipped with a Siemens 840D open architecture CNC. The blank material used for the tests was a 170 mm profile of Aluminum with different hardness. The maximum radial depth of cut was 4.5 mm using Sandvik end-mill tools with two flutes and diameter 8, 12 and 20 mm. 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 measure. A Karl Zeiss model Surfcom 130 digital surface roughness instrument was used to measure surface roughness.

Each record includes information on the following seven main variables used in a cutting process:

- ***Fz*** - feed per tooth
- ***Diam*** - tool diameter
- ***ae*** - radial depth of cut
- ***HB*** - hardness on the type of material
- ***Geom*** - tools geometry
- ***rpm*** - spindle speed
- ***Ra*** - average surface roughness

These factors were recorded in the scope of the Design of Experiment explained in [9] that was used to validate a Bayesian model for prediction of surface roughness. We used the sensor measure of the cutting speed on X axes to

detect when a change had occurred in the experiments. We must point out that the measures for each test were saved individually. Then we jointed 9 of them sequentially in order to have only one dataset with eight changes. The goal is to study the effect of an adaptive window's step in change detection in an industrial problem.

Table 2 shows the obtained results and the delay time (we presented the average results for the required number of examples to detect the 8 changes). In spite of the window's step, the algorithm detects the 8 changes, but with an adaptive window's step (ACWM) the number of examples required to detect a change decreased.

Table 2. Results, for real data, using the FCWM and ACWM.

Industrial Dataset	TP	FP	DelayTime(average)
FCWM	8	0	1760
ACWM	8	0	1365

Medical Dataset - CTGs We have evaluated our detection algorithm on five Fetal Cardiotocographic (CTG) problems, collected at Hospital de São João, Porto. Fetal Cardiotocography is one of the most important methods of fetal well-being assessment. CTG signals contain information about the fetal heart rate (FHR) and uterine contractions (UC).

Five antepartum FHR with a median duration of 60 min (min-max: 59 - 103) were obtained and analyzed by the SisPorto® system. These cases corresponded to a median gestational age of 41 weeks (min-max: 59 - 103).

The SisPorto® system, developed at INEB (Instituto Nacional de Engenharia Biomédica), starts the computer processing of CTG features automatically after 11 min of tracing acquisition and its updated every minute [1], providing estimation of FHR baseline, identifying accelerations and decelerations and quantifying short- and long-term variability according to algorithms described in [2]. Along with this features the system also triggers alerts, such as 'Normality criteria met' alert, 'Non-reassuring alerts' and 'Very non-reassuring alerts' (further details can be founded in [2]). However, the system usually takes about 10 min to detect these different behaviors. In the 'Normal' stage of FHR tracing four different patterns may be considered: *A*) corresponding to calm or non-eye movement (REM) sleep, *B*) active or rapid eye movement (REM) sleep, *C*) calm wakefulness and *D*) active wakefulness [13].

Figure 4 shows an example of the analysis of a CTG exam exactly as it is produced by the SisPorto® system. The FHR and UC tracings are represented at top and at the bottom, respectively. The FHR baseline estimation, accelerations and decelerations and different alerts stages also can be observed in this figure. The 'Normal' stage is represented with a green bar in between the FHR and UC

tracings. The 'Suspicious' is represented with yellow and orange bars and the 'Problematic' with a red bar.

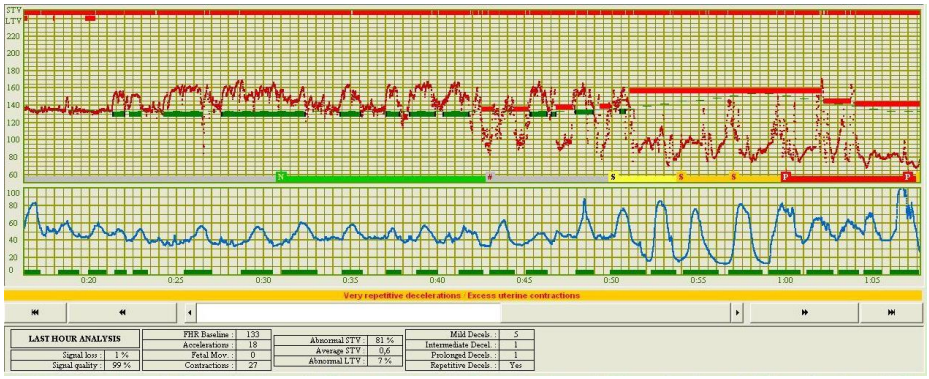


Fig. 4. FHR (top) and UC (bottom) tracings. This window also includes the FHR baseline estimation, accelerations and decelerations and patterns classification.

Our aim was to detect the concept changes detected by SisPorto®, if possible faster. We applied our detection algorithm to the FHR tracings. Because the records contained few observations, we set the input parameters ϵ and δ as 5% and for the decision rule we established the 95th percentile as boundary.

The achieved results are consistent with the system analysis and our algorithm detects the changes between the different stages earlier than the SisPorto® system. Further than the analysis of this program, our algorithm is able to detect some changes between different patterns of the 'Normal' stage. Due to difficulty of ascertain the exact change points between these behaviors we could not perform a delay evaluation. However the preference of an adaptive window's step is again supported by detections results in this dataset.

Numerical High Dimensional Dataset: MAGIC Gamma Telescope Benchmark In order to simulate a data stream with concept changes, we modified the UCI MAGIC Gamma Telescope [6], which consists of 19020 data points in 2 classes with 10 numerical (real) attributes ('fLength', 'fWidth', 'fSize', 'fConc', 'fConc1', 'fAsym', 'fM3Long', 'fM3Trans', 'fAlpha' and 'fDist').

In order to rank the attributes' importance, we create classification trees (based on the algorithm described in [5]). We started by creating a classification tree using all the attributes. Then we take out the attribute that was chosen for the split test at the root and create another classification tree, and repeat the process until the rank is finished. We obtained the following ranking: 'fAlpha', 'fLength', 'fWidth', 'fM3Long', 'fAsym', 'fM3Trans', 'fConc', 'fSize', 'fConc1' and 'fDist'.

For each attribute, we created a single vector composed first for the examples of class 'gamma' (12332 data points) and followed by the examples of class 'hadron' (6688 data points). Figure 5 shows the modified data for each attribute in this dataset.

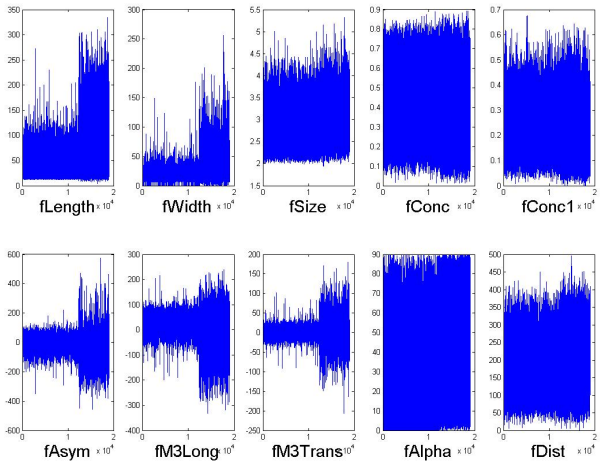


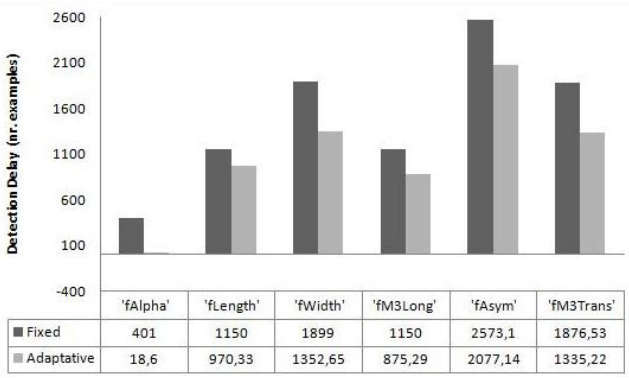
Fig. 5. Data distribution of each attribute in the MAGIC Gamma Telescope dataset.

Because the data is not 'time labeled' and to obtain results independent from the examples order, for each attribute, we shuffled the examples of class 'hadron'. We repeated this strategy obtaining 100 sets for each attribute.

We evaluated both detection algorithms (with a fixed and adaptive cumulative window) in all datasets. Performing the change detection test (using ACWM and FCWM) we expected to detect changes in the top-ranked attributes, around the class change point (12332).

Both methods (ACWM and FCWM) detect the change point for attributes 'fLength', 'fWidth', 'fAsym', 'fM3Long' and 'fM3Trans', in all the 100 datasets of each one. For the rest of the attributes, none of the algorithms detected any change point. Figure 6 shows the achieved results for these attributes. In spite of the approximated delay time, the ACWM requires less data points to detect the change point for all the mentioned attributes. As expected, both algorithms require fewer examples to detect the change in the top-ranked attributes, which is consistent with the tree classification results, since the algorithm described chooses the best attribute to split the tree.

From data characteristics one may try to explore the detected change points. Figure 7 shows the absolute difference between descriptive statistics of examples from class 'gamma' and class 'hadron' (these values are presented as percentage of the corresponding statistic of classe 'gamma'). The descriptive statistics



	'fAlpha'	'fLength'	'fWidth'	'fM3Long'	'fAsym'	'fM3Trans'
Fixed	401	1150	1899	1150	2573,1	1876,53
Adaptative	18,6	970,33	1352,65	875,29	2077,14	1335,22

Fig. 6. Detection delay (nr. of examples) for different attributes using the ACWM and FCWM.

shown are: mean, standard deviation and median. The 6 top-ranked attributes presented higher differences than the rest.

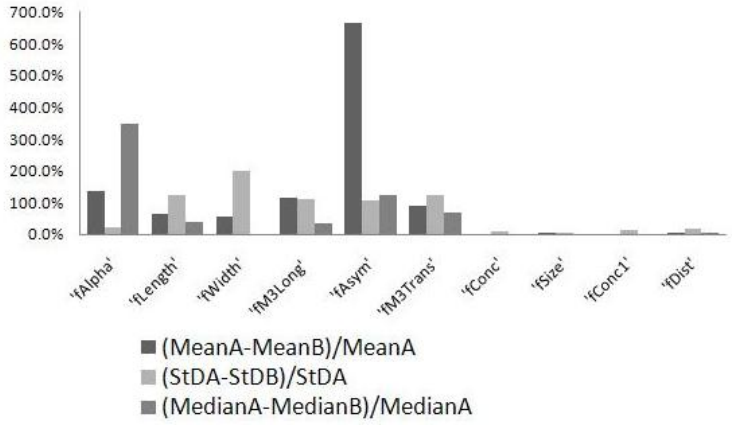


Fig. 7. Absolute percentage difference between descriptive statistics of different classes.

5 Conclusion and Future Research

In this work we address the problem of detecting changes when constructing histograms from time-changing high-speed data streams. Histograms are a widely used tool in exploratory analysis from static data, providing useful graphical information about the distribution of a random variable. They also can be used as a component in more sophisticated data mining algorithms, like pre-processing

(discretization), Bayesian classifiers and decision trees. However, in the context of open-ended data there are few contributions and still missing algorithms to address conveniently the data representation.

The method we present here is capable to understand how the process generating the stream is evolving; providing information of a time window where changes have occurred and adapting the histogram representation to the most current status of nature. For both artificial and real datasets, the results sustain that the algorithm with an adaptive window's step is capable to faster detection rates, using fewer examples to detect changes and reaching better performances. Finally, we must point out that our algorithm can be applied in a large variety of data stream problems, detecting and reacting to changes using few examples with the capacity of being resilient to false alarms when there are no drifts.

As a final conclusion, one can say that the results achieved so far are quite encouraging and motivating to continue this research line. Improvements of this algorithm and applications in more kinds of medical and industrial domains shall be considered. The use of different synopsis techniques and the adaptation of the proposed change detection algorithm to multivariate problems are future research steps.

6 Acknowledgments

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

The work of Pedro P. Rodrigues is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/29219/2006.

The authors also thank to the financial support given by the FEDER, the Plurianual support attributed to LIAAD, project ALES II (POSC/EIA/55340/2004).

References

1. Ayres-de-Campos D., Sousa P., Costa A., Bernardes J. Omniview-SisPorto® 3.5 - a central fetal monitoring station with online alerts based on computerized cardiotocogram+ST event analysis. J. Perinat. Med. 2008; 36:260-4.
2. D. Ayres-de-Campos, J. Bernardes, A. Garrido, J. Marques-de-Sá, L. Pereira-Leite. SisPorto 2.0: a program for automated analysis of cardiotocograms. J Matern Fetal Med. 2000; 9:3118.
3. D. Barbará. Requirements for clustering data streams. *SIGKDD Explorations (Special Issue on Online, Interactive and Anytime Data Mining)*, 3(2):23-27, 2002.
4. M. Berthold and D. Hand. Intelligent Data Analysis - An Introduction. Springer Verlag, 1999.
5. Breiman, L., et al.. Classification and Regression Trees. Chapman & Hall, Boca Raton, 1993.
6. R. K. Bock and P. Savicky , 2007. MAGIC Gamma Telescope Benchmark. http://archive.ics.uci.edu/ml/datasets.html.
7. M. Chabert, D. Ruiz, J-Y. Tourneret. Optimal wavelet for abrupt change detection in multiplicative noise. *IEEE International Conference on Acoustics Speech and Signal Processing*, pages: 1089-1092, May 2004.

8. G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages:281-292, June 2007.
9. M. Correa, M. de J. Ramírez, C. Bielza, J. Pamies, and J.R. Alique. Prediction of surface quality using probabilistic models. *7th Congress of the Colombian Association of Automatic*, Cali, Colombia, 21–24 Mar, 2007. (in Spanish)
10. T. Dasu, S. Krishnan, S. Venkatasubramanian and K. Yi. An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. In Interface 2006 (Pasadena, CA) Report.
11. J. Gama, P. Medas, G. Castillo and P. P. Rodrigues. Learning with Drift Detection. *Advances in Artificial Intelligence - SBIA 2004*, Vol.3171 of Lecture Notes in Computer Science, pages:286-295, São Luiz, Maranhão, Brazil, October 2004. Springer Verlag.
12. J. Gama and C. Pinto. Discretization from Data Streams: applications to Histograms and Data Mining. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages:662-667, 2006.
13. H. Gonçalves, J. Bernardes J, A. Paula Rocha et al. Linear and nonlinear analysis of heart rate patterns associated with fetal behavioral states in the antepartum period. *Early Human Development*, 83(9):585–591, 2007.
14. S. Guha, N. Koudas and J. AndWoo. REHIST: Relative error histogram construction algorithms. In *Proceedings of the VLDB Conference*, pages:300-311, 2004.
15. S. Guha and B. Harb. Wavelet synopsis for data streams: minimizing non-euclidean error. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages:88-97, August 2005.
16. S. Guha, N. Koudas and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396-438, 2006.
17. G. Hulten, L. Spencer and P. Domingos. Mining Time-Changing Data Streams. ACM SIGKDD 2001. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages:97-106, 2001. ACM Press.
18. H. V Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *Proc. of the VLDB Conference*, pages: 275-286, 1998.
19. P. Karras, D. Sacharidis, and N. Mamoulis Exploiting Duality in Summarization with Deterministic Guarantees. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages:380-389, 2007.
20. D. Kifer, S. Ben-David and J.Gehrke. Detecting change in data streams. In *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages:180-191, 2004. Morgan Kaufmann Publishers Inc.
21. R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *Learning for Text Categorization*, pages:33-40, 1998. AAAI Press.
22. R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers.
23. R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281-300, 2004.

24. M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.
25. D.D. Pestana and S.F. Velosa. *Introdução à Probabilidade e à Estatìstica*. Fundação Calouste Gulbenkian, 2002.
26. C. Pinto and J. Gama. Incremental discretization, application to data with concept drift. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages:467-468, March 2007.
27. R. Sebastião and J. Gama Change Detection in Learning Histograms from Data Streams. In *Proceedings of Portuguese Conference on Artificial Intelligence*, Guimarães, Portugal, December 2007.
28. E. J. Spinosa, A. Carvalho and J. Gama. OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages:448-452, March 2007.
29. G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69-101, 1996.