Universitï¿½ degli Studi di Napoli Federico II

Knowledge discovery methods for data streams

Methodological contributions and applications

Lidia Rivoli

Phd Thesis in
Statistics

XXV Ciclo

Dipartimento
di Matematica e Statistica
Università degli Studi di Napoli "Federico II"

via Cintia, Monte Sant'Angelo – 80126 Napoli

Knowledge discovery methods for data streams

Napoli, 29 Marzo 2013

Contents

List of Figures

Introduction

The advances in hardware technologies have made possible to produce data continuously over time and at very high rate. Let's think for instance to sensor networks for controlling security systems, for monitoring electrical consumptions, for detecting environmental pollution, for recording GPS data, to name a few. Data produced in all these framework are usually named *data streams*.

The first feature of data streams is the unbounded and increasing volume of the data. This means that the entire sequence of a data stream can not be stored in the traditional database but, they have to be processed as soon as they are available and then deleted. Obviously, using of algorithms based on multiple scan of data cannot be considered feasible in this context. Conversely, one pass and very fast algorithms could be more suitable to satisfy the storage and computational constraints of such kind of data. The most frequent approaches to deal with this problem consists in data reduction and sliding windows. In both cases, exact answers can not be provided and a trade-off between accuracy and storage/computational constraints is preferred. In particular, these techniques are based on using of *summaries structures* which allow to preserve the information contained in observations even after the data have been lost.

The second characteristic of data streams is related to the process generating data; in fact, examples are not generated randomly according to some stationary probability distribution but the underlying structure can change over time. Thus, a fundamental issue in data stream context is to monitor how the pattern may evolve over time, commonly known as *Concept drift*. Consequently, the methods for data stream mining have to be developed in order to detect and monitor the presence of evolution of data.

In this thesis, after introducing a detailed overview about existing methods for data stream, we introduce new strategies dealing with two main problems: summarization and change detection. In particular, we will propose two innovative approaches.
A first approach deals with summarizing data stream. Unlike the ones existent in data streams literature, it uses in a new way, a tool able to provide an intuitive graphic summarization of data. In fact, it discovers new concepts or behaviors of a fast changing data stream, by means of the comparison with representative statistical and graphical instruments: the histograms. So, histogram becomes a useful tool for assessing changes of data over time.

In the second approach, a method for monitoring the evolution over time of a data stream summarized by histogram data is presented. This proposal is based on an extension of the order statistics to the quantile functions associated to the set of histograms. Thus, we first define the Median, the First and the Third Quartile functions and then, we introduce a generalized representation of the box and whiskers plot in order to identify and classify the histogram outliers arriving along time.

The thesis is organized as follows. In Chapter 1, an introduction about data stream management is presented. Chapter 2 describes some of the methods and techniques introduced for dealing meanly with the two problems we discuss in this thesis: summarization and change detection in data streams. In Chapter 3 and 4, the two innovative strategies are illustrated. Finally in Chapter 5, several experimental results on real applicative fields are presented.

Chapter 1

Data Stream Mining: issues and challenges

The data analysis has been characterized by several stages. The first was the statistical Exploratory Data Analysis (or EDA) [83]. The goal was to explore the available data in order to validate a specific hypothesis. With the advances in computer science, to find computationally efficient solutions to data analysis issues was necessary. So, computational statistics [64] and machine learning [86] fields have arisen. Then, due to the increase in database sizes, machine learning and statistical analysis techniques have been modified and new algorithms have been proposed. So, Data Mining [34] has been stated as interdisciplinary field to extract models and patterns from large amounts of information stored in data repositories.

Nowadays, potentially infinite volumes of data generated continuously by real-time surveillance systems, communication networks, Internet traffic, on-line transactions in the financial market or retail industry, electric power grids, industry production processes, scientific and engineering experiments, remote sensors, and other dynamic environments are also becoming increasingly common.

These data are named *data stream* and may be defined as a sequence of ordered events which arrive at very high rate. According to this definition, we can consider a data stream as a particular case of sequential data [75], [8] indexed by time as well as time series.

Although their definition can be similar, there are several differences between data streams and time series. The main difference lies in the size of data and the manner in which they are collected and processed. A data stream is unbounded and arrives continuously over time. For this reason, it is not possible to store the entire stream permanently in a database as it occurs for time series rather it can be only processed *on the fly*. This means the techniques for analysis of time series cannot be always used to deal with data stream.

In recent years, a new research field, usually referred as *Data stream mining* (in short DSM) has attracted the attention of the scientific community. DSM consists into extracting knowledge from massive, fast changing, temporally ordered and potentially infinite data streams. To achieve this aim, some adjustments of existing techniques and methods have been proposed while others have been introduced just for data streams.

1.1 Data Stream Model

In numerous applicative contexts data takes form of unlimited and continuous data streams rather than bounded data sets. Using a more formal wording:

Definition 1.1. *A data stream is a infinite sequence of couples* (y_j, t_j) *where* y_j *is real-valued observation recorded in time-point* t_j *belonging to* *the discrete time grid* $T = \{t_1, ..., t_j, ...,\} \subseteq \Re$.

The input elements $y_1, \ldots, y_j, \ldots$ arrive describing an underlying function F and, according its values, three different types of data stream model can be detected [74]. In fact, we can consider:

- **insert only model** in which once an element y_i is seen, it can not be changed;
- **insert-delete model** in which each element y_i can be deleted or updated;
- **additive model** on which each y_i is an increment to $F[j] = F[j] + y_i$.

An example for the first type of model is represented by time-series generated by sensor networks, radio frequency identification is a case of Insert-Delete Model and for Additive Model we can consider monitoring the max (or min) of the sum of quantities.

Beyond the specific model, the main properties which identify a data stream are the following [11]:

- data are potentially unbounded in size;
- data elements arrive on-line;
- once an element from a data stream has been processed it is discarded or archived; it cannot be retrieved easily unless it is explicitly stored in memory, which typically is small relative to the size of the data streams;
- the system has no control over the order in which data elements arrive, either within a data stream or across data streams;

- the distribution generating the items can change over time.

Taking into account these properties, different challenges in data stream analysis such as storage, computational, querying and mining are emerged which will discuss in next sections.

1.1.1 Issues of data stream model

In traditional data analysis, data are first stored and then processed using algorithms that make multiple pass over the data. In data stream context, it is not feasible because two main reasons.
Firstly, the volume of data is unbounded and increases continuously over time with a rate more high than processing time. This poses a serious problem because the traditional data bases are not designed to collect the infinite sequence of data. Furthermore, even if every observation had been stored, because of huge amount of data, accessing and processing the collected items many times may become very difficult.
Secondly, data streams are often mined in a distributed fashion namely they come from hundreds of sources (like smart sensors networks) which are connected among them and to a central system. In this case, the basic processing is performed at the sensor which have limited memory capability.

A summary of the differences between traditional and stream data processing is presented in Table 1.1.

Therefore, a key challenge in data stream mining is to overcome the issue of storing data.
A common approach consists in using approximate summary structures, referred to in the literature as *synopses*. In recent years, several synopsis structures have been developed [35], [44] and the main ones include the sampling, wavelets, sketches and histograms. In Chapter 2, we present

	Traditional Methods	**Data Stream Methods**
Number of passes	Multiple	Single
Processing Time	Unlimited	Restricted
Memory Usage	Unlimited	Restricted
Type of Result	Accurate	Approximate
Distributed	No	Yes

Table 1.1: Difference between traditional and stream processing

an overview of the main synopsis highlighting the challenges and trade-offs associated with using different kinds of techniques, and the important research directions for synopsis construction.

From a computational point of view, because it is not possible to access data more than once, the multiple pass algorithm cannot be apply. New algorithms with specific characteristics have to be developed. In particular, they are able to run with high processing time and have to designed to be applied directly on streams instead of storing data beforehand in a database. The fundamental requirements can be identified in the following sentences:

- the algorithms will have to use limited computational resources, in terms of computational power, memory, communication, and processing time;
- the algorithms will have only a limited direct access to data and may have to communicate with other agents on limited bandwidth resources;
- algorithms have to perform only one scan of the data;
- the knowledge about data should be available at any point in time or on user demand.

This means that techniques should process the incoming observation in short and constant times using a very reduced amount of memory and performing a single scan of the data.
Since new data continuously arrive along time, a desirable property of these algorithms is the ability of incorporating new data. This allows to permanently maintain an accurate decision model as data arrive. Such issue requires learning algorithms that can modify the current model whenever new data is available at the rate of data arrival. Moreover, they should forget older information when data is out-dated.

Some supervised learning algorithms are naturally incremental, for example k-nearest neighbors and naive-Bayes. Others, like decision trees, require substantial changes to make incremental induction. Moreover, if the process is not strictly stationary, the target concept could gradually change over time and algorithm have to be carefully designed to work with a clear focus on the evolution of the underlying data.

1.2 Data Stream Mining

A number of algorithms have been proposed for extracting knowledge from streaming data. In this section, we discuss the main data stream mining problem and challenges relative to each of them.

1.2.1 Data Stream Clustering

Clustering is a widely studied problem in data mining literature. In particular, the methods proposed in the data stream framework may concern the examples of a single stream or the streams selves (also called *variables clustering* [39]). Data stream clustering allows to solve the problem of finding

a partition of data into homogenuous groups of items (single observation or stream selves) according to some similarity measure.
In the first case, the aim is mainly to provide a summary of the stream by means of the centroid of each cluster. This is very useful because of the impossibility to store the single observations belonging to the stream.
In the variables clustering or *clustering of time series data streams*, the goal is to discover groups of streams having a similar behavior and to explore the temporal evolution of underlying phenomenon.

Nevertheless, it is very difficult adapt any clustering algorithm to data stream framework. In fact, considering dynamic behavior of data, clustering over data streams should be addressed in an on-line manner using incremental procedure, in order to enable faster adaptation to new concepts and to produce better models over time. Obviously, traditional methods cannot adapt to the high-speed arrival of new examples, and algorithms developed to process data in real time. With respect to clustering analysis, these algorithms should be capable of continuously maintaining a compact description of the most recent data, processing examples in constant time and memory at the rate they arrive.

Interesting proposals on clustering algorithm for single data stream are the CluStream [1] and the STREAM algorithms [49].
CluStream divides the clustering process into on-line and off-line steps. In the on-line step, some summary statistics about the data stream are computed and stored at micro-clusters and the on-line updating of micro-clusters is performed. In the off-line step, a K-means algorithm is computed on the set of the stored summary statistics on the basis of a tilted time frame model. In the chapter , we give a more detailed description of this algorithm.

The STREAM algorithm aims at providing a solution to the k-median problem on data streams. The basic idea is to split data into non overlapping windows, to find a k-median solution on each window and then, to cluster the obtained centers which are weighed by the number of points assigned to them. The algorithm used to find the k-median solution on each chunk of data is a variant of the local search [50] in which an initial solution is refined by making local improvements. According to the authors, their proposal to perform the local search is able to guarantee a constant factor of approximation of the optimal solution.

The clustering of multiple data streams is, instead, a more recent challenge. Some results have been introduced in [16], [18], [78]. In the first, an extension to the data stream framework of the k-means algorithm performed on time series is proposed. The streams are split into non overlapping windows and for each of them, a Discrete Fourier Transform is used to reduce the dimensionality of data, and then, the k-means algorithm on the coefficients of the transformation is performed. The k-means is initialized using the centroids of the clusters of the partition obtained by the latest processed window. Taking into account this, the final data partition only depends from the partitions obtained in previous window and it is not possible to query about the clustering structure over user defined time intervals.

In the second method, named Clustering On Demand (COD), an on-line scan of the data for dimensionality reduction data streams using a multi resolution approach based on wavelet transformation or on piecewise linear regression. An off-line step is performed on the reduced data by using a suitable clustering algorithm in order to obtain the partitioning structure of the streams. This method allows to deal with evolving data streams but the clustering is performed on compressed streams and it does not work if new data streams are added.

The third proposal is a top-down strategy referred as Online Divisive Agglomerative Clustering (ODAC) and provides a hierarchical structure according to a dissimilarity measure based on the correlation among the streams. The procedure which allows to divide the clusters is based on a comparison between the diameter of each cluster and a threshold obtained using Hoeffding limits. The algorithm takes into account the evolution of the data using a criterion to aggregate the leaves still based on the comparison between the diameters of the clusters and the limits of Hoeffding.

Finally, the strategy proposed in [28] for discovering changes in web usage over time can be considered for monitoring the evolution of proximity relations among multiple data streams. The incoming data streams are split into non overlapping batches and then, a clustering algorithm is performed on each batch as soon as it is recorded. In order to deal with evolution in data is performed comparing the partitions intra batch through an appropriate measure.

An interesting two steps clustering algorithm for multiple data stream is furnished in [13]. In the first one, on-line arriving chunks of data are clustered in a predefined number of clusters and a proximity matrix is updated according to the membership of pairs of sequences to the same cluster. In the second, a non-metric multidimensional scaling is performed on the proximity matrix in order to represent the dissimilarities among the streams on a low dimensional space. Finally a k-means algorithm is run to provide the final partitioning of the streams.

1.2.2 Data Stream Classification

In data stream context, the classification represents a challenging problem not only for the very large size of streams of examples but also for the evolution of the data over time. In fact, the presence of changes in underlying data stream implies the need to introduce classification schema able to continuously update itself over time. In particular, the training model has to adapt quickly to new changes selecting dynamically the most appropriate classifier.

A first method for data stream classification is the *Vary Fast Decision Tree* (or VFDT) [33]. It is a decision tree learning method based on Hoeffding trees and it requires a single scan of data by using constant processing time and memory. The main drawback is that it is not feasible in environment which constantly evolves over time.

An on-line algorithm named *On Demand Classification* is proposed in [3]. The aim is to deal with continuous changing in underlying distribution of the data stream. The classification models obtained can be considered sufficiently accurate unlike the static one whose accuracy may be compromised if a sudden change for the data belonging to a specific class occurs. It is based on the idea proposed in CluStream algorithm. In fact, this method exploits the concept of supervised micro-clusters obtained by appropriate modifications to the one proposed in the unsupervised clustering [1]. The supervised micro-clusters are created from a training data and each of them represents a set of points belonging to the same class.

A further classification algorithm is proposed in [88] and it is based on an ensembles of classification models. The use of a combination of classifiers, as an alternative to the traditional use of a single classifier, allows to improve the ability to identify new concepts expressed by the data. This algorithm solves the problem of the expiration of the old data in the model by assigning a weight to each classifier which vary according to the accuracy of the provided estimate. Also, the number of classifiers may change along time without losing the accuracy of the classification process.

1.2.3 Change Detection

In data stream context, it is very hard to assume that observations are generated randomly by a stationary distribution over time. This is realistic in many real applications as real-time surveillance system, telecommunication system, sensor network. In these cases, the distribution or *concept* underlines data may shift time to time and new concepts may emerge.
The *Change detection* is widely dealt in literature [60], [61], [65], [90] and several methods are proposed in DSM especially in the context of learning [36], [54], [62], [88]. As described in the previous paragraph, the main challenge in learning methods is to continuously maintain an accurate decision model updating it when new data arrive and deleting the outdated information.

The nature of change can be very diverse and regards two aspect [40]: the cause and the rate of change. In the first case, changes are due to modifications of hidden and/or observed variables. In the second case, how the change occurs is analyzed.
In [82], *Concept Drift* is used when a gradual changing of distribution generating data is working. If the change is abrupt, the term *Concept Shift* is preferred. It is obvious that a rapid change is more easy to detect than a gradual change. In this case, it may be required several observations to detect the change. In fact, in the transient phase between the two consecutive concepts, the distributions are mixed and the newer distribution can be viewed as noise for the old one. For this reason, the detection algorithm have to be able to capture the new concepts distinguishing it by noise.
At this end, it is possible to evaluate the persistence (sensibility to change) of observations belong to a new distribution respect to simple noise (robustness to noise). In [65], the author gives a definition of concept drift by

means of the consistence and persistence.

Thus, the detection methods must be used in order to obtain meaningful descriptions and a quantification of the changes.
A first approach is proposed in [63], in which the method allows to the detect and estimate the change by comparing the distribution belonging to two different time-windows. The first time-window refers to the distribution representing the past behavior of data and is defined reference window. The second window contains the new incoming data. Both windows contain a fixed number of items and the reference window is updated when a change is detected. In order to decide whether the distributions are different, statistical tests based on Chernoff bound are performed.

Another approach is presented in [38] where the Very Fast Decision Tree (VFDT) algorithm is extended just for dealing with concept drift. The goal is a continuously tracking of differences between two class-distribution of the examples: the distribution when a node was built and the distribution in a time window of the most recent examples.

Change detection may also concern the clustering structure of a set of (or multiple) data streams [78]. In this case, *Structural drift* identifies a point starting from which the clustering structure obtained with previous observation is no longer valid because of new relations of proximity and dissimilarity between the streams. Also in this case, the multiple streams clustering methods should be able to detect and to adapt to these changes in order to maintain an updated clustering structure along time.

1.3 Summary

Nowadays, massive data generally named data streams are constantly produced by several sources. A data stream may be defined as a sequence of ordered observations arriving continuously over time and at very high rate. For dealing with these data it is necessary to satisfy very stringent computational and memory constraints. In fact, the volume of data is unbounded and increases continuously over time with a rate more high than processing time. Furthermore, data streams are often mined in a distributed fashion so, the basic processing is performed at the sensor which have limited memory capability.

Data Stream Mining represents the set of techniques developed just for data stream. They have to able to process data on fly by means of only one scan and to update, in incremental way, the knowledge about data as soon as new data arrive.

Chapter 2

Data Stream summarization and dimensional reduction

In data stream framework, new data are constantly arriving and the size of data grows continuously. This make not feasible to store the entire sequence of a data stream. A typical approach to overcome this storage constraint consists in using of summary structures known as *synopsis* [44].
A synopsis is defined as any data structure which is in size essentially smaller than its native data set. Because new items are constantly incoming along time, they must be designed for being maintained incrementally, that is, updated each time that one observation or a new sequence of observations is collected.

The general idea is that data after processing are discarded or archived and become not easily available anymore [11]. At the same time, summary structures are updated with the new incoming observations and then stored for being used in the knowledge extraction process. Thus, the knowledge is extracted starting from these summaries rather than directly from the observations.

Synopsis are used for summarizing the whole data stream but they also play an important role in query processing over data streams because allows to give answer to the *continuous* queries [23] which are typical in this context.

The choice of a particular summary structure depends on the problem which has to be solved. A synopsis used to answer a query is probably very different from the ones used to deal with a data mining problems such as change detection and classification. According to this premise, summarization is a main task in data stream processing and it has been widely dealt in literature. Main approaches are based on data histograms [37], [14], wavelets [46], sketches [27] and micro-cluster [2].

Another possible approach to make data stream analysis tractable is the dimension reduction. In particular, we can consider dimensionality reduction techniques proposed just for data streams but also suitable adjustment of techniques for dimensionality reduction of stocked time series. Among these we mention: Discrete Fourier Transform [7], Discrete Wavelet Transform (DWT) [25], Piecewise Aggregate Approximation (PAA)[59], Symbolic Aggregate approXimation (SAX) [66], Piecewise Linear Representation (PLR) [72].

In this chapter, we review the main synopses and techniques for dimensionality reduction of data streams.

2.1 Types of synopses

One of the main challenging task in data streams analysis is querying. In order to solve the query estimation problem, a frequent approach consists in considering synopses on time windows. The main time window models considered in the literature are:

- Fixed windows: each window is obtained by considering a fixed number of elements (e.g. last 10 elements). This means that each window has equal size as shown in Fig. 2.1);

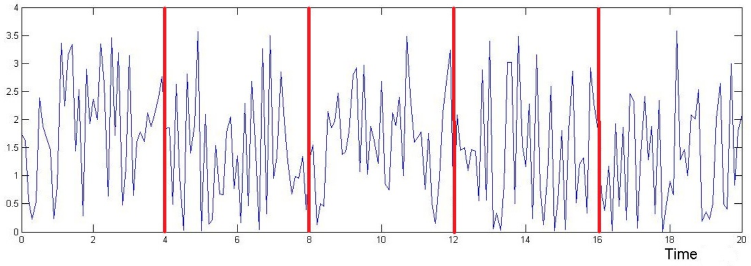


Figure 2.1: A data stream divided in equal size windows

- Landmark windows [42]: each window is obtained by setting an initial time point (landmark) and by increasing the size with the arrival of new data until a new landmark is set to allow the start of a new window.

- Sliding windows [2]: aims at processing only the most recent N observations. Each incoming data element expires after exactly N time steps as in a first in, first out data structure.

- Tilted windows [52]: aims at giving more importance to recent data without discard the oldest information. The most recent data are stored inside the window at the finest detail. Oldest information is stored at a coarser detail, in an aggregated way.

The knowledge extraction using a windows model can be strictly performed on the data defined through a window or by aggregating the results of a processing set of windows.

2.1.1 Sketches

Sketches are widely used data structures for data stream processing able to compute basic statistics about data in sub linear space. Some example of tasks which are dealt using sketches are the computation of quantiles, the detection of distinct items, the selection of frequent items. Unlike to other techniques for data stream summarization, sketches require the definition of an aggregation function. Especially, starting from two summaries of the data of two data streams or from two summaries of two different parts of the same data stream, sketches require the possibility to compute, efficiently, a new data structure which picks the information from the two initial summaries. If this aggregation function computes the sum or multiplication of the two summaries, the sketch is said to be linear.

Several sketches have been developed for data stream processing. Exponential histograms [11]are a deterministic technique for maintaining ϵ-approximate counts and sums over sliding windows in space and time that is significantly sub-linear. Deterministic waves [44] address the histogram computation problem with the same space complexity as exponential histograms, but improve the worst-case update time complexity. The Count Sketch introduced in [24] is a first proposal for addressing the problem of frequency count which requires an accuracy parameter ϵ and ensures a threshold on the probability of error. On the same topic, a widely used tool is the Count-Min Sketch in [27]. The latter has a lower space occupation but is weaker in its approximation guarantee.

2.1.2 Histogram

Among the summarization methods more used for static data sets, an important tool is provided by histograms but they are widely utilized to give a concise representation of data streams too . Allowing consider extreme values and outliers, an histogram can be defined as a set of break points $y_1, \ldots, y_{k-1}$, and a set of frequency counts $f_1, \ldots, f_{k-1}$, that identify k intervals in the range of the distribution generating the data stream.

The algorithm to deal with data stream have to be incremental, namely, to be able to incorporate new data forgetting old and outdated observations. In [37], [39] propose a incremental two-step strategy for constructing histogram data termed Partition Incremental Discretization *(or PiD).* In the first step, the range of histogram and the number K of bin are defined. Without seeing data, the range is divided in K bin and when a new data is available it is associated to a specific bin updating the value of frequencies associated to that bin. In this way, a first histogram is constructed and updated in a on-line manner. Fixed the final number of bin and using only the frequency associated to previous bins, a second histogram is provided on user demand.

The main source of inaccuracy of histogram is that an uniform distribution is assumed in each bin. A suitable solution is to consider equi-depth (or equi-frequency) histograms [73]. In this case, each bin of histogram contains the same number of points. It is evident that the design of equi-depth histogram becomes equivalent to the problem of quantile estimation for a data stream. In fact, equi-depth bins define different quantiles in the data. For this reason, the aim is to develop one-pass and with limited memory algorithm for computing quantile in data stream. Some results are proposed in [47], [69], [70].

Another type of histogram is V-optimal Histogram. It is based on the concept of minimizing over all bins (or buckets) a quantity which is called the weighted variance and denoted by V. In particular,

$$V = \sum_{i=1}^{n} n_i V_i = \sum_{i=1}^{n} \sum_{j} (f_{ij} - \bar{f})^2,$$

where n denotes the number of buckets, n_i is the number of items in bucket i, and V_i is the variance of frequencies in the ith bucket computed comparing the frequency f_{ij} of the jth value and $\bar{f}_i$ is the average frequency in that bucket.

In [57], a method for computing optimal V-Optimal Histograms for a given data set using a dynamic programming is presented and in [51] this algorithm is adapted to sorted data streams. Finally, in [45], the restriction to sorted data stream is removed through an algorithm based on the sketching technique. This algorithm provides a robust histogram of desired accuracy and number of buckets, namely, such that adding a small number of buckets, the representation quality does not improve significantly.

In the next chapters, we present two methods for summarizing and analyzing data stream using histograms.

2.1.3 Micro-clustering

Among the knowledge extraction tools for data streams, clustering is probably the most frequently used in exploratory analysis. Clustering in data stream framework is suitable for grouping items of a single data stream or for grouping the stream selves.

We will deal with only the first type. Because huge amount data stream is potentially unbounded, it would be very difficult and expensive to store each single item of the stream associated to a particular cluster. Then, the goal becomes to provide a summary of stream by means the centroid of each cluster.

Among data stream clustering methods, a recent method to capture summary information is named CluStream [1]. This algorithm is made by an on-line step and by an off-line step. The first one is on-line and it aims at collecting statistical information updating, with the arrival of new data points, summary structures called *micro-clusters*; the second one is off-line and it provides a set of summaries by processing the micro-clusters by means of a classical clustering algorithm like the k-mean.
Generally, this framework is designed for capturing summary information about multidimensional records but here, we define the concept of micro-cluster of unidimensional observations $y_1, \ldots, y_j, \ldots$

Definition 2.1. *A micro-cluster mC for a set of observations* $y_{j_1}, \ldots, y_{j_n}$ *with time points* $t_{j_1}, \ldots, t_{j_n}$ *is the* 5*-tuple* (ssv, sv, sst, svt, n) *wherein:*

- $ssv = \sum_{l=1}^{n} y_j^2$ *is the sum of the squared of the data values;*
- $sv = \sum_{l=1}^{n} y_j$ *is the sum of the data values;*
- $sst = \sum_{l=1}^{n} t_j^2$ *is the sum of the squared of the time stamps;*
- $st = \sum_{l=1}^{n} t_j$ *is the sum of the time stamps;*
- n *is the number of data elements maintained in the micro-cluster.*

Such summary structures satisfy the additivity property which is essential for providing the micro-clusters on a specific time horizon. Furthermore, the larger number of micro-clusters more information about data is detailed. In this regard, we can observe for example the Fig. 2.2.

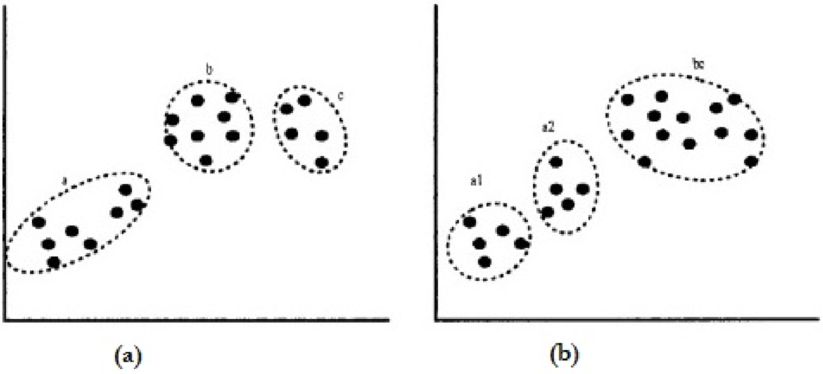


Figure 2.2: Example of the evolution of micro-clusters over time

The Fig. (a) shows 3 clusters denoted by a, b, c, which make a different data structure at a last step Fig. (b). In fact, the cluster a is split in $a1$ and $a2$ whereas b and c are merged in bc. Only if we consider micro-clusters, we are able to capture such evolution.

In the first phase of the on-line step, we need to create the set of the initial Z micro-clusters, namely $\{mC_1, \ldots, mC_z, \ldots, mC_Z\}$. This is done by applying a off-line k-means clustering algorithm to the first $InitNum$ observations.

Then, every time a new data item (y_j, t_j) is collected, it concurs to the updating of the statistics stored in one micro-cluster appropriately selected through an allocation procedure in the set MC. It is allocated to the micro-cluster $mC_z \in MC$ if the distance between y_j and the average value of mC_z is the lowest and the increasing of variance in mC_z is not superior to a threshold value. If the second condition is not satisfied for any micro-cluster, a new one mC_{Z+1} is started and if the number of micro-clusters reaches a threshold value *(th)* which is set in order to keep under control the memory occupation, the two micro-clusters having the nearest average value are mixed into one. Obviously, the number of micro-clusters in the on-line step can vary but has to be less than a fixed value. In order to make this decision, we use the cluster feature vector of Mp to decide if this data point falls within the maximum boundary of the micro-cluster Mp. If so, then the data point Xik is added to the micro-cluster Mp using the CF additivity property. The maximum boundary of the micro-cluster Mp is defined as a factor of t of the RMS deviation of the data points in Mp from the cen- troid. We define this as the maximal boundary factor. We note that the RMS deviation can only be defined for a cluster with more than 1 point. For a cluster with only 1 previous point, the maximum boundary is defined in a heuristic way. Specifically, we choose it to be the distance to the closest cluster.

The second part of the on-line step is named **snapshots recording** and consists in storing the set of micro-clusters on some available media at time stamps detected through a predefined temporal scheduling. With data flowing, this step makes available several snapshots of the micro-clusters which will be used for recovering the summarization of a user defined time period. For example, if the current time is t_{j^*} and the user is interested to a time-horizon h, that is, the user wants to find the status of the micro-clusters formed in $[t_{j^*-h}; t_{j^*}]$, then it is sufficient to recover the snapshot temporally nearest to t_{j^*-h} and the one corresponding to the current time and subtracting the values of the statistics stored in the snapshot related to t_{j^*-h} to the corresponding ones of the current snapshot.

The off-line step of CluStream is a variation of the k-means algorithm where the input data points are the average value of each micro-cluster and the center of each cluster is computed as weighted average of the allocated data points.

2.2 Sampling

Sampling is a very natural way to summarize a data set and it is relatively easy when the size of entire set of observations is fixed in advance. In the data stream context instead, sampling is not a trivial problem because the total amount of data is not available being data stream unbounded in size. In order to deal with this drawback, appropriate algorithms which do not require the knowledge of the size and the availability of the whole set of observations have been proposed.

A known one pass over algorithm for sampling is the *Reservoir Sampling* proposed in [87] and then, it has been also adapted to data streams. It is based on probabilistic insertions and deletions on arrival of new points. It builds a dynamic random sample or *reservoir* which is updated continuously in order to reflect the current history in an unbiased way over time. It detects a fixed size uniform random sample at the beginning of the stream. Then, some elements are selected randomly from the stream with a decreasing probability and replace randomly elements already in the sample. The decreasing probability ensures that the sample is uniform over the entire period.

An adjustment of this approach to the case of weighted sampling is [].

Another problem with sampling is that it might reduce the probability of detecting changes and anomalies. In [4], an interesting sampling algorithm for streams which gradually evolve over time is presented. Another proposals are presented in [12], [22] which address the problem of sampling from sliding windows. In this case, the basic idea is that rather sampling on all of the data seen so far, we can consider only on recent data contained in the most recent window.

2.3 Reduction Techniques for time series

Among dimensionality reduction techniques for data streams, some used in time series framework can be used provided the data stream computational paradigm is satisfied. In fact, a real data stream can be considered as continuously arriving time series e.g. stock markets data. Among dimensionality reduction approaches refer to stoked time series, there are: Discrete Fourier Transform (DFT), Piecewise Linear Representation (PLR), Piecewise Aggregate Approximation (PAA), Symbolic Aggregate ApproXimation (SAA).

2.3.1 The Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is based on Fourier Transform which is first proposed time series data reduction technique and still widely used. A DFT of a given time series is a particular transformation which allows to move from the time domain to the frequency domain. In fact, it is usually employed to analyze the frequencies contained in a sample signal and with respect to the continuous Fourier Transform, it is considered for signals observed in discrete time. A DFT can be considered as a data reduction technique because it is demonstrated that for most real time series, the first few coefficients contain most of the *energy* of a time series [7]. So, it is reasonable to expect those coefficients are able to capture a concise and acceptable representation of the time series. This means that those coefficients can be summarize the entire time series and similarly can be used for the data stream. This property of DFT come from Parseval's theorem which guarantees that the DFT preserve the Euclidean distance between time series. Based on Fourier Transform, a new signal processing technique called Wavelet Transform has been also proposed [25].

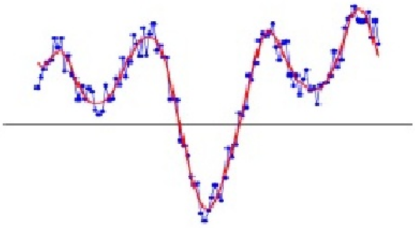


Figure 2.3: A visualization of the DFT dimensionality reduction technique

2.3.2 Piecewise Linear Representation (PLR)

Piecewise Linear Representation (PLR) approximates a time series S , of length n, by means of a number $K << n$ of segments obtained with a linear interpolation or linear regression. In data stream context, this technique is generally used to produce the best approximation such that the error for any segment does not exceed some threshold.

2.3.3 Piecewise Aggregate Approximation (PAA)

PAA is a technique for reducing the length of a time series from n dimensions to N dimensions and it was presented both in [59] and [91] independently. The technique divides time series in equi-sized overlapping windows and calculates the mean of each of them. These values will be used for providing a representation of reduced data. When $N = n$, the transformed

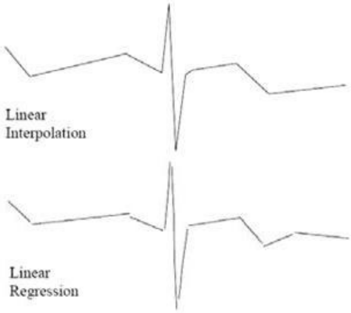


Figure 2.4: A visualization of the PLR dimensionality reduction technique

representation is identical to the original representation. When $N = 1$, the transformed representation is simply the mean of the original sequence. Similarly, it is used to reduce the length of a data stream.

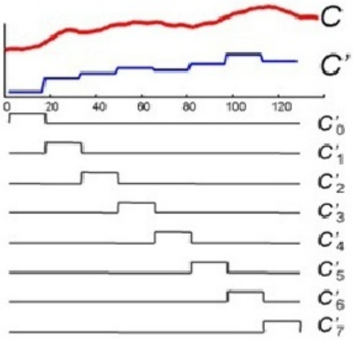


Figure 2.5: A visualization of the PAA dimensionality reduction technique

2.3.4 Symbolic Aggregate ApproXimation (SAX)

SAX introduced in [66] aims at transforming a time series into a sequence of symbols that can be processed using tools for discrete data. The time series is first normalized and then, two steps of discretization are performed. In the first step, the coefficients of PAA are obtained dividing the time series into a fixed number of equal-sized segments. In the second step, it converts the PAA to symbols which belongs to a new alphabet chosen assuming that normalized time series have a standard Gaussian distribution. So, dividing the standard Gaussian distribution in k equi-probable regions, it is possible to map each region to a symbol and then, to process these items (see the Fig. 2.6).

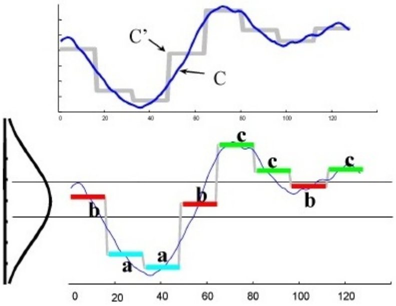


Figure 2.6: A visualization of the SAX dimensionality reduction technique

Recently, a new approach to represent a data stream into a reduced space has been illustrated in [15]. In this work, a real valued data stream is transformed into a string of symbols, each of them includes two components: the first one is a level component and the second one is a shape component. This transformation allows to get a better representation of data while maintaining a strong compression ratio (see the Fig. 2.7).

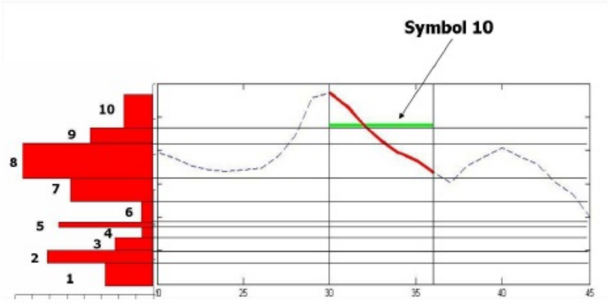


Figure 2.7: A visualization of the SAX dimensionality reduction technique

2.4 Summary

Data Streams pose new challenges both from a storage and computational point of view. A feasible approach is to store sequences of data steam through compact summary structures also named synopsis. So that, it is no longer necessary to store all the single observations but rather the extraction of knowledge will occur from these summaries themselves. Clearly, synopsis of data are not able to capture all the characteristics of the dataset, therefore, approximate answers are produced when using such data structures.

Chapter 3

Data streams reduction by histogram data clustering

In previous chapters, we have discussed about the problem of summarizing data through synopsis structures. Likely, data summarization by histograms is a of the most common techniques since it is parsimonious representation with respect to storage requirements and it provides an idea of the data underlying distribution by giving useful information about location, variability and shape.

Histograms have a long history in data analysis. In fact, they are the most used tool in exploratory data analysis [83]. In Symbolic Data Analysis (SDA) the Histogram Variable is a particular case of symbolic multi-valued modal variable and several techniques for Clustering, Regression and PCA have been proposed in [20], [21] to analyze histogram data. Furthermore, histogram is also the oldest and most widely used density estimator where *density estimation* means the construction of an estimate of the density function from the observed data [81]. Especially, histogram represents of course an extremely useful class of density estimates in the univariate case.

In this chapter, we introduce a new strategy for summarizing data stream by means histogram data. In particular, a set of histograms summarizes groups of data so that each histogram will represent a main concept in the data.
At this aim, we use a clustering algorithm based on the CluStream [1] extended to histogram data. It is an efficient clustering algorithm for data streams where the prototype of each cluster is an histogram and data are allocated to clusters through the L^2 Wasserstein distance.

The chapter is organized as follows. In the section 2, we give some formal definitions for histogram data; in section 3, we furnish a detailed description of the distances proposed for histograms; in sections 4 and 5, we illustrate CluStream and the allocation strategy, respectively; the section 6 ends the paper with an application on simulated data.

3.1 The definition of Histogram

Let Y be a continuous variable defined on a finite support $S = [\underline{y}; \overline{y}]$, where $\underline{y}$ and $\overline{y}$ are the minimum and maximum values of the domain of Y. The variable Y is supposed partitioned into a set of contiguous intervals (bins) $\{I_1, \ldots, I_k, \ldots, I_K\}$, where $I_k = [\underline{y}_k; \overline{y}_k)$. Given n observations on the variable Y, a function $\Psi(I_k) = \sum_{u=1}^{n} \Psi_{y_u}(I_k)$, where $\Psi_{y_u}(I_k) = 1$ if $y_u \in I_k$ and 0 otherwise, is associated with each semi-open interval I_k. Thus, it is possible to associate to I_k an empirical distribution $f_k = \Psi(I_k)/n$.

A histogram of Y is then the graphical representation based on a finite number of rectangles with base the interval I_k along the horizontal axis and the area equals to the relative frequencies f_k associated to I_k.
Having so defined histogram data, we consider a set of N histograms

$\{H_i,\, i = 1, \ldots, N\}$ identifying each histogram by a set of ordered couples, namely, $H_i = \{(I_{i1}, f_{i1}), \ldots, (I_{ik}, f_{ik}), \ldots, (I_{iK_i}, f_{iK_i})\}$, $i = 1, \ldots, N$. For each elementary intervals of H_i, we define the cumulative weights as follows:

$$w_{ik} = \begin{cases} 0 & l = 0 \\ \sum\limits_{h=1,\ldots,k} f_{ih} & k = 1, \ldots, K_i \end{cases} . \tag{3.1}$$

Furthermore, let $U\,(a, b)$ be a uniform density defined on the interval $[a, b]$, we may assume that within each interval $I_{ik} = [\underline{y}_{ik}, \bar{y}_{ik})$ the values are uniformly distributed and also interpret a histogram description as a particular mixture density distribution, i.e.:

$$H_i \sim \sum_{k=1,\ldots,K} f_{ik}\, U(\underline{y}_{ik}, \bar{y}_{ik}).$$

Consequently, the cumulated distribution function *(cdf)* $F_i(y)$ associated to each H_i assumes the following form:

$$F_i(y) = \begin{cases} 0 & \textit{if } y < \underline{y}_{i1} \\ w_{ik-1} + \frac{y - \underline{y}_{ik}}{\bar{y}_{ik} - \underline{y}_{ik}} f_{ik}, & \textit{if } \underline{y}_{ik} \le y < \bar{y}_{ik} \\ 1 & \textit{if } y \ge \bar{y}_{iK_i}. \end{cases} \tag{3.2}$$

Reminding that the quantile function *(qf)* of a probability distribution is the inverse of its cumulative distribution function, the *qf* associated to each H_i is:

$$F_i^{-1}(\xi) = \begin{cases} y_{i1} & \textit{if } \ \xi = 0 \\ \underline{y}_{ik} + \frac{\xi - w_{ik-1}}{w_{ik} - w_{ik-1}} (\bar{y}_{ik} - \underline{y}_{ik}) & \textit{if } \ w_{ik-1} \le \xi < w_{ik} \\ \bar{y}_{iK_i} & \textit{if } \ \xi = 1 \end{cases} \tag{3.3}$$

Both the cumulative distribution function and the quantile function associated to a histogram are piece-wise linear functions as shown in Fig. 3.1.

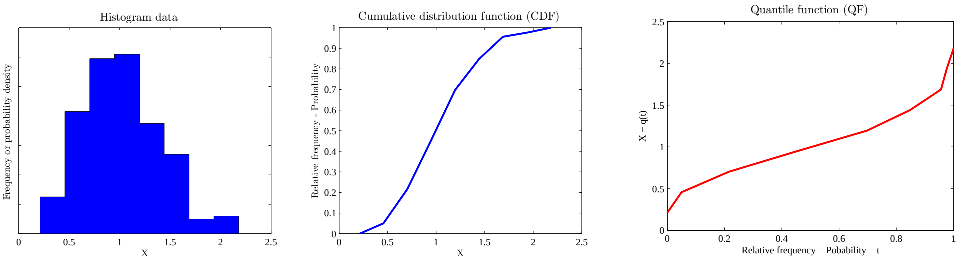


Figure 3.1: From the left to the right: a histogram datum, its cumulative distribution function (*cdf*) and the corresponding quantile function (*qf*).

3.2 Metrics for histogram data

An interesting problem when data are described by histograms concerns the choice of the metric used to compare them. Since histograms can be considered as the representation of empirical frequency distribution, a possibility consists in measuring the dissimilarity between the cumulated distribution functions associated to them. In [43], a good review on metrics between probability measures is proposed. Such metrics are summarized in Table 3.1. Generally, in order to define a metric, it is necessary to consider a measurable space Ω with a σ-algebra $\mathcal{B}$ and let $\mathcal{M}$ be the space of all probability measures on $(\Omega, \mathcal{B})$. In the following, we denote with μ and ν two probability measures (like the π_{ih} are) on Ω. Let f and g be the corresponding density functions with respect to a σ-finite dominant measure λ. It is noteworly that, if $\Omega = \mathbb{R}$, F and G denote just the corresponding distribution functions.

Abbreviation	**Metric**
D	Discrepancy
H	Hellinger distance
I	Relative entropy(or Kullback-Leibler divergence
K	Kolmogorovo (or Uniformm) metric
L	Lévy metric
P	Prokhorov metric
S	Separation distance
TV	Total variation distance
W	Wasserstein(or Kantorovich) metric
χ^2	χ^2 distance

Table 3.1: Some metrics between probability measures.

3.2.1 Discrepancy metric

It is defined on any metric space as:

$$d_D\left(\mu,\nu\right) := \sup_{\text{all closed balls } B} \left|\mu\left(B\right) - \nu\left(B\right)\right|$$

It assumes values in $[0, 1]$. Diaconis (1988, p. 34) showed that it can be used to study weak convergence on random walks on groups and show some bounds for particular distributions using Fourier transformations of probability measures on compact sets.

3.2.2 Hellinger metric

The distance is attributed to Hellinger (1901) that firstly used the quantity $\left(1 - \frac{1}{2}d_H^2\right)$ known as *Hellinger affinity*. For any measurable space, the distance can be formalized as:

$$d_H(\mu,\nu) := \left[\int_\Omega \left(\sqrt{f}-\sqrt{g}\right)^2 d\lambda\right]^{1/2} = \left[2\left(1-\int_\Omega \sqrt{fg}d\lambda\right)\right]^{1/2}.$$

For countable space its version is:

$$d_H(\mu,\nu) := \left[\sum_{\omega\in\Omega}\left(\sqrt{\mu(\omega)}-\sqrt{\nu(\omega)}\right)^2\right]^{1/2}$$

It assumes values in $[0,\sqrt{2}]$.

3.2.3 F-divergence based measures

The *f-divergence* indexes are based on a family of metrics where for every convex function ϕ one may define:

$$d_\phi(\mu,\nu) = \sum_\omega \nu(\omega)\,\phi\left(\frac{\mu(\omega)}{\nu(\omega)}\right)$$

- $\phi(x) = (x-1)^2$ yields d_{χ^2}, the Chi-square measure that thus cannot be considered as a dissimilarity;

- $\phi(x) = x\log x$ yields d_I, the Kullback-Leibler (or Relative entropy) divergence, that is not symmetric and, then, it is not a dissimilarity measure;

- $\phi(x) = |x-1|/2$ yields d_{TV}, the Total variation distance;

- $\phi(x) = (\sqrt{x}-1)^2$ yields d_H^2, the Hellinger distance.

3.2.4 Kolmogorov (or Uniform) metric

It is defined on any metric space as:

$$d_K\left(F,G\right) := \sup_x \left|F\left(x\right) - G\left(x\right)\right|, x \in \mathbb{R}$$

It assumes values in $[0,1]$.

3.2.5 Prokhorov (and Lévi-Prokhorov) metric

It is defined on any metric space as:

$$d_P\left(F,G\right) := \inf\left\{\epsilon > 0\ :\ \mu\left(B\right) \leq \nu\left(B^{\epsilon}\right) + \epsilon, \mathrm{for\,all\,Borel\,sets}\, B\right\}$$

where $B^{\epsilon} = \{x\ :\ \inf_{y\in B}\ d(x,y) \leq \epsilon\}$. It can be also rewritten as:

$$d_P\left(\mu,\nu\right) = \inf\left\{\epsilon > 0; \inf P\left[d(X,Y) > \epsilon\right] \leq \epsilon\right\}.$$

It generalizes the Lévi distance that is defined on $\mathbb{R}$

$$d_L\left(F,G\right) := \inf\left\{\epsilon > 0\ :\ G\left(x-\epsilon\right) - \epsilon \leq F\left(x\right) \leq G\left(x+\epsilon\right) + \epsilon, \forall x \in \mathbb{R}\right\}$$

Also if it not easy to compute, this metric is theoretically important because it permits to compute rate of convergence between two distributions on any separable metric space [55]. The Prokhorov distance between two random variables can be considered as the minimum distance in probability between the two random variables generated by μ and ν.

3.2.6 Total variation

For any measurable space is defined as:

$$d_{TV}(\mu,\nu) := \sup_{A\subset\Omega} |\mu(A) - \nu(A)| = \frac{1}{2}\max_{|h|\leq 1}\left|\int h\, d\mu - \int h\, d\nu\right|$$

where $h : \Omega \to \mathbb{R}$ satisfies $|h(x)| \leq 1$.
For countable spaces it is:

$$d_{TV}(\mu,\nu) := \frac{1}{2}\sum_{x\in\Omega} |\mu(x) - \nu(x)|$$

which is the half L^1 norm between two measures. It assumes values in $[0,1]$. For our aims, we focus our attention on the Wasserstein family of metrics.

3.2.7 Wasserstein distance

The Wasserstein metric has a colorful history with various historical sources [79]. The term *Vasershtein distance* for comparing two probability measures μ and ν on a metric space appeared for the first time in [89]. In particular, it was defined as $l_1(\mu,\nu) = \inf\{Ed(X,Y)\}$ where the inf is with respect to all random variables X, Y associated to measures μ and ν. However, the L^1 metric had been introduced and investigated already in 1940 by Kantorovich for compact metric spaces [58]. In 1914 Gini introduced the L^1 metric in a discrete setting on the real line and Salvemini (in the discrete case) [80] and in [29], the representation of the general case was presented. Mallows introduced the L^2 metric in a statistical context [68]. Moreover, starting from Mallows work, in [19] topological properties are described and applications to statistical problems as the bootstrap are investigated.

The general expression of L^p metric is:

$$d_W^p(\mu,\nu) := \int_0^1 \left| F_j^{-1}(\xi) - F_j^{-1}(\xi) \right|^p d\xi,\ p \geq 1, \tag{3.4}$$

where F_i and F_j are the *cdfs* of μ, ν respectively and the F_i^{-1} and F_j^{-1} their corresponding *qfs*. It is worth noting that main drawback of (3.4) is the invertibilty of cdfs. In [56], an interesting solution to this problem is provided for $p = 2$. In this case, the previous distance can be rewritten as:

$$d_W^2(\mu,\nu) = \int_0^1 \left(F_i^{-1}(\xi) - F_j^{-1}(\xi) \right)^2 d\xi, \tag{3.5}$$

also known as Mallows distance. Let H_i and H_j be two histograms having as cdfs F_i and F_j, respectively. In order to compare them by using the distance (3.5), a set of uniformly dense intervals is derived from the set of the cumulated weights of the two distribution F_i and F_j:

$$\left\{ w_{0i}, ..., w_{ui}, ..., w_{H_i i}, w_{0j}, ..., w_{vj}, ..., w_{H_j j} \right\} \tag{3.6}$$

as reported in [56]. Such set contains the sorted values of (3.6) (without repetition) and it is denoted as

$$\{w_0, ..., w_l, ..., w_L\},$$

where

$$w_0 = 0,\ w_L = 1,\ \max(H_i, H_j) \leq L \leq (H_i + H_j - 1).$$

Thus, the distance (3.5) between two histogram H_i and H_j becomes:

$$d_M^2(H_i, H_j) := \sum_{l=1}^{L} \int_{w_{l-1}}^{w_l} \left(F_i^{-1}(\xi) - F_j^{-1}(\xi) \right)^2 dt. \tag{3.7}$$

At the other hand, each $[w_{l-1}, w_l)$, $l = 1, \ldots, L$ allows us to identify two new uniformly dense intervals, one for i and one for j, having respectively the following bounds:

$$I_{il}^* = [F_i^{-1}(w_{l-1}); F_i^{-1}(w_l)] \quad \text{and} \quad I_{jl}^* = [F_j^{-1}(w_{l-1}); F_j^{-1}(w_l)].$$

Because intervals are uniformly distributed, we may express such intervals by using the function of the center (mid point) and of the radius (half-width) as: $c + r(2\xi - 1)$ for $0 \leq \xi \leq 1$. The equation (3.7) can be rewritten as:

$$d_M^2(H_i, H_j) := \sum_{l=1}^{L} \pi_l \left[(c_{il} - c_{jl})^2 + \frac{1}{3} (r_{il} - r_{jl})^2 \right], \tag{3.8}$$

where $\pi_l = w_l - w_{l-1}$ and centers and radii are given by:

$$c_{il} = (F_i^{-1}(w_l) + F_i^{-1}(w_{l-1}))/2 \; ; \; r_{il} = (F_i^{-1}(w_l) - F_i^{-1}(w_{l-1}))/2.$$

Given a set of N histogram data, by it is possible to define the Average histogram $\bar{H}$. It is computed minimizing the following sum of distances:

$$\begin{aligned} f(\bar{H}|H_1, \ldots, H_N) = & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.9) \\ = \sum_{i=1}^{N} d^2(H_i, \bar{H}) \quad = & \quad \sum_{i=1}^{N} \sum_{l=1}^{L} \pi_l \left[(c_{il} - \bar{c}_l)^2 + \frac{1}{3} (r_{il} - \bar{r}_l)^2 \right], \end{aligned}$$

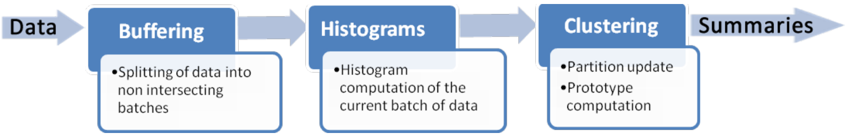

Data
Buffering
•Splitting of data into non intersecting batches
Histograms
•Histogram computation of the current batch of data
Clustering
•Partition update
•Prototype computation
Summaries

Figure 3.2: Processing schema

and it is easy to prove that it holds a minimum when:

$$\bar{c_l} = N^{-1}\sum_{i=1}^{N} c_{il} \quad ; \quad \bar{r_l} = N^{-1}\sum_{i=1}^{N} r_{il}.$$

The Average histogram can be as the centroid (or *barycenter*) of the N histogram data and can expressed by the couples: $\left(\left[\bar{c_l} - \bar{r_l}; \bar{c_l} + \bar{r_l}\right], \pi_l\right)$ of intervals with associated weights π_l.

3.3 CluStream strategy for histogram data

Let $Y = \{(y_1, t_1), \ldots, (y_j, t_j) \ldots\}$ be a data stream whose each observation y_j arrives continuously over time at fixed time stamps t_j, $j = 1, 2, \ldots$. The method we propose adapts the CluStream algorithm (see paragraph 2.1.3) to the processing the histogram data [14] according to the following schema:

- **On-line phase**

  1. Splitting of incoming data into non overlapping batches;
  2. Representation of each data batch through an equi-frequency histogram;
  3. Allocation of the histograms to a Histogram micro-cluster through the L^2 Wasserstein distance;

Symbol	Description
H_i, $i = 1, \ldots,$	Any arriving histogram
HmC_z, $z = 1, \ldots, Z$	Any histogram micro-cluster
Z_{max}	Maximum number of histogram micro-clusters
$\overline{HmC_z}$, $z = 1, \ldots, Z$	Centroid of any histogram micro-cluster
HMC_λ, $\lambda = 1, \ldots, \Lambda$	Any histogram macro-cluster
$\overline{HMC_\lambda}$, $\lambda = 1, \ldots, \Lambda$	Centroid of any histogram macro-cluster
Λ	Number of the required histogram macro-clusters

Table 3.2: Symbols with corresponding description used in CluStream strategy for histogram data.

4. Snapshot recording.

- **Off-line phase**

  1. Clustering of the Histogram micro-Cluster in order to obtain the summarization of the stream.

3.3.1 The on-line phase

Following the previous schema, in the on-line phase the data stream Y is split into time windows $W_i = \{t_j, \ldots, t_{j+S}\}$, $i = 1, 2, \ldots$ where S is the total number of time points contained in W_i. At the other hand, each W_i corresponds to a data batch $Q_i = \{y_j, \ldots, y_{j+S}\}$, $i = 1, 2, \ldots$ containing S observations and such as $Q_i \bigcap Q_{i+1} = \emptyset, \forall i$. Finally, for each data batch Q_i, we get a histogram $H_i = \{(I_{i1}, f_{i1}), \ldots, (I_{ik}, f_{ik}), \ldots, (I_{iK_i}, f_{iK_i})\}$, $i = 1, 2, \ldots$ where I_{ik} are bins obtained partitioning the domain of Q_i and f_{ik} are empirical frequencies associated to I_{ik}.

In this proposal, a first decision regards the type of histograms used

for synthesizing batches Q_i. As it will be apparent below, we have chosen equi-depth histograms, that is, each H_i is portioned in bins which have the same number of data items and $f_{ik} = f_{ik'}$, $\forall k \neq k'$.
This means that, since we have also considered batches of equal width, the number of bins K_i is equal to K for each histogram H_i, $\forall i$. Furthermore, the distance (3.8) between two histograms H_i and H_j is defined for $L = K$, $\pi_l = \frac{1}{K}, \forall l$ and is given by:

$$d^2_M(H_i, H_j) := \sum_{l=1}^{K} \frac{1}{K} \left[(c_{li} - c_{lj})^2 + \frac{1}{3} (r_{li} - r_{lj})^2 \right]. \tag{3.10}$$

The histograms H_i are the data elements which concur to update summary structures we call Histogram micro-clusters (HmC). They are an extension of the concept of the micro-cluster introduced in paragraph 2.1.3.
Unlike, a Histogram micro-cluster stores basic statistics about a set of histograms. Formally, by considering that each bin I_{ik}, $k = 1, \ldots, K_i$ of the histogram H_i, $i = 1, 2, \ldots,$ can be expressed as function of its center and radius, that is $I_{ik} = c_{ik} + r_{ik}(2t - 1)$ for $0 \leq t \leq 1$ and then, a Histogram micro-Cluster HmC can be defined as follows.

Definition 3.1. *A Histogram micro-Cluster HmC for a set of histograms is the $(2K + 1)$-tuple $(\bar{\mathbf{c}}, \bar{\mathbf{r}}, n)$ where:*

- *$\bar{\mathbf{c}}$ is the vector of the centers $\bar{c}_k$ with $k = 1, \ldots, K$ of the bins of the histogram $\bar{H}$;*
- *$\bar{\mathbf{r}}_z$ is the vector of the radii $\bar{r}_k$ with $k = 1, \ldots, K$ of the bins of the histogram $\bar{H}$;*
- *n is the number of histograms which belong to the histogram micro-cluster;*

wherein $\overline{HmC}$ is the histogram which assumes the role of centroid computed as in paragraph 3.2.7.

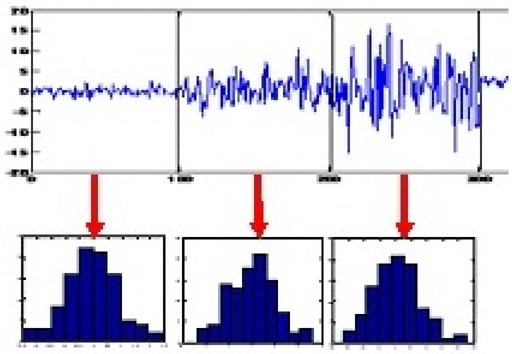


Figure 3.3: Sequence of data and related histogram.

As in the first phase of the CluStream algorithm (see par. 2.1.3), we create a initial set of Z histogram micro-clusters $\{HmC_1, \ldots, HmCz, \ldots, HmC_Z\}$. This is done by applying a off-line k-means clustering algorithm to the first $InitHistNum$ histograms.

Then, every time a new batch of data Q_i is available, the correspondent histogram H_i is computed and it is allocated to nearest micro-cluster HmC_z, selected on basis of the value of proximity between H_i and all centroids $\overline{HmC}_z$, of the $HmC_z,\ z = 1, \ldots, Z$. This means that the allocation occurs by detecting the minimum in the set of following distances:

$$d_M^2(H_i, \overline{HmC}_z) = \sum_{l=1}^{K} \frac{1}{K} \left[(c_{il} - \bar{c}_{zl})^2 + \frac{1}{3} (r_{il} - \bar{r}_{zl})^2, \right], z = 1, \ldots, Z \quad (3.11)$$

Then, once the allocation of H_i to any HmC_z is performed, the number n_z is increased by one unit and the histogram micro-cluster centroid has to be updated.

In our method, the size Z of the set of histogram micro-cluster is not defined a priori but it adapts to the structure of data, however it strongly depends on the choice of a boundary threshold $thHmC$ which fix the size of the histogram micro-clusters. Obviously, a too high value of $thHmC$ involves that only few histogram micro-clusters are generated; conversely, a too low value might generate too many histogram micro-clusters. Furthermore, a too low value of $thHmC$ involves that a lot of processed histograms will not be allocated to existing micro-clusters but they will start new ones. At the opposite, a too high value implies that histograms will be always allocated to some existing micro-cluster and it will be more difficult to capture the emerging concepts. To deal with this issue we introduce an heuristic to set the value of the threshold and a criterion to keep the number of histogram micro-clusters under a maximum value Z_{max} (this allows to keep a constant upper bound of the used memory space). Particularly, we propose to compute the threshold $thHmC$ as follows:

$$thHmC = min\ d(\overline{HmC}_z, \overline{HmC}_{z'})\ \ \forall z, z' = 1, \ldots, Z\ with\ z' \neq z \quad (3.12)$$

that is, $thHmC$ is set to the minimum distance allowed between the histogram micro-cluster centroids.

If the number of histogram micro-clusters grows too much so to exceed the available memory resources, we propose to merge the two nearest histogram micro-clusters into one. The choice is made by evaluating the L^2 Wasserstein distance between all couple of histogram micro-cluster and by selecting those ones closest among them:

$$\begin{cases} \text{If} \quad argmin_{z,z'} d(\overline{HmC}_z, \overline{HmC}_{z'}) \ \forall z, z' = 1, \ldots, Z, \ z \neq z' \\ \quad \Rightarrow \ Merge \ HmC_z, HmC'_z \end{cases}$$

. The updating of the snapshot recording are performed as in CluStream 2.1.3. In fact, we provide a method to obtain the summaries of behavior data for a user-defined time. It is based on the storing of a snapshot of the set of Histograms micro-Cluster HmC_z at a fixed time.

In order to get the summarization of the user defined time horizon, the procedure identifies the snapshot that is temporally closer to the lower end of the time interval (lower snapshot) and the one which is temporally closer to the upper end (upper snapshot). The next step is to remove from the state of the histogram micro-clusters the effects of the updates that occurred before the beginning of the lower snapshot.
Since the centroid $\overline{HmC}_z$ of each histogram micro-cluster is the average of the allocated histograms, it is possible to recover the state of each HmC_z removing what has happened before the beginning of the time slot, by computing a component by component weighted difference between the centroid as available from the upper snapshot and the corresponding HmC_z, obtained from the lower snapshot (the weights are the number of allocations stored in the parameter n_z).

3.3.2 The off-line phase

In order to detect the final synthesis of the stream, the phase off-line analyzes the micro-cluster calculated in step on-line. From the output of the previous step, the obtained centroids Z, together with the number of allocated items n_z (which assumes the role of weight), become the data to be processed by a algorithm like *k-means* which provides, as output, a partition of the Histogram micro-clusters centroids into a set of macro-clusters $\{HMC_1, \ldots, HMC, \ldots, HmC_\lambda\}$ (with $\Lambda < Z$) which are the final summaries of the required time interval and a new set of centroids. Similarly to the k-means, this algorithm minimizes an internal heterogeneity measure:

$$\Delta = \sum_{\lambda=1}^{\Lambda} \sum_{HmC_z \in HMC_\lambda} d(\overline{HmC_z}; \overline{HMC_\lambda}) n_z \tag{3.13}$$

where $d(\overline{HmC_z}; \overline{HMC_\lambda})$ is computed according to the definition (3.8) between the centroid $\overline{HmC_z}$ of histogram micro-cluster HmC_z and the centroid $\overline{HMC_\lambda}$ of histogram macro-cluster HMC_λ.

3.4 An application on simulated data

In this section, the strategy proposed is applied on several simulated data sets. We have generated nine datasets composed by 110000 temporally ordered observations characterized by two main concepts. Especially for all datasets, each concept have been kept for 50000 consecutive and ordered time stamps and the observations are locally independent and identically distributed. A further transition concept made by 10000 items has been introduced in the datasets, especially observations by a mixture of the two involved distributions with weights equal to 0.5 are generated. In all the cases, the main concepts are obtained by a random simulation of two Gaussian distributions having different parameters.

To discover if our strategy recognizes them, we have compared the prototype histograms obtained by the off-line clustering procedure with the ones related to data generated by two Gaussian distributions. The comparison is based on the distance between two histograms H_i and H_j introduced in [85]:

$$d_M(H_i, H_j) = (\bar{x_i} - \bar{x_j})^2 + (\sigma_i - \sigma_j)^2 + 2\sigma_i\sigma_j(1 - \rho(H_i, H_j)) \qquad (3.14)$$

where $\bar{x_i}$, $\bar{x_j}$, σ_i, σ_j and $\rho(x_i, x_j)$ are mean, variance and correlation of the quantile functions associated to two histograms. Through this distance function, we can evaluate the matching of two histograms in terms of location, size and shape. For this reason, we have distinguished three set of experiments. In the first one, the two distribution have the same *mean* μ and different *standard deviation* σ_i and σ_j; in the second one, they have different μ_i and μ_j and same σ values and in the third, they have different μ_i and μ_j and σ_i and σ_j values. The dataset are generated according to the distributions shown in the Table 3.3:

	First Concept		**Second concept**	
Dataset Id	μ_1	σ_1	μ_2	σ_2
1	0	1	0	5
2	0	1	0	7
3	0	1	0	7
4	2	1	5	1
5	2	1	7	1
6	2	1	10	1
7	2	1	5	5
8	3	1	9	2
9	5	1	14	10

Table 3.3: Parameters for simulated datasets

To run our procedure, we need to set the input parameters. The first input parameter is the windows size ws which, for our data, corresponds to 200 observations. According to the rule of the square root, we have set the number of bins K (or $nBin$) of the histograms to 15.

Then, we have fixed the maximum number of the micro-clusters $Zmax = 50$ which represents a good compromise between the detail of summarization and the memory usage and the final number of summaries (macro-clusters) $\Lambda = 2$ consistently with the number of main concepts in the generated data. Finally, the threshold $thHmC$ is set automatically throughout the execution of the algorithm.

The results in Table 3.4 show that for all the datasets, the procedure has been able to catch the main concepts in the data. This emerges by looking at the values of the distance but also at the values of the single components of it. In particular, the values near to 0 for the first and second component, highlight that the obtained histograms have a good match to the original data in terms of average value and standard deviation while values near to 1, for the correlation component, show that there is also a very good match in terms of shape of the histograms.

3.5 Summary

In this chapter, we have introduced a new strategy for summarizing a data stream and then, we have evaluated it on simulated data. Our approach provides, by a two-step clustering algorithm, a set of histograms which describes the main concepts emerging in a fast changing data stream. Unlike existent approaches in data streams literature, we use histograms to summarize the concepts emerging in data and to provide an intuitive graphic representation of them. Further contributions are the introduction of a Wasserstein derived distance to the context of data stream mining, as well as a more computationally efficient expression for comparing equi-frequency histograms.

	First Concept				Second Concept			
Dataset Id	$\mu - \mu_1$	$\sigma - \sigma_1$	ρ	d_M	$\mu - \mu_2$	$\sigma - \sigma_2$	ρ	d_M
1	−0.0496	−0.2422	0.9765	0.3590	−0.3480	−1.3024	0.9809	1.7161
2	−0.0802	−0.1118	0.9860	0.2405	−0.4457	−1.7096	0.9831	2.2537
3	−0.0877	−0.0550	0.9883	0.2101	−0.6223	−2.1930	0.9827	2.9110
4	0.1907	−0.1723	0.9715	0.3800	−0.0652	−0.2621	0.9812	0.3424
5	0.3404	0.0057	0.9656	0.4674	0.072	−0.2399	0.9826	0.3220
6	0.6190	0.1174	0.9304	0.8151	−0.0656	−0.2683	0.9804	0.3509
7	0.1676	−0.08	0.9797	0.3037	−0.2675	−1.4936	0.9786	1.9016
8	−0.1430	−0.5123	0.9811	0.6787	0.4075	0.0793	0.9618	0.5438
9	−0.7666	−2.6773	0.9789	3.5884	0.6431	0.3765	0.9592	0.8425

Table 3.4: Results of the comparison between the histograms emerging from the proposed procedure and the generated data.

Chapter 4

A graphical tool for data stream summarization and monitoring

In the previous chapter, the data stream summarization task has been dealt with a clustering method which provides, as output, a set of histograms, each one representing a distinct concept underlying the data stream.
In this chapter, we propose a more complex summary structure which extends the classic box-plot [83] to a set of quantile functions associated with a set of histograms.
This tool provides a whole graphical representation of the concepts in a stream by analyzing the histograms which synthesize the data contained in each time windows. We introduce a method for monitoring potential outliers in a data stream with respect to the mean or the shape of the observed histograms.

For this purpose, we present new descriptive statistics for histogram variables, and, in particular, new order skewness statistics. In the SDA approach, some basic statistics like the *sample mean* and the *standard deviation* for a histogram variable have been introduced in [17], [20], [21] and [56]. However, there is a lack of consideration about the order statistics. Usually, for defining order statistics, an order relationship is needed. In data analysis, the ordering definition problem is not trivial. For example, it is not possible to define a natural ordering for a set of units described in R^d when $d > 1$. A well known approach is based on the concept of *data depth* [83], [67], [92]. The data depth provides a center outward ordering assigning to an object a value of *centrality* or *depth* with respect to the data set which is as higher as the object is nearer to the center of a data cloud.

The concept of depth has also been proposed in the context of the functional data but the computational constraints do not allow you to apply these techniques effectively in the context of data streams.

Consequently, starting from the properties of a particular families of distances, we propose to use the quantile functions associated with histograms to define the order statistics. Especially, we make reference to a family of L^p Wasserstein distances presented in the previous chapter; in particular, we consider only the L^1 and L^2 norm (that is $p = 1, 2$).
The proposed order relation allows to define new functions representing the order statistics for the set of quantile functions. These functions are still piece-wise linear quantile functions which can be associated with a histogram data uniquely. By using the 1-st and 3-rd Quartiles, Median quantile functions we propose an extension of the classic box-plot for describing a set of histogram-valued data. Finally, some measures of variability and skewness related to the box-plot are discussed.

The chapter is organized as follows. In the section 2, we give a detailed description of the algorithm for computing of order statistics for quantile functions and its computational evaluation; in section 3, we illustrate the procedure to construct the box-plot; in the section 4, we present also some skewness indexes and, finally, the section 5 ends the paper with an application on real data.

4.1 Box and Whiskers plot for quantile functions

Let $Y = \{y_1, \ldots, y_l, \ldots\}$ be an univariate data stream whose observations y_l, $l = 1, 2, \ldots$ arrive continuously over time at a fixed time t_l. The data stream Y is split into non-overlapping time windows $W_i = \{t_j, \ldots, t_{j+S}\}$, $i = 1, 2, \ldots$ having equal size S and the corresponding histogram is computed for each of them.

The data stream Y can be represented by an infinite number of histograms again indicated as $H_i = \{(I_{i1}, f_{i1}), \ldots, (I_{ik}, f_{ik}), \ldots, (I_{iK_i}, f_{iK_i})\}$, $i = 1, 2, \ldots$.

Let us suppose to consider the first N time windows W_i, $i = 1, \ldots, N$ and the associated histogram H_i summarizing the data stream. We define the *quantile function box-plot* [77] as follows.

Definition 4.1. ***Quantile functions box-plot .*** *A box-plot for a set of quantile function is the graphical representation of five quantile functions* $\{Q_{Low}(\xi), Q_1(\xi), Me(\xi), Q_3(\xi), Q_{Upp}(\xi)\}$, $\forall \xi \in [0, 1]$ *where:*

- $Q_{Low}(\xi)$ *is the lower whiskers quantile function;*
- $Q_1(\xi)$ *is the first quartile quantile function;*
- $Me(\xi)$ *is the median quantile function;*
- $Q_3(\xi)$ *is the third quartile quantile function;*

- $Q_{Upp}(\xi)$ *is the upper whiskers quantile function.*

In the next section, we will illustrate the procedure used to define the five quantile functions forming the quantile function box-plot . In particular, since we have used the same procedure for all five quantile functions we will detail our proposal for the Median quantile function.

4.2 The Median quantile function and other order statistics

Our first aim is to define the *Median-qf* and the corresponding *Median-histogram* for a set of histogram data $\{H_i\}_{i=1,\dots,N}$ (see [76]).

Taking into account the proprieties of the median in descriptive statistics and according to [10], the *Median quantile function* can be defined as the histogram H_{Me} obtained by solving the following minimization problem based on L^1 Wasserstein distance expressed in (3.4):

$$\min_{H_{Me}} \sum_{i=1}^{N} d_1(H_i, H_{Me}) = \min_{F^{-1}(\xi)} \sum_{i=1}^{N} \int_{0}^{1} \left| F_i^{-1}(\xi) - F_{Me}^{-1}(\xi) \right| d\xi, \qquad (4.1)$$

where F_i^{-1} and F_{Me}^{-1} are the *qfs* associated with H_i and H_{Me} respectively.

By (4.1), it can be observed that H_{Me} is barycetric with respect to the set of histogram data H_i according to the L^1 Wasserstein distance so as the Average histogram according to the L^2 distance as shown in [56], [85] and in the Chapter 3.

Considering the nature of the data, and the minimization of the function in (4.1), we present a strategy for obtaining a *level-wise median quantile function*, namely, a quantile function that, for each $\xi \in [0;1]$, leaves $N/2$

quantiles before and $N/2$ quantile after the obtained value.
The procedure for computing the Median histogram and the Median qf can be divided in two steps described in the next sections and named *homogenization and selection steps.*

4.2.1 Homogenization step

As seen in the previous chapter, we set $w_{ik} = \sum_{l=1}^{k} f_{il},\ k = 1, \dots, K_i$ as cumulative relative frequencies or *levels*. We firstly homogenize the set of histograms, i.e. we detect a minimum set of values, allowing to define a set of elementary intervals of levels that do not contain angular points. Thus,

$$\mathbf{w} = \{w_{10}, \dots, w_{1K_1}, \dots, w_{i1}, \dots, w_{iK_i}, \dots, w_{N0}, \dots, w_{NK_N}\} \tag{4.2}$$

is the set of the cumulated relative frequencies associated to all histograms $H_i,\ i = 1, \dots, N$. After sorting the element of $\mathbf{w}$ and eliminating the replicated values, the previous set becomes:

$$\mathbf{w} = \{w_0, \dots, w_l, \dots, w_L\}, \tag{4.3}$$

where $w_0 = 0,\ w_L = 1$ and, denoting with $\bar{K} = N^{-1} \sum\limits_{i=1}^{N} K_i$, the value of L varies in

$$\max_{1 \le i \le N} K_i \le L \le (N(\bar{K} - 1) + 1).$$

For each $w_l,\ l = 0, \dots, L$ the values of $F_i^{-1}(w_l) = y_l,\ i = 1, \dots, N$ are known or they can be easily computed by a linear interpolation (because the qfs are piece-wise linear). So, each $H_i,\ i = 1, \dots N$ is described by a new set of L couples $\{(I_{il}^*, f_{il}^*);\ l = 1, \dots, L\}$, where: $I_{il}^* = [y_{l-1}, y_l]$ and $f_{il}^* = w_l - w_{l-1}$.

4.2.2 Median level piece-wise selection step

Fixed $l = 1, \dots, L$, each elementary intervals of levels $[w_{l-1}, w_l)$ contains the segments $F_i^{-1}(\xi)$, (with $w_{l-1} \leq \xi < w_l$) associated to the quantile functions $F_i^{-1}, i = 1, \dots, N$.

Thus after computing the values F_i^{-1} for each level w_{l-1}, (i)-th order statistics $F_{(i)}^{-1}(w_{l-1})$ on the set $\{F_1^{-1}(w_{l-1}), \dots, F_i^{-1}(w_{l-1}), \dots, F_N^{-1}(w_{l-1})\}$ is detected. If there are not intersections between $F_{(i)}^{-1}(\xi)$ and $F_{(j)}^{-1}(\xi)$, $\forall \xi \in [w_l, w_{l+1})$ and $\forall j \neq i$, then $F_{(i)}^{-1}(\xi)$ is the $i - th$ piece-quantile function in $[w_{l-1}, w_l)$. Otherwise, the order of the pieces of the quantile functions changes (as shown in Fig. 4.2) and it is necessary to select the new pieces in position (i)-th.

According this procedure, we define the Median quantile function $Me(\xi)$ for the set $\{F_1^{-1}(\xi), \dots, F_i^{-1}(\xi), \dots, F_N^{-1}(\xi)\}$, $\forall \xi \in [0, 1]$. At this end, if N is odd, we select for each level w_l with $l = 1, \dots, L$ the piece of quantile function in position $(\frac{N+1}{2})$, that is, $F_{(\frac{N+1}{2})}^{-1}(\xi)$, $\forall \xi$ such that $w_{l-1} \leq \xi < w_l$.

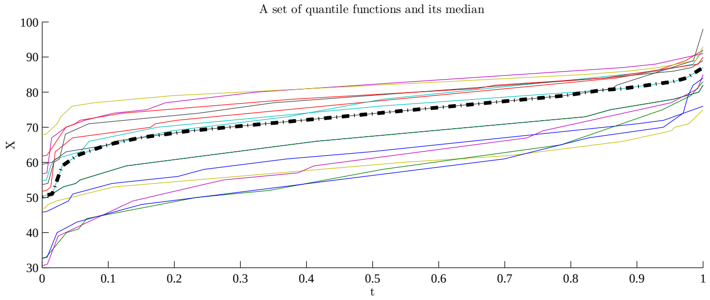


Figure 4.1: The black dotted curve represents the Median-qf obtained with the proposed method.

If N is even, we consider the average between two quantile functions in position $(\frac{N}{2})$ and $(\frac{N}{2}+1)$, that is, $F^{-1}_{(\frac{N}{2})}(\xi)$ and $F^{-1}_{(\frac{N}{2}+1)}(\xi)$, $\forall \xi : w_{l-1} \leq \xi < w_l$, in analogy of the classical case.

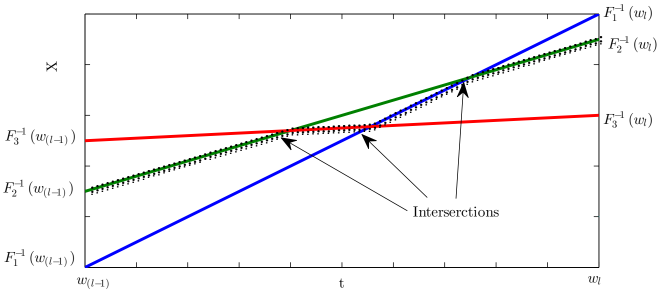


Figure 4.2: Selection of the pieces quantile level of the Median quantile function (the dotted path) in the elementary interval of levels $[w_l; w_{(l+1)}]$ with $N = 3$.

The median quantile function may correspond to an observed quantile function or it is obtained by the selected the different segments in each interval quantile level $[w_{l-1}, w_l)$.

Definition 4.2. *The Median histogram is the histogram associated with the Median quantile function $Me(\xi)$ of the set of the quantile functions $F_i^{-1}(\xi)$, $\forall \xi \in [0,1]$, $i = 1, \ldots, N$.*

Since we may order the quantile functions for a given w_l, this approach proposes a level-wise ordering which however, it is not a full order or semi-ordering relation. Naturally, if for each level w_{l-1} the ordering of the quantile functions is always the same that is, there is no intersections between quantile functions, we can extend the level-wise ordering to a full order relation.

From a computational point of view we evaluate the processing time. The selection step is performed for L times. The maximum number of intersection between the N segments to be evaluated is the order of $O(N^2)$. Thus, taking into account the number of bins and the number of potential intersections, in the worst case, the computational cost of the whole processing is of order:

$$O\left([N(\bar{K}-1)+1]N^2\right) = O(\bar{K}N^3).$$

The first and the third quartile quantile function can be computed by means of the same algorithm used for the median $Me(\xi), \xi \in [0,1]$. In fact, for each $p \cdot N$, $p \in [0;1]$ we can apply the procedure presented in 4.2 and compute the relative order statistics for the set of quantile functions. In order to distinguish such functions from the quantile functions $F_i^{-1}(\xi)$, we denote them as $Q_{(p \cdot N)}(\xi)$.
By analogy with the classical notation, we will denote with $Q_1(\xi)$ the quantile function $Q_{\left(\frac{1}{4} \cdot N\right)}(\xi)$ and with $Q_3(\xi)$ the quantile function $Q_{\left(\frac{3}{4} \cdot N\right)}(\xi)$.
Finally, the proposed algorithm for searching the quantile functions guarantees a univocal correspondence between the histograms and the quantile functions. Thus, the First Quartile histogram H_{Q_1} is associated with $Q_1(\xi)$, the Third Quartile histogram H_{Q_3} with $Q_3(\xi)$ as well as the Median histogram H_{Me} with $Me(\xi)$.

4.3 The inter quartile range and whiskers definitions

After defining $Q_1(\xi)$ and $Q_3(\xi)$, we can extend the concept of *box* and of Inter Quartile Range *(IQR)*.

Definition 4.3. *The region bounded by the piece-wise quantile functions $Q_1(\xi)$ and $Q_3(\xi)$ associated to the First and Third Quartile-histograms is* *defined as the* ***box*** *of the quantile functions box-plot representation.*

We extend also the definition of the IQR by computing the L_1 Wasserstein distance between $Q_3(\xi)$ and $Q_1(\xi)$. In particular, we can give the following definition:

Definition 4.4. *Given $Q_1(\xi)$ and $Q_3(\xi)$, the Inter Quartile Range for the qfs box-plot representation is*

$$IQR = d_W(H_{Q_1}, H_{Q_3}) = \int_0^1 |Q_3(\xi) - Q_1(\xi)| d\xi = \bar{Q_3} - \bar{Q_1}. \tag{4.4}$$

Obviously, this definition is consistent with the metric used for defining the order statistics like the Median and the Quartile functions.

Moreover, considering that $Q_3(\xi) \geq Q_1(\xi)\ \forall \xi \in [0,1]$ IQR can be interpreted as the difference of the mean values of the histograms associated with the two Quartile functions[1].

Finally, for the choice of the whiskers we can consider different criteria for select the quantile functions corresponding to the Q_{Low} and Q_{Upp}. We take into consideration three possible ways.

1. A first one is to choice the lower and the upper bounds according to $Q_{(0)}(\xi)$, and $Q_{(N)}(\xi)$ (namely Min and Max quantile functions). However, this solution can include also extreme or *outlying* quantile functions.

[1]

$$IQR = \int_0^1 |Q_3(\xi) - Q_1(\xi)| d\xi = \int_0^1 (Q_3(\xi) - Q_1(\xi))\, d\xi = \int_0^1 Q_3(\xi) d\xi - \int_0^1 Q_1(\xi) d(\xi) = \bar{Q_3} - \bar{Q_1}$$

2. A second way consists in choosing the bounding qfs according to $Q_{(0.05 \cdot N)}(\xi)$, and $Q_{(0.95 \cdot N)}(\xi)$ (namely 90% most central quantiles). This solution is less sensible to *outlying* quantile functions.

3. According to the formulation given for the IQR, a third way for defining the upper and the lower whiskers consists in translating $Q_3(\xi)$ and $Q_1(\xi)$ of 1.5 times the Inter Quartile Range (IQR). In this case, the whiskers have the same shape of $Q_1(\xi)$ and $Q_3(\xi)$ respectively, and this can be a limit in the interpretation of the final results. Furthermore, H_{Low} will be the histogram associated with $Q_{(\frac{N}{4})}(t) - 1.5 \cdot IQR$, while H_{Upp} is the histogram associated with $Q_{(\frac{3}{4}N)}(t) + 1.5 \cdot IQR$.

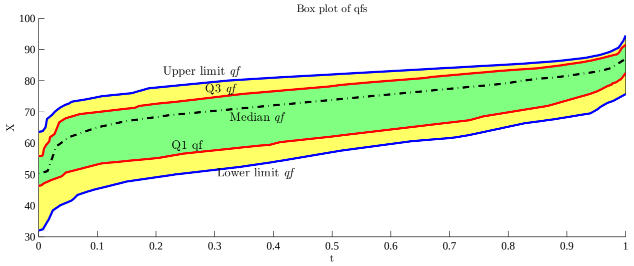


Figure 4.3: The quantile functions box-plot consisting in five qfs: the median, the first and the third quartile qfs delimiting the box, an Upper and a Lower bound qf are the extremes of the whiskers.

4.4 Variability and shape measures

The *quantile functions box-plot* as shown in Fig. 4.3 is constituted by the quantile functions corresponding to the Median $Me(\xi)$ (in the center), the

Symbol	Description
H_i, $i = 1, \ldots,$	Any arriving histogram
$w_l, l = 1, \ldots, L$	relative cumulative frequencies or levels
$F_i(\xi)$	Quantile function associated with H_i
$F_i^c(\xi)$	Quantile function centered by mean value of H_i
$Me(\xi)$	Median quantile function for the set of the F_i
$Me^c(\xi)$,	Median quantile function for the set of the F_i^c
$Q_1(\xi)$	First Quartile quantile function for the set of the F_i
$Q_1^c(\xi)$	First Quartile quantile function for the set of the F_i^c
$Q_3(\xi)$	Third Quartile quantile function for the set of the F_i
$Q_3^c(\xi)$	Third Quartile quantile function for the set of the F_i^c
$Q_{Low}(\xi)$	Lower Whisker quantile function for the set of the F_i
$Q_{Low}^c(\xi)$	Lower Whisker quantile function for the set of the F_i^c
$Q_{Upp}(\xi)$	Upper Whisker quantile function for the set of the F_i
$Q_{Upp}^c(\xi)$	Upper Whisker quantile function for the set of the F_i^c

Table 4.1: Symbols with corresponding description used for defining quantile function box-plot.

$Q_1(\xi)$ and the Third Quartile $Q_3(\xi)$ and similarly by the lower $Q_{Low}(\xi)$ and upper $Q_{Upp}(\xi)$ whiskers.

The characteristics of the box-plot in descriptive statistics can be immediately extended to this new graphical tool giving useful information about the set of the quantile functions associated to histogram data $\{H_i\}_{i=1,\ldots,N}$.

We also propose some indices that can be associated to the box-plot in order to measure the *variability* and the *shape* (skewness) of the set of histogram data (or of their corresponding quantile functions). An extension of the well-known Inter Quartile Range (IQR) has been proposed in (4.4).

However the use of the Wasserstein distance (3.4) in norm L^2 permits the decomposition of the interquartile measure in two parts corresponding to the double effect of the means and of the variability of the quantile functions in the distribution of the histogram data around the Median, that is:

$$\begin{aligned} IQR^2 = d_W^2(H_{Q_1}, H_{Q_3}) = \int\limits_0^1 \left(Q_3(\xi) - Q_1(\xi)\right)^2 d\xi = \\ = \underbrace{\left(\overline{Q_3} - \overline{Q_1}\right)^2}_{(IQR)^2} + \underbrace{\left(s_{Q_3} - s_{Q_1}\right)^2 + 2 s_{Q_3} s_{Q_1} \left(1 - \rho(Q_3, Q_1)\right)}_{\Delta(IQR)^2} \end{aligned} \tag{4.5}$$

Firstly, the IQR can be used to measure the variability among the quantile functions. Secondly, the position of Median quantile function within the box and with respect to the First and The Third Quartile quantile functions can indicate the degree of skewness of the set of quantile functions. If the distribution of the (50%) central quantile functions is symmetric, the $Me(\xi)$ would have the same distance (computed by Wasserstein metric) from $Q_1(\xi)$ and $Q_3(\xi)$; otherwise, it would be skewed.

As measures of shape we propose some skewness indices:

$$A_1 = \frac{d(H_{Q_3}, H_{Me})}{d(H_{Q_1}, H_{Me})} = \frac{\overline{Q_3} - \overline{Me}}{\overline{Me} - \overline{Q_1}}; \tag{4.6}$$

$$A_2 = d(H_{Q_3}, H_{Me}) - d(H_{Q_1}, H_{Me}) = \overline{Q_1} + \overline{Q_3} - 2\overline{Me}; \tag{4.7}$$

Therefore, an extension of the Bowley skewness (also known as quartile skewness coefficient), is:

$$A_3 = \frac{d(H_{Q_3}, H_{Me}) - d(H_{Q_1}, H_{Me})}{d(H_{Q_3}, H_{Q_1})} = \frac{A_2}{IQR}. \tag{4.8}$$

It corresponds to the normalized A_2 index. Since we use the Wasserstein L^1 norm the skewness of the distribution of quantile functions with respect to the Median quantile function can be expressed according to the mean values of the first, median and third quantile function. In order to take into account the skewness of the distribution of the quantile functions also with respect to their variability and shape, we should use the L^2 Wasserstein distance.

Alternatively, we propose to define the index A_2 for each level interval $[w_{l-1}, w_l)$, $l = 1, \dots, L$. Thus, we may express A_2 as a piece-wise function assuming constant values in the quantile intervals $[w_{l-1}, w_l)$ as follows:

$$\begin{aligned} A_2(l) &= \int\limits_{w_{l-1}}^{w_l} |Q_3(\xi) - Me(\xi)| dt - \int\limits_{w_{l-1}}^{w_l} |Q_1(\xi) - Me(\xi)| dt = \\ &= \int\limits_{w_{l-1}}^{w_l} \left[Q_1(\xi) + Q_3(\xi) - 2 \cdot Me(\xi)\right] dt \end{aligned} \tag{4.9}$$

for $w_{l-1} \leq \xi \leq w_l$ and $l = 1, \dots L$ where the number of the quantile levels L are computed during the homogenization step. It gives an information about the skewness of the distribution of quantile functions around the Median quantile function for each quantile level. The value $A_3(l) = 0$ means a symmetry of the pieces of quantile functions between the first quartile and the median and the median and the third quartile; $A3(l) < 0$ ($A3(l) > 0$) means that, the concentration of pieces of quantile functions between the first quartile and the median is on an interval of values larger (smaller) than the ones between the the median and the third quartile.

The A_1 and A_2 indices can be extended to a wider domain of quantile functions considering rather than $Q_1(\xi)$ and $Q_3(\xi)$, with $\xi \in [0,1]$, the quantile functions at 5% and 95% of the distribution (excluding extreme quantile functions which can be outliers). The index A_3 extended to a 5−th and 95−th quantile requires a different normalization than the previous one. In such way we propose:

$$A'_3 = \frac{d(H_{0.95N}, H_{Me}) - d(H_{0.25N}, H_{Me})}{N^{-1} \sum_{i=1}^{N} d(H_i, H_{Me})} \tag{4.10}$$

where the normalizing term $N^{-1} \sum_{i=1}^{N} d(H_i, H_{ME})$ is a sort of *mean absolute deviation from the median measure* among the set of the quantile functions associated to the H_i (for $i = 1, \ldots, N$) and the Median quantile function computed according to the Wasserstein L^1 distance.

4.5 Outlier detection

As said in the introduction of this chapter, we have introduced the quantile functions box-plot in order to detect possible changes in data stream over time and to identify potential outliers.

The first task can be performed by computing the box-plots related to no-overlapping subsets of the data stream, each one including a fixed number of time windows. In this way, from the comparison of the different box-plots, we may monitor their evolution along time.

The second task is based on evaluating the outlyingness of new incoming histograms with respect to a summarization of a first batch of data. At this

end, we use two further different box-plots: the first one for catching the outlyingness of the incoming histograms in mean; the second one for catching the outlyingness in the shape of the corresponding quantile function.
The former is the classical box-plot depicting the five mean values μ_{Q_1}, μ_{Me}, μ_{Q_3}, and μ_{Low} μ_{Upper} associated to the histograms $H_{Q_1}, H_{Me}, H_{Q_3}, H_{Low}$ and H_{Upp} computed on the set of the histograms H_i, $i = 1, \dots, N$.
The latter box-plot is the quantile functions box-plot on the set of the quantile function $F_1^{-1}, \dots, F_N^{-1}$ associated with H_i, $i = 1, \dots, N$ and centered by their mean values. Thus, we denote with $Q^c_{Low}(\xi)$, $Q^c_1(\xi)$, $Me^c(\xi)$, $Q^c_3(\xi)$, $Q^c_{Upp}(\xi)$ the five quantile functions which give rise the quantile functions box-plot.

For each new histogram H_{N+j}, $j = 1, 2, \dots$, its mean value and the shape of the corresponding centered quantile function are evaluated. Since any curves outside the bounds of the quantile function box-plot are identified as potential outliers, we can distinguish four different situations. In fact, if $F^{-1\,c}_{N+j}(t)$ is the centered quantile function associated to the histogram H_{N+j}, $j = 1, 2, \dots$:

1. it can be included in the region between the First Quartile and the Third Quartile quantile functions;

2. it can be included between the lower or the upper whisker quantile functions;

3. it can intersect the lower or the upper whisker qf or both of them;

4. it can be to the below of the the lower whisker or to the above of the upper whisker quantile function.

Naturally, in the fourth case, the histogram has to be considered as an *shape outlier*. The other cases need further attention because a histogram not necessarily differs in mean if its quantile function is not included in the quantile function box-plot. In general, different possibilities may arise as described in Table 4.2: For a quantile function which differs in shape, we also propose a dissimilarity measure in order to evaluate the degree of outlyingness.

Table 4.2:

Outliers classification

Mean values box-plot	**Quantile Functions box-plot**	
	Yes	No
Yes	in mean and in shape	only in mean
No	only in shape	No outlier

If ξ_0 is the x-axis of the intersection point between the potential outlier quantile function and the upper (or lower) whisker quantile function and for $\xi > \xi_0$ the $F_{N+j}(\xi)$ is outside the box-plot, then we define the following measure:

$$RD := \frac{\int_{\xi_0}^{1} \left| F_{N+j}^{-1\,c}(\xi) - Q_p^c(\xi) \right| d\xi}{\int_{0}^{1} \left| F_{N+j}^{-1\,c}(\xi) - Q_p^c(\xi) \right| d\xi} \tag{4.11}$$

where the numerator represents the area beyond the box-plot (or distance between $F_{N+j}^{-1\,c}(\xi)$ and the lower or upper whiskers quantile functions) and the denominator is the total area enclosed by two functions.

Similarly, if ξ_0 is the x-axis of the intersection point between the potential outlier quantile function and the upper (or lower) whisker quantile function, and for $\xi < \xi_0$ the $F^{-1}_{N+j}(\xi)$ is inside the box-plot, then

$$RD' := \frac{\int_0^{\xi_0} \left| F^{-1\,c}_{N+j}(\xi) - Q^c_p(\xi) \right| d\xi}{\int_0^1 \left| F^{-1\,c}_{N+j}(\xi) - Q^c_p(\xi) \right| d\xi}, \; j = 1, 2, \ldots \tag{4.12}$$

is the ratio between the area below the box-plot (or distance between $F^{-1\,c}_{N+j}(\xi)$ and Q^c_{Low} or Q^c_{Upp}) and the total area enclosed by two qfs.
It is worth noting that, the more this ratio is close to 1, the more $F^{-1\,c}_{N+j}(\xi)$ shows a different shape with respect to the empirical distribution of the centered quantile functions. Furthermore, if the $F^{-1\,c}_{N+j}(\xi)$ intersects both whiskers, the numerator of *RD* (or *RD'*) will be equal to the sum of the areas limited by $F^{-1\,c}_{N+j}(\xi)$ with $Q^c_{Low}(\xi)$ and $F^{-1\,c}_{N+j}(\xi)$ with $Q^c_{Upp}(\xi)$.

4.6 Summary

In this chapter, we have introduced a visualization tool for summarizing a data stream and for monitoring its evolution over time. The first novelty is that we have proposed suitable syntheses of quantile functions associated to histogram data representing data stream in each window and a graphical representation by box-plot of the empirical distribution of the qfs. Moreover, our proposal allows to identify potential outlier quantile functions classifying their degree of outlyingness with respect to their mean and their shape by using a classical box-plot and a qf-box-plot representation, respectively. Further investigation will regard feasible strategies for updating quantile function box-plot when it is can be considered outdated due to new changes.

Chapter 5

Experimental results

In this section, we illustrate some experimental results obtained by applying the two methodologies proposed in the previous chapters on two different data stream examples. Thus, first we describe the two data streams and then, we analyze the results for each strategy in the sections 1 and 2.

The first data set is available on-line [1]. It records the electricity demand of the Australian New South Wales Electricity Market from 7 May 1996 to 5 December 1998. In this market, the prices were not fixed but they were affected by the demand and supply so, they were set every five minutes. A key issue of the demand and price change is the time evolution of the electricity market. During the analyzed time period, the electricity market was expanded with the inclusion of adjacent areas, this allowed a more elaborated management of the demand and supply. This dataset has been already described in [36] and in [53], especially, the seasonality of the price construction, the sensitivity to short-term events and weather fluctuations have been analyzed.

[1] Dataset URL:http://194.117.29.249/ $jgama/ales/ales_5$.html

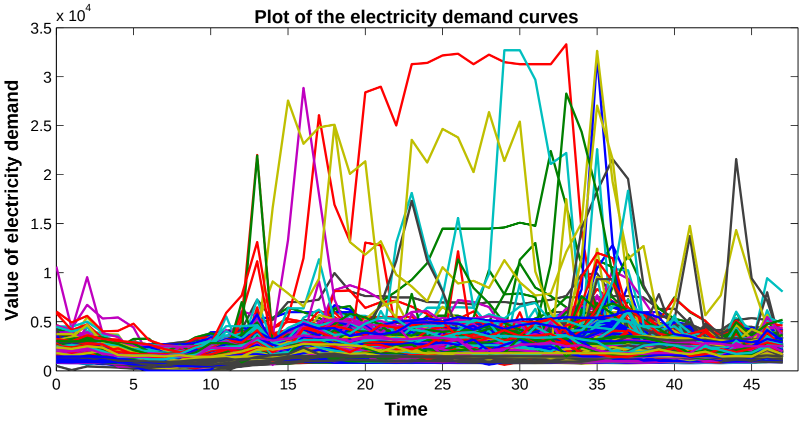


Figure 5.1: Plot of electricity demand curves recorded from 7 May 1996 to 5 December 1998

The dataset contains 45312 instances recorded every half hour, i.e. there are 48 instances for each day. Each example on the dataset has 3 fields: the day of week, the time stamp and the New South Wales electricity demand.

The second data stream was collected by using a *HOBO Data Logger* device placed at the Marine Protected Area of the *Gaiola Underwater Park in the northwestern Gulf of Naples*. In particular, the acquisition of environmental data by this sensor is carried out as part of a regional project for the monitoring of the coastline involving the *Stazione Zoologica Anton Dohrn*. Especially, it is used for recording daily light intensity (expressed by $\mu mol/m^2 \cdot s$) and temperature values (°C) at 2.5 m under the sea level.

We have considered 21634 instances recorded from 22 June to 19 November 2012 every 10 minutes. Data collection is closely linked to daily light hours so, the number of observations is not generally the same but varies according to the season and the light hours. Accordingly, the sensor records

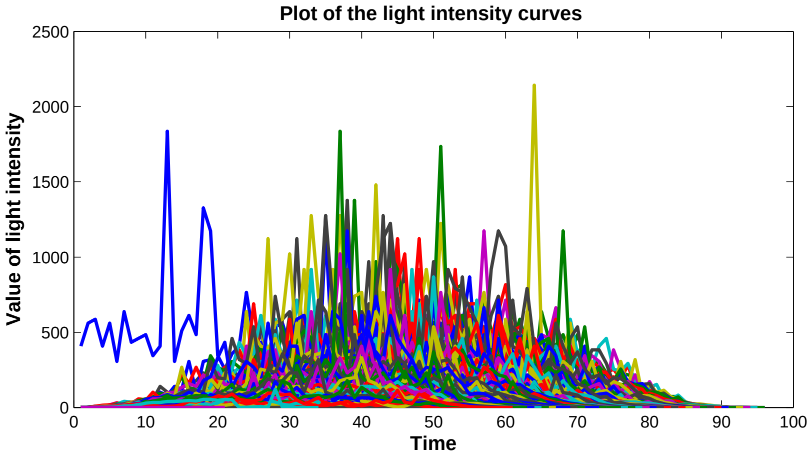


Figure 5.2: Light intensity recorded from 22 June to 19 November 2012.

zero values during the hours of darkness.

In the following sections, we introduce the details of each experimental evaluation. Especially, the chapter is organized as follows. In the section 2, we give a description of experimental results concerning the application of the CluStream strategy for histogram data on both datasets and, in section 3, we illustrate the performance of the box-plot based strategy on the same datasets.

5.1 CluStream strategy for histogram data

In this section, we show as the methodology proposed in the third chapter, can help to catch the main behaviors along a fixed period. We apply the strategy to both data sets presented above.

The assessment of the method requires to set the input parameters for on-line and off-line step. The on-line parameters are:

- the size of each window ws;
- the number of bins of the histograms $nBin$;
- the maximum number of histogram micro-clusters $Zmax$.

The first input parameter depends on the choice of the period of monitoring and the second should be chosen according to the required level of detail in the approximation of the data distribution. Furthermore, it can be noticed that the use equi-depth histograms in our strategy, implies that the value of $nBin$ is related to the number of quantiles of the approximated distribution. The last parameter is determined according the amount of memory available in order to store the histogram micro-clusters. According to this memory constraint, typical value of $Zmax$ is significantly larger than the natural number of the concepts emerging from a massive data stream but also significantly smaller than the number of the histograms (or equivalently, of the time windows) which are computed in a long period of time.

As highlighted in Chapter 3, the boundary threshold $thHmC$ is not an input parameter because its value is determined automatically by algorithm taking into account that a too low value of $thHmC$ involves that a lot of processed histograms will not be allocated to existing micro-clusters but they will start new ones. At the opposite, a too high value implies that histograms will be always allocated to some existing micro-cluster and it will be more difficult to capture the emerging concepts.

Finally, for the off-line step, which is performed taking as input the whole set of HmC_z, $z = 1, \ldots, Z$, we need to fix the maximum number of the macro-clusters Λ.

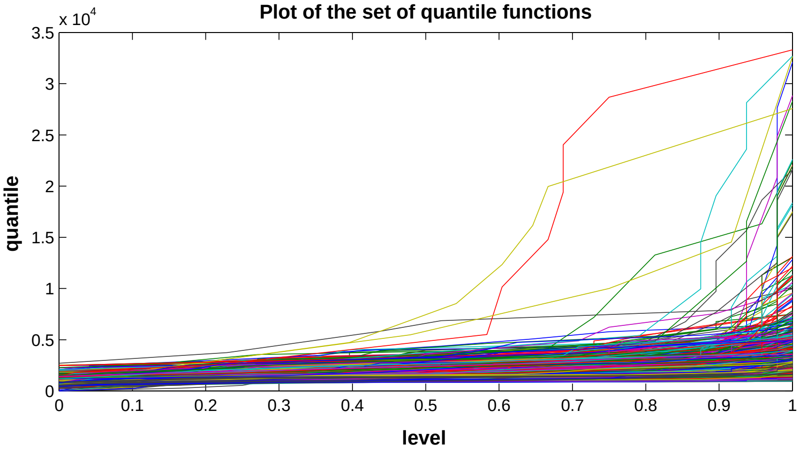


Figure 5.3: Plot of the quantile functions of the electricity demand recorded from 7 May 1996 to 5 December 1998

5.1.1 Results for electricity data set

For the electricity demand dataset, in order to compute each on-line histogram, we have set $ws = 48$. Thus, each time window contains the same number of observations and it corresponds to an entire day of tracking. This means that we have considered 944 time windows.

For each of them, we have computed an equi-depth histogram considering $nBin = 7$ (chosen by using the *square-root criterion*); the corresponding the curve of electricity demand and the corresponding quantile function. The set of the curves representing the electricity demand for the 944 days of monitoring is shown in Fig.5.1 and, the set of the quantile functions associated with each histogram is plotted in Fig.5.3.

The third parameter has to be set so that the number of on-line generated micro-clusters is the highest as possible under the constraint of the

HmC_z	n_z	HmC_z	n_z	HmC_z	n_z	HmC_z	n_z
HmC_1	126	HmC_5	9	HmC_9	3	HmC_{13}	87
HmC_2	76	HmC_6	5	HmC_{10}	106	HmC_{14}	214
HmC_3	113	HmC_7	1	HmC_{11}	60	HmC_{15}	1
HmC_4	16	HmC_8	101	HmC_{12}	12	HmC_{16}	14

Table 5.1: Electricity data: number of the on-line histograms allocated to each histogram micro-cluster

available memory space. This allows to pick at best the different behaviors in data. According to this aspect, a good comprise between the details of summarization and memory usage, the $Zmax$ value is set equal to 100. Finally, the boundary threshold $thHmC$ automatically computed by algorithm is 13000.

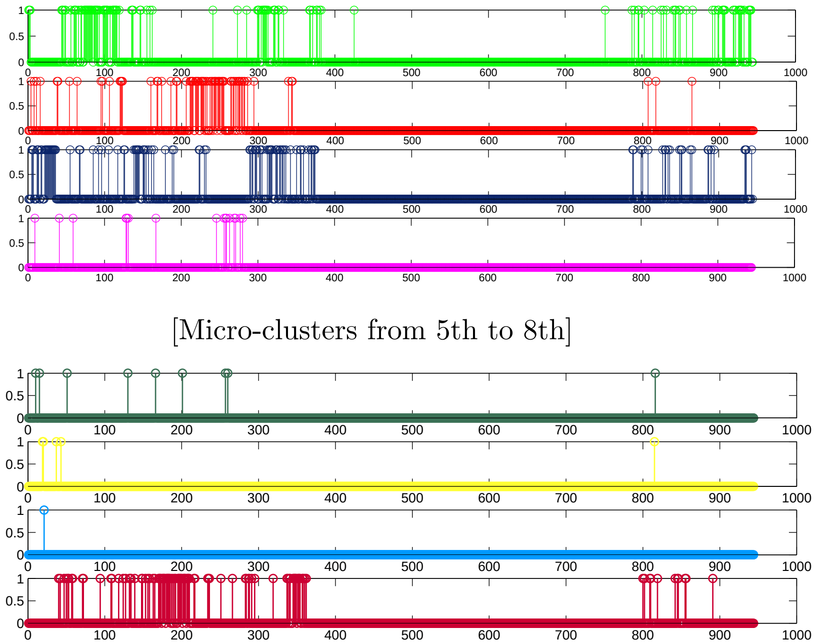
As shown in Tab.5.1, the 944 on-line histograms have been assigned to 16 different histogram micro-clusters. Especially, 12 histogram micro-clusters collect more than 10 histograms so that, these are the ones that record the main concepts in the data. The remaining histogram micro-clusters summarize the anomalous or outlier behaviors.

In Fig.5.4, 5.5, we have also investigated the evolution of the histogram micro-clusters HmC_z, $z = 1, \ldots, Z$, over the 944 time windows. Especially, each plot represents one of the 16 different histogram micro-clusters and describes the membership of each histogram $H_i, i = 1, 2, \ldots$ to the histogram micro-cluster HmC_z, $z = 1, \ldots, 16$ according to a characteristic function. This characteristic function assumes value 1 if H_i belongs to HmC_z and the value 0 otherwise.

For instance, in the top of the Fig.5.4 we can observe as the histograms have been associated to the HmC_1 over the time.

The micro-cluster HmC_1, summarizes, mainly, the data stream behavior

Figure 5.4: Evolution over time of the micro-clusters on the electricity data set

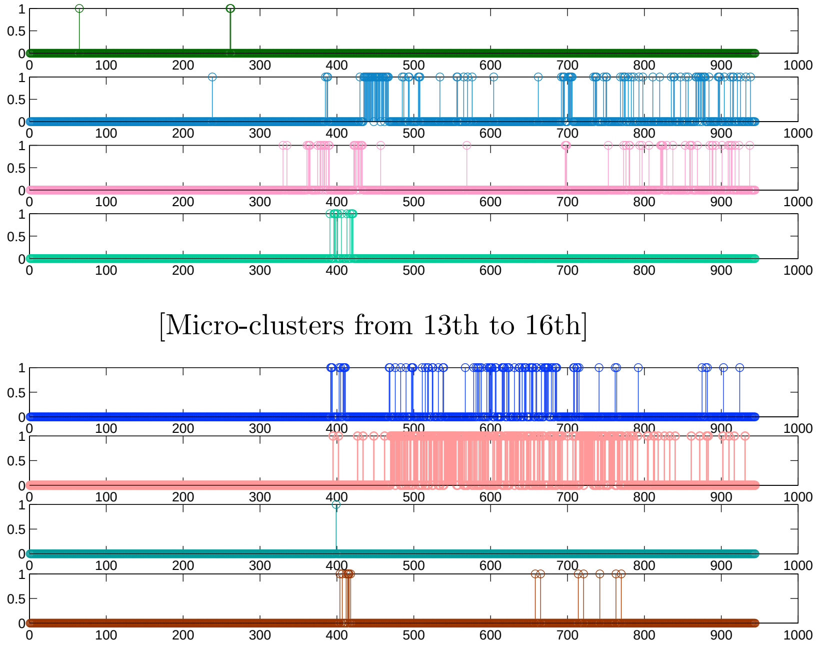


[Micro-clusters from 1st - 4th]

[Micro-clusters from 5th to 8th]

of three temporal intervals. In fact, we note that the histograms have been associated to this histogram micro-cluster between 40-th and 150-th time window, between the 200-th and the 300-th and finally, between the 700-th and the 900-th time window. No on-line histograms are assigned to HmC_1 between the 400-th and the 700-th time window. This means that, HmC_1 represents a concept which occurs in the data quite frequently over time. Conversely, other histogram micro-clusters like HmC_{13} and HmC_{14} represent concepts appearing only from the 400-th time onwards

Figure 5.5: Evolution over time of the micro-clusters on the electricity data set



[Micro-clusters from 9th to 12th]

[Micro-clusters from 13th to 16th]

being entirely absent before.

The off-line procedure, which is performed taking as input the whole set of HmC_z, $z = 1, \dots, 16$, provides a final summarization of the histogram data. We are interested in discovering how the changes occur over the days and if there are dominant structure in the histogram data behaviors. For the selection of the final number of histograms we have taken into account the micro-clusters obtained in on-line step and the value of the optimization function related to the algorithm of the off-line clustering (see Fig. 5.6).

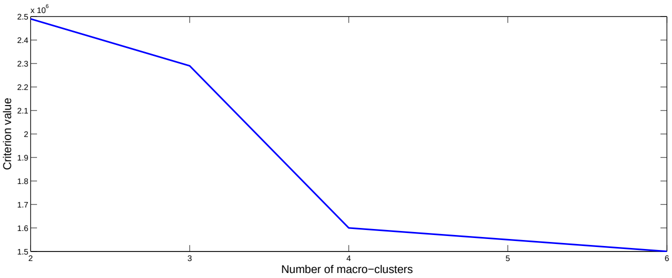


Figure 5.6: Number of micro-clusters with respect to the values of the optimization function.

A reasonable choice for the final number of histogram macro-clusters is 4 whose centroids are represented in Fig. 5.7.

As shown in Table 5.1.1, such histograms are very different in location, variability and shape, so that, we can consider them like a summarization of the 4 main concepts emerging from the data stream.

HMC	Mean	Std.	Skewness	Kurtosis
		Deviation	(Fisher Index)	(Pearson Index)
HMC_1	1964.50	376.95	0.1451	0.0183
HMC_2	1154.12	237.84	0.0654	0.0206
HMC_3	2456.11	417.5457	0.1216	0.0184
HMC_4	1793.48	536.4652	0.4661	0.0205

Table 5.2: Descriptive statistics for the 4 histogram macro-clusters.

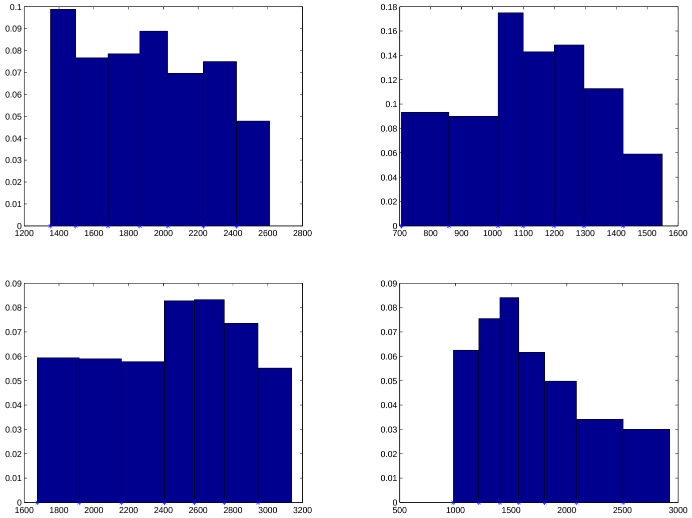


Figure 5.7: The histogram data representing the centroids of the 4 macro-clusters for the electricity data set.

5.1.2 Results for light intensity data set

As in the previous case, we have associated to each day of observation a different time window. Since the light intensity may change from day to day, the number of observation is variable and the window size ws is not set a priori. However, ws varies between 61 to 95 so that, we have chosen $nBin = 8$ (by using the *square-root criterion*). Finally, the $Zmax$ parameter has been chosen equal to 30. The set of the curves representing the light intensity in each day of monitoring is shown in Fig. 5.2 and, the set of the quantile functions associated with each histogram is plotted in Fig. 5.8. From the results (see Tab.5.3), we can observe that the 152 on-line histograms have been assigned to 15 different histogram micro-clusters.

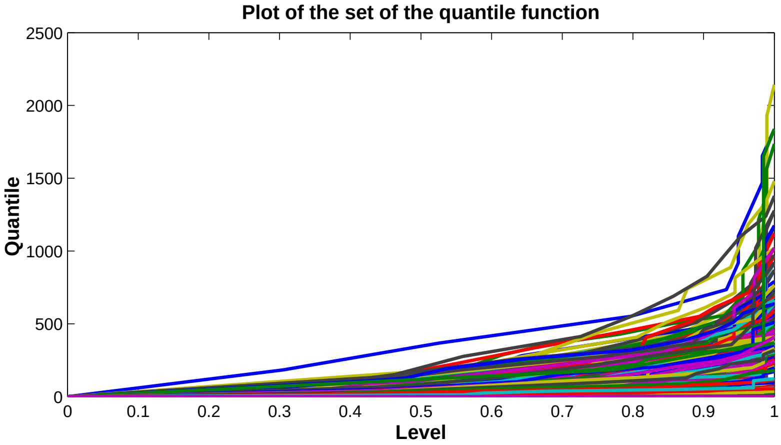


Figure 5.8: Quantile functions of the light intensity recorded from 22 June to 19 November 2012.

Especially, 8 histogram micro-clusters collect more than 10 histograms so, these are the ones recording the main concepts in the data. The remaining histogram micro-clusters summarize the anomalous or outlier behaviors. As in previous case, we have also investigated the evolution of the histogram micro-clusters over the 152 time windows. The allocation of on-line histogram to HmC_z, $z = 1, \ldots, 15$ along time is presented in the Fig. 5.9, 5.10.

The off-line procedure, which is performed taking as input the whole set of HmC_z, $z = 1, \ldots, 15$, provides a final summarization of the data. In order to set the number Λ of macro-clusters, we have considered an optimization criterion whose graph is shown in Fig. 5.11. According to it, we have chosen $\Lambda = 5$. The centroids of the histogram macro-clusters obtained by running the k-mean algorithm are plotted in Fig. 5.12 and the

HmC_k	n_k	HmC_k	n_k	HmC_k	n_k	HmC_k	n_k
HmC_1	12	HmC_5	1	HmC_9	20	HmC_{13}	5
HmC_2	8	HmC_6	2	HmC_{10}	11	HmC_{14}	17
HmC_3	13	HmC_7	6	HmC_{11}	14	HmC_{15}	6
HmC_4	1	HmC_8	25	HmC_{12}	12		

Table 5.3: Light Intensity data: number of the on-line Histograms allocated to each micro-clusters.

descriptive statistics for each of them are reported in Tab. 5.1.2. These histograms can be considered like a summarization of the 5 main concepts emerging from the data stream. Their differences in location, variability and shape are expressed in the Table. 5.1.2.

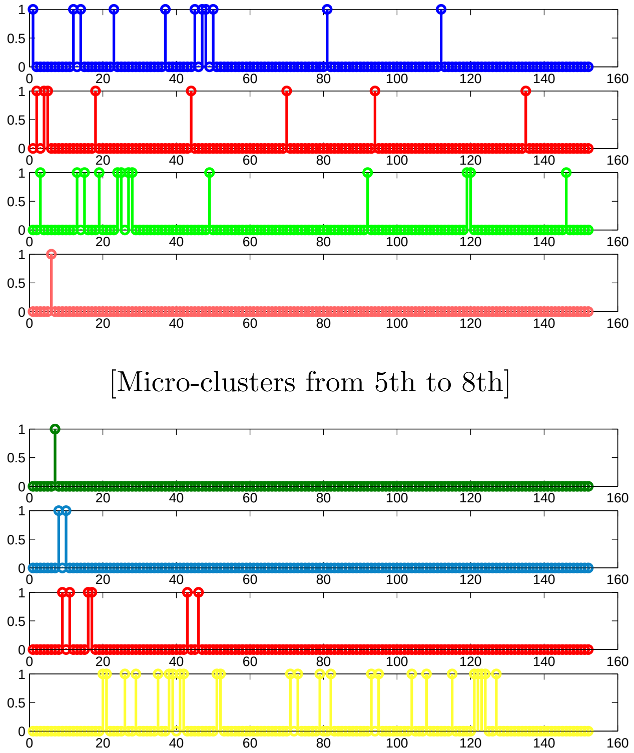
HMC	Mean	Std.	Skewness	Kurtosis
		Deviation	(Fisher Index)	(Pearson Index)
HMC_1	160.02	162.74	1.1122	0.0311
HMC_2	115.52	126.89	0.8191	0.0250
HMC_3	87.20	85.44	0.9751	0.028
HMC_4	56.85	55.95	0.9831	0.0278
HMC_5	32.95	34.41	1.0350	0.0288

Table 5.4: Descriptive statistics for the 5 histogram macro-clusters.

5.2 Boxplot strategy

In this section, we validate the strategy proposed in the Chapter 4 through both the data sets. Also in this case, the first step consists in dividing the data stream in non overlapping time windows and in constructing the

Figure 5.9: Evolution over time of the micro-clusters on the light intensity data set



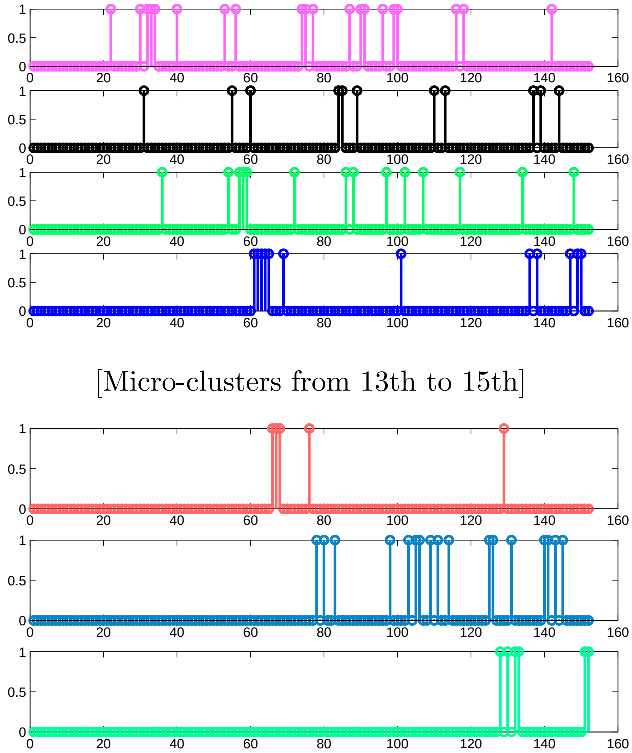
[micro-clusters from 1st to 4th]

[Micro-clusters from 5th to 8th]

histograms associated to each of them. For this reason, we have considered the same time windows size ws and the same number of bins $nBin$ used for the previous strategy.

A parameter to be set for this algorithm is the initial number of histogram data, or equivalently the initial number of time windows, from

Figure 5.10: Evolution over time of the micro-clusters on the light intensity data set



[Micro-clusters from 9th to 12th]

[Micro-clusters from 13th to 15th]

which to build the *quantile function box plot.* As explained in the previous chapter, the latter is defined through five curves, similarly to the five summary statistics defining a classical box-plot. Especially, we have computed the whiskers generalizing the empirical rule, by shifting the $Q_1(\xi)$ and $Q_3(\xi)$ by 1.5 times the IQR.

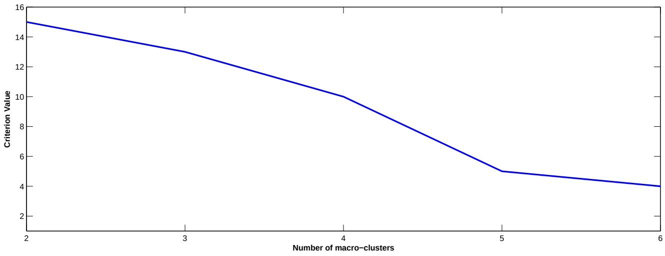


Figure 5.11: Number of micro-clusters with respect to the values of the optimization function for light intensity data set.

5.2.1 Results for electricity data set

As said above, we have considered the same input parameters for computing the set of the on-line histograms. This means that each day of monitoring represents a time window whose size is $ws = 48$ (thus, we have 944 windows), and each equi-depth histogram has the same number of bins $nBin = 8$. Finally, we set the initial number of time windows equals to 50. According this choice, we have constructed the box-plot for the set of the mean values associated with the first 100 histogram data and the box-plot for the set of the corresponding quantile functions as shown in Fig. 5.13. Especially, in Fig. 5.13, the median is the dashed black curve, the first and the third quartile delimit the box in red and, the curve in blue are the upper and the lower whiskers quantile functions. Naturally, the median curves can be interpreted as the most representative observed pattern in the Australian New South Wales Electricity Market.

Comparing the original curves to the quantile functions box-plot, we see that the latter is very useful to summarize over time, the electricity demand and it is very informative to identify the potential outliers. In fact, for each

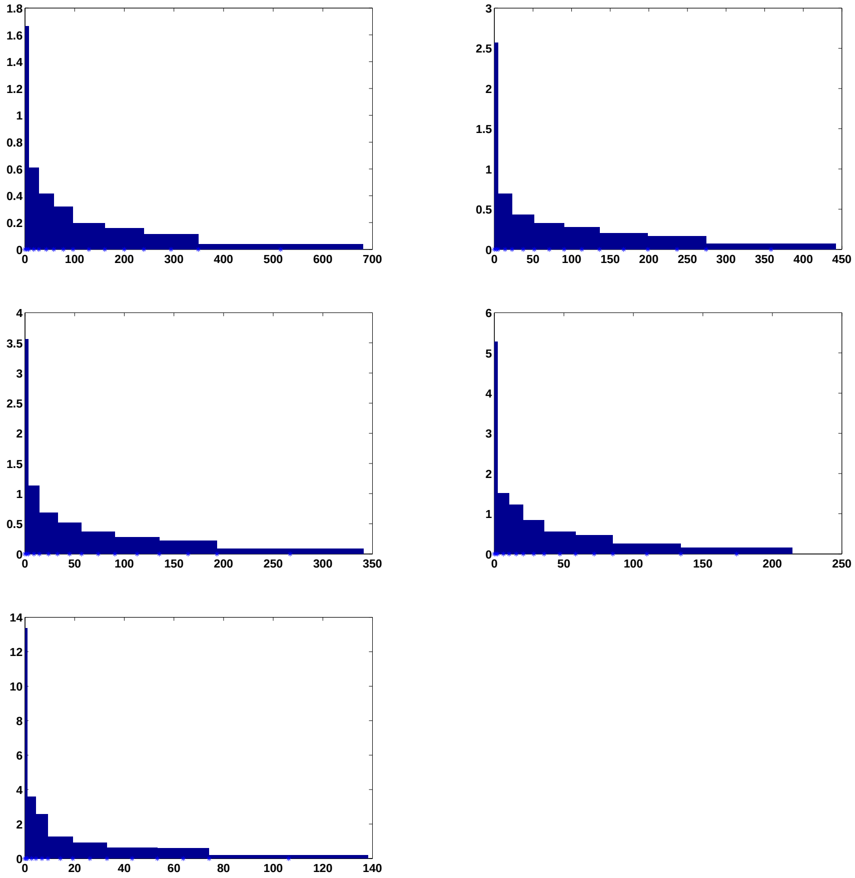


Figure 5.12: The histogram data representing the centroids of the 5 macro-clusters for the light intensity data set.

new histogram H_i, $i = 51, \ldots, 944$ its mean value and the corresponding quantile function have been evaluated with respect to the five summary of the two box-plot.

For example, we can notice a outlier with respect to the mean (red dot in the

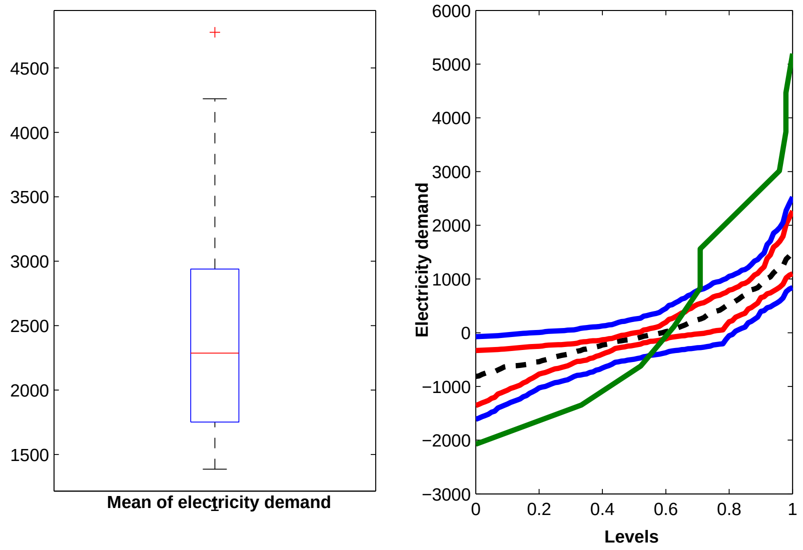


Figure 5.13: From left to right: mean values box-plot and quantile functions box-plot related to the electricity demand.

left box-plot) and the shape (green curve in the right box-plot). Therefore, it represents a histogram whose mean value is higher than upper whisker and in terms of shape, its quantile function intersects the lower and the upper whisker quantile functions. In data stream analysis, the frequency of the observed (potential) outliers is a hint of the evolution of the data stream. Different kinds of outlyingness (with respect to the mean or to the shape, or both) can suggest a different types of evolution of the stream. Consequently, the proposed box-plot can be updated with a new five quantile functions computed considering the new changes occurring along time.

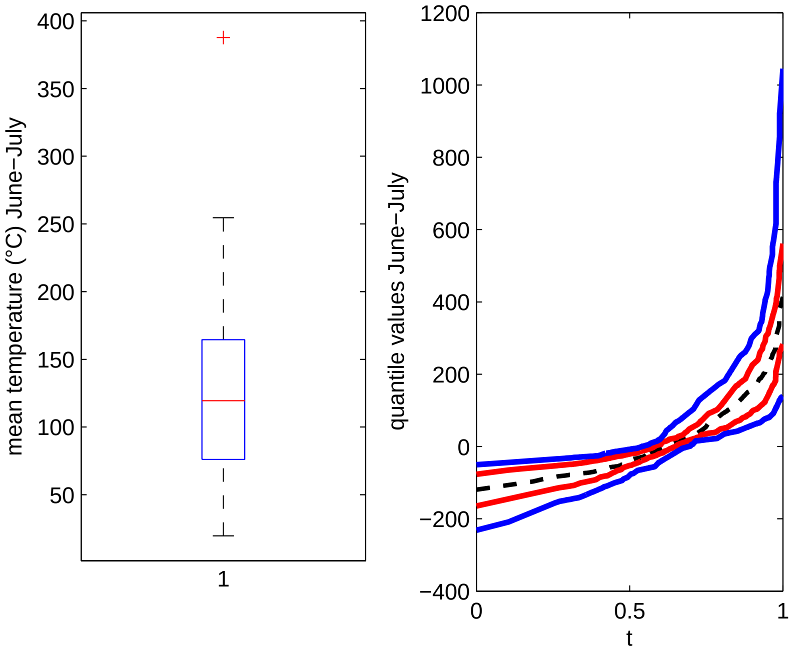


Figure 5.14: From left to right: mean values box-plot and quantile function box-plot related to the light intensity recorded from 22 June to 22 August 2012.

5.2.2 Results for light intensity data set

We have considered each day of monitoring as a different time window whose size ws is not set a priori and for each time window, an equi-depth histogram is computed by using a number of bins equal to 8. Unlike the previous case, we have chosen to compare the box-plots related to different periods of time as shown in Fig. 5.14, 5.15, 5.16. The first box-plot is obtained considering the first 60 time windows in order to capture the behavior of light intensity during the period from 22 June to 22 August 2012; the second one the next 60 time windows related to the period from 22 August to 22 October 2012 and the third one consists in the last 32 time windows for evaluating the period from 22 October to 19 November 2012.

Comparing the three mean values box-plots, we see that are very informative in terms of mean range of variability. Especially the three cases highlight three different mean range of variability. The first from 20 to 250, the second from 10 to 230 and finally the third from 2 to 230.

More interesting results can be observed in terms of changes in the data by the quantile function box-plot. The three boxplots differs mostly for: the median histogram, that can be interpreted as the most representative observed pattern of the light intensity data; the central region, that gives a less biased visualization of the histogram changes.

More information is detected by observing the box (the two red lines) in the three cases. It highlights for the first boxplot that changes of variability in histogram data are lower among levels 0.2 and 0.4, since the trend of the quartile functions are nearest around the median histogram. In the same way but for different levels this happens for the second and third boxplot. Especially the second and the third boxplot shows a low variability in changes among the levels 0.7 and 0.8. Other information about the main changes can be detected for the second and third boxplot by observing the wishers(the blu lines). Especially the third boxplot highlights the presence of a more wide range of variability in the histogram on the contrary to the second one.

In the Fig. 5.14, we notice that there is one detected outlier with respect to the average. Therefore, it represents a histogram whose mean value (red point in the left box plot) is highest in magnitude than upper whisker. In terms of the shape, the curves show a moderate degree of skewness.

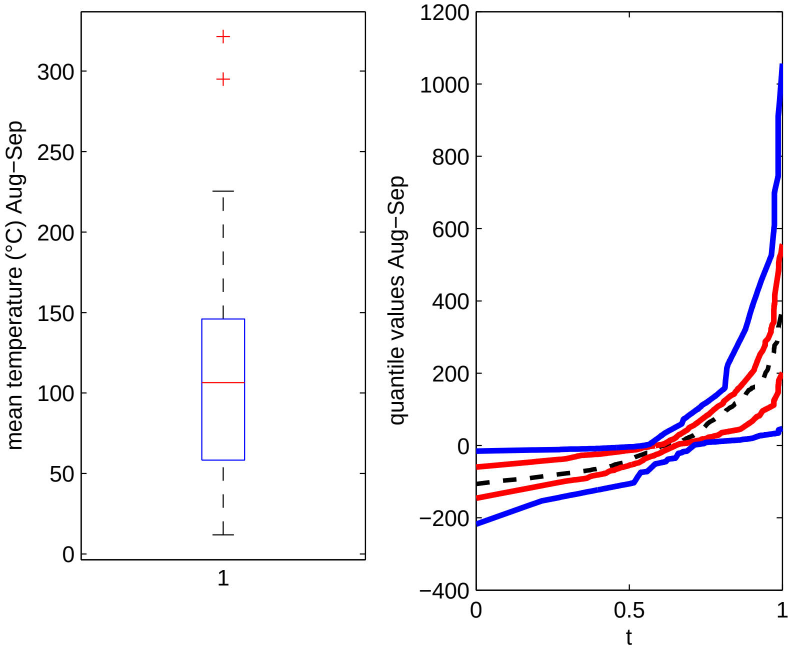


Figure 5.15: From left to right: mean values box-plot and quantile function box-plot related to the light intensity recorded from 22 August to 22 October 2012.

5.3 Summary

This chapter presented the effectiveness of the two proposals described in the previous chapters: the CluStream for quantile functions and quantile functions boxplot representation. These were applied to electricity demand and light intensity datasets. We have shown as the techniques proposed can be used for summarizing, analyzing and detecting outlier in data stream context.

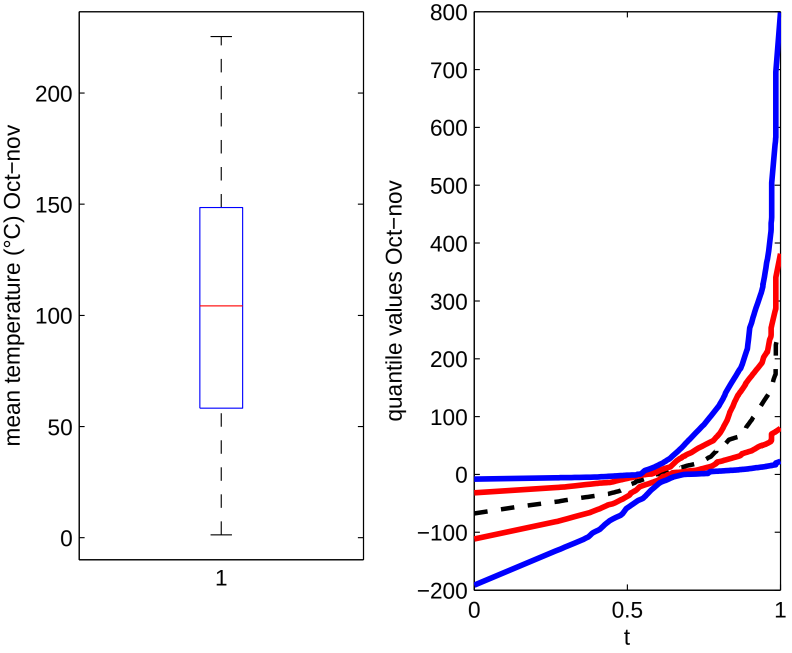


Figure 5.16: From left to right: mean values box-plot and quantile function box-plot related to light intensity recorded from 22 October to 19 November 2012.

Chapter 6

Conclusions and perspectives

In the last years the analysis of data stream has gained a lot of attention. In this thesis two innovative statistical based approaches for addressing the problem of knowledge discovery from data stream have been proposed.

The problems of representing, summarizing and monitoring the evolution over time of data streams have been addressed.

In particular, for what concern the problem of representing data stream, histogram has been proposed in a diverse way as graphical tool for representing changes in the stream. In fact summaries are expressed by a set of histogram that represent main concepts in the data. A finer summarization of the data is then obtained by a CluStream based strategy extended to histogram data.

In relation to the other mentioned problem of monitoring the evolution over time of data streams, a second more complex tool has been proposed: the boxplot for histogram data. It extends the classic idea of box-plot to a set of quantile functions associated to a set of histograms. Especially since histogram summarize a set of time windows, the boxplot monitors the evolution over the time and potential outliers in a data stream with respect to the mean and the shape of the histogram distribution.

Further developments on this work will concern the experimentation of multidimensional graphical tool for summarizes changes in multiple streaming data.

Appendix

```

function [microCluster]=clustering(dati,wS,nBin,sogliaMC)
[r c]=size(dati);
cont=1;
[bin,altezze]=Istogrammi(dati(1,cont:cont+wS-1),nBin);
for i=1:nBin
   centri(i)=(bin(i,1)+bin(i,2))/2;
   raggi(i)=(bin(i,2)-bin(i,1))/2;
end
microCluster(1).centri=centri;
microCluster(1).raggi=raggi;
microCluster(1).num=1;
microCluster(1).t=1;
contMC=1;
contw=1;
while cont<c-wS-1
[bin,altezze]=Istogrammi(dati(1,cont:cont+wS-1),nBin);
for i=1:nBin
centri(i)=(bin(i,1)+bin(i,2))/2;
raggi(i)=(bin(i,2)-bin(i,1))/2;
end
```

```
for i=1:contMC
dist(i)=Distanza(centri,raggi, microCluster(i).centri,
    microCluster(i).raggi);
end
[minDist,idx] = min(dist);
if minDist<=sogliaMC
microCluster(idx).centri=microCluster(idx)centri*
    microCluster(idx).num;
microCluster(idx).centri=microCluster(idx)centri+centri;
microCluster(idx).centri=microCluster(idx).centri/(
    microCluster(idx).num+1);
microCluster(idx).raggi=microCluster(idx).raggi*
    microCluster(idx).num;
microCluster(idx).raggi=microCluster(idx).raggi+raggi;
microCluster(idx).raggi=microCluster(idx).raggi/
    microCluster(idx).num+1);
microCluster(idx).num=microCluster(idx).num+1;
numt=length(microCluster(idx).t);
microCluster(idx).t(numt+1)=contw;
end
if (minDist>sogliaMC)
contMC=contMC+1;
microCluster(contMC).centri=centri;
microCluster(contMC).raggi=raggi;
microCluster(contMC).num=1;
microCluster(contMC).t=contw;
end
cont=cont+wS;
contw=contw+1;
end
```

```matlab
function [istBin]=ConversioneIstogrammi(istCentri,istRaggi)
istCentri=istCentri';
istRaggi=istRaggi';
[r c]=size(istCentri);
for i=1:r
istBin(i,1)=istCentri(i,1)-istRaggi(i,1);
istBin(i,2)=istCentri(i,1)+istRaggi(i,1);
end
```

```matlab
function [dist]=Distanza(centri1,raggi1,centri2,raggi2)
[r c]=size(centri1);
dist=0;
for i=1:r
dist=dist+((centri1(i)-centri2(i))^2-(1/3*(raggi1(i)-raggi2(i))^2));
end
dist=dist/r;
```

```matlab
function [bink1 bink2]=fullClustering(data,ws,nbin,soglia,k)
datab=[data(1:50000);data(50001:60000);data(70001:120000)];
[microCluster]=clustering(datab',ws,nbin,soglia);
[macroCluster,IDX,criterio]=macroClustering(microCluster,k,nbin);
[bink1]=ConversioneIstogrammi(macroCluster(1,1).centri,macroCluster(1,1).raggi);
[bink2]=ConversioneIstogrammi(macroCluster(1,2).centri,macroCluster(1,2).raggi);
```

```
size(microCluster)
```

```
function [bin,altezze]=Istogrammi(dati,numBin)
[r c]=size(dati);
datiOrdinati=sort(dati);
freqBin=floor(c/numBin);
cont=0;
estremoInferiore=min(dati);
bin(1,1)=estremoInferiore;
bin(1,2)=datiOrdinati(1,cont+freqBin);
cont=cont+freqBin;
altezze(1)=freqBin/(bin(1,2)-bin(1,1));
for i=2:numBin
bin(i,1)=bin(i-1,2);
bin(i,2)=datiOrdinati(1,cont+freqBin);
altezze(i)=freqBin/(bin(i,2)-bin(i,1));
cont=cont+freqBin;
end
```

```
function [macroCluster,IDX,criterio]=macroClustering(microCluster,k,nBin)
clear macroCluster
[r c]=size(microCluster);
for i=1:c
IDX(i)=floor(rand*k)+1;
end
for iter=1:10
for j=1:k
```

```
idxCluster=find(IDX==j);
[rclust cclust]=size(idxCluster);
totPesi=0;
for n=1:cclust
totPesi=totPesi+microCluster(idxCluster(n)).num;
end
for n=1:nBin
macroCluster(j).centri(n)=0;
macroCluster(j).raggi(n)=0;
for m=1:cclust
macroCluster(j).centri(n)=macroCluster(j).centri(n)+
    microCluster(idxCluster(m)).centri(n)*microCluster(
    idxCluster(m)).num;
macroCluster(j).raggi(n)=macroCluster(j).raggi(n)+
    microCluster(idxCluster(m)).raggi(n)*microCluster(
    idxCluster(m)).num;
end
end
macroCluster(j).centri=macroCluster(j).centri/totPesi;
macroCluster(j).raggi=macroCluster(j).raggi/totPesi;
end
criterio(iter)=0;
for i=1:c
for j=1:k
dist(j)=Distanza(macroCluster(j).centri, macroCluster(j).
    raggi,microCluster(i).centri, microCluster(i).raggi);
end
[minDist,IDX(i)] = min(dist);
criterio(iter)=criterio(iter)+minDist;
end
end
```

```
[r c]=size(macroCluster);
wS=100
for i=1:c
[istBin]=ConversioneIstogrammi(macroCluster(i).centri,
    macroCluster(i).raggi);
for j=1:nBin
altezze(j)=(wS/nBin)/abs((istBin(j,2)-istBin(j,1)));
end
[datiStack,altezzeStack]=plotIstogramma(istBin,altezze,1);
end
```

```
function [datiStack,altezzeStack]=plotIstogramma(bin,
    altezze,prec)
[r c]=size(bin);
cont2=1;
for i=1:r
ampiezza=bin(i,2)-bin(i,1);
step=ampiezza/prec;
cont=0;
for j=1:prec
dati(i,j)=bin(i,1)+cont;
cont=cont+step;
datiStack(cont2)=dati(i,j);
altezzeStack(cont2)=altezze(i);
cont2=cont2+1;
end
end
figure()
bar(datiStack,altezzeStack,'histc')
```

```
function []=plotmicroClusters(microClusters)
[r c]=size(microClusters);
[t nbins]=size(microClusters(1).centri);
for j=1:nbins
  altezze(j)=100/nbins;
end
for i=1:c
clear istBin;
[istBin]=ConversioneIstogrammi(microClusters(i).centri,
    microClusters(i).raggi);
[datiStack,altezzeStack]=plotIstogramma(istBin,altezze,1)
end
```

```
function [switches]=plotMicroTempo(microCluster,numFinestre
    )
[r c]=size(microCluster);
switches=zeros(c,numFinestre);
for cont=1:c
num=length(microCluster(cont).t);
for j=1:num
switches(cont,microCluster(cont).t(j))=1;
end
end
w=1:numFinestre;
figure
for cont=1:c
subplot(c,1,cont)
```

```
plot(w,switches(cont,:),'.k')
end
```

```
function []=plottingCentroidi(macroCluster,nBin,wS)
[r c]=size(macroCluster);
for i=1:c
[istBin]=ConversioneIstogrammi(macroCluster(i).centri,macroCluster(i).raggi);
for j=1:nBin
altezze(j)=(wS/nBin)/abs((istBin(j,2)-istBin(j,1)));
end
[datiStack,altezzeStack]=plotIstogramma(istBin,altezze,1);
end
```

```
function [vet]=QQPlotIstogrammi(bin, centri, raggi)
[istBin]=ConversioneIstogrammi(centri,raggi);
figure()
plot(bin,istBin)
vet=1;


classdef distribution
%distribution is a class that describes cumulative distribution functions


    properties
        domain %the domain of the distribution
        cumulat %the cdf (must have the same sive of domain)
        m %the mean
        s %the standard deviation
    end

    methods

        function obj=distribution(c) %costruttore
            if nargin>0
                if isa(c,'distribution')
                    obj = c(:).';
                else
                    obj.domain=c(:,1);
                    obj.cumulat=c(:,2);
                    obj.m=0;
                    obj.s=0;
                    obj.m=meanH(obj);
                    obj.s=stdH(obj);
                end
            else
                obj.domain=[];
                obj.cumulat=[];
                obj.m=[];
                obj.s=[];
            end
        end
```

```
    %1D_Squared_Wassertein_L2_distance between two histograms
```

```
function [WD, dm, ds, dr]=WASS_DIST_Q_1D(o1,o2)
WD=0;
dm=0;
ds=0;
dr=0;
if isa(o1,'distribution')&&isa(o2,'distribution')
    % o1=distribution(o1);
    % o2=distribution(o2);
    m1=o1.m;
    m2=o2.m;
    std1=o1.s;
    std2=o2.s;
    r=rho(o1,o2);
    dm=(m1-m2)^2;
    ds=(std1-std2)^2;
    dr=2*std1*std2*(1-r);
else
    if isa(o1,'distribution')
        dm=(o1.m-o2)^2;
        ds=o1.s^2;
    else
        if isa(o2,'distribution')
            dm=(o2.m-o2)^2;
            ds=o2.s^2;
        end
    end
end
WD=dm+ds+dr;
end
```

```
function mu=meanH(o1) % Average value associated to cdf
if isa(o1,'distribution')
    centers=(o1.domain(1:end-1,1)+o1.domain(2:end,1))./2;
    w=(o1.cumulat(2:end,1)-o1.cumulat(1:end-1,1));
    mu=centers'*w;
else
    error('o1 is not a distribution')
end
end
```

```
function s=stdH(o1) %Standard deviation associated to cdf
if isa(o1,'distribution')
    centers=(o1.domain(1:end-1,1)+o1.domain(2:end,1))./2;
    radii=(o1.domain(2:end,1)-o1.domain(1:end-1,1))./2;
    w=(o1.cumulat(2:end,1)-o1.cumulat(1:end-1,1));
    s=w'*(centers.*centers+1/3*radii.*radii)-(o1.m)^2;
    s=sqrt(s);
else
    error('o1 is not a distribution')
end
end
```

```
function DM=mean_set(o1,w)%Wassertein barycenter

n=length(o1);
if nargin<2
    w=ones(n,1);
end
```

```
w=w(:,1)./sum(w(:,1));

DM=o1(1,1);
DM.domain=w(1,1)*DM.domain;
if n>1
    for i=2:n
        tmpo=o1(i,1);
        tmpo.domain=w(i,1)*tmpo.domain;
       DM=DM+tmpo;
    end
   DM.domain=DM.domain;
   DM.m=meanH(DM);
   DM.s=stdH(DM);
end
end

\begin{lstlisting}[frame=trbl]{}
function STD=std_set(o1,mea)%Wassertein std
if nargin<2
    mea=mean_set(o1);
else
    if isa(mea,'distribution')==0
        error('the second input must be a distribution')
    end
end
n=length(o1);
STD=0;
if n>1
    for i=1:n
        STD=STD+(o1(i,1).s)^2+(o1(i,1).m)^2+(mea.s)^2+(mea.m)^2;%+...
        %  -2*(o1(i,1)*mea);
    end
    STD=STD- 2*n*(mea*mea);
    STD=sqrt(STD/n);
end
```

```
function resu=posdis_set(o1,pos) %the pos quantile F^-1(pos) 0<=pos<=1
n=size(o1,1);
if (pos>n)||(pos<0)
    resu=[];
    warning('p must be integer and not grater then the number of distributions');
    return
end

POS_DIS=multiple_register(o1);
nint=size(POS_DIS,1)-1;
resu=[];
tmp2=0;
for i=1:nint
    interv=zeros(n,2);
    for j=1:n
        interv(j,:)=[POS_DIS(i,j) POS_DIS(i+1,j)];
    end
```

```
    function val=LRarea(o1,o2)
%positive and negative area of o1 w.r.t o2
```

```
%o2 is the reference
%area tra due distribuzioni
%val=[LEFTarea RIGTHarea];
        resu=register(o1,o2);
        n=length(resu(:,3));
        val=[0 0];
        for i=1:n-1
            tmp=LRareaint(resu(i,1),resu(i+1,1),resu(i,2),
                resu(i+1,2));
            w=resu(i+1,3)-resu(i,3);
            val=val+w*tmp;
        end
    end
```

```
function qua=quant_set(o1,q) %q-th quantile of o1
n=size(o1,1);
%distribuzione quantile
p1=floor(1+(n-1)*q);
p2=ceil(1+(n-1)*q);
w1=1+(n-1)*q;

if p1==p2
    %una sola distr
    if p1==0;
    else
        resu=posdis_set(o1,p1);
    end
    qua=distribution(resu);
else
    %due distrib
    resu=posdis_set(o1,p1);
    qua1=distribution(resu);
    resu=posdis_set(o1,p2);
    qua2=distribution(resu);
    w(1,1)=(p2-w1)/(p2-p1);
    w(2,1)=1-w(1,1);
    qua=mean_set([qua1;qua2],w);
end

end
```

```
function [distrs, IQR]=boxplot1(o1,whi) %boxplot of qfs
%boxplot di distribuzioni punto punto
% mediana q1 q3 baffi ad 1.5
distrQ1=quant_set(o1,0.25);
distrME=quant_set(o1,0.5);
distrQ3=quant_set(o1,0.75);
IQR=area(distrQ3,distrQ1);
distrMIN=quant_set(o1,0);
distrMAX=quant_set(o1,1);
if nargin<2
    %whisker 1
    distrW1= distrMIN;
    %whisker 2
    distrW2= distrMAX;
else
    if length(whi)<2
        shift=(whi-1)*(distrQ3.m-distrQ1.m);
        %whisker 1
        distrW1=distrQ1;
        distrW1.domain=distrW1.domain-shift;
        distrW1.m=distrW1.m-shift;
        %whisker 2
        distrW2=distrQ3;
        distrW2.domain=distrW2.domain+shift;
        distrW2.m=distrW2.m+shift;
    else
        distrW1=quant_set(o1,min(whi));
        distrW2=quant_set(o1,max(whi));
    end
end
distrs=[distrMIN;distrW1;distrQ1;distrME;distrQ3;distrW2;distrMAX];
end
```

```
function [val, distr]=skewness(o1) %Bowley index extended to qfs
%Bowley measure of skewness
%sk=((Q3-ME)-(ME-Q1))/(Q3-Q1);
distrQ1=quant_set(o1,0.25);
distrME=quant_set(o1,0.5);
distrQ3=quant_set(o1,0.75);
a1=area(distrQ3,distrME);
a2=area(distrQ1,distrME);
a3=area(distrQ3,distrQ1);
val=(a1-a2)/a3;
resu=multiple_register([distrQ3;distrME;distrQ1]);
d1=resu(:,1)-resu(:,2);
d2=resu(:,2)-resu(:,3);
d3=resu(:,3)-resu(:,1);
```

```
num=d1-d2;
 distr=[num./(d3+eps) resu(:,end)];
end
```

```
function g=plot_distr(o1,para) %plot a distribution
n=size(o1,1);
g=0;
if nargin<2
    para='-r';
end
hold on
for i=1:n
    %plotta una distribuzione
    plot(o1(i,1).domain, o1(i,1).cumulat,para);
end
hold off
end
```

```
function plot_hist(o1,col) %plot a histogram
n=size(o1,1);

if nargin<2
    col='r';
end
hold on
for i=1:n
    for j=1:(size(o1(i,1).cumulat)-1)
        %plotta un istogramma
        basemin=o1(i,1).domain(j,1);
        basemax=o1(i,1).domain(j+1,1);
        if (basemax-basemin)>eps
            heigthh=(o1(i,1).cumulat(j+1,1)-o1(i,1).cumulat(j,1))/(basemax-basemin);
        else
            basemax=basemax+eps;
            heigthh=(o1(i,1).cumulat(j+1,1)-o1(i,1).cumulat(j,1))/(basemax-basemin);
        end
        patch([basemin basemax basemax basemin basemin],[0 0 heigthh heigthh 0],col,'FaceAlpha',0.5);
    end
end
hold off
end
end
end
```

Bibliography

[1] Aggarwal, C. (2003). A Framework for Clustering Evolving Data Streams. ACM SIGMOD Conference.

[2] Aggarwal, C. (2007). Data Streams Models and Algorithms. Series: Advances in Database Systems, Vol. 31, Springer.

[3] Aggarwal, C., Han J., Wang, J., Yu, P.S. (2004). On Demand Classification of Data Stream Data Streams. Knowledge Discovery and data Mining (KDD'04), Settle, WA.

[4] Aggarwal, C. (2006). On biased reservoir sampling in the presence of stream evolution. In: 32nd International Conference on Very large data bases, pp. 607–618.

[5] Agrawal, R. and Srikant, R. (1994). Fast Algoritms for Mining Association Rules. In: VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499.

[6] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In: ICDE '95 Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14.

[7] Agrawal, R., Faloutsos, C., Swami, A. (1993). Efficient similarity search in sequence databases. In: 4th International Conference, FODO '93, pp. 69–84.

[8] Antunes, C. M., Oliveira, A. L. (2001). Temporal Data Mining: an overview. Lecture Notes in Computer Science.

[9] Arroyo, J., Mate, C., Roque, A. (2009). Hierarchical clustering for boxplot variables. Studies in Classification, Data Analysis and Knowledge Organization, Part II, pp. 59–66.

[10] Arroyo, J., Mate, C., Roque, A. (2011). Smoothing Methods for Histogram-valued Time Series. An application to Value-at-Risk. Statistical Analysis and Data Mining **4** (2), pp. 216–228.

[11] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom J. (2002). Models and Issues in Data Stream Systems. In: 21st ACM Symposium on Principles of Database Systems.

[12] Babcock, B., Datar, M., Motwani, R. (2002). Sampling from a moving window over streaming data. In: 13th annual ACM-SIAM symposium on Discrete algorithms, pp. 633–634.

[13] Balzanella A., Verde R., Lechevallier, Y. (2010). Clustering highly evolving multiple data streams. In: Actes des rencontres de la Société Francophone de Classification. Saint-Denis de la Reunion, 9-11 giugno 2010. pp. 17–20.

[14] Balzanella, A., Rivoli, L., Verde, R. (2011). Data stream summarization by histograms clustering. Conference proceedings of CLADAG 2011

[15] Balzanella A., Irpino A., Verde R. (2010). Dimensionality reduction techniques for streaming time series: a new symbolic approach. In: Classification as a Tool for Research; vol. 1, p. 381–390, Locarek-Junge, Hermann, Weihs, Claus Eds. ISBN/ISSN: 978-3-642-10744-3

[16] Beringer J., Hullermeier E. (2006). Online clustering of parallel data streams. In: Data and Knowledge Engineering, Vol. 58(2), pp. 180–204.

[17] Bertrand, P., Goupil, F. (2000). Descriptive statistics for symbolic data. In: Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data, pp. 103–124, Springer Berlin Heidelberg.

[18] Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen (2006). Adaptive Clustering for Multiple Evolving Streams. In IEEE Transactions On Knowledge And Data Engineering, Vol. 18, No. 9.

[19] Bickel, P.J., Freedman, D.A. (1981). Some asymptotic theory for the bootstrap. In: Ann. Statist., Vol. 9, pp. 1196–1217.

[20] Billard, L., Diday, E. (2003). From the statistics of data to the statistics of knowledge: Symbolic data analysis. Journal of the American Statistical Association 98, pp. 470–487.

[21] Billard, L., Diday E. (2006). Symbolic Data Analysis: conceptual statistics and data Mining. Wiley series in computational statistics.

[22] Braverman, V., Ostrovsky, R., Zaniolo, C. (2009). Optimal Sampling from Sliding Windows. In: ACM PODS.

[23] Babu, S., Widom, J. (2001). Continuous Queries over Data Streams. SIGMOD Record, Vol. 30 (3).

[24] Charikar M., Chen K., Farach-Colton M. (2004). Finding frequent items in data streams. Theor. Comput. Sci., pp. 312–315

[25] Chan, K., Wai-Chee Fu, A. (1999). Efficient Time Series Matching by wavelets. In Proceedings of the 15th International Conference on Data Engineering.

[26] S. Chaudhuri, R. Motwani, Narasayya, V. (1999). On random sampling over joins. In Proc. of the ACMSIGMOD Intl. Conf. on Management of Data, pp. 263–274.

[27] Cormode G., Muthukrishnan S. (2005). An improved data stream summary: the count-min sketch and its applications. J. Alg., Vol. 55(1), pp.58–75.

[28] Da Silva A., Lechevallier Y., Rossi F., de Carvalho F. A. T. (2009). Clustering Dynamic Web Usage Data. In: Innovative Applications in Data Mining, Studies in Computational Intelligence Volume 169/2009, 71–82, Springer Berlin / Heidelberg

[29] Dall'Aglio, G. (1956). Sugli estremi dei momenti delle funzioni di ripartizione doppia. Ann. Scuola Norm. Sup. Pisa Cl. Sci., Vol. 3(1), pp. 33–74

[30] Diday, E. (1971). La methode des Nuees dynamiques Revue de Statistique Appliquee, 19, 2, 19–34.

[31] Diday, E., Simon, J.C. (1976). Clustering Analysis. In: Fu, K. S. (Eds.): Digital Pattern Recognition. Springer-Verlag, Heidelberg, 47–94.

[32] Dobra, A., Gehrke, J., Garofalakis, M. and Rastogi, R. (2002). Processing complex aggregate queries over data streams. In Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data.

[33] Domingos, P., Hulten, G. (2000). Mining high-speed data streams. Paper presented at the Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.

[34] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases.

[35] Gaber, M. M., Zaslavsky, A., Krishnaswamy, S. (2005). Mining data streams: a review. SIGMOD Rec. 34, 2, pp. 18–26.

[36] Gama, J., Medas, P., Castillo, G., Rodrigues, P. (2004). Learning with drift detection. In SBIA Brazilian Symposium on Artificial Intelligence, pp. 286-295, Springer Verlag.

[37] Gama, J., Pinto, C. (2006). Discretization from data streams: Applications to histograms and data mining. In: Proceedings of the ACM symposium on Applied computing, pp. 662-667. ACM, New York USA

[38] Gama, J., Fernandes, R., Rocha, R. (2006). Decision trees for mining data streams. Intell. Data Anal. 10, 1, 23–45.

[39] Gama, J., Gaber, M.M. (2007). Learning from Data Streams: Processing Techniques in Sensor Networks.Ed. Springer Verlag.

[40] Gama, J. (2010). Knowledge Discovery from Data Streams. Chapman and Hall/CRC Data Mining and Knowledge Discovery Series.

[41] Ganguly, A. R., Gama, J., Omitaomu, O. A., Gaber, M. M., Vatsavai, R. R. (2009). Knowledge discovery from sensor data. CRC Press.

[42] Gehrke, J., Korn, F., Srivastava, D. (2001). On computing correlated aggregates over continual data streams. In: SIGMOD Conference, pp. 13–24. ACM Press, New York.

[43] Gibbs, L. A., Su, F. E. (2002). On Choosing and Bounding Probability Metrics. International Statistical Review, Vol. 70, pp. 419–435.

[44] Gibbons, P. B., Matias, Y. (1998). Synopsis Data Structures for Massive Data Sets. In: External Memory Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science.

[45] Gilbert, A., Guha, S., Indyk, P., Kotidis, Y., Muthukrishnan, S., Strauss, M. (2002). Fast, small-space algorithms for approximate histogram maintenance. In: Annual ACM Symposium on Theory of Computing.

[46] Gilbert, A., Kotidis, Y., Muthukrishnan, S., Strauss, M. (2003). Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. Journal version in IEEE Transactions on Knowledge and Data Engineering, 15(3).

[47] Greenwald, J., Khanna, F. (2001). Space Efficient On-Line Computation of Quantile Summaries. In: Annual ACM Int. Conf. on Management of Data.

[48] Guha, S., Mishra, N., Motwani, R., O'Callaghan, L. (2000). Clustering data streams. In: the Annual Symposium on Foundations of Computer Science, IEEE.

[49] Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L. (2003). Clustering Data Streams: Theory and Practice. TKDE special issue on clustering, vol. 15.

[50] Guha, S., Khuller, S., (1998). Greedy strikes back: Improved facility location algorithms. Proc. SODA, pages 649–657.

[51] Guha, S., Koudas, N., Shim, K. (2001). Data-streams and histograms. In: Annual ACM Symp. on Theory of Computing, pages 471–475.

[52] Han, J., Kamber, M. (2006). Data Mining Concepts and Techniques. Morgan Kaufmann, San Mateo.

[53] Harries, M. (1999). Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales.

[54] Hulten, G., Spencer, L., Domingos, P. (2001) Mining Time-Changing Data Streams, In: Proc. of ACM SIGKDD.

[55] Huber, P.J. (1981). Robust statistic. John Wiley

[56] Irpino, A., Verde, R. (2006). Dynamic clustering of histograms using Wasserstein metric. In: Rizzi A., Vichi M. COMPSTAT 2006 - Advances in computational statistics, pp. 869-876. Heidelberg, Physica-Verlag.

[57] Jagadish, H., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K., Suel, T. (1998). Optimal histograms with quality guarantees. In Proc. of the Intl. Conf. on Very Large Data Bases, pp. 275–286.

[58] Kantorovich, L.V. (1940). On one effective method of solving certain classes of extremal problems. Dokl. Akad. Nauk USSR, Vol. 28, pp. 212–215.

[59] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In proceedings of ACM SIGMOD Conference on Management of Data, pp. 21–24.

[60] Klinkenberg, R., Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. In; Learning for Text Categorization, Madison, WI, pp. 33–40. AAAI Press.

[61] Klinkenberg, R., Joachims, T. (2000). Detecting Concept Drift with Support Vector Machines. In: 17th International Conference on Machine Learning, Stanford, US.

[62] Klinkenberg, R. (2004). Learning Drifting Concepts: Example Selection vs. Example Weighting. In: Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, Vol. 8, No. 3, pages 281–300.

[63] Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases, Vol. 30.

[64] Lauro, N. C. (1996). Computational statistics or statistical computing, is that the question? In: Computational Statistics and Data Analysis, Vol. 23, 191–193.

[65] Lazarescu, M., S. Venkatesh, Bui, H. (2004). Using multiple windows to track concept drift. Intelligent Data Analysis 8 (1), 2960.

[66] Lin J., Keogh E., Lonardi S., Chiu B. (2003). A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD. San Diego, CA. June 13.

[67] Liu, R. Y. and Parelius, J. M., Singh, K. (1999). Multivariate analysis by data depth: Descriptive statistics, graphics and inference. Ann. Stat. Vol. 27(3), pp. 783–858.

[68] Mallows, C.L. (1972). A note on asymptotic joint normality. Annals of Mathematical Statistics, 43(2), pp. 508–515.

[69] Manku, G., Rajagopalan, S., Lindsay, B.G (1998). Approximate medians and other quantiles in one pass and with limited memory. In ACM SIGMOD, pp. 426–435.

[70] Manku, G., Rajagopalan, S., Lindsay, B.G. (1999). Random sampling techniques for space efficient on-line computation of order statistics of large datasets. In ACM SIGMOD, pp. 251–262.

[71] Mitsa T. (2010). Temporal Data Mining. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC

[72] Morinaka, Y., Yoshikawa, M., Amagasa, T., Uemura, S. (2001). The L-index: An indexing structure for efficient subsequence matching in time sequence databases. PAKDD 2011.

[73] Mousavi H., Zaniolo, C. (2011). Fast and accurate computation of equi-depth histograms over data streams. In Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT ’11) ACM, New York, pp. 69–80.

[74] Muthukrishnan, S. M. (2005). Data Streams: Algorithms and Applications. In Foundations and Trends in Theoretical Computer Science, Vol. 1.

[75] Laxman, S., Sastrya, P. S. (2006) A Survey of temporal data mining. Sadhana, Vol. 31.

[76] Rivoli, L., Irpino, A., Verde, R. (2012) The median of a set of histogram data. In: XLVI Riunione Scientifica della Società Italiana di Statistica, CLEUP. ISBN 978-88-6129-882-8.

[77] Rivoli, L., Irpino, A., Verde, R. (2012) Outliers detection for histogram data summarizing data streams. Conference proceedings of CLADAG2012.

[78] Rodriguess P. P., Pedroso J. P. (2006). Hierarchical clustering of time series data streams. In: Sixth SIAM International Conference on Data Mining, pp. 499–503.

[79] Rueshendorff, L. (2001). Wasserstein metric, Encyclopedia of Mathematics. Springer.

[80] Salvemini, T. (1943). Sul calcolo degli indici di concordanza tra due caratteri quantitativi. Atti della VI Riunione della Soc. Ital. di Statistica.

[81] Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. Monographs on Statistics and Applied Probability, Vol. 26, Chapman and Hall, London.

[82] Tsymbal, A. (2004). The problem of concept drift: Definitions and related work. Technical Report. Department of Computer Science, Trinity College, Dublin, Ireland.

[83] Tukey, J. W. (1977). Exploratory Data Analysis. Addison-Wesley, Reading, MA.

[84] Verde, R., Irpino, A. (2007). Dynamic clustering of histogram data: using the right metric. Studies in Classification, Data Analysis, and Knowledge Organization 2007 Part I, pp. 123–134, doi: 10.1007/978-3-540-73560-1, Vol. 12.

[85] Verde, R., Irpino, A. (2010). Ordinary Least Squares for Histogram Data Based on Wasserstein Distance. In: Lechevallier, Y., Saporta, G. - COMPSTAT 2010, pp.581–588. Physica Verlag, Springer, Berlin.

[86] Vapnik, V. (1998) Statistical Learning Theory. Wiley-Interscience, ISBN 0-471-03003-1.

[87] Vitter, J. (1985). Random sampling with a reservoir. ACM Trans. on Mathematical Software, Vol. 11(1).

[88] Wang, H., Fan, W., Yu, P., Han, J. (2003). Mining Concept-Drifting Data Streams using Ensemble Classifiers, In: Proc. of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) Washington DC, USA.

[89] Wasserstein, L.N. (1969). Markov processes over denumerable products of spaces describing large systems of automation. Probl. Inform. Transmission, Vol. 5, pp. 47–52.

[90] Widmer, G., Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine Learning 23, pp. 69–101.

[91] Yi, B., Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. Proceedings of the 26th Int. Conference on Very Large Databases, pp. 385–394.

[92] Zuo, Y. and R. Serfling (2000). General notions of statistical depth function. Ann. Statist, Vol. 28, pp. 461–482