Information Sciences 183 (2012) 35–47

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A clustering algorithm for multiple data streams based on spectral component similarity

Chen Ling [a,b,*], Zou Ling-Jun [a], Tu Li [c]

[a] Department of Computer Science, Yangzhou University, Yangzhou 225009, China
[b] State Key Lab of Novel Software Tech, Nanjing University, Nanjing 210093, China
[c] Department of Computer Science, Jiangyin Polytechnic Institute, Jiangyin 214405, China

ARTICLE INFO

Article history:
Received 12 September 2008
Received in revised form 1 September 2011
Accepted 5 September 2011
Available online 14 September 2011

Keywords:
Data streams
Clustering
Auto-regression model
Spectral component

ABSTRACT

We propose a new algorithm to cluster multiple and parallel data streams using spectral component similarity analysis, a new similarity metric. This new algorithm can effectively cluster data streams that show similar behaviour to each other but with unknown time delays. The algorithm performs auto-regressive modelling to measure the lag correlation between the data streams and uses it as the distance metric for clustering. The algorithm uses a sliding window model to continuously report the most recent clustering results and to dynamically adjust the number of clusters. Our experimental results on real and synthetic datasets show that our algorithm has better clustering quality, efficiency, and stability than other existing methods.

1. Introduction

Today, tremendous amounts of data and potentially infinite volumes of data streams are generated in many applications such as network intrusion detection, financial transaction flows, telephone call records, sensor streams, and meteorological data. Unlike the finite, statically stored data sets, a data stream is massive, continuous, temporally ordered, dynamically changing, and potentially infinite [8,19]. A typical example of stream data is the trading of public securities in the United States. The approximately 50,000 securities generate 100,000 quotes and trades per second [41]. For the stream data applications, the volume of data is usually too large to be stored or scanned more than once. Furthermore, because the data objects can be only sequentially accessed in the data streams, random data access techniques are not practical.

A related development is the increased demand for mining useful information from data streams. Examples of such increased demand are summarisation [41], similarity searches [4,16,21,22,33], classification [1,34], and clustering [2,3,6,10–14,17,18,20,23,26–32,35] of data streams. Clustering of data streams has attracted increasing interest. Data clustering partitions a data set into clusters such that members within the same cluster are similar in a certain sense and members of different clusters are dissimilar. Current clustering techniques can be broadly classified into several categories: partitioning methods (e.g., *k*-means and *k*-medoids), hierarchical methods (e.g., BIRCH [40]), density-based methods (e.g., DBSCAN [25,7]), and grid-based methods (e.g., CLIQUE [5]). However, these methods are designed only for static data sets and cannot be directly applied to data streams.

Recently, an abundant body of research on clustering data in one data stream emerged. O'Callaghan et al. [30] presented an algorithm called STREAM that was based on *k*-means, which adopts a divide-and-conquer technique to process buckets and obtain clusters. Guha et al. [17] also proposed an algorithm using a divide-and-conquer technique to cluster data on a

* Corresponding author at: Department of Computer Science, Yangzhou University, Yangzhou, 225009, China.
E-mail address: lchen@yzcn.net (L. Chen).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.09.004

single stream. CluStream [2] is an algorithm for clustering incoming data streams based on user-specified, online clustering queries. It divides the clustering process into on-line and off-line components. The online component computes and stores summary statistics of the data stream using micro-clustering, while the offline component performs macro-clustering and answers various user queries using the stored summary statistics. Lughofer [24] presents an incremental clustering method for data streams. A split-and-merge strategy is advanced to adjust the number of clusters adaptively during incremental clustering. In these studies, the problem that is addressed focuses on clustering the elements of one data stream. These studies motivate the online compression and offline computation framework but are obviously not applicable to our problem.

In this paper, we study the clustering of multiple and parallel data streams. Our study should be differentiated from some of the previous work on stream data clustering [2,30]. Our goal is to group multiple streams with similar behaviour and trends together, instead of clustering the data records within one data stream. There are various applications where it is desirable to cluster the streams. For example, the price of a stock may rise and fall from time to time. To reduce the financial risk, an investor may prefer to spread his investment over a number of stocks that exhibit different behaviours. Another application is in meteorological research and disaster prediction. Weather forecasters cluster data streams that are taken from different geographical regions and that have similar curvature trends to identify regions with similar meteorological behaviours. Yet another application is a supermarket that records its sales for different types of merchandise. Statistical analysis of the price trends on different items may reveal a high correlation between the sales of these items. Armed with this knowledge, the supermarket can manipulate the prices to maximise its profit.

In the problem of clustering multiple data streams, each data stream is an element to be clustered; this approach focuses on the similarity between the data streams. There are several previous studies on this problem. Yang [37] uses the weighted aggregation of snapshot deviations as the distance measure between two streams, which can reflect the similarity in the data values but ignores the trends in the streams. More recently, a COD framework [15] was suggested, which consists of two phases, the online maintenance phase and the offline clustering phase. Beringer and Huellermeier [9] proposed an online version of the *k*-means clustering algorithm, which uses a discrete Fourier transforms (DFT) approximation of the original data and utilises the low-frequency (instead of all) coefficients to compute the distance between two streams. Their method acts like a low-pass filter to smooth the data stream. Because the DFT transformation preserves Euclidean distances, the DFT distance is equivalent to the Euclidean distance of the smoothed data streams. Beringer and Huellermeier [9] also presented an incremental *k*-means method that can adaptively adjust the number of the clusters. Such adaptation is very important in stream data clustering, because the clustering structure may change over time.

A serious limitation of the previous studies mentioned above is that they are based on the Euclidean distance between data records. Because similar trends in data streams cannot be described by their Euclidean distance, these methods may discard important trend information that is contained in the data streams. For example, in stock markets, the absolute prices of stocks from similar areas or similar industries can be very different but their trends are similar. To capture such similarities, it is more appropriate to use correlation analysis, a statistical method for time series, than to measure the resemblance of the trends found in multiple data streams.

Some real streams or their subsequences may have lag correlations, which are highly similar rising/falling patterns that neglect some lags or shifts in the time axis. For example, stock *A* may affect another stock *B;* in this case, we would observe a rising (or a falling) in the price of stock *A* followed by a corresponding rising (or a falling) in the price of stock *B* after a certain period of time delay. This phenomenon may frequently occur in the stock market. Such streams should be clustered into the same cluster, and the information could be useful for the investors to make decisions. This problem is challenging because, in such streams, clustering and traditional similarity measures such as Euclidean distance or correlation coefficients cannot be helpful for revealing the lag similarity between the data streams. Y. Sakurai et al. presents a method called BRAID [33] to address the problem of discovering lag correlations among multiple streams. Their focus is on time and space efficient methods for finding the earliest peak and the highest peak in the cross correlation functions between all pairs of streams. They compute the cross-correlation at lag lengths that take on the values of a geometric progression.

In this paper, we propose a clustering algorithm for multiple data streams that is based on an auto-regressive (AR) modelling technique to measure the correlations between data streams. The AR model has been successfully used to analyse gene expression data [36,38,39]. Our algorithm uses an estimated frequency spectrum to extract the essential features of the data streams. Each stream is represented as the sum of the spectral components, and the correlation is measured component-wise. Each spectral component is described by four parameters, including the amplitude, phase, damping rate, and frequency. The ε-lag-correlation between two spectral components is calculated and used as a similarity measure in clustering the data streams. Our experimental results on real and synthetic datasets show that our algorithm has better clustering quality, efficiency, and stability than other related methods.

The remainder of this paper is organised as follows. After introducing the basic concepts and problem definitions in Section 2, we propose an algorithm that computes the spectral component-wise correlation in Section 3. Section 4 presents the algorithm for clustering multiple data streams under a sliding window. In Section 5, we show experiment results on synthetic and real data sets, which demonstrate the high accuracy, stability, and speed of our algorithm. We conclude the paper in Section 6.

2. Background

In this section, we introduce the background of this study and several basic concepts.