Procedia Computer Science

Volume 80, 2016, Pages 461–471

ICCS 2016. The International Conference on Computational Science

Improving Multivariate Data Streams Clustering

Christian C. Bones[1], Luciana A. S. Romani[2], and Elaine P. M. de Sousa[1]*

[1] University of São Paulo, São Carlos, SP, Brazil. {chris, parros}@icmc.usp.br
[2] Embrapa Agricultural Informatics, Campinas, SP, Brazil.
luciana.romani@embrapa.br

Abstract

Clustering data streams is an important task in data mining research. Recently, some algorithms have been proposed to cluster data streams as a whole, but just few of them deal with multivariate data streams. Even so, these algorithms merely aggregate the attributes without touching upon the correlation among them. In order to overcome this issue, we propose a new framework to cluster multivariate data streams based on their evolving behavior over time, exploring the correlations among their attributes by computing the fractal dimension. Experimental results with climate data streams show that the clusters' quality and compactness can be improved compared to the competing method, leading to the thoughtfulness that attributes correlations cannot be put aside. In fact, the clusters' compactness are 7 to 25 times better using our method. Our framework also proves to be an useful tool to assist meteorologists in understanding the climate behavior along a period of time.

Keywords: Clustering, Data Streams, Data Mining, Fractal

1 Introduction

Extracting valid and useful knowledge from data streams is an important and costly task for many environmental science fields such as ecology, geology, atmospheric science, to name a few. An increasing number of devices and sensors have generated a huge amount of data incessantly, leading to new challenges and applications. For instance, sensors have been used to monitor the pollution in cities, the level of rivers and the meteorological conditions. Extracting valuable information from these flows of data could be helpful to avoid disasters, such as flooding or the extinction of fragile plants due to sudden changes in temperature.

In this scenario, clustering of data streams becomes an active research topic [2, 12, 7, 5, 18, 14, 17] with applications in several contexts. Clustering aims to group in the same cluster data streams that have similar properties and behavior over time, whereas data streams of different clusters must present dissimilar characteristics. For illustration, clustering techniques could be applied to cluster meteorological data stream sensors that have similar behavior along a period of time.

*The authors are grateful to CAPES, CNPQ and FAPESP for their financial support.

doi:10.1016/j.procs.2016.05.325

However, clustering data streams requires appropriate solutions for challenging issues, mainly: to capture and represent data evolution along the time; to deal with all the attributes of each stream, i.e., multivariate data streams; to take into account the correlation among the attributes; to read data only once; to provide answers as soon as the user demands them; to deal with outlier data streams. Therefore, in the past few years some methods have been proposed to process and analyze flows of data in real time [15, 14, 18, 17, 8, 16, 1], aiming to overcome some of these challenges. However, only few of them try to pull out valuable information and group the entire data streams based on their similar behavior over the time [14, 17, 8, 16]. Furthermore, most of these methods either do not support multivariate data streams [14, 17, 16] or only consider the similarity of attributes independently [15, 13, 8].

In this paper, we propose a new approach to cluster multivariate data streams, taking into account the entire data streams and their (dis)similar behavior along the time, bearing in mind the data streams evolution. Our approach also considers the correlation among the attributes of each data stream, aiming to improve the clusters' quality. We thus propose the framework eFCDS - Evolving Fractal-based Clustering of Data Streams. Its main module is a novel algorithm for clustering multivariate data streams, based on the continuous calculation of the attributes correlation by the fractal dimension. Also, it tracks the evolution of the data streams by checking cluster membership whenever a new value of fractal dimension is obtained. In other words, our method checks whether the data stream still belongs to the same cluster or it should be allocated in another one that better describes its behavior in that period of time. Another module of eFCDS checks whether an outlier data stream could be associated to one of the regular clusters without disrupt the clusters' formation rules. It also detects overlapping between the generated clusters and merge them when their union does not extrapolate a maximum standard deviation.

Finally, we apply our framework to climate data streams from meteorological sensors, which usually have more than one attribute (e.g. temperature and precipitation) and presumably there are some correlations among them. We conducted an experimental study on data from different Brazilian regions, provided by Agritempo[1]. Our results not only indicated that our approach can be useful to assist specialists in analyzing large amounts of climate data, which is relevant to current climate research, but also helps to identify regions with the same behavior along the time.

The rest of this paper is organized as follows. Section 2 presents background concepts and related work. Section 3 describes our approach to cluster data streams. Experimental results are discussed in Section 4 and Section 5 presents final remarks and future work.

2 Background and Related Work

In order to be considered a data stream the data collection must be generated continuously by one or more sources, i.e, $d_1, d_2, ...$ [11]. Moreover, each d_i could have more than one attribute, characterizing it as a multivariate data stream. Formally, let $\mathbb{S} = \{S_1, \ldots, S_n\}$ be a set of data streams sources where each $S_i = \left\{\vec{d_1}, \ldots, \vec{d_\infty}\right\}$ is a multivariate data stream. Also, each S_i is assumed to contain f attributes such that $\vec{d_i} = [a_1, \ldots, a_f]$ is the set of attributes.

Clustering in a data stream environment can be very costly [11], due to some basic requirements that must be presented in the algorithms [3]: (i) Representation of compact size; (ii) Quickly and incremental processing of new data items; (iii) Traceability of changes in groups; (iv) Quick and clear identification of outliers.

[1] www.agritempo.gov.br

Methods that aim to overcome those challenges usually fall into two main categories, namely: *Clustering by example*, in which all data points are clustered independently, regardless of the data sources these data points are from; *Clustering by variable* or *Clustering the entire data stream*, in which sensor data streams are compared with each other and clustered according to their similarity in the analyzed time interval. The general idea of clustering the entire data stream is illustrated in Fig. 1. Notice that clusters can evolve over time ($T1$ to $T3$), i.e. data streams can move from one cluster to another, clusters can disappear or new clusters can be created. In this work we address the clustering by variable problem.

2.1 Clustering by Variable

In the literature, there are few methods which aim to cluster the entire multivariate data streams [8, 15, 13]. But they do not consider the correlation among attributes to cluster the data streams. Instead, attributes are analyzed individually such that two data streams are placed together in the same cluster only if all their related attributes are similar. This approach leads to results that may not correspond to the general behavior of the data streams along the time.

In general, algorithms to cluster data streams fall into one of two main categories: partitioning methods, such as ECM [17] and POD-Clus [8], and hierarchical methods, such as TS-Stream [14] and ODAC [16]. All these methods deal with evolving data streams but, to the best of our knowledge, only POD-Clus supports multivariate data streams. Therefore, POD-Clus is closely related to our work, due to it partitioning strategy and multivariate support.

The **POD-Clus** algorithm (*Probability and Distribution-based Clustering*) [8] seeks to maintain summaries and discard detailed information of the data points, using normal distributions for this purpose. Each POD-Clus' cluster k receives n data points from each incoming data stream and stores some average statistics: such as the average μ, standard deviation σ and the updated covariance matrix, whenever new data arrives. These statistics are used to measure the similarity between data streams considering each attribute or feature independently according to the equation 1.

$$D_{SC} = \sum_{f=1}^{F} \frac{(\mu_{Sf} - \mu_{Cf})^2}{\sigma_{Sf}^2}, \tag{1}$$

where S denotes a data stream, C denotes a cluster, f denotes a data feature (attribute), μ_{Sf} is the mean of data stream S's feature f, σ_{Sf} is the standard deviation of data streams S's feature f, and μ_{Cf} is the mean of the cluster C's feature f.

Although POD-Clus supports multivariate data streams, Equation 1 shows it does not consider the correlation among the data stream attributes to build the clusters. POD-Clus adds a data stream to a cluster only if all its attributes f are similar to the C's features f, regardless of attribute correlations. Thus, as real data usually present correlated attributes, we

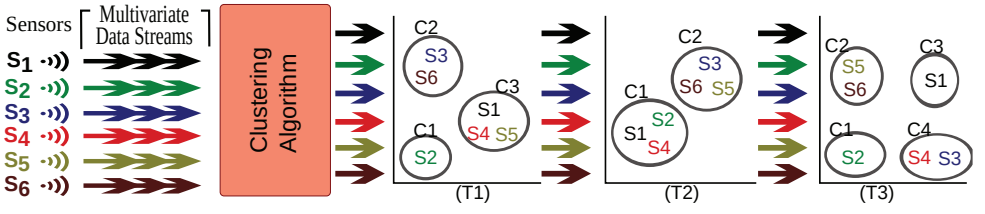


Figure 1: Main idea of clustering multivariate data streams with cluster evolution.

propose a fractal-based approach to improve clustering of entire data streams by considering such correlations and the general behavior of the data streams (see Section 3).

2.2 Fractal Dimension

A fractal is characterized by the self-similarity property, i.e., it is an object that presents roughly the same characteristics when analyzed over a large range of scales. From the Fractal Theory, the Correlation Fractal Dimension D_2 is particularly useful for data analysis, since it can be applied to estimate the intrinsic dimension of real datasets that exhibit fractal behavior, i.e., exactly or statistically self-similar datasets [4]. The Correlation Fractal Dimension D_2 measures the non-uniform behavior of real data considering both linear and nonlinear attribute correlations [9]. Therefore, D_2 represents the dimensionality of the dataset regardless of the dimension of the space defined by its attributes.

Sousa et al. [10] proposed a technique to detect changes in multivariate, evolving data streams based on the information of intrinsic behavior provided by the fractal dimension D_2. The authors also presented the algorithm SID-meter to continuously measure D_2 over time aimed at monitoring the evolving behavior of the data, such that significant variations in successive measures of D_2 can indicate changes in the intrinsic characteristics of the data and in the attribute correlations as well. In this paper, we apply a similar approach to calculate the fractal dimension D_2 of the incoming data streams in order to capture their evolving behavior. We then measure the similarity between data streams based on this behavior aiming for better clusters results.

3 Evolving Fractal-based Clustering Framework

We propose the framework Evolving Fractal-based Clustering of Data Streams, eFCDS for short. A preliminary, naive version of our approach was introduced in [6]. Here, we describe a substantially improved framework, including new solutions to overlapping identification, merge of clusters and outlier detection.

The eFCDS aims to segment a set of data streams $\mathbb{S}$ in a collection $P = \{C_1, \ldots, C_m\}$ of m dissociated clusters. P is built using the fractal dimension analysis of each S_i whilst the related methods employ the raw data points. In other words, the eFCDS groups data streams with similar behavior in an interval of time T_i taking into consideration the correlation among the attributes measured by the fractal dimension D_2. Inside eFCDS, a data stream source S_i will be placed into a cluster C only if the cluster' standard deviation, computed from values of D_2, does not exceed an user-defined threshold σ_{max}. Our method also follows the evolution of the data streams enabling clusters to disappear or be created. After processing every S_i on T_i, the eFCDS detects clusters overlap by identifying data streams sensors that could be member of more than one cluster. If so, the eFCDS tries to merge the clusters or rearrange the sensors in more suitable clusters.

The eFCDS process follows the idea of clustering data streams depicted in Fig. 1. The data stream sources (e.g. sensors) generate new data continuously and send them to eFCDS, which produces evolutionary clusters. Notice that for each interval of time T_i, the gathered data are clustered following the fractal dimension of the available data. As new data arrives, the clusters are created or rearranged to ensure the similarity among their elements along the time. For instance, the cluster disappearance illustrated from period T_1 to T_2 could occur when all members of a cluster are redistributed to others clusters or due to the new merge process explained latter.

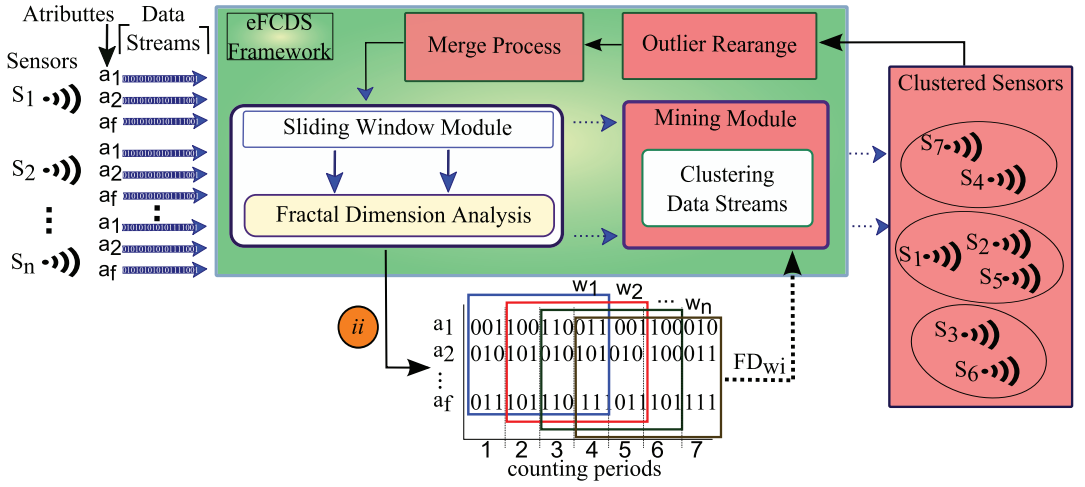


Figure 2: Method of clustering data streams and *(ii)* sliding window of a sensor, counting periods $t = 4$, events $e = 3$.

The main components of our framework work as follows. Data produced by a number of incoming data stream sources are directly forwarded to the eFCDS framework in order to be processed, as shown in Fig. 2.

The first component of the eFCDS is the Sliding Window Module, which receives the data streams and bounds their data in a sliding window fashion. The sliding window (w) specifies the amount of data buffered for the fractal dimensional calculation. The window is divided into counting periods (t), and each period has a defined number of events (e) such that event correspond to a $\vec{d}$ data point. Therefore, $t \times e$ defines the size of the window (l) and e represents its movement step. The size of the window is a user-defined parameter which usually depends on domain expertise. However, it is possible to explore different values of l to evaluate clustering results under distinct temporal granularities. Fig. 2(ii) shows the sliding windows w_i of size l = 12, that means four counting periods t where each t has three events.

As soon as there are enough data fulfilling w, those data can now be processed, i.e., the fractal dimension FD related to w_i is obtained. In such case, the sliding window is shifted forward in e units, releasing memory previously occupied by the oldest e data points in order to buffer the continuous incoming data.

The Fractal Dimension Analyzer is based on the SID-Meter [10] approach to perform the incremental calculus of D_2. Unlike the clustering methods already proposed in the literature, the use of D_2 enables to iterate over data containing more than one attribute, transforming a piece of a multivariate data stream into exactly one value of fractal dimension, summarizing the amount of data inside the window and the correlation among the data stream attributes. Our intuition is that using a technique which capture the correlations among the involved attributes along the time leads to a better recognition of similar data stream sources than using the raw values of the attributes.

Subsequently, the reduced amount of fractal dimension points is sent to the data mining step. The Mining Module is the main component of the eFCDS framework and aims at clustering the fractal dimension points according to their similarity and a user-defined maximum standard deviation (σ_{MAX}), that could be a percentage of the number of attributes. In order to detail the clustering step, let us introduce the Algorithm 1.

Algorithm 1: Evolving Fractal-based Clustering of Data Streams (eFCDS)

Input: The set X containing pairs of streams S_i and its fractal dimension fd;
The partition P of clusters;
The maximum standard deviation σ_{MAX}.
Output: The partition P of clusters with similar data streams.

```
1  foreach x ∈ X do
2      if x.S_i is in a cluster C_j ∈ P then
3          if updateCluster(C_j, x, σ_MAX) = false then
4              if findCluster(x, σ_MAX) = false then
5                  C_new ← createNewCluster (x);
6                  rearrangeCluster(C_j, C_new);
7      else
8          if findCluster(x, σ_MAX) = false then
9              C_new ← createNewCluster (x);
10 rearrangeOutlier (C, σ_MAX);
11 mergeCluster (C, σ_MAX);
12 return P;
```

The Algorithm 1 iterates over the set X comprising the points of fractal correlation (fd) such that each point is labeled with its respective data stream S_i. Initially, it is verified whether or not the data stream source S_i already belongs to some existing cluster C_j composing the partition P (line 2). Supposing S_i was not clustered yet, it is necessary to find for a cluster to insert S_i into. For this purpose, for each cluster $C \in P$, the boolean procedure `findCluster` (line 8) looks for the cluster C_j with the lowest difference between the centroid of C_j and the analyzed point x. This process follows the model introduced in Equation 2, where $C = \{c_1, \ldots, c_k, \ldots, c_n\}$, the quantity of clusters generated so far.

$$\Delta(C, x) = \left(\frac{1}{n} \sum_{k=1}^{n} c_k.fd \right) - x.fd \tag{2}$$

Once the cluster C which minimizes the Equation 2 regarding to the element x is obtained, x is assigned to C if the condition expressed in Equation 3 holds. Otherwise, the new cluster C_{new} containing the element x is created (line 9) and included in the partition P.

$$\sqrt{\frac{1}{n+1}\left(\left(x.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2 + \sum_{k=1}^{n}\left(c_k.fd - \left(\frac{1}{n}\sum_{p=1}^{n} c_p.fd\right)\right)^2\right)} \leq \sigma_{MAX} \tag{3}$$

Considering the data stream source S_i is already part of some cluster, it is necessary to check whether or not S_i remains in the previous assigned cluster. In order to employ such verification (line 3), the Mining module re-execute the calculus of Equation 3 regarding to the new value of the fractal dimension $x.fd$. If so, the statistic (function Δ) of the considered cluster C_j is updated. In the case where the left-hand side of Equation 3 exceeds the user-defined maximum standard deviation σ_{MAX}, the algorithm tries to reallocate the element x in an existing cluster

C_k so as to minimize the value computed in Equation 2 (line 4) and also hold the condition defined in Equation 3. If there is not such cluster C_k satisfying both conditions, a new one (C_{new}) is created to include the element x (line 5). Now, when creating a new cluster, the elements already present in C_j are checked if they are better included in C_{new}, minimizing Equation 2 (line 6). Notice that the `rearrangeCluster` procedure is able to suppress existing (empty) clusters if all of their elements are moved.

After that, the eFCDS iterates over the clusters to identify those with single data streams, i.e., potential outliers. If anyone is found the eFCDS tries to rearrange it on one of the non outlier clusters following the aforementioned criteria. The last step is to check if there is overlapping between clusters (line 11) and merge them according to the *Merge Cluster* algorithm (Alg. 2). It checks whether or not there are some data streams that could be associated to more than one cluster (Alg. 2:line 1). If so, then the respective clusters will be merged if their union does not exceed the maximum standard deviation (Alg. 2:line 3). Otherwise, it is found the bounds of both clusters i and j (Alg. 2:line 4) by getting the S_i that is closest to the C_j centroid and the S_j that is closest to C_i centroid. Both S_i and S_j will be attached to a new cluster ij (Alg. 2:line 5) and all the remain data streams in C_i and C_j will be checked if they stay in their own clusters or if they will be associated to the new cluster C_{ij} (Alg. 2:line 6).

Thus, at the end of one iteration over the set X, the partition P is composed of clusters containing the similar data streams sources in relation to the fractal dimension (line 12). As the sources are continuously generating measures, the sliding window shifts forward and, as soon as there are enough data to repeat the process, the Mining module is re-invoked. Therefore, the obtained clusters evolves to reflect the changes in the gathered data along the time.

Algorithm 2: Merge Cluster

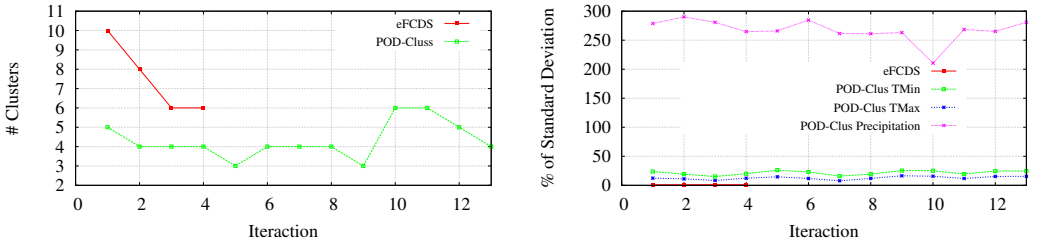
Input: The partition P of clusters;
The maximum standard deviation σ_{MAX}.
Output: The partition P of clusters with similar data streams.

```
1 ov[] ← findOverlap(P);
2 foreach pair(i, j) ∈ ov do
3     if merge(i, j) ≤ σ_MAX then
4         findMargin(i, j);
5         C_new ← createNewCluster (ij);
6         checkReAlloc(C_i, C_j, C_ij) ;
7 return P;
```

4 Experimental Evaluation

In order to evaluate our framework, we employed the following methodology: 1) examining the performance of our approach using a real dataset; 2) using Silhouette measure to assess the clusters' quality; 3) comparing eFCDS' results with the competing method (POD-Clus) previously described in Section 2.1. The goal of this methodology was to analyze three fundamental points: (i) the eFCDS' ability to capture similar data streams behavior along the time, (ii) the clusters' partition cohesion obtained, (iii) the clusters' quality.

The real dataset is composed of 145 data streams generated by the same number of climate sensors, each one collecting 3 distinct daily measures (minimum‖maximum temperature and



(a) Quantity of clusters generated (b) Standard deviation percentage

Figure 3: Comparative main results

precipitation) from January 1990 to December 1994 in different regions of Brazil[2]. The measures measures of this dataset belong, mainly, to the South and Southeast Brazilian regions with different climatic conditions, making it suitable for this type of evaluation. As the country has continental proportions, it is important to automate analysis as proposed by the eFCDS algorithm.

To test our method with the climate sensors dataset, many values of parameter were empirically tested and the best ones were chosen. The sliding window was set with 4 counting periods (t) of 365 events (e) which represent 4 year of data, the maximum standard deviation σ_{MAX} was set to 0.05 that is, a small percentage of 3 (the maximum fractal dimension that could be obtained when the attributes are not correlated with each other). In fact, σ_{MAX} greater than 10% of the maximum fractal dimension generally leads to single cluster.

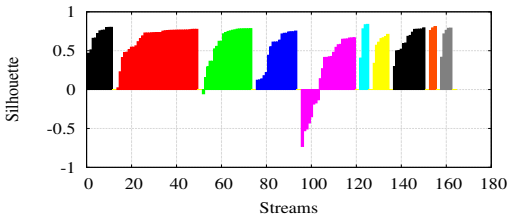
In order to measure clusters' quality, we applied the well-known Silhouette, which is commonly employed to evaluate how well the elements are disposed in their respective clusters. The values of Silhouette range from -1 to 1, such that clusters with measures above 0.5 are considered of good quality.

Ultimately, the eFCDS results were compared against the related method POD-Clus (Section 2.1). POD-Clus was chosen because it aims to build clusters with compact representation, fast data processing, traceability of changes and maintenance of clusters' evolution. As already stated in Section 2.1, POD-Clus supports multivariate data streams, but it does not consider the attributes' correlation. The parameters of POD-Clus were also empirically determined: the sliding windows size equal to 365 with shift of 90 days; three streams are needed to turn an outlier group into a cluster; the cluster merge threshold was 0.3; and all the others parameters were set with default values.

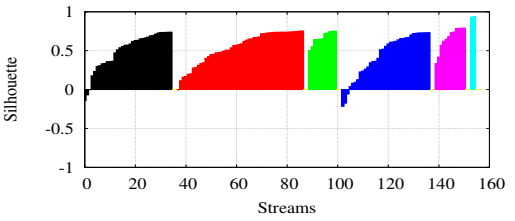
The main results are depicted in Fig. 3. Fig. 3(a) presents the amount of clusters generated by each method in every iteration and the Fig. 3(b) shows the average percentage of the standard deviations for POD-Clus' features and eFCDS fractal dimension. Recall that eFCDS deals with fractal dimension values and the POD-Clus deals with the original measures of temperature (min & max) and precipitation, then to determine the cohesion of the obtained clusters we calculated the percent of standard deviation over the mean of the clusters on each iteration.

Thus, it is possible to notice that the percentage of standard deviation obtained by eFCDS was less than 1% indicating a very high clusters cohesion. On the other hand, the cluster cohesion obtained by POD-Clus ranges from 7% to 25% on minimum and maximum temperature, respectively. Also notice that standard deviation of precipitation was 3 times higher than the mean of the data points. It is due to the particular characteristics of the rainfall in Brazil

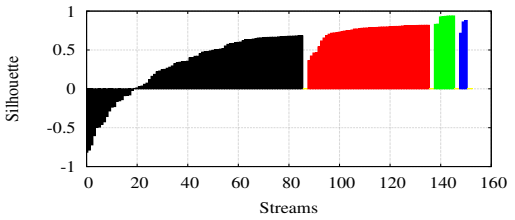
[2]Data provided by Agritempo (www.agritempo.gov.br)



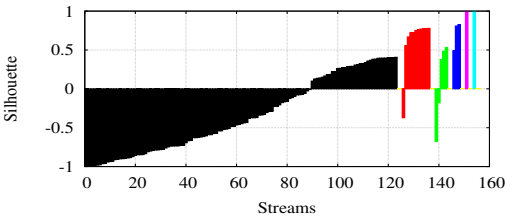
(a) eFCDS' Silhouette at first iteration



(b) eFCDS' Silhouette at last iteration



(c) POD-Clus' Silhouette at first iteration



(d) POD-Clus' Silhouette at last iteration

Figure 4: Comparative Silhouette results

Southeast, where the rain vary from 1200 to 1500 millimeters a year. As POD-Clus considers the attributes individually (see Equation 1), the higher the standard deviation of the attribute the smaller its influence on the distance measure, affecting the quality of the final clusters.

Fig. 4(a) and (b) show the Silhouette results obtained by the eFCDS in the first and last iteration, respectively. The eFCDS built clusters with 67,3% of the data streams Silhouette measures upper to 0.5, in the average. As the figures reveal, the great majority of the data streams were well clustered using eFCDS.

Fig. 4(c) and (d) show the Silhouette values to the clusters generated using the POD-Clus . Differently from eFCDS, the POD-Clus obtained the average of 25.8% considering all iterations. The best result obtained by POD-Clus appears in the first iteration (Fig 4(c)), because as the process goes on, the quality of the clusters decreases with only 10% of the data streams Silhouette upper to 0.5 on the last iteration (Fig 4(d)).

In our experimental evaluation, the eFCDS framework better represented the behavior of different data streams along the time. And, as showed, the POD-Clus distance measure does not deal properly with the precipitation feature, because the rainfall season, in the South and Southeast of Brazil, occurs from October to March and the dry season from July to September, that means a great precipitation variation that POD-Clus was unable to deal with it. Our approach based on fractals achieved promising results to deal with multivariate data streams, helping to capture the behavior of climate streams over time and obtaining clusters with good quality.

5 Conclusion and Future Work

Clustering of data streams is one of the most employed approach to analyze evolving data streams. Nevertheless, the literature provides few methods for clustering the entire data streams and most of them deal with data streams composed of a single attribute or do not apply

appropriate strategies for multivariate flows.

In this paper we described the eFCDS framework to cluster a set of multivariate data streams considering the correlations among their attributes and complying the basic requirements to cluster data streams, namely: to read data only once, to provide answers as soon as the user demands them, to deal with multivariate data streams, to take into account the correlation among the attributes, to capture and represent data evolution along the time, to deal with outlier data streams. It also takes into account the behavior of distinct streams along the time. The proposed framework is composed of minor modules, each one responsible for a specific processing of the data. The core of the eFCDS framework lies in the computation of the fractal dimension of a piece of the data stream and its subsequent clustering. Also, our framework performs the clustering of multivariate data streams supporting their evolution in an incremental way helping to improve the clusters quality. To the best of our knowledge, there is no other clustering method that support multivariate, evolving data streams and clusters them considering attribute correlation.

We performed a set of experimental evaluation of the eFCDS framework and compared it against POD-Clus, in order to verify the capacity of representing similar data streams behavior along the time and keeping the quality of the produced clusters. As the results showed, the use of the fractal dimension allowed to better identify the correlation among the attributes of the data streams and consequently lead better clusters than the POD-Clus did. The eFCDS performed 2.6 times better than the POD-Clus in relation to the cohesion of the clusters, measured by the Silhouette. The eFCDS not only follows the evolution of the data streams identifying new clusters and their disappearance, but also identifies outlier data streams and merge clusters based on a maximum standard deviation.

Our framework also proves to be an useful tool to assist meteorologists in understanding the climate behavior along a period of time without the necessity to analyze the entire dataset manually. They were able to identify the transposition of a sensor from one cluster to another and thus detect those with unpredictable behavior. Then the meteorologists can check if those sensors were affected by extreme weather events, in the analyzed period, that could have changed their expected behavior, such as severe droughts or heat waves.

References

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. On clustering massive data streams: Summarization paradigm. In Charu C. Aggarwal, editor, *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*, chapter 2, pages 169–208. Springer, Yorktown Heights, NY, USA, 2007.

[2] Argenis A Aroche-Villarruel, José Fco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa, and Airel Pérez-Suárez. A different approach for pruning micro-clusters in data stream clustering. In *Pattern Recognition*, pages 33–43. Springer, 2015.

[3] Daniel Barbará. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2):23–27, January 2002.

[4] Alberto Belussi and Christos Faloutsos. Estimating the selectivity of spatial queries using the correlation fractal dimension. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 299–310, 1995.

[5] A. Bifet and G. De Francisci Morales. Big data stream learning with samoa. In *2014 IEEE Int. Conf on Data Mining Workshop*, pages 1199–1202, Dec 2014.

[6] Christian C. Bones, Luciana A. S. Romani, and Elaine P. M. de Sousa. Clustering multivariate climate data streams using fractal dimension. In Vanessa Braganholo, editor, *The 30th Brazilian Symposium on Databases*, volume 30, pages 41–52, Petropolis, RJ, Brazil, October 2015. SBC.

[7] Rattanapong Chairukwattana, Thanapat Kangkachit, Thanawin Rakthanmanon, and Kitsana Waiyamai. Se-stream: Dimension projection for evolution-based clustering of high dimensional data streams. In Van Nam Huynh, Thierry Denoeux, Dang Hung Tran, Anh Cuong Le, and Son Bao Pham, editors, *Knowledge and Systems Engineering*, volume 245 of *Advances in Intelligent Systems and Computing*, pages 365–376. Springer, 2014.

[8] Pimwadee Chaovalit and Aryya Gangopadhyay. A method for clustering transient data streams. In *Proc. ACM SAC '09*, pages 1518–1519, New York, NY, USA, 2009. ACM.

[9] Elaine P.M. de Sousa, Caetano Traina Jr, Agma J.M. Traina, Leejay Wu, and Christos Faloutsos. A fast and effective method to find correlations among attributes in databases. *Data Mining and Knowledge Discovery*, 14(3):367–407, 2007.

[10] Elaine P.M. de Sousa, Agma J.M. Traina, Caetano Traina Jr, and Christos Faloutsos. Measuring evolving data streams behavior through their intrinsic dimension. *New Generation Computing*, 25(1):33–60, 2007.

[11] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528, 2003.

[12] Edwin Lughofer and Moamar Sayed-Mouchaweh. Autonomous data stream clustering implementing split-and-merge concepts–towards a plug-and-play approach. *Information Sciences*, 304:54–79, 2015.

[13] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73, 2014.

[14] Cássio M. M. Pereira and Rodrigo F. de Mello. Ts-stream: clustering time series on data streams. *Journal of Intelligent Information Systems*, 42(3):531–566, 2014.

[15] Muhammad Z. Rehman, Tianrui Li, Yan Yang, and Hongjun Wang. Hyper-ellipsoidal clustering technique for evolving data stream. *Knowledge-Based Systems*, 70(0):3–14, 2014.

[16] Pedro P. Rodrigues, Joao Gama, and Joao P. Pedroso. Hierarchical clustering of time-series data streams. *IEEE TKDE*, 20(5):615–627, 2008.

[17] Harya Widiputra, Russel Pears, and Nikola Kasabov. Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. In Joshua Huang, Longbing Cao, and Jaideep Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, pages 161–172. Springer, Shenzhen, China, 2011.

[18] Xiangliang Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag. Data stream clustering with affinity propagation. *IEEE TKDE*, 26(7):1644–1656, July 2014.