Information Sciences 558 (2021) 174–193

Contents lists available at ScienceDirect

Information Sciences

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

Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach

Laura Melgar-García [a], David Gutiérrez-Avilés [a], Cristina Rubio-Escudero [b], Alicia Troncoso [a,*]

[a] Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain
[b] Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, Seville 41012, Spain

ARTICLE INFO

Article history:
Received 22 July 2020
Received in revised form 13 November 2020
Accepted 30 December 2020
Available online 26 January 2021

Keywords:
Data streaming
Patterns
Real-time
Triclustering

ABSTRACT

Triclustering algorithms group sets of coordinates of 3-dimensional datasets. In this paper, a new triclustering approach for data streams is introduced. It follows a streaming scheme of learning in two steps: offline and online phases. First, the offline phase provides a summary model with the components of the triclusters. Then, the second stage is the online phase to deal with data in streaming. This online phase consists in using the summary model obtained in the offline stage to update the triclusters as fast as possible with genetic operators. Results using three types of synthetic datasets and a real-world environmental sensor dataset are reported. The performance of the proposed triclustering streaming algorithm is compared to a batch triclustering algorithm, showing an accurate performance both in terms of quality and running times.

1. Introduction

Currently, research efforts are focused on developing new methodologies that consider continuous and massive flows of data from a wide variety of sources with the objective of analyzing and taking advantage of them. Therefore, the term Data Streaming that defines these in motion data takes relevance in this context. Nowadays, there are a vast variety of data streaming sources such as Internet of Things (IoT), social media, medical devices, or videos. When dealing with continuous flows of data and real-time characteristics, traditional machine learning methodologies (i.e., clustering, regression, classification) have to be adapted to this new environment [1].

There are two main computational approaches to develop machine learning models for data streaming [1]. On the one hand, incremental learning is found, where the model evolves and adapts incrementally to concept drift [2] in data streams. On the other hand, the offline-online learning emerges, where the model is divided into an offline phase in order to create a summary (or sketch) of the data without time execution restrictions and an online phase to update the synopsis data in real-time as fast as possible.

Among all the streaming analysis tasks, the behavior pattern extraction becomes relevance as it is the base of a vast number of current real-time applications such as customer analysis, fraud detection, or sentimental analysis [3]. Different types of algorithms are found in the literature depending on the type of patterns to be searched. Clustering [4] extracts similar behaviour patterns over all the features analyzed. Biclustering [5] appears as an evolution of clustering since patterns can

* Corresponding author.
E-mail address: atrolor@upo.es (A. Troncoso).

https://doi.org/10.1016/j.ins.2020.12.089
0020-0255/

be extracted from a subset of instances and features. Triclustering [6] allows for considering a third dimension in the dataset, usually time, finding groups of elements with similar behaviour throughout three-dimensional datasets.

In this work, we present a new online triclustering algorithm for data streaming, called STriGen. The STriGen is a learning method based on genetic algorithms, which combines an offline and online phase, to discover collections of resembling patterns in 3D data streams in real time. We show the results obtained from the application of STriGen to three synthetic datasets of different complexity and to a real dataset collected from seven environmental sensors. The results obtained for the synthetic datasets are validated by means of comparing the triclusters found to the real ones, providing quality evaluation measures such as the accuracy and F1 score. The triclusters obtained by the STriGen using the real dataset can not be validated as the synthetic ones since the real triclusters are not known, and therefore we use the Graphical Quality (GRQ) measure described in [7].

The rest of the paper is structured as follows: a summary of the previous research related to the paper's topic is presented in Section 2; Section 3 describes how the offline and online part of the proposed algorithm works; the experimental setting carried out and the results obtained are shown in Section 4; and, finally, the conclusions and future works are provided in Section 5.

2. Related works

In this Section, the current state of the art related to the proposed research is presented. The STriGen algorithm is associated to two main areas: the streaming analysis and the triclustering approach.

Regarding the streaming analysis topic, several proposals of new machine learning approaches adapted to this emerging environment can be found in the literature. In [8], an online version of the support vector machine model was developed to predict air pollutant levels using streaming time series. The authors in [9] presented a streaming version of the linear discriminant analysis algorithm for dimension reduction. In [10], the authors developed a streaming analytics system for large-scale data. An adaptive ensemble learning for online activity recognition from data streams was presented in [11]. Furthermore, efforts to develop tools to facilitate the adaption of classic machine learning models to the streaming environment have been carried out. In this sense, a general adaptive incremental learning framework for streaming data analysis was performed in [12] and, an API to apply machine learning algorithms to data streams, well-known as SAMOA, was introduced in [13].

In streaming environments, both the prompt anomaly detection and the generation of a continuous flow of data are highly explored fields. The authors in [14] presented a real-time methodology to detect cybercrimes related to credit card frauds; another anomaly detection algorithm for streams of data was developed in [15]. In an overview regarding the pattern discovery task in streaming, the authors in [16] proposed a methodology to discover patterns in multiple time-series in streaming. A classification of streaming features based on an emerging-pattern approach was presented in [17]. The authors presented SPADE in [18], which is a shape-based pattern detection method for streaming time-series.

Also, new online machine learning approaches have emerged in the area of evolving systems in clustering due to the rise of the streaming analysis field. In [19], the authors developed a fuzzy-rule-based model with the capability to adapt to the new streams of data and to deal with missing values efficiently. The authors in [20] presented an evolving optimal granular system to perform the approximation of functions such as time-series models, classification or regression functions, demonstrating its outperform when comparing with other evolving methods in multivariate problems. The authors in [21] presented a new approach for online regression and system identification problems in data streams, based on evolving fuzzy systems. An updating of the previous methodology was developed in [22], where the authors presented a new rule splitting framework for generalized evolving fuzzy systems, demonstrating the outperform of this new algorithm against the original version. In [23], the author presented a new online incremental clustering algorithm based on a dynamic cluster merging method, which consisted in the calculation of the covariance matrix of the clusters susceptible to be joined. Furthermore, in [20], the authors developed other evolving systems specializing in evolving fuzzy rule-based models and neuro-fuzzy networks in online and real-time environments. A complete survey of evolving systems dealing with streaming data can be found in [24].

In the streaming analysis field, an important research area is the adaptation of existing methodologies to the streaming environment. Thus, some algorithms are adapted to the streaming environment using incremental learning. For example, K-Means methods for data streaming modify just the calculation process of the closest mean to the new point instead of applying the whole algorithm every time a new point arrives [25]. We can also find a mixed online/offline strategy, which is advantageous as streaming concerns just the online phase. In the offline phase, traditional algorithms are applied to the data without meeting all the requirements of data streaming. During the online phase, selected streaming data are structured, keeping only a statistic summary of them and not the full data. StreamKM++ [26] and CluStream [27] are some examples of this online/offline methodology.

Finally, the triclustering topic emerges as a methodology for gene expression data time series. The goal in triclustering is the same as in clustering, that is, minimizing the intra-triclustering distance and maximizing the inter-triclustering distance. There are different triclustering algorithms approaches depending on their behaviour: iterative searches, distribution param-

eter identification, pattern mining, evolutionary multi-objective optimization, among other options [28]. The authors introduced an algorithm based on the symmetry property of the triclusters in [29]. Lately, an extended and generalized version of the previous proposal was presented in [30]. An evolutionary computation approach was proposed in [31], where the fitness function was defined as a multi-objective measure. Another proposal based on genetic algorithms was developed in [32], where the authors mined the optimal shifting and scaling patterns from 3D-microarrays. The triclustering genetic-based algorithm, TriGen, also used multi-objective optimization approach [6]. The authors in [33] developed a triclustering algorithm based on a statistical methodology for 3D short gene expression data time series datasets. The triclustering techniques have been also applied to other areas. In particular, a triclustering algorithm was used to depict seismogenic zoning over the Iberian Peninsula in [34]. The authors applied triclustering techniques in high-content screening images to identify its components in [35]. Moreover, triclustering techniques have been successfully applied to medical environments, like in [36], where triclustering was applied in combination with random forest to predict the need for non-invasive ventilation in ALS patients. A survey of several existing triclustering algorithms can be found in [37].

3. Materials and methods

This Section describes the proposed algorithm STriGen. Section 3.1 defines and specifies the main characteristics of the triclusters. The STriGen algorithm is presented in Section 3.2, including the type of data that it needs, the description of its two phases and the validation of the resulting triclusters.

3.1. Tricluster definition

Triclustering algorithms are an evolution of clustering algorithms [38]. Specifically, clustering is applied on 2-dimensional datasets and triclustering on 3-dimensional datasets. In both cases, data are a set of instances (rows in a matrix) and features (columns in a matrix) and, just in triclustering, data also contain a set of time points (depths in a matrix).

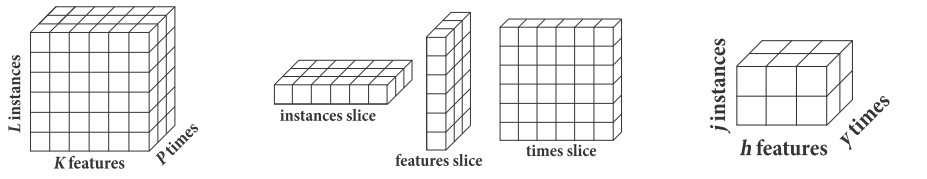
In particular, a cluster is defined as a group of instances over all features. However, a tricluster is defined as a group of instances over a group of features and over a group of time points. For example, in a 3-dimensional dataset with L instances, K features and P time points, a resulting tricluster is a subset composed of $j \leqslant L$ instances, $h \leqslant K$ features and $y \leqslant P$ time points, respectively. An example tricluster formed by $j \times h \times y$ components is represented in Fig. 1c. Fig. 1a represents the 3-dimensional data and its 3 slices are presented in Fig. 1b.

3.2. The STriGen algorithm

STriGen is a triclustering algorithm based on a genetic evolutionary heuristic that finds groups of similar behaviour patterns in 3-dimensional streaming datasets.

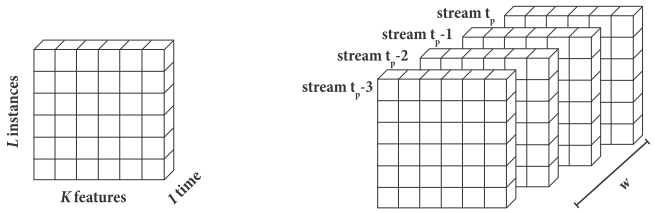
Data streams are continuous flows of data supplying new information. As data can possibly have an infinite volume, there are some constraints that every streaming algorithm needs to satisfy [39]: single-pass and chronological order, i.e., streams are processed one by one, only once and in the order of arrival. In addition, stream models have to incorporate new information updating themselves dynamically and they have to detect and eliminate outdated data effects (also called concept drift detection). Stream models also have to deal with very important constraints as bounded memory and bounded response time. Therefore, only a summary of specific data can be stored and outputs must be provided as fast as possible and the execution time of the learning model must increase linearly according to the number of instances. The STriGen algorithm deals with data streams and satisfies all mentioned constraints.

Moreover, as streams can be infinite, there are different options to define which time window of data has to be considered [1]. The proposed algorithm uses the w sliding window where the model takes into consideration only the most recent instances. In particular, w is an integer number from 2 to the current number of streams. Thus, 2-dimensional data arrive with all L instances and all K features, as defined in Section 3.1, so one stream is formed by $L \times K$ samples, as shown in



(a) 3D data (b) Data slices (c) $j \times h \times y$ tricluster

Fig. 1. Representation of triclusters.



(a) Stream sample (b) Time window

Fig. 2. Representation of streams.

Fig. 2a. Fig. 2b represents a particular case where w is 3 and so, by definition of sliding window, only the 3 most recent streaming samples are considered.

There are two main computational approaches to model data streams [1]: the incremental learning and the offline-online learning. In the first one, incremental learning, the model evolves and adapts incrementally to concept drift in data streams. However, in the offline-online learning, the modeling is divided in an offline phase that creates a summary (or sketch) of the data without execution time restrictions and an online phase to update the synopsis data in real time as fast as possible. The proposed algorithm uses the offline-online learning approach and uses Apache Kafka as streaming platform.Fig. 3 is an overview of the workflow of the proposed algorithm. It starts with the offline phase that creates a summary model containing the components of the triclusters found by the algorithm as explained in Section 3.2.1. Then, the Kafka producer publishes the streaming data in a Kafka topic. A Kafka topic is a container that stores the data streams. The Kafka consumer receives the data in streaming from the Kafka topic and executes the online phase of the algorithm. In this phase online, the initial triclusters resulting from the offline step evolve over time as it receives new data streams to keep the model always updated as explained in Section 3.2.2.

These two phases of the algorithm are described in detail below. The offline phase is described in Section 3.2.1 and the online in the Section 3.2.2.

3.2.1. The offline phase

The offline phase of streaming algorithms processes data in a static or batch mode, i.e., not considering the requirements of the streaming algorithms above-mentioned. However, the output of this phase has to be a summary or sketch of the data that is the base of the online phase.

In particular, the offline phase of the proposed algorithm creates a summary of the components of each tricluster found. STriGen applies evolutionary meta-heuristics of genetic algorithms to find triclusters. The process of natural selection is represented by the selection of the fittest individuals for reproduction in order to produce the offspring of the next generation. Therefore, there are two main actors in the algorithm: population and individuals. One individual is a potential solution of the algorithm, i.e., a tricluster containing instances, features and streams, and a population is composed of several individuals.

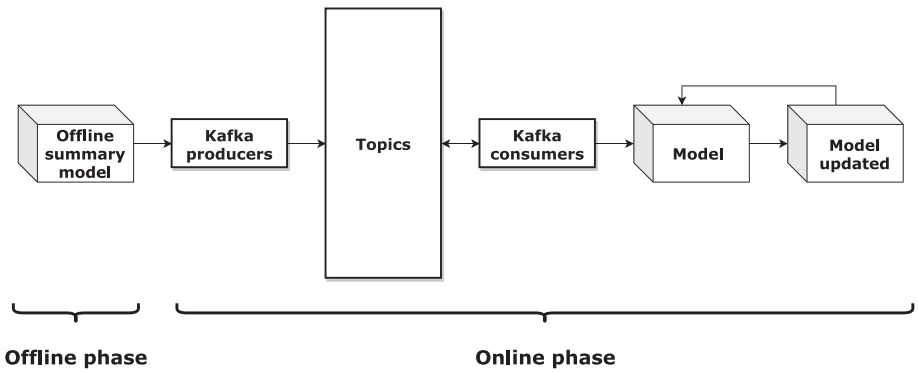


Fig. 3. Offline and online phases of the STriGen.

The algorithm starts creating an initial population with a random selection of instances, features and streams for each individual. Afterwards, four genetic operators are used to make the population of individuals evolves optimizing a fitness function. Once a solution tricluster is found, this process is repeated for all the rest of the desired triclusters but with a different way of creating the initial population. In other words, considering N as the number of triclusters to find, for the rest N-1 triclusters, a percentage of individuals are generated randomly and the others are generated considering the non-explored areas of the previous solution triclusters in order to avoid overlapping solutions.

Inspired by [6], the genetic operators, together with the initialization of the population, are crossover to make connections between individuals and share their components, mutation to make specific changes to individuals and explore new possible components, selection to choose which individual stays in the next generation and evaluation to measure the quality of the population. These operations are made considering some triclustering configuration parameters: the number of solution triclusters, the number of generations, the members of a population, the aleatory factor for the initial population, the selection rate and the mutation probability rate. The crossover is implemented by means of a random one-point crossover [40]. Mutation is made by inserting, deleting or changing instances or features of the current tricluster. The tournament selection mechanism is used to select the best individuals and the evaluation is implemented by means of a fitness function.

The fitness function employed by the proposed algorithm is the Multiple Square Lines (MSL), which is based on the similarity among the angles of the slopes formed by the components of the tricluster at each time point. MSL is completely described in [7]. In addition two terms are added to control the size of the triclusters regarding the number of instances, features and times and to control the overlapping of the triclusters, respectively.

Finally, the measure used to evaluate the quality of a solution tricluster *TRI* is the Graphical Quality (GRQ) measure:

$$GRQ(TRI) = 1 - \frac{MSL(TRI)}{2\pi} \tag{1}$$

A tricluster will have a higher graphical quality as smaller the MSL value is, which is minimized by means of the fitness function.

Finally, the components of the triclusters found in the offline phase are the individuals with the best fitness function value, and the final summary of the offline phase is the initial base of the online phase. A scheme of the main steps of the offline phase is shown in Algorithm 1.

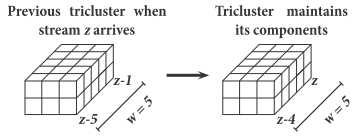
Algorithm 1. Offline phase.

```
Result: summary model with triclusters components
repeat
    initialize population;
    evaluate population;
    repeat
        select individuals;
        crossover individuals;
        mutation individuals;
        evaluate new population;
    until number of generations;
    select the individual that minimizes the fitness function;
    tricluster is made up of the components of the selected individual;
until number of triclusters;
```

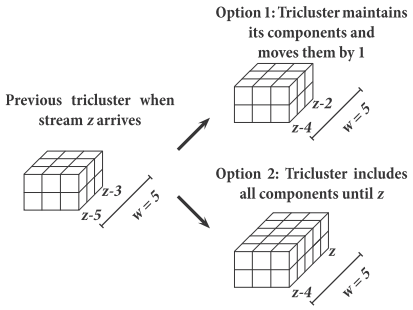
3.2.2. The online phase

Once the summary of the offline phase is ready, the online phase starts. In this phase, the algorithm considers all streaming requirements.

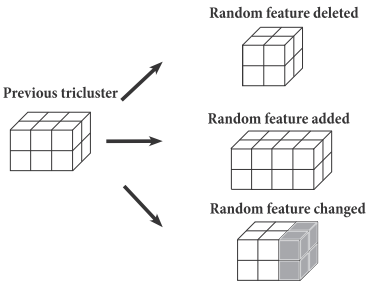
Firstly, as the algorithm uses a sliding window, just the most recent w samples are considered. In particular, if the new stream arrives at the instant point z, all triclusters components that include samples up to the $z - w - 1$ instant point are



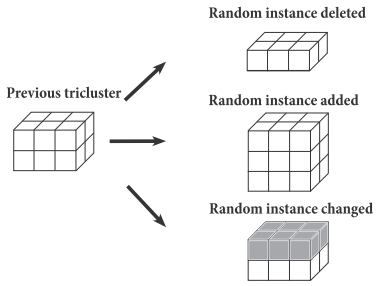
(a) Processing of new stream when the last time point in the previous tricluster is $z-1$.



(b) Processing of new stream when the last time point in the previous tricluster is not $z-1$.



(c) Feature mutation process.



(d) Instance mutation process.

Fig. 4. Examples of the updating operations during the online phase.

deleted, i.e., they are not considered anymore. This is a common procedure in the streaming environment [41]. Thereby, the sliding window model removes the oldest samples every time a new stream sample arrives and so, triclusters contain the most recent data at every moment (see Fig. 2b).

Afterwards, the model has to evolve to include knowledge from the new stream sample. In traditional clustering techniques applied to streaming, as for example K-Means, a common strategy is to add the new instance to its nearest "centroid" if it is within the boundary of the cluster [42]. The STriGen algorithm computes different updating options and selects the one with the highest GRQ value (Eq. 1), that corresponds to the best graphical quality of each tricluster. All updating options try to find the most suitable tricluster's components from the w streams.

One updating option is to maintain the same components of the existing tricluster from the instant point $z-1$ to $z-w$. In other words, including as components at instant or stream z the same instances and features of the previous tricluster. Fig. 4a shows a graphical example of this updating. In the particular case of triclusters that do not include the instant $z-1$ in their components, two updating options are carried out: either adding just the following instant point of the previous tricluster or all of the instant points until z.Fig. 4b is a graphical description of this updating, where the maximum number of instant points that can be included in a tricluster is five, as $w=5$. Thus, the tricluster contains just three instant points ($z-4$ to $z-2$) for the option 1 and the tricluster contains all five possible instant points ($z-4$ to z) for the option 2.

In addition, as the offline phase is a NP hard problem, it is probable that even with the application of these updating heuristics, the solution triclusters obtained might be a local rather than global optimal. To manage this issue and to allow triclusters to evolve and adapt to new data streams, the mutation operator is included as another updating possibility. This genetic operator deletes, changes or/and adds randomly samples (instances or/and features) in the triclusters resulting from the above-mentioned updates. Firstly, the type of mutation is randomly selected: deleting an existing coordinate, changing an existing coordinate to a new one or adding a new coordinate. Then, the specific instance or feature is also randomly selected. If the deleting option is randomly selected and a specific feature of the tricluster is selected, this feature is removed for all the instances and time points of the existing tricluster. In the case of the adding option, if a feature is randomly

selected from the data that are not included in the tricluster, this feature is added for all the instances and time points of the current tricluster and so, all its expression values. For the changing option, the deleting option is firstly executed and then the adding one. The instance mutations follow the same procedure of the feature mutations but considering the instances. Fig. 4c and Fig. 4d are graphical examples of mutation of features and instances, respectively. Once mutation is carried out, the GRQ is computed for all updated triclusters and the tricluster that maximizes the GRQ is selected. This updating process is represented in the pseudo code of the Algorithm 2.

Algorithm 2. Updating of the triclusters.

```
Result: Updated tricluster
z ← current time;
z-t ← last time of the tricluster TRI;
if z − t != z-1 then
    newTRI_1 ← add(z − t + 1,TRI);
    newTRI_2 ← mutate(newTRI_1);
    newTRI_3 ← add([z − t + 1,...,z],TRI);
    newTRI_4 ← mutate(newTRI_3);
    [bestTRI,bestGRQ] ←
      evaluate(newTRI_1,newTRI_2,newTRI_3,newTRI_4);
else
    newTRI_1 ← add(z,TRI);
    newTRI_2 ← mutate(newTRI_1);
    [bestTRI,bestGRQ] ← evaluate(newTRI_1,newTRI_2);
end
return bestTRI and bestGRQ
```

Another important aspect when dealing with stream computing is to consider the concept drift problem. Concept drift occurs when an existing concept changes or a new concept appears [43]. There are different types of concept drift: incremental, gradual, abrupt or reoccurring. When any change starts to occur in expression values of the triclusters components, the GRQ decreases abruptly. In such cases, only one updating is not enough. The parameter *minGRQ* let the algorithm entering in a loop of a maximum of *numIt* iterations or until the GRQ is higher than *minGRQ*. Each iteration corresponds to a mutation updating process. In this way, STriGen can adjust rapidly to small changes but also to abrupt changes detecting new components of the triclusters or making the current ones disappear. This additional updating to deal with changes is represented in the pseudo code of the Algorithm 3.

Algorithm 3. Additional updating.

```
iter ← 0;
while currentGRQ < minGRQ and iter < numIt do
    TRI ← mutate(TRI);
    currentGRQ ← evaluate(TRI);
    iter ← iter+1;
end
return TRI
```

In summary, an overview of the online phase is depicted in the pseudo code of the Algorithm 4.

Algorithm 4. Online phase.

```
Result: Set of updated triclusters S
S ← {};
while new data stream at time z do
    for each TRI in current model do
        TRI ← remove(z − w − 1,TRI);
        [newTRI, currentGRQ] = updating(TRI) //algorithm 2;
        if currentGRQ<minGRQ then
            updatedTRI ← additionalUpdating(newTRI) //algorithm 3;
        else
            updatedTRI ← newTRI;
        end
        S ← S ⋃ updatedTRI ;
    end
end
return S
```

3.2.3. Validation of triclusters

In the case of using synthetic data as experimental data, the triclusters' coordinates that STriGen should find are previously known. To evaluate the performance of the STriGen in this situation two metrics, F1 score and accuracy, are computed. Accuracy is the most used measure in the classification field. It represents the ratio of correctly predicted observation with respect to the total number of observations, and it is appropriate if the dataset is symmetric, that is, when values of false positive and false negatives are almost the same. In our case, the coordinates not being part of a tricluster solution are far more abundant than the ones belonging to a tricluster. Therefore, we also include the F1 score, which is a weighted average of the precision and recall and considers both false positives and false negatives.

In the case of using real data as experimental data, F1 score and accuracy can not be computed. Then, we compute the GRQ value to asses the quality of the triclusters found by the STriGen.

4. Results

This Section presents the results obtained by the application of the STriGen algorithm to different datasets, three synthetics in Section 4.1 and one real in Section 4.2. In particular, Section 4.1.1 describes the synthetics datasets, in which STriGen has been tested. Then, a summary of the main parameter settings can be found in Section 4.1.2. Section 4.1.3 discusses the experimental results obtained by STriGen with different configuration settings.

4.1. Synthetic datasets

Synthetic datasets are essential to evaluate the quality of the algorithm, as we know in advance which should be the results, i.e., triclusters that STriGen must find.

4.1.1. Description of the synthetic datasets

The algorithm is applied on three synthetic datasets that follow additive, multiplicative and dynamic additive models respectively. A full description of these models can be found in [44,45]. This way, we can test how our model is capable to adapt to different types of changes over time in data streams. All three datasets are made up of 800 instances, 4 features

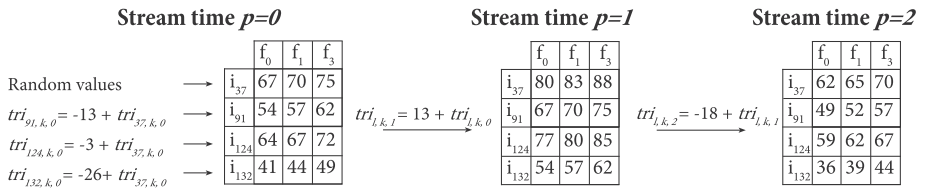


Fig. 5. Example of a tricluster in the additive dataset.

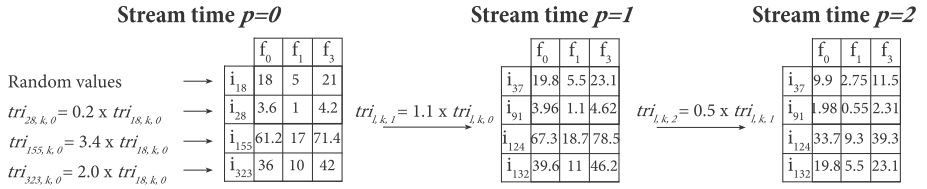


Fig. 6. Example of a tricluster in the multiplicative dataset.

and 500 streams for the two first datasets and 2100 streams for the third dataset. Instances are represented in rows, features in columns and streams represent the depth of each 3D dataset. Each dataset contains three triclusters with a different number of instances (between 25 and 30) and features (between 3 and 4). The components that are not in any tricluster, take random values.

- **Additive dataset:** For the additive synthetic dataset, a value $\gamma_p \in [-30, 30]$ has been added to each expression value in the tricluster at time p in order to create the additive pattern over time. Therefore, a tricluster *tri* at the stream time p in the additive dataset is defined as:

$$tri_{l,k,p} = \gamma_p + tri_{l,k,p-1} \quad 1 \leqslant l \leqslant L \quad 1 \leqslant k \leqslant K \tag{2}$$

  where the tricluster at the initial time, $tri_{l,k,0}$, is generated with random expression values for each feature of the first instance and the expression values for all features of the remaining instances according to the following equation:

$$tri_{l,k,0} = \alpha_l + tri_{1,k,0} \quad 2 \leqslant l \leqslant L \quad 1 \leqslant k \leqslant K \tag{3}$$

  where α_l is a random number between −30 and 30. In addition, an instance or a feature is added or deleted from the components of the tricluster at random times in order to represent a small evolution of the tricluster. The initial expression values for each instance or feature added at random times as a new tricluster component are random, and in the next time they will follow the same behaviour of the rest of the components of the tricluster. An example of a tricluster in the additive dataset is shown in Fig. 5.
- **Multiplicative dataset:** For the multiplicative dataset, each expression value in the tricluster at time p has been multiplied by a value $\sigma_p \in [0.1, 5.0]$ in order to create the multiplicative pattern over time. Thus, a tricluster *tri* at the stream time p in the multiplicative dataset follows the equation:

$$tri_{l,k,p} = \sigma_p \times tri_{l,k,p-1} \quad 1 \leqslant l \leqslant L \quad 1 \leqslant k \leqslant K \tag{4}$$

  where the tricluster at the initial time, $tri_{l,k,0}$, is also generated with random expression values for each feature of the first instance and the expression values for all features of all the other instances according to the following equation:

$$tri_{l,k,0} = \beta_l \times tri_{1,k,0} \quad 2 \leqslant l \leqslant L \quad 1 \leqslant k \leqslant K \tag{5}$$

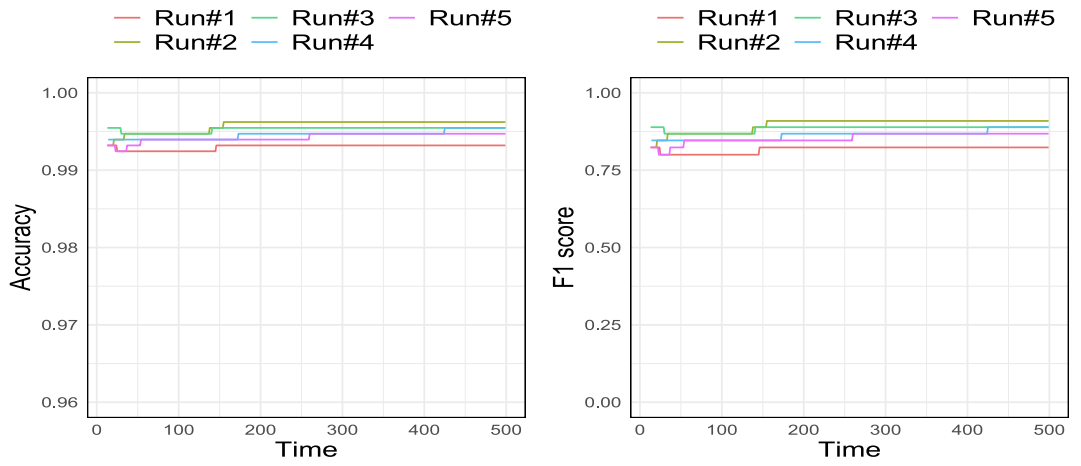
  where β_l is a random number between 0.1 and 5.0. The procedure of adding or deleting instances or features of the tricluster at random times follows the same behaviour explained in the generation of the additive dataset. Fig. 6 shows a graphical example of a tricluster in the multiplicative dataset.
- **Dynamic additive dataset:**The third dataset is a dynamic additive dataset, that is, it is generated with different additive datasets. In particular, it is made up of three different additive models: the first one from stream 0 to 399, the second from 400 to 1299 and the third from 1300 to 2100. The goal of this dataset is to test the performance of the proposed algorithm to abrupt changes.

4.1.2. Experimental setting

There are two main groups of parameters in the STriGen algorithm: typical parameters of the evolutionary algorithm in the offline phase and specific parameters of the STriGen. After a parameter tuning process and considering [6], the traditional triclustering parameters have been fixed. In particular, the number of solutions has been set to 3, the generations to 200, the members of the population to 400, the aleatory probability for the generation of the initial population to 0.2, the selection probability to 0.7 and the mutation probability to 0.4. To assess the influence of the specific parameters of the STriGen, five

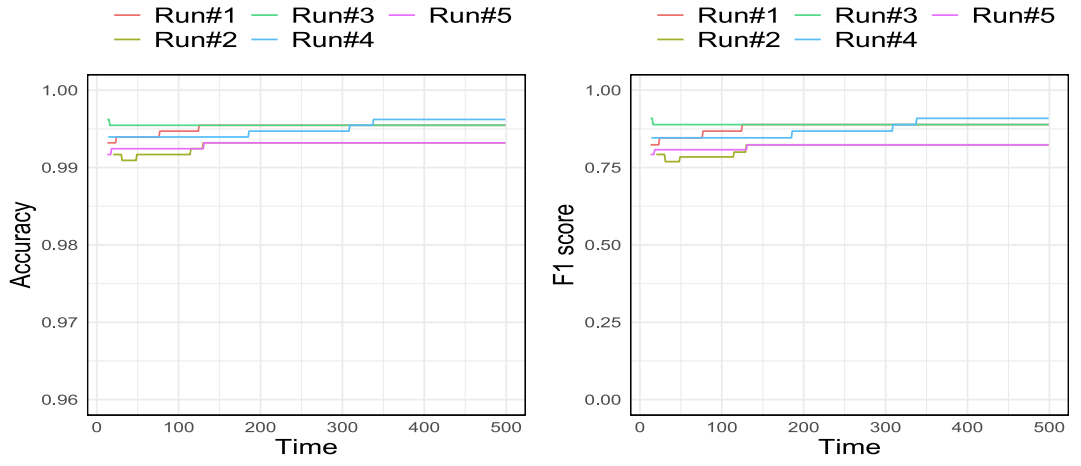
Table 1
STriGen parameters for synthetic datasets.

Parameter	Run 1	Run 2	Run 3	Run 4	Run 5
minGRQ	0.85	0.975	0.975	0.95	0.975
numIt	50	25	25	35	15
w	15	15	10	15	15



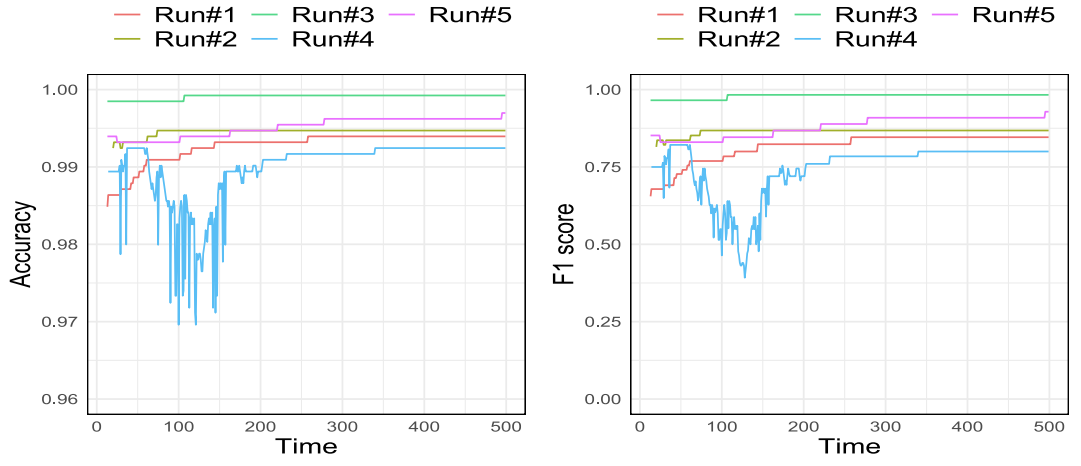
(a) First tricluster.

(b) First tricluster.



(c) Second tricluster.

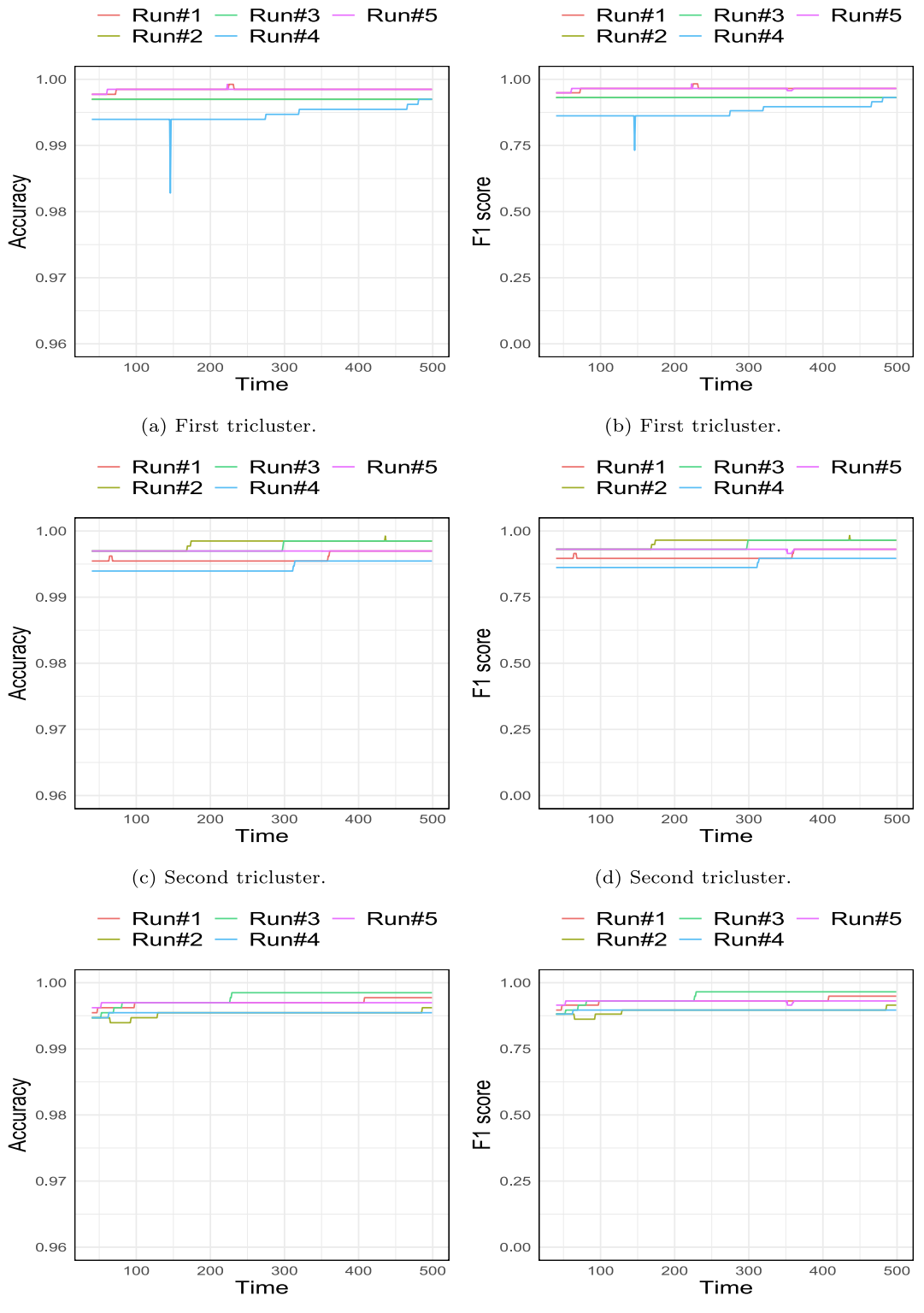
(d) Second tricluster.



(e) Third tricluster.

(f) Third tricluster.

Fig. 7. Accuracy and F1 score of the tricluster solutions in the additive dataset.

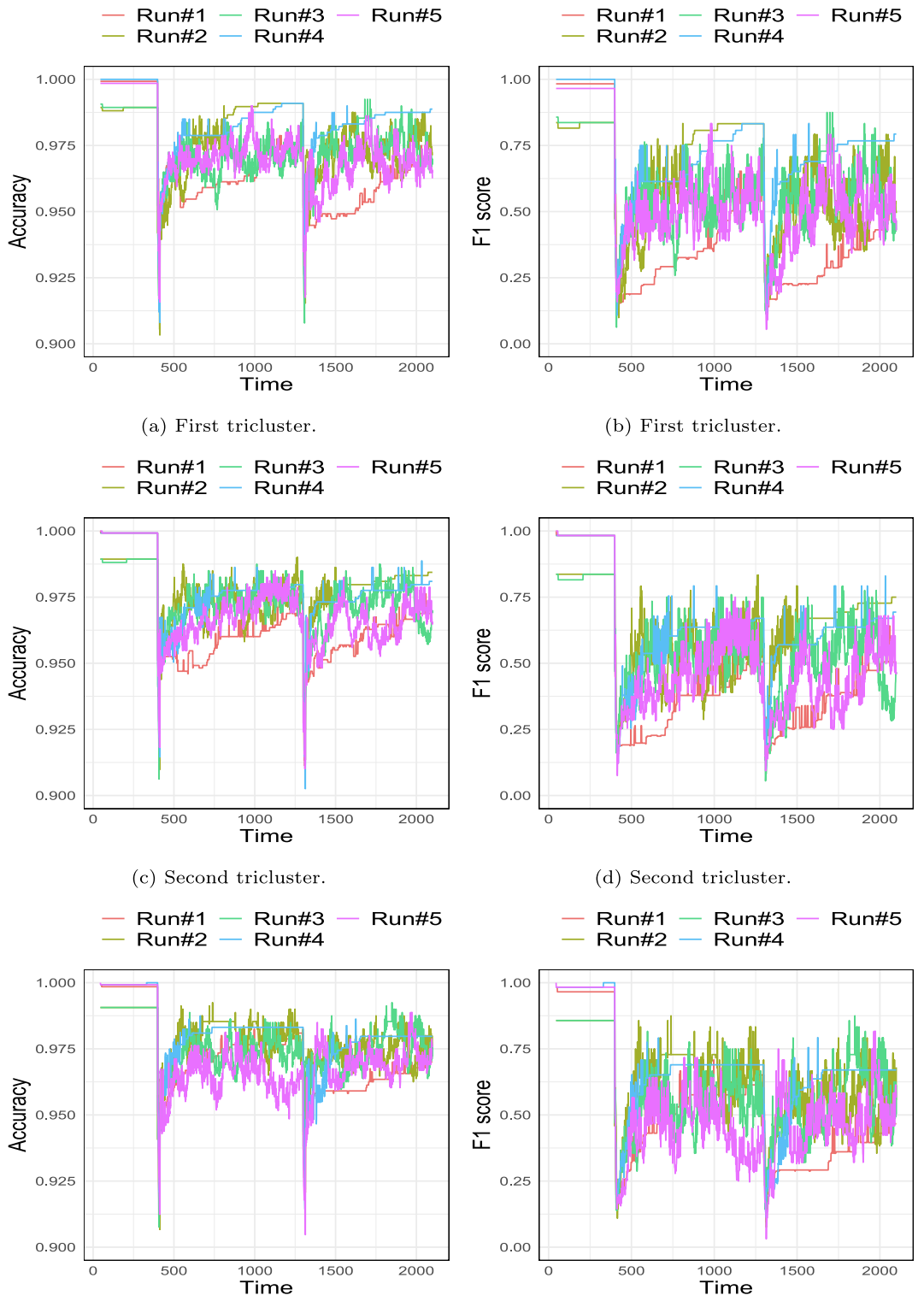


(a) First tricluster. (b) First tricluster.

(c) Second tricluster. (d) Second tricluster.

(e) Third tricluster. (f) Third tricluster.

Fig. 8. Accuracy and F1 score of the tricluster solutions in the multiplicative dataset.



(a) First tricluster.

(b) First tricluster.

(c) Second tricluster.

(d) Second tricluster.

(e) Third tricluster.

(f) Third tricluster.

Fig. 9. Accuracy and F1 score of the tricluster solutions in the dynamic additive dataset.

experiments with different parameters values have been carried out. Experiments shall be referred as 'Run' from this point on-wards. Table 1 presents the parameters for each experiment for all three synthetic datasets.

4.1.3. Analysis of results

Figs. 7–9 present the evolution over time of the accuracy and F1 score of the three triclusters obtained by the STriGen algorithm for the additive, multiplicative and dynamic additive datasets, respectively.

- **Additive dataset:** In Fig. 7, it can be seen that both accuracy and F1 score improve over time with final metrics closer to 1 (the maximum value). The mean value of the accuracy for all fifteen solution triclusters (five "runs" for each of the three triclusters) is 0.995. The maximum value of the F1 score is 0.983 and is reached with the parameters of the third run for the third solution tricluster. The smallest F1 score is 0.8 and is reached for the third solution tricluster but with the configuration of the forth run. Considering that the whole dataset has 800 instances, the STriGen algorithm has good results and, furthermore, it is capable of adapting the solutions to the new streams of data arriving. All triclusters found in the additive dataset follow a similar pattern in the procedure of finding the triclusters, excluding the third tricluster found on the forth run, as shown in Fig. 7e and in Fig. 7f. This exception is due to the high *minGRQ* and high *numIt* fixed for the forth run and because the GRQ obtained in the offline phase is 0.945. Therefore, STriGen has to improve the quality of the third tricluster by introducing more mutations from the beginning and not trying to find the evolution of the values as in the other triclusters. At the end of the forth run, the accuracy is 0.992 and the F1 score is 0.8. Regarding the control parameters, the tricluster with the highest accuracy and F1 score is the third tricluster with the parameters of the third run that get a mean GRQ value of 0.976.
- **Dynamic dataset:** The performance of the algorithm is also accurate for the multiplicative dataset as shown in Fig. 8. The mean value of the accuracy for all fifteen solution triclusters is 0.997. The final F1 score values range between 0.896 and 0.965, which are quite high values. All triclusters follow a similar pattern and they find the optimal components over time. All three triclusters are found for all five runs without any incidence. It is important to consider that the spikes are due to the fact that the F1 score increases when finding one new instance or feature on one particular stream and not gradually. In this case, even if all experiments with different parameters provide similar results, the configuration with the highest accuracy and F1 score is also obtained for the setup of the third run for the third tricluster that reaches a mean GRQ value of 0.987.
- **Dynamic additive dataset:** For the dynamic additive dataset, the components of the triclusters change completely at some streams and the difficulty in finding these new components that differ totally from the previous ones is high as can be seen in Fig. 9. In this way, theoretically, with higher *minGRQ* and *numIt* the algorithm is forced to maintain a high quality level and when it is not reached (usually when abrupt changes appear), it mutates instances and attributes components repeatedly to find the new ones and maintain a good tricluster quality. In addition, in this type of dataset, if w is very high, the algorithm might not notice that an abrupt change occurs until there are more streams of the new pattern than of the previous one. For the experiments of the setup of the third run with a low w, the GRQ value decreases earlier when an abrupt change occurs. The mean value of the accuracy for all the triclusters is 0.974. The evolution pattern of the values is similar each time a new abrupt change happens. When the change occurs, the GRQ, accuracy and F1 score decrease dramatically. Mutations allow to find the new members of the triclusters and so GRQ, accuracy and F1 score increase continuously. The best accuracy and F1 score is 0.9887 and 0.793, respectively, both them are reached for the first solution tricluster for the parameter configuration setup of the fourth run.

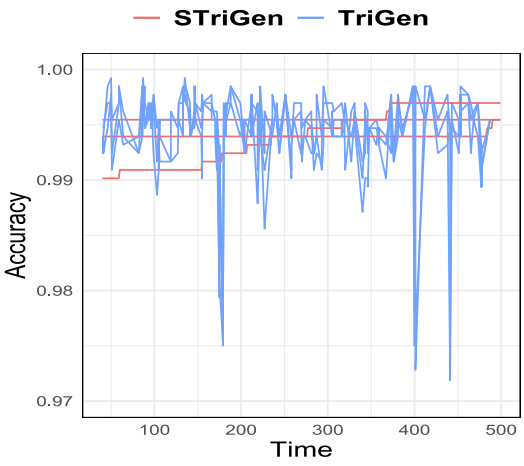
Summarizing, despite abrupt changes in the components of the triclusters, the STriGen algorithm performs correctly and obtains good results in terms of adaptation to concept drift, finding the majority of the new components even if they are totally different. Thus, it can be concluded that the STriGen has a huge potential as it is clearly able to detect patterns in data streams even if they change abruptly along time.

4.1.4. Comparison with TriGen benchmarking algorithm

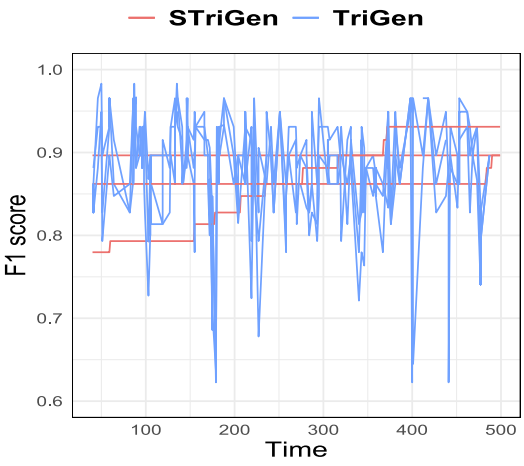
In this Section, the common points and differences between the TriGen and STriGen algorithms are firstly presented. Next, a comparison of the quality of the triclusters by means of F1 and accuracy along with the computational time is carried out for the triclusters found by both algorithms.

Table 2
Similarities and differences between TriGen and STriGen.

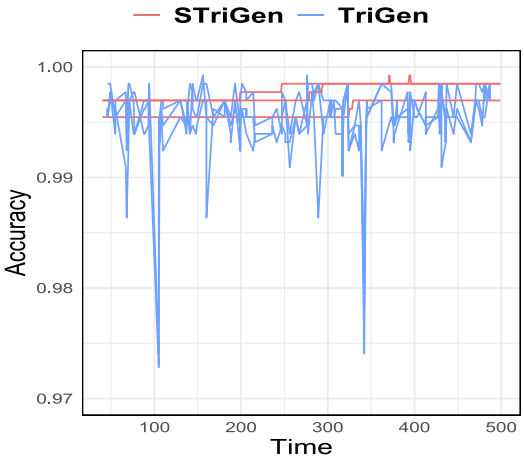
Common features	STriGen new features
• Evolutionary process. • Genetic operators. • Genetic parameters.	• Consecutive instant points. • Mutations keeps the consecutive instant points. • Streaming execution. • Streaming control parameters. • Streaming input dataset.



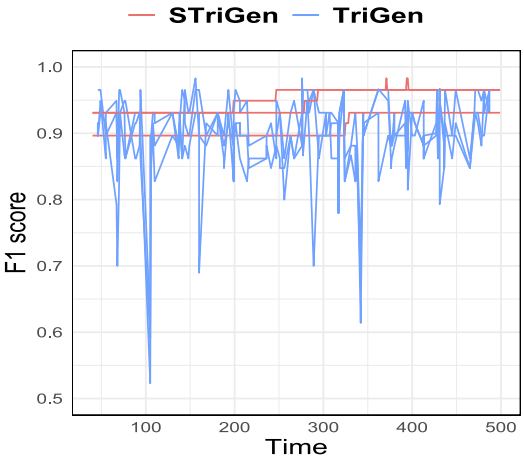
(a) Accuracy for the additive dataset.



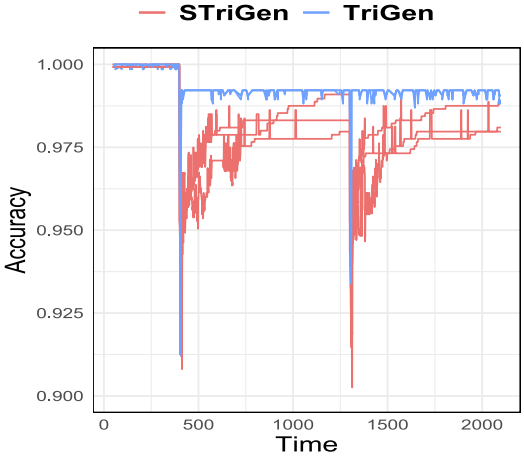
(b) F1 score for the additive dataset.



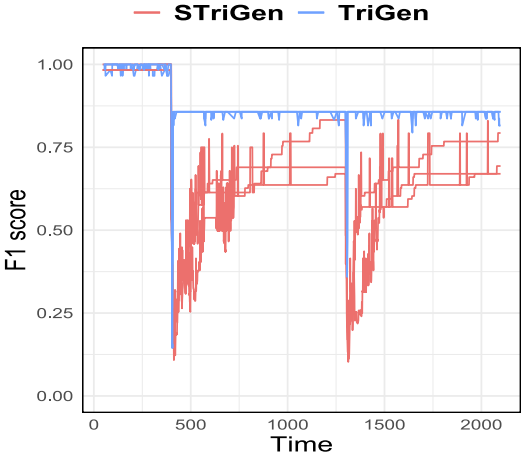
(c) Accuracy for the multiplicative dataset.



(d) F1 score for the multiplicative dataset.



(e) Accuracy for the dynamic additive dataset.



(f) F1 score for the dynamic additive dataset.

Fig. 10. Comparison of the performance for STriGen and TriGen algorithms.

(a) Additive dataset. (b) Multiplicative dataset. (c) Dynamic additive dataset.

Fig. 11. Comparison of the execution times for STriGen and TriGen algorithms.

A summary of the common characteristics between TriGen and STriGen and the new ones included by STriGen is shown in Table 2.

The common characteristics between both algorithms are mainly reduced to the offline phase of STriGen. TriGen and STriGen implement a triclustering algorithm based on an evolutionary process, which minimizes a fitness function to obtain the triclusters. Thus, both approaches share the fitness function to be minimized, and use the same genetic operators and typical control parameters of an evolutionary process in the phase offline of STriGen.

As for the new features of STriGen, there are two main novelties. Firstly, about the instant points of a tricluster, STriGen considers them as a complete and well-formed time series. That is, as a consecutive sequence of time points in opposition to the TriGen algorithm, where this feature is not taken into account. Furthermore, the mutation operator of STriGen respects the consecutive instant point feature of the tricluster when they are altered. This new characteristic is a key-point in the new STriGen algorithm in order to be adapted to the streaming environment.

Finally, the second novelty is the capability of the STriGen algorithm to analyze streaming data. STriGen is run in a streaming environment where it analyses an in-motion input dataset, increasing whenever the time points move forward. Therefore, STriGen can find the evolution of patterns during the arrival of data streams and adapt the model accordingly, in opposition to TriGen, where both the input data and its models are static. In addition, the streaming feature of STriGen implies the inclusion of new control parameters in its online phase.

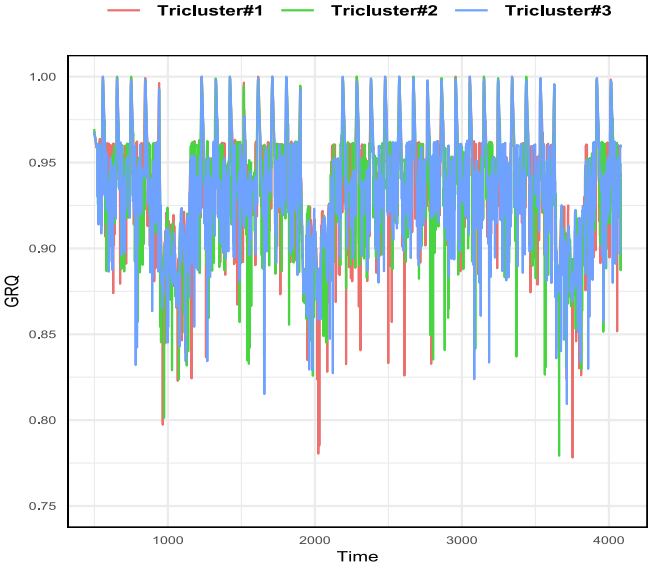


Fig. 12. GRQ values using STriGen for the real sensor dataset.

Next, the results obtained by STriGen have been compared to the TriGen triclustering algorithm [6]. The best STriGen execution for each synthetic dataset between all five "run" is selected to carry out the comparison. The TriGen algorithm is rerun each time a new data stream arrives. At each execution, TriGen considers just the previous w data samples. Fig. 10 shows the mean of the accuracy and F1 score for the three triclusters obtained by the STriGen and TriGen algorithms. It can be observed that the STriGen takes advantage of the sequential behaviour and finds the corresponding components of the triclusters for the additive and multiplicative datasets easily as shown in Fig. 10a, 10b, 10c and 10d. However, as TriGen does not consider the evolution over time of the triclusters components, it does not find the corresponding components as accurately as the STriGen streaming algorithm. In the case of the dynamic additive dataset when abrupt changes occurs, the STriGen algorithm takes some time to find the new components of the triclusters in contrast to TriGen as shown in Fig. 10e and 10f. The main reason is that TriGen considers just the previous w samples and not the evolution over time of the triclusters' components as STriGen does.

One of the critical aspects in streaming models is the bounded response time. Fig. 11 presents the runtimes of the STriGen and TriGen algorithms. The offline phase of STriGen is quite time consuming (first value of the graphic) and the next executions consume almost imperceptible time compared to TriGen execution times, since TriGen has to be recomputed each time a new data stream arrives.

4.2. Real sensor dataset

4.2.1. Description of the real dataset

The real-world dataset contains data of seven sensors that record environmental data such as atmospheric pressure, precipitation, relative humidity, solar radiation, temperature, wind direction and wind speed. These sensors are placed in 12 different areas of Malaga (Spain). Some ones record data in just one location. However, other ones register data from different locations in its territory, for example Malaga capital city that records in 10 different locations.

4.2.2. Experimental setting

The offline phase is computed with the first 500 streams. Afterwards, each time a new stream arrives, the STriGen algorithm updates in quasi real time the triclusters and provides results. The experiment is carried out for 5000 streams, where w is fixed to 20, *minGRQ* to 0.975 and *numIt* to 35 after a process of tuning of the parameters in order to use the most accurate values.

4.3. Analysis of the results

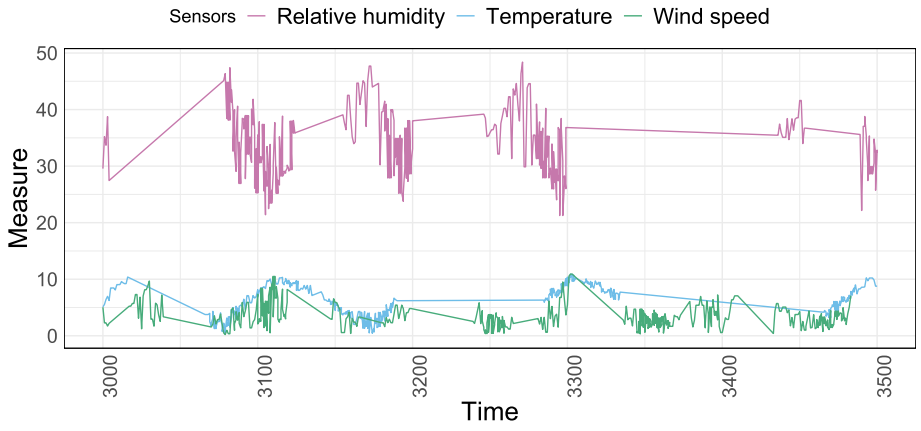
Fig. 12 shows the evolution over time of the values of GRQ for the STriGen algorithm. The average GRQ value for the three triclusters is 0.9468. These GRQ values are between 0.78 and 0.99, so STriGen exhibits a notable improvement.

However, as it is a real dataset and the ground truth is not known, the GRQ quality measure is not sufficient. In order to improve it, a baseline algorithm is used to compare results. The baseline algorithm is a simple triclustering in streaming. In particular, the STriGen algorithm is modified in such a way that the algorithm do not select the best tricluster, i.e. with the best GRQ, as defined in Algorithm 2, but a random tricluster between all the possibilities. This random assignment of the triclusters is the baseline algorithm used to compare the results. The mean and standard deviation of the GRQ values for each tricluster found by the STriGen algorithm and the baseline algorithm are presented in Table 3. The results of the STriGen algorithm are higher than the ones of the baseline. Thus, STriGen is adapting correctly to the evolution of the data streams of the sensors dataset.

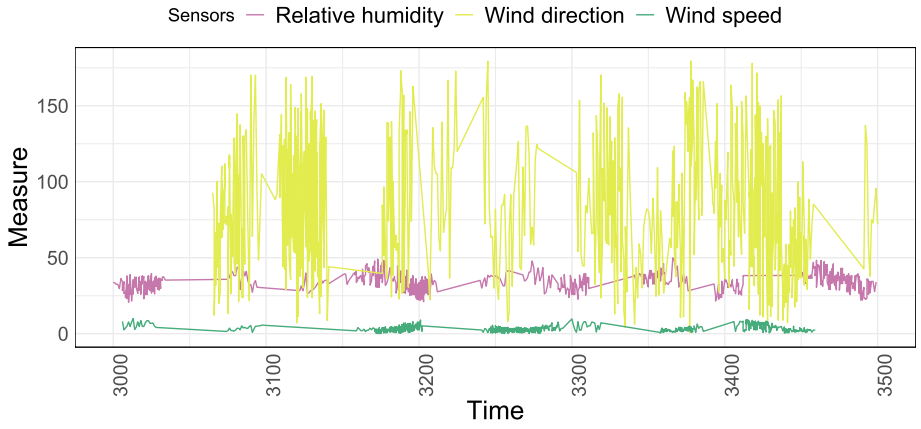
Fig. 13 represents the three triclusters obtained by the STriGen algorithm from time 3000 to 3500 in order to show the usefulness of the patterns found. Note that patterns evolve over time as streaming data arrives, for example at a time z, the components of a tricluster can be totally different from the ones at time $z + 50$. For this reason, this figure represents the evolution of the three triclusters only from time 3000 to 3500 as it is complex to visualize all patterns for each of the 4500 data streams of the dataset. The first tricluster in Fig. 13a corresponds to interior areas, in particular the areas of Antequera and Ronda, and they mainly include components of the relative humidity, temperature and wind speed. The second tricluster in Fig. 13b is made up of the nearest areas to the sea of the capital city of Málaga and they contain mainly components of the relative humidity, wind direction and wind speed sensors. The third tricluster in Fig. 13c contains west coast areas such as the towns of Estepona and Fuengirola, and include mainly instances of the solar radiation, temperature and wind speed sensors. This is just a particular sample of found patterns. In this case, there are three clearly different areas (in-

Table 3
STriGen and baseline comparison.

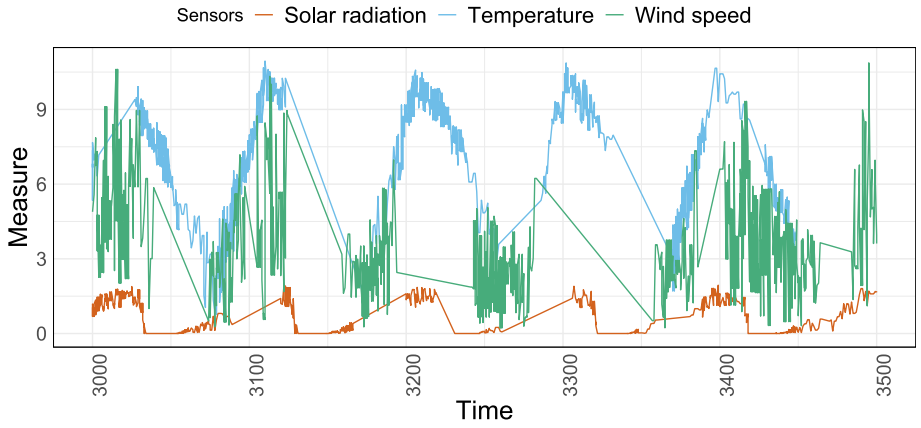
Algorithm	Tricluster 1		Tricluster 2		Tricluster 3	
	Mean	Std	Mean	Std	Mean	Std
STriGen	0.9398	0.0294	0.9390	0.0289	0.9614	0.0299
Baseline	0.7653	0.0314	0.7682	0.0302	0.7665	0.3533



(a) Tricluster#1.



(b) Tricluster#2.



(c) Tricluster#3.

Fig. 13. Triclusters patterns for the real sensor dataset from time 3000 to 3500.

terior, near to the sea in the east-side and near to the sea in the west-side) with different sensors instances. The precipitation and atmospheric pressure sensors are not present in these patterns as the precipitation is zero in all areas and the pressure is very similar in these time intervals. In the same way, Fig. 13 only includes the three more significant sensors for each tricluster, as the other sensors are only present in these triclusters occasionally and are not relevant for the meaning of the patterns.

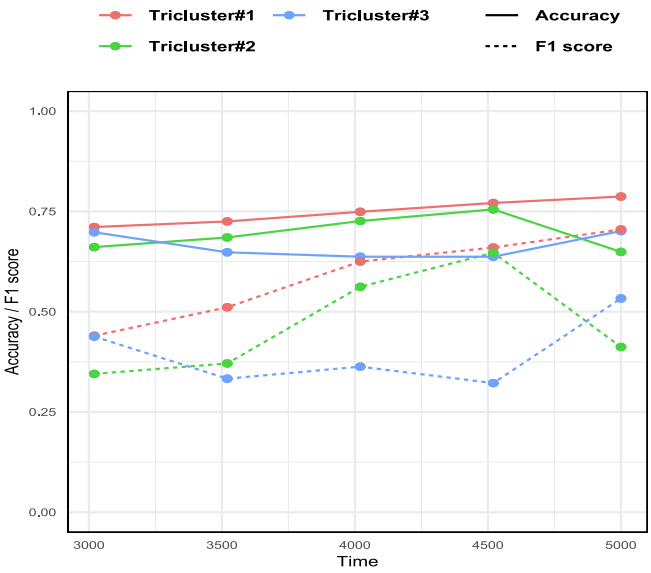


Fig. 14. Comparison of results using TriGen and STriGen for the real sensor dataset.

In addition, the TriGen algorithm has been executed in order to consider its triclusters as the "real" ones and the STriGen triclusters as the "found" ones. Besides the comparison with a baseline algorithm, this is an additional option to deal with real datasets where the ground truth is unknown. This test has been carried out 5 times, in particular for streams from 3000 to 3020, 3500 to 3520, 4000 to 4020, 4500 to 4520 and 4980 to 5000. The TriGen algorithm is executed just in the w last streams, namely 20 streams, and therefore, in this particular case the evolution over the different streams is not considered. Fig. 14 presents the accuracy and F1 score obtained when comparing the TriGen and STriGen results using the real sensor dataset. The highest accuracy is 0.787 and the highest F1 score is 0.705 for the STriGen algorithm. Even if TriGen is not executed in all streams, the F1 score, accuracy and GRQ results of the STriGen ensures its good performance. In addition, the STriGen algorithm keeps improving its results over time.

5. Conclusions

This work has introduced a new triclustering algorithm in the streaming environment. The algorithm consists of two stages: an offline or batch phase and an online phase. The first one creates a sketch or summary model with the triclusters components that optimize the fitness function. This fitness function considers the similarity between the angles of the slopes that represent the values of the tricluster components. In addition, the GRQ quality measure is computed to evaluate the evolution over time of the triclusters. Then, considering the offline summary model, the online phase of STriGen is computed satisfying all the requirements of data streaming, i.e., each sample is processed just once and in the order of its arrival and the model stores a limited amount of data and updates the model with a low computational cost. The STriGen has been proved to be able to discover collections of resembling patterns in 3D stream data. The algorithm has been applied to three synthetic datasets with different characteristics and to one real dataset of environmental sensors. Results have shown that the algorithm detects both little and huge changes of instances and features in triclusters components. The validation of the experiments is carried out comparing the found triclusters to the real triclusters in the case of the synthetic datasets and to the solution found by the TriGen batch triclustering algorithm published in the literature in the case of the real sensor dataset. For both type of datasets, the quality of the triclusters found is similar to the quality of the triclusters obtained by the TriGen with much less execution time. Up to our knowledge, there are not many triclustering streaming algorithms in the literature, so this field of research is noteworthy due to vast amount of streaming data sources available nowadays. Another advantage of the proposed algorithm is the good performance in terms of accuracy and execution times provided for datasets of different types and volumes.

The future work will be focused on addressing when a new stream concept is an outlier pattern and alert about that. In addition, we plan to add different fitness functions and evaluation measures to extend the approach of the STriGen proposed algorithm.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank the Spanish Ministry of Science, Innovation and Universities for the support under the project TIN2017-88209-C2.

References

[1] P. Larrañaga, D. Atienza, J.D. Rozo, A. Ogbechie, C. Puerto-Santana, C. Bielza, Industrial Applications of Machine Learning, CRC Press, 2018.
[2] H. Wang, A. Zubin, Concept drift detection for streaming data, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2015, pp. 1–9.
[3] J. Gama, A survey on learning from data streams: current and future trends, Progr. Artif. Intell. 1 (1) (2012) 45–55.
[4] C. Rubio-Escudero, F. Martínez-Álvarez, R. Romero-Zaliz, I. Zwir, Classification of gene expression profiles: comparison of k-means and expectation maximization algorithms, in: Proceedings of the 8th International Conference on Hybrid Intelligent Systems, 2008, pp. 831–836.
[5] J.A. Hartigan, Direct clustering of a data matrix, J. Am. Stat. Assoc. 67 (337) (1972) 123–129.
[6] D. Gutiérrez-Avilés, C. Rubio-Escudero, F. Martínez-Álvarez, J. Riquelme, TriGen: a genetic algorithm to mine triclusters in temporal gene expression data, Neurocomputing 132 (2014) 42–53.
[7] D. Gutiérrez-Avilés, C. Rubio-Escudero, MSL: a measure to evaluate three-dimensional patterns in gene expression data, Evolut. Bioinform. 11 (2015) 121–135.
[8] B. Zhou, J. Li, X. Wang, Y. Gu, L. Xu, Y. Hu, L. Zhu, Online Internet traffic monitoring system using spark streaming, Big Data Mining Anal. 1 (1) (2018) 47–56.
[9] L.P. Liu, Y. Jiang, Z.H. Zhou, Least square incremental linear discriminant analysis, in: Proceedings of the IEEE International Conference on Data Mining, 2009, pp. 298–306.
[10] C. Za'in, M. Pratama, E. Pardede, Evolving large-scale data stream analytics based on scalable PANFIS, Knowl. -Based Syst. 166 (2019) 186–197.
[11] B. Krawczyk, Active and adaptive ensemble learning for online activity recognition from data streams, Knowl. -Based Syst. 138 (2017) 69–78.
[12] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Trans. Neural Networks 22 (12) (2011) 1901–1914.
[13] A. Bifet, G.F. Morales, Big data stream learning with SAMOA, in: Proceedings of the IEEE International Conference on Data Mining Workshop, 2015, pp. 1199–1202.
[14] S.C. Pallaprolu, R. Sankineni, M. Thevar, G. Karabatis, J. Wang, Zero-day attack identification in streaming data using semantics and Spark, in: Proceedings of the IEEE International Congress on Big Data, 2017, pp. 121–128.
[15] U. Rajeshwari, B.S. Babu, Real-time credit card fraud detection using streaming analytics, in: Proceedings of the 2nd International Conference on Applied and Theoretical Computing and Communication Technology, 2016, pp. 439–444.
[16] S. Papadimitriou, J. Sun, C. Faloutsos, Streaming pattern discovery in multiple time-series, in: Proceedings of 31st International Conference on Very Large Data Bases, vol. 2, 2005, pp. 697–708.
[17] K. Yu, W. Ding, D.A. Simovici, H. Wang, J. Pei, X. Wu, Classification with streaming features: an emerging-pattern mining approach, ACM Trans. Knowl. Discovery Data 9 (4) (2015) 1–31.
[18] Y. Chen, K. Chen, M.A. Nascimento, Effective and efficient shape-based pattern detection over streaming time series, IEEE Trans. Knowl. Data Eng. 24 (2) (2012) 265–278.
[19] C. Garcia, A. Esmin, D. Leite, I. Škrjanc, Evolvable fuzzy systems from data streams with missing values: with application to temporal pattern recognition and cryptocurrency prediction, Pattern Recogn. Lett. 128 (2019) 278–282.
[20] I. Škrjanc, J.A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey, Inf. Sci. 490 (2019) 344–368.
[21] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, Evolving Syst. 6 (4) (2015) 269–292.
[22] D. Leite, G. Andonovski, I. Škrjanc, F. Gomide, Optimal rule-based granular systems from data streams, IEEE Trans. Fuzzy Syst. 28 (3) (2020) 583–596.
[23] I. Škrjanc, Cluster-volume-based merging approach for incrementally evolving fuzzy gaussian clustering—egauss+, IEEE Trans. Fuzzy Syst. 28 (9) (2020) 2222–2231.
[24] D. Leite, I. Škrjanc, F. Gomide, An overview on evolving systems and learning from stream data, Evolving Syst. 11 (2) (2020) 181–198.
[25] M. Ackerman, S. Dasgupta, Incremental Clustering: The Case for Extra Clusters, in: Proceedings of the Neural Information Processing Systems, 2014, pp. 1–13.
[26] M. Capó, A. Pérez, J.A. Lozano, An efficient approximation to the K-means clustering for massive data, Knowl.-Based Syst. 117 (2017) 56–69.
[27] U. Kokate, A. Deshpande, P. Mahalle, P. Patil, Data stream clustering techniques, applications, and models: comparative analysis and discussion, Big Data Cognit. Comput. 2 (4) (2018) 32.
[28] R. Henriques, S. Madeira, Triclustering algorithms for three-dimensional data analysis: a comprehensive survey, ACM Comput. Surv. 51 (5) (2018) 1–43.
[29] L. Zhao, M.J. Zaki, triCluster: an effective algorithm for mining coherent clusters in 3D microarray data, in: Proceedings of the ACM SIGMOD International Conference on Management of data, 2005, pp. 694–705.
[30] H. Jiang, S. Zhou, J. Guan, Y. Zheng, gTRICLUSTER: A More General and Effective 3D Clustering Algorithm for Gene-Sample-Time Microarray Data, in: Proceedings of the Data Mining for Biomedical Applications, 2006, pp. 48–59.
[31] J. Liu, Z. Li, X. Hu, Y. Chen, Multi-objective evolutionary algorithm for mining 3D clusters in gene-sample-time microarray data, in: Proceedings of the IEEE International Conference on Granular Computing, 2008, pp. 442–447.
[32] N. Narmadha, R. Rathipriya, Evolutionary correlation triclustering for 3d gene expression data, in: Innovative Data Communication Technologies and Application, Springer International Publishing, 2020, pp. 637–646.
[33] A. Tchagang, S. Phan, F. Famili, H. Shearer, P. Fobert, Y. Huang, J. Zou, D. Huang, A. Cutler, Z. Liu, Y. Pan, Mining biological information from 3D short time-series gene expression data: the optricluster algorithm, BMC Bioinform. 13 (2012) 54.
[34] F. Martínez-Álvarez, D. Gutiérrez-Avilés, A. Morales-Esteban, J. Reyes, J. Amaro-Mellado, C. Rubio-Escudero, A novel method for seismogenic zoning based on triclustering: application to the iberian peninsula, Entropy 17 (12) (2015) 5000–5021.
[35] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, High-content screening images streaming analysis using the strigen methodology, in: Proceedings of the 35th Annual ACM Symposium on Applied Computing, 2020, pp. 537–539.
[36] D. Soares, R. Henriques, M. Gromicho, S. Pinto, M. de Carvalho, S.C. Madeira, Towards triclustering-based classification of three-way clinical data: A case study on predicting non-invasive ventilation in als, in: Practical Applications of Computational Biology & Bioinformatics, 14th International Conference (PACBB 2020), Springer International Publishing, 2021, pp. 112–122.
[37] P. Mahanta, H.A. Ahmed, D.K. Bhattacharyya, J.K. Kalita, Triclustering in gene expression data analysis: a selected survey, in: 2011 2nd National Conference on Emerging Trends and Applications in Computer Science, 2011, pp. 1–6.
[38] N. Narmadha, R. Rathipriya, Triclustering: an evolution of clustering, in: Proceedings of the Online International Conference on Green Engineering and Technologies, 2016, pp. 1–4.
[39] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: theory and practice, IEEE Trans. Knowl. Data Eng. 15 (3) (2003) 515–528.
[40] D.A. Umbarkar, P. Sheth, Crossover operators in genetic algorithms: a review, ICTACT J. Soft Comput. 6 (1) (2015) 1083–1092.

[41] M. Ghesmoune, M. Lebbah, H. Azzag, State-of-the-art on clustering data streams, Big Data Anal. 1 (1) (2016) 1–27.
[42] M.S. Hammoodi, F. Stahl, A. Badii, Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining, Knowl.-Based Syst. 161 (2018) 205–239.
[43] R.H. Moulton, H.L. Viktor, N. Japkowicz, J. Gama, Clustering in the presence of concept drift, in: Proceedings of the ECML/PKDD Machine Learning and Knowledge Discovery in Databases, 2018, pp. 339–355.
[44] B. Pontes, R. Giráldez, J.S. Aguilar-Ruiz, Quality measures for gene expression biclusters, PLOS ONE 10 (3) (2015) 1–24.
[45] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, On biclustering of gene expression data, Curr. Bioinform. 5 (2010) 204–216.