Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Evolving clustering algorithm based on mixture of typicalities for stream data mining

José Maia [a], Carlos Alberto Severiano Junior [a,b,c], Frederico Gadelha Guimarães [a,b,*], Cristiano Leite de Castro [a,b], André Paim Lemos [a], Juan Camilo Fonseca Galindo [a], Miri Weiss Cohen [d]

[a] *Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil*
[b] *Machine Intelligence and Data Science (MINDS) Laboratory, Federal University of Minas Gerais, Belo Horizonte, Brazil*
[c] *Instituto Federal de Minas Gerais, Campus Sabará, Brazil*
[d] *Department of Software Engineering, Braude College of Engineering, Karmiel, Israel*

ARTICLE INFO

Article history:
Received 15 May 2019
Received in revised form 10 January 2020
Accepted 10 January 2020
Available online 24 January 2020

Keywords:
Clustering
Data stream
Concept drift
Stream data mining
Evolving fuzzy systems

ABSTRACT

Many applications have been producing streaming data nowadays, which motivates techniques to extract knowledge from such sources. In this sense, the development of data stream clustering algorithms has gained an increasing interest. However, the application of these algorithms in real systems remains a challenge, since data streams often come from non-stationary environments, which can affect the choice of a proper set of model parameters for fitting the data or finding a correct number of clusters. This work proposes an evolving clustering algorithm based on a mixture of typicalities. It is based on the TEDA framework and divide the clustering problem into two subproblems: micro-clusters and macro-clusters. Experimental results with benchmarking data sets showed that the proposed methodology can provide good results for clustering data and estimating its density even in the presence of events that can affect data distribution parameters, such as concept drifts. In addition, the model parameters were robust in relation to the state-of-the-art algorithms.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Clustering is a very important learning task for many applications. The scope of this task goes beyond data partitioning, since clustering is also useful for feature engineering [1] and to automatically generate the parameters of other machine learning algorithms like fuzzy systems and Radial Basis Function (RBF) neural networks [2,3].

In the current scenario, the increasing amount of streaming data claims for the development of clustering algorithms with the ability to discover new patterns in online setting. Data streams often come from non-stationary environments with no prior information about the data distribution or the true number of clusters. Further, the data distribution as well as the number of clusters may change over time. Thus, in contrast to batch methods, algorithms for clustering online data streams process data one element at a time, update clusters structure whenever required and store in the memory a small number of elements or prototypes as a representation of the clusters. Nowadays, online clustering becomes an important tool for many applications such as knowledge discovery [4], process fault detection [5,6], recommendation systems and anomaly detection [7].

In this context, many approaches have been proposed for clustering data streams [8]. The incremental version of k-means [9] is a simple but efficient distance-based algorithm. Although incremental k-means presents low computational cost, it requires some prior knowledge about the problem, for example, the number of clusters k that should be known in advance. In addition, distance-based methods often assume that clusters come from the same distribution (or shape) according to the used distance metric.

Another kind of more advanced algorithms first divide the streaming data into micro-clusters and then find the final clusters based on micro-clusters that share common properties. This approach has been proven to be effective to incremental cluster of data stream [11] and is being widely used in this context. Clustream algorithm [12] follows this approach by using samples within a time window to create and update micro-clusters incrementally, however, like incremental k-means, Clustream has problems when dealing with non-spherical clusters.

* Corresponding author at: Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil.

E-mail addresses: jmnt@ufmg.br (J. Maia), fredericoguimaraes@ufmg.br (F.G. Guimarães), crislcastro@ufmg.br (C.L. de Castro), andrelemos@ufmg.br (A.P. Lemos), juankmilofg@ufmg.br (J.C. Fonseca Galindo), miri@braude.ac.il (M.W. Cohen).

https://doi.org/10.1016/j.future.2020.01.017
0167-739X/© 2020 Elsevier B.V. All rights reserved.

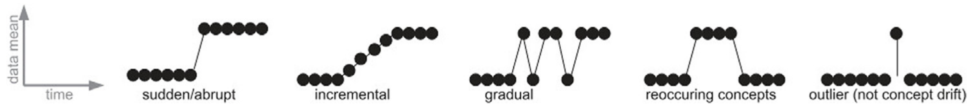


Fig. 1. Patterns of changes over time [10].

Density based algorithms like DBstream [13], Denstream [14] can automatically find the actual number of clusters at each step and create clusters of arbitrary shapes, but the large number of free parameters makes the search for the best model very difficult for different applications [8].

Most of distance-based and density-based algorithms are not able to handle non-stationary data because they have fixed structure and predefined parameters that should be known in advance. Recently many evolving clustering algorithms have been proposed for clustering data streams. These algorithms have the ability to evolve their structure and parameters over time as new data arrives [15–18]. The majority of the evolving clustering algorithms however, are suitable for applications in which the groups represent sequential behaviors or states of a system, like in anomaly detection. This is not the case of many data mining tasks where it is required an online clustering algorithm that is able to properly partition data from different clusters that arrive in random order. The CEDAS algorithm [15] can cluster data from arbitrary shapes that arrive in random order by creating micro and macro-clusters and can adapt to data drift. However, its output is only cluster assignment and it is not possible to discover the densities of the regions in the data space.

In this paper, we aim to address these gaps by presenting a new evolving clustering algorithm called MicroTEDAclus. Similarly to CEDAS [15], this algorithm divides the problem into micro-clusters and macro-clusters.

The micro-clusters are based on the concept of Typicality and Eccentricity Data Analysis (TEDA) [19]. Firstly, the algorithm updates TEDA micro-clusters that have their variance limited by an upper bound threshold that grows dynamically (σ_0) as more data is used on TEDA calculation. Next, the macro-cluster structure is updated. Each macro-cluster is composed by a set of connected micro-clusters that intersect each other. We then filter the macro-cluster structure in order to set active only its densest micro-clusters and estimate the density of a macro-cluster by a weighted sum of the typicalities of its connected micro-clusters, as in a density mixture model. Finally, the algorithm assigns a new data point to the macro-cluster that has the highest membership according the mixture of typicalities score. The proposed evolving clustering algorithm has the following characteristics:

- A single-pass procedure to update cluster prototypes and estimate its densities one sample at a time.
- Cluster data from arbitrary distributions (or shapes) that arrives in an arbitrary order.
- The method does not have user defined parameters to control granularity or micro-clusters radii.
- The output is not only the cluster assignment but also the membership of a new data point to all the existing clusters.

We evaluate the proposed method by comparing MicroTEDAclus with state-of-the-art algorithms in artificial data sets that simulate different situations of concept drift.

The rest of the paper is organized as follows. In Section 2 we discuss the non-stationarity in data streams caused by concept drift events. Section 3 introduces the concepts of TEDA, on which the proposed approach is based. In Section 4 we present a detailed description of the proposed algorithm. In Section 5, the methodology and experiments are described. In Section 6 we present the experimental results. Finally, we conclude the paper with discussions and future work proposals in Section 8.

2. Non-stationarity in data streams

Data streams are data sets generated by some underlying distribution assigned to a temporal aspect. As discussed earlier, data streams often present non-stationary patterns. Given that such non-stationarity could be represented by a sequence of different data distributions (or concepts), the effects can be explained by a set of **concept drifts**, which are changes in the parameters of the data distribution. A qualitative categorization of concept drift is presented in [10]. Sudden or abrupt concept drift occurs when a concept is immediately (in the next time step) replaced by another one with different parameters. The next concept usually presents a significant change if compared to the previous one. Incremental drift can also result in a significant change, but during the process of change there is a sequence of intermediate concepts. In a gradual drift, there is an alternation between components. Fig. 1 illustrates different types of concept drift. It is worth noticing that punctual changes in data, which over time do not represent new distributions, cannot be considered concept drifts but, in most cases, outliers.

Given a data stream whose data objects $o = \langle x, y \rangle$ where x is the feature vector and y is the class label. Each is drawn from a different random variable, X and Y: $x \in dom(X)$, $y \in dom(Y)$ and $o \in dom(X, Y)$. The concepts of this data stream are represented by probabilities associated to its data distribution [20]. It means that a concept drift of duration d occurs when time points t and $t + d$ present different probabilities $P_t(X, Y) \neq P_{t+d}(X, Y)$. It can be caused by changes in $P(X)$ or in $P(Y|X)$. In addition to duration, another factor to be considered in a quantitative characterization of a drift is its magnitude. It represents the distance between the concepts at times t and $t + d$. It can be measured by a distribution distance function. As mentioned in [21], Kullback–Leibler Divergence and Helling Distance are examples of metrics that may be used as distance function for concept drift.

When no labels are available for the problem, concept drift can be related to changes in $P(X)$. Although there is no previous labeling, there is still an underlying data partition structure, which may be different when the data distribution changes. A special case of concept drift, named concept evolution [20], refers to a situation when a novel pattern emerges, as a result of a structural changes in data distribution. Although it is usually related to change in the number of classes, thus being associated with discussions about supervised problems, $P(X)$ is also likely to be affected, which means that concept evolution can be considered in an unsupervised context.

3. TEDA

The proposed algorithm is based on the concepts of TEDA framework [19]. TEDA is an algorithm for anomaly detection that incrementally models a non-parametric data distribution based only on the proximity between data samples.

TEDA is based on the concept of cumulative proximity. Given a d-dimensional input vector $\boldsymbol{x_k} \in \mathbb{R}^d$, in the timestamp k, the cumulative proximity $\pi(.)$ of $\boldsymbol{x}_k$ with respect to all existing data samples, is calculated as:

$$\pi_k(\boldsymbol{x}) = \sum_{i=1}^{k} d(\boldsymbol{x}_k, \boldsymbol{x}_i) \ , \tag{1}$$

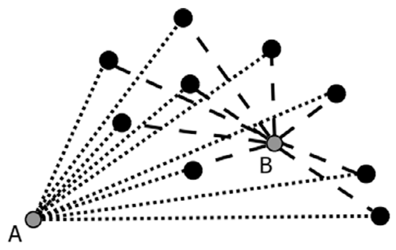


Fig. 2. Illustration of Typicality and Eccentricity concepts in TEDA [18].

where $d(\boldsymbol{a},\boldsymbol{b})$ is the distance between data points $\boldsymbol{a}$ and $\boldsymbol{b}$, k is the timestamp when the data point $\boldsymbol{x}$ is sampled.

From $\pi_k(\boldsymbol{x})$ we calculate the eccentricity $\xi_k(\boldsymbol{x})$, which is a measure of the dissimilarity between the data point $\boldsymbol{x}_k$ with respect to all the data points received until timestamp k.

$$\xi_k(\boldsymbol{x}) = \frac{2\pi_k(\boldsymbol{x})}{\sum_{i=1}^{k}\pi_k(\boldsymbol{x}_i)}\ ,\ \sum_{i=1}^{k}\pi_k(\boldsymbol{x}_i) > 0\ ,\ k > 2. \tag{2}$$

For the case of euclidean distance, eccentricity can be calculated recursively as follows [7].

$$\xi(\boldsymbol{x}_k) = \frac{1}{k} + \frac{(\boldsymbol{\mu}_k - \boldsymbol{x}_k)^T(\boldsymbol{\mu}_k - \boldsymbol{x}_k)}{k\sigma_k^2}, \tag{3}$$

where $\boldsymbol{\mu}_k$ and σ_k^2 are the mean and variance respectively, that can also be recursively updated:

$$\boldsymbol{\mu}_k = \frac{k-1}{k}\boldsymbol{\mu}_{k-1} + \frac{\boldsymbol{x}_k}{k}\ ,\ k \geq 1\ ,\ \boldsymbol{\mu}_1 = \boldsymbol{x}_1. \tag{4}$$

$$\sigma_k^2 = \frac{k-1}{k}\sigma_{k-1}^2 + \frac{1}{k-1}\|\boldsymbol{x}_k - \boldsymbol{\mu}_k\|^2\ ,\ \sigma_1^2 = 0 \tag{5}$$

The typicality $\tau(\boldsymbol{x}_k)$ is the dual of the eccentricity and represents how typical an arbitrary data point $\boldsymbol{x}_k$ is with respect to all the data points received until the timestamp k.

$$\tau(\boldsymbol{x}_k) = 1 - \xi(\boldsymbol{x}_k)\ ,\ k \geq 2 \tag{6}$$

Typicality and Eccentricity concepts are depicted in Fig. 2 [18]. The data point "A" is more distant from the data set than the data point "B", therefore "A" has higher eccentricity and lower typicality than "B".

The normalized eccentricity $\zeta(\boldsymbol{x}_k)$ and typicality $t(\boldsymbol{x}_k)$ can be obtained as follows:

$$\zeta(\boldsymbol{x}_k) = \frac{\xi(\boldsymbol{x}_k)}{2} \tag{7}$$

$$t(\boldsymbol{x}_k) = \frac{\tau(\boldsymbol{x}_k)}{k-2} \tag{8}$$

The normalized eccentricity $\zeta(\boldsymbol{x}_k)$ is used to define a threshold based on the well known Chebyshev inequality for outlier detection [22]. The condition expressed by (9) defines if the actual sample $\boldsymbol{x}_k$ is "$m\sigma$" away from the mean, where m is a constant value that defines how many standard deviations distant from the mean a data sample should be in order to be considered an outlier. In the most of cases, the value of m is defined as 3 [16–18,23].

$$\zeta_k(\boldsymbol{x}) > \frac{m^2+1}{2k},\ m > 0 \tag{9}$$

Despite its robustness to anomaly detection, TEDA needs to be adapted for clustering not ordered data. In this paper we propose a new clustering algorithm based on TEDA micro-clusters created using dynamically changed constraints on the TEDA variance that is detailed in the next section.

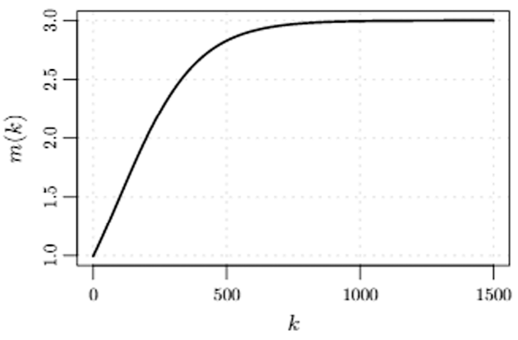


Fig. 3. The empirical function $m(k)$.

4. Proposed approach

Similarly to [15], the proposed approach is divided into two steps. The first step is the micro-cluster update. In this step we create and update micro-clusters based on a constrained version of TEDA-Cloud algorithm [18]. The second step is the macro-cluster update, in which the connected micro-clusters are joined into groups. Finally, the density of each macro-cluster is estimated and a new data sample is assigned to the macro-cluster for which it has highest membership degree.

4.1. Outlier threshold definition

The eccentricity $\xi(\boldsymbol{x}_k)$ is a statistical measure of the dissimilarity between data samples, thus it is more reliable as more data (greater k) is used on its calculation. For example, when we have a few number of data samples, it is difficult to say accurately that $\boldsymbol{x}_k$ is $m\sigma$ away from the mean because we do not have a good estimate of the mean and standard deviation, hence it is better a small value of m and consequently a more restrictive outlier threshold. On the other hand, as the number of data samples used to calculate TEDA increases, one can be more confident to say that $\boldsymbol{x}_k$ is $m\sigma$ away from the mean and the value of m can be larger. In this sense, a TEDA micro-cluster is a slightly modified version of the data clouds, presented in [18], that have its variance constrained in order to prevent a cluster to grow overly and encompass all the data from disjoint clusters. This constraint is applied by dynamically changing the parameter m for the outlier threshold, in the right side of (9), according to an empirical function $m(k)$ defined by (10).

$$m(k) = \frac{3}{1+e^{-0.007(k-100)}} \tag{10}$$

Thus, instead of having a constant value of m we have m as a function of k. The characteristics of the empirical function $m(k)$ are stated below:

1. $m(k)$ starts from 1, because one can easily see from (7) that for $k = 2$ the normalized eccentricity $\zeta(\boldsymbol{x}_2)$ will always be equal to 0.5, in this way the value of m should be at least equal to 1, otherwise every data point will be a cluster. Then, as more data is acquired, the value of m smoothly grows up to 3, where it saturates.
2. $m(k)$ saturates at 3 when $k \approx 1000$. In (9) the value of the threshold starts at different positions depending on the value of m. However, when k becomes high, the threshold for all the values of m is almost a constant. Therefore, we define $k \approx 1000$ as the value in which the empirical function $m(k)$ saturates close to 3.

The parameters of $m(k)$ were designed to obey the above characteristics. Fig. 3 depicts how $m(k)$ behaves for different values of k. Although Fig. 3 depicts a continuous function to facilitate the visualization, in practice $m(k)$ is discrete because k represents discrete timestamps.

4.2. Micro-clusters

A micro-cluster is a granular data structure with an automatically defined outlier parameter m by the function $m(k)$. Let $\mathfrak{m}_i$, $i = 1 \dots n$ be the set of micro-clusters. A prototype of $\mathfrak{m}_i$ is defined by the following parameters, updated every time a new data point is sampled:

- S_k^i: number of data samples;
- $\boldsymbol{\mu}_k^i$: center;
- $(\sigma_k^i)^2$: variance;
- $\xi^i(\boldsymbol{x}_k)$, $\zeta^i(\boldsymbol{x}_k)$: eccentricity and normalized eccentricity;
- $\tau^i(\boldsymbol{x}_k)$, $t^i(\boldsymbol{x}_k)$: typicality and normalized typicality;
- $D_k^i = \frac{1}{\zeta^i(\boldsymbol{x}_k)}$: density;
- $m_k^i(S_k^i)$: outlier threshold parameter;

When the first data sample $\boldsymbol{x}_1$ arrives, the first micro-cluster $\mathfrak{m}_1$ is created with the following parameters:

$$n = 1 \ , \ S_1^1 = 1 \ , \ \boldsymbol{\mu}_1^1 = \boldsymbol{x}_1 \ , \ (\sigma_1^1)^2 = 0 \tag{11}$$

where n is the number of micro-clusters. Note that just a few parameters are calculated when $S_k^i = 1$ because typicality and eccentricity can be only calculated with 2 or more data samples.

When a new data point $\boldsymbol{x}_k$ arrives at timestamp $k > 1$, the algorithm calculates the typicality and eccentricity of $\boldsymbol{x}_k$ to all existing micro-clusters by Eq. (6) and Eq. (3), respectively and check if $\boldsymbol{x}_k$ is an outlier or not :

$$\zeta^i(\boldsymbol{x}_k) > \frac{m_k^i(S_k^i)^2 + 1}{2S_k^i} \tag{12}$$

$$m_k^i(S_k^i) = \frac{3}{1 + e^{-0.007(S_k^i - 100)}} \tag{13}$$

For the calculation of TEDA, at least 2 data samples are needed, so using the condition expressed by (9) with $m \geq 1$ the second data point of the micro-cluster $\mathfrak{m}_i$ will never be considered an outlier even if it is far away from the first data point of $\mathfrak{m}_i$. This property is undesirable when the first two data points are far away from each other and may result in very big micro-clusters that do not model dense regions properly in the data space. Thus, we added a parameter r_0 to limit the variance of each micro-cluster when $S_k^i = 2$ in order to prevent a cluster to grow indefinitely. This parameter modifies the outlier condition expressed by the Eq. (9) just when $k = 2$:

$$\left[\zeta_2^i(\boldsymbol{x}_2) > \frac{(m^i(2))^2 + 1}{4}\right] \textbf{ AND } \left[(\sigma_2^i)^2 < r_0\right] \tag{14}$$

For all the experiments in this work, the parameter r_0 was set to **0.001**. Next, one of two conditions can take place:

Condition 1. $\boldsymbol{x}_k$ is not an outlier for at least one micro-cluster **then** update all the micro-clusters for which this condition holds.

$$\begin{aligned} S_k^i &= S_{k-1}^i + 1 \\ \boldsymbol{\mu}_k^i &= \frac{S_k^i - 1}{S_k^i} \boldsymbol{\mu}_{S_k^i - 1} + \frac{\boldsymbol{x}_k}{S_k^i} \\ (\sigma_k^i)^2 &= \frac{S_k^i - 1}{S_k^i} (\sigma_{k-1}^i)^2 + \frac{1}{S_k^i - 1} \left(\frac{2\|\boldsymbol{x}_k - \boldsymbol{\mu}_k^i\|}{d} \right)^2 \\ \xi^i(\boldsymbol{x}_k) &= \frac{1}{S_k^i} + \frac{2(\boldsymbol{\mu}_k^i - \boldsymbol{x}_k)^T(\boldsymbol{\mu}_k^i - \boldsymbol{x}_k)}{S_k^i(\sigma_k^i)^2 d} \end{aligned} \tag{15}$$

where d is the dimensionality of the data set.

Condition 2. $\boldsymbol{x}_k$ is an outlier for all existing micro-clusters **then** create a new micro-cluster.

$$n = n + 1; \ S_k^n = 1; \ \boldsymbol{\mu}_k^n = \boldsymbol{x}_k; \ (\sigma_k^n)^2 = 0 \tag{16}$$

The micro-cluster update procedure is detailed in Algorithm 1

Algorithm 1: Micro-cluster update

```
Input: $\boldsymbol{x}_k$, $r_0$
Output: $\mathfrak{m}_i$, $i = 1, 2, \dots, n$
begin
    while new samples available do
        if $k == 1$ then
            Set $\mathfrak{m}_1$ parameters as defined in Eq. (11);
        else
            flag ← true;
            for $i = 1 : n$ do
                $\mathfrak{m} \leftarrow \mathfrak{m}_i$;
                if $S_k^i == 2$ then
                    outlier ← condition of Eq. (14);
                else
                    outlier ← condition of Eq. (12);
                end
                if outlier == false then
                    Update $\mathfrak{m}_i$ according to Eq. (15);
                    flag ← false;
                end
            end
            if flag == true then
                Create a new micro-cluster with the parameters of
                Eq. (16);
            end
        end
    end
end
```

The micro-clusters for the data set of Fig. 4(a) are illustrated in Fig. 4(b).

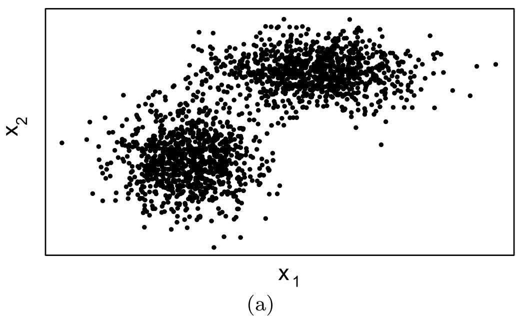
4.3. Macro-clusters

With the micro-clusters for the timestamp k, the macro-cluster update algorithm finds groups of micro-clusters that intersect each other. This procedure is similar to the way CEDAS [15] algorithm finds macro-clusters, by updating an intersection graph and grouping the disjoint connected micro-clusters. The intersection graph is created from the adjacency matrix whose dimensions are equal to the number of micro-clusters. Each element in the matrix is set as 1 if two micro-clusters intersects and 0 otherwise. The condition to verify if two micro-clusters intersect is shown by (17).

$$dist(\boldsymbol{\mu}_k^i, \boldsymbol{\mu}_k^j) < 2(\sigma_k^i + \sigma_k^j) \ , \ \forall i \neq j \tag{17}$$

Using the above condition for an overlapping data set such as in Fig. 4, it is clear that all the micro-clusters can be connected creating only one big macro-cluster. In order to avoid this issue, we apply a filter to activate or not the micro-clusters based on their density. Let $\mathfrak{M}_j = \{\mathfrak{m}_1^j, \mathfrak{m}_2^j, \dots, \mathfrak{m}_l^j\}$, $j = 1, 2, \dots, N$ be the jth macro-cluster composed by a set of connected micro-clusters $\mathfrak{m}^j$. The set of active micro-clusters of the macro-cluster $\mathfrak{M}_j$ are the ones for which the density D_k^l is greater or equal to the average density calculated over all the micro-clusters that belong to $\mathfrak{M}_j$:

$$active(\mathfrak{m}_l^j) = D_k^l \geq mean(D_k^l), \ l = 1, \dots, |\mathfrak{M}_j| \tag{18}$$



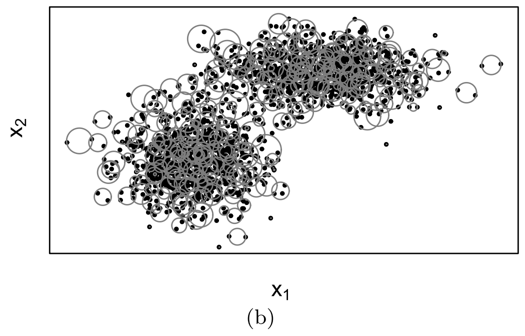


Fig. 4. (a) Data set. (b) Micro-clusters.

The effect of applying this filter is that micro-clusters at low density regions will be inactivated while the ones at high density regions will be active. In other words, we assume that the low density regions indicate separation between overlapping macro-clusters, so the filter will remove from macro-clusters calculations the micro-clusters that do not represent the core pattern at timestamp k.

Finally, the density estimate of each macro-cluster is calculated as a sum of the normalized typicalities of its active micro-clusters, weighted by their normalized density such as a density mixture model, as discussed in [19]:

$$\mathcal{T}_j(\boldsymbol{x}_k) = \sum_{l \in \mathfrak{M}_j} w_k^l t_k^l(\boldsymbol{x}_k) \tag{19}$$

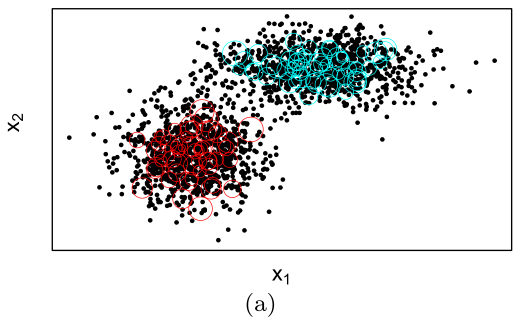
$$w_k^l = \frac{D_k^l}{\sum_{l \in \mathfrak{M}_j} D_k^l} \tag{20}$$

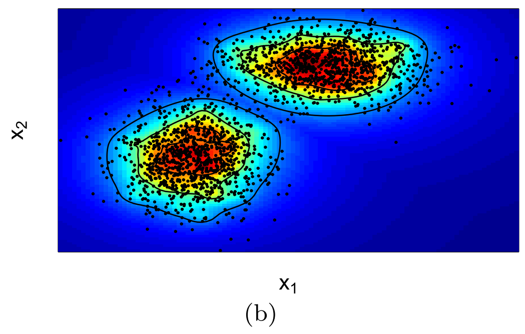
A new data point $\boldsymbol{x}_k$ is assigned to the macro-cluster for which it has the highest mixture of typicalities score $\mathcal{T}_j(\boldsymbol{x}_k)$. The detailed procedure to calculate macro-clusters is presented in Algorithm 2.

Algorithm 2: Macro-clusters update

```
Input: $\boldsymbol{x}_k$, $\mathfrak{m}$
Output: membership degree of $\boldsymbol{x}_k$ for each macro-cluster
begin
    while new samples available do
        $\mathfrak{M} \leftarrow$ Group the micro-clusters that intersect each other
        according the condition expressed in Eq. (17);
        Find the set of active micro-clusters of each
        macro-cluster $\mathfrak{M}_j$, $j = 1, 2, \dots, N$ according Eq. (18);
        Calculate $\mathcal{T}_j(\boldsymbol{x}_k)$ for $j = 1, 2, \dots N$ according Eq. (19);
        Assign $\boldsymbol{x}_k$ to the cluster $j$ it has highest $\mathcal{T}_j(\boldsymbol{x}_k)$;
    end
end
```

An example of how the macro-cluster definition works is illustrated in Fig. 5. Fig. 5(a) depicts the micro-clusters arranged





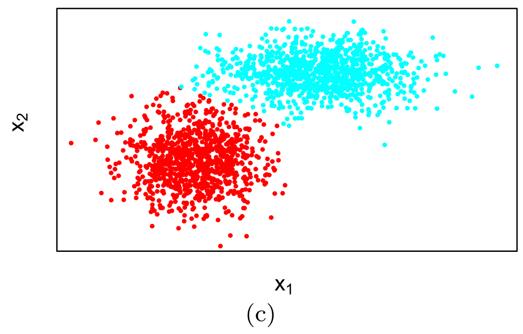


Fig. 5. (a) Macro-clusters. (b) The mixture of typicalities. (c) Cluster assignment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in two different macro-clusters (blue and red). In Fig. 5(b), the density regions are highlighted according to the mixture of typicalities. Fig. 5(c) shows the final cluster structures identified by the algorithm in the data set.

Therefore, at each new data point $\boldsymbol{x}_k$ MicroTEDAclus updates the micro-clusters, recalculates the macro-clusters and returns the cluster for which $\boldsymbol{x}_k$ is more compatible with respect to the mixture of typicalities $\mathcal{T}_j(\boldsymbol{x}_k)$.

5. Methodology and experiments

The experiments in this work were conducted with the objective of evaluating the performance of MicroTEDAclus in a context of data streams where different concept drifts exist. For all the experiments we considered online setup, in which each data point or sliding window is received, processed and discarded. The evaluating metrics were calculated after the whole data set was inserted to the algorithm sample by sample. The following subsections provide more details on data sets, comparison algorithms and evaluation metrics applied in this work.

In order to contribute to the replication of all the results in the paper, we provide full results and all source codes for this article

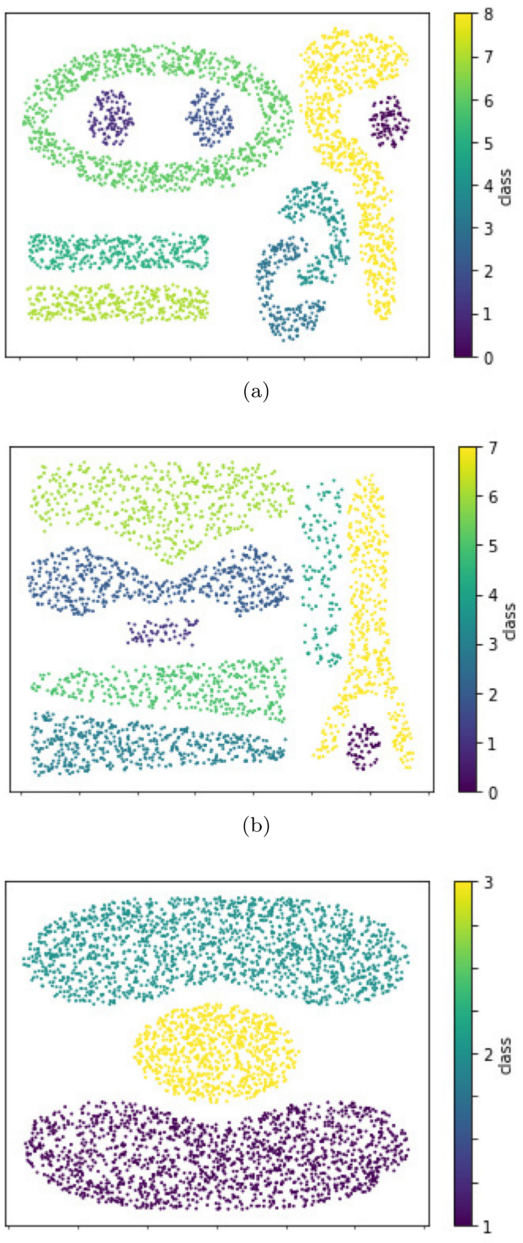


Fig. 6. Static data sets. (a) Static data set 1. (b) Static data set 2. (c) Cassini.

in Github and in the MINDS Laboratory website:
https://github.com/cseveriano/evolving_clustering
http://www.minds.eng.ufmg.br

5.1. Data sets

5.1.1. Static data sets

Before adapting to concept drift, a data stream clustering algorithm should be able to cluster static structures, or its adaptation abilities would not be much relevant. Therefore, three static data sets with difficult static clustering problems were chosen. The static data sets 1 and 2 were introduced in [24], applied for the analysis of Chameleon clustering algorithm. In this work, the data sets are denoted as **Static Data set 1 (ST-D1)** and **Static Data set 2 (ST-D2)**, respectively. **ST-D1**, depicted in Fig. 6(a), contains 3031 samples organized in 9 clusters.[1] **ST-D2**, in Fig. 6(b), contains 2551 samples in 8 clusters.[2]

The third static data set, illustrated in 6(c), is **Cassini (CSN)**, a well known clustering problem with three clusters of uniform density, where two clusters are non-convex shapes bending around a circular cluster in between them. It contains 4000 samples. This dataset was provided by the R tool **stream**, developed by [25].

5.1.2. Concept drift data sets

Three synthetic data sets were designed to simulate concept drifts. The R tool **stream**, developed by [25], was used to create all the data sets. Each data set simulates a different concept drift, as follows:

1. **Stream Benchmark 1 (STR-B1):** benchmark available in R package stream. It consists of 2 clusters moving in a two-dimensional space. One moves from top left to bottom right and the other one moves from bottom left to top right. Both clusters overlap when they meet exactly in the center of the data space.
2. **Stream Benchmark 2 (STR-B2):** benchmark stream 2, available in R tool stream. It consists of 2 static clusters and a third one moving from the first to the second.
3. **Random RBF Generator with Events (RBF):** the generator starts with a fixed number of centroids representing each cluster, which has a random position, a single standard deviation, class label and weight. During the sampling, the cluster attributes are changed, and events such as cluster splitting/merging and deletion/creation are set.

Examples of the initial and final states of **STR-B1** and **STR-B2** data sets are shown in Figs. 7 and 8, respectively. Fig. 9 depicts an output of **RBF**. Fig. 9(a) shows the initial state with 3 clusters. In Fig. 9(b) an event of cluster creation occurs. In Fig. 9(c) a cluster is split. Fig. 9(d) indicates an event of cluster deletion.

5.2. Evaluation

5.2.1. Prequential method

For each data set and clustering algorithm, the Prequential method is applied [26]. In this method, each instance can be used to test before it is used for training. Such mechanism is appropriate for data stream evaluation, where new, unknown data arrives over time. It can also improve the analysis of the clustering algorithms by providing temporal plots of accuracy. Effects on data, such as concept drift may be better identified.

The prequential error P at time t is the accumulated sum of a loss function L between the prediction and observed values:

$$P(t) = \frac{1}{t}\sum_{i=1}^{t} L(\hat{y}_i, y_i) \qquad (21)$$

The experiments in this work define, for each evaluated method, an initial training set and subsequent sliding windows. The initial training set has the same number of samples of a prequential sliding window. The prequential error for each sliding window w is calculated as follows:

$$P_w(t) = \frac{1}{w}\sum_{i=t-w+1}^{t} L(\hat{y}_i, y_i) \qquad (22)$$

[1] https://data.world/cseveriano/clustering-with-concept-drift/workspace/file?filename=chameleon-ds3-clean.csv.

[2] https://data.world/cseveriano/clustering-with-concept-drift/workspace/file?filename=chameleon-ds4-clean.csv.

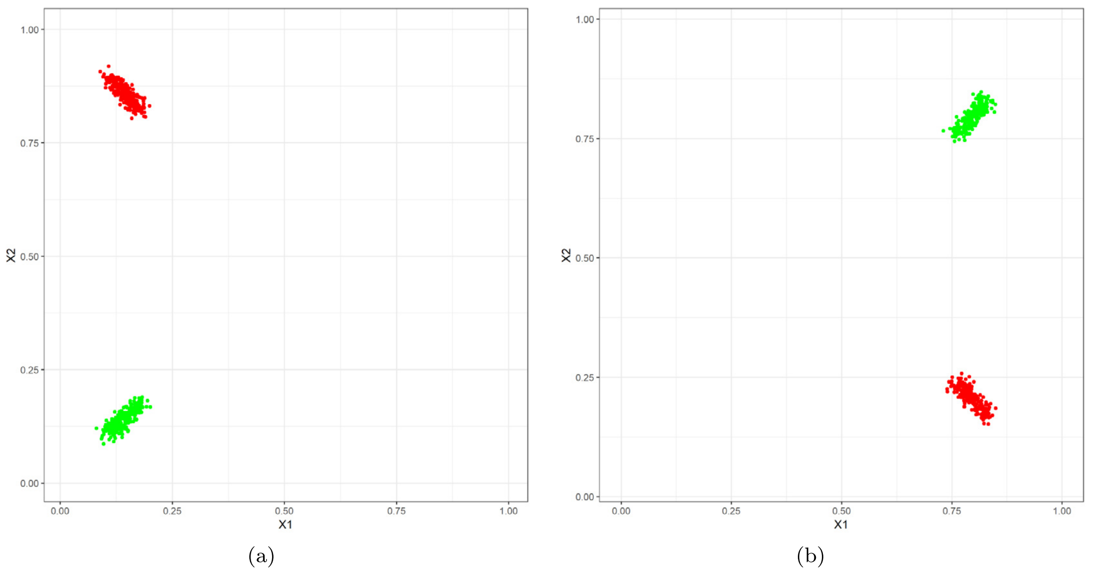


Fig. 7. Stream Benchmark 1. (a) Before concept drift. (b) After concept drift.

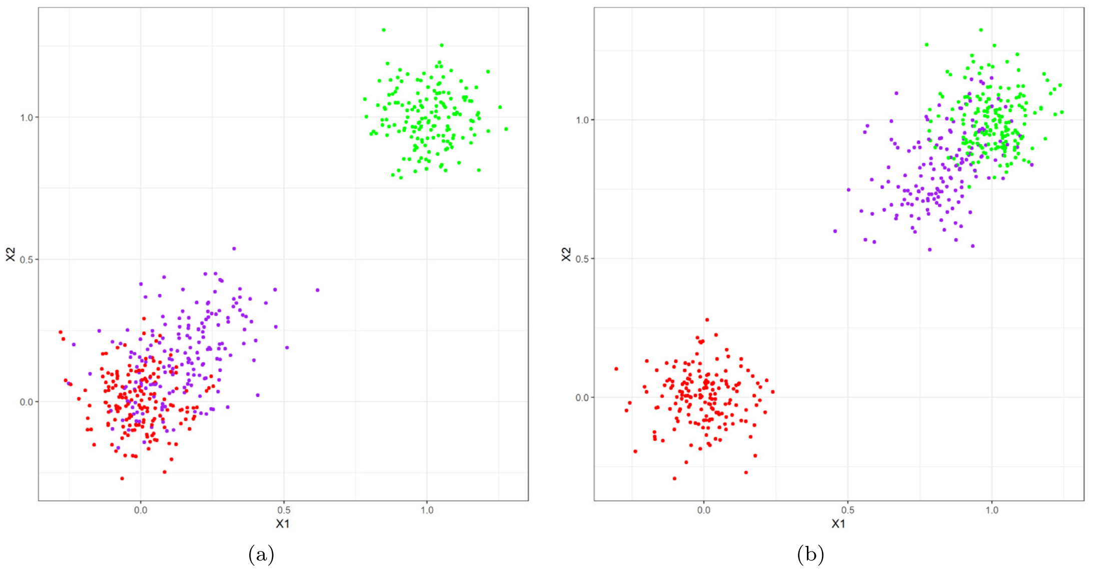


Fig. 8. Stream Benchmark 2. (a) Before concept drift. (b) After concept drift.

Table 1
Experiment settings for the data sets.

Data set	ST-D1	ST-D2	CSN	STR-B1	STR-B2	RBF
Samples	3031	2551	4000	4000	4000	10 000[a]
Clusters	9	8	3	2	3	1-9[b]
Window Size	–	–	–	100	100	100
Windows	–	–	–	40	40	100

[a]After the generation of 1000 samples, one event occurs, which results in 10 events during each RBF experiment.
[b]RBF experiments start with 3 clusters and according to the events of concept evolution, can change the number of clusters. The range is from 1 to 9 clusters.

The settings for the experiments using each data set are described in Table 1. The prequential evaluation was applied only to concept drift data sets. Static data sets were evaluated using all the samples without any window separation and for each run the data was randomly sampled. For each combination of method and data set, the experiment was performed 30 times.

5.2.2. *Adjusted Rand Index*

In this work, Adjusted Rand Index is used as loss function. The Rand index is a measure of the similarity between two data clusterings. This metric was chosen as a loss function to treat the problem of comparison between predicted and observed labels in a more transparent way, since the clustering algorithms evaluated in this work presented different ways of assigning a label index to each detected cluster.

Given a set S of n samples $S = \{o_1, \dots, o_n\}$ and two data clusterings of S to compare, $X = \{X_1, \dots, X_r\}$, a partition of S into r clusters, and $Y = \{Y_1, \dots, Y_s\}$, a partition of S into s clusters, and the following variables:

- a, the number of pairs of elements in S that are in the same cluster in X and in the same cluster in Y
- b, the number of pairs of elements in S that are in different clusters in X and in different clusters in Y

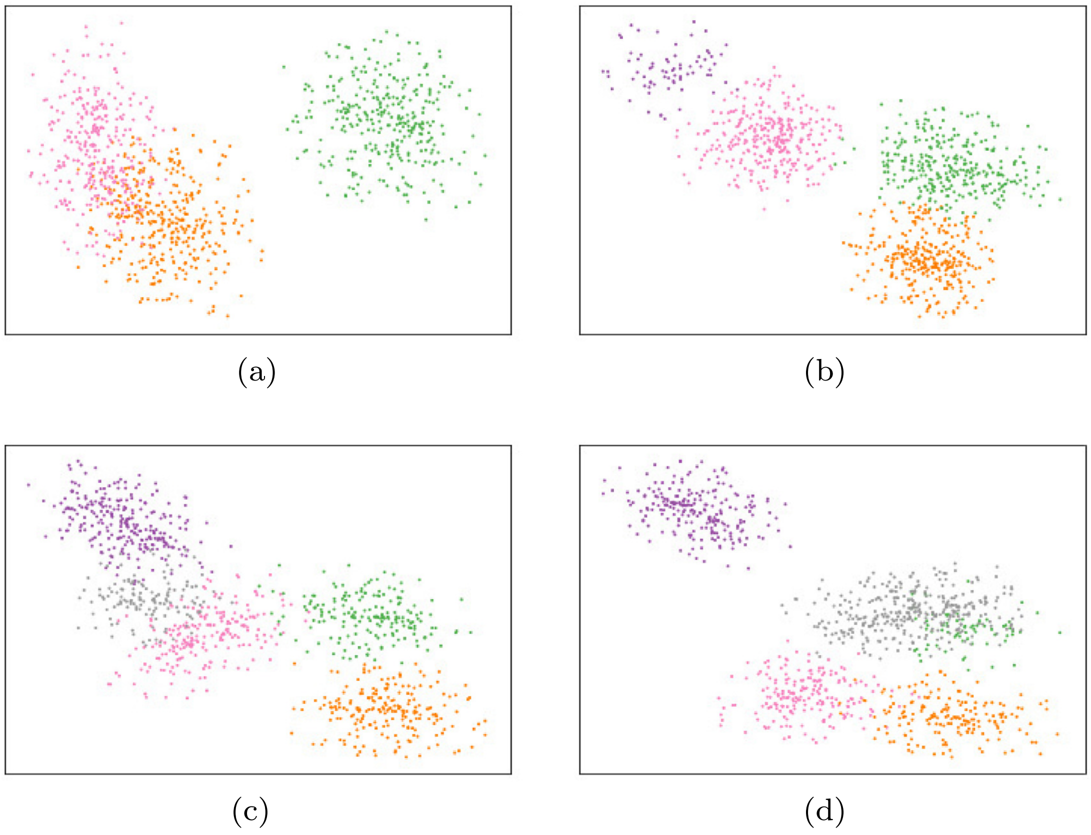


Fig. 9. Random RBF Generator. (a) Initial state. (b) After first event (cluster creation). (c) After second event (cluster split). (d) After third event (cluster deletion).

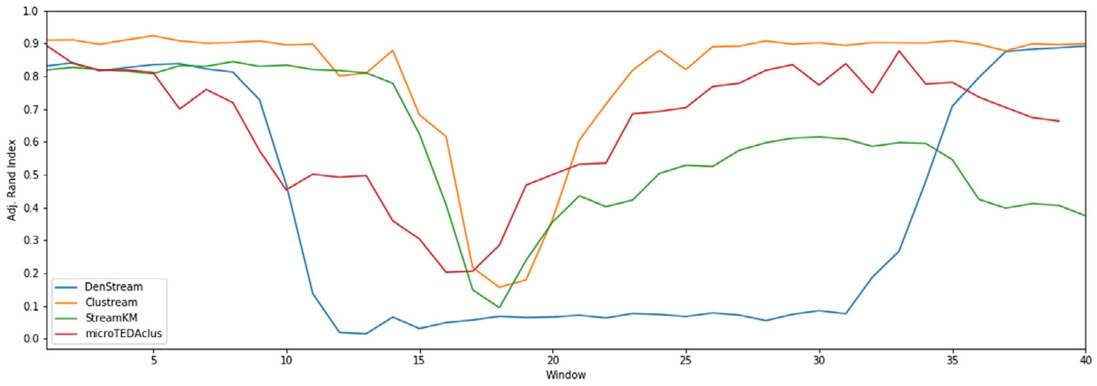


Fig. 10. Prequential evaluation — STR-B1.

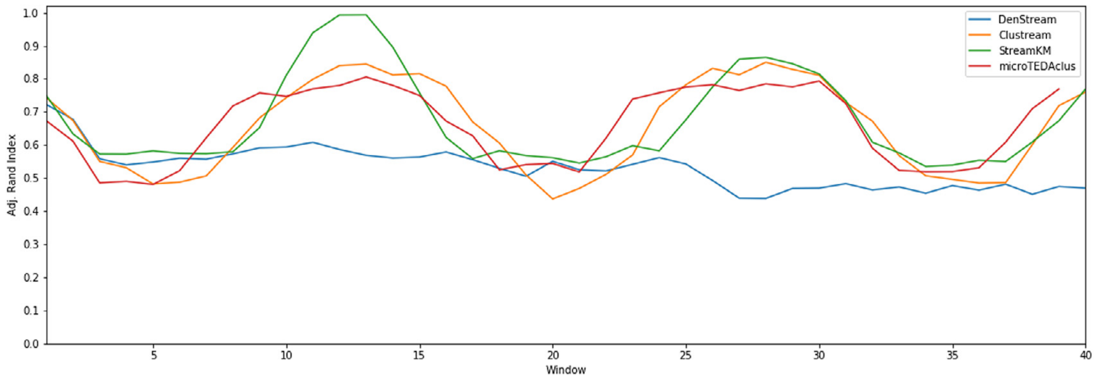


Fig. 11. Prequential evaluation — STR-B2.

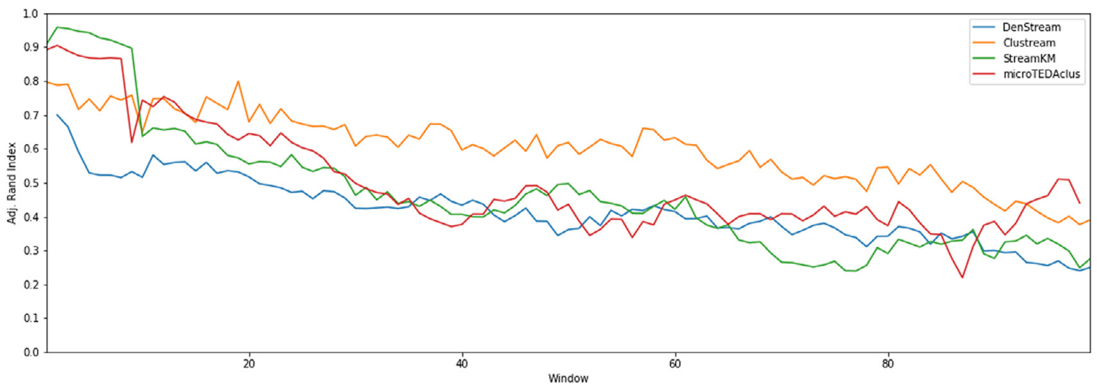


Fig. 12. Prequential evaluation — RBF.

Table 2
Contingency table.

	Y_1	Y_2	…	Y_s	Sums
X_1	n_{11}	n_{12}	…	n_{1s}	a_1
X_2	n_{21}	n_{22}	…	n_{2s}	a_2
⋮	⋮	⋮	⋮	⋮	⋮
X_r	n_{r1}	n_{r2}	…	n_{rs}	a_r
Sums	b_1	b_2	…	b_s	

The Rand index R can be calculated as:

$$R = \frac{a+b}{\binom{n}{2}} \tag{23}$$

The Adjusted Rand index is the adjusted-for-chance version of the Rand index. Given the contingency table (see Table 2).

The Adjusted Rand Index ARI is:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{24}$$

It ensures the score to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the partitions are identical.

5.3. Algorithms

The proposed MicroTEDAclus algorithm is compared with the following data stream clustering algorithms.

1. **Denstream** [14]
2. **CluStream** [12]
3. **StreamKM++** [27]

This work used the implementation of these algorithms available in R package **streamMOA** [28]. The set of hyperparameters of each method was adjusted by a grid search tuning where we chose the best set of parameters for which the algorithm finds the correct number of clusters (or the closest to the correct) for each data set. For each data set, the samples denoted as "Initial training" in Table 1 were applied to parameter tuning. Table 3 lists the range of parameters where the grid search occurred for all the methods.

6. Experimental results

6.1. Static and concept drift data sets

The data sets **STR-B1** and **STR-B1**, whose average of prequential evaluations are shown in Figs. 10 and 11 respectively, denote situations where drifts occur at all the time steps. Consequently, the clustering methods tend to run in a more unstable context. The prequential evaluation for data set **RBF**, illustrated in Fig. 12, also presents situations where concept drifts occur at all time steps, including events such as cluster creation, deletion, merge or split which are set to happen after each 1000 samples (or 10 prequential windows). It is possible to notice that the performance of all the clustering algorithms decrease during the sequence of events. However, MicroTEDAclus tend to adapt to such events over time. The prequential evaluation suggests that the algorithms can adapt to different concept drift events.

Table 4 depicts the statistics of ARI values for each clustering method and data set. In terms of accuracy, Clustream, StreamKM++ and MicroTEDAclus presented comparable results, with small differences between them depending on the experiment and the data set.

In general, the proposed algorithm is more suitable for clustering data stream from arbitrary shapes where incremental drifts with low magnitude are included.

Table 3
Range of parameters for grid search.

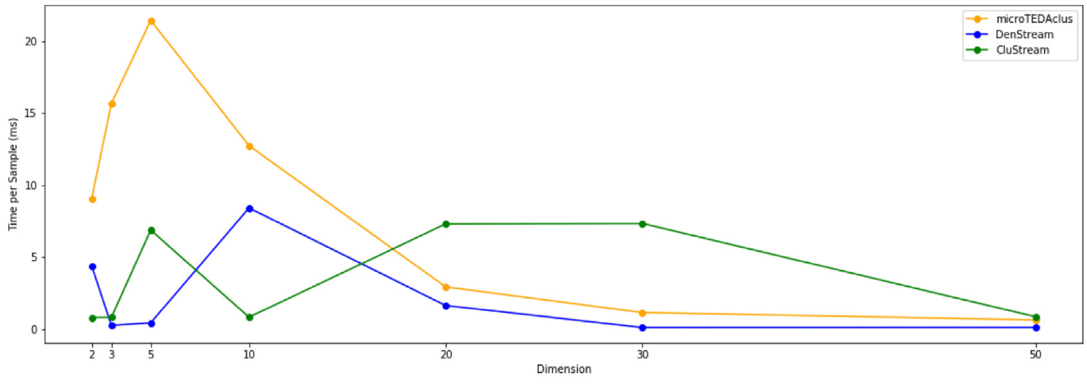
Method	Parameter	Description	Range
DenStream	ϵ	Micro-cluster radius	$\{0.001, \dots, 0.01\}$
	μ	Weight limit	$\{1, \dots, 20\}$
	β	Outlier threshold	$\{0.2, 0.4\}$
CluStream	m	Micro-clusters	$\{10, \dots, 1000\}$
	h	Time window	$\{100, \dots, 1000\}$
	t	Boundary factor	$\{1, \dots, 10\}$
StreamKM++	s	Size of Coreset	$\{10, \dots, 1000\}$
	k	Number of clusters	$\{1, \dots, 10\}$

Table 4
Adjusted rand index for clustering methods.

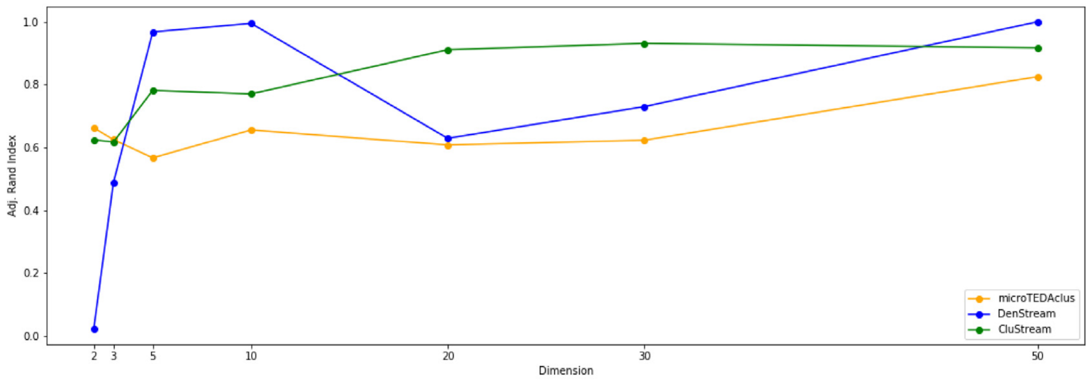
Data set	DenStream	CluStream	StreamKM	MicroTEDAclus
ST-D1	0.21 ± 0.10	0.55 ± 0.03	$\mathbf{0.62 \pm 0.05}$	0.38 ± 0.07
ST-D2	0.22 ± 0.13	0.36 ± 0.07	0.37 ± 0.03	$\mathbf{0.41 \pm 0.05}$
CSN	0.39 ± 0.09	0.40 ± 0.11	$\mathbf{0.48 \pm 0.18}$	0.42 ± 0.16
STR-B1	0.61 ± 0.31	0.57 ± 0.33	0.66 ± 0.27	$\mathbf{0.68 \pm 0.21}$
STR-B2	0.65 ± 0.13	0.65 ± 0.13	0.55 ± 0.04	$\mathbf{0.69 \pm 0.12}$
RBF	0.42 ± 0.10	$\mathbf{0.61 \pm 0.06}$	0.46 ± 0.03	0.50 ± 0.07

6.2. Scalability experiments

The performance of the algorithm when confronted to data sets of larger dimensions was analyzed. Such performance was assessed in terms of processing speed, memory consumption and accuracy. During each experiment, the average time spent to process a sample was used to evaluate processing speed.



(a)



(b)

Fig. 13. Scalability experiments. (a) Average processing time per sample in milliseconds. (b) Adjusted Rand Index.

Memory consumption was monitored in terms of the amount of memory in megabytes allocated at the end of the clustering process. These values are compared with the clustering accuracy, defined in the experiments by the average value of Adjusted Rand Index obtained throughout this process. The experiments were performed on an Intel Xeon Processor (2.30 GHz) and 12 GB RAM. CluStream [12] and DenStream [14] were chosen for the comparison as both models also work with the concept of micro-clusters for clustering. The benchmark algorithms used in this section, as well as microTEDAclus, were implemented in Python programming language.

6.2.1. *Processing speed evaluation*

Datasets in d-dimensional space were used, where d ranges from 2 to 50. Each dataset is composed of three clusters generated from Gaussian distributions. 4000 data points were sampled for each clustering. Each experiment with dimension d was run ten times and the mean values were reported.

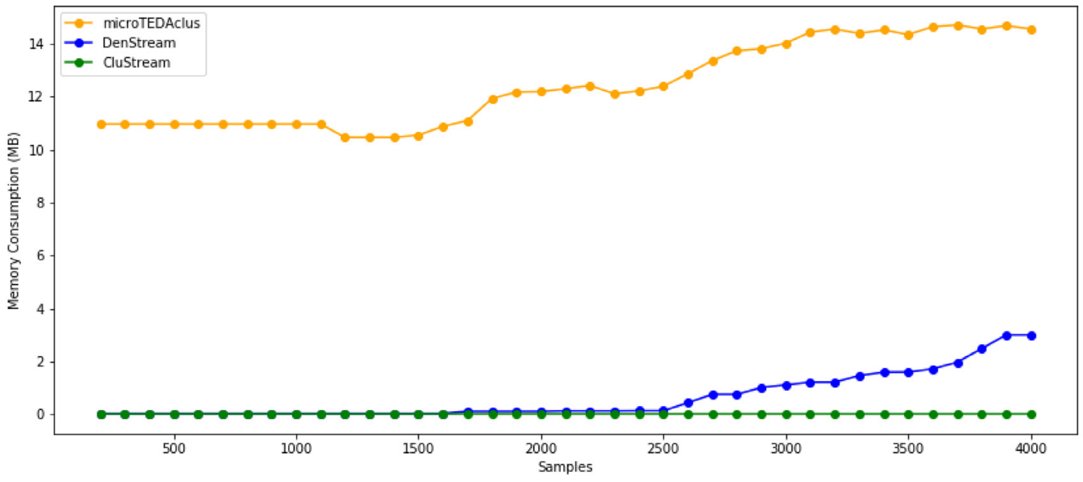
The experiments used the prequential evaluation described in Section 5.2.1 with window size of 100. The first prequential window of each dataset was used for parameter tuning. Fig. 13(a) compares the average processing time per data point in milliseconds of CluStream, Denstream and MicroTEDAClus. Fig. 13(b) shows the accuracy of each algorithm throughout the experiments.

The results point to lower mean values of processing time for DenStream and CluStream compared to microTEDAclus, being this difference more significant in smaller dimensionality datasets (from 2 to 10 dimensions). However, with increasing dimensionality, microTEDAClus can reduce its average values while still maintaining good accuracy results. These results suggest good robustness of the algorithm against larger dimensional databases. Such robustness was also observed during the hyperparameter tuning process. While DenStream and CluStream required a grid search to find the appropriate parameters, the variance limit hyperparameter for microTEDAclus was fixed at 0.001.

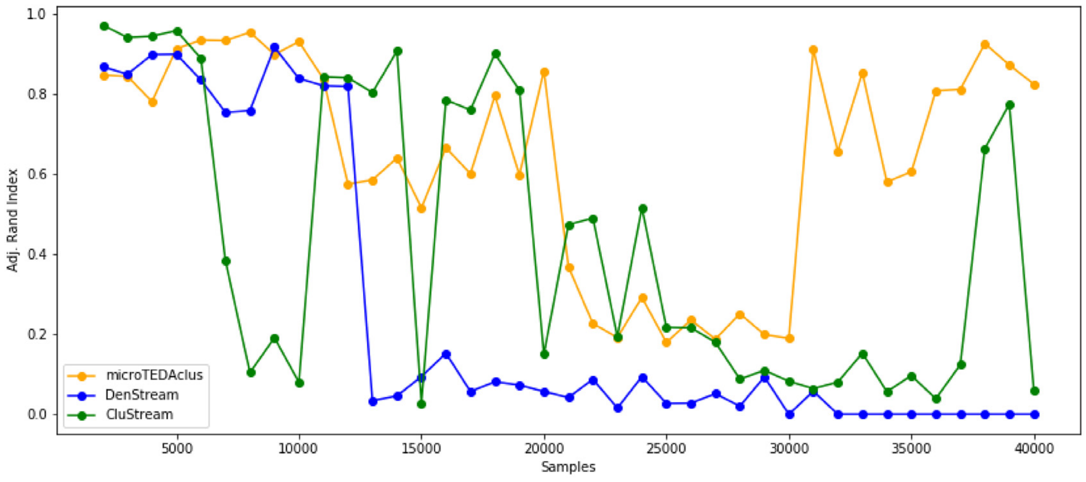
6.2.2. *Memory consumption evaluation*

For the memory consumption experiments the data set **Stream Benchmark 1 (STR-B1)**, described in Section 5.1.2 was monitored. The prequential method was applied to process 4000 samples with window size of 100 samples. Fig. 14(a) depicts the evolution of memory consumption, in megabytes, for each model during data set processing. The recorded values represent the difference between the amount of memory allocated before the start of the algorithms execution and during the processing of each window. Fig. 14(b) represents the Adjusted Rand Index values observed during the experiments.

For all evaluated algorithms, memory consumption is directly related to the creation of micro-clusters. In CluStream, this value remains stable because it works with a fixed number of micro-clusters, defined from the configuration of their hyperparameters. MicroTEDAclus and DenStream have variable values of micro-clusters, which grow throughout the experiment due to the attempt of the algorithms to adapt to the concept drift. MicroTEDA-clus presents a proportionally higher memory consumption than DenStream and CluStream due to its internal data structures for representing macro-clusters and micro-clusters. However, this consumption after an increase to fit the concept drift remains



(a)



(b)

Fig. 14. Memory consumption experiments. (a) Memory in megabytes. (b) Adjusted Rand Index.

stable at the end of the experiment. In addition, microTEDAclus presents a more effective ability to adapt to the drift, as shown in Fig. 14(b). While microTEDAclus was able to return to pre-drift accuracy levels, the other algorithms had their performance reduced when trying to recover from data variations.

7. Computational complexity

Time complexity of the algorithm can be analyzed considering all the clustering steps separately. Micro cluster update step, described in Algorithm 1 is dimensionally dependent during the calculation of variance and eccentricity. Both calculations have complexity of $O(d)$, where d is the dimensionality. Therefore, given that n is the number of samples to be clustered and k is the number of micro clusters, time complexity for micro-cluster update step can be defined as $O(dnk)$. Macro-cluster update, described in Algorithm 2, is performed by calculating Euclidean distance between changed (updated or created) micro-clusters. Given k_{ch} the number of changed micro-clusters, where $k_{ch} \leqslant k$, it is necessary to calculate the Euclidean distance between them, pairwise. The distance calculation is linear in d, or has a complexity of $O(d)$. The total combinations between k_{ch} micro-clusters is $O(k_{ch}$ choose 2), which results in a complexity of $O(k_{ch}^2)$. Macro-cluster update also includes the process of finding the set of active micro-clusters, which is linear in k_{ch}, or $O(k_{ch})$. Therefore, macro-cluster update has complexity of $O(k_{ch}^2 d + k_{ch})$. This value may be limited by the interval between runs of the update process. That is, the shorter the interval between macro-cluster updates, the smaller the k_{ch} and, consequently, the less computational effort associated with the process. Finally, the cluster assignment step depends linearly on the active set of micro-clusters k_{act}, where $k_{act} < k$, which results in a time complexity of $O(nk_{act})$. Space and time complexity are summarized in Table 5.

8. Conclusion and discussions

In this paper we propose a new evolving algorithm called MicroTEDAclus for clustering data streams from arbitrary distributions. The comparison with state of the art algorithms on benchmark problems shows that MicroTEDAclus has competitive performance for online clustering of data streams with arbitrary

Table 5
Clustering complexity for the proposed algorithm.

Time complexity	
Micro-cluster update	$O(dnk)$
Macro-cluster update	$O(k_{ch}^2 d + k_{ch})$
Cluster assignment	$O(nk_{act})$

shapes. However, it is worth noting that MicroTEDAclus was very robust in terms of parameters. The only variable to be tuned was the variance limit r_0, which stayed in the same value for all the experiments. On the other hand, as previously discussed, the other methods evaluated in this work require tuning a large number of free parameters. This is an important obstacle for the algorithm to work well in a streaming data context. It suggests that the evolving capacity of MicroTEDAClus is performed in an iterative and autonomous way, without the need of constant adjustments in its parameters. Such feature can be very useful in a streaming context, where changes in data distributions are not usually known *a priori*.

The proposed algorithm also indicated in the experiments an ability to work with larger data sets. In addition to not requiring storage of information from each sample processed, the algorithm presented a good ability to handle data with higher dimensionality. In addition to presenting good accuracy in high dimensional datasets, all its dimension dependent processing steps have linear complexity. Such aspects reinforces its application to streaming data problems.

As the computational complexity assessment suggests, as well as speed and memory consumption experiments, microTEDAclus performance is directly related to the number of micro-clusters created throughout its clustering process. Therefore, an important optimization aspect can be the development of mechanisms capable of reducing the number of micro-clusters required for the model operation. For instance, a consistent mechanism of identification and removal of micro-clusters formed from outliers or low density regions.

Additionally, the outlier threshold parameter, currently defined by Eq. (13), can be investigated. Different functions can be analyzed for this task. The variance limit hyperparameter r_0, although robust in most experiments, presented some performance variations. It can also be investigated in a sense of being integrated with an autonomous adaptation mechanism.

Another aspect to be advanced in the model is its application to other tasks. Its evolving characteristics and modeling based on concepts such as typicality can be applied as useful tools for an integration with other models assigned to different problems, such as fault detection, forecasting or classification.

In order to contribute to the replication of all the results in the paper, we provide full results and all source codes for this article in Github and in the MINDS Laboratory website:

https://github.com/cseveriano/evolving_clustering
http://www.minds.eng.ufmg.br

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.

Acknowledgments

This work has been supported by the Brazilian agencies (i) National Council for Scientific and Technological Development (CNPq) and (ii) Coordination for the Improvement of Higher Education (CAPES).

References

[1] Q. Song, J. Ni, G. Wang, A fast clustering-based feature subset selection algorithm for high-dimensional data, IEEE Trans. Knowl. Data Eng. 25 (1) (2013) 1–14, http://dx.doi.org/10.1109/TKDE.2011.181.
[2] I. Czarnowski, P. Jędrzejowicz, Kernel-based fuzzy C-means clustering algorithm for RBF network initialization, in: I. Czarnowski, A.M. Caballero, R.J. Howlett, L.C. Jain (Eds.), Intelligent Decision Technologies 2016: Proceedings of the 8th KES International Conference on Intelligent Decision Technologies (KES-IDT 2016) – Part I, Springer International Publishing, Cham, 2016, pp. 337–347, http://dx.doi.org/10.1007/978-3-319-39630-9_28.
[3] J.-S.R. Jang, C.-T. Sun, Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.
[4] J. Gama, Knowledge Discovery from Data Streams, first ed., Chapman & Hall/CRC, 2010.
[5] A. Lemos, F. Gomide, W. Caminhas, Multivariable gaussian evolving fuzzy modeling system, IEEE Trans. Fuzzy Syst. 19 (1) (2011) 91–104.
[6] B.S.J. Costa, P.P. Angelov, L.A. Guedes, Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier, Neurocomputing 150 (2015) 289–303, http://dx.doi.org/10.1016/j.neucom.2014.05.086, Bioinspired and knowledge based techniques and applications The Vitality of Pattern Recognition and Image Analysis Data Stream Classification and Big Data Analytics.
[7] P. Angelov, Anomaly detection based on eccentricity analysis, in: 2014 IEEE Symposium on Evolving and Autonomous Learning Systems, EALS, 2014, pp. 1–8, http://dx.doi.org/10.1109/EALS.2014.7009497.
[8] J.A. Silva, E.R. Faria, R.C. Barros, E.R. Hruschka, A.C.d. Carvalho, J.a. Gama, Data stream clustering: A survey, ACM Comput. Surv. 46 (1) (2013) 13:1–13:31, http://dx.doi.org/10.1145/2522968.2522981.
[9] D.T. Pham, S.S. Dimov, C.D. Nguyen, An incremental k-means algorithm, Proc. Inst. Mech. Eng. C 218 (7) (2004) 783–795, http://dx.doi.org/10.1243/0954406041319509.
[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 44.
[11] M. Ackerman, S. Dasgupta, Incremental clustering: The case for extra clusters, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 27, Curran Associates, Inc., 2014, pp. 307–315.
[12] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th International Conference on Very Large Data Bases - Vol. 29, VLDB '03, VLDB Endowment, 2003, pp. 81–92.
[13] M. Hahsler, M. Bolaños, Clustering data streams based on shared density between micro-clusters, IEEE Trans. Knowl. Data Eng. 28 (6) (2016) 1449–1461, http://dx.doi.org/10.1109/TKDE.2016.2522412.
[14] F. Cao, M. Ester, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: 2006 SIAM Conference on Data Mining, 2006, pp. 328–339.
[15] R. Hyde, P. Angelov, A. MacKenzie, Fully online clustering of evolving data streams into arbitrarily shaped clusters, Inform. Sci. 382 (2017) 96–114.
[16] D. Kangin, P. Angelov, Evolving clustering, classification and regression with TEDA, in: Neural Networks (IJCNN), 2015 International Joint Conference on, IEEE, 2015, pp. 1–8.
[17] B.S.J. Costa, C.G. Bezerra, L.A. Guedes, P.P. Angelov, Unsupervised classification of data streams based on typicality and eccentricity data analytics, in: 2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, 2016, pp. 58–63, http://dx.doi.org/10.1109/FUZZ-IEEE.2016.7737668.
[18] C.G. Bezerra, B.S.J. Costa, L.A. Guedes, P.P. Angelov, A new evolving clustering algorithm for online data streams, in: Evolving and Adaptive Intelligent Systems (EAIS), 2016 IEEE Conference on, IEEE, 2016, pp. 162–168.
[19] P. Angelov, Outside the box: an alternative data analytics framework, J. Autom. Mob. Robot. Intell. Syst. 8 (2) (2014) 29–35.
[20] R.H. Moulton, H.L. Viktor, N. Japkowicz, J. Gama, Clustering in the presence of concept drift, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 339–355.
[21] G.I. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. 30 (4) (2016) 964–994.
[22] J.G. Saw, M.C.K. Yang, T.C. Mo, Chebyshev inequality with estimated mean and variance, Amer. Statist. 38 (2) (1984) 130–132.
[23] D. Kangin, P. Angelov, J.A. Iglesias, Autonomously evolving classifier TEDAClass, Inform. Sci. 366 (2016) 1–11, http://dx.doi.org/10.1016/j.ins.2016.05.012.
[24] G. Karypis, E.-H.S. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer (8) (1999) 68–75.
[25] M. Hahsler, M. Bolanos, J. Forrest, et al., Introduction to stream: An extensible framework for data stream clustering research with R, J. Stat. Softw. 76 (14) (2017) 1–50.
[26] J. Gama, R. Sebastião, P.P. Rodrigues, On evaluating stream learning algorithms, Mach. Learn. 90 (3) (2013) 317–346.

[27] M.R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, C. Sohler, StreamKM++: A clustering algorithm for data streams, J. Exp. Algorithmics 17 (2012) 2–4.

[28] M. Hahsler, J. Forrest, streamMOA: Interface for MOA stream clustering algorithms, 2019, https://CRAN.R-project.org/package=streamMOA, R package version 1.2-1.

José Maia Neto received a B.Sc. degree in Control & Automation Engineering (2015) from College of Science and Technology of Montes Claros (FACIT), a M.Sc. degree (2018) in Communications and Computer Engineering from Federal University of Minas Gerais (UFMG). Since 2018, he is working as Data Scientist in Financial Companies in Brazil, applying machine learning to a wide range of problems involving structured and non structured data. Currently, his research interest involves the application of NLP algorithms to real world problems and incremental self-learning systems.

Carlos Severiano received a B.Sc. degree in Computer Science from Universidade Federal de Minas Gerais (UFMG) in 2003 and a M.Sc. degree in Electrical Engineering from the same university in 2013. He is currently a Ph.D. student at UFMG researching on spatio-temporal forecasting techniques applied to renewable energy problems. Since 2016, he has been working with the Instituto Federal de Minas Gerais (IFMG), Brazil. His current research interests are applications of machine learning, computational intelligence and software engineering.

Frederico Gadelha Guimarães received his B.Eng. and M.Sc. degrees in electrical engineering from the Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil, in 2003 and 2004, respectively. He received a Ph.D. degree in Electrical Engineering from the Federal University of Minas Gerais in 2008, with a one-year visiting student scholarship at McGill University, Montreal, Canada (2006–2007), and a postdoctoral fellowship (2017–2018) at the Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi), linked to the Université Paris-Est Créteil (UPEC), Paris, France. In 2010, he joined the Department of Electrical Engineering, UFMG, and in 2018 he became an Associate Professor. He is responsible for the Machine Intelligence and Data Science Laboratory (MINDS) for computational intelligence research since 2014. Dr. Guimarães has published more than 200 papers in journals, congresses and chapters of national and international books. He has experience in Electrical Engineering and Computer Engineering, with emphasis on optimization, computational intelligence, machine learning, genetic algorithms and evolutionary computation. He is a Senior member of the IEEE Computational Intelligence Society (CIS) and the IEEE Systems, Man, and Cybernetics Society (SMCS).

Cristiano Leite de Castro received a B.Sc. degree in Computer Science (2001) from Federal University of Lavras (UFLA), a M.Sc. and a Ph.D. degree (2004 and 2011) in Communications and Computer Engineering from Federal University of Minas Gerais (UFMG). Between 2007 and 2013, he was Assistant professor of the Computer Science Department at UFLA. Since 2014, he is with the Department of Electrical Engineering at UFMG where he has been teaching and supervising students of the undergraduate courses in Electrical, Systems and Control & Automation Engineering. His interests involve theoretical aspects and applications of machine learning and data mining. Within these broad areas, he has been researching on the following subjects: time-series clustering and forecasting, pattern classification with unbalanced and non-stationary (concept drift) data and design of incremental machine learning algorithms.

Andre Paim Lemos received a B.Sc. degree in Computer Science from Universidade Federal de Minas Gerais (UFMG) in 2003, M.Sc. and Ph.D. degrees in Electrical Engineering from the same university in 2007 and 2011, respectively. Since 2011, he has been working as Assistant Professor in the Department of Electronic Engineering at UFMG. His current research interests are industrial applications of machine learning and computational intelligence, including fault detection and diagnosis and system modeling and identification. Since 2007, he has been working in several R&D projects funded by major Brazilian Companies, such as Petrobras, Gerdau, CEMIG, and CHESF. In 2010 he received the Meritorious Paper Award at the 29th International Conference of the North American Fuzzy Information Processing Society (NAFIPS). In 2011 he received the Best Paper Award at the 30th International Conference of NAFIPS, and in 2012 he received the Best Special Session Paper Award (Adaptive and Dynamic Modeling in Non-Stationary Environments) at the 11th International Conference on Machine Learning and Applications (ICMLA).

Juan Camilo Fonseca Galindo received a B.Sc. degree in Electronic Engineering (2014) from University Francisco de Paula Santander (UFPS), an M.Sc. degree (2016) in Communications and Computer Engineering from Federal University of Minas Gerais (UFMG). Currently, he is a Ph.D. candidate in Communications and Computer Engineering from the Federal University of Minas Gerais (UFMG) and is working as a Software Developer at Loggi, Brazil. His research interests are Trajectory Data Mining applied to Logistics, Clustering Techniques and Pattern Recognition.

Dr. Miri Weiss Cohen is a Senior lecturer in the Department of Software Engineering at the Braude College of Engineering. Miri has completed her B.Sc., M.Sc. and Ph.D. at the Technion — Israel Institute of Technology. Her research interests lie in the area of Machine Learning methodologies in use of optimization methods for engineering problems. In recent years, she has focused on two major topics: prediction and optimization of time series and green energy, and using CT scans for classification cancer stages and 3D volumes using CNN. She has collaborated actively with researchers in several other disciplines related to Machine Learning and engineering design.