An Evolving Approach to Data Streams Clustering Based on Typicality and Eccentricity Data Analytics

Clauber Gomes Bezerra[a,*], Bruno Sielly Jales Costa[b], Luiz Affonso Guedes[c], Plamen Parvanov Angelov[d]

[a] *clauber.bezerra@ifrn.edu.br*
Federal Institute of Rio Grande do Norte - IFRN, Campus EaD
Av. Senador Salgado Filho 1559, Tirol, CEP: 59015-000, Natal, RN, Brazil
[b] *bruno.costa@ifrn.edu.br*
IFRN - Campus Natal Zona Norte
Rua Brusque 2926, Potengi, CEP 59112-490, Natal, RN, Brazil
[c] *affonso@dca.ufrn.br*
Federal University of Rio Grande do Norte - UFRN, Department of Computer Engineering and Automation - DCA
Campus Universitário, Lagoa Nova, CEP: 59078-900, Natal, RN, Brazil
[d] *p.angelov@lancaster.ac.uk*
Lancaster University, Data Science Group, School of Computing and Communications, Lancaster LA1 4WA, United Kingdom

Abstract

In this paper we propose an algorithm for online clustering of data stream. This algorithm is called AutoCloud and it is based on the recently introduced concept of Typicality and Eccentricity Data Analytics, mainly used for anomaly detection tasks. AutoCloud is an evolving, online and recursive technique that does not need training or prior knowledge about the data set to be processed. Thus, AutoCloud is fully online, requiring no offline processing. It allows creation and merging of clusters in an autonomous manner as new data observations become available. The clusters created by AutoCloud are called data clouds, which are structures without pre-defined shape or boundaries. Auto-Cloud allows each data sample to belong to multiple data clouds simultaneously using fuzzy concepts. AutoCloud is also able to handle concept drift and concept evolution, which are problems that are inherent to data streams in general. Since the algorithm is recursive and online, its suitable for applications that requires real-time

*Corresponding author
Email address: clauber.bezerra@ifrn.edu.br (Clauber Gomes Bezerra)

response. We validate our proposal with applications to multiple well known data sets in literature.

Keywords: online clustering, data stream, eccentricity, typicality, anomaly detection

2010 MSC: 00-01, 99-00

1. Introduction

Data clustering is the basis for solving several different classes of problems in various fields of application such as data mining, pattern recognition, data classification, system identification and image segmentation [1, 2, 3]. Due to great demand, the task of clustering data has been object of study of many different authors in the past decades, resulting in a great variety of approaches presented in literature. The intrinsic mechanisms of each of these approaches define its realm of applicability.

Most of the approaches proposed in the literature require some prior knowledge about the analyzed data set [1]. This knowledge is often presented in form of a mathematical model that describes the behavior of the data. In other cases prior training is required using a representative set of data, allowing the algorithm to learn some pattern in this set. For instance, traditional clustering techniques, such as K-means [4], require offline availability of the whole data set from the start. However, there are several problems in which data samples are acquired over time, during the execution of the algorithm, such as general data streams.

A data stream is an ordered sequence of samples, obtained over time, continuously, such as time series. Since the order of the samples is a crucial feature of the problem and they are usually obtained in an online manner, this type of data set require not only an algorithm that can perform clustering online, but also ability to adapt, since very little knowledge about the data is available *a priori* [5]. Thus, the algorithm must be able to handle during data stream analysis the occurrence of problems such as concept evolution and concept drift.

Algorithms and techniques designed handle this and other problems that involve data sets in dynamic and non-stationary environments are often called evolving intelligent systems [6]. One of the main features of such systems is their ability to adapt and evolve autonomously according to the natural changes on the data over time. Evolving intelligent systems are object of study of many authors [7, 8] and solutions based on this concept were recently introduced to many different problems [9], such as systems modeling, process controls, data prediction and classification [10, 11, 12, 13, 14, 15, 16].

Recently, several incremental learning algorithms have been proposed in the context of evolving clustering. Many of them assume a specific format for clusters, such as ellipsoids. It is important to notice that ellipsoids, for example, assume that the data follow a Gaussian distribution. [17] and [18] are two great examples of successful attempts to tackle evolving clustering that are based on such assumption. In [17], the authors propose an interesting approach based on the evolving vector quantization method that was originally published in [18], named AutoClust. The proposed algorithm has merge-and-split functionalities for the clusters. It assumes that the clusters have ellipsoid shapes, and then it uses the geometric information about cluster overlapping and joint cluster homogeneity to merge and/or split them. In [19], the authors present a fuzzy evolving clustering approach that also assumes ellipsoidal shape for the clusters, using Gaussian mixture model and fuzzy inference to define the clusters and their evolution.

In this paper we propose an algorithm for data stream clustering entitled AutoCloud. AutoCloud is fully data driven and does not require specific mathematical models or any prior knowledge about the data set to be analyzed. It is based on the recently introduced concept of Typicality and Eccentricity Data Analytics (TEDA), mostly applied to anomaly detection problems. Although the current form of AutoCloud its eccentricity calculation are based on Euclidean distance, which virtually force the definition of clusters as ellipses, the underlying idea of the algorithm does not assume any pre-defined shape, as other similarity measures can be used and data clouds, per definition, do

not have specific shapes or bounderies [20]. Therefore, AutoCloud is able to perform evolving clustering for data streams fully online and very low computational complexity.

Algorithms for evolving clustering in arbitrary shaped clusters have been introduced and follow the classic DBSCAN algorithm [21], with concepts of micro and macro clusters. These algorithms work generally in two stages: micro-clustering is performed online while macro-clustering is executed offline, as in [22, 23, 24]. From another perspective, [25] proposes a fully online clustering of evolving data streams with arbitrarily shaped clusters, named CEDAS, that updates micro-clusters and macro-clusters in online way. It uses a graph structure to associate a set of micro-clusters to a macro-cluster, where the micro-clusters are the nodes and the edges are its pairs with intersecting micro-clusters. More recently, [26] proposed the BOCEDS algorithm, which is also a fully online density-based clustering algorithm using micro-clusters and macro-clusters concepts for evolving data streams. It uses an energy function based on the spatial information of the data stream for online updating of the micro-clusters. In addition, it adopts a buffer for storing temporarily micro-clusters and when these micro-clusters are non-significant, they are removed from the buffer. Two stage algorithms are hard to compare with the approach presented in this paper, since we decided to focus on a fully online solution. If a scenario permits a training stage, AutoCloud can be easily adapted to perform micro/macro clustering.

AutoCloud is also an evolving algorithm, being able to autonomously adapt to the changes in the data over time, such as concept drift and concept evolution [6]. It is also recursive and online, which means it does not require storing and processing past data samples. This greatly reduces the computational burden, computer- and memory-wise. One of the main practical advantages of AutoCloud is that it introduces an elegant and efficient way of creating new clusters and merging existing ones according to the evolution of data over time.

The remainder of this paper is organized as follows. In Section 2 we present the concepts of TEDA. In Section 3, we detail our proposed technique. In Section 4, we describe and discuss the experiments and results obtained with

the application of AutoCloud to several well-known data sets. Finally, in Section 5, we present our conclusions and discuss potential future work.

2. Typicality and Eccentricity Data Analytics

TEDA is an evolving method for anomaly and outlier detection, introduced by [20]. However, the concepts of typicality and eccentricity have been successfully applied to, among other problems, data classification [27, 28]. The concept of typicality is related to the similarity of a specific n-dimensional data sample to the values of its past readings. Eccentricity, conversely, describes how different a data sample is from the data distribution. Hence, a data sample with high eccentricity (and thus low typicality) is usually an anomaly.

Eccentricity is summarized as the the sum of distances of a particular data sample to all other existing data samples divided by the sums of distances from all data samples to all other data samples. In the realm of data strems, consider a data input $X \in \Re^n$, which consists of a sequence of n-dimensional data samples, i.e. $X = \{x_1, x_2, \ldots, x_k, \ldots\}, \quad x_i \in \mathbb{R}^n, \quad i \in \mathbb{N}$, where k is the discrete time instant in which the sample was acquired. Consider also $d(x_i, x_j)$ as some distance between samples x_i e x_j, in which d can be any type of distance such as Euclidean, cosine or Mahalanobis. For the complete

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

where $\pi_k(x)$ is the sum of distances from a particular sample $x \in X$, for each of the k elements of the data set.

The eccentricity ξ of the data sample x at the time instant k is defined as [20]

$$\xi_k(x) = \frac{2\pi_k(x)}{\sum_{i=1}^{k} \pi_k(x_i)} = 2\frac{\sum_{i=1}^{k} d(x, x_i)}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(x_i, x_j)}, \quad k \geq 2, \quad \sum_{i=1}^{k} \pi_k(x) > 0 \tag{2}$$

However, equation 2 refers to the offline formula of eccentricity. It has been

shown [20] that eccentricity can be derived exactly as

$$\xi_k(x) = \frac{1}{k} + \frac{(\mu_k^x - x_k)^T(\mu_k^x - x_k)}{k[\sigma^2]_k^x}, \quad [\sigma^2]_k^x > 0 \tag{3}$$

where $\xi_k(x)$ is the eccentricity of the sample x_k in relation to all previous samples in the data set, while μ_k^x is the mean and $[\sigma^2]_k^x$ is the aggregated variance of x up to the time instant k. Both μ_k^x and $[\sigma^2]_k^x$ can be recursively updated by

$$\mu_k^x = \frac{(k-1)}{k}\mu_{k-1}^x + \frac{1}{k}x_k, \quad k \geq 1, \quad \mu_0^x = 0 \tag{4}$$

$$[\sigma^2]_k^x = \frac{(k-1)}{k}[\sigma^2]_{k-1}^x + \frac{1}{k}||x_k - \mu_k||^2, \quad k \geq 1, \quad [\sigma^2]_0^x = 0 \tag{5}$$

Conversely, the τ of the sample x at the time instant k can be calculated as a complement of eccentricity, as in [20]:

$$\tau_k(x) = 1 - \xi_k(x) \tag{6}$$

In Section 3, we develop the idea on how eccentricity in its form presented in equation 3 can be used to determine whether a data point belongs to an existing data cloud and how it can be used to determine when a new data cloud must be created to compensate for a potential data drift. Eccentricity and typicality are bounded by [20]

$$0 \leq \xi_k(x) \leq 1, \quad \sum_{i=1}^{k} \xi_k(x_i) = 2, \quad k \geq 2$$

$$0 \leq \tau_k(x) \leq 1, \quad \sum_{i=1}^{k} \tau_k(x_i) = k - 2, \quad k \geq 2$$

$$\sum_{i=1}^{k} \pi_k(x_i) > 0, \quad k \geq 2$$

Finally, the normalized eccentricity $\zeta_k(x)$ and typicality $t_k(x)$ can be ob-

tained by [20]:

$$\zeta_k(x) = \frac{\xi_k(x)}{2}, \quad \sum_{i=1}^{k} \zeta_i(x) = 1, \quad k >= 2 \tag{7}$$

$$t_k(x) = \frac{\tau_k(x)}{k-2}, \quad \sum_{i=1}^{k} t_i(x) = 1, \quad k >= 2 \tag{8}$$

The threshold used to distinguish normal from anomalous data samples is based on the Chebyshev inequality [29], which states that, under any distribution, no more than $1/m^2$ of the data observations are more than $m\sigma$ away from the mean, where σ represents the standard deviation of the data. Thus, a particular data sample x_k is considered to be an anomaly if the condition

$$\zeta_k > \frac{m^2+1}{2k}, \quad m > 0 \tag{9}$$

is satisfied. The parameter m is user-defined and directly affects the sensitivity of the anomaly detector. Although it can be defined using multiple criteria, $m = 3$ is largely used in literature [8, 30] as a standard value and presents satisfactory results for different data sets and different configurations. Once again, equation 9 is central to the method proposed in this paper and in Section 3, we demonstrate how it is used in the determination of membership of points to existing data cloud and potentially triggering of creation of new ones.

3. AutoCloud

The evolving clustering algorithm proposed in this paper is called AutoCloud and is based on TEDA, which makes it suitable for online processing of data streams. Among its characteristics, one can mention:

i) It leverages ever-evolving cluster-like granular structures - the data clouds - where not only the parameters of each granule can be adapted, but also new clusters can be created and existing ones merged, which makes

it suitable for handling dynamic and evolving data (concept drift and concept evolution).

ii) Calculations are recursive, in such a way that it does not require storing previous data samples in memory, executing batch processing tasks nor using sliding windows. Therefore, the resulting algorithm is very fast and and computational cheap, hence, suitable for real-time applications.

iii) It is fully unsupervised, does not require offline training or prior knowledge about the data and can be started from an empty knowledge basis.

3.1. Data Clouds

The granular structures used in AutoCloud are called data clouds [6]. Similarly to traditional clusters, they are local sets of data samples with common properties. Data clouds, however, do not have a particular shape nor predefined boundaries, therefore, are a much more realistic representation of the actual data distribution than traditional fuzzy membership functions (e.g. triangular, trapezoidal, Gaussian), in which, instead, one only targets an approximation of an expected/desired distribution.

Although data clouds do not have specific shapes or boundaries (these are indirectly derived from the type of similarity measure used), they are visually represented as ellipses in this paper.

AutoCloud calculates the data eccentricity ζ_k for each new data sample x_k in relation to each existing cloud independently, as showed in Figure 1. In this example, note that AutoCloud determines the membership of the sample x_k to the two existing clouds c_1 and c_2, based on the eccentricity value ζ_k for each cloud and the dynamic threshold proposed in equation 9. When this condition holds, AutoCloud determines that the membership of x_k to such cloud is significant enough that the x_k *affects* that cloud.

Since AutoCloud is a recursive algorithm, past data samples do not to be stored in memory. Instead, only a small number of statistical variables are

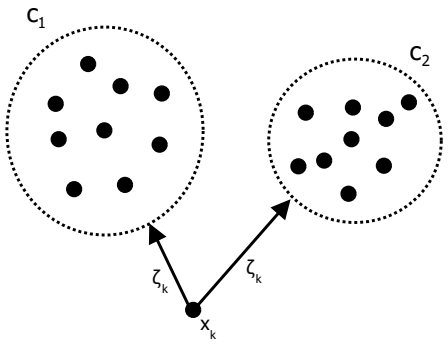


Figure 1: Auto-Cloud overview.

required for each cloud c_i: the number of samples ($\mathbf{s_k^i}$), the mean ($\mu_\mathbf{k}^\mathbf{i}$) and the variance ($[\sigma^\mathbf{2}]_\mathbf{k}^\mathbf{i}$) of samples that *affect/belong* to c_i at the time instant k.

For instance, Figure 2 illustrates a scenario with two data clouds, c_1 e c_2, after the reading of k samples. The number of samples that belong to each cloud is $s_k^1 = 7$ and $s_k^2 = 8$, while *three* samples belong to both c_1 e c_2 simultaneously. Finally, the mean of each existing cloud, μ_k^1, μ_k^2, respectively, are visually represented in the same image by the centers of c_1 and c_2, while the variances $[\sigma^2]_k^1$ and $[\sigma^2]_k^2$ are represented by the spreading/radii of each cloud in the data space.

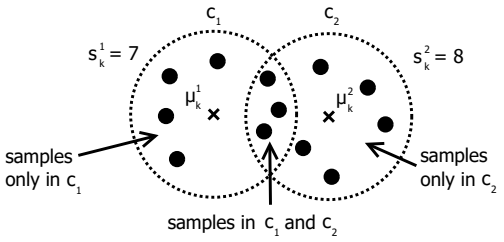


Figure 2: Data clouds c_1 and c_2 at the time instant k.

3.2. Data clouds update

For each new data sample, AutoCloud determines its relevance to each existing cloud based on the value of the normalized eccentricity to c_i. When the

eccentricity is significantly high, the sample x_k is considered *irrelevant* in relation to c_i and, therefore, does not affect that specific cloud. However, if x_k has low eccentricity, AutoCloud determines that the sample *belongs* to c_i (i.e. x_k is similar to the other samples that belong to c_i) and updates that cloud. In that case, the number of samples s_k^i, the mean μ_k^i and the variance $[\sigma^2]_k^i$ are recursively updated to reflect the influence of the new sample on c_i.

It is important to highlight that a data sample can belong to multiple points simultaneously, preserving the nature of fuzzy membership. Alternatively, the algorithm may also determine that x_k has high eccentricity in relation to all existing clouds. In that case, a new cloud is created.

The threshold used to determine whether a sample x_k belongs or not to a cloud c_i, we generalize the equation 9 to reflect local membership to individual clusters:

$$\zeta^i(x_k) <= \frac{m^2+1}{2s_k^i} \tag{10}$$

where m represents the sensitivity of the threshold. Hence, if the condition represented in equation 10 holds, it is determined that x_k belongs to c_i. On the other hand, if the condition does not hold, x_k does not belong c_i.

The remaining equations are also generalized versions of the equations presented in Section 2. Thus, the eccentricity $\xi^i(x_k)$ and the normalized eccentricity $\zeta^i(x_k)$ of the sample x_k in relation to the i-th data cloud are given by

$$\xi^i(x_k) = \frac{1}{[s_k^i]'} + \frac{([\mu_k^i]' - x_k)^T([\mu_k^i]' - x_k)}{[s_k^i]'[[\sigma^2]_k^i]'} \tag{11}$$

$$\zeta^i(x_k) = \frac{\xi^i(x_k)}{2} \tag{12}$$

where $[s_k^i]'$, $[\mu_k^i]'$ and $[[\sigma^2]_k^i]'$ are temporary values for the number of samples, mean and variance of the i-th data cloud, respectively, supposing that x_k belongs to c_i. Such supposition is necessary in order to verify, in sequence, if x_k indeed belongs to c_i. These values are calculated by, respectively

$$[s_k^i]' = s_{k-1}^i + 1 \tag{13}$$

$$[\mu_k^i]' = \frac{[s_k^i]' - 1}{[s_k^i]'}\mu_{k-1}^i + \frac{1}{[s_k^i]'}x_k, \tag{14}$$

$$[[\sigma^2]_k^i]' = \frac{[s_k^i]' - 1}{[s_k^i]'}[\sigma^2]_{k-1}^i + \frac{1}{[s_k^i]'}\|x_k - [\mu_k^i]'\|^2 \tag{15}$$

For each cloud c_i, with $i = [1...N]$, where N is the number of existing clouds, and if equation 10 holds, the values of s_k^i, μ_k^i and $[\sigma^2]_k^i$ are updated, respectively, by $[s_k^i]'$, $[\mu_x^i]'$ and $[[\sigma^2]_k^i]'$, previously calculated by equations 13, 14 and 15. This update must be executed since the supposition that x_k belongs to c_i is true and hence, c_i needs to be updated to reflect the influence of x_k. On the other hand, if equation 10 does not hold, that means x_k does not belong and, therefore, should not affect c_i.

Figure 3 illustrates this procedure by showing three data clouds (c_1, c_2 and c_3) and an input sample x_k. According to the image, AutoCloud calculates the normalized eccentricity of x_k in relation to each cloud, $\zeta^1(x_k)$, $\zeta^2(x_k)$ and $\zeta^3(x_k)$, respectively. In this particular example, it can be seen that x_k belongs to both c_1 and c_3, but not to c_2. Hence, only c_1 and c_3 are updated, while no action is performed for c_2, as shown in Figure 3(b).

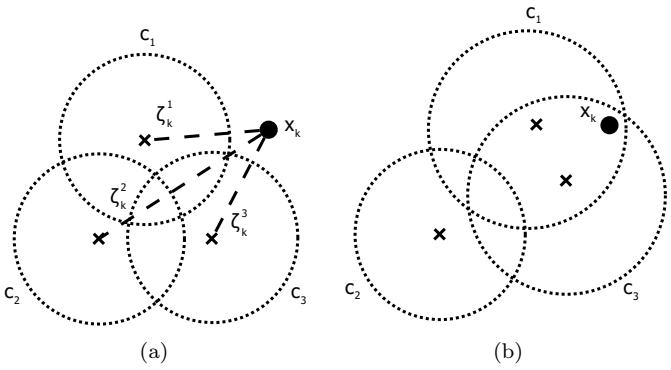


Figure 3: Update of data clouds: (a) c_1, c_2, c_3 and a newly read sample x_k, and (b) c_1, c_2, c_3 after the update

In the case where equation 10 does not hold for any of the existing N clouds, a new data cloud c_{N+1} is created and added to the rule bases. It is initialized with $s_k^{N+1} = 1$, $\mu_k^{N+1} = x_k$ and $[\sigma^2]_k^{N+1} = 0$. As an illustrative example, consider Figura 4, in which the three clouds c_1, c_2 and c_3 and a new data sample x_k are presented. It is easy to observe in Figure 4(a) that x_k is significantly distant from all existing data clouds, hence a new cloud c_4 is created, as shown in Figure 4(b).

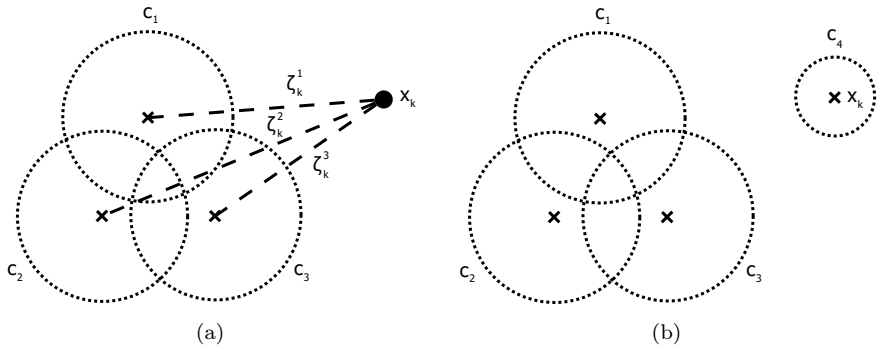


Figure 4: Creation of a new data cloud: (a) three existing clouds, c_1, c_2, c_3 and a new sample x_k and (b) a new cloud c_4 is created

3.3. Merging data clouds

In order to limit the number of clouds and, at the same time, preserve the evolving characteristics of this approach, AutoCloud is able to merge two partially overlapping data clouds, when applicable. Merge is executedwhen the number of overlapping samples, i.e. the number of samples that belong to both clouds simultaneously, is significantly high. This operation is fully autonomous and non-parametric.

Given two clouds c_i and c_j at the time instant k, merge happens when at least one of the following conditions is true:

$$s_k^{c_i \cap c_j} > s_k^i - s_k^{c_i \cap c_j} \tag{16}$$

$$s_k^{c_i \cap c_j} > s_k^j - s_k^{c_i \cap c_j} \tag{17}$$

where $s_k^{c_i \cap c_j}$ corresponds to the number of intersecting samples in c_i and c_j at the time instant k.

In summary, these two conditions tell us that two clouds are merged when the number of samples that belong to both clouds *simultaneously* is higher than the number of samples that belongs to only one of them *separately*. The process of merging overlapping data clouds prevents an uncontrolled growth in the rule/cloud basis.

Figure 5 illustrates the process of merging clouds on AutoCloud. In Figure 5(a) the number of intersecting points in clouds c_1 and c_2 is less than the number of exclusive points in both clouds. Hence, in this case, merge does not take place. Conversely, Figure 5(b) illustrates a concrete merge example, with resulting cloud basis presented in Figure 5(c).

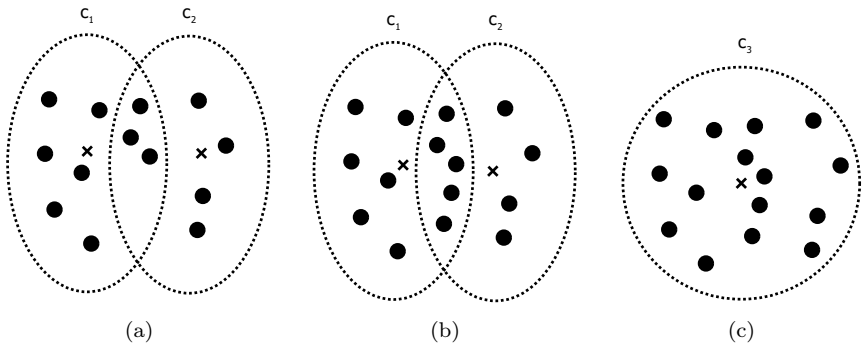


Figure 5: Data cloud fusion: c_1 and c_2, (b) c_1 and c_2 right before merging and (c) resulting cloud c_3 after merging

When two clouds merge, the properties of the resulting cloud are defined as per

$$s_k^l = s_k^i + s_k^j - s_k^{c_i \cap c_j} \tag{18}$$

$$\mu_k^l = \frac{s_k^i \mu_i + s_k^j \mu_j}{s_k^i + s_k^j} \tag{19}$$

$$[\sigma^2]_k^l = \frac{(s_k^i - 1)\sigma_i^2 + (s_k^j - 1)\sigma_j^2}{s_k^i + s_k^j - 2} \tag{20}$$

where the values of μ_k^l and $[\sigma^2]_k^l$ are obtained through a weighted mean of the respective values of each original cloud.

Analyzing the overall update, merge and creation of clouds, it is easy to note that AutoCloud is an algorithm that is able to handle two of the problems that are inherent to data stream analysis: concept drift and concept evolution.

3.4. AutoCloud Algorithm

Finally, Algorithm 1 shows in details each step of AutoCloud. In this example, we use Euclidean distance as the underlying similarity measure between samples and clouds.

For each obtained data sample, AutoCloud determines if it belongs to any of the existing clouds. If that is the case, any of the affected (and only the affected) clouds are updated in order to reflect the addition of this new sample. Else, i.e. the sample is significantly different from *all* existing clouds, a new cloud is created. Finally, each pair of clouds is analyzed in order to determine if merge is necessary. Considering n the number of features in the dataset and C_k the number of existing clusters at the time instant k, the complexity of AutoCloud is $O(n * C^2)$ for a data sample x_k processed at the time instant k.

Analyzing the algorithm, we can verify that at its startup the first data cloud is created and the first two samples obtained ($k = 1$ and $k = 2$) are added to this data cloud. The process of creating new data clouds starts only from the third sample. The reason for this is that the distance between points is relative to the feature ranges. There is no way of determining if 2 points are near or far if there is no pre-defined range, which is the case. That is why a third point is expected before starting by determining how significant the distance between any two points is.

Algorithms 2 and 3 detail the steps for creation and merge of data clouds, respectively.

Algorithm 1: Algorithm *Auto-Cloud* using euclidean distance.

Input: set of samples $X = \{x_1, x_2, x_3, \dots\} \in R^n$

begin
  while $x_k \leftarrow$ *read next sample* **do**
    if $k = 1$ **then**
      `// create data cloud 1 and add` x_1
      $s_1^1 \leftarrow 1$
      $\mu_1^1 \leftarrow x_1$
      $[\sigma^2]_1^1 \leftarrow 0$
      $N \leftarrow 1$
    else
      if $k = 2$ **then**
        `// add` x_2 `in data cloud 1`
        $s_2^1 \leftarrow 2$
        $\mu_2^1 \leftarrow (\mu_1^1 + x_2)/2$
        $[\sigma^2]_2^1 \leftarrow \|x_2 - \mu_2^1\|/2$
      else
        if $k >= 3$ **then**
          forall *existing data cloud* c_i **do**
            calculate $[s']_k^i$, $[\mu']_k^i$ *and* $[[\sigma^2]']_k^i$ *by equations 13, 14 e 15*
            calculate ζ_k^i *by equation 12*
            if $\zeta_k^i <= (m^2 + 1)/(2 * [s']_k^i)$ **then**
              `// add` x_k `in data cloud` i
              $s_k^i \leftarrow [s']_k^i$
              $\mu_k^i \leftarrow [\mu']_k^i$
              $[\sigma^2]_k^i \leftarrow [[\sigma^2]']_k^i$
            else
              $s_k^i \leftarrow s_{k-1}^i$
              $\mu_k^i \leftarrow \mu_{k-1}^i$
              $[\sigma^2]_k^i \leftarrow [\sigma^2]_{k-1}^i$
            end
          end
          if $x_k \notin c_i, \forall i$ **then**
            $createDataCloud(x_k)$
          end
          forall *pair of data clouds* c_i *and* c_j **do**
            $verifyMerge(c_i, c_j)$
          end
        end
      end
    end
  end
end

Algorithm 2: Procedure that create a new data cloud using euclidean distance.

procedure *createDataCloud* (x_k)
    // create new data cloud l
    $n_k^l \leftarrow 1$
    $\mu_k^l \leftarrow x_k$
    $[\sigma^2]_k^l \leftarrow 0$
    $\alpha_{c_l} \leftarrow 1$
    $N \leftarrow N + 1$
end

Algorithm 3: procedure that performs the merge between two data clouds using euclidean distance.

procedure *verifyMerge* (c_i, c_j)
    if $s_k^{c_i \cap c_j} > s_k^i - s_k^{c_i \cap c_j}$ ***or*** $s_k^{c_i \cap c_j} > s_k^j - s_k^{c_i \cap c_j}$ **then**
        // merge data clouds i and j to data cloud l
        calculate s_k^l, μ_k^l *and* $[\sigma^2]_k^l$ *by equations 18, 19 e 20*
        $N \leftarrow N - 1$
    end
end

At the end of each iteration k, AutoCloud outputs the following values:

- The number of samples (s_k^i), the mean (μ_k^i) and the variance ($[\sigma^2]_k^i$) of each existing cloud (c_i) at the time instant k.
- The list with membership degrees (γ_{c_i}) of the sample x_k to each cloud (c_i).

4. Results

To validate AutoCloud, we will present in this section the results obtained by applying the proposed algorithm to data stream clustering problems. These results were obtained using data sets already consolidated and widely used in the field of machine learning. In addition to this data, we also used some artificially generated data to demonstrate some features of AutoCloud.

4.1. Data sets

The data sets used were obtained from a specific repository for data clustering [31]. This repository has some data sets, both artificial and real, that are widely used in data clustering tasks. For each data set, the number of samples and the number of existing clusters are specified. In some cases, the centroid of each group is also available and the information about to which group each sample belongs.

The data sets of the adopted repository are divided into several categories. Among the existing synthetic data categories, we make use of the following: S-sets, A-sets, DIM-sets (high), Shape-sets, and Unbalance.

We have selected some of the data sets belonging to these categories to perform our experiments. Table 1 shows a summary of the selected data sets, indicating the number of samples, N, the number of *clusters*, K, and the size of each of cluster. Note that most of the selected data is 2-dimensional, but this is not a prerequisite for AutoCloud, since it is capable of handling data of any dimensionality. This choice was made only to facilitate the visualization of the data and the obtained results. To prove this, we will also use two high-dimension data sets, `dim512` and `dim1024` which have dimensions 512 and 1024, respectively.

Table 1: Synthetic data sets used in data clustering experiments.

Data set	**Category**	**N**	**K**	**Dimension**
S1	S-sets	5000	15	2
S2		5000	15	2
A1	A-sets	3000	20	2
A2		5250	35	2
dim512	Dim-sets (high)	1024	16	512
dim1024		1024	16	1024
Aggregation	Shape-sets	788	7	2
Compound		399	6	2
Unbalance	Unbalance	6500	8	2

4.2. Experimental Results

AutoCloud was applied to each of the selected data sets in order to get evaluate the proposed clustering approach. Table 2 shows the number of data

clouds obtained, $\hat{K}$, the quality of data clouds obtained, the processing time for each data stream, t_{s}, and the average processing time of each sample, t_{a}. This table also shows the real number of clusters, K, in each data set.

Table 2: Results obtained in the clustering of analyzed data sets.

Data set	**K**	$\hat{\mathbf{K}}$	**Quality**	t_s **(s)**	$t_{\mathbf{a}}$ **(ms)**
S1	15	15	0.30	3.71	0.74
S2	15	15	0.41	3.48	0.70
A1	20	20	0.30	3.0	1.00
A2	35	35	0.23	11.79	2.24
dim512	16	16	0.16	22.52	2.20
dim1024	16	16	0.39	90.13	88.02
Aggregation	7	7	0.52	0,18	0.23
Compound	6	4	0.53	0.07	0.10
Unbalance	8	8	0.34	1.21	0.19

Analyzing the results shown in Table 2, we can verify that AutoCloud was able to accurately identify the actual number of clusters in almost all analyzed data sets. The number of data clouds created by AutoCloud was not equal to the number of clusters in the `compound` base, where AutoCloud created only 4 data clouds, two less than expected. In this experiment, the performance of AutoCloud was satisfactory, regardless of the dimensionality of the data sets analyzed, since it was able to correctly identify the number of clusters in the low and high dimension data sets.

To measure the quality of the obtained data clouds we used the same metric used by [17]. According to this metric, the quality value is calculated using the following equation:

$$quality = \frac{\sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^{2}||\vec{x}_i - \vec{x}_j||}{N min_{i,j=1,\ldots,C,i\neq j}(||\vec{c}_i - \vec{c}_j||)} \tag{21}$$

where C denotes the number of data clouds obtained, N is the total number of samples and μ_{ij} is the membership degree of sample j in data cloud i. The value of μ_{ij} was calculated using the typicality of sample j in data cloud i. The quality values obtained were similar to those obtained by [17] using AutoClust in data sets A1, A2, S1 e S2. The lower calculated value, the better the quality

of the data clouds.

Regarding the processing time required by AutoCloud, we verified that this value is proportional to the size of the analyzed data stream, as well as the dimension of the samples. The largest obtained values were in the processing of the two high dimension data sets used (`dim512` e `dim1024`). It is important to note that the measured processing time does not take into account any sampling rate when reading the data. The average processing time of each sample was obtained by dividing the total processing time of the stream by the number of samples in that stream. Analyzing the obtained values of t_a, AutoCloud proved to be an time-efficient algorithm and adequate to on-line applications. In the worst presented case, each sample was processed at about $88ms$. This result allows AutoCloud to be used in time-constrained applications that require fast responses.

In order to evaluate the quality of the obtained data clouds in the experiments, we have to go beyond the identified number of clusters/data cloud. It is also necessary to verify that the centroid of each identified cloud date was calculated correctly by comparing the obtained with the expected centroids. The obtained centroids in each of the analyzed data sets are shown in Figures 6, 7, 8, 9 and 10. In each of these figures the samples belonging to each data set are shown in blue. The black dots represent the real centroids of the clusters and the red dots correspond to the centroids of data clouds obtained by AutoCloud.

The centroids obtained for data sets S1 and S2 are shown in Figure 6. In the image, it is possible to visualize that the data are grouped in well-defined clusters, but with different shapes, quantity of samples and spreading of samples. In data set S2, the clusters are closer to each other than in S1, thus there is a greater degree of overlap between them in that set. It is also possible to visualize that the centroids of all the data clouds obtained correspond to the expected values in both S1 and S2. This can be verified by observing that the points marked with red `X` coincide with the points marked with the black circle.

The obtained centroids in sets A1 and A2 are shown in Figure 7. We have verified that the clusters are well defined in these two sets. Another character-

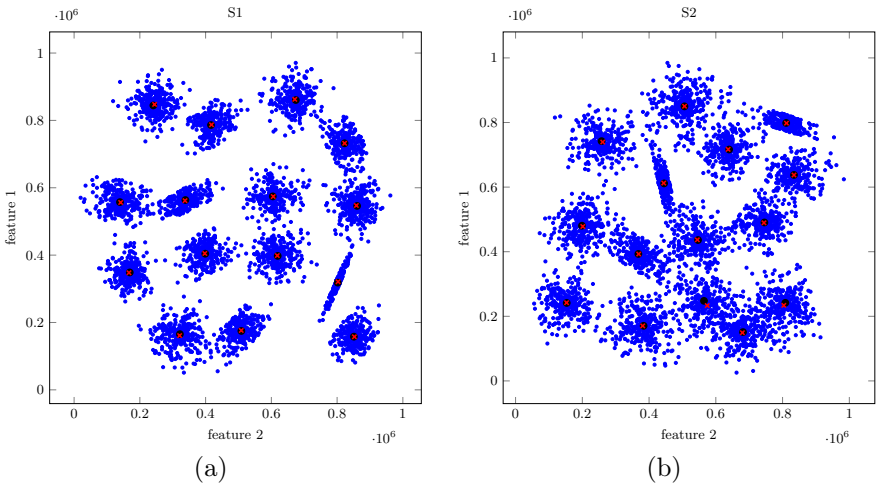


Figure 6: Obtained centroids in clustering of S-Sets: (a) S1 e (b) S2.

istic of these two sets is that the amount of samples in each of the clusters is equal. By observing the Figure, we find that the centroids obtained in all data clouds correspond to the real values expected for each cluster in the two sets.

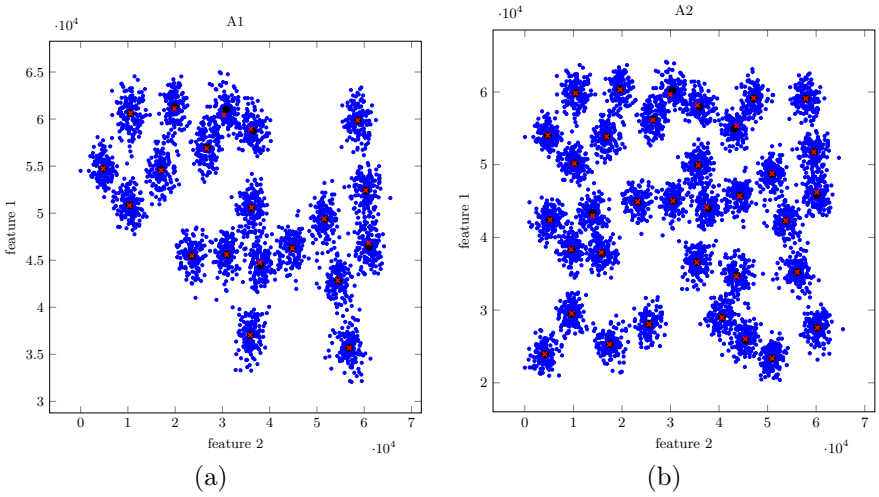
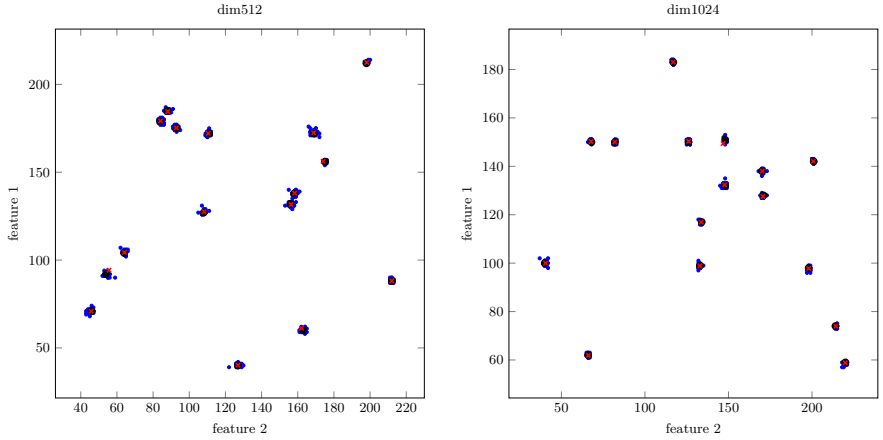


Figure 7: Obtained centroids in clustering of A-Sets: (a) A1 e (b) A2.

Figure 8 shows the obtained centroids in the high-dimension sets dim512 and dim1024. The data from these two sets were presented to AutoCloud in its entirety, without any prior feature selection. However, to facilitate the visual-

ization of the data and the results obtained, the graphs presented in this Figure show only the first two dimensions of the samples. Once again, the centroids in each identified data cloud corresponded to the expected values, which was obtained regardless of the high dimension in analyzed data. In only a few cases, one in the set dim512 and two in the set dim1024, there was a small error in the identified centroids. However, this error was not very significant, still allowing the identification of data clouds correctly.



(a) (b)

Figure 8: Obtained centroids obtidos in clustering of Dim-Sets: (a) dim512 e (b) dim1024.

The obtained results for the Shape-sets used in the experiments are shown in Figure 9. These two data sets are among those analyzed the ones with the least number of samples. In addition, the clusters have different shapes and sizes, but can also be easily identified visually. In the Aggregation set, all centroids were identified quite accurately by AutoCloud. In Compound set, an error occurred in the identification of the clusters. In this data set the number of identified data clouds did not match the number of expected clusters. Analyzing Figure 9 we find that there are two clusters that are comprised in two others clusters. This caused each of these two pairs of clusters to be identified as a single data cloud, reflecting both data clouds unless they should have been identified.

Finally, Figure 10 shows the centoids obtained for data set `Unbalance`. This

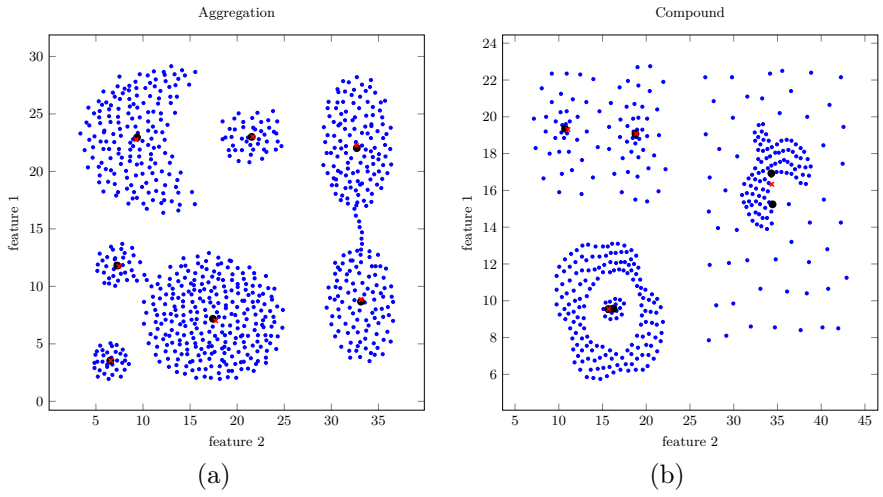


Figure 9: Obtained centroids in clustering of Shape-Sets: (a) Aggregation e (b) Compound.

data set presents 8 clusters and most samples are concentrated in 3 of them. These 3 clusters have 2,000 samples each, while the remaining 5 have only 100 samples each. We can observe in Figure 10 that the centroids were obtained again satisfactorily, regardless of the amount of samples presented by each cluster.

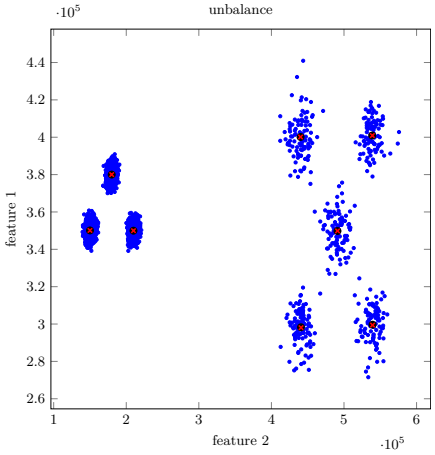


Figure 10: Obtained centroids in clustering of Unbalance.

Based on the presented results, it is possible to note that the AutoCloud was

able to determinate the *clusters* and their respective centers adequately. It is important to highlight all input data were processed in the form of data streams by AutoCloud. More specifically, at the $k-th$ time instant, AutoCloud only has access to the $k-th$ data sample. In addition, AutoCloud does not require storing previous data samples.

Aiming to analyze the influence the order of samples into the data stream for clustering in an evolving context, we used again the `unbalance` data set, but the order of the data was randomized. Figure 11 shows the obtained results with this experiment. Figure 11(a) presents the original data stream and Figure 11(b) presents the randomized data stream. Figure 11(c) and Figure 11(d) present the obtained clusters by AutoCloud for the original data stream and the randomized data stream, respectively. The obtained centroids are indicated by red dots. As expected, for the former case the algorithm obtained 8 clusters, while for the latter it resulted in a single cluster, since the nature of the algorithm requires the samples to be processed in the order they are acquired.

Therefore, it is important to highlight that AutoCloud is not suitable for the analysis of data that does not have a time dependency between samples. However, for the vast majority of real-world problems, one of the main characteristics of data streams is that they carry very strong temporal relationship between neighbor data samples (e.g. In a control process, none of the input/output variables are expected to randomly oscillate between two sequential time instants). That is precisely the type of problem AutoCloud is proposed to address.

The only tunable parameter of AutoCloud is the value of m, that is used to determine the threshold defined by equation 9. All previous presented results used $m = 2$, which is equivalent to using a 2σ threshold under normal distribution. This parameter was set up based in previous successful works [8, 30]. Then, in order to analyze the impact of the parameter m over AutoCloud's performance, we have used $m = 1, 2$ and 3 to data sets used previously. The obtained results are presented in Table 3 and confirm that AutoCloud's sensitivity for creating new clusters increase when value of m decreases and *vice-versa.*

Although we have used only integer values for m, any real positive number

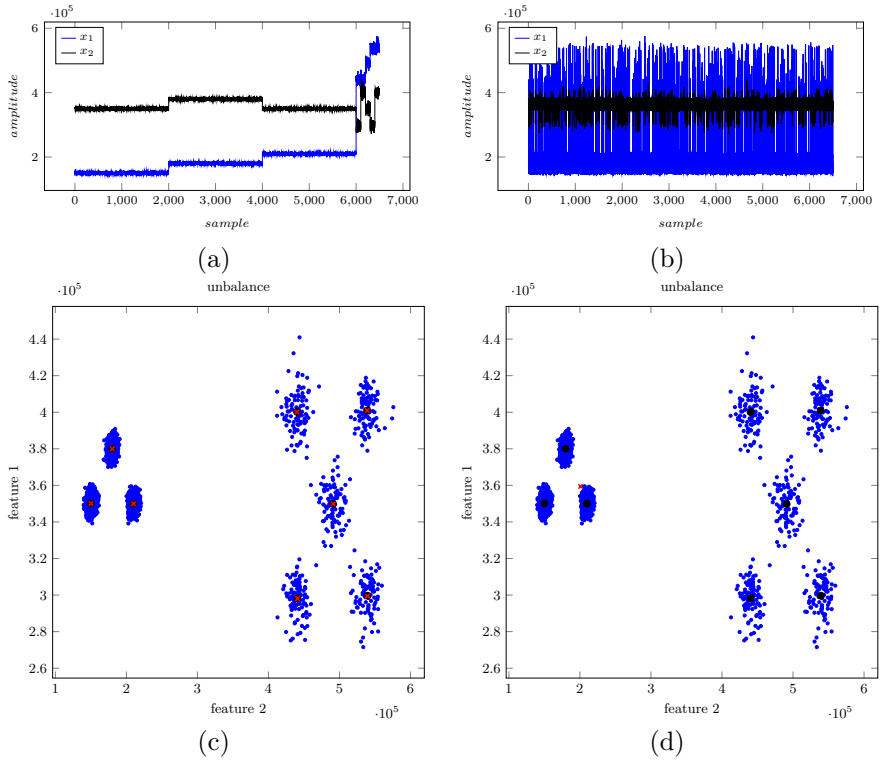


Figure 11: Obtained centroids in clustering of data set `Unbalance` modified: (a) original data stream, (b) randomized data stream, (c) obtained centroids for original data stream and (d) obtained centroids for randomized data stream.

Table 3: Obtained results of analyzed data sets using several values of m.

Data set	**K**	$\hat{\mathbf{K}}_{\mathbf{m=1}}$	$\hat{\mathbf{K}}_{\mathbf{m=2}}$	$\hat{\mathbf{K}}_{\mathbf{m=3}}$
S1	15	114	15	10
S2	15	123	15	1
A1	20	150	20	1
A2	35	250	35	1
dim512	16	30	16	16
dim1024	16	42	16	16
Aggregation	7	278	7	1
Compound	6	126	4	1
Unbalance	8	103	8	8

can be used. Thus, by adjusting this value, AutoCloud is able to achieve different granularization during identification of data clouds. The best value of m will depend on the characteristics of analyzed data distribution. However, once the

AutoCloud is executed online from the very first data sample, m can only be optimized if previous information about the data is available, which is not the case for the problems presented in this paper. As future work, we will investigate how to relate the value of m to some measure of data dispersion. Thus, we could estimate a value of m adaptive, updated for each sample analyzed.

AutoCloud is built to cope with concept evolution by design, since it starts from scratch and creates and updates clusters on-demand. In Figure 12, we illustrate the processes of creation and merging of data clouds for `dim1024`, `aggregation`, and `unbalance` data sets.

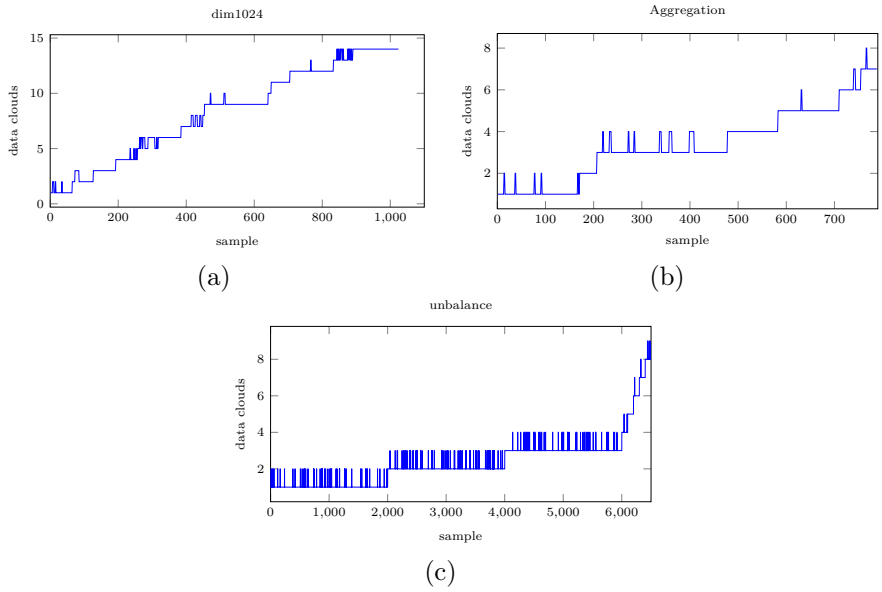


Figure 12: Creation and merging of data clouds in data sets (a) dim1024, (b) Aggregation e (c) Unbalance.

It is possible to observe that, as the data stream is processed, the number of data clouds tends to increase. This is due to the fact that AutoCloud can identify new concepts in the data stream and create new clusters to represent them properly. Complementary, procedures of merging data clouds tend to occur soon after the creation of a new cluster since, in many cases, the algorithm realizes that the data cloud creation was not necessary, thus performing a self-

regulating activity by the merging procedure.

5. Conclusion

In this paper we presented an algorithm for data streams clustering called AutoCloud. The obtained clusters are called data clouds. AutoCloud is an evolving algorithm capable of autonomously identifying changes that occur in the data stream distribution over time. Its evolving properties allow updating of data cloud parameters (concept drift), creation of new data clouds (concept evolution) and merging of existing data clouds, in order to auto-adapt to changes in the data stream over time.

Based on presented results, AutoCloud was able to correctly identify both the quantity and the location of the analyzed clusters. In addition, the proposed strategy for merging of data clouds proved to be quite efficient. Other relevant aspect that must be highlighted is that AutoCloud does not use representation models of data distribution, thus it was able to identify clusters regardless of their format.

Since AutoCloud is executed recursively and does not required data samples to be stored, it is very computationally efficient, processing- and memory-wise.

For future works, in order to improve the AutoCloud we intend to investigate the use of the Mahalanobis distance as metric and to propose efficient strategies to split clouds dynamically. In addition, we will also investigate how to obtain an adaptive value of m, with this value being updated with each new sample analyzed.

[1] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2) (2015) 165–193. doi:10.1007/s40745-015-0040-1. URL https://doi.org/10.1007/s40745-015-0040-1

[2] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, C.-T. Lin, A review of clustering techniques and developments, Neurocomputing 267 (2017) 664 – 681.

doi:https://doi.org/10.1016/j.neucom.2017.06.053.
URL http://www.sciencedirect.com/science/article/pii/S0925231217311815

[3] C. C. Aggarwal, C. K. Reddy, Data Clustering: Algorithms and Applications, 1st Edition, Chapman & Hall/CRC, 2013.

[4] J. Macqueen, Some methods for classification and analysis of multivariate observations, in: In 5-th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[5] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, J. a. Gama, Data stream clustering: A survey, ACM Comput. Surv. 46 (1) (2013) 13:1–13:31. doi:10.1145/2522968.2522981.

[6] P. Angelov, Autonomous Learning Systems: From Data to Knowledge in Real Time, John Willey and Sons, 2012.

[7] F. Gomide, E. Lughofer, Recent advances on evolving intelligent systems and applications, Evolving Systems 5 (4) (2014) 217–218. doi:10.1007/s12530-014-9121-1.
URL https://doi.org/10.1007/s12530-014-9121-1

[8] 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, Information Sciences 490 (2019) 344 – 368.

[9] P. Angelov, D. P. Filev, N. Kasabov, Evolving Intelligent Systems: Methodology and Applications, Wiley-IEEE Press, 2010.

[10] P. Angelov, X. Zhou, Evolving fuzzy-rule-based classifiers from data streams, Fuzzy Systems, IEEE Transactions on 16 (6) (2008) 1462–1475. doi:10.1109/TFUZZ.2008.925904.

[11] J. Iglesias, A. Ledezma, A. Sanchis, P. Angelov, Real-time recognition of calling pattern and behaviour of mobile phone users through anomaly detection and dynamically evolving clustering, Applied Sciences 7 (8) (2017) 1–14. doi:10.3390/app7080798.

[12] B. S. J. Costa, P. Angelov, L. A. Guedes, Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier, Neurocomputing 150 (A) (2015) 289–303. doi:10.1016/j.neucom.2014.05.086.

[13] D. Kangin, P. Angelov, J. A. Iglesias, Autonomously evolving classifier {TEDAClass}, Information Sciences 366 (2016) 1 – 11. doi:http://dx.doi.org/10.1016/j.ins.2016.05.012.

[14] M. Traore, A. Chammas, E. Duviella, Supervision and prognosis architecture based on dynamical classification method for the predictive maintenance of dynamical evolving systems, Reliability Engineering & System Safety 136 (2015) 120 – 131. doi:https://doi.org/10.1016/j.ress.2014.12.005.
URL http://www.sciencedirect.com/science/article/pii/S0951832014003123

[15] M. Pratama, J. Lu, E. Lughofer, G. Zhang, S. Anavatti, Scaffolding type-2 classifier for incremental learning under concept drifts, Neurocomput. 191 (C) (2016) 304–329. doi:10.1016/j.neucom.2016.01.049.
URL https://doi.org/10.1016/j.neucom.2016.01.049

[16] D. Leite, P. Costa, F. Gomide, Evolving granular neural networks from fuzzy data streams, Neural Networks 38 (2013) 1 – 16. doi:https://doi.org/10.1016/j.neunet.2012.10.006.
URL http://www.sciencedirect.com/science/article/pii/S0893608012002791

[17] E. Lughofer, M. Sayed-Mouchaweh, Autonomous data stream clustering implementing split-and-merge concepts - towards a plug-and-play ap-

proach, Inf. Sci. 304 (C) (2015) 54–79. doi:10.1016/j.ins.2015.01.010.
URL https://doi.org/10.1016/j.ins.2015.01.010

[18] E. Lughofer, Extensions of vector quantization for incremental clustering, Pattern Recogn. 41 (3) (2008) 995–1011. doi:10.1016/j.patcog.2007.07.019.
URL http://dx.doi.org/10.1016/j.patcog.2007.07.019

[19] A. Lemos, W. Caminhas, F. Gomide, Adaptive fault detection and diagnosis using an evolving fuzzy classifier, Inf. Sci. 220 (2013) 64–85. doi:10.1016/j.ins.2011.08.030.
URL https://doi.org/10.1016/j.ins.2011.08.030

[20] P. Angelov, Anomaly detection based on eccentricity analysis, in: Proc. IEEE Symposium Series in Computational Intelligence (SSCI 2014), Orlando, Florida, U.S.A., 2014, pp. 1–8.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, AAAI Press, 1996, pp. 226–231.

[22] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, I. Assent, Synchronization-based clustering on evolving data stream, Information Sciences 501 (2019) 573 – 587. doi:https://doi.org/10.1016/j.ins.2018.09.035.
URL http://www.sciencedirect.com/science/article/pii/S0020025518307400

[23] P. Kranen, I. Assent, C. Baldauf, T. Seidl, The clustree: Indexing micro-clusters for anytime stream mining, Knowl. Inf. Syst. 29 (2) (2011) 249–272. doi:10.1007/s10115-010-0342-8.
URL http://dx.doi.org/10.1007/s10115-010-0342-8

[24] C. Isaksson, M. H. Dunham, M. Hahsler, Sostream: Self organizing density-based clustering over data stream, in: Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 264–278. doi:10.1007/978-3-642-31537-4_21.
URL http://dx.doi.org/10.1007/978-3-642-31537-4_21

[25] R. Hyde, P. Angelov, A. Mackenzie, Fully online clustering of evolving data streams into arbitrarily shaped clusters, Information Sciences 382 (2016) 1–41. doi:10.1016/j.ins.2016.12.004.

[26] M. K. Islam, A buffer-based online clustering for evolving data stream, Information Sciences 489 (2019) 113–135.

[27] D. Kangin, P. Angelov, Evolving clustering, classification and regression with TEDA, in: Proc. IEEE The International Joint Conference on Neural Networks (IJCNN 2015), IEEE, 2015, pp. 1–8.

[28] B. S. J. Costa, C. G. Bezerra, L. A. Guedes, P. P. Angelov, Online fault detection based on typicality and eccentricity data analytics, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–6. doi:10.1109/IJCNN.2015.7280712.

[29] J. G. Saw, M. Yang, T. C. Mo, Chebyshev inequality with estimated mean and variance, The American Statistician 38 (2) (1984) 130–132.

[30] G. E. Cook, J. E. Maxwell, R. J. Barnett, A. M. Strauss, Statistical process control application to weld process, IEEE Transactions on Industry Applications 33 (2) (1997) 454–463. doi:10.1109/28.568010.

[31] Clustering datasets - joensuu, https://cs.joensuu.fi/sipu/datasets/, acessado em 27/01/2017 (2015).