A novel algorithm for dynamic clustering: properties and performance

Nathalie Barbosa Roa, Louise Travé-Massuyès, Victor H Grisales

A novel algorithm for dynamic clustering: properties and performance

Nathalie Barbosa Roa, Louise Travé-Massuyès, and Victor Hugo Grisales

***Abstract*—In this paper, we present a novel dynamic clustering algorithm that efficiently deals with data streams and achieves several important properties which are not generally found together in the same algorithm. The dynamic clustering algorithm operates online in two different time-scale stages, a fast distance-based stage that generates micro-clusters and a density-based stage that groups the micro-clusters according to their density and generates the final clusters. The algorithm achieves novelty detection and concept drift thanks to a forgetting function that allows micro-clusters and final clusters to appear, drift, merge, split or disappear. The outlier identification is made in a natural way using micro-clusters density. This algorithm has been designed to be able to detect complex patterns even in multi-density distributions and making no assumption of cluster convexity. The performance of the dynamic clustering algorithm is assessed theoretically through complexity analysis and empirically through a set of experiments that compare the algorithm with other algorithms of the literature on popular data sets that have interesting characteristics regarding the emergence, disappearance and movement of clusters as well as multi-density, non-convexity and noise.**

***Index Terms*—Dynamic clustering, incremental learning, online learning, multi-density, non-convex data sets.**

I. Introduction

With computer storage capacities of today, massive historical data can be recorded and stored in all the socio-economic domains : engineering, environment, finance, etc. The cloud also brings an enormously cost-effective way to increase storage. Hence, huge amounts of data, arising from various sources, are collected and they are available for further analysis. Data can indeed be a tremendous source of information and knowledge if it is mined in a smart way. Machine learning, an essential ingredient in the data mining field, currently face new challenges, in particular, building classification techniques able to handle complex data sets and scaling to big data.

Classification algorithms typically include two phases: a training phase that makes use of a training data set to generate a set of classes considered as the *model* and a recognition phase that uses the model to assign the new objects to one of the classes. The training phase may be supervised if the training data set is labeled, i.e. expertise of the domain has allowed to label each object of this set. In this case, the training phase aims at defining the shape of the classes corresponding to the different concepts existing in the labels and one refers to *supervised classification*. However, data mining and knowledge discovery generally require *unsupervised classification* schemes as the concepts of the domain may not even be (all) known a priori. In this case, also known as *clustering*, the training phase works by grouping the objects according to some predefined criteria. Every cluster is then interpreted as a class for which a concept can be assigned afterwards by an expert.

The above framework defines the *static classification* paradigm in which the model remains unchanged during the recognition phase [1]. However, in evolving environments, this paradigm is not relevant and the model needs to account for the properties of the new objects and adapt continuously. The training and recognition phases are hence interlinked and the task becomes *dynamic classification* and rises the novelty detection and concept drift problems [2], [3].

In fact, due to the arrival of new objects and the discarding of old data being irrelevant, the concepts represented by the classes may change and accordingly a dynamic classifier should be able to [4]:

- Create new clusters in order to represent novel behavior.
- Merge several clusters if the concept represented by one of them can incorporate the others.
- Split one cluster into two or more clusters if new information allow the distinction between them.
- Eliminate old and irrelevant clusters.
- Change the location of the clusters to follow data drift.

These properties will be referred as dynamic structural changes of a classifier.

Dynamic classification has been tackled by several authors. Among the main techniques we can mention: Evolving Clustering [5], [6], Self-Adaptive Feed-Forward Neural Networks (SAFN) [7], LAMDA (Learning Algorithm for Multivariable Data Analysis) [8] and Growing Gaussian Mixture Models (2G2M) [9]. Some of these alternatives are really complex, with high requirements in terms of memory and processor power and hence, not suitable for handling online large amounts of data, such as data arriving in a stream.

Different approaches to data stream clustering have been proposed since the 90's. Among these, incremental approaches have emerged as a solution for incoming data arriving at high rates. This limits the possibility of storing all the samples, and reduces the time available for processing them. Among the main techniques we can mention: CluStream [10], ClusTree [11] and DenStream [12]. These approaches use a two-phase scheme which consists of processing the raw data stream in real time producing some summary data and then using this summary data offline to generate the clusters.

N. Barbosa Roa and L. Travé-Massuyès are with LAAS-CNRS, University of Toulouse, Toulouse, France.

N. Barbosa Roa and V. Hugo Grisales are with Universidad Nacional de Colombia. Bogotá, Colombia.

In other words, the real-time component forms micro-structures characterized by a feature vector that summarizes the information of all individuals contained therein [10], [11], [12], [13], [14]. The use of this vector makes the algorithms suitable for online applications since the storage of each individual becomes unnecessary. These micro-structures may be arranged in an ordered manner (e.g. grids or trees) with the purpose of making the belonging structure location easier. The "offline" component of the algorithms takes the micro-clusters features as source data for final clustering. K-means based techniques are a common choice in the offline component [10], [11], since the one-pass only condition (for high stream rates) is not necessary in this stage (advantage given by the offline nature of the component). Density-based clustering algorithms are also used in the offline component, showing an improvement in their capacity to handle clusters of any shape including non-convex sets [12], [13]. To the best of our knowledge, no streaming clustering technique can handle the five dynamic structural changes described above.

The dynamic clustering algorithm presented in this paper is able to deal with large amounts of data arriving at fast rates by adopting a two stages strategy similar to [10], [11], [12]. The first stage is a fast scale distance-based algorithm that collects, pre-processes and compresses data samples to form so-called *micro-clusters* (μ-clusters). The second stage is a slower scale density-based algorithm that groups the μ-clusters into actual clusters that can be interpreted semantically as classes. The forgetting process implemented allows novelty detection, concept drift detection and identifies outliers in a natural way. The second, density-based, stage detects μ-clusters evolution and changes the dynamic cluster structure accordingly.

In particular, our algorithm is able to deal with multi-density clusters and can hence cope with more complex data sets that previous density-based approaches [12], [14]. The first goal of this paper is to present our algorithm and its enhancements over previous density-based approaches and the second goal is to assess its properties and performance by analyzing its theoretical complexity and by testing it empirically through a set of experiments that compare the algorithm with other algorithms of the literature on popular data sets that have interesting characteristics regarding the emergence, disappearance and movement of clusters as well as distributions exhibiting multi-density, non-convexity and noise. It is shown that the dynamic clustering algorithm proposed in this paper achieves several important properties which are not generally found together in the same algorithm.

The remainder of the paper is organized as follows. Section II presents the distance/density-based two-stages clustering algorithm and the novelty detection, emphasizing the multi-density extension. Section III introduces other properties of the algorithm and assesses the performance of the algorithm by illustrating the results with several known data sets. Section IV position our work with respect to the main references in streaming clustering using the introduced data sets. Finally, section V concludes the paper and outlines future research directions.

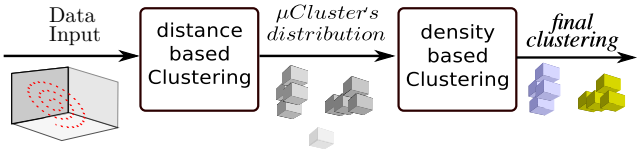


Fig. 1: Global description of the algorithm

II. Two Stages Dynamic Clustering

This paper uses the two stages distance- and density-based clustering approach proposed in [14] modified to be able to discover clusters of different densities. In our proposal both stages work on-line, but operate at different time scales. In addition, μ-clusters of similar densities can form clusters of any shape and any size. This multi-density feature allows the detection of novelty behavior in its early stages when only a few objects giving evidence of this evolution are present.

The first stage operates at the rate of the data stream and creates μ-clusters putting together data samples that are close, in the sense of a given distance, to each other. μ-clusters are stored in the form of summarized representations including statistical and temporal information.

The second stage takes place once each t_{slow} seconds and analyses the distribution of μ-clusters. The density of a μ-cluster is considered as low, medium or high and is used to create the final clusters by a density based approach, i.e. dense μ-clusters that are close enough (connected) are said to belong to the same cluster. Similarly to [13], a cluster is defined as the group of connected μ-clusters where every inside μ-cluster presents high density and every outside μ-cluster exhibits either medium or low density. The above dense μ-cluster structure allows the algorithm to create clusters of non convex shapes even in high dimensional spaces and it has proved outliers rejection capabilities in evolving environments ([12], [14]).

The two stages are further explained in II-A and II-B. The multi-density proposal is explained in II-C.

A. Distance-based clustering stage

Considering a d-dimensional object E qualified by d features, a μ-cluster gathers a group of data samples *close in all dimensions* and whose information is summarized in a characteristic *feature vector* (CF). For a μ-cluster μC_k, CF has the following form:

$$CF_k = (n_k, LS_k, SS_k, t_{lk}, t_{sk}, D_k, Class_k) \qquad (1)$$

where $n_k \in \Re$ is the number of objects in the μ-cluster k, $LS_k \in \Re^d$ is the vector containing the linear sum of each feature over the n_k objects, $SS_k \in \Re^d$ is the square sum of features over the n_k objects, $t_{lk} \in \Re$ is the time when the last object was assigned to that μ-cluster, $t_{sk} \in \Re$ is the time when the μ-cluster was created, D_k is the μ-cluster density and $Class_k$ is the μ-cluster label if known. Using LS_k, SS_k and n_k the variance of the group of objects assigned to the μ-cluster k can be calculated.

The μ-cluster is shaped as a d-dimensional box since the L1-norm is used as distance measure. The distance between an object $E_x = [x^1, \ldots, x^d]^T$ and a μ-cluster μC_k is calculated as the sum of the distances between the vector $LS_k = [c_k^1, \ldots, c_k^d]^T$ and the object value for each feature as shown in equation (2):

$$dis(E_x, \mu C_k) = \sum_{i=1}^{d} \left| x^i - c_k^i \right| \tag{2}$$

The size of the boxes S_k^i along each dimension d_i is set as a fraction of the feature range. This fraction can be established according to the data *context*, i.e. min and max values of the feature, which is used for normalization purposes. If no context is available in advance, it may be established online. The box size for the feature along dimension d_i is found according to (3), where ϕ_i is a constant establishing the fraction:

$$S_k^i = \phi_i |max_i - min_i|, \qquad \forall i = 1, \ldots, d. \tag{3}$$

Whenever an object E arrives, the algorithm searches for the closest μ-cluster. Once found, a maximal distance criterion is evaluated to decide whether or not the object fits inside the μ-cluster hyper-box. If the fitting is sufficient the μ-cluster feature vector is updated using the object information; if not, a new μ-cluster is created with the object information using its time-stamp as cluster time of creation.

B. Density-based clustering

In this stage, density-based clustering is executed over the μ-clusters, which allows the algorithm to find clusters of arbitrary shape. A μ-cluster may be of three different types: dense μ-cluster ($D\mu$-cluster), semi-dense μ-cluster ($S\mu$-cluster) and low density or outlier μ-cluster ($O\mu$-cluster). The difference between each type is established based on dynamic thresholds found locally (see subsection II-C). $S\mu$-clusters may result from an increase in the number of outliers from which one can expect a cluster creation, so they have to be updated more frequently than other μ-clusters.

The density of a μ-cluster μC_k is calculated using the current number of objects n_k and the current hyper-volume of the bounding box $V_k = \prod_{i=1}^{d} S_k^i$, as shown in (4):

$$D_k = \frac{n_k}{V_k}. \tag{4}$$

To find the final clusters, the dense character of a μ-cluster and its neighbors is analyzed with some periodicity. Let μC_{k_α} and μC_{k_β} be two μ-clusters, then μC_{k_α} and μC_{k_β} are said to be *directly connected* if their hyper-boxes overlap in all but φ dimensions. The parameter φ establishes the feature selectivity of the classifier.

A μ-cluster μC_{k_1} is said to be connected to μC_{k_n} if there exists a chain of μ-clusters $\{\mu C_{k_1}, \mu C_{k_2}, \cdots, \mu C_{k_n}\}$ such that μC_{k_i} is directly connected to $\mu C_{k_{i+1}}$ for $i = 1, 2, \cdots, n-1$. A set of connected μ-clusters is said to be a *group*. Groups of μ-clusters are analyzed recursively. A cluster is created if every inside μ-cluster of a group is a $D\mu$-cluster and every border μ-cluster is either a $D\mu$-cluster or an $S\mu$-cluster. Once a cluster is create, the rest of the group is analyzed in turn. $O\mu$-clusters then correspond to the local lowest density samples and do not contribute to the final clusters.

Once the clustering distribution is found, a copy of the μ-clusters distribution is stored as a snapshot that may be examined in the future in order to extract more information about the system evolution. Snapshots are stored following a pyramidal time scheme as the one proposed in [10]. This storage efficiently stacks past snapshots according to several sampling times.

C. Extension to Multi-density Clustering

Density based algorithms as those from [15], [12], [13], [14] group data samples according to their density. The connection between dense samples is key in forming clusters with arbitrary shapes. Nevertheless, in these implementations, the concept of 'dense' is related to a global threshold. The problem in taking a global threshold to identify a sample as dense appears when regions with varied densities are present in the same dataset as can be seen in figure 2. In this case, the above algorithms may misclassify low density samples as noise, as can be seen in figure 2 where DBSCAN fails to detect the two clusters with lower density.

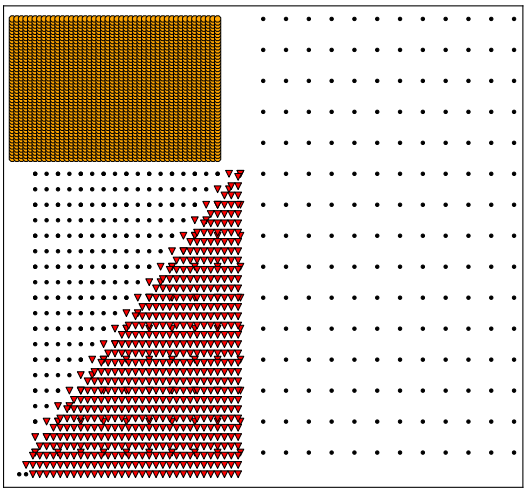


Fig. 2: DBSCAN fails to detect the four different clusters, instead it provides only the highest density clusters (Yellow and Red) and outliers

While it is deemed desirable to be able to learn and recognize clusters of multiple densities, it is also important to maintain the ability to reject outliers. Our solution to multi-density clustering is based on local analysis. Differently from [15], [12], [14], we analyze the dense character of each μ-cluster with respect to the density of the other μ-clusters in the same group.

Two measures are considered as representative of the μ-clusters density distribution in a local sense, the average of the μ-cluster's density in the group and the median. These measures work as thresholds for establishing the dense character of each μ-cluster in the group. The intuition behind the selection of these measures is that the median and average densities of

an heterogeneous group are significantly different, although, if the group is uniformly dense, these two quantities are equal.

In other words, for each group G_k, the μ-clusters having their density higher than or equal to the average density of the group ($avg(D_{G_k})$) and higher than or equal to the median density of the group ($median(D_{G_k})$) are considered as dense. μ-clusters having a density higher than or equal to only one of those measures (either average or median) are considered as $S\mu$-clusters and those with density below both measures are considered as $O\mu$-clusters. Summarizing :

$$\mu C_k\text{dense} \Leftrightarrow D_k \geq median\left(D_{G_k}\right) \wedge D_k \geq avg\left(D_{G_k}\right), \quad (5)$$

$$\mu C_k\text{semi-dense} \Leftrightarrow D_k \geq median\left(D_{G_k}\right) \vee D_k \geq avg\left(D_{G_k}\right), \quad (6)$$

$$\mu C_k\text{outlier} \Leftrightarrow D_k < median\left(D_{G_k}\right) \wedge D_k < avg\left(D_{G_k}\right). \quad (7)$$

Interestingly, using the multi-density scheme helps to better shape clusters that share frontiers, that is, clusters that are side by side, in highly overlapping situations as will be shown in the section III.

D. Novelty detection and cluster evolution

In order to maintain an up-to-date model, more weight should be given to newly incoming data. In our algorithm, μ-clusters are weighted with a decay function dependent on the current time t and the last assignment time t_{lk}. The function is chosen to be $\beta^{-\lambda(t-t_{lk})}$ to emulate an ageing process over a damped window.

When a new d-dimensional object $E = [e_{d_1}, \ldots, e_{d_i}]$, with $i = \{1, \ldots, d\}$ is assigned to a μ-cluster μC_k at t, $t_{lk} = t$. The other attributes of the feature vector are updated as follows:

$$n_k^{(t)} = n_k^{(t-1)}\beta^{-\lambda(t-t_{lk})} + 1 \quad (8)$$

$$LS_{k,d_i}^{(t)} = LS_{k,d_i}^{(t-1)}\beta^{-\lambda(t-t_{lk})} + e_{d_i} \quad \forall\ d_i,\ i = 1, \ldots, d. \quad (9)$$

$$SS_{k,d_i}^{(t)} = SS_{k,d_i}^{(t-1)}\beta^{-\lambda(t-t_{lk})} + e_{d_i}^2 \quad \forall\ d_i,\ i = 1, \ldots, d. \quad (10)$$

Generalizing, if no object is added to a μ-cluster during the time interval $(t, t + \Delta t)$, its CF at $t + \Delta t$ can be calculated from $CF^{(t)}$ using the decay function for the weighted parameters as follows:

$$CF^{(t+\Delta t)} = \beta^{(-\lambda\Delta t)}CF^{(t)} \quad (11)$$

If the object cannot be assigned to any of the existing μ-clusters, a new $O\mu$-cluster is created using the object information as model. We assume that μ-clusters with low density ($O\mu$-cluster), are either outliers or potential clusters in an emerging state. The later case reveals itself with an increment in the cluster density and consequently, this μ-cluster grows into an $S\mu$-cluster as new data arrives.

E. Dynamic clustering algorithm operation with grid indexing

When a new identified object E arrives, the algorithm uses a grid indexing algorithm. The feature space is assumed to be partitioned in the form of a grid, in which each cell is identified by a 3 bits (per dimension) index. The algorithm maps the object information into the grid and verifies the existence of μ-clusters in the corresponding cell. If no μ-cluster exists, it creates a μ-cluster using the data of the object and indexes it with the cell mapping coordinates.

If there are already one or more μ-clusters in that cell, our algorithm finds out which $D\mu$-cluster or $S\mu$-cluster is the closest. The Manhattan distance is used as similarity measure. Once the closest μ-cluster is found, the maximal distance condition is verified to asses whether the object fits inside the maximal possible bounding box MAX_{BB} centered on the μ-cluster. If there is a fit, the object is absorbed by the μ-cluster.

Otherwise our algorithm tries to assign the object to the closest $O\mu$-cluster in the same cell. When the object is assigned, the density of the $O\mu$-cluster is calculated to verify whether or not it has grown into an $S\mu$-cluster. When an $O\mu$-cluster becomes denser it is removed from the outlier list and inserted as $S\mu$-cluster into the active μ-cluster list.

The second stage analyses the μ-cluster density checking for changes. $O\mu$-clusters are evaluated to find out if their density is below a low-density threshold. If that is the case the $O\mu$-cluster is eliminated and no longer considered. Once filtered, the remaining active μ-clusters are analyzed looking for groups. For each group the local density analysis is performed recursively to find all possible clusters and detect local outliers. Locally, the algorithm checks each $S\mu$-cluster to find out whether its density has decreased below the semi-dense threshold. If this is the case, the $S\mu$-cluster is eliminated and with its statistics an $O\mu$-cluster is created. This update may create new low density regions inside a cluster, changing its form and even causing it to split into two or more clusters, following data evolution. In the same way, if inside a group some $O\mu$-clusters have grown into $S\mu$-clusters, two clusters that were separated may be merged.

III. Assessing the properties and performance of the algorithm

In this section the algorithm is analyzed, then tested and compared to other known algorithms to assess its performance.

A. Complexity Analysis

Since the algorithm has two stages that work independently, its complexity can be evaluated independently for the two stages. The complexity of the distance-based stage is $O(dM)$ for each incoming d-dimensional object, where M is the number of μ-clusters. Since the algorithm is incremental, in general $M << n$, n being the number of elements to cluster, which leads us to a global complexity of $O(n)$. It is worth noticing that in the exceptional case of small high dimensional datasets with sparse distributions, the inequalities $M << n$ and $d << n$ do not stand any more. In consequence, in these exceptional cases, the complexity becomes $O(dMn)$.

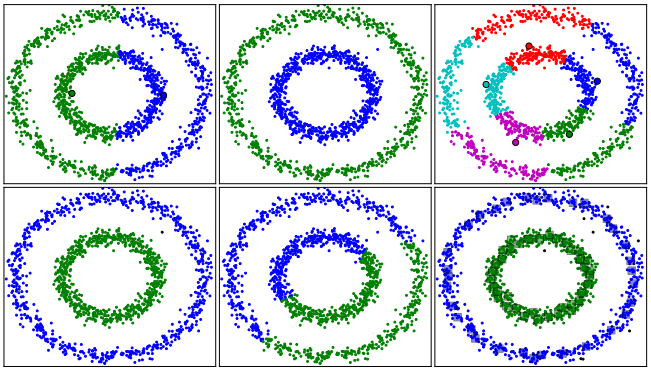


Fig. 3: Comparison of six algorithms for the concentric circles data set. Top left to right: MiniBatch KMeans, Agglomerative Clustering, Affinity Propagation. Bottom: DBSCAN, BIRCH, our algorithm

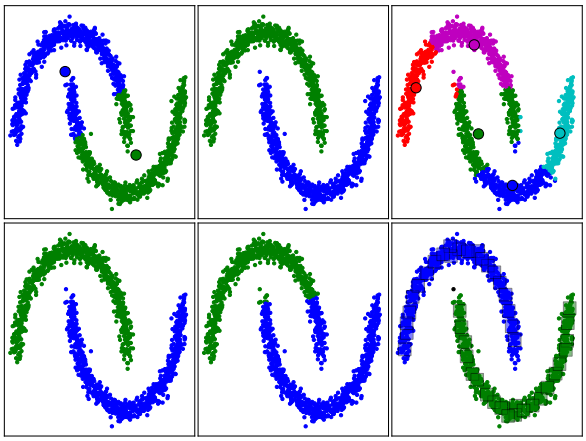


Fig. 4: Comparison of six algorithms for the moons data set. Top left to right: MiniBatch KMeans, Agglomerative Clustering, Affinity Propagation. Bottom: DBSCAN, BIRCH, our algorithm

Nevertheless, small datasets are not the objective of this study. The complexity of the density based stage is $O\left(dM^2\right)$.

B. Algorithm comparison with literature synthetic test cases

The algorithm was faced with several test cases presented in the literature. In Figures 3 to 7, we show the algorithm's clustering results for different test cases compared with the results achieved by those proposing the test. Cluster are represented using a combination of colors and glyph. Samples that belong to the same cluster have both the same tone and the same glyph.

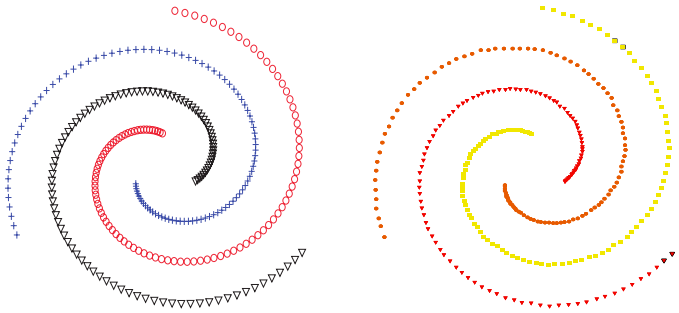
Test cases will show that our algorithm performance is up to the performance of several different clustering algorithms and achieves several important properties which are not generally found together in the same algorithm.

1) Detection of non-convex distributions: The clustering method implemented in our algorithm can detect convex and non convex distributions. To show this feature we have chosen to cluster some synthetic sets available in the *scikit-learn* Python module [16]. Our algorithm is compared to scikit-learn module provided implementations of different clustering algorithms, namely MiniBatch KMeans (MBK-m), Agglomerative Clustering (AC), Affinity Propagation (AP), DBSCAN [15] and BIRCH [17].

Two noisy data sets were evaluated named concentric circles and moons. In each data set 1500 samples were considered. These data sets were generated using the scikit-learn.dataset module. The distributions are time invariant. In consequence, the forgetting process of our algorithm μ-clusters is disabled for this test. [1]

The results of the tests for the concentric circles data set can be seen in Figure 3 and for the moons data set in Figure 4. In the figures the cluster centers found by MBK-m and AP are drawn as colored circles for illustrative purposes as well as our algorithm $D\mu$-clusters (colored squares). MBK-m, AC and BIRCH require the number of clusters as a initial parameter but even with this information, MBK-m and BIRCH are not able to cluster these non convex sets properly. AP does not perform well at all in these distributions. Moreover, it creates a high number of clusters. Our algorithm is able to detect non convex distributions as well as DBSCAN does, since they are both density based. Nevertheless, the test shows that our algorithm rejects more outliers than DBSCAN.

2) Robust path based clustering: Figure 5 shows the three spirals distribution used in [18]. In the centre of each spiral, samples are more abundant and then they become sparser as spirals grow out. This kind of distribution is path based and it is particularly difficult to handle for clustering algorithms only based on distance or only based on density. Since our algorithm uses both, distance and density it can overcome this kind of challenge. It indeed achieves results comparable to those of the original article presented by Chang and Yeung [18] obtained with their robust path-based spectral clustering method as can bee seen in Figure 5.



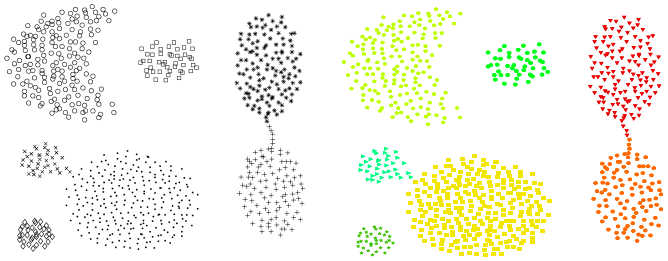
(a) Results in [18] (b) Our results

Fig. 5: Robust path-based spectral clustering and our algorithm tested against the three spirals distribution

3) Clustering aggregation problem: Figure 6 shows the test case used by Gionis *et al.* in its clustering aggregation problem [19]. An intuitively good clustering for this dataset consists of

[1]The algorithm parameters for the test were: MBK-m (# of clusters), AC (linkage `"average"`, affinity `"cityblock"`, # of clusters, connectivity (estimated using n_neighbors=10.)), AP (damping=0.9, preference=-200), BIRCH (# of clusters), DBSCAN (eps=0.2), our algorithm (box relative size=0.06).

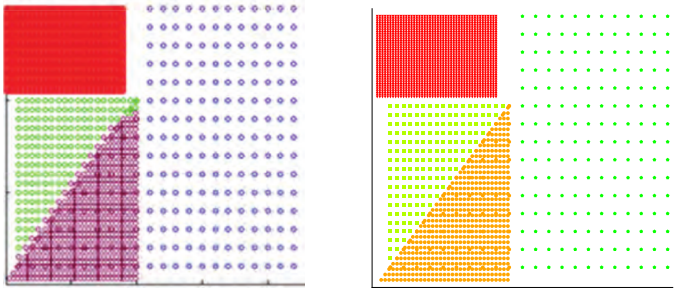
the seven perceptually distinct groups of objects. The authors in [19] ran five different clustering algorithms implemented in MATLAB (single linkage, complete linkage, average linkage, Ward's clustering, and k-means), setting the number of clusters to 7 in each case. None of these five algorithms obtained an appropriate cluster distribution. As seen in Figure 6, our algorithm clusters correctly the test set, achieving by itself the result that Gionis *et al.* achieve aggregating the clustering results of five different algorithms. In addition, our algorithm does not need the number of clusters as input parameter.



(a) Results in [19] (b) Our results

Fig. 6: Gionis *et al.* clustering aggregation results and our algorithm clustering results for the clustering aggregation problem

4) Clustering multi-density distributions: As explained in subsection II-C, our algorithm is capable to cluster multi-density distribution thanks to its local density analysis. To probe this we take the test case introduced in subsection II-C that exhibits varied-density clusters with no separation among them. The original results of [20] and our results are shown in Figure 7.



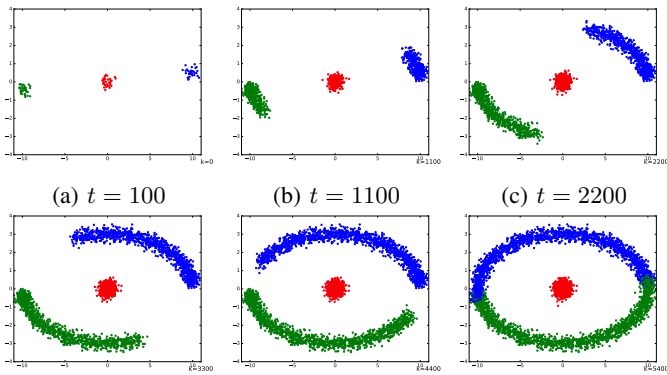
(a) Results in [20] (b) Our results

Fig. 7: Results on varied-density clusters with no separation among them.

The multi-density scheme that we propose helps to better shape clusters that share frontiers. Figure 7 shows that cluster borders are better shape in our results (right) that in the original results of [20] (left). To achieve this, our algorithm finds all possible cluster, and then it analyzes every μ-cluster in the border of the clusters. These μ-clusters are assigned to the connected cluster that has the most similar average density. In that way clusters frontiers can be precisely drawn which is key in highly overlapping distributions.

C. Concept drift problem

In this section, our algorithm is confronted with real concept drift in time varying distributions. Concept drift refers to a subtle change of the underlying data distribution that may (or may not) also involve change in the conditional distribution of the target concept. In order to illustrate this concept, three clearly differentiated distributions are used to form three clusters. Cluster one is represented in green cluster two in blue and cluster three in red in Figure 8. Cluster one and two drift in time until shifting positions. Synthetic data are generated changing the center of the distribution each 100 samples.

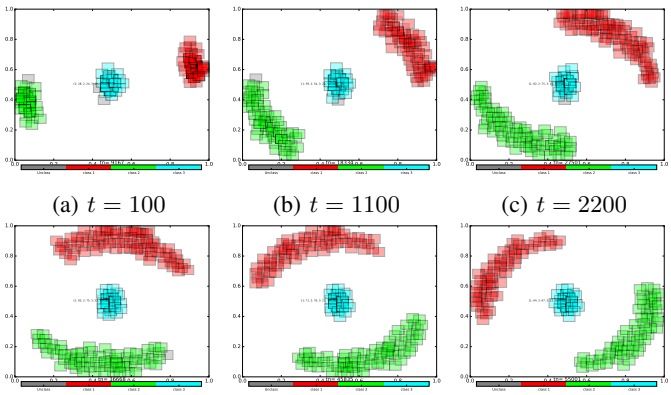


(a) $t = 100$ (b) $t = 1100$ (c) $t = 2200$

(d) $t = 3300$ (e) $t = 4400$ (f) $t = 5400$

Fig. 8: Concept drift toy example

Snapshots showing the distribution of μ-clusters (little boxes) and clusters (same color) found by our algorithm at several time instants are depicted in Figure 9. It can be seen how clusters evolution is tracked thanks to the drift of some of the existent μ-clusters and to the creation of new μ-clusters. Growth in the amount of clusters is particularly visible between snapshots one and two, and again between snapshots two and three. $O\mu$-clusters are represented as gray boxes.



(a) $t = 100$ (b) $t = 1100$ (c) $t = 2200$

(d) $t = 3300$ (e) $t = 4400$ (f) $t = 5400$

Fig. 9: μ-clusters (little boxes) and clusters (same color) distribution obtained with our algorithm for the concept drift toy example

The values used in this experiment were $\beta = 2$ (value par default) and $\lambda = 1/500$. In fact, if β is chosen as 2, the amount

of time required for a cluster to forget half of the known values (half-life) is $\frac{1}{\lambda}$, $\lambda > 0$. The frequency of the second stage is an parameter that can be provided by the user. The default value is defined as 1/100 samples.

It is worth mentioning that, between the algorithm parameters, the forgetting factor is the one that influences the most the global performance. This factor is directly related with cluster reactivity. It should be high enough to assure that micro clusters will be sensitive to distribution changes, but, low enough to avoid over sensitivity, that may reduce the outlier rejection capability. In fact, it should be guarantee that $t_s low < 1/\lambda$ in order to avoid data lost before being clustered. With respect to memory usage, if the forgetting factor is too low, memory issues may arise.

IV. Discussion

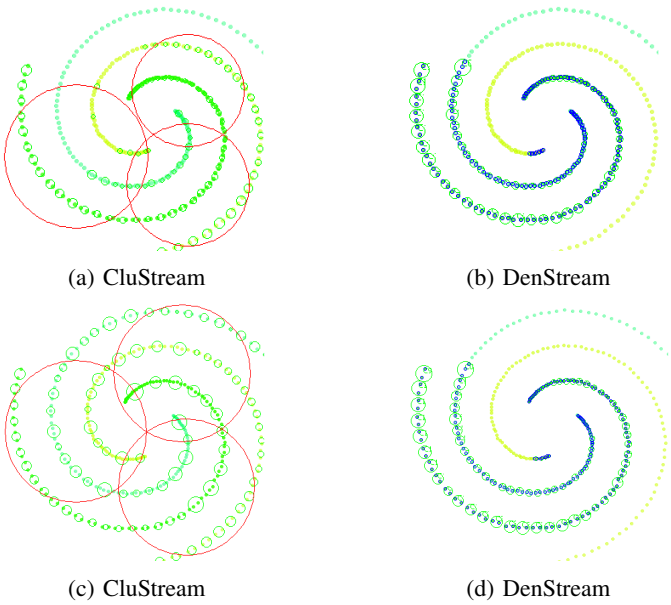
The toy data sets and the generated concept drift data set were selected upon their ability to show complex problems that may arise in real life. Path based clustering, for example, recreates the intuitive reasoning that two objects should be assigned to the same cluster if they can be connected by a mediating path of objects. However, in the best of our knowledge path based clustering is not achieved in dynamic clustering algorithms (stream based or not).

The clustering aggregation problem and the multi-density problem test the ability of clustering overlapping distributions in which densities may vary, as is the case of industrial processes where samples of normal behavior will be more frequent that samples of faulty behavior. Quick detection of faulty behavior and drift of the normal state are also essential.

To test concept drift, an artificial data set was created because, as stated in [21] and [22], an influential problem in most of the real-world data sets is that concept drift manifest over very long periods of time and hence results in huge data sets that are not freely available.

For the test shown in the previous section, the forgetting processes were disabled in the static data sets, thus, only the concept drift experiment was subject to this process. We have shown that our algorithm can deal with these complex problems and in this section we also show that well known stream algorithms cannot handle them. We used the implementations of Clustream [10] and DenStream [12] available in the Massive online analysis open source framework [23].

We chose the robust path based test case shown in subsection III-B2 as first example. The CluStream and Denstream algorithms were confronted to this data set in two scenarios. In the first scenario the time horizon was set to a third of the data length, allowing the algorithms to forget the oldest samples. In the second scenario the time horizon was chosen in order to include all the samples in the dataset. Results for the first scenario are shown at the top of Figure 10. CluStream micro-clusters in Figure 10a are depicted in green and the final cluster result in red. It can be seen that CluStream actually creates the micro-cluster representation for the samples but the final clusters mix samples of the three classes. DenStream micro-clusters are depicted in green in Figure 10b, and the final clusters are represented changing points color to blue.



(a) CluStream (b) DenStream

(c) CluStream (d) DenStream

Fig. 10: Clustering results of streaming algorithms for the path-based test case. Screenshot from MOA software [23]. Neither of these algorithm achieve correct classification. Time horizon was set to 100 samples (a and b) and 300 samples (c and d)

TABLE I: Clustering evaluation of streaming methods over the path-based test case

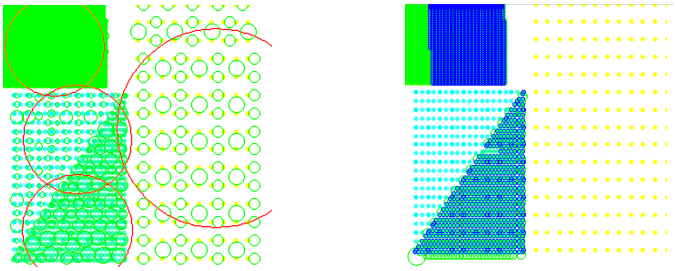
Algorithm	Purity	Precision	Recall	NumClusters
Clustream	0.43	1.0	0.82	3
Denstream	1.0	1.0	0.6	32
Our proposal	1.0	1.0	1.0	3

DenStream, on the other hand, does not mix elements of several classes, but it creates 32 clusters putting apart samples of the same.

The second scenario is the one giving the same importance to all the samples, this scenario was also the test scenario for our algorithm. Clustering results of CluStream and DenStream for this scenario are shown in the bottom of Figure 10. We can see that both algorithms fail again in finding the correct clustering. DenStream results do not change between both scenarios. Table I summarizes the clustering results for the tested algorithms.

As second example, we analyze the streaming algorithm results in the case of multi-density distributions. CluStream and DenStream results for the test case introduced in [20] are shown in figure 11. Once again the tested algorithms fail to detect the data classes correctly. DenStream results are similar to those achieved by DBSCAN shown in Figure 2, that is, it can only detect the two denser classes. Even if in this example the target classes are convex sets, CluStream fails to correctly cluster them due mainly to the overlapping of the distributions. Table II summarize the clustering results.

Dynamic data cause several clustering algorithms to fail because they can not cope with evolution. To illustrate this let us consider the concept drift example of the previous section (Figure 8) and evaluate the clustering performance of



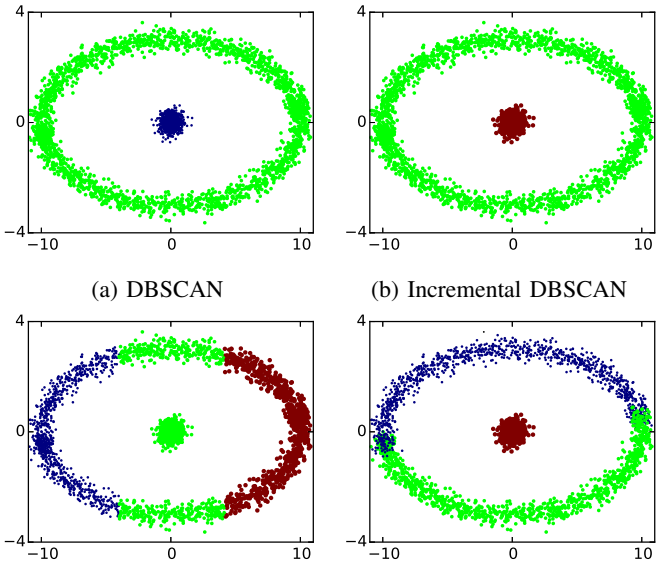
(a) CluStream (b) DenStream

Fig. 11: Clustering results of streaming algorithms for the multi-density test case introduced in [20]. Neither of these algorithm achieve the correct classification. Screenshot from MOA software [23].

TABLE II: Clustering evaluation of streaming methods over the multi-density test case

Algorithm	Purity	Precision	Recall	NumClusters
Clustream	0.83	1.0	0.82	4
Denstream	0.70	1.0	0.69	2
Our proposal	1.0	1.0	1.0	4

several algorithms. Let us start with the DBSCAN algorithm implementation found in the scikit-learn project [16]. DBSCAN clustering results[2] are shown in Figure 12a. Since the DBSCAN implementation does not consider time evolution, it mixes samples of classes 1 (green) and 2 (blue) into one class that is depicted in Figure 12a in green.



(a) DBSCAN (b) Incremental DBSCAN

(c) Mini Batch K-means (d) Our algorithm

Fig. 12: Clustering results of four different algorithms for the concept drift example. Only our algorithm can track the concept drift.

An incremental implementation of DBSCAN was also tested with this dataset. The clustering result can be seen in Figure 12b. Incremental DBSCAN results are similar to DBSCAN results. We also tested the MiniBatchKMeans algorithm, an alternative online implementation of k-means that does incremental updates of the center positions using mini-batches. The implementation of this algorithm can be found in [16]. The number of clusters were given to the algorithm and also the size of the batch. Its results are shown in Figure 12c. Since one of the cluster's distribution is time invariant (cluster 3, in the middle of the figure), MiniBatchKMeans links some samples of the clusters drifting to that cluster, losing clusters' evolution. The final clustering results of this dataset provided by our algorithm are shown in Figure 12d for comparative purposes. It is clear that even if the incremental DBSCAN and MiniBatchKMeans are online batch implementations of clusterers, they do not properly follow the evolution. The efficiency of our algorithm with respect to the concept drift problem, in particular real concept drift [3], compared to these algorithms is clearly illustrated by this experiment.

[2]Parameters for DBSCAN test: $min\_samples = 5$ and $eps = 0.2$.

V. Conclusions

In this paper, a novel algorithm for dynamic clustering, its properties and performance have been explored. The performance of the algorithm is assessed theoretically through complexity analysis and empirically through a set of comparative experiments using popular data sets. The presented dynamic clustering method bases its structure on micro-clusters allowing the handle of non-convex, multi-density clustering with outlier rejection even in highly overlapping situations. This approach to unsupervised learning implements a local-density analysis that allows to detect rare, infrequent behaviors improving novelty detection. This is possible since, with only a few objects, new classes can be characterized and recognized. The algorithm was compared with several different clustering algorithms, both static and dynamic, and it has proved to achieve similar or better results than several of them, hence pushing forward the state of the art. Even more, this proposal has proved to exceed the performance of well-known distance-based and density-based streaming algorithms. The dynamic clustering approach presented in this paper has shown excellent results in presence of concept drift. Future works will include dynamic clustering of dependent and auto-correlated time series, investigating a proper representation of temporal information, and achieving dynamic clustering of multiple time-scale changing patterns.

References

[1] A. Joentgen, L. Mikenina, R. Weber, and H. Zimmermann, "Dynamic fuzzy data analysis based on similarity between functions," *Fuzzy Sets and Systems*, vol. 105, no. 1, pp. 81–90, 1999.

[2] M. Markou and S. Singh, "Novelty detection: a review–part 1: statistical approaches and part 2: neural network based approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[4] L. Angstenberger, *Dynamic fuzzy pattern recognition with applications to finance and engineering*. Springer, 2001.

[5] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 6, pp. 1462–1475, 2008.

[6] P. Angelov, "Fuzzily connected multimodel systems evolving autonomously from data streams," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 4, pp. 898–910, 2011.

[7] K. Li, F. Yao, and R. Liu, "An online clustering algorithm," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, vol. 2. IEEE, 2011, pp. 1104–1108.

[8] T. Kempowsky, A. Subias, and J. Aguilar-Martin, "Process situation assessment: From a fuzzy partition to a finite state machine," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 5, pp. 461–477, 2006.

[9] A. Bouchachia and C. Vanaret, "Incremental learning based on growing gaussian mixture models," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2, IEEE. Elsevier, 2011, pp. 47–52.

[10] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[11] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: indexing micro-clusters for any stream mining," *Knoledge and information systems*, vol. 29, no. 2, pp. 249–272, 2011.

[12] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SDM*, 2006.

[13] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and data mining*, 2007, pp. 133–142.

[14] N. Barbosa, L. T. Massuyes, and V. H. Grisales, "A data-based dynamic classification technique: A two-stage density approach," in *SAFEPROCESS 2015, Proceedings of the 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*. IFAC, 2015, pp. 1224–1231.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databdata with noise," in *KDD*, 1996.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141–182, 1997.

[18] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, 2008.

[19] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, march 2007. [Online]. Available: http://doi.acm.org/10.1145/1217299.1217303

[20] A. Fahim, A.-E. Salem, F. Torkey, M. Ramadan, G. Saake *et al.*, "Scalable varied density clustering algorithm for large datasets," *Journal of Software Engineering and Applications*, vol. 3, no. 06, p. 593, 2010.

[21] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.

[22] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information fusion*, vol. 9, no. 1, pp. 56–68, 2008.

[23] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1859903