

# Efficiency of Clustering Data Streams Based on Micro-Clusters Shared Density

Avula Chitty

Department of CSE, Assistant Professor, Sri Indu College of Engineering And Technology, Hyderabad,  
Telangana, India

## ABSTRACT

As more and a lot of applications produce streaming information, clustering knowledge streams has become a very important technique for data and information engineering. A typical approach is to summarize the information stream in time with an online method into an oversized number of therefore known as micro-clusters. Micro-clusters represent native density estimates by aggregating {the information} of the many data points in an outlined area. On demand, a (modified) typical bunch formula is used in a very second offline step to recluster the micro-clusters into larger final clusters. For reclustering, the centers of the micro-clusters are used as pseudo points with the density estimates used as their weights. However, data concerning density within the area between micro-clusters isn't preserved within the on-line process and reclustering relies on probably inaccurate assumptions concerning the distribution of knowledge inside and between micro-clusters (e.g., uniform or Gaussian). This paper describes DBSTREAM, the primary micro-cluster-based on-line bunch part that expressly captures the density between micro-clusters via a shared density graph. The density data during this graph is then exploited for reclustering supported actual density between adjacent micro-clusters. We have a tendency to discuss the house and time complexness of maintaining the shared density graph. Experiments on a good vary of artificial and real knowledge sets highlight that mistreatment shared density improves bunch quality over alternative popular knowledge stream bunch ways that need the creation of a bigger variety of smaller micro-clusters to realize comparable results.

**Keywords:** Data Mining, Data Stream Clustering, Density-Based Clustering

## I. INTRODUCTION

CLUSTERING data streams has become an important technique for data and knowledge engineering. A data stream is an ordered and potentially unbounded sequence of data points. Such streams of constantly arriving data are generated for many types of applications and include GPS data from smart phones, web click-stream data, computer network monitoring data, telecommunication connection data, readings from sensor nets Stock quotes, etc. Data stream clustering is typically done as a two-stage process with an online part which summarizes the

data into many micro-clusters or grid cells and then, in an offline process, these micro-clusters (cells) are recluster/merged into a smaller number of final clusters. Since the reclustering is an offline process and thus not time critical, it is typically not discussed in detail in papers about new data stream clustering algorithms. Most papers suggest to use a where the micro-clusters are used as pseudo points. Another approach used in Den Stream is to use reach ability where all micro-clusters which are less than a given distance from each other are linked together to form clusters. Grid-based algorithms typically merge adjacent dense grid cells to form larger clusters.

Current reclustering approaches completely ignore the data density in the area between the micro-clusters (grid cells) and thus might join micro-clusters (cells) which are close together but at the same time separated by a small area of low density. To address this problem, Tu and Chen introduced an extension to the grid-based D-Stream algorithm based on the concept of attraction between adjacent grids cells and showed its effectiveness. In this paper, we develop and evaluate a new method to address this problem for micro-cluster-based algorithms. We introduce the concept of a shared density graph, which explicitly captures the density of the original data between micro-clusters during clustering, and then show how the graph can be used for reclustering micro-clusters. This is a novel approach since instead on relying on assumptions about the distribution of data points assigned to a micro-cluster (MC) (often a Gaussian distribution around a center); it estimates the density in the shared region between micro-clusters directly from the data. To the best of our knowledge, this paper is the first to propose and investigate using a shared-density-based reclustering approach for data stream clustering.

## II. RELATED WORK

Density-based clustering is a well-researched area and we can only give a very brief overview here. DBSCAN [10] and several of its improvements can be seen as the prototypical density-based clustering approach. DBSCAN estimates the density around each data point by counting the number of points in a user-specifies-neighbourhood and applies user-specified thresholds to identify core, border and noise points. In a second step, core points are joined into a cluster if they are density-reachable (i.e., there is a chain of core points where one falls inside the eps-neighborhood of the next). Finally, border points are assigned to clusters.

Other approaches are based on kernel density estimation (e.g., DENCLUE [11]) or use shared nearest neighbours (e.g., SNN [12], CHAMELEON

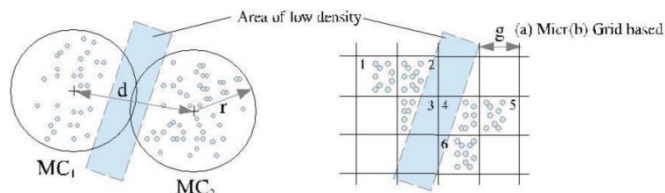
[13]). However, these algorithms were not developed with data streams in mind. A data stream is an ordered and potentially unbounded sequence of data points  $X = \{x_1; x_2; x_3; \dots\}$ . It is not possible to permanently store all the data in the stream which implies that repeated random access to the data is infeasible. In addition, data streams exhibit concept drift over time where the position and/or shape of clusters changes, and new clusters may appear or existing clusters disappear. This makes the application of existing clustering algorithms difficult. Data stream clustering algorithms limit data access to a single pass over the data and adapt to concept drift. Over the last 10 years, many algorithms for clustering data streams have been proposed [5], [6], [8], [9], [14], [15], [16], [17], [18], [19], and [20]. Most data stream clustering algorithms use a two-stage online/offline approach [4]:

**1) Online:** Summarize the data using a set of  $k_0$  micro clusters organized in a space-efficient data structure, which also enables fast lookup. Micro-clusters are representatives for sets of similar data points and are created using a single pass over the data (typically in real time when the data stream arrives) cluster centres and additional statistics as weight (density) and dispersion (variance) typically represent .Micro-clusters. Each new data point is assigned to its closest (in terms of a similarity function) micro-cluster. Some algorithms use a grid instead and non-empty grid cells represent micro clusters (e.g., [8], [9]). If a new data point cannot be assigned to an existing micro-cluster, a new micro cluster is created. The algorithm might also perform some housekeeping (merging or deleting microclusters) to keep the number of micro-clusters at manageable size or to remove noise or information outdated due to concept drift.

**2) Offline:** When the user or the application requires a clustering, the  $k_0$  micro-clusters are replastered into  $k$  ( $k \geq k_0$ ) final clusters sometimes referred to as macro-clusters. Since the offline part is usually not regarded time critical, most researchers only state

that they use a conventional clustering algorithm (typically k-means or a variation of DBSCAN [10]) by regarding the micro-cluster center positions as pseudo-points. The algorithms are often modified to take also the weight of micro-clusters into account.

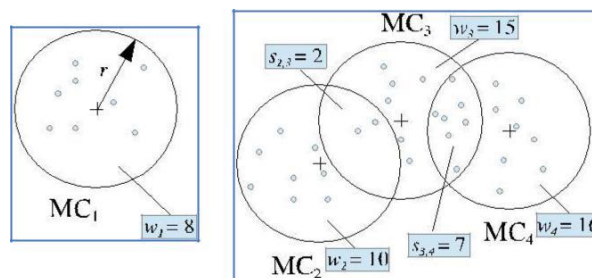
However, these algorithms were not developed with data streams in mind. A data stream is an ordered and potentially unbounded sequence of data points  $X = \{x_1; x_2; x_3; \dots\}$ . It is not possible to permanently store all the data in the stream, which implies that repeated random access to the data is infeasible. Also, data streams exhibit concept drift over time where the position and/or shape of clusters changes, and new clusters may appear or existing clusters disappear. This makes the application of existing clustering algorithms difficult. Data stream clustering algorithms limit data access to a single pass over the data and adapt to concept drift. Over the last 10 years, many algorithms for clustering data streams have been proposed. Most data stream clustering algorithms use a two-stage online/offline approach.



**Figure 1.** Problem with reclustering when dense areas are separated by small areas of low density with (a) micro clusters and (b) grid cells.

Reclustering methods based solely on micro-clusters only take closeness of the micro-clusters into account. This makes it likely that two micro-clusters, which are close to each other, but separated by an area of low density still, will be merged into a cluster. Information about the density between micro-clusters is not available since the information is not recorded in the online step and the original data points are no longer available. Fig. 1a illustrates the problem where the micro-clusters MC1 and MC2 will be merged as long as their distance  $d$  is low. This is even true when density-based clustering methods (e.g., DBSCAN) are used in the offline reclustering

step, since the reclustering is still exclusively based on the micro-cluster centres and weights.



**Figure 2.** MC1 is a single MC. MC2 and MC3 are close to each other but the density between them is low relative to the two MCs densities while MC3 and MC4 are connected by a high-density area.

### III. PROPOSED SYSTEM

In this paper, we develop and evaluate a new method to address this problem for micro-cluster-based algorithms. We introduce the concept of a shared density graph which explicitly captures the density of the original data between micro-clusters during clustering and then show how the graph can be used for reclustering micro-clusters. This is a novel approach since instead on relying on assumptions about the distribution of data points assigned to a microcluster (often a Gaussian distribution around a center); it estimates the density in the shared region between micro clusters directly from the data.

### IV. THE DBSTREAM ONLINE COMPONENT

Typical micro-cluster-based data stream clustering algorithms retain the density within each micro-cluster as some form of weight (e.g., the number of points assigned to the MC). Some algorithms also capture the dispersion of the points by recording variance. For reclustering, however, only the distances between the MCs and their weights are used. In this setting, MCs, which are closer to each other, are more likely to end up in the same cluster. This is even true if a density-based algorithm like DBSCAN [10] is used for reclustering since here only the position of the MC centers and their weights are used. The density in the area between MCs is not

available since it is not retained during the online stage.

The basic idea of this work is that if we can capture not only the distance between two adjacent MCs but also the connectivity using the density of the original data in the area between the MCs, then the reclustering results may be improved. In the following, we develop DBSTREAM, which stands for density-based stream clustering.

### Leader-Based Clustering

Leader-based clustering was introduced by Hartigan [21] as a conventional clustering algorithm. It is straightforward to apply the idea to data streams (see, e.g., [20]). DBSTREAM represents each MC by a leader (a data point defining the MC's center) and the density in an area of a user-specified radius  $r$  (threshold) around the center. This is similar to DBSCAN's concept of counting the points in an  $\epsilon$ -neighborhood, however, here the density is not estimated for each point, but only for each MC which can easily be achieved for streaming data. A new data point is assigned to an existing MC (leader) if it is within a fixed radius of its center. The assigned point increases the density estimate of the chosen cluster and the MC's center is updated to move towards the new data point. If the data point falls in the assignment area of several MCs then all of them are updated. If a data point cannot be assigned to any existing MC, a new MC (leader) is created for the point. Finding the potential clusters for a new data point is a fixed-radius nearest-neighbor problem [22] which can be efficiently dealt with for data of moderate dimensionality using spatial indexing data structures like a k-d tree [23]. Variations of this simple algorithm were suggested in [24] for outlier detection and in [25] for sequence modeling.

### Competitive Learning

New leaders are chosen as points, which cannot be assigned to an existing MC. The positions of these newly formed MCs are most likely not ideal for the clustering. To remedy this problem, we use a

competitive learning strategy introduced in [26] to move the MC centers towards each newly assigned point. To control the magnitude of the movement, we use a neighborhood function  $h_{\delta P}$  similar to self-organizing maps.

### Capturing Shared Density

Capturing shared density directly in the online component is a new concept introduced in this paper. The fact, that in dense areas MCs will have an overlapping assignment area, can be used to measure density between MCs by counting the points, which are assigned to two or more MCs. The idea is that high density in the intersection area relative to the rest of the MCs' area means that the two MCs share an area of high density and should be part of the same macro-cluster. In the example in Fig. 2, we see that MC2 and MC3 are close to each other and overlap. However, the shared weight  $s_2$ ; three is small compared to the weight of each of the two involved MCs indicating that the two MCs do not form a single area of high density. On the other hand, MC3 and MC4 are more distant, but their shared weight  $s_3$ ; four is large indicating that both MCs form an area of high density and thus should form a single macro-cluster.

### Fading and Forgetting Data

To adapt to evolving data streams we use the exponential fading strategy introduced in Den Stream [6] and used in many other algorithms. Cluster weights are faded in every time step by a factor of  $2^{-\alpha}$ , where  $\alpha > 0$  is a user-specified fading factor. We implement fading in a similar way as in D-Stream [9], where fading is only applied when a value changes (e.g., the weight of a MC is updated). For example, if the current time-step is  $t = 10$  and the weight  $w$  was last updated at  $t = 5$  then we apply for fading the factor  $2^{-\alpha(t-t_{\text{last}})}$  resulting in the correct fading for five time steps. In order for this approach to work, we have to keep a time-stamp with the time when fading was applied last for each value that is subject to fading.

## The Complete Online Algorithm

**Algorithm 1** Update DBSTREAM clustering.

**Require clustering** data structures initially empty or 0

MC. Set of MCs

$Mc \in MC$  has elements  $mc = (c; w; t)$ . Center, weight, Last update time

S. weighted adjacency list for shared density graph

$s_{ij} \in S$  has an additional field  $t$ . time of last update

$t$ . current time step

**Require** User-specified parameters

$r$ . clustering threshold

$\alpha$ . fading factor

$tgap$ . cleanup interval

$wmin$ . minimum weight

$\beta$ . intersection factor

1: **function** UPDATE( $x$ ) . new data point  $x$

2:  $N \leftarrow \text{ndFixedRadiusNN}(x; MC; r)$

3: **if**  $|N| < 1$  **then** . create new MC

4: add  $(c = x; t = t; w = 1)$  to MC

5: **else** . update existing MCs

6: **for each**  $i \in N$  **do**

7:  $mci[w] \leftarrow mci[w] \cdot \alpha^{(t - mci[t])} + 1$

8:  $mci[c] \leftarrow mci[c] + h(x; mci[c])(x - mci[c])$

9:  $mci[t] \leftarrow t$

. update shared density

10: **for each**  $j \in N$  where  $j > i$  **do**

11:  $s_{ij} \leftarrow s_{ij} \cdot \alpha^{(t - s_{ij}[t])} + 1$

12:  $s_{ij}[t] \leftarrow t$

13: **end for**

14: **end for**

. prevent collapsing clusters

15: **for each**  $(i; j) \in N \times N$  and  $j > i$  **do**

16: **if**  $\text{dist}(mci[i]; mci[j]) < r$  **then**

17: revert  $mci[i], mci[j]$  to previous positions

18: **end if**

19: **end for**

20: **end if**

21:  $t \leftarrow t + 1$

22: **end function**

Algorithm 1 shows our approach and the used clustering data structures and user-specified

parameters in detail. Micro-clusters are stored as a set MC. Each micro-cluster is represented by the tuple  $\delta c; w; t$  representing the cluster center, the cluster weight and the last time it was updated, respectively. The weighted adjacency list S represents the sparse shared density graph which captures the weight of the data points shared by MCs. Since shared density estimates are also subject to fading, we also store a timestamp with each entry. Fading also shared density estimates is important since MCs are allowed to move which over time would lead to estimates of intersection areas the MC is not covering anymore.

## V. COMPUTATIONAL COMPLEXITY

Space complexity of the clustering depends on the number of MCs that need to be stored in MC. In the worse case, the maximum number of strong MCs at any time is  $tgap$  MCs and is reached when every MC receives exactly a weight of one during each interval of  $tgap$  time steps. Given the cleanup strategy in Algorithm 2, where we remove weak MCs every  $tgap$  time steps, the algorithm never stores more than  $k \frac{1}{4} 2tgap$  MCs.

**Algorithm 2** Cleanup process to remove inactive microclusters

and shared density entries from memory.

**Require:**  $\alpha, \beta, tgap, t, MC$  and S from the clustering.

1: **function** CLEANUP( )

2:  $w_{weak} = 2 \cdot \alpha^{-tgap}$

3: **for each**  $mc \in MC$  **do**

4: **if**  $mc[w] \cdot \alpha^{(t - mc[t])} < w_{weak}$  **then**

5: remove weak  $mc$  from MC

6: **end if**

7: **end for**

8: **for each**  $s_{ij} \in S$  **do**

9: **if**  $s_{ij} \cdot \alpha^{(t - s_{ij}[t])} < \alpha \cdot w_{weak}$  **then**

10: remove weak shared density  $s_{ij}$  from S

11: **end if**

12: **end for**

13: **end function**

The space complexity of MC is linear in the maximal number of MCs  $k_0$ . The worst case size of the adjacency list of the shared density graph  $S$  depends on  $k_0$  and the dimensionality of the data. In the 2D case each MC can have a maximum of  $\lfloor \frac{n}{k_0} \rfloor$  neighbors (at optimal packing). Therefore, each of the  $k_0$  MCs has in the adjacency list  $S$  at most six entries resulting in a space complexity of storing MC and  $S$  of  $O(k_0 n)$ . For higher-dimensional data streams, the maximal number of possible adjacent hyper spheres is given by Newton's number also referred to as kissing number [29]. Newton's number defines the maximal number of hyper spheres which can touch a hyper sphere of the same size without intersecting any other hyper sphere. If we double the radius of all hyper spheres in this configuration then we get our scenario with sphere centers touching the surface of the center sphere. We use  $K_d$  to denote Newton's number in  $d$  dimensions. Newton's exact number is known only for some small dimensionality values  $d$ , and for many other dimensions only lower and upper bounds are known. Note, that Newton's number grows fast, reaches 196,560 for  $d = 24$  and is unknown for most larger  $d$ . This growth would make storing the shared weights for high-dimensional data in a densely packed area very expensive. However, we also know that the maximal neighborhood size  $\lfloor \frac{n}{k_0} \rfloor$ , since we cannot have more neighbors than we have MCs. Therefore, the space complexity of maintaining  $S$  is bounded by  $O(k_0 n)$ . To analyze the algorithm's time complexity, we need to consider all parts of the clustering function. The fixed-radius nearest neighbor search can be done using linear search in  $O(dnk_0)$ , where  $d$  is the data dimensionality,  $n$  is the number of data points clustered and  $k_0$  is the number of MCs. The time complexity can be improved to  $O(dn \log k_0)$  using a special indexing data structure like a  $k$ - $d$  tree [23]. Adding or updating a single MC is done in time linear in  $n$ .

## VI. EXPERIMENTS ANALYSIS

To perform our experiments and make them reproducible, we have implemented/interfaced all algorithms in a publicly available R-extension called stream [30]. Stream provides an intuitive interface for experimenting with data streams and data stream algorithms. It includes generators for all the synthetic data used in this paper as well as a growing number of data stream mining algorithms including clustering algorithms available in the MOA (Massive Online Analysis) framework [31] and the algorithm discussed in this paper. In this paper we use four synthetic data streams called Cassini, Noisy Mixture of Gaussians, and DS3 and DS41 used to evaluate CHAMELEON [13]. These data sets do not exhibit concept drift. For data with concept drift we use MOA's Random RBF Generator with Events. In addition we use several real data sets called Sensor, 2 Forest Cover Type3 and the KDD CUP'99 data4 which are often used for comparing data stream clustering algorithms.

Kremer et al. [32] discuss internal and external evaluation measures for the quality of data stream clustering. We conducted experiments with a large set of evaluation measures (purity, precision, recall, F-measure, sum of squared distances, silhouette coefficient, mutual information, adjusted Rand index). In this study we mainly report the adjusted Rand index to evaluate the average agreement of the known cluster structure (ground truth) of the data stream with the found structure. The adjusted Rand index (adjusted for expected random agreements) is widely accepted as the appropriate measure to compare the quality of different partitions given the ground truth [33]. Zero indicates that the found agreements can be entirely explained by chance and the closer the index is to one, the better the agreement. For clustering with concept drift, we also report average purity and average within cluster sum of squares (WSS). However, like most other measures, these make comparison difficult. For example, average purity (equivalent to precision and part of

the F-measure) depends on the number of clusters and thus makes comparison of clustering's with a different number of clusters invalid. The within cluster sum of squares favors algorithms which produce spherical clusters (e.g., k-means-type algorithms).

## VII. CONCLUSION

This paper, we have developed the first data stream clustering algorithm which explicitly records the density in the area shared by micro-clusters and uses this information for reclustering. We have introduced the shared density graph together with the algorithms needed to maintain the graph in the online component of a data stream mining algorithm. Although, we showed that the worst-case memory requirements of the shared density graph grow extremely fast with data dimensionality, complexity analysis and experiments reveal that the procedure can be effectively applied to data sets of moderate dimensionality. Experiments also show that shared-density reclustering already performs extremely well when the online data stream clustering component is set to produce a small number of large MCs. Other popular reclustering strategies can only slightly improve over the results of shared density reclustering and need significantly more MCs to achieve comparable results. This is an important advantage since it implies that we can tune the online component to produce less micro-clusters for shared In -density reclustering. This improves performance and, in many cases, the saved memory more than offset the memory requirement for the shared density graph.

## VIII. REFERENCES

- [1]. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in Proceedings of the ACM Symposium on Foundations of Computer Science, 12-14 Nov. 2000, pp. 359–366.
- [2]. C. Aggarwal, Data Streams: Models and Algorithms, ser. Advances in Database Systems, Springer, Ed., 2007.
- [3]. J. Gama, Knowledge Discovery from Data Streams, 1st ed. Chapman & Hall/CRC, 2010.
- [4]. J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L.F. d. Carvalho, and J. a. Gama, "Data stream clustering: A survey," ACM Computing Surveys, vol. 46, no. 1, pp. 13:1–13:31, Jul. 2013.
- [5]. C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in Proceedings of the International Conference on Very Large Data Bases (VLDB '03), 2003, pp. 81–92.
- [6]. F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in Proceedings of the 2006 SIAM International Conference on Data Mining. SIAM, 2006, pp. 328–339.
- [7]. 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. New York, NY, USA: ACM, 2007, pp. 133–142.
- [8]. L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Densitybased clustering of data streams at multiple resolutions," ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 3, pp.1–28, 2009.
- [9]. L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 3, pp. 1–27, 2009.
- [10]. L. Ertoz, M. Steinbach, and V. Kumar, "A new shared nearestneighbor clustering algorithm and its applications," in Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining, 2002.
- [11]. P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree:indexing micro-clusters for anytime stream mining," Knowledge and Information Systems, vol. 29, no. 2, pp. 249–272, 2011.
- [12]. A. Amini and T. Y. Wah, "Leaden-stream: A leader density based clustering algorithm over

- evolving data stream," *Journal of Computer and Communications*, vol. 1, no. 5, pp. 26–31, 2013.
- [13]. J. A. Hartigan, *Clustering Algorithms*, 99th ed. New York, NY, USA: John Wiley & Sons, Inc., 1975.
- [14]. E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Data Mining for Security Applications*. Lower, 2002.
- [15]. M. Hahsler and M. H. Dunham, "Temporal structure learning for clustering massive data streams in real-time," in *SIAM Conference on Data Mining (SDM11)*. SIAM, April 2011, pp. 664–675.
- [16]. C. Isaksson, M. H. Dunham, and M. Hahsler, "Sostream: Self organizing density-based clustering over data stream," in *Machine Learning and Data Mining in Pattern Recognition*, ser. *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, vol. 7376, pp. 264–278.
- [17]. "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Self-organized Formation of Topologically Correct Feature Maps, pp. 509–521.
- [18]. M. Hahsler, M. Bolanos, and J. Forrest, *stream: Infrastructure for Data Stream Mining*, 2015, R package version 1.2-2.
- [19]. A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 99, pp. 1601–1604, August 2010.
- [20]. H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer, "An effective evaluation measure for clustering on evolving data streams," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 868–876.
- [21]. J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, pp. 317–346, 2013.
- [22]. A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. *KDD '15*. ACM, 2015, pp. 59–68