An Innovative Framework for Static and Dynamic Clustering using Histogram Models and Wasserstein Distance over Sliding Windows

Xiaotong Qian, Guénaël Cabanes, Parisa Rastin, Mohamed Alae Guidani, Ghassen Marrakchi, Marianne Clausel, Nistor Grozavu

Highlights

An Innovative Framework for Static and Dynamic Clustering using Histogram Models and Wasserstein Distance over Sliding Windows

Xiaotong QIAN, Guénaël CABANES, Parisa RASTIN, Mohamed Alae GUIDANI, Ghassen MARRAKCHI, Marianne CLAUSEL, Nistor GROZAVU

- A fast and versatile framework for static and dynamic clustering over sliding windows.
- Uses histogram models to represent clusters with arbitrary distributions.
- High degree of flexibility in selecting the clustering algorithm to apply.
- Efficient and high quality results are achieved by using Wasserstein distance-based two-sample statistical tests to compare distributions.

An Innovative Framework for Static and Dynamic Clustering using Histogram Models and Wasserstein Distance over Sliding Windows*

Xiaotong QIAN[a], Guénaël CABANES[b], Parisa RASTIN[c], Mohamed Alae GUIDANI[d], Ghassen MARRAKCHI[b], Marianne CLAUSEL[c] and Nistor GROZAVU[a]

[a]ETIS, UMR 8051, CY Cergy Paris Université, Cergy, 95000, France
[b]LIPN, UMR 7030, Université Sorbonne Paris Nord, Villetaneuse, 93430, France
[c]LORIA, UMR 7503, Université de lorraine, Vandoeuvre-lès-Nancy, 54500, France
[d]École nationale supérieure des mines de Nancy, Campus Artem, Nancy, 54042, France

ARTICLE INFO

Keywords:
Unsupervised learning
Static and dynamic clustering
Large datasets
Data streams
Sliding windows
Histogram models
Wasserstein distance

ABSTRACT

In this article, we present an innovative clustering framework designed for large datasets and real-time data streams which uses a sliding window and histogram model to address the challenge of memory congestion while reducing computational complexity and improving cluster quality for both static and dynamic clustering. The framework provides a simple way to characterize the probability distribution of cluster distributions through histogram models, regardless of their distribution type. This advantage allows for efficient use with various conventional clustering algorithms. To facilitate effective clustering across windows, we use a statistical measure that allows the comparison and merging of different clusters based on the calculation of the Wasserstein distance between histograms.

1. Introduction

Machine learning [1] is a prominent field of research that focuses on learning patterns and relationships within datasets to make intelligent predictions or analyses. It includes two main types of algorithms: supervised learning and unsupervised learning. They are often referred to as clustering [2] and classification [3]. While both aim to separate and group objects, there is a fundamental difference between them. In classification, the categories are predetermined and objects are assigned to a specific category (called a label). The labels are therefore necessary to train a classification model. In clustering, on the other hand, labels are not needed to train a clustering model; the goal is to group similar objects based on some definition of distance or similarity. Depending on how the data is processed, clustering analysis can be divided into two forms: conventional clustering and data stream clustering.

Conventional clustering algorithms [4, 5, 6] work on static datasets, that are fixed and remains the same throughout the training. There are several broad categories, such as hierarchical-based, partition-based, graph-based, density-based, model-based, and grid-based clustering. Hierarchical-based clustering iteratively divides the dataset into subsets from top to bottom, or merges individual objects from bottom to top, forming a dendrogram; popular examples of this type include BIRCH [7] and Agglomerative clustering [8]. Partition-based clustering aims to separate objects into K groups by optimizing some criterion such as minimizing the total intra-cluster distance; KMeans [9] and KMedian [10] are typical examples. Graph-based clustering, such as Spectral clustering [11], uses the concept of graphs to represent data points and their relationships. Density-based clustering, such as OPTICS [12], DBSCAN [13], and HDBSCAN [14], relies on notions of density within neighborhoods to determine clusters. Model-based clustering, such as Gaussian Mixture [15], assumes that data points are derived from a combination of Gaussian distributions with different parameters. Finally, grid-based clustering divides the data space into grids and groups data points within

*This work was funded through the ANR project Pro-Text (project N° ANR-18-CE23-0024-01). More details are available at: https://pro-text.huma-num.fr/le-projet/

xiaotong.qian@ensea.fr (X. QIAN); guenael.cabanes@lipn.univ-paris13.fr (.G. CABANES); parisa.rastin@loria.fr (.P. RASTIN); mohamedalae.guidani@mines-nancy.org (.M.A. GUIDANI); ghassen.marrakchi@edu.univ-paris13.fr (.G. MARRAKCHI); marianne.clausel@univ-lorraine.fr (.M. CLAUSEL); nistor.grozavu@cyu.fr (.N. GROZAVU)
ORCID(s): 0009-0005-8249-3156 (X. QIAN)

the same grid such as STING [16] and CLIQUE [17]. These clustering algorithms provide valuable insights and form the basis for data analysis in various fields.

However, data stream clustering [18, 19, 20, 21] differs from conventional clustering. It operates on continuous data streams, eliminating the need to wait for the entire dataset to be collected. This real-time nature allows it to dynamically process incoming data points and adapt the clustering models to any changes in the data distribution over time. Many data stream clustering approaches can be considered extensions or variations of conventional clustering, resulting in different categories within the field. The first proposed data stream clustering STREAM [22] is of partition-based type. This algorithm reads a data stream across fixed size batches or windows, applies KMedian clustering to each batch, and merges the obtained centers into K medians to find final clusters. And few years later, the authors proposed an improved version, STREAMLSEARCH [23], by adding local search techniques. CluStream [24], a well-known hierarchical and partition-based approach, extends the CF tree structure of BIRCH. By using an online-offline algorithm with KMeans, it effectively summarizes micro-clusters and preserves them at specific points in time within a pyramidal time frame. In addition, density-based approaches have been proposed, such as DenStream [25], an extension of DBSCAN. These types of algorithms have the ability to detect clusters of any shape without prior knowledge of the number of clusters, and are capable of dealing with outliers. This can play an important role in real-world applications. Apart from these, there are other types of data stream clustering, including grid-based approaches such as MR-Stream [26] and DGClust [27], as well as model-based algorithms SWEM [28] and GCPSOM [29].

A wide variety of clustering algorithms exist for both static and dynamic datasets. However, no clustering algorithm is universally perfect; each type of clustering algorithm has its advantages and disadvantages. For example, partition-based algorithms have limitations: they require the number of clusters to be specified in advance and struggle with outliers, but they work efficiently and can handle large datasets with relatively low time complexity. Meanwhile, density-based clustering algorithms work on any kind of cluster distribution, automatically detect the number of clusters, and handle outliers very well, but are slower and more difficult to parameterize. Therefore, the application or choice of different algorithms is highly dependent on the specific requirements or needs of the task. In this article, we present a fast, innovative clustering framework adapted to large datasets and data streams that operates over sliding windows and uses histogram models to characterize cluster distributions. Histogram models provide simple representations of clusters without requiring prior knowledge of their distribution. A major advantage of the proposed framework is that it allows fast computational application of any clustering algorithms. To compare and merge different clusters in different windows, we take advantage of the calculation of the Wasserstein distance between histograms [30], facilitating effective cluster analysis and synthesis. A general scheme of the process is illustrated in Fig.1.

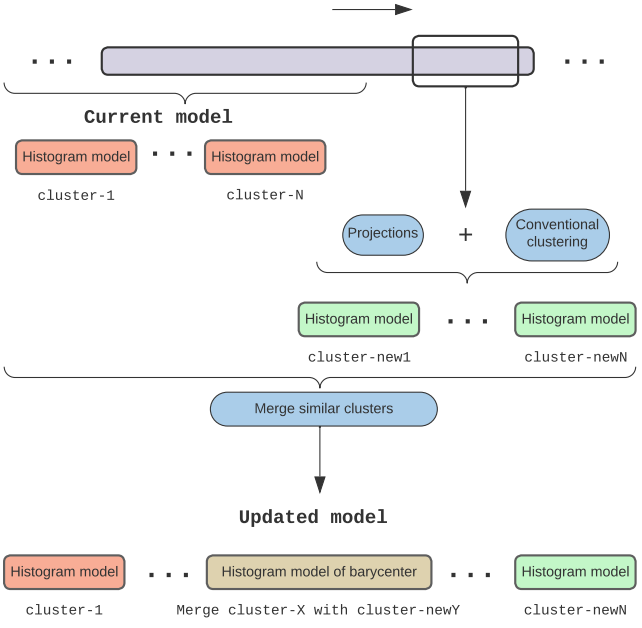

Figure 1: Proposed approach

The paper is organized as follows. In section 2, we provide a brief overview of the theoretical background, including the construction of the histogram models, the computation of the similarities between distributions using Wasserstein distance and the implementation of a two-sample hypothesis testing. In section 3, we delve into the details of the approach, outlining the process of modeling clusters as histograms and the subsequent merging of different windows. In section 4, we conduct experiments to compare the static and dynamic approaches to conventional clustering and data stream clustering on both artificial and real datasets. Finally, in section 5, we summarize the results and discuss potential perspectives for future research.

2. Background

In this section, we present the theoretical background on which our approach is based. In particular, an important goal is to be able to compare the distributions of two clusters efficiently and with low computational cost in order to determine whether they should be considered as different clusters or, on the contrary, should be merged. For this purpose, we base our analysis on a representation of the distribution of each cluster as a set of histograms, and then compare these distributions using a Wasserstein distance adapted to this representation. Finally, a specific two-sample test is computed from this measure to assess the significance of the distance obtained.

2.1. Data distributions using histograms

Estimating probabilistic data distributions [31, 32, 33] is a fundamental part of data science and machine learning, especially in unsupervised learning. Unraveling the distribution of data allows the discovery of hidden patterns in data sets and is a powerful tool for analyzing large amounts of data in an unsupervised manner. In clustering applications [4, 34], clusters are often assumed to be normally distributed [35] (i.e., have a multivariate Gaussian distribution) or sometimes have a more complex distribution, such as gamma distributions [36]. All subsequent analysis is then based on this assumption. These parametric approaches can provide good results, but they have limited applications when it comes to representing clusters of arbitrary distributions, which is an important requirement for many real-world scenarios. One possible solution is to model the distribution as a mixture of simple functions (usually Gaussian). However, Gaussian mixture models [37] can be computationally expensive and slow to train, especially when the number of mixture components is large. They are also sensitive to initialization, and choosing the number of mixture components in the model is often challenging. Another solution, which we focus on in this paper, is to model the cluster distributions using a set of histograms computed from the empirical distribution [38] of the data. Although this representation can lead to a loss of fine variations of the underlying distribution depending on the number of bins, it has the advantage of being independent of any assumptions and is fast and easy to compute. It also greatly simplifies the computation of the Wasserstein distance [39].

The term histogram was first proposed by [40], who described a histogram as a series of rectangles of equal width whose height could represent the number of values falling within the interval formed by their two edges. As mentioned in [41], using histograms to represent data could be a concise and flexible case of symbolic or summarised data analysis when faced with large amounts of data. In this case, the weight of each rectangle of the histogram is no longer the number of observations, but the probability or proportion of values over a set of intervals, formally defined that
A histogram is a model for representing the empirical distribution of a continuous variable Y divided into a set of contiguous I_ϕ intervals (bins) with associated π_ϕ weights. A histogram H is thus represented by a set of Φ ordered pairs (I_ϕ, π_ϕ), where π_ϕ is a non-negative measure of a probability distribution on the domain of Y such that:

$$\begin{cases} \sum_{\phi=1}^{\Phi} \pi_\phi = 1 \text{ with } \pi_\phi \geq 0 \\ I_\phi \cap I_{\phi'} = \emptyset, \phi \neq \phi' \\ \bigcup_{\phi=1,\dots,\Phi} I_\phi = [Y_{min}, Y_{max}] \end{cases}$$

There are two ways of displaying histograms. The first type allows each bin to have a fixed uniform length, but not a uniform weight, expressed as : $|I_\phi| = \frac{Y_{max}-Y_{min}}{H}$. The second type allows each bin to have a fixed uniform weight, but not a uniform length, expressed as : $|\pi_\phi| = \frac{1}{H}$. Fig.2 illustrates the difference.

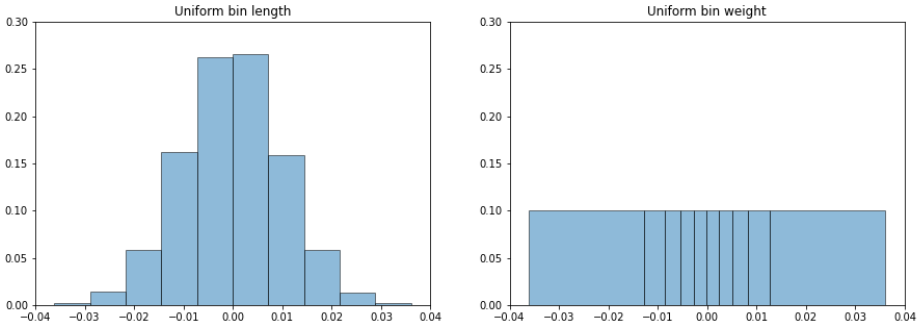

Figure 2: Two types of histogram representations which display a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ where μ is 0 and σ is 0.01

The time complexity of computing a univariate histogram depends on the number of bins Φ in the histogram and the number of data points n being processed which is $O(n * \Phi)$, where Φ is usually much smaller than n and need not be proportional to n. However, computing a multivariate histogram is much more expensive because the number of bins Φ in each dimension should be considered, with a time complexity of $O(n * \Phi^{\upsilon})$, where υ is the dimensionality of the dataset. One way to reduce the computational cost of multivariate histograms is to compute separate independent histogram for each variable, which reduces the complexity to $O(n * \upsilon * \Phi)$, but correlation information is lost in this approach. If some attributes are correlated, we cannot treat their histogram representation independently. For these reasons, in this paper we focus on independent histograms and deal with the loss of correlation information by using random projections which will be presented in section 3.1.

2.2. Distribution comparison

To increase storage efficiency and achieve more consistent clustering results, it is imperative to compare the distributions of different clusters once they have been identified. Then, clusters with similar distributions should be merged to achieve better clustering results. However, comparing different distributions can be challenging and requires a suitable metric to assess the similarity between two distributions μ and υ on the same sample space $\mathcal{X}$. A widely used measure is the Kullback-Leibler divergence (KL divergence expressed in Eq.1) [42], which computes the expectation of the logarithmic difference between the probability distributions μ and υ:

$$KL(\mu, \upsilon) = \int_{\mathcal{X}} \mu(x) log(\frac{\mu(x)}{\upsilon(x)}) dx. \tag{1}$$

Since the KL divergence is not symmetric, the Jensen-Shannon divergence (JS divergence expressed in Eq.2) [43] has been proposed to solve the symmetry problem, known as the total divergence to the average, the square root of the JS divergence is a metric often referred to as the JS distance.

$$JS(\mu, \upsilon) = \frac{KL(\mu, \frac{\mu+\upsilon}{2}) + KL(\upsilon, \frac{\mu+\upsilon}{2})}{2} \tag{2}$$

The KL and JS divergences suffer from several drawbacks, as non robustness to outliers. In addition, KL and JS are not distances and they do not satisfy triangle inequality. For all these reasons, another metric, the so called Wasserstein distance has gained in popularity. Wasserstein distance takes into account the cost of moving the mass from one distribution to another, which helps to mitigate the impact of outliers. The Wasserstein distance is also a continuous function, meaning that small changes in the input distributions result in small changes in the distance measurement. This is not the case with KL divergence, which can exhibit discontinuities and is therefore less suitable for some applications. In addition, Wasserstein distance satisfies the properties of a metric, such as symmetry, triangle inequality, and non-negativity. This property is not satisfied by JS and KL distances, making Wasserstein distance more suitable for use in optimization and clustering algorithms. Finally, Wasserstein distance has a natural interpretation as the minimum cost of transforming one distribution to another, which makes it easier to understand and explain in comparison to other distance measures.

Originally introduced in the context of optimal transport theory [39], this metric quantifies the minimum cost required to transform one probability distribution into another, where cost is defined as the amount of "work" required to move a given amount of mass from one point to another. The traditional approach to estimating the Wasserstein distance is outlined in [44] with a computational complexity of $O(n^3 log(n))$ using the EMD definition of this distance.

$$W_{emd}(\mu, \upsilon) = \min_{\gamma \in \mathcal{P}} \int \|x - y\| \gamma(x, y) dx dy = \mathop{\mathbf{E}}_{(x,y)\sim\gamma} [\|x - y\|] \tag{3}$$

where $\mathcal{P}$ is the set of all joint distributions $\Pi(\mu, \upsilon)$ whose marginals are μ and υ , $\|x - y\|$ gives the cost of transporting a unit of mass from point x to point y. A transport plan to move μ to ν can be described by a function $\gamma(x, y)$ which gives the amount of mass to be moved from x to y.

The Sinkhorn distance [45] is a faster alternative to EMD for solving regularised optimal transport problems with computational complexity up to $O(n^2)$. The cost of mapping μ to υ can be quantified as $\langle \mathcal{P}, M \rangle$, where M is a squared cost matrix, and the Sinkhorn distance directly regularises the original transport problem with a regularisation parameter λ and the entropy h defined as follow:

$$\lambda > 0, W^{\lambda}_{sinkorn}(\mu, \upsilon) := \langle \mathcal{P}^{\lambda}, M \rangle, \ \mathcal{P}^{\lambda} = \text{argmin} \langle \mathcal{P}, M \rangle - \frac{1}{\lambda} h(\mathcal{P}) \tag{4}$$

Computation of the Wasserstein distance can become exceedingly complex for large datasets, making it necessary to explore alternative methods. One such approach, which is specifically designed for histograms, can be found in [30]. This approach introduces an elegant formulation of the Wasserstein distance between two histograms H^i and H^j expressed as :

$$W^2_{hist}(H^i, H^j) = \sum_{\phi=1}^{\Phi} \pi_\phi d^2(I^i_\phi, I^j_\phi) = \sum_{\phi=1}^{\Phi} \pi_\phi \left[(c^i_\phi - c^j_\phi)^2 + \frac{1}{3}(r^i_\phi - r^j_\phi)^2 \right], \tag{5}$$

which simply compute the sum of the differences between each pair of centres C and radius R of two histograms, where π_ϕ is the probability weight associated to each bin, and I^i_ϕ represents an interval $([\underline{y^i_\phi}, \overline{y^i_\phi}])$ of histogram H^i, I^j_ϕ represents an interval $([\underline{y^j_\phi}, \overline{y^j_\phi}])$ of histogram H^j, with:

$$c^i_\phi = \frac{\underline{y^i_\phi} + \overline{y^i_\phi}}{2}; \qquad c^j_\phi = \frac{\underline{y^j_\phi} + \overline{y^j_\phi}}{2}; \qquad r^i_\phi = \frac{\underline{y^i_\phi} - \overline{y^i_\phi}}{2}; \qquad r^j_\phi = \frac{\underline{y^j_\phi} - \overline{y^j_\phi}}{2}.$$

They also propose a way to compute the Wasserstein distance between multivariate histograms. Suppose there are p histogram variables for observation i and j ($H^i_1 \cdots H^i_p$ and $H^j_1 \cdots H^j_p$), under the hypothesis that the variables are independent, the process is expressed like:

$$W^2_{hist}[(H^i_1 \cdots H^i_p), (H^j_1 ... H^j_p)] = \sum_{q=1}^{p} W^2_{hist}(H^i_q, H^j_q) \tag{6}$$

The proposed approach described in this paper is to compute histograms that model the data distribution before computing the distance between the centres and the radius of each interval. The time complexity of computing the Wasserstein distance between histograms is $O(\Phi^{\upsilon})$ for multivariate histograms and $O(\Phi * \upsilon)$ for independent histograms. Overall, since Φ is usually chosen as a constant positive parameter, the total time complexity of this approach for a set of υ independent histograms can be efficient compared to the complexity of multivariate histograms. Moreover, the time complexity of an algorithm also depends on its implementation and hardware. For example, parallelization can greatly reduce the complexity of independent histograms.

It is possible to compute the barycentre of a set of histograms by defining a distance measure between histograms. This barycentre is essentially a single histogram that minimises the squared Wasserstein distance between itself and each member of the set. In other words, it is the optimal representative histogram that captures the essential features of

the entire set. In the case of histograms with uniform bins weight, each one is described by the centres C and radius R of its bins, this involves computing new centres C^b and radius R^b while keeping the weight of each bin constant. The Eq.7, mentioned in [30], shows that this simply requires computing the mean centre and radius of each bin, where N_h is the number of histograms in the set, in our case N_h is set to 2:

$$C^b = N_h^{-1} \sum_{j=1}^{N_h} C^j \text{ and } R^b = N_h^{-1} \sum_{j=1}^{N_h} R^j. \tag{7}$$

2.3. Wasserstein two-samples testing

Performing non parametric two sample testing allows to detect, given samples $X_1, \cdots, X_n \sim \mu$ and $Y_1, \cdots, Y_m \sim \nu$ from the two unknown distributions μ and ν, if they are significantly different. One classical test is the Kolmogorov-Smirnov test (see [46]), or K-S test, which is commonly used to compare an empirical distribution to a theoretical distribution or to compare two empirical distributions. In the case of deciding whether two one-dimensional probability distributions (μ, ν) differ from each other by computing Eq.8:

$$D_{\mu,\nu} = \underset{x}{sup} |F_\mu(x) - F_\nu(x)| \tag{8}$$

and test the null hypothesis $(H_0) : \mu = \nu$ versus $(H_1) : \mu \neq \nu$.

One then rejects the null hypothesis (H_0) and accepts (H_1) with significance level α if

$$D_{\mu,\upsilon} > C(\alpha)\sqrt{\frac{n+m}{n \cdot m}}, \text{ where } C(\alpha) = \sqrt{-ln(\frac{\alpha}{2}) \times \frac{1}{2}}.$$

An alternative distribution free statistical test based on the Wasserstein distance has been proposed in [47]. This test is based on the following preliminary result on the asymptotic empirical Wasserstein distance between two samples. Denote F_n, G_m, the two empirical cumulative distribution functions (CDF) associated to the samples $X_1, \cdots, X_n \sim \mu$ and $Y_1, \cdots, Y_m \sim \nu$. Under the null hypothesis $H_0 : \mu = \nu$, one has

$$T_{m,n} := \frac{n \cdot m}{n+m} \int_0^1 \left(G_m \left(F_n^{-1}(t) - t \right)^2 \right) dt \rightarrow_w Z := \int_0^1 \mathbf{B}(t)^2 \, dt \text{ as } n, m \rightarrow \infty \tag{9}$$

where $B(t)$ represents the Brownian bridge [48]. The Brownian bridge is defined as $B(t) = W_{pr}(t) - \frac{t}{T} W_{pr}(T)$, where $t \in [0, T]$ (in the assumption $T = 1$) and W_{pr} is a standard Wiener process [49]. It describes a random walk from 0 to T starting at 0 and ending at 0, such as $W_{pr}(0) = W_{pr}(T) = 0$. We emphasize that on can simulate the Brownian Bridge and then compute empirically the quantiles of the random variable Z.

Using the test statistic $T_{m,n}$, we then rejects H_0 and accepts H_1 if $T_{m,n} > z_\alpha$ where z_α is the α-quantile of distribution Z. In short, instead of testing the vanishing of Wasserstein distance between G_m and F_n directly, this test computes the Wasserstein distance between $G_m(F_n^{-1})$ and a uniform distribution $\mathrm{U}_{[0,1]}$ on $[0, 1]$ which makes it distribution free.

3. Proposed Approach

The proposed approach uses a non-overlapping sliding window model to run a clustering algorithm on batches of data samples that are briefly held in memory. Then allowing any type of clustering algorithms to be applied across sliding windows. This flexibility is a key strength of our approach, allowing the selection of the most appropriate clustering method for the data at hand. The ability to remove noise using algorithms such as those in the DBSCAN family is particularly valuable in this context, as histograms can be sensitive to extreme values and outliers. However, our approach has shown excellent overall performance in tests with other types of clustering algorithms (see Section 4). This allows incremental clustering of large datasets and handling of data streams. The distribution of each cluster in a window is modeled as a set of unidimensional histograms computed in a random projection space, capturing correlation information between variables without the need to compute a costly multivariate histogram.

The newly computed clusters are compared to clusters found in previous windows using the Wasserstein distance between histogram models Eq.6. A statistical test based on Eq.9 is then performed to discover clusters with similar distributions to previously detected clusters. These clusters are merged and replaced by their barycentric histogram

using a modified form of Eq.11. At the end of this process, the histogram models are stored and the data samples are forgotten before a new window of data samples is placed in memory and processed. The main steps of the approach are shown in Algorithm 1. The remainder of this section provides the details of each step.

Algorithm 1: Proposed approach

Input: Number of projections p, number of bins Φ, data points arriving by windows $\mathbf{X} = \{X[1], X[2]...\}$, conventional clustering algorithm with θ parameters $Clus(\theta)$
Output: Final clusters Θ
$\delta_p \leftarrow$ Determine the similarity threshold δ using Algorithm 2;
$\Theta \leftarrow [\,]$; /* Histogram models stored */
/* Initialization of current window */
nb_w $\leftarrow 1$;
$X_c[nb\_w]) \leftarrow Clus(\theta, X[nb\_w])$; /* Detect clusters in the first window */
$X_c^p[nb\_w] \leftarrow p$ Random Projections of $X_c[nb\_w]$ by Eq.10 ;
$H\_X_c^p[nb\_w] \leftarrow$ Convert all the clusters $X_c^p[nb\_w]$ to histogram models as described in Section. 3.2;
$\Theta \leftarrow \Theta.add(H\_X_c^p[nb\_w])$; /* Add $H\_X_c^p[nb\_w]$ to the entire set Θ */
$nb\_w \leftarrow nb\_w + 1$;
while *length(X[nb_w]) != 0* **do**
  /* While the set of samples in the current window is not empty, do the same as for X[1], except save $H\_X_c^p[nb\_w]$ to the entire set Θ */
  $H\_X_c^p[nb\_w] \leftarrow$ Initialization of current window ;
  for *each cluster i in* Θ **do**
    for *each cluster j in* $H\_X_c^p[nb\_w]$ **do**
      Determine similarity between i and j with respect to Eq.9 ; /* More details in Section.3.3 */
      if *i and j are not significantly different* **then**
        $\Theta \leftarrow \Theta.add(j)$; /* Add j to the entire set Θ */
      end
      else
        Update i which is stored in Θ by the barycenter of i and j using Eq.11
      end
    end
  end
  $nb\_w \leftarrow nb\_w + 1$;
end

3.1. Multivariate histogram extension

Multivariate histogram models to represent the data distribution can be very computationally costly when the number of dimensions is high. Instead of using a multivariate histogram, the distribution can be modeled as a set of univariate histograms at very low computational cost, but all correlation information is lost. One solution inspired by Sliced Wasserstein distance proposed in [50] is to compute random projections of the data onto new axes drawn uniformly on the unit sphere. If the number of new axes is large enough, the resulting model of the distribution is very close to the multivariate model in terms of computing the Wasserstein distance, while reducing the computational complexity very significantly.

The axes of the projections of the data points are obtained via the dot product between the sample vectors X and M_p, the intermediate matrix M_p is expressed in Eq.10 as follow:

$$M_p = \frac{M_\Psi^2}{[\sqrt{\sum_{i=1}^{v}(\psi_i^1)^2}, \cdots, \sqrt{\sum_{i=1}^{v}(\psi_i^p)^2}]}, \qquad X^p = X \cdot M_p \tag{10}$$

where M_Ψ is a matrix which follows a standard normal distribution with shape (v, p), v is the number of variables, and p is the number of projections, and $\psi_i^j \in M_\Psi$.

3.2. Histograms computation

As mentioned in Section 2.1, we favor a fixed weight representation for the histograms, which greatly simplifies the computation of the Wasserstein distance introduced in Eq.5, Eq.6. In the proposed approach, each cluster is represented by a set of independent histograms. For this purpose, the samples of a cluster X_c^p obtained in the previous step must be divided into Φ intervals by computing the 1 to Φ-th quantile of X_c^p along each dimension, keeping the same weight π_ϕ for each interval, which is equal to $\frac{1}{\Phi}$. This process is repeated for all clusters in the current window.

3.3. Cluster similarity test

With respect to Eq.9, the detailed procedures for defining the similarity between two cluster distributions in the form of histograms are as follows:

1. Determine the Brownian bridge threshold δ corresponding to the dimension p using linear regression.
2. Obtain the CDF F_n and G_m for each pairwise histogram distribution μ and υ of two clusters.
3. Compute the composition function of G_m and the inverse function F_n^{-1} of F_n : $G_m(F_n^{-1})$.
4. Compute the sum of WD between each $G_m(F_n^{-1})$ and $\mathrm{U}_{[0,1]}$ by using Eq.6, then multiply with $\frac{n*m}{n+m}$.
5. Compare result of step 4 with the threshold δ.
6. If the result $< \delta$, it means two clusters are similar and must be merged.

To define the threshold δ, we used simulations of Brownian bridges for multivariate observations. Since the threshold δ corresponds to a Wasserstein distance between a random Brownian bridge and the uniform walk $B_t = 0$ with a probability of occurrence below 5%. The algorithm which computes the threshold δ corrspond of corrspoding p variables could be developped as described in Algorithm 2.

Algorithm 2: Computation of threshold δ corresponding to p dimensions

Input: p, M, N
Output: δ_p
for *i in 1,..,p* **do**
  `/* Create M synthetic Brownian Bridges, each one includes N 1d points, i.e. a randomly normal increase/decrease at each time step */`
  B[i] ← Create_Brownian_Bridge(M, N);
  MeanB[i] ← mean(B[i]^2, axis=1) ; `/* Mean of B[i]^2 by columns */`
end
Sum_MeanB ← Sum_MeanB(MeanB, axis=0) ; `/* Sum of MeanB by rows */`
`/* Extract (95%×M)-th element of sorted Sum_MeanB as threshold */`
δ_p ← sort(Sum_MeanB)[M*0.95];

Experiments show that the threshold δ has a linear relationship with the number of dimensions p. Therefore, instead of repeating the process of Algorithm 2, a simple linear regression model trained by δ_1 and δ_2 could be applied to any dimension p, allowing a fast computation of δ_p. Incidentally, the values of M and N do not matter much for the definition of the threshold; to maintain consistency, it is enough to keep them constant for the computation of δ_1 and δ_2. More precisely, we set them both to 10000.

3.4. Merging process

Once similar clusters found, it is necessary to merge them by computing their barycenter in order to reduce the complexity of time and space. Since some clusters may have similar distributions but large differences in the number of data, the barycenter distributions must be computed as a weighted sum of the mean centers and radius by modifying Eq.7:

$$C^b = \frac{\sum_{j=1}^{N_h} n_j * C^j}{\sum_{j=1}^{N_h} n_j} \; ; \; R^b = \frac{\sum_{j=1}^{N_h} n_j * R^j}{\sum_{j=1}^{N_h} n_j} \tag{11}$$

At the end of this step, the process of cluster detection and subsequent merging within a current window is achieved. By repeating this process for each window, it is possible to efficiently handle large datasets and data streams.

4. Experimental protocol and results

4.1. Datasets

To evaluate the proposed approach, we performed a series of experiments on different datasets, including real datasets and artificial datasets, with a variety of number of samples, dimensions and clusters (Tab.1). Detection of clusters in arbitrary shapes plays an important role in the proposed approach, besides real popular datasets used in recent research, artificial datasets with different cluster distributions are necessarily generated. Right after the creation and collection of real and artificial datasets, two ways of dataset preprocessing are followed. This allows clustering in both static and dynamic contexts, demonstrating the versatility and effectiveness of the proposed approach.

Table 1
Datasets summary

Real Datasets	Nb samples	Nb dimensions	Nb clusters
Outdoor	3501	21	40
Gassenor	13377	128	6
IGBN	24150	33	6
IABN	52848	33	6
Rialto	82250	27	10
IIAIN	452044	33	6
IIRIN	452044	33	6
Covtype	549829	10	7

Artificial Datasets	Nb samples	Nb dimensions	Nb clusters
Comet	3,20(*10^4)	50,100	15,50
Meteorite	3,20(*10^4)	50,100	15,50
Square	3,20(*10^4)	50,100	15,50
Gaussian	3,20(*10^4)	50,100	15,50
Moon	3,20(*10^4)	50,100	15,50
Circle	3,20(*10^4)	50,100	15,50
MixSmall	1,2,3 (*10^4)	50,50,50	24,36,48
MixLarge	10,20,50 (*10^4)	100,100,100	54,66,90

4.1.1. Artificial datasets

Generate several artificial databases in different distributions to simulate real scenarios, such as comet, meteorite, square, moon, circular, and Gaussian, by adding orientation to some clusters to replicate the real case. For the square distribution, the first dimension follows a univariate Gaussian distribution, then the remaining dimensions follow a uniform distribution. Then comes the comet distribution, where the first dimension follows a gamma distribution, then the remaining dimensions follow a univariate distribution. Similar to the comet distribution, the meteorite distribution follows a gamma distribution for all dimensions. Another type of distribution is the circle distribution, which is based on cosine and sine elements, and the moon distribution could be treated as a semicircle distribution. Then there is the Gaussian distribution, best known as the multinomial Gaussian distribution. Tab.2 below summarizes the information about each distribution, and also a short example of the visualization of all artificial distributions could be seen in Fig.3

Table 2
Artificial datasets different distributions description

Distribution name	Description	
	First dimension	Rest dimension
Square	Univariate Normal/Gaussian distribution	Uniform distribution
Comet	Gamma distribution	Univariate Normal/Gaussian distribution
Circle	All the dimensions based on cosine and sine element-wise.	
Meteorite	All the dimensions follow a Gamma distribution	
Moon	A half circle	
Gaussian	Multinormal Gaussian distribution	

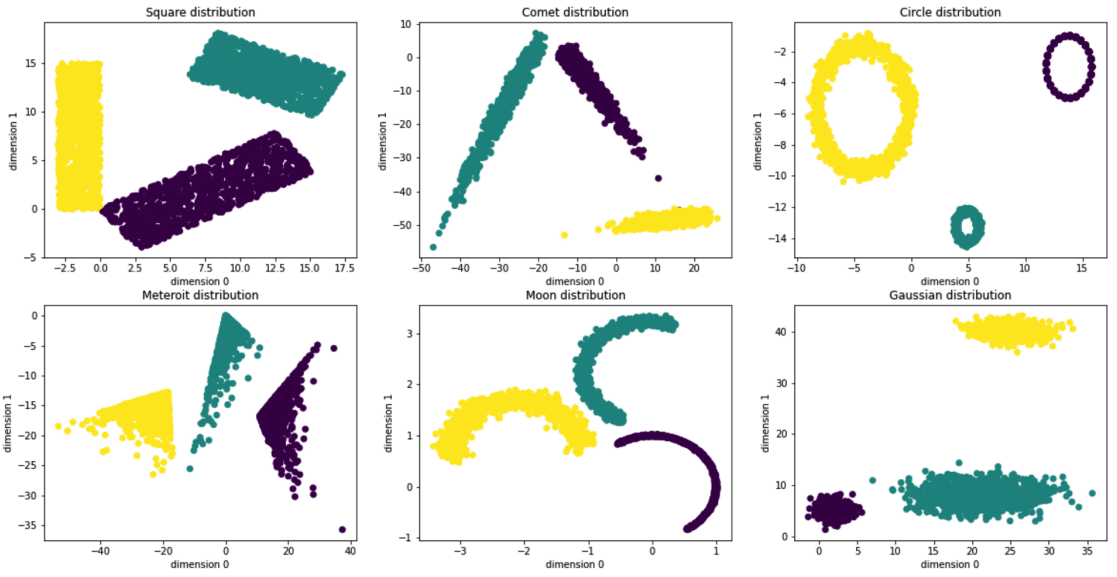

Figure 3: Artificial datasets of different distributions

4.1.2. Real datasets

We also used real datasets that are used in benchmarks for the purpose of data streams clustering, as mentioned in [51]. Each dataset is accompanied by a summary and more detailed information can be found in the same source.

- **Outdoor** [52] : The dataset consists of 4000 images of 40 different objects taken with a smartphone camera in a garden environment. The images were taken under different lighting conditions and from different distances and positions. The dataset uses a 21-dimensional RG chromaticity histogram to represent the examples.
- **Gassor** [53] : This dataset contains 13,910 recordings from 16 chemical sensors measuring six pure gases (ammonia, acetaldehyde, acetone, ethylene, ethanol and toluene) in a gas delivery platform at the University of California, San Diego. The classification task is to identify which gas is being measured.
- **Rialto** [54]: This dataset contains 82,250 examples of ten colourful buildings near the Rialto Bridge in Venice. The dataset was constructed using images extracted from time-lapse videos captured by a fixed-position webcam, covering 20 consecutive days in May-June 2016. The classification task is to identify the correct building, and each class has 8225 examples encoded in a normalised 27-dimensional RGB histogram.
- **Covtype** [55] : This dataset contains 581,012 instances with 54 attributes related to forest cover type in 30×30 cells obtained from the US Forest Service Region 2 Resource Information System. It includes seven class labels.
- **INSECT_** [51]: Theses datasets refers to a significant public health issue that involves the identification of disease-carrying insects using optical sensors. In consideration of the impact of temperature on the data captured by the optical sensor and the consequent conceptual drifts, it is important to carefully evaluate the different modifications made to the datasets, which are also responsible for their respective names. We use the following abbreviations to refer to these datasets:
  - INSECTS_abrupt_balanced_norm called IABN
  - INSECTS_gradual_balanced_norm called IGBN
  - INSECTS_incremental_abrupt_imbalanced_norm called IIAIN
  - INSECTS_incremental_reoccurring_imbalanced_norm called IIRIN

4.1.3. Datasets preprocessing

Since the proposed framework supports both static (batch) and dynamic (sequential) clustering, we prepared artificial and real datasets suitable for both types of clustering. For the static clustering experiment, the goal is to apply conventional clustering on non-overlapping windows to detect clusters, and then merge similar clusters using the proposed approach. This result will be compared to conventional clustering applied to the entire datasets. To ensure the validity of this experiment, each window must contain a certain number of different distributions, and the number of data should remain constant throughout the experiment. For the dynamic clustering experiment, we selected two typical conventional clustering algorithms to apply in our framework. We then compared the results with other popular stream clustering algorithms. To simulate a dynamic process, each data point was assigned a timestamp, which allowed us to set the time interval of the windows and read the data points falling in each interval based on their timestamps.

- **Static clustering (Fig.4)** : In order to evaluate our framework on static context, different distributions must be divided into multiple windows. One approach to accomplish this is by randomly shuffling the datasets, and subsequently selecting the i-th samples from the datasets for each window.

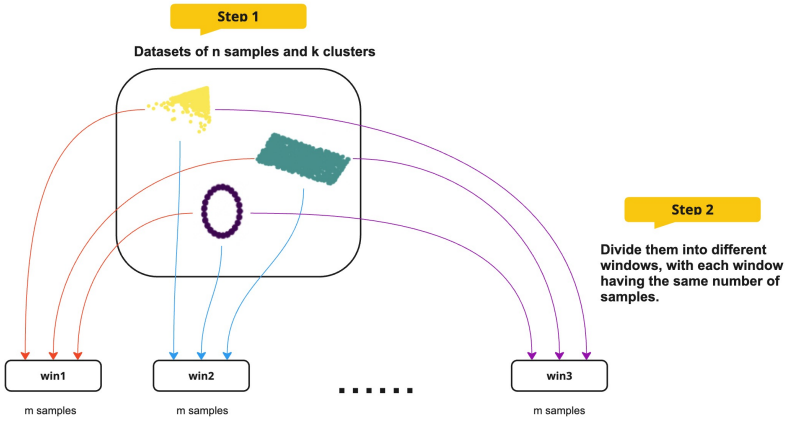

Figure 4: Static clustering

- **Dynamic clustering (Fig.5)** : Randomly assigning timestamps to each data point is necessary to ensure that each cluster remains within a specified time interval during testing of the dynamic model. The data points are then arranged based on their timestamps to allow sequential reading of the datasets, simulating a data stream.

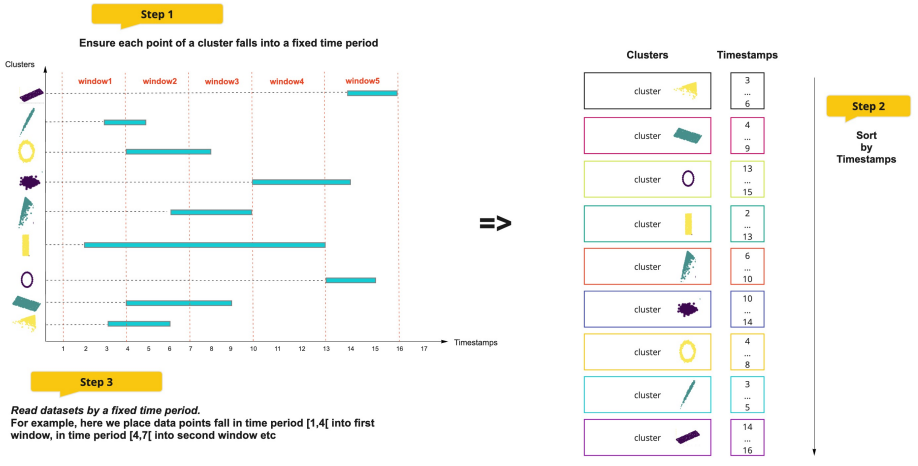

Figure 5: Dynamic clustering

4.2. Clustering evaluation strategy

The Ajusted Rand Index (ARI) and Purity are becoming increasingly popular for assessing the quality of clustering, as they are regularly used in a large number of studies. Both are external indices based on a priori knowledge of the data structure. Internal indices typically check the compactness and separability of clusters and work well to evaluate clusters that are close to a Gaussian distribution. Due to the diversity of cluster distributions in the experimental datasets, we decided to restrict the evaluation to external indices.

- **Ajusted rand index**: By considering all sample pair assignments in both the expected and actual clusterings, as well as counting sample pair assignments in the predicted or actual clusterings. The Rand Index (RI) measures how similar two clusterings are to each other. Thus, regardless of the number of clusters and samples, the Adjusted Rand Index (ARI) is guaranteed to be close to 0.0 for random labeling and exactly 1.0 when the clusterings are identical (up to one permutation).

$$ARI = \frac{RI - Expected\_RI}{max(RI) - Expected\_RI}$$

- **Purity**: Each cluster is given a description based on the class that makes up the majority of the cluster. Purity is then computed as the fraction of all data points divided by the number of successfully matched class and cluster labels. The formula is summarized as follows, where n is the number of samples and K is the number of clusters.

$$Putity = \frac{\sum_{k=1}^{K} max(\text{numer of majority cluster in k})}{n}$$

4.3. Parameter analysis

An analysis of the impact of different parameters of the proposed approach on its quality has been carried out. In particular, the number of bins of the histogram model, the number of random projections, and the number of windows (also related to the number of samples in a window), which could change the quality and complexity of the clustering, all these parameters could have an impact on the result of our approach. We ran each test 10 times and report the average performance with the corresponding standard deviation.

4.3.1. Number of bins

As can be seen from the results in Tab.3, Tab.4 and Fig.6, Fig.7, Fig.8, the proposed approach performs quite similarly regardless of the number of bins, on both artificial and real datasets. In fact, the quality of the clustering did not change significantly when the number of bins was adjusted. However, it is clear that the computation time is higher when the number of bins is increased. Therefore, in the following experiments, we set the number of bins to 10 in order to be efficient without losing quality.

Table 3
Results with increasing the number of bins – Artificial datasets

datasets	NbB	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
MixLarge10	10	0.9725	<1e-4	1.0000	<1e-4	40.4803	1.0068
	20	0.9494	<1e-4	1.0000	<1e-4	45.5781	0.2780
	50	0.9389	<1e-4	1.0000	<1e-4	58.8167	1.1234
	100	0.9628	<1e-4	1.0000	<1e-4	82.6800	0.4657
MixLarge20	10	0.9366	<1e-4	0.9998	<1e-4	127.3219	7.5445
	20	0.9409	<1e-4	0.9998	<1e-4	158.8744	8.2583
	50	0.9401	<1e-4	0.9998	<1e-4	234.8413	5.5816
	100	0.9469	<1e-4	0.9998	<1e-4	366.1205	12.2402
MixLarge50	10	0.9410	<1e-4	0.9996	<1e-4	449.9749	0.2353
	20	0.9298	<1e-4	0.9996	<1e-4	571.5291	0.3471
	50	0.9219	<1e-4	0.9996	<1e-4	951.9568	0.4260
	100	0.9287	<1e-4	0.9996	<1e-4	1530.6935	0.4041

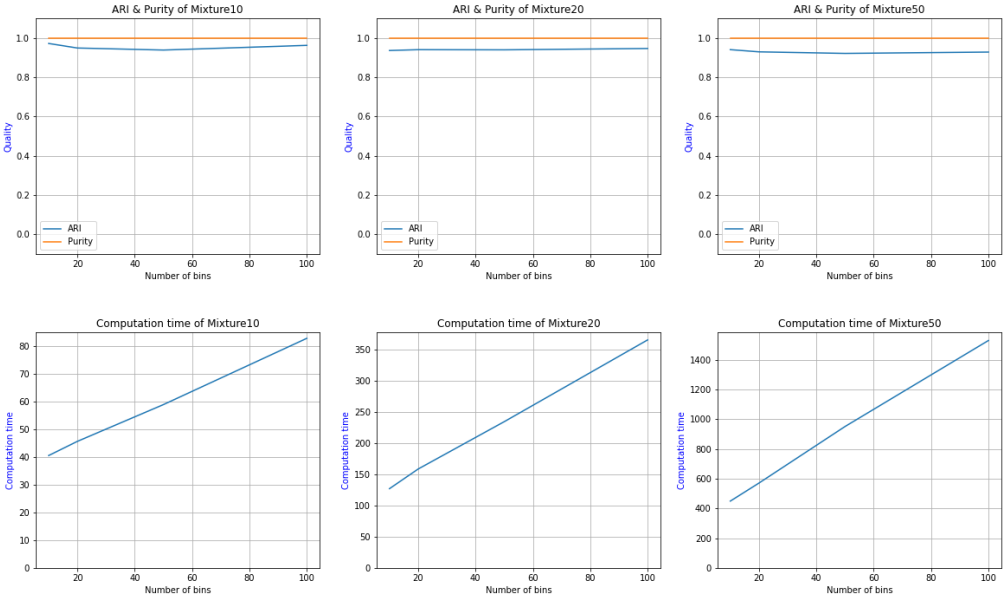

Figure 6: Results with increasing the number of bins – Artificial datasets

Table 4
Results with increasing the number of bins – Real datasets

datasets	NbB	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Outdoor	10	0.5169	<1e-4	0.7870	<1e-4	2.7760	0.1297
	20	0.5304	<1e-4	0.7849	<1e-4	3.6505	0.0690
	50	0.5279	<1e-4	0.7820	<1e-4	7.1185	0.2118
	100	0.5046	<1e-4	0.7817	<1e-4	13.3744	0.6726
Gassenor	10	0.0584	<1e-4	0.9787	<1e-4	49.0299	0.2895
	20	0.0601	<1e-4	0.9770	<1e-4	66.8477	0.8443
	50	0.0607	<1e-4	0.9777	<1e-4	121.3673	0.9275
	100	0.0604	<1e-4	0.9744	<1e-4	215.2406	8.5638
IGBN	10	0.2089	<1e-4	0.4753	<1e-4	6.6979	0.8273
	20	0.2429	<1e-4	0.4829	<1e-4	6.8183	0.4002
	50	0.2254	<1e-4	0.4782	<1e-4	6.1710	0.1387
	100	0.2235	<1e-4	0.4800	<1e-4	6.1976	0.0606
IABN	10	0.1219	<1e-4	0.3033	<1e-4	10.7201	0.6364
	20	0.0884	<1e-4	0.3051	<1e-4	10.1940	0.2250
	50	0.1099	<1e-4	0.3031	<1e-4	10.7351	0.6380
	100	0.1124	<1e-4	0.3051	<1e-4	9.8895	0.1472

datasets	NbB	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Rialto	10	0.0362	<1e-4	0.3103	<1e-4	8.3314	0.5049
	20	0.0370	<1e-4	0.3103	<1e-4	10.2060	0.2524
	50	0.0354	<1e-4	0.3103	<1e-4	13.9765	0.8070
	100	0.0354	<1e-4	0.3103	<1e-4	16.7267	0.6884
IIAIN	10	0.0080	<1e-4	0.4008	<1e-4	256.6696	18.6691
	20	0.0085	<1e-4	0.4008	<1e-4	283.7149	10.1869
	50	0.0089	<1e-4	0.4014	<1e-4	291.8349	11.0537
	100	0.0089	<1e-4	0.4014	<1e-4	268.2696	31.4180
IIRIN	10	0.0392	<1e-4	0.4445	<1e-4	232.7974	14.1282
	20	0.0440	<1e-4	0.4445	<1e-4	251.4768	11.6803
	50	0.0355	<1e-4	0.4445	<1e-4	258.4078	10.4113
	100	0.0410	<1e-4	0.4421	<1e-4	249.5936	11.7942
Covtype	10	-0.0097	<1e-4	0.4924	<1e-4	89.2167	1.1697
	20	-0.0096	<1e-4	0.4924	<1e-4	88.9489	1.0544
	50	-0.0116	<1e-4	0.4924	<1e-4	89.3247	0.9790
	100	-0.0164	<1e-4	0.4924	<1e-4	94.3237	4.3868

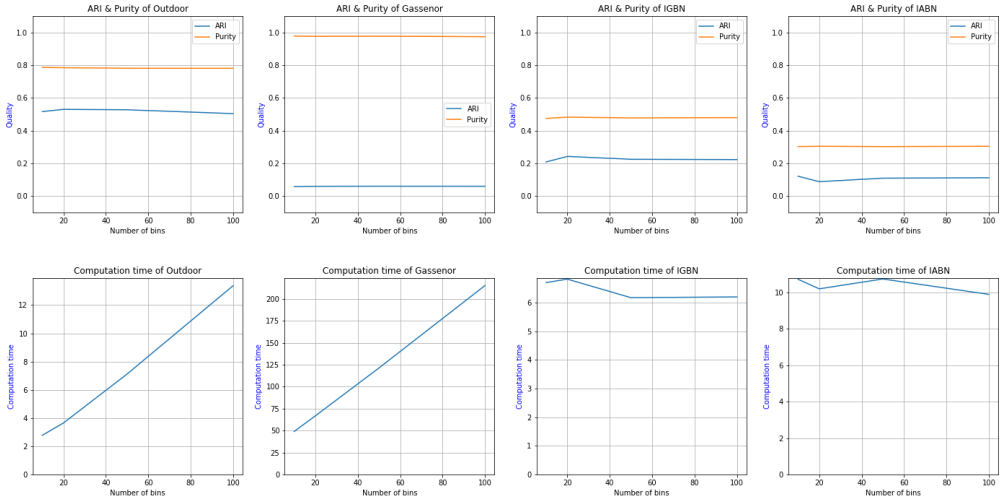

Figure 7: Results with increasing the number of bins – Real datasets (PART I)

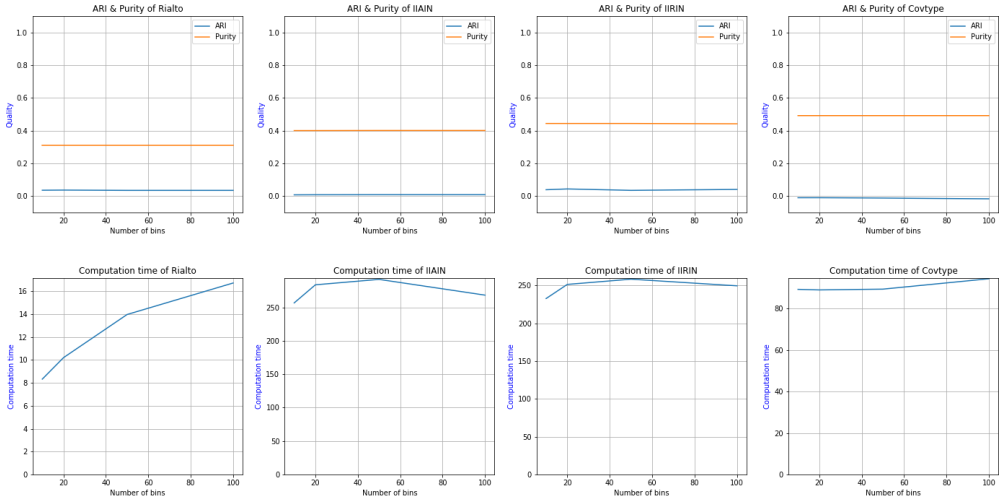

Figure 8: Results with increasing the number of bins – Real datasets (PART II)

4.3.2. Number of projections

We found that the quality of clustering does not vary significantly when the number of projections is changed, both on real and artificial datasets, according to the results shown in Tab.5, Tab.6 and in Fig.9, Fig.10, Fig.11. This seems to indicate that even with a significant reduction in the dimensionality of the data, the richness of the histogram representation of the cluster distributions is sufficient to efficiently detect clusters with similar distributions. In terms of computational time, the algorithm is faster than the version without projections when the number of projections is smaller than the number of variables in the datasets. In addition, as expected, the computation time increases as the number of projections increases. Based on these observations, we decided to set the number of projections to 4 for following experiments.

Table 5
Results with increasing the number of projections - Artificial datasets

datasets	NbP	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
MixLarge10	0	0.9671	<1e-4	0.9997	<1e-4	289.6916	13.0346
	4	0.9897	<1e-4	1.0000	<1e-4	61.9560	4.2505
	8	0.9718	<1e-4	1.0000	<1e-4	58.4983	2.3194
	12	0.9502	<1e-4	1.0000	<1e-4	66.6776	0.5768
	16	0.9685	<1e-4	1.0000	<1e-4	78.0780	1.0313
	20	0.9765	<1e-4	1.0000	<1e-4	78.7753	2.0131
MixLarge20	0	0.9653	<1e-4	0.9997	<1e-4	678.6437	15.4316
	4	0.9728	<1e-4	0.9998	<1e-4	144.3213	6.0264
	8	0.9864	<1e-4	0.9998	<1e-4	165.5495	2.0468
	12	0.9790	<1e-4	0.9998	<1e-4	183.1069	3.2795
	16	0.9892	<1e-4	0.9998	<1e-4	200.9902	2.0494
	20	0.9784	<1e-4	0.9998	<1e-4	229.8228	1.2519
MixLarge50	0	0.9634	<1e-4	0.9996	<1e-4	2513.5941	65.9987
	4	0.9372	<1e-4	0.9998	<1e-4	543.7881	9.3786
	8	0.9522	<1e-4	0.9998	<1e-4	637.2857	22.3597
	12	0.9594	<1e-4	0.9998	<1e-4	693.4370	4.7394
	16	0.9586	<1e-4	0.9998	<1e-4	811.1516	30.8824
	20	0.9671	<1e-4	0.9998	<1e-4	854.3131	31.4934

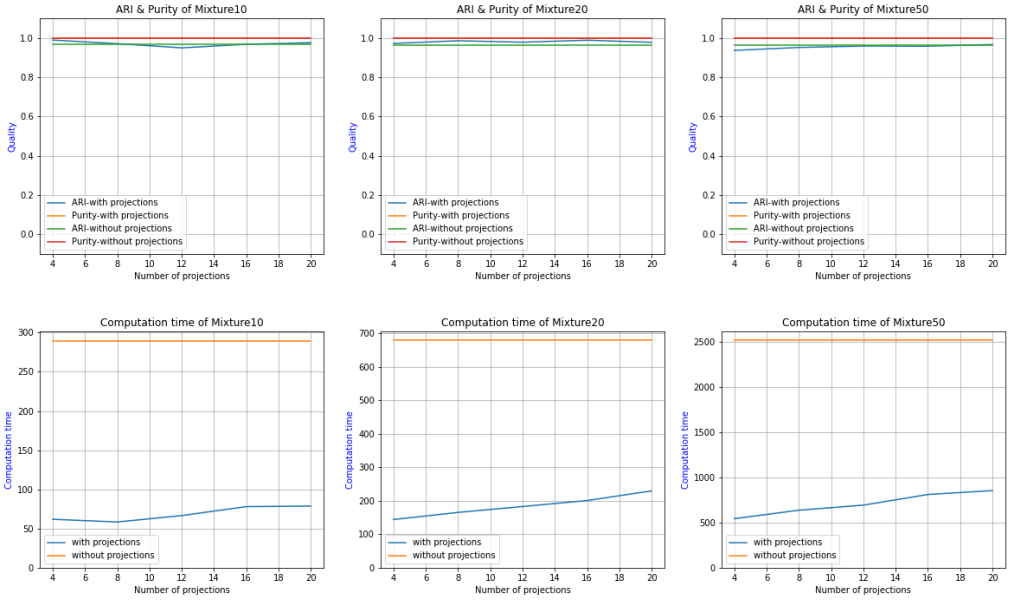

Figure 9: Results with increasing the number of projections – Artificial datasets

Table 6
Results with increasing the number of projections - Real datasets

datasets	NbP	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Outdoor	0	0.4993	<1e-4	0.7792	<1e-4	9.6239	0.2244
	4	0.5578	<1e-4	0.7793	<1e-4	1.6219	0.0096
	8	0.5595	<1e-4	0.7793	<1e-4	3.0240	0.0066
	12	0.5574	<1e-4	0.7765	<1e-4	4.4216	0.0673
	16	0.5577	<1e-4	0.7765	<1e-4	5.6090	0.0991
	20	0.5644	<1e-4	0.7785	<1e-4	6.9491	0.0187
Gassenor	0	0.0945	<1e-4	0.9593	<1e-4	339.4042	3.4804
	4	0.0874	<1e-4	0.9585	<1e-4	12.4324	0.0400
	8	0.0910	<1e-4	0.9585	<1e-4	22.0261	0.2735
	12	0.0886	<1e-4	0.9593	<1e-4	32.1096	0.6993
	16	0.0859	<1e-4	0.9593	<1e-4	40.8242	0.2256
	20	0.0870	<1e-4	0.9593	<1e-4	50.7802	0.1517
IGBN	0	0.3116	<1e-4	0.5980	<1e-4	5.5325	0.0672
	4	0.3057	<1e-4	0.5980	<1e-4	4.5977	0.0059
	8	0.3290	<1e-4	0.5980	<1e-4	4.7244	0.0538
	12	0.3031	<1e-4	0.5980	<1e-4	4.9214	0.1328
	16	0.3017	<1e-4	0.5980	<1e-4	4.9832	0.0221
	20	0.2936	<1e-4	0.5980	<1e-4	5.1623	0.0058
IABN	0	0.1013	<1e-4	0.3080	<1e-4	10.1832	0.2239
	4	0.1051	<1e-4	0.3073	<1e-4	9.0350	0.0137
	8	0.1111	<1e-4	0.3051	<1e-4	9.2985	0.1772
	12	0.0931	<1e-4	0.3051	<1e-4	9.2718	0.0286
	16	0.0938	<1e-4	0.3055	<1e-4	9.4350	0.1233
	20	0.0957	<1e-4	0.3051	<1e-4	9.4273	0.0516

datasets	NbP	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Rialto	0	0.0214	<1e-4	0.7164	<1e-4	1322.2720	9.9595
	4	0.0505	<1e-4	0.7017	<1e-4	130.9979	0.4462
	8	0.0477	<1e-4	0.7082	<1e-4	269.1912	8.6474
	12	0.0490	<1e-4	0.7095	<1e-4	381.9081	2.8852
	16	0.0473	<1e-4	0.7111	<1e-4	542.3676	8.2774
	20	0.0456	<1e-4	0.7108	<1e-4	627.4022	9.3890
IIAIN	0	0.0079	<1e-4	0.3628	<1e-4	221.7884	1.6728
	4	0.0109	<1e-4	0.3628	<1e-4	219.9474	6.8324
	8	0.0070	<1e-4	0.3628	<1e-4	239.6036	14.8937
	12	0.0084	<1e-4	0.3628	<1e-4	231.2133	2.7645
	16	0.0101	<1e-4	0.3628	<1e-4	243.3415	3.2499
	20	0.0081	<1e-4	0.3628	<1e-4	239.5649	1.6757
IIRIN	0	0.0280	<1e-4	0.4445	<1e-4	242.6675	12.3442
	4	0.0487	<1e-4	0.4445	<1e-4	228.7733	2.3927
	8	0.0387	<1e-4	0.4445	<1e-4	233.9548	4.3951
	12	0.0390	<1e-4	0.4445	<1e-4	230.4332	4.4831
	16	0.0390	<1e-4	0.4445	<1e-4	221.2432	1.8827
	20	0.0388	<1e-4	0.4445	<1e-4	222.4964	7.9179
Covtype	0	-0.0053	<1e-4	0.4924	<1e-4	87.4430	0.1671
	4	-0.0188	<1e-4	0.4924	<1e-4	88.1170	1.2591
	8	-0.0139	<1e-4	0.4924	<1e-4	93.5260	0.8076
	12	-0.0148	<1e-4	0.4924	<1e-4	94.6759	0.2335
	16	-0.0112	<1e-4	0.4924	<1e-4	94.3640	0.3727
	20	-0.0086	<1e-4	0.4924	<1e-4	95.0573	0.7504

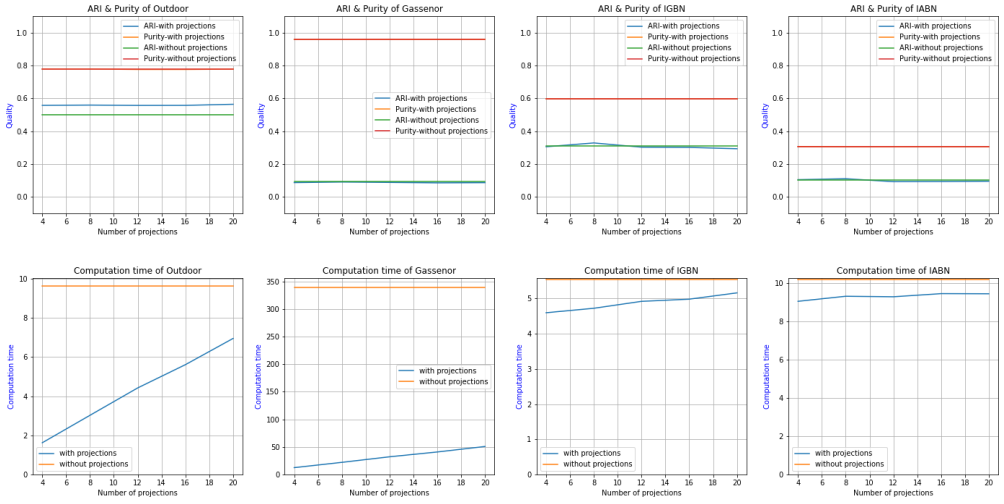

Figure 10: Results with increasing the number of projections – Real datasets (PART I)

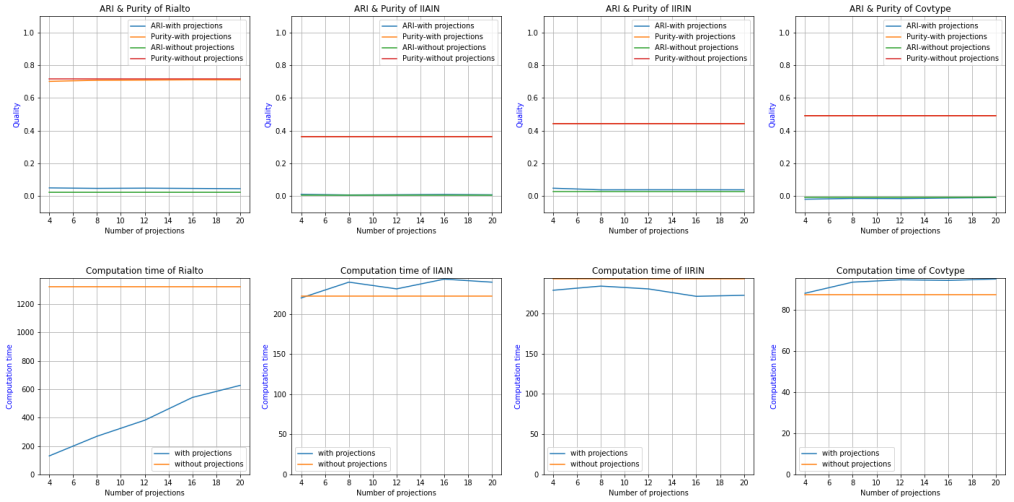

Figure 11: Results with increasing the number of projections – Real datasets (PART II)

4.3.3. Window size

When the datasets are divided into more windows, the proposed cluster similarity test is applied more often on clusters with few samples, potentially leading to more errors, but the process is faster. The aim of testing the effect of changing the number of windows is the same as testing the effect of the number of samples in a window, as well as the effect of the number of times we apply the similarity test. As can be seen in Tab.7, Tab.8 and Fig.12, Fig.13, Fig.14, there is a gradual decrease in quality as the number of windows increases, as the number of comparisons increases, although there is also an initial dramatic decrease in computation time followed by convergence after a certain number of windows. Therefore, to achieve better performance, we set the number of samples or time intervals in each window differently for different datasets. In our experiments, we carefully choose the number of data points (or time intervals) in each batch to ensure the effectiveness of our approach. Striking a balance is essential; using too few data points can result in scattered points, making it difficult to find clusters. Conversely, too many data points would lead to an over-reliance on standard clustering methods within each batch, potentially obscuring the impact of the proposed approach.

Table 7
Results with increasing the number of windows – Artificial datasets

datasets	NbW	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
MixLarge10	20	0.9024	<1e-4	0.9758	<1e-4	68.7511	2.1029
	50	0.7394	<1e-4	0.9225	<1e-4	40.1313	1.2815
	80	0.7338	<1e-4	0.8740	<1e-4	31.9439	1.9729
	110	0.6302	<1e-4	0.8208	<1e-4	31.3047	0.6891
	140	0.5239	<1e-4	0.7629	<1e-4	30.7979	0.7231
MixLarge20	20	0.9458	<1e-4	0.9958	<1e-4	212.3783	30.5535
	50	0.8601	<1e-4	0.9706	<1e-4	113.7186	5.0568
	80	0.8927	<1e-4	0.9515	<1e-4	91.2998	6.8736
	110	0.8368	<1e-4	0.9128	<1e-4	83.2120	3.5792
	140	0.7976	<1e-4	0.8654	<1e-4	73.2229	4.4372
MixLarge50	20	0.9288	<1e-4	0.9970	<1e-4	933.1498	38.5705
	50	0.9320	<1e-4	0.9864	<1e-4	372.6099	0.2560
	80	0.8826	<1e-4	0.9797	<1e-4	263.3228	0.3576
	110	0.8491	<1e-4	0.9746	<1e-4	219.4075	0.3878
	140	0.8254	<1e-4	0.9670	<1e-4	189.6720	0.2960

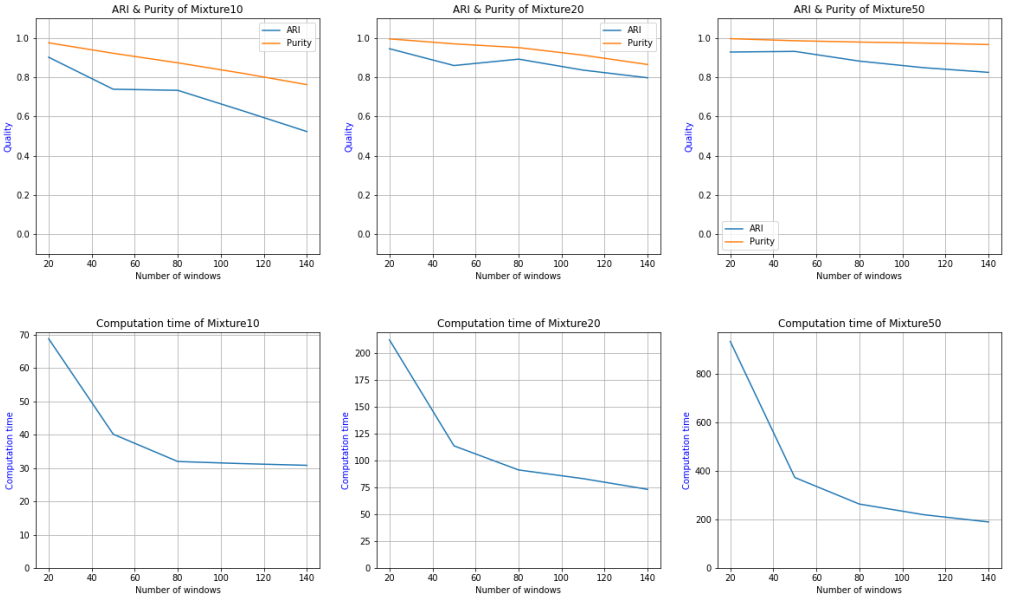

Figure 12: Results with increasing the number of windows – Artificial datasets

Table 8
Results with increasing the number of windows - Real datasets

datasets	NbW	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Outdoor	10	0.4367	<1e-4	0.7295	<1e-4	2.4940	0.0566
	20	0.4337	<1e-4	0.6508	<1e-4	2.2888	0.0750
	30	0.3364	<1e-4	0.5564	<1e-4	2.2959	0.0837
	40	0.2639	<1e-4	0.4925	<1e-4	2.0966	0.0505
	50	0.2656	<1e-4	0.4691	<1e-4	1.8918	0.0208
Gassenor	10	0.0462	<1e-4	0.9830	<1e-4	71.1566	4.4826
	20	0.0613	<1e-4	0.9631	<1e-4	60.7260	2.8672
	30	0.0687	<1e-4	0.9431	<1e-4	56.6898	3.3096
	40	0.0702	<1e-4	0.8658	<1e-4	45.0002	5.6714
	50	0.0778	<1e-4	0.8437	<1e-4	36.0551	2.9504
IGBN	10	0.2892	<1e-4	0.5956	<1e-4	5.5114	0.6831
	20	0.3514	<1e-4	0.6697	<1e-4	3.4527	0.1802
	30	0.3504	<1e-4	0.6754	<1e-4	3.5497	0.5232
	40	0.2851	<1e-4	0.6592	<1e-4	3.6314	0.1029
	50	0.3498	<1e-4	0.6789	<1e-4	1.8776	0.2037
IABN	10	0.1036	<1e-4	0.3011	<1e-4	19.5458	1.7214
	20	0.1151	<1e-4	0.3088	<1e-4	12.1004	0.8460
	30	0.1127	<1e-4	0.3145	<1e-4	9.2761	0.5667
	40	0.1067	<1e-4	0.3175	<1e-4	7.3438	0.3885
	50	0.1237	<1e-4	0.3199	<1e-4	6.6326	1.1436

datasets	NbW	ARI		Purity		Computation time	
		mean	std	mean	std	mean	std
Rialto	10	0.0773	<1e-4	0.3502	<1e-4	10.2090	0.5407
	20	0.0572	<1e-4	0.3206	<1e-4	9.1944	0.1972
	30	0.0694	<1e-4	0.3680	<1e-4	8.6354	0.4201
	40	0.0857	<1e-4	0.4282	<1e-4	9.0779	0.4252
	50	0.1358	<1e-4	0.4599	<1e-4	9.5875	0.5672
IIAIN	10	0.0196	<1e-4	0.3988	<1e-4	311.4166	8.9169
	20	0.0171	<1e-4	0.4184	<1e-4	207.5998	6.8795
	30	0.0167	<1e-4	0.4586	<1e-4	158.4947	3.2622
	40	0.0686	<1e-4	0.5964	<1e-4	129.5866	4.0756
	50	0.0558	<1e-4	0.6018	<1e-4	98.1429	5.6320
IIRIN	20	0.0405	<1e-4	0.4413	<1e-4	263.9345	15.3767
	50	0.0482	<1e-4	0.4478	<1e-4	221.1476	2.5597
	80	0.0503	<1e-4	0.4499	<1e-4	152.4280	2.3740
	110	0.0526	<1e-4	0.4523	<1e-4	130.2410	3.0180
	140	0.0495	<1e-4	0.4579	<1e-4	96.4369	1.1988
Covtype	20	0.0401	<1e-4	0.6189	<1e-4	171.5827	8.3167
	50	0.0059	<1e-4	0.5720	<1e-4	123.6715	10.3234
	80	0.0048	<1e-4	0.5727	<1e-4	98.7537	3.2652
	110	0.0032	<1e-4	0.5911	<1e-4	85.1102	2.2618
	140	0.0218	<1e-4	0.6138	<1e-4	76.2572	1.7298

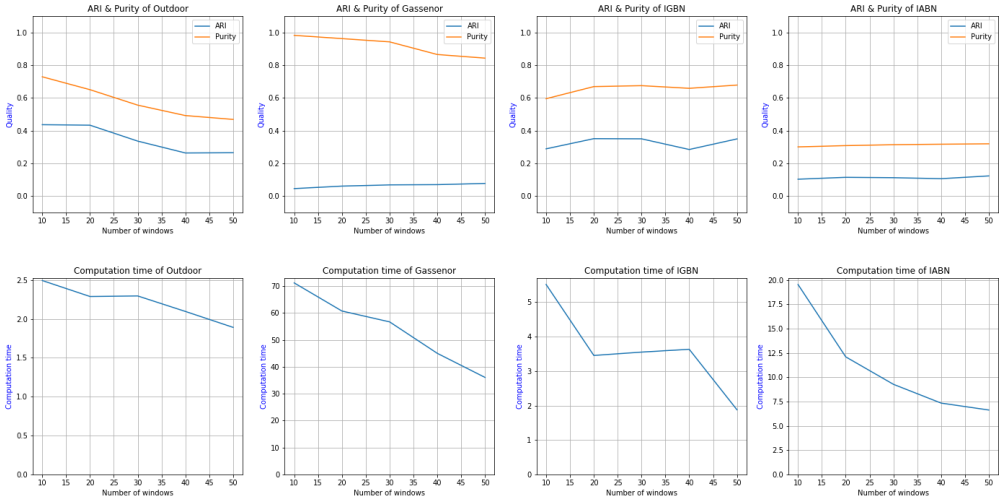

Figure 13: Results with increasing the number of windows – Real datasets (PART I)

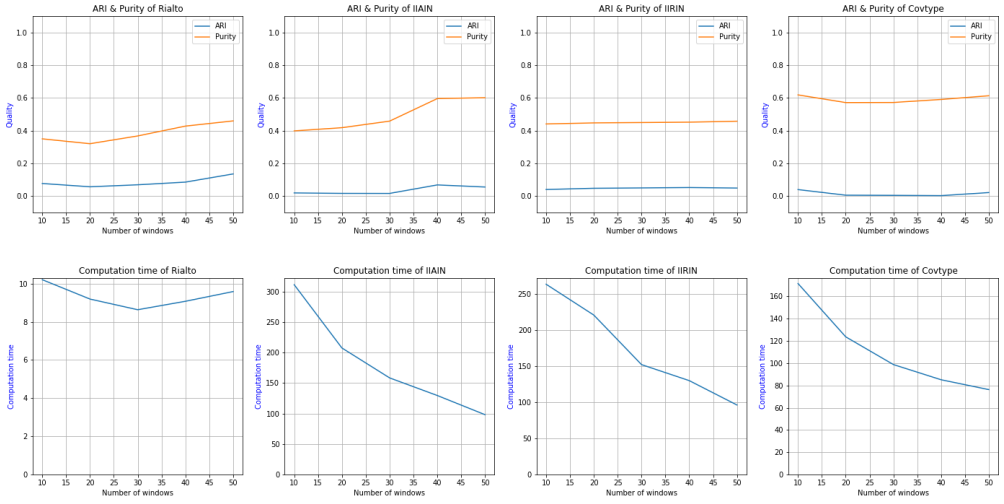

Figure 14: Results with increasing the number of windows – Real datasets (PART II)

To address this, we have developed a systematic configuration for the experiments on both static datasets and dynamic datasets. For smaller datasets, we ensure that the number of data points in each batch is approximately 10% of the total number. For larger datasets, this proportion is reduced to 5%. For artificial datasets, we keep the percentage constant. However, real-world datasets may consist of varying amounts of samples, we allow for slight adjustments to avoid inconsistent number of batches in the analysis. The number of samples in each window for the batch clustering tests are described in Tab.9, the total duration and the chosen time intervals of the windows for the dynamic clustering tests are described in Tab.10.

Table 9
Number of samples in the windows for each dataset

Datasets	Nb_samples	Datasets	Nb_samples	Datasets	Nb_samples	Datasets	Nb_samples
Comet3	3000	Comet20	10000	Meteorite3	3000	Meteorite20	10000
Square3	3000	Square20	10000	Gaussian3	3000	Gaussian20	10000
Moon3	3000	Moon20	10000	Circle3	3000	Circle20	10000
MixSmall1	1000	MixSmall2	2000	MixSmall3	3000	MixLarge10	5000
MixLarge20	10000	MixLarge30	20000	Outdoor	1500	Gassenor	2000
IGBN	3000	IABN	6000	Rialto	8000	IIAIN	8000
IIRIN	40000	Covtype	60000				

Table 10
Total duration and time interval of the windows for each dataset

Datasets	Total_dur	Time_inter	Datasets	Total_dur	Time_inter	Datasets	Total_dur	Time_inter
Comet20	200	4	Meteorite20	200	4	Square20	200	4
Gaussian20	200	4	Moon20	200	4	Circle20	200	4
MixLarge10	100	5	MixLarge20	200	6	MixLarge50	500	7
Outdoor	30	6	Gassenor	50	5	IGBN	50	5
IABN	80	4	Rialto	80	4	IIAIN	150	5
IIRIN	150	5	Covtype	150	5			

4.4. Quality of the distribution comparisons

We performed an experiment to confirm that the statistical test we use to identify similarity between identical distributions works correctly, as this is important for the proposed framework. First, we create pairs of clusters drawn from the same distribution. Each pair is constructed with different distribution types and dimensions. We then apply the similarity test to these pairs. As expected, the results show that the percentage of similar clusters discovered for each distribution is about 95%, in accordance with the chosen threshold $\delta = 5\%$ (see Fig.15).

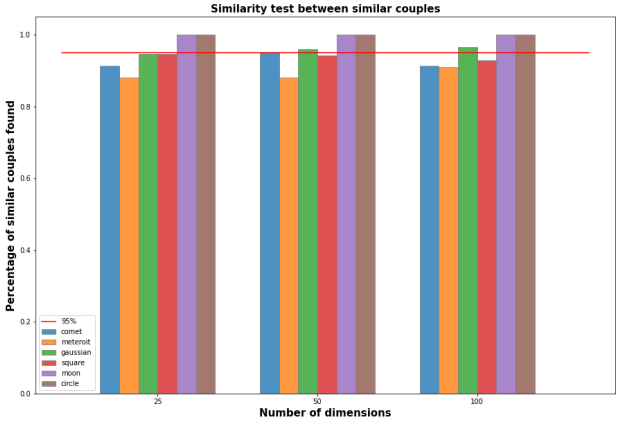

Figure 15: Similarity tests between similar distribution. The red line represent 95% of correct detection.

We also evaluated the robustness of the similarity detection by shifting the position of a distribution by a fraction of its standard deviation σ over a random axis, to check at what level the two clusters are treated as not similar. therefore, we fix a cluster distribution and shift the position of the cluster by $1+\sigma$*std_per. After each shift, we compute the similarity between the two cluster distributions (Fig.16).

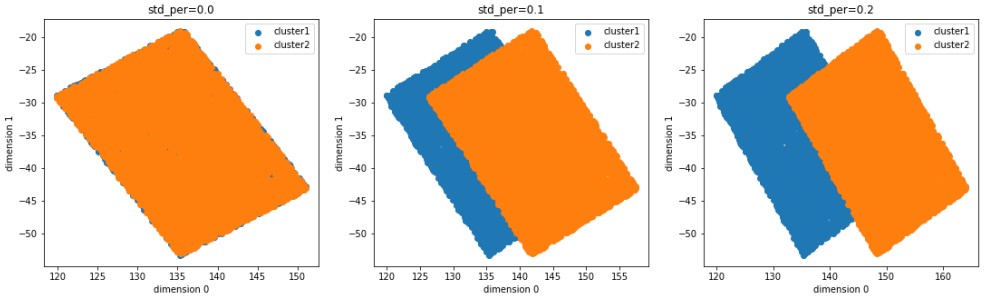

Figure 16: Example of a similarity test between two similar clusters by changing the standard deviation percentage (std_per) to change the position of the second cluster.

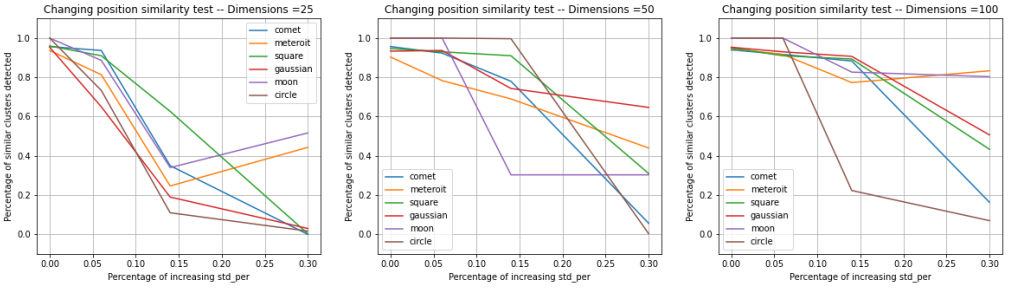

Figure 17: Similarity test on distinct distributions of different dimension numbers by increasing std_per.

For datasets with different dimensions (see Fig.17), as long as the shift remains below about 10% of the standard deviation, we observe only a slow decrease in the proportion of pairs detected as similar, from 95% to 80%. If we further increase the shift, we observe, as expected, a sharp decrease in the proportion of pairs detected as similar.

4.5. Assessment of algorithm quality and complexity

We performed a series of experiments to evaluate the quality and complexity of the resulting clustering in comparison to existing approaches. The proposed framework can be used with any clustering algorithm, depending on the required cluster properties, by working on subsets (windows) of data and comparing the obtained clusters between windows. For static datasets, we expect the proposed framework to reduce the computational speed and memory requirements of most existing clustering algorithms, while preserving their interesting properties, with minimal loss of quality. For dynamic datasets, we expect the proposed framework to be competitive with existing stream clustering approaches in terms of quality and complexity, while allowing great flexibility in the choice of the clustering algorithm to be applied.

4.5.1. Experiments on static datasets

As mentioned above, our approach relies on sliding windows and the basic idea is to first perform conventional clustering on each window, then represent the resulting clusters in a histogram model and merge the windows to obtain the final clustering results. This part of the test compares conventional clustering algorithms applied to the full dataset with the proposed approach using the same clustering algorithms applied to sliding windows (illustrated in Fig.18). The following clustering algorithms were chosen: BIRCH, Agglomerative clustering, OPTICS, KMeans, Gaussian Mixture and HDBSCAN for the comparisons in this section. BIRCH and Agglomerative clustering are only evaluated on short datasets due to their space requirements, while Kmeans and Gaussian Mixture are well suited to large datasets. OPTICS and HDBSCAN both take a long time to run on large datasets. However, since HDBSCAN is still faster than OPTICS, we decided to test OPTICS only on small datasets and HDBSCAN on large datasets. The quality of the clustering was again evaluated using the ARI and Purity indices. The computation time is given in seconds. For all experiments, the mean and standard deviation of the index over five replicates are shown in the tables.

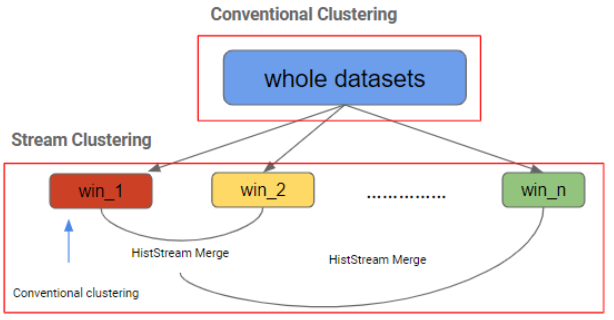

Figure 18: Experiment of clustering on static datasets

The Adjusted Rand Index (ARI) results in Tab.11, Tab.12 and Purity results in Tab.13, Tab.14 of the proposed approach are consistently close to the results of the original clustering, in some cases the proposed approach outperforms conventional clustering. However, the slight losses in quality observed in some cases are compensated by a (sometimes considerable) gain in computing time, as shown in Tab.15, Tab.16. Overall, the clustering process is greatly accelerated in the proposed framework for both real and artificial datasets, without significant loss of quality. This appears to be consistent across a large family of clustering algorithms, allowing for great flexibility in the use of the framework. To provide a comprehensive visualization of the overall comparison between the proposed and the original clustering approach, we computed the mean value of each index and presented the results in the form of histograms. The purpose was to take a general view of differences between them. The histogram results are shown in Fig.19, Fig.20 and Fig.21.

Table 11
ARI results of static clustering experiments - small datasets

Datasets	Birch				Agglomerative				OPTICS			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	0.9304	<1e-4	0.959	<1e-4	0.9325	<1e-4	0.9584	<1e-4	0.3366	<1e-4	0.7455	<1e-4
Meteorit3	1	<1e-4	1.0	<1e-4	1	<1e-4	0.9991	<1e-4	1	<1e-4	0.8356	<1e-4
Circle3	0.9815	<1e-4	0.9632	<1e-4	0.9811	<1e-4	0.9661	<1e-4	0.0254	<1e-4	0.9454	<1e-4
Gaussian3	1	<1e-4	0.8958	<1e-4	1	<1e-4	0.7091	<1e-4	1	<1e-4	0.9994	<1e-4
Square3	1	<1e-4	0.9779	<1e-4	1	<1e-4	1.0	<1e-4	1	<1e-4	0.9932	<1e-4
Moon3	1.0	<1e-4	0.9035	<1e-4	1.0	<1e-4	0.9043	<1e-4	0.8648	<1e-4	0.9775	<1e-4
MixSmall1	0.9818	<1e-4	0.955	<1e-4	0.9792	<1e-4	0.9563	<1e-4	0.6958	<1e-4	0.9526	<1e-4
MixSmall2	0.9272	<1e-4	0.8787	<1e-4	0.9306	<1e-4	0.8808	<1e-4	0.9881	<1e-4	0.955	<1e-4
MixSmall3	0.9944	<1e-4	0.9839	<1e-4	0.9944	<1e-4	0.9838	<1e-4	0.9957	<1e-4	0.9842	<1e-4
Outdoor	0.0032	<1e-4	0.4246	<1e-4	0.4044	<1e-4	0.4446	<1e-4	0.2483	<1e-4	0.2648	<1e-4
Gassenor	0.1266	<1e-4	0.2747	<1e-4	0.1266	<1e-4	0.2837	<1e-4	0.0126	<1e-4	0.0218	<1e-4
IGBN	0.0092	<1e-4	0.2859	<1e-4	0.0653	<1e-4	0.227	<1e-4	0.006	<1e-4	0.0127	<1e-4
IABN	0.1106	<1e-4	0.0999	<1e-4	0.1098	<1e-4	0.062	<1e-4	0.0032	<1e-4	0.0074	<1e-4
Total mean	0.6973	-	0.7386	-	0.7326	-	0.7212	-	0.552	-	0.6689	-

Table 12
ARI results of static clustering experiments - large datasets

Datasets	KMeans				GaussianMixture				HDBSCAN			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	1.0	<1e-4	0.931	<1e-4	0.9789	0.0149	0.9318	0.0141	0.2997	<1e-4	0.9358	<1e-4
Meteorit20	1.0	<1e-4	0.9873	0.0001	0.9983	0.0024	0.9632	0.0172	0.7406	<1e-4	0.9476	<1e-4
Circle20	1.0	<1e-4	0.7651	0.0008	0.911	<1e-4	0.7795	0.0119	0.1399	<1e-4	0.9059	<1e-4
Gaussian3	1.0	<1e-4	0.9742	<1e-4	1.0	<1e-4	0.9782	0.0038	0.9087	<1e-4	0.9823	<1e-4
Square20	1.0	<1e-4	0.9843	0.0005	1.0	<1e-4	0.9446	0.011	0.6336	<1e-4	0.9635	<1e-4
Moon20	1.0	<1e-4	0.9754	<1e-4	1.0	<1e-4	0.972	0.0048	0.2689	<1e-4	0.9635	<1e-4
MixLarge10	1.0	<1e-4	0.9764	0.0008	0.9935	0.0046	0.9319	0.0129	0.4662	<1e-4	0.9749	<1e-4
MixLarge20	0.9358	0.0018	0.8832	0.0053	0.9395	0.0026	0.8639	0.0103	0.1874	<1e-4	0.9789	<1e-4
MixLarge50	0.9479	<1e-4	0.8555	0.0039	0.9261	0.0089	0.792	0.0039	0.1769	<1e-4	0.9475	<1e-4
Outdoor	0.4108	0.0076	0.4535	0.0115	0.4322	<1e-4	0.454	0.0092	0.0	<1e-4	0.5633	<1e-4
Gassenor	0.138	<1e-4	0.2776	0.008	0.1231	<1e-4	0.3065	0.0112	0.3829	<1e-4	0.0442	<1e-4
IGBN	0.0805	0.0003	0.2171	0.0092	0.1663	<1e-4	0.2491	0.0066	0.0	<1e-4	0.306	<1e-4
IABN	0.0998	0.0175	0.0779	0.0011	0.2617	<1e-4	0.1133	0.0142	0.0	<1e-4	0.1165	<1e-4
Rialto	0.0677	<1e-4	0.0632	0.002	0.0579	<1e-4	0.048	0.0006	0.0558	<1e-4	0.0809	<1e-4
IIAIN	0.0485	0.0002	0.0213	0.0008	0.2986	<1e-4	0.0319	0.0016	0.0	<1e-4	0.0209	<1e-4
IIRIN	0.0485	0.0001	0.0369	0.0013	0.1781	<1e-4	0.0567	0.0009	0.0	<1e-4	0.0728	<1e-4
Covtype	0.0144	0.0001	0.0146	<1e-4	0.0197	<1e-4	0.0147	<1e-4	0.0	<1e-4	0.0227	<1e-4
Total mean	0.576	-	0.5585	-	0.605	-	0.5548	-	0.2506	-	0.5781	-

Table 13
Purity results of static clustering experiments - small datasets

Datasets	Birch				Agglomerative				OPTICS			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	0.999	<1e-4	0.9996	<1e-4	0.999	<1e-4	0.9996	<1e-4	1.0	<1e-4	1.0	<1e-4
Meteorit3	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4
Circle3	0.9985	<1e-4	0.9989	<1e-4	0.9985	<1e-4	0.9993	<1e-4	1.0	<1e-4	1.0	<1e-4
Gaussian3	1	<1e-4	0.9997	<1e-4	1	<1e-4	0.9997	<1e-4	1	<1e-4	1.0	<1e-4
Square3	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4
Moon3	1	<1e-4	0.9977	<1e-4	1	<1e-4	0.9977	<1e-4	1	<1e-4	1.0	<1e-4
MixSmall1	0.9987	<1e-4	0.9993	<1e-4	0.9987	<1e-4	0.9993	<1e-4	1.0	<1e-4	1.0	<1e-4
MixSmall2	0.9983	<1e-4	0.9988	<1e-4	0.9983	<1e-4	0.9989	<1e-4	1.0	<1e-4	1.0	<1e-4
MixSmall3	0.999	<1e-4	0.9998	<1e-4	0.999	<1e-4	0.9998	<1e-4	1.0	<1e-4	1.0	<1e-4
Outdoor	0.0566	<1e-4	0.6174	<1e-4	0.5643	<1e-4	0.6557	<1e-4	0.9546	<1e-4	0.9314	<1e-4
Gassenor	0.4252	<1e-4	0.6797	<1e-4	0.4252	<1e-4	0.6797	<1e-4	0.9929	<1e-4	0.9866	<1e-4
IGBN	0.2349	<1e-4	0.5568	<1e-4	0.3205	<1e-4	0.6316	<1e-4	0.659	<1e-4	0.6975	<1e-4
IABN	0.3079	<1e-4	0.3743	<1e-4	0.3488	<1e-4	0.4136	<1e-4	0.6698	<1e-4	0.6466	<1e-4
Total mean	0.7706	-	0.8632	-	0.8194	-	0.875	-	0.9443	-	0.9432	-

Table 14
Purity results of static clustering experiments - large datasets

Datasets	KMeans				GaussianMixture				HDBSCAN			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	1.0	<1e-4	1.0	<1e-4	0.9998	0.0002	1.0	0.0001	1.0	<1e-4	1.0	<1e-4
Meteorit20	1.0	<1e-4	1.0	<1e-4	0.9999	0.0001	0.9999	<1e-4	1.0	<1e-4	1.0	<1e-4
Circle20	1.0	<1e-4	0.9999	<1e-4	0.9997	<1e-4	0.9998	0.0001	1.0	<1e-4	0.9999	<1e-4
Gaussian3	1	<1e-4	1.0	<1e-4	1	<1e-4	0.9999	<1e-4	1	<1e-4	1.0	<1e-4
Square20	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4
Moon20	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4	1	<1e-4	1.0	<1e-4
MixLarge10	1.0	<1e-4	0.9997	0.0001	0.9988	0.0008	0.9992	0.0002	1.0	<1e-4	1.0	<1e-4
MixLarge20	0.9995	<1e-4	0.9991	<1e-4	0.9991	0.0005	0.9993	0.0003	1.0	<1e-4	0.9999	<1e-4
MixLarge50	0.9997	<1e-4	0.9998	<1e-4	0.9992	0.0004	0.9996	<1e-4	1.0	<1e-4	1.0	<1e-4
Outdoor	0.573	0.0013	0.6554	0.0045	0.5886	<1e-4	0.6722	0.0071	0.0531	<1e-4	0.8765	<1e-4
Gassenor	0.4302	<1e-4	0.6685	0.0013	0.394	<1e-4	0.7273	0.0026	0.8349	<1e-4	0.9868	<1e-4
IGBN	0.3256	0.0002	0.6106	0.0048	0.4095	<1e-4	0.6878	0.0093	0.2841	<1e-4	0.5544	<1e-4
IABN	0.3351	0.0158	0.4579	0.0014	0.4547	<1e-4	0.5443	0.0081	0.2304	<1e-4	0.3002	<1e-4
Rialto	0.2242	0.0002	0.3313	0.0001	0.2053	<1e-4	0.2949	0.0018	0.8237	<1e-4	0.3354	<1e-4
IIAIN	0.3822	0.0002	0.4391	0.0012	0.6605	<1e-4	0.5633	0.0128	0.2983	<1e-4	0.4016	<1e-4
IIRIN	0.3823	<1e-4	0.4991	0.0019	0.5301	<1e-4	0.6189	0.0086	0.2983	<1e-4	0.5235	<1e-4
Covtype	0.4838	0.0001	0.631	0.0004	0.4799	<1e-4	0.6173	<1e-4	0.474	<1e-4	0.5554	<1e-4
Total mean	0.7139	-	0.7818	-	0.7482	-	0.8073	-	0.7233	-	0.7961	-

Table 15
Computation time results of static clustering experiments - small datasets

Datasets	Birch				Agglomerative				OPTICS			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet3	46.3985	5.5252	7.8408	0.2432	127.4078	58.3112	5.0261	0.3873	493.3792	29.1054	163.4676	4.2873
Meteorit3	67.0437	7.5685	8.1117	0.4121	98.8683	3.322	4.8845	0.1614	918.6827	38.0698	150.9396	2.9005
Circle3	58.8365	14.6916	7.6885	0.1152	385.324	380.004	4.6353	0.0457	549.9497	93.6811	200.4851	9.3847
Gaussian3	50.6037	6.429	7.9339	0.2602	149.0743	54.4083	4.9113	0.1212	907.9354	7.8397	156.0655	15.702
Square3	48.9298	13.2206	10.005	0.3386	269.3275	26.8078	5.5559	0.3193	517.4142	57.988	183.2336	5.6596
Moon3	50.6882	8.0583	8.957	0.7769	116.288	12.7159	5.4233	0.4177	616.3526	81.1579	177.2624	2.3017
MixSmall1	10.3563	1.1373	2.0381	0.0417	11.8551	0.8239	1.266	0.0215	227.0029	9.3106	31.7193	0.9437
MixSmall2	54.2983	11.1377	6.5279	0.1404	46.1001	1.5442	4.8924	0.6459	544.6624	77.89	139.1024	4.3467
MixSmall3	90.4021	9.0464	11.9336	0.1881	173.7018	75.2559	8.8753	0.3032	762.1336	181.8225	291.2263	7.3133
Outdoor	0.0625	<1e-4	1.0264	0.0304	0.3124	<1e-4	0.8213	0.0089	4.5836	1.5515	10.9769	0.1763
Gassenor	22.9573	2.1986	3.4747	0.0463	19.5265	1.6089	2.1071	0.0456	335.5294	21.1148	225.4194	8.4252
IGBN	0.7426	0.0064	0.6852	0.0368	34.4075	3.6584	3.4804	0.1498	422.94	131.5297	161.5363	2.3473
IABN	0.9206	0.0009	1.3271	0.0074	48190.2563	3384.3683	15.9417	0.2998	974.3189	0.373	687.5644	21.508
Total mean	38.6339	-	5.9654	-	3817.1115	-	5.217	-	559.6065	-	198.3845	-

Table 16
Computation time results of static clustering experiments - large datasets

Datasets	KMeans				GaussianMixture				HDBSCAN			
	Original		Proposed		Original		Proposed		Original		Proposed	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	23.7342	2.7909	19.3367	1.4584	222.9943	78.1578	65.8177	7.7575	4213.044	171.5366	214.1387	13.7232
Meteorit20	24.1478	1.002	18.9577	0.441	132.9496	34.5254	85.2217	13.1	4406.3028	56.387	222.485	14.6033
Circle20	16.4333	1.3872	20.6704	0.5538	113.1192	2.2735	104.3281	13.3199	3383.8675	187.8437	289.1871	1.5691
Gaussian3	23.5632	0.4888	14.6609	0.2494	115.2474	1.3591	58.9519	5.7402	3979.6592	184.0601	144.8482	1.269
Square20	34.9393	1.6008	13.419	0.4361	115.2435	1.9313	57.7548	7.9632	5379.3736	519.3697	153.2965	2.101
Moon20	27.8314	4.1625	12.889	0.3042	114.9479	1.6865	44.9077	1.4486	3994.7321	237.7477	142.0085	2.0998
MixLarge10	18.3237	3.3988	14.3273	0.363	72.1962	5.9587	42.0488	6.5004	1205.1306	74.8272	49.496	0.8893
MixLarge20	50.8988	1.6117	26.4944	2.3093	391.9952	121.8181	130.6982	16.6862	5014.7048	408.9864	191.2873	4.0184
MixLarge50	95.487	27.4479	79.6922	5.5884	1076.0308	390.7638	812.5653	88.7552	18195.6259	4904.3931	787.7254	55.2137
Outdoor	0.302	0.0074	1.035	0.0493	0.7661	0.0128	1.9281	0.1739	0.2656	0.0128	2.9949	0.0866
Gassenor	0.4061	0.0663	0.5567	0.0129	19.5284	1.9068	4.7923	0.6046	16.7923	0.3322	71.8682	1.6906
IGBN	0.5071	0.0175	1.1919	0.0778	5.2655	2.509	2.4238	0.4223	12.4552	2.8957	5.7869	0.1156
IABN	0.9893	0.181	1.6901	0.0311	15.5038	1.708	11.095	0.3244	35.2257	2.4508	17.0926	0.0204
Rialto	2.7374	0.3745	3.2711	0.0336	14.2576	1.0034	14.3153	1.2942	12.315	0.1322	10.8885	0.2875
IIAIN	11.1205	0.7061	15.6908	1.4261	262.0022	2.598	220.7938	25.0763	2503.7866	263.0432	367.5863	38.6229
IIRIN	30.0352	2.5654	15.2497	1.9226	292.2506	9.6818	192.2129	15.15	2871.1593	13.8162	353.6794	11.6252
Covtype	12.7106	1.3458	6.836	0.261	88.7571	1.03	28.6675	0.561	3187.8646	138.1439	102.8748	3.6112
Total mean	22.0098	-	15.6452	-	179.5915	-	110.5013	-	3436.0179	-	183.9555	-

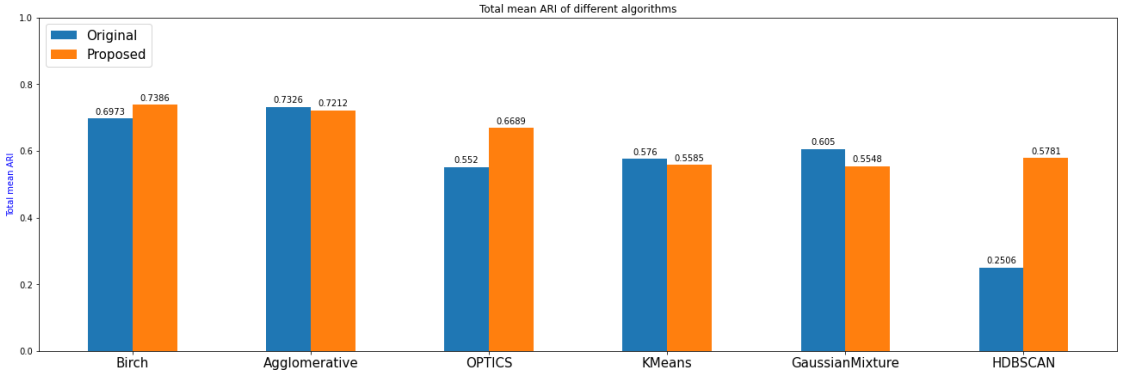

Figure 19: Total mean ARI of different algorithms

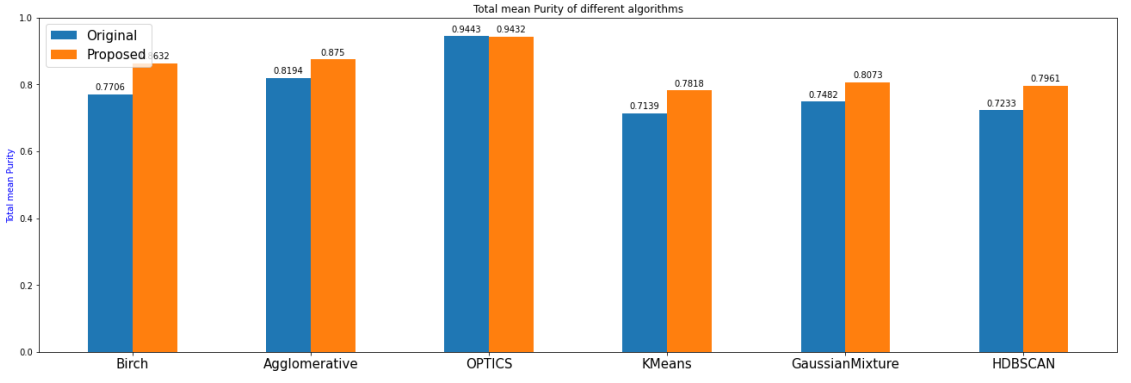

Figure 20: Total mean Purity of different algorithms

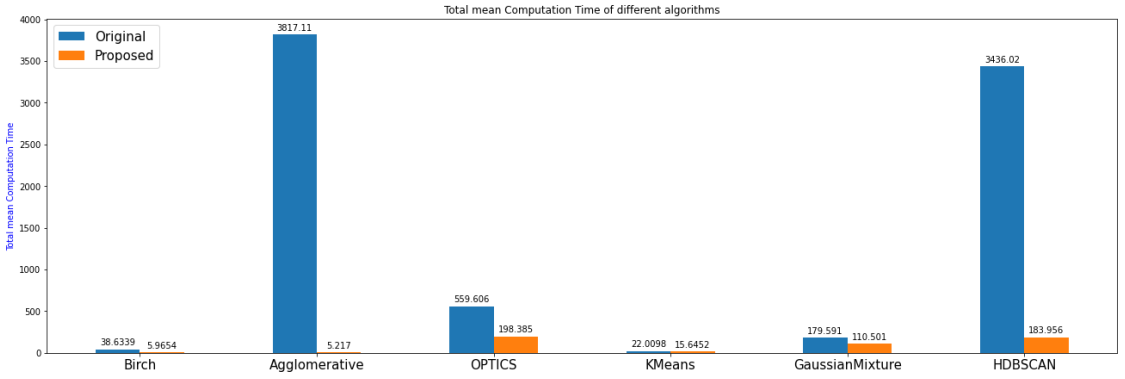

Figure 21: Total mean Computation time of different algorithms

4.5.2. Experiments on dynamic datasets

In order to evaluate whether our approach works well on dynamic datasets and streams, we compared it with traditional approaches such as CluStream and DenStream, presented in section 1, as well as two other approaches already implemented in the Python package River [56]: SequentialKmeans [57], which is a mini-batch KMeans clustering with a new parameter *halflife* that indicates the amount by which the cluster centres should be shifted. The cluster centroid is shifted towards the most recent observation. StreamKmeans [23] is a variant of the original STREAMLSEARCH algorithm, which uses the incremental KMeans algorithm instead of the KMedians obtained by LSEARCH. This allows the algorithm to update KMeans without re-initialising, saving a significant amount of CPU power. We implemented CluStream and DenStream following the parameters proposed in the corresponding paper.

Since there are several possible choices of conventional clustering for our approach, we decided to use HDBSCAN and name this approach HistStream(HDBSCAN) to compare with DenStream, since HDBSCAN is a density-based clustering that can achieve cluster identification for any distribution without the number of clusters as an input. In addition, since KMeans is widely used and is the basis of most partition-based clustering, we also tested KMeans with our framework and name this approach HistStream(KMeans), which could therefore be compared with other partition-based algorithms such as SequentialKmeans, StreamKmeans and CluStream.

Table 17
ARI results of dynamic clustering experiments

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	0.7955	<1e-4	0.677	0.0005	0.8516	0.0107	0.9713	<1e-4	0.9321	<1e-4	0.9537	<1e-4
Meteorit20	0.8112	<1e-4	0.6125	0.0034	0.8713	0.0021	0.9559	0.0002	0.933	<1e-4	0.8531	<1e-4
Circle20	0.8849	<1e-4	0.7163	0.0027	0.8174	0.0452	0.8399	<1e-4	0.9508	<1e-4	0.8059	<1e-4
Gaussian3	0.9999	<1e-4	0.9996	<1e-4	0.8892	0.0049	0.969	0.0008	0.9798	<1e-4	0.7939	<1e-4
Square20	0.9412	<1e-4	0.7284	0.0058	0.9507	0.0001	0.9784	<1e-4	0.9901	<1e-4	0.9452	<1e-4
Moon20	0.8476	<1e-4	0.5151	0.0091	0.8572	0.008	0.9741	<1e-4	0.944	<1e-4	0.9907	<1e-4
MixLarge10	0.8324	<1e-4	0.7487	0.0103	0.9236	0.0259	0.9566	0.0062	0.9823	<1e-4	0.875	<1e-4
MixLarge20	0.9369	<1e-4	0.885	0.0016	0.8841	0.0215	0.885	0.0038	0.9621	<1e-4	0.9104	<1e-4
MixLarge50	0.8576	<1e-4	0.8507	0.0022	0.0036	<1e-4	0.8696	0.0025	0.9544	<1e-4	0.875	<1e-4
Outdoor	0.0909	<1e-4	0.0477	0.0027	0.367	0.0052	0.45	<1e-4	0.5843	<1e-4	0.1301	<1e-4
Gassenor	0.0385	<1e-4	0.0425	<1e-4	0.0588	0.0002	0.402	0.0001	0.8362	<1e-4	0.1978	<1e-4
IGBN	0.0101	<1e-4	0.2069	0.0892	0.0406	0.0023	0.2076	<1e-4	0.3301	<1e-4	0.0839	<1e-4
IABN	0.0046	<1e-4	0.0605	0.0074	0.0545	0.0003	0.0698	0.0003	0.243	<1e-4	0.0714	<1e-4
Rialto	0.0017	<1e-4	0.0002	0.0001	0.0517	0.0036	0.0514	0.0002	0.0802	<1e-4	0.0655	<1e-4
IIAIN	0.0	<1e-4	0.0087	0.0043	0.0421	0.0077	0.0202	<1e-4	0.0122	<1e-4	0.0395	<1e-4
IIRIN	0.0016	<1e-4	0.0195	0.0116	0.0331	0.002	0.0331	0.0007	0.0318	<1e-4	0.0259	<1e-4
Covtype	0.0063	0.0034	0.0015	0.0004	0.0114	0.0009	0.0057	0.0015	0.0092	<1e-4	0.0	<1e-4
Total mean	0.4742	-	0.4189	-	0.4534	-	0.567	-	0.6327	-	0.5069	-

Table 18
Purity results of dynamic clustering experiments

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	0.8709	<1e-4	0.7531	0.0005	0.9627	0.006	1	<1e-4	1	<1e-4	1.0	<1e-4
Meteorit20	0.859	<1e-4	0.6799	0.0069	0.9079	0.0043	1.0	<1e-4	0.9999	<1e-4	1.0	<1e-4
Circle20	0.9549	<1e-4	0.8063	0.0051	0.9206	0.0047	0.9998	<1e-4	1.0	<1e-4	0.9998	<1e-4
Gaussian3	1.0	<1e-4	0.9997	<1e-4	0.9179	<1e-4	1.0	<1e-4	1.0	<1e-4	0.8565	<1e-4
Square20	0.9301	<1e-4	0.7653	0.0141	0.9535	<1e-4	0.9999	<1e-4	1.0	<1e-4	0.9994	<1e-4
Moon20	0.8918	<1e-4	0.6072	0.0094	0.9181	0.01	1.0	<1e-4	0.9998	<1e-4	0.995	<1e-4
MixLarge10	0.8978	<1e-4	0.833	0.0089	0.9597	0.0088	0.9996	<1e-4	1.0	<1e-4	0.9996	<1e-4
MixLarge20	0.9347	<1e-4	0.8736	0.0018	0.9437	0.0113	0.9998	<1e-4	0.9998	<1e-4	0.9992	<1e-4
MixLarge50	0.9403	<1e-4	0.9206	0.0024	0.1055	<1e-4	0.9998	<1e-4	0.9998	<1e-4	0.9996	<1e-4
Outdoor	0.2031	<1e-4	0.1724	0.0015	0.5004	0.0133	0.6786	<1e-4	0.8928	<1e-4	0.1969	<1e-4
Gassenor	0.3093	<1e-4	0.3181	<1e-4	0.3882	0.0019	0.6922	0.001	0.805	<1e-4	0.4967	<1e-4
IGBN	0.2138	<1e-4	0.3853	0.0738	0.268	0.0105	0.5453	0.0001	0.5743	<1e-4	0.3789	<1e-4
IABN	0.202	<1e-4	0.266	0.0122	0.295	0.0003	0.4523	0.0009	0.5175	<1e-4	0.2772	<1e-4
Rialto	0.1225	<1e-4	0.1086	0.001	0.1953	0.0038	0.3224	0.0001	0.3132	<1e-4	0.2115	<1e-4
IIAIN	0.298	<1e-4	0.3227	0.0075	0.3848	0.0122	0.4284	<1e-4	0.3674	<1e-4	0.4315	<1e-4
IIRIN	0.3156	<1e-4	0.3539	0.0233	0.3685	0.0011	0.5127	0.0023	0.4426	<1e-4	0.3764	<1e-4
Covtype	0.4762	0.0017	0.4742	0.0003	0.4846	0.0001	0.5318	0.0005	0.5312	<1e-4	0.474	<1e-4
Total mean	0.6129	-	0.5671	-	0.6161	-	0.7743	-	0.7908	-	0.6878	-

Table 19
Computation time results of dynamic clustering experiments

Datasets	SequentialKmeans		StreamKmeans		CluStream		HistStream				DenStream	
							Kmeans		HDBSCAN			
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Comet20	601.1789	0.8627	4173.0255	5.8701	111.1533	10.8181	20.0096	0.0576	136.5564	1.9086	103.7651	2.658
Meteorit20	598.7418	0.3835	4048.359	0.5028	99.2521	4.647	20.7868	0.1174	127.1455	1.1586	9854.3365	87.156
Circle20	612.8237	3.016	4321.1806	116.6523	100.8072	5.9999	29.4217	0.5364	494.6203	7.7403	141.2185	0.5147
Gaussian3	603.9073	0.2887	4199.7799	3.861	107.0685	9.2846	24.4203	0.4745	76.8281	0.503	3105.9656	6.6823
Square20	602.799	0.6457	4173.3107	3.6656	106.2824	11.4305	25.6883	2.3608	82.2653	0.9315	4769.8329	32.2027
Moon20	637.8218	3.3549	4435.9815	18.294	93.5763	5.8634	25.0204	3.4944	96.8381	1.1616	85.1582	2.9777
MixLarge10	343.2683	0.5161	2501.5632	5.0294	56.7119	4.6763	14.6774	0.6988	52.6882	1.3952	15544.6931	345.2638
MixLarge20	841.1053	1.8018	7051.6518	51.9163	137.4155	9.9892	33.4324	2.2048	179.6185	11.3344	2597.6667	59.2815
MixLarge50	2867.8369	7.7141	32481.0794	1056.8821	318.0409	8.9821	117.0566	10.4723	483.7691	17.7896	15544.6931	345.2638
Outdoor	1.9826	0.0119	12.0543	0.0459	18.3913	0.9477	1.3999	<1e-4	3.128	<1e-4	0.3405	0.0117
Gassenor	7.6274	0.2071	19.1491	0.7072	1.0658	0.0581	0.6815	0.2587	4.3237	0.0406	8.5452	0.099
IGBN	3.7845	0.2189	6.1328	0.1192	2.9484	2.3777	0.699	0.1686	9.8378	0.0485	2.9768	0.1999
IABN	8.4413	0.2436	14.6793	0.5363	4.2191	0.8793	1.6813	0.1438	14.7541	0.0692	2.6717	0.1316
Rialto	15.5729	0.0744	47.5999	0.2665	12.239	1.1187	2.8241	0.1433	11.8558	0.0565	8.0523	0.1978
IIAIN	70.0952	3.2963	125.4809	3.5934	30.6166	0.1338	10.3653	<1e-4	312.6661	<1e-4	82.7187	0.2978
IIRIN	68.3161	0.9216	123.9349	0.8913	30.1934	0.0635	24.901	0.784	378.9039	12.6035	103.2595	1.5032
Covtype	32.2408	0.4028	86.62	1.5781	43.3229	1.1053	12.2326	1.1983	142.838	3.6063	21.7024	0.1732
Total mean	465.7379	-	3989.5049	-	74.9003	-	21.4881	-	153.4492	-	3057.5057	-

The results shown in Tab.17, Tab.18, and Tab.19 provide compelling evidence of the superiority of HistStream(KMeans) over other algorithms in terms of clustering quality across multiple scenarios, consistently maintaining competitive performance. In particular, HistStream(KMeans) exhibits significantly faster running time compared to both SequentialKmeans and StreamKmeans, while still being slightly faster than CluStream. Similarly, HistStream(HDBSCAN) consistently achieves better clustering quality than DenStream and shows faster speed, although the difference occasionally appears small. Another interesting observation is that the stability of DenStream seems worse, especially when dealing with larger datasets, compared to the consistent performance of HistStream(HDBSCAN). To gain a more complete understanding of the overall performance differences between the algorithms, we performed calculations for the mean value of each index. The results, as shown in Fig.22, clearly

highlight the better performance of the proposed approach over other methods in terms of clustering quality, without any significant compromise in computational efficiency. Moreover, interestingly, the application of the proposed framework with HDBSCAN and KMeans reveals noticeable differences, although not statistically significant. These differences highlight the distinct advantages of each conventional clustering method, such as the efficiency of KMeans and the better clustering quality of HDBSCAN on arbitrary clusters. This observation shows the exceptional flexibility of the proposed framework in adapting to different tasks by judiciously selecting different conventional clustering algorithms.

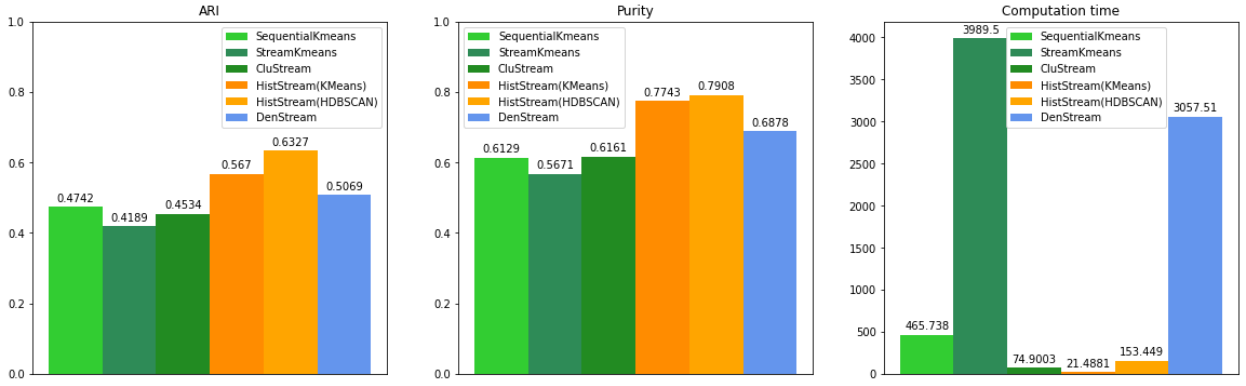

Figure 22: Total mean ARI & Purity & Computation time of different algorithms

5. Conclusion

A significant advancement in clustering, especially for large datasets, is the modelling of cluster distributions using innovative representations. In this research, we present a novel approach designed for both static and dynamic clustering, based on a histogram representation and the use of Wasserstein distance to compare distributions. The proposed framework can be used with any clustering algorithm, depending on the required cluster properties, by working on subsets of data and comparing the obtained clusters between subsets. The main goal is to reduce computation time and improve clustering quality. Specifically, experiments on real and artificial datasets show that the proposed framework can reduce the computational speed and memory requirements of most traditional clustering algorithms with minimal loss of quality. For dynamic datasets, the proposed framework is competitive with existing stream clustering approaches in terms of quality and complexity, while allowing great flexibility in the choice of clustering algorithm to be applied.

However, we observed that the clustering quality depends on the size of the data subsets (windows) and may start to degrade as the size decreases, with occasional sharp drops, while the computational time decreases significantly with smaller windows. This observation suggests that the proposed approach is sensitive to the number of samples in each window. Furthermore, the critical window size seems to vary between datasets. Therefore, a critical focus for our future studies will be to automatically determine the optimal number of samples for each window to obtain the best trade-off between quality and speed, and to propose more robust clustering results.

In conclusion, our research highlights a promising new technique that uses histogram modelling with Wasserstein distance for clustering in both static and dynamic scenarios. It provides an efficient option for real-world applications, offering significant advantages in terms of computational time and clustering quality.

References

[1] E. Alpaydin, Introduction to machine learning, MIT press, 2020.
[2] E. Diday, J. Simon, Clustering analysis, Digital pattern recognition (1976) 47–94.
[3] G. H. Ball, Classification analysis. (1970).
[4] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2015) 165–193.
[5] 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.
[6] G. J. Oyewole, G. A. Thopil, Data clustering: Application and trends, Artificial Intelligence Review 56 (7) (2023) 6439–6475.
[7] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, ACM sigmod record 25 (2) (1996) 103–114.
[8] F. Murtagh, P. Legendre, Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion?, Journal of classification 31 (2014) 274–295.
[9] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the royal statistical society. series c (applied statistics) 28 (1) (1979) 100–108.
[10] M. Charikar, S. Guha, É. Tardos, D. B. Shmoys, A constant-factor approximation algorithm for the k-median problem, Journal of Computer and System Sciences 65 (1) (2002) 129–149.
[11] N. Cristianini, J. Shawe-Taylor, J. Kandola, Spectral kernel methods for clustering, Advances in neural information processing systems 14 (2001).
[12] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, Optics: Ordering points to identify the clustering structure, ACM Sigmod record 28 (2) (1999) 49–60.
[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: kdd, Vol. 96, 1996, pp. 226–231.
[14] C. Malzer, M. Baum, A hybrid approach to hierarchical density-based cluster selection, in: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), IEEE, 2020, pp. 223–228.
[15] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, Vol. 4, Springer, 2006.
[16] W. Wang, J. Yang, R. Muntz, et al., Sting: A statistical information grid approach to spatial data mining, in: Vldb, Vol. 97, 1997, pp. 186–195.
[17] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, 1998, pp. 94–105.
[18] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, A survey on data stream clustering and classification, Knowledge and information systems 45 (2015) 535–569.
[19] S. Mansalis, E. Ntoutsi, N. Pelekis, Y. Theodoridis, An evaluation of data stream clustering algorithms, Statistical Analysis and Data Mining: The ASA Data Science Journal 11 (4) (2018) 167–187.
[20] U. Kokate, A. Deshpande, P. Mahalle, P. Patil, Data stream clustering techniques, applications, and models: comparative analysis and discussion, Big Data and Cognitive Computing 2 (4) (2018) 32.
[21] A. Zubaroğlu, V. Atalay, Data stream clustering: a review, Artificial Intelligence Review 54 (2) (2021) 1201–1236.
[22] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 359–366. doi:10.1109/SFCS.2000.892124.
[23] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, Streaming-data algorithms for high-quality clustering, in: Proceedings 18th International Conference on Data Engineering, IEEE, 2002, pp. 685–694.
[24] C. C. Aggarwal, S. Y. Philip, J. Han, J. Wang, A framework for clustering evolving data streams, in: Proceedings 2003 VLDB conference, Elsevier, 2003, pp. 81–92.
[25] F. Cao, M. Estert, W. Qian, 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.
[26] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, K. Zhang, Density-based clustering of data streams at multiple resolutions, ACM Transactions on Knowledge discovery from Data (TKDD) 3 (3) (2009) 1–28.
[27] J. Gama, P. P. Rodrigues, L. Lopes, Clustering distributed sensor data streams using local processing and reduced communication, Intelligent Data Analysis 15 (1) (2011) 3–28.
[28] X. H. Dang, V. C. Lee, W. K. Ng, K. L. Ong, Incremental and adaptive clustering stream data over sliding window, in: Database and Expert Systems Applications: 20th International Conference, DEXA 2009, Linz, Austria, August 31–September 4, 2009. Proceedings 20, Springer, 2009, pp. 660–674.
[29] T. Smith, D. Alahakoon, Growing self-organizing map for online continuous clustering, in: Foundations of Computational Intelligence Volume 4: Bio-Inspired Data Mining, Springer, 2009, pp. 49–83.
[30] A. Irpino, R. Verde, A new wasserstein based distance for the hierarchical clustering of histogram symbolic data, in: Data science and classification, Springer, 2006, pp. 185–192.
[31] B. W. Silverman, Density estimation for statistics and data analysis, Vol. 26, CRC press, 1986.
[32] J. A. Rice, Mathematical statistics and data analysis, Cengage Learning, 2006.
[33] G. H. Givens, J. A. Hoeting, Computational statistics, Vol. 703, John Wiley & Sons, 2012.
[34] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transactions on neural networks 16 (3) (2005) 645–678.
[35] A. Lyon, Why are normal distributions normal?, The British Journal for the Philosophy of Science (2014).
[36] P. J. Davis, Gamma function and related functions, Handbook of mathematical functions with formulas, graphs, and mathematical tables (1972) 253–293.
[37] S. J. Prince, Computer vision: models, learning, and inference, Cambridge University Press, 2012.

[38] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, L. E. Meester, A Modern Introduction to Probability and Statistics: Understanding why and how, Vol. 488, Springer, 2005.
[39] L. V. Kantorovich, Mathematical methods of organizing and planning production, Management science 6 (4) (1960) 366–422.
[40] K. Pearson, Contributions to the mathematical theory of evolution, Philosophical Transactions of the Royal Society of London. A 185 (1894) 71–110.
[41] L. Billard, E. Diday, From the statistics of data to the statistics of knowledge: symbolic data analysis, Journal of the American Statistical Association 98 (462) (2003) 470–487.
[42] S. Kullback, R. A. Leibler, On information and sufficiency, The annals of mathematical statistics 22 (1) (1951) 79–86.
[43] D. Endres, J. E. Schindelin, A new metric for probability distributions, IEEE Transactions on Information Theory 49 (2003) 1858–1860.
[44] Y. Rubner, C. Tomasi, L. J. Guibas, The earth mover's distance as a metric for image retrieval, International journal of computer vision 40 (2) (2000) 99–121.
[45] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Advances in neural information processing systems 26 (2013).
[46] V. W. Berger, Y. Zhou, Kolmogorov–smirnov test: Overview, Wiley statsref: Statistics reference online (2014).
[47] A. Ramdas, N. G. Trillos, M. Cuturi, On wasserstein two-sample testing and related families of nonparametric tests, Entropy 19 (2) (2017) 47.
[48] D. Revuz, M. Yor, Continuous martingales and Brownian motion, Vol. 293, Springer Science & Business Media, 2013.
[49] R. Durrett, Probability: theory and examples, Vol. 49, Cambridge university press, 2019.
[50] N. Bonneel, J. Rabin, G. Peyré, H. Pfister, Sliced and radon wasserstein barycenters of measures, Journal of Mathematical Imaging and Vision 51 (1) (2015) 22–45.
[51] V. M. A. Souza, D. M. Reis, A. G. Maletzke, G. E. A. P. A. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Mining and Knowledge Discovery 34 (2020) 1805–1858. doi:10.1007/s10618-020-00698-5.
[52] V. Losing, B. Hammer, H. Wersing, Interactive online learning for obstacle classification on a mobile robot, in: 2015 international joint conference on neural networks (ijcnn), IEEE, 2015, pp. 1–8.
[53] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sensors and Actuators B: Chemical 166 (2012) 320–329.
[54] V. Losing, B. Hammer, H. Wersing, Knn classifier with self adjusting memory for heterogeneous concept drift, in: 2016 IEEE 16th international conference on data mining (ICDM), IEEE, 2016, pp. 291–300.
[55] J. A. Blackard, D. J. Dean, Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables, Computers and electronics in agriculture 24 (3) (1999) 131–151.
[56] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al., River: machine learning for streaming data in python (2021).
[57] D. Sculley, Web-scale k-means clustering, in: Proceedings of the 19th international conference on World wide web, 2010, pp. 1177–1178.