BILINGUAL PUBLISHING GROUP

Journal of Electronic & Information Systems

https://journals.bilpubgroup.com/index.php/jeis

ARTICLE

Improving Fast Density Peak Clustering Using Mass Distance: m-FDPC

Mouloud Merbouche , ***Célia Tso*** , ***Jérémie Sublime*** *

ISEP – Paris Institue of Digital Technologies, 75006 Paris, France

ABSTRACT

In this work, we introduce m-FDPC, a mass-based variant of the Fast Density Peak Clustering (FDPC) algorithm, aimed at improving both performance and ease of use in unsupervised learning tasks. Traditional FDPC relies on Euclidean distance and requires careful parameter tuning and data normalization, which can significantly affect clustering outcomes—especially for heterogeneous or high-dimensional datasets. To address these challenges, m-FDPC replaces the conventional Euclidean metric with a mass-based distance measure derived from isolation forests, a method originally designed for anomaly detection. This substitution allows the algorithm to capture local data density and structure more naturally, while eliminating the need for normalization and simplifying the choice of key parameters such as cutoff distance and density thresholds. Comprehensive experiments on synthetic and real-world datasets demonstrate that m-FDPC not only matches or surpasses the performance of well-established clustering techniques such as DBSCAN, K-means, and Euclidean FDPC, but also offers greater robustness, scalability, and interpretability, particularly in high-dimensional or unevenly distributed data scenarios. Results evaluated through metrics like Matching Score and Silhouette Score confirm the algorithm's superior ability to detect meaningful cluster structures with minimal user intervention. Overall, m-FDPC provides a more efficient, adaptive, and user-friendly framework for density-based clustering, making it a promising tool for diverse applications in data mining, anomaly detection, and exploratory data analysis.

Keywords: Clustering; Isolation Forests; Density-Based Methods

*CORRESPONDING AUTHOR:

Jérémie Sublime, ISEP – Paris Institue of Digital Technologies, 75006 Paris, France; Email: jsublime@isep.fr

ARTICLE INFO

Received: 20 June 2025 | Revised: 13 August 2025 | Accepted: 23 August 2025 | Published Online: 30 August 2025
DOI: https://doi.org/10.30564/jeis.v7i2.11528

CITATION

Merbouche, M., Tso, C., Sublime, J., 2025. Improving Fast Density Peak Clustering Using Mass Distance: m-FDPC. Journal of Electronic & Information Systems. 7(2): 51–65. DOI: https://doi.org/10.30564/jeis.v7i2.11528

1. Introduction

The introduction should briefly place the study in a broad context and highlight why it is important, in particular, in relation to the current state of research in the field. Finally, it can conclude with a brief statement of the aim of the work and a comment about whether that aim was achieved[1]. Clustering is an important Machine Learning task that consists of finding interesting structures or groups of similar elements within a dataset in an unsupervised way. With clusters coming in all shapes and forms and no universal evaluation metrics due to the universal nature of this task, a large number of clustering algorithms exist in the literature (including the K-means algorithm[1] which is the most famous clustering method) with varying efficiency depending on the dataset nature, the number of features, and the clusters' nature (shape, size, density, variability and number).

Among the more successful families of clustering algorithms, we can mention density-based clustering methods that have proven effective at detecting clusters without assuming any specific shape or properties. In this family of methods, we can find very famous algorithms like DBSCAN[2], Density Peak Clustering (DPC)[3], or its evolutions, the Fast Search and Find of Density Peaks algorithm (FDPC)[4,5]. However, despite their efficiency and repeated success, these algorithms suffer from the following major weaknesses: First, they have a high complexity ranging in the $O(N^2)$ to $O(N^2 \log N)$, where N is the size of the dataset. Second, they usually rely on the computation of a distance matrix between all data elements whose complexity in $O(DN^2)$ (for D variables) not only makes the bulk of the algorithm computational complexity, but also tends to make these methods ineffective in high dimension since it often rely on the Euclidean distance, which is known not to scale well. Last but not least, these methods rely on several parameters that are not easy to set up properly and are key to their success or failure to detect good clusters.

Within this context, in this paper, we propose a mass-based version of the Fast Search and Find of Density Peaks algorithm (FDPC). By doing so, we reduce several of the previously mentioned problems as follows:

- The mass-based distance doesn't suffer from the same scaling issues as Euclidean-based distances. Furthermore, because it is based on isolation forests[6], this distance is also known to be very effective at separating clusters[7]. As we will show, these key factors greatly improve the efficiency of the m-FDPC method compared to the original DPC algorithm in high dimension.
- Because it is a normalized distance, it also significantly simplifies the parameter optimization of DPC-based methods, regardless of the dataset.
- Finally, thanks to recent works on efficiently computing mass-based distance matrices[8], it is possible to optimize the computation of mass-based distance matrices and make it 10 times faster.

While other methods have already considerably sped up the original DPC algorithm[9,10], most of them remain constrained by the computation of the distance matrix and stick with the Euclidean distance. This is where the originality of our paper lies, as it changes the nature and computation of this matrix, which is necessary before doing any approximation on the neighboring graph and is the heart of the DPC algorithm.

The remainder of this paper is organized as follows: First, we will present the state of the art on the DPC algorithm and mass-based distances for density-based clustering. Secondly, we will introduce our modified version of the FDPC algorithm using mass-based distance. Then, we will present our experimental results and comparisons with other clustering methods over several datasets and scenarios. Finally, this paper will end with a conclusion and some insights about our future works.

2. State of the Art

2.1. Density-Based Clustering and the Density Peaks Algorithm

TThe main principle of clustering is to split datasets into clusters by grouping together points belonging to the same high-density areas and that are separated from other clusters by lower-density areas. Based on this definition, which does not assume any shape for the clusters (unlike the K-means algorithm, which detects mostly spherical clusters), density-based clustering algorithms have become popular clustering methods as they seek clusters
directly based on the aforementioned definition. Famous legacy clustering methods belonging to this family include

famous algorithms such as Meanshift[11], DBSCAN[2], and OPTICS[12].

The Density peak clustering algorithm (DPC)[3] and its evolutions are parts of recent density-based methods which have shown strong capabilities to extract clusters even in complex scenarios with a lot of noise, or clusters in contacts, clusters with different densities, clusters with complex shapes, etc.

Like most other density-based methods, DPC relies on the use of a distance matrix between all data points, which by default is computed using the Euclidean distance over all *D* variables of the dataset:

$$d(x,y) = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2} \tag{1}$$

This distance matrix is essential for identifying the density of data points and determining the relationships between them, which will then be used to detect cluster centers. To do so, the following steps are followed:

1. For each data point x_i, the local density ρ_i is computed based on other points in its neighborhood.
2. From the previous step, the distance δ_i is computed which represents the distance of any point x_i to the closest higher density point. A graph is then built by linking each point to its nearest higher density neighbor[13].

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j>\rho_i} \left(d\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right)\right) & \text{if } \exists j \text{ such that } \rho_j > \rho_i \\ \max\limits_{j} \left(d\left(\overrightarrow{x}_i, \overrightarrow{x}_j\right)\right) & \text{for the densest point} \end{cases} \tag{2}$$

3. Finally, The clusters are extracted by turning into a cluster center each point whose closest higher density neighbor distance δ_i is too big based on a chosen threshold. The clusters are then propagated to all points linked to them in the graph.

Several methods exist to compute ρ_i. The first one from the original DPC algorithm consists in simply counting the numbers of points within a fixed distance parameter (or radius) d_c (See Equation (3)).

$$\begin{aligned} \rho_i &= \textstyle\sum_{j\neq i} \chi(d(x_i,x_j) - d_c) \\ \chi(d) &= \begin{cases} 1, & \text{if } d < 0 \\ 0, & \text{if } d \geq 0 \end{cases} \end{aligned} \tag{3}$$

Another possibility[14] that removes the radius parameter d_c is to computed ρ_i using a Gaussian or RBF-like kernel as shown in Equation (4).

$$\rho_i = \sum_{j\neq i} exp^{-d(x_i,x_j)^2} \tag{4}$$

While the second method removes the radius parameter and is, in theory, more "correct", it is also more computationally intensive as it forces the computation of *N-1* exponentials, most of which will have a value very close to 0.

After calculating the parameters ρ and δ for all points, it is possible to plot a decision graph (also called a Gamma Graph) with ρ on the x-axis and ρ on the y-axis. By placing each point on the graph, we can visualize the cluster centers (also called density peaks). The DPC algorithm selects cluster centers based on high local density and distance to denser points. The remaining points are assigned to the cluster of their nearest denser neighbor, and outliers are points whose local density falls below an absolute threshold *min_density*. When the noise option is enabled, such points are labeled *-1* (noise). Below is **Algorithm 1**, which presents the main steps of the Density Peak Clustering (DPC) method.

Algorithm 1. DPC Algorithm.

Require: The dataset $X = \{x_i\}_{i\in\mathbb{N}}$, the threshold distance parameter d_c
Ensure: The label vector of cluster index
1: Compute distance matrix.
2: Compute ρ_i and δ_i for each point.
3: Plot decision graph and select cluster centers according to ρ_i and δ_i.
4: Assign the remaining non-central points to clusters.
5: If $\rho_i <$ *min_density* then, if *noise* is enable, label x_i as *-1* (outlier).
6: Return label.

2.2. Isolation Forests and Mass-Based Distance for Clustering

Since our proposed method relies on the use of the mass-based distance (MBD), we will do a state-of-the-art review of where this distance comes from and how it is computed.

The Isolation Forest (iForest) is an anomaly detection algorithm designed to identify outliers in a dataset[6]. It is particularly well-suited for high-dimensional data and large datasets. The algorithm works as follows: The algorithm

builds an ensemble of S binary trees $T_{\mathrm{i,i}\in 1,\mathrm{S}}$. In each tree, for each node, the dataset (or a sample) is split randomly by selecting a feature and a random split value within the range of that feature. The process is iterated until each data point is isolated in a leaf.

Unlike traditional distance-based or density-based anomaly detection methods, the Isolation Forest relies on a unique concept: anomalies are easier to isolate than normal points due to their rarity and distinctiveness. By doing so, a distance based on an isolation forest is more effective than an Euclidean-based one as it is built based on the data themselves and uses outliers and anomalies to build the tree, which results in exploiting a lower-dimension manifold embedded around the data rather than all the data space, which in high dimension can be mostly empty.

In the rest of this work, we will refer to:

- A **leaf node** as a terminal node; it is a node of maximum depth.
- The **root node** as the first node created; it contains the entire sample.
- The **mass** of a node as the number of data points (observations) present in the node.

Using the following diagram (**Figure 1**), it is possible to see the structure of a complete binary tree:

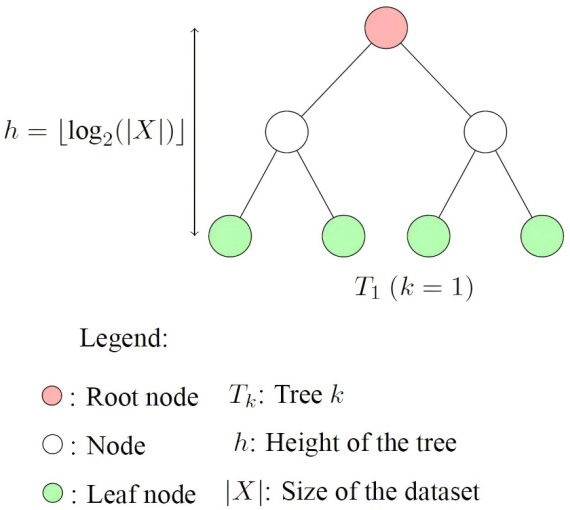

Figure 1. Complete Binary Tree.

We can observe in this diagram that there are three possible types of nodes. Moreover, the purpose of this algorithm is to distribute the data into different nodes, thus dividing the data. Each time a node is split, its mass becomes zero; the total mass (size of the initial sample) is therefore divided among all the leaf nodes.

From there, the main idea of the mass-based distance MBD is to use the iTree instead of the Euclidean distance: The distance between two points will be based on the average distance between their leafs in the different trees of the iForest, as shown in Equation (5), which we will detail in the next section presenting our algorithm.

This use of the mass-based distance to replace the Euclidean distance has already been successfully tested for DBSCAN[15] as well as the original DPC algorithm, but without optimized computation of the MBD matrix[16].

What we propose is an improvement over both works by using both the fast version of the DPC algorithm, and the fast version of the mass-based distance matrix computation in order to obtain an algorithm that is both a lot better than DBSCAN and OPTICS, but also much faster than earlier versions of DPC and mass-based methods.

3. Improving Fast Density Peaks Clustering

As stated in the introduction, the main limitation of all density-based algorithms (including DPC) is the computation and use of the distance matrix. Using Euclidean-based distances for this type of clustering presents two main challenges:

1. Euclidean-based distance matrices are slow to compute: $O(N^2D)$ for a dataset with N elements over D dimensions.
2. Due to the curse of dimensionality[17], Euclidean-based distances tend to become ineffective in high dimension due to strong differences over specific features being lost when averaged over too many features.

Within this context, mass-based distance applied to the DPC algorithm solves both issues by allowing to compute the distance matrix faster and providing a distance that, in high dimension, is more discriminating than the Euclidean distance. Indeed, the mass-based distance is built in a customized way that matches the data, and the distance is then computed based on the lowest common ancestor in the tree. It has recently been demonstrated that this type of distance relying on isolation kernels is better than Euclidean-based distances when it comes to high dimensional data[18] — the

proof relies on isolation kernels being based on Voronoi diagrams[19]. It follows that with a better and more discriminating distance, better clusters should be found.

3.1. Improving the Core Algorithm

In order to improve the Euclidean FDPC clustering, it is possible to change the metric to use a fast-computed mass-based distance matrix instead of an Euclidean distance matrix. This is the m-FDPC algorithm.

To perform this clustering, we first need the mass-based distance matrix. Here are the different steps of the algorithm to compute the mass-based distance matrix[8,20]:

1. The first step is to input a dataset X into the root node.
2. Then, the data included in the first node (root node) will be divided into two child nodes (left and right): The criterion for node divisibility is as follows: we randomly choose an attribute q, then among all the points present in the root node, we note:
   - q_{max} the maximum value observed among all observations (rows) for attribute q.
   - q_{min} the minimum value observed among all observations (rows) for attribute q.
3. A split point p (a value randomly chosen among all those of attribute q) is selected such that $q_{min} < p < q_{max}$. If the q^{th} attribute value for any $x \in X$(for the root node) is less than p, then x becomes part of the left child X_l, otherwise, it goes to the right child X_r. This step is performed recursively to divide the new nodes. The construction of the tree continues until the maximum height $h = \lfloor \log_2(|X|) \rfloor$ is reached or each point is isolated.

Now, after generating the forest using the Isolation Forest algorithm, it is possible, thanks to the different trees obtained, to calculate the MBD for each pair of points; we must therefore calculate $C_2^{|X|}$ distances in total. The idea is to calculate the MBD for each tree by computing the distance between two points as follows:

$$MBD_i(x,\ y) = Mass_i\left(LCA\left(Leaf_i(x), Leaf_i(y)\right)\right) \tag{5}$$

where:

- $MBD_i(x,\ y)$ is the function that returns the distance calculated for a single tree.
- $LCA\left(Node_1, Node_2\right)$ is the function that returns the node of the lowest common ancestor of the two nodes.
- $Leaf_i(x)$ is the function that returns the number of the leaf node.
- $Mass_i(Node)$ is the function that returns the mass of the node in tree T_i.

From this relation, it is possible to determine the MBD for the forest (it is simply the average of the MBD for all the trees):

$$MBD(x,\ y) = \frac{1}{S}\sum_{i=1}^{S} MBD_i(x,\ y) \tag{6}$$

Here is what an MBD matrix looks like (**Table 1**):

Table 1. MBD matrix.

	$\overrightarrow{x}_1$	$\overrightarrow{x}_2$	$\overrightarrow{x}_3$
$\overrightarrow{x}_1$	$MBD(\overrightarrow{x}_1,\ \overrightarrow{x}_1)$	$MBD(\overrightarrow{x}_1, \overrightarrow{x}_2)$	$MBD(\overrightarrow{x}_1,\ \overrightarrow{x}_3)$
$\overrightarrow{x}_2$	$MBD(\overrightarrow{x}_2,\ \overrightarrow{x}_1)$	$MBD(\overrightarrow{x}_2,\ \overrightarrow{x}_2)$	$MBD(\overrightarrow{x}_2,\ \overrightarrow{x}_3)$
$\overrightarrow{x}_3$	$MBD(\overrightarrow{x}_3,\ \overrightarrow{x}_1)$	$MBD(\overrightarrow{x}_3,\ \overrightarrow{x}_2)$	$MBD(x,\ \overrightarrow{x}_3)$

Below is the corresponding improved DPC algorithm that integrates the use of MBD distance (**Algorithm 2**):

Algorithm 2. m-FDPC Algorithm.

Require: The dataset $X = \{x_i\}_{i\in\mathbb{N}}$, the threshold distance parameter d_c
Ensure: The label vector of cluster indices
1: Fast compute the mass-based distance matrix.
2: Use the regular DPC algorithm with the computed matrix.
3: Identify outliers: if $\rho_i < min\_density$ then, if *noise* is enable, label x_i as *-1* (outlier).
4: Return label.

It is pertinent to elaborate on Step 3 of **Algorithm 2** (and similarly, Step 5 of **Algorithm 1** for DPC) concerning outlier identification. These steps refine the general outlier handling step by using a direct, parameter-driven mechanism for greater control and reproducibility. In our implementation, after computing the local density ρ_i for each point, a point is explicitly marked as noise (labeled *-1*) if its density falls below a user-defined threshold, *min_density*, and a

noise flag is active. This approach offers more direct control over the sensitivity of outlier detection compared to rules based on relative density. The effectiveness of this explicit, *min_density*-based approach is demonstrated in our Results Analysis section with the custom dataset designed for this purpose.

3.2. Algorithmic Complexity

In this section, we discuss the asymptotic time complexity of our m-FDPC algorithm and compare it with Euclidean FDPC. In particular, we focus on the complexity associated with computing the mass-based distance (MBD) matrix.

As previously mentioned, Euclidean FDPC relies on an all-pairs Euclidean distance matrix, which entails a complexity of $O(N^2D)$ for N data points and D dimensions. Although the clustering steps within Euclidean FDPC can be accelerated to $O(N \log N)$, the overall complexity remains dominated by the $O(N^2D)$ distance matrix computation. Similarly, standard density-based algorithms like DBSCAN or OPTICS also require $O(N^2)$ complexity due to their pairwise distance computations, each involving $O(D)$ operations per pair.

In contrast, our m-FDPC algorithm employs the mass-based distance (MBD) metric, which can be computed more efficiently using the fastMBD algorithm[15]. Without optimization, a naive MBD (nMBD) approach would require multiple insertions and Lowest Common Ancestor (LCA) queries, resulting in $O(N^2 \log S)$ complexity, where S is the sample size used to construct the iForest. The fastMBD technique significantly reduces this complexity by:

1. Constructing the iForest once.
2. Processing all data points in each iTree via a single top-down traversal.
3. Determining MBD values for all pairs of points during this traversal, without the need for separate leaf node detection or repeated LCA computations.

By performing the necessary computations during a single, systematic pass through each iTree, fastMBD calculates the complete $N \times N$ MBD matrix in $O(N^2)$ time. This approach removes the explicit dependence on D for the distance calculation stage and provides a substantial improvement over the naive MBD method.

In summary, the total complexity for m-FDPC is $O(N^2)$, achieved through the integration of the fastMBD approach. This complexity represents a significant practical advantage compared to the $O(N^2D)$ complexity of Euclidean-based methods, making m-FDPC more efficient and scalable, particularly in medium to high-dimensional datasets.

All results are summed up in **Table 2**.

Table 2. Complexity Comparison of FDPC with Euclidean and Mass-Based Distance Matrices.

Algorithm	DPC	Euclidean FDPC	m-FDPC
Distance Matrix Computation	$O(N^2D)$	$O(N^2D)$	$O(N^2)$[15]
Clustering Alg.	$O(N^2)$	$O(N \log N)$	$O(N \log N)$
- ρ calculation	$O(N^2)$	$O(N \log N)$	$O(N \log N)$
- δ calculation	$O(N)$	$O(N)$	$O(N)$
- Cluster assignment	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
- SearchCluster function	$O(N)$	$O(N)$	$O(N)$
Total Complexity	$O(N^2D)$	$O(N^2D)$	$O(N^2)$

3.3. Further Considerations on Speeding the Algorithm

While mass-based distance primarily affects the computation and complexity of the distance matrix, it can also be used to further accelerate the clustering algorithm itself. Owing to their similarity with KD-trees[21] (both rely on binary trees, recursive partitioning, and feature-based splits), which have historically been used for nearest-neighbor search, the binary trees from Isolation Forests could likewise be leveraged to approximate the search for the nearest highest-density point in the DPC algorithm.

This strategy is appealing in principle, as it could yield a clustering algorithm faster than $O(N^2)$, down to $O(N \log N)$[15,22]. However, it should be noted that in Isolation Forests, splits are random, designed to measure anomaly scores from path length. KD-trees, by contrast, use deterministic splits to enable efficient nearest-neighbor or range queries. Thus, using Isolation Forests to approximate the search for the nearest highest-density point would effectively layer one approximation (nearest highest-density

point) on top of another (distance estimation based on a non-deterministic method). In other words, this approach would trade precision for speed.

Another possibility to speed up the DPC algorithm would be to combine our mass-based distance approach with other approximation methods for the DPC algorithm that already exist in the literature and can reach a complexity as low as $O(N \log N)$ in best-case situations [9,10].

While both are possible depending on the context, in the remainder of this paper, we opted for the exact search of the nearest highest-density point using the original algorithm.

4. Experimental Results

4.1. Datasets

To assess the quality of our proposed method and compare it to other clustering strategies, we used the following datasets (**Table 3**):

Table 3. Datasets.

Dataset	Rows	Variables	Clusters	Noise
Iris	150	4	3	No
WDBC	569	30	2	No
Seeds	210	7	3	No
Digits	1797	64	10	No
Custom	1075	2	3–5	Yes
Flame	240	2	2	No
Aggregation	788	2	7	No

With regards to our Custom dataset, it contains 5 true labels from which we generated the data. However, due to one clustering being voluntarily small and another being sparse and with a high standard deviation, these could be considered noise depending on the algorithm's settings.

4.2. Experimental Setting and Data Preparation

We implemented and compared several clustering approaches: Euclidean FDPC utilizing an Euclidean distance matrix, m-FDPC (Mass-Based Fast Density-Peaks Clustering) leveraging a mass-based distance matrix, as well as DBSCAN and K-means algorithms to provide a comprehensive comparison between mass-based clustering and other methods relying on Euclidean distance.

Our methodology encompasses the following key steps: we used several standard datasets, including Breast Cancer Wisconsin, Seeds, Digits, Iris, and the shape (non-spherical) datasets Flame and Aggregation, as well as a custom-generated dataset. Each dataset was preprocessed appropriately to suit the requirements of the clustering algorithms. Specifically, for all Euclidean clustering methods, we normalized the data using the MinMaxScaler (a function from the scikit-learn library) to ensure that all feature values were scaled between 0 and 1. This normalization is crucial because the Euclidean distance matrix can exhibit large variance in values, potentially skewing the clustering results. However, this normalization process can lead to a loss of information by compressing the natural variation within the data, which is a notable drawback for Euclidean distance-based clustering and highlights a key advantage of the mass-based distance.

A distinctive aspect of our methodology is the differential treatment of data normalization based on the clustering approach employed:

- Euclidean FDPC (Euclidean Distance): For Euclidean FDPC, we applied normalization to the dataset using the MinMaxScaler. This step ensures that all feature values are within the $[1]$ range, stabilizing the Euclidean distance calculations by preventing features with larger scales from disproportionately influencing the distance metrics.
- m-FDPC (Mass-Based Distance): Conversely, for m-FDPC, we did not perform any additional normalization on the dataset. The mass-based distance matrix inherently produces values within the $[1]$ range, removing the need for prior normalization. This inherent normalization simplifies the preprocessing pipeline for m-FDPC and maintains the natural distribution of the data's mass-

based relationships.

One of the significant advantages of using m-FDPC lies in the ease of parameter selection. Given that the mass-based distance matrix is already normalized, determining optimal parameters such as *r-values*, *min-density* values, and *max-distance* values becomes more straightforward. These parameters are crucial for defining the density peaks and determining cluster centers. In contrast, for Euclidean FDPC, the variability in the Euclidean distance matrix necessitates a more careful and often dataset-specific parameter tuning process to achieve meaningful clustering results.

By leveraging the normalized nature of the mass-based distance matrix in m-FDPC, we observed that parameter ranges could be standardized across different datasets, enhancing the robustness and reproducibility of the clustering outcomes. This uniformity reduces the complexity involved in the parameter optimization phase, making m-FDPC a more efficient and user-friendly approach for density-based clustering.

After preparing the datasets, we applied both Euclidean FDPC and m-FDPC algorithms, as well as the DBSCAN and K-means algorithms, to identify clusters within each dataset. For evaluation, we employed metrics such as the Silhouette Score [23] and the Davies-Bouldin Index [24] to assess the quality of the clusters formed. Additionally, we utilized Principal Component Analysis (PCA) for dimensionality reduction to visualize the clustering results effectively.

4.3. Experimental Results

Our methodology highlights the streamlined preprocessing and parameter optimization processes afforded by m-FDPC, positioning it as a more accessible and efficient alternative to traditional Euclidean FDPC in density-based clustering tasks.

There are several important considerations to analyze the results shown in **Appendix A**, **Table A1**:

- We deliberately selected datasets where cluster formations naturally align with class distributions. This choice was essential for our evaluation strategy, as our clustering score (matching score) relies on the confusion matrix to assess performance.
- The matching score is a supervised metric and requires available labels. This was the case in our experimental setting for evaluation purposes, but it is not common for real clustering applications.
- Clustering indexes tend to be biased towards specific algorithms. For instance, the Davies-Bouldin index is the metric for the K-means algorithm and tends to be very heavily biased towards spherical clusters found by this method, regardless of their real quality [25,26].
- m-FDPC is a stochastic method that does not rely on multiple initializations (in contrast to scikit-learn's K-means++, typically run with several restarts). To assess reliability and robustness, we computed an additional metric.

The methodology adopted to quantify this stability for the m-FDPC stochastic algorithm is detailed as follows. For a fixed set of its hyperparameters, the entire clustering process was executed independently N times. Based on our experimental scripts, this value N was typically 50. Each of these N executions yielded a set of cluster labels, denoted as $P_1, P_2, \ldots, P_N$.

To assess the agreement between any two of these N partitions, for example, P_i and P_j, we employed the Adjusted Rand Index (ARI). The ARI is a measure of similarity between two data clusterings that considers all pairs of samples and counts pairs that are assigned to the same or different clusters in the predicted and true clusterings; it then adjusts this score to account for chance groupings. The ARI produces a value between -1 and +1, where +1 indicates perfect agreement between the partitions, values near 0 indicate a similarity no better than random assignment, and negative values suggest disagreement. We denote this as $ARI(P_i, P_j)$.

The overall stability score, S, for the tested configuration of hyperparameters, was then determined by calculating the arithmetic mean of the ARI scores obtained from all unique pairs of partitions generated across the N runs. The total number of such unique pairs is given by the combination $\binom{N}{2}$. The general formula for this stability score S is therefore:

$$S = \frac{1}{\binom{N}{2}} \sum_{1 \le i < j \le N} ARI(P_i, P_j) \tag{7}$$

This is the specific case where $N = 50$ (as indicated for stochastic algorithms in our comparative analysis script), the number of unique pairs $\binom{50}{2}$ is 1225. The formula applied

then becomes:

$$S = \frac{1}{1225} \sum_{i=1}^{49} \sum_{j=i+1}^{50} ARI(P_i, P_j) \tag{8}$$

A stability score S approaching 1 implies a high stability, indicating that the algorithm consistently generates very similar, if not identical, clustering results across repeated executions with the same parameters. Conversely, a lower score suggests greater variability in the partitions produced by the algorithm.

To optimize hyperparameters, we used the Optuna framework (a Bayesian parameter optimization method[27]) for most algorithms and datasets. Our optimization strategy with Optuna focused on maximizing the Silhouette coefficient rather than stability. This decision was made because calculating stability for mass-based distance algorithms already requires computing the mass-based distance matrix 50 times, which would have resulted in excessive computational iterations if incorporated directly into the optimization objective. Specifically, we ran Optuna for 200 trials to find parameters that maximize the Silhouette coefficient. Subsequently, for the best parameters identified, we calculated the stability metric. This two-step approach allowed us to identify parameter sets that provide both high clustering quality (as measured by the Silhouette coefficient) and good stability.

However, several exceptions required manual parameter tuning:

- For the Euclidean FDPC algorithm with the WDBC dataset, we manually fine-tuned optimal parameters after finding that Optuna failed to produce satisfactory results.
- Similarly, for the Euclidean FDPC algorithm with the Custom dataset, manual parameter tuning proved more effective than Optuna's suggestions.
- For the Euclidean FDPC algorithm with the Flame and Aggregation datasets, we also fine-tuned the parameters manually based on the silhouette index, matching index and visual quality of the results.
- For the Euclidean FDPC algorithm with the Digits dataset, as well as for DBSCAN across datasets, we encountered significant challenges in parameter identification, even through manual testing.
- For the m-FDPC algorithm, we relied on manually fine-tuned parameters that consistently yielded strong performance.

4.4. Result Analysis

The results show that m-FDPC achieves the best Matching Score on five datasets (Iris, WDBC, Digits, Custom, Aggregation) and is within 0.05 of the best on the remaining two (Seeds and Flame). It also attains the highest Silhouette on WDBC (and on Digits among the evaluated methods). **Table A1** summarizes the best results achieved using the optimal parameters for each dataset and each algorithm. This presentation allows for a synthetic comparison of the performance of different methods in terms of quality indices, thereby highlighting the best clustering achieved for each case. We note that for stochastic algorithms, the Silhouette and Davies–Bouldin values shown in the figures correspond to a single run and may therefore differ from those listed in **Table A1**, whereas for deterministic algorithms, the values coincide.

We observe that the *Matching Score*, computed directly from the confusion matrix (excluding noise, i.e., unassigned points), is given by:

$$\begin{aligned} MS &= \tfrac{1}{N_{assigned}} \sum_{i=1}^{K} \max_{j} n_{ij} \ , \\ N_{assigned} &= \sum_{i=1}^{K} \sum_{j} n_{ij} \ , \end{aligned} \tag{9}$$

where n_{ij} is the number of points in cluster i with true label j. This score is almost always highest for the m-FDPC algorithm. This indicates a more precise correspondence between the detected clusters and the actual data classes.

Regarding quality indexes, the results are shown in **Table A1**, where the best scores are underlined. We can see that unsurprisingly, the K-means algorithm achieves the best DBI scores. Our method, however, has the best matching score on average and achieves good performances on the Silhouette index. This proves that m-FDPC is competitive in matching cluster assignments to true class labels and gives good clustering results: while it does not always have the best clustering scores, it is never far behind the other methods.

Furthermore, while the DBI is generally higher (worse) than K-means, it is best on Flame and lower than K-means on Aggregation. For DBSCAN comparison, m-FDPC shows superior performance in Matching Score and Silhouette on WDBC and Seeds, while DBSCAN maintains a lower DBI on Iris and Custom; conversely, m-FDPC attains a lower DBI on Flame and Aggregation. Moreover, compared to

Euclidean FDPC, our m-FDPC often shows improved performances (especially on Matching Score), highlighting the effectiveness of using mass-based distance. The Flame dataset presents a case where performances are highly competitive; our m-FDPC method achieves the best DBI score (1.096) while remaining a strong competitor on the Matching Score (0.983 vs. 0.996) and Silhouette index.

Another advantage of m-FDPC lies in the simplicity of parameter selection. The use of a mass-based distance matrix eliminates the need for prior data normalization, which simplifies the parameterization process. In contrast, algorithms based on Euclidean distance, such as Euclidean FDPC and K-means, require prior data normalization, which can complicate the selection of optimal parameters. Because of this, it was easier to obtain satisfactory results with m-FDPC in terms of parameterization compared to Euclidean FDPC, K-means, or DBSCAN. For instance, for the *Digits* dataset, it was particularly difficult to find appropriate parameters for Euclidean FDPC and DBSCAN using a Euclidean distance matrix.

We can see in **Table A1** that for these two cases, we were unable to find any relevant results. On the other hand, m-FDPC was able to generate useful clusters without requiring complex adjustments, thanks to the use of the mass-based matrix.

As mentioned earlier, to facilitate the interpretation of results, we employed Principal Component Analysis (PCA) to reduce the dimensionality of the datasets to two dimensions: PC1 and PC2. **Figure 2** presents the clusters obtained for the *Wisconsin Breast Cancer Diagnostic* (WDBC) dataset using the Euclidean FDPC algorithm with an explained variance of 70.38%.

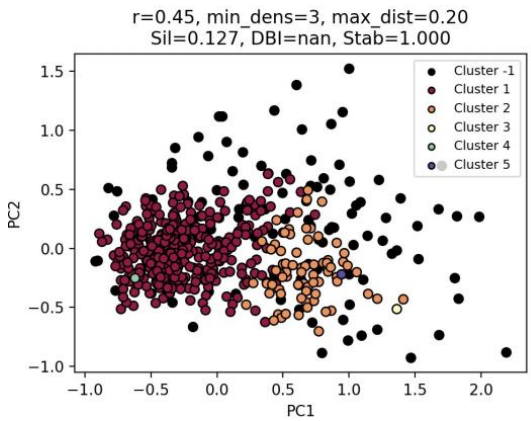

Figure 2. Euclidean FDPC clustering visualization for the WDBC dataset.

As illustrated in **Figure 3**, the clusters obtained with m-FDPC are significantly better defined than those obtained with Euclidean FDPC, using the best parameters.

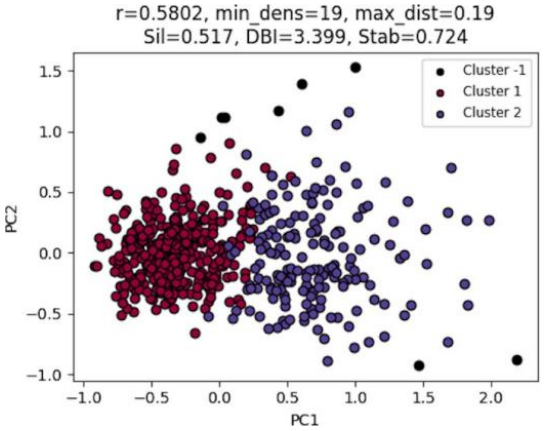

Figure 3. m-FDPC clustering visualization for the WDBC dataset.

While Euclidean FDPC manages to identify the two main clusters, the presence of substantial noise complicates the interpretation of the results. In contrast, m-FDPC successfully forms more compact and less noisy clusters, thereby enhancing the overall quality of the clustering.

Figure 4 presents the clusters obtained for the *Iris* dataset using the Euclidean FDPC algorithm after a PCA retaining 95.89% of the variance. Compared to **Figure 5**, which represents the clusters with m-FDPC, the clusters obtained with FDPC are less defined using the best parameters. For Euclidean FDPC, there is a considerable amount of noise, and we can only identify 2 clusters, whereas we need to have 3 different clusters. In contrast, for m-FDPC, the noise is negligible.

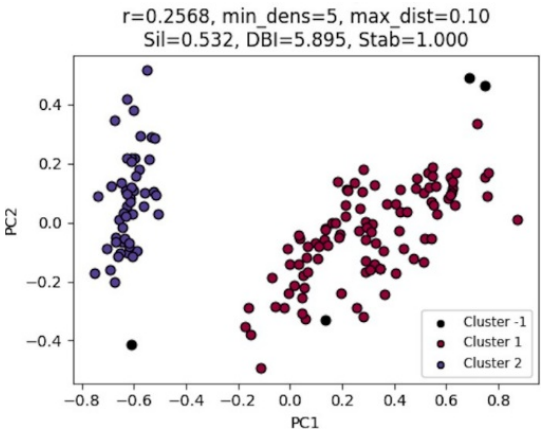

Figure 4. Euclidean FDPC clustering visualization for the Iris dataset.

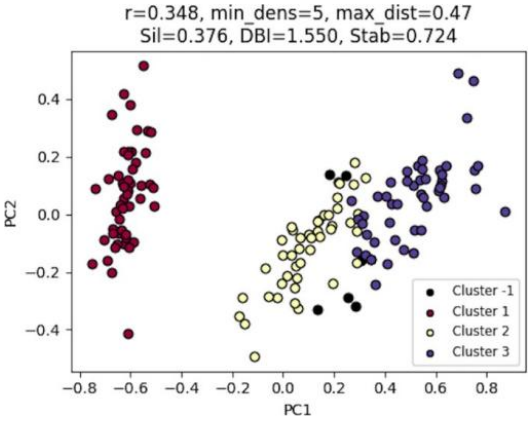

Figure 5. m-FDPC clustering visualization for the Iris dataset.

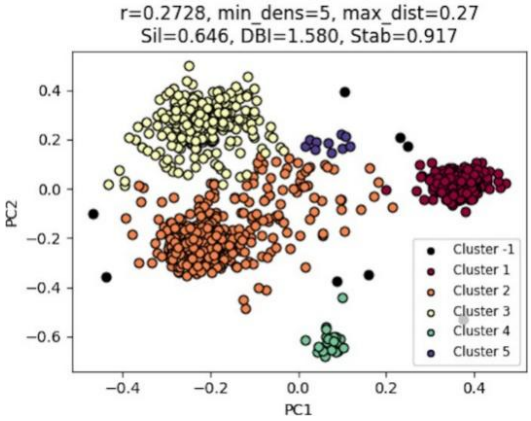

Figure 7. m-FDPC clustering visualization for the custom dataset.

For the best clusters obtained after optimizing the parameters, we observe that the quality indices show improvement. The Davies-Bouldin Index is 5.895 for Euclidean FDPC and 2.621 for m-FDPC, demonstrating a clear improvement.

The custom dataset (1075 points, 5 true labels, 2 dimensions) was specifically included not only to test performance on varying cluster shapes and densities but also to serve as a "toy dataset" for evaluating the outlier handling capabilities discussed in **Section 3**. This dataset includes a sparse group of points designed to emulate outliers or low-density formations.

Figures 6 and **7** show the clustering results for Euclidean FDPC and m-FDPC, respectively, on this custom dataset. For Euclidean FDPC (**Figure 6**), with parameters *r = 0.2*, *min_density = 1*, *max_dist = 0.2* (as per **Table A1**), the low *min_density* threshold resulted in all points being assigned to one of the three identified clusters, without explicit noise detection.

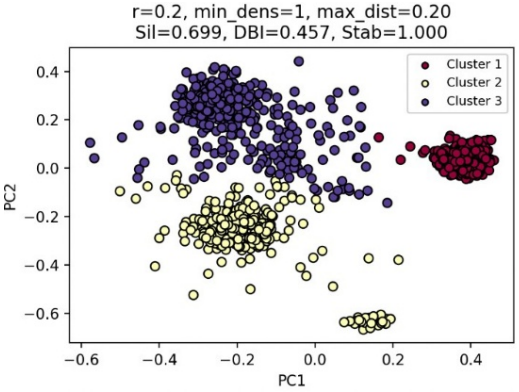

Figure 6. Euclidean FDPC clustering visualization for the custom dataset.

In contrast, m-FDPC (**Figure 7**), using parameters *r = 0.2563*, *min_density = 3* and *max_dist = 0.2743*, successfully identified five clusters and, significantly, a number of points classified as noise (visible as isolated points or explicitly labeled as 'Cluster -1' if using visualization parameters similar to those that produced. This ability to isolate outliers is a direct result of the *min_density* threshold in our FDPC implementation, effectively demonstrating the practical application of the refined outlier handling mechanism.

While Euclidean FDPC showed better DBI for this specific run (0.457 vs. 2.538), m-FDPC's higher Matching Score (0.823 vs. 0.791) and its capacity to delineate noise underscore its utility in scenarios with complex data structures and potential outliers. This provides empirical support for the benefits of the detailed outlier treatment approach.

Furthermore, the numerical and visual results on the Aggregation dataset show the limits of clustering indexes. For instance, as shown in **Figure 8**, it is the original Euclidean FDPC that has the best results in terms of Davies-Bouldin and Silhouette indexes. However, when looking at the visualization and the matching indexes, it is clear that our m-FDPC method (**Figure 9**) is better: not only does it have higher matching scores, but we can see that it manages to capture 2 small clusters that the original FDPC did not find. In contrast, the figures clearly show that the K-means algorithm (**Figure 10**) provides a completely arbitrary clustering.

We also evaluated the consistency of our m-FDPC by computing stability *S* over 50 independent runs (i.e., recomputing the mass-based distance matrix and reclustering each time).

This analysis was specifically focused on m-FDPC, as the other algorithms are either deterministic or artificially

stable. For instance, DBSCAN and Euclidean FDPC are fully deterministic, thus trivially yielding $S = 1.00$. While K-means is non-deterministic, the standard Scikit-learn implementation is actually a K-means++ algorithm run multiple times, which makes its output highly consistent.

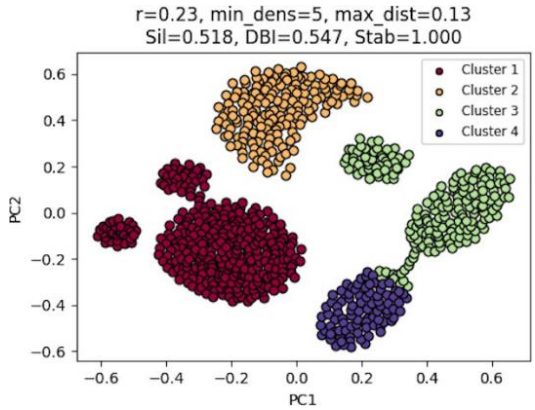

Figure 8. Euclidean FDPC clustering visualization for the aggregation dataset.

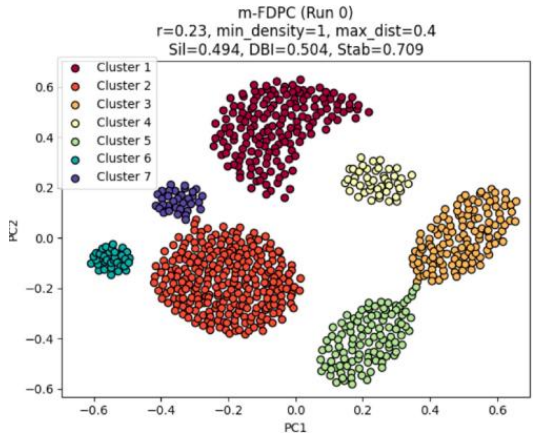

Figure 9. m-FDPC clustering visualization for the aggregation dataset.

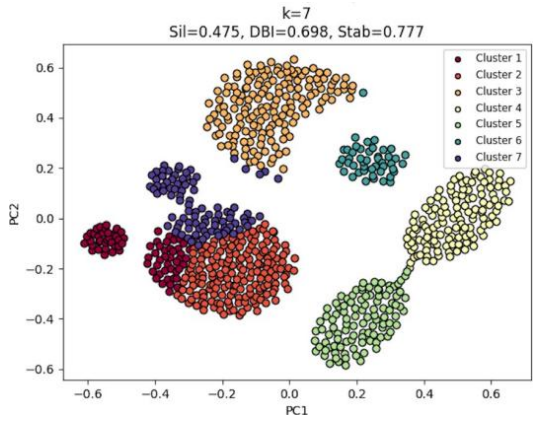

Figure 10. K-means clustering visualization for the aggregation dataset.

As reported in **Table A1**, m-FDPC achieves stability scores in the range $0.552 \leq S \leq 0.917$. For example, on the Iris and WDBC datasets, it attains $S = 0.724$, indicating that even though the mass-based distance is rebuilt on each run, the resulting partitions remain largely consistent. On the higher-dimensional Digits dataset, stability dips to $S = 0.652$, reflecting greater variability in the isolation forest splits under random sampling. These results show that the randomness from recomputing mass-based distances does not significantly affect m-FDPC's consistency, and that its stability remains competitive given the notable gains in cluster quality and the simplified parameter initialization.

The experiments demonstrate that m-FDPC offers not only better clustering quality but also greater ease of use in terms of parameter selection. By utilizing a mass-based distance, m-FDPC simplifies the parameterization process and eliminates the need for prior data normalization, making the algorithm more accessible and efficient. Furthermore, the results show that m-FDPC outperforms traditional Euclidean distance-based methods, particularly in high-dimensional and complex datasets where conventional methods face significant limitations. It also performs strongly on shape (non-spherical) datasets such as Flame and Aggregation. These advantages make m-FDPC an effective solution for clustering.

5. Conclusions and Future Works

In this paper, we presented a new variant of the Fast Density Peak Clustering (FDPC) algorithm by replacing the Euclidean distance with a mass-based distance (m-FDPC). More specifically, we integrated a fast computation of the mass-based distance matrix to enhance the efficiency of density peak detection. This approach allowed us to leverage the properties of the mass-based distance—already recognized for its robustness in high-dimensional settings and its intrinsic normalization—without the need for prior data normalization. Furthermore, we used the latest version of an optimized algorithm to compute the mass-based distance, which considerably reduced its computation time.

Our experiments demonstrated that m-FDPC provides improved clustering quality, exhibiting a stronger ability to separate data groups under various conditions. In our experiments (**Table A1**), m-FDPC achieves Matching Score and

Silhouette Score on a par with—and often exceeding—those obtained with classic Euclidean-metric FDPC, DBSCAN, and K-means++. Across the datasets where both methods were evaluated, m-FDPC attains a higher Matching Score than Euclidean-FDPC, DBSCAN, and K-means (on average +14%, +19%, +4%). Moreover, the parameter selection process, simplified by the mass-based distance, proved particularly advantageous: critical parameter values are more easily determined and more robust to data variations, which gives our method a decisive edge over other density-based clustering algorithms, where an extensive grid search can be needed to find the optimal parameters.

The initial objective, which was to assess the contribution of mass-based distance for accelerating and simplifying the implementation of FDPC, has thus been fully achieved. Not only does the new m-FDPC approach validate the performance and robustness gains, but it also confirms the relevance of reconsidering the distance metric used for density peak detection.

However, it is important to remember that while our method has been shown to be more efficient and faster than other clustering methods from different families, it still has limitations. Like many density-based clustering algorithms, it will struggle with clusters that cumulate the following difficulties: varying density clusters being in contact with one another.

In future works, it would be interesting to further assess the impact of isolation forest construction (the foundation of the mass-based distance) on the algorithm's performance and scalability. Additionally, incorporating mass-based approaches into other clustering algorithms, as well as exploring parallel optimizations is also something worth exploring in the future. Furthermore, combining mass-based distance computation with an already sped-up version of the DPC algorithm, such as the sparse dual approximation version [9], would also be a potentially interesting future lead.

Finally, extending this approach to Big Data environments or real-time data would open up new opportunities to test the robustness and flexibility of m-FDPC in various applied contexts.

Author Contributions

This work was conducted as part of an "introduction to scientific research" class, followed by M.M. and C.T. The research was done under the supervision of J.S., who came up with the subject. Most of the implementation, experiments, visualization as well as the paper writing were done by M.M. and C.T. J.S. was in charge of the supervision, formal analysis, validation, proofreading, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding

This work received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The source code can be found on Github: https://github.com/Mougli1/IX.2408.

Acknowledgments

We would like to thank Professor Lionel Trojman, who through his introduction to scientific research class made this project possible.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix A

Table A1. Summary of clustering performances with optimal parameters.

Dataset	Iris	WDBC	Seeds	Digits	Custom	Flame	Aggregation
Euclidean FDPC							
r	0.2568	0.45	0.3441	–	0.2	0.15	0.23
min_density	5	3	9	–	1	5	5
max_dist	0.1013	0.2	0.3921	–	0.2	0.25	0.13
DBI	5.895	1.875	1.788	–	0.457	1.128	0.547
Silhouette	0.532	0.127	0.367	–	0.699	0.335	0.518
Matching Score	0.660	0.705	0.833	–	0.791	0.996	0.841
Our m-FDPC							
r	0.348	0.5802	0.4238	0.45	0.2728	0.5335	0.23
min_density	5	19	13	3	5	5	1
max_dist	0.47	0.1906	0.151	0.3	0.27	0.55	0.4
DBI	2.621	3.921	1.635	2.106	2.538	1.096	0.697
Silhouette	0.449	0.453	0.344	0.181	0.640	0.34	0.418
Matching Score	0.933	0.928	0.843	0.795	0.823	0.983	0.996
Stability	*0.724*	*0.724*	*0.751*	*0.652*	*0.917*	*0.552*	*0.709*
DBSCAN							
ε	0.179	0.6201	0.1999	–	0.06	0.1	0.0892
min_samples	2	2	3	–	22	10	28
DBI	2.150	1.629	1.657	–	1.222	1.554	0.996
Silhouette	0.446	0.325	0.262	–	0.638	0.258	0.447
Matching Score	0.673	0.612	0.590	–	0.785	0.975	0.983
K-means							
k	3	2	3	10	5	2	7
DBI	0.787	1.136	0.877	1.927	0.587	1.103	0.749
Silhouette	0.483	0.385	0.422	0.131	0.680	0.380	0.462
Matching Score	0.887	0.914	0.890	0.782	0.813	0.846	0.903

Note: The best performances results have been underlined.

References

[1] MacQueen, J.B., 1967. Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1967; pp. 281–297.

[2] Ester, M., Kriegel, H.-P., Sander, J., et al., 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.

[3] Rodriguez, A., Laio, A., 2014. Clustering by Fast Search and Find of Density Peaks. Science. 344(6191), 1492–1496. DOI: https://doi.org/10.1126/science.1242072

[4] Wang, S., Wang, D., Li, C., et al., 2016. Clustering by Fast Search and Find of Density Peaks With Data Field. Chinese Journal of Electronics. 25(3), 397–402.

[5] Liu, R., Wang, H., Yu, X., 2018. Shared-Nearest-Neighbor-Based Clustering by Fast Search and Find of Density Peaks. Information Sciences. 450, 200–226. DOI: https://doi.org/10.1016/j.ins.2018.03.031

[6] Liu, F.T., Ting, K.M., Zhou, Z.-H., 2008. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.

[7] Ting, K.M., Zhu, Y., Carman, M., et al., 2016. Overcoming Key Weaknesses of Distance-Based Neighbourhood Methods Using a Data Dependent Dissimilarity Measure. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1205–1214.

[8] Ahlawat, N., Awekar, A., 2022. Scaling Up Mass-Based Clustering. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, 17–21 October 2022; pp. 3781–3785.

[9] Floros, D., Liu, T., Pitsianis, N., et al., 2018. Sparse Dual of the Density Peaks Algorithm for Cluster Analysis of High-Dimensional Data. In Proceedings of the 2018 IEEE High Performance Extreme Computing Conference, Waltham, MA, USA, 25–27 September 2018; pp. 1–14.

[10] Sieranoja, S., Fränti, P., 2019. Fast and General Density Peaks Clustering. Pattern Recognition Letters. 128, 551–558. DOI: https://doi.org/10.1016/j.patrec.2019.10.019

[11] Fukunaga, K., Hostetler, L., 1975. The Estimation of the Gradient of a Density Function, With Applications in Pattern Recognition. IEEE Transactions on Information Theory. 21(1), 32–40. DOI: https://doi.org/10.

1109/TIT.1975.1055330

[12] Ankerst, M., Breunig, M.M., Kriegel, H., et al., 1999. OPTICS: Ordering Points to Identify the Clustering Structure. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadelphia, PA, USA, 1–3 June 1999; pp. 49–60.

[13] Wang, Y., Qian, J., Hassan, M., et al., 2023. Density Peak Clustering Algorithms: A Review on the Decade 2014–2023. Expert Systems with Applications. 238(7), 121860. DOI: https://doi.org/10.1016/j.eswa.2023.121860

[14] Du, M., Ding, S., Jia, H., 2016. Study on Density Peaks Clustering Based on k-Nearest Neighbors and Principal Component Analysis. Knowledge-Based Systems. 99, 135–145.

[15] Ahlawat, N., 2024. Isolation Forest Based Efficient Unsupervised Machine Learning Algorithms [PhD thesis]. Indian Institute of Technology Guwahati: Guwahati, India. pp. 1–150.

[16] Ling, D., Xiao, X., 2018. Mass-Based Density Peaks Clustering Algorithm. In Proceedings of the International Conference on Intelligent Information Processing, Nanning, China, 19–22 October 2018; pp. 40–48.

[17] Chen, L., 2009. Curse of Dimensionality. In: Liu, L., Özsu, M.T. (Eds.). Encyclopedia of Database Systems. Springer: Boston, MA, USA. pp. 545–546.

[18] Wang, J., Ji, C., Liu, F., et al., 2025. A Band Selection Approach Based on a Mass-Based Metric and Shared Nearest-Neighbours for Hyperspectral Images. IET Image Processing. 19(1), e70165. DOI: https://doi.org/10.1049/ipr2.70165

[19] Ting, K.M., Washio, T., Zhu, Y., et al., 2021. Breaking the Curse of Dimensionality With Isolation Kernel. arXiv preprint. arXiv:2109.14198. DOI: https://doi.org/10.48550/arXiv.2109.14198

[20] Bhattacharjee, P., 2024. Density-Based Mining Algorithms for Dynamic Data: An Incremental Approach [PhD thesis]. Indian Institute of Technology Guwahati: Guwahati, India. pp. 1–200.

[21] Bentley, J.L., 1975. Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM. 18(9), 509–517. DOI: https://doi.org/10.1145/361002.361007

[22] Brown, R.A., 2015. Building a Balanced k-d Tree in O(kn log n) Time. Journal of Computer Graphics Techniques. 4(1), 50–68.

[23] Rousseeuw, P.J., 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Journal of Computational and Applied Mathematics. 20, 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7

[24] Davies, D.L., Bouldin, D.W., 1979. A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1(2), 224–227. DOI: https://doi.org/10.1109/TPAMI.1979.4766909

[25] Vendramin, L., Campello, R.J.G.B., Hruschka, E.R., 2010. Relative Clustering Validity Criteria: A Comparative Overview. Statistical Analysis and Data Mining: The ASA Data Science Journal. 3(4), 209–235. DOI: https://doi.org/10.1002/sam.10080

[26] Ikotun, A.M., Habyarimana, F., Ezugwu, A.E., 2025. Cluster Validity Indices for Automatic Clustering: A Comprehensive Review. Heliyon. 11(2), e41953. DOI: https://doi.org/10.1016/j.heliyon.2025.e41953

[27] Akiba, T., Sano, S., Yanase, T., et al., 2019. Optuna: A Next-Generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631.