Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 502 – 517

doi:10.3906/elk-2105-44

Clustering with density based initialization and Bhattacharyya based merging

Erdem KÖSE, Ali Köksal HOCAOĞLU*
Department of Electronics Engineering, Faculty of Engineering, Gebze Technical University, Gebze, Kocaeli, Turkey

Received: 06.05.2021 • **Accepted/Published Online:** 26.07.2021 • **Final Version:** 21.03.2022

Abstract: Centroid based clustering approaches, such as k-means, are relatively fast but inaccurate for arbitrary shape clusters. Fuzzy c-means with Mahalanobis distance can accurately identify clusters if data set can be modelled by a mixture of Gaussian distributions. However, they require number of clusters apriori and a bad initialization can cause poor results. Density based clustering methods, such as DBSCAN, overcome these disadvantages. However, they may perform poorly when the dataset is imbalanced. This paper proposes a clustering method, named clustering with density initialization and Bhattacharyya based merging based on the fuzzy clustering. The initialization is carried out by density estimation with adaptive bandwidth using k-Nearest Orthant-Neighbor algorithm to avoid the effects of imbalanced clusters. The local peaks of the point clouds constructed by the k-Nearest Orthant-Neighbor algorithm are used as initial cluster centers for the fuzzy clustering. We use Bhattacharyya measure and Jensen inequality to find overlapped Gaussians and merge them to form a single cluster. We carried out experiments on a variety of datasets and show that the proposed algorithm has remarkable advantages especially for imbalanced and arbitrarily shaped data sets.

Key words: Infinite mixture models, density estimation, Jensen inequality, bandwidth selection, optimal number of clusters, arbitrarily shaped clusters

1. Introduction

Clustering is an unsupervised problem of finding natural groups. It is one of the most difficult tasks for pattern recognition and machine learning. Difficulty arises from the nature of clustering. A data set may or may not possess a clustering structure and clustering algorithms may impose a clustering structure. Determining the presence or the absence of a clustering structure in a dataset is usually easy in low-dimensional settings such as the three-dimensional features. But, this time, the data set may be very complex to obtain sensible groups. When the dimensionality increases, it will be more difficult to verify whether the dataset possesses a clustering structure and to evaluate the results of clustering algorithms. These complexities have led to different clustering algorithms with some restrictions on the shape of clusters such as spiral, hyper-ellipsoidal, etc.

One of the basic and also a popular clustering algorithm is the k-means algorithm [1–3]. It is a centroid-based clustering algorithm. The k-means algorithm determines k centroids and associates points to the nearest centroid. Bezdek et. al. [4] introduced Fuzzy C-means (FCM) algorithm to improve the basic k-means algorithm by using fuzzy measures to associate points to clusters. Gueorguieva et al. [5] proposed a new algorithm called Fuzzy C-means with Mahalanobis Distance (MFCM) to improve the conventional FCM algorithm by accurately identifying cluster shapes using Mahalanobis distance. Centroid-based methods generally require number of clusters as input. To determine the number of clusters in a data set for centroid-based methods, randomized

*Correspondence: khocaoglu@gtu.edu.tr

algorithms [6, 7] were introduced. Centroid based methods mostly have initialization problems, but they have the advantage of easily assigning a new point to one of existing clusters due to their parametric nature.

Compared to centroid-based clustering, density-based clustering attempts to identify dense clusters of points, which leads to learning clusters of arbitrary shape. Density based clustering (DBSCAN) [8, 9], ordering points to identify the clustering structure (optıcs) [10], density based cluster estimation (DENCLUE) [11, 12], Gaussian density distance (GDD) [13] and others [14–17] are the examples for density based clustering algorithms. Density based methods use Kernel density estimation (KDE) or k-nearest neighbor (kNN), and they solve some problems like estimating optimal cluster count, but they cause new problems like kernel bandwidth. The density based methods do not require a user input for the number of clusters. But, they require one to specify the bandwidth parameter.

There are also several geometric approaches for clustering using Delaunay triangulation [18–20], Voronoi diagrams [21] or fuzzy c-means based ones [22]. Although geometrical approaches are computationally intensive for high dimensions, they can easily find hyperellipsoidal shaped and imbalanced clusters.

Other methods that do not require the number of clusters as a user input are subtractive [23, 24], Kluster [25] and evidential [26]. It is also worth noting that comprehensive surveys of clustering algorithms are done by [27, 28]. The reader may refer to them for other types of clustering methods beyond the scope of this study. Also, there are researches on clustering datasets [29–33]. We will use some of them and real world GPS trajectory [34] to validate our proposed method.

We focus on the density based algorithms as they are easy to initialize. However, they have the disadvantage that they cannot perform well when there are large differences in densities. To find better bandwidth for kernel density estimation, we are inspired from geometric approaches and propose k-nearest orthant-neighbor (k-NON) method. We form numerous subclusters and use Bhattacharyya measures to find overlapped ones and merge them to form a single cluster. The proposed algorithm is called clustering with density based initialization and Bhattacharyya based merging (CDIBM). It has an increased performance on Gaussian based data, and a decreased performance in thin-line clusters. However, this is not a big disadvantage because we mostly encounter Gaussian-shaped data as central limit theorem states that if independent random variables are added, normalized sum of them tends to form a normal distribution even if original random variables do not have a normal distribution, and any data can be modeled as a mixture of different signals in nature.

The rest of the paper is organized as follows. Section 2 consists the technical background of the proposed algorithm. In Section 3, we describe the proposed algorithm. In Section 4, we present the benchmark datasets and discuss the performance of the proposed algorithm. Section 5 concludes the paper.

2. Technical background

This section presents the building blocks of the proposed CDIBM algorithm, including the technical background information and the proposed k-NON method. In this section, we also discuss the reasons for utilizing these methods. The terminologies and notations presented in this chapter are used in the next section to define our clustering algorithm.

Let $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N-1}\}$ be a set, which contains univariate, independent and identically distributed N samples from an unknown d-dimensional distribution. i^{th} sample from $\mathbf{X}$ is $\mathbf{x}_i = \{x_{i,0}, x_{i,1}, ..., x_{i,d-1}\}$, where $0 \leq i < N$. We want to find optimal clusters in set $\mathbf{X}$ without prior knowledge about cluster count and

cluster shape. A summary of the clustering algorithms is given in the first section to find clusters for a given set $\mathbf{X}$. They have different strengths and also drawbacks, but our aim is to build a hybrid clustering algorithm using their strengths and avoiding their drawbacks.

Fuzzy C-Means with Mahalanobis distance (MFCM) can provide the parameters of multivariate Gaussian distribution, μ_j and $\mathbf{\Sigma}_j$, where μ_j is center and $\mathbf{\Sigma}_j$ is covariance matrix of the clusters to be identified. A major drawback for MFCM is that it requires prior knowledge about the number of clusters. We propose a hybrid approach based on both KDE and k-NN to initialize MFCM.

It is well known that the performance of KDE is mainly dependent on the bandwidth parameter of the kernel. Since a fixed bandwidth for all mixtures is usually not a valid assumption, adaptive KDE -[35, 36]- is utilized when number of samples varies greatly for each mixture. The hybrid approach that we propose does not use a fixed bandwidth parameter and constructs bandwidth matrices based on local properties. We present this algorithm in the next sub-section.

The proposed CDIBM algorithm utilizes MFCM to determine subclusters, and it is discussed in Section 2.2. Bhattacharyya distance [37] is used to obtain the amount of overlap among the sub-clusters with multivariate Gaussian distributions and Jensen inequality [38] is used to find sub-clusters to construct arbitrarily shaped main clusters. The mathematical background for this cluster merging approach is provided in Section 2.3 and Section 2.4.

2.1. Kernel density estimation

Nonparametric density estimation methods are used for clustering with different approaches, but imbalanced and hyperellipsoid shaped data become a problem with fixed bandwidth [39]. Zhu et. al. [15] showed that fixed bandwidth for each point is a problem for varying densities. This indicates that one needs to use adaptive bandwidth for each point [35, 36]. Adaptive bandwidth problem for KDE was studied by [21, 40, 41] using geometric approaches, but when high dimensional data occur, these methods fail.

Our algorithm use neither KDE nor k-NN to find densities directly. Our proposed algorithm CDIBM is a hybrid approach using the both. An orthant in d-dimensions is intersection of d mutually orthogonal half-spaces. By independent selections of half-space signs, there are 2^d orthants in d-dimensional space. Our approach picks k points from each 2^d orthants from d dimensional space for central point $\mathbf{x}_i$ and constructs a bandwidth matrix $\mathbf{H}_i$ from this neighborhood. This method is called k-NON.

Our aim is to find k nearest neighbors from each orthant to construct a point cloud $R_{\mathbf{x}_i}$ as shown in Figure 1 around the central point $\mathbf{x}_i$. The k-NON algorithm given in Algorithm (1) constructs these point clouds. We, then, calculate the bandwidth matrix $\mathbf{H}_i$ using $R_{\mathbf{x}_i}$ for each data point $\mathbf{x}_i$.

We calculate a distance vector $\mathbf{y}_i = [y_{i,n}]$, where $n \neq i$, from a reference point $\mathbf{x}_i$ to all other points in $\mathbf{X}$ as given in Equation (1). We also calculate orthant indices $\mathbf{z}_i = [z_{i,n}]$, which gives orthant index of a point $\mathbf{x}_n$ relative to the reference point $\mathbf{x}_i$ as given in Equation (2), where the function $u(.)$ is the unit step function. We partition $\mathbf{y}_i$ with respect to $\mathbf{z}_i$ and construct neighborhood set $R_{\mathbf{x}_i}$ in Algorithm 1. In an algorithmic point of view, $R_{\mathbf{x}_i}$ is a vector array composed of indeces of the k-nearest orthant neighbours.

$$y_{i,n} = \|\mathbf{x}_i - \mathbf{x}_n\| \tag{1}$$

$$z_{i,n} = \sum_{m=0}^{d-1} 2^m u(x_{i,m} - x_{n,m}) \tag{2}$$

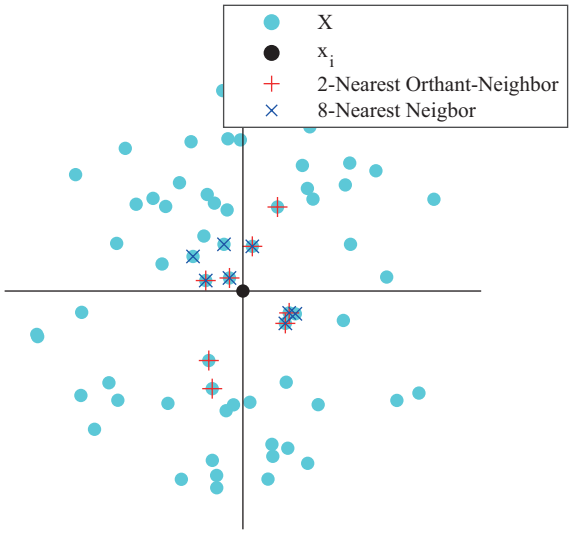


Figure 1. Neighbor example for a sample point $x_i \in X$ in 4-orthant space.

Algorithm 1 can be explained as following. We sort $\mathbf{y}_i$ and get sorted index vector $\tilde{\mathbf{n}}$. Starting from the nearest point to $\mathbf{x}_i$, we determine which $\mathbf{x}_{\tilde{\mathbf{n}}(n)}$ point belongs to m^{th} orthant with the help of $z_{i,\tilde{\mathbf{n}}(n)}$. We pick the first k nearest points to $\mathbf{x}_i$ for each orthant and save their indices to $R_{\mathbf{x}_i}$.

Algorithm 1: k-NON algorithm.

Input: $\mathbf{y}_i$, $\mathbf{z}_i$, k (nearest neighbor count in each orthant)
Output: $R_{\mathbf{x}_i}$: indices of k nearest neighbor in each orthant for each i

$\tilde{\mathbf{n}} \leftarrow sortindex\,(\mathbf{y}_i, ascend)$ // Sorted index vector
$l \leftarrow 0$
for $m \leftarrow 0$ *to* $2^d - 1$ **do**
  $temp \leftarrow 0$
  for $n \leftarrow 0$ *to* $N - 1$ **do**
    if $z_{i,\tilde{\mathbf{n}}(n)} = m$ **then**
      $temp(l) \leftarrow \tilde{\mathbf{n}}(n)$
      $l \leftarrow l + 1$
    end if
    if $l = k$ **then break**
  end for
  $R_{\mathbf{x}_i}(m) \leftarrow temp$
end for

Equation (3) gives bandwidth covariance matrix, where $|R_{\mathbf{x}_i}|$ denotes the number of elements of $R_{\mathbf{x}_i}$. After this, Gaussian KDE can be performed using Equation (4). A sample probability density function (PDF) result can be seen in Figure 2 for the dataset given in [32]. The function given in (4) is used for determining maximum density points locally among the dataset to obtain initial sub-cluster centers. Equation (5) gives

generalized set for (4).

$$\mathbf{H}_i = \frac{1}{|R_{\mathbf{x}_i}|} \sum_{\mathbf{x}_{\tilde{n}} \in R_{\mathbf{x}_i}} \left(\mathbf{x}_{\tilde{n}} - \mathbf{x}_i\right) \left(\mathbf{x}_{\tilde{n}} - \mathbf{x}_i\right)^T \tag{3}$$

$$f_{\mathbf{H}_i}\left(\mathbf{x}_i\right) = \frac{|R_{\mathbf{x}_i}|}{N\left(2\pi\right)^{0.5d} \det\left(\mathbf{H}_i\right)^{0.5}} \tag{4}$$

$$f_{\mathbf{H}} = \left\{ f_{\mathbf{H}_0}\left(\mathbf{x}_0\right), f_{\mathbf{H}_1}\left(\mathbf{x}_1\right), , ..., f_{\mathbf{H}_{N-1}}\left(\mathbf{x}_{N-1}\right) \right\} \tag{5}$$

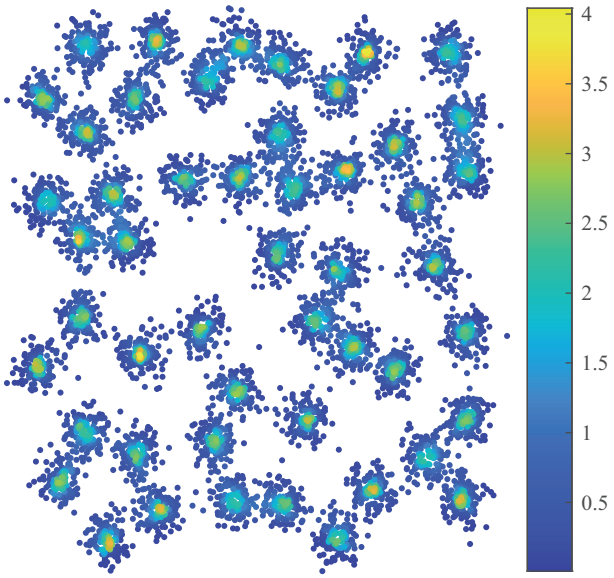


Figure 2. Probability density of an example independent-identically distributed Gaussian mixture.

2.2. Fuzzy c-means clustering with Mahalanobis distance

We use parameters from Section 2.1 to initialize Fuzzy C-Means algorithm to achieve accurate clusters. Our clustering metric for C-Means is Mahalanobis distance [5] in order to identify clusters modeled as Gaussian mixtures. The squared Mahalanobis distance $Q_{i,j}$ from $\mathbf{x}_i$ to the cluster j when the cluster has a distribution of $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$ is given in equation (6).

$$Q_{i,j} = \left(\mathbf{x}_i - \mu_j\right)^T \mathbf{\Sigma}_j^{-1} \left(\mathbf{x}_i - \mu_j\right) \tag{6}$$

Equation (6) is not useful for membership calculation because of different covariance matrix volumes. For membership calculations, the cluster volumes are kept constant $\rho > 0$ and the modified covariance matrix $\mathbf{\Sigma}_j^{mod}$ in equation (7) is used.

$$\mathbf{\Sigma}_j^{mod} = \frac{\mathbf{\Sigma}_j}{\left(\rho \det\left(\mathbf{\Sigma}_j\right)\right)^{1/d}} \tag{7}$$

Thus, $\det\left(\mathbf{\Sigma}_j^{mod}\right) = 1/\rho$ equation is obtained. The modified Mahalanobis distance $Q_{i,j}^{mod}$ given in equation (8) is used instead of equation (7).

$$Q_{i,j}^{mod} = \left(\mathbf{x}_i - \mu_j\right)^T \left(\mathbf{\Sigma}_j^{mod}\right)^{-1} \left(\mathbf{x}_i - \mu_j\right) \tag{8}$$

Membership of x_i to $\mathcal{N}\left(\mu_j, \mathbf{\Sigma}_j^{mod}\right)$ is calculated using equation (9), where $m > 1$ is fuzzification parameter. If fuzziness parameter m in equation (9) decreases to 1, clustering will be crispier.

$$r_{i,j} = \left(\sum_{l=0}^{M-1} \left(\frac{Q_{i,j}^{mod}}{Q_{i,l}^{mod}} \right)^{\frac{1}{m-1}} \right)^{-1} \tag{9}$$

Cluster center μ_j and cluster covariance matrix $\mathbf{\Sigma}_j$ are calculated using equation (10) and (11), respectively.

$$\mu_j = \frac{\sum_{i=0}^{N-1} r_{i,j}^m \mathbf{x}_i}{\sum_{i=0}^{N-1} r_{i,j}^m} \tag{10}$$

$$\mathbf{\Sigma}_j = \frac{\sum_{i=0}^{N-1} r_{i,j}^m \left(\mathbf{x}_i - \mu_j\right)\left(\mathbf{x}_i - \mu_j\right)^T}{\sum_{i=0}^{N-1} r_{i,j}^m} \tag{11}$$

2.3. Bhattacharyya distance

Bhattacharyya measure [37] is a well known method for determining divergence or overlap between two different multivariate Gaussian distributions. We use it to identify clusters to be merged. The Bhattacharyya distance between the distributions f_j and $f_{\tilde{j}}$ are given in equation (12) [37].

$$D_{j,\tilde{j}} = -\ln\left(BC\left(f_j, f_{\tilde{j}}\right)\right) = -\ln\left(\int_{-\infty}^{\infty} \sqrt{f_j\left(x\right) f_{\tilde{j}}\left(x\right)} dx\right) \tag{12}$$

For multivariate normal distributions, to measure overlap between cluster i and cluster j with distributions $\mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ and $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$, equation (12) results in equation(13).

$$D_{j,\tilde{j}} = \frac{D_{B_{Mah}}}{8} + \frac{D_{B_{det}}}{2} \tag{13}$$

where $D_{B_{Mah}}$ and $D_{B_{det}}$ are (14) and (15), respectively.

$$D_{B_{Mah}} = \left(\mu_j - \mu_{\tilde{j}}\right)^T \left(\frac{\mathbf{\Sigma}_j + \mathbf{\Sigma}_{\tilde{j}}}{2}\right)^{-1} \left(\mu_j - \mu_{\tilde{j}}\right) \tag{14}$$

$$D_{B_{det}} = \ln\left(\frac{\det\left(\frac{\mathbf{\Sigma}_j + \mathbf{\Sigma}_{\tilde{j}}}{2}\right)}{\sqrt{\det\left(\mathbf{\Sigma}_j\right) \det\left(\mathbf{\Sigma}_{\tilde{j}}\right)}} \right) \tag{15}$$

2.4. Jensen inequality for chi-squared distributions

If we assume we have point cloud from a distribution $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$, but we want the points from most probable quantiles, we must use inequalities like Chebyshevs. In multidimensional Gaussian case, distance from points $\mathbf{X}$ to the distribution $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$, forms Mahalanobis distance random variable $Q_{i,j}$.

$$Q_{i,j} = (\mathbf{x}_i - \mu_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{x}_i - \mu_j) \tag{16}$$

Sum of d standard normal random variables has chi-square distribution with d degrees of freedom [42]. So, random variable $Q_{i,j}$ will be chi-squared distributed from definition. Jensen's inequality for chi-squared distributions respecting to Lipschitz functions of Gaussian variables [38] is given in equation (17),

$$Pr\left[Q_{i,j} > d\,(1+t)\right] \leq \exp\left(\frac{-dt^2}{2}\right) \tag{17}$$

We can modify equation (17) as equation (18)

$$Pr\left[Q_{i,j} < d\,(1+t)\right] \geq 1 - \exp\left(\frac{-dt^2}{2}\right) \tag{18}$$

where $\alpha = 1 - \exp\left(\frac{-dt^2}{2}\right)$ is quantile portion we desired, where $0 < \alpha < 1$. We can extract t via equation (19).

$$t = \sqrt{-2\ln(1-\alpha)/d} \tag{19}$$

After finding t, we can find the threshold Mahalanobis distance $d(1+t)$, which shows α portion of region lies in $Q_{i,j}$ random variable, where $0 < \alpha < 1$.

3. Algorithm

This section introduces our method in algorithmic perspective using the methods previously explained. The flowchart of the algorithm is shown in Figure 3. The algorithm starts with estimating bandwidth for each point in a dataset using the k-NON. It uses these bandwidths to estimate KDE. Then, it finds subclusters using the maximum density point in $R_{\mathbf{x}_i}$ groups. These subclusters are used to initialize fuzzy c-means with Mahalanobis distance to find more accurate Gaussian subclusters $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$. Finally, we use Bhattacharyya measure [37] and Jensen inequality to find overlapped Gaussians and merge them to form a single cluster.

3.1. Initialization with adaptive KDE

We find density estimate $f_{\mathbf{H}}$ using Eq. (4) for each $\mathbf{x}_i$. Initially, a label *unprocessed* (value 0) is assigned for each $\mathbf{x}_i$. Then, we find $center_{pdf}$ and $neigh_{pdf}$ for each *unprocessed* $\mathbf{x}_i$ and corresponding neighborhood $R_{\mathbf{x}_i}$. If $center_{pdf}$ is smaller than all $neigh_{pdf}$ for $\mathbf{x}_i$, then $\mathbf{x}_i$ cannot be a candidate as a cluster center, and, therefore, it is labeled as *not a candidate* (value -1). If $center_{pdf}$ is bigger than all $neigh_{pdf}$ for $\mathbf{x}_i$, then $\mathbf{x}_i$ is as labeled as *subcluster center* (value 1), and the neighboring points are labeled as *not a candidate* in order not to evaluate them as a candidate as a cluster center.

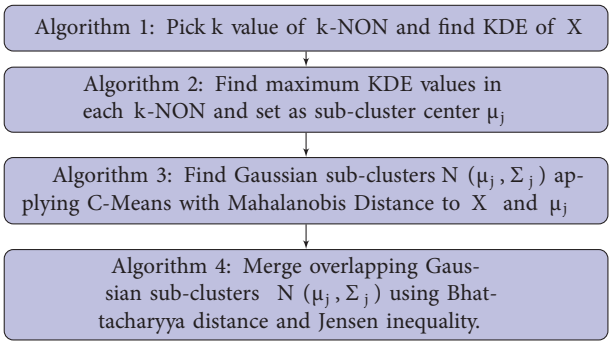


Figure 3. Algorithm flowchart of the proposed algorithm.

The initialization algorithm, Algorithm 2, returns the number of points labeled as *subcluster center* as the number of subclusters and returns these points as the center of subclusters. At this stage, covariance matrices for each subcluster is set to identity-matrix. These results are used to initialize MFCM.

Algorithm 2: Initialization.

Input: $\mathbf{X}$, k: nearest neighbor count in each orthant
Output: μ, $\mathbf{\Sigma}$, M: initial estimate for the number of sub-clusters

$[f_{\mathbf{H}}, R] \leftarrow kernelPDF(\mathbf{X}, k)$ // Equation (4)
$cnd \leftarrow 0_{1,N}$ // All $\mathbf{X}$ elements are unprocessed
$j \leftarrow 0$
for $i \leftarrow 0$ *to* $N-1$ **do**
  if $cnd\,(i) = 0$ **then**
    $center_{pdf} \leftarrow f_{\mathbf{H}}\,(i)$ // Single PDF value
    $neigh_{pdf} \leftarrow f_{\mathbf{H}}\,(R_{\mathbf{x}_i})$ // PDF value vector
    if $centr_{pdf}$ *bigger than each of* $neigh_{pdf}$ **then**
      $cnd\,(i) \leftarrow 1$ // Candidate is a center
      $cnd\,(R_{\mathbf{x}_i}) \leftarrow -1$ // Candidate cannot be a center
      $\mu_j \leftarrow \mathbf{x}_i$
      $\mathbf{\Sigma}_j \leftarrow I_d$ // I_d is $d \times d$ identity matrix
      $j \leftarrow j + 1$
    else if $centr_{pdf}$ *smaller than one of* $neigh_{pdf}$ **then**
      $cnd\,(i) \leftarrow -1$ // Candidate cannot be a center
    end if
  end if
end for
$M \leftarrow j$ // Number of subclusters

3.2. Gaussian Sub-clusters with Fuzzy C-Means

We estimate $\mathcal{N}(\mu_j, \mathbf{\Sigma}_j)$ for each sub-clusters using MFCM in Section 2.2 with initial values from Section 3.1.

3.3. Bhattacharyya linkage

We find Bhattacharyya distance for each sub-cluster pair j and $\tilde{j}$ to determine if they are overlapped. If the distance is larger than a threshold Q_{limit}, then the two sub-clusters are assumed to be not overlapped.

Otherwise, the distance below the threshold is seen as an evidence of an overlap and the subclusters j and $\tilde{j}$ are labeled as *overlapped* by setting the related elements of the evidence matrix to *true*: $\mathbf{E}_{j,\tilde{j}} = true$, and $\mathbf{E}_{\tilde{j},j} = true$ (see Algorithm 3 for an algorithmic view). This matrix is later used to perform merging not only the subclusters j and $\tilde{j}$, but also all clusters overlapped by any of these clusters directly or indirectly. For example, if $\mathbf{E}_{i,j} = true$, $\mathbf{E}_{i,k} = true$ and $\mathbf{E}_{k,m} = true$, then the subclusters i, j, k and m are merged resulting in a single cluster and if this is the l^{th} linked sub-cluster, then the l^{th} linked subcluster index vector $\mathbf{LSC}_l$ is set to $\{i, j, k, m\}$ to indicate that these subclusters are merged.

Algorithm 3: Bhattacharyya linkage.

Input: D from (13), α from (19)
Output: **LSC** (Array of linked sub-cluster index vectors)

$Q_{limit} \leftarrow d\left(1 + \sqrt{\left(-2\log(1-\alpha)\right)/d}\right)$
$\mathbf{E} \leftarrow I_{M,M}$ // Threshold matrix, initially identitiy
for $j \leftarrow 0$ *to* $M-2$ **do**
  for $\tilde{j} \leftarrow j+1$ *to* $M-1$ **do**
    $\mathbf{E}_{j,\tilde{j}} \leftarrow u\left(Q_{limit} - D_{j,\tilde{j}}\right)$ // u: Unit-step function
    $\mathbf{E}_{\tilde{j},j} \leftarrow \mathbf{E}_{j,\tilde{j}}$
  end for
end for
$[row_{ind}, col_{ind}] \leftarrow find(\mathbf{E} = 1)$ // Corresponding indices of 1s
LSC $\leftarrow unique([row_{ind}, col_{ind}])$ // Unique couples remains
$N_{\mathbf{LSC}} \leftarrow |\mathbf{LSC}|$ // Number of vectors in **LSC**
$Concatenate_{en} \leftarrow$**true**; $Break_{en} \leftarrow$**false**
while $Concatenate_{en} =$***true*** **do**
  for $l \leftarrow (N_{\mathbf{LSC}} - 1)$ *to* 0, *step:*-1 **do**
    for $\tilde{l} \leftarrow (l-1)$ *to* 0, *step:*-1 **do**
      $N_{l,\tilde{l}} \leftarrow |\mathbf{LSC}_l| + |\mathbf{LSC}_{\tilde{l}}|$ // Sum of element counts
      $Union_{tmp} \leftarrow \mathbf{LSC}_l \cup \mathbf{LSC}_{\tilde{l}}$
      if $|Union_{tmp}| \neq (N_l + N_{\tilde{l}})$ **then**
        $\mathbf{LSC}_l \leftarrow Union_{tmp}$
        Remove $\tilde{l}^{th}$ vector from **LSC**
        $Break_{en} \leftarrow$**true**
        break
      end if
    end for
    if $Break_{en} =$***true*** **then**
      $Break_{en} \leftarrow$**false**
      break
  end for
  if $N_{\mathbf{LSC}} = |\mathbf{LSC}|$ **then** $Concatenate_{en} \leftarrow$**false**
end while

3.4. Final clustering

Each point is assigned to the nearest subcluster using Mahalanobis distance. Then, we obtain the final clusters by merging the sub-clusters indicated by the linked subcluster index vector $\mathbf{LSC}_l$, where $l = 1, 2, ..., |\mathbf{LSC}|$. Algorithm 4 gives an algorithmic view for this step.

Algorithm 4: Final clustering.

Input: X, μ, $\mathbf{\Sigma}$, ρ, **LSC**
Output: $Cluster_{new}$
for $j \leftarrow 0$ *to* $M-1$ **do**
  $\mathbf{\Sigma}_j^{mod} = \mathbf{\Sigma}_j / \left((\rho \det(\mathbf{\Sigma}_j))^{1/d} \right)$
  for $i \leftarrow 0$ *to* $N-1$ **do** $Q_{i,j}^{mod} = (\mathbf{x}_i - \mu_j)^T \left(\mathbf{\Sigma}_j^{mod}\right)^{-1} (\mathbf{x}_i - \mu_j)$
end for
for $i \leftarrow 0$ *to* $N-1$ **do** $Cluster_{out}(i) \leftarrow \underset{j}{\operatorname{argmin}} \left(Q_{i,j}^{mod}\right)$
if $\mathbf{LSC}_l \neq \emptyset$ **then**
  $N_{union} \leftarrow |\mathbf{LSC}_l|$ // Number of elements in $\mathbf{LSC}_l$
  for $l \leftarrow 0$ *to* $N_{union} - 1$ **do**
    for $i \leftarrow 0$ *to* $N-1$ **do**
      if $Cluster_{out}(i) \in \mathbf{LSC}_l$ **then** $Cluster_{new}(i) \leftarrow l$
    end for
  end for
end if

4. Performance

We used five different algorithms and nine different synthetic datasets to validate the proposed algorithm. Figure 4 provides a quick overview of the results. Datasets used to validate the proposed algorithm are, in the order in which they are given in Figure 4, a) Two circles [31], b) Two half-circles [29], c) Three spiral [31], d) Gaussian mixture [32, 43], e) Imbalanced, [32, 43], f) Overlapped Gaussian mixture, g) Hepta [33], h) Atom [33] and i) Chainlink [33] We used adjusted rand index (ARI) and normalized mutual information (NMI) [44] for performance evaluation and comparison with other algorithms in Table 1 and 2, respectively because ARI has no assumption on the cluster structure, and NMI can easily measure random label assignments [45].

The proposed algorithm has two parameters to affect its performance: k, the number of neighbors for the k-NON algorithm, and α, the Jensen inequality parameter. Figure 5 shows the effects of the parameters k and α for all of the test dataset. Both NMI and ARI are between 0.7 and 0.95 for these parameter ranges. Based on this analysis, we use $k = 6$ and $\alpha = 0.3$ in the rest of the experiments.

The algorithms used for the comparison are spectral [46, 47], Average-linkage agglomerative [48], DBSCAN [8], OPTICS [10] and GDD [13]. Compared algorithms are used via Scikit-learn [45]. Parameters of algorithms used in the experiments are given in Table 3, 4 and 5 to optimize their performance for each individual dataset.

Table 1, 2 and Figure 4 show that the proposed algorithm fails only for dataset (c) because the Gaussianity assumption for the subclusters is not met for this dataset. This is a known and only drawback for the proposed algorithm in terms of cluster shape. The proposed algorithm is successful on the rest of the datasets with an average ARI of 0.952 and average NMI of 0.954. If the datasets given the worst performances are ignored for

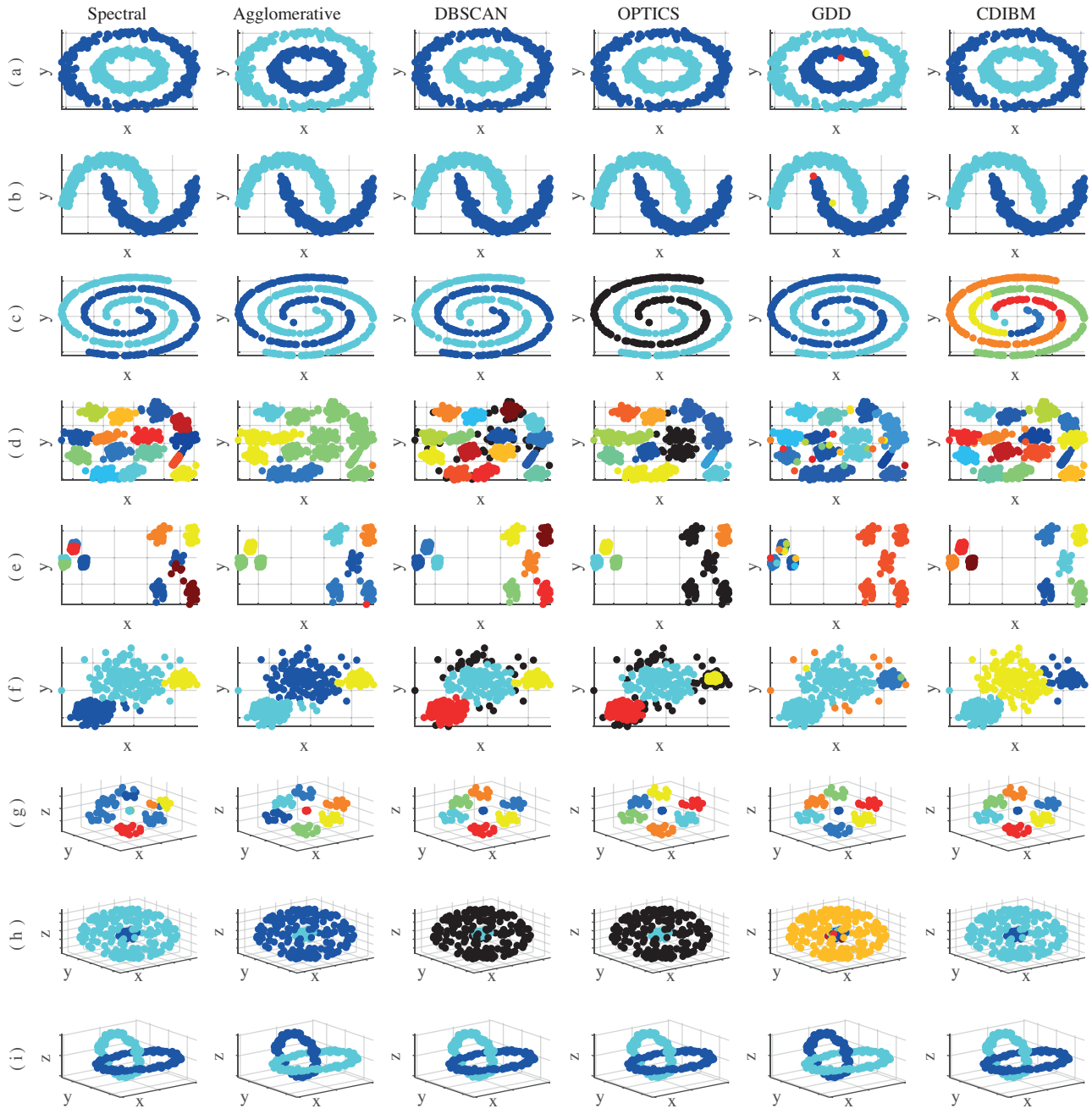
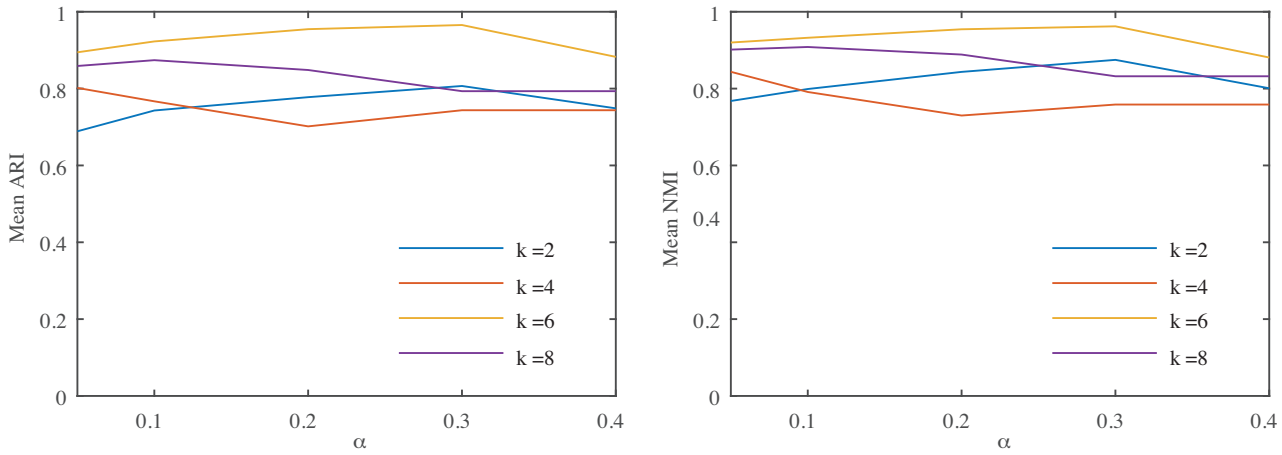


Figure 4. Obtained clusters from synthetic datasets for different algorithms. Colors represent different clusters, black means noise for DBSCAN and OPTICS.

each algorithm, the best performance for both minimum and average ARI and NMI metrics is provided by the proposed algorithm. The averages for ARI and NMI become 0.988 and 0.985, respectively for the proposed algorithm, while the figures are 0.986 and 0.981 for DBSCAN, which gives the most competitive performance. When evaluating these results, it must be taken into account that a fixed parameter set is used for the proposed algorithm, while the parameters are optimized individually for the rest of the algorithms for each dataset. This



(a) Mean ARI of dataset.

(b) Mean NMI of dataset.

Figure 5. Sweep of α and k parameters and corresponding ARI and NMI results for CDIBM.

Table 1. Adjusted Rand Index results for reference algorithms and datasets.

Dataset	Spectral	Agglom.	DBSCAN	OPTICS	GDD	CDIBM
(a)	**1.000**	0.997	**1.000**	**1.000**	0.996	**1.000**
(b)	**1.000**	**1.000**	**1.000**	**1.000**	0.997	**1.000**
(c)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.670
(d)	**0.979**	0.183	0.944	0.670	0.796	**0.979**
(e)	0.446	0.612	0.999	0.993	0.983	**1.000**
(f)	**0.941**	0.926	0.877	0.711	0.559	0.924
(g)	0.465	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
(h)	**1.000**	**1.000**	0.941	**1.000**	0.917	**1.000**
(i)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Min.	0.446	0.183	**0.877**	0.670	0.559	0.670
Avg.	0.870	0.857	**0.973**	0.930	0.916	0.952

is an advantage of the proposed algorithm, and it doesn't require fine tuning the parameters for each dataset to achieve this performance.

We also investigated the performance of our algorithm on the real world GPS trajectory [34] dataset, and the results are given in Table 6. The best performance is provided by the GDD algorithm in terms of both NMI and ARI. The proposed algorithm is among the best ones.

Density initialization and cluster formation are the two parts of the proposed algorithm having the most of the computational cost. Density initialization requires calculating the determinant of the matrix $\mathbf{H}$, and it is done N times for all $\mathbf{X}$. This has a complexity of $\mathcal{O}\left(N^2 d^{2.373}\right)$ [49]. We use Fuzzy C-Means with Mahalanobis Distance, which is very similar to Expectation Maximization for Gaussian Mixture Models, with a complexity of $\mathcal{O}\left(N^2 M N_{iter}\right)$ for N_{iter} number of iterations [27]. The computational cost of the rest of the algorithm is proportional to either M or N, and, therefore, they are negligible. In total, the cost of the proposed algorithm is $\mathcal{O}\left[N^2(d^{2.373} + M N_{iter})\right]$.

Table 2. Normalized Mutual Information results for reference algorithms and datasets.

Dataset	Spectral	Agglom.	DBSCAN	OPTICS	GDD	CDIBM
(a)	**1.000**	0.993	**1.000**	**1.000**	0.989	**1.000**
(b)	**1.000**	**1.000**	**1.000**	**1.000**	0.992	**1.000**
(c)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.707
(d)	**0.984**	0.571	0.949	0.883	0.916	0.982
(e)	0.689	0.809	0.997	0.969	0.944	**1.000**
(f)	**0.915**	0.898	0.862	0.732	0.671	0.901
(g)	0.744	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
(h)	**1.000**	**1.000**	0.902	**1.000**	0.842	**1.000**
(i)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Min.	0.689	0.571	**0.862**	0.732	0.671	0.707
Avg.	0.925	0.919	**0.967**	0.953	0.928	0.954

Table 3. Compared algorithms' parameters in general.

Algorithm	**Parameters**
Spectral	Cluster count, eigen_solver="arpack", affinity="nearest_neighbors"
Agglomerative	Cluster count, linkage="average", affinity="cityblock"
GDD	No input parameters
CDIBM	α=0.3, k=6, FCM_iteration=10, m=1.1

Table 4. Dataset specific parameters for DBSCAN algorithm.

Parameters	**Dataset**									
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Real world
min_samples	0.3	0.3	0.3	0.1	0.3	0.2	0.5	0.15	0.3	0.0003
eps	4	4	4	10	4	10	4	10	4	4

Table 5. Dataset specific parameters for OPTICS algorithm.

Parameters	**Dataset**									
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Real world
x_samples	0.25	0.25	0.25	0.1	0.25	0.035	0.25	0.25	0.25	0.0005
xi	20	20	20	50	20	5	20	20	20	4
min_cluster_size	None	None	None	None	None	0.2	None	None	None	None

Table 6. NMI and ARI performance metrics of algorithms for real world dataset.

Metric	Spectral	Agglom.	DBSCAN	OPTICS	GDD	CDIBM
ARI	0.149	0.027	0.107	0.005	**0.280**	0.115
NMI	0.572	0.390	0.523	0.519	**0.644**	0.557

Table 7. Computational complexities of used algorithms.

Algorithm	Spectral	Agglom.	DBSCAN	OPTICS	GDD	CDIBM
Complexity	$\mathcal{O}\left(N^3\right)$	$\mathcal{O}\left(N^2\log(N)\right)$	$\mathcal{O}\left(N\log(N)\right)$	$\mathcal{O}\left(N\log(N)\right)$	$\mathcal{O}\left(N^2\right)$	$\mathcal{O}\left[N^2(d^{2.373}+MN_{iter})\right]$

We compared the complexities in Table 7. The proposed algorithm depends on the dimension exponentially, but it brings a new perspective and may lead to new studies to decrease its computational complexity.

5. Conclusion

The proposed algorithm uses a centroid based method and it is initialized with a kernel density estimation. Bhattacharyya measure is used to determine subclusters to be merged. We use Jensen inequality to find optimal clusters. The proposed algorithm has remarkable advantages especially for imbalanced and arbitrary shaped data and doesn't require one to pre-specify the number of clusters. It fails only for very thin shaped clusters. It overperforms OPTICS and GDD on syntetic data, and it shows a similar performance with DBSCAN. The computational complexity of the proposed algorithm is mostly affected by the kernel density initialization algorithm. A computationally efficient initialization algorithm can be integrated as future work.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. Erdem Köse gave the idea and did the experiments. Erdem Köse and Ali Köksal Hocaoğlu interpreted the results and wrote the paper.

References

[1] Forgy EW. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics 1965; 21: 768-769.

[2] Bandyopadhyay A, Deb K, Das A, Bag R. Impulse noise removal by k-means clustering identified fuzzy filter: a new approach. Turkish Journal of Electrical Engineering & Computer Sciences 2020; 28 (5): 2838-2862. doi: 10.3906/elk-1910-34

[3] Rezaee MJ, Eshkevari M, Saberi M, Hussain O. GBK-means clustering algorithm: An improvement to the K-means algorithm based on the bargaining game. Knowledge-Based Systems 2021; 213: 106672. doi: 10.1016/j.knosys.2020.106672

[4] Bezdek JC, Ehrlich R, Full W. FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences 1984; 10 (2-3): 191-203. doi: 10.1016/0098-3004(84)90020-7

[5] Gueorguieva N, Valova I, Georgiev G. M&MFCM: fuzzy c-means clustering with Mahalanobis and Minkowski distance metrics. Procedia Computer Science 2017; 114: 224-233. doi: 10.1016/j.procs.2017.09.064

[6] Tellaroli P, Bazzi M, Donato M, Brazzale AR, Drăghici S. Cross-clustering: a partial clustering algorithm with automatic estimation of the number of clusters. PloS One 2016; 11 (3): e0152333. doi: 10.1371/journal.pone.0152333

[7] Gupta A, Datta S, Das S. Fast automatic estimation of the number of clusters from the minimum inter-center distance for k-means clustering. Pattern Recognition Letters 2018; 116: 72-79. doi: 10.1016/j.patrec.2018.09.003

[8] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining; Portland, Oregon, USA; 1996. vol. 96, pp. 226-231. doi: 10.5555/3001460.3001507

[9] Schubert E, Sander J, Ester M, Kriegel HP, Xu X. DBSCAN revisited, revisited: why and how you should still use DBSCAN. ACM Transactions on Database Systems 2017; 42 (3): 1-21. doi: 10.1145/3068335

[10] Ankerst M, Breunig MM, Kriegel HP, Sander J. OPTICS: ordering points to identify the clustering structure. ACM Sigmod record 1999; 28 (2): 49-60. doi: 10.1145/304181.304187

[11] Hinneburg A, Gabriel HH. Denclue 2.0: Fast clustering based on kernel density estimation. In: 7th International Symposium on Intelligent Data Analysis; Ljubljana, Slovenia; 2007. pp. 70-80. doi: 10.1007/978-3-540-74825-0_7

[12] Rehioui H, Idrissi A, Abourezq M, Zegrari F. DENCLUE-IM: A new approach for big data clustering. Procedia Computer Science 2016; 83: 560-567. doi: 10.1016/j.procs.2016.04.265

[13] Güngör E, Özmen A. Distance and density based clustering algorithm using Gaussian kernel. Expert Systems with Applications 2017; 69: 10-20. doi: 10.1016/j.eswa.2016.10.022

[14] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science 2014; 344 (6191): 1492-1496. doi: 10.1126/science.1242072

[15] Zhu Y, Ting KM, Carman MJ. Density-ratio based clustering for discovering clusters with varying densities. Pattern Recognition 2016; 60: 983-997. doi: 10.1016/j.patcog.2016.07.007

[16] Sieranoja S, Fränti P. Fast and general density peaks clustering. Pattern Recognition Letters 2019; 128: 551-558. doi: 10.1016/j.patrec.2019.10.019.

[17] Yu H, Chen L, Yao J. A three-way density peak clustering method based on evidence theory. Knowledge-Based Systems 2021; 211: 106532. doi: 10.1016/j.knosys.2020.106532

[18] Eldershaw C, Hegland M. Cluster analysis using triangulation. In: Computational Techniques and Applications: CTAC 97; Adelaide, Australia; 1997. pp. 201-208. doi: 10.1142/3834

[19] Liu D, Nosovskiy GV, Sourina O. Effective clustering and boundary detection algorithm based on Delaunay triangulation. Pattern Recognition Letters 2008; 29 (9): 1261-1273. doi: 10.1016/j.patrec.2008.01.028

[20] Deng M, Liu Q, Cheng T, Shi T. An adaptive spatial clustering algorithm based on Delaunay triangulation. Computers, Environment and Urban Systems 2011; 35 (4): 320-332. doi: j.compenvurbsys.2011.02.003

[21] Azzalini A, Torelli N. Clustering via nonparametric density estimation. Statistics and Computing 2007; 17 (1): 71-80. doi: 10.1007/s11222-006-9010-y

[22] Kesemen O Tezel Ö, Özkul E, Tiryaki BK. Fuzzy c-Means Directional Clustering (FCMDC) algorithm using trigonometric approximation. Turkish Journal of Electrical Engineering & Computer Sciences 2020; 28 (1): 140-152. doi: 10.3906/elk-1903-118

[23] Chiu SL. Fuzzy model identification based on cluster estimation. Journal of Intelligent & Fuzzy Systems 1994; 2 (3): 267-278.

[24] Yager RR, Filev DP. Generation of fuzzy rules by mountain clustering. Journal of Intelligent & Fuzzy Systems 1994; 2 (3): 209-219.

[25] Estiri H, Omran BA, Murphy SN. kluster: an efficient scalable procedure for approximating the number of clusters in unsupervised learning. Big Data Research 2018; 13: 38-51. doi: 10.1016/j.bdr.2018.05.003

[26] Zhang ZW, Liu Z, Martin A, Liu ZG, Zhou K. Dynamic evidential clustering algorithm. Knowledge-Based Systems 2021; 213: 106643. doi: 10.1016/j.knosys.2020.106643

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

[28] Mostafa SM. Clustering Algorithms: Taxonomy, Comparison, and Empirical Analysis in 2D Datasets. J. Artif. Intell. 2020; 2 (4): 189. doi: 10.32604/jai.2020.014944

[29] Jain AK, Law MH. Data clustering: A user's dilemma. In: PReMI: International Conference on Pattern Recognition and Machine Intelligence; Kolkata, India; 2005. pp. 1-10. doi: 10.1007/11590316_1

[30] Gionis A, Mannila H, Tsaparas P. Clustering aggregation. ACM Transactions on Knowledge Discovery from Data 2007; 1 (1): 4-es. doi: 10.1145/1217299.1217303

[31] Chang H, Yeung DY. Robust path-based spectral clustering. Pattern Recognition 2008; 41 (1): 191-203. doi: 10.1016/j.patcog.2007.04.010

[32] Fränti P, Sieranoja S. K-means properties on six clustering benchmark datasets. Applied Intelligence 2018; 48 (12): 4743-4759. doi: 10.1007/s10489-018-1238-7

[33] Thrun MC, Ultsch A. Clustering benchmark datasets exploiting the fundamental clustering problems. Data Brief 2020; 30: 105501. doi: 10.1016/j.dib.2020.105501

[34] Cruz MO , Macedo H, Guimaraes A. Grouping similar trajectories for carpooling purposes. In: 2015 Brazilian Conference on Intelligent Systems (BRACIS); Natal, Brazil; 2015. pp. 234-239. doi: 10.1109/BRACIS.2015.36

[35] Terrell GR, Scott DW. Variable kernel density estimation. The Annals of Statistics 1992; 20 (3): 1236-1265.

[36] Hazelton ML. Variable kernel density estimation. Australian & New Zealand Journal of Statistics 2003; 45 (3): 271-284. doi: 10.1111/1467-842X.00283

[37] Bhattacharyya A. On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society 1943; 35: 99-109.

[38] Wainwright MJ. High-dimensional statistics: A non-asymptotic viewpoint. Cambridge, UK: Cambridge University Press, 2019.

[39] Guidoum AC. Kernel Estimator and Bandwidth Selection for Density and its Derivatives: The kedd Package. arXiv preprint 2015. arXiv: 2012.06102

[40] Browne M. A geometric approach to non-parametric density estimation. Pattern Recognition 2007; 40 (1): 134-140. doi: 10.1016/j.patcog.2006.05.012

[41] Peterka T, Croubois H, Li N, Rangel E, Cappello F. Self-adaptive density estimation of particle data. SIAM Journal on Scientific Computing 2016; 38 (5): 646-666. doi: 10.1137/15M1016308

[42] Bar-Shalom Y, Li XR, Kirubarajan T. Estimation with applications to tracking and navigation: theory algorithms and software. USA: John Wiley & Sons, 2004.

[43] Fränti P, Sieranoja S. How much can k-means be improved by using better initialization and repeats?. Pattern Recognition 2019; 93: 95-112. doi: 10.1016/j.patcog.2019.04.014

[44] Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research 2010; 11: 2837–2854.

[45] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 2011; 12: 2825-2830.

[46] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000; 22 (8): 888-905. doi: 10.1109/34.868688

[47] Stella XY, Shi J. Multiclass Spectral Clustering. In: Proceedings Ninth IEEE International Conference on Computer Vision; Nice, France; 2003. pp. 313-319. doi: 10.1109/ICCV.2003.1238361

[48] Davies DL, Bouldin DW. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1979; PAMI-1(2): 224-227. doi: 10.1109/TPAMI.1979.4766909

[49] Kaltofen E, Villard G. On the complexity of computing determinants. Computational Complexity 2005; 13 (3): 91-130. doi: 10.1007/s00037-004-0185-3