ImpKmeans: An Improved Version of the K-Means Algorithm, by Determining Optimum Initial Centroids, based on Multivariate Kernel Density Estimation and Kd-Tree

Ali Şenol

Department of Computer Engineering, Faculty of Engineering, Tarsus University, Engineering Faculty, 33400 Tarsus, Mersin, Turkey
alisenol@tarsus.edu.tr

Abstract: K-means is the best known clustering algorithm, because of its usage simplicity, fast speed and efficiency. However, resultant clusters are influenced by the randomly selected initial centroids. Therefore, many techniques have been implemented to solve the mentioned issue. In this paper, a new version of the k-means clustering algorithm named as ImpKmeans shortly (An Improved Version of K-Means Algorithm by Determining Optimum Initial Centroids Based on Multivariate Kernel Density Estimation and Kd-tree) that uses kernel density estimation, to find the optimum initial centroids, is proposed. Kernel density estimation is used, because it is a nonparametric distribution estimation method, that can identify density regions. To understand the efficiency of the ImpKmeans, we compared it with some state-of-the-art algorithms. According to the experimental studies, the proposed algorithm was better than the compared versions of k-means. While ImpKmeans was the most successful algorithm in 46 tests of 60, the second-best algorithm, was the best on 34 tests. Moreover, experimental results indicated that the ImpKmeans is fast, compared to the selected k-means versions.

Keywords: k-means; clustering; kernel density estimation; centroid initialization; kd-tree

1 Introduction

Clustering algorithms are unsupervised approaches, that separate data into groups, that are called clusters, according to included similarities and dissimilarities [1] [2]. Clustering approaches aim to maximize as much as possible both the similarities among the data in the same group and also the dissimilarity among the data in the different groups. In general terms, clustering algorithms are divided into five parts which are partitioning-based methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods; and DBSCAN [3], OPTICS [4], k-means [5], Affinity Propagation [6], Agglomerative Clustering [7], HDBSCAN [8], and MCMSTClustering [9] are some examples of clustering algorithms. Some of the application areas of clustering are pattern recognition [10, 11], machine learning [12] [13], bioinformatics [14] [15], data mining [16] [17], web mining [1] [18], stream mining [12] [19], etc.

Basic k-means clustering was proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation to define linearly separable clusters. It is one of the partitioning-based clustering algorithms that divides the dataset into k clusters over randomly selected initial centroids. Although k-means is efficient and easy to use, it encounters problems if the dataset is not linearly separable. The main problems related to k-means are as follows:

- The final clusters are dependent on randomly selected centroids. As shown in Figure 1, if the randomly selected centroids are not located optimal, it fails while defining clusters.
- K-means clustering assumes that the shape of the clusters to be found is spherical. However, a minority of the datasets are spherical, and majorities are arbitrary in real-life.
- K-means cannot handle outliers because it partitions the data into k clusters without searching for outliers.
- It encounters some problems if the sizes of clusters are different.
- If the clusters are not linearly separable or overlapped, k-means encounters some issues.

Since the basic k-means clustering algorithm was proposed, many variants of it have been proposed to deal with mentioned issues that are given above [20]. Kernel k-means has been proposed to overcome the problem of identifying clusters that cannot be linearly separated [21] [22]. By using kernel methods, kernel k-means can define non-spherical clusters. However, kernel k-means run-time complexity is high, and its time complexity is high. On the other hand, to meet the need for selecting optimal initial centroids, many advanced versions of k-means were proposed, like k-means++ [22], and algorithms like Fuzzy C-Means, to automatically determine the number of clusters [23] [24]. In k-means++, cluster centers are chosen more innovatively to avoid complete randomness. Centroids are chosen step by step according to the centroids selected before to minimize the cost. This approach makes k-means++ better than basic k-means. However, this approach is not easy to perform. Fritzke proposed K-means-u*, an improved version of k-means++, to improve the limits of k-means++ [25]. But, used operations increase the complexity of the algorithm significantly. Another version of k-means to overcome the issue of selection of initial centroids was proposed by Zhang et al. [26]. In their study, although the accuracy of clustering results improved, it is unsuitable for big datasets because the algorithm's time complexity is high. Zhang et al. [27] proposed an advanced version of k-means based on density canopy to feed the k-means with the best initial centers. They used the canopy algorithm to find the best values for k and initial centroids for k-means clustering. Although it effectively improves clustering quality, their method increases the algorithm's time complexity. As understood from explained versions of k-means clustering algorithm, we need a new k-means-based clustering algorithm that can cluster the dataset more accurately and quickly.

This study proposes a novel approach that uses multivariate kernel density estimation to find optimum initial centroids for k-means. Since kernel density estimation is a nonparametric probability density function (KDE), we use it to find denser regions to select them as initial centroids. It can find denser regions and the degree of density in any data distribution. The main contributions that this article has and state-of-the-art algorithms do not have are summarized as follows:

- The accuracy of k-means clustering algorithms increases thanks to using kernel density estimation to detect centroids of clusters.
- Because the detected centroids are also the final centroids, our approach does not need an iterative procedure to reach final clusters. Final clusters are formed in the first iteration. This method makes our approach very fast, when compared with existing methods.

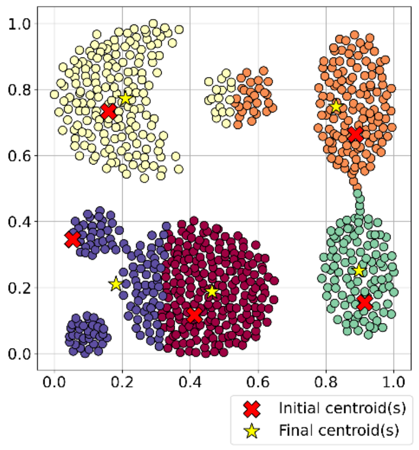

Figure 1
Final clusters of basic k-means according to randomly selected initial centers

The rest of the paper is organized as follows. In the 2nd section, related methods are explained, while in the 3rd section, the problem is stated. Then in the 4th section, details about the proposed algorithm are provided. Then, details of the experimental study are shared in the 5th section, while the work is concluded in the 6th section.

2 Preliminaries

2.1 Basic K-means

K-means clustering is based on the partitioning approach and is the basic clustering algorithm of clustering techniques. Its procedure is simple and effective if the shapes of clusters are spherical and there are no outliers. It uses an iterative approach over randomly selected initial centroids to reach the final cluster. But, just as an example is illustrated in Figure 2, initial centroids affect the final clusters directly. The main objective of iterations is to minimize the standard deviation of the dataset. The objective function of k-means is given in Equation (1).

$$J = min \sum_{j=1}^{k} \sum_{x_i \in C_i} \left\| x_i - \mu_j \right\|^2 \qquad (1)$$

where k is the number of clusters, μ_j is the centroid of the j^{th} cluster, x is a data point, $\left\| x_i - \mu_j \right\|^2$ is the distance from the data point x_i to the cluster center, which is μ_j of the j^{th} cluster. Let X be the data points that construct the dataset; the pseudo-code of

```
Algorithm 1: Standard k-means
  Input: Data points $X = x_1, x_2, x_3, \ldots, x_n \subseteq \mathbb{R}^d$;
  k;
  Output: A set of $k$ centroids: $C = c_1, c_2, c_3, \ldots, c_k \subseteq \mathbb{R}^d$;
  Initialize $C \leftarrow c_1, c_2, c_3, \ldots, c_k \subseteq \mathbb{R}^d$ at random
  while $C$ has not converged do
      $S_i \leftarrow \emptyset, \forall \in [k]$;
      foreach $x_i \in X$ do
          $j^* \leftarrow argmin_j \|x_i - c_j\|$;
          $S_{j*} \leftarrow S_{j*} \cup \{x_i\}$
      end
      $c_j \leftarrow \frac{1}{|S_j|} \sum_{x \in S_j} x, \forall j \in [k]$;
  end
```

k-means is given in Algorithm 1.

2.2 Kernel Density Estimation (KDE)

In the literature, two types of density estimation methods are commonly used. These methods are parametric and nonparametric approaches. Parametric methods like the Gaussian method assume all the data distribution is uniform and most data is gathered around the center in the circle with a standard deviation radius. In contrast, nonparametric methods assume there may be more than one denser area among the data. Namely, according to the parametric methods, there is only one peak on the curve; nonparametric methods assume there may be more than one peak. The probability density function of the univariate normal distribution with mean μ and variance σ^2 is given in Equation (2).

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left[\frac{x-\mu}{\sigma}\right]^2/2} \tag{2}$$

where the value of x is in $-\infty < x < \infty$ interval. On the other hand, in addition to assuming that there may be more than one peak on the curve, nonparametric distribution estimation methods may not be uniform. Let $X = [X_1, \dots, X_n]^T$ be an n-dimensional vector of multivariate Gaussian distribution of n-dimensional mean vector $\mu \epsilon \boldsymbol{R}^n$ and $\sum$ the covariance matrix of $n\ x\ n$ dimensions. Therefore, the multivariate kernel distribution equation will be Equation (3) [28].

$$p(x, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(\pi - \mu)^T \Sigma^{-1}(x - \mu)) \tag{3}$$

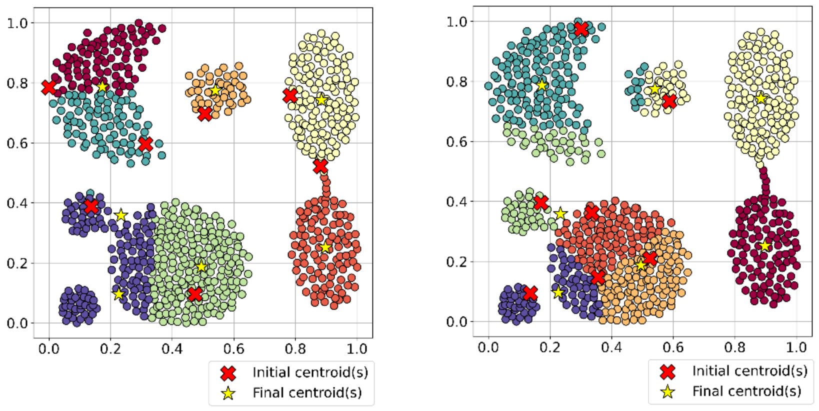

Figure 2
Two examples of the effect of randomly selected centroids on final clusters in standard k-means

As a nonparametric method, kernel density estimation tries to estimate where any new incoming data to locate according to existing data. Owing to this ability, KDE is used in many areas like machine learning, healthcare systems, stock markets, etc. [2]. As we mentioned earlier and as in the example in Figure 3, there can be multiple density peaks on the curve, and there are many types of KDE functions, known as smoothing functions, as given in Figure 4. Then, KDE is calculated as given in Equation (4), where K(.) is one of the functions in Figure 4.

$$\widehat{P_n}(n) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \tag{4}$$

3 Problem Statement

The most important problem related to k-means is centroid initialization. Since the initial centroids are selected randomly in standard k-menas, both final clusters and the accuracy might be affected directly in a negative way. Although many advanced versions of k-means have been proposed, the time complexities of these algorithms are very high. There is still a need for new k-means versions that can determine the best initial centroids and have low time complexity. In this study, we propose a new version of k-means to overcome the mentioned issues.

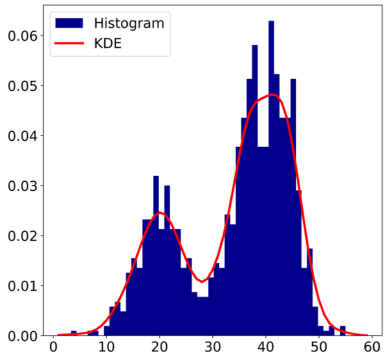

Figure 3

The relationship between histograms and peaks in KDE

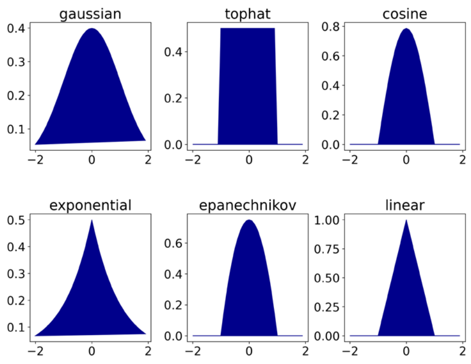

Figure 4

Types of kernel density estimation curves

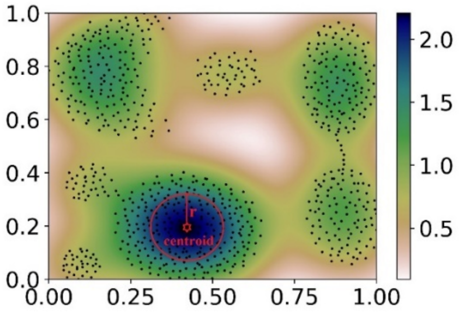

Figure 5

Example of the initial centroid and the radius used to determine the ignorance area on the Aggregation dataset

4 The Proposed Algorithm

This section describes the details of the proposed algorithm. In this study, we propose a new version of k-means clustering to overcome the issue of initial centroids determination. We try to detect peak points in the dataset to reach the goal. To find peak points, we used multivariate kernel density estimation. The purpose of finding peak points is to select determined peak points as initial centroids for k-means. Namely, the points with high KDE are candidates for initial centroids. Therefore, we select k points as initial centroids. However, as shown in Figure 5, selected k points with the highest values may be too close to each other. If these points were chosen as the initial centroids, the accuracy of the final clusters would be reduced. Therefore, in ImpKmeans, we use one more predefined parameter: ignorance radius. When searching for a new initial centroid using kd-tree and range search, we ignore data around previously selected centroids that are inside the radius of ignorance. Now, let's give more details and define the parameters used in ImpKmeans.

4.1 Definitions

Definition 1 (ignorance radius - ignorance_r): The ignorance radius determines the ignorance area around each selected centroid. This approach makes it possible for our algorithm to overcome local maxima. As given in Figure 5, if we didn't use this approach, all initial centroids would be selected from the same denser regions. This approach makes it possible to select initial centroids from different denser regions.

Definition 2 (the number of clusters - k): The number of clusters is the predefined number of clusters the user enters. However, this does not necessarily mean that there will always be k final clusters. In ImpKmeans, the formed clusters may be less than the selected k value.

Definition 3 (MultiKDE): As processed data is multidimensional, in ImpKmeans, we calculate the multivariate kernel density estimation value for each data. In addition to applying KDE to univariate data, we can apply it to multivariate datasets. To adapt the KDE to process multivariate datasets, we should use a kernel constructed by a product kernel or a radial basis function to process multidimensional datasets. Let's handle a 2-dimensional dataset. Let $X = (X_1, X_2, X_3, \dots, X_d)'$ be a sample of multivariate random variables with the density of *f(x)* defined on R^d and $\{x_1, x_2, x_3 \dots, x_n\}$ be an independent sample taken from *f(x)*. Then the multivariate kernel density estimation is calculated by Equation (5), where *K(.)* is a multivariate kernel function, and *h* is a positive bandwidth matrix.

$$\widehat{fh}(x) = \frac{1}{n|h|^{-\frac{1}{2}}} \sum_{i=1}^{n} K(h^{-\frac{1}{2}}(x - X_i)) \tag{5}$$

Definition 4 (kd-tree-based rangesearch): Figure 6 shows that kd-tree is a tree data type that can process multidimensional datasets. While placing the data into the tree, it evaluates one dimension in each step. On the other hand, a rangesearch operation is an operation that is performed on any kd-tree to find the data inside a circle of radius of *r*. The reason to use this approach in our algorithms is that the kd-tree and the rangesearch have low computational complexity

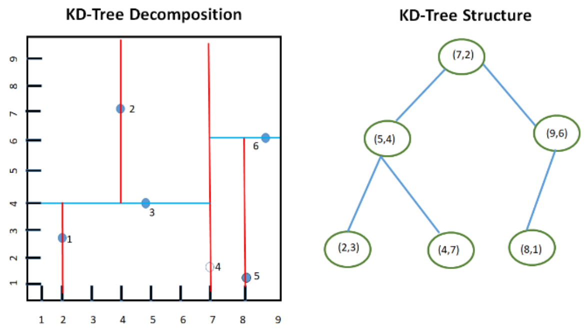

Figure 6
Decomposition of the kd-tree dataset

Algorithm 2: ImpKmeans

```
Input: Data points $X = x_1, x_2, x_3, \dots, x_n \subseteq \mathbb{R}^d$;
k;
r;
Output: A set of $k$ centroids: $C = c_1, c_2, c_3, \dots, c_k \subseteq \mathbb{R}^d$;
$IntitialCentroids \leftarrow findInititalCentroids(X, k, r)$;
foreach $i \in k$ do
    $InitCentroids[i] \leftarrow argmax(MultiKDE)$;
    $MultiKDE \leftarrow deleteMultiKDE(r, InitCentroids[i])$
end
$S_i \leftarrow \emptyset, \forall \in [k]$;
foreach $x_i \in X$ do
    $j^* \leftarrow argmin_j \|x_i - c_j\|$;
    $S_{j*} \leftarrow S_{j*} \cup \{x_i\}$
end
$c_j \leftarrow \frac{1}{|S_j|} \sum_{x \in S_j} x, \forall j \in [k]$;
```

Algorithm 3: findInitialCentroids

```
Input: Data points $X = x_1, x_2, x_3, \dots, x_n \subseteq \mathbb{R}^d$;
k;
r;
Output: A set of $k$ centroids: $C = c_1, c_2, c_3, \dots, c_k \subseteq \mathbb{R}^d$;
foreach $i \in k$ do
    $MultiKDE \leftarrow \frac{1}{n|h|^{-\frac{1}{2}}} \sum_{i=1}^{n} K(h^{-\frac{1}{2}}(x - X_i))$;    ▷calculate MultiKDE
    $InitCentroids[i], j \leftarrow argmax(MultiKDE)$;    ▷find peak point
    $kdtree \leftarrow KDTree(X)$;    ▷place data into kd-tree
    $ind \leftarrow rangesearch(kdtree, r, x_j)$;    ▷find the data inside the ignorance area
    $X \leftarrow delete(X, ind)$;    ▷delete the data in ignorance data
end
return $InitCentroids$
```

4.2 Algorithm

As we explained above, the main contributions of ImpKmeans are that it can detect the best initial centroids for k-means and does not need an iterative search method to reach final clusters. As an example, illustrated in Figure 5, KDE makes it possible to find the denser areas, and ignorance radius makes it possible to find the initial centroids in different regions. ImpKmeans algorithm is divided into two parts:

- In the initial stage, kernel density estimation finds peak points in the dataset. k points that minimize the cost are selected as initial centroids.
- In the second stage, initial centroids proposed by kernel density estimation are given to basic k-means as initial centroids, and the dataset is clustered according to these centroids in only one iteration.

According to the abovementioned equations and explanations about our method, the pseudo-code of ImpKmeans is given Algorithms 2 and 3.

4.3 Time Complexity

Let *n* be the number of vectors of d-dimensions, *k* be the number of clusters in the dataset, and *i* be the number of iterations needed to be converged; the comparison of the time complexity of the proposed algorithm with the state-of-the-art algorithm is given in Table 1. Because we use kd-tree construction and range search on it, the time complexity of our algorithm is the summation of *O(dn logn)* for constructing the kd-tree and O($dn^{1-\frac{1}{d}+k}$) for reangesearch operation. In addition, *O(nkd)* is the complexity of assigning the data to selected initial centroids. Therefore, the allover time complexity of our algorithm is O(*dn logn* + $dn^{1-\frac{1}{d}+k}$ + *nkd*). This complexity could be simplified as O($dn^{1-\frac{1}{d}+k}$). As our algorithm does not use an iterative approach, it is expected to be faster, compared with the other algorithms.

Table 1
Time complexity comparison of algorithms

Algorithm	Complexity
k-means	*O(nkdi)*
k-mediods	$O(n^2kdi)$
k-means++	$O(n^2k^2di)$
FCM	*O(nkdi)*
ImpKmeans	*O(nkd)*

5 Experimental Study

5.1 Development Environment

In this study, to measure the efficiency of our algorithm, we tested it on synthetic and real datasets in the Anaconda environment by using Python programming language with the needed libraries. To measure its clustering accuracy and speed, we compared it with some state-of-the-art algorithms like k-means, kmeans++, k-medoids, and Fuzzy C-Means. All experimental studies were performed on a computer with 16 GB RAM, an Intel i7 processor, and Windows 11 operating system installed.

5.2 Experimental Setup

To be sure that each data is in the same range and to select parameters easily, the data were normalized with the min-max normalization. The equation of min-max normalization is given in Equation (6).

$$z_{ij} = \frac{x_{ij} - minx_j}{\max x_j - \min x_j} \tag{6}$$

Additionally, we used ARI (Adjusted Rand Index), Purity, and Silhouette Index to evaluate and compare the clustering quality of the algorithms. Equations about these indices are given in Equations (7), (8), and (9), respectively, where *n_{ij}, a_i, b_j,* be values obtained from the contingency table, k the number of clusters, and c and t are the maximum count of data in the related clusters.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_i}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{a_i}{2} + \sum_j\binom{b_i}{2}\right] - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_i}{2}\right]/\binom{n}{2}} \tag{7}$$

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j \left|c_i \cap t_j\right| \tag{8}$$

$$SI = \frac{1}{n}\sum_{i=1}^{k}\sum_{x \in C_i} \frac{b(\boldsymbol{x}) - a(\boldsymbol{x})}{max\ (a(\boldsymbol{x}), b(\boldsymbol{x}))} \tag{9}$$

where *a(x)* is the average distance to all the data of the cluster that *x* is in, and *b(x)* is the average distance to all the data of the closest cluster that x is not in.

5.3 Used Datasets

Synthetic and real datasets were used in the experimental study to compare the success of our algorithm with the state-of-the-art algorithms. Since the main purpose of our approach is to improve the accuracy of k-means and reduce the time complexity, the selected datasets are spherical in general. On the other hand, to measure the efficiency of our algorithm on the imbalanced dataset, we select some imbalanced datasets like Outliers, Aggregation, and Thyroid. Details of the datasets used in the experimental study are given in Table 2.

Table 2
Used datasets

Dataset	Type	# of Features	# of data	# of class	Reference
Outliers	Synthetic	2	700	4	[29]
Corners	Synthetic	2	2000	4	[29]
Iris	Real	4	150	3	[30]
Breast Cancer	Real	8	699	2	[30]
Aggregation	Synthetic	2	788	7	[31]
Thyroid	Real	4	215	2	[30]
Xclara	Synthetic	2	3000	3	[32]
Twenty	Synthetic	2	1000	20	[33]
2d-10c	Synthetic	2	2990	10	[33]
2d-20c	Synthetic	2	1517	20	[33]
2d-3c	Synthetic	2	625	3	[33]
2d-4c	Synthetic	2	1260	4	[33]
D31	Synthetic	2	3100	31	[34]
R15	Synthetic	2	600	15	[34]
Diamond9	Synthetic	2	3000	9	[33]
Sizes1	Synthetic	2	1000	4	[33]
DS-850	Synthetic	2	850	5	[33]
Fourty	Synthetic	2	1000	40	[33]
S-set1	Synthetic	2	5000	16	[33]
St900	Synthetic	2	900	9	[33]

5.4 Experimental Procedure and Parameter Setting

In the experimental study, we used a random search method with randomly selected parameters to reach the best results for each algorithm. We run each algorithm on each dataset 100 times with randomly selected parameters of each algorithm for each index (ARI, Purity, and SI). The highest obtained value of each index on each dataset was the best value for the selected algorithm. Similarly, the parameters enabling us to reach this value were the best. On the other hand, we also compared the speed of algorithms on selected datasets.

5.5 Results on Both Synthetic and Real Datasets

We used the procedure explained in Section 5.4 to find the best parameters for each algorithm and clustering results. Obtained results are shown in Tables 3, 4 and 5. The ARI values of ImpKmeans, k-means, k-medoids, FCM, and k-medoids are shared in Table 3. Additionally, visual results are provided in Figure 7. According to the results, it is obvious that our algorithm is more successful on 16 datasets over 20 datasets. On the other hand, k-means, k-means++, k-medoids, and FCM were the best on 9, 12, 6 and 8 datasets over 20 datasets, respectively. Iris, D31, Sizes1, and DS-850 were the datasets in that our algorithm was not the most successful. But the ARI values that our algorithm achieved were very close to the best values. Therefore, we can say that our algorithm is the best in datasets by the aspect of ARI.

Regarding Purity, our algorithm was the most successful on 15 datasets over 20, while k-means, k-means++, k-medoids, and FCM were the best on 11, 7, 5 and 12 datasets over 20 datasets, respectively. When we examined our algorithms' results on the datasets in which it was not the best; its Purity values were very close to the best. So, in Purity, our algorithm is very competitive compared to the other algorithms. As for SI, it was seen that our algorithm was the most successful on 15 datasets over 20 with k-means++, while k-means, k-medoids, and FCM were the most successful on 9, 6 and 8 datasets. As real datasets, we tested the algorithms on Iris, Breast Cancer, and Thyroid. Our algorithm was more successful in Breast Cancer and Thyroid. As for Iris, our algorithm is very close to the best values. Consequently, as presented in Table 6, our algorithm appears to be more successful when compared with the other algorithms.

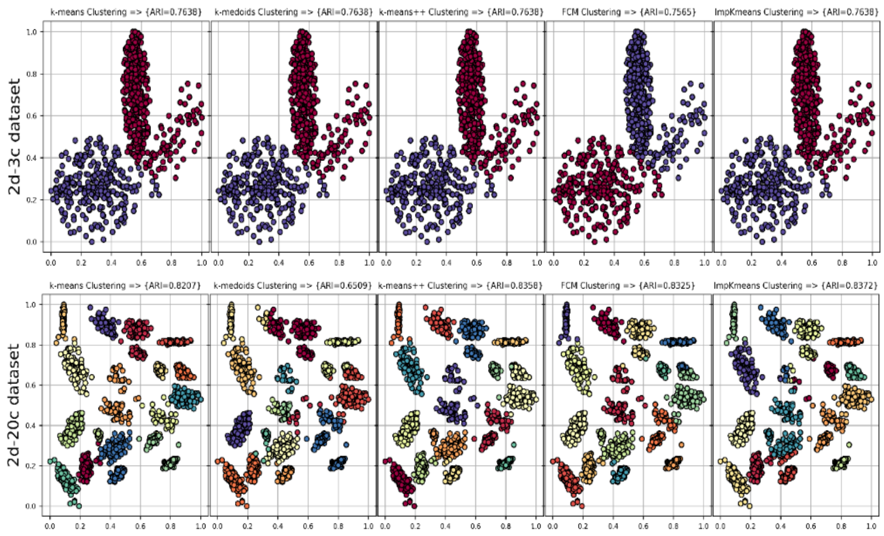

Figure 7
Cont.

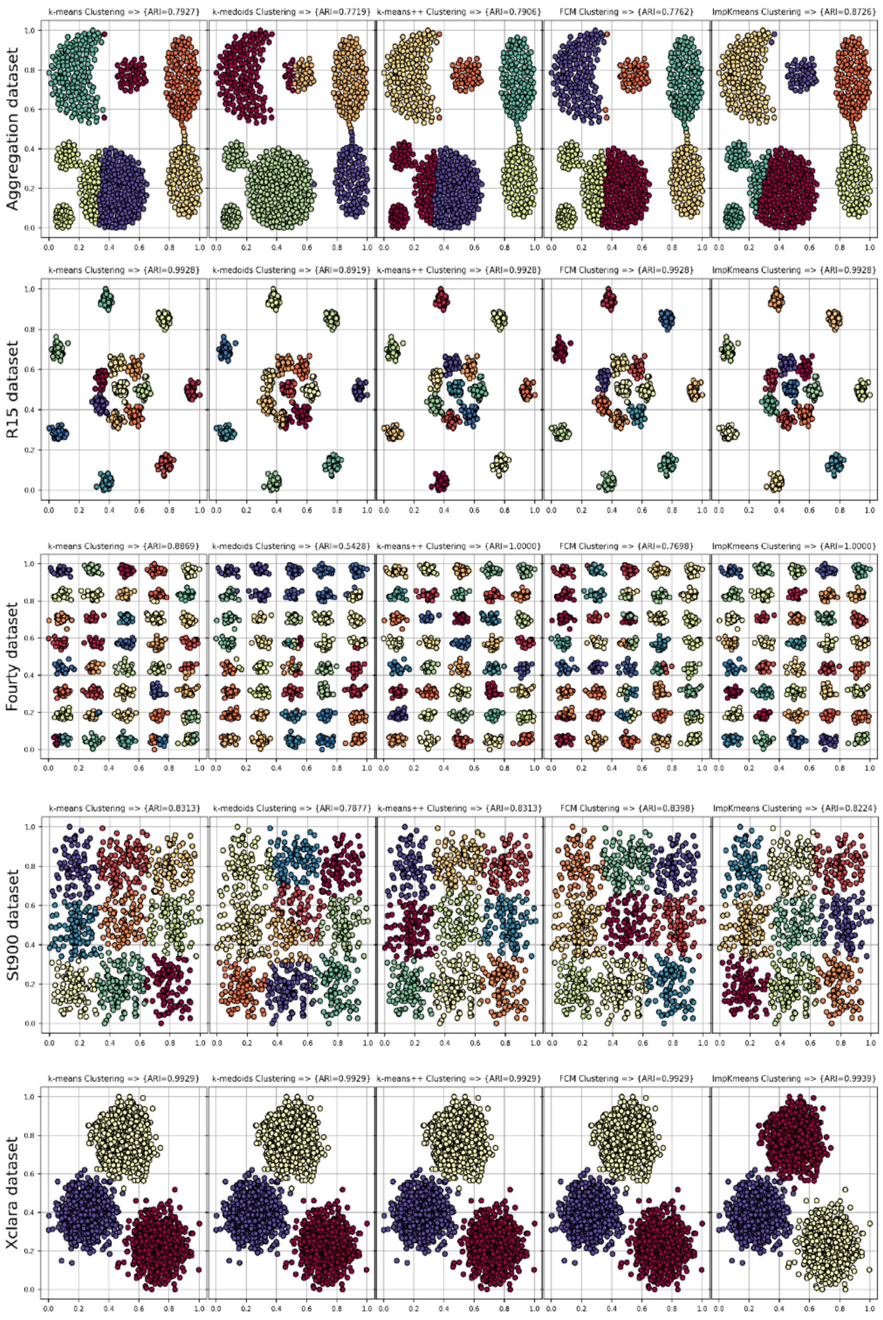

Figure 7
Cont.

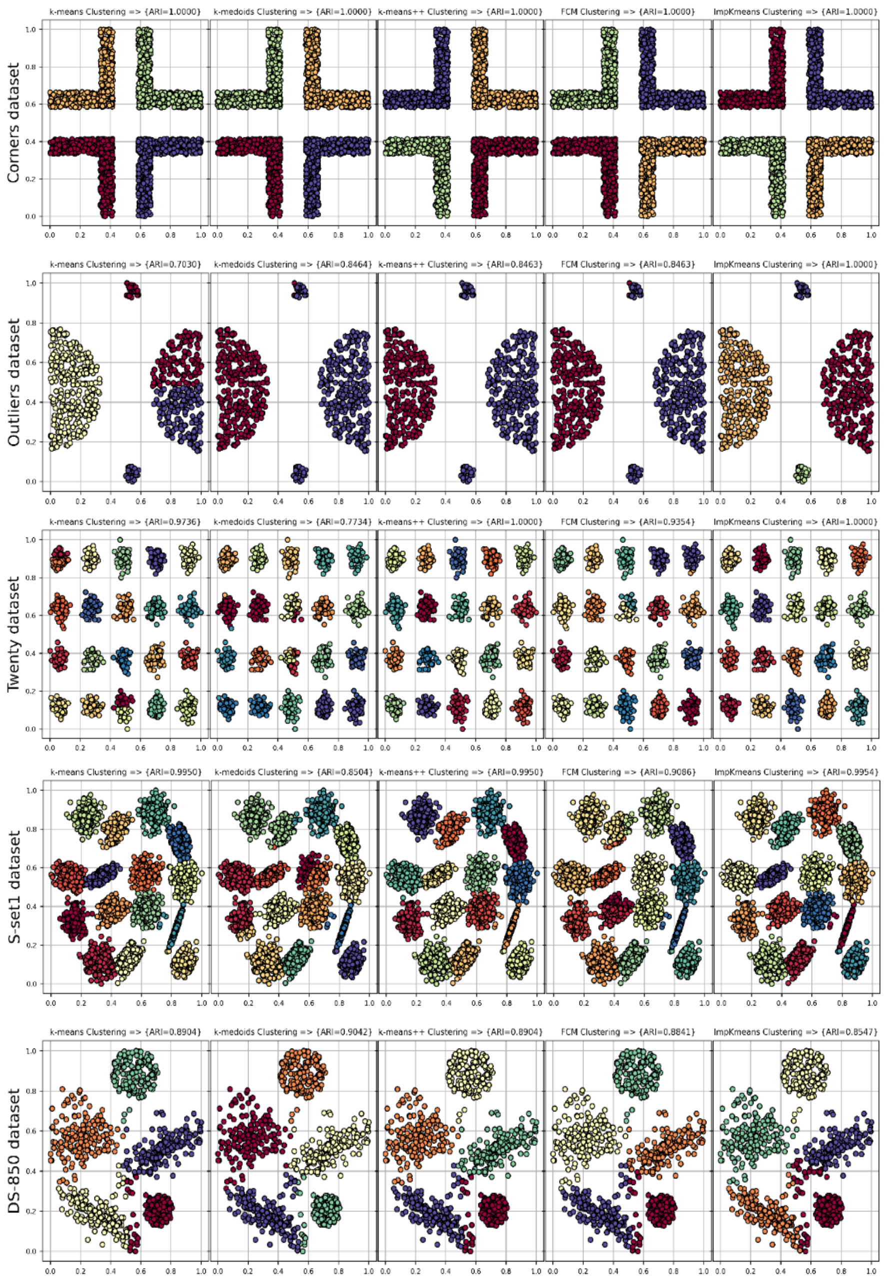

Figure 7

Cont.

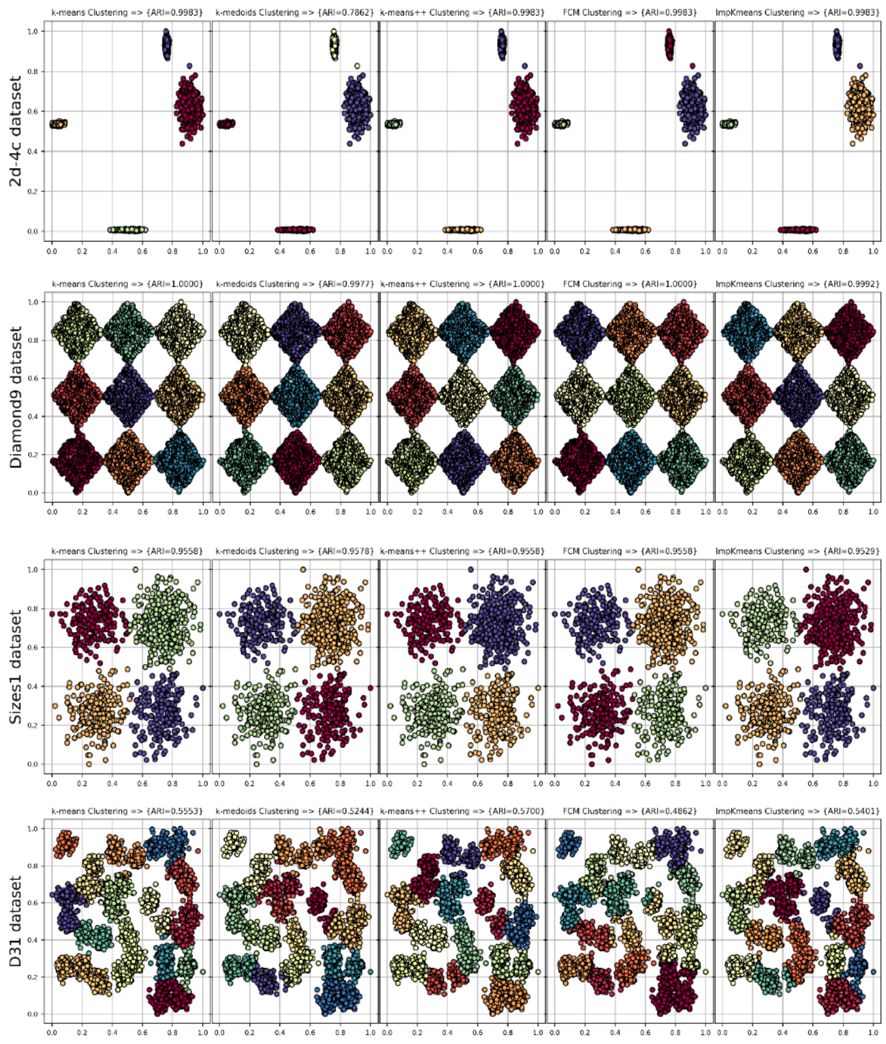

Figure 7

Visual clustering results (ARI values) of algorithms on the synthetical datasets

Table 3

ARI evaluation

	FCM	Lloyd's k-means	k-means++	k-medoids	ImpKmeans
Outliers	0.8463	0.703	0.8463	0.8464	**1**
Corners	**1**	**1**	**1**	**1**	**1**
Iris	0.7287	0.7163	0.7163	**0.743**	0.7163
Breast Cancer	0.8178	**0.8391**	**0.8391**	0.8284	**0.8391**

Aggregation	0.7762	0.7927	0.7906	0.7719	**0.8239**
Thyroid	0.6927	0.6283	0.6283	0.2055	**0.8434**
Xclara	**0.9929**	**0.9929**	**0.9929**	**0.9929**	**0.9929**
Twenty	**1**	0.9736	**1**	0.7734	**1**
2d-10c	**0.9967**	**0.9967**	**0.9967**	0.804	**0.9967**
2d-20c	0.8325	0.8207	0.8358	0.6509	**0.8377**
2d-3c	0.7565	**0.7638**	**0.7638**	**0.7638**	**0.7638**
2d-4c	**0.9983**	**0.9983**	**0.9983**	0.7862	**0.9983**
D31	0.5254	0.5553	**0.57**	0.5244	0.5489
R15	**0.9928**	**0.9928**	**0.9928**	0.8919	**0.9928**
Diamond9	**1**	**1**	**1**	0.9977	**1**
Sizes1	0.9558	0.9558	0.9558	**0.9578**	0.9558
DS-850	0.8841	0.8904	0.8904	**0.9042**	0.8904
Fourty	0.9006	0.8869	**1**	0.5428	**1**
S-set1	0.995	0.995	0.995	0.8504	**0.9954**
St900	**0.8398**	**0.8313**	**0.8313**	0.7877	**0.8313**

Table 4
Purity evaluation

	FCM	**Lloyd's k-means**	**k-means++**	**k-medoids**	**ImpKmeans**
Outliers	**1**	**1**	**1**	**1**	**1**
Corners	**1**	**1**	**1**	**1**	**1**
Iris	**0.98**	0.96	0.9667	0.9667	0.9733
Breast Cancer	0.9671	0.97	0.97	**0.9742**	**0.9742**
Aggregation	0.9949	**0.9962**	0.9949	**0.9962**	0.9949
Thyroid	0.9488	**0.9628**	0.9535	0.9023	**0.9628**
Xclara	**0.9997**	0.9987	0.999	0.9977	**0.9997**
Twenty	**1**	**1**	**1**	0.8	**1**
2d-10c	**0.9997**	0.9993	0.9993	0.8013	**0.9997**
2d-20c	**0.8471**	0.8154	0.8451	0.6711	0.8457
2d-3c	0.9902	**0.9944**	0.993	0.993	**0.9944**
2d-4c	**1**	**1**	**1**	**1**	**1**
D31	**0.4835**	**0.4835**	0.4823	0.4777	**0.4835**
R15	**0.9967**	**0.9967**	**0.9967**	0.9317	**0.9967**
Diamond9	**1**	**1**	**1**	0.999	**1**
xxSizes1	0.984	**0.986**	0.983	0.983	0.985
DS-850	0.9953	0.9988	0.9988	0.9918	**1**
Fourty	0.95	0.925	**1**	0.475	**1**
S-set1	0.9976	0.9976	0.9976	0.869	**0.9978**
St900	**0.9256**	0.9211	0.9211	0.9033	0.9211

Table 5
SI evaluation

	FCM	Lloyd's k-means	k-means++	k-medoids	ImpKmeans
Outliers	0.6128	0.5173	**0.6136**	0.6119	**0.6136**
Corners	0.5697	0.5534	0.5698	0.4693	**0.5699**
Iris	0.618	**0.6295**	**0.6295**	**0.6295**	**0.6295**
Breast Cancer	0.597	0.5966	0.5966	**0.5968**	0.5966
Aggregation	0.5279	0.5365	0.5365	**0.5385**	0.5365
Thyroid	0.5382	0.5755	**0.5852**	0.1909	**0.5852**
Xclara	**0.6945**	**0.6945**	**0.6945**	**0.6945**	**0.6945**
Twenty	**0.738**	0.6993	**0.738**	0.5472	**0.738**
2d-10c	**0.8368**	**0.8368**	**0.8368**	0.724	**0.8368**
2d-20c	0.6133	0.5967	**0.6172**	0.5374	0.6171
2d-3c	0.5517	0.5557	0.5557	0.5557	**0.5562**
2d-4c	**0.8738**	**0.8738**	**0.8738**	0.7184	**0.8738**
D31	0.4606	0.4821	**0.4832**	0.4483	0.4791
R15	**0.7528**	**0.7528**	**0.7528**	0.6843	**0.7528**
Diamond9	**0.5487**	**0.5487**	**0.5487**	0.5486	**0.5487**
xxSizes1	**0.5934**	**0.5934**	**0.5934**	**0.5934**	**0.5934**
DS-850	0.5635	0.5646	0.5646	**0.5652**	0.5646
Fourty	0.6082	0.6206	**0.6852**	0.4324	**0.6852**
S-set1	**0.7116**	**0.7116**	**0.7116**	0.6116	**0.7116**
St900	0.4417	**0.4436**	**0.4436**	0.4201	**0.4436**

Table 6
Overall cluster quality comparisons of the algorithms

Algorithms	ARI	Purity	SI	Total
k-means	9	11	9	29
m-medoids	6	5	6	17
k-means++	12	7	**15**	34
FCM	8	12	8	28
ImpKmeans	**16**	**15**	**15**	**46**

5.6 Speed Analysis

As we explained in Table 1, our algorithm is expected to be fair regarding run-time complexity. Because our algorithm does not use any iterative approach. In our approach, the time-consuming stage is the initial stage, in which the kernel density estimation based on initial centroids is determined. Experimental studies also support our idea, as seen in Figure 8. On the other hand, in some datasets, like Twenty, in which the number of clusters is high compared to the others, the consumed time in the ImpKmeans is slightly higher. In other words, we can say that the higher the number of clusters, the higher the run-time complexity for ImpKmeans.

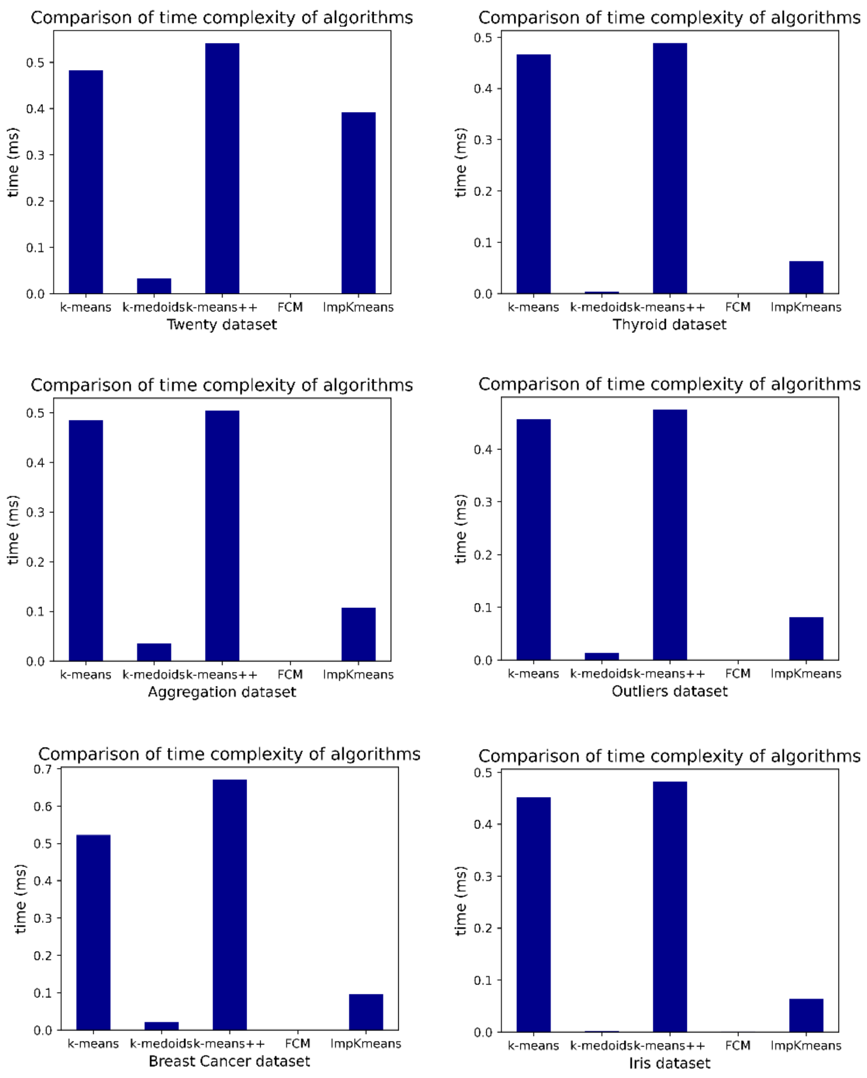

Figure 8

Time comparison of the algorithms on some of the used datasets

Conclusion and Future Work

In this study, we proposed a new advanced version of k-means, named "ImpKmeans", shortly to overcome the issues of initial centroid determination and the time complexity, that the other versions of k-means face. Our approach gets its power from using multivariate kernel density estimation, to find the denser regions among the data. Eligible k number of peaks, are selected as initial centroids, according to density. This approach makes our algorithm superior to the compared algorithms, in terms of clustering quality. Moreover, since the selected initial centroids are also the final cluster centroids, our algorithm produces the final clusters, in only one iteration. This approach makes our algorithm both, more effective and faster.

A significant experimental result was observed while testing algorithms on Outliers and Thyroid datasets. Our algorithm reached 100% and 84.34%, while the second-best algorithm reached 84.64% and 69.27% clustering quality, respectively. As the experimental studies also support, our algorithm has both successful clustering quality and low time complexity.

In the future, plans to examine various studies, addressing datasets with arbitrary-shaped clusters, will be conducted.

Code availability

Python implementation of the proposed clustering algorithm is shared on GitHub (https://github.com/senolali/ImpKmeans).

References

[1] Aggarwal, C. C. and C.K. Reddy, Data Clustering: Algorithms and Applications. 2014: CRC Press Taylor and Francis Group

[2] Węglarczyk, S. Kernel density estimation and its application. in ITM Web of Conferences. 2018, EDP Sciences

[3] Ester, M., H.-P. Kriegel, J. Sander, and X. Xu, 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. 1996, AAAI Press: Portland, Oregon. pp. 226-231

[4] Ankerst, M., M. M. Breunig, H.-P. Kriegel, and J. Sander, OPTICS: ordering points to identify the clustering structure. SIGMOD Rec., 1999. 28(2): pp. 49-60

[5] Lloyd, S. P., Least squares quantization in PCM. IEEE Trans. Inf. Theory, 1982. 28: pp. 129-136

[6] Frey, B. J. and D. Dueck, Clustering by Passing Messages Between Data Points. 2007, 315(5814): pp. 972-976

[7] Lance, G. N. and W. T. Williams, A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. The Computer Journal, 1967, 9(4): pp. 373-380

[8] Campello, R. J. G. B., D. Moulavi, and J. Sander. Density-Based Clustering Based on Hierarchical Density Estimates. in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2013

[9] Şenol, A., MCMSTClustering: defining non-spherical clusters by using minimum spanning tree over KD-tree-based micro-clusters. Neural Computing and Applications, 2023. 35(18): pp. 13239-13259

[10] Sathya, B. and R. Manavalan, Image Segmentation by Clustering Methods: Performance Analysis. International Journal of Computer Applications, 2011, 29: pp. 27-32

[11] Li, C., F. Kulwa, J. Zhang, Z. Li, H. Xu, and X. Zhao, A Review of Clustering Methods in Microorganism Image Analysis, in Information Technology in Biomedicine, E. Pietka, et al., Editors. 2021, Springer International Publishing: Cham. pp. 13-25

[12] Şenol, A. and H. Karacan, A Survey on Data Stream Clustering Techniques. European Journal of Science and Technology, 2018(13): pp. 17-30

[13] Kumar, V., M. S. Chauhan, and S. Khan, Application of Machine Learning Techniques for Clustering of Rainfall Time Series Over Ganges River Basin, in The Ganga River Basin: A Hydrometeorological Approach. 2021, Springer, pp. 211-218

[14] Yu, Z., H.-S. Wong, and H. Wang, Graph-based consensus clustering for class discovery from gene expression data. Bioinformatics, 2007, 23(21): pp. 2888-2896

[15] Zou, Q., G. Lin, X. Jiang, X. Liu, and X. Zeng, Sequence clustering in bioinformatics: an empirical study. Briefings in bioinformatics, 2020, 21(1): pp. 1-10

[16] Han, J., M. Kamber, and J. Pei, Data mining concepts and techniques third edition. The Morgan Kaufmann Series in Data Management Systems, 2011. 5(4): pp. 83-124

[17] Sabor, K., D. Jougnot, R. Guerin, B. Steck, J.-M. Henault, L. Apffel, and D. Vautrin, A data mining approach for improved interpretation of ERT inverted sections using the DBSCAN clustering algorithm. Geophysical Journal International, 2021

[18] Rambabu, M., S. Gupta, and R. S. Singh, Data Mining in Cloud Computing: Survey, in Innovations in Computational Intelligence and Computer Vision. 2021, Springer. pp. 48-56

[19] Şenol, A. and H. Karacan, Kd-tree and adaptive radius (KD-AR Stream) based real-time data stream clustering. Journal of the Faculty of EngineeringArchitecture of Gazi University, 2020. 35(1): pp. 337-354

[20] Şenol, A., Standard Deviation-based Centroid Initialization for K-means, in 3rd International Anatolian Congress on Scientific Research. 2022: Kayseri. pp. 523-530

[21] Schölkopf, B., A. Smola, and K. Müller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 1998, 10(5): pp. 1299-

1319

[22] Arthur, D. and S. Vassilvitskii, k-means++: The advantages of careful seeding. 2006, Stanford

[23] Bellman, R., R. Kalaba, and L. Zadeh, Abstraction and pattern classification. Journal of Mathematical Analysis and Applications, 1966. 13(1): pp. 1-7

[24] Ruspini, E. H., A new approach to clustering. Information and Control, 1969. 15(1): pp. 22-32

[25] Fritzke, B., The k-means-u* algorithm: non-local jumps and greedy retries improve k-means++ clustering. CoRR, 2017. abs/1706.09059

[26] Ze-bao, Z. J. C. S., Algorithm for Initialization of K-Means Clustering Center Based on Optimized-Division. 2009

[27] Zhang, G., C. Zhang, and H. Zhang, Improved K-means algorithm based on density Canopy. Knowledge-based systems, 2018. 145: pp. 289-297

[28] Şenol, A., VIASCKDE Index: A Novel Internal Cluster Validity Index for Arbitrary-Shaped Clusters Based on the Kernel Density Estimation. Computational Intelligence and Neuroscience, 2022. 2022: p. 4059302

[29] Kools, J. 6 functions for generating artificial datasets. 2023 July 10, 2023]; Available from: https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets

[30] Dua, D. and C. Graff. UCI Machine Learning Repository. 2021; Available from: http://archive.ics.uci.edu/ml

[31] Gionis, A., H. Mannila, and P. Tsaparas, Clustering aggregation. ACM Trans. Knowl. Discov. Data, 2007. 1(1): p. 4-es

[32] Zelnik-Manor, L. and P. Perona, Self-tuning spectral clustering, in Proceedings of the 17th International Conference on Neural Information Processing Systems. 2004, MIT Press: Vancouver, British Columbia, Canada. pp. 1601-1608

[33] Parmar, M. Clustering Datasets. 2022 8.8.2022]; Available from: https://github.com/milaan9/Clustering-Datasets

[34] Veenman, C. J., M. J. T. Reinders, and E. Backer, A Maximum Variance Cluster Algorithm. IEEE Trans. Pattern Anal. Mach. Intell., 2002. 24: p. 1273-1280