A Kernel Clustering Algorithm Based on Diameters*

M. Fernanda P. Costa (✉)[1][0000−0001−6235−286X], Ana Maria A. C. Rocha[2][0000−0001−8679−2886], and Edite M. G. P. Fernandes[2][0000−0003−0722−9361]

[1] Centre of Mathematics,
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
mfc@math.uminho.pt
[2] ALGORITMI Center,
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
{arocha,emgpf}@dps.uminho.pt

Abstract. This paper analyzes an iterative kernel partitioning clustering algorithm that dynamically merges, removes and adds clusters using some characteristics, like the radii and diameters of the clusters, and distance between centers. The clustering is carried out in feature space in terms of a kernel function so that non-linearly separable clusters are identified. The preliminary experiments with seven datasets show that the proposed algorithm is able to successfully converge to the expected clustering. It is also shown that the algorithm performance is sensitive to the parameter σ of the Gaussian kernel.

Keywords: Partitioning Clustering · Kernel function · Cluster Diameter and Radius

1 Introduction

Clustering is an unsupervised machine learning task and consists of grouping a set of data points in a way that similarity of the elements in a group – also called cluster – is maximized, whereas similarity of elements in two different groups, is minimized [1,2]. Applications of clustering are very common in areas like data mining, bioinformatics, pattern recognition and image processing [3].

The partitioning and hierarchical clustering are the most popular. The well-known K-means clustering is a partitioning type clustering [4] that subdivides the dataset into K clusters, where K is specified *a priori*, and uses the centroid (mean) of the data points belonging to a cluster as the representative point of that cluster. Partitioning and hierarchical clustering are suitable to find convex clusters since they work well when clusters are compact and separated. However, they inaccurately identify clusters when non-convex clusters and noise points are

* This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020, UIDB/00013/2020 and UIDP/00013/2020 of CMAT-UM.

present. When a partitioning clustering algorithm cannot identify clusters that are non-linearly separable in the input space, the use of a kernel function solves the problem [5]. The goal is to use an appropriate nonlinear function that maps the points in the input space to points in some higher-dimensional feature space, known as the kernel space. Then, the kernel clustering method can partition the data points that are linearly separable in the new space [2].

Unlike K-means and pure hierarchical clustering, the partitioning clustering algorithm of this study is able to iteratively adjust the number of clusters. The algorithm may merge, remove and add clusters depending on some pre-defined conditions. During the initialization of the algorithm, a number of clusters is required to start the points assigning process, although this number is updated whenever clusters are merged, removed or split. The kernel clustering algorithm uses a distance-based methodology, a Gaussian kernel function to carry out the clustering in the feature space, and concepts like the cluster diameter, and the cluster radius, to select a cluster to be split, and to be merged, respectively. Thus, the algorithm proceeds as follows. First, at each iteration, the algorithm identifies the pair of clusters that may be merged and checks if a cluster center may be deleted, and consequently the cluster is removed. Second, it identifies the cluster that may be split according to conditions that involve the size of the two clusters with closest centers. Finally, each point in the dataset is assigned to the closest mean in feature space, resulting in a new clustering.

The paper is organized as follows. Section 2 briefly describes the ideas of a cluster analysis and Sect. 3 presents the details of the new kernel clustering diameter-based algorithm. In Sect. 4, the results of clustering seven sets of data points with two attributes are shown. Finally, Sect. 5 contains the conclusions of this work.

2 Cluster Analysis

Let a set of n data points (or vectors) of dimension a, the so-called attributes, be given. These points can be represented by a data matrix $\mathbf{X}$ with n vectors/points of dimension a. Each element $X_{i,j}$ corresponds to the jth attribute of the ith point [6]. Thus, given $\mathbf{X}$, a partitioning clustering algorithm tries to find a partition $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ of K clusters (or groups), in a way that similarity of the points in the same cluster is maximum and points from different clusters differ as much as possible. In hard clustering, it is required that the partition satisfies three conditions:

1. each cluster should have at least one point, i.e., $|C_k| \neq 0$, $k = 1, \ldots, K$;
2. a point should not belong to two different clusters, i.e., $C_k \bigcap C_l = \emptyset$, for $k, l = 1, \ldots, K$, $k \neq l$;
3. each point should belong to a cluster, i.e., $\sum_{k=1}^{K} |C_k| = n$;

where $|C_k|$ is the number of points in cluster C_k. Since there are several ways to partition the points and maintain these properties, a fitness function should be provided to evaluate the adequacy of the partitioning. Thus, the goal in the

clustering problem is to find an optimal partition, $\mathcal{C}^*$, that gives the optimal (or near-optimal) adequacy, when compared to all the other feasible solutions.

3 Kernel Clustering Diameter-based Algorithm

To partition data points into clusters, the distance-based clustering algorithms are the most popular, and are used in a large variety of applications [7]. In this section, a clustering approach that uses the distance between specific points and a kernel function is described.

The algorithm uses the kernel trick to carry out the clustering in feature space in terms of the Gaussian kernel function. Although the algorithm starts by defining a set of K clusters, it has mechanisms to try to find the optimal (or near-optimal) clustering by merging and removing clusters, and adding (splitting an existing cluster) based on some characteristics of the current clusters. In particular, the largest distance between points in a cluster (diameter), the smallest distance between cluster points, and the largest distance between the points in a cluster and the center point (radius), are considered. It is termed Kernel Clustering Diameter-based algorithm (KCDbased algorithm).

At the initial step, a number K of clusters must be provided and the corresponding centers $m_1, m_2, \ldots, m_K$ are randomly selected from the set of data points $\mathbf{X}$. Then, at each iteration, the KCDbased algorithm merges two featured clusters, removes a rather small cluster, adds a new cluster by splitting a specific cluster, according to some conditions, and ends up by assigning the dataset points to the existent clusters.

Next, the details of each procedure are presented.

After a set of points being randomly selected from $\mathbf{X}$, to define the initial cluster centers $m_k, k = 1, \ldots, K$ (given K), each point X_i $(i = 1, \ldots, n)$ is assigned to a cluster. To find which cluster to assign the point X_i, the simplest idea is to find the closest distance from X_i to the m_k, $k = 1, \ldots, K$.

In a kernel function context, the details to compute the distance from a point X_i to the centers m_k, $k = 1, \ldots, K$ follow. Let the cluster centers in the feature space be given as $m_k^\phi, k = 1, \ldots, K$ where

$$m_k^\phi = \frac{1}{|C_k|} \sum_{X_i \in C_k} \phi(X_i), \tag{1}$$

and ϕ is an appropriate nonlinear function that maps the points in the input space to points in the higher-dimensional feature space [5]. In this space, the distance of a point $\phi(X_i)$ to the mean m_k^ϕ is defined as

$$\|\phi(X_i) - m_k^\phi\|^2 = \|\phi(X_i)\|^2 - 2\phi(X_i)^T m_k^\phi + \|m_k^\phi\|^2$$

and consequently, using (1) and the kernel function $\mathbf{K}(\cdot, \cdot) = \phi(\cdot)^T \phi(\cdot)$:

$$\|\phi(X_i) - m_k^\phi\|^2 = \mathbf{K}(X_i, X_i) - \frac{2}{|C_k|} \sum_{X_j \in C_k} \mathbf{K}(X_j, X_i) + \frac{1}{|C_k|^2} \sum_{X_j \in C_k} \sum_{X_l \in C_k} \mathbf{K}(X_j, X_l). \tag{2}$$

Algorithm 1 presents the details of the KCDbased algorithm.

Algorithm 1 KCDbased Algorithm

Require: n (number of dataset points), $X_i, i = 1, \ldots, n$ (points of the dataset); $It_{\max}$
1: Set initial K and $N_{\min}$; Set $It = 1$
2: Randomly select a set of K points from the dataset $\mathbf{X}$ to initialize the K cluster centers m_k, $k = 1, \ldots, K$
3: Assign each point $X_i \in \mathbf{X}$ to the cluster based on the closest center, using Algorithm 2, to define K clusters $\mathcal{C}_K = \{C_1, \ldots, C_K\}$
4: **repeat**
5: Compute the distance between the two closest cluster centers, $D_{\min}$, using (7) and identify the indices k_i and k_j of the centers
6: Compute the largest distance of each center $m_{k_i}^{\phi}$ (resp. $m_{k_j}^{\phi}$) to the points in cluster C_{k_i} (resp. C_{k_j}), r_{k_i} (resp. r_{k_j})
7: Compute the average distance of each center $m_{k_i}^{\phi}$ (resp. $m_{k_j}^{\phi}$) to the points in cluster C_{k_i} (resp. C_{k_j}), $\bar{r}_{k_i}$ (resp. $\bar{r}_{k_j}$)
8: Merge the 2 clusters with closest cluster centers (based on $D_{\min}$, r_{k_i}, r_{k_j}, $\bar{r}_{k_i}$ and $\bar{r}_{k_j}$) and remove the cluster with fewer points (based on $N_{\min}$), using Algorithm 3
9: **for** $k = 1$ to K **do**
10: Compute distances $d_{j,l} = \|\phi(X_j) - \phi(X_l)\|^2$ for all $X_j, X_l \in C_k$ using (6)
11: Identify diameter $D_1^k = \max d_{j,l}$ and the minimum distance $D_0^k = \min d_{j,l}$
12: **end for**
13: Identify the cluster C_{k_i}, where the index $k_i = \arg\max_k(D_1^k - D_0^k)$, and the points $X_{i_1}, X_{i_2} \in C_{k_i}$ corresponding to $D_1^{k_i}$
14: Split the cluster C_{k_i} using Algorithm 4
15: Assign points to the cluster with closest center, using Algorithm 2
16: Set $It = It + 1$
17: **until** $It > It_{\max}$ OR number of points in clusters has not changed for two consecutive iterations
18: **return** K^* and $\mathcal{C}^* = \{C_1^*, \ldots, C_K^*\}$.

3.1 Assign Points to Clusters

To assign point X_i to the closest cluster center, the following condition is set:

$$\begin{aligned} X_i \in C_{k_i} \text{ such that } k_i &= \arg\min_{k=1,\ldots,K} \left(\|\phi(X_i) - m_k^{\phi}\|^2 \right) \\ &= \arg\min_{k=1,\ldots,K} \left(SqN_k - 2Avg_{i,k} \right) \end{aligned} \tag{3}$$

where the SqN_k (squared norm of cluster means) for each $k = 1, \ldots, K$ is given by:

$$SqN_k = \frac{1}{|C_k|^2} \sum_{X_j \in C_k} \sum_{X_l \in C_k} \mathbf{K}(X_j, X_l), \tag{4}$$

and the average kernel value for X_i and C_k, $Avg_{i,k}$ ($i = 1, \ldots, n$ and $k = 1, \ldots, K$), is

$$Avg_{i,k} = \frac{1}{|C_k|} \sum_{X_j \in C_k} \mathbf{K}(X_j, X_i). \tag{5}$$

We note that in the last expression of (3), the term $\mathbf{K}(X_i, X_i)$ has not been included because it is the same for all the clusters.

Subsequently, the distance of a point $\phi(X_j)$ to another point $\phi(X_l)$ of the dataset in feature space, can be computed as

$$\begin{aligned} d_{j,l} = \|\phi(X_j) - \phi(X_l)\|^2 &= \|\phi(X_j)\|^2 - 2\phi(X_j)^T\phi(X_l) + \|\phi(X_l)\|^2 \\ &= \mathbf{K}(X_j, X_j) - 2\mathbf{K}(X_j, X_l) + \mathbf{K}(X_l, X_l). \end{aligned} \tag{6}$$

The Algorithm 2 contains the steps required to the process of assigning points to clusters.

Algorithm 2 Assign points to clusters

Require: K, n, $X_i, i = 1, \ldots, n$, $C_k, k = 1, \ldots, K$
for $k = 1$ to K **do**
  Compute squared norm of cluster means SqN_k using (4)
end for
for all X_i, $i = 1, \ldots, n$ **do**
  for all C_k, $k = 1$ to K **do**
    Compute average kernel value $Avg_{i,k}$ using (5)
  end for
end for
Set $C_k = \emptyset, k = 1, \ldots, K$
for $i = 1$ to n **do**
  for $k = 1$ to K **do**
    Compute $d_{i,k} = SqN_k - 2Avg_{i,k}$
  end for
  Assign point X_i to cluster C_{k_i} by identifying the index $k_i = \arg\min_{k=1,\ldots,K} d_{i,k}$
end for
for $k = 1$ to K **do**
  if $|C_k| = 0$ **then**
    Remove C_k
  end if
end for
Update K
return C_k, $k = 1, \ldots, K$

3.2 Merge Two Clusters

The two clusters with the closest centers are merged if the distance between their centers satisfies some conditions based on their radii. Let the radius of a cluster be defined by the largest distance from its center to the points of the cluster, r_k, $k = 1, \ldots, K$. The average of all the distances from the center to the points of a cluster is denoted averaged radius and is represented by $\bar{r}_k$. The two clusters with the closest centers may be merged if the distance between their centers, $D_{\min}$, is not greater than the sum of their radii and at the same time less than the average of their averaged radii multiplied by a positive factor ≥ 1,

where $D_{\min}$ is computed using

$$\begin{aligned} D_{\min} &= \min_{i=1,\ldots,K, j>i} \left(\|m_i^\phi - m_j^\phi\|^2 \right) \\ &= \min_{i=1,\ldots,K, j>i} \left(\|m_i^\phi\|^2 - 2\bar{m}_{i,j} + \|m_j^\phi\|^2 \right) \\ &= \min_{i=1,\ldots,K, j>i} \mathbf{K}(m_i^\phi, m_i^\phi) - 2\bar{m}_{i,j} + \mathbf{K}(m_j^\phi, m_j^\phi), \end{aligned} \tag{7}$$

and $\bar{m}_{i,j}$, the average kernel value for C_i and C_j, is given by

$$\bar{m}_{i,j} = \frac{1}{|C_i||C_j|} \sum_{i=1,\ldots,k} \sum_{j>i} \mathbf{K}(m_i^\phi, m_j^\phi).$$

See details of the merge and remove clusters algorithm in Algorithm 3.

Algorithm 3 Merge and remove clusters

Require: n, K, C_{k_i}, C_{k_j}, $D_{\min}$, $\bar{r}_{k_i}$, $\bar{r}_{k_j}$, r_{k_i}, r_{k_j}, $N_{\min}$

if $D_{\min} \leq (r_{k_i} + r_{k_j})$ AND $D_{\min} < \dfrac{\bar{r}_{k_i} + \bar{r}_{k_j}}{2} \dfrac{n}{|C_{k_i}| + |C_{k_j}|}$ **then**
  Include points of C_{k_i} into cluster C_{k_j} and remove C_{k_i}
  Update K and $|C_{k_j}|$; set $N_0 = |C_{k_j}|$ and $N_1 = 0$
else
  Set $N_0 = |C_{k_j}|$ and $N_1 = |C_{k_i}|$
end if
Identify the index k_l such that $|C_{k_l}| = \min_k |C_k|$
if $|C_{k_l}| < N_{\min}$ **then**
  Define all points in C_{k_l} as 'noise/outlier'
  Remove C_{k_l}
  Update K
end if
return C_k, $k = 1, \ldots, K$, N_0, N_1.

3.3 Split a Cluster

Based on the diameter of a cluster, defined as the largest distance between points in that cluster, and on the minimum distance between points of the cluster, the algorithm may split the cluster into two clusters. Let D_1 be the diameter of a cluster and D_0 be the minimum distance between points of the cluster. The candidate cluster to be split has the largest difference $D_1 - D_0$. We choose to split that cluster if its diameter exceeds the (previously identified) smallest distance between centers and if its size (number of points in that cluster) exceeds the size of the cluster when two clusters are merged or the size of the largest cluster previously identified for the merging process. Algorithm 4 describes the main steps of the splitting process.

Algorithm 4 Split a cluster

Require: K, k_i, C_{k_i}, $D_1^{k_i}$, $X_{i_1}, X_{i_2} \in C_{k_i}$, $D_{\min}$, N_0, N_1
1: **if** $D_1^{k_i} > D_{\min}$ AND $|C_{k_i}| > \max(N_0, N_1)$ **then**
2: Assign points $X_i \in C_{k_i}$ to the new clusters with the selected centers X_{i_1} and X_{i_2} to define clusters C_{K+1} and C_{K+2}
3: **end if**
4: Set $C_{k_i} \leftarrow C_{K+2}$; $K \leftarrow K+1$
5: **return** C_{k_i} and C_K

4 Computational Results

In this section, some preliminary results are shown. Seven datasets of points with two attributes are used to test the algorithm.

The Algorithm 1 is coded in MATLAB®. Its performance depends on the parameter of the Gaussian kernel, σ, where for U and V vectors,

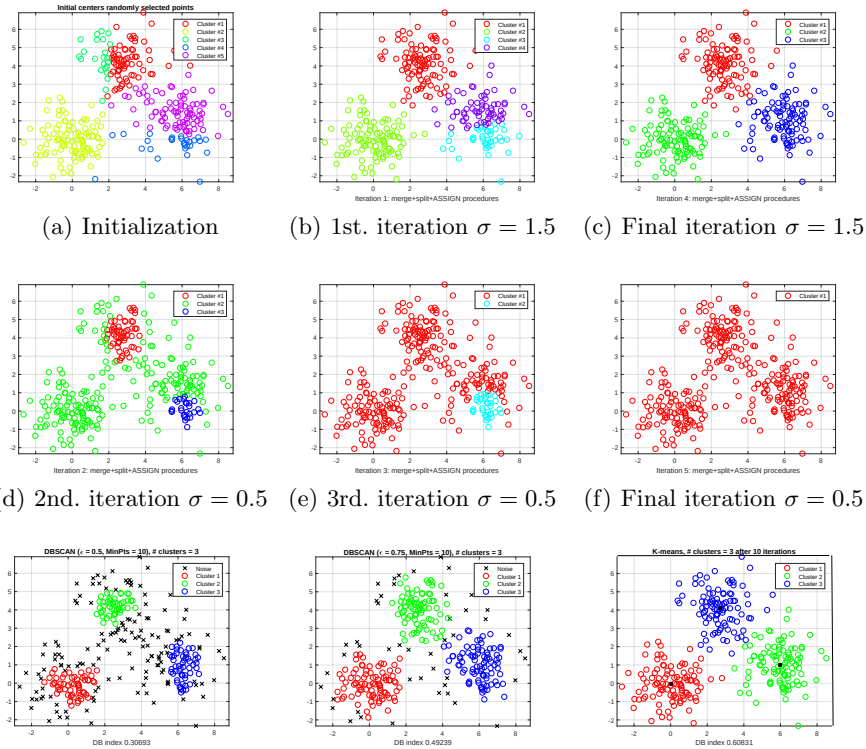
$$\mathbf{K}(U,V) = \exp(-\gamma\|U-V\|_2^2) \text{ and } \gamma = (2\sigma^2)^{-1}.$$

Note that this kernel value $\mathbf{K}$ is inversely related to the distance between the two vectors. In practice, different values for σ are usually set by trial and error. Other parameters that have not much influence in the algorithm performance are initial $K = \max\{v, \min\{2, \lceil 0.005n \rceil\}\}$, where v is a positive integer in $[2, 6]$, $N_{\min} = \max\{5, \lceil 0.01n \rceil\}$ and $It_{\max} = 10$. The algorithm is assumed to converge when the cluster points do not change for two consecutive iterations.

The clustering obtained in this study is compared with those obtained by two well-known partitioning algorithms, DBSCAN and K-means. DBSCAN is a density-based clustering algorithm that is designed to discover clusters and noise in data [8]. Noise points are outliers that do not belong to any cluster. There are two parameters that must be provided before the algorithm stars, ϵ and $MinPts$. A point is assigned to a cluster if it satisfies the condition that its ϵ neighborhood contains at least a minimum number of neighbors, $MinPts$. The distance metric used to measure the similarity between observations is the Euclidean distance (default). K-means is an iterative partitioning algorithm that assigns the n data points to one of K clusters, where K must be provided at the beginning of the algorithm [9]. Each cluster is represented by the centroid and the clustering aims to minimize the sum of squared distances of the points to the centroids. The default distance metric for the minimization is the Euclidean distance. The maximum number of iterations, $MaxIter$ is also provided to the algorithm.

The first two datasets of points are available in the internet. The others are randomly generated datasets. In Figs. 1 - 7 below, the results obtained for Problems 1 - 7 are depicted.

Problem 1. 'mydata'. 300 data points and two attributes:
available at https://yarpiz.com/64/ypml101-evolutionary-clustering [10].



(a) Initialization (b) 1st. iteration $\sigma = 1.5$ (c) Final iteration $\sigma = 1.5$

(d) 2nd. iteration $\sigma = 0.5$ (e) 3rd. iteration $\sigma = 0.5$ (f) Final iteration $\sigma = 0.5$

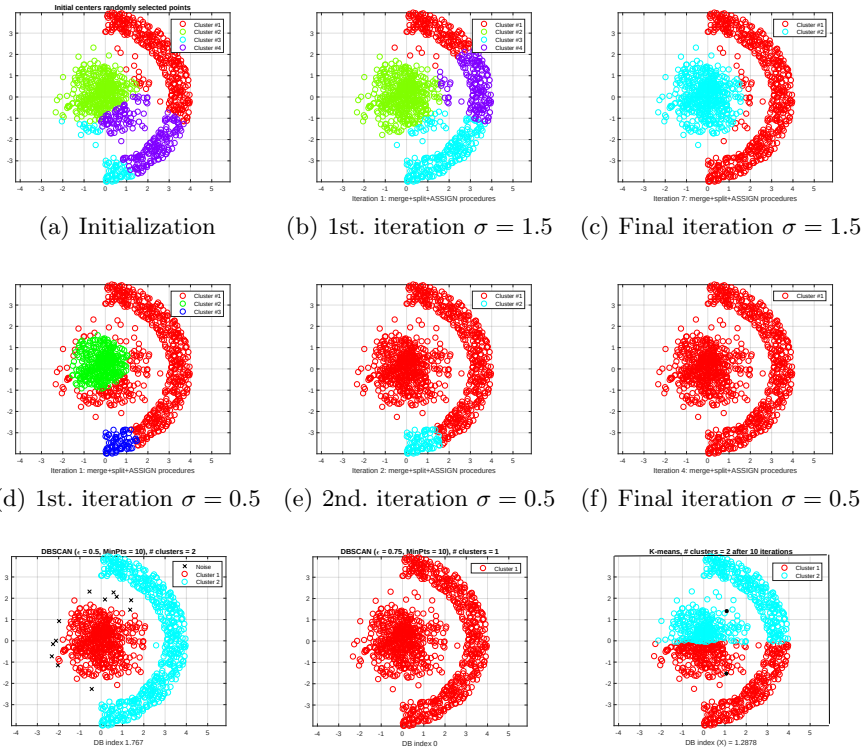
(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.75$ (i) K-means (10 iterations)

Fig. 1. Clustering process of Problem 1. Plots (a), (b) and (c) are from our algorithm with $\sigma = 1.5$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 0.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

Algorithm 1 was tested with Problem 1 starting with 5 clusters (see Fig. 1). Plot (a) shows the initial clustering with the 5 clusters. Plots (b) and (c) show the obtained well succeeded clustering (1st. iteration and 4th. iteration respectively) using $\sigma = 1.5$ in the Gaussian kernel. However, when $\sigma = 0.5$ is used, the iterative process starts with the clustering shown in plot (a) - the initialization - and converges to one cluster (see plots (d), (e) and (f)). Plots (g) and (h) show the DBSCAN clustering when the parameter ϵ is set to 0.5 and 0.75, respectively, with $MinPts = 10$ (for best clusterings) and both clustering identify 3 groups, but with plenty of noise points. Finally, plot (i) shows the K-means successful clustering when $K = 3$ is provided.

Problem 2. 'mydataDBSCAN'. 1000 data points and two attributes: available at https://yarpiz.com/255/ypml110-dbscan-clustering [11].

With Problem 2, the two values (1.5 and 0.5) of σ were tested, and the results are depicted in Fig. 2. Starting with 4 clusters, plots (b) and (c) show a



(a) Initialization (b) 1st. iteration $\sigma = 1.5$ (c) Final iteration $\sigma = 1.5$

(d) 1st. iteration $\sigma = 0.5$ (e) 2nd. iteration $\sigma = 0.5$ (f) Final iteration $\sigma = 0.5$

(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.75$ (i) K-means (10 iterations)

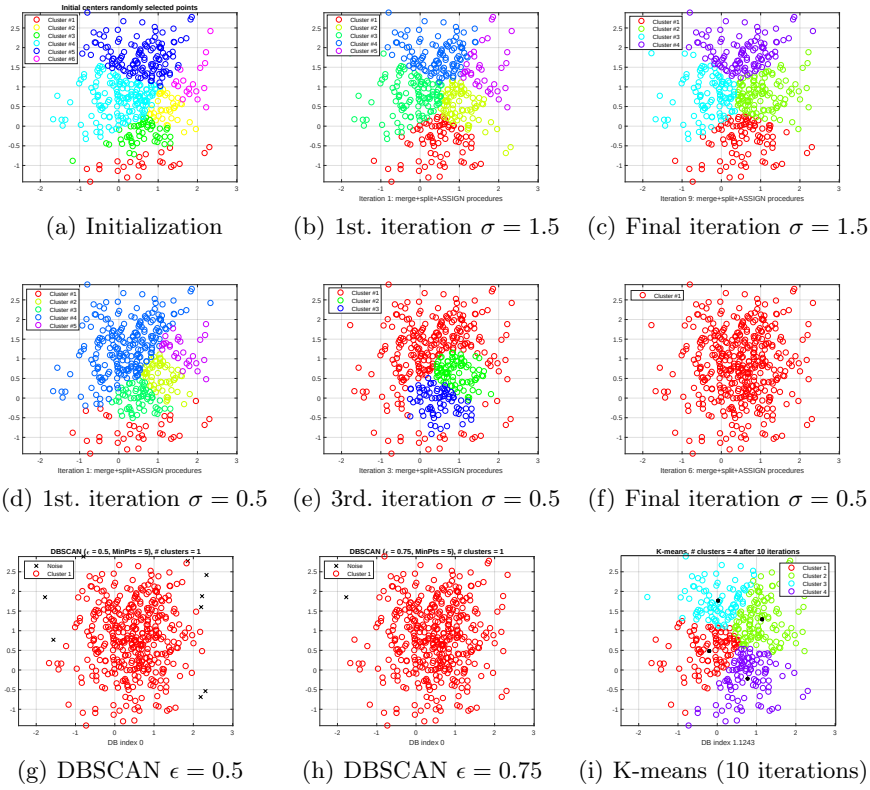
Fig. 2. Clustering process of Problem 2. Plots (a), (b) and (c) are from our algorithm with $\sigma = 1.5$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 0.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

successful clustering of Algorithm 1, while plots (d), (e) and (f) (starting with the same 4 clusters) display an unsuccessful clustering. Plots (g) and (h), with $\epsilon = 0.5$ and $\epsilon = 0.75$ respectively (with $MinPts = 10$), show a successful and an unsuccessful clustering of DBSCAN. As expected, K-means clustering (with provided $K = 2$) is unable to identify non-linearly separable clusters.

Problem 3. 400 data points with two attributes: (Four clusters with generated points using multivariate Normal distributions with equal variances.)

```
Sigma = [0.5 0.05; 0.05 0.5];  f1 = mvnrnd([0.5 0],Sigma,100);
f2 = mvnrnd([0.5 0.5],Sigma,100); f3 = mvnrnd([0.5 1],Sigma,100);
f4 = mvnrnd([0.5 1.5],Sigma,100); X = [f1;f2;f3;f4];
```

When solving Problem 3, and using both $\sigma = 1.5$ and $\sigma = 0.5$, the performance of the KCDbased algorithm, shown in Fig. 3, is similar to the two previous examples. When $\sigma = 1.5$, and starting from the initial clustering (shown in plot



(a) Initialization (b) 1st. iteration $\sigma = 1.5$ (c) Final iteration $\sigma = 1.5$

(d) 1st. iteration $\sigma = 0.5$ (e) 3rd. iteration $\sigma = 0.5$ (f) Final iteration $\sigma = 0.5$

(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.75$ (i) K-means (10 iterations)

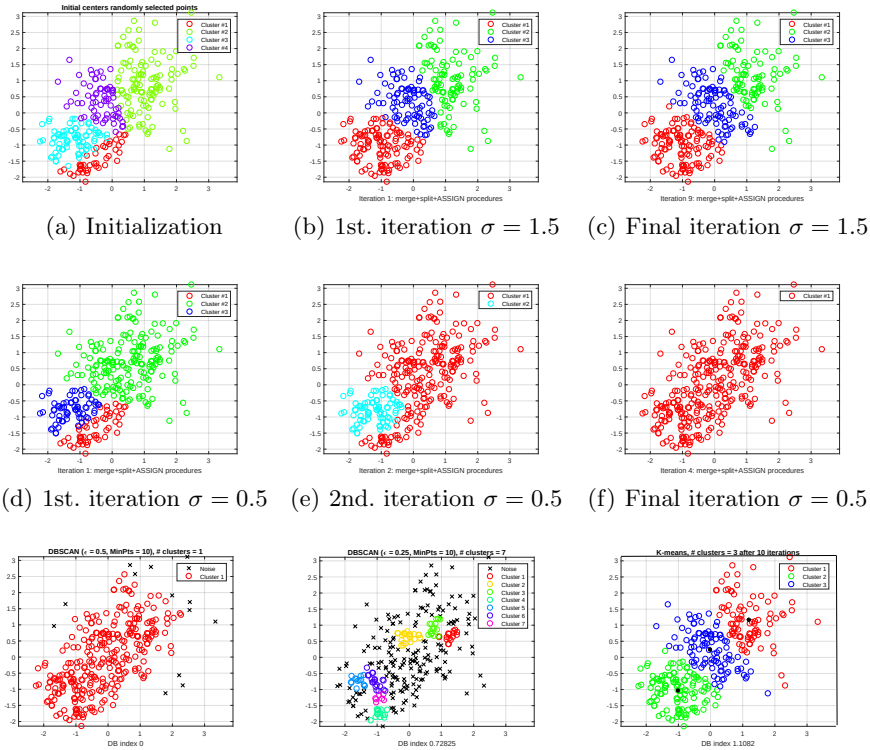
Fig. 3. Clustering process of Problem 3. Plots (a), (b) and (c) are from our algorithm with $\sigma = 1.5$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 0.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

(a)), the algorithm is able to identify 4 clusters (plots (b) and (c)). The 4 clusters are not identified by DBSCAN when ϵ is set to 0.5 and 0.75, using $MinPts = 5$. The same is true when $MinPts = 10$ is used. Setting $K = 4$, K-means identifies 4 linearly separable clusters.

Problem 4. 300 data points with two attributes: (Three clusters with generated points using Uniform distributions.)

```
f1 = randn(100,2)*0.75+ones(100,2); f2 = randn(100,2)*0.5-ones(100,2);
f3 = randn(100,2)*0.75; X = [f1;f2;f3];
```

To cluster the dataset in Problem 4, $\sigma = 1.5$ and $\sigma = 0.5$ are also used (see Fig. 4). The performance of the KCDbased algorithm is similar to the previous examples. We note that DBSCAN identifies one cluster with $\epsilon = 0.5$ and $MinPts = 10$, and only for $\epsilon = 0.25$ is able to identify more than one cluster but with plenty of noise points (see plots (g) and (h)). K-means converges by linearly separating 3 clusters (plot (i)).



(a) Initialization (b) 1st. iteration $\sigma = 1.5$ (c) Final iteration $\sigma = 1.5$

(d) 1st. iteration $\sigma = 0.5$ (e) 2nd. iteration $\sigma = 0.5$ (f) Final iteration $\sigma = 0.5$

(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.25$ (i) K-means (10 iterations)

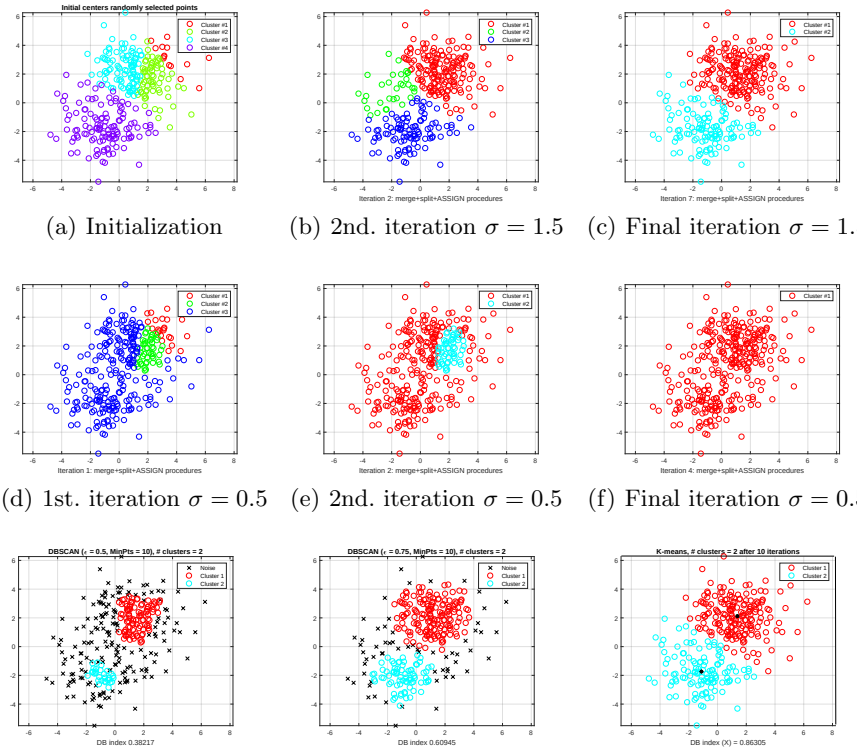
Fig. 4. Clustering process of Problem 4. Plots (a), (b) and (c) are from our algorithm with $\sigma = 1.5$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 0.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

Problem 5. 300 data points with two attributes: (Two clusters with generated points using multivariate Normal distributions.)

```
mu1 = [1 2]; sigma1 = [3 .2; .2 2]; mu2 = [-1 -2]; sigma2 = [2 0; 0 1];
f1 = mvnrnd(mu1,sigma1,200); f2 = mvnrnd(mu2,sigma2,100); X = [f1;f2];
```

To cluster the set of points in Problem 5, $\sigma = 1.5$ and $\sigma = 0.5$ are also used. The initial clustering (plot(a)), based on 4 clusters, is the same for both experiments. The performance of the KCDbased algorithm is similar to the previous examples, as shown in Fig. 5. With the DBSCAN, $MinPts$ is set to 10 to show the best clustering found (plot (g) for $\epsilon = 0.5$ and plot (h) for $\epsilon = 0.75$). K-means linearly separates 2 clusters (plot (i)).

Problem 6. 200 data points and two attributes: (Two clusters with generated points giving two moons, distorted with some added noises.)



(a) Initialization (b) 2nd. iteration $\sigma = 1.5$ (c) Final iteration $\sigma = 1.5$

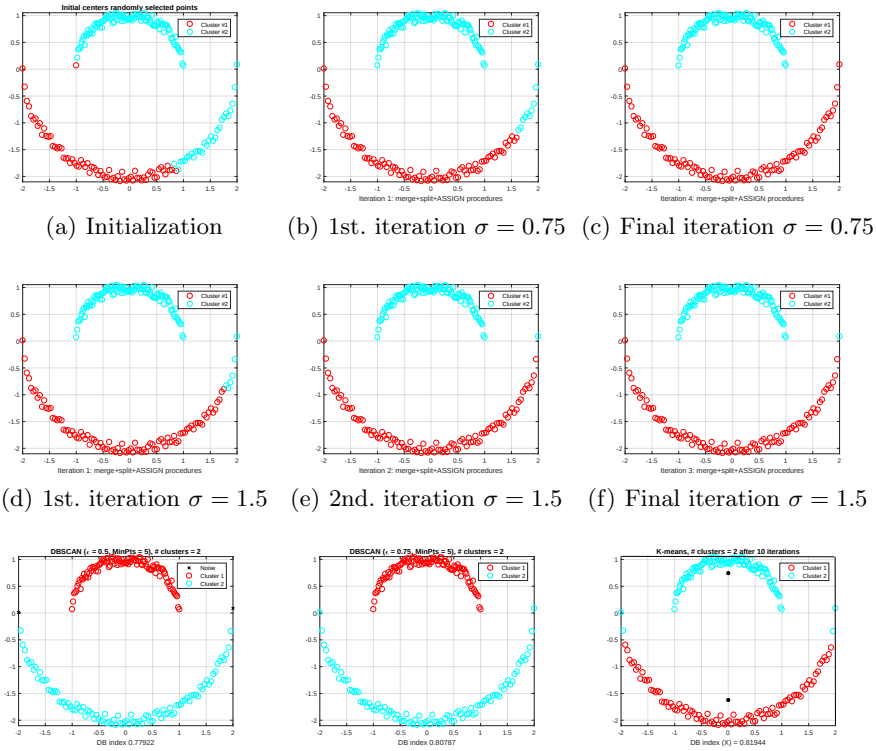
(d) 1st. iteration $\sigma = 0.5$ (e) 2nd. iteration $\sigma = 0.5$ (f) Final iteration $\sigma = 0.5$

(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.75$ (i) K-means (10 iterations)

Fig. 5. Clustering process of Problem 5. Plots (a), (b) and (c) are from our algorithm with $\sigma = 1.5$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 0.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

```
for i=1:100
   f1(i) = (-1+2*(i-1)/99, sqrt(1-(-1+2*(i-1)/99)^2)+unifrnd(-0.1,0.1));
   f2(i) = (-2+4*(i-1)/99, -sqrt(4-(-2+4*(i-1)/99)^2)+unifrnd(-0.1,0.1));
end
X = [f1;f2];
```

The data points in Problem 6 constitute 2 non-convex clusters. The clustering obtained by the Algorithm 1 when $\sigma = 0.75$ successfully identifies the 2 clusters (plots (a), (b) and (c)) of Fig. 6. However, when $\sigma = 1.5$, there is a point that is not assigned to the corresponding cluster when the algorithm stops (see plots (d), (e) and (f)). DBSCAN successfully identifies the 2 clusters (setting $MinPts$ to 5), although for $\epsilon = 0.5$ defines 2 noise points (see plots (g) and (h)). The K-means clustering (plot (i)), with provided $K = 2$, identifies 2 clusters as if they were linearly separable.



(a) Initialization (b) 1st. iteration $\sigma = 0.75$ (c) Final iteration $\sigma = 0.75$

(d) 1st. iteration $\sigma = 1.5$ (e) 2nd. iteration $\sigma = 1.5$ (f) Final iteration $\sigma = 1.5$

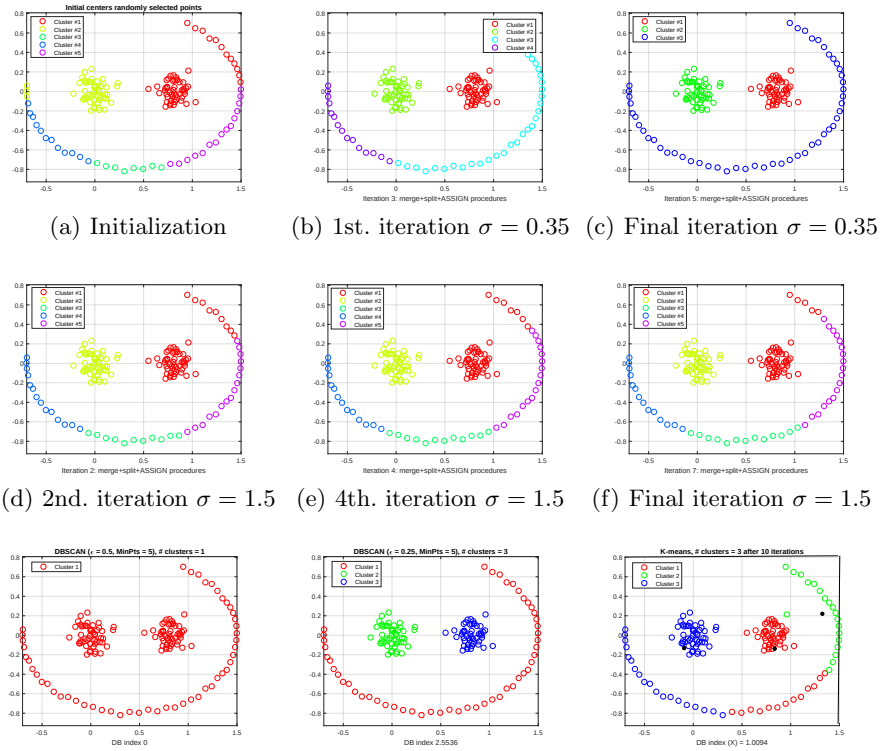
(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.75$ (i) K-means (10 iterations)

Fig. 6. Clustering process of Problem 6. Plots (a), (b) and (c) are from our algorithm with $\sigma = 0.75$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 1.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

Problem 7. 150 data points and two attributes: (Three clusters with two spherically shaped clusters inside 3/4 of a perturbed circle.)

```
for i=1:50
  f1(i) = (0.4+1.1*sin(2*pi*(30+5*(i-1))/360),
           0.8*cos(2*pi*(30+5*(i-1))/360)+unifrnd(-0.025,0.025));
end
Sigma = [0.01 0; 0 0.01]; mu1 = [0 0]; mu2 = [0.8 0];
f2 = mvnrnd(mu1,Sigma,50); f3 = mvnrnd(mu2,Sigma,50); X = [f1;f2;f3];
```

The data points in Problem 7 define non-linearly separable clusters. A successful clustering is obtained by the KCDbased algorithm with a small value of σ (0.35), starting with an initial clustering of 5 groups, as shown in plots (a), (b) and (c) of Fig. 7. When σ is set to a larger value (1.5), the clustering process evolves very slowly and the algorithm stops with still 5 clusters - plots (d), (e) and (f). DBSCAN successfully identifies the 3 clusters when ϵ is set to 0.25 (plot



(a) Initialization (b) 1st. iteration $\sigma = 0.35$ (c) Final iteration $\sigma = 0.35$

(d) 2nd. iteration $\sigma = 1.5$ (e) 4th. iteration $\sigma = 1.5$ (f) Final iteration $\sigma = 1.5$

(g) DBSCAN $\epsilon = 0.5$ (h) DBSCAN $\epsilon = 0.25$ (i) K-means (10 iterations)

Fig. 7. Clustering process of Problem 7. Plots (a), (b) and (c) are from our algorithm with $\sigma = 0.35$ in the Gaussian kernel, (d), (e) and (f) are from our algorithm with $\sigma = 1.5$, (g) and (h) come from DBSCAN, and (i) is from K-means.

(h)), but converges to just one cluster when $\epsilon = 0.5$ (plot (g)). The $MinPts$ was set to 5 so that the successful clustering in plot (h) is obtained. As expected, K-means (setting $K = 3$) converges to 3 clusters assuming that they are linearly separable.

5 Conclusions

A kernel-based partitioning clustering algorithm is presented to address the issue related to the identification of non-linearly separable clusters. The Kernel Clustering Diameter-based algorithm iteratively merges clusters, based on the radii and average radii of the two clusters with the closest centers, and removes a cluster that has a very small number of points. The algorithm is also able to split a cluster that has the largest difference between its diameter and the smallest distance between its cluster points. To activate the splitting process, the cluster diameter and the number of points in the cluster have to satisfy some conditions.

In the kernel sense, assigning points to clusters involves minimizing the sum of the squared distances of the points to the centers in the feature/kernel space. The Gaussian kernel is used to compute the squared distances of the points to the centers, between centers, and between points of a cluster.

The experiments with seven datasets show that the performance of the KCD-based algorithm is sensitive to the value of the parameter σ, in the Gaussian kernel. A large value gives a successful clustering for datasets with clusters not well separated, and a small value is better when the points in the dataset form well separated clusters. Furthermore, larger values of σ imply larger Gaussian kernel values.

The experiments with this algorithm will be extended to higher dimensional datasets. Furthermore, as the iterations proceed, the smallest distance between points of two clusters will be used to identify well separated clusters and select an adequate value for the parameter σ.

Acknowledgments. The authors wish to thank the three anonymous referees for their comments and suggestions to improve the paper.

References

1. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Ann. Data Sci. **2**(2), 165–193 (2015)
2. Mohammed, J.Z., Meira Jr., W.: Data Mining and Machine Learning: Fundamental Concepts and Algorithms, 2nd Edition, Cambridge University Press (2020)
3. Greenlaw, R., Kantabutra, S.: Survey of clustering: algorithms and applications. Int. J. Inf. Retr. Res. **3**(2), 29 pages (2013)
4. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In L.M. Le Cam, J. Neyman (Eds.), Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, vol. 1, 281–297 (1967)
5. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**, 1299–1319 (1998)
6. Jain A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
7. Xu, R., Wunsch II, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
8. Ester, M., Kriegel, H.-P., Sander, J., Xiaowei, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining, 226–231. Portland, OR: AAAI Press, (1996)
9. Lloyd, S. P.: Least Squares Quantization in PCM. IEEE Trans. Inf. Theory **28**, 129–137 (1982)
10. Heris, M.K.: Evolutionary Data Clustering in MATLAB. https://yarpiz.com/64/ypml101-evolutionary-clustering, Yarpiz, (2015)
11. Heris, M.K.: DBSCAN Clustering in MATLAB. https://yarpiz.com/255/ypml110-dbscan-clustering, Yarpiz, (2015)