Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 13 (2018) pp. 137-148

Dynamic Algorithm based on split and merge for Data Streams Clustering

Chedi Ounali[1], Fahmi Ben Rejab[2] and Kaouther Nouira[3]

[1]Université de Tunis, ISGT,
LR99ES04 BESTMOD, Le Bardo Tunisia
chedy.ounelly@gmail.com

[2]Université de Tunis, ISGT,
LR99ES04 BESTMOD, Le Bardo Tunisia
fahmi.benrejab@gmail.com

[3]Université de Tunis, ISGT,
LR99ES04 BESTMOD, Le Bardo Tunisia
kaouther.nouira@planet.tn

Abstract: **Clustering is a widely used technique. It is a discipline that aims to reveal groups, or clusters of similar entities in data, but data in our case are large and continuously generated. In addition, we have to deal with the change at the level of clusters number that can be extended or reduced at every moment. We present a new technique allowing the merge and split of clusters while receiving streaming data in order to create stable clusters without losing information or retraining from scratch.**
Keywords: Clustering; Clusters; Data stream; K-Means; Dynamic clustering; Split; Merge.

I. Introduction

Clustering is an unsupervised learning method as it classifies data-sets without any a prior knowledge. It has been used in different fields such as bio-informatics, image processing, genetics, speech recognition, market research, document classification, anomalies detection and weather classification [1].
There are various algorithms for the data clustering. The k-means algorithm is one of the most popular, it is very simple in operation, suitable for unraveling compact clusters and a fast iterative algorithm[9]. k-means algorithm divides N elements from data-set for k clusters that used center-based clustering methods [9]. Consequently, the main challenge for these clustering methods is in determining the number of clusters[1]. In general, the number of clusters has been set by users or archives from knowledge of research[10]. But a bad choice of the number of cluster can lead to a wrong distribution of observations. Thats how the term adaptive clustering was born. Their for incremental remains to observations and to the most adequate cluster to the element [11]. In our previous paper Incremental k-means based on split technique [26], we proposed a new version of k-means to detect the cluster to be split. However, it does not cover the issue presented by Big Data which is processing real-time data or to deal with the shrink of clusters .
In this research, we propose a dynamic clustering algorithm based on k-means to deal the with incoming data at real time. In addition, it allows the update of clusters distribution without retraining from scratch by going from k cluster to $k+1$ clusters or $k-1$ clusters based statical and mathematical tools. The rest of our paper is organized as follow: Section 2 contains the standard version of K-Means. Section 3 deals with the notion of dynamic clustering. Section 4 presents the proposed approach. Section 5 contains the results of experimentation, and the last section presents the conclusion.

II. K-Means algorithm

The k-means algorithm is a partitional clustering method proposed in 1967 by MacQueen [27], it can be described by the following steps [2].

1. Choose initial centroids $m_{1..k}$ of the clusters $C_{1..k}$
2. Calculate new cluster membership. A feature vector x_j is assigned to the cluster C_i if and only if :
$$I = arg_{k=1...k} min||x_j - m_k||^2 \quad (1)$$
3. Recalculate centroids for the cluster according.
$$m_i = \frac{1}{|c_i|} \sum x_j \quad (2)$$
4. If none of the cluster centroids have changed, finish the algorithm. Otherwise go to Step (2).

The standard k-means always retrain from scratch if their is a modification in the data-set which causes a big loss of time, thats why the use of incremental clustering has been improved last years.

A. Distance measures

There are different methods that the algorithm K-means uses for distance measure between these distance measures there are:

1. **Euclidean distance and squared Euclidean distance**, are generally calculated from row data and not from standardized data. The advantage of the Euclidean distance is that the addition of a new element cannot influence the measure of distance between two other elements, it is calculated as follow [3]:

$$D.E(a,b) = \sqrt{\sum_{i=1}^{k}(a_i - b_i)^2} \quad (3)$$

2. **Manhattan distance**, consider that the shortest path between two points in the xy-plain is the hypotenuse which refer to the Euclidean distance, the Manhattan distance measure will be like [4]:

$$D.E(a,b) = \sum_{i=1}^{k}|a_i - b_i| \quad (4)$$

3. **The Jacquard distance**, is a metric measure that inform how dissimilar tow sets are. It represents a complement to the Jacquard index, and it is obtained by subtracting the Jaccard coefficient from one. The Jaccard distance is represented as follows [5]:

$$D.J(a,b) = \frac{\sum_{i=1}^{n}(a_i - b_i)^2}{\sum_{i=1}^{n}(a_i^2 + \sum_{i=1}^{n}(b_i^2 - \sum_{i=1}^{n} a_i b_i)} \quad (5)$$

K-means cant deal with incremental datasets. Whenever new elements added, the algorithm needs to retrain from scratch which causes loss of information and time where time is such challenge for this kind of algorithms. So to deal with this problem incremental k-meas was proposed.

III. Dynamic clustering

A large area of research in clustering has focused on improving the clustering process such that the clusters are not dependent on the initial identification of cluster representation. There are versions of adaptive clustering that allows the regeneration of clustering procedure from scratch to response to the change of elements but those techniques can produce large deference in term of the size of clusters and a huge waste of time[12]. To avoid the regeneration from scratch, the split technique was induced in clustering [13]. It was used to create an incremental clustering procedure, but the problem was always which cluster should split. The following approaches are typically used for the selection of the cluster [14][15]:

1. Complete partition: every cluster is split, so obtaining a complete binary tree.
2. The cluster having the largest number of elements is split.
3. The cluster with the highest variance.
4. The shape of the cluster.

The above criterion are extremely simple. The first criteria split every cluster that provide a complete tree, but it completely ignores the issue of the quality of the clusters. The second one is also very simple: it does not provide a complete tree, but it has the advantage of yielding a balanced tree, where the leaves have approximately the same size. The tow last criteria is the most sophisticated in relation with the tow previous, since it is based upon a simple but meaningful property of a cluster. This is the reason why highest variance criteria is the most commonly used criterion for cluster selection.
Adding to those there are other version of incremental clustering. Qiu et al [6] represented a clustering boundary detection method by the transformation of affine space, this method where argued by Tong et al [7] by claiming that boundary points are essential for clustering due to their representation of the distribution of the dataset. In literature we found that all works either search for a new manner to create a new algorithm that allows the incremental in the clustering level or used the split technique but with a simple criterion in the phase of the choice of the cluster to split. One of the most known algorithm based on k-means and uses the split technique to create new clusters is Bisecting k-means [8]; but bisecting k-means creates a complete tree from the k-cluster that given at the beginning. In our case we are searching for the cluster that had the most spreading out elements and none of these algorithm can respond to the problem. However, dynamic clustering aims to group data being accumulated and updates clusters based on the last clustering result. The strategy used by incremental k-means optimizes the clustering process and reduces the time needed to get final results, where time is a critical factor for the usability of some application that uses that type of algorithm [18].
Dynamic clustering as a form of unsupervised online/incremental machine learning algorithm considers two concepts [19]:

1. Incrementality of the learning methods to divide the clustering model.
2. Self-adaptation of the learned model (parameters and structure).

Works that involves Dynamic clustering will be presented next:

1. Dynamic Clustering of Data with Modified K-Means Algorithm [28], it is very difficult to fix the number of clusters in advance. The proposed method deals with both the cases, for known number of clusters in advance as well as unknown number of clusters. The user has the flexibility either to fix the number of clusters or input the minimum number of clusters required. In the former case it works same as K-means algorithm. In the latter case the algorithm computes the new cluster centers by incrementing the cluster counter by one in each iteration until it satisfies the validity of cluster quality.
2. Dynamic clustering and management of mobile wireless sensor networks [29], represents a self-organizing

and adaptive Dynamic Clustering (DCMDC) solution to maintain networks. This solution is based on dividing the network into well delimited clusters. Incrementally adding the number of clusters by creating new ones from the misclassified elements.

3. Fully Dynamic k-Center Clustering [30], many real-world applications might need to deal with arbitrary deletion and insertions. For example, one might need to remove data items that are not necessarily the oldest ones, because they have been flagged as containing inappropriate content or due to privacy concerns. Clustering trajectory data might also require to deal with more general update operations. The algorithm is based on $(2+\epsilon)$ approximation for the k-center clustering problem with small amortized cost under the fully dynamic adversarial model. In such a model, points can be added or removed arbitrarily, provided that the adversary does not have access to the random choices of the algorithm.

For the incremental phase it will be based on the split process but the number of clusters can decrease not only grown. In the literature the merge of clusters where based on CF-Trees and its additive properties [20]. To cover the issue of the grown and decrease of number of clusters we propose a new algorithm based on k-means which allows the split and merge of clusters while receiving data streams. This algorithm is based on different criterion from statistics.

A. Used Indexes

Methods that can be used to split clusters and other that allows the merge of two clusters are against each other. One is searching for dispersion within clusters. The second is looking for the intra-clusters similarity.

1. **Split techniques**

   There are different criterion that can be the base of a split procedure in a group of elements. Between those we can find Sum of Squared Error known as SSE and the index of dispersion.

   - **Sum of Squared Error**
     SSE is the sum of the squared differences between each observation and its group's mean. It can be used as a measure of validation within a cluster. If all elements within a cluster are identical, the SSE would then be equal to zero. SSE is a criteria for testing the quality of clustering algorithms and used also in decision trees as a split criteria for nodes [20].

     $$SSE_{x \in C} = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{kj} - c_i) \tag{6}$$

   - **Index of dispersion**
     The index of dispersion or also known as variance to mean value, is a common used index in statistics and probability theory. It quantifies if observations in a data set are dispersed or highly related to its centroids. The index of dispersion is the square of the standard deviation divided by the mean of the observation [21].

     $$\frac{\sigma^2}{\mu} \tag{7}$$

     Where the standard deviation is calculated as:

     $$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{n} (x_i - \mu)^2} \tag{8}$$

     It is used to quantify either if data elements are close to their center or if they are widely dispersed. A low standard deviation means that data are optimally clustered. On the other hand, high value of standard deviation indicates that data points are spread out over a wider range of values [22].

2. **Merge techniques**

   Merging two sets of data refers to join elements of both of them together. In clustering for merging two clusters, the main task refers to the choice of the most similar two. There are different indexes that can give data similarity. Between those indexes there are:

   - **Davies-Bouldin index**
     Davies-Bouldin is based on the idea that the intra-cluster similarity should be as lower as possible. On the other hand, cluster compactness must be high. The index is based on the index of similarity, its manner of calculation is shown next. The similarity index reflect how much two clusters are related to each other [23].

     $$SI_{ij} = \frac{I(c_i) + I(c_j)}{I(c_i, c_j)} \tag{9}$$

   - **Centroid distance:**
     Centroid distance is the most commonly used distance. It is based on the distance between two different clusters centers. This index is used in different machine learning algorithms such as KNN [24].

IV. Proposed approach : D-kmeans

Our solution consists of using two alternative operations on clusters, namely: split and merge. The split operation gives the opportunity of dividing a cluster into two sub-clusters. On the other hand, the merge of two clusters refers to create one cluster based on the two chosen ones. Our proposal consists of four stages as shown in Figure 1, starting from the data preprocessing until the stage of split or merge of cluster. In the following sections, each step will be detailed separately for better understanding of the different aims of the proposed approach.

A. Clustering Phase

- **Step 1: K-means**
  At this level, we aim to create k clusters using the standard k-means algorithm. It represents the initialization phase of the whole functionality of the proposed

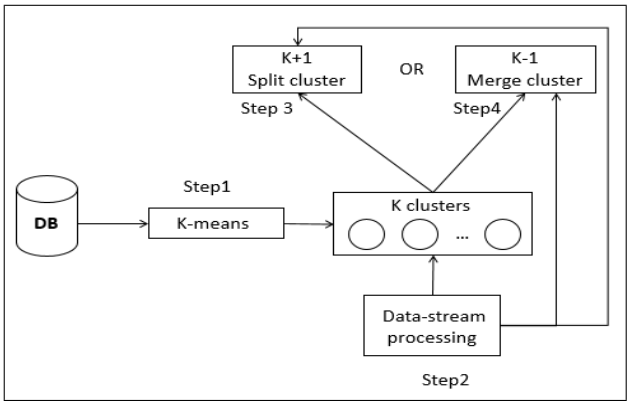


Figure. 1: Workflow of D-kmeans

approach. Defining the number of clusters has no impact on the process of our approach.

- **Step 2: Data-streams preprocessing**
  This phase consists of sending vectors of data to each cluster based on the Euclidean distance, and at each iteration, the center of the cluster where the stream data has been inserted will be updated and all inserted elements in each cluster should be verified based on the new clusters centroids. Data-stream could be interrupted at any time to response to the user demand of modifying the number of clusters.

The pseudo-code of the data-stream preprocessing algorithm is presented next.

Input clusters = $\{c_1, c_2, ...c_k\}$, Streaming file F
Output Streams file F
Begin
  While (DataIncoming = True) AND (F = NotEmpty) **do**
    clustering (C, firstLine.toVector())
    delete (F, firstLine)
    For $(j = 1, j < k)$ **do**
      update (C_j)
    end For
  end While
Return F
End.

- **Streaming file:**
  It represents a file that contains the rest of the dataset after the creation of the k clusters using simple k-means. Every ten seconds, ten lines from that file will be collected in order to create true real time streams.

- **Clustering():**
  This function seeks for the most suitable cluster that could contain the new coming object based on Euclidean distance, the function compares the distance between clusters centroids and the object. Line.ToVector() gives the opportunity of creating an instance from the streaming file line.

- **Delete():**
  After adding the incoming element, it will be deleted from the streaming file to avoid the redundancy of elements insertion into clusters.

- **Update():**
  Aims to update the distribution of elements between clusters and also updating clusters centers to be ready for collecting new incoming data.

B. Split process

Our proposed split process takes into consideration the adding a new cluster to the actual distribution of cluster without retraining from scratch. In order to showcase this work there is an ultimate challenge witch is; witch cluster should we choose to split. In literature like it was induced before there are different criterion to chose the cluster to split.
Our proposed split process is based on three different steps to get final clusters, they are organized as follow:

1. Calculate Score for all clusters
   - Calculate SSE
   - Calculate Dispersion Index
2. Searching for the highest Score
3. Split the cluster with the highest Score using K-Mean

For the sake of clarity Fig.1 shows the main different stages from the choice of the cluster to split until getting the k+1 clusters as output.

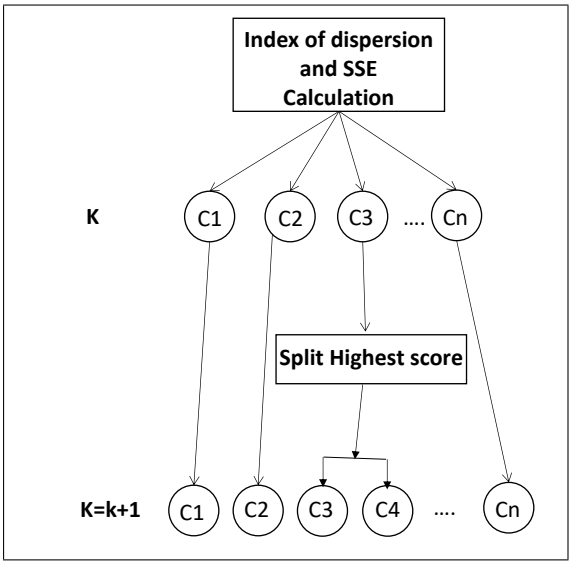


Figure. 2: Structure of the incremental k-Means

In our work we use both of Sum of Squared Error (SSE) calculation and Index of dispersion at first level and the size of clusters at a second level. Through a way or an other those tow indexes refers to all other split criterion that have been mentioned; the shape, size and the variance of clusters, because it calculate the dispersion of elements from the centroid of the cluster[16].
The split process is based on three different split criterion that have been induced from statistic and other used with other machine learning algorithms. The first split criteria is SSE (Sum of Error Square) which used by CF-trees to split nodes and used as an evaluation criteria for clustering algorithms. The second used index is the index of dispersion, and the last one is the size of clusters.

The split process begins by calculating both of SSE and index of dispersion of each cluster. If the two indexes indicate two different clusters, then moving to the third criteria which is size of cluster. In this case, the choice will be the cluster that have the largest number of elements between the two chosen. Otherwise, the algorithm will split the cluster that had the biggest SEE and Dispersion at the same time with out referring to the size criteria.
the split process will be done using Simple K-means algorithm

Pseudo-code of the split process

Input: clusters = $\{c_1, c_2, ...c_k\}$
Output: clusters = $\{c_1^{'}, c_2^{'}, ...c_k^{'}, c_{k+1}^{'}\}$
Begin
$c^{'}$ =c
For each cluster $c_i^{'}$ in $\{c_1^{'}, c_2^{'}, ...c_k^{'}\}$ **do**
computeSSE ($c_i^{'}$)
computeID ($c_i^{'}$)
end For
For (j=1, j<= k) **do**
If $SSE(cluster_i) = Max(SSE(c_1^{'}), .., SSE(c_k^{'}))$ **then**
$Number_S \leftarrow j$
end If
If $ID(cluster_i) = Max(ID(c_1^{'}), .., ID(c_k^{'}))$ **then**
$Number_I \leftarrow j$
end If
end For
If $(Number_S = Number_I)$ **then**
Split($cluster_{Number_S}$)
Else If $(cluster.Size(cluster_{Number_S}) > cluster.Size(cluster_{Number_I}))$ **then**
Split($cluster_{Number_S}$)
Else
Split($cluster_{Number_I}$)
end If
Return clusters = $\{c_1^{'}, c_2^{'}, ...c_k^{'}, c_{k+1}^{'}\}$
End.

- **Compute(cluster):**
  - **Input:** Cluster *i*.
  - **Output:** SSE and ID for cluster *i*.

  This method gives the opportunity to calculate the SSE of each cluster and also the calculation of the Index of Dispersion of each cluster.

- **Max(clusters):**
  - **Input:** Clusters.
  - **Output:** Index of cluster.

  The function Max() searches into the collection of clusters SSE and clusters Index of Dispersion for the highest values.

- **Split(cluster):**
  - **Input:** Clusters *i*.
  - **Output:** Tow sub-clusters.

  The function split() aims to run the simple k-means algorithm with number of clusters k=2.

- **cluster.Size(cluster):**
  - **Input:** Clusters *i*.
  - **Output:** Size of the cluster *i*.

  It gives the size of a cluster. The size of a cluster refers to the number of elements within the cluster.

- $Number_S$ and $Number_I$ are two variable that contain the index of the cluster that have the highest SSE and the cluster that have the highest ID.

C. Merge process

Merging two clusters aims to find the most coherent clusters. It searches into k clusters for the two closest clusters that can be merged, it is based on two indexes. The first one is Closest Centers of clusters, and the second is the Davis-Bouldin Index. Figure 3 shows the main idea of the merge technique functionality. For the first merge criteria, it calculates the

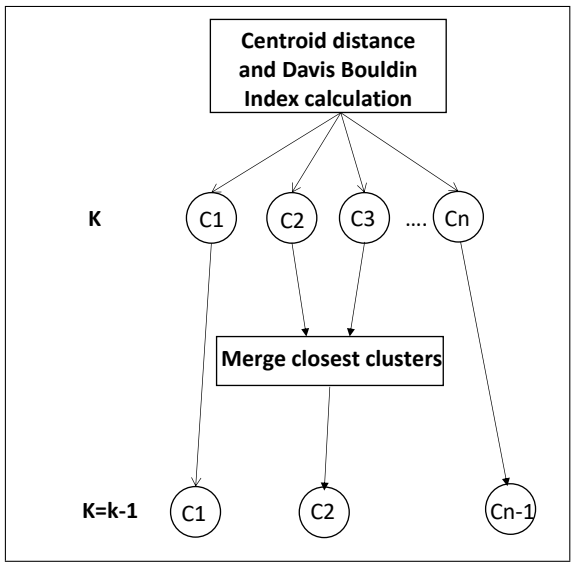


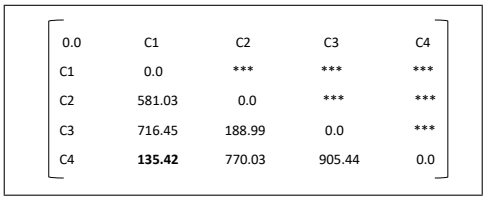
Figure. 3: Workflow of the merge technique

center of each cluster and creates a matrix that contains all distances between each pair of clusters. Table 1 shows the number of elements of each cluster for an example dataset D1 with initialization of cluster number k=4, Figure 4 shows the matrix that contains centroid distances for each cluster.

Table 1: Number of instances per cluster, k=4

Datasets	**D1**
Cluster 1	104
Cluster 2	206
Cluster 3	180
Cluster 4	93

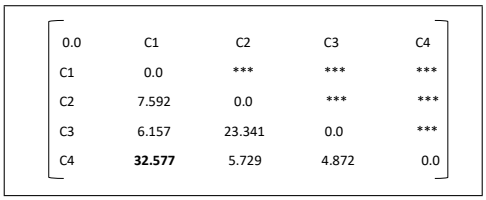
The first line and the first column represent centroids of different clusters. In our example, we have four clusters. Centroids distance matrix is a square matrix where each row represents the distance between two different clusters centroids



0.0	C1	C2	C3	C4
C1	0.0	***	***	***
C2	581.03	0.0	***	***
C3	716.45	188.99	0.0	***
C4	**135.42**	770.03	905.44	0.0

Figure. 4: Centroids Distance Index per cluster

using the Euclidean distance. Clusters centroids will be calculated like:
($CC = \sum \frac{x_i}{n}$), where x_i represents the element number *i* from the current cluster and *n* is the total number of elements within this cluster. Between those distances that have been calculated, the algorithm will take the two closest clusters that have the lowest distance. Moving to the second merge criteria which is Davis Bouldin Index, Figure 5 shows an example of the Davis Bouldin Index matrix with the same four clusters used previously with the Centroids Index criteria. Contrary to Centroids Index, the Davis Bouldin Index will



0.0	C1	C2	C3	C4
C1	0.0	***	***	***
C2	7.592	0.0	***	***
C3	6.157	23.341	0.0	***
C4	**32.577**	5.729	4.872	0.0

Figure. 5: Davis Bouldin Index matrix per cluster

take into consideration the two clusters that have the highest index. The matrix shows the degree of similarity between clusters, the bigger it is the closer clusters are. The similarity calculated by the Davis Bouldin Index is shown in equation 7:

$$SI_{ij} = \frac{I(c_i) + I(c_j)}{I(c_i, c_j)} \tag{10}$$

Where:

- $I(c_i)$ represents the average of distance between each element and its corresponding cluster centroid.
- $I(C_i, C_j)$ is the distance between the two clusters centroids.

The merge process is based on two indexes criterion as shown.. The Centroids Index searchs for the minimum distance into a matrix of distances between clusters. Otherwise, the Davis Bouldin Index generates a matrix of similarity indexes between clusters, the two clusters that have the biggest index are the best to be merged. In the Figure 3 lowest Centroid Index where presented in bold which refers to clusters (2 and 3). Also, Davis Bouldin Index presented in Figure 4 refers to clusters (2 and 3). In our case both Davis Bouldin Index And Centroids Index indicates the same couple of clusters to be merged. Clusters that have the biggest index or the closest centroid distance are not always the best choice for the merge procedure if both indexes are not in agreement about the clusters to merge. In this case, both clusters resulting from the two indexes will be merged. The cluster that will be maintained is the cluster which has the lowest SSE.

The pseudo-code of the merge process

Input: clusters = $\{c_1, c_2, ...c_k\}$
Output: clusters = $\{c_1^{'}, c_2^{'}, ...c_{k-2}^{'}, c_{k-1}^{'}\}$
Begin
  $c^{'}$ =c
  For each cluster A_i in $\{c_1^{'}, c_2^{'}, ...c_k^{'}\}$ **do**
   For each cluster B_j in $\{c_1^{'}, c_2^{'}, ...c_k^{'}\}$ **do**
    $CD \leftarrow computeCentroidDistance(A_i, B_j)$
    $DB \leftarrow computeDavisBouldinIndex(A_i, B_j)$
   end For
  end For
  For each cluster A_i in $\{c_1^{'}, c_2^{'}, ...c_k^{'}\}$ **do**
   For each cluster B_j in $\{c_1^{'}, c_2^{'}, ...c_k^{'}\}$ **do**
    If $computeCentroidDistance(A_i, B_j)$ = Min(CD) **then**
     $Clust1 \leftarrow A_i$
     $Clust2 \leftarrow B_j$
    end If
    If $computeDavisBouldinIndex(A_i, B_j)$ = Max(DB) **then**
     $Clust3 \leftarrow A_i$
     $Clust4 \leftarrow B_j$
    end If
   end For
  end For
  If (Clust1=Clust3) and (Clust2=Clust4) **then**
    Merge (Clust1, Clust2)
    Delete (Clust1)
    Delete (Clust2)
  end If
    $SSE1 \leftarrow SSE(Merge(Clust1, Clust2)$
    $SSE2 \leftarrow SSE(Merge(Clust3, Clust4)$
  end If
  If SSE1 $<$ SSE2 **then**
    $NewClust \leftarrow Merge(Clust1, Clust2)$
    Delete (Clust1)
    Delete (Clust2)
  Else
    $NewClust \leftarrow Merge(Clust3, Clust4)$
    Delete (Clust1)
    Delete (Clust2)
  end If
Return clusters = $\{c_1^{'}, c_2^{'}, ...c_{k-2}^{'}, c_{k-1}^{'}\}$
End.

- **Compute(clusters):**
  - **Input**: Clusters
  - **Output**: Table

  This function has two functionalities: The first one is the generation of a matrix (CD) that contains distances between each two clusters centroids, and a second function which allows the creation of matrix (DB) that contains the Davis Bouldin Index between each pair of clusters.

- **Compute(clusters):**
  - **Input**: Clusters
  - **Output**: Table

This function has two functionalities: The first one is the generation of a matrix (CD) that contains distances between each two clusters centroids, and a second function which allows the creation of matrix (DB) that contains the Davis Bouldin Index between each pair of clusters.

- **Merge(clusters):**
  - **Input**: cluster *i*, cluster *j*
  - **Output**: Table

  The merge function provides the opportunity of creating one cluster from the two most similar clusters that have been chosen using both of indexes.

- **Compute(clusters):**
  - **Input**: Clusters
  - **Output**: Table

  This function has two functionalities: The first one is the generation of a matrix (CD) that contains distances between each two clusters centroids, and a second function which allows the creation of matrix (DB) that contains the Davis Bouldin Index between each pair of clusters.

- **Min(matrix)/ Max(matrix):**
  - **Input**: Matrix of indexes
  - **Output**: Highest index

  Min searches into the centroids distances matrix for the smallest distance between two clusters. Otherwise, Max() function does the opposite work, it searches into the Davis Bouldin Index matrix for the biggest index between two clusters, that reflects how much elements of both clusters are close to each other. The higher is the index, the best are clusters to be merged.

- **Delete(cluster):**
  - **Input**: cluster *i*

  It allows removing clusters that have been merged after the creation of the new cluster that contains elements of both of them.

- **Clust1,2,3,4:**
  Clust goes from 1 to 4, they are temporary clusters that will contain elements of clusters that have been chosen from both of indexes and ready for the test. We use those temporary clusters to not lose information within the chosen clusters.

- **NewClust:**
  It represents the new cluster that will be added to the main distribution of clusters. NewClust contains elements of the most similar pair of clusters.

V. Experimentation

A. Framework

The evaluation is divided into two parts: the first part will be dedicated to the split process against the Incremental k-means algorithm, and the second part contains a comparison between the merge process and Incremental k-means with different numbers of clusters. Those comparisons have as purpose the evaluation of the memberships of elements within clusters using different sizes of clusters in connection with the run time.
Four different datasets are used to evaluate the performance of our approach. Three real datasets (Airlines, Bank-data, 3D road network) and a simulated data base (BNG (vehicle)). The number of attributes are not the same for all of the datasets as shown in Table 4.1. Also, there are no missing values. Airline dataset, bank-data and 3D road network dataset were obtained from the UCI repository [1] and the last dataset BNG (vehicle) was obtained from OpenML [2]. Table 2 shows a description in terms of number of instances and attributes for the different datasets that have been used for the experiments.

Table 2: Description of the used datasets

Datasets	#Instances	Attributes
Airlines (A)	539382	8
Bank-data (BD)	600	12
3D road network (RD)	434874	4
BNG (vehicle) (VH)	792698	19

- **Airlines:** Is a real dataset Inspired in the regression dataset from Elena Ikonomovska. The task is to predict whether a given flight will be delayed, given the information of the scheduled departure. The dataset is composed of 539383 instances distributed between 8 attributes.
- **Bank-data:** : Contains 600 instances, that refer to 12 different attributes concerning some accounts in a bank. Bank-data contains real information about bank clients.
- **3D road network:** 3D road network with highly accurate elevation information from Denmark used in eco-routing and fuel/Co2-estimation routing algorithms. This Dataset contains 434874 instances and 4 attributes.
- **BNG (vehicle):** Vehicle is a simulated dataset that contains 792698 elements and 19 attributes.

B. Evaluation criteria

Clustering analysis does not have any solid evaluation measure that can be used to evaluate the outcome of different clustering algorithms [**?**]. But based on the result of the clustering process, we can find how similar objects within clusters are and how distinct generated clusters are. In our evaluation we used two different measures: The first one is Sum

[1] https://archive.ics.uci.edu/ml/index.php

[2] https://www.openml.org/d/268

of Squared Error (SSE), and the second is Silhouette Index. Both indexes give a feedback of the similarity inter-cluster and intra-clusters. In addition, we compare our approach against the k-means algorithm in term of run-time.

- **Sum of Squared Error**
  SSE as mentioned, calculates the dispersion of elements of a cluster in relation with their centroid. Its equation is shown as follows:

$$SSE_{x \in C} = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} - c_i) \tag{11}$$

- **Silhouette Index**:
  It refers to a method for interpretation and validation of clusters. The silhouette is based on cohesion / separation. It searchs how much an object *i* is related to its cluster compared to other clusters. The results are in the range of [-1..1]. The higher is the result the best, is the clustering. Silhouette can be calculated with any distance measure like Euclidean or Manhattan [25]. Equation 9 shows how the silhouette index is measured:

$$S(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{12}$$

  Where:

  - a(i): represents the distance between *i* and all other objects in the same cluster. It refers to the degree of membership of the element *i* to the cluster that was inserted into. The Smallest is the value of a(i), the better is the assignment.
  - b(i) is the smallest value of the average of distances between *i* and all other clusters. The lowest value of b(i) refers to the second best fit cluster that *i* can be inserted into.
  - **Run-time**
    The run-time represents the time needed to create the final result from the beginning of the procedure until getting the final distribution of elements between clusters.

C. Results and Discussion

In this part, we study the results collected from different experiments. We evaluate the performance of the proposed Dynamic algorithm for data streams clustering denoted D-kmeans, compared to Incremental k-means algorithm. This work is mainly devoted to testing the scalability of our proposed approach, showing how it capes with the weaknesses of the Incremental k-means.
For the sake of clarity, the evaluation of the split process and merge process will be done separately following different steps of the split process and merge process.

Split process evaluation

To investigate the capability of the split process, Table 3 summarizes the number of elements within each cluster after the initialization using k-means algorithm and the reception of data streams. The choice of the cluster to split will follow the split process steps:

1. Calculate SSE and ID
2. Verify clusters to be split.
3. Rerunning k-means with k equal to two on the chosen cluster.

Table 3: Number of instances when k=3

Datasets	A	BD	RD	VH
Cluster 1	175085	259	208329	232555
Cluster 2	254354	183	87929	256205
Cluster 3	109944	157	138616	303938

After the split of clusters which is based on SSE and ID for the choice of the cluster to split, results will be compared against results obtained from Incremental k-means with the same number of clusters.
Split process steps will be done on each of the four datasets. Table 4 contains the results of SSE for each cluster in every dataset. Table 5 presents the Indexes of dispersions of each cluster. Elements written in bold represent highest SSE and highest ID for each cluster. Clusters to be split after the use of split criterion of D-kmeans are shown in Table 6, and Table 7 presents the new distribution of elements between cluster after the split.

Table 4: SSE of clusters when k=3 (E10)

Databases	A	BD	RD	VH
Cluster 1	102,02	16,51	**0.317**	3,05
Cluster 2	**318,48**	**22,02**	0.134	**9,23**
Cluster 3	41,22	12,52	0.022	5,15

Table 5: Index of dispersion for each cluster

Databases	A	BD	RD	VH
Cluster 1	43.93	165.2	4.127	8.36
Cluster 2	**55.29**	**189.2**	**4.208**	**12.15**
Cluster 3	40	170.8	4.224	9.17

Table 6: Clusters to split

Databases	A	BD	RD	VH
Cluster 1	175085	259	**208329**	232555
Cluster 2	**254354**	**183**	87929	**256205**
Cluster 3	109944	157	138616	303938

Table 7: Number of instance when k=4 using D-kmeans

Databases	A	BD	RD	VH
Cluster 1	175085	259	**99480**	232555
Cluster 2	**165714**	**108**	**108849**	**128110**
Cluster 3	**88640**	**75**	87929	**128095**
Cluster 4	109944	157	138616	303938

The first evaluation criteria is SSE, Table 8 contains different SSE of each dataset, the first row is dedicated to the incremental k-means and the second row contains SSE obtained using D-kmeans. For a better comprehension of SSE results given for both algorithms, Figure 6 shows different obtained results. Results in Figure 6 were normalized using the equation 13.
From Figure 6, we can conclude that the dispersion of elements within clusters according to SSE are almost the half of the dispersion obtained using k-means.

$$Value = \frac{Value - min}{max - min} \tag{13}$$

To verify results obtained using SSE in relation with elements dispersion within clusters we use the Silhouette Index with the same distribution of clusters, Figure 7 presents different Silhouette Indexes for each dataset using both algorithms: Incremental k-means and D-kmeans. Silhouette Index and SSE represented in Figure 6 and Figure 7, shows that our D-kmeans algorithm outperforms the incremental k-means in terms of stability of clusters. Clusters generated from D-kmeans respect better the basic notion of clustering (maximization of the coherence within cluster and minimization of the similarity between clusters).
Next, we compare the needed time for the execution of Incremental k-means versus that required from D-kmeans. From the execution time shown in Figure 8, we can conclude that D-kmeans split process uses less time than the Incremental k-means, and the offset is proportional the size of the dataset. Our split approach outperformed the Incremental k-means algorithm in terms of processing performance.
Our split process outputs clusters that respects better basics of clustering in term dispersion of elements within and between clusters. Besides, the quality of clustering, D-kmeans split process uses less time during its execution. **Merge process evaluation**
For the evaluation of the merge process, we use the same datasets for the test of the split process but with different initialization of clusters number. At this level, we fix k=4. Table 9 summarizes the number of elements within each cluster with regards to the four used datasets. As used previously in the case of the evaluation of the split process, experiments of the merge process will look also into the dispersion of elements within and between clusters. It follows the steps of the merge process:

1. Calculate Centroids distances and Davis Bouldin Index for each cluster
2. Verify clusters to merge.
3. Merge the two chosen clusters of each dataset.

Figure 9 and Figure 10 present centroids indexes and Davis Bouldin Indexes for the VH datasets. We can notice that cluster number 2 and cluster number 3 are going to be merged as they are cluster that have lowest centroids distance and highest Davis Bouldin Index. Table 10 contains clusters to be merged of each dataset after they have undergone

Table 8: SSE Total (E10)

Datasets	Incremental k-means	D-Kmeans
A	608.1	160.8
BD	24.1	12.5
RD	0.017	0.0096
VH	18.3	12.07

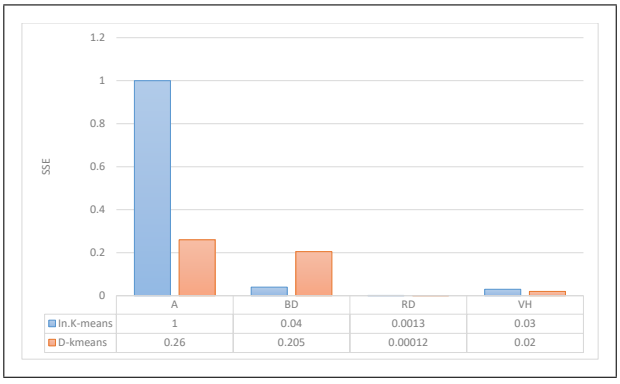


Figure. 6: SSE Total for each dataset (Split process)

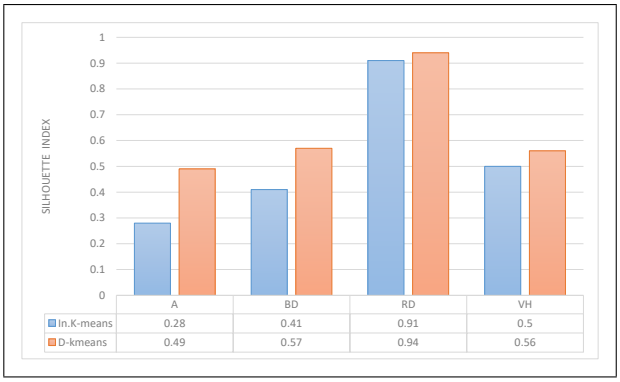


Figure. 7: Silhouette Index for each dataset (Split process)

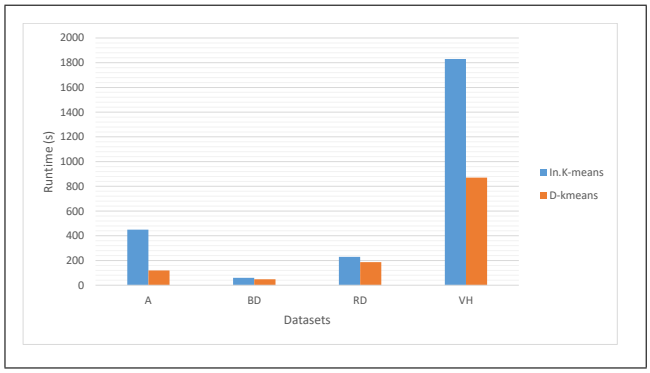


Figure. 8: Run-time obtained by In.K-means and D-kmeans

Table 9: Number of instances when k=4

Datasets	**A**	**BD**	**RD**	**VH**
Cluster 1	124351	209	199760	140558
Cluster 2	207723	163	59143	139227
Cluster 3	86492	129	44090	150273
Cluster 4	120817	98	131881	147841

0.0	C1	C2	C3	C4
C1	0.0	***	***	***
C2	32.955	0.0	***	***
C3	34.198	**906.423**	0.0	***
C4	131.909	26.367	27.157	0.0

Figure. 9: Centroids Distance Index per cluster (VH dataset)

0.0	C1	C2	C3	C4
C1	0.0	***	***	***
C2	32.955	0.0	***	***
C3	34.198	**906.423**	0.0	***
C4	131.909	26.367	27.157	0.0

Figure. 10: Davis Bouldin Index matrix per cluster (VH dataset)

Table 10: Clusters to be merge

Datasets	**A**	**BD**	**RD**	**VH**
Cluster 1	**124351**	209	199760	140558
Cluster 2	207723	163	59143	**139227**
Cluster 3	**86492**	**129**	**44090**	**150273**
Cluster 4	120817	**98**	**131881**	147841

the same procedure as clusters of VH dataset. The new distribution of elements between clusters using D-kmeans is presented in Table 11. Clusters presented in bold are those who have been merged. Table 12 contains SSE for each

Table 11: Merged cluster using D-kmeans

Datasets	**A**	**BD**	**RD**	**VH**
Cluster 1	**210843**	209	199760	140558
Cluster 2	207723	163	59143	**289500**
Cluster 3	120817	**227**	**175971**	147841

dataset using Incremental k-means and D-kmeans. For a better comprehension of the obtained result concerning SSE, Figure 11 represents the normalized results in the Table 12. We can notice in Figure 11 that elements within clusters

Table 12: SSE per dataset (E10) (merge process)

Datasets	Incremental k-means	Dynamic K-means
A	461.72	249,73
BD	51.05	48.12
RD	0.473	0.461
VH	17.43	15.59

obtained from D-kmeans merge process results in less dispersed clusters than those generated from Incremental k-means. To check the quality of clusters we evaluate our merge process using the Silhouette index. Figure 12, contains Silhouette Index for both algorithms on the four used datasets for experiments. From Figure 12 we notice that D-kmeans merge process gives better clusters quality than Incremental k-means. The time needed to get final results from D-kmeans is lower than that needed by Incremental

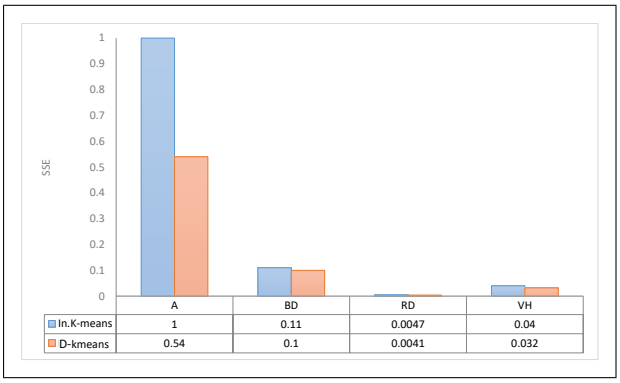


Figure. 11: SSE per dataset (E10) (merge process)

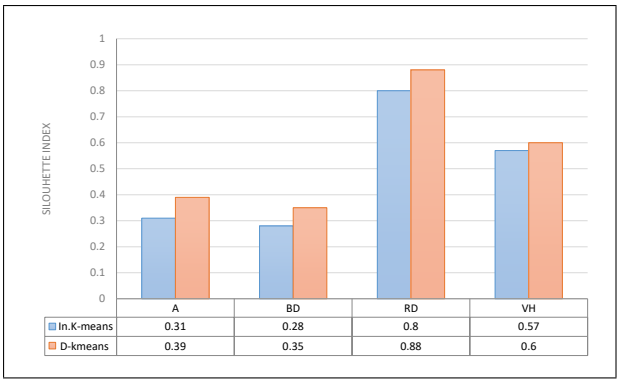


Figure. 12: Silhouette Index for each dataset (merge process)

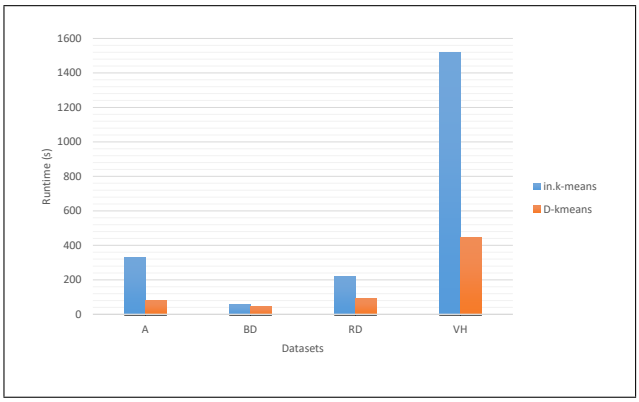


Figure. 13: Run-time for the merge process and D-kmeans

k-Means as shown in Figure 13. Also the execution time for Incremental k-means grows faster with bigger dataset.

We conclude that our D-kmeans outperformed Incremental K-means in terms of cluster quality either in the split process or merge process. In addition the required run-time to get final results is less with D-kmeans than Incremental K-means.

VI. Conclusion

We have proposed a new dynamic clustering algorithm, our proposal is a dynamic technique based on the split and the merge of clusters that have the most spreading out elements. Experimental results demonstrate that our approach perform better then the incremental k-means in term of element dis-

tribution between clusters. The split procedure reduces the SSE for each cluster which provide a lower SSE (total) than that given by incremental K-Means, also the merge process gives better results in term of time and element dispersion intra and inter clusters. As a result our proposed algorithm out performed the incremental k-means algorithm especially when it is related to large data sets.
In future work, we intend to use an heuristic to fix the initial number of clusters. Furthermore, in our work we handled clusters that their number were initialized randomly. However, with the intention to extend our approach to handle bigger size clusters for that we have to ameliorate the used criterion for the split and merge. We intend to ameliorate the quality of clusters by using fuzzy networks..

References

[1] A.Yadav and S.Dhingra, A REVIEW ON K-MEANS CLUSTERING TECHNIQUE, International Journal of Latest Research in Science and Technology, Volume 5, Issue 4: Page No.13-16, July - August 2016.

[2] P.Y.Zhou and K.C.C.Chan, A Model-Based Multivariate Time Series Clustering Algorithm, Springer International Publishing Switzerland,W.-C. Peng et al. (Eds.): PAKDD 2014 Workshops, LNAI 8643, pp. 805817, 2014.

[3] PI.Dalatu, A.Fitriantoa and Aida Mustaphab, Hybrid distance functions for K-Means clustering algorithms, Statistical Journal of the IAOS -1 (2017).

[4] T.Strauss and M.j.Von Maltitz, Generalising Ward's Method for Use with Manhattan Distances PLoS ONE 12(1): e0168288. 2017.

[5] V.B.Surya Prasath, H.Arafat, A.Alfeilatb, O.Lasassmehb, A.B.A.Hassanatb .Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier A Review. Preprint submitted to Elsevier. August 16, 2017.

[6] Li X, Han Q, Qiu B. (2017) A clustering algorithm with affine space-based boundary detection. Applied Intelligence 2:113

[7] Tong Q, Li X, Yuan B. (2017) A highly scalable clustering scheme using boundary information. Pattern Recognition Letters 89:17.

[8] R.R. Patil, A.Khan, Bisecting K-Means for Clustering Web Log data, International Journal of Computer Applications, vol(16), 2015

[9] M.Cap, A.Prez, and J.A.Lozano, An efficient K-means clustering algorithm for massive data, JOURNAL OF LATEX CLASS FILES, VOL. 14, NO. 8, August 2015.

[10] J. Han and M. Kamber, Fast kernel classifiers with online and active learning, Data mining concepts and techniques (2nd ed.), vol. 6, pp. 15791619, 2006.

[11] R. Mall, A. Ahmad, and J. Lamirel, Comportement comparatif des methodes de clustering incrmentales et non incrmentales sur les donnes textuelles htrogenes, (2014).

[12] J.Bao, W. Wang, T.Yang and G.Wu, An incremental clustering method based on the boundary profile. PLoS ONE 13(4), (2018).

[13] Y. Zhanga, K. Lib, H. Guc, and D. Yanga, Adaptive split-andmerge clustering algorithm for wireless sensor networks, International Workshop on Information and Electronics Engineering (IWIEE) (2012).

[14] M. Savaresi, L. Boley, D.S. Bittanti, and G. Gazzaniga, Choosing the cluster to split in bisecting divisive clustering algorithms.

[15] k.Jain and C. Dubes, Algorithms for clustering data, Prentice-Hall advance reference series, (1988).

[16] T. Thinsungnoena, N. Kaoungkub, P. Durongdumronchaib, K. Kerdprasopb, and N. Kerdprasopb, the clustering validity with silhouette and sum of squared errors, Proceedings of the 3rd International Conference on Industrial Application Engineering, 2015.

[17] P. I.Brahmi and S. Ben Yahia, Dtection des anomalies base sur le clustering, (2014)

[18] A. Yadav and G. Singh, Incremental K-means Clustering Algorithms: A Review, International Journal of Latest Trends in Engineering and Technology (IJLTET), 2015, vol. 5, pp. 136-140

[19] J. Bao and W. Wang and G.Wu, An incremental clustering method based on the boundary profile, PLOS ONE, 2018, vol. 13, pp. 1-19

[20] J.A. Silva, E. R. Faria, R.C. Barros, E. R. Hruschka and A.C. P. L. F. De Carvalho, Data Stream Clustering: A Survey, Journal of the ACM, 2014, Vol. 5, pp. 1-37

[21] K.R. Clarke, M.G. Chapman, P.J. Somerfield and H.R. Needham, Dispersion-based weighting of species countsin assemblage analyses, 5th International Conference on Leadership, Technology, Innovation and Business Management, 2006, Vol. 320, pp. 11-27

[22] M.F Al-Saleh and A.E Yousif, Properties of the Standard Deviation that are Rarely Mentioned in Classrooms, AUSTRIAN JOURNAL OF STATISTICS, 2009,vol. 38, pp. 193-202

[23] M. Ghribi, P. Cuxac, J.C. Lamirel and A. Lelu, Mesures de qualite de clustering de documents : Prise en compte de la distribution des mots cls, in proceedings of the 10th Conference Internationale Francophone, 2011, vol. 10, pp. 1-14

[24] S. Mehta, X. Shen, J. Gou and D. Niu, A New Nearest Centroid Neighbor Classifier Based on K Local Means Using Harmonic Mean Distance, Information, 2018, vol. 9, pp. 234-250

[25] G. Songtao, D.X. Luna, S. Divesh and Z. Remi, Record linkage with uniqueness constraints and erroneous values, Proceedings of the VLDB Endowment, vol. 3, pp. 417-428, 2010.

[26] C.Ounali, F. BenRejeb and K. N.Ferchichi, Incremental k-means based on split technique, International Conference on Intelligent Systems Design and Applications. ISDA 2018, AISC 941, pp. 110, 2020.

[27] J. MacQueen, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Some methods for classification and analysis of multivariate observations, 1967, vol. 1, pp. 281-297.

[28] A. Shafeeq and K.S. Haresha, Dynamic clustering of data with modified k-means algorithm, in proceedings of the International conference on information and computer networks, 2012, vol. 27, pp. 221-225.

[29] A. Abuarqouba and M. Hammoudehb and B. Adebisi and S. Jabbar and A. Bounceur and H. Al-Bashar, Dynamic clustering and management of mobile wireless sensor networks, Computer Networks, 2017, vol. 117, pp. 62-75.

[30] H. Chan and A. Guerqin and M. Sozio, In proceedings of the Fully Dynamic k -Center Clustering, World Wide Web Conference, 2018,pp. 579-587.

Author Biographies

Chedi Ounali is a doctoral student at Tunis University of Tunisia. His research interests are artificial intelligence, data mining, and machine learning. Ounali received a master degree in Business Computing from Tunis university of Tunisia. He is a member of BESTMOD Laboratory. Contact him at: chedy.ounelly@gmail.com.

Fahmi Ben Rejab is an associate professor at Tunis University of Tunisia. His research interests are medical informatics, artificial intelligence, data mining, and machine learning. Ben Rejab received a PhD in Business Computing from Tunis university of Tunisia. He is a member of BESTMOD Laboratory. Contact him at: fahmi.benrejab@gmail.com.

Kaouther Nouira is an associate professor at Tunis University of Tunisia. Her research interests are medical informatics, artificial intelligence, and deep learning. Nouira received a PhD in Business Computing from Tunis university of Tunisia. She is Head of the E-Health Research Team at BESTMOD Laboratory. Contact her at: kaouther.nouira@planet.tn.