RESEARCH PAPER
International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009

A Modified *k*-means Algorithm to Avoid Empty Clusters

Malay K. Pakhira
Kalyani Government Engineering College, Kalyani, West Bengal, INDIA
Email: malay_pakhira@yahoo.com

***Abstract*—The *k*-means algorithm is one of the most widely used clustering algorithms and has been applied in many fields of science and technology. One of the major problems of the *k*-means algorithm is that it may produce empty clusters depending on initial center vectors. For static execution of the *k*-means, this problem is considered insignificant and can be solved by executing the algorithm for a number of times. In situations, where the *k*-means is used as an integral part of some higher level application, this empty cluster problem may produce anomalous behavior of the system and may lead to significant performance degradation. This paper presents a modified version of the *k*-means algorithm that efficiently eliminates this empty cluster problem. We have shown that the proposed algorithm is semantically equivalent to the original *k*-means and there is no performance degradation due to incorporated modification. Results of simulation experiments using several data sets prove our claim.**

***Index Terms*—Empty clusters, initial centers, *k*-means, modified *k*-means.**

I. Introduction

Clustering [1]-[4] is an unsupervised classification mechanism where a set of patterns (data), usually multi-dimensional, is classified into groups (clusters) such that members of one group are similar according to a predefined criterion. Clustering of a set forms a partition of its elements chosen to minimize some measure of dissimilarity between members of the same cluster. Clustering algorithms are often useful in various fields like data mining, pattern recognition, learning theory etc. The *k*-means [4] is possibly the most commonly-used clustering algorithm. It is most effective for relatively smaller data sets. The *k*-means finds a locally optimal solution by minimizing a distance measure between each data and its nearest cluster center [5].

Many parallel versions of the *k*-means algorithm [6]-[10] use the basic serial *k*-means at their core. Besides, a number of stochastic clustering algorithms make use of the basic *k*-means or some of its variations. Very often these algorithms are based on Simulated Annealing [11]-[12] or Genetic Algorithms [13]-[14].

The *k*-means clustering algorithm faces two major problems. One is the problem of obtaining non-optimal solutions. As the algorithm is greedy in nature, it is expected to converge to a locally optimal solution only and not to the global optimal solution, in general. This problem is partially solved by applying the *k*-means in a stochastic framework like simulated annealing (SA) and genetic algorithm (GA) etc. The second problem is that of empty cluster generation. This problem is also referred to as the singularity problem in literature. Singularity in clustering is obtained when one or more clusters become empty. Both the problems are caused by bad initialization. Algorithms that use the *k*-means at their core suffer from the empty cluster problem too. A very common practice to solve these problems is to repeat the initialization until we receive a set of good initial center vectors. In practice, after center initialization, we assign elements to the concerned clusters. If an empty cluster is found, at this early stage, a re-initialization takes place. This process is repeated until all non-empty initial clusters are formed. This is, however, an ad-hoc technique. Several other methodical approach are also found. In [15], a refinement approach is proposed, where starting with a number of initial samples of the data set we can obtain a number of sets of center vectors. These center vectors then pass through a refinement stage to generate a set of so called good starting vectors. In [16], a genetically guided *k*-means has been proposed where possibility of generation of empty clusters is treated in the mutation stage. Several *k-d*-tree based methods are found in [17] and [18]. Another approach to initialize cluster centers based on values for each attribute of the data set, has been proposed in [19]. These methods are time costly and may not be applicable by keeping the *k*-means's inherently simple structure.

In this paper, we have presented a modified *k*-means algorithm which eliminates the problem of generation of empty clusters (with some exceptions). Here, the basic structure of the original *k*-means is preserved along with all its necessary characteristics. A new center vector computation strategy enables us to redefine the clustering process and to reach our goal. The modified algorithm is found to work very satisfactorily, with some conditional exceptions which are very rare in practice.

II. The *k*-Means Algorithm For Clustering

The *k*-means algorithm [4] is very commonly used for clustering data. To handle a large data set, a number of different parallel implementations of the *k*-means have also been developed. Here, we shall provide a brief description of the serial *k*-means and some of its properties relevant for the current work.

In crisp partitional clustering, a set D of N patterns $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of dimension d is partitioned into K clusters

denoted by $\{C_1, C_2, \ldots, C_K\}$ such that the sum of within cluster dispersions, i.e., the clustering metric $\mathcal{M}$, as given in (1), becomes the minimum.

$$\mathcal{M}\ (C_1,\ C_2,\ \ldots,\ C_K) = \sum_{k=1}^{K} \sum_{x_j \in C_k} ||\ \mathbf{x}_j - \mathbf{z}_k\ ||^2 \quad (1)$$

Here, $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K\}$ is the set of cluster centers.

A. Basic k-means Algorithm

The basic *k*-means algorithm is commonly measured by any of intra-cluster or inter-cluster criterion. A typical intra-cluster criterion is the squared-error criterion (Equation 1). It is the most commonly used and a good measure of the within-cluster variation across all the partitions. For the current work, we consider intra-cluster squared-error function to evaluate the present scheme of clustering. In basic *k*-means algorithm, a set D of *d*-dimensional data is partitioned into *K* clusters, starting with a set of *K* randomly generated initial center vectors. The process iterates through the following steps:

- assignment of data to representative centers upon minimum distance, and
- computation of the new cluster centers.

The process stops when cluster centers (or the metric $\mathcal{M}$) become stable for two consecutive iterations. The basic *k*-means algorithm is greedy in nature.

The time complexity of the *k*-means clustering algorithm is ***O(TKN)*** where *N* is the number of input patterns, *K* is the desired number of clusters and *T* is the number of iterations needed to complete the clustering. The high time complexity makes *k*-means unsuitable for large applications. Being a greedy algorithm, it often converges to a local minima, may produce empty clusters, but performs well enough for smaller data sets.

III. The Modified *k*-Means Algorithm

The clustering schemes employing the basic *k*-means algorithm (Section II) face a significant problem of generation of empty clusters at different instances due to bad initialization. In this section, we report a scheme to address the above issue of generation of empty clusters and propose an algorithm to modify the center vector updation procedure of the basic *k*-means that denies the possibility of empty clusters. We denote this new algorithm as *m_k*-means algorithm.

A. Basis of the Proposed Scheme

In the basic *k*-means algorithm, an iteration starts with a set of old center vectors $\mathbf{z}_k^{(\text{old})}$, the data elements are distributed among clusters depending on minimum Euclidean distance, and then a set of new cluster centers $\mathbf{z}_k^{(\text{new})}$ is generated by averaging the data elements. This center updation procedure in the original *k*-means algorithm can be mathematically described as:

$$\Rightarrow\ \mathbf{z}_k^{(new)} \leftarrow \frac{1}{n_k}\{\sum_{x_j \in C_k} (\mathbf{x}_j)\} \quad (2)$$

where, n_k is the number of elements in cluster C_k. If the new centers $\mathbf{z}_k^{(\text{new})}$ do not match exactly with the old centers $\mathbf{z}_k^{(\text{old})}$, the *k*-means algorithm enters into the next iteration assuming $\mathbf{z}_k^{(\text{new})}$ as $\mathbf{z}_k^{(\text{old})}$.

In case of the *m_k*-means algorithm, the computation of new center vectors differs from that in the *k*-means algorithm. Here, the old center vectors $\mathbf{z}_k^{(\text{old})}$ are assumed to be members of the concerned clusters along with the allocated data items. In this scheme, we deny the formation of empty clusters. Therefore, center updation procedure in *m_k*-means can be written as :

$$\Rightarrow \mathbf{z}_k^{(new)} \leftarrow \frac{1}{n_k+1}\{\sum_{x_j \in C_k} (\mathbf{x}_j) + \mathbf{z}_k^{(old)}\} \quad (3)$$

Equation 3 indicates that every cluster should always contain at least one element. One may guess that incorporation of $\mathbf{z}_k^{(\text{old})}$ in computation of $\mathbf{z}_k^{(\text{new})}$, may affect the rate of convergence to final cluster centers and the quality of clustering solutions. However, since the value of $\mathbf{z}_k^{(\text{old})}$ soon becomes well inside the concerned cluster (within a few iterations), the effect of $\mathbf{z}_k^{(\text{old})}$ will be very insignificant, and for a large data set it can be assumed to be negligible. The following subsection reports the design of the present algorithm, referred to as *m_k*-means clustering algorithm.

B. The m_k-means Algorithm

The execution steps of the *m_k*-means algorithm to form clusters are essentially similar to those of the original *k*-means algorithm. The processor maintains the cluster structures in its own local memory and iterates through the steps of the *m_k*-means algorithm to evaluate a final set of cluster centers **Z**. The execution steps to be followed are summarized below.

The *m_k* -means Algorithm

Input: a set D of *d*-dimensional data and an integer *K*.
Output: *K* clusters
begin
  randomly pick *K* points $\in$D to be initial means;
  while measure $\mathcal{M}$ is not stable **do**
    begin
      compute distance $d_{kj} = ||\mathbf{x}_j - \mathbf{z}_k||^2$ for each k, j where $1 \le k \le K$ and $1 \le j \le N$, and determine members of new *K* subsets based upon minimum distance to $\mathbf{z}_k$ for $1 \le k \le K$;
      compute new center $\mathbf{z}_k$ for $1 \le k \le K$ using (3);
      compute $\mathcal{M}$;
    end
end

The above algorithm reveals that the new clustering scheme is exactly similar to the original *k*-means algorithm except the only difference at the center computation step. In the following subsection, we shall try to prove that the *m_k*-means algorithm converges to *k*-means centers and the rate of convergence is almost equal to that of the original *k*-means algorithm.

C. Proof of Convergence for the m_k-means Algorithm

As mentioned in Section III-A, incorporation of $\mathbf{z}_k^{(\text{old})}$ in computing $\mathbf{z}_k^{(\text{new})}$ may affect the rate of convergence of the present algorithm. However, this effect is insignificant and for a large data set it can be assumed to be negligible. An analytical measure is described in this subsection to estimate the effect of the old centers while computing new ones.

Let us consider that at iteration t, data items are distributed among correct clusters, but center vectors are yet to be stable. Let, at this stage, a certain cluster be represented by $C = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ and the corresponding center is $\mathbf{z}^{(t)} = \mathbf{x}$ (say), where

$$\mathbf{x} \neq \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m}{m}$$

due to the initial center effect. Therefore, the following is a sequence of center vectors obtained in the subsequent iterations:

$$\mathbf{z}^{(t)} = \mathbf{x}$$

$$\mathbf{z}^{(t+1)} = \frac{\sum_{i=1}^{m}\mathbf{x}_i + \mathbf{z}^{(t)}}{m+1} = \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} + \frac{\mathbf{x}}{m+1}$$

$$\mathbf{z}^{(t+2)} = \frac{\sum_{i=1}^{m}\mathbf{x}_i + \mathbf{z}^{(t+1)}}{m+1} = \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} + \frac{\sum_{i=1}^{m}\mathbf{x}_i}{(m+1)^2} + \frac{\mathbf{x}}{(m+1)^2}$$

$$\mathbf{z}^{(t+3)} = \frac{\sum_{i=1}^{m}\mathbf{x}_i + \mathbf{z}^{(t+2)}}{m+1} = \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} + \frac{\sum_{i=1}^{m}\mathbf{x}_i}{(m+1)^2} + \frac{\sum_{i=1}^{m}\mathbf{x}_i}{(m+1)^3} + \frac{\mathbf{x}}{(m+1)^3}$$

Proceeding in a similar fashion, we get

$$\mathbf{z}^{(t+u)}$$

$$= \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} + \frac{\sum_{i=1}^{m}\mathbf{x}_i}{(m+1)^2} + \cdots + \frac{\sum_{i=1}^{m}\mathbf{x}_i}{(m+1)^u} + \frac{\mathbf{x}}{(m+1)^u}$$

$$= \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1}\left[1 + \frac{1}{(m+1)} + \cdots + \frac{1}{(m+1)^{u-1}}\right] + \frac{\mathbf{x}}{(m+1)^u}$$

$$= \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} \times \frac{1 - \left(\frac{1}{m+1}\right)^u}{1 - \frac{1}{m+1}} + \frac{\mathbf{x}}{(m+1)^u}$$

For large values of m or u, $1/(m+1)^u \rightarrow 0$. Hence,

$$\mathbf{z}^{(t+u)} = \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m+1} \times \frac{m+1}{m} = \frac{\sum_{i=1}^{m}\mathbf{x}_i}{m} \qquad (4)$$

which is the desired cluster center (using *k*-means). And the additional u iterations needed for final convergence can be expressed by the condition $1/(m+1)^u \rightarrow 0$. For large data sets (large m), even for a small u this condition holds. That is, the effect of the initial centers toward convergence is negligible.

D. Proof of Generation of Non-empty Clusters

In the above subsection, we have shown that the *m_k*-means algorithm converges to the *k*-means centers. In the following we show that the *m_k*-means algorithm generates non-empty clusters only as opposed to the conventional *k*-means. The problem of empty clusters occurs when the initial center vectors $\mathbf{z}_1^{(0)}, \mathbf{z}_2^{(0)}, \cdots, \mathbf{z}_K^{(0)}$ are such that any two or more of them are either equal or very close to each other. In such a situation, after assignment of data to clusters, data elements will be assigned to only one of the clusters with nearly equal centers, and the others remain empty.

Let D = $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ be a data set with $(m+n)$ data elements forming two well separated clusters X and Y as shown in Figure 1. If we apply serial *k*-means algorithm to partition this data set into two clusters, we get the following center vectors $\mathbf{z}_1 = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m}{m}$ and $\mathbf{z}_2 = \frac{\mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n}{n}$.

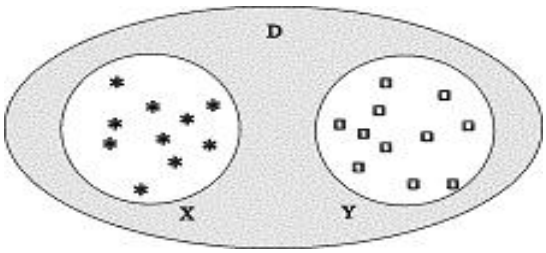

Fig.1. Sample data set with two clusters.

Now let us consider execution of *k*-means and *m_k*-means on the above data set with identical initial centers.

Execution using *k*-means:

Let the two cluster centers be initialized as $\mathbf{z}_1^{(0)}$ and $\mathbf{z}_2^{(0)}$ respectively (where, $\mathbf{z}_1^{(0)} = \mathbf{z}_2^{(0)}$), then all the data elements go to either cluster 1 or cluster 2 only. If we assume all data go to cluster 1, then the data set will be partitioned as $D_1 = D$ and $D_2 = \emptyset$, producing new centers as

$$\mathbf{z}_1^{(1)} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n}{m+n}, \text{ and } \mathbf{z}_2^{(1)} = \infty.$$

Here, the superscripts of $\mathbf{z}$s indicate iteration number.

Execution using *m_k*-means:

In this case, the center vectors will become

$$\mathbf{z}_1^{(1)} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n + \mathbf{z}_1^{(0)}}{m+n+1} \text{ and } \mathbf{z}_2^{(1)} = \mathbf{z}_2^{(0)}.$$

Thus after the first iteration, $\mathbf{z}_1^{(1)} \neq \mathbf{z}_2^{(1)}$.

Now, in a general situation, when the number of clusters K is greater than *2*, we can arrive at the same conclusion according to the following logic.

Let, $\mathbf{z}_1$, $\mathbf{z}_2$, …, $\mathbf{z}_K$ be the K centers, which are initialized to $\mathbf{z}_1^{(0)}, \mathbf{z}_2^{(0)}, \cdots, \mathbf{z}_K^{(0)}$ respectively. In the worst case, we can assume that all the clusters are initialized to the same value, i.e., $\mathbf{z}_1^{(0)} = \mathbf{z}_2^{(0)} = \cdots = \mathbf{z}_K^{(0)} = \mathbf{z}^{(0)}$ (say). Then, after the first iteration, we have

$$\mathbf{z}_1^{(1)} \neq \mathbf{z}^{(0)}$$

and

$$\mathbf{z}_2^{(1)} = \mathbf{z}_3^{(1)} = \cdots = \mathbf{z}_K^{(1)} = \mathbf{z}^{(0)}$$

because all the elements will be occupied by a single cluster. After the second iteration,

$$\mathbf{z}_1^{(1)} \neq \mathbf{z}_2^{(1)} \neq \mathbf{z}^{(0)}$$

and

$$\mathbf{z}_3^{(1)} = \mathbf{z}_4^{(1)} = \cdots = \mathbf{z}_K^{(1)} = \mathbf{z}^{(0)}$$

Therefore, after the Kth iteration, we will get K different center vectors and, hence, all non-empty clusters can be ensured. The above derivation, unfortunately, will not be applicable in either of the following two conditions:

- if all the center vectors are initialized to the centroid of the data set, and
- if any $\mathbf{z}_i^{(j)}$ becomes equal to $\mathbf{z}^{(0)}$ even after getting data elements assigned to it during iteration number j (where, $j<K$).

In the former instance, for a two cluster situation we have, $\mathbf{z}_1^{(0)} = \mathbf{z}_2^{(0)} = \dfrac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n}{m+n}$

Let, in this case, all the elements are assigned to the first cluster keeping the other empty. Then, we have

$$\mathbf{z}_1^{(0)} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n}{m+n}$$

$$\mathbf{z}_1^{(1)} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n + \mathbf{z}_1^{(0)}}{m+n+1} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_m + \mathbf{y}_1 + \mathbf{y}_2 + \cdots + \mathbf{y}_n}{m+n} = \mathbf{z}_1^{(0)}$$

This means that there is no change of the cluster centers even after the ireration is over. The same logic is equally applicable for $K > 2$ also.

Example:

A run of the proposed clustering algorithm is next described for illustration. Let us consider a 1-dimensional data set $\{\mathbf{x}_1 = 12, \mathbf{x}_2 = 2, \mathbf{x}_3 = 7, \mathbf{x}_4 = 5, \mathbf{x}_5 = 19, \mathbf{x}_6 = 14, \mathbf{x}_7 = 8, \mathbf{x}_8 = 18, \mathbf{x}_9 = 17, \mathbf{x}_{10} = 13, \mathbf{x}_{11} = 15, \mathbf{x}_{12} = 9\}$.

The initial cluster centers are intentionally set to the same value 6, i.e., $\mathbf{z}_1 = 6$, $\mathbf{z}_2 = 6$ and $\mathbf{z}_3 = 6$. We shall show how the *m_k*-means algorithm classifies the data set into three clusters *C(1), C(2)* and *C(3)*. It may be noted here that one could have initialized the center vectors to random values also.

The formation of clusters in different iterations of the present *m_k*-means scheme is shown in Table I. It may be noted that the modified clustering scheme takes 11 iterations to converge to a solution with $\mathcal{M}$ (metric) = 20.428574. The partitions become {7, 8, 9}, {2, 5}, and {12, 13, 14, 15, 17, 18, 19}. In this case, the *m_k*-means is found to produce a good clustering. Under a similar initial condition, the *k*-means algorithm fails to cluster data and two empty clusters would be produced.

TABLE I.
ELEMENTS AND CENTERS OF CLUSTERS DURING ITERATIONS

Iter No.	Clus-ters	$\mathbf{z}^{old}$	Elements in Clusters	$\mathbf{z}^{new}$	Metric
0	C(1) C(2) C(3)	6.000 6.000 6.000	2,5,7,8,9,12,13,14,15,17,18,19 -- --	11.154 6.000 6.000	54.6923
1	C(1) C(2) C(3)	11.1546.000 6.000	12,13,14,15,17,18,19,9 2, 5, 7, 8 -	14.239 5.600 6.000	29.0000
2	C(1) C(2) C(3)	14.239 5.600 6.000	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.280 4.200 7.500	20.7799
3	C(1) C(2) C(3)	15.280 4.200 7.500	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.410 3.733 7.857	20.5350
4	C(1) C(2) C(3)	15.410 3.733 7.857	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.426 3.578 7.969	20.4575
5	C(1) C(2) C(3)	15.426 3.578 7.969	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.428 3.526 7.992	20.4361
6	C(1) C(2) C(3)	15.428 3.526 7.992	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.509 7.998	20.4305
7	C(1) C(2) C(3)	15.429 3.509 7.998	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.503 8.000	20.4291
8	C(1) C(2) C(3)	15.429 3.503 8.000	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.501 8.000	20.4287
9	C(1) C(2) C(3)	15.429 3.501 8.000	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.500 8.000	20.4286
10	C(1) C(2) C(3)	15.429 3.500 8.000	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.500 8.000	20.4286
11	C(1) C(2) C(3)	15.429 3.500 8.000	12,13,14,15,17,18,19 2, 5 7, 8, 9	15.429 3.500 8.000	20.4286

IV. EXPERIMANTAL RESULTS

This section provides the performance comparison of the conventional *k*-means and *m_k*-means in terms of rate of convergence and the quality of the solution $\mathcal{M}$. The efficiency of the present *m_k*-means algorithm is then compared with respect to conventional *k*-means algorithm regarding generation of empty clusters.

A. Comparison of k-means and m_k-means Algorithms in Terms of Rate of Convergence and Quality of Solutions

In this subsection, we report the performances of conventional *k*-means and the *m_k*-means algorithms in terms of their rate of convergence and the quality of solutions obtained while executed on different artificial and real life data sets.

Four artificial and four real-life data sets are considered for these experimentation. The artificial data sets are *Circular_5_2, Circular_6_2, Spherical_4_3* and *Elliptical_10_2* [14]. The names imply the structure of underlying clusters, concatenated with the number of clusters present in the data and its dimensions (Table II). For example, in *Circular_5_2* data set the clusters are circular in nature. There are five clusters and 2 dimension. The real-life data sets are *Iris* [20], *Crude_Oil* [21], *Breast_Cancer* and *Color_Moments* [22].

The *Color_Moments* data is derived from 68,040 images of the corel image features collection of the UCI KDD archive [22]. The number of clusters for the *Iris*, *Crude_Oil* and *Breast_Cancer* data sets are known to be 3, 3 and 2 respectively. For the *Color_Moments* data, we assumed the number of clusters to be 10.

For the purpose of comparison, concerned algorithms are executed on various data sets. The results of comparison are given in Table III. The *Circular_5_2* and *Iris* data sets and the corresponding clustered sets (using *m_k*-means) are shown in Figures 2 to 5. The *Circular_6_2, Spherical_4_3* and *Elliptical_10_2* data sets are not shown because they form visually well separate clusters. From Table III, it can be observed that the number of iterations needed for the *m_k*-means algorithm are almost similar to those required for the *k*-means algorithm. However, the *m_k*-means, required a few more iterations in some cases. This is because, the data sets are relatively smaller. Values of the clustering metric that indicate a measure of goodness of clustering are found to be almost similar in all the instances. Here, square root of distances are used to compute metric $\mathcal{M}$..

B. Comparison of k-means and m_k-means Algorithms in Terms of Empty Cluster Generation

Here, we shall show experimentally that the *m_k*-means algorithm is able to overcome the empty cluster problem of the original *k*-means. We do not claim that the *m_k*-means is able to solve the problem entirely. Under certain specific situations, as discussed earlier, it produces empty clusters. However, the possibility of occurrence of such results is very small and condition specific. We chose the *Iris* data set for experimentation with the empty cluster problem in association with the *k*-means and *m_k*-means algorithms, because this data set is considered to be a benchmark for experiments in machine learning, clustering and other related fields. We have also used the *Circular_5_2* data set for comparison.

TABLE II.
DESCRIPTION OF DATA USED

Data Sets	Data Size	Dimension	No. of Clusters
Circular_5_2	250	2	5
Circular_6_2	300	2	6
Spherical_4_3	400	3	4
Elliptical_10_2	500	2	10
Iris	150	4	3
Crude_Oil	57	5	3
Breast_Cancer	683	9	2
Color_Moments	68040	9	10

Simulation experiments are performed on the *Iris* data set by varying the number of desired clusters using the *k*-means and *m_k*-means algorithms. We performed experiments under two different initial conditions as mentioned below:

- **Normal condition**: the center vectors are generated randomly, as usual.
- **Extreme condition**: all the center vectors are identical (initialized to the same value).

It has been observed that using *m_k*-means, in extreme condition, for smaller number of clusters there is no chance of generation of empty clusters. However, as the number of clusters grows, the possibilities of empty clusters also grow due to the reasons mentioned earlier. Results of simulation experiments, over the *Iris* data set, are shown in Table IV. To show goodness or badness of clustering, we have executed both the algorithms (with identical and random initializations) for 3 clusters over the *Iris* data. Results are shown in Table V. Here, good partitions contains (50, 38/39, 62/61) elements, and bad partitions have (96/97, 22/21, 32) elements in 3 clusters as shown in Table VI. The *m_k*-means is found to

TABLE III.
ITERATION NUMBERS AND METRIC VALUES

Data Sets	*k*-means		*m_k*-means	
	Iter No.	Metric	Iter No.	Metric $\mathcal{M}$
Circular_5_2	6	327.530	8	328.457
Circular_6_2	6	374.541	7	374.542
Spherical_4_3	4	749.978	8	749.978
Elliptical_10_2	14	949.389	12	949.345
Iris	5	97.205	10	97.225
Crude_Oil	16	279.743	12	279.743
Breast_Cancer	3	2986.958	6	2986.960
Color_Moments	152	133181.281	96	132938.563

produce all good partitions, when executed for *Circular_5_2* data with 5 clusters (5×50 = 250 elements), for both the above mentioned initial conditions, and no instance of empty or bad clusters were found.

From Table IV it is seen that, if random cluster initialization is used, we do not face the empty cluster problem, not even for a single instance, when the *Iris* data is clustered using the *m_k*-means by varying the number of clusters from 2 to 10. When all the cluster centers are initialized to the same value, we see that the singularity problem do not arise for number of clusters 2 to 6. With higher number of clusters, we encounter increasing number of instances of empty cluster situations. It is to be noted here that, for identical initializations (same value for all centers) the *k*-means fails to partition a data set and all clusters except one remain empty.

TABLE IV.
% OF EMPTY CLUSTERS USING IRIS DATA

Number of Clusters	2	3	4	5	6	7	8	9	10
m_k-means (identical initialization)	0	0	0	0	0	5	15	20	35
m_k-means (random initialization)	0	0	0	0	0	0	0	0	0
k-means (random initialization)	0	0	3	9	12	19	17	32	28

TABLE V.
% CLUSTERING FOR IRIS DATA WITH 3 CLUSTERS

Algorithm	% of Bad partitions	% of Good partitions	% of Empty partitions
m_k-means (identical initialization)	35	65	00
m_k-means (random initialization)	19	81	00
k-means (random initialization)	19	80	01

Although it is seen from Table IV that with identical initialization the *m_k*-means fails when number of clusters is greater than 6, it can be said that for the *Iris* data, which ideally have only 2 or 3 clusters, a value greater than 6 is too high. It is also shown here that in case of normal random initialization, the *k*-means fails to cluster the data set for a moderate number of instances, even for smaller number of clusters. For above experiments, we have executed both the *m_k*-means and the *k*-means for 100 simulation runs for each number of clusters. Therefore, from the experimental results, we can say that the *m_k*-means can handle empty cluster problem very efficiently.

TABLE VI.
GOOD AND BAD PARTITIONS FOR DATA SETS

Instances of Good or bad partitions	Bad partitions	Good partitions
No. of elements in 3 Clusters of *Iris* data	97, 32, 21 or 97, 31, 22 or 96, 32, 22	61, 50, 39 or 62, 50, 38
No. of elements in 5 Clusters of *Circular_5_2* data	No Bad Results obtained	55, 54, 51, 48, 42 or 52, 51, 50, 49, 48 or 53, 52, 50, 48, 47 or 56, 54, 50, 48, 42

V. CONCLUSIONS

This paper proposes a modified version of the well known *k*-means clustering algorithm. The modified algorithm maintains all important characteristic features of the basic *k*-means and at the same time eliminates the possibility of generation of empty clusters. It has been shown that the present algorithm is semantically equivalent to the serial *k*-means algorithm. A detailed comparison of this new algorithm with the basic *k*-means has been reported. Experimental results show that the proposed clustering scheme is able to solve the empty cluster problem, to a great extent, without any significant performance degradation.

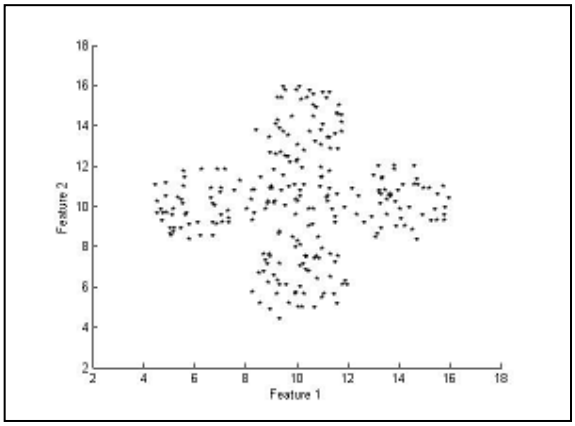

Fig. 2. Original *Circular_5_2* data

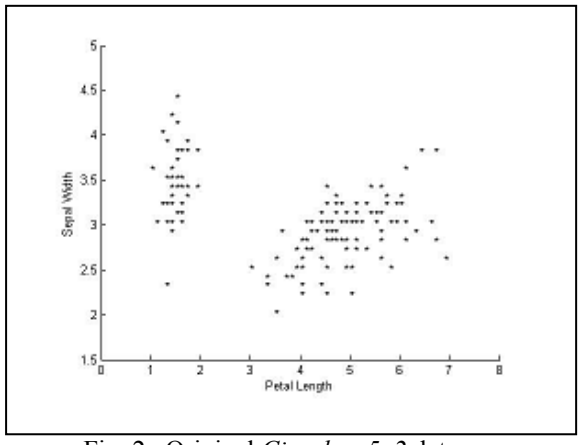

Fig. 3. Original *Iris* data

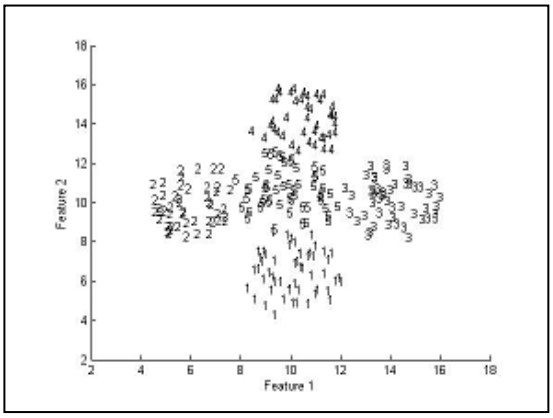

Fig. 4. *Circular_5_2* data clustered by *m_k*-means

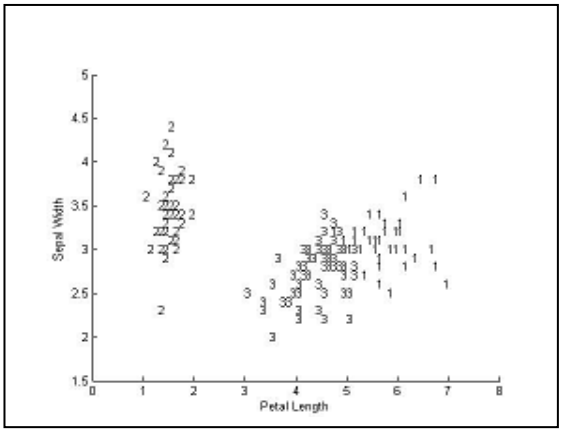

Fig 5. *Iris* data clustered by *m_k*-means

ACKNOWLEDGMENT

This research is partly supported by a sponsored project, Number: 8023/ BOR/ RID/ RPS-109/2007-08, funded by *All India Council for Technical Education* (AICTE), Government of India.

REFERENCES

[1] J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles. Reading: Addison-Wesley, 1974.
[2] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[3] M. R. Anderberg, Cluster Analysis for Application. Academic Press, 1973.
[4] J. B. McQueen, "Some methods of classification and analysis in multivariate observations," in *Proc. Of fifth Barkley symposium on mathematical statistics and probability*, pp. 281 - 297, 1967.
[5] S. Z. Selim and M. A. Ismail, "K-means type algorithms: a generalized convergence theorem and characterization of local optimality," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 6, No. 1, pp. 81--87, 1984.
[6] H. Tsai, S. Horng, S. Tsai, S. Lee, T. Kao, and C. Chen. "Parallel clustering algorithms on a reconfigurable array of processors with wider bus networks," in *Proc. IEEE International Conference on Parallel and Distributed Systems*, 1997.
[7] S. Kantabutra and A. Couch. "Parallel K-means clustering algorithm on NOWs," in *NECTEC Technical Journal*, Vol. 1, No. 6, pp. 243–247, 2000.
[8] A. K. Jain, R. P. Duin, and J. Mao, "Statistical Pattern Recognition:A Review," in *IEEE Transaction on Pattern Analysis and Machine Intelligence* , vol. 22, No. 1, pp. 04--37, 2000.
[9] R. Jin, A. Goswami and G. Agarwal, "Fast and Exact Out-of-Core and Distributed K-Means Clustering," in *Knowledge Information Systems*, vol. 10, pp. 17-40, 2006.
[10] I. S. Dhillon and D. S. Modha, "A Data-Clustering Algorithm on Distributed Memory Multiprocessors," in *Proceedings of KDD-WS on High Performance Data Mining*, 1999.
[11] S. J. Selim and K. Al-Sultan, "A simulated annealing algorithm for the clustering problem," in *Pattern Recognition* , vol. 24, pp. 1003-1008, 1991.
[12] S. Bandyopadhyay, U. Maulik, and M. K. Pakhira, "Partitional clustering using simulated annealing with probabilistic redistribution," in *International Journal Pattern Recognition and Artificial Intelligence*, vol. 15, pp. 269--285, 2001.
[13] U. Maulik and S. Bandyopadhyay, "Genetic algorithms based clustering technique," in *Pattern Recognition*, vol. 33, pp. 1455-1465, 2000.
[14] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "A Study of Some Fuzzy Cluster Validity Indices, Genetic Clustering and Application to Pixel Classification," in *Fuzzy Sets and Systems*, vol. 155, pp. 191--214, 2005.
[15] P. S. Bradley and U. M. Fayyad, "Refining Initial Points for K-means Clustering," in *Technical Report* of Microsoft Research Center, Redmond,California, USA, 1998.
[16] F. X. Wu, "Genetic weighted k-means algorithm for clustering large-scale gene expression data," in *BMC Bioinformatics*, vol. 9, 2008.
[17] A. Likas, Vlassis and J. J. Verbeek, "The global k-means clustering algorithm," in *Pattern Recognition* , vol. 36, no. 2, pp. 451-461, 2003.
[18] S. Deelers and S. Auwatanamongkol, "Enhancing k-means algorithm with initial cluster center derived from data partitioning along the data axis with the highest variance," in *International Journal of Computer Science*, vol. 2, no. 4, pp. 247-252, 2007.
[19] S. S. Khan and A. Ahmed, "Cluster center initialization for K-means algorithm," in *Pattern Recognition Letters*, vol. 25, no. 11, pp. 1293-1302, 2004.
[20] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 3, pp. 179–188, 1936.
[21] R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis. Prentice-Hall, 1982.
[22] Machine Learning Database available at http://kdd.ics.uci.edu.