International Journal of Machine Learning and Cybernetics (IJMLC) manuscript No.
(will be inserted by the editor)

A hybrid approach to speed-up the *k*-means clustering method

T. Hitendra Sarma · P. Viswanath · B. Eswara Reddy

Received: date / Accepted: date

Abstract *k*-means clustering method is an iterative partition based method which for finite data-sets converges to a solution in a finite time. The running time of this method grows linearly with respect to the size of the data-set. Many variants have been proposed to speed-up the conventional *k*-means clustering method. In this paper, we propose a prototype based hybrid approach to speed-up the *k*-means clustering method. The proposed method, first partitions the data-set into small clusters (grouplets), which are of varying sizes. Each grouplet is represented by a prototype. Later, the set of prototypes is partitioned into k clusters using the modified *k*-means method. The modified k-means clustering method is similar to the conventional k-means method but it avoids empty clusters(the clusters to which no pattern is assigned) in the iterative process. In each cluster of prototypes, each prototype is replaced by its corresponding set of patterns (which formed the grouplet) to derive a partition of the data-set. Since this partition of the data-set can deviate from the partition obtained using the conventional *k*-means method over the entire data-set, a correcting step is proposed. Both theoretically and experimentally the conventional *k*-means method and the proposed hybrid method (augmented with the correcting step) are shown to yield the same result (provided, the initial k seed points are same). But, the proposed method is much faster than the conventional one. Experimentally, the proposed method is compared with the conventional method and the other recent methods that are proposed to speed-up the k-means method.

Keywords *k*-means clustering method · prototypes · hybrid clustering · leaders clustering method

T. Hitendra Sarma · P. Viswanath
Department of Computer Science and Engineering, Rajeev Gandhi Memorial College of Eng. and Technology-Nandyal 518501,A.P., India.
E-mail: {hitendrasarma, viswanath.p}@ieee.org

B. Eswara Reddy
Department of Computer Science and Engineering, JNTUA College of Engineering, Anantapur-515002, A.P.,
India.
E-mail: eswarcsejntu@gmail.com

1 Introduction

Data clustering is a process of identifying the natural grouping that exist in a given data-set, such that the patterns in the same cluster are more similar and the patterns in different clusters are less similar. It has been considered as an important tool in various applications like biology, marketing, information retrieval, remote sensing, statistics, pattern recognition, image processing, text mining, *etc.*, [20] [17]. Clustering algorithms can be broadly divided into two groups, *viz., hierarchical* and *partitional* [20]. Hierarchical clustering algorithms recursively find nested clusters either in agglomerative (bottom-up) mode or in divisive (top-down) mode while partitional clustering algorithms find all the clusters simultaneously as a partition of the data-set [48] [20]. Apart from this, the clustering methods can also be categorized into *density based methods, grid based methods, model based methods, etc.*, [17] [21]. In case of datasets with high dimensionality, the subspace clustering methods are proved to be efficient for mining clusters in different subspaces [33]. For further comprehensive study on various clustering methods refer [19], [22], [5] and [18].

The most popular, simplest and efficient partitional clustering method is the *k*-means clustering method [31]. Recently, it has been identified that the *k*-means method is one of the most influential algorithms that have been widely used in the data mining community [47]. It is generally effective in producing good results in various applications [31] [20]. An iterative procedure for the *k*-means problem is proposed

by Llyod [29]. Llyod's *k*-means method partitions the data-set into k clusters, where each cluster is represented by its mean. Recently, Liang *et. al.*, proposed a k-means like clustering approach for clustering data in 2D space, but uses a steiner point or curvature-centroid [15] as a representative of each cluster [23]. Further, some variants of the conventional k-means method are successfully applied in soft subspace clustering [14]. The time complexity of this method is proportional to the product of the number of patterns in the data-set, dimensionality of the data-set, number of clusters required and the number of iterations performed till convergence. In this iterative process there may be some clusters to which no pattern is assigned. Such clusters are called empty clusters. Calculating the mean of such empty clusters is not possible. Empty cluster problem is one of the drawbacks of the k-means method. This problem can be avoided by taking each old mean pattern as one of the patterns in its new cluster, in each iteration [32]. However, if the data-set can not fit in the primary memory then it needs to be brought from the secondary memory several times. This is the typical case with a large data-set. Hence, this method is a slower one, especially for large data-sets [25]. Several attempts were made by researchers to make the *k*-means method efficient. This paper proposes a simple hybrid scheme to speed-up the *k*-means clustering method.

In the iterative process of the *k*-means method, it is necessary to compute the distance between each pattern to each one of the k means, in each iteration. Here, the distance computation is the time consuming step. The computational cost of *k*-means method depends on the number of distance computations. One approach to reduce the overall computational cost of the *k*-means method is to reduce the number of distance computations [9]. This is the motivating step of our proposed method. To reduce the number of distance computations, instead of considering each pattern in the data-set, the proposed method works only with a few selected prototypes. The proposed method is briefly outlined as follows. It runs in two stages. First, the data-set is partitioned into small clusters of varying sizes, called grouplets. Each grouplet is represented by a prototype. The set of prototypes is partitioned using the modified *k*-means method [32]. Finally, a correcting step is performed to get the clustering result which is same as that is obtained using the conventional method which uses the entire data-set in each iteration. In most of the cases the correcting step is not required. However, the concept of deriving the grouplets of varying sizes helps to speed-up this correcting step. Both theoretically and experimentally the proposed method has been shown to yield the same result as the conventional *k*-means method, provided the initial cluster centers are same, but in a reduced time. Recently, Braga *et. al.*, have proposed a similar kind of prototype based hybrid approach to speed-up SVM learning, where the prototypes are generated by using the conventional k-means method [1].

A preliminary version of the present paper was presented at a conference [42]. The present paper has several improvements, which are both theoretical and experimental, and are enumerated below.

1. Theoretically it is shown that the proposed method produces the same clustering result as that of the conventional k-means method (see Theorem 1, Lemma 1 and Theorem 2).
2. Grouplets of varying sizes, which are automatically generated, *i.e.*, without the need of any input parameter (like τ in [42], which is given as input, which does not vary)
3. Avoiding the empty cluster problem.
4. Rigorous experimental study using several real world and synthetic data-sets.
5. Comparison between the present method and the method proposed in [42], which clearly establishes the superiority of the method proposed in this paper.

The remaining paper is organized as follows. Section 2 briefly reviews the *k*-means and modified k-means clustering methods. Section 3 describes some of the existing techniques to speed-up the *k*-means clustering method. Section 4 gives an outline of the leaders clustering method and its modified version proposed in this paper. The proposed hybrid method is described in section 5. Experimental results are given in Section 6, and Section 7 gives some of the conclusions.

2 The *k*-means clustering method and its modified version

This section briefly reviews the *k*-means method and its modified version, which avoids the empty cluster problem.

k-means clustering method is an iterative method. It partitions a data-set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ into k clusters such that all patterns in a given cluster are closest to the center of that cluster. Each cluster is represented by its mean (centroid). The means are found such that the criterion $J = \sum_{i=1}^{n} ||x_i - m(x_i)||^2$ is minimized, where $m(x_i)$ is the nearest mean to the pattern x_i [27] [31]. The iterative process includes the following steps:

– Assign each pattern to its nearest mean, and
– Compute the means of the new clusters.

The iterative process stops when cluster centers (means) become stable for two consecutive iterations. The time complexity of *k*-means method is $O(nkdt)$, where n is the size of the data-set, and k is the number of clusters derived, d is the dimensionality of the data-set and t is the number of iterations taken till convergence.

The running time of this method grows linearly with respect to the size of the data-set. The method performs well for small data-sets, but it may produce empty clusters. Empty clusters are the clusters to which no pattern is assigned [32]. Empty cluster problem can be solved by taking each old mean pattern as one of the patterns in its corresponding new cluster, in each iteration. That is, if $m_j^{(r-1)}$ is the mean of the j^{th} cluster in $(r-1)^{th}$ iteration, then mean of the j^{th} cluster in r^{th} iteration *i.e.,* $m_j^{(r)}$ is calculated as follows.

$$m_j^{(r)} = \frac{\sum_{x_i \in C_j^{(r)}} x_i + m_j^{(r-1)}}{|C_j^{(r)}|+1}. \quad (1)$$

where $C_j^{(r)}$ is the j^{th} cluster in r^{th} iteration.

The k-means method along with this modification is given in the Algorithm 1. Malay *et.al.,* proved that Algorithm 1 will converge to the same means as the conventional k-means method which will not use old means as patterns in their new clusters [32]

Algorithm 1 Modified k-means($\mathcal{D}$, k)

1. Randomly choose k patterns from $\mathcal{D}$ as the initial seed points. Let this be $M^{(0)} = \{m_1^{(0)}, m_2^{(0)}, \ldots, m_k^{(0)}\}$.
2. Let $r = 0$; /* r is the iteration number */
repeat
3. $r = r + 1$;
4. Form k clusters by assigning each pattern in $\mathcal{D}$ to its nearest mean in $M^{(r-1)}$.
5. Find new centroids (means) of the k clusters, i.e., $M^{(r)} = \{m_1^{(r)}, m_2^{(r)}, \ldots, m_k^{(r)}\}$, where each $m_j^{(r)}$ is calculated using equation (1).
until $(M^{(r)} == M^{(r-1)})$
6. Output $M^{(r)}$.

This method is a slower one in case of large data-sets. So many techniques are proposed in the literature to speed-up the *k*-means method for large data-sets, some of them are briefly reviewed in the next Section.

3 Review of existing methods to speed-up the *k*-means clustering method

The methods proposed to speed-up the *k*-means clustering method can be divided in to two categories. First, the methods which produces approximate solution. Second, the methods which produces the same solution as obtained using the conventional *k*-means clustering method.

There are several approximate methods like single pass *k*-means method which scans the data-set only once to produce the clustering result [20]. However the result can considerably deviate from the result obtained by the conventional method. Another approach to speed-up the *k*-means clustering method is by bootstrap averaging. The approach of bootstrap averaging applies *k*-means method on small bootstrapped samples of the data-set, till convergence, and finally averaging the similar cluster centers to obtain a single model that contains *k* clusters [10]. Other approximate methods are to keep important points in a buffer while discarding unimportant points as done by Bradley *et. al.* [8]. Later Fredrik *et. al.* [12] improved this idea to speed-up *k*-means method. Domingos *et. al.* [11] proposed a fast *k*-means algorithm which gives a better approximate solution, with statistically bounded loss in the clustering quality.

There are other improvements to speed-up the *k*-means method without compromising with the the quality. Alsabti *et. al.* [2] proposed a new technique to reduce the computational cost of the method, where the patterns are organized in a data-structure called *kd*-tree such that the patterns that are closest to a cluster center can be found in a reduced time. Further, Pelleg *et. al.* [34] [35] used the same data-structure to store the information of the distances from patterns to the cluster centers, which can make each iteration of *k*-means significantly faster. Kanungo *et. al.* [26] proposed a refined and better analyzed method over these methods called the *filtering algorithm*. It proposes a small variation of the *kd*-tree called *balanced box-decomposition tree (BBD-tree)* to reduce the number of centers to be searched for a set of points which are enclosed in a hyper-rectangle. Further, several researchers concentrated on reducing the number of distance computations to speed-up the *k*-means method. Fahim *et al.* [38] proposed an efficient method of finding the nearest cluster center for each pattern more quickly by using the information from its previous iteration. If a pattern has been identified to belong to the same cluster for few successive iterations then further distance computations to find its nearest mean are avoided, thereby reducing the unnecessary computational cost. Later, Abdul Nazeer *et. al.* [24] used a variant of the method presented in [38] to speed-up the iterative process. Recently, Shi Na *et. al.* [39] used the similar concept and along with that two simple data-structures are used for storing and passing useful information from one iteration to its next iteration. Several other efficient versions of *k*-means includes, Genetic *k*-means [28] and Fast Genetic *k*-means [30], CoreMeans [13], Fast and Exact K-means clustering method (FEKM) [37], Compare-means method and Sort-means methods [41]. Recently Hitendra *et.al.,* [42] proposed a prototype based hybrid approach, called *lk*-means clustering method, to speed-up the *k*-means clustering method. The method first partitions the data-set into a number of small clusters, where all clusters are of same size. Each small cluster (grouplet) is represented by a prototype. Then all these small clusters are again grouped into *k* clusters by applying *k*-means method over the set of prototypes. In this approach, grouplets derived are of same size and is given as an input parameter. This paper presents

an extended and critically analyzed version of this prototype based hybrid method, where grouplets of varying sizes are automatically found. More recently Bidyut *et. al.,* proposed a similar kind of hybrid approach to speed-up the single-link clustering method [6].

4 Generating Prototypes

Prototypes are generated using a fast clustering method called the leaders clustering method [40].

4.1 The leaders clustering method [40]

Leaders method scans the data-set only once to find a partition of the given data-set. This method is briefly outlined for the sake of completeness. For further extensive study and the properties of the leaders method refer [44], [43] and [45]. Each cluster is represented by a patten called *leader* and all other patterns in the cluster are its *followers*. For a given threshold distance τ, the leaders method works as follows. It maintains a set of leaders $\mathcal{L}$, which is initially empty and is incrementally built. For each pattern x in the data-set $\mathcal{D}$, if there is a leader $l \in \mathcal{L}$ such that distance[1] between x and l is less than or equal to τ, then x is assigned to the cluster represented by l. In this case, we call x as a *follower* of the leader l (l is a follower to itself). Note that even if there are many such leaders, only the first one which is at a distance less than or equal to τ is chosen. If there is no such leader then x itself becomes a new leader and is added to $\mathcal{L}$. The algorithm outputs the set of leaders $\mathcal{L}$. Along with the leaders, some additional statistics of each grouplet are required in our hybrid approach. Hence the leaders method is modified. The following subsection explains the proposed modified leaders clustering method.

4.2 Modified Leaders Clustering Method

Following are the modifications of the leaders method proposed in this paper.

- The leaders clustering method is modified to derive clusters of varying size using varying threshold τ, instead of a fixed τ.
- Each leader l has its own threshold value, $\tau(l)$. The threshold values are calculated by applying *k*-means on a small random sample of $\mathcal{D}$ (which will be explained later in the following sections).
- For a pattern x, among the leaders that are less than or equal to the distance $\tau(l)$ from it, the leader that is at the least distance is chosen. In earlier versions of the leaders clustering method [42], a leader, which is at a distance less than or equal to the threshold that is encountered first is chosen. This modification is needed, because of the variable threshold values used in the current method.
- To store along with leaders, their threshold values, count, which is the number of patterns that are present in its cluster (this is similar to that done in [44]) and linear sum of all followers.

The output of the modified method is the set of leaders along with their threshold values, number of followers(count), linear-sum of followers of each leader. The data-set is rearranged and stored according to these clusters. This is done in a way that, given a leader l and its threshold $\tau(l)$, it should be easy to extract all its followers, without scanning the data-set from beginning to end. That is, the data-set has to be stored indexed with respect to leaders.

For each leader l its threshold value $\tau(l)$ is found as follows. First, a random sample $\mathcal{S}$ of a small size is derived from the data-set $\mathcal{D}$. Efficient algorithms for drawing a sample randomly from data-set are proposed in [46]. *k*-means method is applied on this random sample and the set of converged means *i.e.,* $M^{(\mathcal{S})} = \{m_1^{(\mathcal{S})}, m_2^{(\mathcal{S})}, \ldots, m_k^{(\mathcal{S})}\}$ is obtained. The converged means need not be same as obtained by using the entire data-set. However the deviation between these two results can be reduced by selecting the sample which is of moderate size. Theoretically it is proved that by using chernoff bounds, we can derive values for sample sizes for which the deviation is low [16].

Let x be the pattern which is going to become the new leader and $m_i^{(\mathcal{S})}$ and $m_j^{(\mathcal{S})}$ be the first two nearest means of x. First we compute the values of a and b as given below.

$$a = \frac{||m_i^{(\mathcal{S})} - m_j^{(\mathcal{S})}||}{2} \tag{2}$$

$$b = x \cdot \frac{(m_i^{(\mathcal{S})} - m_j^{(\mathcal{S})})}{||m_i^{(\mathcal{S})} - m_j^{(\mathcal{S})}||} \tag{3}$$

where a is half of the distance between the two means $m_i^{(\mathcal{S})}$ and $m_j^{(\mathcal{S})}$ and b is the distance between the first nearest mean of x and the image of x projected onto the line joining the means $m_i^{(\mathcal{S})}$ and $m_j^{(\mathcal{S})}$. Finally, the threshold of the new leader x *i.e.,* $\tau(x)$ is

$$\tau(x) = \rho(a - b) \tag{4}$$

where ρ is a positive constant. In the paper ρ used is 0.5. The modified version of leaders method is given in the Algorithm 2.

[1] For the sake of simplicity, we assume that the patterns are from a Euclidean space and Euclidean distance is used. Whereas the proposed methods are applicable with any distance metric.

Algorithm 2 Modified-Leaders($\mathcal{D}$)

$\mathcal{L} = \emptyset$;
for each $x \in \mathcal{D}$ **do**
  Find a $l \in \mathcal{L}$ which is nearest to x, such that $||l - x|| \leq \tau(l)$
  if there is no such l or when $\mathcal{L} = \emptyset$ **then**
    find the threshold value of the new leader x i.e., $\tau(x)$ using equation (4).
    $\mathcal{L} = \mathcal{L} \cup \{x\}$;
    $count(x) = 1$;
    linear-sum$(x) = x$;
    $followers(x) = \{x\}$;
  else
    $count(l) = count(l) + 1$;
    linear-sum$(l) =$ *linear-sum*$(l) + x$;
    $followers(l) = followers(l) \cup \{x\}$;
  end if
end for
Output:
$\mathcal{L}^* = \{< l, \tau(l), count(l),$ *linear-sum*$(l), followers(l) > |l$ is a leader $\}$.

5 Proposed prototype based hybrid approach

This section presents our prototype based hybrid approach to speed-up the *k*-means method. The proposed method is called the *lk*-means clustering method with Varying Threshold (*lk*-means-CMVT). Preliminary version of this approach, called *lk*-means clustering method was presented at a conference [42]. For the sake of convenience in the comparative study, the earlier proposed method is referred as *lk*-means clustering method with fixed threshold (*lk*-means-CMFT). Before explaining our proposed method, first, the earlier version, *i.e.*, *lk*-means-CMFT [42] is explained in brief.

5.1 *lk*-means Clustering Method with Fixed Threshold (*lk*-means-CMFT) [42]

lk-means-CMFT runs in two stages *viz.*, *lk*-means-CMFT-*first-stage* and *lk*-means-CMFT-*correcting-stage*. First the leaders method is applied over the given data-set to divide the data-set into small grouplets. Each grouplet is represented by a prototype called leader. The set of leaders along with some additional statistics like: the number of followers of each leader(count), the linear-sum of the followers of each leader are also computed in a single-scan. Now, the *lk*-means-CMFT-*first-stage* basically runs the Lloyd's k-means method, but using the set of leaders, their count and linear-sum values. The method is iterated till it converges. A partition of the data-set can be obtained by assigning all the followers to the clusters to which their leader belongs. But there are some leaders whose all followers may not be assigned to the cluster to which it belongs. Next section explains this scenario in detail. Later, the second stage, called the *lk*-means-CMFT-*correcting-stage*, reassigns the followers of such conflicting leaders to their nearest clusters and the method iterates till convergence.

The threshold value τ which is used to generate the grouplets has to be given as an input parameter. The running time of this method is highly influenced by this parameter. Experimentally it is analyzed on various bench mark data-sets, that, if the value of τ increases then the running time of the first stage decreases but the running time of the correcting stage increases. The concept of varying threshold proposed in our method will speed-up both first stage and the correcting stages. The following subsection presents our proposed method in detail.

5.2 *lk*-means Clustering Method with Varying Threshold (*lk*-means-CMVT)

The proposed method, *lk*-means-CMVT, runs in two stages. In the first stage, called *lk*-means-CMVT-*first-stage*, the data-set is divided in to a number of grouplets by employing the modified leaders clustering method as given in Algorithm 2. Later, the modified k-means clustering method is executed using leaders set, their count and linear-sum values, to derive a partition of the leaders set. Then, a second stage, called the *lk*-means-CMVT-*correcting-stage* is done to see that the results are not deviated from that of the *k*-means method which uses the entire data-set. Finally each leader is replaced by its grouplet to obtain a partition of the data-set. These two steps along with their formal properties are described as follows.

5.2.1 lk-means-CMVT-*first-stage*

This is same as the modified k-means method, but is applied using the leaders set (instead of the entire data-set). The method starts with initially chosen random patterns from the data-set as its seed-points, and in each iteration, each leader is assigned to the nearest mean pattern to form the k clusters of leaders. The new means (*i.e.*, the new centroids) of each cluster (cluster of leaders) is found as follows. Let $l_1, l_2, \ldots, l_p$ be the leaders in j^{th} cluster in r^{th} iteration *i.e.*, $C_j^{(r)}$. Then its new centroid is

$$m_j^{(r)} = \frac{\sum_{j=1}^{p} \textit{linear-sum}(l_j) + m_j^{(r-1)}}{\sum_{j=1}^{p} count(l_j) + 1}.$$

The method is iterated till it converges. Theorem 1 formally establishes that the first-stage definitely converges.

Theorem 1 *lk-means-CMVT-first-stage converges.*

Proof: Create a synthetic data-set $\mathcal{D}'$ as described. For all leaders (that are derived according to the modified-leaders method given in Algorithm 2), add each leader l to $\mathcal{D}'$ $count(l)$ times. So, the size of $\mathcal{D}$ and $\mathcal{D}'$ are same. Now, take the k

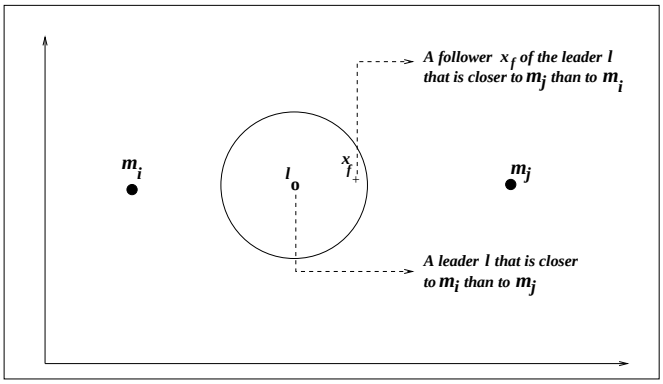

Fig. 1 Reason for deviation in the result

seed points exactly as that taken by the *lk*-means-CMVT-*first-stage*, but run the conventional *k*-means algorithm using the data-set $\mathcal{D}'$. Since the conventional *k*-means algorithm working with data-set $\mathcal{D}'$ converges [7], the modified k-means algorithm also converges [32]. Hence *lk*-means-CMVT-*first-stage* also converges, because both are same □

lk-means-CMVT-*first-stage* derives a partition of the set of leaders $\mathcal{L}$. By replacing each leader by the set of its followers, we get a partition of the data-set $\mathcal{D}$. Let this partition be π^l, and that obtained by employing the conventional *k*-means method over the entire data-set (keeping the seed points same) be π. Now, π^l need not be same as π, because there may be a leader l which is assigned to a mean m_i (according to *lk*-means-CMVT-*first-stage*), but a follower of l may be actually closer to some other mean m_j such that $m_i \neq m_j$. See Figure 1 which illustrates this. Let x_f be a follower of the leader l. So, this situation arises when $||l - m_i|| < ||l - m_j||$, but $||x_f - m_i|| > ||x_f - m_j||$. The pattern x_f according to the conventional k-means method should be included with the cluster for which m_j is the mean, but *lk*-means-CMVT-*first-stage* assigned this to the mean m_i.

To characterize these kinds of errors formally, we define safe-leader and border-leader as follows.

Definition 1 Safe-leader: A leader l is said to be a safe-leader, if l is assigned to a mean m_i according to lk-*means-CMVT*-first-stage, and there is no another mean m_j such that $||l - m_j|| - \tau(l) \leq ||l - m_i||$.

Definition 2 Border-leader: A leader l is said to be a border-leader, if l is assigned to a mean m_i according to lk-*means-CMVT*-first-stage, but there is another mean m_j such that $||l - m_j|| - \tau(l) \leq ||l - m_i||$.

The concept of safe and border leaders is illustrated in the Figure 2. In Figure 2 there are four leaders, say l_1, l_2, l_3, l_4. Let a_1, a_2, a_3 and a_4 are the thresholds of the leaders

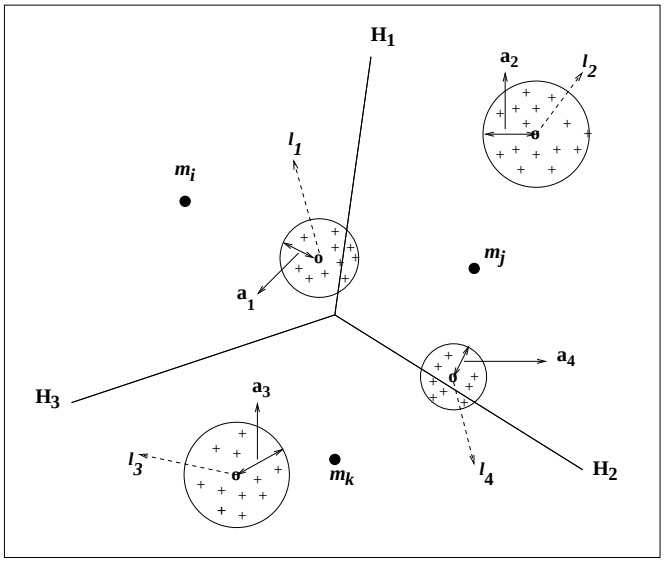

Fig. 2 Safe-leaders and border-leaders

l_1, l_2, l_3, l_4 respectively. *i.e.*, $a_1 = \tau(l_1)$, $a_2 = \tau(l_2)$, $a_3 = \tau(l_3)$, and $a_4 = \tau(l_4)$. Here H_1,H_2 and H_3 are the lines which bisects the lines joining m_i and m_j, m_j and m_k, and m_k and m_i, respectively. Therefore it forms three voronoi regions corresponding to m_i, m_j and m_k.

Since l_2 belongs to m_j according to lk-*means-CMVT*-first-stage and there is no other mean m (where m is either m_i or m_k) satisfying the inequality $||l - m|| - \tau(l_2) \leq ||l - m_j||$. By the definition of safe-leader, l_2 is a safe leader. Similarly l_3 is also a safe leader which belongs to m_k. Where as the leader l_1 is a border leader, since l_1 belongs to m_i according to lk-*means-CMVT*-first-stage but there exists another mean m_j such that $||l - m_j|| - \tau(l_1) \leq ||l - m_i||$. Similarly l_4 is a border leader.

The set of such border-leaders can be found as part of last iteration of the first-stage. If the set of border-leaders is empty, then the result of *lk*-means-CMVT-*first-stage* (*i.e.*, the partition π^l) is same as that obtained by the conventional k means method which uses the entire data-set $\mathcal{D}$ (*i.e.*, the partition π). To formally establish this (Theorem 2) the following Lemma is used.

Lemma 1 *Let $\pi = \{B_1, B_2, \ldots, B_k\}$ be a partition of the data-set $\mathcal{D}$. Let m_i be the centroid for the block B_i, for $i = 1$ to k. Then π is a solution to which the conventional k-means method converges if and only if for each pattern x, if x is in block B_i, then $||x - m_i|| \leq ||x - m_j||$ for all $j \neq i$.*

Proof: The criterion $J = \sum_{i=1}^{n} ||x_i - m(x_i)||^2$ is minimized only when $m(x_i)$ is the nearest mean to x_i, otherwise the Algorithm goes to the next iteration where it moves x_i to the cluster corresponding to its nearest mean □

Theorem 2 *If the set of border-leaders is empty, then the clustering result obtained by lk-means-CMVT-first-stage, where*

each leader is replaced by its followers, is same as that obtained by the conventional k-means method which uses the entire data-set, provided the initial seed points are same in both cases.

Proof: Let $\pi^l = \{B_1^{(l)}, B_2^{(l)}, \ldots, B_k^{(l)}\}$ be the partition of the data-set obtained using *lk*-means-CMVT-*first-stage*. Initially *lk*-means-CMVT-*first-stage* gives a partition of the leaders set, and by replacing each leader by its followers we get π^l. If $l_1, l_2, \ldots, l_p$ are the leaders in a cluster(block). Then its centroid is obtained by

$$\frac{\sum_{j=1}^{p} \textit{linear-sum}(l_j)}{\sum_{j=1}^{p} count(l_j)}.$$

This is the actual centroid of the cluster consisting of patterns followers(l_1)$\cup$followers(l_2)$\cup\ldots\cup$followers(l_p). Since there are no border leaders, for all patterns in this cluster its centroid is the nearest mean (otherwise there is a border leader). Since this is true for each cluster of π^l, according to Lemma 1, π^l is a solution obtained by the conventional k-means method. Since the conventional k-means method and *lk*-means-CMVT-*first-stage* are initialized with same seed points, their solutions are also same □

5.2.2 lk-means-CMVT-*correcting-stage*

If the set of border leaders is non-empty, then π^l and π need not be same. A correcting step is applied over the result of *lk*-means-CMVT-*first-stage* in order to get the same clustering result as that would have obtained by using the conventional k-means method. Only the followers of border leaders are reconsidered in the correcting stage as given in Algorithm 3. Each border leader l is removed from its cluster and each follower of this leader l is reconsidered as an individual pattern and assigned to its nearest mean. To avoid empty clusters, through out the iterative process old mean patterns are reconsidered in their new clusters in the next iteration. The process is repeated till convergence. In this correcting stage the proposed method *lk*-means-CMVT gets advantage over the earlier version *i.e.,* *lk*-means-CMFT.

In the *lk*-means-CMFT, all leaders are having a fixed threshold, *i.e.,* the threshold values of the safe-leader and border-leader are same. The number of border leaders depends on the threshold value *i.e.,* τ. As we are considering only those patterns which are the followers of the border leaders, the value of τ plays a significant role in the total running time of *lk*-means-CMFT. If the value of τ is zero then each pattern becomes a leader and hence the running time of *lk*-means-CMFT-*first-stage* is equal to the conventional *k*-means method and the second stage can be avoided as there will be no border leader. If the value of τ is large then the time required for the *lk*-means-CMFT-*first-stage* will be less, but the number of border leaders will increase and hence the running time for the correcting step is high.

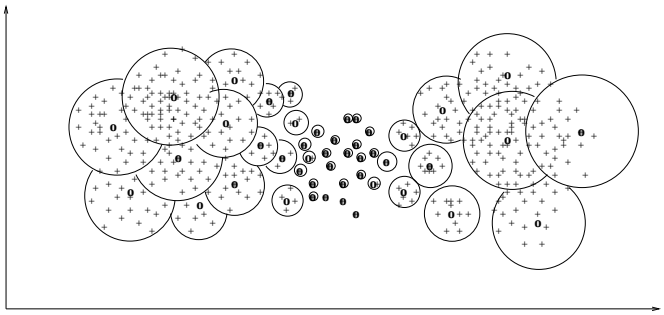

Fig. 3 Leaders with variable threshold values

The present method *lk*-means-CMVT, applies the modified leaders method as given in Algorithm 2, to reduce the overall running time. Algorithm 2 generates leaders of varying sizes such that the threshold values of the safe-leaders are larger when compared to the threshold values of the border leaders. Hence the number of followers of the safe-leaders becomes more and the number of followers of the border leaders becomes less. As a result the time required for the *lk*-means-CMVT-*first-stage* as well as for the *lk*-means-CMVT-correcting-stage will be reduced. There by reducing the overall running time. Figure 3 gives an illustration of the leaders of varying thresholds after the first stage.

Experimentally it is shown that our method, which uses varying threshold will take reduced time for the correcting stage when compared to that the previous method, which uses a fixed threshold.

5.3 Time complexity analysis of *lk*-means-CMVT

The time complexity of the conventional *k*-means method is $O(dnmk)$, where d is the dimensionality of the data-set, n is the size of the data-set, m is the number of iterations taken till convergence and k is the number of clusters derived.

In the proposed method, the leaders set is derived in a single scan of the data-set. The data-set is re-stored indexed w.r.t the leaders, so that the followers of border leaders can be easily extracted which are needed in the correcting stage. *lk*-means-CMVT-*first-stage* works with the leaders set. The number of leaders depends on the varying threshold value τ, The threshold for each new leader can be fixed using the equation (4). The set of leaders is typically very small (when compared with the entire data-set) which totally fits in the primary memory in most of the cases (according to the experimental studies performed).

Let the number of leaders be r. Then the time complexity of the *lk*-means-CMVT-*first-stage* is equal to $O(drm'k)$, where m' is the number of iterations in the *lk*-means-CMVT-*first-stage*. The *lk*-means-*correcting-stage* depends on the set of border leaders. Its time complexity is $O(dr_bm''k)$, where r_b is the number of border leaders, m'' is the number

Algorithm 3 *lk*-means-*correcting-stage*($M^{(0)}$)

/* $M^{(0)} = \{m_1^{(0)}, m_2^{(0)}, \ldots, m_k^{(0)}\}$ is the set of means that is obtained as output of *lk*-means-*first-stage* */
1. Assign each leader to its nearest mean in $M^{(0)}$ to obtain a partition of the set of leaders which is $\pi_{\mathcal{L}} = \{L_1, L_2, \ldots, L_k\}$.
2. For $j = 1$ to k, find *Total-Sum*(L_j) = $\sum_{\forall l \in L_j}$ *linear-sum*(l), and *Total-Count*(L_j) = $\sum_{\forall l \in L_j}$ *count*(l);
if there is any L_j to which no leader is assigned **then**
  the corresponding $m_j^{(0)}$ of such cluster is reconsidered into L_j.
  /* *i.e.,* *Total-Sum*(L_j)=$m_j^{(0)}$ and *Total-Count*(L_j)=1;*/
end if
3. $i = 0$.
repeat
  4. $i = i + 1$;
  5. Find the set of border leaders.
  for each border leader l **do**
    6. Find nearest mean for l in $M^{(i-1)}$. Let this be $m_l^{(i-1)}$ which corresponds to the cluster L_l.
    7. *Total-Sum*(L_l) = *Total-Sum*(L_l) $-$ *linear-sum*(l);
    8. *Total-Count*(L_l) = *Total-Count*(L_l) $-$ *count*(l);
    for each $x \in$ *follower*(l) **do**
      9. Find the nearest mean of x in $M^{(i-1)}$. Let this be $m_x^{(i-1)}$ and the corresponding cluster be L_x;
      10. *Total-Sum*(L_x) = *Total-Sum*(L_x) + x;
      11. *Total-Count*(L_x) = *Total-Count*(L_x) + 1;
    end for
  end for
  12. Find new means,
  $M^{(i)} = \{$*Total-Sum*(L_j)/*Total-Count*$(L_j) \mid j = 1$ to $k\}$;
until $(M^{(i)} == M^{(i-1)})$
13. Output $M^{(i)}$.

of iterations of this stage and this is typically very less compared with m or m'. Hence the total time complexity of the proposed method is $O(n + drm' + dr_b m''k)$. Whereas the filtering approach [26] has time complexity $O(dn \log n + 2^d mk \log n)$. The preprocessing time complexity of the proposed method is only $O(n)$ in contrast to that of the filtering method which has $O(n + drm'k)$ preprocessing time complexity.

6 Experimental study

The objective of this experimental study is to establish the practical efficiency of the proposed hybrid method. First, to establish the superiority of the present method over the previous preliminary version, we compare the total running times taken by these two methods. Running times of both the first stage and the correcting stages are individually analyzed over some bench mark data-sets. Later, we analyze the running time of the proposed method over various synthetic data-sets whose dimensionality ranges from 2 to 40. This is to prove that our method is more efficient than one of the best index based method which is called *the filtering method* [26]. Finally the running times of the four methods *viz.,* conventional *k*-means method, filtering method, *lk*-means-CMFT and *lk*-means-CMVT are compared to show that our method is more efficient than the other methods irrespective of dimensionality.

Table 1 Details of data-sets used

Data-set	Size of the data-set(n)	Number of dimensions (d)	Number of Clusters (k)
PENDIGITS data-set	10992	16	10
OCR data-set	10003	192	10
LIR data-set	20000	16	26
Synthetic data-sets SD_2 to SD_{40}	60000	2 to 40	3

Experiments are conducted on a PC with an *intel P4 processor (3.2 Ghz)* with 512 MB RAM. Totally 42 data-sets are used in our experimental study. These data-sets includes some bench mark data-sets and synthetic data-sets. The bench mark data-sets which are employed in this study includes Pendigits data-set, Letter Image Recognition (LIR) data-set and Optical Character Recognition (OCR) data-set. Pendigits data-set and Letter Image Recognition data-sets are available at the UCI Machine Learning Repository [36]. OCR data-set is also used in [3], [4] and [42]. A series of synthetic data-sets of varying dimensionality are artificially generated. The process of generating these artificial data-sets is explained in the following subsection.

6.1 Generating Synthetic Data-sets

Totally, 39 different synthetic data-sets $SD_2, SD_3, \ldots, SD_{40}$ of dimensionality ranging from 2 to 40 are generated as follows. Each data-set has 60000 patterns. Each data-set is generated from a tri-modal Gaussian distribution $p(x) = \frac{1}{3}N(\mu_1, \Sigma_1) + \frac{1}{3}N(\mu_2, \Sigma_2) + \frac{1}{3}N(\mu_3, \Sigma_3)$. For dimensionality 2, $\mu_1 = (0, 0), \mu_2 = (5, 5), \mu_3 = (-5, 5)$; for dimensionality 3, $\mu_1 = (0, 0, 0), \mu_2 = (5, 5, 5), \mu_3 = (-5, 5, -5)$; and so on up to dimensionality 40. In Each case the covariance matrix is taken as the Identity matrix of size $d \times d$, where d is the dimensionality of the data. The details of both bench mark data-sets as well as the synthetic data-sets which are used in the experimental study are given in the Table 1.

6.2 *lk*-means-CMFT Vs *lk*-means-CMVT

This subsection experimentally establishes the superiority of our method over its preliminary version. Running times of both first stage and the correcting stages are individually analyzed over the bench mark data-sets *viz.,* Pendigits data-set, Optical Character Recognition (OCR) data-set

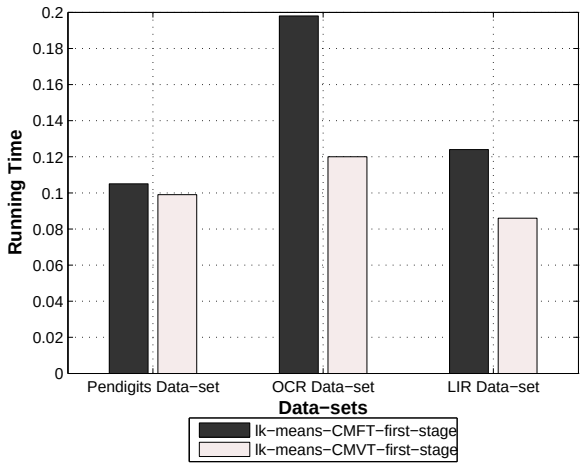

Fig. 4 Comparing running times of lk-means-CMFT-first-stage and lk-means-CMVT-first-stage

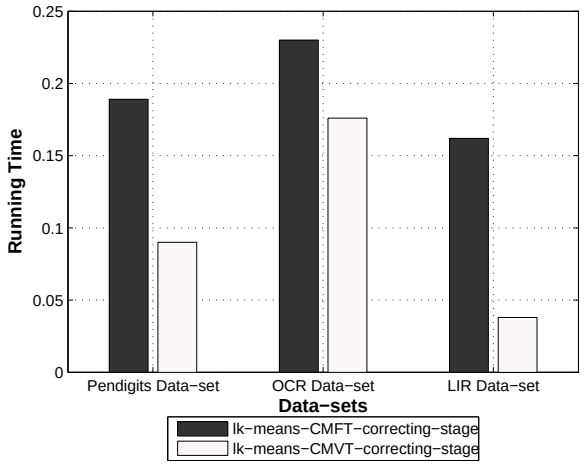

Fig. 5 Comparing running times of lk-means-CMFT-correcting-stage and lk-means-CMVT-correcting-stage

and Letter Image Recognition (LIR) data-set. In *lk*-means-CMFT, to fix the parameter τ, a random sample is derived form the given data-set and the average of all the distances between every pair of distinct random patterns, which are taken from that random sample, is calculated. Finally some percentage of such average distance is taken as the value of τ. In this experimental study, the parameter τ is taken as 50% of the average pairwise distances in a random sample of 5% of the data-set. In the proposed method the parameter τ is automatically calculated using the procedure which is explained in the section 4. Figure 4 and Figure 5 gives the running times of both stages in *lk*-means-CMFT and *lk*-means-CMVT. Form these two figures, it follows that the proposed method has reduced time both in the first stage as well as in the second stages. Hence, the total running time of the proposed method is less when compare to its preliminary version.

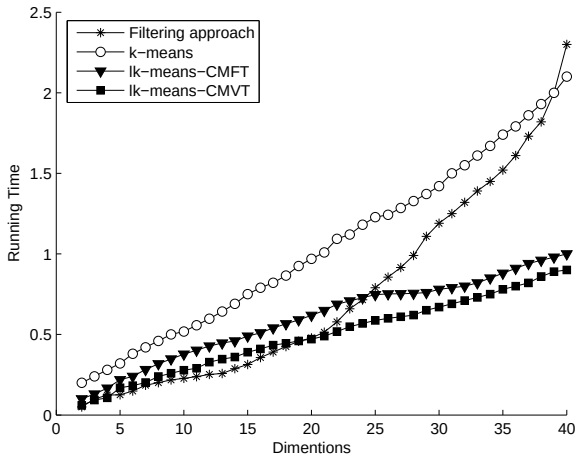

Fig. 6 Comparing the time taken by various methods

6.3 Comparative Study

This subsection compares the running times of the four clustering methods *viz.*, conventional *k*-means method, Filtering approach, *lk*-means-CMFT and the proposed method *i.e.*, *lk*-means-CMVT. Table 2 gives the running times of these four methods over the bench mark data-sets. The running times of all these four methods over the synthetic data-sets whose dimensionality ranges from 2 to 40 are plotted in the Figure 6. The conventional k-means method takes entire data-set in each iteration and its total running time depends on the size of the data-set. The filtering approach [26] shows good performance over the data-sets with low dimensionality. But, it has high running time when compared with the conventional k-means method for high dimensional data-sets. Further, *lk*-means-CMFT is much faster than the conventional k-means method as well as the filtering method, but its running time is highly influenced by the input parameter τ. The proposed method, *lk*-means-CMVT overcomes this drawback. Finally, form Figure 6, it follows that the proposed method is much faster than the remaining three methods irrespective of the dimensionality.

7 Conclusions

The paper presented a hybrid approach to speed-up the *k*-means clustering method. The proposed method is a prototype based one, where the prototypes called leaders are derived from the data-set using the leaders clustering method. Each leader represents a grouplet and the size of these grouplets are varied. Along with each leader some additional information, like its threshold, number of followers, etc., are stored without any additional computational burden. The proposed method has two stages, *viz.*, the first-stage and the correcting-stage. It is formally argued that the correcting-

Table 2 Comparing running times on various data-sets used

Data-set	Running Time of lloyd's *k*-means(In seconds)	Running Time of filtering method (In seconds)	Running Time of *lk*-means-CMFT (In seconds)	Running Time of *lk*-means-CMVT (In seconds)
PENDIGITS	0.463	0.326	0.294	0.189
OCR Dat Set	0.950	1.355	0.428	0.296
LIR Data Set	0.495	0.371	0.286	0.124

stage is not required at all in some favoring situations. The proposed method is suitable to work with high dimensional large data-sets.

References

1. Almedia, M. B., Braga, A., Braga, J. P.: Svm-km: Speeding svms learning with a priory cluster selection and k-means. In: Proceedings of the sixth Brazilian Symposium on Neural Networks, pp. 162–167 (2000)
2. Alsabti, K., Ranka, S., Singh, V.: An efficient k-means clustering algorithm. In: Proceedings of First Workshop High Performance Data Mining (March 1998)
3. Ananthanarayana, V., Murty, M., Subramanian, D.: An incremental data mining algorithm for compact realization of prototypes. Pattern Recognition **34**, 2249–2251 (2001)
4. Babu, T.R., Murty, M.N.: Comparison of genetic algorithms based prototype selection schemes. Pattern Recognition **34**, 523–525 (2001)
5. Berkhin, P.: Survey Of Clustering Data Mining Techniques. Technical Report, Accure Software (2002)
6. Bidyut Kr. Patra, Sukumar Nandi, Viswanath, P.: A distance based clustering method for arbitrary shaped clusters in large datasets. Pattern Recognition **44**, 2862–2870 (2011)
7. Bottou, L., Bengio, Y.: Convergence properties of the k-means algorithms. In: Advances in Neural Information Processing Systems 7, pp. 585–592. MIT Press (1995)
8. Bradley, P.S., Fayyad, U., Raina, C.: Scaling clustering algorithms to large databases. In: Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining, pp. 9–15. AAAI Press (1998)
9. Chitta, R., Murthy, M.N.: Two-level k-means clustering algorithm for k-τ relationship establishment and linear-time classification. Pattern Recognition **43**, 796–804 (2010)
10. Davidson, I., Satyanarayana, A.: Speeding up k-means clustering by bootstrap averaging. IEEE ICDM (2004)
11. Domingos, P., Hulten, G.: A general method for scaling up machine learning algorithms and its application to clustering. Eighteenth International Conference on Machine Learning (2001)
12. Farnstrom, F., Lewis, J., Elkan, C.: Scalability for clustering algorithms revisited. SIGKDD Explorations **2**(1), 51–57 (2000)
13. Gereon Frahling, Christian Sohler: A fast k-means implementation using coresets. In: Proceedings of the twenty-second annual symposium on Computational geometry (SCG '06), pp. 135–143. ACM Press, New York, USA (2006)
14. Gongde Guo, Si Chen, Lifei Chen: Soft subspace clustering with an improved feature weight self-adjustment mechanism. International Journal of Machine Learning and Cybernetics pp. DOI: 10.1007/s13,042–011–0038 (2011)
15. Grunbaum B: Convex Polytopes, 2 nd edn. Springer, New York (2003)
16. Guha, S., Rastogi, R., Shim, K.: Cure:an efficient clustering algorithm for large databases. In: Proceedings of Conference Management of Data (ACM SIGMOD'98), pp. 73–84 (1998)
17. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, second edition edition (2000)
18. Hartigan, J.A.: Clustering Algorithms. John Wiley & Sons, New York (1975)
19. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, N.J. (1988)
20. Jain, A., Murthy, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys **31**(3), 264–323 (1999)
21. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recognition Letters (2009)
22. Jain, A.K., Duin, P., Mao, J.: Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(1), 4–37 (2000)
23. Jiuzhen Liang, Wei Song: Clustering based on steiner points. International Journal of Machine Learning and Cybernetics pp. DOI:10.1007/s13,042–011–0047–7 (2011)
24. K.A. Abdul Nazeer, M. P. Sebastian: Improving the accuracy and efficiency of the k-means clustering algorithm. In: Proceedings of the World Congress on Engineering 2009 Vol I. London, U.K. (2009)
25. Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., Wu, A.: The analysis of a simple k-means clustering algorithm. Proc. 16th Ann. ACM Symp. Computational Geometry pp. 100–109 (June 2000)
26. Kanungo, T., Mount, D.M., Netanyahu, N.S., christine D. Piatko, Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence **24(**7), 881–892 (2002)
27. Kaufman, L., Rousseeuw, P.: Finding groups in data: An introduction to cluster analysis. John Wiley & Sons, New York (1990)
28. Krishna, K., Murty, M.: Genetic k-means algorithm. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics **29**(3), 433–439 (June 1999)
29. Lloyd, S.P.: Least squares quantization in PCM. IEEE Transactions on Information Theory **28**, 129–137 (1982)
30. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.J.: Fgka: A fast genetic k-means clustering algorithm. In: Proc. ACM Symposium on Applied Computing, pp. 622–623 (2004)
31. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
32. Pakhira, M.K.: A modified k-means algorithm to avoid empty clusters. International Journal of Recent Trends in Engineering **1**(1), 220–226 (May 2009)
33. parsons L, Haque E, Liu H: Subspace clustering for high dimensional data: A review. ACM SIGKDD Explore Newsletter **6**(1), 90–105 (2004)
34. Pelleg, D., Moore, A.: Accelerating exact k-means algorithms with geometric reasoning. In: Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 277–281 (Aug 1999)

35. Pelleg, D., Moore, A.: x-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the 17th International Conference on Machine Learning (July 2000)
36. P.M.Murphy: UCI Repository of Machine Learning Databases [http:// www.ics.uci.edu/mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA (1994)
37. R, J., A, G., G, A.: Fast and exact out-of-core and distributed k-means clustering. Knowledge and Information Systems **10**(1), 17–40 (2006)
38. Fahim A.M and Salem A. M, Torkey A, Ramadan M. A: An efficient enhanced k-means clustering algorithm. Journal of Zhejiang University **10**(7), 1626–1633 (2006)
39. Shi Na, Liu Xumin, Guan Yong: Research on k-means clustering algorithm: An improved k-means clustering algorithm. In: Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI), pp. 63–67 (April 2010)
40. Spath, H.: Cluster Analysis Algorithms for Data Reduction and Classification. Ellis Horwood, Chichester, UK (1980)
41. Steven J. Phillips: Acceleration of k-means and related clustering algorithms. In: Proceedings of Algorithms Engineering and Experiments(ALENEX02), pp. 166–177. Springer-Verlag, Berlin Heidelberg (2002)
42. T. Hitendra Sarma, Viswanath, P.: Speeding-up the k-means clustering method: A prototype based approach. In: Proc. 3rd Int. Conf. on Pattern Recognition and Machine Intelligence(PReMI)LNCS 5909, pp. 56–61. Springer-Verlag, Berlin Heidelberg (2009)
43. Vijaya, P., Murty, M.N., Subramanian, D.K.: Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognition Letters **25**, 505–513 (2004)
44. Viswanath, P., Pinkesh, R.: l-dbscan : A fast hybrid density based clustering method. In: Proceedings of the 18th Intl. Conf. on Pattern Recognition (ICPR-06), vol. 1, pp. 912–915. IEEE Computer Society, Hong Kong (2006)
45. Viswanath, P., V. Suresh Babu: *Rough-DBSCAN* : A fast hybrid density based clustering mehtod for large data sets. Pattern Recognition latters (2009), doi:10.1016/j.patrec.2009.08.008 (2009)
46. Vitter, J.: Random sampling with a reservoir. ACM Transactions on Mathematical Software **11**(1), 37–57 (1985)
47. Wu, X., Kumar, V., J. Ross Quinlan, Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.: Top 10 algorithms in data mining. Knowledge and Information Systems **14**(1), 1–37 (2008)
48. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks **16**(3), 645–678 (2005)