Chapter 2

Literature Survey

In this Chapter, first, the conventional k-means and kernel k-means clustering methods along with their merits and demerits are discussed in detail. Later, the existing methods in the literature that are proposed to speed-up the k-means and kernel k-means clustering methods are discussed. Finally, the proposed techniques to speed-up the k-means and kernel k-means clustering methods are outlined.

2.1 k-means clustering method

The k-means clustering method is the most widely used and well studied partition based clustering method that is based on minimizing a formal objective function, which is proposed by J.Mac Queen [8]. Given a dataset $\mathcal{D}$ which has n patterns in real d-dimensional space, $\mathbb{R}^d$, and an integer k, the problem is to determine a set of k patterns in $\mathbb{R}^d$, called k centers such that the mean squared distance from each pattern to its nearest center is minimized. This measure is often called the squared-error distortion [78] and this type of clustering methods are also called variance based clustering methods [129].

One of the most popular heuristics for solving the k-means problem is based on a simple iterative scheme for finding a locally minimal solution, it is also called Lloyd's k-means algorithm [22]. Throughout the thesis, the Lloyd's k-means clustering method is simply

called as k-means clustering method. Lloyd's algorithm is based on the simple observation that the optimal placement of a center is at the mean of the associated cluster. Each cluster is represented by its mean. The means have to be found such that the criterion $J = \sum_{j=1}^{k} \sum_{X \in C_j} ||X - M_j||^2$ is minimized, where M_j is the mean of the cluster C_j. Note that, finding a globally optimal solution for this problem is known to be a NP-hard problem (even for $k = 2$) [5].

In detail, k-means clustering method randomly selects k patterns from the dataset $\mathcal{D}$ which is of size n, as initial cluster centers. These initial centers are also called seed-points. Let $\mathcal{M}^{(0)} = \{M_1^{(0)}, M_2^{(0)}, \ldots, M_k^{(0)}\}$ be the set of initial seed points. Remaining (*n-k*) patterns are assigned to their nearest cluster centers. New centroid (mean) of each cluster is computed. Each pattern $X \in \mathcal{D}$ is again assigned to the nearest center and new centers are again found. This process is iterated until all centers (means) remain unchanged in two successive iterations.

The time complexity of the k-means method is $O(nkt)$, where n is the number of patterns in the dataset, k is the number of clusters and t is the number of iterations till the convergence. Lloyd's method is guaranteed to find a locally optimal clustering (hence the term local search algorithm), but this local optimum may be arbitrarily worse than the true optimum even for fixed n, k, and d [129]. The iterative procedure given by Lloyd [22] to find the k means is given in the Algorithm 2.1.

2.1.1 Limitations of the k-means method

The k-means method is simple to analyze and easy to implement. Hence, it has been widely used in data analysis, pattern recognition and image processing *etc.* [130] However, it has

Algorithm 2.1 k-means clustering method ($\mathcal{D}$, k)

Randomly choose k patterns as initial cluster centers from $\mathcal{D}$.
Let this be $\mathcal{M}^{(0)} = \{M_1^{(0)}, M_2^{(0)}, \ldots, M_k^{(0)}\}$.

Let $i = 0$; /* i is the iteration number */

repeat

$i = i + 1$;

Form k clusters by assigning each pattern in $\mathcal{D}$ to its nearest mean in $\mathcal{M}^{(i-1)}$.

Find new centroids (means) of the k clusters, *i.e.,* $\mathcal{M}^{(i)} = \{M_1^{(i)}, M_2^{(i)}, \ldots, M_k^{(i)}\}$.

until $(\mathcal{M}^{(i)} == \mathcal{M}^{(i-1)})$

Output: $\mathcal{M}^{(i)}$.

several limitations. The major limitations of k-means method are given below.

1. *Number of clusters*: The number of clusters must be determined before the iterative process begins.

2. *Sensitivity to initial conditions (local minima problem)* : The clustering result is extremely sensitive to the initial seed-points.

3. *Noise and empty clusters problem*: Noise, or outliers and empty clusters (to which no patterns are assigned) deteriorates the quality of the k-means clustering result.

4. *Inability to cluster non-linearly separable data*: It fails to identify non-linearly separable clusters in the input space.

5. *Scalability and efficiency in case of large datasets*: It does not scale well, since it needs to scan the dataset in each iteration.

The above mentioned limitations of the k-means method are also well studied. The present thesis focuses on the last two limitations. In the following, some of the existing improvements proposed to overcome the first four limitations are summarized. Whereas

the exiting improvements proposed to overcome the last two limitations will be elaborately discussed in the subsequent sections.

- *Number of clusters*:

  The value of k, *i.e.,* the number of clusters should be given as an input parameter for the k-means clustering method. How to select the number of clusters is among the key issues, which often calls upon prior knowledge about the application domain. The selected number of clusters reflects the desired trade-off between the two trivial solutions: at one extreme the whole dataset is put in a single cluster (maximum compression); at the other extreme, each point becomes a cluster (maximum accuracy).

  Solving the selection of a correct cluster number has been tried in two ways. The first one invokes some heuristic approaches. The clustering algorithm is run many times with the number of clusters gradually increasing from a certain initial value to some threshold value. The second is to formulate cluster number selection by choosing a component number in a finite mixture model. The earliest method for solving the model selection problem may be to choose the optimal number of clusters by Akaikes Information Criterion (AIC) or its extensions [131], [132]. Schwarzs Bayesian interface criterion-(BIC), Rissanens minimum description length-(MML), Bezdeks partition coefficients-(PC), Gap statistics and Dirichlet Process (DP) are other commonly used approaches for deciding the number of clusters [133], [5], [132]. Hansen *et.al.,* [134] used a related approach using the principle of Minimum Description Length (MDL) for selecting the number of clusters. Figueiredo *et.al.,* [135] used the minimum message length (MML) criteria in conjunction with the Gaussian mixture model (GMM) to estimate k. An interesting improvement over the k-means method,

called ISODATA [23], deals with the estimation of k. The X-means, proposed by Pelleg *et.al.,* [136], can not only accelerate the iterative process but also find the best estimate for the number of clusters in the k-means method. A generalization of conventional k-means clustering algorithm, called k^*-clustering method [98], performs correct clustering without predetermining the exact number of clusters. More Recently, Krista *et.al.,* [132] proposed another extended version of the k-means method, called *An efficient k'-means clustering algorithm* which performs correct clustering without pre-assigning the number of clusters.

- *Sensitivity to initial conditions (local minima problem)*:

  k-means method does not guarantee unique clustering because we get different results with randomly chosen initial clusters. The clustering result is extremely sensitive to the initial seed-points. This is also called cluster initialization problem. This leads to the local minima problem.

  Several attempts have been reported to solve the cluster initialization problem. A recursive method for initializing the means by running k clustering problems was discussed in [78]. Bradley and Fayyad [91], [137] proposes a procedure that refines the initial point to a point likely to be close to the modes of the joint probability density of the data. Khan and Ahmad [92] proposed an algorithm, called *the cluster center initialization algorithm (CCIA),* for computing initial cluster centres for the k-means clustering method. Redmond *et.al.,* [138] used a special data structure called *kd*-tree to solve this cluster initialization problem. Genetic algorithms have been developed for selecting the seed-points for k-means clustering method [93]. Steinley and Brusco [139] evaluated twelve procedures proposed in the literature for initializing k-means clustering

and to introduce recommendations for best practices. Lu *et.al.,* [94] proposed a hierarchical initialization approach for k-means clustering method. In [140], the centroids obtained are consistent with the distribution of data, which produced clusters with better accuracy, compared to the original k-means algorithm. Xiaoping Qing *et.al.,* [141] presented a method to compute initial cluster centers for k-means clustering based on an efficient technique for estimating the modes of a distribution. Some other related methods are discussed in [142], [143].

Many studies have attempted to combine k-means with other heuristic algorithms to prevent k-means from falling into local minima. Meila *et.al.,* [144] have shown that with a large probability, k-means will converge to global minima when the clusters are well separated. The stochastic optimal techniques like simulated annealing (SA) and genetic algorithms (GA) can find the global optimum with the price of expensive computation. Murthy *et.al.,* [145] designed a new hybrid scheme called, Genetic K-Means Algorithm (GKA) in order to do global search and fast convergence. Similar kind of genetic algorithm called, k-means with genetic algorithm (KGA) was proposed in [146]. Both, Genetic K-Means Algorithm (GKA) and k-means with genetic algorithm (KGA) used k-means to find the local minima and genetic algorithm to search for the global minimum. FGKA: A Fast Genetic K-means Clustering Algorithm is faster than GKA [147]. The *global k-means algorithm* proposed by Likas *et.al.,* [148] is a significant improvement of the k-means algorithm. It is an incremental algorithm that dynamically adds one cluster center at a time and uses each data point as a candidate for the k^{th} cluster center. However, this approach is not efficient since it is very time consuming, as n applications of k-means algorithm are made, where n is the size of

the dataset. Further, the authors suggest two procedures to reduce computational load. Recently, Bagirov *et.al.,* [149] have developed a new version of the global k-means algorithm, called the *modified global k-means algorithm*. Some other related methods were presented in [150], [151], [152].

- *Noise and empty clusters problem*

  The quality of the k-means clustering result will be deviated because of noise, or outliers in the data. Even if a pattern is quite far away from the cluster centroid, it is still forced into a cluster and thus distorts the cluster shapes. Both ISODATA [23] and PAM [84] will consider the effect of outliers in clustering procedures. The splitting operation in ISODATA eliminates the possibility of elongated clusters. PAM utilizes real data point (medoids) as the cluster representatives and avoid the effect of outliers.

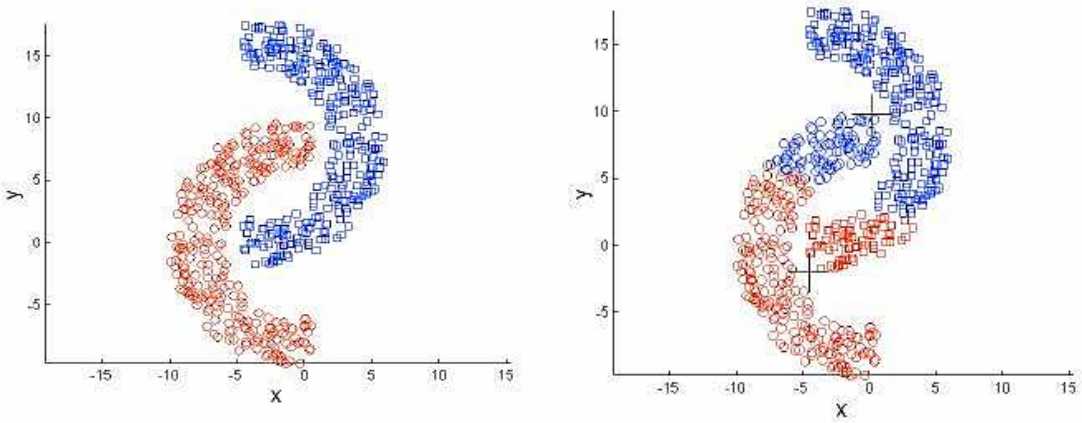
  In the iterative process of k-means method, there may be some clusters to which no pattern is assigned. Such clusters are called *empty clusters*. These empty clusters deteriorates the quality of the k-means clustering result. Some approaches have studied this empty cluster problem. Malay *et.al.,* [153] proposed an improvement over the k-means method called, the *modified k-means clustering method*, which overcomes this empty cluster problem by taking each old mean pattern as one of the patterns in its new cluster, in each iteration.

  In some papers, this empty cluster problem is called as under-utilization or dead-unit problem. One of the recent papers [132] proposed an enhanced version of k-means which resolves the dead-unit problem along with other draw backs of the method. The k^*- clustering method [98] is a generalization of the conventional k-means clustering

method. This method can find elliptical-shaped as well as ball-shaped clusters efficiently, without dead-unit problem.

- *Inability to cluster non-linearly separable dataset*:

  k-means clustering method inherently assumes that the clusters must be separated by a hyperplane; this follows from the fact that squared Euclidean distance is used as the distortion measure. In other words, the k-means method can find convex shaped clusters only. See Figure 2.1. In Figure 2.1(a) the banana dataset has two clusters which are non-convex in shape. The clustering result of the k-means method (for the same number of clusters *i.e.*, for $k = 2$) is as shown in Figure 2.1(b).



(a) Dataset before clustering (b) k−means clustering result (k=2)

Figure 2.1: k-means result for non-convex shaped clusters

Kernel k-means clustering method [90] is an extension of the standard k-means method which has been proved to be efficient in identifying linearly inseparable and non-convex shaped clusters in input space. Kernel k-means clustering method and its variants will be discussed in detail in the next section.

- *Scalability and efficiency in case of large datasets*:

  The k-means is a multi-scan method. It assumes that the entire dataet is memory resident. If the main memory is not enough to store the entire dataset, then the dataset has to be scanned once in each iteration which is a time consuming process. Further, the k-means method requires time proportional to the product of number of patterns and number of clusters per iteration. Finding the nearest mean for each pattern in each iteration is a time consuming process in case of large datasets.

  Scalability of the k-means clustering algorithm, especially for large or high dimensional datasets, has received extensive attention recently. Scalable k-means [154] uses buffering and a two-stage compression scheme to either compress or discard patterns to enhance the performance of k-means clustering method. Relational k-means [155] uses the block and incremental concept to provide a more stable clustering result than scalable k-means [154]. Huang *et.al.,* [97] proposed some extensions to the k-means method for clustering large datasets with categorical values. Some other scalable methods are proposed in [156], [12].

  The issue of speeding-up the k-means method for large datasets is well studied in the literature. Some speeding-up techniques produce the same result as the conventional k-means method [129], whereas the others may produce an approximate clustering result, but in reduced time [91]. These methods are discussed in detail in Section 2.3.1.

2.2 Kernel k-means clustering method

Kernel k-means clustering method [90] is a nonlinear extension of the k-means clustering method [157], [158]. It has been proved to be effective to identify clusters which are non-isotropic and non-linearly separable in the input space [159].

Kernel k-means clustering method is an iterative method, wherein, first the data points are mapped from the input space to a higher dimensional feature space through a non linear transformation $\phi(\cdot)$ and then minimizes the clustering error, similar to that in the k-means clustering method [160], but in the induced feature space. Mapping the data points form the input space to a higher dimensional kernel induced feature space results in linear separators in the induced space which corresponds to nonlinear separators in the input space. Thus, kernel k-means method avoids the limitation of linearly separable clusters in the input space that k-means suffers from. Figure 2.2 gives an illustration of how the linearly inseparable data in the input space gets linearly separable in the induced feature space.

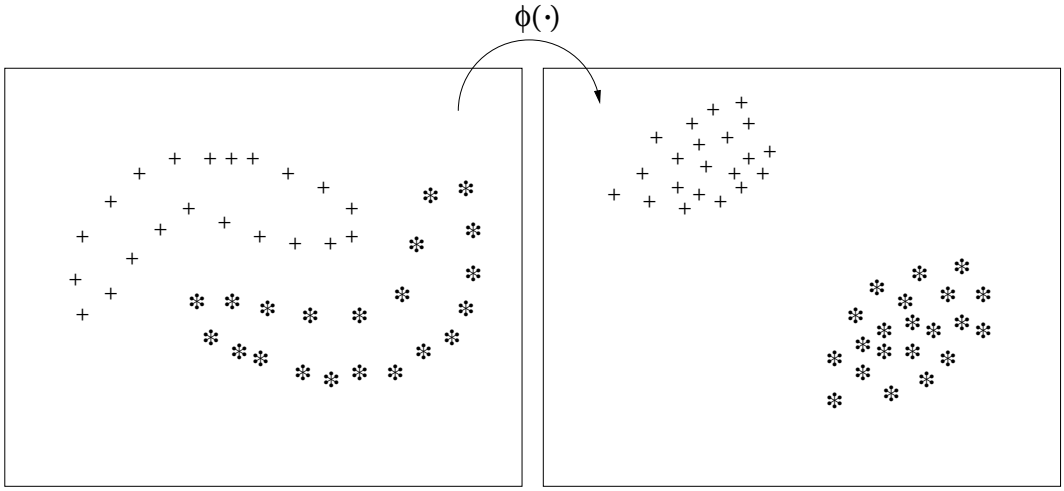


Figure 2.2: Linear separability of clusters in the induced feature space

The squared euclidean distance between two patterns X and Y in the induced feature space *i.e.,* $||\phi(X) - \phi(Y)||^2$ can be calculated using the dot products. The dot product between X and Y in the feature space, which is $\phi(X) \cdot \phi(Y)$, can be computed as a function $K(X, Y)$, where $K : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ is called the *kernel function*. Here $\mathcal{D}$ is the dataset and $\mathbb{R}$ is the set of real numbers. Because of this, the distance between any two patterns in the induced space can be found without the need for an explicitly known $\phi(\cdot)$. This is often known as the kernel trick and it is valid for the transformations that satisfy Mercer's conditions [44], [10].

Some standard kernel functions are given below.

- Linear kernel : $K(X_i, X_j) = X_i \cdot X_j$
- Polynomial kernel of degree d : $K(X_i, X_j) = (X_i \cdot X_j + 1)^d$, where d is a positive integer.
- Radial(RBF)kernel : $K(X_i, X_j) = exp(-\frac{||X_i - X_j||^2}{2\sigma^2})$, where $\sigma \in \mathbb{R}$.

The method randomly selects k patterns from the dataset $\mathcal{D}$ as the initial cluster centers or seed-points. Let $\mathcal{M}^{(0)}$ be a set of initial seed-points. Note, $\mathcal{M}^{(0)} = \{M_1^{(0)}, M_2^{(0)}, \ldots, M_k^{(0)}\}$ are points in the input space. For each pattern $X \in \mathcal{D}$, assign X to the mean $M_i^{(0)}$ such that $M_i^{(0)}$ is at the nearest distance from X in the kernel induced feature space. Let the resultant initial partition be $\Pi^{(0)}$. The kernel k-means method takes k and $\Pi^{(0)}$ as input parameters and produces a partition of the entire dataset, $\Pi_{\mathcal{D}}$ as the output.

The objective function is to minimize the criterion function

$$J = \sum_{j=1}^{k} \sum_{X_i \in C_j} ||\phi(X_i) - M_j||^2 \tag{2.1}$$

where M_j is the mean of the cluster C_j, for $j = 1, 2, \ldots, k$, in the induced space and

$$M_j = \sum_{X_l \in C_j} \frac{\phi(X_l)}{|C_j|}. \tag{2.2}$$

Here the key issue is that, the cluster center M_j can not be expressed explicitly (which is in contrast to the k-means method), as the explicit form of $\phi(\cdot)$ is not known. Because of this, the time complexity of the kernel k-means becomes quadratic whereas the k-means has linear time complexity, *w.r.t* the size of the dataset.

The Euclidean distance between two data points $\phi(X_i)$ and $\phi(X_j)$ in the induced space, can be found using a kernel function, as given below.

$$\begin{aligned} ||\phi(X_i) - \phi(X_j)||^2 &= \phi(X_i) \cdot \phi(X_i) - 2\phi(X_i) \cdot \phi(X_j) + \phi(X_j) \cdot \phi(X_j) \\ &= K(X_i, X_i) - 2K(X_i, X_j) + K(X_j, X_j). \end{aligned} \tag{2.3}$$

Further, $||\phi(X_i) - M_j||^2$ can be calculated without knowing the transformation $\phi(\cdot)$ explicitly, as given below:

$$\begin{aligned} ||\phi(X_i) - M_j||^2 &= ||\phi(X_i) - \sum_{X_l \in C_j} \frac{\phi(X_l)}{|C_j|}||^2 \\ &= K(X_i, X_i) + \mathrm{F}(X_i, C_j) + \mathrm{G}(C_j), \end{aligned} \tag{2.4}$$

where

$$\begin{aligned} F(X_i, C_j) &= -\tfrac{2}{|C_j|} \textstyle\sum_{X_l \in C_j} \phi(X_l) \cdot \phi(X_i) \\ &= -\tfrac{2}{|C_j|} \textstyle\sum_{X_l \in C_j} K(X_l, X_i) \end{aligned} \tag{2.5}$$

and

$$\begin{aligned} \mathrm{G}(C_j) &= \tfrac{1}{|C_j|^2} \textstyle\sum_{X_l \in C_j} \sum_{X_s \in C_j} \phi(X_l) \cdot \phi(X_s) \\ &= \tfrac{1}{|C_j|^2} \textstyle\sum_{X_l \in C_j} \sum_{X_s \in C_j} K(X_l, X_s) \end{aligned} \tag{2.6}$$

The iterative process of kernel k-means method is outlined in the Algorithm 2.2.

Algorithm 2.2 kernel k-means clustering method ($\mathcal{D}$, k, $\Pi^{(0)}$)

---

1. For each cluster C_j , find $|C_j|$ and $\mathrm{G}(C_j)$.

2. Compute $\mathrm{F}(X_i, C_j)$ for each X_i and for each cluster C_j.

3. Assign each $X_i \in \mathcal{D}$ to a cluster C_l such that $l = \min_j\{\mathrm{F}(X_i, C_j) + \mathrm{G}(C_j)\}$

4. Update M_j, for $j = 1$ to k.

5. Repeat step 2 through step 5 till convergence.

Output: The final Partition $\Pi_{\mathcal{D}} = \{C_1, C_2, \ldots, C_k\}$.

In the iterative process of the kernel k-means method, it is required to calculate $K(X_i, X_j)$ for $i, j = 1, 2, \ldots, n$ in each iteration. A matrix called the *kernel matrix* $H = [K_{ij}]_{n \times n}$ is precomputed and stored before starting the iterative process, where the $(i, j)^{th}$ entry is $K_{ij} = K(X_i, X_j)$. Note that, kernel matrix is a positive-semi definite matrix. Here n is the size of the dataset. The time and space complexity of the kernel k-means method is $O(n^2)$.

2.2.1 Limitations of kernel k-means clustering method

Kernel k-means clustering method is more efficient than the k-means clustering method in identifying non-linearly separable and arbitrary shaped clusters in the input space. However, kernel k-means method also has the following limitations.

1. *Sensitivity to initial conditions*: The kernel k-means clustering result depends on the initial seed-points.

2. *Scalability and efficiency in case of large datasets*: The time and space complexities of the kernel k-means clustering method are $O(n^2)$. Because of the quadratic time and space complexities the kernel k-means method is not a suitable one to work with large datasets.

Several improvements of the kernel k-means have been proposed to cater the above mentioned drawbacks. Some of them are summarized below.

- *Sensitivity to initial conditions*

  The clustering result obtained by the kernel k-means clustering method is highly influenced by the initial seed-points. Poor initial seed-points leads the convergence to poor local minima [157]. Likas *et.al.,* [158] proposed a deterministic and incremental approach to the kernel k-means method called the *global kernel k-means method* which deals with the cluster initialization problem and also avoids the poor local minima problem. Kenari *et.al.,* [161] proposed an intelligent weighted kernel k-means clustering method which can handle high dimensional data. Some related methods include

soft geodesic kernel k-means method [62] and the semi-supervised kernel fuzzy c-means method [162] *etc.* Several other related methods are reviewed in[163], [48], [164].

- *Scalability and efficiency in case of large datasets*

  To the best of our knowledge there are only a few improvements that are proposed to handle the time and space complexity problems of the kernel k-means method. Rudnichy *et.al.,* [160] proposed a block based approach to address the space complexity of the method for large datasets. In this approach, the kernel matrix H is computed before starting the iterative process and it is stored in the secondary memory. That is the size of H can be theoretically extended to as large as the entire disk. Later, the kernel matrix H is split into blocks, where size of each block is determined according to the I/O capability and the available main memory size. In each iteration, each block is moved as a whole from secondary memory to the main memory and processed. So the number of I/O operations for each iteration is equal to the number of blocks, but it is a costly operation when there are more number of blocks. Although this block based approach handles the space complexity, it still requires the computation of the full kernel matrix. Hence this approach is also not a good choice for large datasets as the time requirement is not reduced.

  Recently, Radha Chitta *et.al.,* [128] proposed two methods, *viz.,* two-step kernel k-means method and the approximate kernel k-means clustering method, which reduces the time and space requirements and produces a similar clustering result as the kernel k-means method. These methods are discussed in detail in the a Section 2.3.2.

2.3 Focus of the thesis and related work

The focus of the thesis is to speed-up the k-means and kernel k-means clustering methods for large datasets. Several techniques were proposed in the literature to speed-up the k-means clustering method, whereas to speed-up the kernel k-means clustering method, only a few methods were proposed, so far. This section gives an outline of all the exiting techniques in the literature which are proposed to speed-up the k-means and kernel k-means clustering methods.

2.3.1 Existing methods to speed-up the k-means clustering method

The two main factors that causes the k-means method to be slow are: computing many point-center distances and multiple scans of the dataset till convergence. The problem of speeding-up the k-means method has been extensively studied by many researchers. Several techniques were proposed to speed-up the k-means method for the past few years. These techniques can be broadly classified into two categories. 1) The methods which require some pre-processing like, creating indexes using some additional data structures to speed-up the clustering process. These methods are called the *index-based methods*. 2) The methods which reduce the number of times of scanning the entire dataset. The methods which require only one dataset scan are called the *single-scan methods*.

- **Index-based methods:**

  The computation time of k-means is reduced by some pre-processing as done in [165], parallelization [166] and intelligently setting the initial cluster positions [137].

  Alsabti *et.al.,* [167] 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.,* [136] 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.* [129] proposed a refined and better analyzed method over these methods called the *filtering algorithm* (FA). 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. The FA maintained, for each node of the *kd*-tree, a set of candidate cluster centers. The candidates were pruned, as they were propagated to the children nodes. For each leaf node having a single data point, the candidate was the nearest center to the data point. To our knowledge, the FA is the best for k-means clustering if the data dimension is moderate. The main drawback of FA is that its computational complexity grows exponentially with the dimensionality of the data *i.e.,* d. Recently, Lai and Liaw [168] proposed a modified filtering algorithm to reduce the computing time of *kd*-tree based algorithm. In [169], Lai *et al.,* used the information of cluster activity to speed-up the cluster center generation process.

All the above mentioned clustering methods pass no information from one stage of iteration to the next. Fahim *et.al.,* [170] 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., [142], [143] used some variants of the method presented in [170] to speed-up the iterative process. Recently, Shi Na *et.al.,* [171] 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.

- **Single-pass methods:**

  Scanning entire dataset once in each iteration makes the k-means more expensive, particularly for large disk-resident datasets. Considerable research has focused on designing clustering algorithms that require only one scan over the dataset. In all these methods, it is assumed that only a portion of the dataset can reside in the memory, and require a single scan through the dataset.

  The first single-pass method, called *single-pass k-means clustering method* was proposed by Bradley *et.al.,* [91]. This method used several data compression techniques to limit the memory usage. The method requires to scan the dataset only once to produce the clustering result. However, the result may be considerably deviated from that obtained by the conventional method. Later, Fredrik *et.al.,* [172] improved this idea to speed-up the clustering process. They proposed a new method called *simple single-pass k-means method* which is an improved version of the *single-pass k-means method*. The *simple single-pass k-means method* is much faster than the *single-pass k-means method*. Further, the quality of the final clustering result is also improved. Domingos *et.al.,* [156] and Guha *et.al.,* [173] proposed some other faster versions of the k-means algorithm which gives a better approximate solution.

Note that, all the above methods provide approximate solutions, possibly with deterministic bounded loss in the quality of the solutions. Recently, Goswami *et.al.,* [174] proposed another method called the *Fast and Exact k-means* method. Initially, it apples the conventional k-means on a small sample of the dataset to create the initial cluster centers followed by one full scan over the entire data to adjust these centers. The method produces the clustering result which is the same as that obtained by using the conventional k-means method.

- **The other related methods:**

  Some other methods which do not fit in the above categorization are explained bellow.

  Elkan *et.al.,* [175] presented a highly effective k-means algorithm which eliminates a large number of distance calculations between points and centers. It does this without indexing the data. Instead, it uses efficiently-updated bounds on center-point distances to determine when exact distance calculations can be avoided. Elkans algorithm exploits the triangle inequality to avoid many distance computations, and is the fastest current algorithm for high-dimensional data.

  Other strategies for accelerating k-means include sub-sampling large datasets as done in [137], [172], as well as finding initializations that will place centers near their final positions. Partial distance search [176] is another way to accelerate algorithms like k-means that need to identify closest points. Hochbaum and Shmoys [177] proposed their furthest-first algorithm as an approximate solution to the k-center problem. It has been popular to use this algorithm as an initialization technique to accelerate the k-means method.

Bootstrap averaging [178], Genetic k-means [145] and Fast Genetic k-means [147], CoreMeans [179], Compare-means method and Sort-means methods [180] *etc.* are some other related methods.

2.3.2 Existing methods to speed-up the kernel k-means method

This subsection outlines the recent improvements over the kernel k-means method that are proposed to address the running time and space requirement problems of the kernel k-means clustering method for large datasets.

Radha Chitta *et.al.,* [128] proposed a simple and naive approach, called the *two-step kernel k-means method*, for reducing the computational complexity of the method. In the first step, a random sample of q data points is selected from the dataset, and the optimal cluster centers only based on the sampled points is found. Later, in the second step every unsampled data point is assigned to the cluster whose center is at the nearest. This approach has reduced both time complexity and memory requirements. However, the clustering result obtained using this method depends on the selected random sample. The clustering result obtained using this method need not be same as that obtained using the conventional kernel k-means method, unless it is provided with a sufficiently large sample of data points.

Further, Radha Chitta *et.al.,* [128] proposed a randomized method called the *approximate kernel k-means* to reduce the total running time and space requirements of the conventional kernel k-means clustering method. In this method a random sample, say B, which is of size q is selected and the kernel similarity matrix, called H_B, between the data points in $\mathcal{D}$ and the sampled q data points is calculated. The size of the matrix H_B is $n \times q$ which is very less when compared to the size of the full kernel matrix H which is of size $n \times n$, for

$q << n$. The iterative process starts with an initial partition. In each iteration, the matrix H_B is used to estimate the closest cluster center for each data point. The process iterates till convergence. The time complexity of this method is $O(q^2kr + qnkr + q^2n)$, where r is the number of iterations till convergence. The space complexity of this method is $O(nq)$.

2.4 An outline of the present work

Both k-means and kernel k-means clustering methods are iterative methods. The overall running times of both the clustering methods directly proportional to the number of distance computations carried out in each iteration. One way to speed-up the k-means or kernel k-means clustering methods is to reduce the number of distance computations. This is the inspiring step to our proposed prototype based hybrid schemes to speed-up the k-means and kernel k-means clustering methods.

In order to reduce the number of distance computations, the proposed methods selects a few selected prototypes that represents the entire dataset. Later the k-means or kernel k-means clustering methods is allowed to work with only with this reduced set instead of the entire dataset. However, the final clustering result may be a deviated one when compared to that obtained using the entire dataset. Some correcting steps are proposed to reduce such deviations. The thesis argues that in most of the cases such correcting steps are not required. Both theoretically and experimentally the proposed prototype based hybrid methods are shown to be much faster than the conventional methods.

Along with these prototype based hybrid methods, the thesis also proposes a single-pass approach to speed-up the kernel k-means clustering method which is motivated form the *simple single-pass k-means method* [172] and the two-step kernel k-means clustering

method [128]. The proposed *single-pass kernel k-means method* produces approximate clustering result when compared to that obtained by using the conventional kernel k-means method, but in a reduced time.

2.4.1 Proposed techniques: In brief

This section gives an outline of the proposed techniques to speed-up the k-means and kernel k-means clustering methods.

- **To speed-up the k-means clustering method**

    1. *lk*-means Clustering Method with Fixed Threshold (*lk*-means-CMFT):

        This method works in two stages. In the first stage, the dataset is partitioned a number of grouplets. Each grouplet has a fixed (constant) size and is represented by a prototype called its *leader*. The k-means clustering method is applied on the set of leaders, to derive a partition of it. If each leader is replaced by its grouplet, we get a partition of the entire dataset. The clustering result obtained at this stage may not be same as that obtained using the conventional k-means method over the entire dataset. In the second stage, a correcting step is proposed in order to get the same clustering result as obtained by the conventional k-means method. Both theoretically and experimentally, it is established that *lk*-means-CMFT is faster than the conventional k-means clustering method. The efficiency of this method depends on the the size of the grouplets. However, finding the optimal size of the grouplets, in advance, is difficult as well as a time consuming task.

2. *lk*-means Clustering Method with Varying Threshold (*lk*-means-CMVT):

   This is an improved version of the above method *i.e., lk*-means-CMFT which addresses the problem of finding the optimal size for the grouplets. It works similar to *lk*-means-CMFT, but the key difference is that, each grouplet has its own size and it is fixed on fly. Theoretically and experimentally, it is proved that *lk*-means-CMVT will produce the same clustering result as that obtained using the conventional k-means method, but in a faster pace. Further, it is also shown that *lk*-means-CMVT is more efficient than *lk*-means-CMFT.

- **To speed-up the kernel k-means clustering method:**

1. Kernel *lk*-means Clustering Method with Fixed Threshold

   (Kernel-*lk*-means-CMFT):

   In this method, first, the dataset is partitioned into a number of grouplets which are of same sizes such that these grouplets are formed in the kernel induced feature space, where as each grouplet is represented by a prototype called its leader, which is a pattern in the input space. Later, the set of leaders is again grouped into k clusters by using the kernel k-means clustering method. Finally, each leader is replaced by its grouplet to get the partition of the entire dataset. Both theoretically and experimentally it is shown that, the kernel *lk*-means-CMFT produces a similar or same (under some favorable conditions) clustering result when compared with that obtained using the conventional kernel k-means method, but in a reduced time. However, the efficiency of this method depends on the size of the grouplets (τ). But the optimal size of the grouplets is difficult to find before hand.

2. Kernel *lk*-means Clustering Method with Varying Threshold (Kernel-*lk*-means-CMVT):

   This is an improvement over the above method *i.e.*, kernel-*lk*-means-CMFT. Note, the clustering result of this method is independent of the parameter τ, which is the size of the grouplets, as used in kernel-*lk*-means-CMFT. It works similar to kernel-*lk*-means-CMFT, but the key difference is that, each grouplet in the induced space has its own size and it is fixed on fly. Kernel-*lk*-means-CMVT is more efficient than kernel-*lk*-means-CMFT both in terms of clustering quality and the running time.

3. *single-pass kernel k-means:*

   The method works in two stages. Initially, the method applies the conventional kernel k-means method on a small random sample and derives a partition of it. For each cluster in the partition of the random sample, its center (mean) is found by using the gradient descent method. Finally, each unsampled pattern in the dataset is assigned to its nearest cluster center to derive a partition of the entire dataset. The *Single-pass kernel k-means method* gives a great reduction on total running time, but the efficiency of this method depend on the size of the initial random sample.

2.5 Conclusions

In this chapter, first, the conventional k-means and kernel k-means clustering methods are discussed in detail along with their limitations. Existing methods that are proposed to speed-up the k-means and kernel k-means are reviewed. Finally, the proposed techniques to speed-up the k-means and kernel k-means clustering methods are outlined.