Journal of Computations & Modelling, vol.6, no.3, 2016, 43-85
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2016

Automatic *K*- Expectation Maximization (A *K*-EM) Algorithm for Data Mining Applications

Archit Harsh[1] and John E. Ball[2]

Abstract

A non-parametric data clustering technique for achieving efficient data-clustering and improving the number of clusters is presented in this paper. K-Means and Expectation-Maximization algorithms have been widely deployed in data-clustering applications. Result findings in related works revealed that both these algorithms have been found to be characterized with shortcomings. K-Means does not guarantee convergence and the choice of clusters heavily influenced the results. Expectation-Maximization's premature convergence does not assure the optimality of results and as with K-Means, the choice of clusters influence the results. To overcome the shortcomings, a fast automatic *K*-EM algorithm is developed which provides optimal number of clusters by employing various internal cluster validity metrics, thereby providing efficient and unbiased results.

[1] Electrical and Computer Engineering department, Mississippi State, University, Starkville, MS, 39759. E-mail: archit.harsh89@gmail.com, ah2478@msstate.edu
[2] Electrical and Computer Engineering Department, Mississippi State, University, Starkville, MS, 39759. E-mail: jeball@ece.msstate.edu

Article Info: *Received* : August 9, 2016. *Revised* : September 11, 2016.
Published online : October 1, 2016.

The algorithm is implemented on a wide array of data sets to ensure the accuracy of the results and efficiency of the algorithm.

Mathematics Subject Classification: Big Data Clustering; Machine Learning; Algorithms

Keywords: K-Means; Expectation-Maximization; Data Clustering

1 Introduction

Clustering algorithms partition a dataset into several groups such that points in the same group are similar to each other and points across groups are different from each other [2]. The clustering algorithms are categorized into three main categories, namely partition based clustering, nearest-neighbor based clustering, and density-based clustering. Among them, the widely known clustering techniques include AGNES (Agglomerative Nesting), DIANA (Divisive Analysis), CLARA (Clustering Large Applications), PAM (Partitioning around medoids), BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), *K*-Medoids, FCM (Fuzzy C-Means), *K*-Means and EM (Expectation-Maximization) algorithms [1]. The K-Means algorithm [3] is the most popular and widely known clustering algorithm because of its simplicity and efficiency. Originally developed for and applied to the task of vector quantization, K-Means has been used in a wide assortment of applications. Studies have shown that it is a good approach to cluster data. However, K-Means does not assure the best representation for the data as it uses distances from the centers of clusters to determine which sample belongs to which class. It has also been proved that, with K-Means, there is no guarantee for optimal clustering, since the convergence depends on the number of clusters provided (k) which is often not known in advance. Moreover, K-Means is not considered as the best choice of clustering

due to its poor convergence and predefined choice of clusters, *k*. The EM is a model-based approach to solving clustering problems. It is an iterative algorithm that is used in scenarios where the data is incomplete or contains missing values. Unlike partition-based or hard membership algorithms, EM is known to be a suitable optimization algorithm for constructing proper statistical models of the data. EM is widely used in applications such as computer vision, speech processing and pattern recognition etc. EM aims at finding clusters such that maximum likelihood of each clusters is obtained. In EM, each observation belongs to each cluster with a certain probability. EM starts with an initial estimate for the missing variable and iterates to find the maximum likelihood (ML) estimates for these variables. Maximum likelihood methods estimate the parameters by values that maximize the sample's conditional probability for an event. EM is conventionally used with mixture models. In EM clustering, the number of clusters that are desired is predetermined. It is initialized with values for unknown (hidden) variables. Since EM uses maximum likelihood it most likely converges to local maxima, around the initial values. Hence, the selection of initial values is critical. The proposed approach works in two phases: the first phase consists of development of automatic K-Means, which is a non-parametric approach, where the optimal number of clusters are chosen based on the internal clustering metrics. The initial cluster centers chosen by this approach serves the basis for the second phase. The second phase consists of applying the EM algorithm. EM takes these centers as its initial variables and iterates to find the local maxima. Hence, clusters that are well distributed using K-Means and clusters that are compact are obtained using EM. The algorithm is tested using various datasets of varying sizes, shapes and dimensions. The results show that the proposed approach guarantees convergence and provide optimal clusters with reduced time complexity.

2 Technical Background

Clustering is a division of data into groups of similar objects, i.e. objects which share similar characteristics with those of the rest within that cluster. Each group, called a cluster, consists of objects that are similar to themselves and dissimilar to objects of other clusters. From a machine learning perspective, clusters often lead to discovery of hidden patterns in the data, aiding in knowledge discovery and extraction. The search of clusters is often termed as unsupervised clustering, and the resulting system represents a data concept [4]. Clustering is often categorized into three different categories: unsupervised, supervised and semi-supervised. Unsupervised clustering techniques assume that no prior knowledge of data is known in advance. Supervised clustering techniques relies on some sort of prior knowledge about the data. In that sense, they are supervised. Semi-supervised clustering is a bridge between the unsupervised and supervised clustering. Clustering algorithms have been a vital part of statistics [5] and science [6]. The classical introduction of these algorithms into pattern recognition framework is given in [7]. In the literature, various categories of clustering algorithms have been presented and developed. For the reader's convenience, a classification of these algorithms is provided.

2.1 Clustering Algorithms

The clustering algorithms have been widely classified in three different categories:

1. Hierarchical Methods.
2. Partitioning Methods.
3. Density based Methods.

Hierarchical clustering build clusters gradually (as crystals are grown). These methods are further subdivided into agglomerative and divisive [8, 9] methods. Agglomerative methods tends to “chain” the clusters within different clusters. On the other hand, divisive clustering tends to divide the clusters into different families, thereby creating compact and distinct clusters. Examples of agglomerative clustering include SLINK [10], COBWEB [11], CURE [12] and CHAMELEON [13]. Partitioning algorithms build clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets, or try to identify clusters as areas highly populated with data. They are further classified into probabilistic clustering (EM [14, 15], SNOB [16], AUTOCLASS [17], MCLUST [18]), *K*-Medoids methods (PAM [9], CLARA [9], CLARANS [19]), and K-Means methods (K-Means [20, 21], K-Means++ [42], harmonic means [27], fuzzy c means [43, 44]). Such methods concentrate on how well points fit into their clusters and tend to build clusters of proper convex shapes.

Density based clustering algorithms try to discover dense connected components of data, which are flexible in terms of their shape and structure. Examples include DBSCAN [22, 45], OPTICS [23], GDBSCAN [24], etc.

In the literature, partition-based clustering techniques have gained reputation and popularity based on their simplistic implementation, scalability and applications. The K-Means algorithm is by far the most popular clustering tool used in scientific and industrial applications. The name originates from representing each of K clusters, C_j by the mean c_j of its points, called centroid. Various approaches have been proposed in the research community to improve and modify this basic idea. FCM, K-Means++, harmonic means are among the most important advances in K-Means.

Despite its popularity and simplicity, K-Means approaches suffers from two major shortcomings. It requires a user-centric parameter k, number of clusters, in advance, which is often not available and heavily influences the results. Because of its iterative procedure and random initialization of centers, it can suffer from

convergence issues. Since, K-Means tends to provide a local minimum instead of a global one, it tends to terminate prematurely, leading to poor convergence in specific data sets. In [25], the researchers developed AK-Means based on K-Means which is a non-parametric approach and the clusters are optimally chosen by the state-of-the art internal cluster validity metrics. In [26], the researchers came up with a hybrid K-Means expectation maximization approach. The idea put forth was to combine the K-Means and EM algorithm for better convergence. However, the researchers only provide the results on only one data set which sets a limiting factor for applicability of the approach. In this paper, we develop an efficient Automatic *K*-EM approach which modifies the ideas presented in [25] and [26] and attempt to provide wide array of results with numerous data sets of varying types, shapes, sizes and dimensions. For evaluation of clusters, two widely known internal cluster validity metrics were used: namely elbow plots and the Silhouette index, which are discussed in detail in the upcoming sections.

2.2 Technical Description

2.2.1 Fast K-Means

In this paper, we used K-Means based on weighted average instead of normal mean, to get new clusters. This approach is faster and more efficient than the default Matlab implementation of K-Means which is based on K-Means++. We have modified the current Matlab implementation of K means++ by vectorizing some of the parameters and setting suitable number of replications and iterations for faster processing and efficiency.

Let $\{x_1,.....x_p\}$ be a set of P real numbers. The number of iterations is given as r. The weighted average (WA) at the r^{th} iteration is given by [27].

$$\mu^{r+1} = \sum_{p=1,P} w_p{}^{(r)} \mathbf{x}_p \tag{1}$$

An initial mean is taken and weight, w_p, is obtained for x_p. In equation (2), $w_p^{(r)}$ is the weight calculated at the rth iteration. Feature vectors are assigned to each cluster point, empty or small clusters are eliminated. Cluster centers are replaced with weighted averages and feature vectors are reassigned. This process is repeated until the algorithm converges to a local minimum.

2.2.1.1 K-Means performance function

The K-Means algorithm works by partitioning the data set into *k* clusters, $S = \{S_1, ...S_k\}$ by placing each data point in the nearest cluster. It intends to find the local optimal set of centers that minimizes the total within cluster variance, which is formally defined as K-Means performance function [27],

$$Perf_{KM}(X, M) = \sum_{k=1}^{K} \sum_{x \in S_i} \| \mathbf{x} - \mathbf{m_k} \|^2 \quad (2)$$

where, the k^{th} center, $\mathbf{m_k}$, is the centroid of the k^{th} partition. The double summation in (3) can be alternatively expressed as a single summation over all data points, adding only the distance to the nearest center expressed by the minimum function *MIN*.

$$Perf_{KM}(X, M) = \sum_{i=1}^{N} MIN\{\| \mathbf{x_i} - \mathbf{m}_k \|^2\} \quad (3)$$

The K-Means algorithm starts with an initial set of clusters and then iterates through the following steps:

1. For each data point, locate the closest center m_k, and assign the data point to the k^{th} cluster.

2. Re-evaluate all the centers. The k^{th} center becomes the centroid of the k^{th} cluster.

3. Iterate through steps 1 and 2 until the clusters do not change significantly.

We will direct our readers to [28, 29] for the detailed proof of the local optimum. Also, K-Means is a NP-hard problem, as such no significant work proves that it provides global optimum [30].
For most time, the algorithm reach a stable partition in a finite number of steps for finite datasets. The cost per iteration is O(*kDN*).

2.2.2 Expectation Maximization (EM) clustering algorithm

The EM algorithm is considered a variant of K-Means which is based on mixture models. It is often referred to as an optimization strategy for K-Means as it doesn't require a training phase. The algorithm follows an iterative approach to find the parameters of the probability distribution that have the maximum likelihood of its attributes. In this work, EM based on Gaussian mixture models (GMM) is used. The input parameters for the algorithm are the data set (x), the total number of clusters (*k*), the accepted error to converge, also known as tolerance, (*e*) and the maximum number of iterations, $iter_{max}$. The algorithm starts with an initialization phase, where the parameters are initialized and then iterates through the expectation and maximization phase until convergence. For each iteration, the algorithm goes through the expectation phase, called the E-step, which estimates the probability of each point belonging to that cluster, followed by the Maximization phase, which re-estimates the parameter vector of the probability distribution of each class. The termination phase constitutes the convergence of the parameters. It is to be stated here that the convergence doesn't necessarily imply global optimum as it may be prematurely terminated.
To avoid this, we performed multiple iterations of the algorithm and took the average number of iterations (steps) to converge. This strategy is widely used in literature, especially when using partition-based clustering approaches.

2.2.2.1 Initialization phase

Each class *j*, of *k* clusters, is constituted by the parameter vector $(\boldsymbol{\theta})$, consisting of the vector mean $(\boldsymbol{\mu})$ and the covariance matrix $(\mathbf{P}_j)$, which represents the Gaussian probability distribution used to characterize the observed and hidden entities of the data set x.

$$\boldsymbol{\theta}(\mathrm{t}) = \boldsymbol{\mu}_{\mathbf{j}}(t), \mathbf{P}_{\mathbf{j}}(t),\ j = 1, 2, ..., k \tag{4}$$

At *t=0,* the algorithm randomly generates the initial mean and covariance matrix. The EM algorithm aims to approximate the parameter vector $(\boldsymbol{\theta})$ of the real distribution.

2.2.2.2 E-step

This step is responsible for estimating the probability of each element belonging to a specific cluster (Bayesian rule). Each element of the data is composed by an attribute vector $(\mathbf{x}_k)$. The association of the point to the cluster is estimated by the maximum likelihood of that point in comparison to other elements of the cluster [31].

$$p(C_j \mid \mathbf{x}_i) = \frac{p\left(\mathbf{x}_i \mid C_j\right) p\left(C_j\right)}{\sum_{k=1}^{K} p\left(\mathbf{x}_i \mid C_j\right) p\left(C_j\right)} \tag{5}$$

where, $p\left(\mathbf{x}_i \mid C_j\right)$ is the prior probability with Gaussian distribution and $p\left(C_j\right)$ is the mixing probability.

2.2.2.3 M-step

This step is responsible for updating the parameters computed in E-step. First, the mean $(\boldsymbol{\mu}_j)$ of classes *j* is computed using the mean of all points in function of the relevance degree (association) of each point.

$$\boldsymbol{\mu}_j(t+1) = \frac{\sum_{k=1}^{N} p(\mathrm{C}_j \mid \mathbf{x}_k)\mathbf{x}_k}{\sum_{k=1}^{N} p(\mathrm{C}_j \mid \mathbf{x}_k)} \tag{6}$$

Secondly, the covariance matrix is computed for the next iteration by applying the Bayes theorem [46], based on the conditional probabilities of the class occurrence. The covariance matrix is given by:

$$\sum\nolimits_{(t+1)} = \frac{\sum_{k=1}^{N} p(\mathrm{C}_j \mid \mathbf{x}_k)(\mathbf{x}_k - \boldsymbol{\mu}_j(t))^T(\mathbf{x}_k - \boldsymbol{\mu}_j(t))}{\sum_{k=1}^{N} p(\mathrm{C}_j \mid \mathbf{x}_k)} \tag{7}$$

Finally, the probability of occurrence of each class is computed through the mean of probabilities in function of the relevance degree of each point from the class.

$$p_j(t+1) = \frac{1}{N}\sum_{k=1}^{N} p(\mathrm{C}_j \mid \mathbf{x}_k) \tag{8}$$

2.2.2.4 Steps for implementation of EM

1. Mean and standard deviation of the data is randomly initialized.
2. Expectation step.
3. Maximization step.
4. Convergence Step.

At each iteration, a convergence test is performed which verifies if the difference of the attributes vector of an iteration to the previous iteration is smaller than an acceptable error tolerance, provided by the user-defined parameter.

If $\left(\left\|\boldsymbol{\theta}(t+1)-\boldsymbol{\theta}(t)\right\| < e\right)$

Stop

else

call E-step

end; (where e is the accepted tolerance and $\boldsymbol{\theta}(t)$ is the estimate at time t).

At each step, the algorithm estimates a new attribute with a maximum likelihood, no necessarily global, which reduces its complexity. However, depending on the dispersion of the data and its volume, the algorithm can also be terminated due to the maximum number of iteration defined.

2.2.2.5 Performance function of Expectation Maximization

The EM algorithm estimates the centers, **X**, the co-variance matrix, $\sum_k$ and the mixing probabilities **p**. The performance function is given by [27, 95] as

$$Perf_{EM}(\mathbf{X},\mathbf{M},\sum_k,\mathbf{p}) = -\log\left\{\prod_{x\in S}\left[\sum_{k=1}^{K} p_k \frac{1}{\sqrt{(2\pi)^D \det\left(\sum_k\right)}} \exp(-(\mathbf{x}_i-\mathbf{m}_k)\sum_k{}^{-1}(\mathbf{x}_i-\mathbf{m}_k)^T)\right]\right\} \quad (9)$$

E-step: Estimates the association/likeliness of x belonging to the k^{th} cluster:

$$p(\mathbf{m}_k \mid \mathbf{x}) = \frac{p(\mathbf{x}\mid\mathbf{m}_k)\,p(\mathbf{m}_k)}{\sum_{x\in S} p(\mathbf{x}\mid\mathbf{m}_k)\,p(\mathbf{m}_k)} \quad (10)$$

where, $p(x|m)$ is the prior probability with Gaussian distribution, and $p(\mathrm{m}_k)$ is the mixing probability.

M-step: Updating the membership of each attribute using the fuzzy membership function from the E-step, this step updates the new center locations, new

covariance matrices, and new mixing probabilities that maximize the performance function [27].

$$\mathbf{m}_k = \frac{\sum_{x \in S} p(\mathbf{m}_k \mid \mathbf{x})\mathbf{x}}{\sum_{x \in S} p(\mathbf{m}_k \mid \mathbf{x})}$$

$$\sum_k = \frac{\sum_{\mathbf{x} \in S} p(\mathbf{m}_k \mid \mathbf{x})(\mathbf{x} - \mathbf{m}_k)^T (\mathbf{x} - \mathbf{m}_k)}{\sum_{x \in S} p(\mathbf{m}_k \mid \mathbf{x})} \tag{11}$$

$$p(\mathbf{m}_k) = \frac{1}{|\mathrm{S}|} \sum_{\mathbf{x} \in S} p(\mathbf{m}_k \mid \mathbf{x})$$

3 Proposed Method

The Automatic *K*-Expectation Maximization algorithm is divided into two phases: The automatic K-Means initialization phase and the EM iterative phase.

3.1First Phase

3.1.1 Automatic K-Means

In the initialization phase, the automatic K-Means variation of *K*- Means is used to classify the data into the number of desired clusters by employing three well-known internal cluster validity metrics; namely, elbow plots [33], average Silhouette index [34] and CH index [35]. The merit for selecting these metrics comes from the fact that these indices have been a part of statistical community since 1950's and provide good merit for the choice of optimal clusters. They are easy to implement and generally yield desirable clusters.

The proposed algorithm starts by setting an upper bound on the choice of clusters, where *k* is taken to be $n^{1/2}$, where *n* is the number of data points in the

provided data set. This choice is influenced by the fact that usually the number of clusters lies between 2 to $n^{1/2}$, as reported by Pal and Bezdek in [36].

Next, it applies a deterministic initialization procedure proposed by the authors in [37]. Fast K-Means algorithm is applied on these initial *k* centroids, and centroid of the smallest cluster is removed. It restarts again with the remaining centroids. At each iteration, the maximum of the CH cluster validity index [35] of the current iteration is stored. The motivating factor for choosing this index comes from the fact that it is relatively easy to compute and implement and it generally provides robust results in comparison to other cluster validity metrics. Additionally, we also implemented two other validity indices as mentioned before, to add more weight to the optimal clusters. The process is continued until k=2.

Finally, the algorithm outputs the number of clusters, the centroids, and the corresponding labels. It also provides the mean Silhouette value which indicates the suitability of the clusters with respect to the data set. In practice, a mean Silhouette value closer to unity indicates good fit.

3.1.2 Internal Cluster Validity metrics

The motivation for employing internal cluster validity metrics originates from underlying principle of the proposed work that “No prior knowledge of data is provided in advance. It is essentially an unsupervised clustering approach.”

Internal cluster validity metrics, as its name suggests, provides the cluster validation internally without relying on prior knowledge of the data in opposition to the external cluster validity metrics, which rely on prior knowledge of data.

Some common external validity metrics include entropy, purity, etc. In literature [38], it has been proven that the results provided by both these metrics are almost identical. In some scenarios, external cluster validity metrics perform better than the internal cluster validity metrics. It is quite intuitive as they always rely on prior information which make them realistically impossible to implement in practical scenarios.

In the proposed method, we used three internal cluster validity metrics namely, elbow plots, the Silhouette index, and the CH index.

Elbow plots are visual in nature, meaning the user has to look for an “elbow” in the plot. The plot is an indicator of the suitability of clusters where the user specify a range of clusters to be examined. The “elbow” indicates the desired cluster to be chosen. The elbow method seeks the percentage of variance in the data taken to be a function of number of clusters. As stated before, since the users specify the range of clusters, more often, the first clusters add a lot of variance (more contribution), but at some point the gain drops, giving an angle “elbow” in the graph. The elbow indicates the desired number of clusters.

After the K-Means initialization phase, elbow plot is generated to provide a method of validation for the Silhouette index and the CH index. The Silhouette index measures the cohesion based on the distance between all the points in the same cluster and the separation based on the nearest neighbor distance. In practice, a larger average Silhouette index indicates higher clustering accuracy.

The CH (Carlinski-Harabasz) index is the ratio of between-cluster variance and within cluster variance, a.k.a variance ratio criterion (VRC). It takes a range of clusters and outputs the estimate the ratio. A higher value indicates good number of clusters as it maximizes the between-cluster variance which is desirable.

3.1.3 Objective function of Automatic K-Means

The proposed approach is a non-parametric clustering approach where the number of clusters k is optimally chosen by employing three internal cluster validity metrics. The algorithm fundamentally works as a K-Means algorithm with an added feature of internal cluster validity metrics implemented internally. As mentioned in section 2, the performance function of K-Means is given by equation 1.

Since, the proposed approach is independent on the number of clusters. The cost per iteration is $O(DN)$, where, D is the number of dimensions of the data and N is

the number of data points in the dataset. As the time-complexity of the proposed approach is independent of *k*, it is expected to run in linear time. To validate the run time-complexity, we have provided line plots to convey the same to the readers. As expected, the proposed method proceeds linearly with the number of dimensions and the number of data- points. This validation is discussed in the upcoming sections.

Finally, we compared the performance of the proposed algorithm with state-of-the art clustering techniques viz. *K*-Medoids, K-Means++ and FCM. The results show that the proposed approach is better in terms of time complexity and convergence rate and is scalable to higher dimensional datasets.

3.1.4 Pseudo-Code

Algorithm 1 Fast Automatic K-Means

1: Load Data set - ***a = load(data);***

2: Define rows and columns of the data-matrix

[n,p] = size(a)

where, *n*=rows and *p*=cols

3: Set the cluster upper bound as reported in [36] to be $n^{1/2}$

4: for each *k*, number of clusters, do

5: Run Fast K-Means clustering

6: Calculate elbow plots

7: end for

8: Initialization phase:

9: Run Fast K-Means on the range of clusters defined in step 3.

10: Calculate the CH index, reported in [35].

11: **while** $k > 2$, **do**

12: Find the minimum index and store the indices in an empty matrix

13: Decrement *k*

14: Perform fast K-Means
15: Compute the CH index
16: Store the maximum index and the corresponding *k* value.
17: end while
18: **for** *k* computed from step 16, **do**
19: Compute the Silhouette index, reported in [34].
20: **end for**
21: Plot the clustering results
22: Plot the Silhouette index
23: Print the optimal *k* value
24: Print the mean Silhouette index

3.2 Second Phase

The second phase consists of the EM algorithm, which takes the initial centroids and clusters provided by the first phase and iteratively runs EM algorithm for optimizing results.

First, the number of clusters and cluster centers are selected from the Automatic K-Means algorithm. The algorithm proceeds by computing the maximum likelihood estimation for the *k* clusters and calculate new centers for each of these clusters. The algorithm moves through the Expectation and Maximization phases in a traditional loop-back fashion. First, it runs the E-step for estimating the input parameters and then updates the parameters by running the Maximization step. The process is repeated until a user defined tolerance *e* is reached.

3.2.1 Objective Function of the proposed approach

Applying the identity transformation/mapping to the performance function of Automatic K-Means and EM, the following equations are obtained:

$$Perf_{KM}(X, M) = -\log\left(\prod_{i=1}^{N} \exp(-MIN\{\|\mathbf{x}-\mathbf{m}\|^2, \mathbf{m} \in \mathrm{M}\})\right) \quad (12)$$

$$Perf_{EM}(\mathrm{X}, \mathrm{M}) = -\log\left(\prod_{i=1}^{N}\sum_{l=1}^{k} p_l \frac{1}{\sqrt{(2\pi)^D}} \exp(-\|\mathbf{x}-\mathbf{m}\|^2)\right) \quad (13)$$

where, *X* is the data-set and *M* is the centroids.

The expressions inside the brackets are a result of linear mixing of exponential functions. The constant $\frac{1}{\sqrt{(2\pi)^D}}$ does not impact the performance function. Hence, it does not change the locations of the optima of the function.

3.2.2 Pseudo code

Algorithm 2: *K*-EM Algorithm based on GMM

1: Output Parameters: **W**, **M**, and **V**

W = Estimated weights of GM

M = Estimated mean vectors of GM

V= Estimated covariance matrices of GM

2: Input Parameters: X, *k*, l_{tol}, $iter_{max}$, *plot*, *Init*

X = Given Data

k = Number of clusters (optimally chosen from algorithm 1)

l_{tol} = Tolerance value, denoted as *e*, in earlier notations

$iter_{max}$ = maximum number of allowed iterations

plot = 1 for 1-D, 2 for 2-D

Init = Structure of initial W, M, V

3: Initialize the parameters by running automatic K-Means on the input data

4: Perform the Expectation step using the equations listed in 6 and 11

5: Obtain the log-likelihood estimate of the input parameters

6: Initialize number of iterations

7: **While** the value of the log likelihood estimate is less than tolerance value & number of iterations is less than or equal to $iter_{max}$, **do**
8: Perform the Maximization step using the equations listed in
9: Loop back and forth between these two steps until convergence
10: end while
11: Plot the log-likelihood estimate versus the number of iterations
12: Plot the Clustering results

3.3 Datasets

3.3.1 Types of Datasets:

The proposed approach is tested and evaluated on a number of data sets with varying size, shape and dimensions. We implemented the algorithm on 20 different data sets categorized as follows:

1. Real Data sets: These data sets are available as an open-source on UCI repository [39] which is a data-collection website maintained by UCI for machine-learning applications. Example data sets include Wine, Breast, Iris and yeast. The "Real" data sets are obtained from real world scenarios and experiments. They provide a strong validation of the algorithm as they depict natural data representation and are widely used in literature.

2. Synthetic Data sets: These data sets are generated artificially to provide more insight about the algorithm and its potential limitations. For the evaluation of this algorithm, synthetic data sets of higher dimensions and big data-sizes were generated. These data sets validate the scalability and time-complexity of the algorithm.

3. Shape Data sets: These data sets are available on [94]. The shape data sets are chosen to evaluate the suitability of the algorithm on different data structures. For instance, spherical, cylindrical etc. The motivation for choosing these data sets come from the fact that by nature, K-Means does not perform well on non-convex

structures and more often, fail to converge. The proposed algorithm was tested to overcome this limitation and outperforms K-Means.

4. Higher Dimension Data sets: These data sets are artificially generated to validate the scalability of the proposed approach in higher dimensions.

5. Miscellaneous Data sets: These data sets are a combination of shape, size and dimensions.

For each of these categories, we selected four data sets on which we implemented our algorithm and evaluated the performance. The following table lists the type of data sets in each category.

Table 1 Types of Data sets

Category	Data set	Data Description
Real	Breast	This data set contains the records of patients suffering from breast-cancer. There are two classes: Benign and Malignant. Source: [39]
Real	Iris	It is a widely-known data set taken from fisher's Iris data. It contains four species of Iris, each with three classes. Source: [39]
Real	Wine	It contains three different classes of wine found in North America. Source: [39]
Real	Thyroid	Thyroid viruses categorized in two classes. Source: [39]
Miscellaneous	R15	The first occurrence of R15 data set is found in [49].
Miscellaneous	A1 A2 A3	A data sets were used in a variety of clustering tasks. The data sets were first implemented in [50].
Shape	S1 S2	S Data sets comes under the category of shape Data sets, i.e., they consist of varying

Category	Data set	Data Description
	S3 S4	shapes with different degree of overlap. Source: [48]
Higher Dimensions	Dim32	Gaussian data set with 32 dimensions.
Higher Dimensions	Dim64	Gaussian data set with 64 dimensions.
Higher Dimensions	Dim256	Gaussian data set with 256 dimensions.
Higher Dimensions	Dim1024	Gaussian data set with 1024 dimensions.
Synthetic	Synthetic1	Synthetic1 is an artificially generated data set with one million data points.
Synthetic	Synthetic2	Synthetic2 is an artificially generated data set with two million data points.
Synthetic	Synthetic3	Synthetic3 is an artificially generated data set with three million data points.
Synthetic	Synthetic4	Synthetic4 is an artificially generated data set with four million data points.

4 Implementation Details

4.1 Software Used

The algorithm is designed and developed in MATLAB. The reason for choosing this programming language is fundamentally based on the fact that MATLAB is a very powerful tool for efficiently handling large memory requirements involved in scientific and engineering problems. Moreover, the majority of the clustering algorithms are already a part of this language which makes it relatively easy to validate the results and evaluate the performance. The R programming language was also used. R is an open-source language preferred by data-scientists and data-analysts as it supports a wide array of clustering tools and efficiently handle huge data structures. This language is used to validate the results initially carried out in MATLAB.

4.2 System specifications

The algorithm is tested and developed on Dell Inspiron N5010 computer with 8GB of RAM and an Intel i5 processor with 2.4 GHz processing speed.

5 Results and Discussions

5.1 Performance Metrics

To evaluate the performance of the algorithm, we implemented the following performance metrics: Number of clusters versus actual clusters, run time complexity, convergence and scalability. The chosen metrics validate the proposed approach in terms of convergence, scalability, and time-complexity.

Additionally, we provide a comparative study between the proposed method and the state-of-the-art clustering algorithms, specifically K-Means++, K-Medoids, and FCM for validation purposes.

5.1.1 Number of clusters versus actual clusters

This metric validate the algorithm's accuracy for finding desirable clusters. For accomplishing this, we used a wide array of data sets with known number of clusters. The choice of these data sets confirm the efficiency of the algorithm as the ground-truth is available and known. As it is a non-parametric approach, this metric is critical for testing the algorithm. In the following table, we provide the results of clustering obtained by our algorithm on different data sets. Additionally, we also list the mean Silhouette index value which is a strong indicator for goodness of fit of the clusters. Ideally, the value closer to one indicate better clustering results.

As clearly evident from the table, the proposed approach yields perfect results in most scenarios. The results are validated by comparing with the ground truth as

mentioned in the table. Also, the mean Silhouette value is closer to unity in most cases which is a desirable property. We also provide the clustering plots as obtained by the algorithm. The plots are generated in MATLAB with different color markers suggesting different clusters.

For better understanding of the results, we provide results for two data sets in each category. Among them, we selected one simple (simple structure) data set and one highly convoluted data set for clustering. By doing so, we attempt to provide a fair understanding of the results to the readers. Additionally, we also provide one such scenario where the algorithm fail to perform due to the nature of the data set.

Table 2 listing the results is on the following page.

Table 2: Experimental Results of Application of A*K*-EM on Different Data set

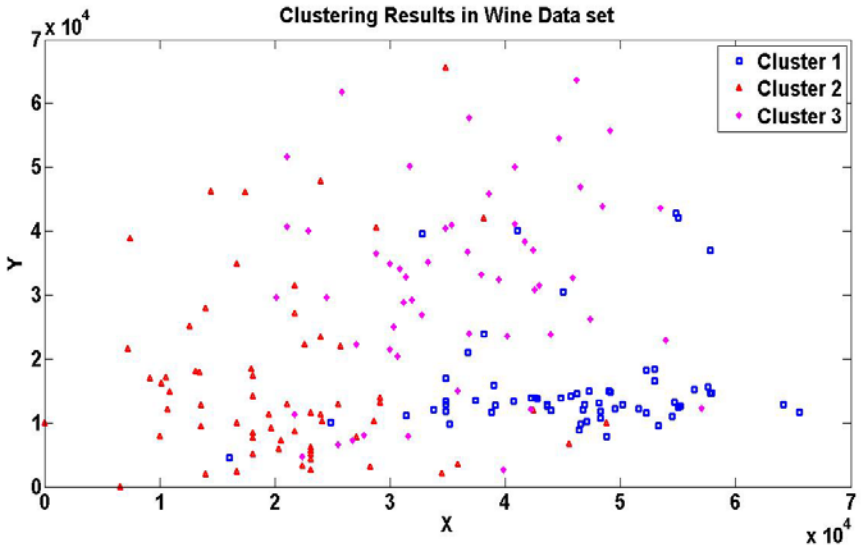
Data set	**Data points**	**Category**	**Dimension/ Features**	**True *k***	**A*K*-EM estimated *k***	**Mean Silhouette index**
Breast	699	Real	9	2	2	0.7542
Iris	150	Real	4	3	3	0.7786
Wine	178	Real	13	3	3	0.7503
Thyroid	215	Real	5	2	2	0.7773
R15	600	Miscellaneous	15	15	15	0.9543
A1	3000	Miscellaneous	20	20	20	0.7892
A2	5250	Miscellaneous	35	35	35	0.7911
A3	7500	Miscellaneous	50	50	20	0.7949
S1	5000	Shape	15	15	15	0.8803
S2	5000	Shape	15	15	15	0.8009
S3	5000	Shape	15	15	15	0.6659
S4	5000	Shape	15	15	15	0.6446

Dim32	1024	Higher Dimensions	32	16	16	0.9962
Dim64	1024	Higher Dimensions	64	16	16	0.9985
Dim256	1024	Higher Dimensions	256	2	2	0.9991
Dim1024	1024	Higher Dimensions	1024	16	16	0.9998
Synthetic1	1,000,000	Synthetic	5	5	5	0.7887
Synthetic2	2,000,000	Synthetic	5	5	5	0.7654
Synthetic3	3,000,000	Synthetic	5	5	5	0.7896
Synthetic4	4,000,000	Synthetic	5	5	5	0.7994

5.1.1.1 Clustering plots

1. Wine Data set

The following plot shows the clustering results for the wine data set.



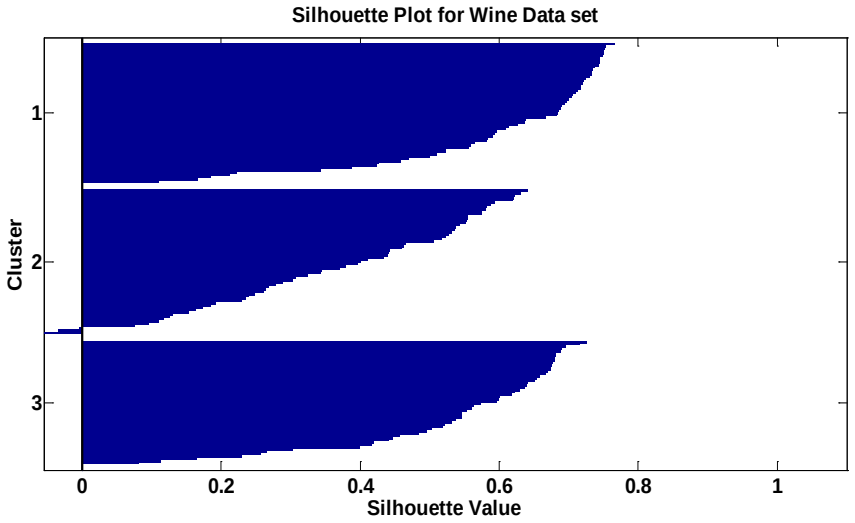


Figure 1: Clustering Result for Wine Data set

As clearly evident from the plots, the proposed algorithm correctly identifies three clusters. Additionally, the Silhouette plot validates the results giving three distinct clusters. These plots are provided to present a visual understanding of the results to the readers.

2. Iris Data set

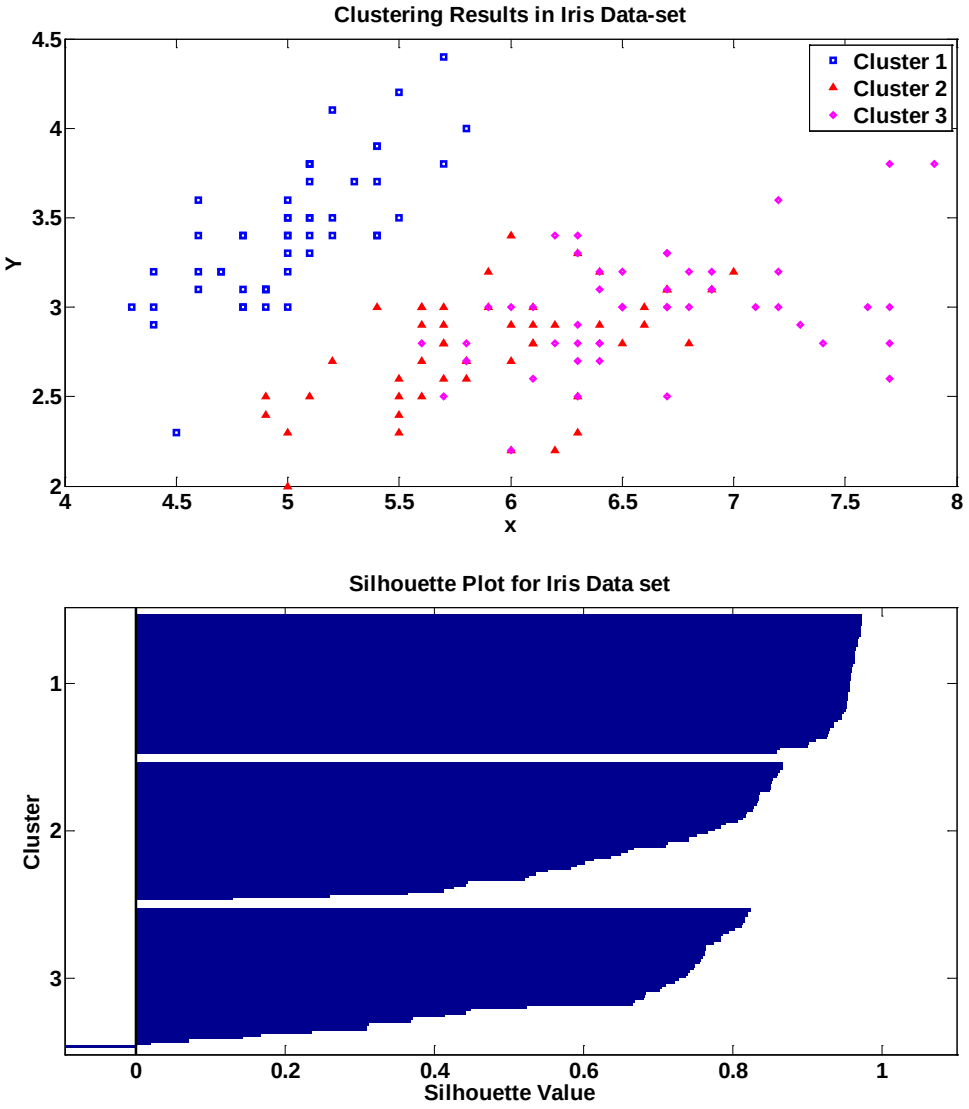


Figure 2: Clustering Result in Iris Data set

The aforementioned plot presents the clustering results in Iris Data set. It is quite evident that the algorithm correctly determines the three clusters which match with the original data description. The Silhouette plot agrees with the results.

3. R15 data set

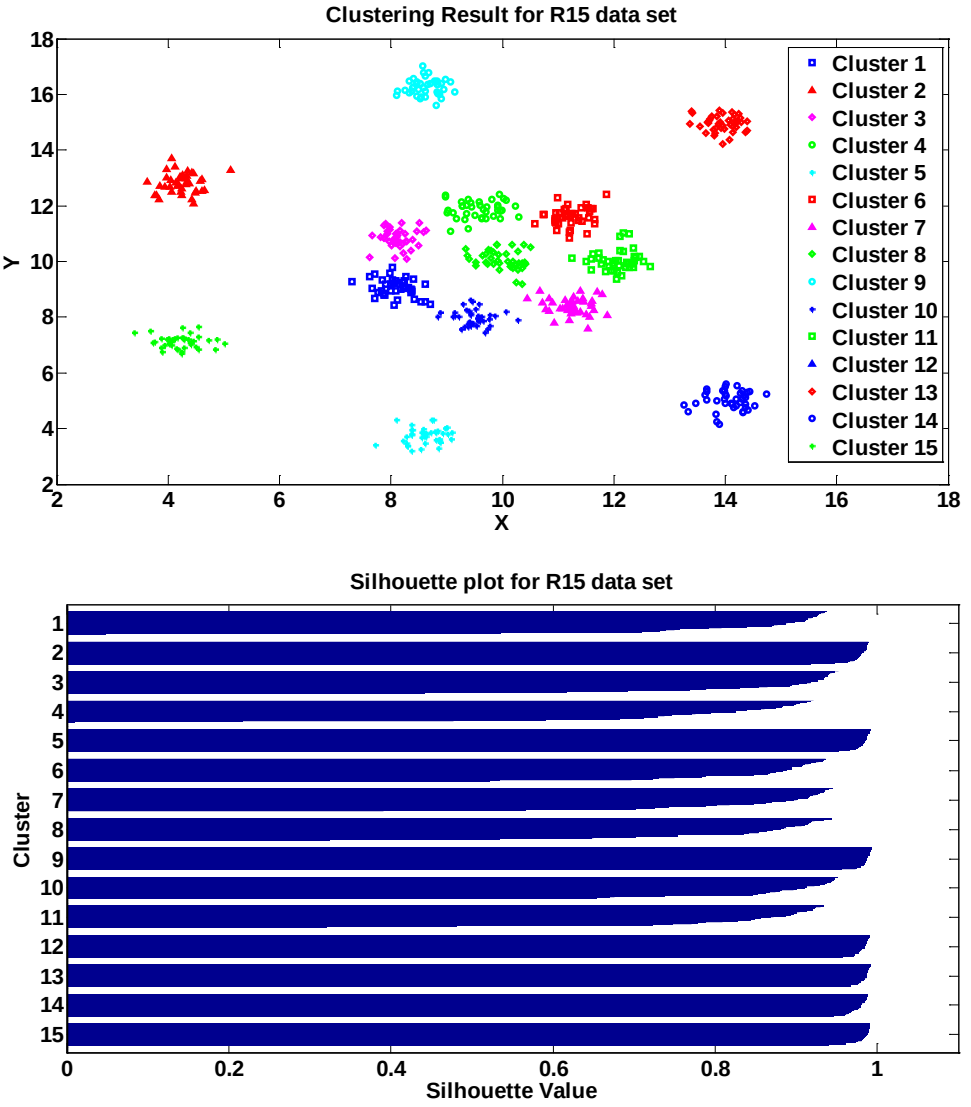


Figure 3: Clustering Result in R15 data set

The aforementioned plot indicates the clustering results in R15 data set. Through visual inspection, one can easily witness the nature of the data. The inside ring of clusters are correctly identified by the proposed method. However, one can also argue that due to the highly convoluted nature of data, the inside rings can be merged as a single cluster. The algorithm performs extremely well in this scenario giving distinct clusters. The Silhouette plot also confirms the results.

4. A3 data set

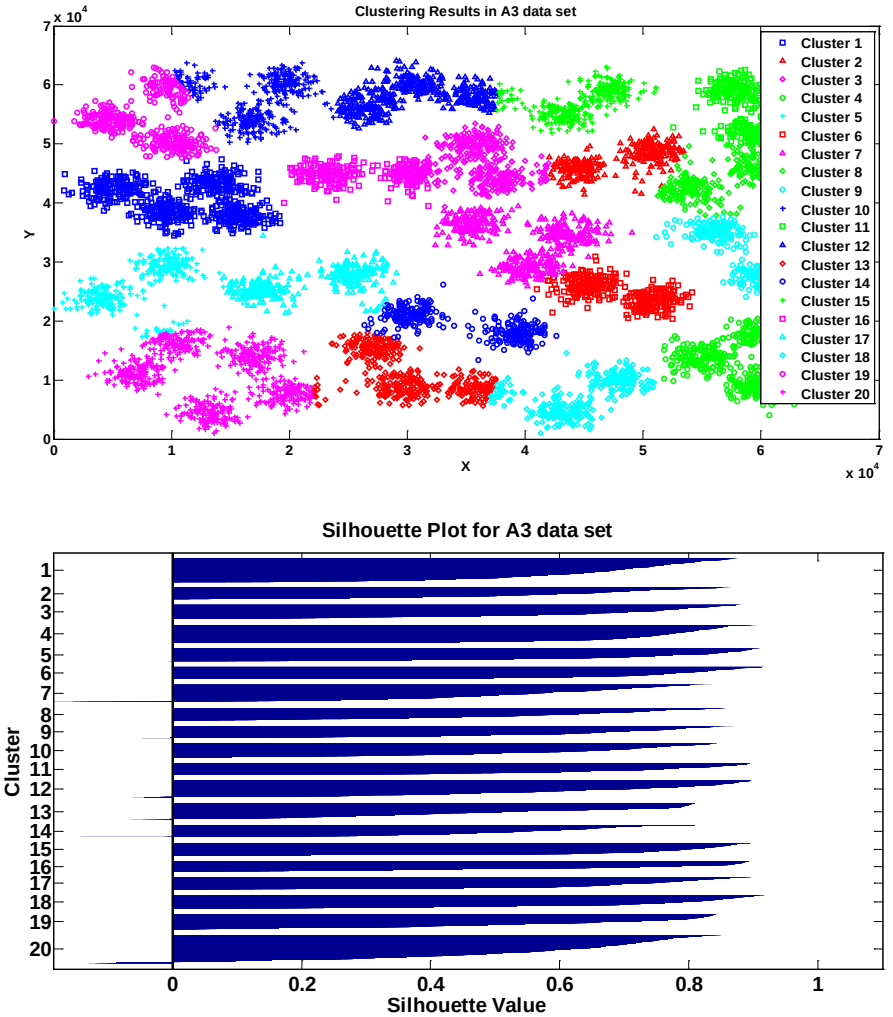


Figure 4: Clustering Result in A3 data set

The given plot provides the clustering results for A3 data set. For this specific dataset, the proposed algorithm fails to correctly identify the number of clusters. It identifies 20 clusters against the true value of 50. The possible reason for failure is that the clusters are highly overlapping and there is no clear distinction between them. This result is critical for the readers as it highlights the limitation of the proposed method. As previously mentioned, K-Means is inherently build to utilize Euclidean distance as the metric, it tends to fail when the clusters are highly

constricted. I would also like to state that in such scenarios, all the Partitional based clustering techniques tend to fail as they rely heavily on some specific objective function, which may perform poorly in highly convoluted data sets.

The following plots are generated as a result of implementation of the algorithm on shape data sets.

5. S1 data set

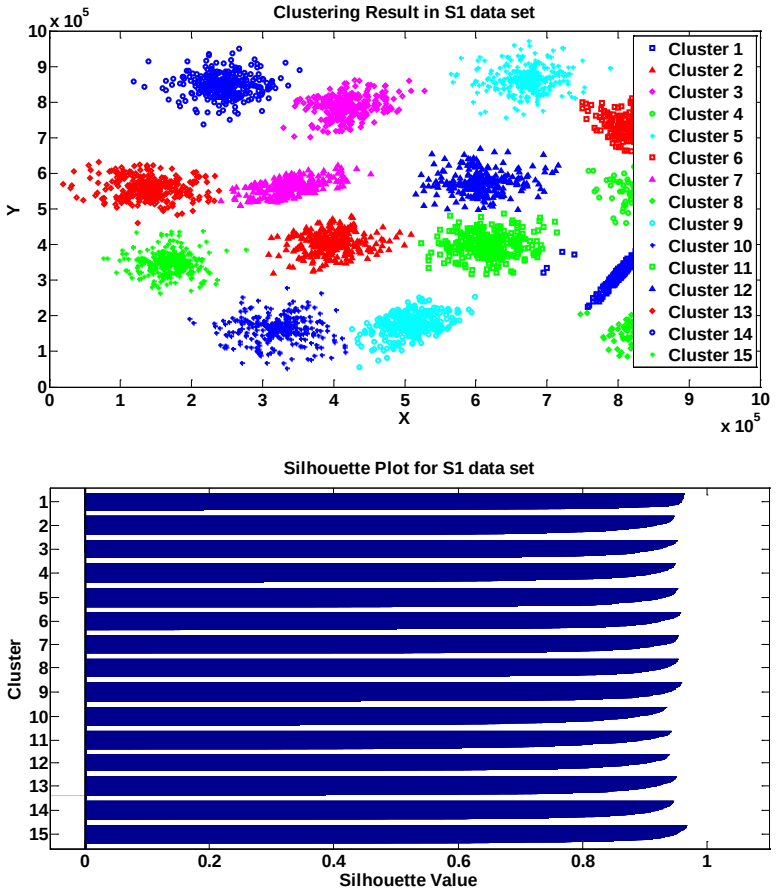


Figure 5: Clustering Result in S1 data set

In the aforementioned plot, the proposed method is applied to S1 shape data set. It correctly identifies the number of clusters, as expected. The Silhouette plot validates the results.

6. S4 data set

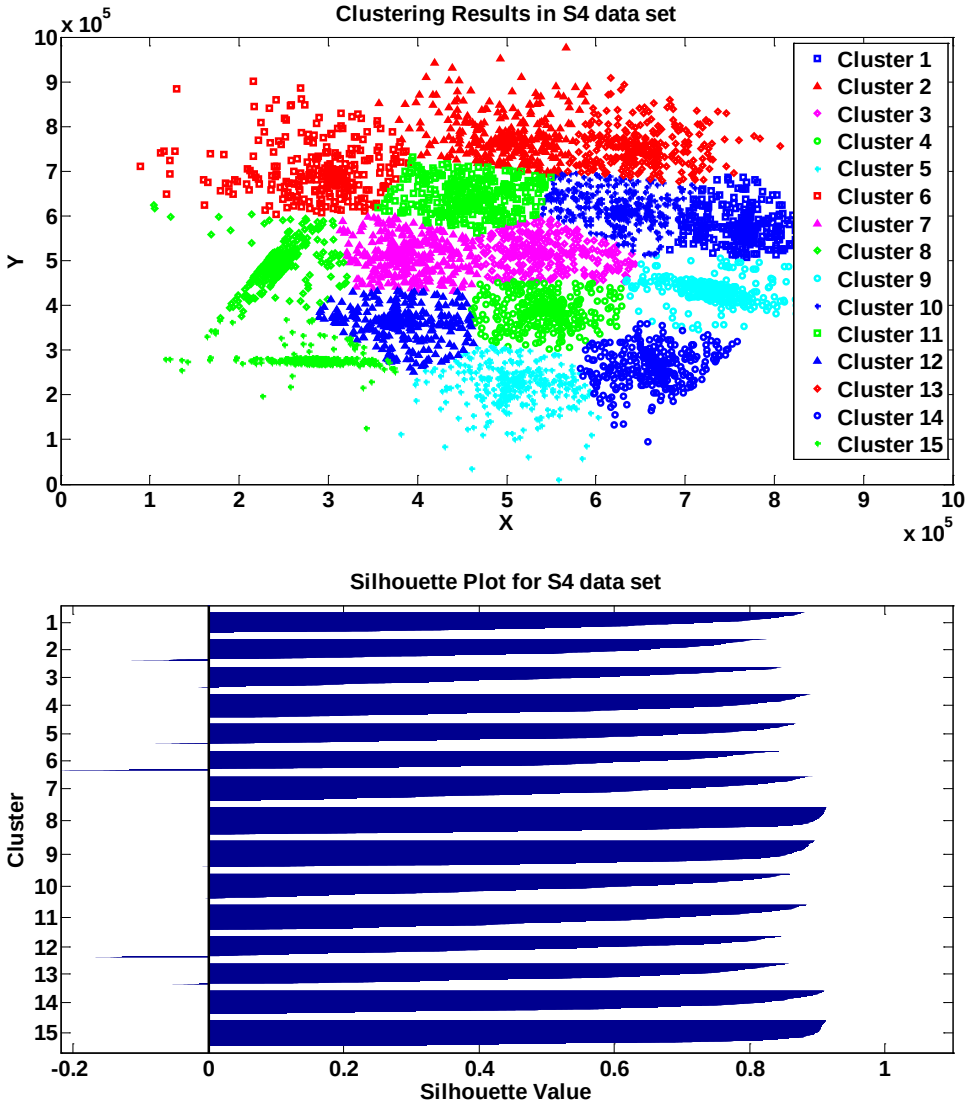


Figure 6: Clustering Result in S4 data set

This result is of high importance due to the highly constricted nature of data. My proposed algorithm performs well in this data set, correctly identifying 15 clusters. The Silhouette plot compliments the results, as can be visually inspected.

7. Dim256 data set

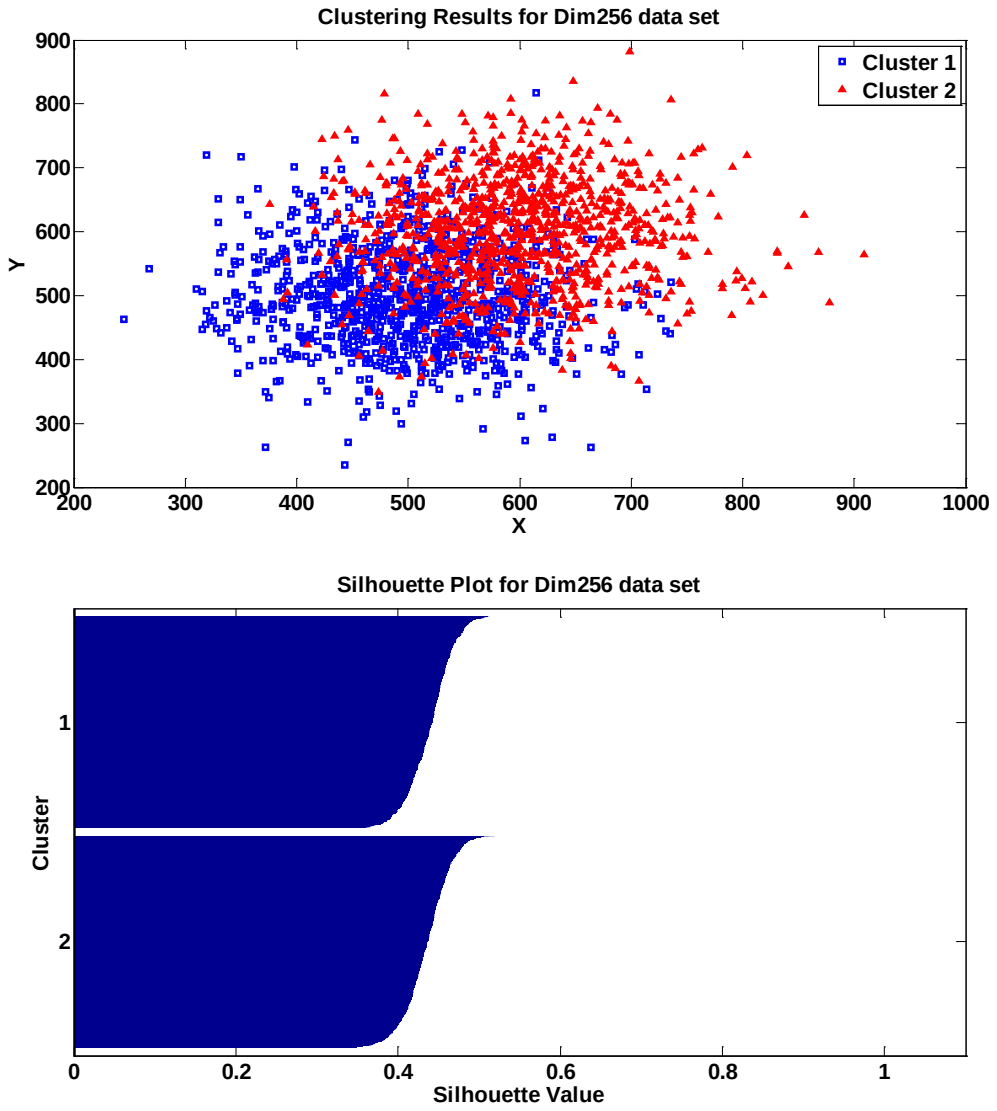


Figure 7: Clustering Result in Dim256 data set

This plot provides clustering results for Dim256 data set. This data set has 256 dimensions/features. The proposed method correctly identifies the desired number of clusters, validating its suitability in higher dimensional data sets. Furthermore, the Silhouette plot validates the clusters

8. Dim1024

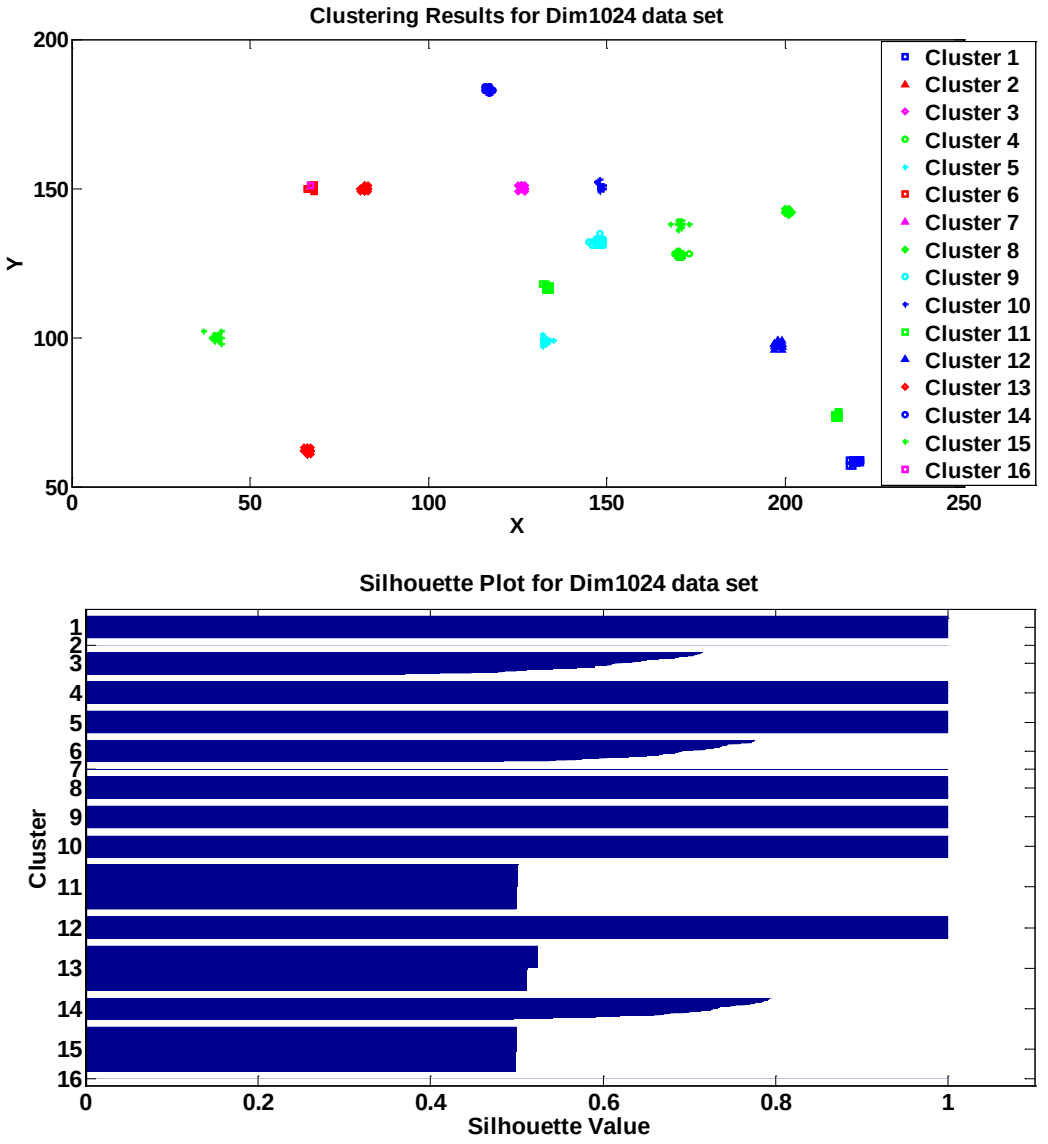


Figure 8: Clustering Result in Dim1024 data set

As expected, the algorithm correctly yields 16 clusters in 1024 dimensions validating the algorithm's performance in higher dimensions.

9. Synthetic 1 data set

10.

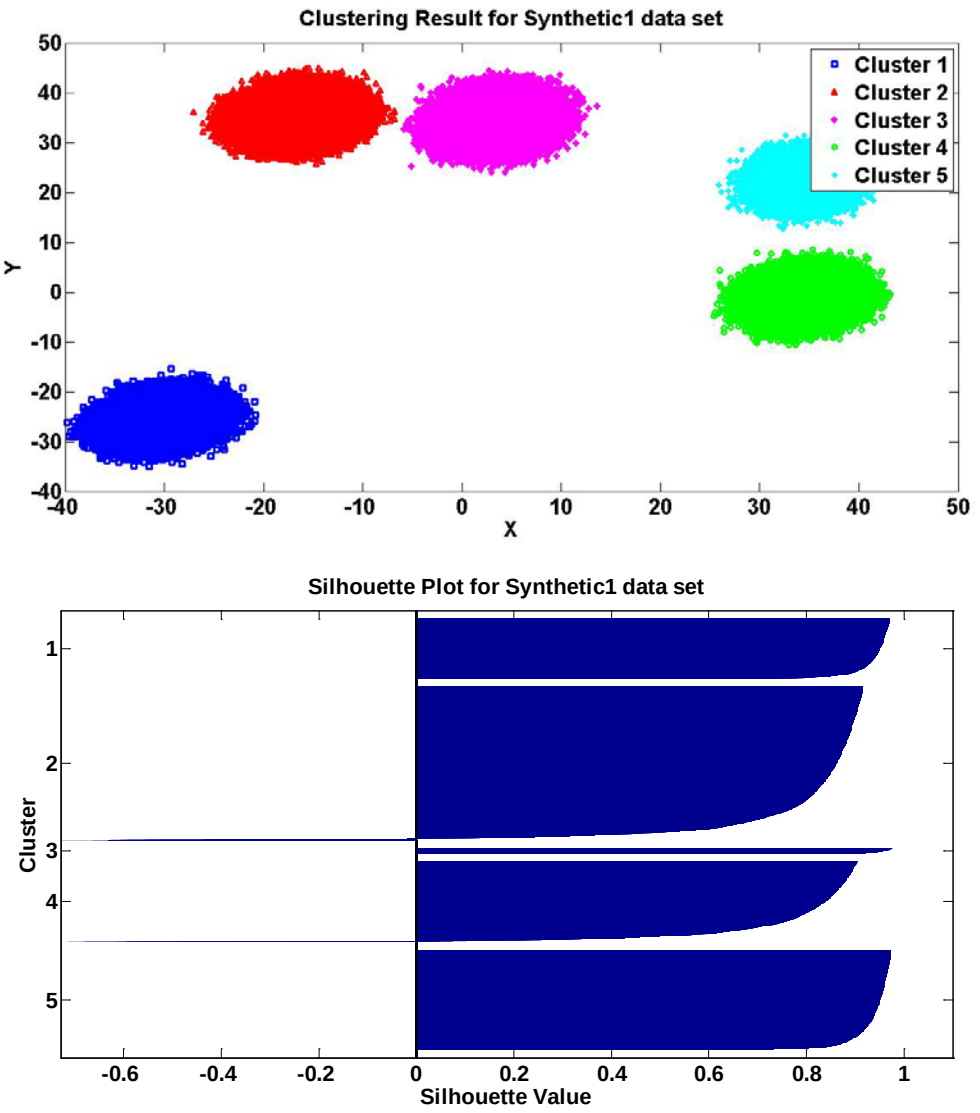


Figure 9: Clustering Results in Synthetic1 data set

The above plot shows the clustering plot for synthetic1 data set. The data set consists of 1 million data points. The aim of this plot is to provide validation of the proposed method in big data sets. The proposed algorithm correctly identifies 5 clusters. The Silhouette plot also confirms the same.

11. Synthetic4 data set

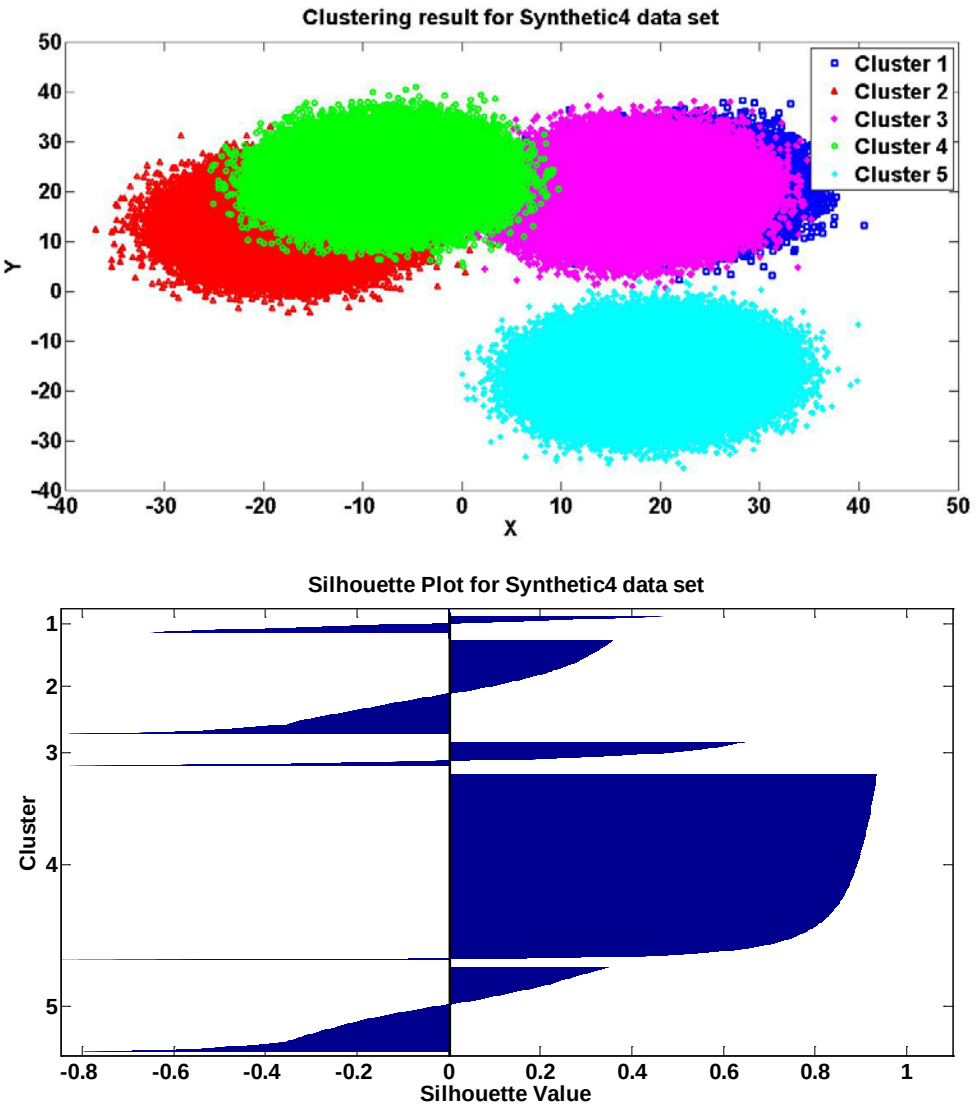


Figure 10: Clustering Result in Synthetic4 data set

This plot is highly critical from the point of clustering as the Silhouette plot indicates the maximum index when $k = 4$. For the same reason, I employed two other internal cluster validity metric, namely the CH index and the elbow plot. The highest average value of the CH index and Silhouette index is taken and k is chosen. The detailed values are listed in table 2.

It is to be noted that the cluster validity metrics rely on information of the data and hence, no single cluster validity metric can provide sufficient information on the desired number of clusters. All of them are data-driven.

5.1.2 Run time complexity

This metric is implemented to evaluate the time-complexity of the given approach. The results are compared with the time-complexity of three state-of-the-art algorithms. Our approach performs better than the other algorithms with respect to computational and processing time for most scenarios. Due to the random initialization phase of all the algorithms, we performed 20 different runs for each of the method and calculated the average time complexity.

Figure 11 provides computational time complexity of the given approach versus the number of data points.

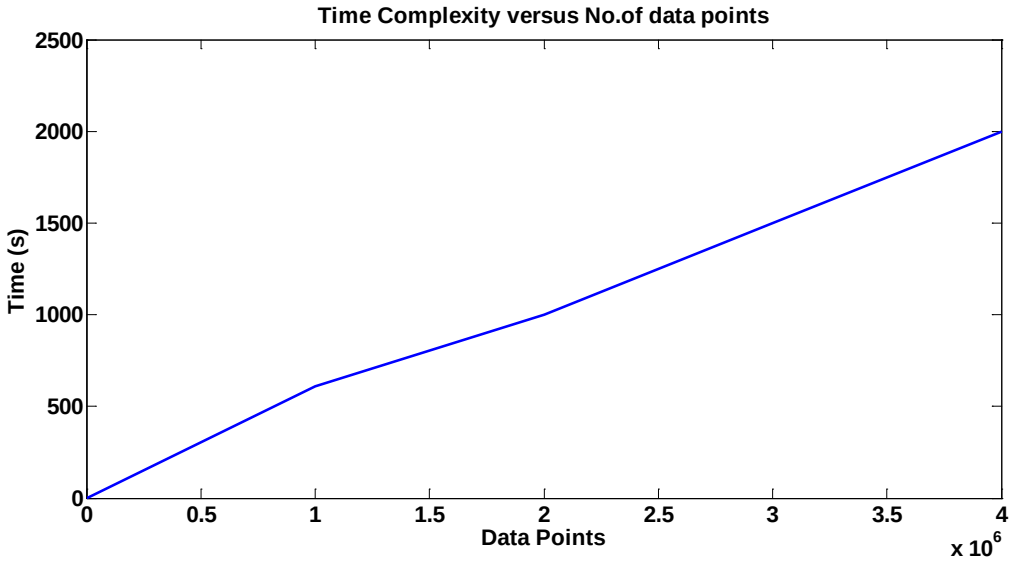


Figure 11: Time complexity versus Data Points

As expected, the time-complexity of the given approach proceeds nearly linearly with the number of data points. The given approach takes a maximum of 2000 seconds for the data set containing 5,000,000 points.

5.1.3 Comparison with the state-of-the art algorithms

Figure 12 provides the run time complexity of the given approach in comparison with three well-known clustering techniques viz. K-Means++, K- Medoids, and FCM. For the purpose of validation, we provide two separate plots for time-complexity versus the number of data-points and dimensionality. These plots individually show the dependence on the number of data-points and dimensionality. FCM runs in quadratic time, as expected. However, since it is the best-case comparison, we additionally provide the true value of *k* and calculated the average time over twenty multiple iterations.

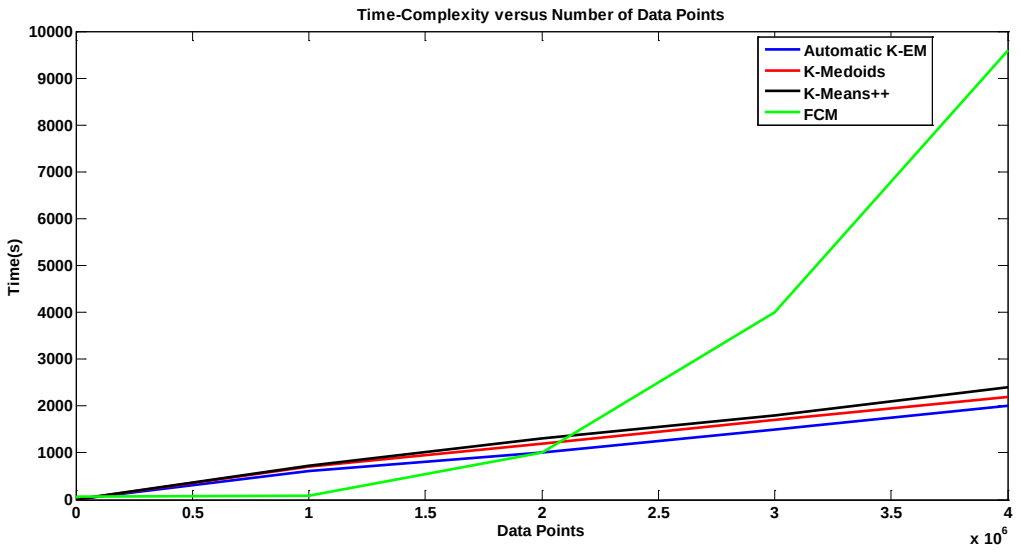


Figure 12: Comparison of average time complexity with state-of-the-art algorithms

It is quite evident from the plot that the proposed approach outperforms the other well-known techniques in terms of run time complexity. Due to the random initialization strategy of these algorithms, each algorithm is run multiple times and an average time complexity is calculated to provide unbiased results.

5.1.4 Convergence

As a widely known fact, the partitioning based algorithms suffer from convergence issues as they are based on some sort of statistical parameter like mean, centroid, median, etc. which inherently require a regular distribution of data and are also inherently designed to be random in nature. Due to these limitations, this performance metric was used to test the convergence of my algorithm. The following table lists the convergence vs the number of iterations for each of these methods. The proposed approach takes three iterations on an average to converge in comparison to the other techniques.

Table 3: Comparison of convergence with state-of-the-art clustering algorithms

Algorithms Tested	**Average No. of Iterations**	**Convergence Achieved?**
K-Medoids	6	Yes
FCM	20	Yes
K-Means++	20	No
A-KEM	3	Yes

5.1.5 Scalability

We implemented this performance metric to test the scalability of the proposed approach. For this purpose, we tested the algorithm on higher dimensional data sets. Figure 13 provides the results of scalability. We plotted the time in seconds versus the number of dimensions for each of the methods. The proposed approach scales well with the dimensions and yield good results.

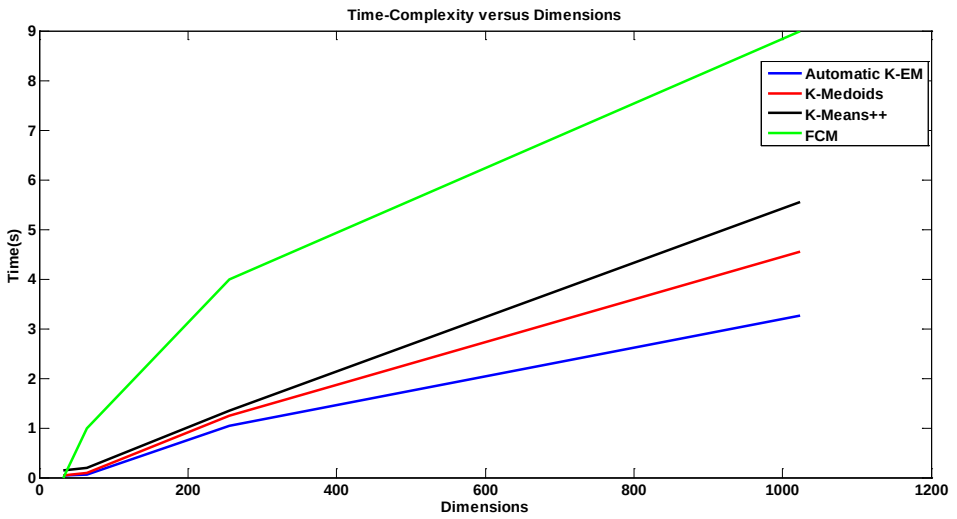


Figure 13: Comparison of Scalability with state-of-the art cluster techniques

6 Discussions

Based on the aforementioned performance metrics, it is quite evident that the proposed approach performs well and yields good results on data sets with different attributes and characteristics. Moreover, the given method outperforms the state-of- the-art K-Means clustering techniques in most scenarios. The results claim that the proposed method is efficient, accurate and is suitable for various degrees of data-sets, making it desirable for data mining applications.

7 Limitations

The proposed approach provides a non-parametric clustering algorithm for data-mining applications. As such, the given approach provides consistent results based on the internal cluster validity metrics. However, there is no concrete proof that these metrics yield perfect results on every data set. Additionally, since the

proposed approach is based on K-Means, there is a possibility that the algorithm might fail to converge under specific conditions imposed by the type of data set. Fundamentally, we have implemented an upper bound on the number of clusters to be $n^{1/2}$ as reported in the literature. It would be interesting to find a tighter upper bound to minimize the processing time when considering huge data sets.

8 Conclusions

In this paper, a novel non-parametric clustering technique is presented. The proposed method is non-parametric in a sense that it optimally identifies the number of cluster employing three internal cluster validity indices. The algorithm is tested on a wide array of data sets in terms of varying shape, size and dimensions. The results are also compared with three well-known state-of-the-art clustering techniques. The results prove that the proposed method yields good results and performs better than the other state-of-the-art techniques in most scenarios. The proposed algorithm is proven to run in linear time with a decent convergence rate in most tested scenarios. It scales well with higher dimensions making it apt for big data applications.

9 Future Work

In future work, it will be of interest to find a tighter upper bound on the number of clusters, in order to reduce the number of computational steps of the proposed approach. As a potential alternative, it would be interesting to exploit the triangle inequality following the method developed by Elkan in [40].

A possible improvement of the proposed method will consist of trying different similarity measures instead of Euclidean distance, which is implemented by default. This might aid in enhancing the cluster accuracy. For future work, we will also try to implement the algorithm on noisy data sets containing outliers. As the

partitioning based techniques are non-robust to outliers, it would be of interest to devise a technique which can automatically handle outliers while clustering. In [41], we have developed an onion-peeling outlier detection approach which might aid in achieving the same.

References

[1] Z. Huang, Extensions to the K-Means Algorithm for Clustering Large Datasets with Categorical Values, *Data Mining and Knowledge Discovery*, **2**, (1998), 283-304.

[2] O. Benli and A. Botsali, An Optimization-Based Decision Support System for a University Timetabling Problem: An Integrated Constraint and Binary Integer Programming Approach, 2004. Available from: http://www.csulb.edu/~obenli/research/benli-botsali.pdf [Oct 2005].

[3] J. Macqueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, CA, 1, (1967), 281-297.

[4] B. A. Rahumath and R. Remya, A Comparison of Clustering Techniques in Data Mining, *International Journal of Computing and Technology*, 1(4), (2014).

[5] P. Arabie and L. J. Hubert, *An overview of combinatorial data analysis, Clustering and classification*, World Scientific Publsihing Co., NJ, (1996), 5-63.

[6] D. Massart and L. Kaufman, *The interpretation of Analytical Chemical Data by the Use of Cluster Analysis*, John Wiley & Sons, New York, NY, (1983).

[7] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, NY, (1973).

[8] A. Jain and R. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ. (1988).

[9] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, Ney York, NY, (1990).

[10] R. Sibson, SLINK: An optimally efficient algorithm for the single link cluster method, *Computer Journal*, 16 (1973), 30-34.

[11] D. Fisher, An analysis of recent work on clustering algorithms, *Technical Report UW-CSE01-03-02*, University of Washington, (1973).

[12] S. Guha, R. Rastogi, and K. Shim, CURE: An efficient clustering algorithm for large Databases, *Proceedings of the ACM SIGMOD conference*, (1998), 73-84, Seattle, WA.

[13] G. Karypis, E.-H. Han, and V. Kumar, CHAMELEON: A hierarchical clustering algorithm using dynamic modelling, *COMPUTER*, 32, (1999a), 68-75.

[14] T. Mitchell, *Machine Learning*. McGraw-Hill, New York, NY, (1997).

[15] A. Dempster, N. Laird, and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, Series B, 39(1), (1977), 1-38.

[16] C. Wallace and P. Freeman, Estimation and inference by compact coding, *Journal of the Royal Statistical Society*, Series B, 49(3), (1987), 240-265.

[17] P. Cheeseman and J. Stutz, Bayesian Classification (Autoclass): Theory and Results, *Advances in knowledge discovery and data-mining*, AAAI press/MIT press, (1996).

[18] C. Frayley and A. Raftery, MCLUST: Software for model-based cluster and discriminant analysis, *The Computer Journal*, 41(8), (1999), 578-588.

[19] R. Ng and J. Han, Efficient and effective clustering methods for spatial data mining, *Proceedings of the 20th conference on VLDB*, (1994), 144-155, Santiago, Chile.

[20] J. Hartigan, *Clustering Algorithms*, John Wiley & Sons, New York, NY, (1975).

[21] J. Hartigan and M. Wong, Algorithm AS136: A k-means clustering algorithm, *Applied statistics*, 28, (1979), 100-108.

[22] H.-P. Kriegel, B. Seeger, R. Scheider, and N. Beckmann, The R*-tree: an efficient access method for geographic information systems, *Proceedings of the International Conference on Geographic Information Systems*, Ottawa, Canada, (1990).

[23] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander, OPTICS: Ordering points to identify clustering structure, *Proceedings of the ACM SIGMOD Conference*, 49-60, Philadelphia, PA, (1999).

[24] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, Density-based clustering in spatial Data-bases: the algorithm GDBSCAN and its applications, *Data Mining and Knowledge Discovery*, 2(2), (1984), 169-194.

[25] K. Omar, F. Ramdani, and B. Tadili, AK-Means: an automatic clustering algorithm based on K- Means, *Journal of Advanced Computer Science and Technology*, 4(2) (2015), 231-236.

[26] A. Adebisi, O. Olusayo, O. Olatunde, A. Bola, and A. Titilayo, Development of a hybrid K-Means-expectation maximization clustering algorithm, *Journal of Computations and Modelling*, 2(4), (2012), 1-23.

[27] B. Zhang, M. Hsu and U. Dayal, *K-Harmonic Means – A Data clustering Algorithm*, Software Technology Laboratory, HP Laboratories, Palo Alto, (1999).

[28] Z. Zhang, B.T. Dai, and A. Tung, On the Lower Bound of Local Optimums in K-Means Algorithm, *IEEE Computer Society*, (2006).

[29] M. Yan, *Methods of determining the number of clusters in a data set and a new clustering criterion*, Ph.D. thesis, Virginia Polytechnic Institute and State University, (2005).

[30] A. Vattani, The hardness of *k*-means clustering in the plane, available from: https://cseweb.ucsd.edu/~avattani/papers/kmeans_hardness.pdf

[31] N. Sara, A. Rawa, V. Gregory, A modified fuzzy k-means clustering using Expectation-Maximization, *IEEE International Conference on Fuzzy Systems*, Vancouver BC, (2006), 231-235.

[32] Y. Zhang, X. Li, Y. Ye, and X. Xu, Information-theoretic Agglomerative K-Means, *Information Technology Journal*, 10, (2011), 2420-2426.

[33] R. L. Thorndike, Who belongs in the family?, *Psychometrica* 18(4), (1953), 267-276.

[34] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Computational and Applied Mathematics*, 20, (1987), 53-65.

[35] T. Calinkski and J. Harabrasz A dendrite method for cluster analysis, *Communications in Statistics*, 3, (1974), 1-27.

[36] N. R. Pal and J. C. Bezdek, J.C., On cluster Validity for the Fuzzy c-Means Model, *IEEE Transactions on Fuzzy systems*, 3, (1995), 370-379.

[37] O. Kettani, B. Tadili and F. Ramdani, A deterministic k-means algorithm based on nearest neighbor search, *International Journal of Computer Applications*, 0975-8887, 63(15), (2013).

[38] E. Rendon, I. Abundez, A. Arizmendi and E. M. Quiroz, Internal versus External cluster validation indexes, *International Journal of Computers and Communications*, 5(1), (2015).

[39] A. Asuncion and D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA, (2007).

[40] C. Elkan, Using the triangle inequality to accelerate k-means, ICML 2003 Conference Proceedings, 147-153, (2003).

[41] A. Harsh, J. E. Ball and P. Wei, Onion-Peeling Outlier Detection in 2-D data-sets, *International Journal of Computer Applications*, 139(3), (2016).

[42] D. Arthur and S. Vassilvitskii, K-Means++: the advantages of careful seeding, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society of Industrial and Applied Mathematics, Philadelphia, PA, USA, (2007), 1027-1035.

[43] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, NY, (1981).

[44] J. C. Dunn, A fuzzy Relative of the ISODATA Process and Its use in detecting Compact Well-Separated Clusters, Journal of Cybernetics 3(3), (1973), 32-57.

[45] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD-96, AAAI Press, (1996), 226-231.

[46] T. Bayes and P. Richard, An essay towards solving a problem in the Doctrine of Chance, *Philosophical Transactions of the Royal Society of London*, (1763), 370-418.

[47] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, *The American Statistician*, 46(3), (1992), 175-185.

[48] Clustering datasets, Speech and Image Processing Unit, School of Computing, University of Eastern Finland, Joensuu, Finland.
URL: http://cs.joensuu.fi/sipu/datasets/

[49] C. J. Veenman, M. J. T. Reinders and E. Backer, A maximum variance cluster algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2002), 24(9), 1273-1280.

[50] I. Karkkainen and P. Franti, Dynamic local search algorithm for the clustering problem, *Research Report A-2002-6*, 2006.