INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

An Effective Initialization Method Based on Quartiles for the K-means Algorithm

Trushali Jambudi[1]*, **Savita Gandhi**[2]

1 Research Scholar, Department of Computer Science, Gujarat University, India
2 Former Professor and Head, Department of Computer Science, Gujarat University, India

OPEN ACCESS

Received: 08-04-2022

Accepted: 30-07-2022

Published: 07-09-2022

Citation: Jambudi T, Gandhi S (2022) An Effective Initialization Method Based on Quartiles for the K-means Algorithm. Indian Journal of Science and Technology 15(35): 1712-1721. https://doi.org/10.17485/IJST/v15i35.714

* **Corresponding author**.

talk2trushali@gmail.com

Funding: None

Competing Interests: None

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: This study aims to speed up the K-means algorithm by offering a deterministic quartile-based seeding strategy for initializing preliminary cluster centers for the K-means algorithm, enabling it to efficiently build high-quality clusters. **Methods:** We have investigated various cluster center initialization approaches in literature and presented our findings. For the K-means algorithm, we here propose a novel deterministic technique based on quartiles for finding initial cluster centers. To obtain the preliminary cluster centers, we have applied our suggested approach to the data set. The initial cluster centers determined by our suggested method are then entered into the K-means algorithm. The proposed seeding method is evaluated on sixteen benchmark clustering data sets: five synthetic and eleven real data sets. Python is used to run the simulation. **Findings:** Based on empirical results from experiments, it is evident that our proposed cluster center initialization method allows the K-means algorithm to form clusters with SSE values comparable to the minimum SSE values produced by repeated Random or K-means++ initializations. Furthermore, our deterministic initialization strategy assures that the K-means algorithm converges faster than the Random and K-means++ initialization techniques. **Novelty:** In this study, we explore the potential of quartile-based seeding as a technique of accelerating the K-means algorithm. Needless to add, as our seeding method is deterministic, the requirement to run K-means repeatedly with different stochastic initializations is completely eliminated. Also, our initialization strategy assures that there is remarkable saving in execution time as compared to the Random and K-means++ initialization techniques. Moreover, it is found that after initializing with our offered method, the solution obtained with just a single run of K-means produces optimal clusters. **Applications:** Our proposed seeding technique will be helpful for initializing the K-means algorithm in time-sensitive applications, applications managing large amounts of data, and applications looking for deterministic cluster solutions.

Keywords: Kmeans Algorithm; Initialization Method; Speeding Kmeans; Quartiles; Clustering; Deterministic Initialization Method

1 Introduction

For clustering data into meaningful groups, K-means is a widely used technique. The K-means clustering algorithm groups the input N data points into k clusters based on the similarity of data so as to reduce to minimum the sum of squared distance between each data point and the cluster center nearest to it. In the beginning, K-means requires k, i.e., the number of clusters to be formed as input while values of the initial k required by the algorithm are selected randomly. The initial k centroids which are randomly initialized are then amended by the two-step process of assignment and update. The assignment step consists of assigning every point to the cluster with the centroid nearest to that point, whereas in the update step the center of each of the k clusters is recomputed as the mean of all the data points in that cluster. These two steps of assignment and update make up one iteration of K-means algorithm (1–3). K-means is well-liked in a variety of study fields due to its simplicity, adaptability, and effectiveness. For instance, Gouho et al. used the K-means technique in (3) to normalize the noise-polluted diagram, after which they entered the new modified diagram into a hidden multi-layered Neural Network. Health, marketing, business, finance, and social studies are some other fields in which K-means is applied.

The fundamental issue in obtaining optimal clustering results in the partitioning of algorithms like K-means is the seeding selection mechanism. In reality, the K-means clustering framework's seeding procedure is the first stage and a key factor in the optimization process. The K-means algorithm, while highly popular for quickly and efficiently being able to cluster large data, is also sensitive to the selection of initial centers of the k clusters to be formed. K-means does not assure unique clustering since we get different results when the centers are randomly initialized. If the initial centroids are not chosen correctly, it might result in empty clusters, slower convergence and a greater chance of being stuck in bad local minima. Since it is commonly known that the best clustering outcomes may be obtained with the right seeding selection mechanism, various studies have been done on the subject. The K-means++ technique is one of the most well-liked seeding strategies suggested to deal with this issue for the K-means algorithm (4,5). Fig. 1 shows the plot whileTable 1 shows the CPU Time, SSE and Silhouette score for single run as well as 10 runs of K-means algorithm with Random and K-means++ cluster center initialization methods for R15 (6) data set.

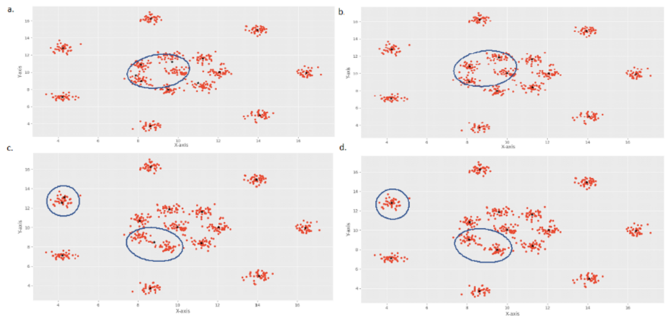


Fig 1. Comparison of Cluster formations for R15 data set by K-means algorithm initialized with Random and K-means ++ methods a. Clusters formed for R15 data set by K-means algorithm when initialized once with Random method, b. Clusters formed for R15 data set by K-means algorithm after 10 Random initializations, c. Clusters formed for R15 data set by K-means algorithm when initialized once with K-means++ method, d. Clusters formed for R15 data set by K-means algorithm after 10 K-means++ initializations.

Table 1. K-means clustering results with Nondeterministic center initialization methods

Data Set	Initialization Method	Number of Runs	CPU Time (seconds)	SSE	Silhouette score
R15	Random	1	0.25	174.961	0.67
		10	0.4	108.62	0.75
	K-means++	1	0.21	162.90	0.68
		10	2.10	108.62	0.75

The CPU Time, SSE, and Silhouette score for a single run and ten runs of the K-means algorithm utilizing two nondeterministic center initialization approaches, Random and K-means++.

From Fig. 1 and Table 1 It is evident that when we initialized and executed the K-means algorithm just once using non-deterministic center initialization methods like Random and K-means++, we did not get clusters of optimal quality given by SSE value and Silhouette score. Moreover, there is considerable difference in the best and worst results given by the K-means algorithm due to the high variability in the initial centers selected by the random and K-means++ initialization methods. Hence, to get optimal cluster quality we have to initialize and execute the K-means algorithm multiple times and select the best result but the total time required to achieve the results increases with the number of runs. From this we can conclude that the selection of initial centers affects the quality of clusters formed by the K-means algorithm and for non-deterministic center initialization methods such as Random and K-means++, the performance of K-means algorithm can be considerably improved by running the algorithm many times. However, while for a small value of k, it is possible to run K-means several times with different initial center values and selecting the best result; but, as the value of k increases, a proper stable initialization method which would yield deterministic results is required. Moreover, there is no mention in literature regarding the number of runs that are required to achieve optimal results. In literature, the authors have executed non-deterministic methods for 50 to 100 runs to get best results. The number of required runs depends on the data set and is often left to the user who may not make an optimal choice, again justifying the need for a deterministic initialization method.

Various deterministic and non-deterministic initialization methods have been analyzed and their characteristics explained in literature(5,7). While, non-deterministic initialization methods often produce highly variable results across multiple runs and need to be executed multiple times and the output of the run that gives the least SSE is taken as the result; deterministic methods produce identical results across multiple runs and need to be executed only once. Therefore, the total computational cost of a non-deterministic method is often significantly higher than that of a deterministic method. Moreover, the outcomes of clustering may be less optimal in case of nondeterministic seeding. Hence, in real world applications, deterministic initialization methods are desirable.

During the past few years, a number of seeding methods have been proposed, however, the issue of seeding selection continues to be a concern for research. Franti and Sieranoja(4) carried out a thorough investigation into the seeding selection problem and divided it into six categories: random points, farthest point heuristic, sorting heuristic, density-based, projection-based, and splitting technique. Despite their efforts, it was unclear which category or seeding technique is more effective. Seman and Sapawi(5) makes use of a seeding method that takes advantage of zero-point multidimensional spaces. This seeding approach seems to be very well suited and to work solely for the k-AMH clustering framework in order to achieve cluster optimality and stability. However, may not be helpful in case of the fuzzy c-means and k-means frameworks as these employ the mean as their center of gravity. In(7), an empirical comparison is carried out among the various stochastic and deterministic methods of centroid initialization including random and K-means++, for the K-means variations on synthetic data. They demonstrate that, on average, more advanced initialization strategies reduce the requirement for intricate clustering strategies. Deterministic techniques outperform stochastic techniques, too. While repeated application of simpler stochastic approaches can produce superior grouping, deterministic approaches can be competitive and produce a decent clustering solution when execution time is taken into account. In research by Fahim Ahmed(8) an algorithm is proposed to select a good value for k and initial centers. Their algorithm includes a step of pre-processing to get the value of k and the initial centers before the K-means algorithm is applied. The initial clusters are formed using a density-based method that relieves the user from providing the number of clusters and in each formed cluster its mean value is used as the initial center in K-means. Two parameters required by the DBSCAN method are also taken as input in this method. The computation time increases with increase in the data set size. Ahmed Fahim(9) has proposed a method which identifies the correct value of k and also the initial centers for K-means algorithm. This proposed method aims to improve the cluster quality and the number of iterations of the K-means algorithm. DBSCAN method is used as a pre-processing step which does not ask for k as input and produces cluster centers which are taken as initial centers for the K-means algorithm. In(10) quantile values of the data are used to initialize K-means in each sub-division step, making it non-random in nature. The user inputs five parameters into the algorithm. Therefore, to apply this approach, the user must possess extensive subject expertise. If the input parameters are not carefully set for the proposed DWMB and the results are provided with only a few fixed input parameters, the method may not perform well. Yang Jie et al. provide an adaptive initialization approach (AIMK) for the K-means algorithm in(11). The method has a high temporal complexity but can adapt to different dataset attributes. It is challenging to use AIMK on big or high-dimensional datasets because of the temporal complexity. The AIMK-RS, which is based on random sampling and reduces the time complexity of the AIMK, is presented as a solution to this issue. The average performance of 100 runs is taken as the actual performance of the AIMK-RS because Random Sampling is utilized and the AIMK-RS is stochastic. In(12) Andrei Ciuparu and Raul Mureşan offer Gradient-k, an update to the k-means algorithm that enhances clustering precision and lowers the quantity of convergence iterations needed. The computational scalability of Gradient-k to the dataset's dimensionality is poor. Gradient-k is not practical for datasets with large dimensionality since the computational complexity grows exponentially with dimensionality. They conclude that in order to improve speed, accuracy, and the number of parameters, it is preferable to develop a more intelligent initialization that allows the starting points' locations to be more informed. A new clustering algorithm developed by A detailed comparison with 8 different initialization techniques on 32 real and 12,228 synthetic datasets is carried out in(13); It was established that random centroids and Max-min algorithms often perform below par and that there are considerably better alternatives with equivalent computational requirements. However, their results do not clearly point out a single technique as being consistently better than others. The literature assessment shows that the seeding approach needs to be enhanced further, and efforts to achieve the essential optimality, notably for k-means, are still under investigation.

From the review of the related works, the following is concluded:

• Centers from highly dense regions are good representatives.

• Centers should be far from each other to maximize inter-cluster distances.

• Within cluster SSE should be minimized.

• The algorithm should require a minimum CPU time to converge to enable it to scale well even in case of large data sets.

• Deterministic initialization methods perform better than non-deterministic methods due to their lower computation cost and are therefore preferable when dealing with large data sets.

• When initialized with a non-deterministic initialization method, the difference in the best and the worst results achieved by the K-means algorithm is very high, acknowledging the need for a deterministic initialization method which would ensure stable results.

There is no definitive conclusion in the literature about the best seeding technique or category, hence there is no practical answer, however efforts are still being made in this direction. In this paper we propose a new deterministic initialization method based on quartiles named Quartile based deterministic Initialization Method for K-means algorithm (QuIK) to find better initial centroids for the K-means algorithm. The initial centers are chosen by our proposed method from the k densest locations, which finds dense areas based on quantiles. We have measured the performance of QuIK using quality measurement criteria as well as efficiency (speed) assessment criteria. Experiments show that our method guarantees faster convergence along with optimal cluster formations and being deterministic it gives constant results across multiple runs, and so needs to be executed only once unlike the non-deterministic methods which produce highly variable results across multiple runs thereby requiring to be executed multiple times.

2 Methodology

Our proposed method QuIK, takes as input the data set to be clustered and the number of clusters to be formed (k), and gives k initial cluster centers as output these cluster centers are input in the traditional K-means algorithm for forming k clusters. Figure 2, shows this methodology of our proposed algorithm.

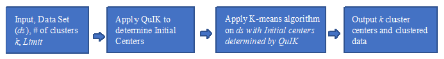

Input, Data Set (*ds*), # of clusters *k*, *Limit*
Apply QuIK to determine Initial Centers
Apply K-means algorithm on *ds with Initial centers determined by QuIK*
Output *k* cluster centers and clustered data

Fig 2. Methodology of Proposed work

Next, we explain the concepts related to understanding our method and thereafter we describe in detail our proposed method for determining initial cluster centers for K-means algorithm.

2.1 Related Concepts for QuIK

Here we explain the background concepts required for understanding QuIK:

1. **The K-means algorithm**[2]

Input: Dataset, *k*

Output: *k* means (one mean for each cluster)

1. Select k initial centers randomly
2. Assign each object to its closest center
3. Compute new means for clusters
4. Repeat steps 2 and 3 until there is no change in any of the cluster means

2. **Quartiles**

An important measure that provides suggestions regarding a distribution's center and spread is Quartiles. Q1, Q2 and Q3 are the first, second and third quartiles. Where Q1 is the lowest 25% of the data, Q2 is the median or center of the data and Q3 is the lowest 75% or highest 25% of the data. The Q1, Median and the Q3 together give good information in case of both symmetric as well as skewed data distributions and hence can prove to be good representatives of a data's centers[2].

3. Bins

For an attribute, *a*, the data distribution of *a* is partitioned into disjoint subsets which are denoted as buckets or bins[2]. Frequently, buckets denote continuous ranges for the given attribute.

2.2 The Quartile based deterministic Initialization method for the K-means algorithm (QuIK):

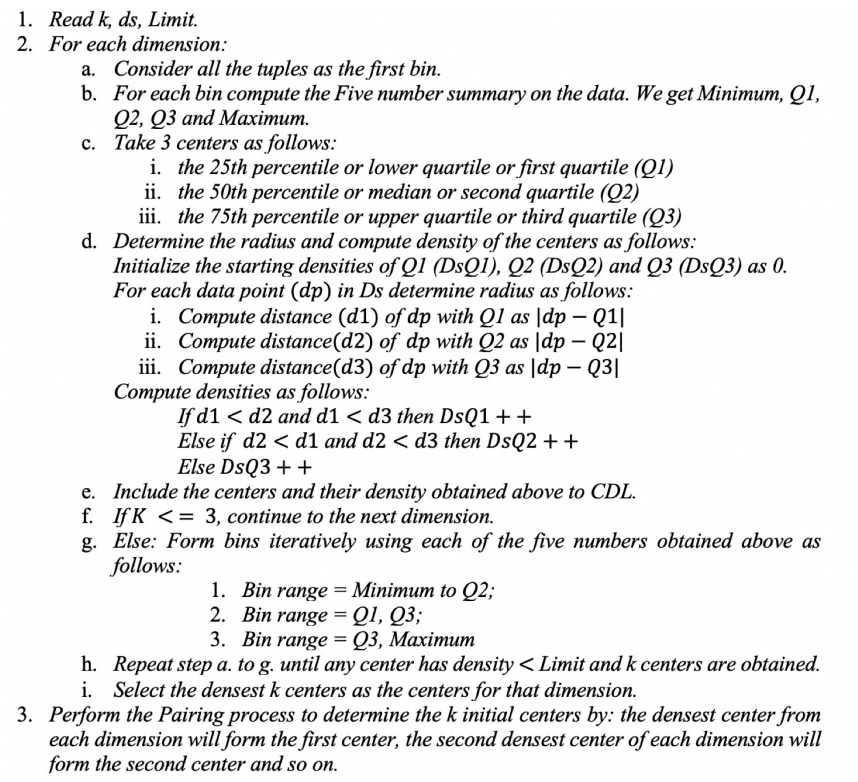
The initial centers algorithm is as follows:

Inputs:

- *Data Set (Ds) to be clustered*
- *k = number of clusters to be formed*
- *Limit = A percentage of number of records in (ds)*
- ***Note:*** *We recommend that the value of Limit be taken as 10%.*

Output: *k initial centers*

Method:



```
1. Read k, ds, Limit.
2. For each dimension:
    a. Consider all the tuples as the first bin.
    b. For each bin compute the Five number summary on the data. We get Minimum, Q1,
       Q2, Q3 and Maximum.
    c. Take 3 centers as follows:
        i. the 25th percentile or lower quartile or first quartile (Q1)
        ii. the 50th percentile or median or second quartile (Q2)
        iii. the 75th percentile or upper quartile or third quartile (Q3)
    d. Determine the radius and compute density of the centers as follows:
       Initialize the starting densities of Q1 (DsQ1), Q2 (DsQ2) and Q3 (DsQ3) as 0.
       For each data point ($dp$) in Ds determine radius as follows:
        i. Compute distance ($d1$) of dp with Q1 as $|dp - Q1|$
        ii. Compute distance($d2$) of dp with Q2 as $|dp - Q2|$
        iii. Compute distance($d3$) of dp with Q3 as $|dp - Q3|$
       Compute densities as follows:
            If $d1 < d2$ and $d1 < d3$ then $DsQ1 ++$
            Else if $d2 < d1$ and $d2 < d3$ then $DsQ2 ++$
            Else $DsQ3 ++$
    e. Include the centers and their density obtained above to CDL.
    f. If $K <= 3$, continue to the next dimension.
    g. Else: Form bins iteratively using each of the five numbers obtained above as
       follows:
            1. Bin range = Minimum to Q2;
            2. Bin range = Q1, Q3;
            3. Bin range = Q3, Maximum
    h. Repeat step a. to g. until any center has density < Limit and k centers are obtained.
    i. Select the densest k centers as the centers for that dimension.
3. Perform the Pairing process to determine the k initial centers by: the densest center from
   each dimension will form the first center, the second densest center of each dimension will
   form the second center and so on.
```

2.3 Proposed Method Description:

The algorithm takes as input data set, k: the number of clusters to be formed and Limit: the minimum percentage of tuples required in a cluster. For each dimension we first compute the values of Q1, Q2 and Q3 which represent the 25th percentile value, 50th percentile value and 75th percentile value from the input data. These three quartile values are selected as the first 3 centers. We then compute the density for each of these centers. We include the centers and their densities obtained above in the center density list (CDL). If $k \leq 3$, we continue processing for the next dimension as we already have 3 centers and their density; else we proceed iteratively to form bins until the stopping criterion is met. From each bin we determine 3 centers as Q1, Q2, and Q3 and compute density for these centers and we add the centers and their density to the CDL The stopping criterion for $k > 3$ is that when we get a Center with density less than Limit and if k centers are obtained in the CDL. In this case the algorithm sorts the centers in the CDL by their density and returns.

• Forming Bins: In each iteration we compute the three quartiles i.e., Q1, Q2 and Q3. In the first iteration, these three values act as the first three centers and also determine the first bin. We compute the distance of each data point in the bin to each of these three quartiles and a data point is assigned to its closest quartile. We use the quartiles obtained above to form new bins and thereby obtain more centers as follows: Bin 2 = [Minimum, Q2]; Bin 3 = [Q1, Q3]; and Bin 4 = [Q2, Maximum]. For the three centers obtained here we again compute density and include the center and its density in the CDL. This process is repeated until we find a center with density less than Limit and k centers are obtained.

• Measuring Distance: The Distance between an object p and Quartile Q, for the i^{th} dimension, denoted by $dist\ (p_i,\ Q_i)$, is computed as $\min\left({}_i - Q1_i|\ , {}_i - Q2_i|\ , {}_i - Q3_i| \right)$ (1)

• Radius - For each bin we have the minimum value, maximum and Q1, Q2 Q3 as center values. Compute the distance of each point in the bin with the minimum, maximum and center values of that bin. Assign the data point to the radius of the center if it is closest to the center value. All the points close to a center will form the center's radius.

• Computing density: A centers' density is determined by the number of points in the centers' radius.

• Our stopping criterion "Limit" – We recommend the value of limit be set to 10%. The value for parameter Limit is also required as input from the user. This parameter Limit is used as a stopping criterion and helps restrict the number of iterations. When in algorithm we arrive at a stage where a center has a density less then Limit, we stop. However, if at least k centers are not in the center density list, our algorithm will ignore the value of Limit and will not terminate till k centers are added in the center density list.

• Pairing - The densest quartile of each dimension will form the first center, the second densest quartile of each dimension will form the second center and so on.

• Reduced computation – Our algorithm is based on the distance of data points to prospective centers. This distance is determined by computing the difference of each data point in a Bin with that bin's quartiles. Hence, for each point in a bin we need to compute the distance with only 3 values and this reduces the computation considerably. Also, where $k \leq 3$, our algorithm requires only one iteration in each dimension as in the first iteration itself we have Q1, Q2 and Q3 as the first 3 centers.

• From each bin, the Quartiles are chosen as centers which represent the spread of data and can prove to be good estimates of cluster centers.

3 Results and Discussion

3.1 Evaluation Criteria

SSE, Silhouette score, Purity and CPU time are used to measure the performance of our proposed method. While SSE, Purity and Silhouette score measure the quality of the clusters formed, CPU Time measures efficiency(7,13–15). These evaluation criteria are explained below:

1. **Sum of Squared Error (SSE)**: The sum of squared error (SSE) determines within cluster variation**,** between all objects in cluster C_i and the center c_i, and can be used to measure the quality of cluster C_i For each object in each cluster, the distance from the object to its cluster's centroid is squared, and the distances are totaled(2).

The SSE value would be 0 in case where all observations in a cluster are same. Hence we need to achieve a low value for SSE so as to decrease dissimilarity within a cluster. If we get a high value of SSE, it would mean that there is high dissimilarity within that cluster(2,14,15).

2. **Purity**: It is an external validity index that measures the degree of similarity between the clustering solution formed by a clustering method and that specified by the given class labels(7).

3. **Silhouette Score**: The mean Silhouette coefficient $s(i)$, measures the correctness of a data point's allocation to a cluster. The value of $s(i)$ lies between the range -1 and 1. The data point can be said to be correctly clustered if we get the value of $s(i)$ near 1. If the value of $s(i)$ is near -1, it would be appropriate to assign the data point to its neighboring cluster. An $s(i)$ near 0 means overlapping clusters(7).

4. **CPU Time**: CPU time or the processing time measured in seconds is the total time taken by the Initial center determination stage and the clustering stage. A low value is desirable as it indicates better algorithm efficiency(10,13).

5. **Number of Iterations**: When initialized by a certain Initialization Method, this is the number of iterations that K-means has to complete before achieving convergence. It is an efficiency metric that is independent of the CPU architecture, compiler, language used for programming, and implementation style(13).

3.2 Data Sets Description

We have carried out experiments on a number of synthetic and Real clustering data sets. The synthetic data sets used are A2, D31, R15, S3 and S4(6). We also have used the real datasets Iris, Thyroid disease, Seeds, Wine, Ecoli, Data User Modelling, Shuttle, Person Activity, MAGIC Gamma Telescope(16) and two wine quality data sets(17). These selected data sets are the standard clustering data sets and are suitable for testing the efficiency of our proposed method as these include data sets of varied characteristics including data sets with varied value for k as well as a varied number of dimensions and small as well as large number of instances. Table 2 describes the characteristics of these data sets.

Table 2. Description of Data Sets and Limit value used

Data set name	Number of instances	Number of attributes	Number of Clusters	Limit (%)
A2	5250	2	35	05
D31	3100	2	31	05
R15	600	2	15	15
S3	5000	2	15	10
S4	5000	2	15	10
Iris	150	5	3	
Thyroid disease	215	5	3	
Seeds	210	7	3	
Wine	180	13	3	
White Wine Quality	4898	11	7	30
Red Wine Quality	1600	11	4	35
E Coli	336	8	8	15
Data User Modelling	403	5	4	25
Person Activity	1,64,860	3	11	24
Magic Gamma Telescope	19,020	10	2	
Shuttle	58,000	9	7	12

3.3 Experiments and Discussion

All simulations of our proposed method as well as the K-means algorithm are carried out in Python. As explained in operational flow above, QuIK takes as input ds: the data set to be clustered, the number of clusters k and Limit: the percentage of number of instances in ds that is the minimum required density in each cluster and produces k initial centers as output. These initial centers are then fed as input in the traditional K-means algorithm.

In Table 2 we have given the value of Limit for each data set used in the simulation of our proposed method. As the number of clusters increases, the density of each cluster i.e., the number of points in each cluster decreases. Hence, we need to set a low value for Limit. We can have a high value for Limit where the number of clusters is less. In cases where $k \leq 3$, the value of Limit does not have any impact as our algorithm will execute only 1 iteration for each dimension. As mentioned earlier, the centers are not entirely dependent on Limit as in case k centers are not available in the CDL, our algorithm will ignore the value of Limit and not return. However, in the case of a low value of k, convergence is faster when we set Limit to a high value. For example, as you can see in Table 2, in the Red Wine Quality data set, $k = 4$ so we were able to set Limit = 35%, whereas for A2 and D31 data sets, k is 35 and 31 respectively so we set Limit = 5%.

The CPU time, SSE, Silhouette score and Purity obtained by taking 1 run for K-means centers initialized with the centers produced by QuIK method are compared with that of the K-means centers initialized with Random and K-means++ center initialization methods for 10 runs as both the Random and K-means++ initialization methods are non-deterministic. Table 3 shows this comparison of CPU time and the number of iterations. Here the CPU time is the Total CPU time for determining and initializing the initial centers and generating clusters thereafter by K-means algorithm. The number of iterations in Table 3 is the total of 10 runs in case of Random and K-means++ initialization methods whereas the QuIK method being deterministic needs to be executed only once so there we have given the number of iterations for a single run. Figure 3 shows the chart of comparison of CPU time given in Table 3 and Figure 4 shows the chart of comparison of number of iterations given in Table 3. This shows that for all datasets except the shuttle data set, the K-means algorithm requires less CPU time for convergence when initialized with QuIK method and the QuIK method enables the K-means algorithm to converge in considerably fewer number of iterations for all the data sets in the experiment.

Therefore, from Table 3, Figure 3 and Figure 4, it is clear that the CPU time and number of iterations required by K-means algorithm are reduced considerably when initialized with the cluster centers determined by QuIK method compared to centers initialized by K-means++ method and Random method. In Table 4 we compare the cluster quality formed when the centers for the K-means algorithm are initialized by QuIK method with that of the clusters formed by initializing the K-means with K-means++ method and Random method. This is measured by executing the K-means algorithm once with centers initialized by QuIK method and for 10 runs each with centers initialized by Random method and K-means++ method and selecting the outcome with minimum SSE as results from these 10 runs. In Figure 5 we show the chart of comparison of Silhouette score

Table 3. Comparison of CPU Time

Data set name	CPU Time (Seconds)			Number of Iterations		
	QuIK	Random	K-means++	QuIK	Random	K-means++
A2	0.297	2.68	6.12	12	190	76
D31	0.201	2.34	5.10	11	187	106
R15	0.074	1.44	2.46	6	90	38
S3	0.2	2.00	3.40	11	236	84
S4	0.203	2.06	3.57	15	297	173
Iris	0.09	1.30	1.50	8	66	63
Thyroid disease	0.092	1.39	1.18	10	96	87
Seeds	0.09	1.30	1.10	4	81	52
Wine	0.15	1.74	1.64	5	72	98
White Wine Quality	0.47	2.06	4.78	27	271	340
Red Wine Quality	0.28	1.83	2.96	40	400	319
E Coli	0.174	2.00	2.24	10	175	133
Data User Modelling	0.12	1.40	1.80	18	193	177
Person Activity	3.96	7.3	6.48	16	714	459
Magic Gamma Telescope	0.3	3.6	3.3	24	254	248
Shuttle	5.05	2.83	1.12	19	238	51

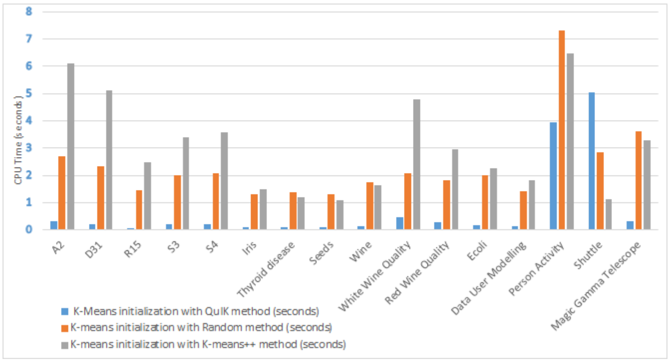


Fig 3. Comparison of Total CPU Time

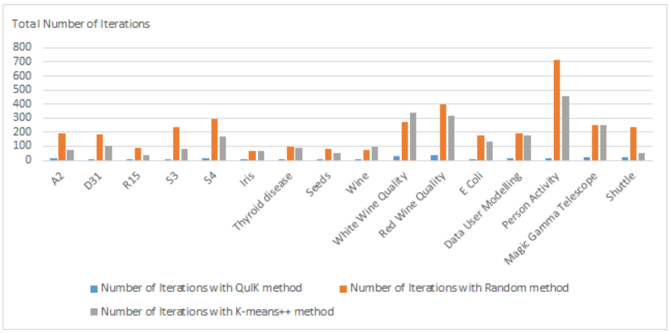


Fig 4. Comparison of Total Number of Iterations

given in Table 4.

Table 4. Cluster quality evaluation

Data set	SSE			Silhouette Score			Purity		
	QuIK	Random	K-means++	QuIK	Random	K-means++	QuIK	Random	K-means++
A2	2.2E+10	2.3E+10	2.03E+10	0.58	0.53	0.60			
D31	4182.56	4223.72	3393.31	0.55	0.55	0.58			
R15	108.62	157.36	108.62	0.75	0.68	0.75			
S3	1.9E+13	1.7E+13	1.7E+13	0.47	0.49	0.49			
S4	1.8E+13	1.7E+13	1.6E+13	0.45	0.46	0.47			
Iris	78.94	78.94	78.94	0.55	0.55	0.55	0.89	0.89	0.89
Thyroid disease	4.6E+14	4.6E+14	4.6E+14	0.6	0.6	0.6	0.88	0.87	0.87
Seeds	588.78	587.32	587.32	0.47	0.47	0.47	0.89	0.89	0.89
Wine	2E+06	2E+06	2E+06	0.57	0.57	0.57	0.7	0.7	0.7
White Wine Quality	1E+06	1E+06	1E+06	0.32	0.32	0.32	0.45	0.45	0.45
Red Wine Quality	283161	283161	283161	0.49	0.49	0.49	0.49	0.49	0.49
E Coli	15.5	14	13.95	0.31	0.24	0.25	0.83	0.82	0.83
Data User Modelling	70.3	68.9	68.9	0.18	0.19	0.2	0.53	0.62	0.63
Person Activity	31625.5	31595.7	31596.64				0.46	0.46	0.46
Magic Gamma Telescope	2E+08	2E+08	2E+08	0.43	0.43	0.43	0.65	0.65	0.65
Shuttle	8.4E+08	7E+08	4.35E+08	0.38	0.41	0.98	0.84	0.83	0.79

SSE, Silhouette score, and Purity values for K-means algorithm initialized with QuIK method are from a single run of the K-means algorithm, but for K-means algorithm initialized with Random and K-means++ method, these values are the best values from 10 runs of the K-means algorithm.

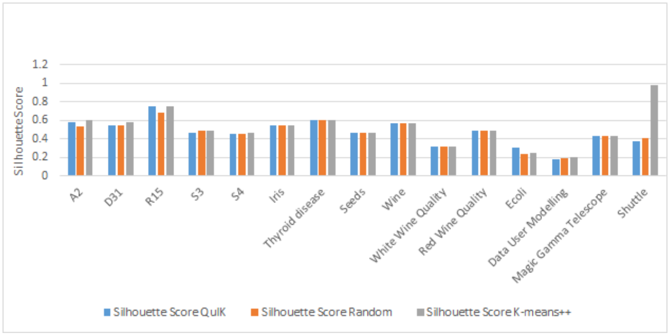


Fig 5. Comparison of Silhouette Score

From Table 4 and Figure 5 it is clear that the quality of clusters produced by single run of K-means when initialized with our proposed method QuIK is as good as the quality of clusters produced by the K-means algorithm with Random and K-means++ initialization methods with 10 runs.

4 Conclusion

In this paper we present an original novel method named QuIK for generating k initial centers for the K-means algorithm. Our proposed deterministic seeding method, accelerates the K-means algorithm and enables it to efficiently build high-quality clusters. When initialized with the initial centers determined by QuIK, the K-means algorithm can rapidly produce good quality clusters with values of SSE, Purity, and Silhouette score comparable to the best results obtained by multiple Random as well as K-means++ initializations, as evidenced by empirical results drawn from experiments on synthetic and real benchmark datasets.

As per the results obtained, in all except one data set, the K-means algorithm requires significantly less number of iterations and CPU time when initialized with the QuIK approach compared to when initialized with the Random and K-means++ methods. The minimum time reduction achieved by our proposed method is for Person activity data set, which is 45.75 % against Random initialization method and 38.89% against K-means++ initialization method; whereas the maximum time reduction achieved is for R15 data set, which is 94.86 % against Random initialization method and 96.99 % against K-means++ method. The average execution time reduction achieved is 70.31 % against Random method and 67.94% against K-means++ method. Moreover, unlike the stochastic initialization methods where a good number of runs (50 – 100) are needed, our novel QuIK technique has to be executed only once as it is deterministic in nature, making it appropriate for real world applications. As a result, the seeding selection would no longer be a barrier to achieve the best clustering outcomes for each run. Moreover, the QuIK technique through the parameter limit, ensures that no empty clusters arise because the initial centers are picked to match the minimum density criteria. The parameter Limit ensures that each of the specified initial centers has a minimal number of points within its proximity. The suggested QuIK has the prospective to be employed in the development of clustering tools in the future, especially for numerical clustering because of its optimality and constancy.

References

1) Jambudi T, Gandhi S. A New K-means - Based Algorithm for Automatic Clustering and Outlier Discovery. In Information and Communication Technology for Intelligent Systems. Singapore. Springer. 2019. Available from: https://doi.org/10.1007/978-981-13-1747-7_44.
2) Han J, Pei J, Tong H. Data Mining Concepts and Techniques. In: and others, editor. The Morgan Kaufmann Series in Data Management Systems. Elsevier. 2022. Available from: https://www.elsevier.com/books/data-mining/han/978-0-12-811760-6.
3) Gouho JB, Karim S, Aka B, Babri M. Automatic Modulation Classification Based on In-Phase Quadrature Diagram Constellation Combined with a Deep Learning Model. *Indian Journal of Science and Technology*. 2020;13(2):200–212. Available from: https://doi.org/10.17485/ijst/2020/v13i02/148648.
4) Fränti P, Sieranoja S. K-means properties on six clustering benchmark datasets. *Applied Intelligence*. 2018;48(12):4743–4759. Available from: https://doi.org/10.1007/s10489-018-1238-7.
5) Seman A, Sapawi AM. An Optimal and Stable Algorithm for Clustering Numerical Data. *Algorithms*. 0197;14(7):197–197. Available from: https://doi.org/10.3390/a14070197.
6) Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*. 2009;47(4):547–553. Available from: https://doi.org/10.1016/j.dss.2009.05.016.
7) Vouros A, Langdell S, Croucher M, Vasilaki E. An empirical comparison between stochastic and deterministic centroid initialisation for K-means variations. *Machine Learning*. 2021;110(8):1975–2003. Available from: https://doi.org/10.1007/s10994-021-06021-7.
8) Fahim A. K and starting means for k-means algorithm. *Journal of Computational Science*. 2021;55:101445–101445. Available from: https://doi.org/10.1016/j.jocs.2021.101445.
9) Fahim A. Finding the Number of Clusters in Data and Better Initial Centers for K-means Algorithm. *International Journal of Intelligent Systems and Applications*. 2020;12(6):1–20. Available from: https://doi.org/10.5815/ijisa.2020.06.01.
10) Rehman AU. Samir Brahim Belhaouari. Divide well to merge better: A novel clustering algorithm. *Pattern Recognition*. 2022;122. Available from: https://doi.org/10.1016/j.patcog.2021.108305.
11) Jie Y, Yu-Kai W, Xin Y. Lin Chin-Teng. Adaptive Initialization Method for K-Means Algorithm. *Frontiers in Artificial Intelligence*;4. Available from: https://www.frontiersin.org/article/10.3389/frai.2021.740817.
12) Ciuparu A, Mureșan RC. Gradient-k: Improving the Performance of K-Means Using the Density Gradient. *bioRxiv*. 2022;03(30):486343. Available from: https://doi.org/10.1101/2022.03.30.486343.
13) Celebi ME, Kingravi HA, Vela PA. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*. 2013;40(1):200–210. Available from: https://arxiv.org/pdf/1209.1960.pdf?ref=https://githubhelp.com.
14) Jambudi T, Gandhi S. Analytical review of K-means based algorithms and evaluation methods. In: 12th International Conference on Advances in Computing, Control, and Telecommunication Technologies. 2021;p. 479–486. Available from: https://thegrenze.com/pages/servej.php?fn=8_1.pdf&name=AnalyticalReviewofK-MeansbasedAlgorithmsandEvaluationMethods&id=834&association=GRENZE&journal=GIJET&year=2021&volume=7&issue=1.
15) Jambudi T, Gandhi S. Analysing the effect of different Distance Measures in K-means Clustering Algorithm. *GLS KALP-Journal of Multidisciplinary Studies*. 2021;1(3):49–56. Available from: http://glskalp.in/index.php/glskalp/article/download/15/11.
16) Dua D, Graff C, Uci. Machine Learning Repository. Irvine, CA: University of California. School of Information and Computer Science. 2019. Available from: http://archive.ics.uci.edu/ml.
17) 11p, Cortez A, Cerdeira F, Almeida T, Matos J, Reis. 11P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. 2009. . Available from: http://www3.dsi.uminho.pt/pcortez/wine/.