Journal of Big Data

RESEARCH Open Access

Big data: an optimized approach for cluster initialization

Marina Gul[1] and M. Abdul Rehman[1*]

*Correspondence:
rehman@iba-suk.edu.pk

[1] Department of Computer Science, Sukkur IBA University, Sukkur, Pakistan

Abstract

The k-means, one of the most widely used clustering algorithm, is not only faster in computation but also produces comparatively better clusters. However, it has two major downsides, first it is sensitive to initialize *k* value and secondly, especially for larger datasets, the number of iterations could be very large, making it computationally hard. In order to address these issues, we proposed a scalable and cost-effective algorithm, called *R-k-means,* which provides an optimized solution for better clustering large scale high-dimensional datasets. The algorithm first selects *O(R)* initial points then reselect *O(l)* better initial points, using distance probability from dataset. These points are then again clustered into *k* initial points. An empirical study in a controlled environment was conducted using both simulated and real datasets. Experimental results showed that the proposed approach outperformed as compared to the previous approaches when the size of data increases with increasing number of dimensions.

Keywords: k-means, Cluster initialization, Large scale data

Introduction

Clustering plays a crucial role in unsupervised learning, encompassing a wide array of applications across various domains. The primary goal of clustering is to organize data in a manner that promotes grouping of similar datapoints within the same clusters, while ensuring that dissimilar clusters are well-separated. It is presumed that the number of clusters and the initial points are known. There are several clustering techniques developed to find the patterns in unlabeled data such as Partition based clustering [1], Hierarchical Agglomerative clustering (HAC) [2], DBSCAN [3], Gaussian Mixture Models (GMM) [4], and Spectral Clustering [5].

The clustering problem is commonly defined as the problem of minimizing objective function while clustering the datapoints. The most commonly used objective function in clustering is sum of squared error (SSE) [6], which is computed as the squared distance between the datapoints and their respective clusters centroid. Thus, the aim of the objective function is to find the clusters with minimum internal variance. Generally, the selection of an appropriate objective function depends on the specific problem, data characteristics, and the desired outcome of the clustering task.

Springer Open

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

The k-means [7] is one of the simplest and familiar clustering algorithms, based on Lloyd's algorithm [8]. It works by dividing the data points into k clusters, where k is specified by the user. The algorithm assigns each data point to the cluster whose mean (centroid) is closest to the datapoint. The k-means clustering is a popular and easy-to-implement algorithm that can be used for a variety of applications, such as image processing [9] and image segmentation [10], data summarization [11], text clustering [12] and sound source angle estimation [13]. However, it has some limitations, including the need to specify the number of clusters in advance, and the sensitivity of the results to the initial placement of the centroids.

Although k-means widely used in clustering, its non-probabilistic nature and adoption of a simple radial distance metric to assign cluster membership make it challenging in terms of performance, especially for high dimensional large scale datasets [14]. It thus really becomes challenging for existing k-means based clustering algorithms, in the big data domain [15], to cluster the data in an optimal way [16]. One of the main problems with k-means is the cluster initialization as the initial selection of centroids greatly influences the performance of K-means. Different initializations can lead to different final clustering results. If the initial centroids are randomly chosen, the algorithm may converge to a suboptimal solution or get stuck in local optima. Due to its iterative nature of k-means, the algorithm may converge to local optima rather than the global optimum. Poor initialization can also lead to clusters with unequal size. There has been a significant emphasis in recent research on cluster initialization methods specifically designed for large-scale, high-dimensional datasets as better initialization dramatically improved the performance of Lloyd iteration in terms of convergence and quality [17].

A significant advancement in this direction was made by k-means++ algorithm [18], in which the initial point is chosen randomly, and subsequent points are selected using probability distribution that ensures the selected center is dissimilar to the ones already chosen. The downside of k-means++ is that the initialization phase requires k sequential passes over data. This is because the selection of new points relies on the previously selected points, making it challenging to parallelize. Another approach, known as k-means|| [19], proposed a variant of k-means++, specifically designed for parallel initialization of cluster centroids. The algorithm speeds up the process of k-means++ by sampling l times more points in each round independently. Independent sampling speeds up the process of the initialization but the quality of selected centers is not good as k-means++ leading to increase in Lloyd iteration for convergence.

In this work, we propose a variant of the k-means++ clustering algorithm, which is comparatively scalable and cost-effective for clustering large scale high-dimensional datasets. Keeping k-means|| as a baseline, the proposed algorithm introduces one more optimization factor R, which ensures the selected clusters are far away from each other.

The proposed algorithm aims to minimize the problem of centroid local minima by proposing an $R = \theta k$ optimization factor. The R is an optimization factor that selects the center more than desired centers i.e. $R \geq k$. The main idea behind our algorithm is to select $O(k)$ points in each round and pass these points to k-means++ to select $O(l)$ points that are far away from each other. The process repeats for $O(logn)$ times and finally leaves with $O(llogk)$ points which then again reclusters into k points. The algorithm guarantees to reduce the cost of Lloyd's step.

In order to evaluate the proposed algorithm, an experimental evaluation is conducted using some real and artificial datasets to compare the clustering quality of *R-k-means* with the state-of-the-art clustering techniques. The SSE, a very popular internal evaluation metric, is calculated before and after initialization process. For statistical evaluation, the one-way analysis of variance (ANOVA) is used to determine whether there is any significant difference between the performance of the proposed algorithm as compared to k-means++ and k-means||.

The main contributions of this study are the following:

- In "The proposed algorithm: R-k-means" section, we introduce a scalable algorithm named R-k-means, specifically designed for clustering large-scale datasets. Our algorithm incorporates a novel factor R, which enhances the initialization process in k-means, leading to improved clustering results.
- In "Algorithm evaluation" section, we present an empirical evaluation of our proposed algorithm, showcasing its effectiveness in the context of clustering large datasets. The evaluation encompasses various datasets, highlighting the algorithm's performance and its capability to handle big data.
- In "Result validation" section, we provide a statistical evaluation of the performance of our proposed algorithm.

The rest of the paper is organized as follows; in "Related work" section we present a detailed discussion of related works to show the research gap. In "The proposed algorithm: R-k-means" section, we present our proposed algorithm with detailed illustration. In "Empirical evaluation of proposed algorithm" section, we discuss the results of our empirical experiments. Finally, in "Conclusion" section conclusion is presented.

Related work

The problem of Clustering has been addressed in a variety of contexts. There are several different variants for the k-means algorithm available in the literature, covering from initial k parameter selection to generating proper "seeding" with different objective function and data reduction schemes to reduce the number of iterations.

The k-means clustering has emerged as one of the most rated data mining algorithms [20] due to its simplicity and ease of usage. However, the algorithm usually influenced by the number of clusters and how each cluster is initialize. In general the validity indices can be used to find the optimal number of clusters, divided into two categories: internal and external index [21]. External indices uses the prior structure or reference results label to find the number of clusters [22, 23]. Internal indices used the internal data for finding the goodness of cluster structure. Silhouette Width (SW), Dunn's index, Davies–Bouldin index (DB), Bayesian information criteria, Calinski and Harabasz index (CH) and Gap statistic are popular internal validity indices for k-means clustering algorithm. Other approaches are also proposed in literature, MM [24], U-k-means [25], X-means [26], G-means [27] uses validity indices as a model and range of cluster numbers.

Cluster with different k initial values produce different clustering results especially for large scale datasets. In many k-means investigations, the best initial values of k are determined using a subsample of the data. The continuous k-means algorithm [28] selects

referenced points as a random sample from the whole datasets and in each iteration examines only random sample of datapoints. The algorithm is faster but the result is not necessarily global minimum. In [29] the k-means Mod algorithm is applied on the *J* random subsample of dataset to choose *k* centroids and again run k-means algorithm on selected points for each subsample. The final k centers are chosen based on the minimal distortion value. The algorithm perform better for small datasets. Similarly, the algorithm [30] divides each attribute of the datasets into *k* fixed number of cluster and compute the percentile. The attribute values calculated using mean and standard deviation that serves as the seed for that attributes. The density-based data condensation method is used to merge the resulting centroids into *k* cluster. The algorithm reduces the cost of Lloyd's step however handling high dimensional data is challenging. In [31] greedy deletion procedure is used to select *k* centroids from the bulk of random points in the dataset. The Ball-k-means seeding step that considers the ball of radius around each center and moves the center to the centroid of the ball, is used to obtain the final centers. They also showed that Lloyd's algorithm performs well if the data satisfies a natural separation condition of clustering and return optimal clustering. The algorithm provides the optimal initial centers that required no or minimum Lloyd iteration. But the overall running time is $O(nkd+3kd)$. In the same context [32], the input datasets is divided into *m* number of groups and runs k-means++ in each group. The algorithm selects $3log(k)$ points in each iteration and at the end, it reclusters these $3mlog(k)$ points into *k* using s scheme or any of the k-means algorithm. The advantage of this algorithm over k-means++ is that it can be implemented in parallel, as each group of input is assigned to different machines but the running time of partition does not improve when the number of machines surpasses the threshold. In [33], the data is sampled from t-mixture distribution. The t-mixture distribution is heavy tailed Gaussian distribution. This t-mixture model based distributed data is then analyzed from the aspect of loss function. The proposed method is stable in terms of variance of multiple results.

Some approaches also focus on the overall computational complexity associated with the Lloyd's step in k-means algorithm. The QuicK-means [34], which is based on the Fast Transform by reducing the complexity of applying linear operators in high dimension by approximately factorizing the corresponding matrix into few sparse factors. The approach more focuses on fast convergence of clusters and hence optimizes Lloyd's steps, however ignoring cluster initialization. The Ball k-means algorithm [35] divides different cluster, represents as ball, into active, stable and annular area. The distance calculation is performed only on annular area of neighboring clusters. Another notable approach presented in a literature I-k-means-+ [36], which iteratively remove and divide pair of clusters and perform re-clustering. The solution used to minimize the objective function of clustering. The PkCIA [37] computes initial cluster centers by using eigenvector as an indexes. The approach enable to identify meaningful clusters.

To overcome the issue of accelerating the clustering process, many parallelization techniques are employed. The parallel k-means clustering algorithm [38] use MapReduce framework to handle large scale data clustering. The map function assigns each point to closest center and reduce function updates the new centroids. To demonstrate the wellness of algorithm, different experiments perform on scalable datasets. Another MapReduce based method [39], reduces the MapReduce job as it uses one MapReduce

job to select k initial centers. The k-means++ initialization phase runs on mapper and the weighted k-means++ runs on the reducer phase. It overcomes the problem of k-means|| to run multiple Map-Reduce jobs for initialization. Parallel batch k-means [40] divide the dataset into equal partition by preserving the characteristics of the data. The k-means apply on each partition to reduce computational complexity of big dataset but not provide accurate no of clusters. In the same environment, two initialization methods for the k-means [41] were proposed. The first method used the divide and conquers strategy on k-means|| approach using subset sampling. In the second approach random projection was used along with subsampling, to project high dimension space into lower dimension space and then initialization perform. The algorithm guarantees to perform better than state-of-the-art methods. In [42], another recent entropy based initialization method is proposed. The algorithm uses Shannon's entropy based objective function for similarity measure. The proposed algorithm also aims to detect the optimal no of k for faster convergence. In [43], the random initialization method is merging the bootstrap technique. First, the algorithm applies k-means to B number of bootstrap replications of data and selects k initial centers from each bootstrap dataset. Then clustering is performed on $B*k$ set of centers, to get the k new clusters. Instead of selecting the average points, the deepest point is considered a cluster center. The algorithm aims to perform better than the previous proposal algorithm of initialization. In [44] an algorithm named as pattern-based clustering for categorical datasets, uses MFIM (Maximal frequent item sets mining) algorithm to find list of MFIs for initial cluster. Then it uses a kernel density estimation (KDE) method to estimate the local density of datapoints for the formation of cluster. Another technique [45] employs KDE, to create the balance between majority and minority clusters by estimating the better approximation of the distribution. In general, KDE based clustering techniques perform well for data with complex distribution however require high computation. In [46] the density based clustering algorithm (DBSCAN) used as a preprocessing step, to find the initial cluster center before applying k-means algorithm. In [47] K-means9+ model the comparison steps after randomly chosen centroids improved, by comparing with the current and eight nearest neighbor cluster partitions. The algorithm improve the efficiency by reducing the unnecessary comparison. In [48] the new algorithm FC-K-means improved the clustering performance by preventing some cluster centroids from updating in all iteration by fixing them on real world condition. In [49] power k-means++ the combination of power k-means and k-means++ presented to improve the clustering performance. The algorithm utilizes the k-means++ for good initial starting points then final alternative cluster centers using power k-means algorithm.

In summary, the aforementioned approaches highlight the strengths and limitations of different variants of k-means, aiming to enhance the overall performance of clustering. However, it is important to note that no single approach is universally applicable to all situations, and there are still numerous research gaps and challenges that need to be addressed. One significant challenge is the cluster initialization step, as the selection of initial centroids profoundly impacts the performance of k-means. Different initialization methods can lead to varying clustering outcomes, with random initialization often resulting in suboptimal solutions or local optima convergence. Inadequate initialization can also lead to clusters of unequal sizes. Recent research has placed considerable

emphasis on developing cluster initialization methods tailored for large-scale, high-dimensional datasets. Improved initialization techniques, such as the k-means|| [19], have shown good results in terms of convergence and clustering quality. The paper proposes an algorithm that aims to minimize the problem of centroid local minima, ensuring cost-effective and comparatively scalable improvements in the initialization phase to achieve better clustering results with comparatively good performance, especially for large scale high-dimensional datasets.

The proposed algorithm: *R-k-means*

The algorithm, known as *R-k-means* (*k*, *l*, *R*), is a variant of k-means++ inspired by k-means|| for initializing the centers. While the proposed algorithm is largely inspired by k-means||, it also uses an oversampling factor l and proposed optimization factor R. In (1) step the proposed algorithm chooses l, R constants and k number of desired clusters. It then picks an initial center (say, uniformly at random) and computes the $\psi \leftarrow \phi X(C)$ i.e. the sum of all smallest 2-norm distances (Euclidean Distance) from all points set X to all points from C. In other words, for each point in X, the algorithm will find the distance to the closest point in C. In the end, it computes the sum of all those minimal distances, one for each point in X. It then runs $log(\psi)$ iterations as mentioned in (3) step. In each iteration, it selects $l*R$ center points using probability distance measurement and then runs $log\ (l*R)$ times and reclusters the selected C' point into l points by using k-means++ to ensure that intra-cluster distance between points is far away from each other. In each iteration, the algorithm includes selected points from C'' into C. After the completion of the iteration, the algorithm reclusters the selected weighted points into k clusters. For reclustering of Step 8 k-means++ is used.

Algorithm 1 *R-k-means* ($\{x_1, x_2, \ldots, x_n\}, k, l, R$)

1. Select a sample point x_i randomly from X
2. Calculate the $\psi \leftarrow \varphi X(C)$
3. **for** $log(\psi)$ times **do**
4. C' = Sample $R*l$ points using distance probability $R.l.D(x)^2 / \varphi X(C)$
5. C'' = Select l points from C' using k-means++
6. $C = C \cup C''$
7. **end for**
8. Set weighting to the points in C
9. Recluster the weighted points in C into k clusters

Empirical evaluation of proposed algorithm

In this section, the results of *R-k-means*, k-means++, and k-means|| have been analyzed on 08 different datasets using the same control environment.

Experimental setup

The sequential version of the k-means algorithm is evaluated on a single machine quad-core 2.5 GHz processor and 16 GB memory. The parallel version of the algorithm is run

by using a Hadoop cluster of 40 nodes, created on Microsoft Azure, each with 16 GB of memory. The datasets used in the experiment are discussed in the next section.

Datasets

The datasets that are used in the experiments are the same as those are used in [18, 19, 29] algorithms. A number of the datasets analyzed in previous work were not particularly large, the main objective of the proposed algorithm is to cluster large datasets, which are difficult to fit in the main memory. Three large datasets ActivityRecognition, 3DRoadNetwork, and the AlltheNews datasets along with 3 benchmark datasets are used for the experiments. These datasets are taken from the UCI Machine learning repository. Two other synthetic datasets are also used. The Summary of all datasets is presented in Table 1.

Optimal number of *k*

As discussed before, in k-means clustering the number of clusters *k* is already randomly selected prior to running the algorithm. There are different ways to determine the right number of *k*. To demonstrate the performance and quality evaluation of the proposed algorithm in a more transparent manner, we select the initial value of *k* that fits the data. To determine which number of clusters *k* is more optimum for the dataset, or find cluster fitness, two well-known techniques on a random subset (samples) of data are used, i.e., the Silhouette Score and Elbow Method using SSE. These methods are standard evaluation methods for choosing the optimum number of clusters.

An Elbow analysis is used to visually observe the number of clusters in each dataset. The Fig. 1 demonstrates the number of *k* (x-axis) for each dataset against computed SSE values. An optimum number of *k* can be obtained with minimum SSE value. It should be noted here that after determining the range of *k* from 2 to 14 according to the empirical rules, WCSS (Within-Cluster Sum of Square) is the sum of the squared distance between each point and the centroid in a cluster is calculated for each value of *k*. When plotting the WCSS against the number of clusters, the resulting graph exhibits an elbow shape. The point where the graph's slope exhibits a sudden change indicates the optimal number of clusters. For example, in the case of the iris dataset, k = 3 represents the optimal number of clusters, while for the News dataset, k = 5 is deemed optimal.

Table 1 Summary of dataset

Dataset	No of instances	Dimensions
Iris	150	4
Sonar	208	60
Wine	178	13
Activity recognition (AR)	43,930,257	16
3D road network (3DSN)	434,874	4
All the news (News)	143,000	184,933
Norm10	10,000	5
Norm25	10,000	15

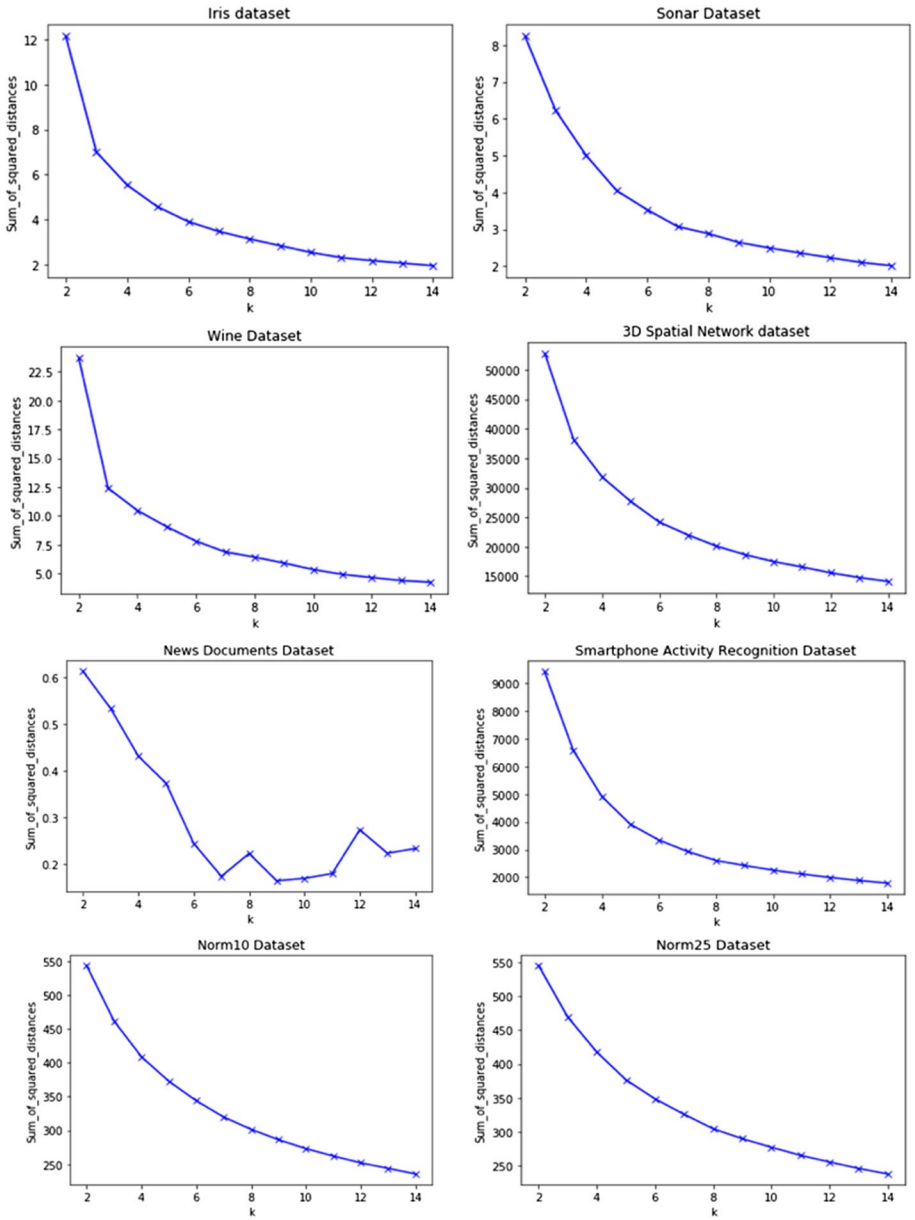

Fig. 1 Elbow analysis on different dataset

The Table 2 presents Silhouette analysis to cross validate the result of Elbow method. This technique provides a measure of how well each data point fits within its assigned cluster and aids in determining the number of clusters. The Table 2 shows the Silhouette score of each dataset. The highest silhouette coefficient value suggests that the point is well-matched to its own cluster and poorly matched to another cluster. For instance, values like 0.84 (k = 2) and 0.75 (k = 3) in 'iris' dataset. It is important to note that in our analysis, we examined the results of both techniques and selected the most appropriate value for *k* that satisfies both approaches.

Table 2 Silhouette analysis on different dataset

Dataset/k	2	3	4	5	6	7	8	9	10	11	12	13	14
Iris	0.75	0.84	0.68	0.60	0.56	0.50	0.44	0.43	0.47	0.48	0.47	0.43	0.46
Sonar	0.33	0.31	0.30	0.27	0.26	0.28	0.26	0.24	0.22	0.22	0.21	0.20	0.19
Wine	0.81	0.82	0.72	0.73	0.71	0.73	0.71	0.68	0.67	0.66	0.68	0.67	0.66
3D SN	0.83	0.84	0.73	0.85	0.78	0.76	0.72	0.75	0.67	0.71	0.68	0.6	0.70
News	0.61	0.53	0.43	0.67	0.24	0.17	0.22	0.16	0.16	0.17	0.27	0.22	0.23
AR	0.71	0.47	0.63	0.74	0.64	0.63	0.57	0.56	0.56	0.57	0.52	0.48	0.51
Norm10	0.23	0.24	0.24	0.23	0.23	0.24	0.27	0.22	0.23	0.28	0.23	0.23	0.22
Norm25	0.10	0.10	0.19	0.19	0.10	0.08	0.09	0.07	0.09	0.09	0.08	0.06	0.06

Algorithm evaluation

In order to demonstrate a comparative evaluation, all eight datasets were utilized to compute and compare the objective functions, namely inter-cluster and intra-cluster sum of squared errors (SSEs), for three algorithms: k-means++, k-means||, and the proposed algorithm R-k-*means*, using various threshold values for l and R factors ("The proposed algorithm: R-k-means" section). The evaluation of these approaches was conducted in two phases, which include the initialization phase and the final cluster formation phase, as better initialization leads to improved cluster formation. In initialization phase of *R-k-means*, multiple data points, say R, are drawn in each iteration C_i from the dataset and producing l estimates of the true cluster locations using k-means++. To find the best initial centroids, these l points (C solutions, each having l clusters) are weight into k centroids in an "optimal" fashion.

The Fig. 2 and Table 3 demonstrate the evaluation of initialization phase using inter-cluster SSEs (y-axis), threshold value of l (x-axis), indicating dissimilarities between clusters. In the smaller datasets, k-means|| demonstrates good performance in the 'iris' dataset with SSE = 61.608 when $l = 4$. However, *R-k-means* performs well with SSE = 84.63 when $l = 6$ and R = 10 in the same dataset. In the 'sonar' dataset, *R-k-means* outperforms other algorithms with SEE = 50.9349 at $l = 5$ and $R = 15$. The performance improvement is even more significant, with higher SSEs, in the larger document datasets (3D SN, News, and Activity), where *R-k-means* consistently outperforms k-means|| and k-means++ algorithms.

Figure 3 and Table 4 illustrate the evaluation results of the final cluster formation phase using the intra-cluster SSEs (y-axis) and the threshold value of l (x-axis), indicating similarities within clusters. In all datasets, *R-k-means* outperforms other algorithms by achieving the smallest SSE values when larger values of l and R are selected. This suggests that *R-k-means* algorithm consistently produces better quality clusters compared to the other approaches.

Result validation

To conduct a statistical evaluation of the results obtained from the proposed algorithm compared to k-means++ and k-means||, a one-way ANOVA test is employed. The null hypothesis (H_0) in Eq. (4) assumes that there is no improvement in the clustering results, and all approaches perform equally. The alternative hypothesis (H_A)

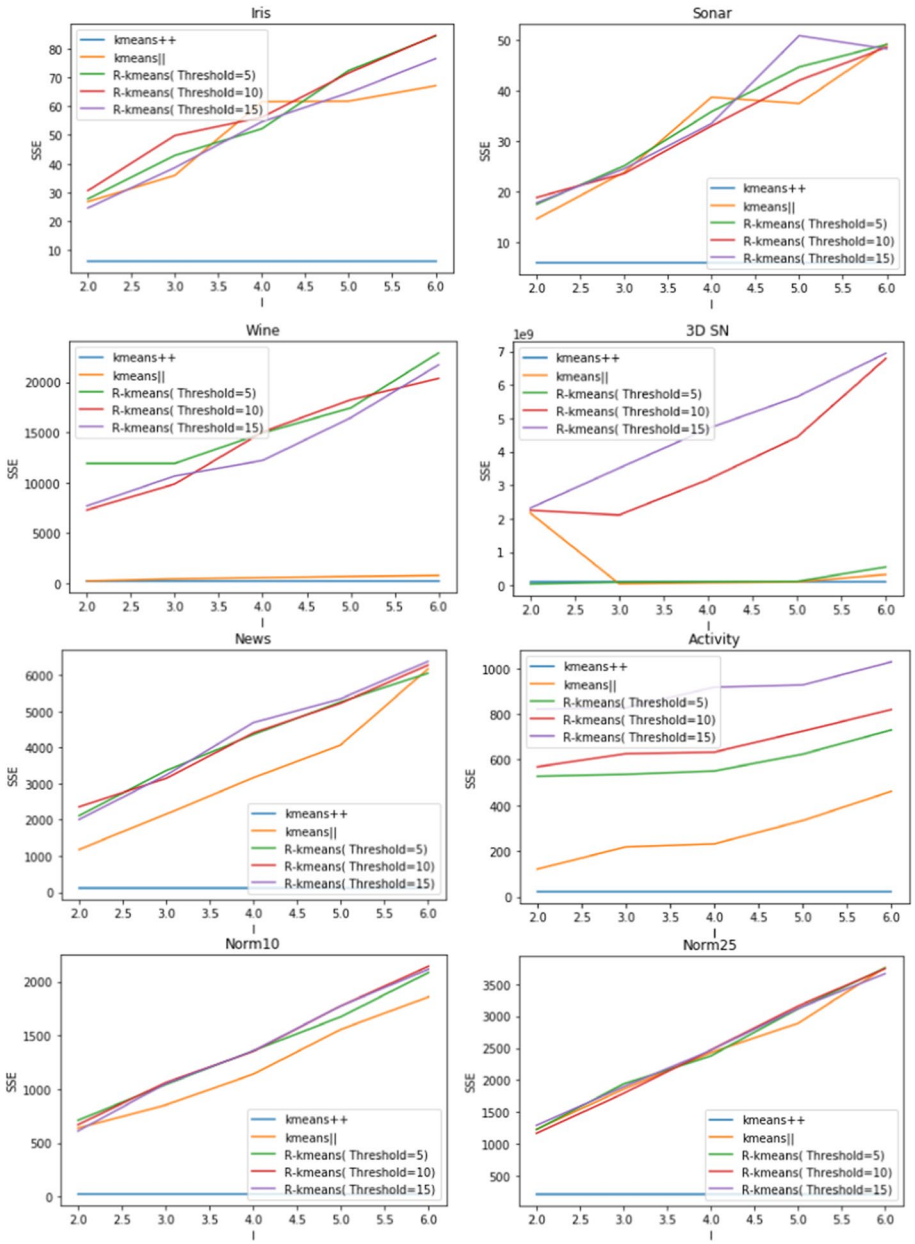

Fig. 2 Inter-cluster SSE (initialization phase)

states that there is a statistically significant difference in performance between the proposed approach and the other two methods.

$$H_0 = \mu_1 = \mu_2 = \mu_3 \neq H_A \tag{1}$$

The one-way ANOVA test compares the means of multiple groups and determines whether there is a significant difference among them. In this case, the group means

Table 3 Inter-cluster SSE

Dataset	*l*	k-means++	k-means ||	*R-k-means*		
				R = 5	R = 10	R = 15
Iris (k = 3)	2	6.1616	26.7986	27.7711	30.6586	24.6123
	3		35.9525	42.9021	49.8446	38.6484
	4		61.6088	52.2022	56.2073	54.6286
	5		61.7990	72.3874	71.5324	64.7145
	6		67.1798	84.4240	84.6325	76.5402
Sonar (k = 2)	2	5.8666	14.5736	17.4731	18.8364	17.7819
	3		23.8098	25.1271	23.5792	24.5343
	4		38.7443	35.8578	33.0370	33.5313
	5		37.4966	44.7290	42.0556	50.9349
	6		49.2615	49.1634	48.6287	48.2908
wine (k = 3)	2	250.9660	229.6661	7676.1558	7284.0679	11,915.8242
	3		452.9283	10,663.0315	9865.4811	11,915.8242
	4		560.4012	12,216.4932	15,047.1598	14,934.4016
	5		667.6036	16,447.1180	18,236.0528	17,421.5611
	6		778.0889	21,710.9044	20,372.1126	22,901.5713
3D SN (k = 5)	2	109,336,880	2,168,104,903.3345	48,660,297.0000	2,247,790,621.4291	2,314,450,425.5880
	3		53,853,088.0000	101,224,306.0000	2,107,780,481.0000	3,508,908,898.8092
	4		85,897,954.0000	114,341,298.6666	3,167,177,199.0000	4,691,745,030.8011
	5		108,397,304.0000	123,914,049.0000	4,432,592,406.0000	5,637,602,936.0409
	6		331,134,543.0000	556,311,668.0000	6,783,567,832.4096	6,939,717,354.0766
News (k = 5)	2	114.3128	1178.0515	2360.5534	2113.9152	2006.8279
	3		2156.5824	3144.6057	3360.4814	3236.0935
	4		3163.3691	4404.6432	4359.0314	4689.82083
	5		4072.5487	5349.1998	5265.0942	5228.0226
	6		6176.2299	6383.6909	6057.0527	6280.4491
Activity (k = 5)	2	23.4528	121.2921	527.6653	569.1061	822.1117
	3		218.7024	536.2599	626.7691	828.0986
	4		231.9427	550.8495	633.4962	917.1622
	5		333.8185	624.6799	724.9864	927.2576
	6		461.4125	730.1079	819.8806	1028.3068
Norm10 (k = 4)	2	24.3682	636.3063	709.4414	667.8098	608.8425
	3		851.1451	1041.1056	1060.4802	1049.3394
	4		1138.5454	1356.5306	1350.7130	1355.4381
	5		1555.7457	1673.0125	1774.5042	1773.9882
	6		1857.9609	2084.9132	2142.4280	2115.3574
Norm25 (k = 5)	2	214.9796	1229.5679	1223.6761	1165.7223	1291.2499
	3		1856.8133	1939.1858	1791.1910	1893.9185
	4		2429.2542	2378.6777	2469.8412	2465.6334
	5		2888.2870	3115.1088	3158.8707	3122.1037
	6		3760.7232	3755.8775	3743.6885	3665.1731

being compared are the results obtained by k-means++ (μ_1), k-means|| (μ_2), and the proposed approach (μ_3).

Figure 4 shows the result of the ANOVA test. To perform the one-way ANOVA test, the performance metric (Intra-Cluster SSE) is collected for each algorithm across multiple datasets presented in Table 3. The null hypothesis is then tested by analyzing the variance between the groups (algorithms) and the variance within each group. The

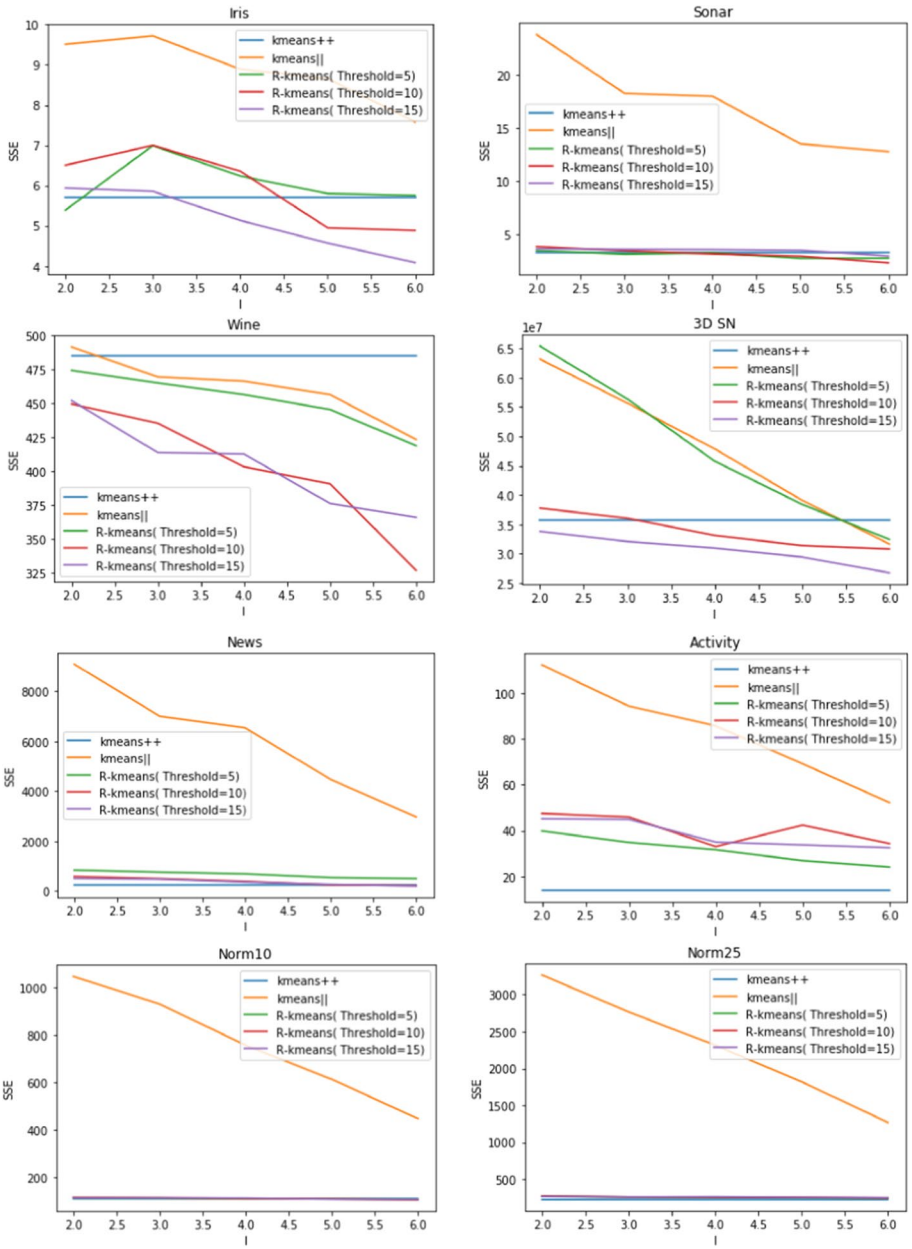

Fig. 3 Intra-cluster SSE (cluster formation phase)

result of the statistical test showed that the proposed approach in fact performed well as p-value (0.019) is smaller than the alpha (0.05). So the H_0 hypothesis is rejected.

Conclusion

Clustering large-scale data is a challenging task. This work addressed the problem of initialization of the k-means clustering algorithm for the large datasets, especially for the document data. The traditional k-means algorithm largely depends on the choice

Table 4 Intra-cluster SSE

Dataset	L	k-means++	k-means ||	*R-k-means*		
				R = 5	R = 10	R = 15
Iris (k = 3)	2	5.7119	9.7201	6.9859	6.50141	5.9335
	3		9.5087	6.2300	4.94409	5.8526
	4		8.8875	5.7965	4.88033	4.0785
	5		8.6536	5.7509	6.99476	6.1276
	6		7.5775	5.3867	6.35059	5.5642
Sonar (k = 2)	2	3.3063	23.7412	3.4541	3.8480	3.6334
	3		18.2249	3.1234	3.4419	3.5586
	4		17.9539	2.7326	3.1443	3.5812
	5		13.4780	3.2455	2.9392	3.4850
	6		12.7424	2.7532	2.3037	2.9509
wine (k = 3)	2	485.2552	491.5231	514.2313	449.5273	451.9927
	3		469.4520	494.9853	435.3800	413.7944
	4		466.3222	466.3913	403.3321	412.7368
	5		456.3817	445.4646	390.8077	376.2936
	6		423.4088	418.8286	327.0550	366.0236
3D SN (k = 5)	2	35,774,691.2030	63,158,109.2354	65,386,303.8797	37,779,415.4668	33,818,355.0888
	3		55,670,026.2025	56,386,320.7979	36,065,470.1777	32,102,586.3758
	4		47,881,019.6847	45,789,201.6702	33,133,511.8313	30,977,219.6847
	5		39,122,421.3758	38,423,140.1365	31,410,769.4442	29,458,130.3463
	6		31,666,529.8399	32,459,008.2043	30,822,221.3333	26,777,827.2025
News (k = 5)	2	224.7326	9082.4200	835.5889	581.1772	501.8262
	3		7001.9904	753.8404	496.2172	476.0039
	4		6541.1025	684.8287	385.8096	359.0314
	5		4467.0103	537.8884	244.5760	265.0942
	6		2970.6630	495.4273	220.2111	201.4371
Activity (k = 5)	2	14.0528	112.1903	39.8633	47.4728	45.2055
	3		94.2314	34.8781	45.9747	44.9542
	4		85.6380	31.6925	33.0593	35.0243
	5		69.1579	26.9614	42.4164	33.7983
	6		52.1811	24.1458	34.3696	32.5727
Norm10 (k = 4)	2	107.6781	1048.8821	110.6411	113.1431	111.2958
	3		931.2911	109.8669	111.2486	110.7953
	4		757.3920	106.3967	108.3090	110.3505
	5		612.9243	106.6067	106.5647	105.5661
	6		447.1270	104.9621	103.4869	104.7452
Norm25 (k = 5)	2	227.3719	3264.0002	268.6444	273.4111	267.8086
	3		2765.1543	260.5138	261.8034	261.0960
	4		2314.9855	255.6187	257.9374	264.5306
	5		1819.8593	249.0356	254.3362	256.0727
	6		1268.7172	244.7055	246.3955	250.6534

of cluster initial centers. The k-means++ is the most popular technique to deal with the issue of initialization in k-means, however due to its sequential nature, it is hard to apply in big data scenarios. A promising approach, known as k-means||, has recently emerged to address the issue of initialization in k-means for big datasets. This paper

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Kmeans++	40	546687600.5	13667190.01	1.34105E+15
Kmeanparallel	40	2747427217	68685680.41	1.19068E+17
KmeasnAdvance	40	23092549671	577313741.8	2.72544E+18

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	7.72569E+18	2	3.86285E+18	4.072084056	0.019514251	3.073762904
Within Groups	1.10988E+20	117	9.48617E+17			
Total	1.18714E+20	119				

Fig. 4 ANOVA analysis

proposes an algorithm, called *R-k-means* as a variant of k-means++, to offer a comparatively better solution to the problem of cluster initiation for big datasets. Using inter and intra cluster SSEs as an evaluation metrics, experimental results show that the proposed approach performs comparatively better. At each iteration of cluster initialization process, SSE of proposed approach is greater than that of k-means++ and k-means||, which shows that the centers selected in each iteration are far away from each other, thus reducing the cost of convergence in Lloyd's algorithm. The quality of final clusters was also assessed by using intra SSE, a very popular metric for cluster evaluation. The results also shows that SSE of proposed approach is much lesser than that of others, suggesting better clusters. Finally, in order to statistically validate the performance, one-way ANNOVA was performed. The result of the statistical test shows that the proposed approach in fact performs well as p-value (0.019) is smaller than the alpha (0.05).

Appendix

In this section, the overview of the basic composition of the existing clustering algorithms is discussed.

k-means

The k-means, depicted in Algorithm 2, clustering method, based on expectation maximization (EM) algorithm, divides the group of n objects into k partition clusters and measures similarity by calculating the distance between each of the items with its mean value i.e. centroid. The k-means clustering splits objects n into clusters k with each object in a cluster matching the nearest mean; for k-means clustering k (the clustering mean) must be picked before the clustering process and computed from data. The k must be chosen prior to clustering and must be computed from data. The aim is

to produce exactly k different clusters of greatest possible distinction by minimize the objective function O and maximize the L i.e. inter-cluster distance.

Definition 1 The equation below represents the objective function O where k is the number of centroids and m is the number of object assign to a particular cluster centroid.

$$O = \sum_{j=1}^{k} \sum_{i=1}^{m} d(i,j) \tag{2}$$

Here O is the distance between points within the cluster i.e. Sum of squared distances. We use Euclidean metric for distance measurement, which is defined as:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 \cdots + (x_{im} - x_{jm})^2} \tag{3}$$

Definition 2 Let we have $X = x_1, x_2, x_3, \ldots, x_n$ data points, assuming that n objects are clustered into k clusters, the inter-cluster distance is defined as the sum of the difference of mean distances of all clusters to the mean of the entire samples.

$$L = \sum_{i=1}^{k} |m_i - m| \tag{4}$$

Here L is calculated using m_i the mean of cluster c_i and the m the mean of all n data points. The k-means algorithm is defined as follow:

Algorithm 2 k-means($\{x_1, x_2, \ldots, x_n\}, k$)

1. Select k sample point randomly from X
2. **repeat:**
3. Assign all points to cluster centroids $c_i \in X$
4. Recompute centroids of each cluster
5. **until** the centroids stop convergence

Initially, the algorithm chooses the k centroids randomly from the space of data and assigns each object from data space to the centroid which has the minimum distance from the point. When all objects are assigned to the group, then the position of these k centroids is recomputed by calculating the mean of the points assigned to them on the basis of distance measure. These steps have to be repeated until convergence or until the centroids no longer move. The k-means algorithm is based on the famous Lloyd's algorithm. The k-means algorithm is scalable, computationally faster, and produces a tighter cluster for small-scale datasets but there are two major issues when it is applied to big data. It is sensitive to initialize k value and for large datasets, the number of iterations can be very large, making it computationally expensive. Respectively, each step of the

method needs computation of the distance between every pair of the data points and the inter-distance comparisons.

k-means++

Various methods have been devised to solve the problems of k-means, for instance, k-means++, depicted in Algorithm 3, proposed an improved local potential version of k-means with the D^2 weighting. The algorithm focuses on the initialization of k clusters, to improve the quality of the cluster and to minimize the number of iterations. The k-means++ chooses centers one by one in a controlled fashion. It selects the first center randomly from the dataset and then each subsequent center is selected using the probability proportional to the overall SSE given by the previously selected centroids. Preferably the algorithm achieves good clustering by preferring the centers that are far away from the previously selected points.

Algorithm 3 k-means++($\{x_1,x_2,\ldots, x_n\}, k$)

1. Select a sample point x_i randomly from X
2. $C=x_i$
3. **while** C < k **do**
4. Take new point x_i using probability $D(x)^2 / \sum_{x\in X} D(x)^2$
5. $C=CU\{x_i\}$
6. **end while**
7. **repeat**
8. Assign all points to cluster centroids C
9. Recompute centroids of each cluster
10. **until** the centroids stop convergence

The algorithm chooses a center x_i randomly from a dataset then other $k\text{-}1$ centers are chosen one by one from the dataset with probability $D(x)^2 \Big/ \sum_{x\in X} D(x)^2$.

The process is sequential, it thus repeats until k objects are selected for centers, making the complexity $O\ (n\ k\ d)$, for n points in d dimension, same as that of a single Lloyd iteration. The central downside of k-means++, from a scalability point of view, is of inherent sequential nature—the choice of the next center is conditioned to the current set of centers.

k-means||

Another improved form of k-means is k-means||, basically designed to overlay the drawback of k-means++ in terms of scalability. The k-means||, depicted in Algorithm 4, uses oversampling factor $l=\omega(k)$ for choosing k centers. Like k-means++ the algorithm selects the first center randomly from the dataset and then other centers are chosen with probability $l\, D(x)^2 \Big/ \sum_{x\in X} D(x)^2$.

The algorithm selects first point x_i randomly and then computes the initial cost (ψ) after this selection. It then iterates for log (ψ) times, in each iteration it selects l centroids from the X dataset. The number of centroids is expected to be l time $log(\psi)+1$, which is more

than the number of k. In order to reduce the selected centroids, the algorithm assigns weight to these selected centers and then recluster the weighted points into k. The k-means parallel runs in the fewer number of iterations, better in terms of running time, and clustering cost is expected to be much better than random initialization but it is not guaranteed that selected centroids are far away from each other, increasing the cost of re-clustering.

Algorithm 4 k-means‖ ($\{x_1, x_2, \ldots, x_n\}$, k, l)

1. Select a sample point x_i randomly from X
2. $C = x_i$
3. Calculate the $\psi \leftarrow \varphi X(C)$
4. **for** $log(\psi)$ times **do**
5. Z = Sample l points using distance probability $l.D(x)^2/\varphi X(C)$
6. $C = CUZ$
7. **end for**
8. Set weighting to the points in C
9. Recluster the weighted points in C into k clusters
10. **repeat**
11. Assign all points to cluster centroids C
12. Recompute centroids of each cluster
13. **until** the centroids stop convergence

Acknowledgements
Not applicable.

Author contributions
Both authors contributed equally to this work. MAR formulated and designed the solution, reviewed the manuscript, and supervised the whole work. MG contributed to the implementation of the research, processed the experimental data, drafted the manuscript and designed the figures.

Funding
The authors received no Funding for this research study.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Yes.

Competing interests
The authors declare that they have no competing interests.

Received: 26 April 2023 Accepted: 4 July 2023
Published online: 20 July 2023

References

1. MacQueen J. Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley symposium on mathematics. Statistics and probability. Berkeley: University of California Press; 1967. p. 281–97.
2. Ward JH Jr. Hierarchical grouping to optimize an objective function. J Am Stat Assoc. 1963;58(301):236–44.
3. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd, vol. 96, No. 34; 1996. p. 226–31.
4. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc: Ser B (Methodol). 1977;39(1):1–22.

5. Ng A, Jordan M, Weiss Y. On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems, 14; 2001.
6. Aloise D, Deshpande A, Hansen P, Popat P. Np-hardness of Euclidean sum-of-squares clustering. Mach Learn. 2009;75(2):245–8.
7. Jain AK. Data clustering: 50 years beyond k-means. Pattern Recogn Lett. 2010;31(8):651–66.
8. Lloyd S. Least squares quantization in PCM. IEEE Trans Inf Theory. 1982;28(2):129–37.
9. Kwedlo W, Czochanski PJ. A hybrid MPI/OpenMP parallelization of k-means algorithms accelerated using the triangle inequality. IEEE Access. 2019;7:42280–97.
10. He L, Zhang H. Kernel k-means sampling for Nyström approximation. IEEE Trans Image Process. 2018;27(5):2108–20.
11. Ahmed M. Data summarization: a survey. Knowl Inf Syst. 2019;58(2):249–73.
12. Alhawarat M, Hegazi M. Revisiting k-means and topic modeling, a comparison study to cluster Arabic documents. IEEE Access. 2018;6:42740–9.
13. Yang X, Li Y, Sun Y, Long T, Sarkar TK. Fast and robust RBF neural network based on global k-means clustering with adaptive selection radius for sound source angle estimation. IEEE Trans Antennas Propag. 2018;66(6):3097–107.
14. McCallum A, Nigam K, Ungar LH. Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining. 2000. p. 169–78.
15. Oussous A, Benjelloun FZ, Lahcen AA, Belfkih S. Big data technologies: a survey. J King Saud Univ Comput Inf Sci. 2018;30(4):431–48.
16. Sreedhar C, Kasiviswanath N, Reddy PC. Clustering large datasets using k-means modified inter and intra clustering (KM-I2C) in Hadoop. J Big Data. 2017;4(1):1–19.
17. Fränti P, Sieranoja S. How much can k-means be improved by using better initialization and repeats? Pattern Recognit. 2019;93:95–112.
18. Arthur D, Vassilvitskii S. k-means++: the advantages of careful seeding. Technical report, Stanford; 2006.
19. Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable k-means++. arXiv preprint. 2012. arXiv:1203.6402.
20. Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Philip SY, et al. Top 10 algorithms in data mining. Knowl Inf Syst. 2008;14(1):1–37.
21. Rendón E, Abundez I, Arizmendi A, Quiroz EM. Internal versus external cluster validation indexes. Int J Comput Commun. 2011;5(1):27–34.
22. Lei Y, Bezdek JC, Romano S, Vinh NX, Chan J, Bailey J. Ground truth bias in external cluster validity indices. Pattern Recogn. 2017;65:58–70.
23. Wu J, Chen J, Xiong H, Xie M. External validation measures for k-means clustering: a data distribution perspective. Expert Syst Appl. 2009;36(3):6050–61.
24. Jahan M, Hasan M. A robust fuzzy approach for gene expression data clustering. Soft Comput. 2021;25(23):14583–96.
25. Sinaga KP, Yang MS. Unsupervised k-means clustering algorithm. IEEE Access. 2020;8:80716–27.
26. Pelleg D, Moore AW, et al. X-means: extending k-means with efficient estimation of the number of clusters. In: Icml. 2000. p. 727–34.
27. Hamerly G, Elkan C. Learning the k in k-means. In: Advances in neural information processing systems; 2003. p. 16.
28. Faber V. Clustering and the continuous k-means algorithm. Los Alamos Sci. 1994;22(138144.21):67.
29. Bradley PS, Fayyad UM. Refining initial points for k-means clustering. In: ICML. 1998. p. 91–9.
30. Khan SS, Ahmad A. Cluster center initialization algorithm for k-means clustering. Pattern Recogn Lett. 2004;25(11):1293–302.
31. Ostrovsky R, Rabani Y, Schulman LJ, Swamy C. The effectiveness of Lloyd-type methods for the k-means problem. J ACM. 2013;59(6):1–22.
32. Ailon N, Jaiswal R, Monteleoni C. Streaming k-means approximation. In: NIPS. 2009. p. 10–8.
33. Li Y, Zhang Y, Tang Q, Huang W, Jiang Y, Xia ST. tk-means: a robust and stable k-means variant. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2021. p. 3120–4.
34. Giffon L, Emiya V, Kadri H, Ralaivola L. QuicK-means: accelerating inference for K-means by learning fast transforms. Mach Learn. 2021;110:881–905.
35. Xia S, Peng D, Meng D, Zhang C, Wang G, Giem E, Wei W, Chen Z. Ball *k* k-means: fast adaptive clustering with no bounds. IEEE Trans Pattern Anal Mach Intell. 2020;44(1):87–99.
36. Ismkhan H. Ik-means−+: an iterative clustering algorithm based on an enhanced version of the k-means. Pattern Recogn. 2018;79:402–13.
37. Manochandar S, Punniyamoorthy M, Jeyachitra RK. Development of new seed with modified validity measures for k-means clustering. Comput Ind Eng. 2020;141: 106290.
38. Zhao W, Ma H, He Q. Parallel k-means clustering based on MapReduce. In: IEEE international conference on cloud computing. 2009. p. 674–9.
39. Xu Y, Qu W, Li Z, Min G, Li K, Liu Z. Efficient k-means++ approximation with MapReduce. IEEE Trans Parallel Distrib Syst. 2014;25(12):3135–44.
40. Alguliyev RM, Aliguliyev RM, Sukhostat LV. Parallel batch k-means for Big data clustering. Comput Ind Eng. 2021;152: 107023.
41. Hämäläinen J, Kärkkäinen T, Rossi T. Scalable initialization methods for large-scale clustering. arXiv preprint. 2020. arXiv:2007.11937.
42. Chowdhury K, Chaudhuri D, Pal AK. An entropy-based initialization method of k-means clustering on the optimal number of clusters. Neural Comput Appl. 2021;33(12):6965–82.
43. Torrente A, Romo J. Initializing k-means clustering by bootstrap and data depth. J Classif. 2020;38:1–25.
44. Duy-Tai D, Van-Nam H. *k*-PbC: an improved cluster center initialization for categorical data clustering. Appl Intell. 2020;50(8):2610–32.
45. Bortoloti FD, de Oliveira E, Ciarelli PM. Supervised kernel density estimation K-means. Expert Syst Appl. 2021;168: 114350.
46. Fahim A. K and starting means for k-means algorithm. J Comput Sci. 2021;55: 101445.

47. Abdulnassar AA, Nair LR. Performance analysis of Kmeans with modified initial centroid selection algorithms and developed Kmeans9+ model. Meas Sens. 2023;25: 100666.
48. Ay M, Özbakır L, Kulluk S, Gülmez B, Öztürk G, Özer S. FC-Kmeans: fixed-centered K-means algorithm. Expert Syst Appl. 2023;211: 118656.
49. Li H, Wang J. Collaborative annealing power k-means++ clustering. Knowl-Based Syst. 2022;255: 109593.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com