IIOAB JOURNAL
ISSN: 0976-3104

ARTICLE

EXPERIMENTAL ANALYSIS OF SEQUENTIAL CLUSTERING ALGORITHM ON BIG DATA

Ajay Kumar and Shishir Kumar*

Dept. of Computer Science & Engineering, Jaypee University of Engineering & Technology, Guna (MP), INDIA

ABSTRACT

In the past decade, commercial and scientific organization generates a large volume of data. Clustering of Big data finds application in many areas. For example, in pattern recognition, information extraction, image processing, computational statistics, and data mining. Over the years, we find many methods in the literature for clustering large data set and most of these methods uses, a distance based similarity measures to assign data points to a cluster. This paper considers the case of finding the cluster in big data that cannot fit into the physical processor memory or the data is accessible in batches. In the present work, we have modified the distance-based similarity measure to discover the similarity between an unlabeled data point and a cluster for assignments of cluster label to unlabeled data points. The data points first sampled according to available physical memory and then passed to the next step for further processing. We have used the k-means algorithm and its variants to assign the class or label to the first sample into a legitimate cluster. Also, we have compared the results with sequential k-means, batched-mode k-means, and online k-means algorithm for the clustering of the big data set. Our experimental results based on running time and the quality of the clusters shows the viability of proposed work.

KEY WORDS
Clustering, Sequential k-means, online k-means, Incremental clustering, k-means, k-means++

Published: 2 December 2016

INTRODUCTION

In recent years, we have seen that commercial and scientific organization generates a large volume of data. Clustering of such data finds application in many areas. However, clustering of large data set requires a different approach than traditional clustering algorithm because this data set may not fit into the processor's physical memory. Another situation that may challenge the clustering algorithm is the use of different data access model when the data is available in a stream, increments, or in batches [1].

The primary need of the clustering algorithm to handle above situation is to reuse the clustering results on the addition of new data to the existing dataset in the form of batches or according to the data access model. Liberty et al. show three type of data access model for the handling of a large dataset, the offline mode, stream mode and online mode [1].

In the offline access mode, the data points are accessible ahead of time, and the data access model is unrestricted. Therefore, finding the optimal solution is hard. Recently, with an end goal to make adaptive sampling methods more scalable, Bahmani et al. presented k-means|| algorithm that decreases the number of passes required for the data, and empowers enhanced parallelization [2].

In the streaming model, the data is processed in one pass and keep a small amount of information about the data. In any case, it must yield cluster centers when the stream ends. This limited data access model requires new algorithmic thoughts for clustering. In this model, the new data points get their cluster with the help of previously computed centers [3, 4].

An online k-means algorithm must allocate points to cluster throughout the run of the algorithm. In this model, data points arrived one by one in an arbitrary order. At the point when new point arrives, the algorithm should either include it in one of the current clusters or create another cluster. This setting is entirely harder than the streaming model. From another viewpoint, the online algorithm can be convertible to a streaming model. Also, one has to keep enough statistics for every cluster so that it can be used toward the end of the stream [1].

Due to simplicity and ease of implementation, the k-means algorithm and its variants are used to cluster large dataset in the numerical domain [5]. In this method, every cluster is drawn by a center that is computed by the mean value of the entire data points present in the cluster [6]. The algorithm uses a straightforward distance measure to give a suitable cluster label to each non-clustered data points.

We see that the Euclidean distance measure from the mean value of other data points in the cluster is not adequate for a non-uniformly distributed cluster. Therefore, there is a need to address similarity measures based on the distribution of data objects in a cluster. In this paper, we have analyzed the use of a new similarity measure in light of the spread (largest and least value of every cluster) of data in every cluster and its mean value (centroids). The algorithm, named as Incremental Batched Clustering Algorithm (IBCA), gives a suitable cluster level to each non-clustered data point that is coming as batches or stream mode.

In this work, we have divided the task of assignments of cluster label to a non-clustered data point in three stages: batch creation, cluster analyses, and data labeling phase. In batch formation stage, we have sampled an appropriate size of data according to the main memory of the system. In cluster analysis stage, a suitable clustering algorithm, for example, k-means, k-means++, k-medoids and fuzzy c-mean, is applied on the dataset to get the initial cluster representative. In data labeling stage, based on the new

***Shishir Kumar**
Email: dr.shishir@yahoo.com
Tel.: +91-9407327719

Guest editors: N. Arun Kumar & P. Mohammed Shakeel

similarity coefficient, each unlabeled data point gets an appropriate cluster label corresponding to the maximal resemblance. The experiments on the benchmark datasets show the effectiveness of the proposed framework. We have also analyzed the effect of the initial seed selection of the k-means algorithm on a large dataset.

The rest of paper is organized as follows. In Section 2, the related works are reviewed. In Section 3, the problem statement and several variants of k-means are presented. Section 4 describes our proposed method and Section 5 illustrates experimental setup and results. In Section 6, we conclude the paper.

REALATED WORKS

Several methodologies have been documented in the literature for the study of the structure of large data sets [7]. Many applications need clustering of datasets that is too large to fit into the usable physical memory, or the clustering data is available in batches. Both the circumstances influence the need for incremental clustering algorithm [8].

The investigation of incremental clustering has begun with Hartigan's clustering algorithm that makes utilization of a threshold value to decide if a data objects can be put in a current cluster or it should form another cluster independently [9]. Several other algorithms use a sample data having a size equivalent to accessible physical memory [10].

The BIRCH clustering algorithm is well known for the clustering of streaming data. This algorithm has two stages: in the initial step, it examines the database and after that builds a tree comprising data of clusters. In the next step, BIRCH prunes the tree by eliminating of sparse nodes and producing new unique cluster. However, the technique has a drawback of limited memory of its leaves. In addition, the algorithm will not execute well if the cluster does not have spherical shapes because the BIRCH algorithm controls the cluster's boundary by applying the approximation of diameter [11].

The maximum margin clustering uses one data point at one time to decide which cluster the new data point has a place [12]. In another approach, a distance based incremental clustering technique that can discover clusters of arbitrary shapes and sizes in dynamic datasets have been introduced in [13]. Incremental clustering is also referred as a single pass clustering while the conventional algorithm is similar to multi-pass clustering. The thought is to cluster a manageable amount of data and preserve the results for the next block until all the data blocks are processed [8].

Shindler et al. consider the k-means issue where the data is too large to be stored in physical main memory and data must be acquired sequentially [14]. They introduce an improved algorithm for the k-means in a streaming model. Ester et al. have modified DBSCAN algorithm that is appropriate for clustering where dataset are often updated [15].

Aaron et al. demonstrated a dynamic incremental k-means clustering algorithm to discuss seeding issue, the sensitivity of the algorithm to the order of the data, and the number of clusters for the k-means algorithm [16]. Om and Sangam introduced a hybrid data-labeling algorithm for clustering mixed datasets [17]. This algorithm utilizes distance comparability measures for the numerical data to give cluster level to unlabeled data points.

Based on literature survey we observe that most of the method uses a distance based similarity measure for clustering large datasets. In the next section, we present the k-means and its variants that are used for handling large dataset.

K-means and related

The traditional k-means algorithm is simple optimization model in unsupervised learning process [6]. Given the set, $D = \{x_1, x_2 \ldots, x_n\}$, of n data points in R^n, the objective is to minimize the cost of discovering k clusters, $C = \{c1, c2, \ldots, ck\}$, with k center, $\{m1, m2, \ldots, mk\}$ by using equation-1.

$$V(C) = \sum_{i=1}^{n} \min_{j=1}^{k} \| x_i - m_j \|^2 \quad (1)$$

Where m_j, is the mean of the cluster, ck. The norm $\| * \|$ can be any vector norm on R^n. Minimizing the squared error is an NP-hard problem [18]. The k-mean technique is an iterative process and often terminates near a local minimum, but it is very sensitive to the initial set of centers. Algorithm-1 shows the working of the k-means algorithm.

Algorithm 1 : k-means (R, k)
Step 1. Select k first centers $\{m1, m2, \ldots, mk\}$ randomly from R

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

Step 2.	Repeat
Step 3.	Partition R into k subsets, $\{c1, c2, \ldots, ck\}$, such that R_i, $1 \le i \le k$, contains all points whose nearest center is mi
Step 4.	Replace the current set of centers by a new set of centers $\{m1, m2, \ldots, mk\}$, such that center mi, $1 \le i \le k$, is the center of gravity of R_i
Step 5.	Until the set of centers has not changed

Clustering method expects that the whole data set is available ahead of time. In a few cases, it is possible that the data is large and cannot match the memory of the processor, or the data is available in batches, so a substitute technique must be adapted. The k-means and various other iterative optimization algorithms related to the k-means algorithm uses one of two fundamental modes of data access model for dealing large data, batch mode, and on-line mode. In an online mode or sequential mode, the parameters of the cluster are recalculated at every addition of new data points of the training set. Similarly, in batch mode, the data is accumulated with time, and a while later the clustering is performed on each batch of data. The last centers of cluster i are utilized as the initial center for the next approaching batch i+1 [19].
In the sequential k-means, the data points are assigned to a cluster and immediately update the cluster centers. Because of the substantial overhead of recalculating centers, the resource requirement of sequential k-means may be a restrictive element. Even so, in specific situations, the data may appear one at a time. In these cases, it may be desirable to update the results per data occurrence as opposed to waiting for the aggregation of the whole data set.

In the next sub-section, we discuss various variants of k-means algorithm documented in the literature that is dealing with large data. Further to this, we also analyze the effect of initial seed selection methods on the quality of clustering results on the sequential k-means algorithm.

The Sequential Block K-means Algorithm

The sequential block mode is a trade-off between the stringent computational prerequisites of the sequential k-means and the need to work on data online. For this situation, the clustering happens on accumulated block of data. Every block is going through l epoch of k-means where the last centers of block l are utilized as the initial center for block l+1. In many cases l=1. In this sense, the algorithm uses another multistage clustering, for example, the pyramid k-means algorithm [19, 16]. The block sequential algorithm employed in an iterative manner where on each iteration performs l epochs of k-means on a single block of data elements at once.

Single Pass k-means

Single Pass k-means (SQKM) is an adaptation of the online fuzzy c-means algorithm discussed in [10]. Give N be the number of data objects in a dataset, which is too large to fit in main memory; let n signify the number of points that can be stacked into memory. SQKM divides the N points in the big dataset into s pieces of small data as $s = (\frac{N}{n})$. Further, k-means algorithm is used to find k groups from these segments.

The first set of data now replaced by the k-weighted means, $\{m_{w,i} : 1 \le i \le k\}$, where the weights are the numbers of data objects in every group, $\{w_i = |X_i| : 1 \le i \le k\}$.These k weighted centroids are then converged with the next data segment. Followed by applying weighted k-means (WKM) to this combined set with the centroids from the past k-means run. The procedure is repeated until the entire dataset is stacked and processed. In the wake of acquiring the last k centroids, the big data are labeled according to the similarity measures. The two most vital disclaimers about its effectiveness are:

- At every step the SQKM faces same limitation and problem as the k-means algorithm and
- The output is obviously subject to the way big data are divided in s sample. Each chunk or sample of SQKM is treated as one input, and thus SQKM is sequential.

K-MEANS++ Clustering Algorithm

The k-means algorithm is sensitive to the initial set of center. To improve the quality of k-means algorithm k-means++ algorithm [20] was introduced. It chooses the initial set of center with probability proportional to its squared distance from the nearest center already chosen. The k-means++ algorithm for choosing k initial center is given in Algorithm-2

Algorithm 2: k-means++ (R, k)	
Step 1.	choose an initial center m1 uniformly at random from R
Step 2.	$M \leftarrow m_1$
Step 3.	for i = 2 to k
Step 4.	choose the next center mi at random from R, where the probability of each $r \in R$ is given by D2(r, M)/cost(R, M)

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

Step 5. $M \leftarrow M \cup m_i$
Step 6. repeat
Step 7. Partition R into k subsets $\{c1, c2, \ldots, ck\}$, such that R_i, $1 \le i \le k$, contains all points whose nearest center is mi
Step 8. replace the current set of centers by a new set of centers $\{m1, m2, \ldots, mk\}$, such that center mi, $1 \le i \le k$, is the center of gravity of R_i
Step 9. until the set of centers has not changed

Partitioning about medoids (PAM)

Partitioning about medoids (PAM) cluster the data objects about k medoids. The algorithm takes the instance of the steepest ascent hill climber, using a direct swap neighborhood operation. Best objects are picked in each iteration that conveys the best clustering result. The objective function uses the sum of the distance from each object to the closest medoid. As the search time of the algorithm is slow, the initial set of medoids builds in a greedy fashion. Starting with an empty set of medoids, data objects are added one at a time until k medoids have been selected. At each step, the new medoid is chosen to minimize the objective function.

PROPOSED WORK

Most of the partitional clustering techniques method makes use of the Euclidean distance between the cluster centers and the data points to allocate a data point into the cluster. In this paper, we propose new similarity measures for the sequential k-means clustering. The algorithm is abbreviated as Incremental Batched Clustering Algorithm (IBCA) and is capable of handling large and streaming numerical data.

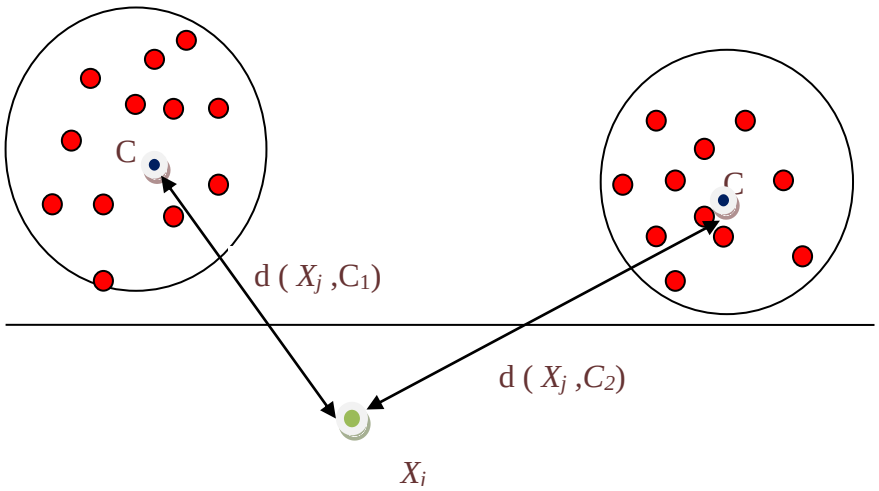
The sequential k-means algorithm divides the N points in the big dataset into "s" pieces of small data. On the arrival of a new data point, the algorithm uses the previously computed centroids for the assignment of cluster label. The process is repeated until the entire dataset is stacked and processed. Often the similarity measures that are used in handling sequential data is based on the Euclidean distance between the cluster representative and the data objects. However, this strategy does not consider the spread of data inside the group for appointing the cluster level to unlabeled data points. Upon arrival of a data point, Xi the sequential k-means algorithm uses the similarity measures defined in equation-2 to provide cluster label to data point Xi [21].

$$D_{(X_i,C_j)} = \sum_{i=1}^{n} \min_{j=1}^{k} \| X_i - C_j \|^2 \quad (2)$$

Where $D_{(X_i,C_j)}$ is the distance of data object X_i with cluster center C_i.

Now consider the case where a data point arrives in unexpected order. In the sequential approach, the first block of data is divided into two clusters having centroids C1 and C2 with the help of traditional k-means algorithm. Now on the arrival of a new data point Xi., following two possibilities is possible which are shown in the [Fig. 1] and [Fig. 2].

- If the distances between Xi and cluster centers are equal, then the algorithm is not able to decide the correct cluster label of data point Xi.
- If Xi is close to the boundary points of cluster C_1 or C_2. In this case, we have to check $D_{\max}(X_1,C_1) < D_{\max}(X_1,C_2)$.



THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

Fig. 1: First Case when d (Xi, C1) = d (Xi, C2), i.e., distance from the centroids are equal.

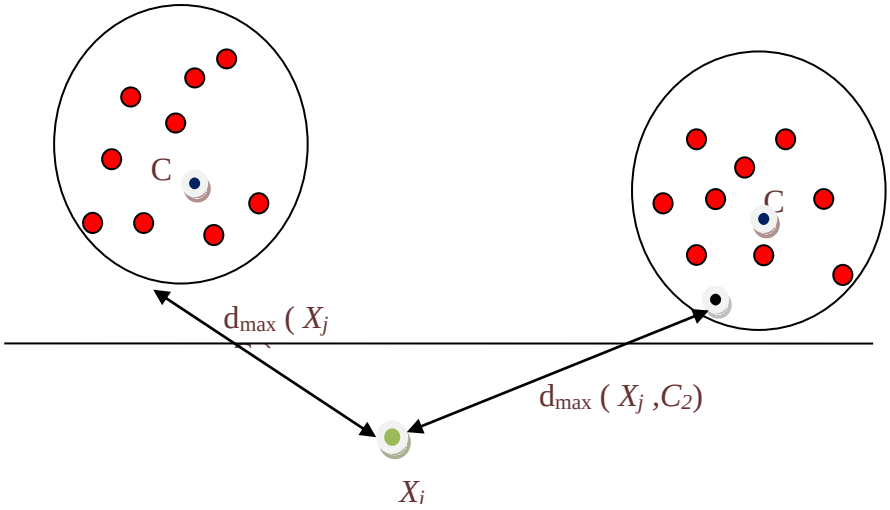


Fig. 2: The Second case when the new data point is close to the boundary points of a cluster.

Therefore, to discover a similarity coefficient for the unlabeled data on cluster C_i Euclidian distance method is used. Equation-3 gives the Euclidian distance between an unlabeled data point x_j and mean estimation of cluster ci.

$$D_{mean}\left(x_j,c_i\right)=\sum_{j=1}^{n}\sum_{i=1}^{k} min \parallel x_j - mean(c_i) \parallel^2 \quad (3)$$

Where, $D_{mean}\left(x_j,c_i\right)$ is the Euclidean distance between the unlabelled data point x_j and the centroid of the cluster c_i . The $mean(c_i)$ represents the mean value of all the data points in the cluster c_i or centroids.

Similarly, the equation-4 and 5 gives the Euclidian distance between an unlabeled data point x_j and the nearest boundary points of a cluster C_i .

$$D_{max}\left(x_j,c_i\right)=\sum_{j=1}^{n}\sum_{i=1}^{k} min \parallel x_j - max(c_i) \parallel^2 \quad (4)$$

Where, $D_{max}\left(x_j,c_i\right)$ represents the Euclidean distance between the unlabelled data point, x_j and the maximum value of cluster c_i .

$$D_{min}\left(x_j,c_i\right)=\sum_{j=1}^{n}\sum_{i=1}^{k} min \parallel x_j - min(c_i) \parallel^2 \quad (5)$$

Where, $D_{max}\left(x_j,c_i\right)$ represents the Euclidean distance between the unlabelled data point x_j and the minimum value of cluster c_i .

So based on the above cases equation-6 shows a new similarity measure (SC) between the new incoming data points x_j and the cluster c_i .

$$SC\left(x_j,c_i\right)=\min\left(D_{mean}\left(x_j,c_i\right)+D_{max}\left(x_j,c_i\right)+D_{min}\left(x_j,c_i\right)\right) \quad (6)$$

We have considered the mean and spread of the data inside the cluster as a similarity measures to give cluster labeled to unlabeled data points. The proposed approached is explained with the help of exam.1.

Example-1: Consider the artificial dataset given in [Table 1]. The dataset comprises of two numerical attribute A1, A2, and having nineteen data samples. These data samples are divided into three cluster c1, c2, and c3. Each cluster has five data points. Based on similarity coefficient described in equation-6, the cluster label to unlabelled data points x16, x17, x18, and x19 in clusters c1, c2, and c3 are obtained as follows:

$$SC\left(x_{16},c_1\right)=\min\left(D_{mean}\left(x_{16},c_1\right)+D_{max}\left(x_{16},c_1\right)+D_{min}\left(x_{16},c_1\right)\right)$$

$$=18.04217+24.73863+15.65248= 58.43328241$$

According to the above-calculated similarity coefficient values, we can assign c1 cluster label to, x_{16} since the distance between the data point and the value of SC is minimum as compared to other clusters. For other unlabeled data points, a similar approach has been used and it is summarized in [Table 2]. Final cluster assignment to unlabelled data points is given in [Table 3].

The example shows that our proposed similarity coefficient preserves the characteristics of clustering, i.e., both high intra-cluster similarity and small inter-cluster similarity.

Table 1: Artificial dataset

Data point	A1	A2	Cluster
x_1	11	52	c1
x_2	13	45	c1
x_3	14	44	c1
x_4	15	57	c1
x_5	19	40	c1
x_6	31	60	c2
x_7	35	62	c2
x_8	34	71	c2
x_9	39	73	c2
x_{10}	30	78	c2
x_{11}	61	51	c3
x_{12}	64	56	c3
x_{13}	67	58	c3
x_{14}	72	59	c3
x_{15}	76	55	c3
x_{16}	25	33	?
x_{17}	33	39	?
x_{18}	36	73	?
x_{19}	16	51	?

Table 2: Distance matrix of unlabelled data points w.r.t mean

		c_1	c_2	c_3	Cluster assignment
D_{mean}	x_{16}	18.04217	35.80894	44.99822	c_1
	x_{17}	20.49195	30.25029	36.01722	c_1
	x_{18}	33.34247	39.42436	11.65504	c_3
	x_{19}	3.757659	24.75237	30.1171	c_1
D_{max}	x_{16}	24.73863	47.12749	57.24509	c_1
	x_{17}	22.80351	39.45884	47.42362	c_1
	x_{18}	23.34524	5.830952	42.37924	c_2
	x_{19}	6.708204	35.4683	60.53098	c_1
D_{min}	x_{16}	15.65248	27.65863	40.24922	c_1
	x_{17}	22.02272	25.23886	30.46309	c_1
	x_{18}	41.40048	21.40093	33.30165	c_2
	x_{19}	12.08305	9.486833	45	c_2

Table 3: Sum of distance matrix and final cluster assignment for unlabelled data points

	c_1	c_2	c_3	Cluster labeled
x_{16}	58.43328	110.5951	142.4925	c_1
x_{17}	65.31817	94.94799	113.9039	c_1
x_{18}	98.08818	66.65624	87.33593	c_2
x_{19}	22.54891	69.7075	135.6481	c_1

Given X data points, the proposed algorithm incrementally arrives at a cluster solution upon receiving a new data point, Xj as follows:

Step 1. Compute the sample data from the sequential data X, s= (n/p), p is the available physical memory, n is the number data point.

Step 2. Normalize the X by min-max normalization algorithm.

Step 3. Apply k-means (or any other partitioned clustering algorithm) clustering algorithm on sample dataset X to create Ci partition. Assigned obtained centroid as $Centroid^{old}$ and cluster label as $Cluter\_label^{old}$

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

Step 4. Find the minimum, maximum, and means of each cluster Ci
Step 5. Upon receiving a new data point Xi, assign a cluster label as i by using the similarity coefficient defined in equation-6.
Step 6. Obtained $Centroid^{new}$
Step 7. Update the $centroid^{old}$ = $Centroid^{new}$ and append the cluster label to $Cluter\_label^{old}$.
Step 8. X=X+1; go to step 5.

The pseudo code of the proposed algorithm is given in [Fig. 3].

Algorithm: IBCA
Input: Dataset X, number of cluster k
Output: Cluster labeled to unlabeled of data points

```
(n×m) ←size (X)
Compute the sample data size, s= (n/p), p is the available physical memory
Initialize the variable seed as k*m array
Initialize min_data_cluster, max_data_cluster, distance as an (k*m) empty array
Load X as an s sized sample dataset
Apply any clustering algorithm on sample dataset to create partitions
Assign obtained centroid to seed array
for i=k+1 to n do
   for j=1 to k do
      Compute the minimum, maximum and mean value of the ith numerical attribute on jth cluster and
assign these value to min_data_cluster[i][j], max_data_cluster[i][j], and mean_data_cluster[i][j]
respectively
    end
for p=1 to L do
        initialize max_similarity=0
    for i=1 to k
 find the similarity between a Pth unlabelled data points and a ith cluster according to similarity
coefficient (6)
     if max_similarity < similarity then
        max_similarity=similarity
        label=p
     end
    end
 assign label value to unlabelled data points as cluster label
end
```

Fig. 3: The pseudo-code of the proposed algorithm.

EXPERIMENTAL SETUP AND RESULTS

We ran the proposed algorithm on several datasets. The datasets are taken from the UCI machine learning repository, and the summaries of their characteristics are given in [Table 4] [22]. The dataset s1, s2, s3, and s4 are artificial datasets consisting of Gaussian clusters with the same variance but increasing overlap. Some of these datasets is available on the web page https://cs.uef.fi/sipu/datasets as mentioned in the literature [23]. All the experiments are performed on Intel Core™ i3-2348M machine with 2GB of RAM.

Table 4: Data Set description

Dataset	Data Set Characteristics	Number of features	Number of data objects	Number of Clusters
Iris	Multivariate	4	150	3
Wine	Multivariate	13	178	3
Thyroid	Multivariate	5	215	3
WDBC	Multivariate	32	569	2
s1-s4	Artificial	2	5000	15

We have compared the performance of the proposed method with other variants of sequential k-means algorithm. We have used running time, Dunn's, and Davies-Bouldin (DB) index as parameters for the comparison of experimental results. Several runs of each algorithm have been performed, and most significant digits of the result are shown.

The Dunn's index measure is used to find the compactness of the data points within the cluster and cluster's separation (minimum distance between clusters). The maximum value of the index represents

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

the right partitioning. The Davies-Bouldin index (DB) criterion is based on a ratio of within-cluster and between-cluster distances. A Smaller value of DB shows a better clustering solution [24].

First, we made running time analysis of the proposed method (IBCA) with Block-Mode-k-means, Online-k-means, and Sequential-k-means. By analyzing, the results, we have observed that the proposed approached have better time complexity with Sequential k-means and its other variants for the data set used in the experiments, see [Fig. 4]. We have also analyzed the seeding issue of the k-means algorithm and find that the running time is higher than the proposed work, but the quality can be improved significantly by choosing proper initial centers.

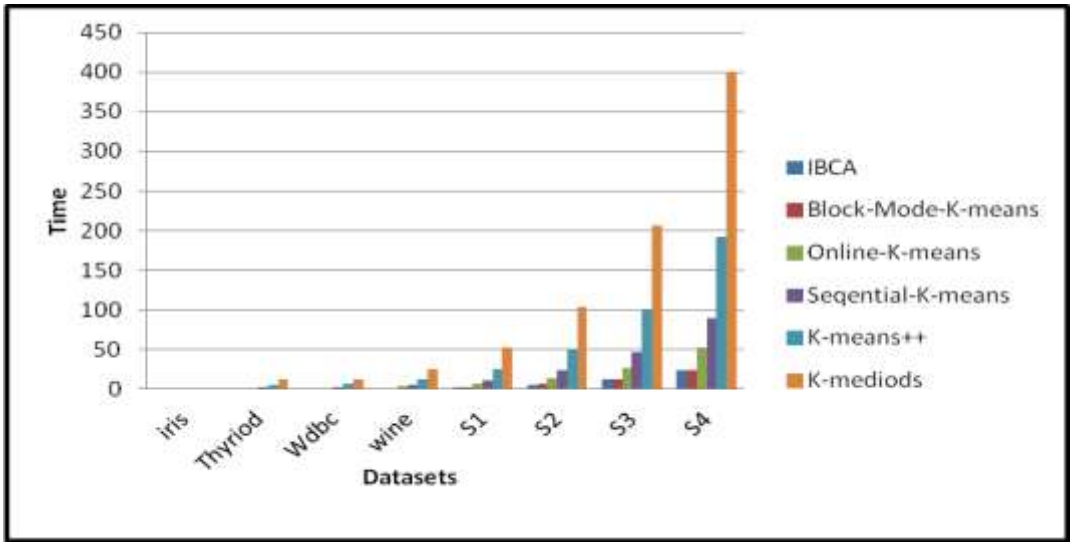


Fig. 4: Running time analysis of different variants of Sequential k-means algorithm.

As mentioned in the proposed approached the first block of data is clustered with the k-means algorithm can have poor clustering results due to a random selection of initial centers. Therefore an initialization algorithm k-means++ and k-medoids has been used in the initial phase. However, with the good initial guess of centroids consume some amount of time compared to other algorithms, but the quality of the cluster is good.

The next set of experiment examines the quality of clustering results on various dataset mentioned in the [Table 4]. Results on these data sets are tabulated in the [Table 5-12]. The experimental results show that the quality of the clustering results of the proposed algorithm is equivalent to the traditional sequential k-means algorithm except for the dataset S1 see Table-9. However, other algorithms perform better than the proposed algorithm, but the running cost of these algorithms is high; see [Fig. 5] and [Table 5-12].

Table 5: Comparative results on the iris dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	**0.776**	**0.9449**	**1.3442**	**1.2427**	**1.2124**
	Block-Mode-K-means	2.0197	1.3261	1.4959	1.3375	2.0279
	Online-K-means	2.0197	1.5961	2.1545	1.9433	1.994
	Sequential-K-means	**0.776**	**0.9449**	**1.3442**	**1.2427**	**1.2124**
	K-means++	2.219	1.4856	1.5067	1.8818	1.6197
	K-medoids	2.219	1.6488	1.4959	1.252	1.9557
DB	IBCA	0.9714	1.283	0.699	0.6673	0.6466
	Block-Mode-K-means	0.7004	0.6694	0.656	0.6593	0.7629
	Online-K-means	0.7004	0.649	0.6894	0.7209	0.7043
	Sequential-K-means	0.9714	1.283	0.699	0.6673	0.6466
	K-means++	0.5114	0.5956	0.641	0.7269	0.7812
	K-medoids	0.5114	0.6401	0.656	0.6494	0.7622

Table 6: Comparative results on the iris dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	0.6907	0.6821	0.783	0.715	0.673
	Block-Mode-K-means	1.1739	0.6364	0.8521	0.9842	1.3211
	Online-K-means	1.1699	0.6415	0.8926	1.0284	1.2896
	Sequential-K-means	0.6907	0.6821	0.783	0.715	0.673
	K-means++	1.3304	0.65	1.4457	0.9607	1.2934

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

	K-medoids	1.0369	0.6148	0.9629	1.0635	1.2934
DB	IBCA	2.3034	2.1126	1.8226	1.7863	1.865
	Block-Mode-K-means	1.5655	1.7807	1.8474	1.857	1.3683
	Online-K-means	1.4912	1.7821	1.9328	1.6605	1.188
	Sequential-K-means	2.3034	2.1126	1.8226	1.7863	1.865
	K-means++	0.9941	1.7547	1.1976	1.8361	1.3746
	K-medoids	1.747	1.8287	1.6531	1.4858	1.3746

Table 7: Comparative results on the thyroid dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	**1.382**	**1.2703**	**1.2955**	**0.7612**	**0.8163**
	Block-Mode-K-means	1.3711	1.295	1.4407	0.6743	1.4785
	Online-K-means	1.1973	1.2043	1.1764	0.697	1.4785
	Sequential-K-means	1.382	1.2703	1.2955	0.7612	0.8163
	K-means++	1.3821	1.247	1.3719	2.0212	1.4785
	K-medoids	1.2612	1.2138	1.1245	0.6743	1.4812
DB	IBCA	0.8864	1.304	1.3246	2.3526	1.692
	Block-Mode-K-means	1.0319	1.3567	1.3497	0.8977	0.7712
	Online-K-means	1.2869	1.0813	1.1516	0.8867	0.7712
	Sequential-K-means	0.8864	1.304	1.3246	2.3526	1.692
	K-means++	1.0377	1.3829	1.3264	0.7543	0.7712
	K-medoids	1.3343	1.3413	1.4192	0.8889	0.7591

Table 8: Comparative results on the Wdbc dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	1.1907	1.0946	1.1667	1.2507	1.2609
	Block-Mode-K-means	1.2707	1.1479	1.2403	1.26	1.281
	Online-K-means	1.2707	1.1644	1.2403	1.2594	1.2526
	Sequential-K-means	1.1907	1.0946	1.1667	1.2507	1.2609
	K-means++	1.2707	1.1479	1.2403	3.0377	1.2727
	K-medoids	1.2707	1.1644	1.2403	1.2599	1.2526
DB	IBCA	1.432	1.4879	1.4218	1.3247	1.3138
	Block-Mode-K-means	1.3388	1.4304	1.337	1.308	1.3093
	Online-K-means	1.3388	1.4167	1.337	1.3083	1.3205
	Sequential-K-means	1.432	1.4879	1.4218	1.3247	1.3138
	K-means++	1.3388	1.4304	1.337	0.6383	1.3123
	K-medoids	1.3388	1.4167	1.337	1.3081	1.3205

Table 9: Comparative results on the S1 dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	0.0434	0.0429	0.0958	0.0992	0.1262
	Block-Mode-K-means	0.6396	0.572	0.5384	0.5308	0.5361
	Online-K-means	0.856	1.3119	1.1437	1.4073	4.1649
	Sequential-K-means	1.1144	0.8647	0.672	0.7699	0.9218
	K-means++	1.3588	1.2671	1.0654	1.2126	0.5689
	K-medoids	0.649	0.6163	0.5797	0.4326	0.4884
DB	IBCA	1.1144	0.8647	0.672	0.7699	0.9218
	Block-Mode-K-means	0.6292	0.7492	0.5244	0.4113	0.3685
	Online-K-means	0.6688	0.6497	0.5373	0.4185	0.3103
	Sequential-K-means	0.0434	0.0429	0.0958	0.0992	0.1262
	K-means++	0.572	0.6521	0.5375	0.3991	0.3812
	K-medoids	0.7194	1.2003	1.0195	1.4404	0.5075

Table 10: Comparative results on the S2 dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	**0.059**	**0.0836**	**0.1142**	**0.1766**	**0.216**
	Block-Mode-K-means	0.5403	0.2731	1.0723	0.44	0.4402
	Online-K-means	1.0036	1.3485	1.1676	1.0043	0.8079
	Sequential-K-means	**0.059**	**0.0836**	**0.1142**	**0.1766**	**0.216**
	K-means++	1.113	1.0976	1.3025	0.7557	0.8944
	K-medoids	0.7646	1.1433	1.3437	1.1472	0.7811
DB	IBCA	0.7464	1.0436	0.7911	0.7799	0.7587
	Block-Mode-K-means	0.6532	0.6729	0.5079	0.5297	0.5748
	Online-K-means	0.7687	0.7557	0.6258	0.495	0.4731

	Sequential-K-means	0.7464	1.0436	0.7911	0.7799	0.7587
	K-means++	0.735	0.7254	0.6134	0.5707	0.5398
	K-medoids	0.7429	0.7054	0.6002	0.5154	0.6024

Table 11: Comparative results on the S3 dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	**0.0602**	**0.1643**	**0.2089**	**0.3142**	**0.4914**
	Block-Mode-K-means	0.9009	1.1579	1.2956	1.7059	1.3395
	Online-K-means	1.0932	1.3964	1.4017	1.4246	1.4246
	Sequential-K-means	**0.0602**	**0.1643**	**0.2089**	**0.3142**	**0.4914**
	K-means++	0.9422	1.3489	1.1727	1.7691	1.2206
	K-medoids	1.0335	1.4559	1.3767	1.3117	2.3305
DB	IBCA	0.9687	0.8973	0.8483	0.7616	0.7421
	Block-Mode-K-means	0.7107	0.7161	0.7131	0.633	0.6753
	Online-K-means	0.6666	0.7699	0.7149	0.6355	0.6355
	Sequential-K-means	0.9687	0.8973	0.8483	0.7616	0.7421
	K-means++	0.7874	0.7674	0.7116	0.6319	0.6335
	K-medoids	0.6929	0.7318	0.7121	0.6763	0.5164

Table 12: Comparative results on the S4 dataset

	Sample Size	15%	25%	50%	75%	100%
Dunn	IBCA	0.1991	0.1189	0.1258	0.158	0.2634
	Block-Mode-K-means	0.6495	1.1921	0.9446	1.1518	1.3645
	Online-K-means	0.9852	1.3183	1.2853	1.2128	1.8504
	Sequential-K-means	0.1991	0.1189	0.1258	0.158	0.2634
	K-means++	1.1877	1.5787	1.5168	1.6513	1.7998
	K-medoids	0.5339	0.9537	1.4689	1.5466	1.5538
DB	IBCA	0.8064	0.7998	0.8714	0.859	0.8131
	Block-Mode-K-means	0.7108	0.7036	0.7518	0.7325	0.6596
	Online-K-means	0.6802	0.6723	0.7119	0.658	0.5596
	Sequential-K-means	0.8064	0.7998	0.8714	0.859	0.8131
	K-means++	0.6326	0.6841	0.6847	0.6578	0.6002
	K-medoids	0.6892	0.742	0.6477	0.6177	0.641

SUMMARY

In this work, we propose an efficient method for clustering large data that does not fit into the system available physical memory. The key idea in handling the large dataset is to divide it into a fixed sample size and then perform clustering on each sampled data. We have used results of the previously computed cluster along with its boundary point to assign the cluster labeled to unlabeled data points. A series of experiments are performed on the benchmark datasets suggest that the proposed work can be utilized as an alternative to sequential k-means for clustering large data set as it has better running time complexity and similar cluster quality outcome. The Proposed IBCA algorithm competes with sequential k-means along the time and accuracy dimensions

CONFLICT OF INTEREST
There is no conflict of interest.

ACKNOWLEDGEMENTS
None

FINANCIAL DISCLOSURE
None

REFERENCES

[1] Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. [2016] an Algorithm for Online K-Means Clustering. Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX). 81-89
[2] Bahmani, Bahman, et al. [2012] Scalable k-means++. PVLDB. 622-633.
[3] Ailon, Nir, Jaiswal, Ragesh and Monteleoni, Claire. [2009] Streaming k-means approximation. NIPS. 10-18.
[4] Ackermann, R Marcel, et al. [2012] StreamKM++: A clustering algorithm for data streams. ACM Journal of Experimental Algorithmic.17(2):1-30
[5] Wu X, et al. [2007] Top 10 algorithms in data mining. Knowledge and Information System. 14(1): 1-37.
[6] MacQueen, J. [1967]5th Berkeley symposium on mathematical statistics and probability. 281-297.
[7] Jain AK. [2010] Data clustering: 50 years beyond k-means. Pattern Recognition Letters. 31: 651-666.

THE IIOAB JOURNAL
Guest editors: N. Arun Kumar & P. Mohammed Shakeel

[8] Gupta C, Grossman R. [2004] A Single Generalized Incremental Algorithm for Clustering. The Fourth SIAM International Conference on Data Engineering. 234-242.
[9] Hartigan JA. [1975]. Clustering Algorithms. New York: John Wiley & Sons.
[10] Hore, Prodip, et al. [2009] A Scalable framework for segmenting magnetic resonance images. J. Signal Process. Syst., 183-203.
[11] ZHANG T, RAMAKRISHNAN R, LIVNY M. [1996] an efficient data clustering method for very large databases. In Proceedings of the ACM SIGMOD. International Conference on Management of Data (SIGMOD '96). 103–114.
[12] Saradhi V Vijaya, Abraham P Charly. [2016] Incremental maximum margin clustering. Pattern Analysis and Applications. 19(4): 1057-1067.
[13] Patra, Bidyut kumar, et al. [2013] Distance based Incremental Clustering for Mining Clusters of Arbitrary Shapes. Pattern Recognition and Machine Intelligence. 229-236.
[14] Shinder, Michael, Wong, Alex and Meyerson. [2011] Fast and Accurate k-means For Large Datasets. NIPS. 2375-2383.
[15] Ester, M, et al. [1998] Incremental Clustering for Mining. Proceedings of the 24th International. Conference on Very Large Databases (VLDB'98). 323-333.
[16] Bryant Aaron, et al. [2014] Dynamic Incremental K-means Clustering. IEEE, International Conference on Computational Science and Computational Intelligence. 308-313.
[17] Ravi Sankar, Sangam ,Hari, Om. [2014], Hybrid data labeling algorithm for clustering large mixed data type. Journal of intelligent information system, 1-21.
[18] NP-hardness of Euclidean sum-of-squares clustering. Aloise, D, et al. [2009], Machine Learning, pp. 245-248.
[19] Tamir DE, Park C , Yoo B. [2007] The Validity of Pyramid K-means. Proc. SPIE conference on Optics and Photonics / Optical Engineering and Applications. 1225-1233.
[20] Arthur D, Vassilvitskii, S. New Oreeans LA. [2007] K-means++: the advantages of careful seeding. ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and applied Mathematics. 1027-1035.
[21] Chakraborty, S and Nagwani, N.K. [2011] Analysis and study of Incremental K-Means clustering algorithm. International conference in High Performance Architecture and Grid Computing. 338-341.
[22] Lichman M. {UCI} Machine Learning Repository. [2013] http://archive.ics.uci.edu/ml.
[23] Franti P, Virmajoki, O. [2006] Iterative shrinking method for clustering problems. Pattern Recognition. 761-765.
[24] Mirkin, B. [1996] Mathematical Classification and clustering. Springer.