An Efficient K-Means with Good Initial Starting Points

Fahim A.M.[1] Salem A. M.[2] Torkey F. A.[3] Ramadan M. A.[4] Saake G.[5]

[1]*Faculty of information, Otto-von-Guericke-University, Magdeburg, Germany, ahmmedfahim@yahoo.com*

[2]*Faculty of Computers & Information, Ain Shams University, Cairo, Egypt, absalem@asunet.shams.edu.eg*

[3]*Kaferel shiekh University, Kaferel sheikh, Egypt, fatorkey@Yahoo.com*

[4]*Faculty of Science, Minufiya University, Shbien el koum, Egypt, mramadan@mailer.eun.eg*

[5]*Faculty of information, Otto-von-Guericke-University, Magdeburg, Germany, saake@iti.cs.uni-magdeburg.de*

Abstract

The k-means algorithm is one of the most widely used methods to partition a dataset into groups of patterns. However, the k-means method converges to one of many local minima. And it is known that, the final result depends on the initial starting points (means). We introduce an efficient method to start the k-means with good starting points (means). The good initial starting points allow the k-means algorithm to converge to a "better" local minimum, also the number of iteration over the full dataset is decreased. Our experimental results show that, good initial starting points lead to improved solution.

Keywords: *clustering algorithms, k-means algorithm, and data clustering.*

1. Introduction

Classifying objects according to similarities is the base for much of science. Organizing objects into sensible grouping is one of the most fundamental modes of understanding and learning. Cluster analysis is the study of algorithms for grouping or classifying objects [8]. So a cluster is comprised of number of similar objects collected or grouped together. Clustering is a process in which a group of unlabeled objects are partitioned into a number of sets so that similar objects are assigned to the same cluster, and dissimilar objects are assigned to different clusters. There are two goals of clustering algorithms: (1) determining good clusters and (2) doing so efficiently. Clustering has become a widely studied problem in a variety of application domains including data mining and knowledge discovery [4], [7], data compression and vector quantization [5], pattern recognition and pattern classification [2], neural networks, artificial intelligence, and statistics.

Several clustering algorithms have been proposed. These algorithms can be broadly classified into hierarchical and partitioning clustering algorithms [8]. Hierarchical algorithms decompose a database *D* of *n* objects into several levels of nested partitioning (clustering), represented by a dendrogram (tree). There are two types of hierarchical algorithms; an agglomerative that builds the tree from the leaf nodes up, whereas a divisive builds the tree from the top down. Partitioning algorithms construct a single partition of a database *D* of *n* objects into a set of *k* clusters, such that the objects in a cluster are more similar to each other than to objects in different clusters.

The k-means clustering algorithm is the most commonly used [8] because it can be easily implemented, speed convergence to local minimum. However this local minimum depends on the initial starting means. In this paper, we introduce an efficient method to obtain good initial starting means, so the final result will be better than that of randomly selected initial starting means. How to get good initial starting means becomes an important operational objective. To solve this problem, [1] proposed an algorithm based on selecting *J* of subsamples, cluster them independently producing *J* estimates of the true cluster locations, and apply the k-means on this result to get the refined initial starting points. While this method can improve the final results but the final results depend on the quality of the selected subsamples, number of samples and the size of sample. We

propose an efficient method for implementing the k-means algorithm. It can produce better clustering results on most cases. Our algorithm scans the dataset block by block; produces *k* representative objects for each block, the algorithm keeps the results from each block, and applies the k-means algorithm to the collected results from all blocks to get the initial starting points. In other words, the proposed algorithm compressed the data into smaller dataset by producing *k* means from each block, if the dataset contains *j* blocks, then the compressed data will contains *k*j* objects. Our algorithm scans the original dataset two times and produces better clusters. This paper is organized as follows, in section 2 we present the algorithms related to our work. In section 3 we present our proposed algorithm which we refer to it as k-means block. We present some experimental results in section 4 and conclude with section 5.

2. Related Work

There are many algorithms for clustering datasets. The k-means clustering is the most popular method used to divide *n* patterns $\{x_1, \ldots, x_n\}$ in *d* dimensional space into *k* clusters[8]. The result is a set of *k* centers, each of which is located at the centroid of the partitioned dataset. This algorithm can be summarized in the following steps:

1. **Initialization**: Select a set of *k* starting points $\{m_j\}$, $j = 1, 2, \ldots, k$. The selection may be done in random manner or according to some heuristic.
2. **Distance calculation**: For each pattern x_i , $1 \le i \le n$ compute its Euclidean distance to each cluster centroid m_j, $1 \le j \le k$, and then find the closest cluster centroid m_j and assign the object x_i to it.
3. **Centroid recalculation**: For each cluster j, $1 \le j \le k$ recompute cluster centroid m_j as the average of the data points assigned to it.
4. **Convergence condition**: Repeat steps 2 and 3 until convergence.

To choose a proper number of clusters *k* is a domain dependent problem. To resolve this, some researchers have proposed methods to perform k-clustering for various numbers of clusters and employ certain criteria for selecting the most suitable value of *k* [11] and [10].

Several variants of the k-means algorithm have been proposed. Their purpose is to improve efficiency or find better clusters. Improved efficiency is usually accomplished by either reducing the number of iterations to reach final convergence or reducing the total number of distance calculations, you can see [3], [6]. The k-means algorithm randomly selects *k* initial cluster centers from the original dataset. Then, the algorithm will converge to the actual cluster centers after several iterations. Therefore, choosing a good set of initial cluster centers is very important for the algorithm. However, it is difficult to select a good set of initial cluster centers randomly.

Bradley and Fayyad have proposed an algorithm for refining the initial cluster centers. We present a simple review of this algorithm. For more details see [1]. This algorithm select *j* subsamples from the dataset, apply the k-means algorithm on each subsample independently. If, at termination of K-Means, there are empty clusters then reassigning all empty clusters to points farthest from their respective centers. At the end of this phase there will be set of data contains *k*j* objects, since each subsample represented by *k* representative objects. Then they apply the k-means for the second time to produce the *k* refined initial centers from *k*j* objects. Note that, in this algorithm the k-means algorithm will be applied *j* times, in each time it takes the *k* representative for a sample as starting point, and distributes the *k*j* objects over these representatives, keeps the mean squared error for each sample of starting point, this process is similar to CLARA algorithm [9] with small difference, this difference is the CLARA restricts the goodness of representative to one sample only, while this algorithm test the goodness of representative over all samples. This algorithm can be summarized in the following steps:

0. $CM = \varphi$
1. For i=1,…,J
   a. Let S_i be a small random subsample of data
   b. Let CM_i = KMeans (SP, S_i, K)
   c. $CM = CM \cup CM_i$
2. $FMS = \varphi$
3. For i=1,…,J
   a. Let FMi = KMeans(CM_i, CM, K)
   b. Let $FMS = FMS \cup FMi$
4. Let $FM = \underset{FM_i}{ArgMin}\{Distortion(FM_i, CM)\}$
5. Return (FM)

In this algorithm the KMeans function takes three input parameters; they are the initial starting points (SP), the random subsample (S_i) and the required number of clusters (K). This algorithm based on samples, the authors use sample of size 1% of the full dataset size, and use 10 subsamples. By examining step 3 of the algorithm, we find that the final result produced from one sample of size 10% from the full dataset. And the good result based on sample is not necessary good for the full dataset. Our proposed algorithm is similar to this algorithm, but it does not depend on sample. Our algorithm partitions the dataset into blocks and applies the k-means on each block. Our proposed method uses a simple process to obtain initial cluster centers from a compressed dataset. These centers are very close to the actual cluster centers of the original dataset, and so only a few iterations are needed for convergence.

3. The Proposed Algorithm

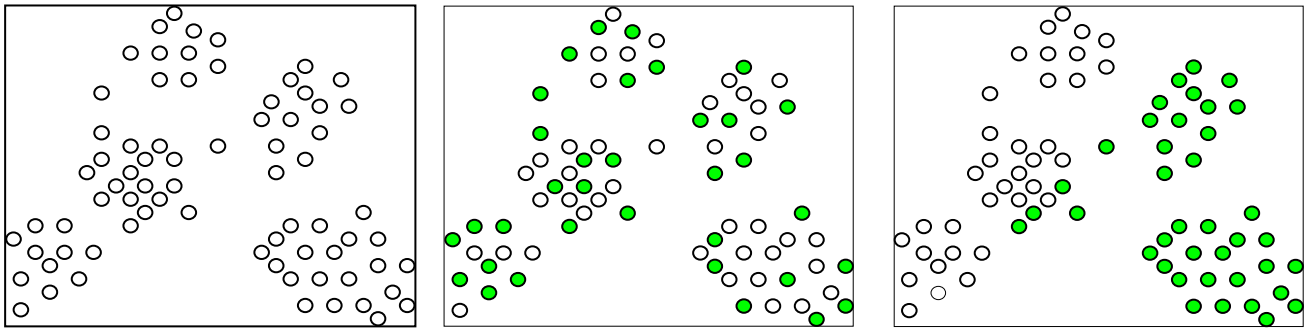
In this section we describe our algorithm. That produces good starting points for the k-means algorithm instead of selecting them randomly. And this will leads to better clusters at the final result than that of the original k-means.

3.1 The proposed Methodology

The main idea of this algorithm is to compress the dataset into finite number of representative. Each representative is the mean value of some data points form a small cluster. In our algorithm we compress the data set of size N into smaller data set of size $k*m$; where k is the required number of partition for each block, m is the number of blocks. This process has been done at the first phase. In the second phase we apply the k-means on the compressed dataset, to get the k representative points that will be the initial starting points for the k-means on the full dataset. The idea of compression of dataset comes from the BIRCH algorithm. The BIRCH algorithm [12] compresses the dataset into cluster feature vectors based on some statistics like the size of the leaf node and the branching factor for the internal nodes. The pre-clustering algorithm employed by BIRCH is to reduce the size of data set, is incremental and approximate. During pre-clustering, the entire database is scanned, and cluster summaries are stored in memory in a data structure called the CF-tree. For each successive data point, the CF-tree is traversed to find the closest cluster to it in the tree, and if the point is within a threshold distance of the closest cluster, it is absorbed into it. Otherwise, it starts its own cluster in the CF-tree. Once the clusters are generated, a final labeling phase is carried out in which using the centroids of clusters as seeds, each data point is assigned to the cluster with the closest seed.

3.2 The proposed K-means Block Algorithm.

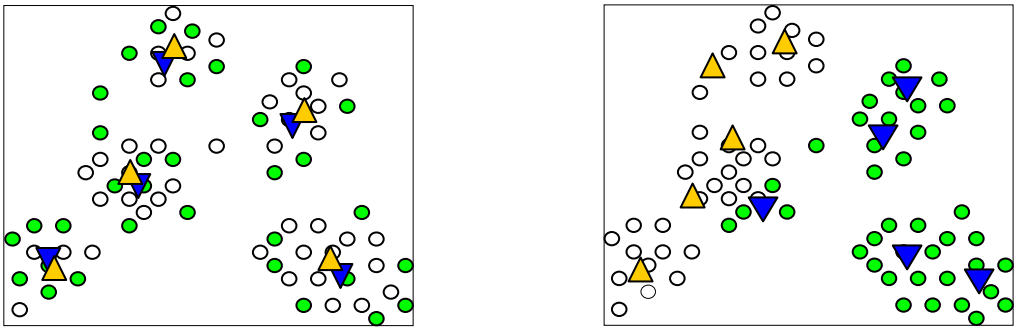
Returns to our proposed algorithm, we divide the data set into 10 blocks in our experimental results. The size of block varies according to the size of data. We apply the k-mean on each block independently, to produce *k* centers for each block, so if the dataset partitioned into *m* blocks, then the compressed data will contains *m*k* objects. Each block is small compared with the full dataset. Also the compressed data is very small regards to the full dataset. We apply the k-means on the compressed data to produce *k* centroids. We take the resulting *k* centroids to be the initial starting points for the k-means algorithm that works on the full dataset. In this case the number of iteration will be decreased, and a better means will be founded. Our algorithm regards the effect of each point on the means, and does not restrict the effect of the sample like in [1]. Our algorithm scan the data two times; at the first phase to compress the dataset, the second time when distribute the full dataset points over the initial centroids produced from the compressed dataset. When we look at each block, we may find that the block represents a sample of dataset, if the block contains objects distributed over all the space, so the *k* representative means for this block will be lie near to the final means of the full dataset. On the other side, if the block contains objects that fall in a small portion of the space, then the *k* representatives for this block will be very close to each other and some of them will be merged together when applying the k-means on the compressed dataset. Figure 1 explains our idea, the dataset contains five clusters, and we partition the dataset into two blocks. Figure 1.a represents the dataset, Fig.1.b and 1.c represent two probability for the data in each block; in Fig.1.b the block represents a sample of points distributed over all the data space, while in Fig.1.c the block represents an area of the data space.



a) Dataset b) Two blocks of the data c) Two blocks of the data

Fig. 1: Dataset contains 5 clusters, the 2 blocks of the data may be like as in b or c

When we apply the k-means on the two blocks represented by Fig 1.b and 1.c we will obtain the result shown in Figure 2



a) Result from Fig. 1.b b) Result from Fig. 1.c

△The k means of white block (white circles), ▽ the k means of green block (green circles)

Fig. 2: The compressed dataset generated from 2 blocks of data

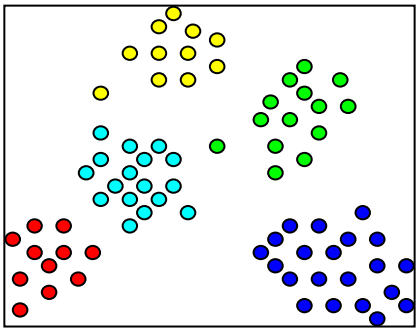


Fig. 3 the final 5 clusters

At this point we obtain the compressed dataset that contains *m*k* (*k* representative points for each block) means resulting from the *m* blocks. We apply the k-means algorithm on this compressed dataset to find the final *k* means. This final *k* means will be the initial starting points for applying the k-means algorithm over the full dataset. Returns to our example, the final *k* means for the compressed dataset will be nearly the same for the two probabilities of the blocks in Fig. 1. And the final result of our algorithm will be as in Figure 3.

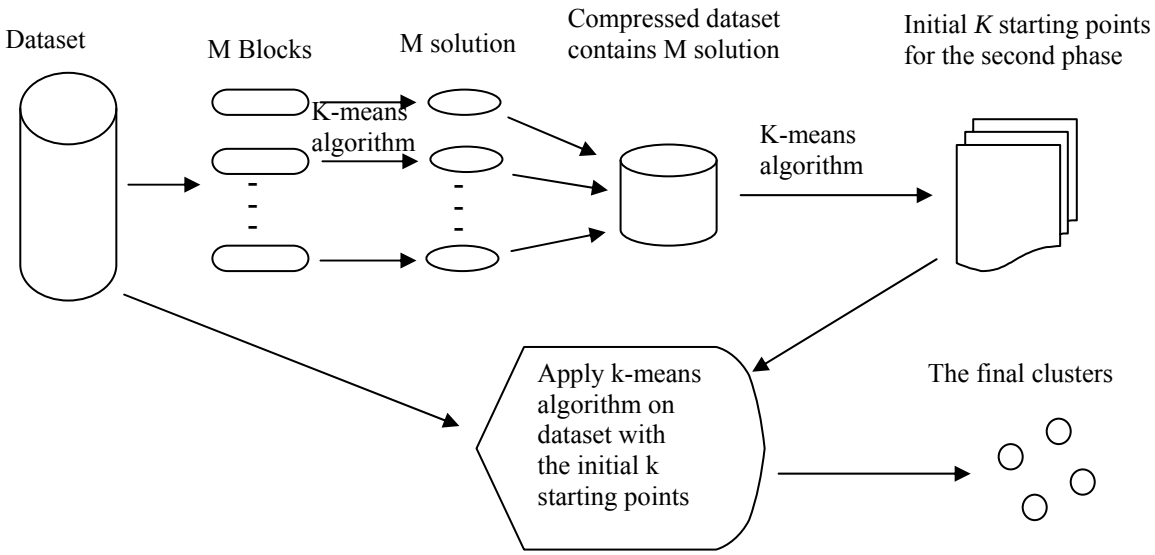


Fig. 4.a: An overview of the k-means block

1. Set the size of the block
2. i=0
3. While not end of file
4. Read the $Block_i$
5. k-means($Block_i$, *k*)
6. (Write/Append) the means to output file
7. i=i+1
8. End while
9. k-means(compressed dataset, *k*)
10. k-means(dataset, final means, *k*)

Fig. 4.b: The main steps of the proposed algorithm.

Fig. 4: The K-means block algorithm

From the previous Figures (1 to 3) we note that; when the block represents a sample distributed over all the data space, the generated means will be very condensed on specified areas as in Fig 2.a. But when the block represents a sample distributed over certain area of the data space, the generated means will be scattered over the data space, but we find that these means scattered over groups as in Fig.2.b. Figure 4 summarizes the steps of the k-means block algorithm.

Note that, in figure 4.b, the algorithm determines the size of each block and the user should determine the required number of partitions in each block. The value of *k* may be changed when applying the k-means on the compressed and the full dataset. From experimental results it will be better when we use small value for *k* at the first phase of our algorithm, at the second phase we change the value of *k* to the required number of clusters as a final results. So the value of *k* in steps 9, 10 may be different from the value of *k* in step 5. In line 10, the k-means start with the *k* points generated from the compressed dataset in line 9 and applied on the full dataset.

3.3 Computational Complexity

The k-means block algorithm is primarily intended to work on large datasets. Since our method works on small blocks of the data, the initialization of good starting points is fast. We use block sizes of 10% of the full dataset size, so to get the compressed dataset it will require $O(mbkl)$; m is the number of blocks, b is the size of block, k is the number of clusters and l is the number of iterations. At the worst case this time will be equals to the time required by the k-means to cluster the full dataset. The time required to get the final k means from the compressed dataset is negligible, since the compressed dataset is very small compared with the full dataset. At the final phase we apply the k-means over the full dataset starting with the final k means resulting from the compressed dataset, this lead to a few number of iteration of the k-means. The time complexity of the second phase is $O(nk)$, where n is the size of the full dataset, and k is the number of clusters. So the overall time complexity will be $O(nk + mbkl)$. At the worst case this time will be the running time of the k-means two times.

4. Experimental Results

We have evaluated our proposed algorithm on both synthetic and real datasets. We compared our results with that of the original k-means algorithm in terms of the total quality of clusters and the execution time for both algorithms. We give a brief description of the datasets used in our algorithm evaluation. Table 1 shows some characteristics of these datasets.

Table 1: Characteristic of the Datasets

Dataset	**Number of Records**	**Number of Attributes**	**Type of dataset**
Db1	3159	2	Synthetic
Letters	20000	16	real
Iris	150	4	real
Abalone	4177	8	real

Table 2: The results of the proposed algorithm compared with that of k-means

dataset	k	k-means			k-means block		
		MSE	iterations	Time in sec.	MSE	iterations	Time in sec.
Db1	3	5.221	13	< 1	4.776	2	< 1
letters	26	139.315	18	18	135.778	11	27
letters	30	161.203	45	51	159.659	14	36
letters	20	112.765	23	18	112.718	13	21
letters	10	65.210	8	3	64.267	5	7
Iris	3	1.942	4	< 1	1.942	4	< 1
Abalone	29	8.913	4	< 1	7.582	4	2

Note that: The value of *k* in table 2 is fixed for the two phase of the algorithm.

From the results in table 2 and in figures from 5 to 8, we find that our proposed algorithm produces better results than that of the k-means algorithm; our proposed algorithm is designed for large datasets. And it produces better result as the dataset contains large number of clusters. In this case the run time of the proposed algorithm will be smaller than that of the k-means. And we can decrease the running time by inserting small value for k when applying the k-means on the blocks provided that the compressed dataset contains number of points larger than the final number of clusters in the full dataset. For example, when we partition the letters dataset into 30 clusters, we partition each block into 10 clusters, the compressed dataset contains 100 representative objects, and then we partitioned this dataset into 30 clusters to get the initial starting points for the k-means in the next phase. The MSE is 156.397171, the iteration is 22, and the running time is 30 second as shown in Figure 8. And this result is better than that exist in table 2. This occurs because the final result will depends on the size of the block, number of clusters, this number may be constant along the run time of the algorithm, and we can make it varies on the compressed dataset. When the dataset contains large number of clusters (>40) it is better to partition each block to small number of clusters, provided that the resulting compressed dataset contains sufficient number of points(> the number of clusters in the full dataset).

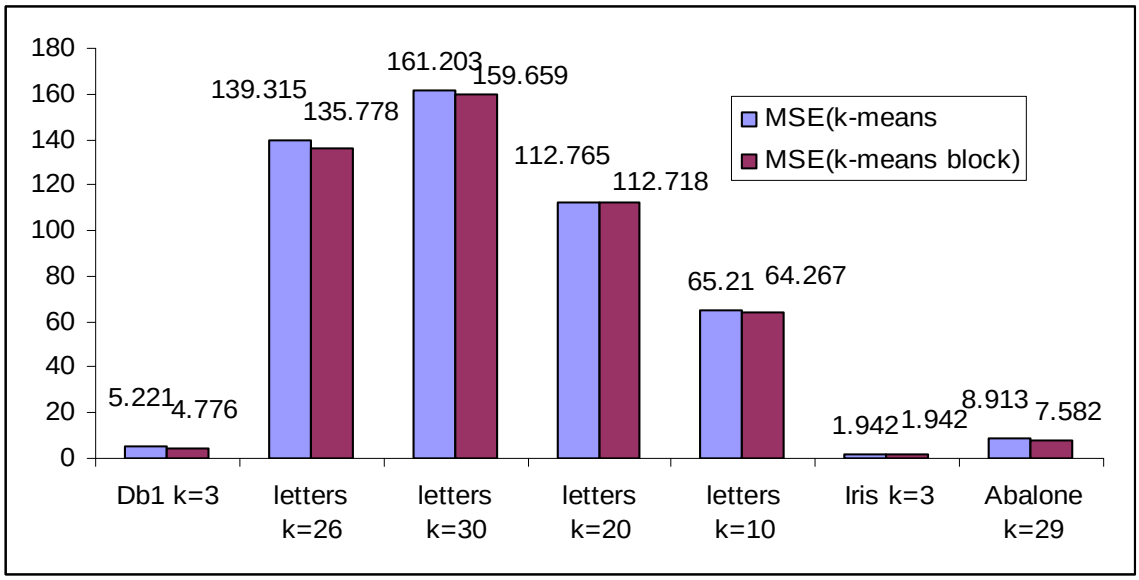


Fig. 5: mean square error comparison between k-means and k-means block

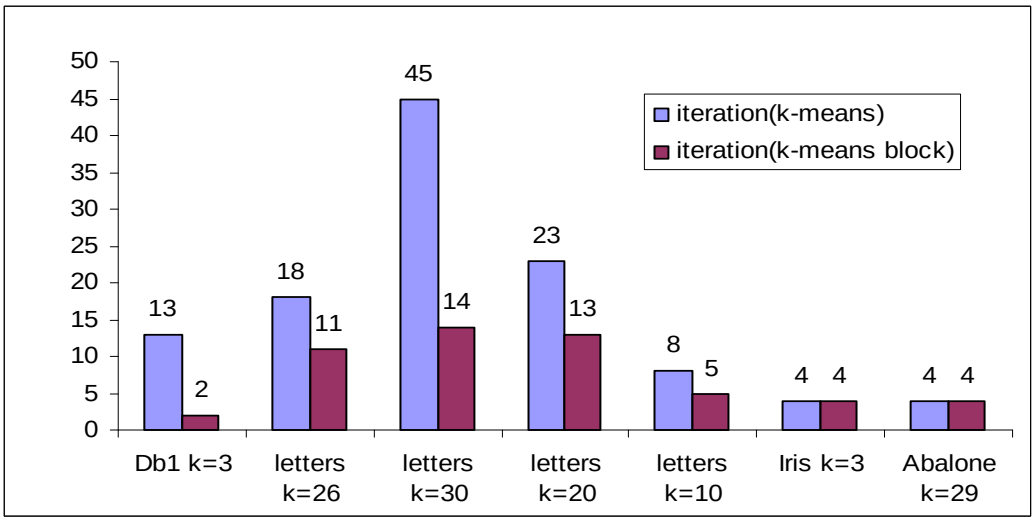


Fig. 6: Iteration comparison between k-means and k-means block

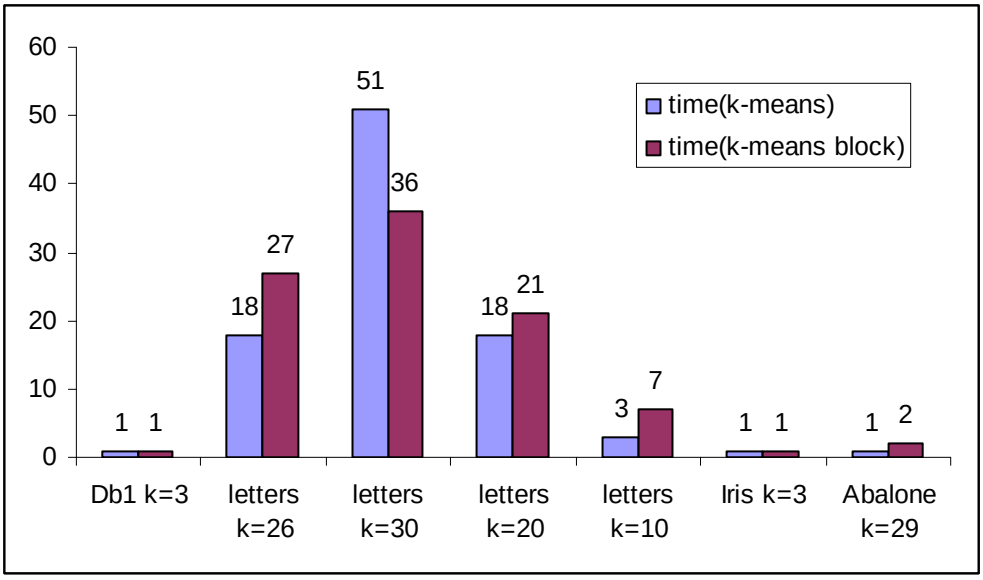


Fig. 7: Run time comparison between k-means and k-means block

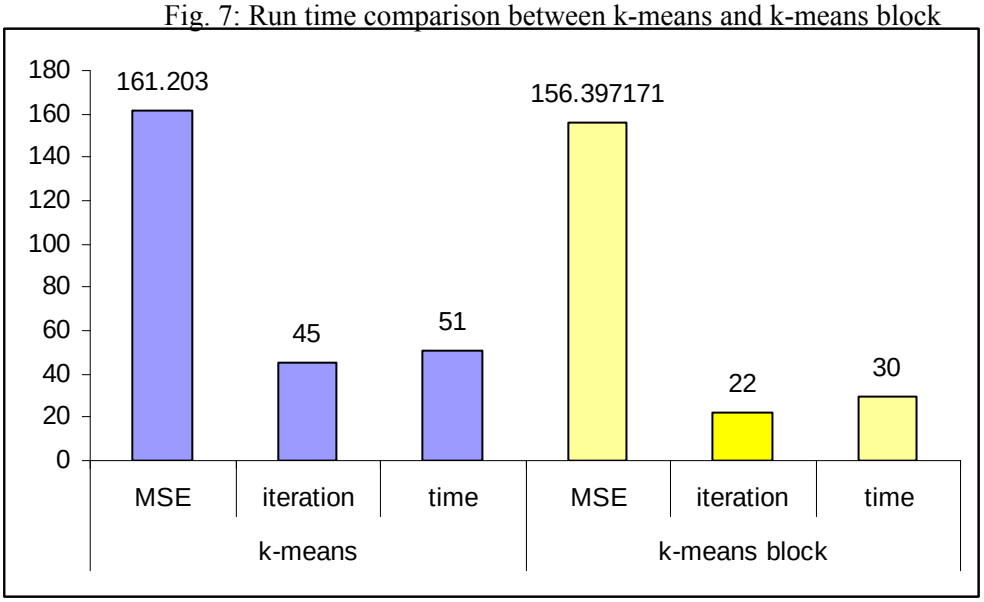


Fig. 8: Great improvement in run time, iteration and MSE, when each block is portioned into small number of clusters, producing the required number of clusters at the final phase.

Table 3: The results of the proposed algorithm compared with that of k-means
On letters data set with small value for *k* at the first phase and large value for *k* at the second phase

K 1st Phase	K 2nd phase	k-means			k-means block		
		MSE	iterations	Time in sec.	MSE	iterations	Time in sec.
10	30	161.203	45	51	156.397	22	30
20	60	277.430	30	69	269.835	12	38
10	90	387.869	18	62	372.735	16	59

From our experimental results in table 3, we recommend to use small value for *k* when we apply the k-means on the blocks. And the actual value for *k* when applying the k-means on the compressed dataset and the full dataset. These results are shown in Figures 8, 9, and 10. We use the letters dataset since it is the largest dataset we have, it contains 20000 points in 16 dimensions.

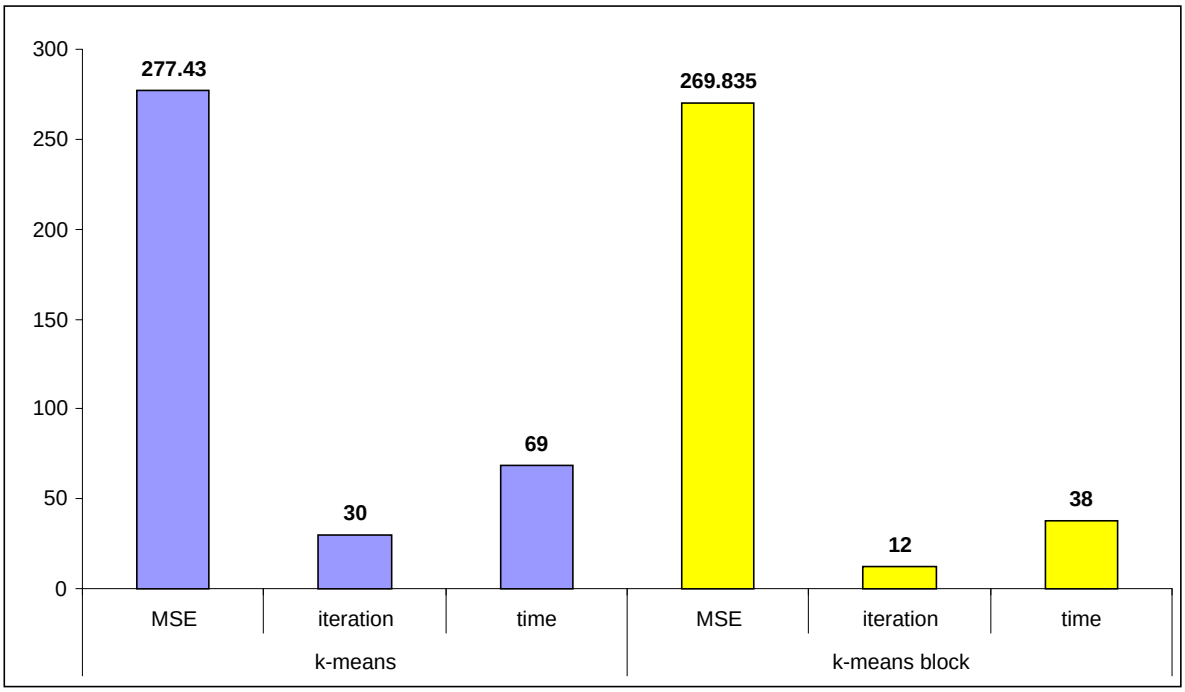


Fig. 9: Great improvement in iteration and MSE, when each block is portioned into 20 clusters, producing 60 clusters at the final phase.

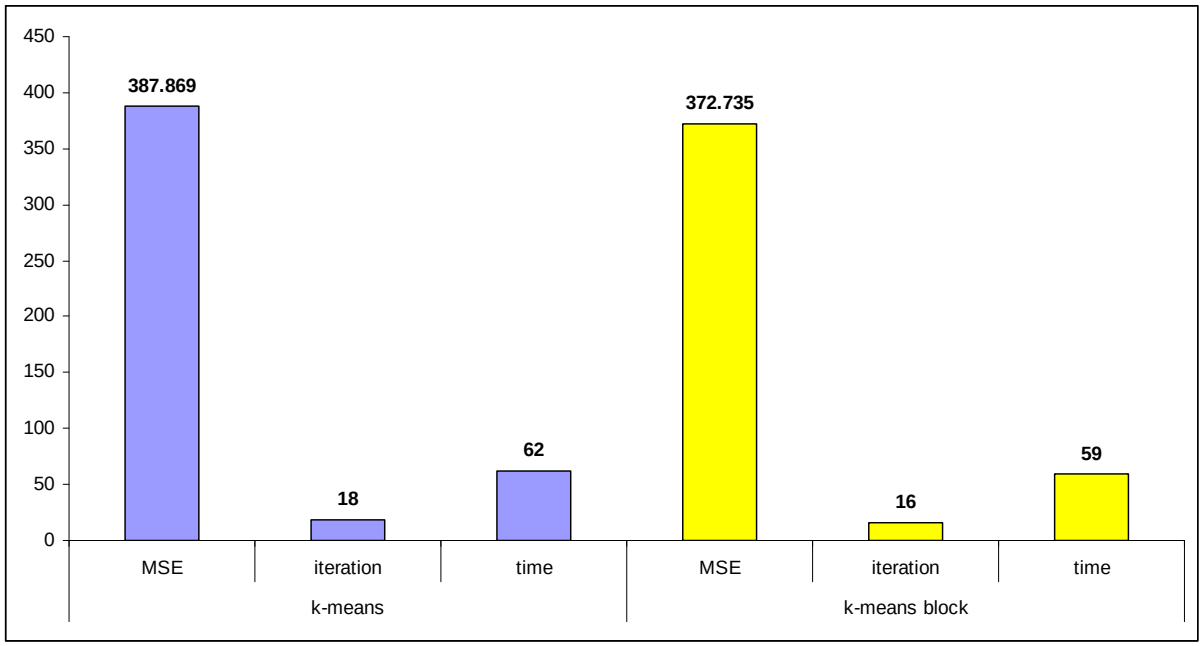


Fig. 10: Great improvement MSE, when each block is portioned into 10 clusters, producing 90 clusters at the final phase.

So our proposed algorithm is suitable for large datasets, and it perform well when the dataset contains large number of clusters, and this proves that the good initial starting points lead to a better clustering solution.

5. Conclusion

In this paper we have presented a simple idea that make the k-means more efficient and produce good quality clusters. Our idea depends on the good selection of the starting points for the k-means. To select a good initial solution we partition the dataset into blocks, and applied the k-means on each block, we get a compressed dataset, we apply the k-mean on it with the same value of *k* or larger to get good starting means, and redistributes the points in the full dataset over these starting means. Our experimental results demonstrated that the proposed algorithm produces better results than that of the k-means algorithm.

References

[1] Bradley P. S., Fayyad U. M. "Refining Initial Points for K-Means Clustering", Proc. of the 15th International Conference on Machine Learning (ICML98), J. Shavlik (ed.), Morgan Kaufmann, San Francisco, 1998, pp. 91-99.

[2] Duda, R.O., Hart, P.E., "Pattern Classification and Scene Analysis". John Wiley & Sons, New York, 1973.

[3] Fahim A. M., Salem A. M., Torkey F. A. and Ramadan M. "An efficient enhanced k-means clustering algorithm". Journal of Zhejiang University Science A, 2006, vol 7(10), pp. 1626-1633.

[4] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy,R., "Advances in Knowledge Discovery and Data Mining". AAAI/MIT Press, 1996.

[5] Gersho, A., Gray, R.M., 1992. "Vector Quantization and Signal Compression". Kluwer Academic, Boston.

[6] Hung M., WU J., Chang J. and Yang D.,"An Efficient k-Means Clustering Algorithm Using Simple Partitioning", journal of Information Science and Engineering, 2005, vol. 21, pp. 1157-1177.

[7] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," Data Mining and Knowledge Discovery, 1998, vol. 2, , pp. 283-304.

[8] Jain, A.K., Dubes, R.C., "Algorithms for Clustering Data". Prentice-Hall Inc., 1988.

[9] Ng R. T., Han J.: "Efficient and Effective Clustering Methods for Spatial Data Mining", Proc. 20th Int. Conf. On Very Large Data Bases, Santiago, Chile, Morgan Kaufmann Publishers, San Francisco, CA, 1994, pp. 144-155.

[10] Pham D. T., Dimov S. S., and Nguyen C. D., "Selection of *k* in K-means clustering". Mechanical Engineering Science, 2004, vol. 219. pp.103-119.

[11] Ray S. and Turi R. H., "Determination of number of clusters in k-means clustering and application in colour image segmentation.", in Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques, 1999, pp. 137-143,.

[12] Zhang T., Ramakrishnan R., Linvy M.: "BIRCH: An Efficient Data Clustering Method for Very Large Databases". Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM Press, New York, 1996, pp.103-114.

This article contains

10 Figures

3 Tables

Article received: 2008-06-02