IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-2 E-ISSN: 2347-2693

A Fast Global *k*-means Algorithm for Datasets having Streaming Behavior

Purnendu Das[1], Bishwa Ranjan Roy[2*], Sanju Das[3]

[1]Dept. of Computer Science, Assam University, Silchar, India
[2*]Dept. of Computer Science, Assam University, Silchar, India
[3]Dept. of Computer Science, Assam University, Silchar, India

[*]*Corresponding Author: brroy88@gmail.com, Tel.: +91-94356-73134*

Available online at: www.ijcseonline.org

Received: 20/Jan/2018, Revised: 30/Jan/2018, Accepted: 14/Feb/2018, Published: 28/Feb/2018

***Abstract*—**The *k*-means algorithm and its variations are known to be fast clustering algorithms. However, they are sensitive to the choice of starting points and are inefficient for solving clustering problems in large datasets. Recently, incremental approaches have been developed to resolve difficulties with the choice of starting points. The global *k*-means and the fast global *k*-means algorithms are based on such an approach. They iteratively add one cluster center at a time. Numerical experiments show that these algorithms considerably improve the k-means algorithm. However, they require storing the whole affinity matrix or computing this matrix at each iteration. This makes both algorithms time consuming and memory demanding for clustering even moderately large datasets. Also the continuously arriving data stream has become common phenomenon for many fields recent years; for example, sensor networks, web click stream and internet traffic flow. Researchers proposes many innovative technologies to manage such streaming datasets. Finding efficient data stream mining algorithm has become an important research subject. In this paper we propose a fast global *k*-means algorithm for datasets having streaming behavior. Experiment shows that our proposed algorithm is more efficient than the fast global *k*-means algorithm in case of streaming datasets.

***Keywords*—** k-Means, Global k-Means, Fast Global k-Means, Data Streaming.

I. INTRODUCTION

The continuously arriving data stream has become common phenomenon for many fields recent years; for example, sensor networks, web click stream and internet traffic flow. Researchers proposes many innovative technologies to manage such streaming datasets. Finding efficient data stream mining algorithm has become an important research subject. Data stream [1] is potential infinite, with uncertain arriving speed and can be scanned one pass. The processing of data stream has to implement within a limited space (memory) and a strict time constraint. Due to this, an efficient data stream mining algorithms must satisfy a more strict demand.

A fundamental problem that frequently arises in a great variety of fields such as pattern recognition, image processing, machine learning and statistics is the clustering problem [2]. In its basic form the clustering problem is defined as the problem of finding homogeneous groups of data points in a given data set. Each of these groups is called a cluster and can be defined as a region in which the density of objects is locally higher than in other regions.

The simplest form of clustering is partitional clustering which aims at partitioning a given data set into disjoint subsets (clusters) so that specific clustering criteria are optimized. The most widely used criterion is the clustering error criterion which for each point computes its squared distance from the corresponding cluster center and then takes the sum of these distances for all points in the data set. A popular clustering method that minimizes the clustering error is the *k*-means algorithm. However, the *k*-means algorithm is a local search procedure and it is well known that it suffers from the serious drawback that its performance heavily depends on the initial starting conditions [2]. To treat this problem, several other techniques have been developed that are based on stochastic global optimization methods (e.g. simulated annealing, genetic algorithms). However, it must be noted that these techniques have not gained wide acceptance and in many practical applications the clustering method that is used is the *k*-means algorithm with multiple restarts [3], [4], [5], [6].

Different approaches to improve the efficiency of the *k*-means algorithm have been proposed [7], of which incremental ones are among the most successful. In these approaches clusters are computed incrementally by solving all intermediate clustering problems. The global *k*-means

algorithm (GKM) proposed in [8] and the modified global *k*-means algorithm (FGKM) proposed in [9] are incremental clustering algorithms. Results of numerical experiments presented in [9] show that these algorithms allow one to find global or a near global minimizer of the cluster (or error) function.

Global *k*-means clustering algorithm (GKM), which constitutes a deterministic effective global clustering algorithm for the minimization of the clustering error that employs the *k*- means algorithm as a local search procedure. The algorithm proceeds in an incremental way: to solve a clustering problem with M clusters, all intermediate problems with $1, 2, \ldots, M$ 1 clusters are sequentially solved. The basic idea underlying the proposed method is that an optimal solution for a clustering problem with M clusters can be obtained using a series of local searches (using the *k*-means algorithm). At each local search the M 1 cluster centers are always initially placed at their optimal positions corresponding to the clustering problem with M 1 clusters. The remaining M^{th} cluster center is initially placed at several positions within the data space. Since for $M = 1$ the optimal solution is known, we can iteratively apply the above procedure to 2nd optimal solutions for all *k*- clustering problems $k = 1, \ldots, M$. In addition to effectiveness, the method is deterministic and does not depend on any initial conditions or empirically adjustable parameters. These are significant advantages over all clustering approaches mentioned above.

A new version of the modified global *k*-means algorithm (FGKM) is proposed in [9]. An auxiliary cluster function has been applied to generate a set of starting points lying in different parts of the dataset. The *k*-means algorithm is applied starting from these points to minimize the auxiliary cluster function and the best solution is selected as a starting point for the next cluster center. Exploit the information gathered in previous iterations of the incremental algorithm to avoid computing the whole affinity matrix. Also the triangle inequality for distances is used to avoid unnecessary computations. The results demonstrate that the FGKM is far more efficient than the GKM.

A lot of work has already done in the area of clustering the streaming datasets. The STREAM algorithm is proposed to cluster data streams in [10]. It first determines the size of sample. If the size of data chunk exceeds the sample size, a LOCALSEARCH procedure is invoked to obtain the clusters of the chunk. Finally, the LOCALSEARCH is applied to all the cluster centers generated in the previous iterations. The extended *k*-means algorithm and the VFKM algorithm is proposed in [11]. It is guaranteed that the model produced does not differ significantly from the one that would be obtained with infinite data. In [12], the CluStream algorithm is proposed to cluster evolving data streams. CluStreams idea is dividing the clustering process into an online component which periodically stores detailed summary statistics and an offline component which uses only this summary statistics. A pyramidal time frame in conjunction with a micro-clustering approach is used to deal with the problems of efficient choice, storage, and use of this statistical data for a fast data stream.

However as per our knowledge, no work has been done to propose any fast global *k*-means algorithm for datasets having streaming behavior. As explained earlier, streaming dataset has different properties than normal dataset. Applying fast global *k*-means directly does not give the optimum results for stream dataset. Our work is inspired by the weighted fuzzy c-means algorithm proposed for streaming dataset in [13]. In this paper, we proposed a fast global *k*-means algorithm (WFGKM) for the datasets having streaming behavior. We compare our proposed algorithm with the fast global *k*-means algorithm (FGKM). Our experiments finds that the proposed WFGKM algorithm is more efficient than fast FGKM.

The rest of the paper is organized as follows. In the next section we discussed related works. Section II gives the background details required for this paper. We explained our proposed algorithm in section III. The experimental comparisons and analysis are given in section IV. Finally we conclude the paper in section V.

II. RELATED WORK

A. Global k-means algorithm (GKM)

Suppose we are given a data set $X = \{x_1, \ldots, x_N\}$, $x_n \in R^d$. The M -clustering problem aims at partitioning this data set into M disjoint subsets (clusters) $C_1, \ldots, C_M$, such that a clustering criterion is optimized. The most widely used clustering criterion is the sum of the squared Euclidean distances between each data point x_i and the centroid m_k (cluster center) of the subset C_k which contains x_i. This criterion is called clustering error and depends on the cluster centers $m_1, \ldots, m_M$:

$$E(m_1, \ldots\ldots\ldots, m_N) = \sum_{i=1}^{N}\sum_{k=1}^{M} I(x_i \in C_k)||x_i - m_k||^2, \quad (1)$$

Where $I(X) = 1$ if X is true and 0 otherwise.

The *k*-means algorithm gives locally optimal solutions with respect to the clustering error. It is a fast iterative algorithm that has been used in many clustering applications. It is a point- based clustering method that starts with the cluster

centers initially placed at arbitrary positions and proceeds by moving at each step the cluster centers in order to minimize the clustering error. The main disadvantage of the method lies in its sensitivity to initial positions of the cluster centers. Therefore, in order to obtain near optimal solutions using the *k*-means algorithm several runs must be scheduled differing in the initial positions of the cluster centers.

The global *k*-means clustering algorithm constitutes a deterministic global optimization method that does not depend on any initial parameter values and employs the *k*-means algorithm as a local search procedure. Instead of randomly selecting initial values for all cluster centers as is the case with most global clustering algorithms, global *k*-means proceeds in an incremental way attempting to optimally add one new cluster center at each stage.

More specifically, to solve a clustering problem with *M* clusters the method proceeds as follows. Start with one cluster ($k = 1$) and find its optimal position which corresponds to the centroid of the data set *X*. In order to solve the problem with two clusters ($k = 2$), perform *N* executions of the *k*- means algorithm from the following initial positions of the cluster centers: the first cluster center is always placed at the optimal position for the problem with $k = 1$, while the second center at execution *n* is placed at the position of the data point x_n, ($n = 1, \ldots, N$). The best solution obtained after the *N* executions of the *k*-means algorithm is considered as the solution for the clustering problem with $k = 2$. In general, let $(m_1^*(k), \ldots, m_N^*(k))$ denote the final solution for *k*-clustering problem. Once the solution for the (k1)-clustering problem is found, the solution of the *k*-clustering problem is as follows: Perform *N* runs of the *k*- means algorithm with *k* clusters where each run *n* starts from the initial state $(m_1^*(k1), \ldots, m_{k-1}^*(k1), x_n)$. The best solution obtained from the *N* runs is considered as the solution $(m(k), \ldots, m_k^*(k))$ of the *k*-clustering problem. The above algorithm finally obtain a solution with *M* clusters having also found solutions for all *k*-clustering problems with $k < M$.

The latter characteristic can be advantageous in many applications where the aim is also to discover the correct number of clusters. To achieve this, one has to solve the *k*-clustering problem for various numbers of clusters and then employ appropriate criteria for selecting the most suitable value of *k* [11]. In this case, the proposed method directly provides clustering solutions for all intermediate values of *k*, thus requiring no additional computational effort.

The rationale behind the proposed method is based on the following assumption: an optimal clustering solution with *k* clusters can be obtained through local search (using *k*-means) starting from an initial state with

- the $k - 1$ centers placed at the optimal positions for the $(k - 1)$-clustering problem and
- the remaining k^{th} center placed at an appropriate position to be discovered.

B. The fast global k-means algorithm (FGKM)

Based on the general idea of the global *k*-means algorithm, several heuristics can be devised to reduce the computational load without significantly affecting the quality of the solution. To make the execution of global *k*-means algorithm faster a modified global *k*-means algorithm called fast global *k*- means (FGKM) has been proposed in [14], [15], [16]. In this algorithm, during each iteration of the incremental procedure, instead of executing *k*-means for all the data variables in the data set and decide the next cluster, it selects a single data from the entire data set as the initial center for the next cluster and continue with *k*-means algorithm. The selection of the single data from the data set is done by the following procedure. In order to compute an initial center, define v_i for each object x_i as following:

$$v_i = \sum_{j=1}^{N} \frac{d_{ij}}{\sum_{l=1}^{N} d_{jl}}, i = 1, \ldots \ldots, N \qquad (2)$$

The point that minimizes v_i is the one which has a comparatively high density around it, that is to say the sample with the minimum v_i tends to be the best center of a cluster. Another parameter is required to obtain the next initial cluster center. Suppose that the solution of the (k-1)-clustering problem is $(m_1^*(k-1), \ldots, m_{k-1}^*(k-1))$ and a new cluster center (i.e., the k^{th} initial center) is added at the location x_i that minimizes f_i as defined in Equation 3. Then we execute the *K*-means algorithm to obtain the solution with *k* clusters.

$$f_i = \frac{v_i}{\sum_{j=1}^{k-1} d(x_i, m_j^{(k-1)})}, i = 1, \ldots \ldots, n \qquad (3)$$

The addition of the parameter (i.e. the denominator of f_i) ensures that the new cluster center could be far away from the existing cluster centers. It should be noted that the new center we computed it by Equation 3 is an optimal initial cluster center.

The algorithm can be described as follows:

1) (Initialization) Calculate the distance between each pair of all the objects based on Euclidean distance, then calculate v_i for each object as defined in Equation 2. Select the point that minimize v_i as the first center. Set $q = 1$

2) (Update centroids) Apply *k*-means algorithm and preserve the best *q*-partition obtained and their cluster centers ($m_1, m_2, \ldots, m_q$).
3) (Stopping criterion) Set $q = q + 1$. If $q > M$, then
Stop
4) (Select the new cluster center) Calculate f_i for object x_i as defined in Equation 3. Select the point which has the minimum value of f_i as the new cluster center, now the initial center is ($m_1, m_2, \ldots, m_q, x_i$) and go to Step2.

This version of the GKM algorithm has an excellent feature that it requires much less calculation amount and shows less computational complexity. The distance between each pair of objects is computed only once, which contributes to the excellent feature. At the same time, the selection of the next cluster initial center can avoid the impact of noisy data on the clustering result. This proposed algorithm will be compared with GKM algorithm and its variation in the next section.

III. METHODOLOGY

In this section we discuss our proposed weighted fast global *k*-means algorithm (WFGKM). The WFGKM algorithm is proposed for dataset having streaming behavior. Considering a data set $X = x_1, x_2, x_3, \ldots, x_N$, the algorithm partitions X into M clusters $C = C_1, C_2, \ldots, C_M$ and find out each clusters center so that the cost function (objective function) of dissimilarity measure is minimization or below a certain threshold. The objective function is same as given in Equation1. The cluster centers are defined as $m_1, \ldots, m_M$. But for the streaming dataset, first we divide data stream into chunks $X_1, X_2, \ldots, X_s$ according to the reaching time of data, and the size of each chunk is determined by main memory of the processing system, let $n_1, n_2, \ldots, n_s$ be the data numbers of chunks $X_1, X_2, \ldots, X_s$ respectively . Due to its stream setting, a time weight $w(t)$ is imposed on each data representing the datum influence extent on the clustering process, and

$$\int_{f_0}^{f_c} w(t)dt = 1$$

Where t_0 is the initial time of stream and t_c is the current time

The main idea of WFGKM is renewing the weighted clustering centers by iterations till the cost function gets a satisfying result or the number of iteration is to a tolerance. Moreover, during the processing, we give the singleton a constant weight as 1. The procedure is presented as follow:

1) Import the chunk X_l ($1 \le l \le s$).
2) Update the weight of cluster centroids.

- If $l = 1$: Apply FGKM to gain cluster centroids v_i, $i = 1, \ldots, M$,and compute:

$$w_i' = \sum_{j=1}^{n_1} (u_{ij}) w_j, \qquad 1 \le i \le M$$

Where $w_j = 1, \forall \le j \le n_1$.

$$w_i' = \sum_{j=1}^{n_1+M} (u_{ij}) w_j, \qquad 1 \le i \le M$$

The centroid weight w_i then updates as $w_i = w'_i$

3) Update cluster centroids as updated in FGKM algorithm
4) Compute objective function:

$$E(m_1, \ldots\ldots\ldots, m_N) = \sum_{i=1}^{c+n_1} \sum_{k=1}^{M} I(x_i \in C_k) * w_k * ||x_i - m_k||^2$$

Stop if objective function is minimization or concentrate on a certain value, or its improvement over previous iteration is below a certain threshold, or iterations reach a certain tolerance value.

5) Compute a new U using Equation 4. Go to step 2.
6) If l = s then stop, else go to step 1.

U can be calculated as:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} (\frac{d_{ik}}{d_{jk}})^{2/(m'-1)}} \qquad (4)$$

Where u_{ik} is the membership value of the k^{th} data x_k in the i^{th} cluster.

As GKM and FGKM algorithms are on the total dataset, data stream may contain a very large data set, so allowing GKM or FGKM to deal with data stream directly may consume significant amounts of CPU time to converge, or result in an intolerable iteration quantity. Our proposed WFGKM reduces such complexities and take much lesser execution time and memory as compared to both GKM and FGKM.

IV. EXPERIMENTAL ANALYSIS

We implemented both FGKM and WFGKM in java to experiment the performance of the algorithms. Our experiment shows that WFGKM performs better than FGKM for dataset having streaming behavior. We use six datasets: car, iris, KDD- CUP, nursery, Ozone and spa-base. All the datasets are available in http://archive.ics.uci.edu/ml/datasets .html.

A. Execution Time

We consider the execution time for WFGKM by adding the execution of each data-segment. Since the dataset has streaming behavior there is no guarantee about the delay of arriving the data. Hence we ignore the data arrival delays from execution time. Table I, II and III shows the comparison of execution time for both FGKM and proposed WFGKM. We can see that WFGKM reduces execution time on an average of 65.53%. Figure 1 shows the same execution time comparison in terms of graph chart.

Clusters	Baseline	Proposed	Improvement (%)
8	1859	165	91.12
10	2262	324	85.67
12	2659	434	83.67
14	3101	654	78.91
16	3522	875	75.15
18	4072	1020	74.95
20	4361	1491	65.81

TABLE I: Execution time comparison on KDDCUP Dataset

Clusters	Baseline	Proposed	Improvement (%)
8	3450	127	96.31
10	4232	217	94.87
12	5017	357	92.88
14	5812	612	89.47
16	6608	790	88.04
18	7325	1080	85.25
20	8103	1550	80.87

TABLE II: Execution time comparison on LETTER Dataset

Clusters	Baseline	Proposed	Improvement (%)
8	1833	101	94.48
10	2255	179	92.06
12	2667	295	88.93
14	3059	437	85.71
16	3501	705	79.86
18	3944	924	76.57
20	4294	1346	68.365

TABLE III: Execution time comparison on Nursery Dataset.

Dynamic nature of execution time: It has been observed that both FGKM and WFGKM shows slight variance in execution time for multiple runs with same parameters. This is because of the assumption taken for fast global K-means. But the variations are not very high and can be tolerable. Still for accurate results we ran each algorithm multiple times (with same data-set and parameters) and taken average of them as final execution time.

A. Memory Used

Since WFGKM process data as number of chunks we calculated the memory consumption of each chunk separately and take the largest value as the final memory consumption for sWFCM-HD. Figure 4 shows the percentage of improvement in terms of memory consumption by proposed (WFGKM) as compared to the baseline algorithm. The improvement is more than 68% for all three datasets. Baseline algorithm (FGKM) uses entire dataset at a time and hence it requires enough memory to hold the entire dataset. This is the reason why baseline requires much higher memory than our proposed algorithm.

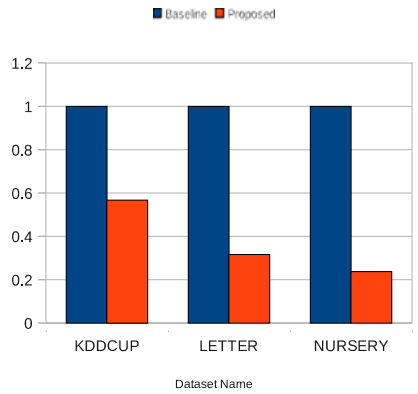
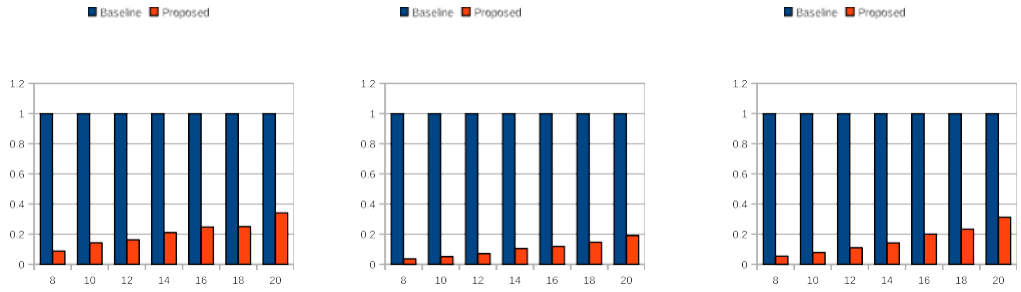


Fig. 4: Comparison of memory consumption in proposed WFGKM over baseline (FGKM).

Memory issues: The proposed WFGKM has a member ship matrix which is not there in FGKM. When the number of segments in WFGKM is less the memory required by the membership matrix is higher and consume the benefit gained from WFGKM. But as the number of segments increase the benefit can be observed. In our experiments we assumed the number of segments as 50; which is normal for todays streaming data-sets.

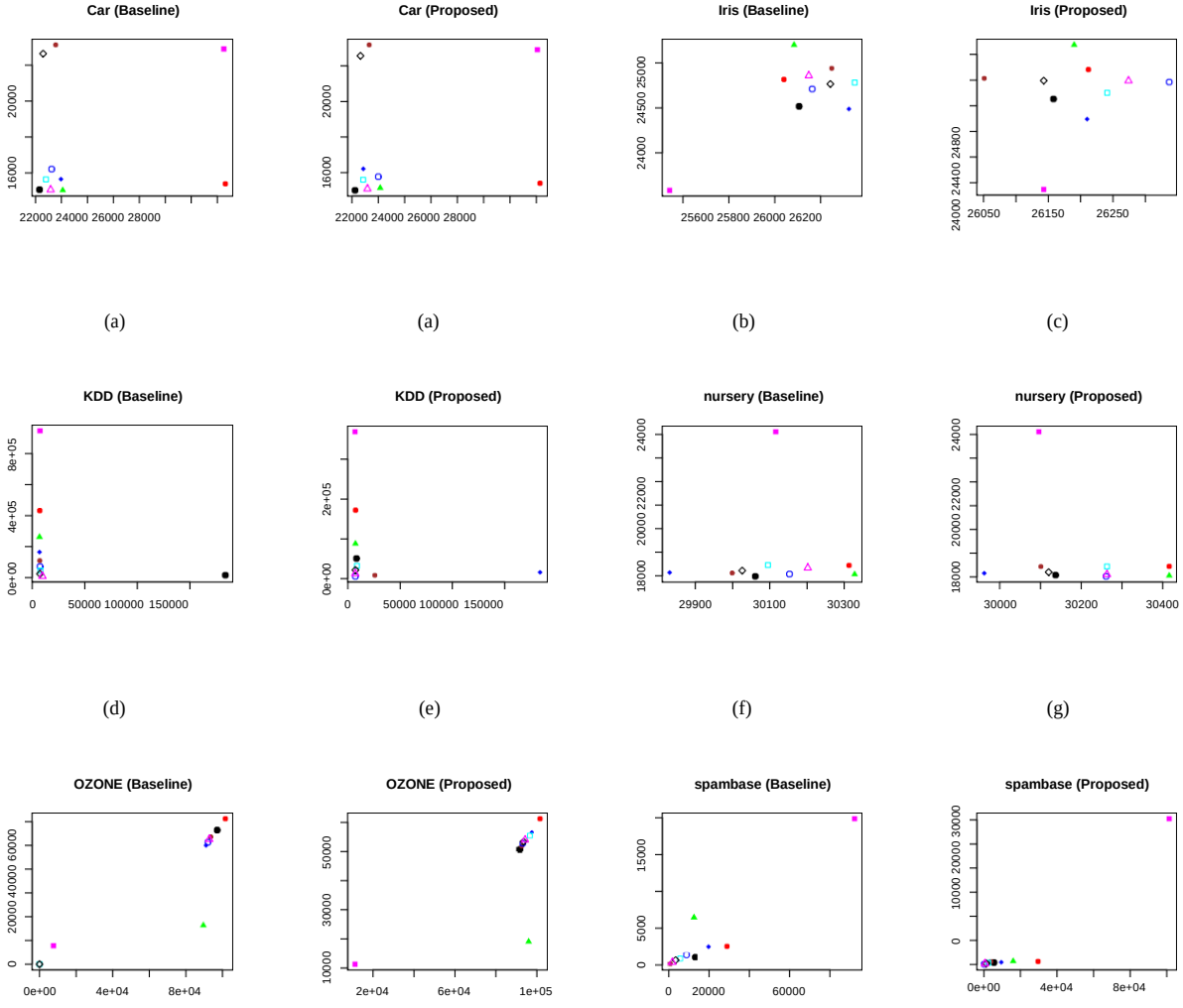
Cluster analysis

This section describes the cluster based analysis of our proposed algorithm. Different datasets has been used in this section for analysis. Each dataset is executed for 10 clusters and the results are shown for both the baseline as well as for the proposed algorithm.



(a) KDD Cup 1999 Data Set. (b) Letter Recognition Data Set. (c) Nursery Data Set.

Fig. 1: Execution time comparison of proposed WGFKM over baseline (FGKM).



(a) (a) (b) (c)

(d) (e) (f) (g)

(h) (i) (j) (k)

Fig. 2: Cluster-center diagram of different datasets for both baseline (FGKM) and proposed (WFGKM) algorithm. Each diagram has a heading mentioning its dataset name and the algorithm (baseline or proposed).

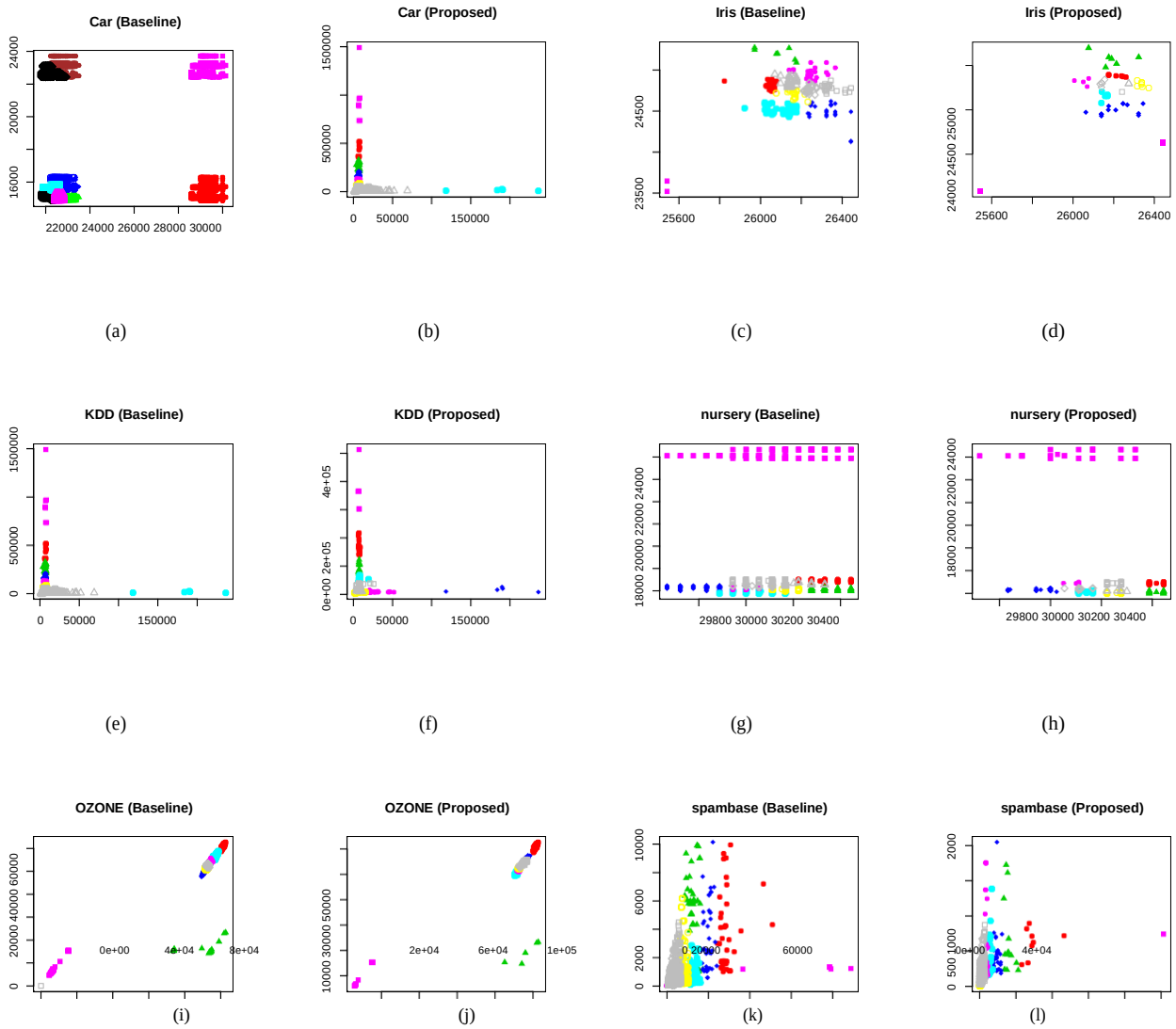


Fig. 3: Cluster diagram of different datasets for both baseline (FGKM) and proposed (WFGKM) algorithm. Each cluster diagram has a heading mentioning its dataset name and the algorithm (baseline or proposed).

Figure 2 shows the cluster-centers of each dataset separately. The graphs of both baseline and proposed for each dataset are given one after another and can be recognized by the heading given in each graph. From the figure it can be observed that the cluster-centers are almost same in both baseline and proposed algorithm. Hence our proposed algorithm gives the huge improvement in execution time and memory used (as discussed in Section IV-A and Section IV-B) without any major clustering differences. The cluster-diagram for each dataset (both for baseline and proposed) is given in Figure 3. Note that the proposed algorithm divides the dataset into multiple segments and hence the cluster size of baseline and proposed are not same. But their patters are almost looks similar (except car).

From the above analysis it can concluded that the proposed clustering algorithm reduces the execution time and memory consumption without any major clustering variations.

V. CONCLUSION AND FUTURE SCOPE

The *k*-means algorithm and its variations are known to be fast clustering algorithms. However, they are sensitive to the choice of starting points and are inefficient for solving clustering problems in large datasets. Recently, incremental approaches have been developed to resolve difficulties with the choice of starting points. The global *k*-means and the fast global *k*-means algorithms are based on such an approach. They iteratively add one cluster center at a time. Numerical experiments show that these algorithms considerably improve the k-means algorithm. However, they require storing the whole affinity matrix or computing this matrix at each iteration. This makes both algorithms time consuming and

memory demanding for clustering even moderately large datasets. Also the continuously arriving data stream has become common phenomenon for many fields recent years; for example, sensor networks, web click stream and internet traffic flow. Re- searchers proposes many innovative technologies to manage such streaming datasets. Finding efficient data stream mining algorithm has become an important research subject. In this paper we propose a fast global *k*-means algorithm for datasets having streaming behaviour. Experiment shows that our pro- posed algorithm is more efficient than the fast global *k*-means algorithm in case of streaming datasets.

REFERENCES

[1] Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS '02, 2002, pp. 1–16.

[2] L. Bai, J. Liang, C. Sui, and C. Dang, "Fast global k-means clustering based on local geometrical information," *Information Sciences*, vol. 245, no. 0, pp. 168 – 180, 2013.

[3] I. Fodor. (2002) A Survey of Dimen sion Reduction Techniques. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.5098

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *SCIENCE*, vol. 290, pp. 2323–2326, 2000.

[5] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[6] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1:1–1:58, Mar. 2009.

[7] A. Jain and R. Dubes, Eds., *Algorithms for Clustering Data*. Prentice Hall, 1988.

[8] A. Likas, M. Vlassis, and J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 35, no. 2, pp. 451–461, 2003.

[9] A. Bagirov, "Modified global k-means algorithm for sum-of-squares clustering problem," *Pattern Recognition*, vol. 41, pp. 3192–3199, 2008.

[10] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Data Engi- neering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 685–694.

[11] P. Domingos and G. Hulten, "A general method for scaling up machine learning algorithms and its application to clustering," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, 2001, pp. 106–113.

[12] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clus tering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases - Volume 29*, ser. VLDB '03, 2003, pp. 81-92

[13] R. Wan, X. Yan, and X. Su, "A weighted fuzzy clustering algorithm for data stream," in *Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management- Volume 01*, ser. CCCM '08, 2008, pp. 360–364.

[14] L. Bai, J. Liang, C. Sui, and C. Dang, "Fast global k-means clustering based on local geometrical information," *Information Sciences*, vol. 245, no. 0, pp. 168 – 180, 2013.

[15] J. Xie and S. Jiang, "A simple and fast algorithm for global k-means clustering," in *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, vol. 2, March 2010, pp. 36– 40.

[16] L.-E. Sal, J. Carrasco-Ochoa, and M.-T. Fco, "Fast global k-means with similarity functions algorithm," in *Intelligent Data Engineering and Automated Learning IDEAL 2006*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4224, pp. 512–521.

Authors Profile

Dr. Purnendu Das is an assistant professor of the Department of Computer Science, Assam University, Silchar. He has pursued Ph.D. degree from Tripura University. He has published researched papers in many reputed journals.

Bishwa Ranjan Roy is an assistant professor of the Department of Computer Science, Assam University, Silchar. He has pursued M.Tech. degree from NIT Silchar. He has published researched papers in many reputed journals.

Dr. Sanju Das is a technical assistant of the Department of Computer Science, Assam University, Silchar. He has pursued Ph.D. degree from Assam University. He has published researched papers in many reputed journals.

