

Implementing K-means Algorithm using Row store and Column store databases: A case study

Suresh.L¹, Jay.B.Simha², and Rajappa Velur³

Professor, Cambridge Institute of Technology, CSE Dept, Bangalore, India

Email: suriakls@gmail.com

² Chief Technology Officer, Abiba systems, Bangalore, India

Email: jbsimha@abibasystems.com

³ Dean, Academics, Cambridge Institute of Technology, Bangalore, India

Email: rajuvelur@rediffmail.com

Abstract – K-means Clustering is an important algorithm for identifying the structure in data. K-means is the simplest clustering algorithm [8]. This algorithm uses as input a predefined number of clusters i.e., the K from its name. Mean stands for an average, an average location of all the members of a particular cluster. In this work, a novel approach to seeding the clusters with a latent data structure is proposed. This is expected to minimize the need for number of clusters apriory, Time for convergence by providing near optimal cluster centers. Also these algorithms are tested on the latest standards for data warehouse – the column store databases.

Index Terms—Clustering, K-means, SQL, Median Projection, Median selection

I. INTRODUCTION

Clustering is one of the important techniques of data mining [6, 13]. Clustering becomes an indispensable requirement while dealing with immense volume of data. Clustering algorithms [2,1] partition a dataset into several groups such that points in the same group are close (similar) to each other and points across groups are far (different) from each other [7, 13].

K-means algorithm [8] is chosen for clustering large datasets.

II. LITERATURE SURVEY

K-means Algorithm [8, 11] is a popular and most widely used clustering algorithm. It is important algorithm for identifying the structure in data. K-means is the simplest clustering algorithm. This algorithm uses as input a predefined number of clusters i.e., K from its name. Mean stands for an average, an average location of all the members of a particular cluster.

Bottou and Bengio [12] have proposed convergence properties of K-means algorithm. The original K-means algorithm works with many resident data, but disk resident options are also available.

Ordenez[3] introduced three SQL implementations of the popular K-means clustering algorithm to cluster large datasets and integrate it with a RDBMS. 1) a straightforward translation of K-means computations into SQL, 2) an optimized version based on improved data organization, efficient indexing, sufficient statistics, and

rewritten queries and 3) an incremental version that uses the optimized version as a building block with fast convergence and automated reseeding [3].

However, it is observed in the literature that very little work has been done to seed the initial cluster centers as well as setting the number of clusters using SQL based algorithms. These two steps considerably influence the quality of clusters.

Proposed work:

In this research an attempt has been made to develop algorithm in SQL for

- Identifying appropriate number of clusters
- Seeding initial cluster centers.
- K-means clustering with pre-seeded cluster means.

Implementing computationally intensive algorithms on databases are severely restricted due to the data handling mechanism implemented in the DBMS. Currently there are different versions of DBMS/DBMS like systems for handling large datasets. Most of the DBMS have been designed to work with the relational constraints keeping in mind the requirement of OLTP systems [5, 10, 14].

Most of the systems are implemented with row stored data structure where every tuple of a table will be stored as variable length row in a file. Even though the actual implementation varies across the vendors the general scheme will facilitate relational algebra. These row stored RDBMS are available in both memory model like SQLite [5] and MonetDB[14] or disk based architecture like MYSQL [10] and Postgre.

However the limitation of holding larger than memory datasets requiring complex page swapping mechanism leading to unavailability of these systems for expensive analytical functions.

The latest researches in handling large datasets with analytical functions have been shown to dramatically improve the performance of analytical queries. These systems use column stored architecture where data will be stored in column fashion than a row fashion. Both main memory [14] and disk based [17] versions are available.

The SQL implementation of K-means algorithm has been tested on row stored databases and found to perform exceptionally well on large datasets [3]. However no work has been done in evaluating performance of K-means algorithm with column stored database on large datasets. Therefore in this research an

empirical evaluation of the proposed extended K-means algorithm has been carried out.

The results indicate superiority of column store architectures in some areas of computation while row oriented architectures score in some area.

III. EXPERIMENTATION

A. Research Work

In this research work an attempt has been made to implement a novel approach for identifying the number of clusters and seeding the K-means algorithm. Further the modified K-means algorithm has been implemented in SQL and the performance of SQL code has been tested on row store and column store DBMS.

B. SQL Implementation:

The SQL implementation in SQL having D data points and C cluster centers has been solved in [4] and improved version has been implemented in this research. The basic framework is:

- 1) Create, index and populate tables.
- 2) Initialize the cluster center table with C points randomly selected from dataset D
- 3) Compute repeatedly till $|KM(X,C)^i - |KM(X,C)^{i-1}| < \epsilon$ where ϵ is user defined threshold for the error.

C. Strategy for Implementation:

The process starts with selection of cluster centers from the data table (Y) into table C. Next the distance of each of the data points (records) in data table (Y) with each of the cluster center from table C, are inserted into the distances table (YD).

Next the records are assigned to the clusters based on minimum distance and attached with a class label of the cluster they belong to in table MD. Subsequently the new cluster centers are computed by joining tables C2 and Y and inserted into table CN. If the CN and C are different (based on chosen criteria), then the process is repeated till convergence. Figure 1 shows process flow diagram. The data structures used in the implementation are shown in table 1.

D. Novel Seeding algorithm:

In order to mitigate this limitation a novel seeding algorithm based on median projection is proposed. The basic idea of our algorithm is to identify the median gaps in the data in each of the columns and then projecting these median gaps to the other columns.

Intuitively it is evident in a given scalar set D (database column) the presence of large gap will lead to existence of distinct clusters.

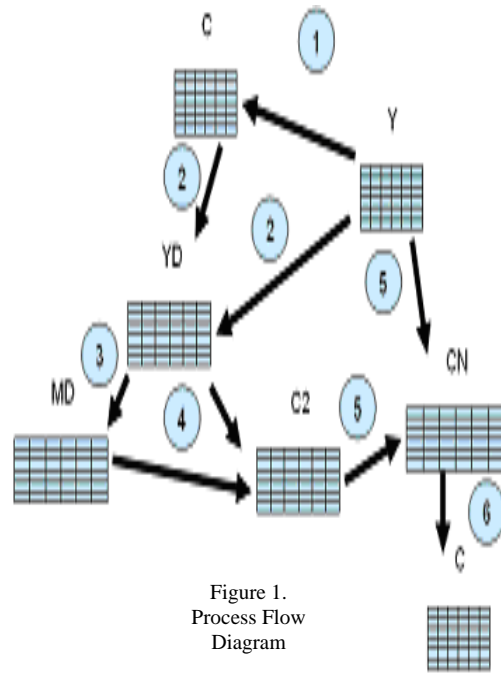


Figure 1. Process Flow Diagram

TABLE 1: DATA STRUCTURE USED

Table	Primary Key	Columns	No. of rows	Contents
Y	RID	y1, y2, ..., yp	N	Records
C	CID	y1, y2, ..., yp	K	Cluster centers
YD	RID	d1, d2, ..., dp	KN	Distances
MD	RID	D	N	Minimum
C2	None	cid, rid	N	Classified
CN	CID	y1, y2, ..., yp	K	New cluster

This property of data is exploited in our approach to find the number of clusters and provide initial seed clusters. The algorithm actually consists of 2 procedures namely Median Search and median projection which are briefly discussed below.

E. Median Projection Algorithm

- For each of the column mark the medians as given in median search algorithm.
- Select the column with least number of medians to the left and most number of medians to the right.
- Use an agglomerative technique to merge the cluster centers till optimal clusters are achieved.
- Compute new cluster centers and provide them as seeds of the K-means.

F. Median Search Algorithm

- Select and sort the data in a column.
- Compute the difference between two successive points.
- Rearrange the data points based on the difference

value.

- Select top 5% of the difference value and set them as medians.
- Increase sample size to 6% and verify the cluster centers using the selected medians. If the difference between 2 successive iterations of identifying the cluster centers is statistically insignificant or we have reached 10% of the points for selecting the medians then stop projection.

G. Data

Data for the exploitation has been developed using random number generator with 3 well defined clusters. The datasets contains 6 columns. Two different approaches to dataset creation has been developed.

1) In the first approach, the numbers of columns are fixed and only different number of records of the order 10K, 100K and 1000K with 3 distinct clusters are generated. This set is used to verify

a) Seeding of the cluster centers.

b) Vertical scalability of the algorithm implemented in SQL.

2) In the Second approach, the number of records has been kept constant at 10K and only numbers of columns are varied. This is used to validate the proposed algorithm on horizontal scalability.

H. Validation

The proposed implementation is expected to perform linearly on increase of number of records and number of columns and number of clusters. To verify this conjecture, we have evaluated the run time of the proposed implementation for different number of clusters, different data sizes and different number of columns. In order to verify quality of cluster formed inter-cluster distance has been analyzed. Further the run times for different data sizes on standard K-means and modified K-means will be analyzed.

IV. RESULTS AND DISCUSSION

It can be observed from table 2 and 3 that the runtime of the algorithm is linear with respect to the datasets. The inter cluster distances which measure the quality of clusters also seems to be improved. The proposed algorithm provides initial cluster centers through median projection, appears to improve the performance significantly.

The average run time of the proposed algorithm is 30-40% less than the K-means algorithm implemented without seeding the initial cluster centers. This can be attributed to better convergence with minimum number of passes over datasets. Further median projection algorithm seems to work on the synthetic datasets used in the research.

Another interesting observation in figure 5 and 6 is non-availability of the data for the proposed algorithm in certain parameter settings. This is due to the fact that the proposed median projection algorithm clearly identifies the latent structure in the dataset. On the other hand the number of clusters can be forced on standard K-means even when the numbers of clusters are limited.

These results are encouraging and the proposed algorithm meets all the criteria set initially during research namely- scalability, quality of cluster and true unsupervised learning. The results of the experiments on the synthetic datasets using SQL K-means algorithm seems to be promising. It can be seen that SQL K-means scales linearly on all the three dimensions – number of attributes, number of records and number of clusters.

TABLE 2. K-MEANS

Data size	Number of Clusters	Inter Cluster distance	Run time
10K	2	124	14
	3	110	22
	4	102	30
	5	92	37
100K	2	118	142
	3	106	225
	4	101	306
	5	91	378
1000K	2	135	1430
	3	114	2300
	4	104	3100
	5	96	3760

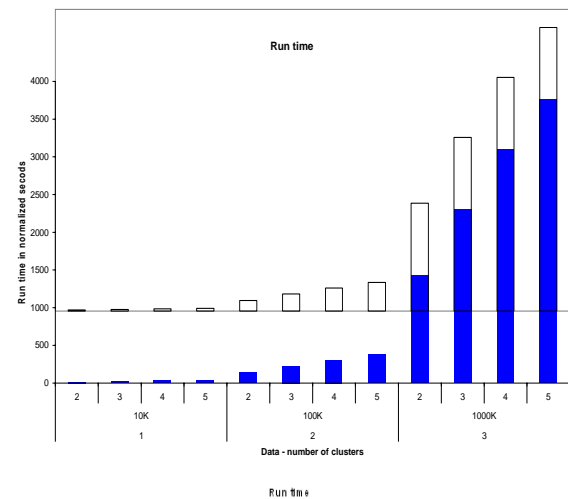


Figure 2. Runtime for K-means

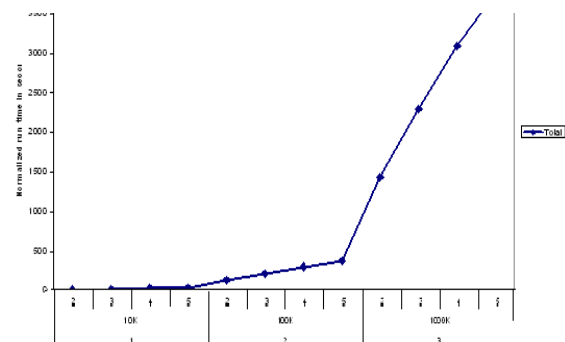


Figure 3. Runtime for K-means

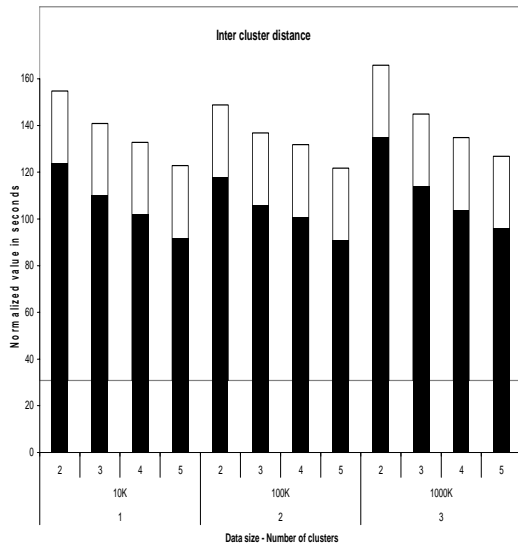


Figure 4. Inter cluster distance for K-means

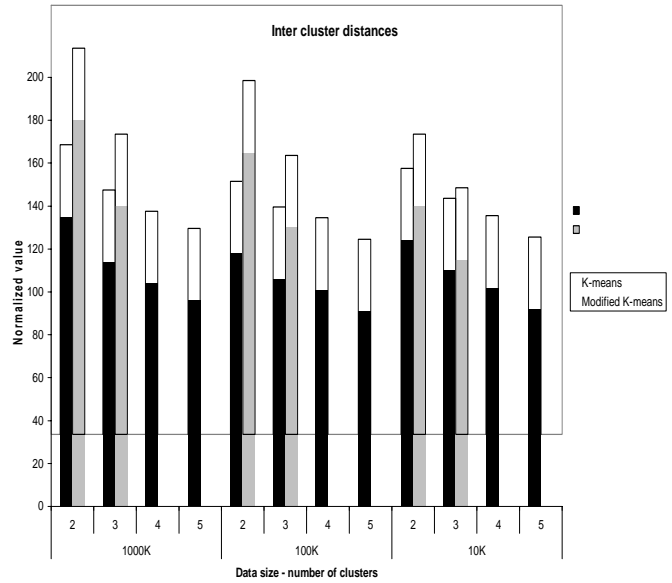


Figure 6. comparison of inter cluster distance of K-means with modified K-means

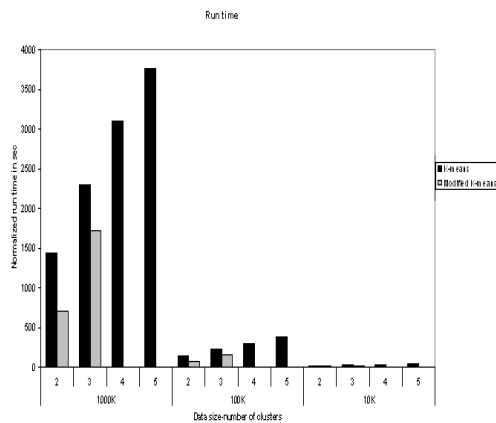


Figure 5. Comparisons of K-means with Modified K-means

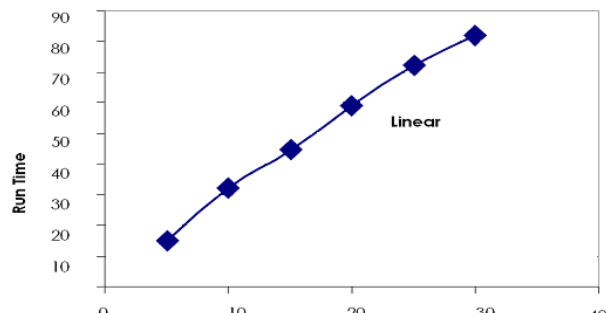


Figure 7. Runtime for K-means with varying number of columns

Table 3. Modified K-Means

Data size	Number of Clusters	K-means	Modified K-means
1000K	2	135	180
	3	114	140
	4	104	
	5	96	
100K	2	118	165
	3	106	130
	4	101	
100K	5	91	
	2	118	165
	3	106	130
10K	4	101	
	5	91	
	2	124	140
	3	110	115
	4	102	
	5	92	

From figure 7, it is observed that SQL based clustering algorithm scales linearly with number of columns. This reconfirms the observation made in Ordonez [7]. From figure 8, it can be observed that Column store representations take less time compare to Row-store databases.

Data size	Number of clusters	Row store		Column store	
		SQLite	MySQL	MonetDB	LucidDB
10K	2	13	14	14	14
	3	20	22	21	22
	4	30	30	29	32
	5	33	37	35	36
100K	2	135	142	133	136
	3	227	225	220	220
	4	300	306	304	305
	5	379	378	365	370
1000K	2	1500	1430	1440	1435
	3	2390	2300	2310	2300
	4	3200	3100	3080	3084
	5	3700	3760	3600	3655

Figure 8. Results of running SQL clustering on different databases

V. CONCLUSIONS

The proposed implementations allow clustering of large datasets stored inside a relational DBMS, eliminating the need to export data. We concentrated on defining suitable tables, indexing them and optimizing queries for clustering.

In this research an attempt has been made to develop a K-means algorithm to specify the number of clusters a priori so that convergence can be achieved fast. Our novel approach exploits the latent structure in the data by partitioning the columns along medians. Subsequently these medians are projected across other columns to get the number of clusters and the initial cluster centre.

Once the seeds are identified standard K-means algorithm has been implemented using suitable tables in discs and optimized queries. The results of the proposed approach are promising and the algorithm implemented in SQL scales linearly in all dimensions tested. This is expected to provide scalability for large datasets and reduction in pre-processing overheads for in-database mining.

The results of implementing the proposed approach on both row store and column store DBMS have shown mixed results. The aggregations on column store were efficient compared to row store and the difference calculations in a row store DBMS outperformed that on column store DBMS. Additional optimization on column store DBMS is required to improve its overall effectiveness.

REFERENCES

[1] Anjan Goswami, Ruoming Jin, Gagan Agrawal, "Fast and Exact Out-of-Core K-Means Clustering". Proceedings of the Fourth IEEE International Conference on Data Mining, Pages: 83-90, 2004.
 [2] Bradley.B.S, Usama Fayyad and Cory Reina, "Scaling clustering algorithms to large databases". Proceedings of

the Fourth International Conference on Knowledge Discovery and Data Mining, Pages 9-15, August 1998.

[3] Carlos Ordonez, "Programming the K-means Clustering Algorithm in SQL", 2002.

[4] Carlos Ordonez, "Integrating K-means clustering with Relational DBMS using SQL". IEEE transactions on Knowledge and Data engineering, Vol. 18, no. 2, February 2006.

[5] Chris Newman, "SQLite: A Practical guide for using, administering and programming Database bundled with PHP5".

[6] Cooley.R and Srivastava.J, "Data Preparation for Mining". Journal of Knowledge and Information Systems, Vol.1, No.1, 1999.

[7] Dietterich.G, "Machine Learning".

[8] Hartigan J.A and Wong M A, "A K-means clustering algorithm". Applied Statistics, 28(1):100-108, 2001.

[9] Herbert A. Edelstein, "Introduction to Data Mining and Knowledge Discovery". 3rd edition, Published in two crow's corporation, 1999.

[10] Imielinski.T and Virmani.A, "MSQL: A Query Language for Database Mining". Data Mining and Knowledge Discovery, vol. 3, no. 4, Pages 373-408, 1999.

[11] Khaled Alsabti, Sanjay Ranka and Vineet Singh, "An Efficient K-means Clustering Algorithm".

[12] Leon Bottou and Yoshua Bengio. "Convergence properties of the K-means algorithms". Advances in Neural Information Processing Systems, Vol 7, Pages 585-592. The MIT Press, 1995.

[13] Lorinda Visnik, "Clustering Techniques". A technical White paper, pages 2-8, 2002.

[14] Maarten Vermeij1, Wilko Quak1, Martin Kersten2, Niels Nes2 1 TUDelft, OTB, "MonetDB, a novel spatial column-store DBMS". Section GIS-technology.

[15] Maimon.O and Rokesh.L (Eds.), "Data mining and knowledge discovery handbook". Springer, 2005, Pages. 399-416.

[16] Murthy.M.N, Jain A.K and Flynn P.J, "Data Clustering: A Review". ACM Computing Surveys, Vol. 31, No. 3, September 1999.

[17] Tapas kanungo, Natan S, Natanyahu, Angela Y. Wu. "An efficient K-Means Clustering Algorithm: Analysis an Implementation". IEEE Transactions on Knowledge and Data Engineering, Vol. 13, NO.3, May/June 2001.

[18] Zhang.T, R. Ramakrishnan, and Livny.M, "BIRCH: An Efficient Data Clustering Method for Very Large Databases". Proceedings of the ACM SIGMOD Conference, Pages 103-114, 1996.