A STUDY ON CLUSTERING LARGE DATASETS USING IN-DATABASE APPROACHES

A THESIS

Submitted by

SURESH L

for the award of the degree

of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
UNIVERSITY
(Declared U/S 3 of the UGC Act, 1956)
CHENNAI 600 095
JANUARY 2010
BONAFIDE CERTIFICATE

Certified that, this thesis titled “A STUDY ON CLUSTERING LARGE DATASETS USING IN-DATABASE APPROACHES” is the bonafide work of Mr.SURESH.L, who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. Jay B.Simha
Research Supervisor

Dr. Bharatheeesh Jayasimha
Chief Technology Officer
Abica Systems Pvt. Ltd.
17/12-30, 1st Floor, 30th Main,
BSK 3rd Stage, Bangalore-560 085.
DECLARATION

I declare that thesis entitled “A STUDY ON CLUSTERING LARGE DATASETS USING IN-DATABASE APPROACHES”, submitted by me for the degree of Doctor of Philosophy in Computer Science and Engineering in the record of work carried by me during the period from June 2004 to Jan 2010 under the guidance of Dr. JAY B.SIMHA, and has not formed basis for the award of any degree, diploma, associate ship, fellowship, titles in this or any other university or similar institution of higher learning.

Signature of the candidate

SUresh L
ABSTRACT

Data mining can be comprehended as a process of extraction of knowledge hidden in extremely large datasets. Data mining techniques can significantly boost the ability to analyze the data.

Clustering is one of the important techniques in data mining. Clustering becomes an indispensable requirement while dealing with immense volume of data. Clustering algorithms partition a dataset into several groups such that the intra cluster similarity is maximized and the inter cluster similarity is minimized. Existing clustering algorithms, such as K-means, PAM, CLARANS, DBSCAN, CURE, and ROCK are designed to find clusters that fit some static models. These algorithms can breakdown if the choice of parameters in the static model is incorrect with respect to the data set being clustered, or if the model is not adequate to capture the characteristics of clusters.

K-means Clustering is an important algorithm for identifying the structure in data. K-means is the simplest clustering algorithm. This algorithm is based on distance minimization between assumed cluster center and data points assigned to the cluster. This algorithm takes a predefined number of clusters and
an acceptable threshold of errors between ideal convergences to actual convergence as input. Then a set of records equal to predefined numbers of clusters are chosen randomly and an iterative process towards converging the cluster centers will be carried out till an acceptable error rate defined by the threshold is achieved. Though simple in nature, K-means algorithm has some limitations.

1) When the clustering is used as an unsupervised learning algorithm for data mining, it is difficult to provide the number of clusters apriori as the structure is not known.

2) Since the records are chosen randomly the order of the records determine the quality of the clusters, which requires repetitive runs to get a better set of cluster centers. This increases the search space and time.

3) Since the records are chosen randomly the time for convergence for a solution with acceptable error rate is non-deterministic and can be quite high.

In addition most of the implementations of the standard K-means algorithm are based on data structures which are assumed to fit into memory. Working on larger datasets requires sampling or data reduction in some form which may lead to information loss if not properly tuned.

In order to minimize some of the limitations of the K-means algorithm, in this research a novel approach for finding the cluster centers has been proposed. Subsequently the K-means algorithm has been implemented in a declarative language provided by the DBMS to provide an ability to handle massive datasets that cannot fit into memory.
The novel seeding algorithm is based on an assumption of potential possibility of existence of gaps in an ordered sequence of an attribute value set. Once these gaps are identified and statistically validated for sufficient differentiation they will be projected across other attributes which may not have such kind of properties. The empirical experimentation on synthetic and real world datasets indicate this approach can substantially improve the quality of clusters by means of both identifying the structure of the clusters in the data and providing initial cluster centers which are near optimal. Since the proposed algorithm works on the basis of sorting the data it can be easily implemented in a declarative language like SQL.

SQL implementation of K-means algorithm has been tested on row stored database and found to perform exceptionally well on large datasets. However no work has been done in evaluating performance of K-means algorithm with column stored database on large datasets. In this research an empirical evolution 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.

However one of the limitations of the current research is that the implementation has been tested only on numerical datasets which lend themselves well for ordered sequencing or sorting. Use of non-numeric attribute is handled through representation of categorical attributes as numeric attributes through flattening.
THESIS ORGANIZATION

This thesis has eight chapters and each chapter is further divided into different sections. This abstract describes each chapter briefly.

CHAPTER 1

In section 1.1 KDD process as accepted by the practitioners and academic research groups are discussed. In section 1.2 the open issues of the KDD process are discussed. In section 1.3 open issues in Clustering process for large datasets in terms of runtime or quality of results are discussed and in the last section 1.4, the scalable algorithms for Clustering are discussed.

CHAPTER 2

Chapter 2 discusses background and review of literature. It focuses on Clustering Techniques, Partitioning and the role of Clustering techniques in Data mining. Historical and recent developments in the existing field of clustering in general and K-means in particular are reviewed.
CHAPTER 3

This chapter focuses on the objective of the present research work carried out. This chapter mainly focuses on the proposed work. Initially the limitations of K-means are discussed. Later the focus is on the pros and cons of implementing K-means using SQL are discussed. Finally the focus is on the proposed work.

CHAPTER 4

This chapter describes focuses on the K-means algorithm implementation strategies, selection of random centers, Complexity analysis of K-means algorithm, frame work to build the algorithms and the parameters used to specify the distance between data points in the clustering are measured. Implementation of K-means using SQL is discussed with an example.

CHAPTER 5

In this chapter a seeding algorithm to identify the number of clusters and the initial seeds will be discussed. Implementation of Median Search and Median projection using SQL are discussed with an example.

CHAPTER 6

Results of the experimental work done and its analysis are explained in this chapter. We have evaluated the run time of the proposed implementation for different number of clusters, different data sizes and different number of columns. Further the run times for different data sizes on standard K-means and modified K-means are analyzed. Comparison of inter cluster distance of K-means with modified K-means are discussed.
CHAPTER 7

In this chapter Case study on Segmentation and profiling churn behavior in Telecom sector is analyzed and discussed. A case study of the proposed framework in churn analysis and modeling in telecom has been carried out to test the suitability of framework for industrial applications. Application of the framework has shown promising results.

CHAPTER 8

In Chapter 8, the thesis concludes with a summary of the major contributions, critical evaluation and discussions on scope for future work.
ACKNOWLEDGEMENT

I am truly grateful to my mentor, Dr Jay B.Simha, Chief Technology Officer, Abiba Systems, Bangalore for his intellectual support and persistent motivation that have guided me throughout my graduate career.

I owe a special gratitude to Dr. Rajappa Velur, Dean Academics, CITech, Bangalore for his deep concern, perpetual encouragement, excellent guidance, kind co-operation and support during my research work.

I also owe my sincere thanks to Dr.Iyengar S.S Chairman Computer Science Department Louisiana State University USA, Dr.K.Chidananda Gowda Vice chancellor Kuvempu University, Dr. P. Nagabhushan University of Mysore Mysore, Dr.Jayaram Dean Computer science SBJCE, Dr. Balasubramanya murthy Principal PESIT, Dr.Arun Biradar HOD KSIT, Dr.Rajendra Hegde HOD RGIT and Dr.Ravishankar B.V BMSCE for their help and critical comments.

Particularly I am indebted to Dr.P.Aravindan Dean-Research, Dr.CyrilRaj HOD CSE Dept, Dr.S.Sendilvelan Dean Engineering and Technology and Mr.Anand co-ordinator, Dr.MGR Educational and Research Institute, University for providing a friendly atmosphere.

My special thanks to Sri.D.K.Mohan Chairman and Dr. H.Maheshappa Principal, CITech for their support, in attending International Conferences and completing the work in time.
My Sincere thanks to Prof. Sashikumar D.R, Dr.Rudresh kumar, Mr.Guruprasad.S and to all my colleagues and friends in CITech/PATE team for their co-operation and support.

Finally, I would like to express my appreciation to all the family members who has given moral support to carry out this research work.

Suresh.L

CONTENTS

LIST OF FIGURES…………………………………………………………….. i
LIST OF ALGORITHMS…………………………………………………… iii
LIST OF TABLES…………………………………………………………… iv
LIST OF SYMBOLS……………………………………………………… v
LIST OF ABBREVIATIONS…………………………………………… vi
THESIS ABSTRACT……………………………………………………….. vii

1 INTRODUCTION…………………………………………………… 1
   1.1 Introduction TO KDD Process………………………………… 1
   1.2 Open Issues in KDD Process………………………………… 3
   1.3 Open Issues in Clustering………………………………… 6
       1.3.1 Feature Selection or extraction……………………… 10
       1.3.2 Clustering Algorithm Design/Selection…………… 10
       1.3.3 Cluster Validation………………………………… 11
   1.4 Scalable algorithms for Clustering …………………… 12

2 LITERATURE SURVEY…………………………………………… 15
   2.1 Setting the scene ……………………………………………… 15
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Classification of Clustering Algorithms</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Hierarchical Method</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Density Based Method</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Grid Based Methods</td>
<td>19</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Model Based Method</td>
<td>20</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Neural Network Method</td>
<td>20</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Partitioning Methods</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>OBJECTIVE OF THE PRESENT WORK</td>
<td>34</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Objectives</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Proposed Work</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>IN-DATABASE CLUSTERING USING SQL</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>37</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Distance measures</td>
<td>38</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Selection of Initial means</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Cluster Validity Measures</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Frame work</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>K-means Clustering</td>
<td>49</td>
</tr>
<tr>
<td>4.4.1</td>
<td>K-means Method</td>
<td>49</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Complexity Analysis of K-means</td>
<td>56</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Remarks</td>
<td>59</td>
</tr>
<tr>
<td>4.5</td>
<td>K-means Using SQL</td>
<td>60</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Introduction</td>
<td>60</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Data Structures used</td>
<td>61</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Implementation Strategy</td>
<td>62</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIG. NO.</th>
<th>TITLE OF FIGURE</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>KDD Process</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Multiple isolated KDD Instances of the same data</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Procedure of Cluster analysis</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Clustering of a Set of points based on K-means Method</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>K initial Means</td>
<td>49</td>
</tr>
<tr>
<td>4.2</td>
<td>K clusters are created by associating every observation with the nearest mean</td>
<td>50</td>
</tr>
<tr>
<td>4.3</td>
<td>The centroid of each of the K clusters becomes the new means</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Repetition till convergence</td>
<td>51</td>
</tr>
<tr>
<td>4.5</td>
<td>K-means Process</td>
<td>53</td>
</tr>
<tr>
<td>4.6</td>
<td>Process flow diagram</td>
<td>63</td>
</tr>
<tr>
<td>4.7</td>
<td>Centroid re-computation</td>
<td>68</td>
</tr>
<tr>
<td>4.8</td>
<td>Stopping Criteria</td>
<td>69</td>
</tr>
<tr>
<td>6.1</td>
<td>DBMS used in the Research</td>
<td>100</td>
</tr>
<tr>
<td>6.2</td>
<td>Runtime for K-means</td>
<td>102</td>
</tr>
<tr>
<td>6.3</td>
<td>Inter cluster distance for K-means</td>
<td>104</td>
</tr>
<tr>
<td>6.4</td>
<td>Comparison of Inter cluster distance of K-means with Modified K-means</td>
<td>106</td>
</tr>
<tr>
<td>FIG. NO</td>
<td>TITLE OF FIGURE</td>
<td>PAGE NO.</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>6.5</td>
<td>Runtime Comparison of K-means with Modified K-means.</td>
<td>108</td>
</tr>
<tr>
<td>6.6</td>
<td>Runtime for K-means with varying Number of Columns….</td>
<td>109</td>
</tr>
<tr>
<td>7.1</td>
<td>Architecture of the Proposed system..................................................</td>
<td>116</td>
</tr>
<tr>
<td>7.2</td>
<td>Typical Patterns in Time series trends.............................................</td>
<td>118</td>
</tr>
<tr>
<td>7.3</td>
<td>Typical Times series comparison......................................................</td>
<td>120</td>
</tr>
<tr>
<td>7.4</td>
<td>Prediction with a Probabilistic Projection model.................................</td>
<td>123</td>
</tr>
<tr>
<td>7.5</td>
<td>Sample churn in different regions...................................................</td>
<td>125</td>
</tr>
<tr>
<td>7.6</td>
<td>Projections for contract type with actual values.................................</td>
<td>126</td>
</tr>
<tr>
<td>7.7</td>
<td>Projections for contract type by region with a actual values.</td>
<td>127</td>
</tr>
<tr>
<td>7.8</td>
<td>Projections for contract type by region with Product type…</td>
<td>127</td>
</tr>
</tbody>
</table>
# LIST OF ALGORITHMS

<table>
<thead>
<tr>
<th>ALG. NO</th>
<th>TITLE OF ALGORITHM</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>K-medoids Algorithm</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>The K–means Algorithm for partitioning based on the mean value of the objects in the cluster</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Steps of Classical K-means Algorithm</td>
<td>52</td>
</tr>
<tr>
<td>4.2</td>
<td>Pseudo code for K-means</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>Implementation strategy for SQL K-means</td>
<td>63</td>
</tr>
<tr>
<td>4.4</td>
<td>Selecting Random Centers</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td>SQL code for K-means</td>
<td>81</td>
</tr>
<tr>
<td>5.1</td>
<td>Median Search Algorithm</td>
<td>87</td>
</tr>
<tr>
<td>5.2</td>
<td>Median Projection Algorithm</td>
<td>89</td>
</tr>
<tr>
<td>5.3</td>
<td>Modified Median Projection Algorithm</td>
<td>90</td>
</tr>
<tr>
<td>5.4</td>
<td>Median Projection using SQL</td>
<td>97</td>
</tr>
<tr>
<td>7.1</td>
<td>Computing the time series rank</td>
<td>119</td>
</tr>
<tr>
<td>7.2</td>
<td>Algorithm for Prediction</td>
<td>122</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO</th>
<th>TITLE OF TABLE</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Similarity and Dissimilarity measures of Quantitative Features</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Computational Complexity of Clustering Algorithms</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Data Structures used for SQL K-means</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Selecting Random records</td>
<td>65</td>
</tr>
<tr>
<td>6.1</td>
<td>Run time for K-means</td>
<td>101</td>
</tr>
<tr>
<td>6.2</td>
<td>Inter cluster distance of K-means</td>
<td>103</td>
</tr>
<tr>
<td>6.3</td>
<td>Inter cluster distance for K-means with Modified K-means</td>
<td>105</td>
</tr>
<tr>
<td>6.4</td>
<td>Runtime Comparisons of K-means with Modified K-means</td>
<td>107</td>
</tr>
<tr>
<td>6.5</td>
<td>Runtime for K-means with varying number of columns</td>
<td>109</td>
</tr>
<tr>
<td>6.6</td>
<td>Results of running SQL Clustering on different databases</td>
<td>111</td>
</tr>
<tr>
<td>7.1</td>
<td>Prediction accuracy comparison</td>
<td>128</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀</td>
<td>For Every.</td>
</tr>
<tr>
<td>∈</td>
<td>Belongs to</td>
</tr>
<tr>
<td>⊗</td>
<td>Binary Multiplication operation</td>
</tr>
<tr>
<td>⊕</td>
<td>Binary Addition operation</td>
</tr>
<tr>
<td>(\ln)</td>
<td>Log(_2)</td>
</tr>
<tr>
<td>(\sqrt{n})</td>
<td>Square root on (n)</td>
</tr>
<tr>
<td>≠</td>
<td>Not Equals</td>
</tr>
<tr>
<td>⇔</td>
<td>Implies.</td>
</tr>
<tr>
<td>≡</td>
<td>Congruence</td>
</tr>
<tr>
<td>(O)</td>
<td>Time / Space Complexity of an algorithm</td>
</tr>
</tbody>
</table>
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>ABBREVIATION</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cid</td>
<td>Cluster id</td>
</tr>
<tr>
<td>CLARA</td>
<td>Clustering LARge Applications</td>
</tr>
<tr>
<td>CLARANS</td>
<td>Clustering Large Application based upon RANdomized Search.</td>
</tr>
<tr>
<td>CURE</td>
<td>Clustering Using REpresentatives</td>
</tr>
<tr>
<td>DBMS</td>
<td>DataBase Management System</td>
</tr>
<tr>
<td>DB</td>
<td>DataBase</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density Based Spatial Clustering of Applications with Noise</td>
</tr>
<tr>
<td>DDL</td>
<td>Data Definition Language</td>
</tr>
<tr>
<td>DML</td>
<td>Data Manipulation Language</td>
</tr>
<tr>
<td>DMQL</td>
<td>Data Mining Query Language</td>
</tr>
<tr>
<td>EM</td>
<td>Expected Maximization</td>
</tr>
<tr>
<td>KDD</td>
<td>Knowledge Data Discovery</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Average Error</td>
</tr>
<tr>
<td>NBD</td>
<td>Negative Binomial Distribution</td>
</tr>
<tr>
<td>OLAP</td>
<td>On-Line Analytical Processing</td>
</tr>
<tr>
<td>OLTP</td>
<td>On-line Transaction Processing</td>
</tr>
<tr>
<td>OPTICS</td>
<td>Ordering Points To Identify the Clustering Structure</td>
</tr>
<tr>
<td>PAM</td>
<td>Partitioning Around Medoids</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Data Base Management System</td>
</tr>
<tr>
<td>Rid</td>
<td>Row id</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>SBG</td>
<td>Shifted Beta Geometric</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Introduction to KDD process

KDD refers to the overall process of discovering useful knowledge from databases. KDD consists of several steps. Data mining refers to a particular step in the overall KDD process. Data mining is the application of specific algorithms for extracting patterns, which then will be interpreted and evaluated to produce knowledge from data [US01].

![Figure 1.1: KDD process](image-url)
Knowledge discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [FR91]. KDD process has data selection, preprocessing, transformation, data mining and interpretation steps as shown in Figure 1.1 [US01]. Composition of these steps constitutes the KDD process.

Knowledge discovery is a process as depicted in figure 1.1 and consists of an iterative sequence of the following steps.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Data Cleaning</td>
<td>To remove noise and inconsistent data.</td>
</tr>
<tr>
<td>2. Data Integration</td>
<td>Data may be present in more than one source. In this step, data from multiple sources are combined.</td>
</tr>
<tr>
<td>3. Data selection</td>
<td>Relevant data for the analysis task are retrieved from the database.</td>
</tr>
<tr>
<td>4. Data Transformation</td>
<td>Data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.</td>
</tr>
<tr>
<td>5. Data Mining</td>
<td>It is an essential process where intelligent methods are applied in order to extract data pattern.</td>
</tr>
<tr>
<td>6. Pattern evaluation</td>
<td>Identifying the truly interesting patterns representing knowledge based on some interesting measures.</td>
</tr>
<tr>
<td>7. Knowledge presentation</td>
<td>Representing the knowledge extracted in the user readable format. Visualization and Knowledge representation techniques are used to present the mined knowledge to the user.</td>
</tr>
</tbody>
</table>
Knowledge discovery in databases (KDD) and data mining offer enterprises and other large organizations, a set of methods to discover previously unknown relations among data which are often called patterns. With these patterns enterprises gain deeper insight into their context and are able to use them for better decision-making.

The term data mining denotes the step of the KDD process in which sophisticated methods analyze the data for patterns of interest [AG95] [FR91]. There are varieties of Data mining techniques available. Clustering, classification, and association rule mining are the most-commonly used data mining techniques [MI04].

Clustering is one of the important and most widely used exploratory data mining technique. There are varieties of clustering techniques [RU05]. One of the major limitations of these algorithms excepting DBSCAN, BIRCH and CLARANS is that they are limited to available memory size i.e., the data size should be smaller than the available memory. Even though some techniques like sampling [WE04] has been used, SQL implementation has been found to handle large datasets efficiently [OR02].

1.2 Open Issues of the KDD Process

Although there exist approaches that improve the runtime of instances of the KDD process, time is still an open issue of the KDD process and quality of results is the other one. Open issues concerning the runtime of instances of the KDD processes are:
**Issue1:** Algorithms with long runtime or more iteration during an instance of the KDD process increases the runtime of an instance. If the data set which an algorithm analyzing is very large, then runtime of an algorithm is high even for well scaled algorithm. The consequence of a long runtime of an algorithm is that the analyst who started it is slowed down in ones work by the system until the algorithm terminates.

**Issue2:** Iterating steps in an instance of the KDD process is necessary because the best values of the used algorithm’s parameters and the best set of data for a specific analysis are initially unknown.

**Issue3:** It is common practice to run, data mining algorithms several times, each time with a different combination of parameter values and data. More precisely, the analyst chooses the data and values that will most likely bring the best results best to one’s current knowledge. Even an unsuccessful attempt commonly gives the analyst a better insight into the relations among the analyzed data.

When an analyst re-analyses with different settings the results may not be very similar. Obviously, processing the same intermediate results more than once means waste of time. Yet, previous work introduces techniques which are able to re-use results of a previous analysis when parameters change. If an analyst uses a different subset of the data he/she must compute all results of this new analysis from scratch.
The analyses that share the computation of some intermediate results are not limited to analysis of the same instance of the KDD process. Moreover, a set of analysis can also share several tasks or parts of them. Especially if these analysis use the same data set and also share the same type of analysis, several intermediate results can be identical.

Figure 1.2 shows two instances of the KDD process that use the same data set to analyze. Although the goals of both instances are different, both instances need to cluster the same table (parameter values and used attributes differ).
Pre-processing and most data mining methods require accessing all data which is time-consuming in large databases exceeding 1GB in size. Hence the algorithms that fail to scale linearly are inapplicable to large data sets. Although some data mining algorithms like the clustering algorithms BIRCH [ZH96] and CLARANS [RU05] are very efficient and scale better in the number of tuples, the minimum time needed for pre-processing and data mining is still too long for interactively analyzing the data.

A long runtime of an algorithm or a pre-processing task means that an analyst has to wait longer for a computer to finish its job then he/she can analyze the resulting patterns. Zhang et.al. [ZH96] states that, “the most time-consuming part of data-mining algorithms is scanning the data while operations in main memory are negligible”. As a consequence, approaches of improving data mining algorithms must focus on minimizing the number of required database scans. Another approach is to implement clustering in DBMS, which is the major focus area of this research.

Quality is always an additional issue while considering the runtime. Yet, improving quality and runtime do not exclude each other. Thus, maintaining the quality of the results and achieving the same with less time complexity is a challenging task in the KDD process.

1.3 Open issues in Clustering

We are living in a world full of data. Every day, people encounter a large amount of information and store or represent it as data, for further analysis and
management. One of the vital means in dealing with these data is to classify or group them into a set of categories or clusters. Actually, as one of the most primitive activities of human beings [MQ01], classification plays an important and indispensable role in the data segmentation.

In order to learn a new object or understand a new phenomenon, people always try to seek the features that can describe it, and further compare it with other known objects or phenomena, based on the similarity or dissimilarity, generalized as proximity, according to some certain standards or rules.

Basically, classification systems are either supervised or unsupervised, depending on whether they assign new inputs to one of a finite number of discrete supervised classes or unsupervised categories, respectively [BE02] [NI03] [GR98]. In supervised classification, the mapping from a set of input data vectors to a finite set of discrete class labels is modeled in terms of mathematical function. The values of these parameters are determined (optimized) by an inductive learning algorithm (also termed inducer), whose aim is to minimize an empirical risk (function related to inductive principle) on a finite data set [GR98] [NI03] [BE02].

In unsupervised classification, called clustering or exploratory data analysis, no labeled data are available [JA99] [J108]. The goal of clustering is to separate a finite unlabeled data set into a finite and discrete set of “natural” hidden data structures. It is noteworthy that clustering differs from multidimensional scaling (perceptual maps), whose goal is to depict all the evaluated objects in a way that minimizes the topographical distortion while using as few dimensions as possible.
As pointed out by Jain [JA99], “In cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create “interesting” clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups”.

Clustering algorithms partition data into a certain number of clusters (groups, subsets or categories). Clustering is a division of data into groups of similar objects, representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. It models data by its clusters. Both the similarity and the dissimilarity should be examinable in a clear and meaningful way [FO65].

Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning and the resulting system represents a data concept. From a practical perspective, clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology and many others.

Clustering is useful in several exploratory pattern-analysis, grouping, decision-making and machine-learning situations, including data mining, document retrieval, image segmentation and pattern classification. However, in many such problems, there is little prior information (e.g., statistical models) available about the data and the decision-maker must make as few assumptions
about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment of their structure.

The procedure for Cluster analysis is as shown in the figure 1.3.

![Figure 1.3 Procedure of cluster analysis.](image)

**Figure 1.3 Procedure of cluster analysis.**
1.3.1 Feature selection or extraction.

Feature selection chooses distinguishing features from a set of candidates, while feature extraction utilizes some transformations to generate useful and novel features from the original ones. Both are very crucial to the effectiveness of clustering applications. Elegant selection of features can greatly decrease the workload and simplify the subsequent design process. Generally, ideal features should be of use in distinguishing patterns belonging to different clusters, immune to noise, easy to extract and interpret[AG01][AG00].

1.3.2 Clustering algorithm design or selection.

This step is usually combined with the selection of a corresponding proximity measure and the construction of a criterion function. Patterns are grouped according to whether they resemble each other. The proximity measure directly affects the formation of the resulting clusters. Almost all clustering algorithms are explicitly or implicitly connected to some definition of proximity measure. Once a proximity measure is chosen, the construction of a clustering criterion function makes the partition of clusters an optimization problem, which is well defined mathematically. Clustering is ubiquitous and a wealth of clustering algorithms has been developed to solve different problems in specific fields. However, there is no clustering algorithm that can be universally used to solve all problems [JA99] [KA07].
1.3.3 Cluster validation.

Given a data set each clustering algorithm can always generate a division, no matter whether the structure exists or not. Moreover different approaches usually lead to different clusters; and even for the same algorithm, parameter identification or the presentation order of input patterns may affect the final results. Therefore, effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms. These assessments should be objective and have no preferences to any algorithm. Also, they should be useful for answering questions like how many clusters are hidden in the data, whether the clusters obtained are meaningful or just an artifact of the algorithms or why we choose some algorithm instead of another [MA02]. Generally, there are three categories of testing criteria: external criteria, internal criteria, and relative criteria. These are defined on three types of clustering structures, known as partitioned clustering, hierarchical clustering, and individual clusters [HAL01].

External criteria are based on some pre-specified structure, which is the reflection of prior information on the data and used as a standard to validate the clustering solutions. Internal tests are not dependent on external information (prior knowledge). On the contrary, they examine the clustering structure directly from the original data [JI08][MA02]. Relative criteria deals with the evaluation of clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different input parameter values [MA02].

The measure of a good cluster is based on intra-class and inter-class similarity. A good clustering method will produce high quality clusters with high
intra-class similarity and low inter-class similarity. In other words, data items contained in a cluster should be similar to each other, but not similar to items contained in other clusters. The quality of a clustering depends on the similarity measure used by the method, its implementation and its ability to discover some or all of the hidden cluster patterns. Cluster analysis is highly subjective since it is hard to define “similar enough” or “good enough” in clusters. The quality of a clustering method is also measured by its ability to discover some or the entire hidden pattern. The quality of Clustering can be assessed based on a measure of dissimilarity of objects.

The ultimate goal of clustering is to provide users with meaningful insights from the original data, so that they can effectively solve the problems encountered.

### 1.4 Scalable algorithms for Clustering

K-means is a data mining algorithm which performs clustering. As mentioned in the previous section, clustering is dividing a dataset into a number of groups such that similar items fall into same groups. Clustering uses unsupervised learning technique which means that resulting clusters are not known before the execution of clustering algorithm, unlike the case in classification. Some clustering algorithms takes the number of desired clusters as input while some others decide the number of result clusters themselves.

K-means algorithm uses an iterative procedure in order to cluster database. It takes the number of desired clusters and the initial means as inputs and produces
final means as output. If the algorithm is required to produce K clusters then there will be K initial means and K final means. After termination of K-means clustering, each object in data set becomes a member of one cluster. This cluster is determined by searching throughout the means in order to find the cluster with nearest mean to the object. Minimum distanced mean is considered to be the mean of cluster to which examined object belongs.

K-means algorithm has some limitations, they are

- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples. In this research, a seed selection algorithm based on median projection is proposed.

- The results produced depend on the initial values for the means, and it frequently happens that sub-optimal partitions are found. The standard solution is to try a number of different starting points. As the median projection provides almost optimal cluster centers this problem is mitigated to a large extent.

- The results depend on the value of K. Since, the proposed algorithm provides optimal clusters possible; if ‘k’ is less than this theoretical maximum then ‘K’ means can be selected from maximum number of clusters possible. If ‘k’ is greater than maximum clusters possible, the algorithm reduces ‘k’ to maximum number of clusters possible recommended by the proposed algorithm.

This last problem is troublesome, since we often have no way of knowing how many clusters exist. Finding the optimal number of clusters for any given data set is a challenging task. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given
criterion; we need to be careful because increasing k results in smaller error function values by definition, but also an increasing risk of over fitting. An approach to address this problem is proposed in this research using median projection algorithm. Once the number of seeds is selected using median projection algorithm, the K-means algorithm is implemented.

The proposed algorithms are implemented using Structured Query Language. The advantages of implementing K-means using SQL are

- It is available in any relational DBMS and isolates the application programmer from internal mechanisms of the DBMS.
- Many datasets are stored in a relational database. Trying different subsets of data points and dimensions is more flexible, faster and easier to do inside a DBMS with SQL queries than outside with alternative tools.
- Managing large datasets without DBMS is tedious task.
- Space management, fault tolerance, secured access, concurrency control, etc., are automatically handled by DBMS.

It is expected that the proposed approaches excel in terms of runtime complexity, scalability and quality of clusters.

This dissertation focuses on the problem of pattern discovery using in-database clustering implemented in SQL and the time-space complexity for handling large datasets, which otherwise cannot be fitted into main memory. Recently, many approaches have been introduced to improve techniques specifically for clustering/segmenting large datasets. However, these approaches try to optimize a clustering analysis with an assumption of memory resident data.
CHAPTER 2

LITERATURE SURVEY

2.1 Setting the scene

Humans have a strong temptation to place objects, people and ideas into groups. For many scientific and computational problems, clustering is a key tool for ensuring data consistency, cost minimization and computational efficiency [CH98].

Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details but achieves simplification. Clustering models the data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics and numerical analysis [ES02]. From machine learning or Database Management System perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning and the resulting system represents a data concept [MI97]. From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, Customer Relationship Management, Marketing, Medical diagnostics, Computational biology and many others [MIT02].
Clustering is the subject of active research in several fields such as statistics, pattern recognition and machine learning. This survey focuses on clustering in data mining. Data mining, adds to clustering, the complications of very large datasets with many attributes of different types. This imposes unique computational requirements on relevant clustering algorithms. A variety of algorithms have recently emerged, meet these requirements and were successfully applied to real-life data mining problems [BE02][AL01]. The goal of this survey is to provide a comprehensive review of different clustering techniques in data mining and provide sufficient evidence for the proposed research.

Clustering as discussed earlier, divides data into groups of similar objects. Each group called cluster consists of objects that are similar between themselves and dissimilar to objects of other groups. Representing data by fewer clusters necessarily looses certain fine details but achieves simplification. It represents many data objects by few clusters and hence, it models data by its clusters [VI02].

Clustering is useful in several exploratory pattern-analysis [AG95], grouping [FO65], decision-making [ES02] and machine-learning situations, including data mining, document retrieval, image segmentation and pattern classification [KA07]. However, in many such problems there is less prior information available about the data and the decision-maker must make few assumptions about the data. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment.
2.2 Major Classification of Clustering Algorithms

The number of clustering algorithms available is large. The choice of clustering algorithm depends both on the type of data available and on the particular purpose and application [JI06].

There are different approaches for clustering. In the next section few clustering techniques are discussed.

Clustering

Hierarchical  Partitioned  Density  Grid  Model  Statistics  Neural
   Based      Based     Based     Based    Network

Agglomerative
Divisive

2.2.1 Hierarchical Method.

Hierarchical clustering builds a cluster hierarchy or in other words, a tree of clusters, also known as a dendrogram. Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent. Such an approach allows exploring data on different levels of granularity. A Hierarchical Method creates a hierarchical decomposition of given set of data objects. A Hierarchical Method can be classified as Agglomerative or Divisive based on how the decomposition is formed.
The Agglomerative approach, also called the “bottom up” approach starts with each object forming a separate group. It successively merges the objects or groups close to one another until all of the groups are merged into one or until the termination condition holds [VI02].

The divisive approach, also called “top down” approach starts with all the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters until eventually each object is in one cluster or until a termination condition holds.

Hierarchical Methods suffer from the fact that once a merge or split is done, it can never be undone. It leads to less computation cost without checking for combinatorial options. The problem, it cannot handle exceptional conditions. The major advantages of hierarchical clustering methods are embedded flexibility regarding the level of granularity, ease of handling any forms of similarity or distance and applicability to any attributes types [BE02].

2.2.2 Density Based Method.

The idea in Density methods is to continue growing the given cluster as long as the density (number of objects or data points) in the neighborhood exceeds some threshold, i.e., for each data point within a given cluster the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape [JA99].
Based on the notion of density-reach ability, a density-based clustering algorithm, DBSCAN is developed for clustering data in a database. It checks the ε-neighborhood of each point in the database. If the ε-neighborhood of a point p contains more than minimum points, a new cluster with p as a core object is created. It then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters. The process terminates when no new point can be added to any cluster [JI06] [JI08].

2.2.3 Grid based Method.

A Grid based method quantizes the object space into a finite number of cells which form a grid structure. It then performs all of the clustering operations on the grid structure (i.e., on the quantized space). The main advantage of this approach is its fast processing time which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the optimized space [VI02].

Some typical examples of the grid-based approach include STING [JI08], which explores statistical information stored in the grid cells; Wave Cluster [JI08], which clusters objects using a wavelet transform method; and CLIQUE [JI08], which represents a grid and density-based approach for clustering in high dimensional data space.
2.2.4 Model based Method.

A Model based method hypothesizes a model for each of the clusters and finds the best fit of the data to that model. A model based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking “noise” or outliers into account and thus yielding robust clustering methods [JI08].

Some clustering algorithms integrate the ideas of several clustering methods, so that it is difficult to classify a given algorithm as uniquely belonging to only one clustering method category. The Model based clustering methods have two major approaches: Statistical approach and Neural network approach.

2.2.5 Neural network approach

The best known Neural network approach in clustering is SOM (Self Organizing feature Map) [DE01]. It can be viewed as a nonlinear projection from an multidimensional input space onto a lower-order (typically 2-dimensional) regular lattice of cells. Such a mapping is used to identify clusters of elements that are similar (in an Euclidean sense) in the original space [JI08] [RZ85].

There are two methods of the neural networks approach in clustering. The first is competitive learning and the second is self organizing feature maps, both of which involve competing neural units.
2.2.6 Partitioning Method

Given a database of n objects or data tuples, a partitioning method constructs k partitions of the data, where each partition represents a cluster and \( k \leq n \). It classifies the data into k groups, which together satisfy the following requirements.

1. Each group must contain at least one object
2. Each object must belong to exactly one group. Given k, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique which attempts to improve the partitioning by moving objects from one group to another [KI04].

To achieve global optimality in Partitioning based clustering, most applications adopt the following heuristic methods.

1. K-means algorithm - each cluster is represented by the mean value of the objects in the cluster.
2. K-medoids algorithm - each cluster is represented by one of the objects located near the center of the cluster.

The heuristic clustering method works well for finding spherical shaped clusters in small to medium sized databases. For finding cluster with complex shapes and for clustering very large data sets partitioning based methods need to be extended [BE02].

2.2.6.1 The K-medoids method
Algorithm: 2.1 K-medoids Algorithm

**Input:** The number of clusters k and a database containing n objects.

**Output:** A set of K clusters which minimizes the sum of the dissimilarities of all the objects to their nearest medoids.

**Method:** The K-medoids algorithm is implemented as follows:

1) arbitrarily choose K objects as the initial medoids.

2) repeat
   
a) assign each object to the cluster corresponding to the nearest medoid.
   
b) calculate the objective function, which is the sum of dissimilarities of all the objects to their nearest medoid.
   
   swap the medoid x by an object y if such a swap reduces the objective function.

3) until no change.

The K-means algorithm is sensitive to outliers since an object with some extremely large value may substantially distort the distribution of data. Instead of taking the mean value of the objects in a cluster as a reference point, one may take a representative object in a cluster called a medoid, the most centrally located point in a cluster. Thus the partitioning method can still be performed based on the principle of minimizing the sum of the dissimilarities between each object and with its corresponding reference point which forms the basis of the K-medoids
PAM (Partition Around Medoids) is a K-medoids type clustering algorithm. It finds k clusters in n objects by first finding a representative object (medoid) for each cluster. The initial set of medoids could be arbitrarily selected. It then iteratively replaces one of the medoids by one of the non-medoids as long as the total distance of the resulting clustering is improved [ES02].

2.2.6.2 The CLARA method

A typical K-medoids partition algorithm like PAM works effectively for small data sets but it does not scale well for larger data sets. To deal with larger data sets a sampling based method called CLARA (Clustering LARge Applications) was used [ES02].

The idea of CLARA is as follows: Instead of taking the whole set of data into consideration, only a small portion of the real data is chosen as a representative of the data and medoids are chosen from this sample using PAM. If the sample is selected in a fairly random manner, it correctly represents the whole data set and the representative objects (medoids) chosen will therefore be similar to those chosen from the whole data set. The effectiveness of CLARA depends on the sample size. CLARA will never find the best clustering. This is exactly the tradeoff for efficiency. A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased [JA99].
2.2.6.3 The CLARANS method

To improve the quality and scalability of CLARA, another clustering algorithm called CLARANS (Clustering Large Applications based upon RANdomized Search) was proposed by Han [HA94]. It is also a K-medoids type algorithm and combines the sampling technique with PAM. However, unlike CLARA, CLARANS does not confine itself to any sample at any given time. While CLARA has a fixed sample at every stage of the search, CLARANS draws a sample with some randomness in each step of the search. The clustering process can be presented as searching a graph where every node is a potential solution, i.e., a set of $k$ medoids. The clustering obtained after replacing a single medoid is called the neighbour of the current clustering.

The number of neighbours to be randomly tried is restricted by a parameter. If a better neighbour is found, CLARANS moves to the neighbour's node and the process starts again; otherwise the current clustering produces a local optimum. If the local optimum is found, CLARANS starts with new randomly selected nodes in search for a new local optimum [JA99].

CLARANS has been experimentally shown to be more effective than both PAM and CLARA. The computational complexity of CLARANS for each iteration is basically linearly proportional to the number of objects. CLARANS also enables the detection of outliers, i.e., the points that do not belong to any cluster [JI08].
2.2.6.4 K-means Method

K-means Algorithm [HA01][AL01] is a popular and most widely used clustering algorithm. It is an important algorithm for identifying the structure of 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 location of all the members of a particular cluster.

The K-means algorithm takes the input parameter, K and partitions a set of n objects into K clusters so that the resulting intra-cluster similarity is high whereas the inter-cluster similarity is low. Cluster similarity is measured with respect to the mean value of the objects in the cluster, which can be viewed as the cluster's "center of gravity".

The algorithm proceeds as follows: First it randomly selects K objects each represents a cluster mean or center. For each of the remaining objects an object is assigned to the cluster to which it is the most similar based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the squared-error criterion is used, defined as

\[ E = \sum_{i=1}^{k} \sum_{x \in C_i} [x - m_i]^2 \]

Where x is the point in space representing the given object and m_i is the mean of cluster C_i. This criterion tries to take the resulting K clusters as compact and separate as possible. The K-means procedure is summarized in algorithm 2.2.
Algorithm 2.2: The K-means algorithm for partitioning based on the mean value of the objects in the cluster.

| **Input:** | The number of clusters K, and a database containing n objects. |
| **Output:** | A set of K clusters which minimizes the squared-error criterion. |
| **Method:** | The k-means algorithm is implemented as follows. |
| | 1) arbitrarily choose K objects as the initial cluster centers. |
| | 2) repeat. |
| | 3) (re) assign each remaining object, to the cluster with which the object is the most similar based on the mean value of the objects in the cluster. |
| | 4) update the cluster means, i.e., calculate the mean value of the objects for each cluster. |
| | 5) until no change. |

The algorithm attempts to determine K partitions that minimize the squared-error function. It works well when the clusters are compact clouds that are well separated from one another. The pictorial representation of working of K-means algorithm is shown in the figure 2.1.
Figure 2.1: Clustering of a set of points based on the K-means method.

The method is relatively scalable and efficient in processing large data sets because the computational complexity of the algorithm is $O(nKt)$, where $n$ is the total number of objects, $K$ is the number of clusters, and $t$ is the number of iterations. Normally $K \leq n$ and $t \leq n$. The method often terminates at a local optimum [JI08].
The K-means method can be applied only when the mean of a cluster is defined. This may not be the case in some applications such as when data with categorical attributes are involved. The necessity for users to specify K, the number of clusters, in advance can be seen as a disadvantage. The k-means method is not suitable for discovering clusters with non-convex shapes or clusters of very different size. Moreover, it is sensitive to noise and outlier data points since a small number of such data can substantially influence the mean value [AL01] [GO04].

There are quite a few variants of the K-means method which differ in the selection of initial K-means, the calculation of dissimilarity and the strategies to calculate cluster means.

Another variant to K-means is K-modes method by Huang [HU98] which extends the K-means paradigm to cluster categorical data by replacing the means of clusters with modes using new dissimilarity measures to deal with categorical objects and using a frequency-based method to update modes of clusters. The k-means and the k-modes methods can be integrated to cluster data with mixed numeric and categorical values which is called the k-prototype method.

Another variant to k-means is called EM (Expectation Maximization) which extends the K-means paradigm in a different way [OR00]. Instead of assigning each point to a dedicated cluster, it assigns each point to a cluster according to some weight representing the probability of membership. Therefore, new means are computed based on weighted measures.
A recent effort on scaling K-means algorithm, proposed by Bradely et.al., is based on an idea of identifying regions of the data that are compressible, regions that must be maintained in main memory and the regions that are discardable. A point is discardable if its membership in a cluster is ascertained and such a point can be discarded and only the summarized clustering feature of such discarded points should be retained. A point is compressible, if it is not discardable but belongs to a tight sub cluster and such a sub cluster is summarized using its summarized clustering feature. If a point is neither discardable nor compressible such a point should be retained in main memory.

Bottou et.al., [BO01] has proposed convergence properties of K-means algorithm. The original K-means algorithm works with many resident data and disk resident options are also available.

In spite of the above improvements, the K-means algorithm retains the following limitations:

1. Requirement of apriori number of clusters is paradoxical since in an unsupervised learning the structure will be unknown.
2. Multiple dataset scan required for convergence makes it impractical to use on large datasets.

The first problem has been solved by hierarchical clustering on a sample data, then using the cluster centers as seeds or by cluster merging. The second problem has been solved by number of algorithms or approaches by focusing on reducing number of passes or working on samples [BR98] [HA02]. However, the major problems are, either they provide approximate solutions (possibly with deterministic or probabilistic bounds on the quality of solutions) or random clusters means varying widely across samples.
Domingos et.al., [DE01] proposed a faster sub linear version of K-means using sampling or similar statistical bound. The algorithm consists of number of runs of K-means with sample where in every iteration, sample size is increased to maintain the loss bound from the multi-pass K-means. The object is to converge to a solution which is close to that of a multi-pass K-means by a predefined bound with good probability.

Nittel [NI03] proposed to apply K-means algorithm to cluster massive datasets scanning the dataset only once. His algorithm splits the entire dataset into chunks and each chunk can fit into the main memory. Then it applies K-means on each chunk of data and merges the clustering results by another K-means type algorithm. Good results are shown for a real dataset. However, no theoretical bounds on the results are established.

These algorithms cannot maintain the exact result which will be obtained using a multi-pass K-means algorithm. All of the above efforts are on reducing the number of passes on the data. Our focus is developing an algorithm to address the above. This necessitates implementing in-database K-means clustering algorithms for which SQL seems to be better candidate.

The use of SQL for data mining has been researched and well documented in literature. Association rule mining is explored in [AG94] [AG01], general data mining primitives for printing table and sampling gas been proposed in [RA05], aggregate functions provided has SQL extensions in [ME98], MSQL language for association rule mining has been proposed in [IM99].
SQL like operator to express a broad class of association rule mining has been given in [SI03]. Primitives to mine decision tree are introduced in [DE01][SA01]. EM algorithm and K-means algorithm are proposed in [OR022].

SQL extensions to perform spreadsheet like operations are introduced in [WI03]. This shows SQL by its implementation in a standard DBMS, capable of handling massive datasets and can provide an in-database mining platform for implementing clustering algorithms.

Ordonez[OR04] has focused on writing efficient SQL code to implement K-means instead of proposing yet another fast clustering algorithm for large datasets. Implementing these algorithms requires a high level programming language to manage memory and perform complex hierarchical operations.

A fast K-means prototype to cluster transaction datasets using disk-based matrices is presented in [OR04]. The disk based implementation and the SQL-based implementation represents complementary solutions to implement K-means in a relational DBMS, but authors believe that the SQL based solution will become more valuable as CPU utilization increases and hardware cost decreases.

Ordonez [OR02] 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.
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. This motivates the proposed work to develop efficient algorithms.

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 [NE01][WI01][IM99]VI02].

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 store RDBMS are available in both memory model like SQLite [NE01] or disk based architecture like MYSQL [IM99] and Postgre [NM01].

However the limitation of holding larger than memory datasets requiring complex page swapping mechanism lead to unavailability of these systems for expensive analytical functions. The current research in handling large datasets with analytical functions has 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 [MA01] and disk based [ST05] versions are available.
Abadi et al., compare Column stores database with row store database and how different are they really when start using both the database [AB08] [HA06].

The SQL implementation of K-means algorithm has been tested on row stored databases and found to perform exceptionally well on large datasets [OR06] [OR02]. However, evaluating performance of K-means algorithm with column stored database on large datasets is a challenging task. 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. This study justifies the comment made by Alan Halverson [HA01].
CHAPTER 3

OBJECTIVE OF THE PRESENT WORK

3.1 Introduction

There exist many efficient clustering algorithms in the data mining literature [JI06][VI02][BE02]. Most of them follow the approach proposed in [ZO06], minimizing disk access and doing most of the work in main memory. However, many of these algorithms are hard to implement inside a real DBMS where the programmer needs to worry about storage management, concurrent access, memory leaks, fault tolerance, security and so on.

The latest research in handling large datasets with analytical functions has been shown dramatically to improve the performance of analytical queries. These systems use column store architecture where data will be stored in column fashion rather than row fashion.

**K-means algorithm has the following limitations:**

1) When the clustering is used as an unsupervised learning algorithm for data mining, it is difficult to provide the number of clusters apriori as the structure is not known.
2) Since the records are chosen randomly the order of the records determine the quality of the clusters, which requires repetitive runs to get a better set of cluster centers. This increases the search space and time.

3) Since the records are chosen randomly the time for convergence for a solution with acceptable error rate is non-deterministic and can be quite high.

To overcome the above limitations of K-means Clustering algorithm the proposed algorithms were developed using SQL.

3.2 Objectives:

The objectives of this research work are:

- Seeding of K-means algorithm to accelerate the convergence.
- Evaluating the performance of our modified K-means algorithm on column store databases on both in-memory and disk based models.

3.3 Proposed work:

In this research work, clustering large datasets using in-database approach are proposed. 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.
- Developing user driven clustering method.
The novel seeding algorithm is based on an assumption of potential possibility of existence of gaps in an ordered sequence of an attribute value set. Once these gaps are identified and statistically validated for sufficient differentiation they will be projected across other attributes which may not have such kind of properties. The empirical experimentation on synthetic and real world datasets indicate this approach can substantially improve the quality of clusters by means of both identifying the structure of the clusters in the data and providing initial cluster centers which are near optimal. Since the proposed algorithm works on the basis of sorting the data it can be easily implemented in a declarative language like SQL.

The results produced in the standard approach depend on the initial values for the means and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points. However, in our approach the median projection provides almost optimal cluster centers, this problem is mitigated to a large extent.

Another algorithm is also proposed to build user driven clustering methods for the real time applications using ranking. An application of this method is studied as a case study. The Proposed algorithms are used to build user driven clustering methods for real time applications. An example of this is performance of retention programs across regions or products or dealers in a telecom company.
CHAPTER 4

IN-DATABASE CLUSTERING USING STRUCTURED QUERY LANGUAGE

4.1 Introduction

K-means is data mining algorithm which performs clustering. Clustering is dividing a dataset into a number of groups such that similar items fall into same groups [KA01]. Clustering uses unsupervised learning technique which means that resulting clusters are not known before the execution of clustering algorithm unlike the case in classification. Some clustering algorithms takes the number of desired clusters as input while some others decide the number of resulting clusters themselves.

K-means algorithm uses an iterative procedure in order to cluster database [KA01]. It takes the number of desired clusters and the initial means as inputs and produces final means as output. If the algorithm is required to produce K clusters then there will be K initial means and K final means. On completion, K-means algorithm produces K final means which answers why the name of algorithm is K-means.

After termination of K-means clustering, each object in dataset becomes a member of one cluster. This cluster is determined by searching throughout the
means in order to find the cluster with nearest mean to the object. Shortest distanced mean is considered to be the mean of cluster to which examined object belongs. K-means algorithm tries to group the items in dataset into desired number of clusters. To perform this task it makes some iteration until it converges. In each iteration calculated means are updated such that they become closer to final means, finally the algorithm converges and stops performing iterations.

Expected convergence of K-means algorithm is illustrated in the figures 4.1 to 4.4 with $K = 3$. Algorithm converges in three iterations in the illustrated example. Blue points represent the initial means which may be gathered randomly. Purple points stands for the intermediate means. Finally, red points represent the final means which are also the results of K-means clustering. As in the literature, calculated means move to the cluster centroids by each iteration of K-means algorithm. When they reach to the cluster centroids, the algorithm converges.

### 4.1.1 Measurement of Distance between Objects and Means

Different techniques can be used in K-means clustering in order to measure the distance between objects and means. Most popular two distant metrics are Manhattan Distance and Euclidean Distance.

For clustering tuples, there must exist a measurement of distances for each attribute and a way to combine distances of different attributes. Distance functions fulfill the function of combining distance of several continuously and interval scaled attributes to a common distance.
### Table 4.1 Similarity and Dissimilarity measures of Quantitative Features

<table>
<thead>
<tr>
<th>Measures</th>
<th>Formula</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinKowski Distance</td>
<td>$D_{ij} = \left( \sum_{i=1}^{d}</td>
<td>x_{il} - x_{jl}</td>
</tr>
<tr>
<td>Euclidian Distance</td>
<td>$D_{ij} = \left( \sum_{i=1}^{d}</td>
<td>x_{il} - x_{jl}</td>
</tr>
<tr>
<td>City – block Distance</td>
<td>$D_{ij} = \sum_{i=1}^{d}</td>
<td>x_{il} - x_{jl}</td>
</tr>
<tr>
<td>Sup distance</td>
<td>$D_{ij} = \max_{1 \leq j \leq d}</td>
<td>x_{il} - x_{jl}</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>$D_{ij} = (x_i - x_j)^T S^{-1} (x_i - x_j)$, where $S$ is the within-group covariance matrix.</td>
<td>Invariant to any nonsingular linear transformation. $S$ is calculated based on all objects. Tend to form hyper ellipsoidal clusters.</td>
</tr>
<tr>
<td>Pearson correlation</td>
<td>$D_{ij} = \frac{1-r_{ij}}{2}$, where $r_{ij} = \frac{\sum_{i=1}^{d} (x_{il} - \bar{x}<em>i)(x</em>{jl} - \bar{x}<em>j)}{\sqrt{\sum</em>{i=1}^{d} (x_{il} - \bar{x}<em>i)^2 \sum</em>{i=1}^{d} (x_{jl} - \bar{x}_j)^2}}$</td>
<td>Not a metric. Derived from correlation coefficient. Unable to detect the magnitude of differences of two variables.</td>
</tr>
<tr>
<td>Point symmetry Distance</td>
<td>$D_{ir} = \min_{1 \leq j \leq N}$ ( \frac{</td>
<td></td>
</tr>
<tr>
<td>Cosine similarity</td>
<td>$S_{ij} = \cos \alpha = \frac{x_i^T x_j}{</td>
<td></td>
</tr>
</tbody>
</table>
A distance function is a function that takes two tuples as its input and returns a float variable, the distance that indicates the dissimilarity of both tuples. The table 4.1 depicts the Similarity and Dissimilarity measures of Quantitative Features.

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances.

The Euclidian distance, the Manhattan distance and the Minkowski distance are popular distance functions for interval scaled attributes.

The Euclidian distance $d_E$ of two tuples is the length of the distance vector of the vectors representing both tuples in a Euclidian vector space, i.e., if two tuples are represented by the vectors $\mathbf{x}$ and $\mathbf{y}$ the Euclidian distance is the length of the vector $\mathbf{x} - \mathbf{y}$ which is

$$d_E(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=0}^{d} (x_i - y_i)^2}$$

Euclidean distance is the square root of addition of squared differences between corresponding dimensions of object and the mean. Since Euclidean distance is the most common distance metric, especially when dealing with multi-dimensional data, Euclidean distance is used for K-means clustering in this project as a distance metric.

Manhattan distance is one of the simplest metric. This metric is the absolute value of difference between object and the mean. Manhattan distance $d_M$ is the sum of the absolute values of the differences of the tuples in each attribute.

$$d_M(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$$
The Minkowski distance is the generalized distance function of Manhattan distance and Euclidean distance. Minkowski distance has a parameter $p$ that determines the exponent of the difference of the attribute values of the tuples. 

$$d_p(x, y) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

Manhattan distance is a Minkowski distance with parameter $p = 1$. The Euclidean distance is a Minkowski distance with parameter $p = 2$.

The specific requirements of the KDD process instance determine which distance function is the one to prefer. The Euclidean distance will be the most frequently used distance function when analyzing quantitative data.

### 4.1.2 Selection of Initial Means

Selecting of initial means is up to the developer of clustering system [BR98]. This selection is independent of K-means clustering, because these means are inputs of K-means algorithm. Some developers prefer to select initial means randomly from dataset while some others prefer to produce initial points randomly.

It is known that selection of initial means affects the execution time and success of K-means algorithm. Some strategies are developed to gather better results considering the initial means. The simplest of these strategies is to execute K-means algorithm with different sets of initial means and then select the best results. But this strategy is hardly feasible especially for serial K-means when dataset is large.
Another strategy to gather better clustering results is to refine initial points [BR98]. If it is possible to begin K-means algorithm with initial means which are closer to final means, it is strongly possible that number of iterations that the algorithm needs to converge will decrease which also lessens the required time for conversion and increase the accuracy of final means.

4.2 Cluster validity Measures

The clustering algorithms are based on some assumptions in order to define a partitioning of a data set. As a consequence, they may behave in a different way depending on:

- The features of the data set (geometry and density distribution of clusters).
- The input parameters values.

Clustering is an unsupervised process since there are no predefined classes and no examples that would indicate grouping properties in the data set. The majority of the clustering algorithms behave differently depending on the features of the data set and the initial assumptions for defining groups. Therefore, in most applications the resulting clustering scheme requires some sort of evaluation as regards its validity [MA02].

The procedure of evaluating and assessing the results of a clustering algorithm is termed as Cluster validity. There are three approaches for cluster validity.

1. External criteria:
It deals with evaluating the results of a clustering algorithm based on a pre-specified structure, which is imposed on a data set and reflect our intuition about the clustering structure of the data set.

2. Internal criteria:

It deals with evaluating the results of a clustering algorithm in terms of quantities that involve the vectors of the data set.

3. Relative criteria:

It deals with the evaluation of a clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different input parameter values.

The first two approaches are based on statistical tests and their major drawback is their high computational cost. Moreover, the indices related to these approaches aim at measuring the degree to which a data set confirms an apriori specified scheme. On the other hand, the third approach aims at finding the best clustering scheme that a clustering algorithm can define under certain assumptions and parameters [MA02]. The major parameters are Dunn index, Davies-Bouldin index, C-index, Inter cluster distance and Cluster density. Each of them are discussed in brief below.

**Dunn index:**

The Dunn index defines the ratio between the minimal intracluster distance to maximal intercluster distance.

The index is given by: \[ D = \frac{d_{\text{min}}}{d_{\text{max}}} \]

Where \( d_{\text{min}} \) denote the smallest distance between two objects, from different clusters, and \( d_{\text{max}} \) the largest distance of two objects from the same cluster. The Dunn index is limited to the interval \([0, \infty]\) and should be maximized[DU74].

**Davies-Bouldin index:**
This index, DB, is defined as:

$$DB = \frac{1}{n} \sum_{j=1, i\neq j}^{n} \max \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

where $n$ is the number of clusters, $\sigma_i$ is the average distance of all patterns in cluster $i$ to their cluster center $c_i$, $\sigma_j$ is the average distance of all patterns in cluster $j$ to their cluster center $c_j$ and $d(c_i, c_j)$ the distance of cluster centers $c_i$ and $c_j$. Small values of DB correspond to clusters that are compact and whose centers are far away from each other. Consequently, the number of clusters that minimizes DB is taken as the optimal number of clusters.

**C-index:**

The C-index is defined as:

$$C = \frac{S - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}}$$

Where $S$ is the sum of distances over all pairs of objects form the same cluster, $n$ is the number of those pairs and $S_{\text{min}}$ is the sum of the $n$ smallest distances if all pairs of objects are considered. Likewise $S_{\text{max}}$ is the sum of the $n$ largest distances out of all pairs. The C-index is limited to the interval $[0, 1]$ and should be minimized.

**Inter cluster distance:** The distance across clusters.

$$d_i = |c_1 - c_2|$$

$\text{Avg}(d_i)$ is statistically differentiable.
Inter-cluster distance measured by within-cluster sum of squares. Measures cluster "compactness".

For one cluster r:

\[ D_r = \sum_i \sum_j \left| x_i - x_j \right|^2 \]

\[ = 2n_r \sum_i \left| x_i - \bar{x} \right|^2 \]

For all k clusters:

\[ W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \]

**Cluster Density:** How many objects are covered within ±2σ. It is a Standard deviation of points/distances on spherical clusters. We expect 99% to be covered. It is assumed 95% will be within ±2σ

4.3 Framework - Definitions:

**Tuple:** A tuple is a set of attributes which constitutes a record. Formally a tuple is represented as

\[ T = \{ t_i \forall t \in \|t\| i \in \text{C N} \} \]

where \( t_i \) is the \( i^{th} \) attribute of tuple T and t takes the values in the domain of t \( (\text{dom}(t)) \).

It is composed of set of attributes (column or fields).

**Attribute:** Each attribute value is a constant taken from the attribute domain.

Database instance is the data in DB at a given instance of time.

D is called domain of \( A_i \) and denoted by \( \text{dom}(A_i) \)

**Relation:** is a set of tuples (records), i.e., a table whose rows are tuples (records).
Denoted by $R(A_1,A_2,A_3,\ldots, A_n)$ is made up of a relation name $R$ and a list of attributes $A_1,A_2,A_3,\ldots, A_n$.

**Predicates:** is selectivity refers to the number (or proportion) of records which satisfy a particular predicate.

**The natural join:** is formally defined as Cartesian product followed by selection with equality predicates on the columns with identical names.

\[ Q \leftarrow R <\text{List } 1> \ast S <\text{List } 2> \]  

$<$List 1$>$ specifies a list of $i$ attributes from $R$ and $<$List 2$>$ specifies a list of $i$ attributes from $S$.

\[ A(x,y) \Join B(y,z) = \sigma_{A_{xy}=B_y}(A(x,y) \Join B(y,z)) \]  

The two identical $y$ columns in the resulting relation are merged into a single column. Natural join is performed by equating all attributes pairs of two relations.

**Union:** is formally defined as set union restricted to a common set of relation attributes.

\[ A(x,y) \Join B(y,z) = (\pi_{A(x,y)} A(x,y) ) \cup (\pi_{B(y,z)} B(y,z)) \]  

Unlike traditional, set-based union, there is no requirement that relational operand signatures should match.

**Selection:** selects a subset of tuples (records) from a relation (table) which satisfy some logical predicate.

\[ \sigma_{p(x,y)} A(x,y,z) = A(x,y,z) \Join P(x,y) \]  

**Projection:** is generalized union between $A$ and empty relation $Y(y)$

\[ \pi_{r} A(x,y) = A(x,y) \Join Y(y) \]  

The projection operator selects a subset of attribute.

A projection is a unary operation written as $\pi_{a_1,\ldots,a_n}(R)$ as where $a_1,\ldots,a_n$ is a set of attribute names. The result of such projection is defined as the set that is obtained when all tuples in $R$ are restricted to the set $\{a_1,\ldots,a_n\}$. 
**Data vector:** represents a set of scalar values grouped together representing an entity of interest.

Let \( D = \{ d_i \} \forall i \in N \) where \( d_i \) is a scalar and \( N \) is integer.

**Cluster Center:**

Let \( C = \{ c_i \} \forall i \in N \) where \( c_i \) is a scalar and \( N \) is an integer. \( C \) has the same structure as \( D \) as defined in Data vector. ‘\( C \)’ is also called as the ‘means’ in the K-means algorithm.

**Distance:**

Let \( \partial = C \otimes D \) where \( C \) is a cluster center vector, \( D \) is data vector and \( \otimes \) is a similarity/ distance operator defined over \( C \) and \( D \). \( \otimes \) Can take various forms including numeric and non-numeric similarity/distance functions.

Table and Clusters: basic input for many clustering algorithm is the table structure. A table structure is a data set containing \( N \) vectors having \( C \)(table X) or \( D \) (table Y) structures. Cluster is subset of table with group of objects.

<table>
<thead>
<tr>
<th>tuple #1</th>
<th>tuple #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>tuple #2</td>
<td>tuple #2</td>
</tr>
<tr>
<td>tuple #3</td>
<td>tuple #3</td>
</tr>
<tr>
<td>tuple #4</td>
<td>tuple #K</td>
</tr>
<tr>
<td>tuple #N</td>
<td></td>
</tr>
</tbody>
</table>

Table X (data)  Table Y (clusters)

It is assumed that, Number of clusters is less than the number of records \( (K < N) \).

### 4.4 K-Means Clustering

#### 4.4.1 The K-means Method
The K-means algorithm is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The K-Means clustering algorithm has been discovered and rediscovered by researchers in different fields many times.

The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define ‘k’ centroids, one for each cluster. These centroids should be selected randomly, since different locations causes different results. The better choice is to place them as much as possible far away from each other. The working of K-means is as shown in the figure 4.1 to 4.4.

![Figure 4.1: K initial "means"](image)

*Figure 4.1: K initial "means"* (in this case k=3) are randomly selected from the data set (shown in color).

The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done.
Figure 4.2: K clusters are created by associating every observation with the nearest mean.

Figure 4.3: The Centroid of each of the K clusters becomes the new means.

At this point we need to re-calculate K new centroids as bary centers of the clusters resulting from the previous step. After we have these K new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that
the K centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

![Figure 4.4: Steps 2 and 3 are repeated until convergence has been reached](image)

Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \]

where \( ||x_i^{(j)} - c_j||^2 \) is a chosen distance measure between a data point \( x_i^{(j)} \) and the cluster centre \( c_j \), is an indicator of the distance of the n data points from their respective cluster centers.

Let ‘distortion’ be the distance aggregate

\[ \text{Distortion} = \sum_{i=1}^{R} (x_i - c_j)^2 \] ................................. (1)
Now the problem is to minimize this aggregate as

$$\frac{\partial \text{Distortion}}{\partial c_j} = \frac{\partial \sum (x_i - c_j)^2}{\partial c_j \text{ i} \in 0 \text{ owned by } (c_j)}$$

$$= 0 \text{ (for a minimum)} \quad (2)$$

The heuristic solution for (2) algorithm 5.1 is given by Forgey [FO65] and is given below.

**Algorithm 4.1: Steps of Classical K-means**

- Calculate initial means.
- Assign objects into clusters by using initial means.
- Do while objects move to another clusters
  - Recalculate means of clusters by using objects belonging to them.
  - Assign objects into clusters by using calculated means.
- End of while (Convergence of the algorithm).

The details of the functions required for implementing a K-means algorithm are discussed in the following paragraphs. It consists of the steps as mentioned in the discussion above and the flow chart in figure 4.5.
As stated earlier, K-means algorithm takes initial means as input. Then it iterates and updates the means in each iteration. Each updates to means in iterations makes those means closer to final means. This is why K-means algorithm converges after a number of iterations. Initial means and produced subsequent means are used to assign objects into clusters. Initially, objects are assigned into clusters that have the nearest mean to them by using initial means which are supplied to the algorithm as input. This is the first iteration of the algorithm.

When all objects are assigned into clusters, cluster means are recalculated by using the objects in the clusters. These means are supposed to be closer to final
means when compared with initial ones. Next all objects are reassigned to clusters by using new means.

This is the conclusion of second iteration. Probably, some objects will move to different clusters when using new means considering their clusters with the previous means. These iterations of K-means algorithm continues until no object moves to another cluster between the iterations. This converge the algorithm.

Based on the computational details functions discussed above, a Pseudo code of the K-means algorithm is as shown in the Algorithm 4.2

It can be observed that steps of K-means algorithm are very simple and straightforward, but the task performed is great as a result of the iterative structure of algorithm. Simplicity, execution in linear time and the ability of successful clustering makes the K-means algorithm, one of the most common clustering algorithms in different domains [HA01].

Although it can be proved that the procedure will always terminate [HA01], the K-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum
Algorithm 4.2: Pseudo code for K-means

```plaintext
begin
    m[] = SelectInitialCentroids(D,k);
    repeat
        for j = 1 to k do
            s[j] = (0,0,0, . . . ,0); // s[] – family of vectors of size dim(D)
            num[j] = 0; // num[]- number of points in each cluster
            d[j] = 0; //cl[] – actual clusters
        end for
        for each x € D do
            cluster = argmin_{j=1,...,k} (dist(x,m_j)); // assign x to the cluster
            d[cluster] = d[cluster] U {x};  
            s[j] = s[j] + x;
            num[j] = num[j] + 1;
        end for
        for j := 1 to k do
            m[j] = \frac{s[j]}{num[j]};
        end for
        until isStoppingCondition(m[], d[]) = true;
    output d[];
    isStoppingCondition( m[],d[])
    for i = 1 to length [m] do
        diff = | m_i – d_i |
        if diff < threshold
            return true
        end if
    end for.
end
```

The algorithm is significantly sensitive to the initial randomly selected cluster centers [HA02]. This can be mitigated to some level by multiple runs of K-means along with different cluster centers of similar results, can be taken as approximately correct cluster center. The k-means algorithm can be run multiple times to reduce this effect.

### 4.4.2 Complexity analysis of K-means algorithm

**Time Complexity:** $O(mnk)$ ($n = |D|$, $k$ – number of clusters, $m$ – number of iterations)
In each iteration the time complexity is $O(kn)$ where $K = \text{number of clusters or centroids}$ and $n = \text{number of objects}$.

If total number of iterations in the worst case are $m$, then the upper bound or worst case complexity is $T(n) = O(mkn)$.

Consider $n$ records and $k$ clusters. As discussed in the algorithm every record in the data is associated with each of the cluster centers. This give rise to $O(nk)$ time complexity for one iteration. When this process is continued for $m$ iterations the time complexity will be increased to $O(mnk)$.

The algorithm attempts to determine $k$ partitions that minimize the squared error function. It works well when the clusters are compact clouds that are rather well separated from one another. The method is relatively scalable and efficient in processing large datasets because the computational complexity of the algorithm is $O(mkn)$, where $n$ is the total number of objects, $k$ is the number of clusters and $m$ is the number of iterations. Normally $k < n$ and $m < n$. The method often terminates at a local optimum. Fortunately, since the distance metric and the distortion function are very subjective, it makes sense to use heuristics like early
termination with maximum number of iteration and/or termination after achieving an acceptable threshold to achieve sub-optimal solutions.

**Data Complexity:** \( m \cdot B(D) \) where \( B(D) \) – number of disk blocks in holding \( D \) and \( m \) is number of iterations

**Space Complexity:** \( O((n + K)m) \) where \( n \) is the total number of objects, \( K \) is the number of clusters or centroids and \( m \) is the number of iterations.

Table 4.2 shows different clustering algorithms and their time and space complexities, concludes from table K-means algorithm is best suits for multidimensional data.

### Table 4.2 Computational Complexity of Clustering Algorithms

<table>
<thead>
<tr>
<th>Cluster Algorithm</th>
<th>Complexity</th>
<th>Handling high dimensional data</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means</td>
<td>( O(NKm) ) (time) ( O(N+K) ) (space)</td>
<td>Yes</td>
</tr>
<tr>
<td>Fuzzy c-means</td>
<td>Near ( O(N) )</td>
<td>No</td>
</tr>
<tr>
<td>Hierarchical clustering</td>
<td>( O(N^2) ) (time) ( O(N^2) ) (space)</td>
<td>No</td>
</tr>
<tr>
<td>CLARA</td>
<td>( O(K(40 + K)^2 + K(N - K)) ) (time)</td>
<td>No</td>
</tr>
<tr>
<td>CLARANS</td>
<td>Quadratic in total performance</td>
<td>No</td>
</tr>
<tr>
<td>BIRCH</td>
<td>( O(N) ) (time)</td>
<td>Yes</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>( O(N \log N) ) (time)</td>
<td>Yes</td>
</tr>
<tr>
<td>CURE</td>
<td>( O(N_{sample}^2 \log N_{sample}) ) (time) ( O(N_{sample}) ) (space)</td>
<td>Yes</td>
</tr>
<tr>
<td>Wave Cluster</td>
<td>( O(N) ) (time)</td>
<td>No</td>
</tr>
<tr>
<td>DENCLUE</td>
<td>( O(N \log N) ) (time)</td>
<td>Yes</td>
</tr>
<tr>
<td>FC</td>
<td>( O(N) ) (time)</td>
<td>Yes</td>
</tr>
<tr>
<td>CLIQUE</td>
<td>Linear with number of objects, Quadratic with the number of</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### 4.4.3 Remarks

This is a simple version of the k-means procedure. It can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some limitations:

- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples. In this research, a seed selection algorithm based on median projection is proposed.

- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points. As the median projection is theoretically provides almost optimal cluster centers, this problem is mitigated to a large extent.

- The results depend on the value of K. Since, the proposed algorithm provides optimal clusters possible, if ‘k’ is less than this theoretical maximum, then ‘K’ means can be selected from maximum number of clusters possible. If ‘k’ is greater than maximum clusters possible, the algorithm reduces ‘k’ to maximum number of clusters possible recommended by the proposed algorithm.

The last problem is particularly troublesome, since we often have no way of knowing how many clusters exist. In the example shown below, the same
algorithm applied to the same data produces the following 3-means clustering. It is difficult to decide whether it is better or worse than the 2-means clustering.

Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion. We need to be careful because increasing k results in smaller error function values by definition, but also an increasing risk of over fitting. An approach to address this problem is proposed in the chapter 5.

4.5 K-means Using SQL

4.5.1 Introduction

There are three reasons for implementing k-means algorithm in SQL. First, most of the implementations for k-means in a procedural language like C, C++ or Java. Using SQL makes writing data mining algorithm easier, since SQL is declarative and there is no need to handle data at the lowest level.

Second, these algorithms assume that all the data can fit into memory. This creates a problem, when the data to be explored is larger than the available memory, which in reality is always true. Even though some clever algorithms use sampling or incremental learning, the ability to handle large
data is still a problem with these algorithms.

Third, building the cluster model is only one part of the problem. Accessing the data from the sources like a data warehouse and transforming them outside the database is another major problem. Using SQL this problem can be minimized, since the data will be analyzed within the database, without the need for additional overheads.

In this research an attempt has been made to implement k-means algorithm using SQL for in-database analysis of the large datasets.

4.5.2 Data Structures Used

The different data structures (tables) used in implementing the SQL K-means are given in table 4.3.

<table>
<thead>
<tr>
<th>Table</th>
<th>Primary Key</th>
<th>Columns</th>
<th>No. of rows</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>RID</td>
<td>$y_1, y_2, \ldots, y_p$</td>
<td>N</td>
<td>Records</td>
</tr>
<tr>
<td>C</td>
<td>CID</td>
<td>$y_1, y_2, \ldots, y_p$</td>
<td>K</td>
<td>Cluster centers</td>
</tr>
<tr>
<td>YD</td>
<td>RID</td>
<td>$d_1, d_2, \ldots, d_p$</td>
<td>KN</td>
<td>Distances</td>
</tr>
<tr>
<td>MD</td>
<td>RID</td>
<td>D</td>
<td>N</td>
<td>Minimum distance</td>
</tr>
<tr>
<td>C2</td>
<td>None</td>
<td>Cid, Rid</td>
<td>N</td>
<td>Classified records</td>
</tr>
<tr>
<td>CN</td>
<td>CID</td>
<td>$y_1, y_2, \ldots, y_p$</td>
<td>K</td>
<td>New cluster centers</td>
</tr>
</tbody>
</table>

The table Y is used to store the data and it is the table where the input is
The table C is used to store the cluster centers, which are generated. The table YD is used to store the distances. In table D the cluster id and minimum distance is stored. The table C2 is temporary table to store row id and cluster id. The new cluster centers are computed and stored in table CN. Now the values in CN and C are compared. If they are different the process is repeated till the convergence otherwise the process is terminated.

4.5.3 Implementation Strategy

The strategy for implementing SQL K-means is as shown in the algorithm 4.3 and the process flow is shown in figure 4.6.

The strategy to implement is as follows:

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 distance 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 CN and C are different (based on chosen criteria), then the process is repeated till convergence. Higher threshold leads to faster convergence and lower threshold leads to slower convergence.
Algorithm 4.3: Implementation Strategy for SQL K-means

1. Select the number of clusters randomly
   - Top K records.
   - Filter with Order by.
   - Filter with MOD operator.
2. Compute distance matrix
   - Join records and clusters table to compute distances.
   - Select the one cluster per record.
3. Check the movement of the points
   - re-compute new cluster centers.
   - check the difference between old and new centers.
4. Repeat the steps 2 and 3 till the difference is less than threshold.
The queries used for implementing the K-means are given in algorithm 4.5.4. The code shown is only for one attribute to demonstrate the compactness and readability of the code. However, the concept can be generalized to accommodate any number of attributes as restricted by the underlying RDBMS.

4.5.3.1 Seed the Cluster Centers:

One of the important steps in building clustering algorithm specifically the K-means clustering algorithm is to seed the cluster centers. Since we are implementing the K-means in SQL, different available functions of RDBMS are utilized. Using these functions it is possible for us to provide randomize records for the cluster centers. Here numbers of clusters are selected randomly.

Sampling is used to estimate the results of the aggregated queries, to retrieve a sample of records from a database query for subsequent processing, for internal use of the query optimizer and to provide privacy protection for records in the database.

A random sampling query returns a random sample i.e., a randomly selected subset of the results of a relational retrieval query. In general, the sampling operator might appear anywhere in a nested query where a SELECT could appear.

To begin with we have to identify top K records. There are three methods for the same.
Table 4.4 Selecting Random records.

<table>
<thead>
<tr>
<th>Database</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonetDb</td>
<td>SELECT column k from table ORDER BY Random()</td>
</tr>
</tbody>
</table>
| MySQL       | ```
SELECT ALL FROM TABLE WHERE ROW NUMBER < K
  select top k from table
SELECT column FROM table ORDER BY RAND( ) LIMIT 1
SELECT column FROM ( SELECT column FROM table
  ORDER BY dbms_random.value) WHERE rownum = 1.
``` |
| LuCidDB     | SELECT TOP k column FROM table ORDER BY NEWID()                             |
| SQLite      | SELECT ALL FROM TABLE LIMIT K
  K is number of records or number of clusters. |

Method I

If it is MySQL server we can just say
```
SELECT ALL FROM TABLE WHERE ROW NUMBER < K
```

If it is MySQL/POSTGRE SQL then we can use the command,
```
SELECT ALL FROM TABLE WHERE ROW NUMBER < K records.
```

If it is open source systems like SQLite (memory resident) we can use
```
SELECT ALL FROM TABLE LIMIT K, where K is the number of records
or number of clusters.
```

For example: Consider the table DATA with 100 records
When the following statement is submitted:

```
SELECT * FROM DATA WHERE ROW_ID < 3
```

The result is

<table>
<thead>
<tr>
<th>row_id</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

**Method II**

The method II to select top k records is using filter with ORDER BY. We can do the same thing especially if we know or assume that data is not randomly stored or the records are not randomly stored in the database. It is better to go with ORDER BY most of the times for practical reasons. We go with ORDER BY clause on a particular column which is numeric in nature.

eg. SELECT TOP K records from the table order by some column.
Before selecting the records it sorts all the records. After sorting one may be interested in only first N records, since the sorting of complete records is not useful.

For example

When the following statement is submitted-

```
SELECT VALUE FROM DATA WHERE ROW_ID < 4 ORDER BY VALUE;
```

The result is

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>50</td>
</tr>
</tbody>
</table>

Method III

The Third approach to select top K records is using MOD operator. The MOD operator will basically give us the integer numbers which uses an integer value and modulus of number of records is given as record set.

Eg- We are having 100 records and we are taking MOD of 10 will give us 10 records or record set of 10.

In each such cases it is similar to top K records but the filtering is done through the MOD operator which may be the better method for providing the random numbers.

For example

When the following statement is submitted,

```
SELECT VALUE FROM DATA WHERE ROW_ID ≤ RANDOM()% (SELECT MAX(ROWID) FROM TABLE) LIMIT 4;
```
The result: generates 4 random values for a field VALUE.

Algorithm 4.4 gives the selection of Random centers. Figure 4.7 shows the procedure to perform Centroid re-computation and figure 4.8 shows the stopping criteria.

- Let \( m_{t,1}, \ldots, m_{t,k} \) are the k centroids on step \( t \geq 1 \).
- Let \( C_{t,1}, \ldots, C_{t,k} \) be the k clusters assigned on step \( t \).
- The new centroids \( m_{t+1,1}, \ldots, m_{t+1,k} \) are computed as means of points in their clusters on previous iteration:
  \[
  m_{t+1,i} = \frac{1}{|C_{t,i}|} \sum_{x \in C_{t,i}} x_i
  \]

**Algorithm 4.4: Selecting Random Centers.**

- Pick \( k \) random points from the datasets. (Seeds).
- Use the following selection procedure Select Centroids:
  1. Compute the Centroid \( c \) of the entire dataset \( D \).
  2. First Centroid, \( m_1 \) is the \( x \in D \), such that \( d(m_1,c) = \max_{x \in D} d(x,c) \) (the point further away from the centroid).
  3. Pick \( m_2 \) such that \( d(m_1,m_2) = \max_{x \in D} d(m_1,x) \).
  4. Pick \( m_i \), such that 

The k-means algorithm uses one of the following stopping criteria: (for convergence)

1. No (or minimum) reassignment of points between clusters;
2. No (or minimum) change in cluster centroids;
3. Insignificant decrease in the sum of squared error:

\[
SSE = \sum_{j=1}^{k} \sum_{x \in C_{t,j}} d(x,m_{t,j})
\]
4.5.3.2 Compute distance matrix

- Join records and clusters table to compute distances.
- Select the one cluster per record.

The distance of each data points (records) in data table Y with each of the cluster center from table C are computed using Euclidean distance measures are stored in the table YD. Here the records are inserted into YD table by fetching and comparing the data from both the table Y and C. The code is

```sql
INSERT into YD (Rid, Cid, dist)
( SELECT Y.Rid, C.Cid, abs((C.c1-Y.y1) * (C.c1-Y.y1) + (C.c2-Y.y2) * (C.c2-Y.y2) + . . . (C.cn-Y.yn) * (C.cn-Y.yn))
  FROM Y,C);
```

Select the one cluster per record: The clusters are grouped based on the Cluster id and the cluster with minimum distance is selected and the result is stored in the table MD. The process is repeated for all the clusters.

```sql
INSERT into MD (Rid, dist)
( SELECT Rid, min(dist) FROM YD GROUP BY Rid);
```

4.5.3.3 Check the movement of the points

To re-compute new cluster centers, the two tables are used, namely C2 and CN. 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. The code is

```sql
INSERT into C2 (Rid, Cid)
(SELECT YD.Rid, min(YD.Cid)
from YD, MD where MD.Rid = YD.Rid and
    MD.dist = YD.dist group by MD.Rid);

INSERT into CN (Cid, y1, y2,...,yp)
(SELECT C2.Cid, avg(Y.y1), avg(Y.y2),..., avg(Y.yp)
from Y, C2
where Y.Rid = C2.Rid group by C2.Cid);
```

C2 is a temporary table to store the row id and cluster id. The database C2 is populated by referring table YD and MD. To populate C2, first the records are grouped based on row id of MD (minimum distance entries) and later performing natural join operations using tables MD and YD performing join conditions on distance and row id’s.

Now new cluster centers are computed by performing Natural join operation on temporary Table C2 and original data table Y.

4.5.3.4 Repeating the process.

Compute the difference between old and new cluster centers. If they are same process is terminated otherwise process is repeated till the difference is less than threshold.

If the CN and C are different then the process is repeated till convergence.

```sql
SELECT max(abs(C.y1-CN.y1)),
    max(abs(C.y2-CN.y2)), ..., max(abs(C.yp-CN.yp))
from C, CN where C.Cid = CN.Cid;
```
Clean before repeat the process.

In the next section 4.5.4 the table design is discussed. The work is demonstrated with a sample data.

**4.5.4 Example for SQL Implementation**

The following tables are created to implement K-means algorithm using SQL.

1) **Table Y**: is to store the data. This table is taken as data. It contains 5 records as shown in table Y. The table Y has sample data with row id 1, 2, 3, 4 and 5.

```sql
CREATE TABLE Y (Rid int, Ydata float); // DDL to create the table
INSERT INTO Y VALUES (1,1.0); // DML to populate tables
INSERT INTO Y VALUES (1,2.0);
INSERT INTO Y VALUES (1,3.0);
INSERT INTO Y VALUES (1,4.0);
INSERT INTO Y VALUES (1,5.0);
```

<table>
<thead>
<tr>
<th>Rid</th>
<th>Ydata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
2) **Table C**: Selection of cluster center from Y to C, randomly the cluster centers are selected and stored in a table.

\[
\text{(SELECT Rid, Ydata FROM Y ORDER BY Ydata LIMIT K) INTO C}
\]

If we assume K = 2, we get 2 records in table C. This is only a typical result, but any result is sure to provide convergence with threshold conditions.

CREATE TABLE C (Cid int, c1 float);
Table C

<table>
<thead>
<tr>
<th>Rid</th>
<th>Cid</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5.0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6.0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

3) Table YD: 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 YD.

CREATE TABLE YD (int, Cid int, dist float);

INSERT INTO YD (Rid, Cid, dist) (select Y.Rid, C.Cid, abs (C.c1 – Y.y1) from Y, C);

The records are inserted into YD table by fetching and comparing the data from both the tables Y and C.

Table YD

<table>
<thead>
<tr>
<th>Rid</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
</tr>
</tbody>
</table>
4) **Table MD:** The table with record number and the cluster to which it belongs to is stored in the table i.e., the cluster with minimum distance

```sql
CREATE TABLE MD (Rid int, dist float);
INSERT INTO MD (Rid, dist) (select Rid, min (dist) from YD group by Rid);
```

<table>
<thead>
<tr>
<th>Rid</th>
<th>Cid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

5) **Table C2:** It is a temporary table to store the row id and cluster id

```sql
CREATE TABLE C2 (Rid int, Cid float);
INSERT INTO C2 (Rid, Cid) select YD.Rid, min(Yd.Cid) from YD,MD where MD.Rid = YD.Rid and MD.dist = YD.dist group by MD.Rid;
```

<table>
<thead>
<tr>
<th>Rid</th>
<th>Cid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

6) **Table CN:** New cluster centers are computed by joining table C2 and Y

```sql
CREATE TABLE CN (Cid int, c1 float);
INSERT INTO CN (Cid, c1) (SELECT C2.Cid, avg (Y.Ydata) from Y, C2 where Y.Rid = C2.Cid group by C2.Cid);
```
If the CN and C are different then the process is repeated till convergence.

SELECT MAX (abs (C.c1 – CN.c1)) from C, CN where C.Cid = CN.CID;

<table>
<thead>
<tr>
<th>Max(abs(C.c1-CN.C1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>0.5</td>
</tr>
</tbody>
</table>

Since new cluster centers (CN) and initial clusters (C) are not the same the process is repeated for next iteration.

II Iteration:

1) **Table Y**: the contents are same since it is the actual data.

<table>
<thead>
<tr>
<th>Rid</th>
<th>Ydata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
</tr>
<tr>
<td>4</td>
<td>6.0</td>
</tr>
<tr>
<td>5</td>
<td>7.0</td>
</tr>
</tbody>
</table>

**Table Y**
2) **Table C**: Existing the tuples are deleted and the new cluster center are populated from new cluster table (CN).

```
DELETE FROM C;
INSERT INTO C (Cid, C1) as select Cid, C1 from CN;
```

3) **Table YD**: 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 YD.

```
<table>
<thead>
<tr>
<th>Rid</th>
<th>Cid</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5.0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.5</td>
</tr>
</tbody>
</table>
```

4) **Table MD**: The table with record number and the cluster to which it belongs to is stored in the table i.e. the cluster with minimum distance.
5) **Table C2**: It is a temporary table to store the row id and cluster id

<table>
<thead>
<tr>
<th>Rid</th>
<th>Cid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

6) **Table CN**: New cluster centers are computed by joining table C2 and Y

<table>
<thead>
<tr>
<th>Cid</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>6.5</td>
</tr>
</tbody>
</table>

7) Since the new cluster center (CN) and current cluster center (C) are same. Now objects doesn’t move group anymore. Thus the computation of K-mean clustering has reached its stability and no more iteration is needed. The process terminates.

Now the Output is:  
Cluster1: 1, 2 and 3  
Cluster2: 4, 5

The complete SQL code for a sample implementation is given below in the algorithm 4.5.
This implementation of K-means clustering has resulted in two interesting features.

1) The code complexity and readability has been drastically improved in a declarative language like SQL.

2) The ability to handle large datasets which cannot be fit into memory is transferred to DBMS. This achieves scalability across number of rows.

One of the limitations is scalability across columns which are limited by the DBMS.

The experimental evaluation of the proposed approach is discussed in Chapter 6.

Algorithm 4.5: SQL Code for K-means

```sql
CREATE TABLE Y (Rid int, Ydata float); // DDL to create the table
CREATE TABLE C (Cid int, c1 float) // populate the data
(SELECT Rid , Ydata FROM Y ORDER BY Ydata LIMIT K) INTO C
CREATE TABLE YD( Rid int, Cid int, dist float);
INSERT INTO YD(Rid, Cid, dist) (select Y.Rid, C.Cid,abs(C.c1 – Y.y1) from Y, C);
CREATE TABLE MD (Rid int, dist float);
INSERT INTO MD (Rid, dist) (select Rid, min (dist) from YD group by Rid);
CREATE TABLE C2 (Rid int, Cid float);
INSERT INTO C2(Rid, Cid) select YD.Rid, min(Yd.Cid) from YD,MD where  MD.Rid = YD.Rid and MD.dist = YD.dist group by MD.Rid;
CREATE TABLE CN (Cid int, c1 float);
INSERT INTO CN (Cid, c1) (SELECT C2.Cid, avg (Y.Ydata) from Y, C2 where Y.Rid = C2.Rid group by C2.Cid);
SELECT MAX (abs (C.c1 – CN.c1) from C, CN where C.Cid = CN.CID;
DELETE from C;
DELETE into  C (Cid, y1, y2,..., yp) (select Cid, y1, y2,..., yp from CN);
DELETE from CN;
DELETE from YD;
DELETE from MD;
```
4.6 Conclusion

This implementation of K-means clustering using SQL has resulted in two interesting features.

1) The code complexity and readability has been drastically improved in a declarative language like SQL. This is validated with the code. Compared with the code written in procedural language like C++ and Java the complexity in all the level is reduced.

2) The ability to handle large datasets which cannot be fit into memory is transferred to DBMS. This achieves scalability across number of rows.

Still one of the limitations seems to be scalability across columns which are limited by the DBMS.
CHAPTER 5

MODIFIED SQL K-MEANS ALGORITHM

5.1 Introduction

In this chapter a seeding algorithm to identify the number of clusters and the initial seeds are discussed.

Key Points:

1. In K-means cluster centers will be randomly selected. This leads to suboptimal structure identification latent in the data. This needs proper selection of cluster centers.
2. Prior center selection will empirically reduce the time and space complexities by easy convergence.
3. In numerical data there is a possibility that the Euclidean distance measures which assume spherical clusters may be affected by outliers; there by distorting the structure information, hence new methods for cluster selection and arbitrary shape selection are needed.
4. In this research a novel method of selection of the cluster centers has been proposed.
5. The center promise of our approach is a possible presence of natural median in any dimension which can be easily found with sorting and first differences.

6. It is assumed that probability of any cluster overlap in the median separated space is minimal to improbable.

7. The basic algorithm selects the cluster centers.

It can be observed that K-means algorithm has limitation of need for the number of clusters during the initialization of the algorithm. This however is paradoxical since in an unsupervised learning paradigm, pre-specifying parameters will result in expected structure rather than actual latent structure in the data.

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.

In a simpler sense this is equivalent to using Group by clause of SQL in categorized data. Since our algorithm has to work on numerical data, we are proposing the algorithm to identify median on columns.

Another drawback of K-means algorithm as discussed in the previous chapter is scalability both in terms of number of observations (row) and number of attributes (column). In order to mitigate these limitations, SQL implementation of K-means is proposed as discussed in the previous chapter.

In order to optimize the number of clusters and seed the number of cluster centers the algorithm proposed in this chapter can be used.

5.2 Conceptual basis for the Proposed Algorithm

In order to implement the proposed algorithm for median projection certain
assumptions are made. These assumptions are discussed in detail in this section.

**Assignment 1:** There are at least 2 segments in any one of the columns in the database. i.e., at least one axis has 2 clusters.

**Justification:** If there are no segments possible, i.e., if all records have same value, it can be easily represented by a single record.

Assignment 2: Distance measures will not find natural segments in the data.

**Justification:** If natural segments are known, then seeding and apriory setting of K-means is not correct or required.

How can we force 3 segments when only 2 segments are possible.

**Lemma 3:** Segmentation is possible by projecting medians in multi dimensional spaces.
Justification: Presence of medians will provide natural hierarchy of segments across multidimensional spaces (can provide a natural dendogram).

Lemma 4: Working on individual columns retaining the record identity will provide extending scope for parallel implementation.

Justification: It is optimal to perform the operations on individual columns.

5.3 Cluster Seeding

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

5.3.1 Median Search method

The procedure for median search is shown in the algorithm 5.1

**Algorithm 5.1: Median Search**

- 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.

In the median search the consecutive differences between any 2 data points in the column is taken as a possible median point. i.e., where a cut can be made so that portions on both sides of the median constitutes a cluster. In order to do this it is sufficient if we consider a single column at a time, the actual
procedure consists of few simple and sequential steps as discussed below.

In the first step, the column data with positional identity [record integrity, retained with record id] is sorted in the ascending or descending order. In this research sorting in descending order of a magnitude of the indexed value of the column has been used. The sorting can be from Brute force Search algorithm which may have time complexity of \(O(n^2)\). The sophisticated sort algorithm like Quick sort takes a time complexity of \(O(n \log n)\). However in this research since the sorting is implemented in DBMS, we may not have flexibility to choose the sorting algorithm. Hence any algorithm provided by DBMS is taken as unit time complexity at an abstraction level and will be implemented using SQL.

In the second step the differences between consecutive points of the sorted values in the column will be computed. Euclidean distance computations are used since they are simple. This can be easily implemented using self join with the same table or with simple column addition and insertion of the column data into new column.

In the next stage the computed difference between two successive points is ordered in descending order of the magnitude. From this list of ordered differences, top 5% of the values are selected as medians and compared with incremental sample of 6%. If the median values are not changing between 5% and 6%, the median value obtained in the 5% will be set as the cut of points for developing the clusters. In case of change in the median values between samples, similar incremental samples will be experimented till we reach 10% of the sample. For every new iteration the previous set of medians will be computed with new set of medians and an exit criterion is checked. Currently we are setting the upper
limit of the sample size to be 10% heuristically. If there is no upper limit set for exit criteria, there is every possibility that the algorithm will end up with segments having single record. Further, in case of similar median values in the frozen list of median set, one of them will be taken as representative for that group, which is usually last one. The median search algorithm is presented in the figure 5.1. The ability of the column store DBMS has been exploited in this research in an elegant way.

### 5.3.2 Median Projection method

For each column mark the medians as given in median search algorithm 5.2.

**Algorithm 5.2: Median Projection**

- 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.

The above two algorithms 5.1 and 5.2 are integrated as follows.

**Algorithm 5.3: Modified Median Projection method**
Once the medians for all the columns in the database are estimated, they can be projected across to get a better idea of number of possible clusters. In order to do this the number of medians in each column will be counted. The column with highest number of medians is most likely to give micro clusters and can be selected and projected across other column to provide basic set of means or cluster centers.

If the number of clusters in K-means algorithm is less than the possible number of clusters in the data [Clue provided by the proposed algorithm], the K-means are randomly selected from the list of medians provided by the projection algorithm. On the other hand if the number of clusters is more than what is possible, then the value of K is set to maximum number of clusters that are possible and the K-means are initialized with the list derived from the proposed algorithm.

- Select and sort a column.
- Compute the difference between two successive points.
- Find the statistical validity of larger values with smaller values.
- If significant difference exists mark it as a median.
- Repeat step1 to step4 for all columns.
- Order columns by decreasing number of medians (from left to right.
- Project the medians in the most varied column as number of clusters.
- For each median block compute average and set it as the seed.
The actual algorithm for Median Projection is given in figure 5.2. Using the Median search and Projection algorithm the complete median search algorithm for identification of clusters in a given dataset is given in the succeeding section. The SQL code to implement Median Projection algorithm is shown in the algorithm 5.4.

The median projection algorithm can be applied for multiple columns in parallel.

5.4 Demonstration of Median Projection Algorithm

In the following sections median projection algorithm is explained for single and two dimensional data by considering sample data set.

5.4.1 Single Dimension:

The following steps demonstrate the working of Median projection method for Single Dimension for sample data.

1. Consider the Data Vector 6 1 3 5 2 7

<table>
<thead>
<tr>
<th>Rid</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Now if data can be plotted graphically it looks like a vector as shown below.
2. The difference between 2 consecutive points in the vector is computed.

<table>
<thead>
<tr>
<th>Rid</th>
<th>C1</th>
<th>C2</th>
<th>diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

It can be visualized graphically as shown below:

3. Now if the differences are ordered, we get a vector as shown below:

```
3-4  2-5  5-3  4-1  1-6
```
4) If the values in the above vector are tested for equality or togetherness with some statistical test, we can validate both the number of clusters possible in the data and belongingness of each data point into one of the clusters.

For example: now let us make a null hypothesis $H_0$: Difference between records 3-4(2) and 1-3(1) are same and the alternative hypothesis $H_1$: difference is not same. This can be easily validated and null hypothesis is rejected. Therefore there is a possibility that the difference between the records 3 and 4 is a possible median.

Now all the records to the left of the median will form one cluster and the right of the median will form another cluster. This procedure automatically identifies the latent structure without complicated computations.

Logically the 2 clusters formed are whose means are 2 and 6 respectively.

5.4.2 Two Dimension:

The following steps demonstrate the working of Median projection
method for Two Dimension for sample data.

1) Data vector

<table>
<thead>
<tr>
<th>Rid</th>
<th>X1</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

1. Sorted data Vector is

<table>
<thead>
<tr>
<th>Rid</th>
<th>X1</th>
<th>Rid</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Difference between Successive Points are as follows for X1

<table>
<thead>
<tr>
<th>Rid</th>
<th>C1</th>
<th>C2</th>
<th>diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

0 1 2 3 4 5 6 7
3. Difference between Successive Points are as follows for X2

<table>
<thead>
<tr>
<th>Rid</th>
<th>C1</th>
<th>C2</th>
<th>diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

4. Search for larger differences

1 1 2 1 1
Ordered differences are

5. Possible cuts

\[
\begin{array}{cccc}
1 & 2 & 5 & 6
\end{array}
\]

6. Cluster Profile

Number of Clusters : 2
Cluster Centers : \{(x,2), (y,6)\}
\{(2,2), (1,6)\}

The graphical representation of cluster centers is

5.5 Median Projection Algorithm using SQL

**Algorithm: 5.4 Median Projection using SQL**

//Sort the column in descending order and store in a table T1
SELECT C1 INTO T1 ORDER BY C1 DESC

//Modify the table t1 by adding the column DIFF to store the difference value of two columns.
ALTER TABLE T1 ADD COLUMN (DIFF INT)

//Insert the differences between successive values in the column
INSERT INTO T1 VALUES (C2) ( SELECT Temp1.Rid, Temp2.Rid, ABS((Temp1.C1-Temp2.C1)) from T1 temp1, T1 temp2
where temp1.Rid = temp2.Rid+1;
5.6 Conclusion

The empirical evaluation suggests the cluster quality of the proposed method is superior to ordinary K-means algorithm. The proposed algorithm utilizes the inherent strength of modern database systems mainly the column stored databases by proper use of the sorting and the first difference selection.

The center promise of our approach is a possible presence of natural median in any dimension which can be easily found with sorting and first differences.

It is assumed that probability of any cluster overlap in the median separated space is minimal to improbable. In order to optimize the number of clusters and seed the number of cluster centers the algorithm developed is useful.

In the present study a novel method of selection of the cluster centers has been proved. The median projection algorithm can be applied for multiple columns in parallel.
CHAPTER 6

RESULTS AND ANALYSIS

6.1 Data

Data for the exploitation has been developed using random number generator with three well defined clusters. The datasets contains six columns.

Two different approaches to dataset creation have been developed.

1) In the first approach, the number of columns is fixed and only different number of records of the order 10K, 100K and 1000K with three 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 number of columns are varied. This is used to validate the proposed algorithm on horizontal scalability.

6.2 Validation

The proposed implementation is expected to perform linearly on increase of number of records, 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 time for different data sizes on standard K-means and
modified K-means are analyzed.

6.3 Platforms

The experiments were conducted on a Linux based desktop system with 2 GB RAM. The DBMS used was given in figure 4. The SQL code generation was done in Python and the connection to DBMS was done through JDBC library. Synthetic data used in the experimentation, was generated using IBM data generator. In the experimentation the number of attributes, number of records and number of cluster are varied to build the performance profile of the proposed algorithm.

<table>
<thead>
<tr>
<th>Processing Approach</th>
<th>Disk</th>
<th>MySQL</th>
<th>Lucid DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>SQLite</td>
<td>MonetDB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Row store</td>
<td>Column store</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.1: DBMS used in the Research

6.4 Results and Discussions

One of the major limitations of K-means implementations in other languages is management of the large datasets. Implementation of the clustering algorithm in SQL can be delighted to the DBMS itself, which is designed to handle massive datasets. It can be observed from table 6.1 and figure 6.2, runtime of the K-means algorithm using SQL is linear with respect to the datasets.
Table 6.1: Runtime for K-means

<table>
<thead>
<tr>
<th>Data size</th>
<th>Number of clusters</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>400</td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>1450</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3050</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3850</td>
</tr>
</tbody>
</table>
The experiments are conducted on scalability of the proposed algorithm as the data size increases and number of clusters increases. It can be observed from figure 6.2, the scalability of K-means which is linear is maintained in the SQL implementation also. Further it can be observed that the cluster quality improves as the number of clusters increases. This confirms that the proposed algorithm is also preserving the quality of the clusters. The run time for different data sizes is linear. This is relevant since not much difference has been observed, when the data size increases. If the same has been implemented in a procedural language we could have expected above linear times due to data handling overheads.

In order to establish the effectiveness of the proposed algorithm against the standard K-means algorithm we have chosen inter-cluster distance as criteria. The inter cluster distance for K-means are shown in the table 6.2 and represented in the figure 6.3.
## Table 6.2: Inter cluster distance of K-means

<table>
<thead>
<tr>
<th>Data size</th>
<th>Number of clusters</th>
<th>Inter Cluster distance</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>2</td>
<td>124</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>110</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>102</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>92</td>
<td>37</td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>118</td>
<td>142</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>106</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>101</td>
<td>306</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>91</td>
<td>378</td>
</tr>
<tr>
<td>1000K</td>
<td>2</td>
<td>135</td>
<td>1430</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>114</td>
<td>2300</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>104</td>
<td>3100</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>96</td>
<td>3760</td>
</tr>
</tbody>
</table>
The inter cluster distance for K-means with Modified K-means are shown in the table 6.3 and represented in the figure 6.4.

The modified K-means algorithm which is used for initializing the cluster centers using median projection is providing satisfactory results. It can be observed that whenever the number of clusters is less than the maximum possible number of clusters the proposed algorithm provides better quality clusters. This is clearly given in table 6.3.

Table 6.3: Inter cluster distance for K-means with Modified K-means
<table>
<thead>
<tr>
<th>Data size</th>
<th>Number of Clusters</th>
<th>K-means</th>
<th>Modified K-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000K</td>
<td>2</td>
<td>135</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>114</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>118</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>106</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>2</td>
<td>124</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>110</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>92</td>
<td></td>
</tr>
</tbody>
</table>
Figure 6.4: Comparison of Inter cluster distance of K-means with modified K-means

The inter cluster distances which measure the quality of clusters also seem to be improved. The proposed algorithm provides initial cluster centers through median projection, appears to improve the performance significantly.

It can also be observed that the runtime characteristics of SQL K-means are retained in the modified approach also. This is shown in table 6.4 and figure 6.5.

One of the important observations in this data is the relative run time of the standard K-means and modified K-means [both in SQL]. It is worthy to note that the modified version is comparatively faster. This can be explained by reduced number of iterations due to pre-seeding property of the modified algorithm.
## Table 6.4: Runtime Comparisons of K-means with Modified K-means

<table>
<thead>
<tr>
<th>Data size</th>
<th>No of Clusters</th>
<th>Run time</th>
<th>K-means</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000K</td>
<td>2</td>
<td>1550</td>
<td>700</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2400</td>
<td>1750</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>150</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>250</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>350</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>2</td>
<td>30</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Another interesting observation from the experiments is the ability of the proposed algorithm to restrict the number of clusters to maximum possible. Therefore, the algorithm is behaving as expected when it will not provide the details for the $4^{th}$ and $5^{th}$ clusters when K is set to 5, but data contains only 3 clusters.

Further interesting behavior of the proposed algorithm can be observed in the table 6.5 and figure 6.6. As the number of clusters increases the runtime also increases linearly. This is beneficial when large numbers of clusters are expected in the analysis.

<table>
<thead>
<tr>
<th>Number of Columns</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000K</td>
<td>2</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td>100K</td>
<td>1000</td>
</tr>
<tr>
<td>10K</td>
<td>1500</td>
</tr>
<tr>
<td>5000</td>
<td>3000</td>
</tr>
<tr>
<td>15000</td>
<td>5000</td>
</tr>
<tr>
<td>20000</td>
<td>7000</td>
</tr>
<tr>
<td>25000</td>
<td>9000</td>
</tr>
<tr>
<td>30000</td>
<td>11000</td>
</tr>
<tr>
<td>35000</td>
<td>13000</td>
</tr>
<tr>
<td>40000</td>
<td>15000</td>
</tr>
</tbody>
</table>

**Figure 6.5: Runtime Comparison of K-means with Modified K-means**

Table 6.5: Runtime for K-means with varying number of columns
It is worth noting from figure 6.4 and 6.5 is non-availability of the results for the proposed algorithm in certain parameter settings. For example, when the number of cluster is set to 4 or 5 there is no results for the proposed algorithm. 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

<table>
<thead>
<tr>
<th>No of clusters</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>45</td>
</tr>
<tr>
<td>20</td>
<td>58</td>
</tr>
<tr>
<td>25</td>
<td>71</td>
</tr>
<tr>
<td>30</td>
<td>82</td>
</tr>
</tbody>
</table>

**Figure 6.6: Runtime for K-means with varying number of columns**

It is worth noting from figure 6.4 and 6.5 is non-availability of the results for the proposed algorithm in certain parameter settings. For example, when the number of cluster is set to 4 or 5 there is no results for the proposed algorithm. 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
It is observed from figure 6.6, SQL based clustering algorithm scales linearly with number of data points limited.

Another major result of the research is comparison of row store and column store version of databases [DBMS] in in-memory and disk based storage modes. The results of running SQL Clustering algorithms on different databases are shown in table 6.6. It can be observed that there is not much difference between the performance of row store and column store versions. However, column store has shown superiority with respect to the selection of column, aggregation results for re-computing the cluster centers. Further the study is required to implement better algorithm in column store DBMS.

The proposed algorithm provides initial cluster centers through median projection, appears to improve the performance significantly.

<table>
<thead>
<tr>
<th>Data size</th>
<th>Number of clusters</th>
<th>Row store</th>
<th>Column store</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SQLite</td>
<td>MySQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MonetDB</td>
<td>LucidDB</td>
</tr>
</tbody>
</table>

Table 6.6: Results of running SQL clustering on different databases
### 6.5 Conclusion:

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 standard K-means algorithm implemented without seeding the initial cluster centers. This can be attributed to better convergence with minimum number of

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>13</th>
<th>14</th>
<th>14</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>20</td>
<td>22</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>37</td>
<td>37</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>100K</td>
<td>2</td>
<td>142</td>
<td>142</td>
<td>133</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>225</td>
<td>225</td>
<td>220</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>306</td>
<td>306</td>
<td>304</td>
<td>305</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>378</td>
<td>378</td>
<td>365</td>
<td>370</td>
</tr>
<tr>
<td>1000K</td>
<td>2</td>
<td>1430</td>
<td>1430</td>
<td>1440</td>
<td>1435</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2300</td>
<td>2300</td>
<td>2310</td>
<td>2300</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3100</td>
<td>3100</td>
<td>3080</td>
<td>3084</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3760</td>
<td>3760</td>
<td>3600</td>
<td>3655</td>
</tr>
</tbody>
</table>
passes over datasets. Further median projection algorithm seems to work on the synthetic datasets used in the research.
CHAPTER 7

SEGMENTATION AND PROFILING
CHURN BEHAVIOR IN TELECOM – A CASE STUDY

In this chapter a case study on real world data set to test and demonstrate the proposed algorithm will be discussed.

7.1 INTRODUCTION

Today's businesses have sophisticated data analysis requirements. The metrics or analyses of a business data can be difficult to obtain. To calculate a meaningful metric, business analysts often use spreadsheets to manually analyze data. Manual analysis is tedious and time-consuming process. Most applications fail to deliver useful metrics that provide unique insights into an organization's performance. Useful metrics highlight significant performance measures of the business. Typically, business analysts must execute multiple queries and other time-consuming manual interventions to produce these metrics. Despite the time-consuming effort analysts must start the process from start to obtain follow-up information such as an explanation of a particular anomaly in a metric.
Business data is stored on a database/s. These databases are operated with associated database servers, which manage the storage and retrieval of records from the databases. Analytical servers have additionally been provided to format database queries or information requests sent from a client user interface to the database server for handling. The analytical servers are used to improve the efficiency of the database accesses and to provide metrics of interest to the user from the retrieved records from the database.

One of the major approaches to business analytics is to identify the impending change in trend before it accelerates. This kind of early warning systems is necessary and will be useful in various scenarios like trading financial securities, predicting sales performance, analyzing the churn, etc.

In this work an attempt has been made to develop a system for analyzing the multidimensional data, ranking the time series and predicting the time series to the future. The framework consists of a cross tab query to provide the transposed time series data for selected dimensions, similarity search module which will rank the time series data and a prediction module based on probabilistic forecasting model.

K-means Clustering algorithm is defined by data or derived by data. In some cases we need a clustering algorithm based on user driven i.e., clustering technique using in-database capabilities.

The need for user driven clustering method is to segment and rank historic and predictive information for user defined segmentation. An example of this is performance of retention programs across regions or products or dealers in a telecom company. It requires the projection of churn trend across different
segments defined by user to identify pockets of interest. In this research an attempt has been made to provide such a system.

7.2 ARCHITECTURE

The proposed framework consists of three major components:

- An analytical query system based on an advanced column store database system, which provides the aggregated time series data from a star schema data mart.
- A time series ranking module which ranks the time series with an adoptive algorithm.
- A prediction module, which provides a simple but effective parametric model building capabilities.

A simplified architecture diagram of the proposed system is shown in figure 7.1. These components are integrated on a common platform of technologies-SQL. The integration of the components is done through java/python.

7.2.1 Analytical Query System

There has been a significant amount of excitement and work on column-oriented database systems.
These database systems have been shown to perform more than an order of magnitude better than traditional row-oriented database systems on analytical workloads such as those found in data warehouses, decision support and business intelligence applications [AB08]. In fact there are arguments against the multi dimensional data cube approach used by some vendors, favoring column store databases. One of the arguments is that column store databases provide near full support for SQL. In this research work an open source column store database called MonetDB [MA01] is used as analytical server database.

It is assumed that the data model for the analytical database follows the star schema due to several reasons stated by Kimball et.al [KI04]. This type of schema
de-normalizes the dimension tables to facilitate the faster joins on the queries, reducing the latency. Time is defined as a permanent dimension in the schema to facilitate creation of time series data against user selected dimension list.

The major function in the query system is the SQL query model for a cross tab. This function will be executed by the DBMS and provides the result set for further analysis.

A typical cross tab provides aggregated factual values for selected dimensions in the required format. Generally, the cross tab is a tuple \((X_i, Y_j, F_k)\) where \(X_i\) are dimensional values plotted on rows of a spreadsheet, \(Y_j\) are the dimensional values plotted on columns of a spreadsheet and \(F_k\) are aggregated/functional factual values constituting the row column interaction i.e. the cells. Several commercial systems provide some means of generating cross tabs from the data.

In this research \(X_i\) is allowed to take theoretically unlimited number of dimensions (practically limited to six) and time dimension is fixed as \(Y_j\). User selected facts are aggregated as \(F_k\) values.

### 7.2.2 Ranking Time Series

Once the time series data for the user defined criteria is extracted from the query system, it will be processed by the time series ranking module. The trend i.e., time series can have some prominent patterns which are of interest to business analysts. Some of them are like:

1. Vary considerably over the past few periods
2. Increase greatly
3. Drop drastically
4. Increase greatly and then drop drastically
5. Perform differently than the total trend

A typical comparison of the time series is given in figure 7.2. Points 1 through 4 basically translate into the statistical spread of a time-series values around the mean of the respective time series. Point 5 needs comparative or reference data.

![Typical patterns in time series trends](image)

**Figure 7.2: Typical patterns in time series trends**

A simple pattern based approach has been developed in this research work to compare the time series data. The ranking of time series is done through automated sorting of patterns. In order to sort the time series values, the spread of each series is computed and compared with the spread of all the series. Large variances suggest a very different development, while small variances indicate a similar development pattern.
Algorithm 7.1: Computing the time series rank

1. For each series
   i. Compute the mean value of the series.
      \[ X_{im} = \frac{\sum X_{in}}{n} \]
      Where \( X_{im} \) = mean of the series \( X_i \)
      \( X_{in} \) = \( n^{th} \) value of the series \( X_i \)
   ii. Compute the normalized time series
2. Compute the average for the set of time series
   \[ X_{ij} = \frac{X_{in}}{X_{im}} \]
   Where \( X_{ij} \) is the normalized value of series \( I \)
   such that \( j=n \) for all \( n(0,n) \)
   \( X_{in} \) = actual value of index \( n \) of series \( i \)
   \( X_{im} \) = mean value of series \( i \)
3. For each series compute the squared difference with the overall averages
   \[ \Delta_i = \sum (X_{im} - X_m)^2 \]
   Where \( \Delta_i \) = squared distance of series \( i \) with overall average
   \( X_{im} \) value of series \( i \) (normalized)
   \( X_m \) = \( m^{th} \) value of overall average
4. Sort the time series based on the difference \( \delta \) computed in step 3.

Since the values for each series are dissimilar, it is not possible to compare the series values directly. In order to make the series comparable, the series will be normalized by dividing the individual values of the series by series mean. After normalization of data, square of sum of the differences of individual values in the time series with that of the overall mean vector values is computed. This results in
scalar values for each series. Ranking of these series of scalars will provide statistically valid ranks for the time series.

The algorithm for computing the ranking of time series is given in 7.1.A, typical visual presentation of the above processing steps is shown in the figure 7.3.

![Figure 7.3: Typical time series comparison](image)

It can be observed that comparison of each time series with the time series for reference indicates that proposed ranking is very reliable in calculating similarities. Since this benchmark automatically adapts to the underlying data, it also delivers trustworthy conclusions for a completely different set of time series.

### 7.2.3 Prediction with a Probabilistic Projection Model

Customer defection is a prominent issue in subscription based industry like mobile phones, credit cards, internet services, etc. The major characteristics of customer behavior prediction are contractual agreement between the company and its customers, acquisition cost for new customers, availability of large datasets of
behavior at the customer level. This can provide an ability to predict defection point of individual customer [FA06].

A standard approach for modeling defection/churn is to use survival analysis like Keplan-Meyer approach or Cox regression or using neural networks. However, several problems limit their usefulness when dealing with ad-hoc analysis in a practical situation. Some of them are like –

1. Irrational projection from standard statistical parametric models like multiple linear regression [for e.g. the expected value can become negative - which in reality is absurd- when projected beyond the limits] [FA09].

2. Some fixed time models like logistic regression, decision tree can provide good diagnostic information but fail to provide time to defect/churn [MI04].

3. Non-parametric models like Keplan-Meyer tables [ME04] though simple require human evolution which limits the usage in a dynamic setting like interactive analytics.

4. Though non-parametric methods like Cox regression and neural networks [OP03] show promising results their computational complexity is prohibitive.

The prediction of customer defection also called churn is an important function in customer retention. However, there is a very little work in aggregate projection [PA09] and almost no published work in integrating this with a decision support system as per our survey. This necessitates use of simple but robust parametric model for churn/defection projection. Hardie et.al [PA09] has used shifted geometric negative binomial distribution to predict the customer churn. The model is simple yet extremely effective. It is based on computing the gamma
value in the equation as shown in algorithm 7.2. Actual implementation of this algorithm is given below in figure 7.4.

This analysis can be done in two modes, manual and automatic. In both the modes the process remains the same but the space in which the analysis is carried out will differ. In manual mode user will select the dimensions of interest. However in the automatic mode a pre-defined structure for hierarchical analysis is followed.

The actual process consists of,

1) Computing SPG/NBD parameters for the selected dimensional values.
2) Predicting the churn/retention for subsequent periods of interest and
3) Ranking the time series of predicted values.

**Algorithm 7.2: for Prediction**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Assume $\alpha = 1, \beta = 1$ and LL = 1</td>
</tr>
<tr>
<td>2.</td>
<td>for $T = 1$ to $t$&lt;br&gt;   - If $t = 1$ $P(T) = \alpha / (\alpha + \beta)$&lt;br&gt;   - else $P(T) = (\beta + t - 2) / (\alpha + \beta + t - 1)$</td>
</tr>
<tr>
<td>3.</td>
<td>for $T = 1$ to $t$&lt;br&gt;   - If $t=1$ $S(T) = 1 - P(T)$&lt;br&gt;   - else $P(T) = S(T - 1) - P(T)$</td>
</tr>
<tr>
<td>4.</td>
<td>for $T = 1$ to $t$&lt;br&gt;   - Prop(T) = proportion survived till time T taken from cross tab</td>
</tr>
<tr>
<td>5.</td>
<td>$L(\alpha, \beta</td>
</tr>
<tr>
<td>6.</td>
<td>Use a solver to optimize log likely hood of (5) and to find $\alpha$ and $\beta$</td>
</tr>
<tr>
<td>7.</td>
<td>For $T=(t+1)$ to $(t+n)$&lt;br&gt;   - Compute $\gamma(T) = (\beta + t - 1) / (\alpha + \beta + t - 1)$</td>
</tr>
<tr>
<td>8.</td>
<td>Repeat the process from 1 till all the time series are analyzed and predicted.</td>
</tr>
</tbody>
</table>
Figure 7.4: Prediction with a probabilistic projection model.

7.3 CASE STUDY

A case study on real world data and decision support requirement is discussed. The proposed approach has been tested on a real world telecom datasets. Actual details of the dimensions or facts used and the identity of the operator are kept confidential due to sensitivity of the issue.

In total of about 17 dimensions, 7 demographic, 3 products related and 7 behavioral dimensions with 7 facts for usage and revenue has been used.
A customer base of about 100,000 has been used in testing the approach. The time related data of all these customers on all the dimensions for a period of 24 months are selected for churn projection analysis. The data has been sanitized to protect the business interests. The data is drawn from the data warehouse into the churn data mart. Both manual as well as automated ranking analysis of the projection is carried out.

A sample analysis report for the case study has been shown in figure 7.5. It shows the regions where the trends for churn are present town. Based on the interest, decision maker can choose the graphs for which further analysis can be done, if used in the manual mode and all the analysis is done at the back end, if used in the automatic mode.
Figure 7.5: Sample churn in different regions.

Figure 7.6, 7.7, 7.8 shows typical results from the proposed approach. It can be observed that the results from proposed approach are significantly similar to the actual results.

Figure 7.6: Projections for contract type with actual values.
In order to evaluate the suitability of the prediction algorithm, it has been compared with the results of Multi Linear Regression (MLR). Table 7.1 shows the results for different queries executed in manual mode. It can be observed that the MAE (Mean Average Error) and RMSE (Root Mean Square Error) are superior from the proposed system than from regression. The statistical test for change in the average error has confirmed that the proposed approach significantly out
performs other methods on aggregate projection. Further data for the Cox regression has to be modified to suit the software requirements [R].

Table 7.1: Prediction accuracy comparison

| Query | Proposed method | | | | MLR |
|-------|-----------------|-----------------|-----------------|-----------------|
|       | MAE | RMSE | MAE | RMSE |
| 1     | 1529 | 3534 | 10000 | 18000 |
| 2     | 155 | 1707 | 1200 | 2300 |
| 3     | 51 | 186 | 400 | 1000 |
| 4     | 38 | 123 | 450 | 1127 |
| 5     | 37 | 73 | 480 | 1400 |
| 6     | 162 | 267 | 1500 | 4000 |
| 7     | 9 | 37 | 85 | 320 |
| 8     | 29 | 75 | 120 | 350 |
| 9     | 98 | 103 | 400 | 500 |
| 10    | 15 | 59 | 75 | 405 |

Since our approach uses an integrated database dependent model, it is an order of magnitude factor than other approaches on performance. However,
automated analysis could not be tested on all the algorithms due to computational requirements. This again proves the effectiveness of the proposed approach in automated analysis.

7.4 CONCLUSION

Trend analysis with time series ranking prediction is one of the important analytical functions in business analytics. This requires a new architecture with state of art components to provide information on demand. In this research, an attempt has been made to develop an analytical server to provide these features.

The proposed architecture is built using column store database with time series ranking using pattern recognition and time series projection using probabilistic prediction model. A case study on telecom churn has been carried out and results are promising. The work is under progress to provide survival analytics capabilities at the individual level within the proposed framework.
8.1 Conclusion

Clustering is a process of grouping data items based on a measure of similarity, the same set of data items often needs to be partitioned differently for different applications hence clustering is a subjective process. A possible solution lies in reflecting this subjectivity in the form of knowledge. This knowledge is used either implicitly or explicitly in one or more phases of clustering.

A good clustering method will produce high quality clusters with high intra-class similarity and low inter-class similarity. The quality of a clustering result depends on both the similarity measure used by the method and its implementation. The quality of a clustering method is also measured by its ability to discover some or the entire hidden pattern.

Clustering large datasets has some important issues like excessive time, Computational complexity and so on. The existing systems have not utilized the capabilities of DBMS effectively. In the present work the capabilities of DBMS are used efficiently to optimize the performance.

**Objective of the proposed work**
• Identifying appropriate number of clusters.
• Seeding initial cluster centers.
• K-means clustering with pre-seeded cluster means.
• Developing user driven clustering method.

Achieved in this Work

The proposed implementations allow clustering of large datasets stored inside a relational DBMS, eliminating the need to export data using standard SQL. No special extensions to SQL for data mining were needed. We concentrated on defining suitable tables, indexing them and optimizing the queries for clustering.

An attempt has been made to develop a K-means algorithm to specify the number of clusters apriori so that convergence can be achieved fast. Once the seeds are identified standard K-means algorithm has been implemented using suitable tables in discs and optimized queries. The results 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 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 standard 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.

These results are encouraging and the proposed algorithm meets all the criteria set initially,

- Scalability
- Quality of cluster
- 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
- Number of clusters.

However, the comparison of this implementation with memory based algorithms could not be carried out due to large data size which could not fit into the memory.

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 out performed that on column store DBMS.

Compared to stand alone programs (like Weka and R), the capability of our approach to handle massive data sets without resorting to additional
hardware proves the efficacy of the technique. In addition the algorithms were implemented in a declarative fashion, very high level of abstraction is achieved thereby minimizing code maintenance.

The need for user driven clustering method is to segment and rank historic and predictive information for user defined segmentation. The performance of retention programs across regions or products or dealers in a telecom company are observed for segmentation and results are promising. It requires the projection of churn trend across different segments defined by user to identify pockets of interest. In this work analytical server using column store database with many features were developed.

8.2 Open issues which need research attention.

• Currently the implementation works only for numerical data. The assumption we have made in proposing this algorithm is that categorical data can be overlaid on the segments derived from numerical data to make any meaningful references. The algorithm can be extended to incorporate qualitative/categorical data.

• In our work all the columns/variables are given equal weight in computing K-means. However, the algorithm can be extended in which all the columns/variables are given unequal weights.
• The clustering quality is still measured with equal cost on every record. Weighing of the records can be added as an extension to improve the cluster quality.
#APPENDIX – 1

#SQLDM-Kmeans algorithm implementation for MonetDB. Tested on V5
#Developer: Suresh.L

#Dependencies:
#database table 'datatable'
#Number of clusters
#Working
#1. Cluster centers are sampled with 'insert into cc (cid, x1) select rid, x1 from data
order by Random() limit ?'
#2. Compute distances of each record with cluster centers with 'insert into dxy (rid, cid, dist) select data.rid, cc.cid, abs(cc.x1-data.x1) from data'
#3. Currently Hamming distance is implemented. Other distance measure can be used to replace it
#4. Compute the new cluster centers with 'insert into ncc(cid, x1) select cid, avg(data.x1) from mindxy, data where mindxy.rid = data.rid group by cid'
#5. Compute the cluster center difference with 'select max(abs(cc.x1-ncc.x1)) from cc, ncc where cc.cid=ncc.cid'

#Import the required DLLs
import dbapi
import odbc

#Assumes a connection named cdb, user = 'test' and password = 'test'
#Set the connection
con = odbc.connect("cdb/test/test")

#Create the data table
con.execute('create table data(rid int, x1 int)')

#Create the cluster center table
con.execute('create table cc(cid int, x1 int)')

#Create the distances table
con.execute('create table dxy(rid int, cid int, dist int)')

#Create minimum distance table
con.execute('create table mindxy(rid int, cid int, dist int)')

#Create the new cluster center table
con.execute('create table ncc(cid int, x1 int)')

#Create a cursor to handle SQL
cur = con.cursor()

#Populate the data table
#Assume the following data for testing
d = [(1,1), (2,2), (3,5), (4,6), (1,2), (2,1), (5,5)]
#Populate the table
for val in d:
    cur.execute('insert into data values(?, ?)', val)

#Populate the cluster center table
noc = raw_input('Enter number of clusters: ')
numOfClusters = (noc,)
#numOfClusters = (2,)
cur.execute('insert into cc (cid, x1) select rid, x1 from data order by Random() limit ?', numOfClusters)
#Adjust/set these parameters as needed
delta = raw_input('Enter the allowable difference')
#delta = 5
#Set the parameters for looping to converge
repeat = 'yes'
cycle = 1
#Loop while difference is greater than delta
while (repeat == 'yes' and cycle < 10):
    #test the cluster centers table bieng populated
    #print cur.execute('select rid, x1 from cc').fetchall()
    #Populate the distances table
    #cur.execute('insert into dxy (rid, cid, dist) values(1,2,3)')
    cur.execute('insert into dxy (rid, cid, dist) select data.rid, cc.cid, abs(cc.x1-data.x1) from data, cc')

    #Populate minimum distance table
    cur.execute('insert into mindxy(rid, cid, dist) select rid, cid, min(dist) from dxy')

    #Compute new cluster centers and populate the NCC table
    cur.execute('insert into ncc(cid, x1) select cid, avg(data.x1) from mindxy, data
    where mindxy.rid = data.rid group by cid')

    #Check the differences. if there is a difference delete old cluster centers and
   repeat from compute distances table
    difference = cur.execute('select max(abs(cc.x1-ncc.x1)) from cc, ncc where cc.cid=ncc.cid').fetchall()[(0)][0]
    if difference < delta:
        repeat = 'no'
break
cycle = cycle + 1
print 'repeat =',repeat, 'difference = ', difference
cur.execute('delete from cc')
cur.execute('delete from dxy')
cur.execute('delete from mindxy')
cur.execute('insert into cc select * from ncc')
cur.execute('delete from ncc')
print cur.execute('select * from cc').fetchall()


[FA01] Fahim.A.M, Salem A.M, Torkey F.A, Ramadan M.A ”An efficient enhanced k-means clustering algorithm”, Journal of Zhejiang University SCIENCE A ISSN 1009-3095 (Print); ISSN 1862-1775 (Online).


LIST OF PUBLICATIONS

Journals


International Conferences


4. “Analysis and Prediction Server with Column Store Database – A Case Study in Telecom Churn”, Proceedings of the IEEE International Conference on


National Conferences
1. “A Case Study on emulation of Column Oriented database design in Row Oriented data bases”, In the Proceedings of National Conference on Knowledge Engineering and its Applications, Bangalore , India,17-18th October 2009.

2 “Survey on Data mining Algorithms”, Published a paper in the National Conference on Web Engineering and applications, **Bhubaneswar**, India, Dec 2006.
VITAE

Suresh L was born on 24\textsuperscript{th} October 1967, at Bagur, Hassan, Karnataka, India. He has done Bachelor of Engineering in Computer Science and Engineering, Gogte Institute of Technology, Belgaum, Karnataka, India in the year 1990, and Master of Science in Computer Science from Birla Institute of Technology, Pilani, Rajasthan State in the year 1994. He joined the doctoral program at Dr.M.G.R University in 2004 on a part-time basis. He worked as Lecturer in the Department of Computer Science and Engineering for six Years in R.V.College of Engineering, Bangalore. He worked as Assistant professor and Head of the Department in the Department of Information Science and Engineering for twelve Years in B.M.S College of Engineering, Bangalore. He is currently working as Professor and Head of the Department, Department of Computer Science and Engineering in Cambridge Institute of Technology, Bangalore from last two years. His areas of interests are Data Mining, Data Structures, Algorithm Design and Distributed Computing. He has published papers in three International Journals, eleven International Conferences and eight National Conferences. He has attended various workshops and Conferences. He conducted various technical workshops and conferences in the Department/College.