Efficient clustering techniques for managing large datasets

Vasanth Nemala
University of Nevada Las Vegas

EFFICIENT CLUSTERING TECHNIQUES FOR MANAGING LARGE DATASETS

by

Vasanth Nemala

Bachelor of Technology, Computer Science
Jawaharlal Nehru Technological University, India
2007

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2009

ABSTRACT

Efficient Clustering Techniques for Managing Large Datasets

by
Vasanth Nemala

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

The result set produced by a search engine in response to the user query is very large. It is typically the responsibility of the user to browse the result set to identify relevant documents. Many tools have been developed to assist the user to identify the most relevant documents. One such a tool is clustering technique. In this method, the closely related documents are grouped based on their contents. Hence if a document turns out to be relevant, so are the rest of the documents in the cluster. So it would be easy for a user to sift through the result set and find the related documents, if all the closely related documents can be grouped together and displayed.

This thesis deals with the computational overhead involved when the sizes of document collections grow very large. We will provide a survey of some clustering methods that efficiently utilize memory and overcome the computational problems when large datasets are involved.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express the appreciation to my committee chair, Dr. Kazem Taghva for all his support and guidance through every stage of this thesis research. Without his guidance and persistent help, completion of this thesis would not have been possible.

I would like to thank my committee members, Dr. Gewali and Dr Datta, whose work demonstrated to me how we can stand in life and achieve lot of knowledge with simplicity. Special thanks to Dr. Venki for being a part of my committee.

I would also like to extend my appreciation towards my parents and my friends for always being there for me through all phases of my work, for their encouragement and patience and giving me their invaluable support without which I would never be where I am today.

CHAPTER 1

INTRODUCTION

Constant advance in science and technology makes collection of data and storage much easier and very inexpensive than ever before. This led to the formation of enormous datasets in science, government and industry, which should be processed or sorted to get useful information.

For example if we consider the results generated by a search engine for a particular query, user has to sift through the long lists and find the desired solution. But this job can be very difficult for the user if there are millions of web pages displayed as solutions for a given query. Thus Clustering techniques can be very useful in grouping the closely related solutions of a given query and displaying the results in the form of clusters so that the unrelated documents can be avoided even without taking a glimpse at them.

The main idea behind clustering any set of data is to find inherent structure in the data, and interpret this structure as a set of groups, where the data objects within each cluster should show very high degree of similarity known as intra-cluster similarity, while the similarity between different clusters should be reduced.

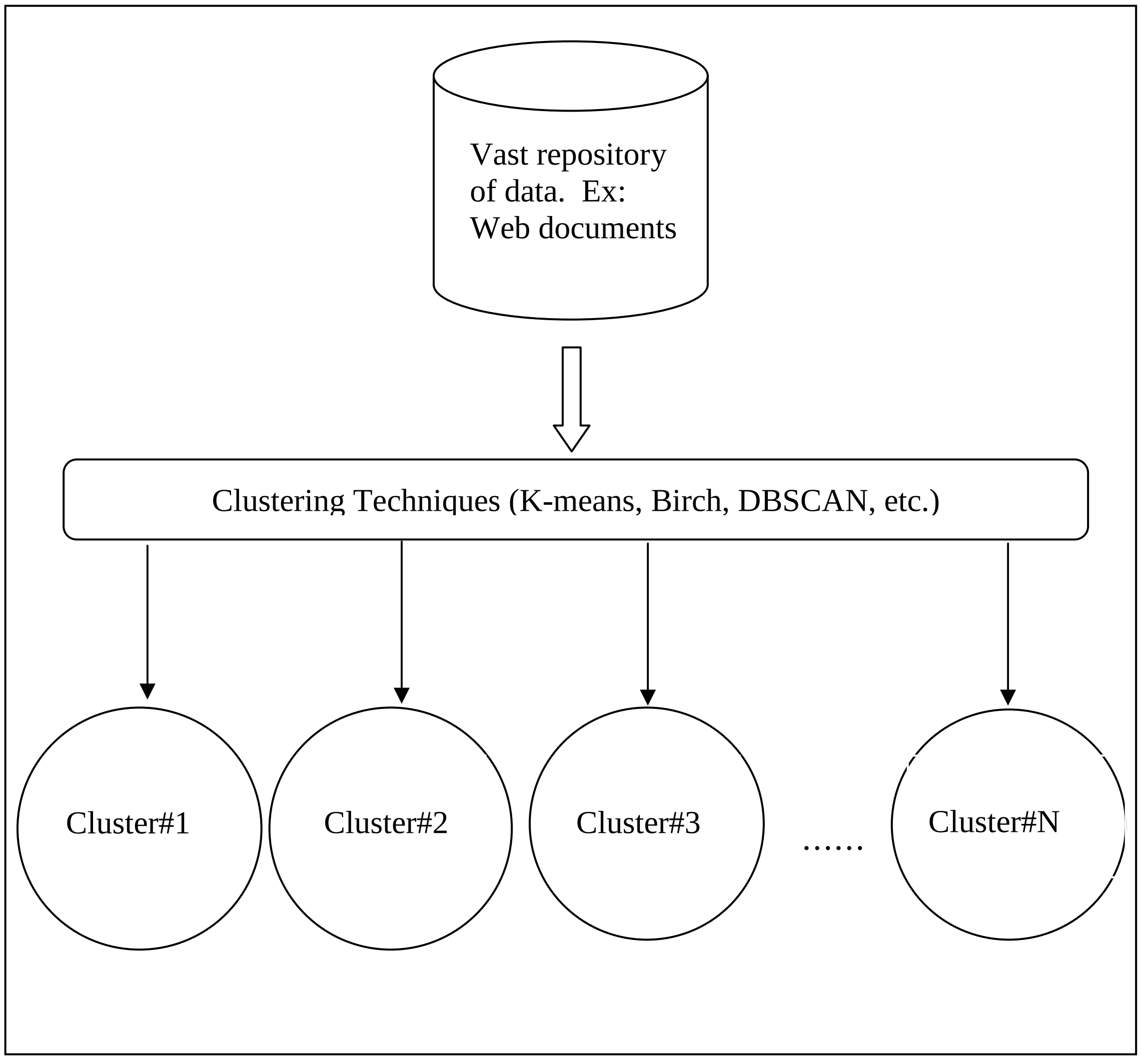

Vast repository of data. Ex: Web documents
Clustering Techniques (K-means, Birch, DBSCAN, etc.)
Cluster#1
Cluster#2
Cluster#3
……
Cluster#N

Figure1. Depicting the entire clustering process

Clustering is employed in many areas like

- News articles: Classifying daily news articles into different groups like sports, highlights, business and health etc.
- Classification of web documents (WWW): The results given out by the search engines can be clustered according to the degree of similarity for the given query.

- Exploring market: Given a large database with every individual customer past purchase records, finding groups of customers with similar behavior.
- Research projects: Collecting large amount of data daily from sensors will go useless if certain conclusions or not made. Finding necessary relations in collected data and classifying them could draw helpful conclusions.
- Earthquake studies: Identifying dangerous zones by clustering observed earthquake epicenters.

The main problems associated with the traditional clustering algorithms are handling multidimensionality and scalability with rapid growth in size of data. The increase in size of data increases the computational complexities which have a devastating effect on the run-time and memory requirements for large applications.

In this thesis, first we present all the major clustering techniques in brief and then we discuss about the drawbacks of first generation clustering algorithms. Then we signify how current clustering algorithms overcome the drawbacks of traditional clustering algorithms. Finally we present three clustering techniques in detail that have revolutionized clustering in their era of discovery.

CHAPTER 2

CLUSTERING: UNSUPERVISED OR MACHINE LEARNING

Clustering is a division of data into groups of similar objects. Each group (= a cluster) consists of objects that are similar between themselves and dissimilar to objects of other groups. From the machine learning perspective, Clustering can be viewed as unsupervised learning of concepts [5].

A simple, formal, mathematical definition of clustering, as stated in [6] is the following: let $X \in R^{m \times n}$ is a set of data items representing a set of m points x_i in R^n. The goal is to partition X into K groups C_k such that every data that belongs to the same group are more "alike" than data in different groups. Each of the K groups is called a cluster. The result of the algorithm is an injective mapping $X \rightarrow C$ of data items Xi to clusters Ck.

In recent years, drastic change in use of web and improvement in communication in general has led to store loads and loads of information in databases. This requirement made lot of researchers to think about ways of information retrieval and categorizing the data, so that meaningful information can be retrieved.

As stated in [7] Unsupervised learning, refers to that class of machine learning approach where the system produces certain sequence of outputs based on a set of given inputs without any response from its environment.

This chapter deals with the basic concepts of clustering and different types of clustering algorithms and goes on to describe how the current generation techniques are more advanced and how they overcome the drawbacks of first generation clustering techniques.

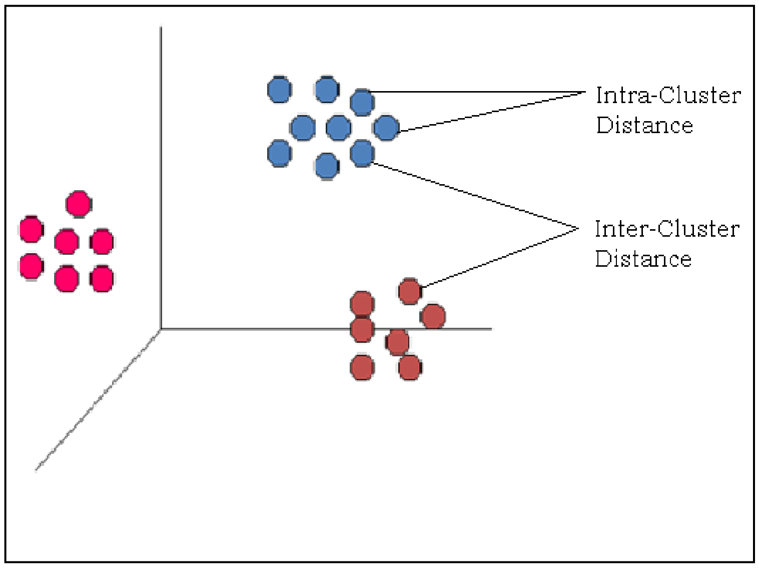


Figure 2. Euclidean Distance Based Clustering in 3-D space

- Intra-cluster distances are minimized and
- Inter-cluster distances are maximized.

2.1 Different Types of Clustering Algorithms

Clustering can be done in many different ways; each clustering technique produces different types of clusters. Some take input parameters from the user like number clusters to be formed etc, but some decide on the type and amount of data given. The main developments have been the introduction to density based and grid based clustering methods. Clustering algorithms can be classified into five distinct types:

- Partitioning methods;
- Hierarchical methods;
- Model-based methods;
- Density based methods; and
- Grid based methods.

Partitioning Methods

If a database containing n data objects is given, then a partitioning method constructs k clusters of the data where $k<=n$ and k is the input parameter provided by the user. That is, it classifies the data into k groups which should satisfy the following conditions: (1) each group must contain at least one data object and (2) each data object should belong to only one group. The second requirement becomes easy in fuzzy k-mean clustering in which one object can be resembled by two or more groups.

With *k* as the given number of partitions to be made, the partitioning method creates an initial partition. Then more number of iterations are followed in which objects are moved from one group to other making sure that in-cluster similarity is more than similarity with objects in other clusters.

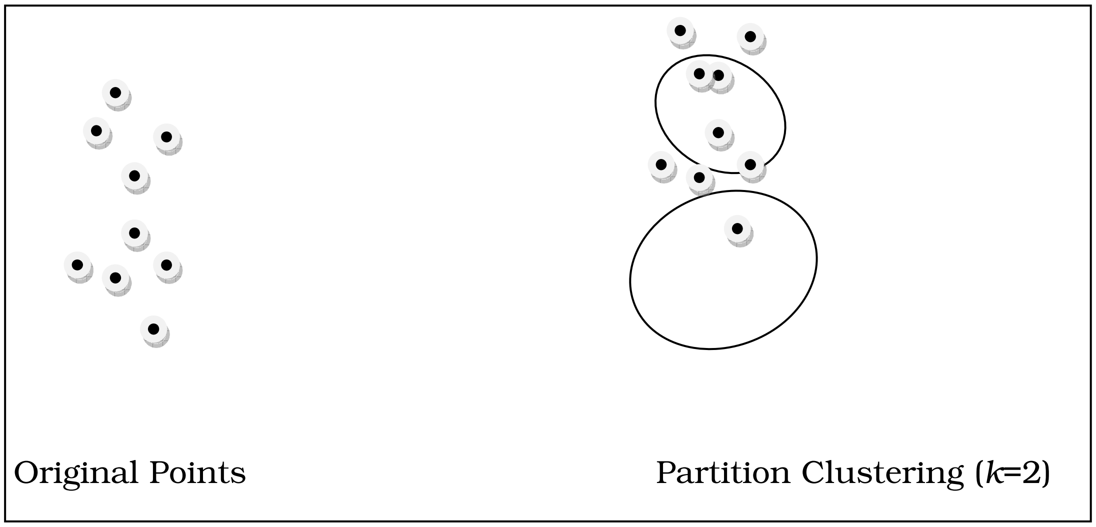


Figure 3: Data objects before and after partitioning

Popular Partitioning Methods: K-Means and K-Medoids

The most well-known and commonly used partitioning methods are k-means proposed by (Mac Queen 1967) and k-medoids proposed by (Kaufman and Rousseeuw 1987).

2.1.1 The K-Means Method: Centroid-Based Technique

The k-means algorithm takes input *k* from the user and partitions *n* data objects into *k* clusters so that the resulting intra-cluster similarity is very high and inter-cluster similarity is very low. The cluster similarity is calculated based on the mean value of the objects in the cluster. First, it randomly picks *k* data objects as the mean or centroid points. For each of the remaining objects, an object assigned to the centroid to which it is 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 till good clusters are formed. Typically, squared root function is used for this which can be 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 function tries to make the clusters as separate as possible. The k-means algorithm as in [8] is

Input: The number of clusters to be formed *k*, and number of data objects contained in the database *n*.

Output: Set of clusters k, which minimizes the squared error function.

Algorithm:

1) Randomly pick *k* objects as initial centroids;
2) Repeat;
   a. Assign the remaining objects to the cluster mean's that are most similar to each of the objects.
   b. Update the cluster means.
3) Until no change;

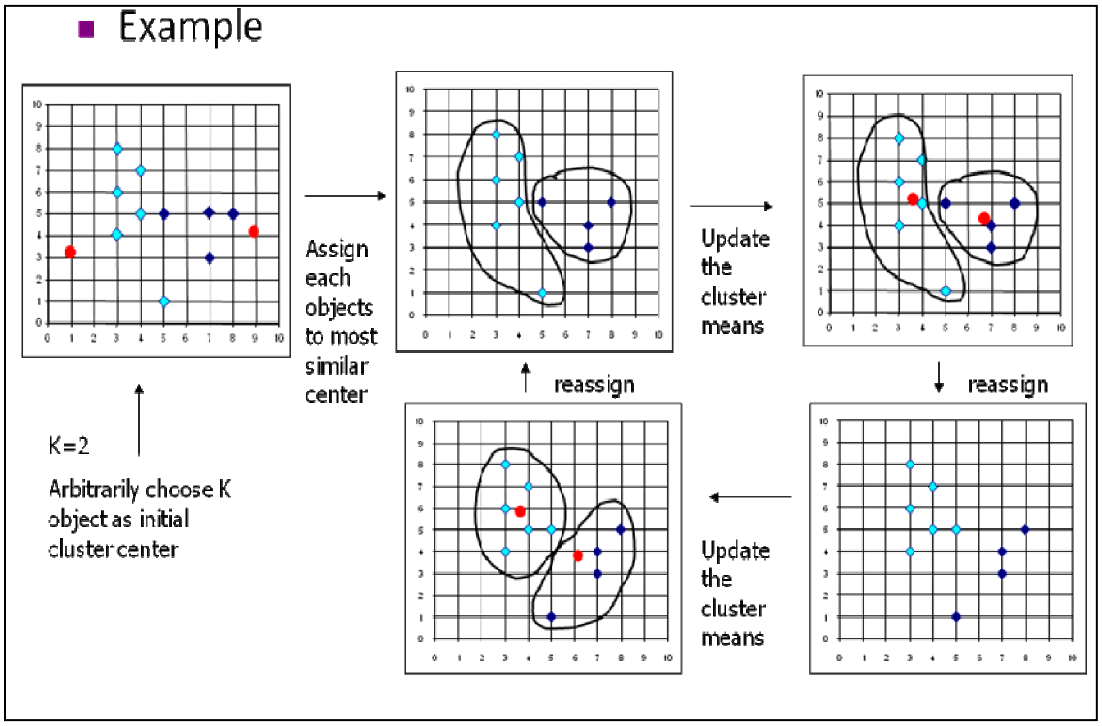

Example
Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means
K=2
Arbitrarily choose K object as initial cluster center

Figure 4: Clustering a set of points based on k-means method [8]

The method is relatively scalable and efficient in handling large data sets because the computational complexity of the method 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<<n and t<<n so, the method often ends up at local optimum. But the draw backs of this method are;

1) It can be applied only when mean of a cluster is defined, but when data with categorical attributes is involved it cannot be the case;
2) The user should specify the number of clusters *k* in advance and
3) It is sensible to noise and outlier data points.

2.1.2 The K-Medoids Method: Representative Point Based

K-medoids algorithm was developed to overcome the drawbacks of k-means method which was very sensitive to outliers. An object with some extremely large value may substantially distort the distribution of data in k-means method. So instead of taking the mean value of objects in a cluster as a reference point, an object that is most centrally located in the cluster can be taken as a representative object, called as medoid. Thus the partitioning method can be performed by minimizing the sum of dissimilarities between each object and with its corresponding reference point. The algorithm of k-medoids algorithm as in [8] is

1) Arbitrarily choose k objects as 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;
   c. Swap the medoid x by an object y if such a swap reduces the objective function;
3) Until no change;

This algorithm creates k partitions for n given objects. Initially k medoids are selected which are located more centrally in each cluster, the algorithm repeatedly tries to make a better choice of medoids by analyzing all the possible pairs of objects.

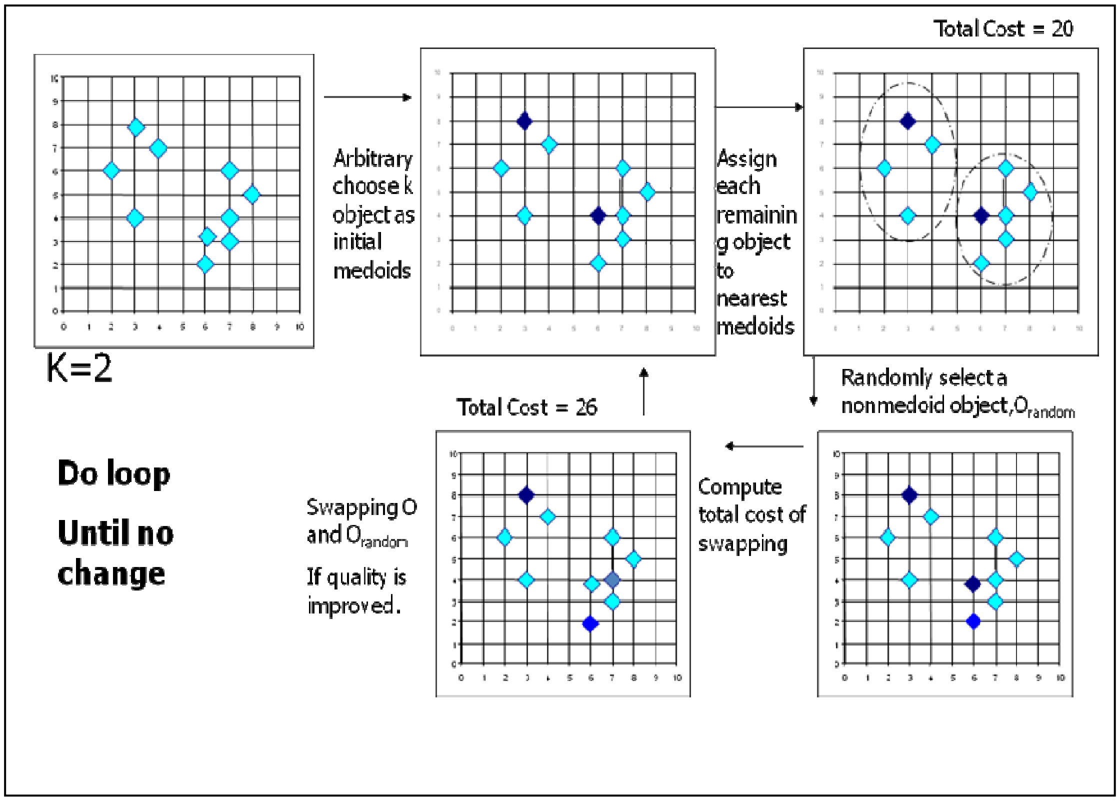

Total Cost = 20
Arbitrary choose k object as initial medoids
Assign each remaining object to nearest medoids
K=2
Randomly select a nonmedoid object, O_{random}
Total Cost = 26
Do loop
Until no change
Swapping O and O_{random}
If quality is improved.
Compute total cost of swapping

Figure 5: Clustering using k-medoid method [8]

The measure of clustering quality is calculated for each such combination. The best points chosen in one iteration are selected as medoids in next iteration. The cost of a single iteration is $O(k(n-k)^2)$. For very large values of n and k the computational cost can be very high.

The k-medoids method is more robust than k-means because it is less influenced by outliers or other extreme values than mean. But its processing is very costly than k-means method and it also has the drawback of user providing the input parameter *k* (number of clusters to be formed).

Hierarchical Methods

The set of given data objects are partitioned in form of a tree like structure or nested clusters in hierarchical clustering. The hierarchical methods can be classified into two types.

- Agglomerative and
- Divisive

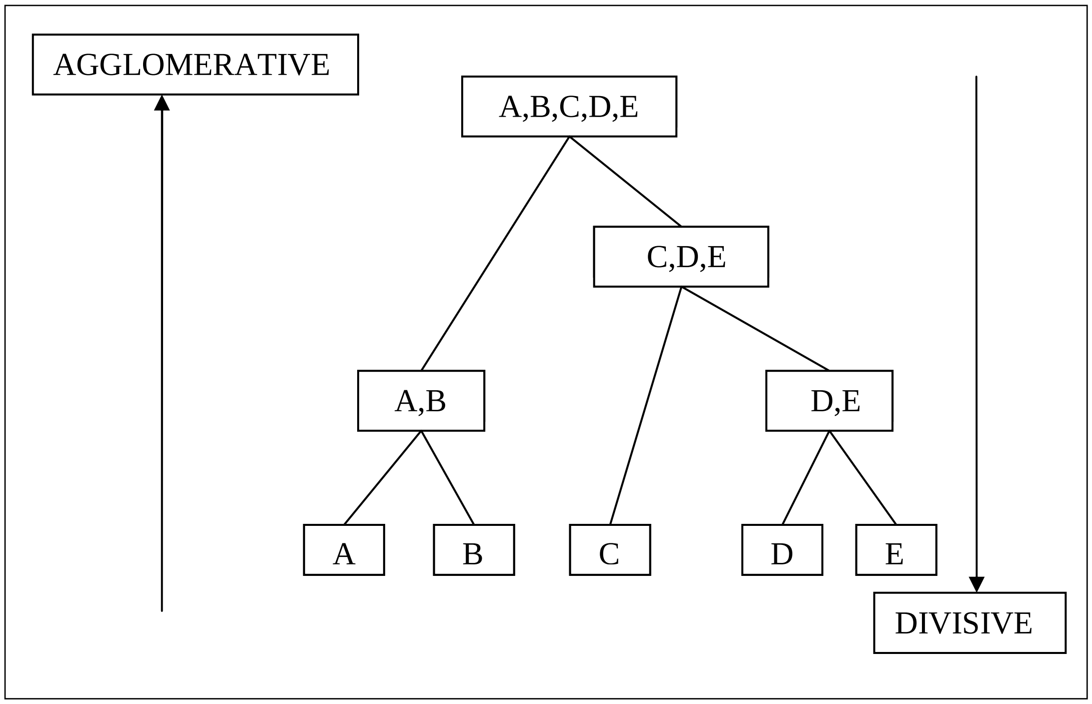


Figure 6: Agglomerative and Divisive clustering

In agglomerative method also known as bottom-up approach, each object forms a separate group. It successively merges the groups close to one another by checking the similarity function, until all the groups are merged into one, that's until the top most level of hierarchy is reached or until a termination condition holds. In divisive clustering also known as top-down approach, initially all the objects are grouped into a single cluster which can also be called as parent. 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.

2.1.3 Agglomerative Method

This method begins by treating each object as an individual cluster and then proceeds by merging two nearest clusters. The distance between any two clusters m and n is defined by a metric $D_{m,n.}$ Metrics can be single-link, complete-link and group average etc. A general class of metrics was given by Lance and Williams [1]. If $D_{k,ij}$ be the distance between cluster k and the union of cluster *i* and cluster *j*, then:

$$D_{k,ij} = \alpha_i D_{k,i} + \alpha_j D_{k,j} + \beta D_{i,j} + \gamma \,|\, D_{k,i} - D_{k,j} \,|$$

The agglomerative method is as follows:

- Consider each object to be an atomic cluster. The (n x n) distance matrix represents the distance between all possible pairs of clusters.
- Find the smallest element in the matrix. This corresponds to the pair of clusters that are most similar. Merge these two clusters, say m and n, together.
- Measure the distances between the newly formed cluster and the

other remaining clusters using a distance function. Delete the row and column of m and overwrite row and column of cluster n with the new values.

- If the current number of clusters is more than k then go to step 2; otherwise stop. The merging process can continue until all the objects are in one cluster.

The advantages of hierarchical methods are that they are easy to implement computationally. They are able to tackle larger datasets than the k-medoids method and we can run the algorithm without providing the input *k* (the number of clusters to be formed). The drawbacks of agglomerative method are:

- The algorithm has $O(n^3)$ time complexity. Even though the order of the distance matrix decreases with each iteration, the cost of Step2 on iteration k is $O((n - k)^2)$, and we are guaranteed (n - k) iterations before we get to k;
- The clusters produced are heavily dependent on the metric $D_{i,j}$. Different metrics can produce different clusters. For instance, the complete-link metric tends to produce spherical clusters, whereas the single-link metric produces elongated clusters [1].

2.1.4 Divisive Method

The contrast procedure of agglomerative clustering is the divisive method. Initially all the data objects are considered in one cluster. Then for each object the degree of irrelevance is measured and the most irrelevant data object is split from the main cluster and a new cluster is formed with only that data object in it. The highest degree of irrelevance of an object corresponds to the one that is most distant from all other objects in that cluster. Let the average distance between object *i* and the cluster C_j be defined as [1]:

$$D_{i,Cj} = \frac{1}{n_j} \sum_{x_h \in C_j} d(x_i, x_h)$$

The most irrelevant object splits off and forms a new cluster. This is equivalent to splitting the cluster with the largest diameter. The process continues until it satisfies certain termination condition, such as a desired number of clusters are formed or the distance between two closest clusters is above a certain threshold distance. These methods face the difficulty of making a right decision of splitting at a high level. The algorithm for divisive method is [1]:

- Select the cluster containing the most distant pair of objects. This is the cluster with the largest diameter.
- Within this cluster, find the object with the largest average distance from the other objects. Remove the object from the cluster, allowing it to form a new atomic cluster.
- For object *h* in the cluster being split, calculate the average distance between it and the current cluster; and the average distance between the object and the new cluster. If the distance to the new cluster is less than the distance between it and the

current cluster, move the object *h* to the new cluster. Loop over all the objects in the cluster.

- If no objects can be moved, but the current number of clusters is greater than *k*, go to step1. Otherwise stop.

The drawbacks of divisive method are:

- The time complexity of algorithm is O(n^3), O(n^2) on the step1 of the algorithm for each iteration. Moreover there are expensive calculations that may take place in step3 of the algorithm.
- In step 3 the group averages between an object and the new and existing clusters need to be recalculated after an object is moved. This will be costly in terms of number of calculations and the amount of storage required.
- The method only searches one of the N(n,k) possible partitions.

In hierarchical clustering once a split or merge is done, it cannot be undone. This fact acts as both key to success and drawback for hierarchical clustering. The firmness of hierarchical method leads to less computational cost without a combinatorial number of choices but the main problem with it is invalid decisions cannot be corrected.

Hierarchical clustering methods are simple but encounter problems at making critical decisions for selection of correct merge are split points. Such a decision is critical because once a group of objects is merged or split, the process at the next step will work on the newly generated clusters. It will never undo what was done previously nor perform object swapping between clusters. Thus merge or split if not done wise may result in low quality clusters. These methods have scaling problem since the decision of merge or split needs to examine and evaluate a good number of objects or clusters.

Hierarchical clustering can be improved by integrating this method with other clustering techniques for multiple phase clustering. One such method known as BIRCH, first partitions objects hierarchically using tree structures and then applies other clustering techniques to produce refined clusters. This method will be discussed in chapter 4 in detail.

Model-Based Methods

The rapid growth in size of datasets has led to increased demand for very good clustering methods for analysis, while at the same time introducing some constraints in terms of memory usage and computational time. Model-based clustering a relatively recent development (McLachlan and Basford 1988, Banfield and Raftery 1993, Mclachlan and Peel 2000, Fraley and Raftery 2002) has shown good performance in many applications. A model-based method hypothesizes a model for each of the clusters, and finds the best fit of the data to that model [8].

In model-based clustering, the data $(X_1,\ldots\ldots,X_n)$ are assumed to be generated by a mixture model with density

$$\prod_{i=1}^{n} \sum_{k=1}^{G} T_k \, f_k(X_i|\theta_k),$$

where $f_k(X_i|\theta_k)$ is a probability distribution with parameters θ_k, and T_k is the probability of belonging to the k^{th} component or cluster. Most often f_k are taken to be multivariate normal distributions θ_k parameterized by their means μ_k and covariance's $\sum_k$.

Basic Model-Based Clustering Strategy [9]:

1. Determine the minimum and maximum number of clusters to consider (Gmin, Gmax), and a set of candidate parameterizations of the Gaussian model.
2. Do EM for each parameterization and each number of clusters Gmin, ... , Gmax, starting with conditional probabilities corresponding to a classification from unconstrained model-based hierarchical clustering.
3. Compute BIC for the mixture likelihood with the optimal parameters from EM for Gmin, ... , Gmax clusters.
4. Select the model (parameterization / number of clusters) for which BIC is maximized.

In model-based clustering a model can be formulated and fit to the data. The process of selecting a model places a great deal of supervision, suggesting that the user has reasonable of knowledge about the structure of the data.

2.1.5 EM Algorithm

Problem of clustering a set of objects can be considered as a missing data problem and the missing data can be completed by filling it with observations made on the objects as well as the labels that allocate the objects to the correct corresponding groups. Before clustering, the labels are missing and all we have are the measurements recorded for the objects.

The EM algorithm is the standard way of analyzing statistical models that have missing data. The EM algorithm proceeds by estimating the missing data (the E-Step) and then estimating the parameters of the model, through maximum likelihood (the M-Step).

EM algorithm requires the entire collection of objects and their clusters to be represented by a statistical model. The data can be considered as a random sample from a mixture of several probability distributions. These probability distributions define the clusters. Each object is generated by one and only one of these distributions; hence belong to one and only one cluster. The likelihood of the data has a multinomial form and can be defined as:

$$L(\psi) = \prod_{i=1}^{n} f(x_i, \psi)$$

where ψ is a set of parameters specifying the current model and $f(x_i, \psi)$ is the probability distribution function of the mixture distribution. As each object has been generated by one and only one of these distributions, the joint density of x_i and z_i, $f(x_i, z_i ; \psi)$ can be defined as

$$\prod_{j=1}^{k} (\pi_j f_j(x_i))^{z_{ij}}$$

where π_j represents the prior probability that object *i* came from the component density $f_j(.)$; the component densities typically depend on additional parameters. The z_{ij} are 0-1 indicator variables, indicating whether object *i* was generated by $f_j(.)$ or not and $z_i = (z_{1i},..., z_{ki})$ is the vector containing the k indicator variables for object *i*, $\sum_{j=1}^{k} z_{ji} = 1$ $\forall i=1,...,$ *n*. These indicator variables represents missing labels.

EM algorithm is robust in nature because it can fit an otherwise intractable statistical model to the data. The drawbacks of the EM algorithm are:

- It scans the entire dataset for every iteration;
- By imposing a statistical model on the data, we are depending on a huge amount of prior knowledge that may or may not be available.
- The linear rate of convergence will make it very slow for complex models and large datasets.
- The EM algorithm is dependent on its starting point.

Density Based Methods

Density-based clustering methods are based on a local cluster criterion. Clusters are assumed as regions in the data space in which the objects are dense and the clusters are separated by regions of low object density. These regions have an arbitrary shape and the data points inside a cluster may be arbitrarily distributed.

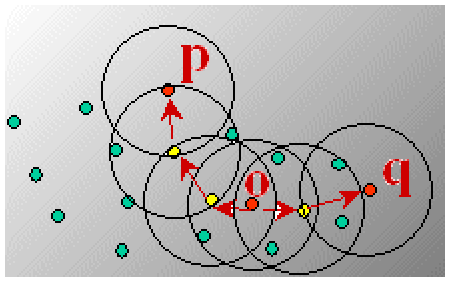


Figure 7: Density-Based clustering [11]

The idea is to increase the size of the cluster with data objects as long as the density 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. Hence the density-based clustering can filter out noise and discover clusters of arbitrary shape.

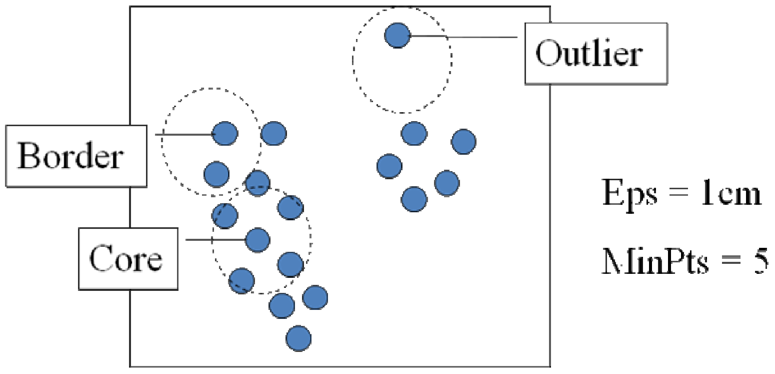


Figure 8: Defining basic terms of density-based Clustering

General clustering algorithms are attractive for the task of cluster identification in spatial databases. But, the application to large spatial databases rises the following requirements for clustering algorithms:

- Minimum amount of domain knowledge to determine the input parameters.
- Discovering clusters with good efficiency and arbitrary shape on large databases.

The very prominent first generation clustering algorithms offer no solution to the combination of these requirements. The main idea is that for each object of a cluster, the neighborhood of a given radius (ϵ) (called ϵ–neighborhood) has to contain at least minimum number of data objects. An object, that is within a given radius (ϵ) containing minimum number of neighborhood objects is called as core object [8].

- An object p is directly density-reachable from object q with respect to radius (ϵ) and a minimum number of points in a set of objects D if p is within the ϵ–neighborhood of q which contains at least a minimum number of points [8].
- An object p is density-reachable from object q with respect to radius (ϵ) and a minimum number of points in a set of objects D if there is a chain of objects $p_1, \ldots\ldots, pn$, $p_1 = q$ and $p_n = p$ such that for $1 \leq i \leq n$, $p_i \in D$ and p_{i+1} is directly density reachable from p_i with respect to ϵ and minimum points [8].

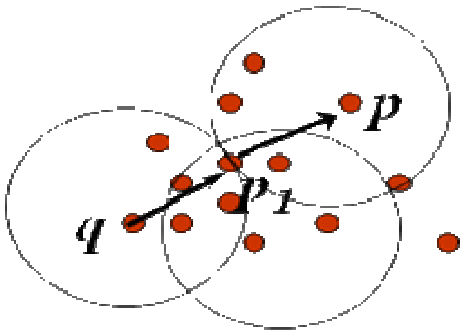


Figure 9: Density-reachability

- An object p is density-connected to object q with respect to radius (ϵ) and a minimum number of points in a set of objects D if there is an object $o \in D$ such that both p and q are density-reachable from o with respect to ϵ and minimum number of points [8].

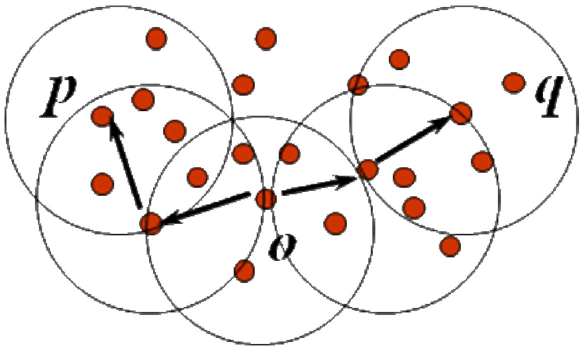


Figure 10: Density-connectivity

Density reachability is the transitive closure of direct density reachability, and this relation is asymmetric. Only core objects are mutually density reachable. A density-based cluster is a set of density-connected objects which is maximal with respect to density-reachability, and every object not contained in any cluster is noise [8].

In this thesis we will see DBSCAN relying on a density-based notion of clusters. DBSCAN requires only one input parameter and helps the user in determining an appropriate value for it. We will see in detail about the DBSCAN algorithm in chapter 4.

Grid-Based Methods

A grid-based clustering technique takes in the object space and quantizes it into a finite number of cells forming a grid structure. Then the method performs all the operations on that grid structure. The main advantage of this method is its fast processing time which is independent of the number of objects, and dependent only on the number of cells in each dimension in the quantized space. Grid-based methods use a single uniform grid mesh to partition the entire problem domain into the cells and the data objects located within a cell are represented by the cell using a set of statistical attributes from the objects. Clustering is performed on the grid cells, instead of database itself. Since the size of the grid is much less than the number of data objects, the processing speed can be significantly improved.

Most of the data mining applications require the clustering algorithms to find clusters embedded in subspaces of high dimensional data, scalability, non-presumption of any canonical data distribution, and insensitivity to the order of input records.

2.1.6 CLIQUE (CLustering In QUEst)

Clique is a grid-based clustering technique that satisfies each of the above requirements. It identifies dense clusters in subspaces of maximum dimensionality. It generates cluster descriptions in form of DNF expressions that are minimized for each of comprehension. It produces identical results irrespective of the order in which the input records are presented and does not presume any specific mathematical form for data distribution [12].

Clique is not purely a grid-based clustering technique. It can be considered as both density-based and grid-based clustering technique. It can automatically identify subspaces of a high dimensional data space that allow better clustering than original space.

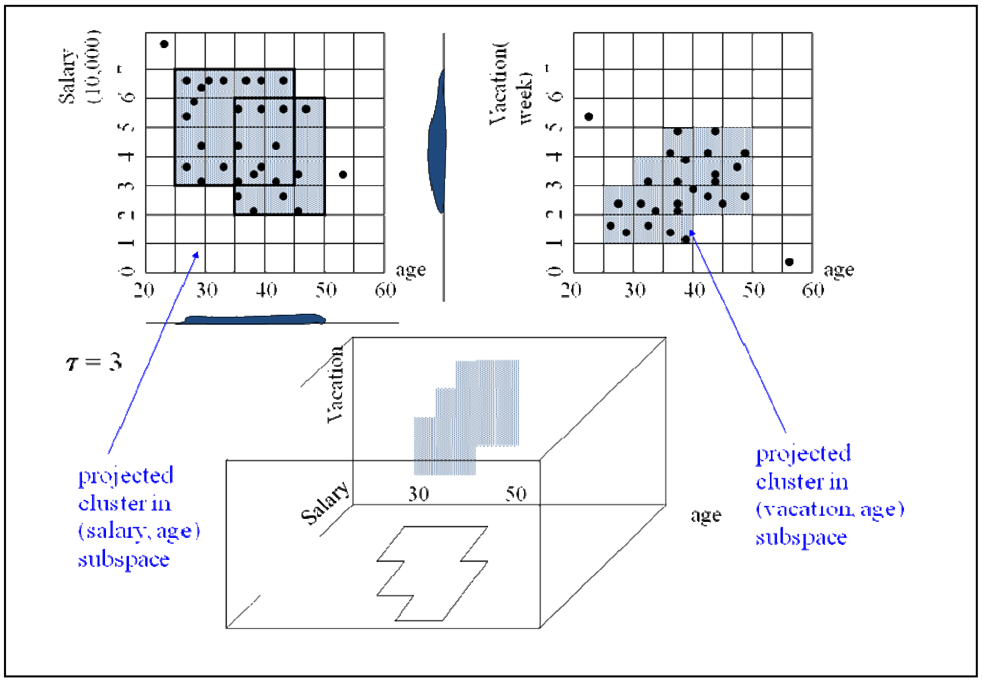


Figure 11: Representing **CLIQUE** clustering technique

Clique technique partitions each dimension into the same number of equal length interval. It partitions an m-dimensional data space into non-overlapping rectangular units. A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter. A cluster is a maximal set of connected dense units within a subspace.

Partition the data space and find the number of points that lie inside each cell of the partition.

- Identify the subspaces that contain clusters using the Apriori principle and then identify clusters:

a. Determine dense units in all subspaces of interests.

b. Determine connected dense units in all subspaces of interest.

- Generate minimal description for the clusters

a. Determine maximal regions that cover a cluster of connected dense units for each cluster

b. Determination of minimal cover for each cluster

Strengths and Weaknesses of CLIQUE:

- It automatically finds subspaces of the highest dimensionality such that high density clusters exist in those spaces.
- It is insensitive to the order of input records and does not presume some canonical data distribution.
- It scales linearly with the size of input and has good scalability as the number of dimensions in the data increases.

The only weakness of this clustering technique is the accuracy of the clustering result may be degraded at the expense of simplicity of the method.

CHAPTER 3

CLUSTERING LARGE DATASETS

After discussing about all the major types of clustering algorithms, we come to a stage where we can form a conclusion, why the traditional clustering algorithms have problems with the large datasets. We notice that if we want to reduce the algorithm's computational complexity then we need to stop working on distance space. Methods that include distance-space functions seem to have more scalability problems than their vector-space counterparts. Calculating and storing the relationship between all the possible pairs of n objects is $O(n^2)$. Calculating the distance between two objects can be expensive when the measurements are of high dimension. There is no predefined method to choose the "center" of a cluster and finding one via some ad-hoc method adds to the computational cost.

It is very clear that vector-space methods enjoy some advantages over distance-based methods. If we want to impose a statistical model on the vector-space, then we can use statistics estimated from objects in the cluster to represent it. The ability of the vector-space models to calculate "reliable" representations of each cluster can be used to improve storage and calculation costs. Research has been focused on the scalability of clustering algorithms, the effectiveness of techniques for clustering complex shapes and types of data, high-dimensional clustering techniques and methods for clustering mixed numerical and categorical data in large databases.

3.1 Essential Requirements of Clustering:

- **Scalability**: Many clustering algorithms perform well with small datasets containing less than 200 data objects. But if the objects number is in millions clustering techniques may lead to biased results.
- **Different types of Attributes should be Dealt**: Many clustering algorithms are designed to deal with numerical data. However, many applications may require clustering other types of data, such as binary, categorical and ordinal data, or mixtures of these types.
- **Discovering Arbitrary Shaped Clusters:** Many clustering algorithms find clusters based on Euclidean or Manhattan distance measures. Algorithms based on such distance measures tend to find clusters with spherical shape with similar size and density. However, a cluster can be of any shape. It should be very important to develop algorithms that find arbitrary shaped clusters.

- **Minimum requirement of domain knowledge:** Many clustering algorithms require users to initially give the input parameters like number of desired clusters etc., The clustering results are often very sensitive to input parameters. It is difficult to determine many parameters by the user especially for datasets that contain high-dimensional data.
- **Ability to deal with noisy data:** Most of the real time large databases have outliers, missing, unknown and erroneous data. Some of the clustering algorithms are sensitive to that kind of data and may lead to clusters of poor quality.
- **Insensitivity to the Order of Input Records:** Some clustering methods are sensitive to the order of input records passed to them. However the order in which the input records are given to the clustering algorithms they should be able to produce same clusters in any of the way.
- **High Dimensionality:** Many clustering algorithms are good at handling low dimensional data, involving only two to three dimensions. The clustering algorithms should be able to cluster data objects in high-dimensional space, especially considering that data in high-dimensional space can be very sparse and highly skewed.
- **Interpretability and Usability:** Users may expect the clustering

results to be interpretable, comprehensible and usable. Clustering may need to be tied up with some specific semantic interpretations and applications. It is important to know how an application goal may influence the selection of clustering methods.

3.2 Improving Traditional Methods:

Large datasets can be clustered by extending the existing methods so that they can cope with a large number of objects. The focus is on clustering large number of data objects rather than a small number in high dimensions. In partitioning methods the improved and refined method over k-medoids is CLARANS developed by Ng and Han (1994).

3.2.1 CLARANS

CLARANS stands for "Clustering Large Applications based on RANdomized Search". Instead of exhaustively searching a random subset of objects, CLARANS proceeds by searching a random subset of the neighbors of a particular solution, *S*. Thus the search for the best representation is not confined to a local area of the data.

The CLARANS algorithm is assisted by two parameters: MAX_{neigh}, the maximum number of neighbors of *S* to access; and MAX_{sol}, the number of local solutions to obtain. The CLARANS algorithm is as follows [1]:

1. Set *S* to be an arbitrary set of *k* representative objects. Set $i=1$.
2. Set $j = 1$.
3. Consider a neighbor *R* of *S* at random. Calculate the total swap contribution of the two neighbors.

4. If R has a lower cost, set $R = S$ and go to step 2. Otherwise increase j by one. If $j \leq \text{MAX}_{\text{neigh}}$ go to Step 3.
5. When $j > \text{MAX}_{\text{neigh}}$, compare the cost of S with the best solution found so far. If the cost of S is less, record this cost and the representation. Increment i by one. If $i > \text{MAX}_{\text{sol}}$ stop, otherwise go to step 1.

3.2.2 Fractionization and Refractionization

Fractionization was a way of adapting any hierarchical clustering method so that it can deal with large datasets. The main idea was to split the data into "manageable" subsets called fractions and then apply the hierarchical methods to each fraction. The clusters resulting from the fractions are then clustered into k groups by the same clustering method. The number of groups to be estimated should be provided in advance.

Fractionization

Let n be the number of data objects and M be the maximum number of data objects that can be handled by clustering procedure in a reasonable time. The fractionization algorithm is as follows:

1. Split n objects into fractions of size M each.
2. Cluster each fraction in αM clusters, where $\alpha<1$. Summarize each new cluster by its mean. These cluster means are referred as meta-observations and are treated as they were data.
3. If the number of meta-observations is greater than M, then the

clustering method still cannot process them. Go to step 1 treating the meta-observations as if they were data.

4. Cluster the meta-observations into *k* clusters using the desired clustering method.
5. Classify each individual as belonging to the cluster with the nearest mean.

The computational effort required to cluster each fraction is $O(M^2)$ and so is independent of *n*. On the i^{th} iteration where there are $\alpha^{i-1}(n/M)$ fractions to be processed by the clustering method which means that the total running time is linear in n and decreasing in α. But still this procedure has problems like specifying the number of clusters in advance and involving the formation of meta-observations.

Refractionization

Refractionization is the repeated application of the fractionization that processes fractions based on the clusters resulting from the previous fractionization iteration. The refractionization algorithm is almost the same as fractionization algorithm except in 5th step, fractions for the next iteration are created as the clusters are formed in step 4. As soon as a cluster has M objects in it, consider it to be a fraction and remove it from the process. Finally classify each individual as belonging to a cluster based on its sufficient statistics.

The limitations of refractionization method are: Let K_g be the true number of groups in the data, K_f be the number of fractions after splitting the data and K_m be the number of meta-observations after clustering each fraction.

- If $K_g \leq K_m$ then fractionization method is enough and we don't need to refractionize anymore. This is because the fractions are more likely to be good. In the case where $K_g \geq K_m$, some meta-observations must be based on clusters that have two or more groups. Hence they are not good;
- Refractionization will not recover the true groups where $K_g > K_f K_m$. There will be a fraction that contains more than K_m groups, leading to impurity.

3.2.3 Mrkd-Trees: an Implementation of EM Algorithm

The EM algorithm discussed in earlier chapters is relatively slow. The main drawback of EM algorithm is it scans the entire dataset on each iteration. Mrkd-tree suggested by Moore (1999) reduces the number of times the data is accessed. Mrkd stands for "multiple-resolution k-dimension", where k is the number of dimensions in the data.

An mrkd-tree is a binary tree consisting of nodes which contain pieces of information. Partitioning the dataset recursively the tree is built in such a way that the partitions adapt to the local density of the data. Nodes owning more number of points are called as denser regions and nodes that own fewer points are called as sparse regions. A node can be referred to as either a leaf node or a non-leaf node. A non-leaf node has two children, which in turn are nodes that own the two disjoint subpartitions of their parent's data points. Every node in the tree stores the following information:

- The bounds of the hyper-rectangle that contains all the objects owned by the node; and
- A set of statistics summarizing the data owned by the node.

If a node is a non-leaf node then it also contains:

- The value on which the partition of the data is made and the dimension to which this value refers.

These values are used while traversing the tree for the data in a particular region of the sample space. The tree is constructed using a top-down recursive procedure. The EM algorithm can be invoked on an mrkd-tree by calling some function m(·) on the root node; where m(·) is a function that returns the set of sufficient statistics for a given node [1]. If m(·) is called on a leaf node, r, then we calculate:

$$\bar{z}_j = P_r(x \in C_j \mid \bar{x}, \varphi) = \frac{f(\bar{x} \mid x \in C_j, \varphi) \Pr(x \in C_j, \varphi)}{\sum_{h=1}^{k} f(\bar{x} \mid x \in C_j, \varphi) \Pr(x \in C_j, \varphi)}$$

Where $\bar{x}$ is the centroid of the points owned by the node and C_j refers to cluster j for j=1,2,....,k. Returning the approximation to the sufficient statistic

$$\sum_{i=1}^{n_r} z_{ij} \approx \bar{z}_j \times n_r$$

$$\sum_{i=1}^{n_r} z_{ij} x_i \approx \bar{z}_j \times n_r \times \bar{x}$$

$$\sum_{i=1}^{n_r} z_{ij} x_i x_i^T \approx \bar{z}_j \times n_r \times \bar{x} \times s_r$$

For j = 1,2,…,k, where n_r is the number of points owned by node r and s_r is the sample covariance of the data owned by node r. If m(·) is called on a non-leaf node, the function is called recursively on its children.

3.2.4 Outlier Analysis:

Some data objects often exist that do not comply with the general behavior or models of the data. Such sets of objects which are inconsistent with the remaining set of data are called as outliers of the dataset. Outliers can be caused by inherent data variability. The data is more similar and doesn't have more variations, if the size of the dataset is very less. But when we consider millions of data objects as one dataset then the data varies widely. For example the salary of the chief executive officer of a company could naturally standout as an outlier among the salaries of the employees in a company.

Many clustering algorithms are trying to eliminate outliers or minimize the influence of outliers. In some cases, the outliers themselves might be of particular interest. Given a set of data points n and the number of outliers k, finding top k outlier points which are considerably dissimilar from the remaining data would fetch some analysis. The most effective ways for outlier detection are data visualization methods.

CHAPTER 4

WORKING AND RESULTS OF THREE MAJOR CLUSTERING TECHNIQUES

In this chapter we will see three important clustering algorithms that have revolutionized the clustering field in their respective era of discovery. The first one is architecture for efficient document clustering and retrieval on a dynamic collection of newspaper texts by Alan F. Smeaton, Mark Burnett, Francis Crimmins and Gerard Quinn. Second one is very famous BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) by Tian Zhang, Raghu Ramakrishnan, and Miron Livny. The last one is density-based approach DBSCAN by Martin Ester, Hans-peter Kriegel, Jörg Sander and Xiaowei Xu.

4.1 Efficient Document Clustering and Retrieval on a Dynamic Collection of Newspaper Texts:

This technique uses a number of short-cuts to make the process computable for large collection of online newspapers. This technique is extensible to dynamic updates of the data and it is implemented on an archive of the Irish Times newspaper.

A collection of newspaper articles are unlike the normal web documents because they are normally of the same magnitude, varying from a paragraph to a number of columns. News paper articles are timely based documents meaning they are related to their date of publication. But most distinguishing characteristic of news paper articles is that they are related to previously or subsequently published articles. For example an amendment initially is released to the media so that people can read and know about it. Then news comes out saying that the amendment is passed and will be starting from so and so date. Then the pros and cons about the amendment may be discussed by the news paper editorial. So this might take months or years together but all the articles published are related to that particular amendment. What this means for a collection of newspaper articles is that the dependencies between and among articles is potentially huge and these dependencies cannot be ignored when it comes to navigating the archive. This technique has explored document clustering as a technique for generating links between related documents as the collection is updated and presenting these links as a result of a search.

4.1.1 System Architecture and Algorithm:

The main reason for cluster analysis having very less impact on information retrieval involving large datasets is the overhead in generating the cluster structures. In this technique N × k similarity matrix was used for N documents, instead of a full N × N matrix structure. The value of k is a compromise between clustering efficiency and storage requirements against completeness of the clustering [13].

Meta-document descriptors are assigned to each document automatically by this clustering technique. The automatic classification allows the use of cluster descriptors to provide a higher level description of a document. These descriptors characterize and summarize the contents of the cluster. Agglomerative hierarchical type of clustering is used which was discussed earlier in chapter two. The method used is hierarchical because the cluster formed is hierarchically structured with closely related data objects at the leaves of the cluster tree, and less closely related at the root. Complete-link clustering method has been used which uses the smallest similarity within a cluster as the cluster similarity, and every data object within the cluster is related to every other with at least the similarity of the cluster. Complete link clustering is known to produce large number of small, tightly bound clusters which correspond to the large number of real world events reported in a newspaper.

For testing the appropriateness of this clustering technique the developers have collected 100 Mbytes of text from almost 34,768 individual newspaper articles. This technique requires the similarity of each document to every other document is known. For this purpose a conventional retrieval technique has been used with term weighting to compute inter-document similarity. Treating every document in turn as a query and using a custom search engine to rank it against the existing collection of documents. Document collection greater in size than about 10,000 cause problems, in addition to computational costs, the similarity matrix requires $N \times N-1$ units of memory per matrix element for storage. If each matrix element requires 4 bytes then for 1,000,000 documents the memory required would be 4Tb. For this problem a solution was developed in the current technique by implementing $N \times k$ matrix, where k is a constant value and $k << N$. Instead of considering similarity score of a document with all the rest of the documents in the collection, just consider the top k similarity scores which will produce good clusters with less computational overhead. Since the similarity scores drop rapidly down the hit list, the results can be comparable after some point of k. Testing done by developers revealed that k=30 gave results comparable to k=40, mainly because there are frequently few if any documents that are cluster candidates lower than the 30th place on the search list [13].

Because of $N \times k$ similarity matrix, small clusters are formed but these do not get integrated into a single overall cluster. The custom search engine used in this clustering procedure includes three thresholds designed to decrease the computation time but does this without the loss of retrieval effectiveness. The first one called as postings list threshold processes only some portion of the postings list entry for a given search term. The second one called as query term threshold processes only some portion of the search terms, depending on the length of the posting list entries. The third threshold creates a reduced set of document registers. While processing entries in the inverted file, only the first DR unique document identifiers will be assigned similarity scores.

Results:

Shared SUN UltraSparc with 128 Mbytes RAM but no local disks was used to index almost 35,000 documents of Irish Times collection in less than 2 hours. N × k similarity matrix was constructed in about 11.75 hours and the clusters were generated in about 8 minutes. Since, dynamic data updates were implemented in the system daily 300 news stories can be indexed and added to overall inverted file in about 15 seconds. Each document was added to the reduced similarity matrix by running it as a query and updating all matrix entries at a rate of almost 50 documents per minute, and the computation of the clustering takes about 8 minutes [13].

4.2 BIRCH: An Efficient Data Clustering Method for Very Large Databases

BIRCH clusters incrementally and dynamically the multi-dimensional input data points to produce best quality clustering with the available memory and time constraints. BIRCH is the first clustering technique in the field of databases to handle "noise" (data points that vary widely from the pattern of original dataset) effectively. Clustering is a procedure of identifying sparse and denser regions in a given dataset. Besides, the derived clusters can be visualized more efficiently and effectively than the original dataset [14].

Considering that the amount of memory is much less than the dataset size and in process of minimizing the time required for I/O, this technique has been designed and developed to deal with very large databases. This technique's I/O cost is linear in the size of the dataset. A single scan of the dataset is enough to produce good clusters and additional scans which are optional can be made to improve the quality of clusters.

4.2.1 Background of BIRCH

BIRCH technique is local meaning clustering decision is made without scanning all the data points or available clusters. It treats the data space as uneven distribution of data points and hence not every data object is equally important in clustering. Denser region in data space is considered to be a cluster while the sparse regions are avoided optionally assuming that they are outliers. It makes use of available memory to extreme by deriving the finest possible subclusters while minimizing I/O costs by using an in-memory, height-balanced and highly-occupied tree structure. Thus using all these features makes BIRCH's running time linearly scalable. *Given N d-dimensional data points in a cluster:* $\{\vec{X}_i\}$, *where* $i = 1, 2, \ldots\ldots, N$. *we define:*

$$\text{Centroid: } \vec{X}0 = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$$

Radius (R): average distance from member points to centroid

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}0)^2}{N}\right)^{\frac{1}{2}}$$

Diameter (D): average pair-wise distance within a cluster

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Given the centroids of two clusters, We define the centroid Euclidean distance (D0) and centroid Manhattan distance (D1) of the two clusters as:

$$D0 = ((\vec{X}0_1 - \vec{X}0_2)^2)^{\frac{1}{2}}$$

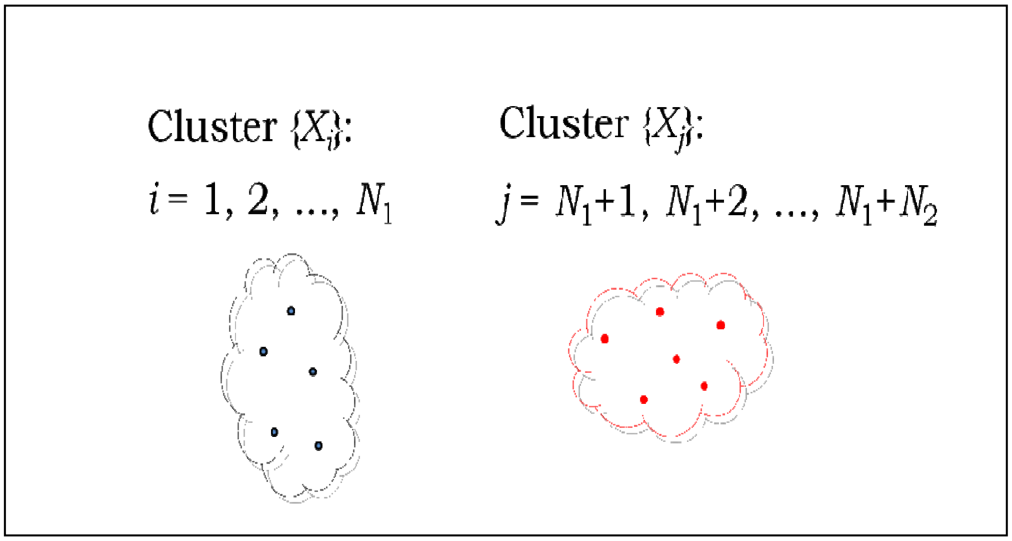
$$D1 = |\vec{X}0_1 - \vec{X}0_2| = \sum_{i=1}^{d} |\vec{X}0_1^{(i)} - \vec{X}0_1^{(i)}|$$

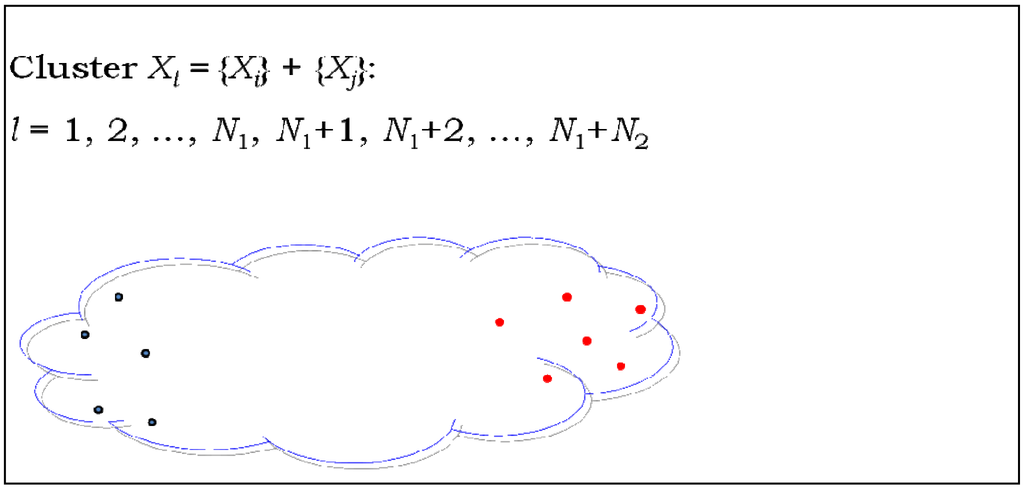
Given N_1 *d-dimensional* data points in a cluster: $\{\vec{X}_i\}$ where $i = 1, 2, \ldots, N_1$, and N_2 data points in another cluster: $\{\vec{X}_j\}$ where $j = N_1+1, N_1+2, \ldots, N_1+N_2$, the average inter-cluster distance D2, average intra-cluster distance D3 and variance increase distance D4 of the two clusters are defined as:

$$D2 = \left(\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{N_1 N_2}\right)^{\frac{1}{2}}$$

$$D3 = (\frac{\sum_{i=1}^{N_1+N_2}\sum_{j=1}^{N_1+N_2}(\vec{X}_i - \vec{X}_j)^2}{(N_1+N_2)(N_1+N_2-1)})^{\frac{1}{2}}$$

$$\mathrm{D4} = (\sum_{k=1}^{N_1+N_2}(\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2}\vec{X}_l}{N_1+N_2})^2 - \sum_{i=1}^{N_1}(\vec{X}_i - \frac{\sum_{l=1}^{N_1}\vec{X}_l}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2}(\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2}\vec{X}_l}{N_2})^2)^{\frac{1}{2}}$$




Cluster $X_l = \{X_i\} + \{X_j\}$:
$l = 1, 2, \ldots, N_1, N_1+1, N_1+2, \ldots, N_1+N_2$

4.2.2 Clustering Feature and CF Tree:

A clustering feature is a triple summarizing the details about the cluster. Given N d-dimensional data points in a cluster: {$\overrightarrow{X_i}$}, where i = 1, 2,..., N, the clustering feature vector of the cluster is defined as a triple: CF = (N, $\overrightarrow{LS}$, SS), where N is the number of data points in the cluster, $\overrightarrow{LS}$ is the linear sum of N data points and SS is the square sum of the N data points. Let CF_1 = (N_1, $\overrightarrow{LS}_1$, SS_1) and CF_2 = (N_2, $\overrightarrow{LS}_2$, SS_2) be the CF vectors of two disjoint clusters. Then the CF vector of the cluster formed by merging the two disjoint clusters is: $CF_1 + CF_2 = (N_1 + N_2, \overrightarrow{LS}_1 + \overrightarrow{LS}_2, SS_1 + SS_2)$.

CF Tree:

A CF tree is a height balanced tree consisting branching factor B and threshold T as two main parameters. Each non-leaf node contains at most B entries in the form of [CF_i, $child_i$], where i = 1,2,..., B. $child_i$ is a pointer to its i-th child node, and CF_i is the CF of the sub-cluster represented by this child [14]. A leaf node contains at most L entries, each of the form [CF_i,], where i = 1,2,..., L. Each leaf node also has two pointers "previous" and "next" which are used to chain all leaf nodes together for efficient scans. All entries in a leaf node satisfy a threshold requirement with respect to a threshold value T. The tree size is smaller if the T value is larger. If a node is required to fit in a page of size P, once the dimension d of data space is given then the sizes of leaf and non-leaf entries are known and finally B and L are determined by P. So performance tuning can be done by varying P value.

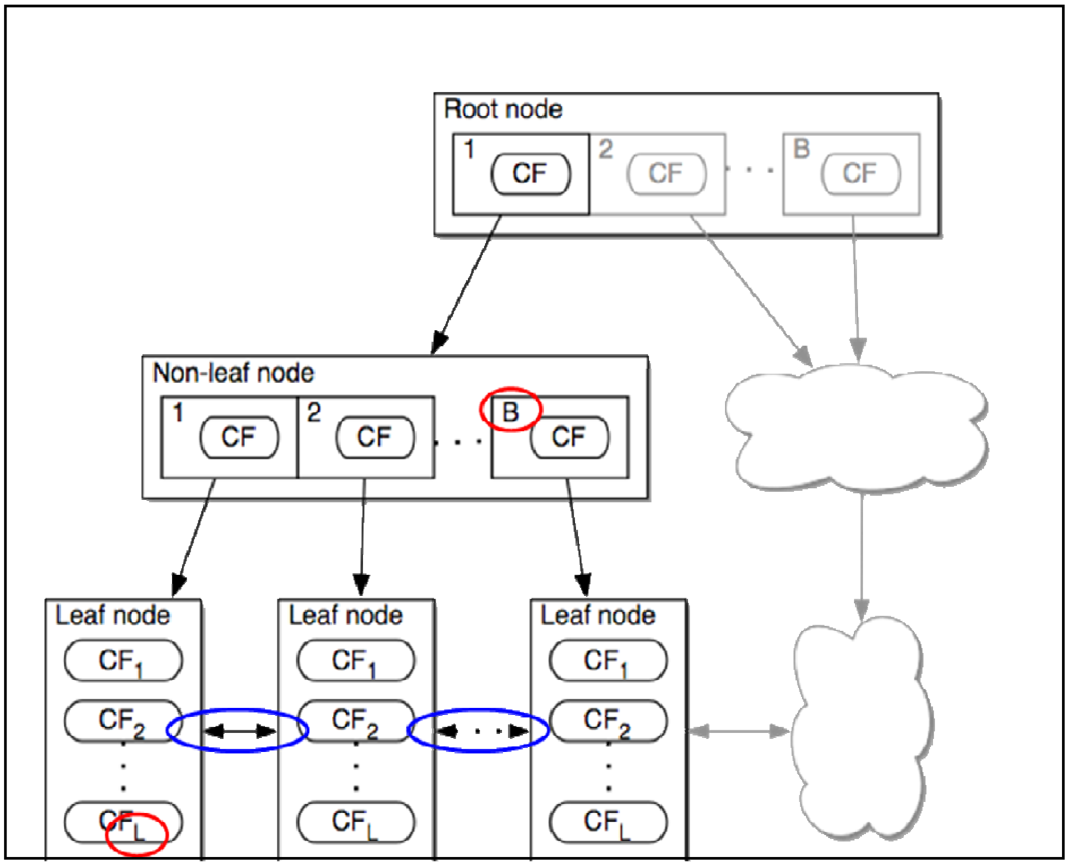


Figure 12. Different levels of a CF Tree

Insertion into a CF Tree:

- Finding the appropriate leaf by recursively descending the CF tree starting from the root and choosing the closest child node according to a chosen distance metric: D0, D1, D2, D3 or D4.
- If the closest CF leaf node cannot absorb (violating the threshold

condition), make a new CF entry. If there is no room for the new leaf node then split the parent node.

- Modify the path to the leaf by updating CF's on the path or splitting nodes.
- If the space is not enough then threshold value can be increased, by doing this CF's absorb more data.
- Due to the size restriction of each node they can hold only limited number of entries and because of this; natural clusters are not formed always.

BIRCH Algorithm

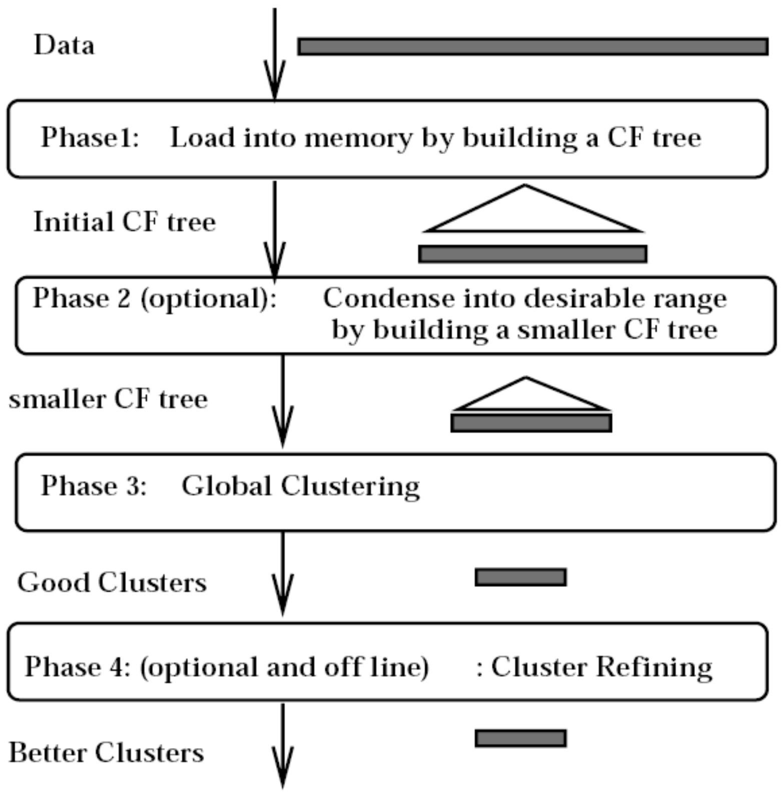

Data
Phase1: Load into memory by building a CF tree
Initial CF tree
Phase 2 (optional): Condense into desirable range by building a smaller CF tree
smaller CF tree
Phase 3: Global Clustering
Good Clusters
Phase 4: (optional and off line) : Cluster Refining
Better Clusters

Figure: 13 Different phases of BIRCH algorithm

Phase 1 scans all the data and builds an initial in-memory CF tree using the available memory and recycling space on disk. This CF tree represents the clustering information of the dataset as clear as possible. The denser regions are grouped as fine subclusters and sparse data points are removed as outliers, and phase 1 creates an in-memory summary of the data. This phase is fast because there are no I/O operations needed and the problem of clustering the original dataset is reduced to a smaller problem of clustering subclusters in the leaf entries.

Phase 2 is optional acting as a bridge to phase 1 and phase 3; it scans the leaf entries in the initial CF tree to rebuild a smaller CF tree, and removes more outliers along with grouping crowded subclusters into larger ones.

Phase 3 is the global clustering phase where existing cluster algorithm is used on CF entries. This phase helps in fixing the problem where natural clusters span nodes. After phase 3, a set clusters are obtained that captures the major distribution pattern in the data. But still some minor and localized inaccuracies might exist.

Phase 4 is optional and entails the cost of additional passes over the data to correct those inaccuracies and refine the clusters further. In phase 4 the centroids of the clusters produced by phase 3 as seeds are used and data points are redistributed to its closest seed to obtain a set of new clusters.

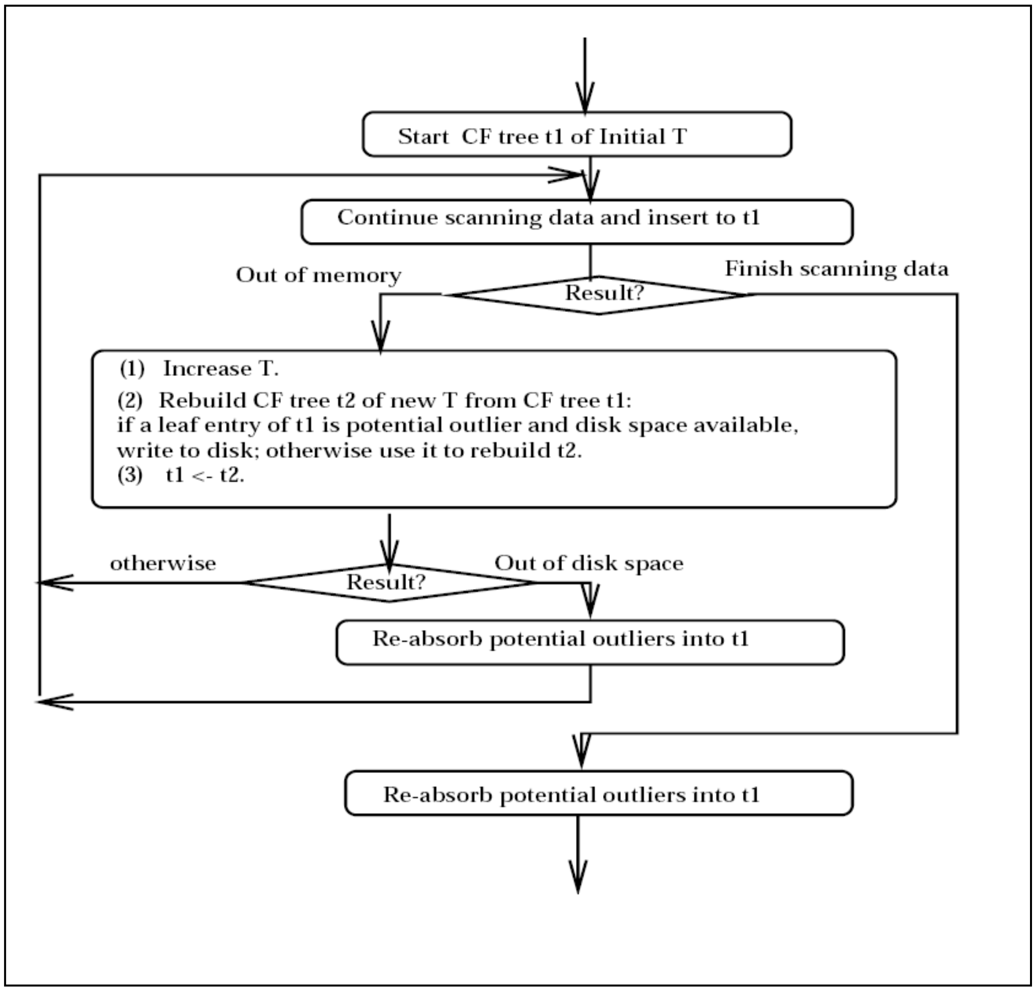

Start CF tree t1 of Initial T
Continue scanning data and insert to t1
Out of memory
Result?
Finish scanning data
(1) Increase T.
(2) Rebuild CF tree t2 of new T from CF tree t1:
if a leaf entry of t1 is potential outlier and disk space available,
write to disk; otherwise use it to rebuild t2.
(3) t1 <- t2.
otherwise
Result?
Out of disk space
Re-absorb potential outliers into t1
Re-absorb potential outliers into t1

Figure 14. Control flow of phase 1

In the above diagram we can see the detailed procedure of phase 1. Starting with an initial threshold value, it scans the entire data and inserts points into the tree. If the method runs out of memory while scanning the data then it increases the threshold value, rebuilds a new smaller CF tree, by re-inserting the leaf entries of the old tree. After the old leaf entries have been re-inserted, the scanning of the data is resumed from the point at which it was stopped.

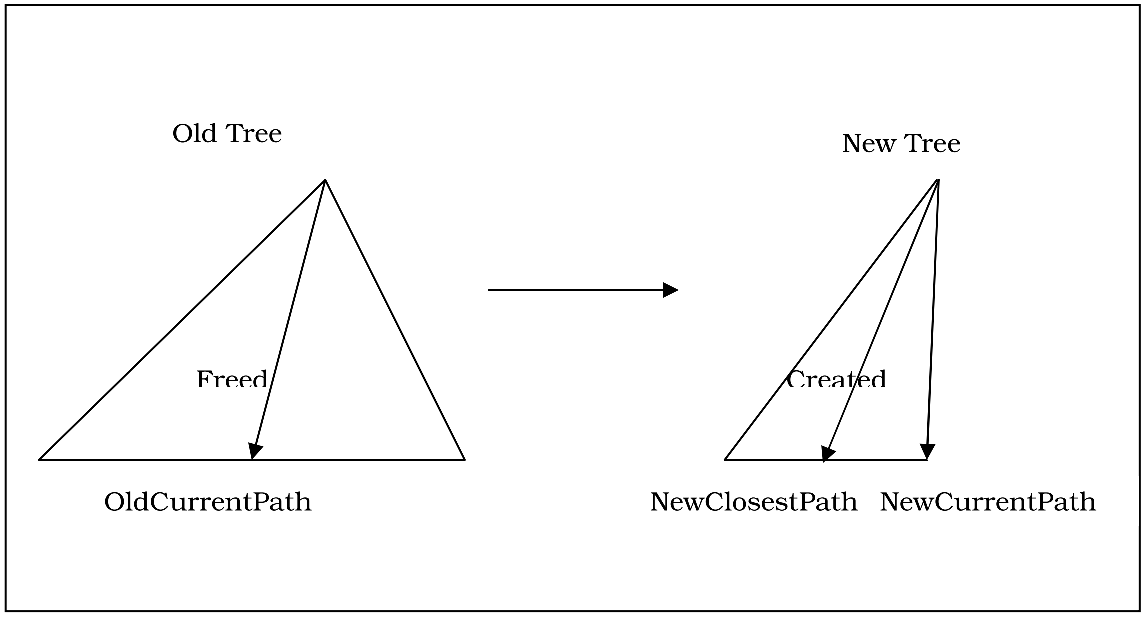


Figure 15. Rebuilding CF Tree

With the natural path order, it scans and frees the old tree path by path and at the same time, creates the new tree path by path. “OldCurrentPath” starts with the left most path of the old tree and new tree starts with the null. For OldCurrentPath the algorithm is [14]:

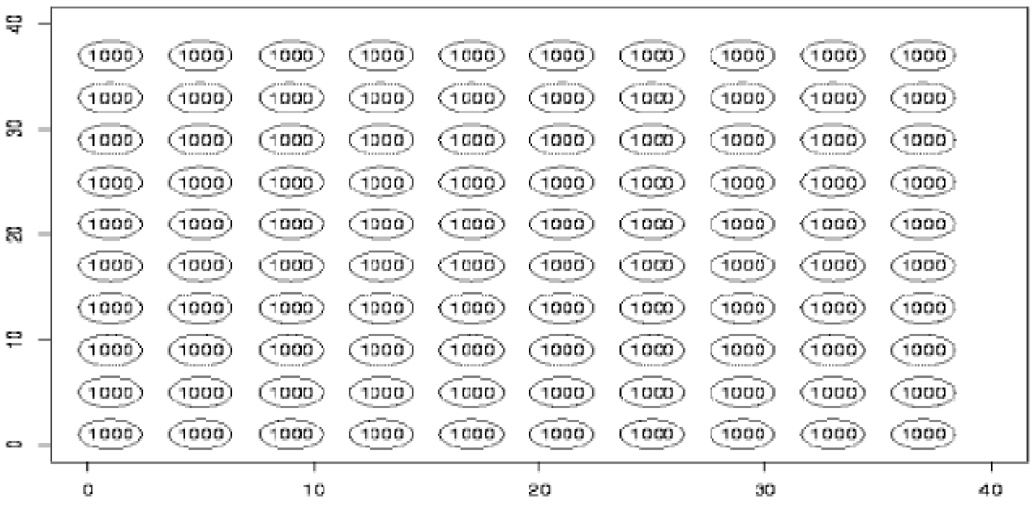
- Create the corresponding NewCurrentPath in the new tree by adding nodes to the new tree exactly the same as in the old tree, so that the new tree ever becomes larger than old tree.
- With the new threshold, each leaf entry in “OldCurrentPath” is tested against the new tree to see if it can fit in the “NewClosestPath” that is found top-down with the closest criteria in the new tree. If it is true then the “NewClosestPath” is before the “NewCurrentPath” then it is inserted in the “NewClosestPath”.

- Once all the entries in "OldCurrentPath" are processed, the unwanted nodes along "OldCurrentPath" can be freed.
- "OldCurrentPath" is assigned to the next path in the old tree if there is one and the above steps are repeated.

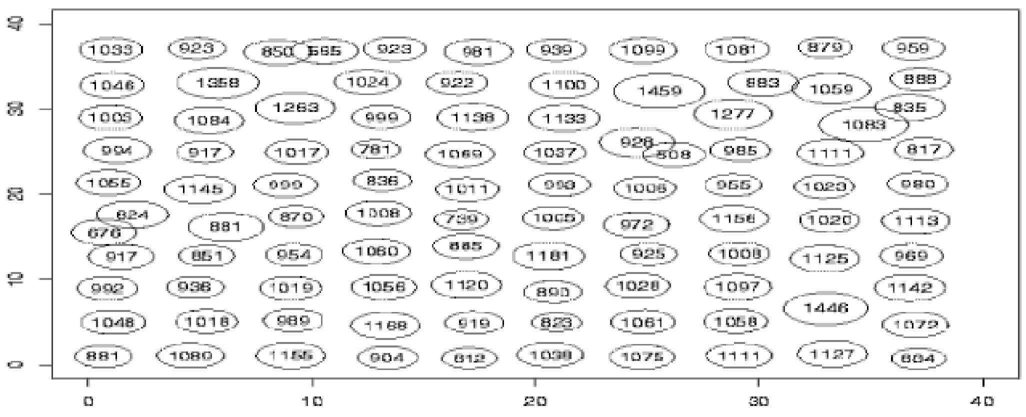
Results:

- *Input parameters:*
- Memory (*M*): 5% of data set
- Disk space (*R*): 20% of *M*
- Distance equation: *D*2
- Quality equation: weighted average diameter (*D*)
- Initial threshold (*T*): 0.0
- Page size (*P*): 1024 bytes

Intended Clustering Result:



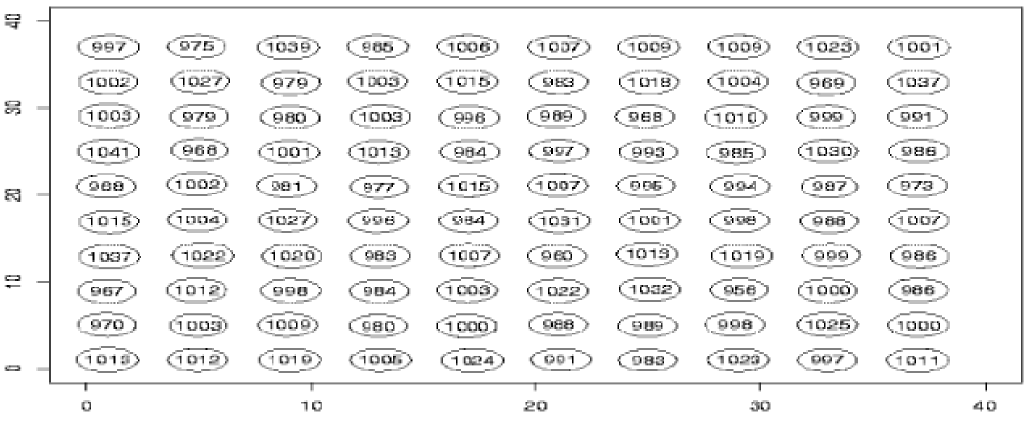
CLARANS Clustering



DS	Time	D	# Scan	DS	Time	DS	# Scan
1	932	2.10	3307	1o	794	2.11	2854
2	758	2.63	2661	2o	816	2.31	2933
3	835	3.39	2959	3o	924	3.28	3369

Table 1: Results of CLARANS

BIRCH Clustering:



DS	Time	D	#Scan	DS	Time	D	# Scan
1	11.5	1.87	2	1o	13.6	1.87	2
2	10.7	1.99	2	2o	12.1	1.99	2
3	11.4	3.95	2	3o	12.2	3.99	2

Table 2: Results of BIRCH

Initial scan is from disk and subsequent scans are in memory. When using Phase 4, page size can vary from 256 to 4096 without much effect on the final results. Results generated with low memory can be compensated for by multiple iterations of phase 4.

4.3 DBSCAN

DBSCAN is a density-based clustering technique, designed to discover efficient and good clusters with arbitrary shapes. This clustering technique requires only one input parameter and helps user in determining an appropriate value for it. The detailed introduction of this density-based technique has been discussed earlier in chapter 2.

4.3.1 A Density Based Notion of Clusters

Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). These regions may have an arbitrary shape and the points inside a region may be arbitrarily distributed. The main idea is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points.

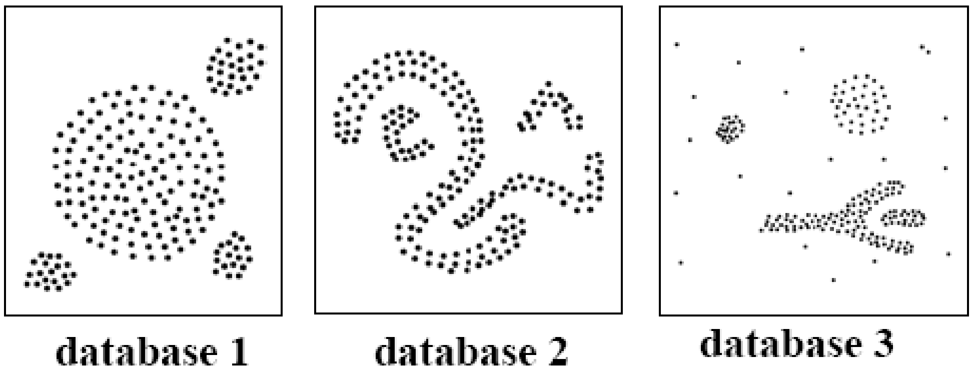


Figure 16: Sample databases showing obvious clusters and noise

The Eps-neighborhood of a point p, denoted by N_{Eps} (p), is defined by N_{Eps} (p) = {q Є D | dist (p, q) ≤ Eps} where the shape of the neighborhood is determined by the choice of a distance function for two points p and q, denoted by dist(p, q). Minimum number of points within the specified distance metric is denoted as "MinPts". But there are two kinds of points in a cluster, points inside the cluster are called as core points and points on the border of the cluster are called as border points. In general an Eps-neighborhood of a border point has less number of points when compared to Eps-neighborhood of a core point. Therefore the minimum number of points should be set to a low value in order to include all points belonging to the same cluster. For every point p in a cluster C there is a point q in C so that p is inside of the Eps-neighborhood of q and N_{Eps} (q) contains at least MinPts points [15].

An object *p* is directly density-reachable from object *q* with respect to Eps and MinPts, if *p* is within the Eps–neighborhood of *q* which contains at least a minimum number of points (MinPts).

An object *p* is density-reachable from object *q* with respect to Eps and MinPts, if there is a chain of objects $p_1, \ldots\ldots, pn$, $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i. Density-reachability is a canonical extension of direct density-reachability. This relation is transitive but not symmetric [15].

An object *p* is density-connected to object *q* with respect to Eps and MinPts, if there is an object $o \in D$ such that both *p* and *q* are density-reachable from *o* with respect to Eps and MinPts [15].

Let D be a database of points. A cluster C with respect to Eps and MinPts is a non-empty subset of D satisfying the following conditions [15]:

The Algorithm:

First, DBSCAN begins with an arbitrary point p and retrieves all points that are density-reachable from p with respect to Eps and MinPts. If p is a core point then DBSCAN yields a cluster with respect to Eps and MinPts, but if p is a border point then no point is density-reachable from p and DBSCAN shifts to next point in the database. Global values are used for Eps and MinPts because of which DBSCAN can merge two clusters into one cluster, if two clusters of different density are close to each other.

```
DBSCAN (SetOfPoints, Eps, MinPts)
// SetOfPoints is UNCLASSIFIED
    ClusterId: = nextId (NOISE);
        FOR i FROM 1 TO SetOfPoints.size DO
            Point: = SetOfPoints.get(i);
            IF Point.ClId = UNCLASSIFIED THEN
            IF ExpandCluster (SetOfPoints, Point,ClusterId, Eps, MinPts)
            THEN ClusterId: = nextId (ClusterId)
            END IF
            END IF
        END FOR
END; // DBSCAN
```

In the above algorithm drawn from [15] SetOfPoints is either the whole database or a discovered cluster from a previous run. Eps and MinPts are global density parameters determined either manually or according to the heuristics. The function SetOfPoints.get (i) returns the ith element of SetOfPoints. ExpandCluster is the main function that is responsible for the working of DBSCAN clustering technique.

Results:

- DBSCAN is more effective in discovering clusters of arbitrary shape than CLARANS.
- DBSCAN can identify noise whereas CLARANS cannot.
- Runtime of CLARANS is comparatively very large.

- CLARANS cannot be applied for large databases.

Results show that DBSCAN outperforms CLARANS by a factor of at least 100 in terms of efficiency.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The main objective of this thesis was to survey the most important clustering algorithms and determine which of them can be used for clustering large datasets. Extending or improving basic models of clustering as discussed in chapter 3 can help in some ways to deal with large datasets but the most successful clustering methods stored summary statistics in trees. Building a tree requires only single scan of data and inserting a new object into an existing tree is usually very simple. By limiting the amount of memory available in the tree building process, it is possible for the tree to adapt to fit into main memory.

This thesis focuses on inspection of most important clustering algorithms and further we have discussed the key concepts that allow the current clustering methods to manage very large datasets. Determining clusters of arbitrary shape, identifying outliers as sparse regions and providing computational speed-ups through ignoring sparse regions of the data space were the essential steps found in most of the current clustering methods.

An optimally efficient tree-based data structure should be ascertained for clustering problems. Multi-resolution clustering techniques (i.e. ability to detect clusters with in a cluster) need to be formalized. The ability to cluster data arriving in a constant stream should be considered. Tree-based data structures within the online systems should be explored as they are likely to be very effective. The below is a list of all the clustering methods and their corresponding run times along with other specifications.

	Run Time $O(\cdot)$	Estimate k	Arbitrary Shapes	Handle Noise	One Scan of Data	Will Stop
k-means	n					
k-medoids	n^2			✓		
Agglomerative	n^3					✓
Divisive	n^2					✓
EM Algorithm	n					
Fract.	n				✓	✓
Refract.	n	✓				
BIRCH	n			✓	✓	✓
Mrkd-EM	n.log n				✓	
DBSCAN	n.log n	✓	✓	✓	✓	✓
DENCLUE	n	✓	✓	✓	✓	✓
DBCLASD	n.log n	✓	✓	✓	✓	✓
STING	n	✓	✓	✓	✓	✓
SOON	n^2	✓				

Table 3: Run Times and Properties of Clustering Algorithms

BIBLIOGRAPHY

1. Clustering Large Datasets by D. P. Mercer
http://www.stats.ox.ac.uk/~mercer/documents/Transfer.pdf

2. Amit Singhal, ‘Modern Information Retrieval: A Brief Overview’, IEEE Data Engineering Bulletin, Volume 24, pages 35-43, 2001.

3. C. J. Van Rijsbergen, ‘Information Retrieval’, Second Edition, Chapters 1, 6 & 7, Information Retrieval Group, University of Glasgow, London: Butterworths, 1979.

4. Ian H. Witten, Alistair Moffat, and Timothy C. Bell, ‘Managing Gigabytes’, Second Edition, Chapter 4, Morgan Kaufmann Publishers, Inc, San Francisco, May 1999.

5. Ricardo Baeza Yates, Berthier Riberio Neto, ‘Modern Information Retrieval’, Chapter 1, Addison Wesley, Addison Wesley Longman, 1999.
http://people.ischool.berkeley.edu/~hearst/irbook/1/node2.html.

6. T. Graepel. Statistical physics of clustering algortihms. Technical Re-port 171822, FB Physik, Institut fur Theoretische Physic, 1998.

7. Zoubin Ghahramani, ‘Unsupervised Learning’, Chapters 1 & 2, University of College, London, September 2004.

8. Data Mining - Concepts and Techniques by Jiawei Han and Micheline Kamber. Chapter 8(Cluster Analysis).

9. Incremental Model-Based Clustering for Large Datasets With Small Clusters by Chris Fraley, Adrian Rafteryz and Ron Wehrensy : Technical Report No. 439

10. Dr. E. Garcia, ‘The Classic Vector Space Model: Description, Advantages and Limitations of Vector Space Model’, Article 3 of series Term Vector Theory and Keyword Weights.
http://www.miislita.com/term-vector/term-vector-3.html

11. S. E. Robertson, C. J. van Rijsbergen and M. F. Porter, 'Probabilistic models of Indexing and Searching', Proceedings of the 3rd annual ACM conference on Research and development in information retrieval, Cambridge, England, Page(s): 35 - 56, June 1980.

12. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications by Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos and Prabhakar Raghavan.

13. Architecture for efficient document clustering and retrieval on a dynamic collection of newspaper texts by Alan F. Smeaton, Mark Burnett, Francis Crimmins and Gerard Quinn.

14. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) by Tian Zhang, Raghu Ramakrishnan, and Miron Livny.

15. Density-based approach DBSCAN by Martin Ester, Hans-peter Kriegel, Jörg Sander and Xiaowei Xu.

16. Information retrieval group by Keith van Rijsbergen, Test Collections.
http://ir.dcs.gla.ac.uk/

17. Kiran Pai, 'A simple way to read an XML file in Java', 2002.
http://www.developerfusion.com/code/2064/a-simple-way-to-read-an-xml-file-in-java/

18. HappyCoders,'TokenizingJavasourcecode'(n.d.)
http://www.java.happycodings.com/Core_Java/code84.html

19. Martin Porter, 'The Porter Stemming Algorithm', Jan 2006.
http://tartarus.org/~martin/PorterStemmer/.

20. Gerald Salton and Chris Buckley, 'Improving Retrieval Performance by Relevance Feedback', Readings in information retrieval, Page(s): 355-364, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1997.

21. Diane Kelly and Nicholas J. Belkin, 'Exploring Implicit Sources of User Preferences for Relevance Feedback During Interactive Information Retrieval', School of Communication, Information and Library Studies, Rutgers, The State University of New Jersey.

VITA

Graduate College
University of Nevada, Las Vegas

Vasanth Nemala

Home Address:
1555 E Rochelle Ave, Apt#247
Las Vegas, NV 89119

Degrees:
Bachelor of Technology in Computer Science, 2007
Jawaharlal Nehru Technological University

Thesis Title: Efficient Clustering Techniques for Managing Large Datasets

Thesis Examination Committee:
Chair Person, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr. Ajoy K. Datta, Ph.D.
Committee Member, Dr. Laxmi P. Gewali, Ph.D
Graduate College Representative, Dr. Muthukumar Venkatesan, Ph.D.