A Survey on Clustering Algorithms and Complexity Analysis

Sabhia Firdaus[1], Md. Ashraf Uddin[2]

[1]Department of Computer Science and Engineering, Bangladesh University of Business and Technology
Dhaka, Bangladesh

[2]Department of Computer Science and Engineering, Jagannath University
Dhaka, Bangladesh

Abstract
Clustering is a technique to group the data objects in such a way that the data objects of a group are similar to each other in that group and dissimilar to the data objects of other groups. It is being performed for the diverse purpose such as statistical data analysis and exploratory data mining. This algorithm is applied based on the data object types, the complexity of the algorithm, and appropriateness for a specific grouping. The list of cluster algorithm includes possibly more than one hundred published clustering algorithms but most of them do not provide models for cluster. In this paper, we analyze the complexity of the mostly used clustering algorithms.

Keywords- *Clustering technique, clustering algorithm' data grouping, algorithm complexity*

1. Introduction

Classification and cluster are important techniques that partition the objects that have many attributes into meaningful disjoint subgroups so that objects in each group are more similar to each other in the values of their attributes than they are to objects in other group.
There is a major difference between cluster analysis and classification. In supervised classification, the classes are defined, the user already knows what classes there are, and some training data that is already labeled by their class membership is available to train or build a model. In cluster analysis, one does not know what classes or clusters exist and the problem to be solved is to group the given data into meaningful cluster. Just like application of supervised classification, cluster analysis has applications in many different areas such as in marketing, medicine, business. Practical applications of cluster analysis have also been found in character recognition, web analysis and classification of documents, classification of astronomical data and classification of objects found in an archaeological study. Work on clustering has been carried out in statistics for many years. Most of the algorithms developed are based on some concept of similarity or distance so that groups of objects that are similar or near to each other and dissimilar or far away from other objects may be put together in a cluster. The technique may appear simple but it can be rather challenging because to find objects that are similar amongst a large collection of objects requires comparing each object with every other object which may be prohibitively expensive for large sets. Also, the cost of computing distance between objects grows as the number of attributes grows. Finally it is easier to compute distances between objects with numeric attributes but computing distances between categorical attributes is more difficult.

The classical clustering algorithms developed in statistics assumed small datasets but with the advent of computing, bar codes and the web, users wish to apply cluster analysis to large datasets. Old methods have therefore been modified to manage large datasets and new methods are being developed.

1.2 Issues of Cluster analysis [1]

- Finding the desired features of cluster analysis
- How similarity should be measured given the datasets that are being analyzed.
- Finding the best cluster analysis method for the given dataset
- How should a very large data set be clustered efficiently
- How should data with a large number of attributes be clustered efficiently
- How should the result of cluster analysis be evaluated

1.3 Requirements of Clustering in Data mining [6]:

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

1.4 Type of clustering method

The clustering methods may be divided into the following categories:

Partitional method: Partitional methods obtain a single level partition of objects. These methods usually are based on greedy heuristics that are used iteratively to obtain a local optimum solution. Partitional clustering algorithm includes Minimum Spanning Tree, Squared Error Clustering Algorithm, K-means clustering, Nearest neighbor algorithm, partitioning around medoids (PAM) algorithm, Bond Energy Algorithm, Clustering with Genetic Algorithm, Clustering with Neural network.

Hierarchical method: Hierarchical methods obtain a nested partition of the objects resulting in a tree of clusters. This methods either start with one cluster and then split into smaller clusters(called divisive or top down) or start with each object in an individual cluster and then try to merge similar clusters into larger and larger clusters(called agglomerative or bottom up). There are some other clustering of this type such as BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies), ROCK (A Hierarchical Clustering Algorithm for Categorical Attributes). Chameleon (A Hierarchical Clustering Algorithm Using Dynamic Modeling).

Density- Based method: In this class of methods, typically for each data point in a cluster, at least a minimum number of points must exist within a given radius. Density based methods include DBSCAN(A Density-Based Clustering Method on Connected Regions with Sufficiently High Density), OPTICS(Ordering Points to Identify the Clustering Structure), DENCLUE(Clustering Based on Density Distribution Functions) [1].

Grid-based method: In this class of methods, the objects space rather than the data is divided into a grid. Grid partitioning is based on characteristics of the data and such methods can deal with non-numeric data more easily. Grid-based methods are not affected by data ordering. STING (Statistical Information Grid), Wave Cluster (Clustering Using Wavelet Transformation), CLIQUE are some example of grid-based method [1].

Model-based method: A model is assumed, perhaps based on a probability distribution. Essentially the algorithm tries to build clusters with a high level of similarity within them and a low level of similarity between them. Similarity measurement is based on the mean values and the algorithm tries to minimize error function. Expectation-Maximization, Conceptual Clustering, and Neural Network Approach is the example of model-based methods [3].

Clustering High-Dimensional Data: CLIQUE (A Dimension-Growth Subspace Clustering Method), PROCLUS (A Dimension-Reduction Subspace Clustering Method), Frequent Pattern-Based Clustering Methods are used for high dimensional data.

Constraint-Based Cluster Analysis: Constraint-based clustering finds clusters that satisfy user-specified preferences or constraints. Depending on the nature of the constraints, constrain-based clustering may adopt rather different approaches. Few categories of constraints include constraints on individual objects, constraints on the selection of clustering parameter, constraints on distance or similarity functions [3].

2. Cluster Algorithms

Cluster analysis has been an area of research for several decades and there are too many different methods for all to be covered even briefly. Many new methods are still being developed. In this section we discuss some popular and mostly used clustering algorithm and present their complexity, advantage and disadvantage.

2.1 Road Map

In this section we now discuss the clustering algorithms.

K-means: K-means is the simplest and most popular classical clustering method that is easy to implement.

The classical can only be used if the data about all the objects is located in the main memory. The method is called K-means since each of the K clusters is represented by the mean of the objects within it [1].

Expectation Maximization Method: In contrast to the K-means method, the Expectation Maximization(EM) method is based on the assumption that the objects in the dataset have attributes whose values are distributed according to some(unknown) linear combination(or mixture) of simple probability distributions. While the K-means method involves assigning objects to cluster to minimize within-group variation, the EM method assigns objects to different clusters with certain probabilities in an attempt to maximize expectation (or likelihood) of assignment [1].

PAM algorithm: The PAM (partitioning around medoids) algorithm, also called the K-medoids algorithm, represents a cluster by a medoid. Using a method is an approach that handles outliers well [2].

Bond Energy Algorithm: The bond energy algorithm (BEA) was developed and has been used in the database design area to determine how to group data and how to physically place data on a disk. It can be used to cluster attributes based on usage and then perform logical or physical design accordingly [2].

Clustering with Genetic Algorithms
Clustering with Neural Networks: Neural networks (NNs) that use unsupervised learning attempt to find features in the data that characterize the desired output. They look for clusters of like data. These types of NNs are often called self-organizing neural networks. There are two basics of unsupervised learning: noncompetitive and competitive [2].

Nearest Neighbor Algorithm: An algorithm similar to the single link technique is called the nearest neighbor algorithm. With this serial algorithm, items are iteratively merged into the existing clusters that are closet. In this algorithm a threshold, t is used to determine if items will be added to existing clusters or if a new cluster is created [2].

Agglomerative Algorithm: Agglomerative algorithms start with each individual item in its own cluster and iteratively merge clusters until all items belong in one cluster. Different agglomerative algorithms differ in how the clusters are merged at each level [2].

Divisive Clustering: With divisive clustering, all items are initially placed in one cluster and clusters are repeatedly split in two until all items are in their own cluster. The idea is to split up clusters where some elements are not sufficiently close to other elements [2].

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) Algorithm: BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering (at the initial micro clustering stage) and other clustering methods such as iterative partitioning (at the later macro clustering stage). It overcomes the two difficulties of agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step [3].

ROCK (A Hierarchical Clustering Algorithm for Categorical Attributes): ROCK (Robust clustering using links) is a hierarchical clustering algorithm that explores the concept of links (the number of common neighbors between two objects) for data with categorical attributes [3].

CURE (Clustering Using Representative) algorithm: One objective for the CURE clustering algorithm is to handle outliers well. It has both a hierarchical component and a partitioning component [2].

Chameleon (A Hierarchical Clustering Algorithm Using Dynamic Modeling): Chameleon is a hierarchical clustering algorithm that uses dynamic modeling to determine the similarity between pairs of clusters. It was derived based on the observed weak-ness of the two hierarchical clustering algorithms: ROCK and CURE [3].

DBSCAN (Density-Based Clustering Method based on connected Regions with sufficiently High Density): DBSCAN (Density –Based Spatial Clustering of Applications with Noise) is a density based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial database with noise. It defines a cluster as a maximal set of density-connected points [3].The method was designed for spatial database but can be used in other applications. It requires two input parameters: the size of the neighborhood(R) and the minimum points in the neighborhood (N). Essentially these two parameters determine the density within the clusters the user is willing to accept since they specify how many points must be in a region n. The number of points not only determines the density of acceptable clusters but it also determines which objects in their neighborhood. The size parameter R determines the size of the clusters found. If R is big enough, there would be one big cluster and no outliers. If R is small, there will be small dense clusters and there might be many outliers.

OPTICS (Ordering Points to Identify the Clustering Structure): Although DBSCAN can cluster objects given input parameters such as ε and MinPts, it still leaves the user with the responsibility of selecting parameter values that will lead to the discovery of acceptable clusters [3].

DENCLUE (Clustering Based on Density Distribution Functions): DENCLUE is a clustering method based on a set of density distribution functions. The method is built on the following ideas :(1) the influence of each data point can be formally modeled using a mathematical function called an influence function, which describes the impact of a data point within its neighborhood;(2) the overall density of the data space can be modeled analytically as the sum of the influence function applied to all data points and (3) clusters can then be determined mathematically by identifying density attractors, where density attractors are local maxima of the overall density function[3].

STING (Statistical Information Grid): STING is a grid-based multi resolution clustering technique in which the spatial area is divided into rectangular cells. There are usually several levels of such rectangular cells corresponding to different levels of resolution, and these cells form hierarchical structure: each cell at high level is partitioned to form a number of cells at the next lower level. Statistics information regarding the attributes in each grid cell (such as the mean, maximum, and minimum values) is precomputed and stored [3].

Wave Cluster (Clustering Using Wavelet Transformation): Wave cluster is multi resolution clustering algorithm that first summarizes the data by imposing a multidimensional grid structure onto the data space. It then uses a wavelet transformation to transform the original features space, finding dense regions in the transformed space. In this approach, each gird cell summarizes the information of a group of points that map into the cell. This summary information typically fits into main memory for use by the multi resolution wavelet transform and the subsequent cluster analysis [3].

Conceptual Clustering: Conceptual clustering is a form of clustering in machine learning that, given a set of unlabeled objects, produces a classification scheme over the objects. Unlike conventional clustering, which primarily identifies groups of like objects, conceptual clustering goes one step further by also finding characteristics descriptions for each group where each group represents a concept or class. Hence, conceptual clustering is two step process: clustering is performed first, followed by characterization. Here, clustering quality is not solely a function of the individual objects. Rather, it incorporates factors such as the generality and simplicity of the desired concept descriptions [3].

CLIQUE (A Dimension Growth Subspace Clustering Method): CLIQUE was the first algorithm proposed for dimension-growth subspace clustering in high-dimensional space. In dimensional-growth subspace clustering, the clustering process starts at single-dimensional subspaces and grows upward to higher-dimensional ones [3].

PROCLUS (A Dimensional Reduction Subspace Clustering Method): PROCLUS is a typical dimension reduction subspace clustering method. That is instead of starting from single-dimensional attribute by finding an initial approximation of the clusters in the high dimensional attribute space. Each dimension is then assigned a weight for each cluster, and the updated weights are used in the next iteration to regenerate the clusters [3].

Frequent Pattern Based Clustering Methods: Frequent pattern mining the name implies, searches for patterns (such as sets of items or objects) that occur frequently in large data sets. Frequent pattern mining can lead to the discovery of interesting associations and correlations among data objects. The idea behind frequent pattern-based cluster analysis is that the frequent patterns discovered may also indicate clusters [3].

2.2 K-means Algorithm

The K-means algorithm may be described as follows:

1. Select the number of clusters. Let this number be K
2. Pick K seeds as centroids of the k clusters. The seeds may be picked randomly unless the user has some insight into the data.
3. Compute the Euclidean distance of each object in the dataset from each of the centroids.

4. Allocate each object to the cluster it is nearest to base on the distances computer in the previous step.
5. Compute the centroids of the clusters by computing the means of the attribute values of the objects in each cluster.
6. Cheek if the stopping criterion has been met (e.g. the cluster membership is unchanged) if yes go to step 7. If not, go to step 3.
7. [optional] One may decide to stop at this stage or to split a cluster or combine two clusters heuristically until a stopping criterion is met.

2.2.1 Starting values for the K-means method

Often the user has little basis for specifying the number of clusters and starting seeds. This problem may be overcome by using an iterative approach. For example, one may first select three clusters and choose three starting seeds randomly. Once the final clusters have been obtained, the process may be repeated with a different set of seeds. Attempts should be made to select seeds that are as far away from each other as possible. Also, during the iterative process if two clusters are found to be close together, it may be desirable to merge them. Also, a large cluster may be split in two if the variance within the cluster is above some threshold value. Another approach involves finding the centroid of the whole dataset and then perturbing this centroid value to find seeds. Yet another approach recommends using a hierarchical method like the agglomerative method on the data first, since that method does not require starting values, and then using the results of that method as the basis for specifying the number of clusters and starting seeds. Yet another approach is based on using a number of small samples of the given data and using random seeds for each simple. The result of cluster analysis of these samples can help in finding seeds for the whole data [1].

2.2.2 Ways for making K-means algorithm more scalable

A recent approach to scaling the k-means algorithm is based on the idea of identifying three kinds of regions in data: regions that are compressible, regions that must be maintained in main memory and regions that are discard able. An object is discard able if its membership in a cluster is ascertained. An object is compressible if it is not discard able but belongs to a tight sub cluster. A data structure known as a clustering feature is used to summarize objects that have been discarded or compressed. If an object is neither discard able nor compressible, then it should be retained in main memory. To achieve scalability, the iterative clustering algorithm may only include the clustering features of the compressible objects and the objects that must be retained in main memory, thereby turning a secondary memory based algorithm into a main memory based algorithm. An alternative approach to scaling the k-means algorithm explores the micro clustering idea, which first groups nearby objects into micro clusters and then performs k-means clustering on the micro clusters [1].

2.2.3 Strength and Weakness of K-means Algorithm

K-means is simple and can be used for a wide variety of data types. It is also quite efficient even though multiple runs are often performed. Some variants including bisecting k-means are even more efficient and are less susceptible to initialization problems. K-means is an iterative greedy method. A number of iterations are normally needed for convergence and therefore the dataset is processed a number of times. If the data is very large and cannot be accommodated in the main memory the process may become inefficient. Although the k-means method is most widely known and used, there are a number of issues related to the method that should be understood.

1. The k-means method needs to compute Euclidean distances and means of the attribute values of objects within a cluster. The classical algorithm therefore is only suitable for continuous data. K-means variations that deal with categorical data are available but not widely used.
2. The k-means method implicitly assumes spherical probability distributions.
3. The results of the k-means method depend strongly on the initial guesses of the seeds.
4. The k-means method can be sensitive to outliers. If an outlier is picked as a starting seed, it may end up in a cluster of its own. Also if an outlier moves from one cluster to another during iterations, it can have a major impact on the clusters because the means of the two clusters are likely to change significantly.
5. Although some local optimum solutions discovered by the K-means method are satisfactory, often the local optimum is not as good as the global optimum.
6. The K-means method does not consider the size of the clusters. Some clusters may be large and some very small.

7. The K-means does not deal with overlapping clusters.

2.2.4 Time and space complexity

The space requirements for K-mean are modest because only the data points and centroids are stored. Specifically, the storage required is O((m+k)n), where m is the number of points and n is the number of attributes. The time requirements for K-means are also modest-basically linear in the number of data points. In particular, the time required is O(I*k*m*n), where I is the number of iterations required for convergence, as mentioned I is often small and can usually be safely bound as most changes typically occur in the first few iterations. Therefore K-means is linear in m, the number of points and is efficient as well as simple provided that K, the number of cluster is significantly less than m [4].

2.3 Expectation Maximization

The EM (Expectation Maximization) algorithm is a popular iterative refinement algorithm that can be used for finding the parameter estimates, it can be viewed as an extension of the K-means paradigm, which assigns an object to the cluster with which it is most similar, based on the cluster mean. Instead of assigning each object to a dedicated cluster, EM assigns each object to a cluster according to a weight representing the probability of membership. In other words, there are no strict boundaries between clusters. Therefore, new means are computed based on weighted measure [1]. The algorithm is described as follows [3].

1. Make an initial guess of the parameter vector: This involves randomly selecting K objects to represent the cluster means or centers (as in K-means partitioning) as well as making guesses for the additional parameters.
2. Iteratively refine the parameters(or clusters) based on the following two steps:

a) Expectation Step: Assign each object x_i to cluster C_k with the probability $(x_i \epsilon\, C_k) = P(C_k|x_i) = \frac{p(C_k)p(x_i|C_k)}{p(x_i)}$, where $p(x_i|C_k) = N(m_k, E_k(x_i))$ follows the normal(i.e., Gaussian) distribution around mean, m_k, with expectation, E_k. In other words, this step calculates the probability of cluster membership of objectsx_i for each of the clusters. These probabilities are the expected cluster memberships for objectsx_i.

b) Maximization Step: Use the probability estimates from above to re-estimate (or refine) the model parameters. For example, $m_k = \frac{1}{n}\sum_{i=1}^{n}\frac{x_i P(x_i \epsilon\, C_k)}{\sum_j P(x_i \epsilon\, C_j)}$ this step is the maximization of the likelihood of the distributions given the data [3].

2.3.1 Strength and Weakness of EM

The EM algorithm is simple and easy to implement. In practice, it converges fast but may not reach the global optima. Convergence is guaranteed for certain forms of optimization functions. Bayesian clustering methods focus on the computation of class-conditional probability density. They are commonly used in the statistics community. In industry, AutoClass is a popular Bayesian clustering method that uses a variant of EM algorithm. The best clustering maximizes the ability to predict the attributes of an object given the correct cluster of the object [4].

2.3.2 Time and Space Complexity

The computational complexity is linear in d (the number of input features), n (the number of objects) and t (the number of iterations) [2].

2.4 PAM (Partitioning around medoids)

Initially, a random set of k items is taken to be the set of medoids. Then at each step, all items from the input dataset that are not currently medoids are examined one by one to see if they should be medoids. That is, the algorithm determines whether there is an item that should replace one of the existing medoids. By looking at all pairs of medoid, non-medoid objects, the algorithm chooses the pair that improves the overall quality of the clustering the best and exchanges them. Quality here is measured by the sum of all distances from a non-medoid object to the medoid for the cluster it is in. An item is assigned to the cluster represented by the medoid to which it is closest (minimum distance). We assume that K_i is the cluster represented by medoid t_i. Suppose t_i is a current medoid and we wish to determine whether it should be exchanged with a non-medoid t_h. We wish to do this swap only if the overall impact to the cost (sum of the distances to cluster medoids) represents an improvement [2].

2.4.1 Algorithm of PAM

The most common realization of *k*-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follow [5]:

```
1. Initialize: randomly select k of the n data points as the medoids
2. Associate each data point to the closest medoid. ("closest" here is defined using any valid distance metric, most commonly Euclidean distance, Manhattan distance orMinkowski distance)
3. For each medoid m
    1. For each non-medoid data point o
        1. Swap m and o and compute the total cost of the configuration
4. Select the configuration with the lowest cost.
5. repeat steps 2 to 4 until there is no change in the medoid.
```

2.4.2 Strength and Weakness of PAM

More robust than k-means in the presence of noise and outliers; because a medoid is less influenced by outliers or other extreme values than a mean. It is relatively more costly and its complexity is O (i k (n-k)2),where i is the total number of iterations, is the total number of clusters, and n is the total number of objects. Relatively it is not so much efficient. It needs to specify k, the total number of clusters in advance. Result and total run time depends upon initial partition [7].

2.5 Bond Energy Algorithms

The idea is that attributes that are used together form a cluster and should be stored together. In distributed database, each resulting cluster is called a vertical fragment and may be stored at different sites from other fragments.

2.5.1 Algorithm of Bond Energy Algorithms

The basic steps of this clustering algorithm are [2]:

1. Create an attribute affinity matrix in which each entry indicates the similarity matrixes are based on the frequency of common usage of attribute pairs.
2. The BEA then converts this similarity matrix to a BOND matrix in which the entries represent a type of nearest neighbor bonding based on probability of co-access. The BES algorithm rearranges rows or columns so that similar attributes appear close together in the matrix.
3. Finally, the designer draws boxes around regions in the matrix with high similarity.

2.5.2 Strength and Weakness of Bond Energy Algorithm

BEA has gained wide recognition and remains the algorithm of choice for a number of applications. One such use arises in manufacturing. In these applications, parts or machines with similar features are grouped into families in a process referred to as cell formation.BEA is also popular for database design. The goal here is to determine sets of attributes that are accessed by distinct sets of applications, using a process referred to as vertical fragmentation.BEA has also been used for analyzing program structure in the field of software engineering. The locations of all of the components, their respective calls, and the depth of nested calls all contribute to the difficulties that can be expected during the debugging and maintenance phases of a program's life. Due to the fact that these phases are generally much more expensive than the other phases, structural improvements are valuable. BEA has been used to determine the placement of components with good results [8].

2.6 Clustering with Genetic Algorithm

2.6.1 Genetic Algorithm

Basic step of genetic algorithm is described below

```
1. t =0
2. initialize population P(t)
3. compute fitness P(t)
4. t=t+1
5. if termination criteria achieved go to step10
6. select P(t) from P(t-1)
7. crossover P(t)
8. mutate P(t)
9. go to step 3
10. output best and stop
```

2.6.2 Genetic Algorithm for clustering

Clustering using Genetic Algorithm is discussed as follows [9]:

- Euclidean distances of the points from their respective cluster centers. Mathematically, the clustering metric μ for the K clusters $C_1, C_2, \ldots, C_k$ is given by

$$\mu(C_1, C_2, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - z_i||$$

- The task of the GA is to search for the appropriate cluster centers $z_1, z_2, \ldots, z_k$ such that the clustering metric μ is minimized.
- String representation
  - Each string is a sequence of real numbers representing the K cluster centers. (Float-point representation)
  - An N-dimensional space, the length of a chromosome is N*K words.
- Population initialization
  - The K cluster centers encoded in each chromosome are initialized to K randomly chosen points from the data set. This process if repeated for each of the P chromosomes in the population, where P is the size of the population.
- Fitness computation
  - In the first phase, the clusters are formed according to the centers encoded in the chromosome under consideration.
  - Assign each point x_i, i=1,2,3,...,n to one of the clusters C_i with center z_j such that $|x_i - z_j| < ||x_i - z_p||$, p=1,2,…,K and p$\neq j$.
  - After the clustering is done, the cluster centers encoded in the chromosome are replaced by the mean points of the respective clusters. For cluster C_i, the new center z_i is computed as

$$z_i^* = \frac{1}{n_i} \sum_{x_j \in C_i} x_j, i = 1,2, \ldots, k$$

  - Subsequently, the clustering metric μ is computed as follows:

$$\mu = \sum_{i=1}^{K} \mu_i, \mu_i = \sum_{x_j \in C_i} ||x_j - z_i||$$

  - The fitness function is defined as $f = \frac{1}{\mu}$.

2.7 Clustering with Neural Network (NN)

A self-organizing feature map or self-organizing map is an NN approach that uses competitive unsupervised learning [17].

- The Self-Organizing Map was developed by professor Kohonen. The SOM has been proven useful in many applications.
- One of the most popular neural network models. It belongs to the category of competitive learning networks.
- Based on unsupervised learning, which means that no human intervention is needed during the learning and that little need to be known about the characteristics of the input data.
- Use the SOM for clustering data without knowing the class memberships of the input data. The SOM can be used to detect features inherent to the problem and thus has also been called SOFM, the Self-Organizing Feature Map.
- Provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimensional space onto a plane.
- The property of topology preserving means that the mapping preserves the relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data. Also, the SOM has the capability to generalize
- Generalization capability means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to.

2.7.1 Input and Output

- Training data: vectors X
  - Vectors of length *n*
    $(x_{1,1}, x_{1,2}, ..., x_{1,i},\dots, x_{1,n})$
    $(x_{2,1}, x_{2,2}, ..., x_{2,i},\dots, x_{2,n})$
    …
    $(x_{j,1}, x_{j,2}, ..., x_{j,i},\dots, x_{j,n})$
    …
    $(x_{p,1}, x_{p,2}, ..., x_{p,i},\dots, x_{p,n})$
  - Vector components are real numbers
- Outputs
  - A vector, Y, of length *m*: $(y_1, y_2, ..., y_i,\dots, y_m)$
  - Sometimes m < n, sometimes m > n, sometimes m = n
  - Each of the *p* vectors in the training data is classified as falling inone of *m* clusters or categories
  - That is: Which category does the training vector fall into?
- Generalization
  - For a new vector: $(x_{j,1}, x_{j,2}, ..., x_{j,i},\dots, x_{j,n})$
  - Which of the *m* categories (clusters) does it fall into?

2.7.2 Network Architecture

- Two layers of units
  - Input: n units (length of training vectors)
  - Output: m units (number of categories)

- Input units fully connected with weights to output units
- Intra-layer ("lateral") connections
  - Within output layer
  - Defined according to some topology
  - No weight between these connections, but used in algorithm for updating weights

2.7.3 Overall Algorithm of SOM

- Training
  - Select output layer topology
  - Train weights connecting inputs to outputs
  - Topology is used, in conjunction with current mapping of inputs to outputs, to define which weights will be updated
  - Distance measure using the topology is reduced overtime; reduces the number of weights that get updated per iteration
  - Learning rate is reduced over time
- Testing
  - Use weights from training

2.7.4 SOM algorithm

- Select output layer network topology
  1. Initialize current neighborhood distance, D(0), to a positive values
- Initialize weights from inputs to small random values
- Let t = 1
- While computational bounds are not exceeded do
  1. Select an input samplei_l
  2. Compute the square of the Euclidean distance of i_l from weight vectors(w_j) associating with each output node $\sum_{k=1}^{n}(i_{l,k} - w_{j,k}(t))^2$
  3. Select output node j^*that has weight vector with minimum value from step 2
  4. Update weights to all nodes within a topological distance given by D(t) from j^*, using the weight update rule:
     $w_j(t+1) = w_j(t) + \eta(t)(i_l - w_j(t))$
  5. Increment t
- End while Learning rate generally decreases with time:$0 < \eta(t) \le \eta(t-1) \le 1$

2.8 K-nearest neighbor Algorithm

Intuitively, the nearest neighbor chain algorithm repeatedly follows a chain of clusters $A \rightarrow B \rightarrow C \rightarrow$ where each cluster is the nearest neighbor of the previous one, until reaching a pair of clusters that are mutual nearest neighbors.

2.8.1 Algorithm of K-nearest neighbor

More formally, the algorithm performs the following steps [12]:

- Initialize the set of active clusters to consist of *n* one-point clusters, one for each input point.
- Let *S* be a stack data structure, initially empty, the elements of which will be active clusters.
- While there is more than one cluster in the set of clusters:
    - If *S* is empty, choose an active cluster arbitrarily and push it onto *S*.
    - Let *C* be the active cluster on the top of *S*. Compute the distances from *C* to all other clusters, and let *D* be the nearest other cluster.
    - If *D* is already in *S*, it must be the immediate predecessor of *C*. Pop both clusters from *S*, merge them, and push the merged cluster onto *S*.
    - Otherwise, if *D* is not already in *S*, push it onto *S*.

If there may be multiple equal nearest neighbors to a cluster, the algorithm requires a consistent tie-breaking rule: for instance, in this case, the nearest neighbor may be chosen, among the clusters at equal minimum distance from *C*, by numbering the clusters arbitrarily and choosing the one with the smallest index [12].

2.8.2 Strength and Weakness of K-nearest Neighbor Algorithm

- Strengths
    - Simple to implement and use
    - Comprehensible–easy to explain prediction
    - Robust to noisy data by averaging k-nearest neighbors. Some appealing applications (will discuss next in personalization)
- Weaknesses:
- Need a lot of space to store all examples.
- Takes more time to classify a new example than with a model (need to calculate and compare distance from new example to all other examples) [21].

2.8.3 Time and Space complexity

Each iteration of the loop performs a single search for the nearest neighbor of a cluster, and either adds one cluster to the stack or removes two clusters from it. Every cluster is only ever added once to the stack, because when it is removed again it is immediately made inactive and merged. There are a total of $2n - 2$ clusters that ever get added to the stack: *n* single-point clusters in the initial set, and $n - 2$ internal nodes other than the root in the binary tree representing the clustering. Therefore, the algorithm performs $2n - 2$ pushing iterations and $n - 1$ popping iterations, each time scanning as many as $n - 1$ inter-cluster distances to find the nearest neighbor. The total number of distance calculations it makes is therefore less than $3n^2$, and the total time it uses outside of the distance calculations is $O(n^2)$.

Since the only data structure is the set of active clusters and the stack containing a subset of the active clusters, the space required is linear in the number of input points [12].

2.9 Agglomerative methods

The basic idea of the agglomerative method is to start out with n clusters for n data points, that is, each cluster consisting of a single data point. Using a measure of distance, at each step of the method, the method merges two nearest clusters, thus reducing the number of clusters and building successively larger clusters. The process continues until the required number of clusters has been obtained or all the data points are in one cluster. The agglomerative method leads to hierarchical clusters in which at each step we build larger and larger clusters that include increasingly dissimilar objects.

2.9.1 Algorithm of Agglomerative method

The agglomerative method is basically a bottom-up approach which involves the following steps. An implementation however may include some variation of these steps [1].

1. Allocate each point to a cluster of its own. Thus we start with n clusters for n objects.
2. Create a distance matrix by computing distances between all pairs of clusters either using, for example, the single-link metric or the complete-link metric. Some other metric may also be used. Sort these distances in ascending order.
3. Find the two clusters that have the smallest distance between them
4. Remove the pair of objects and merge them.
5. If there is only one cluster left then stop.
6. Compute all distances from the new cluster and update the distance matrix after the merger and go to Step 3.

2.9.2 Strength and Weakness of Agglomerative method

More generally, such algorithms are typically used because the underlying application, e.g., creation of taxonomy, requires a hierarchy. Also, there have been some studies that suggest that these algorithms can produce better-quality clusters. However, agglomerative hierarchical clustering algorithms are expensive in terms of their computational and storage requirements. The fact that all merges are final can also cause trouble for noisy, high-dimensional data, such as document data. In turn, these two problems can be addressed to some degree by first partially clustering the data using another technique, such as K-means [4].

2.9.3 Time and Space Complexity

The basic agglomerative hierarchical clustering algorithm just presented uses a proximity matrix. This requires the storage of $\frac{1}{2}m^2$ proximities (assuming the proximity matrix is symmetric) where m is the number of data points. The space needed to keep track of the clusters is proportional to the number of clusters, which is $m-1$, excluding single to n clusters. Hence, the total space (complexity is $O(m^2)$). The analysis of the basic agglomerative hierarchical clustering algorithm (is also straight forward with respect to computational complexity. $O(m^2)$ time is required to compute the proximity matrix. After that step, there are $m-1$ iterations involving steps 3 and 4 because there are m clusters at the start and two clusters are merged during each iteration. If performed as a linear search of the proximity matrix, then for the i^{th} iteration, step 3 requires $O(m-i+1)$ time, which is proportional to the current number of clusters squared. Step 4 only requires $O(m-i+1)$ time to update the proximity matrix after the merger of two clusters. A cluster merge r affects only$O(m-i+1)$proximities for the techniques that we consider. Without modification, this would yield (a time complexity of $O(m^3)$). If the distances from each cluster to all other clusters are stored as a sorted list (or heap), it is possible to reduce the cost of finding the two closest clusters to $O(m-i+1)$. However, because of the additional complexity of keeping data in a sorted list or heap, the overall time required for a hierarchical clustering is $O(m^2 \log m)$. The space and time complexity of hierarchical clustering severely limits the size of data sets that can be processed. We discuss scalability approaches for clustering algorithms, including hierarchical clustering techniques [13].

2.10 Divisive Hierarchical Method

The divisive method is the opposite of the agglomerative method in that the method starts with the whole data set as one cluster and then proceeds to recursively divide the cluster into two sub-clusters and continues until each cluster has only one object or some other stopping criterion has been reached. There are two types of divisive methods [1]:

Monothetic: It splits a cluster using only one attribute at a time. An attribute that has the most variation could be selected.

Polythetic: It splits a cluster using all of the attributes together. Two clusters far apart could be built based on distance between objects.
A typical polythetic divisive method works like the following [1]:

1. Decide on a method of measuring the distance between two objects. Also decide a threshold distance.
2. Create a distance matrix by computing distances between all pairs of object within the cluster. Sort these distances in ascending order.
3. Find the two objects that have the largest distance between them. They are the most dissimilar objects.
4. If the distance between the two objects is smaller than the pre-specified threshold and there is no other cluster that needs to be divided then stop, otherwise continue.
5. Use the pair of objects as seeds of a K-means method to create two new clusters.
6. If there is only one object in each cluster then stop otherwise continue with step 2.

In this the above method, we need to resolve the following two issues:

- Which cluster to split next?
- How to split a cluster?

2.10.1 Selection of cluster to split next

There are a number of possibilities when selecting the next cluster to split:

1. Split the clusters in some sequential order.
2. Split the cluster that has the largest number of objects.
3. Split the cluster that has the largest variation within it.

The first two approaches are clearly very simple but the third approach is better since it is based on the sound criterion of splitting a cluster that has the most variance.

2.10.2 Method to split a cluster

In the method presented above, we used a simple approach for splitting a cluster based on the distances between the objects in the cluster. A distance matrix is created and the two most dissimilar objects are selected as seeds of two new clusters. The K-means method is then used to split the cluster [1].

2.11 BIRCH clustering algorithm

To understand the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) clustering algorithm, the following background knowledge is needed [14].

- BIRCH is local in that each clustering decision is made without scanning all data points.
- BIRCH exploits the observation that the data space is usually not uniformly occupied, and hence not every data point is equally important for clustering purposes.
- BIRCH makes full use of available memory to derive the finest possible subclusters (to ensure accuracy) while minimizing I/O costs (to ensure efficiency).
- Given N d-dimensional data points in a cluster:$\{\vec{x}_i\}$ where i = 1,2,…,N, the centroid, radius and diameter is defined as:

$$\vec{x}_0 = \frac{\sum_{i=1}^{N} \vec{x}_i}{N}$$

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{x}_i - \vec{x}_0)^2}{N}\right)^{\frac{1}{2}}$$

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

- We treat $\vec{x}_0$ R and D as properties of a single cluster

- Next between two clusters, we define 5 alternative distances for measuring their closeness.
  - Given the centroids of two clusters: ,the two alternative distances D0 and D1 are defined as:

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

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

- Given N1 d-dimensional data points in a cluster: $\{\vec{x}_i\}$ where i = 1,2,…N1, and N2 data points in another cluster: $\{\vec{x}_j\}$ where j = N1+1,N1+2,…,N1+N2, the average inter-cluster distance D2, average intra-cluster distance D3 and variance increase distance D4 of the two clusters are defined as:

$$D2 = \left(\frac{\sum_{i=1}^{N1}\sum_{j=N1+1}^{N1+N2} (\vec{xi} - \vec{xj})^2}{N1N2}\right)^{\frac{1}{2}}$$

$$D3 = \left(\frac{\sum_{i=1}^{N1+N2}\sum_{j=1}^{N1+N2} (\vec{xi} - \vec{xj})^2}{(N1+N2)(N1+N1-1)}\right)^{\frac{1}{2}}$$

$$D4 = \sum_{k=1}^{N1+N2} \left(\vec{xk} - \frac{\sum_{l=1}^{N1+N2} \vec{xl}}{N1+N2}\right)^2 - \sum_{i=1}^{N1} \left(\vec{xi} - \frac{\sum_{l=1}^{N1} \vec{xl}}{N1}\right)^2 - \sum_{j=N1+1}^{N1+N2} \left(\vec{xj} - \frac{\sum_{l=N1+1}^{N1+N2} \vec{xl}}{N2}\right)^2$$

- Actually, D3 is D of the merged cluster.
- D0, D1, D2, D3, D4 is some measurement of two clusters. We can use them to determine if two clusters are close.

Clustering Feature:

- A Clustering Feature is a triple summarizing the information that we maintain about a cluster.
- Definition. Given N d-dimensional data points in a cluster:$\{\vec{x}_i\}$ where i = 1,2,…N, the Clustering Feature (CF) vector of the cluster is defined as a triple:$CF = (N, \vec{LS}, SS)$ where N is the number of data points in the cluster, $\sum_{i=1}^{N} \vec{x}_i$ is the linear sum of the N data points, i.e., $\vec{LS}$ and SS is the square sum of the N data points, i.e., $\sum_{i=1}^{N} \vec{x}_i^{\,2}$
- CF Additive Theorem
  - Assume that CF1=(N1, $\vec{LS}$,SS1), and CF2 =(N2, $\vec{LS}$,SS2) are the CF vectors of two disjoint clusters. Then the CF vector of the cluster that is formed by merging the two disjoint clusters, is:

$$CF_1 + CF_2 = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

  - From the CF definition and additive theorem, we know that the corresponding X0, R, D, D0, D1, D2, D3 and D4 can all be calculated easily.

CF Tree

- A CF tree is a height-balanced tree with two parameters:
    - Branching factor B. Each non leaf node contains at most B entries of the form[cfi,childi], where i = 1,2,…,B, childi is a pointer to its i-th child node, and CFi is the CF of the sub-cluster represented by this child. A leaf node contains at most L entries, each of the form [CFi], where i = 1, 2, …L.
    - Initial threshold T. Which is an initial threshold for the radius (or diameter) of any cluster. The value of T is an upper bound on the radius of any cluster and controls the number of clusters that the algorithm discovers.

- CF Tree will be built dynamically as new data objects are inserted. It is used to guide a new insertion into the correct subcluster for clustering purpose. Actually, all clusters are formed as each data point is inserted into the CF Tree.
- CF Tree is a very compact representation of the dataset because each entry in a leaf node is not a single data point but a subcluster.

Insertion into a CF Tree

We now present the algorithm for inserting an entry into a CF tree. Given entry "Ent", it proceeds as below:

1. **Identifying the appropriate leaf:** Starting from the root, according to a chosen distance metric D0 to D4 as defined before, it recursively descends the CF tree by choosing the closest child node.
2. **Modifying the leaf:** When it reaches a leaf node, it finds the closest leaf entry, and tests whether the node can absorb it without violating the threshold condition.
    - If so, the CF vector for the node is updated to reflect this.
    - If not, a new entry for it is added to the leaf.
        - If there is space on the leaf for this new entry, we are done.
        - Otherwise we must split the leaf node. Node splitting is done by choosing the farthest pair of entries based on the closest criteria.
3. **Modifying the path to the leaf:** After inserting "Ent" into a leaf, we must update the CF information for each nonleaf entry on the path to the leaf.
    - In the absence of a split, this simply involves adding CF vectors to reflect the additions of "Ent".
    - A leaf split requires us to insert a new nonleaf entry into the parent node, to describe the newly created leaf.
        - If the parent has space for this entry, at all higher levels, we only need to update the CF vectors to reflect the addition of "Ent".
        - Otherwise, we may have to split the parent as well, and so on up to the root.
4. **Merging Refinement:** In the presence of skewed data input order, split can affect the clustering quality, and also reduce space utilization. A simple additional merging often helps ameliorate these problems: suppose the propagation of one split stops at some nonleaf nodeN_j, i.e., N_j can accommodate the additional entry resulting from the split.
    a. Scan N_j to find the two closest entries.
    b. If they are not the pair corresponding to the split, merge them.
    c. If there are more entries than one page can hold, split it again.
    d. During the resplitting, one of the seeds attracts enough merged entries, the other receives the rest entries.

In summary, it improve the distribution of entries in the closest two children, even free a node space for later use.

2.10.1 Algorithm for BIRCH

- Phase1: Load into memory by building a CF tree
- Phase 2: Condense into desirable range by building a smaller CF tree
    - To meet the need of the input size range of Phase 3
- Phase 3: Global clustering
    - Approach: re-cluster all subclusters by using the existing global or semi-global clustering algorithms
- Phase 4: Global clustering
    - Approach: use the centroids of the cluster produced by phase 3 as seeds, and redistributes the data points to its closest seed to obtain a set of new clusters. This can use existing algorithm [14].

2.10.2 Strength and Weakness of BIRCH

- Scales linearly: finds a good clustering with a single scan and improves the quality with a few additional scans.
- Computation complexity of the algorithm is O(n), where n is number-objects.
- Handles only numeric data, and sensitive to the order of the data record [15].

2.10.3 Time and Space complexity of BIRCH

- Time Complexity Analysis for Phase 1
    - for inserting all data : O(dimension × the number of data point × the number of entries per node × the number of nodes by following a path from the root to leaf)
    - for reinserting leaf entries : O(the number of rebuilding × dimension × the number of leaf entries to reinsert × the number of entries per node ×the number of nodes by following a path from the root to leaf)
- Time Complexity Analysis for other Phase
    - The phase 2 is similar to phase 1
    - The phase 3 & 4 rely on the adopted methods.

2.11. Robust Clustering Algorithm for Categorical Attributes (ROCK) clustering algorithm

2.11.1 Short comings of Traditional methods [16]

- Traditional clustering methods can be classified into:
    - Partitional clustering which divides the point space into k clusters that optimize a certain criterion function, such as $E = \sum_{i=1}^{k} \sum_{\vec{x} \in C_i} d(\vec{x}, \overrightarrow{m_i})$
    - Hierarchical clustering which merges two clusters at each step before reaching a user defined condition.
- Problems of using partitional clustering
    - Large number of items in database while few items in each transaction
    - A pair of transactions in a cluster may only have fewer items in common
    - The size of transactions is not the same.
    - Result: partitional clustering tends to split clusters to minimize the criterion function.
- Problems of using hierarchical clustering
    - Ripple effect as the cluster size grows. The number of attributes appearing in the centroid goes up while their value in the centroid decreases.
    - This makes it very difficult to distinguish the difference between two points differ on few attributes or two points differ on every attribute by small amounts.

Introduction of LINK

- The main problem of traditional hierarchical clustering is local properties involving only the two points are considered.
- Neighbor
    - If the two points are similar enough with each other, they are neighbor
- Link

- The link for pair of points is: the number of common neighbors.

Obviously, link incorporates global information about the other points in the neighborhood of the two points. The larger the link, the larger probability that this pair of point are in the same clusters [16].

- Criterion Function

$$E_l = \sum_{i=1}^{k} \eta_i \times \sum_{p_q p_r \in C_i} \frac{link(p_q, p_r)}{\eta_i^{1+2\int(\theta)}}$$

- Goodness Function

$$g(C_i, C_j) = \frac{Link(C_i, C_j)}{(\eta_i + \eta_j)^{1+2\int(\theta)} - \eta_i^{1+2\int(\theta)} - \eta_j^{1+2\int(\theta)}}$$

Suppose in C_i each point has roughly $\eta_i^{\int(\theta)}$ neighbors.

The author's choice for basket data is: $\int(\theta) = \frac{1-\theta}{1+\theta}$

2.12 Summary of Hierarchical Methods

Agglomerative clustering finds distance between all clusters (starting from each object being in a cluster of its own) and merges the closet clusters until there is just one cluster or some other stopping criterion is met. Since the method requires distances between each object, a distance matrix $n \times n$ must be stored making it difficult for very large database. The size of the matrix can be reduced by committing some entities. For example, large distances are of little interest in the agglomerative method while the opposite is true for the divisive method. It should be noted that in the agglomerative method, any distance metric may be used. It need not be used on Euclidean distance [1].

2.12.1 Advantages of the hierarchical approach [1]

1. The threshold approach can provide more insight the data by showing a hierarchy of clusters than a flat structure created by a partitioning method like the K-means method.
2. Hierarchical methods are conceptually simpler and can be implemented easily.
3. In some applications only proximity data is available and then the hierarchical approach may be better.
4. Hierarchical methods can provide clusters at different levels of granularity.

2.12.2 Disadvantages of the hierarchical approach [1]

1. The hierarchical methods do not include a mechanism by which objects that have been incorrectly put in a cluster may be reassigned to another cluster.
2. The time complexity of hierarchical methods can be shown to be $O(m^3)$.
3. The distance matrix requires $O(m^2)$ space and becomes very large for a large number of objects.
4. Different distance metrics and scaling of data can significantly change the results.

Hierarchical methods are quite different from partition-based clustering. In hierarchical methods there is no need to specify the number of clusters and the cluster seeds. The agglomerative approach essentially starts with each object in an individual cluster and then merges cluster to build larger and larger clusters. The divisive approach is the opposite of that. There is no metric to judge the quality of the results.

Different distance functions can lead to different shapes of hierarchical clusters, but hierarchical clusters provide a useful display of cluster relationships.

2.13 CURE (Clustering using representation)

CURE handles limited memory by obtaining a random sample to find the initial clusters. The random sample is partitioned, and each partition is then partially clustered. These resulting clusters are then completely clustered in a second pass. The sampling and partitioning are done solely to ensure that the data (regardless of database size) can fit into available main memory. When the clustering of the sample is complete, the labeling of data on disk is performed. A data item is assigned to the cluster with the closet representative points. The basic steps of CURE for large database are:

1. Obtain a sample of the database
2. Partition the sample into p partitions of size $\frac{n}{p}$. This is done to speed up the algorithm

because clustering is first performed on each partition.

3. Partially cluster the points in each partition using the hierarchical algorithm. This provides a first guess at what the clusters should be. The number of clusters is $\frac{n}{pq}$ for some constant q.
4. Remove outliers. Outliers are eliminated by the use of two different techniques. The first technique eliminates clusters that grow very slowly. When the number of clusters is below a threshold, those clusters with only one or two items are deleted. It is possible that close outliers are part of the sample and would not be identified by the first outlier elimination technique. The second technique removes very small clusters toward the end of the clustering phase.
5. Completely cluster all data in the sample. Here, to ensure processing in main memory, the input includes only the cluster representative point to it. These sets of representative points are small enough to fit into main memory, so each of the n points must be compared to ck representative points.

2.13.1 Complexity of CURE Algorithm

The time complexity of CURE is O($n^2 \log n$), while space is $O(n)$. This is worst-case behavior. The improvements proposed for main memory processing certainly improve on this time complexity because the entire clustering required. When clustering is performed on the complete database, a time complexity of only $O(n)$ is required. A heap and K-D tree data structure are used to ensure this performance [2].

2.14 Chameleon (A Hierarchical Clustering Algorithm)

Chameleon uses a k-nearest neighbor graph approach to construct a sparse graph, where each vertex of the graph represents a data object, and there exists an edge between two vertices (object) if one object is among the k-most-similar objects of the other. The edges are weighted to reflect the similarity between objects. Chameleon uses a graph partitioning algorithm to partition the k-neighbor graph into a large number of relatively small subclusters. It then uses an agglomerative hierarchical clustering algorithm that repeatedly merges subclusters based on their similarity. To determine the pairs of most similar subclusters, it takes into account both the interconnectivity as well as the closeness of the clusters. Algorithm of Chameleon is described as follows.

- Two-phase approach
  - Phase –I: Uses a graph partitioning algorithm to divide the data set into a set of individual clusters.
  - Phase –II: uses an agglomerative hierarchical mining algorithm to merge the clusters. Using Dynamic Modeling):
- Chameleon(Phase-II) takes into account
  - Inter Connectivity
  - Relative closeness
  - Hence, chameleon takes into account features intrinsic to a cluster.

Constructing a sparse graph

- Data points that are far away are completely avoided by the algorithm (reducing the noise in the dataset)
- Captures the concept of neighborhood dynamically by taking into account the density of the region.
- Partition the KNN graph such that the edge cut is minimized.
  - Reason: Since edge cut represents similarity between the points, less edge cut => less similarity.
- Multi-level graph partitioning algorithms to partition the graph [18]

The graph-partitioning algorithm partitions the k-nearest-neighbor graph such that it minimizes the edge cut. That is, a cluster C is partitioned into sub clustersC_i and C_j so as to minimize the weight of the edges that would be cut should C be bisected into C_i and C_j. Edge cut is denoted EC (C_i, C_j) and assesses the absolute interconnectivity between clusters $C_i$and C_j[3].

Cluster Similarity

Chameleon determines the similarity between each pair of clusters C_i and C_j according to their relative interconnectivity, RI (C_i,C_j), and their relative closeness, RC (C_i, C_j):

- The relative inter-connectivity, RI (C_i, C_j), between two clusters, C_i and C_j is defined as the absolute interconnectivity between C_i and C_j, normalized with respect to the internal interconnectivity of the two clusters, C_i and C_j. That is,
- $$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}$$

Where $EC_{\{C_i, C_j\}}$ is the edge cut, defined as above, for cluster containing both C_i and C_j. Similarly EC_{C_i} $(or\ EC_{C_j})$ is the minimum sum of the cut edges that partition $C_i(or\ C_j)$ into two roughly equal parts.

The relative closeness, RC (C_i, C_j), between a pair of cluster, C_i and C_j is the absolute closeness between C_i and C_j, normalized with respect to the internal closeness of the two clusters, C_i and C_j. It is defined as

$$RC(C_i, C_j) = \frac{\overline{S_{EC_{\{C_i, C_j\}}}}}{\frac{|C_i|}{|C_i| + |C_j|}\overline{S_{EC_{C_i}}} + \frac{|C_j|}{|C_i| + |C_j|}\overline{S_{EC_{C_j}}}}$$

Where $\overline{S_{EC_{\{C_i, C_j\}}}}$ is the average weight of the edges that connect vertices in C_i to vertex in C_j and$\overline{S_{EC_{C_i}}}$ (or $\overline{S_{EC_{C_i}}}$) is the average weight of the edges that belong to the cut bisector of cluster C_i(or C_j).

2.14.1 Time and Space Complexity of Chameleon

Chameleon has been shown to have greater power at discovering arbitrarily shaped clusters of high quality than several well-known algorithms such as BIRCH and density based DBSCAN. However, the processing cost for high-dimensional data may require$O(n^2)$ time for n objects in the worst case [3].

2.15 DBSCAN clustering algorithm

DBSCAN (density based spatial clustering of applications with noise**)** is an example of a density based method for clustering. The method was designed for spatial databases but can be used in other applications. It requires two input parameters: the size of the neighborhood(R) and the minimum points in the neighborhood (N). Essentially these two parameters determine the density within points in the clusters the user is willing to accept since they specify how many points must be in a region. The number of points not only determines the density of acceptable clusters but it also determines which objects will be labeled outliers or noise. Objects are declared to be outliers if there are few other objects in their neighborhood. The size parameter R determines the size of the clusters found. If R is big enough, there would be one big cluster and no outliers. If R is small, there will be small dense clusters and there might be many outliers [1].

We know define a number of concepts that are required in the DBSCAN method:

1. Neighborhood: The neighborhood of an object y is defined as all the objects that are within the radius R from y.
2. Core object: An object y is called a core object if there are N objects within its neighborhood.
3. Proximity: Two objects are defined to be in proximity to each other if they belong to the same cluster. Object x_1 is in proximity to object x_2 if two conditions are satisfied:
(a) The objects are close enough to each other, i.e. within a distance of R.
(b) x_2 is a core object as defined above.
4. Connectivity: two objects x_1 and x_n are connected if there is a path or chain of objects $x_1, x_2, \dots, x_n$ such that each x_{i+1} is in proximity to object x_i.

We now outline the basic algorithm for density-based clustering

1. Select values of R and N
2. Arbitrarily select an object p.
3. Retrieve all objects that are connected to p, given R and N.
4. If p is a core object, a cluster is formed
5. If p is a border object, no objects are in its proximity. Choose another object. Go to step 3
6. Continue the process until all of the objects have been processed.

2.15.1 Strength and Weakness of DBSCAN

Because DBSCAN uses a density-based definition of a cluster, it is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes. Thus, DBSCAN can find many clusters that could not be found using K-means. As indicated previously, however, DBSCAN has trouble when the clusters have widely varying densities. It also has trouble with high-dimensional data because density is more difficult to define for such data. Finally, DBSCAN can be expensive when the computation of nearest neighbors requires computing all pairwise proximities, as is usually the case for high-dimensional data [7].

2.15.2 Time and Space Complexity of DBSCAN

The basic time complexity of the DBSCAN algorithm is O (m × time to find points in the Eps - neighborhood), where m is the number of points. In the worst case, this complexity is O(m^2). However, in low-dimensional spaces, there are data structures, such as kd-trees, that allow efficient retrieval of all points within a given distance of a specified point, and the time complexity can be as low as$O(m \log m)$. The space requirement of DBSCAN, even forhigh-dimensional data, is$O(m)$because it is only necessary to keep a small amount of data for each point, i.e., the cluster label and the identification of each point as a core, border, or noise point.

2.16 OPTICS (Ordering Points to Identity the clustering Structure)

Ordering points to identify the clustering structure (OPTICS) is an algorithm for finding density-based clusters in spatial data. It was presented by Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel and Jörg Sander. Its basic idea is similar to DBSCAN but it addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density. In order to do so, the points of the database are (linearly) ordered such that points which are spatially closest become neighbors in the ordering. Additionally, a special distance is stored for each point that represents the density that needs to be accepted for a cluster in order to have both points belong to the same cluster. This is represented as a dendrogram.

2.16.1 The basic idea of OPTICS

Like DBSCAN, OPTICS requires two parameters: ε, which describes the maximum distance (radius) to consider, and$MinPts$, describing the number of points required to form a cluster. A point p is a core point if at least $MinPts$points are found within itsε - neighborhood $N_\varepsilon(p)$. Contrary to DBSCAN, OPTICS also considers points that are part of a more densely packed cluster, so each point is assigned a core distance that describes the distance to the $MinPts$ th closest point:

$$core - distance_{\varepsilon_1, MinPits}(p) = \begin{cases} UNDEFINED\ if\ |N_\varepsilon(p)| < MinPits \\ distance\ to\ the\ MinPits - th\ closet\ point\ otherwise \end{cases}$$

The reachability-distance of a point p from another point o is the distance between p and o, or the core distance of:

$$reachability - distance_{\varepsilon_1, MinPits}(p, o) = \begin{cases} UNDEFINED\ if\ |N_\varepsilon(p)| < MinPits \\ \max\ (reachability - distance_{\varepsilon_1, MinPits}(p, o), distantce(p, o))\ otherwise) \end{cases}$$

If o and p are nearest neighbors, this is the$\varepsilon' < \varepsilon$ we need to assume in order to have o and p belong to the same cluster.

Both the core-distance and the reachability-distance are undefined if no sufficiently dense cluster (w.r.t.ε) is available. Given a sufficiently large ε, this will never happen, but then every ε-neighborhood query will return the entire database, resulting in run time $O(n^2)$. Hence, the ε parameter is required to cut off the density of clusters that is no longer considered to be interesting and to speed up the algorithm this way.

The parameter ε is strictly speaking not necessary. It can be set to a maximum value. When a spatial index is available, it does however play a practical role when it comes to complexity. It is often claimed that OPTICS abstracts from DBSCAN by removing this parameter, at least to the amount of only having to give a maximum value. The parameter is strictly speaking not necessary. It can be set to a maximum value. When a spatial index is available, it does however play a practical role when it comes to complexity. It is often claimed that OPTICS abstracts from DBSCAN by removing this parameter, at least to the amount of only having to give a maximum value.

2.16.2 Complexity of OPTICS

Like DBSCAN, OPTICS processes each point once, and performs one -neighborhood query during this processing. Given a spatial index that grants a neighborhood query in$O(\log m)$ runtime, an overall runtime of $O(m\log m)$ is obtained. The authors of the original OPTICS paper report an actual constant slowdown factor of 1.6 compared to DBSCAN. Note that the value of might heavily influence the cost of the algorithm, since a value too large might raise the cost of a neighborhood query to linear complexity. In particular, choosing $\varepsilon < \max d(x,y)$(larger than the maximum distance in the data set) is possible, but will obviously lead to quadratic complexity, since every neighborhood query will return the full data set. Even when no spatial index is available, this comes at additional cost in managing the heap. Therefore, ε should be chosen appropriately for the data set.

2.17 DENCLUE (Clustering Based on Density Distribution Functions)

DENCLUE (DENsity-based CLUstEring) is a clustering method based on a set of density distribution functions. The method is built on the following ideas: (1) the influence of each data point can be formally modeled using a mathematical function, called an *influence function*, which describes the impact of a data point within its neighborhood; (2) the overall density of the data space can be modeled analytically as the sum of the influence function applied to all data points; and (3) clusters can then be determined mathematically by identifying *density attractors*, where density attractors are local maxima of the overall density function. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be objects or points inF^d, a *d*-dimensional input space. The influence function of data object $\boldsymbol{y}$ on $\boldsymbol{x}$ is a function,$F_B^y : F^d \to F_0^+$, which is defined in terms of a basic influence function F_B:

$$F_B^y(x) = F_B(x,y)$$

This reflects the impact of $\boldsymbol{y}$ on $\boldsymbol{x}$. In principle, the influence function can be an arbitrary function that can be determined by the distance between two objects in a neighborhood. The distance function, $d(\boldsymbol{x}, \boldsymbol{y})$, should be reflexive and symmetric, such as the Euclidean distance function. It can be used to compute a *square wave influence function,*

$$f_{square}(x,y) = \begin{cases} 0 \; if \; d(x,y) > \sigma \\ 1 \; otherwise \end{cases}$$

Or a Gaussian influence function

$$f_{Gauss}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

2.17.1 Advantage of DENCLUE

The major advantages of DENCLUE has are

- It has a solid mathematical foundation and generalizes various clustering methods, including partitioning, hierarchical, and density-based methods;
- It has good clustering properties for data sets with large amounts of noise;
- It allows a compact mathematical description of arbitrarily shaped clusters in high dimensional data sets;
- It uses grid cells, yet only keeps information about grid cells that actually contain data points. It manages these cells in a tree-based access structure, and thus is significantly faster than some influential algorithms, such as DBSCAN.

However, the method requires careful selection of the density parameter and noise threshold, as the selection of such parameters may significantly influence the quality of the clustering results [19].

2.18 STING (Statistical Information Grid)

STING is a grid-based multi resolution clustering technique in which the spatial area is divided into rectangular cells. There are usually several levels of such rectangular cells corresponding to different levels of resolution, and these cells form a hierarchical structure: each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information regarding the attributes in each grid cell (such as the mean, maximum, and minimum values) is precomputed and stored.

2.18.2 The STING clustering method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and store beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
    - Count, mean, s, min, max
    - Type of distribution-normal, uniform, etc
- Use a top-down approach to answer spatial data queries.
- Start from a pre-selected layer-typically with a small number of cells
- For each cell in the current level compute the confidence interval

2.18.3 Advantage of STING

STING offers several advantages: (1) the grid-based computation is query-independent, because the statistical information stored in each cell represents the summary information of the data in the grid cell, independent of the query; (2) the grid structure facilitates parallel processing and incremental updating; and (3) the method's efficiency is a major advantage: STING goes through the database once to compute the statistical parameters of the cells, and hence the time complexity of generating clusters is O(n), where n is the total number of objects. After generating the hierarchical structure, the query processing time is O(g), where g is the total number of grid cells at the lowest level, which is usually much smaller than n.

2.19 Wave Cluster (Clustering Using Wavelet Transformation)

Wave Cluster is a multi resolution clustering algorithm that first summarizes the data by imposing a multidimensional grid structure onto the data space. It then uses a wavelet transformation to transform the original feature space, finding dense regions in the transformed space. In this approach, each grid cell summarizes the information of a group of points that map into the cell. This summary information typically fits into main memory for use by the multi resolution wavelet transform and the subsequent cluster analysis. A wavelet transform is a signal processing technique that decomposes a signal into different frequency sub bands. The wavelet model can be applied to d-dimensional signals by applying a one-dimensional wavelet transform d times. In applying a wavelet transform, data are transformed so as to preserve the relative distance between objects at different levels of resolution. This allows the natural clusters in the data to become more distinguishable. Clusters can then be identified by searching for dense regions in the new domain [3].

2.19.1 Advantage of Wavelet clustering

- It provides unsupervised clustering. It uses hat-shaped filters that emphasize regions where the points cluster, while suppressing weaker information outside of the cluster boundaries. Thus, dense regions in the original feature space act as attractors for nearby points and as inhibitors for points that are further away. This means that the clusters in the data automatically stand out and "clear" the regions around them. Thus, another advantage is that wavelet transformation can automatically result in the removal of outliers.
- The multiresolution property of wavelet transformations can help detect clusters at varying levels of accuracy.

Wave Cluster is a grid-based and density-based algorithm. It conforms with many of the requirements of a good clustering algorithm: It handles large data sets efficiently, discovers clusters with arbitrary shape, successfully handles outliers, is insensitive to the order of input, and does not require the specification of input parameters such as the number of clusters or a neighborhood radius. In experimental studies, Wave Cluster was found to outperform BIRCH, CLARANS, and DBSCAN in terms of both efficiency and clustering quality. The study also found Wave Cluster capable of handling data with up to 20 dimensions [3].

2.19.2 Complexity of Wave let cluster

Wavelet-based clustering is very fast, with a computational complexity of O(n), where n is the number of objects in the database. The algorithm implementation can be made parallel [3].

2.20 CLIQUE (A Dimension Growth Subspace Clustering Method)

The ideas of the CLIQUE clustering algorithm are outlined as follows.

- Given a large set of multidimensional data points, the data space is usually not uniformly occupied by the data points. CLIQUE's clustering identifies the sparse and the "crowded" areas in space (or units), thereby discovering the overall distribution patterns of the data set.
- A unit is dense if the fraction of total data points contained in it exceeds an input model parameter. In CLIQUE, a cluster is defined as a maximal set of connected dense units.

2.20.1 CLIQUE (Clustering in QUEST)

- Automatically identifying subspace of a high dimensional data space that allows better clustering than origin.
- CLIQUE can be considered as both density based and grid based
    - It partitions each dimension into the same number of equal length interval
    - It partitions an m-dimensional data space into no overlapping rectangular units
    - A unit is dense if the fraction of total points contained in the unit exceeds the input model parameter
    - A cluster is a maximal set of connected dense within a subspace

2.20.2 CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspace that contain clusters using the A priori principle
- Identify clusters
    - Determine dense units in all subspace of interests
    - Determine connected dense units in all subspace of interests
- Generate minimal description for the clusters
    - Determine maximal regions that cover a cluster of connected dense units for each cluster Determination of minimal cover for each cluster

2.20.3 Strength and Weakness of CLIQUE

- Strength
    - Automatically finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
    - insensitive to the order of records in input and does not presume some canonical data distribution
    - scales linearly with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
    - The accuracy of the clustering result may be degraded at the expense of simplicity of the method [20]

2.21 PROCLUS (A Dimensional Reduction Subspace Clustering Method)

PROCLUS (PROjected CLUStering) is a typical dimension-reduction subspace clustering method. That is, instead of starting from single-dimensional spaces, it starts by finding an initial approximation of the clusters in the high-dimensional attribute space. Each dimension is then assigned a weight for each cluster, and the updated weights are used in the next iteration to regenerate the clusters. This leads to the exploration of dense regions in all subspaces of some desired dimensionality and avoids the generation of a large number of overlapped clusters in projected dimensions of lower dimensionality.

PROCLUS finds the best set of medoids by a hill-climbing process similar to that used in CLARANS, but generalized to deal with projected clustering. It adopts a distance measure called *Manhattan segmental distance*, which is the Manhattan distance on a set of relevant dimensions. The PROCLUS algorithm consists of three phases: *initialization*, *iteration*, and *cluster refinement*. In the *initialization* phase, it uses a greedy algorithm to select a set of initial medoids that are far apart from each other so as to ensure that each cluster is represented by at least one object in the selected set. More concretely, it first chooses a random sample of data points proportional to the number of clusters we wish to generate, and then applies the greedy algorithm to obtain an even smaller final subset for the next phase. The *iteration* phase selects a random set of *k* medoids from this reduced set (of medoids), and replaces "bad" medoids with randomly chosen new medoids if the clustering is improved. For each medoid, a set of dimensions is chosen whose average distances are small compared to statistical expectation. The total number of dimensions associated to medoids must be *k_l*, where *l* is an input parameter that selects the average dimensionality of cluster subspaces. The *refinement* phase computes new dimensions for each medoid based on the clusters found, reassigns points to medoids, and removes outliers.

Experiments on PROCLUS show that the method is efficient and scalable at finding high-dimensional clusters. Unlike CLIQUE, which outputs many overlapped clusters, PROCLUS finds non overlapped partitions of points. The discovered clusters may help better understand the high-dimensional data and facilitate other subsequence analyses [3].

2.22 Frequent Pattern Based Clustering Methods

This section looks at how methods of *frequent pattern mining* can be applied to clustering, resulting in frequent pattern–based cluster analysis. Frequent pattern mining, as the name implies, searches for patterns (such as sets of items or objects) that occur frequently in large data sets. Frequent pattern mining can lead to the discovery of interesting associations and correlations among data objects. The idea behind frequent pattern–based cluster analysis is that the frequent patterns discovered may also indicate clusters. Frequent pattern–based cluster analysis is well suited to high-dimensional data. It can be viewed as an extension of the dimension-growth subspace clustering approach. However, the boundaries of different dimensions are not obvious, since here they are represented by sets of frequent item sets. That is, rather than growing the clusters

dimension by dimension, we grow sets of frequent item sets, which eventually lead to cluster descriptions. Typical examples of frequent pattern–based cluster analysis include the clustering of text documents that contain thousands of distinct keywords, and the analysis of microarray data that contain tens of thousands of measured values or "features." In this section, we examine two forms of frequent pattern–based cluster analysis: *frequent term–based text clustering* and *clustering by pattern similarity in microarray data analysis*. In frequent term–based text clustering, text documents are clustered based on the frequent terms they contain. Using the vocabulary of text document analysis, a term is any sequence of characters separated from other terms by a delimiter. A term can be made up of a single word or several words. In general, we first remove nontext information (such as HTML tags and punctuation) and stop words. Terms are then extracted. A *stemming algorithm* is then applied to reduce each term to its basic *stem*. In this way, each document can be represented as a set of terms. Each set is typically large. Collectively, a large set of documents will contain a very large set of distinct terms. If we treat each term as a dimension, the dimension space will be of very high dimensionality! This poses great challenges for document cluster analysis. The dimension space can be referred to as *term vector space*, where each document is represented by a term vector. This difficulty can be overcome by *frequent term–based analysis*. We can mine a set of frequent terms from the set of text documents. Then, instead of clustering on high-dimensional term vector space, we need only consider the low-dimensional frequent term sets as "cluster candidates". Notice that a frequent term set is not a cluster but rather the description of a cluster. The corresponding cluster consists of the set of documents containing all of the terms of the frequent term set. A *well-selected subset* of the set of all frequent term sets can be considered as a clustering.
"*How, then, can we select a good subset of the set of all frequent term sets?*" This step is critical because such a selection will determine the quality of the resulting clustering. Let *Fi* be a set of frequent term sets and *cov*(*Fi*) be the set of documents covered by *Fi*. That is, *cov*(*Fi*) refers to the documents that contain all of the terms in *Fi*. The general principle for finding a well-selected subset, *F*1, . . . , *Fk*, of the set of all frequent term sets is to ensure that (1) S*ki* =1*cov*(*Fi*) = *D* (i.e., the selected subset should cover all of the documents to be clustered); and (2) the overlap between any two partitions, *Fi* and *Fj*(for *i* 6= *j*), should be minimized. An overlap measure based on entropy9 is used to assess cluster overlap by measuring the distribution of the documents supporting some cluster over the remaining cluster candidates. An advantage of frequent term–based text clustering is that it automatically generates a description for the generated clusters in terms of their frequent term sets. Traditional clustering methods produce only clusters—a description for the generated clusters requires an additional processing step. Another interesting approach for clustering high-dimensional data is based on pattern similarity among the objects on a subset of dimensions. Here we introduce the p Cluster method, which performs clustering by pattern similarity in microarray data analysis. In DNA microarray analysis, the expression levels of two genes may rise and fall synchronously in response to a set of environmental stimuli or conditions. Under the p Cluster model, two objects are similar if they exhibit a *coherent pattern on a subset of dimensions*. Although the magnitude of their expression levels may not be close, the patterns they exhibit can be very much alike [3].

2.23 Conceptual Clustering

- Unsupervised, spontaneous - categorizes or postulates concepts without a teacher
- Conceptual clustering forms a classification tree - all initial observations in root - create new children using single attribute (not good), attribute combinations (all), information metrics, etc. - Each node is a class
- Should decide quality of class partition and significance (noise)
- Many models use search to discover hierarchies which fulfill some heuristic within and/or between clusters - similarity, cohesiveness, etc [22].

2.23.1 Cobweb

- Cobweb is an incremental hill-climbing strategy with bidirectional operators - not backtrack, but could return in theory
- Starts empty. Creates a full concept hierarchy (classification tree) with each leaf representing a single instance/object. You can choose how deep in the tree hierarchy you want to go for the specific application at hand
- Objects described as nominal attribute-value pairs
- Each created node is a probabilistic concept (a class) which stores probability of being matched (count/total), and for each attribute,

probability of being on, P(a=v|C), only counts need be stored.

- Arcs in tree are just connections - nodes store info across all attributes (unlike ID3, etc.)

2.23.2 Category Utility: Heuristic Measure

- Tradeoff between intra-class similarity and inter-class dissimilarity - sums measures from each individual attribute
- Intra-class similarity a function of P(Ai = Vij|Ck), Predictability of C given V - Larger P means if class is C, A likely to be V. Objects within a class should have similar attributes.
- Inter-class dissimilarity a function of P(Ck|Ai = Vij), Predictiveness of C given V - Larger P means A=V suggests instance is member of class C rather than some other class. A is a stronger predictor of class C.

2.23.3 Category Utility Intuition

- Both should be high over all (most) attributes for a good class breakdown
  - Predictability: P(V|C) could be high for multiple classes, giving a relatively low P(C|V), thus not good for discrimination
  - Predictiveness: P(C|V) could be high for a class, while P(V|C) is relatively low, due to V occurring rarely, thus good for discrimination, but not intra-class similarity
  - When both are high, get best categorization balance between discrimination and intra-class similarity

For each category sum predictability times predictiveness for each attribute weighted by P(Ai = Vij), with k proposed categories, i attributes, j values/attribute

The expected number of attribute values one could guess given C

Conclusion

In this survey paper, our target is to present the most popular clustering algorithm for the purpose of academic benefit. In future, we present the algorithm with sample example and implementation.

References

[1]. Gupta, G. K. *Introduction to data mining with case studies*. PHI Learning Pvt. Ltd., 2006.

[2]. Dunham, Margaret H. *Data mining: Introductory and advanced topics*. Pearson Education India, 2006.

[3]. Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[4]. http://wwwusers.cs.umn.edu/~kumar/dmbook/ch8.pdf

[5]. http://en.wikipedia.org/wiki/K-medoids

[6]. http://www.di.unipi.it/~iwona/Clustering.ppt

[7]. Singh, Shalini S., and N. C. Chauhan. "K-means v/s K-medoids: A Comparative Study." *National Conference on Recent Trends in Engineering & Technology*. 2011.

[8]. Climer, Sharlee, and Weixiong Zhang. "Rearrangement clustering: Pitfalls, remedies, and applications." *The Journal of Machine Learning Research* 7 (2006): 919-943.

[9]. Maulik, Ujjwal, and Sanghamitra Bandyopadhyay. "Genetic algorithm-based clustering technique." *Pattern recognition* 33.9 (2000): 1455-1465.

[10]. Murtagh, Fionn (2002), "Clustering in massive data sets", in Abello, James M.; Pardalos, Panos M.; Resende, Mauricio G. C., *Handbook of massive data sets*, Massive Computing **4**, Springer, pp. 513–516, ISBN 978-1-4020-0489-6.

[11]. Murtagh, Fionn (1983), "A survey of recent advances in hierarchical clustering algorithms", *The Computer Journal* **26** (4): 354–359,doi:10.1093/comjnl/26.4.354.

[12]. http://en.wikipedia.org/wiki/Nearest neighbor_chain_algorithm.

[13]. Kumar, Vipin, Pang-Ning Tan, and Michael Steinbach. "Cluster analysis: basic concepts and algorithms." *Introduction to data mining* (2006): 487-586.

[14]. www.comp.nus.edu.sg/~conggao/cs6203

[15]. M. Kuchaki Rafsanjani, Z. Asghari Varzaneh, N. Emami Chukanlo Pages: 229 - 240"A survey of hierarchical clustering algorithms",The Journal ofMathematics and Computer Science, December, 2012.

[16]. http://www.powershow.com/view/27ce1-NTBmO/ROCK_A_ROBUST_CLUSTERING_ALGORITHM_FOR_CATEGORICAL_ATTRIBUTES_powerpoint_ppt_presentation

[17]. http://genome.tugraz.at/MedicalInformatics2/SOM.pdf

[18]. http://www.cc.gatech.edu/~lingliu/courses/cs4440/notes/PrashantChari.ppt

[19]. https://sites.google.com/a/kingofat.com/wiki/data-mining/cluster-analysis

[20]. http://www.slideshare.net/pierluca.lanzi/machine-learning-and-data-mining-09-clustering-densitybased-gridbased-modelbased?from_search=4

[21]. http://www2.mccombs.utexas.edu/faculty/Maytal.Saar-Tsechansky/Teaching/MIS_382N/Fall2004/Slides/K-NN.ppt

[22]. http://axon.cs.byu.edu/~martinez/classes/778/Papers/Clustering.ppt

Sabiha Firdaus was born in Dhaka, Bangladesh, in 1985. She received the Bachelor of Science degree in Computer Science and Engineering from Manarat International University (MIU), Dhaka, Bangladesh in 2008 and Master of Science degree in Computer Science and engineering from East West University (EWU), Dhaka, Bangladesh in 2011. Then she joined Bangladesh University of Business and Technology (BUBT), Dhaka, Bangladesh at the Department of Computer Science and Engineering as a lecturer. Her research interests include algorithm, clustering, artificial intelligence, database, data mining, and so forth.

Ashraf Uddin received his B.S. and M.S. degrees in computer science and engineering from the University of Dhaka, Dhaka, Bangladesh, in 2010 and 2012, respectively. He worked as a Faculty Member in the Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka, Bangladesh, and in the Department of Computer Science and Engineering, City University, Dhaka, Bangladesh, in the Department of Computer Science and Engineering at Mawlana Bhashani Science and Technology University and in the Department of Computer Science and Engineering at University of Barisal. Currently he is serving as a Lecturer in the department of Computer Science and Engineering, Jagannath University, Dhaka, Bangladesh. His research interests include modeling, analysis, and optimization of protocols and architectures for underwater sensor networks, artificial intelligent, data mining, and so forth.