IMPROVING FUZZY C-MEANS THROUGH BIASING FUZZY MEMBERSHIP MATRIX

By

Muhammad Nufail Farooqi
2009-NUST-MS PhD-CSE-18
MS-09

Submitted to the Department of Computer Engineering in fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

SOFTWARE ENGINEERING

Supervised By

Prof. Dr. Muhammad Younus Javed

College of Electrical & Mechanical Engineering

National University of Science & Technology

DECLARATION

I hereby declare that I have developed this thesis entirely on the basis of my personal efforts under the sincere guidance of my supervisor Prof. Dr. Muhammad Younus Javed. All the sources used in this thesis have been cited and the contents of this thesis have not been plagiarized. No portion of the work presented in this thesis has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

Muhammad Nufail Farooqi

ABSTRACT

Clustering is one of the important data mining tasks. It is the process of partitioning dataset into clusters so that similar objects are placed into the same cluster while dissimilar objects are placed into different clusters. Many clustering techniques have been developed and studied. Based on different theories and methodologies, their types include partitional, hierarchical, density-based, grid-based and model-based clustering algorithms.

A partitional clustering algorithm Fuzzy c-means is a well-known and widely used algorithm for data clustering. Fuzzy c-means uses a fuzzy membership matrix U to represent the degree of membership for data points with the clusters. In this thesis, a new step is introduced to Fuzzy c-means clustering algorithm. In this new step, the fuzzy membership matrix is biased through a suggested multiplying factor to improve its accuracy. Experimental results on both synthetic and real data have shown the better performance attained by improved Fuzzy c-means in comparison to classical Fuzzy c-means algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 01 INTRODUCTION TO CLUSTERING

Clustering, also known as unsupervised classification is the process of categorizing a collection of objects into different categories or clusters; objects inside the same cluster are similar to each other as compared to objects that are contained in a different cluster. Clustering is employed in many areas some application areas are Data Mining, Image Segmentation etc.

1.1 Definition of Cluster

Over the last 50 years, thousands of clustering algorithms have been developed [1]. However, there is still no formal uniform definition of the term cluster. In fact, formally defining cluster is difficult and may be misplaced [2].

Although no formal definition of cluster exists, there are several operational definitions of cluster. For example, Bock [3] suggested that a cluster is a group of data points satisfying various plausible criteria such as

a) Share the same or closely related properties;
b) Show small mutual distances;
c) Have "contacts" or "relations" with at least one other data point in the group;
d) Can be clearly distinguishable from the rest of the data points in the dataset.

Carmichael [4] suggested that a set of data points forms a cluster if the distribution of the set of data points satisfies the following conditions:

a) Continuous and relatively dense regions exist in the data space; and
b) Continuous and relatively empty regions exist in the data space.

Everitt [5] also summarized several operational definitions of cluster. For example, one definition of cluster is that a cluster is a set of data points that are alike and data points from different clusters are not alike. Another definition of cluster is that a cluster is a set of data points such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.

1.2 Data Clustering

Data clustering is a process of assigning a set of records into subsets, called clusters, such that records in the same cluster are similar and records in different clusters are quite distinct [6]. Data clustering is also known as cluster analysis, segmentation analysis, taxonomy analysis, or unsupervised classification.

The term record is also referred to as data point, pattern, observation, object, individual, item, and tuple [7]. A record in a multidimensional space is characterized by a set of attributes, variables, or features.

A typical clustering process involves the following five steps [6]:

a) Pattern representation;
b) Dissimilarity measure definition;
c) Clustering;
d) Data abstraction;
e) Assessment of output.

In the pattern representation step, the number and type of the attributes are determined. Feature selection, the process of identifying the most effective subset of the original attributes to use in clustering, and feature extraction, the process of transforming the original attributes to new attributes, are also done in this step if needed.

In the dissimilarity measure definition step, a distance measure appropriate to the data domain is defined. Various distance measures have been developed and used in data clustering [7]. The most common one among them, for example, is the Euclidean distance.

In the clustering step, a clustering algorithm is used to group a set of records into a number of meaningful clusters. The clustering can be hard clustering, where each record belongs to one and only one cluster, or fuzzy clustering, where a record can belong to two or more clusters with probabilities. The clustering algorithm can be hierarchical, where a nested series of partitions is produced, or partitional, where a single partition is identified.

In the data abstraction step, one or more prototypes (i.e., representative records) of a cluster are extracted so that the clustering results are easy to comprehend. For example, a cluster can be represented by a centroid.

In the final step, the output of a clustering algorithm is assessed. There are three types of assessments: external, internal, and relative [8]. In an external assessment, the recovered structure of the data is compared to a priori structure. In an internal assessment, one tries to determine whether the structure is intrinsically appropriate to the data. In a relative assessment, a test is performed to compare two structures and measure their relative merits.

1.3 Dissimilarity and Similarity Measures

Dissimilarity or distance is an important part of clustering, as majority of the clustering algorithms rely on some distance measure to define the clustering criteria. Since records might have different types of attributes, the appropriate distance measures are also different.

The choice of distances is important for applications, and the best choice is often achieved via a combination of experience, skill, knowledge, and luck. Here, some of the commonly used distances for measurement of numerical data are discussed.

1.3.1 Euclidean Distance

The most common distance measure for continuous data is the Euclidean distance. Given two data points x and y in a d-dimensional space, the Euclidean distance between the two data points is defined as

$$\mathrm{D_{euc}}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{d}}(\mathrm{x_i} - \mathrm{y_i})^2} \qquad (1.1)$$

where xi and yi are the ith components of x and y, respectively.

The Euclidean distance measure is a metric [9].

Clustering algorithms that use the Euclidean distance tend to produce hyper spherical clusters. Clusters produced by clustering algorithms that use the Euclidean distance are invariant to translations and rotations in the data space [10]. One disadvantage of the Euclidean distance is that attributes with large values and variances dominate other attributes with small values and variances. However, this problem can be alleviated by normalizing the data so that each attribute contributes equally to the distance.

The squared Euclidean distance between two data points is defined as

$$\mathrm{D_{squ}}(\mathrm{x},\mathrm{y}) = \sum_{\mathrm{i}=1}^{\mathrm{d}}(\mathrm{x_i} - \mathrm{y_i})^2 \qquad (1.2)$$

1.3.2 Manhattan Distance

Manhattan distance is also called “city block distance” and is defined to be the sum of the distances of all attributes. That is, for two data points x and y in d-dimensional space, the Manhattan distance between them is

$$D_{man}(x,y) = \sum_{i=1}^{d} |x_i - y_i| \tag{1.3}$$

1.3.3 Minkowski distance

The Euclidean distance and the Manhattan distance are special cases of the Minkowski distance, which is defined as

$$D_{min}(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{1.4}$$

Where $p \geq 1$.

1.3.4 Mahalanobis Distance

Mahalanobis distance [8, 11] can alleviate the distance distortion caused by linear combinations of attributes. It is defined by

$$\mathrm{D_{mah}}(\mathrm{x},\mathrm{y}) = \sqrt{(\mathrm{x}-\mathrm{y})\Sigma^{-1}(\mathrm{x}-\mathrm{y})^{\mathrm{T}}} \tag{1.5}$$

Where Σ is the covariance matrix of the data set. Therefore, this distance applies a weight scheme to the data.

1.4 Types of Clustering Algorithms

1.4.1 Hierarchical Clustering Algorithms

Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a dendrogram. Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent. Such an approach allows exploring data on different levels of granularity. Hierarchical clustering methods are categorized into agglomerative (bottom-up) and divisive (top-down) [8, 12]. An agglomerative clustering starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters. A divisive clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved.

Examples of these algorithms are LEGCLUST, BRICH (Balance Iterative Reducing and Clustering using Hierarchies), CURE (Cluster Using Representatives), and Chemeleon.

Advantages of hierarchical clustering include:

a) Embedded flexibility regarding the level of granularity

b) Ease of handling any forms of similarity or distance

c) Consequently, applicability to any attribute types

Disadvantages of hierarchical clustering are related to:

a) Vagueness of termination criteria

b) The fact that most hierarchical algorithms do not revisit once constructed(intermediate) clusters with the purpose of their improvement

1.4.2 Partitional Clustering Algorithms

In contrast to hierarchical clustering, which yields a successive level of clusters by iterative fusions or divisions, partitional clustering assigns a set of data points into K clusters without any hierarchical structure. This process usually accompanies the optimization of a criterion function.

A partitional clustering algorithm is a clustering algorithm that divides a dataset into a single partition. Partitional clustering algorithms can be further classified into two categories: hard clustering algorithms and fuzzy clustering algorithms.

1.4.2.1 Hard Clustering Algorithms

In hard clustering, algorithms assign a class label $l_i \in \{1, 2, . . . , k\}$ to each object x_i to identify its cluster class, where k is the number of clusters. In other words, in hard clustering, each object is assumed to belong to one and only one cluster.

Mathematically, the result of hard clustering algorithms can be represented by a k×n matrix

$$U = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{k1} & \cdots & u_{kn} \end{pmatrix} \qquad (1.6)$$

where n denotes the number of records in the data set, k denotes the number of clusters, and u_{ji} satisfies

$$u_{ji} \in \{0,1\}, 1 \leq j \leq k, 1 \leq i \leq n$$

$$\sum_{j=1}^{k} u_{ji} = 1, 1 \leq i \leq n,$$

$$\sum_{i=1}^{n} u_{ji} > 0, 1 \leq j \leq k,$$

First constraint implies that each object either belongs to a cluster or not. Second constraint implies that each object belongs to only one cluster. And the last one implies that each cluster contains at least one object, i.e., no empty clusters are allowed.

K-means algorithm is an example of hard clustering algorithms.

1.4.2.2 Fuzzy Clustering Algorithms

In fuzzy clustering, the assumption is relaxed so that an object can belong to one or more clusters with probabilities. The result of fuzzy clustering algorithms can also be represented by a k × n matrix U defined in equation (1.6) with the following relaxed constraints:

$$u_{ji} \in [0,1], 1 \leq j \leq k, 1 \leq i \leq n$$

$$\sum_{j=1}^{k} u_{ji} = 1, 1 \leq i \leq n,$$

$$\sum_{i=1}^{n} u_{ji} > 0, 1 \leq j \leq k,$$

Fuzzy c-means is an example of fuzzy clustering algorithms.

1.4.3 Grid-Based Clustering Algorithms

Grid-based clustering algorithms are very efficient for clustering very large datasets since these algorithms perform clustering on the grid cells rather than the individual data points. A typical grid-based clustering algorithm consists of the following basic steps [13]:

a) Construct a grid structure by dividing the data space into a finite number of cells;

b) Calculate the density for each cell;

c) Sort the cells based on their densities;

d) Identify cluster centers;

e) Traverse neighbor cells.

Grid based algorithm quantize the object space into a finite number of cells that forms a grid structure. Operations are done on these grids. The advantage of this method is lower processing time. Clustering complexity is based on the number of populated grid cells and does not depend on the number of objects in the dataset. The major features of this algorithm are:

a) No distance computations.

b) Clustering is performed on summarized data points.

c) Shapes are limited to union of grid-cells.

d) The complexity of the algorithm is usually O(Number of populated grid-cells)

STING is an example of grid based clustering algorithm.

1.4.4 Density-Based Clustering Algorithms

Density-based clustering algorithms are clustering algorithms that are capable of finding arbitrarily shaped clusters. In density-based clustering, a cluster is defined as a dense region surrounded by low-density regions. Usually, density-based clustering algorithms do not require specifying the number of clusters since these algorithms can automatically detect clusters and the number of clusters [14]. A limitation of some density based clustering algorithms is that it is hard to determine certain parameters, such as the density threshold.

Density based algorithm continue to grow the given cluster as long as the density in the neighborhood exceeds certain threshold. This algorithm is suitable for handling noise in the dataset. The following points are enumerated as the features of these algorithms.

a) Handle clusters of arbitrary shape

b) Handle noise

c) Needs only one scan of the input dataset.

d) Needs density parameters to be initialized.

DBSCAN, DENCLUE and OPTICS are examples for density based clustering algorithm.

1.4.5 Subspace Clustering Algorithms

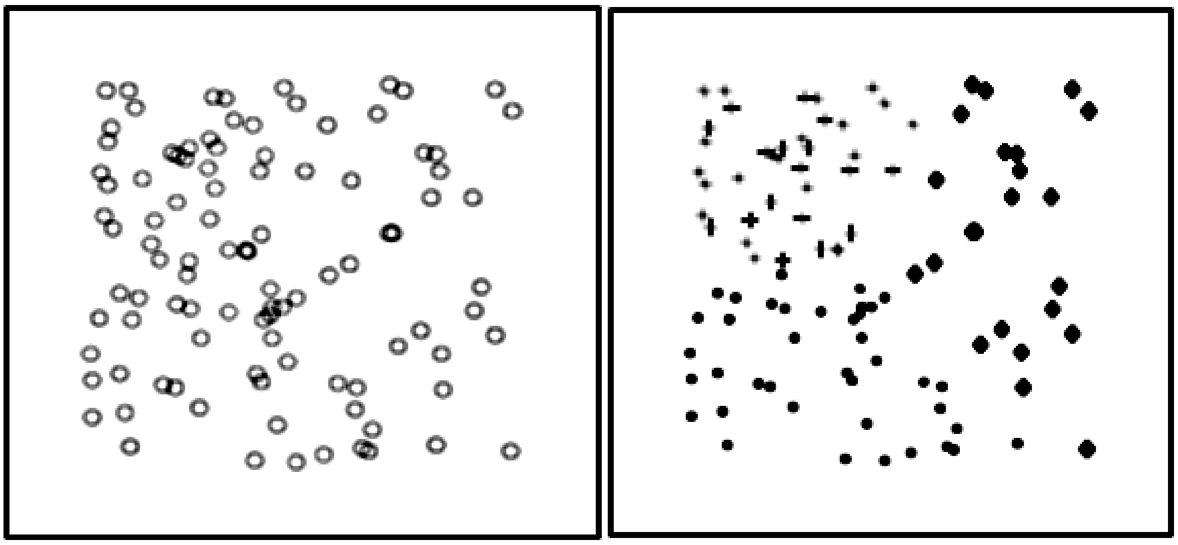
Almost all conventional clustering algorithms do not work well for high dimensional datasets due to the following two problems associated with high dimensional data. First, the distance between any two data points in a high dimensional space becomes almost the same [15]. Second, clusters of high dimensional data are embedded in the subspaces of the data space and different clusters may exist in different subspaces [16].Subspace clustering algorithms are clustering algorithms that are capable of finding clusters embedded in subspaces of the original data space.

Most subspace clustering algorithms can be classified into two major categories [17]: top-down algorithms and bottom-up algorithms. In top-down subspace clustering, a conventional clustering is performed and then the subspace of each cluster is evaluated. In bottom-up subspace clustering, dense regions in low dimensional spaces are identified and then these dense regions are combined to form clusters.

Examples of top-down subspace clustering algorithms include PART [18], PROCLUS, ORCLUS [19], and FINDIT, and δ-cluster. Examples of bottom-up subspace clustering algorithms include CLIQUE [16], ENCLUS, MAFIA, CLTree, DOC, and CBF.

1.5 Cluster Validity

Clustering algorithms tend to find clusters in the data irrespective of the fact whether any clusters are present or not. Figure 1.1(a) shows a dataset with no natural clustering; the points here were generated uniformly in a unit square. However, when the K-means algorithm is run on this data with K = 3, three clusters are identified as shown in Figure 1.1(b). Cluster validity refers to formal procedures that evaluate the results of cluster analysis in a quantitative and objective fashion [8]. In fact, even before a clustering algorithm is applied to the data, the user should determine if the data has any clustering tendency.



(a) **(b)**

Figure 1.1: Cluster validity. (a) A dataset with no "natural" clustering; (b) K-means partition with K=3.

Cluster validity indices can be defined based on three fundamental criteria: internal criteria, relative criteria, and external criteria [8]. Both internal and external criteria are related to statistical testing.

In the external criteria approach, the results of a clustering algorithm are evaluated based on a pre-specified structure imposed on the underlying dataset. Usually, external criteria require using the Monte Carlo simulation to do the evaluation [20]. Hence cluster validity based on external criteria is computationally expensive.

In the internal criteria approach, the results of a clustering algorithm are evaluated based on quantities and features inherited from the underlying dataset. Cluster validity based on internal criteria can be used to assess results of hierarchical clustering algorithms as well as partitional clustering algorithms.

In the relative criteria approach, the results of a clustering algorithm are evaluated with other clustering results, which are produced by different clustering algorithms or the same algorithm but with different parameters. For example, a relative criterion is used to compare the results produced by the k-means algorithm with different parameter k (the number of clusters) in order to find the best clustering of the dataset.

1.6 Clustering Applications

Clustering has been applied in a wide variety of fields, as illustrated below with a number of typical applications [2, 21, 22].

a) Engineering (computational intelligence, machine learning, pattern recognition, mechanical engineering, electrical engineering):
Typical applications of clustering in engineering range from biometric and speech recognition, to radar signal analysis, information compression, and noise removal.

b) Computer Sciences:
There are more and more applications of clustering in web mining, spatial database analysis, information retrieval, textual document collection, and image segmentation.

c) Life and medical sciences:
These areas consist of the major application fields of clustering in its early stage and will continue to be one of the main playing fields for clustering algorithms. Important applications include taxonomy definition, gene and protein function identification, disease diagnosis and treatment.

d) Astronomy and earth sciences:
Clustering can be used to classify stars and planets, investigate land formations, partition regions and cities, and study rivers and mountains systems.

e) Social Sciences:

Interesting applications can be found in behavior pattern analysis, relation identification among different cultures, construction of evolutionary history of languages, analysis of social networks, archeological finding and artifact classification, and the study of criminal psychology.

f) Economics (marketing, business):

Studying customer characteristics and buying patterns recognition, grouping of firms, and stock markets trend analysis all benefit from the use of clustering.

CHAPTER 02 CLUSTERING TECHNIQUES

In the following chapter, some of the clustering techniques are discussed, mainly partitional clustering algorithms (k-means type). The order in which these algorithms are described is how improvements were made to these algorithms with the passage of time. And in the last a grid based clustering STING and density based clustering approach DBSCAN are discussed.

2.1 K-Means Clustering

As the most popular and the simplest partitional clustering algorithm, the k-means algorithm has a long history. In fact, the algorithm was independently discovered by several people from different scientific fields [1]. Since then many variations of the k-means algorithm have been proposed. It partitions the data into K clusters, represented by their centers or means. The center of each cluster is the mean of all the objects inside that cluster.

Let X = $\{X_1, X_2, \ldots. X_n\}$ be a collection of *n* objects where each object X_i is represented as $[x_{i,1}, x_{i,2}, \ldots. x_{i,m}]$, where *m* is the number of numerical attributes. To partition *X* into *k* clusters represented by C = $[c_1, c_2 \ldots. c_k]^T$, k × m matrix containing cluster centers, objective function of equation (2.1) is used

$$J(X, C) = \sum_{j=1}^{k} \sum_{i=1}^{n} D_{i,j} \qquad (2.1)$$

Here, square of Euclidean norm is used as distance measure, i.e,

$$D_{i,j} = \sum_{l=1}^{m} \left(x_{i,l} - c_{j,l}\right)^2, \ 1 \le i \le n, \ 1 \le j \le k \qquad (2.2)$$

K-means algorithm iteratively optimize the objective function by assigning objects to nearest cluster centers and again calculating new cluster center based on that assignment.

Assign object X_i to cluster center c_j such that

$$\arg min_j D_{i,j}, \ 1 \le i \le n, \ 1 \le j \le k \qquad (2.3)$$

Update cluster centers using

$$c_{j,l} = \frac{1}{n_j} \sum_{i=1}^{n_j} X_{i,l}^{j}, \ 1 \le j \le k, \ 1 \le l \le m \qquad (2.4)$$

Where n_j is number of objects in jth cluster and X_i^j is the ith object inside jth cluster.

```
k-means Clustering Algorithm:

  1) Initialize cluster centers.
  2) Repeat
       a. Calculate distances $D_{i,j}$ using equation (2.2)
       b. Assign object $X_i$ to cluster center $c_j$ on the bases of equation (2.3)
       c. Re-compute the cluster means of any changed clusters using (2.4);
```

The purpose of k-means algorithm is simply to find cluster centers in the given data with number of clusters specified in advance. Problem with k-means algorithm is that it is very sensitive to initial cluster centers i.e. different variation of initial cluster centers can produce different end results. In addition to this another drawback associated with k-means is that the number of exact clusters that are present in given the data should be known in advance. While in real world datasets the number of cluster existing in the data is usually unknown.

2.2 Fuzzy K-Means Clustering

The fuzzy k-means algorithm is also referred to as the fuzzy c-means (FCM) algorithm, which was developed by [23] and improved by [24]. Fuzzy k-means clustering algorithm is an extension of k-means clustering algorithm with a fuzzy membership matrix embedded in its objective function. The end results produced by fuzzy k-means clustering algorithm are same as that of k-means clustering but with some improvements in accuracy of determining the cluster centers. The objective function for fuzzy c-means clustering algorithm is given in the following equation (2.5)

$$J_a(\mathrm{X},\mathrm{C},\mathrm{U}) = \sum_{j=1}^{k}\sum_{i=1}^{n}(u_{i,j})^a D_{i,j}\ ,\ 1 \le a < \infty \quad (2.5)$$

Subject to the following condition,

$$\sum_{j=1}^{k} u_{i,j} = 1,\ \ u_{i,j} \in (0,1],\ 1 \le i \le n \qquad (2.6)$$

Where U = $[u_{i,j}]$ is an n × k partition matrix and $u_{i,j}$ corresponds to the association degree of membership that ith object of the given data is having with jth cluster center c_j.

To update the fuzzy membership matrix equation (2.7) is used

$$u_{i,j} = \frac{1}{\sum_{l=1}^{k}\left(\frac{D_{i,j}}{D_{l,j}}\right)^{2/(m-1)}}; \quad 1 \le i \le n,\ 1 \le j \le k \quad (2.7)$$

And to update cluster centers following equation (2.8) is used

$$c_{j,l} = \frac{\sum_{i=1}^{n}(u_{i,j})^m x_{i,l}}{\sum_{i=1}^{n}(u_{i,j})^m},\ 1 \le j \le k,\ 1 \le l \le m \qquad (2.8)$$

Fuzzy c-means Clustering Algorithm:

```
1) Initialize cluster centers
2) Repeat
        a. Compute U using equation (2.7)
        b. Compute C using equation (2.8)
3) Until small change in U
4) Output results.
```

Fuzzy c-means clustering algorithm has the same problems as simple k-means clustering algorithms i.e. its end results sensitivity to initial cluster centers and the number of clusters in given data must be known in advance.

2.3 Competitive Agglomeration Clustering

To deal with the problem i.e. exact number of clusters present in the given data is to be known in advance, faced by k-means and fuzzy k-means clustering algorithms an approach called competitive agglomeration [25] is used. Competitive agglomeration combines the advantages of both hierarchal and partitional clustering. This is done by adding another term to fuzzy k-means objective function. The additional term used in competitive agglomeration controls the number of clusters during the execution of algorithm. Competitive agglomeration starts with initial guess, about the clusters present in the data, considerably more than the clusters present in the data so that the final result will be the exact number of clusters. Starting with more number of initial centers than the original sensitivity to initialization of cluster centers is also reduced. The objective function is as follows

$$\mathrm{J(X,C,U)} = \sum_{j=1}^{k}\sum_{i=1}^{n}(u_{i,j})^2 D_{i,j} - \propto \sum_{j=1}^{k}\left[\sum_{i=1}^{n} u_{i,j}\right]^2 \quad (2.9)$$

Subject to the following condition

$$\sum_{j=1}^{k} u_{i,j} = 1,\ u_{i,j} \in (0,1],\ 1 \leq i \leq n \tag{2.10}$$

Equation (2.11) is used to update the fuzzy membership matrix

$$u_{i,j} = \frac{[1/D_{i,j}]}{\sum_{l=1}^{k}[1/D_{i,l}]} + \frac{\alpha(z)}{D_{i,j}}\left(\sum_{l=1}^{n} u_{l,j} - \frac{\sum_{l=1}^{k}[1/D_{i,l}]\sum_{y=1}^{n} u_{y,l}}{\sum_{l=1}^{k}[1/D_{i,l}]}\right) \tag{2.11}$$

Where *z* is the iteration number and

$$\alpha(z) = \eta_0 exp(-z/\tau)\frac{\sum_{j=1}^{k}\sum_{i=1}^{n}(u_{i,j})^2 D_{i,j}}{\sum_{j=1}^{k}\left[\sum_{i=1}^{n} u_{i,j}\right]^2} \tag{2.12}$$

And to update cluster centers equation (2.13) is used

$$c_{j,l} = \frac{\sum_{i=1}^{n}(u_{i,j})^2 x_{i,l}}{\sum_{i=1}^{n}(u_{i,j})^2},\ 1 \leq j \leq k,\ 1 \leq l \leq m \tag{2.13}$$

The Competitive Agglomeration Algorithm:

```
1) Fix the maximum number of clusters $C = C_{rnax}$;
2) Initialize iteration counter $k = 0$;
3) Initialize the fuzzy C partition $U^{(0)}$;
4) Compute the initial cardinalities $N_i$ for $1 < i < C$
5) Repeat
    a. Compute $d^2(x_j, \beta_i)$ for $1 < i < C, 1 < j < N$.
    b. Update $\alpha(z)$ by using equation (2.12).
    c. Update the partition matrix $U(z)$ by using equation (2.11).
    d. Compute the cardinality $Ni$ for $1 < i < C$.
    e. If $(N_i < \varepsilon_l)$ discard cluster $\beta_i$.
    f. Update the number of clusters C.
    g. Update the prototype parameters.
    h. $z = z + 1$.
6) Until (prototype parameters stabilize).
```

2.4 Agglomerative Fuzzy K-Means Clustering

Another effort for clustering discussed here. Which is known as agglomerative fuzzy k-means clustering [26] uses similar method as competitive agglomerative clustering but with the modified additional term in objective function. The objective function is as follows

$$\mathrm{J}(\mathrm{X}, \mathrm{C}, \mathrm{U}) = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{i,j} D_{i,j} - \lambda \sum_{j=1}^{k} \sum_{i=1}^{n} u_{i,j} \, log u_{i,j} \qquad (2.14)$$

Subject to the condition that

$$\sum_{j=1}^{k} u_{i,j} = 1, \;\; u_{i,j} \in (0,1], \; 1 \le i \le n \qquad (2.15)$$

To update fuzzy membership matrix

$$u_{i,j} = \frac{\exp\left(\frac{-\mathrm{D}_{\mathrm{i,j}}}{\lambda}\right)}{\sum_{l=1}^{k} exp\left(\frac{-\mathrm{D}_{\mathrm{l,j}}}{\lambda}\right)} \qquad (2.16)$$

And to update cluster centers

$$c_{j,l} = \frac{\sum_{i=1}^{n} u_{i,j} x_{i,l}}{\sum_{i=1}^{n} u_{i,j}}, \;\; 1 \le j \le k, \; 1 \le l \le m \qquad (2.17)$$

The value of λ starts from a low value such as 0.1 and increase linearly as iteration progresses.

The Agglomerative Fuzzy K-Means Algorithm:

```
1) Set the penalty factor λ. randomly choose initial points $C^{(0)} = \{C_1, C_2, ..., C_k\}$.
   Determine $U^{(0)}$ such that $J(U^{(0)}, C^{(0)})$ is minimized by using (2.16). Set t = 0.
2) Let $Z = Z^t$, solve problem J(U, Z) to obtain $U^{t+1}$.
       a. If $J(U^{t+1}, Z) = J(U^t, Z)$ output $(U^t, Z)$ and stop, otherwise, go to step (3).
3) Let $U=U^{t+1}$ solve problem P(U, Z) to obtain $Z^{t+1}$.
       a. IF $P(U, Z^{t+1}) = P(U,Z^t)$, output $(U, Z^t)$ and stop, otherwise, set t = t + 1 and
          go to step (2).
```

This approach minimizes the objective function (2.14), which is the sum of the objective function of the fuzzy k-mean and the entropy function. The initial number of clusters is set to be larger than the true number of clusters in a given data set. With the entropy cost function, each initial cluster centers will move to the dense centers of the clusters in a data set. These initial cluster centers are merged at the same location, and the number of the determined clusters is just the number of the merged clusters in the output of the algorithm.

2.5 The Attributes-Weighting Clustering Algorithm

The clustering problem has been studied extensively and many algorithms have been developed according to the specific domain requirements. Usually, the distance or dissimilarity functions of these algorithms involve all attributes of the data set. This is applicable if all or most attributes are important to every cluster. However, the clustering results become less accurate if a significant number of attributes are not important to some clusters. In the literature, the problem of the selection of important attributes is solved by feature selection techniques. These techniques derive a set of important features from all the attributes for the clustering analysis. The feature selection can be viewed as a preprocessing step in the clustering procedure. The limitation of this preprocessing step is that the clustering algorithm is still based on the whole set of new features. However, each cluster may have a different set of important attributes, and each cluster may contain some unimportant features. To tackle the above clustering problem, the weighted dissimilarity measure [27] is defined. In addition to finding out the cluster of objects belong to; the weighted dissimilarity measure also gives the set of important attributes for each cluster. Here the term "a set of attributes with high weights" will be used to describe the set of attributes with the highest power in distinguishing the objects of a cluster from other objects. Additional feature selection techniques cannot handle these characteristics of clusters.

Let X is a set of n objects described by m attributes. The attributes-weighting clustering algorithm to cluster X into k clusters can be stated as an algorithm which attempts to minimize the objective function (2.18)

$$\mathrm{J(X, C, U, W)} = \sum_{j=1}^{k}\sum_{i=1}^{n}\sum_{l=1}^{m} u_{i,j} w_{j,l}{}^{\beta}\left(x_{i,l} - c_{j,l}\right)^2 \qquad (2.18)$$

Subject to the conditions

$$\sum_{j=1}^{k} u_{i,j} = 1,\ \ u_{i,j} \in \{0,1\},\ 1 \leq i \leq n \qquad (2.19)$$

$$\sum_{l=1}^{m} w_{j,l} = 1, \quad 0 \le w_{j,l} \le 1, \; 1 \le j \le k \tag{2.20}$$

Where k , n , and m are respectively the number of clusters, objects, and dimensions. $x_{i,l}$ is the value of the lth dimension of the ith object. C = [$c_{j,l}$] is a k-by-m matrix, and $c_{j,l}$ is the value of the lth dimension of the jth cluster center. U = [$u_{i,j}$] is a k-by-n matrix, and $u_{i,j}$ is the degree of membership of the ith object belonging to the jth cluster. W = [$w_{j,l}$] is a k-by-m matrix, and $w_{j,l}$ is the weight of the lth dimension in the jth cluster. β is a parameter that is greater than 1.

In (2.18), U, W and C are three unknowns, and each of them is calculated by taking the other two constant. They are presented as follows:

$$u_{i,j} = \begin{Bmatrix} 1, if \; \sum_{l=1}^{m} w_{j,l}\left(x_{i,l} - c_{j,l}\right)^2 \le \sum_{l=1}^{m} w_{z,l}\left(x_{i,l} - c_{z,l}\right)^2, 1 \le z \le k \\ 0, otherwise \end{Bmatrix} \tag{2.21}$$

$$c_{j,l} = \frac{\sum_{i=1}^{n} u_{i,j} x_{i,l}}{\sum_{i=1}^{n} u_{i,j}} \tag{2.22}$$

$$w_{j,l} = \frac{1}{\sum_{t=1}^{m} \left[\frac{\sum_{i=1}^{n} u_{i,j}\left(x_{i,l} - c_{j,l}\right)^2}{\sum_{i=1}^{n} u_{i,j}\left(x_{i,t} - c_{j,t}\right)^2} \right]^{1/(\beta-1)}} \tag{2.23}$$

The Attributes-Weighting Clustering Algorithm:

1) Initialize $C^{(1)}$ and set $W^{(1)}$ all entries equal to 1/m, set t=1.
2) Determine $U^{(t+1)}$ using (2.21)
3) If $J(C^{(t)},U^{(t+1)},W^{(t)}) = J(C^{(t)},U^{(t)},W^{(t)})$ then stop, otherwise go to step 4.
4) Determine $C^{(t+1)}$ using (2.22)
5) If $J(C^{(t+1)},U^{(t+1)},W^{(t)}) = J(C^{(t)},U^{(t+1)},W^{(t)})$ then stop, otherwise go to step 6.
6) Determine $W^{(t+1)}$ using (2.23)
7) If $J(C^{(t+1)},U^{(t+1)},W^{(t+1)}) = J(C^{(t+1)},U^{(t+1)},W^{(t)})$ then stop, otherwise go to step 2.

To evaluate the performance and efficiency of the attributes-weighting algorithm and compare it with the standard k-means algorithm (the algorithm without weighting on the attributes), several tests were carried out using these algorithms on both real and artificial data sets. Experimental results had shown the improvement in the new algorithm

2.6 Entropy Weighting K-Means Clustering

2.6.1 Entropy

Entropy [47] is a measure of uncertainty regarding a discrete random variable. For many purposes, the Shannon entropy is the only measure needed. Shannon entropy is defined by

$$H(X) = -\sum_{x=0}^{\infty} P(X = x)\log(P(X = x)) \qquad (2.24)$$

The Shannon entropy is a fundamental measure in information theory. It was introduced by Claude E. Shannon, now considered the father of information theory, in [48]. Much can be said about its properties, its uniqueness, and its relation with the thermodynamical entropy in physics, but we will only scratch a little bit on the surface here. One way of understanding it better is to rewrite the definition as

$$H(X) = E\left(-\log\left(P_X(X)\right)\right) \qquad (2.25)$$

where $P_X(X)$ is the probability ascribed to the value of X that turns out to be correct. This makes $P_X(X)$ a discrete random variable, but not necessarily with integer values.

Now it is clear that the Shannon entropy is the expectation value of –log(p) where p is the probability assigned to the measured value of the random variable. –log(p) can be interpreted as the needed length, in bits, of a message communicating a measurement that had probability p, which makes the Shannon entropy a measure of the expected message length needed to communicate the measured value of a random variable. The Shannon entropy of a uniform random variable with n possible values is

$$H(Q_n^0) = E\left(-\log\left(\frac{1}{n}\right)\right) = \log(n) \qquad (2.26)$$

which means that we need log(n) bits to communicate one choice from n different equally likely states. Without qualifiers, the word entropy and a non-subscripted H normally refers only to Shannon entropy.

2.6.2 Entropy weighting k-means algorithm

Entropy weighting k-means algorithm is a new k-means type algorithm for soft subspace clustering of high-dimensional sparse data [28]. In this algorithm, it is assumed that the weight of a dimension in a cluster represents the probability of contribution of that dimension in forming the cluster. The entropy of the dimension weights represents the certainty of dimensions in the identification of a cluster. Therefore, the objective function (2.18) is modified by adding the weight entropy term to it so that it can simultaneously minimize the within cluster dispersion and maximize the negative weight entropy to stimulate more dimensions to contribute to the identification of clusters. In this way, it could avoid the problem of identifying clusters by few dimensions in sparse data.

$$\mathrm{J(X,C,U,W)} = \sum_{j=1}^{k}\left[\sum_{i=1}^{n}\sum_{l=1}^{m} u_{i,j} w_{j,l}\left(x_{i,l}-c_{j,l}\right)^2 + \gamma \sum_{l=1}^{m} w_{j,l} log w_{j,l}\right] \quad (2.27)$$

Subject to the conditions

$$\sum_{j=1}^{k} u_{i,j} = 1,\ \ u_{i,j} \in \{0,1\},\ 1 \le i \le n \quad (2.28)$$

$$\sum_{l=1}^{m} w_{j,l} = 1,\ \ 0 \le w_{j,l} \le 1,\ 1 \le j \le k \quad (2.29)$$

The first term in (2.27) is the sum of the dispersions within a cluster, and the second term the negative weight entropy. The positive parameter γ controls the strength of the incentive for clustering on more dimensions. Solving (2.27) for minimization C, U and W are determined using following equations:

$$u_{i,j} = \left\{\begin{matrix} 1, if\ \sum_{l=1}^{m} w_{j,l}\left(x_{i,l}-c_{j,l}\right)^2 \le \sum_{l=1}^{m} w_{z,l}\left(x_{i,l}-c_{z,l}\right)^2, 1 \le z \le k \\ 0, otherwise \end{matrix}\right\} \quad (2.30)$$

$$c_{j,l} = \frac{\sum_{i=1}^{n} u_{i,j} x_{i,l}}{\sum_{i=1}^{n} u_{i,j}} \quad (2.31)$$

$$w_{j,l} = \frac{\exp(\frac{D_{j,l}}{\gamma})}{\sum_{t=1}^{m} \exp(\frac{D_{j,t}}{\gamma})} \quad (2.32)$$

Where $D_{j,l} = \sum_{i=1}^{n} u_{i,j}\left(x_{i,l}-c_{j,l}\right)^2$

The input parameter γ is used to control the size of the weights as follows:

γ > 0. In this case, according to (2.32), $w_{j,l}$ is inversely proportional to $D_{j,l}$. The smaller $D_{j,l}$, the larger $w_{j,l}$, the more important the corresponding dimension.

γ = 0. $w_{j,l}$ is equal to one, indicating that the index l' has the smallest value of $D_{j,l}$. The other weights $w_{j,l}$ for $l \neq l'$ are equal to zero. Each cluster contains only one important dimension. It may not be desirable for high-dimensional data sets.

$\gamma < 0$. In this case, according to (2.32), $w_{j,l}$ is directly proportional to $D_{j,l}$. The larger $D_{j,l}$, the larger $w_{j,l}$. This is contradictory to the original idea of dimension weighting. Therefore, γ cannot be smaller than zero.

```
Entropy Weighting K-Means Clustering Algorithm:

  1) Input the number of clusters k and parameter $\gamma$.
  2) Randomly choose k cluster centers and set all initial weights to 1/m.
  3) Repeat
        a. Update the partition matrix U by (2.30).
        b. Update the cluster centers C by (2.31).
        c. Update the dimension weights W by (2.32).
  4) Until (the objective function obtains its local minimum value).
```

This algorithm simultaneously minimizes the dispersion within cluster and maximizes the negative weight entropy in the clustering process. Because this clustering process awards more dimensions to make contributions to identification of each cluster, the problem of identifying clusters by few sparse dimensions can be avoided. As such, the sparsity problem of high-dimensional data is tackled. The experimental results on both synthetic and real data sets have shown that the new algorithm outperformed other k-means type algorithms, for example, Bisecting k-means and FWKM, and subspace clustering methods, for example, PROCLUS, HARP, LAC, SCAD1, and COSA, in recovering clusters. Except for clustering accuracy, the new algorithm is scalable to large high-dimensional data and easy to use because the input parameter γ is not sensitive. The weight values generated in the clustering process are also useful for other purposes, for instance, identifying the keywords to represent the semantics of text clustering results.

2.7 Improved Entropy Weighting K-Means Clustering

The objective of traditional k-means algorithm is to make the distances of objects in the same cluster as small as possible, but another objective, the distances among objects from different clusters is not taken into account. Here an improved k-means algorithm satisfying both of the above objectives is presented. In improved entropy weighted k-means clustering [29] the cost function of entropy weighting k-means clustering algorithm is modified by adding a variable that is relevant linearly to the square sum of distances from the mean of all objects and the means of all clusters. Improved objective function is given below in (2.33).

$$J(X,C,U,W)=\sum_{j=1}^{k}\left[\frac{\sum_{i=1}^{n}\sum_{l=1}^{m}u_{i,j}w_{j,l}\left(x_{i,l}-c_{j,l}\right)^{2}}{\sum_{l=1}^{m}\left(c_{j,l}-\bar{x}_{l}\right)^{2}}\right]+\sum_{j=1}^{k}\left[\gamma\sum_{l=1}^{m}w_{j,l}\,log w_{j,l}\right] \quad (2.33)$$

Subject to the conditions

$$\sum_{j=1}^{k}u_{i,j}=1,\ \ u_{i,j}\in\{0,1\},\ 1\le i\le n \qquad (2.34)$$

$$\sum_{l=1}^{m}w_{j,l}=1,\ \ 0\le w_{j,l}\le 1,\ 1\le j\le k \qquad (2.35)$$

Here, $\bar{x}$ is the mean of all objects, $\bar{x}_l$ is the value of l^th dimension of $\bar{x}$, where $\bar{x}_l=\frac{1}{n}\sum_{i=1}^{n}x_{i,l}$. If n>1 the new algorithm is available, otherwise, $\sum_{l=1}^{m}\left(c_{j,l}-\bar{x}_l\right)^2$ is zero and the cost function J(X, C, U, W) is not computable. The denominator is a variable and is linear to the square sum of the distances from the mean of all objects to the means of all clusters.

Minimizing the objective function in (2.33) subject to the constraints (2.34) and (2.35), the three unknown are calculated as follows

$$u_{i,j}=\begin{cases}1, if\ \sum_{l=1}^{m}w_{j,l}\left(x_{i,l}-c_{j,l}\right)^{2}\le\sum_{l=1}^{m}w_{z,l}\left(x_{i,l}-c_{z,l}\right)^{2}, 1\le z\le k\\ 0, otherwise\end{cases} \quad (2.36)$$

$$c_{j,l}=\frac{\sum_{i=1}^{n}u_{i,j}x_{i,l}}{\sum_{i=1}^{n}u_{i,j}} \qquad (2.37)$$

$$w_{j,l}=\frac{\exp\left(\frac{-\Psi_{j,l}}{\gamma}\right)}{\sum_{t=1}^{m}\exp\left(\frac{-\Psi_{j,t}}{\gamma}\right)} \qquad (2.38)$$

Here $\Psi_{j,l}=\frac{\sum_{i=1}^{n}u_{i,j}\left(x_{i,l}-c_{j,l}\right)^{2}}{\sum_{l=1}^{m}\left(c_{j,l}-\bar{x}_{l}\right)^{2}}$

```
Improved Entropy Weighting K-Means Clustering Algorithm:

5) Input the number of clusters k and parameter $\gamma$.
6) Randomly choose k cluster centers and set all initial weights to 1/m.
7) Repeat
     a. Update the partition matrix U by (2.36).
     b. Update the cluster centers C by (2.37).
     c. Update the dimension weights W by (2.38).
8) Until (the objective function obtains its local minimum value).
```

The results of experiment on Iris data have shown that the proposed algorithm outperform the traditional k-means algorithm and the EWKM algorithm.

2.8 STING

Wang [30] proposed a STatistical INformation Grid-based clustering method (STING) to cluster spatial databases. The algorithm can be used to facilitate several kinds of spatial queries. The spatial area is divided into rectangle cells, which are represented by a hierarchical structure. Let the root of the hierarchy be at level 1, its children at level 2, etc. The number of layers could be obtained by changing the number of cells that form a higher-level cell. A cell in level i corresponds to the union of the areas of its children in level i + 1. In the STING algorithm, each cell has 4 children and each child corresponds to one quadrant of the parent cell. Only two-dimensional spatial space is considered in this algorithm.

For each cell, the attribute-dependent and attribute-independent parameters are defined as follows (used in the STING Algorithm):

n	– number of objects (points) in the cell;
m	– mean of all values in a cell;
s	– standard deviation of all values of the attribute in a cell;
min	– the minimum value of the attribute in a cell;
max	– the maximum value of the attribute in a cell;
dist	– the type of distribution that the attribute value in a cell follows.

Given the hierarchical structure of grid cells, we can start with the root to calculate the likelihood that a cell is relevant to the query at some confidence level using the parameters of a cell.

The STING algorithm

```
Construct the grid hierarchical structure according to the database and generate the
parameters of each cell;
Determine a layer to begin with;
For each cell in the layer, compute the confidence interval of the probability that the cell is
relevant to the query;
if the layer is not the bottom layer then
        Go to the next level in the hierarchy structure and go to step 3 for the relevant cells
        of the higher-level layer;
else if the specification of the query is met then
        Find the regions of relevant cells and return those regions that meet the
        requirements of the query;
else
        Reprocess the data in the relevant cells and return the results that meet the
        requirements of the query;
end if
```

STING has a run time complexity of O(L), where L is the number of cells at the lowest level. Because L is usually much smaller than the number of points in the data set, STING achieves better performance in simulation studies than other algorithms, such as BIRCH. STING is also extended as STING+ [31] to deal with dynamically evolving spatial data while maintaining the similar hierarchical structure. STING+ supports user defined triggers, which are decomposed into sub - triggers associated with cells in the hierarchy. STING+ considers four categories of triggers based on the absolute or relative condition on certain regions or features.

It has much less computational cost than other approaches. The I/O cost is low since it usually keeps the STING data structure in memory. Both of these factors will speed up the processing of spatial data query tremendously. In addition, it offers an opportunity for parallelism (STING is trivially parallelizable). All these advantages benefit from the hierarchical structure of grid cells and the statistical information associated with them.

2.9 DBSCAN

Ester [32] proposed a density-based clustering algorithm called DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to discover arbitrarily shaped clusters. Only one input parameter is required, and the algorithm also supports the user in determining an appropriate value for this input parameter.

To describe the algorithm, start with some definitions and notation. An important concept in density-based algorithms is the ε-neighborhood of a point. Let x be a point. Then the ε-neighborhood of x is denoted by $N_\varepsilon(x)$ and is defined as follows.

ε-neighborhood of a point. The ε-neighborhood of a point x is defined as

$$N_\varepsilon(x) = \{y \in D: d(x,y) \le \varepsilon\} \qquad (2.39)$$

where D is the data set and d(·, ·) is a certain distance function.

Directly density-reachable. A point x is said to be directly density reachable from a point y (with respect to ε and N_{min}) if

a) $x \in N_\varepsilon(y)$;
b) $|N_\varepsilon(y)| \ge N_{min}$, where $|N_\varepsilon(y)|$ denotes the number of points in $N_\varepsilon(y)$.

Directly density-reachable is symmetric for pairs of core points (points inside a cluster), but it is in general not symmetric in case one core point and one border point (a point on the border of a cluster) are involved. As an extension of directly density-reachable, density-reachable, defined below, is also not symmetric in general. But density-connected is a symmetric relation.

Density-reachable. A point x is said to be density-reachable from a point y if there is a sequence of points $x = x_1, x_2, \ldots, x_i = y$ such that x_l is directly density-reachable from x_{l+1} for $l = 1, 2, \ldots, i - 1$.

Density-connected. Two points x and y are said to density-connected with respect to ε and N_{min} if there exists a point z such that both x and y are density-reachable from z with respect to ε and N_{min}.

A cluster is then very intuitively defined as a set of density-connected points that is maximal with respect to density-reachability. Mathematically, they can be defined as.

Cluster. Let D be a data set. A cluster C with respect to ε and N_{min} is a nonempty subset of D satisfying the following conditions:

a) $\forall x, y \in D$, if $x \in C$ and y is density-reachable from x with respect to ε and N_{min}, then $y \in C$(maximality).
b) $\forall x, y \in C$, x and y are density-connected with respect to ε and N_{min} (connectivity).

The noise is a set of points in the data set that do not belong to any cluster. It can be seen from definition that a cluster contains at least N_{min} points. DBSCAN starts with an arbitrary point x and finds all points that are density-reachable from x with respect to ε and N_{min}. If x is a core point, then a cluster with respect to ε and N_{min} is formed. If x is a border point, then no points are density-reachable from x and DBSCAN visits the next unclassified point. DBSCAN may merge two clusters if the two clusters are close to each other. In DBSCAN, the distance between two clusters C_1 and C_2 is defined as $d(C_1,C_2) = \min_{x\in C1,y\in C2} d(x, y)$. DBSCAN tends to merge many slightly connected clusters together.

DBSCAN requires two parameters, ε and N_{min}. These two parameters are used globally in the algorithm, i.e., the two parameters are the same for all clusters, so to choose the two parameters in advance is not easy. However, a heuristic is developed in DBSCAN to determine the parameters ε and N_{min} of the "thinnest" cluster in the database. This heuristic is called the sorted k-dist graph. Let $F_k : D \rightarrow R$ be a function defined as

$$F_k(x) = \text{distance between x and its } k^{th} \text{ nearest neighbor.}$$

Then $F_k(D)$ is sorted in descending order and plotted in a two-dimensional graph. ε is set to $F_4(z_0)$, where z_0 is the first point in the first "valley" of the graph of the sorted 4-dist graph, since the k-dist graph for $k > 4$ does not significantly differ from the 4-dist graph [Ester, 1996]. Hence N_{min} is set to 4 for all two-dimensional data.

DBSCAN has several variations and extensions. DBSCAN to GDBSCAN, which can cluster point objects and spatially extended objects with spatial and nonspatial attributes as well.

GDBSCAN generalizes DBSCAN in two ways: the notion of a neighborhood and the measure in a neighborhood. PDBSCAN is a parallel version of DBSCAN. DBSCAN is also extended to incremental clustering. An algorithm called DBCluC (Density-Based Clustering with Constraints) based on DBSCAN is developed to cluster spatial data in the presence of obstacles.

CHAPTER 03 DESIGN AND IMPLEMENTATION

In order to confirm the study carried out, the improvements suggested in the fuzzy c-means clustering algorithm are modeled and integrated. This chapter provides a detail discussion about the design and implementation of the additional features.

3.1 Motivation

The aim of a cluster analysis is to partition a given set of data or objects into clusters (subsets, groups, classes). This partition should have the following properties:

a) Homogeneity within the clusters, i.e. data that belong to the same cluster should be as similar as possible.
b) Heterogeneity between clusters, i.e. data that belong to different clusters should be as different as possible. [33]

From the above definition it is clear that the result of clustering, i.e. resultant clusters, should have two properties one is similarity of data in same cluster and other is dissimilarity of data w.r.t other clusters. Most of the clustering algorithms designed so far consider only the first property to distribute data among different clusters. The reason behind the use of single property may be that addressing one property indirectly incorporates the second property of a cluster. Addressing single property at a time in clustering may be enough but the fact that using both the properties of a cluster exclusively in parallel to partition data in different clusters will definitely be having an edge. To incorporate both properties of a cluster in the process of data analysis some efforts made by various scientists resulted in a positive outcome. Here, another effort is presented towards the said technique in order to analyze its effectiveness.

3.2 Related Work

The second objective of clustering, which is to make the distance of objects from different clusters as significant as possible, is not taken into account in traditional clustering algorithm. Taoying Li and Yan Chen [29] proposed an improved algorithm by modifying the cost function of entropy weighted k-means clustering algorithm (EWKM algorithm).The improved algorithm adjusted the existing cost function by adding a variable relevant to the distances between the mean of all objects and the mean of each cluster. The results of experiments on Iris data have shown that their improved algorithm outperformed the traditional k-means algorithm and the EWKM algorithm.

3.3 Improved Fuzzy c-means Clustering Algorithm

To achieve maximum dissimilarity among objects of different clusters, it is proposed to multiply the fuzzy membership matrix by a biasing factor. The biasing factor that will be used here is the square of Euclidian distance between the object and the mean of remaining cluster centers.

As the distance between the object and the mean of remaining cluster center increases multiplying the membership of that object to its cluster centers with an increased biased factor will increase its membership value. On the other hand if the distance is less so it will also increase the membership value but the increase here will be less significant as compared to the first case. Thus biasing the membership matrix leaves the matrix in a state having increased membership values for objects having more distance between them and other clusters.

The objective function for this is same as fuzzy c-means mentioned below

$$J_a(\mathrm{X},\mathrm{C},\mathrm{U}) = \sum_{j=1}^{k}\sum_{i=1}^{n}(u_{i,j})^a D_{i,j}\ ,\ 1 \le a < \infty \tag{3.1}$$

Subject to the following condition,

$$\sum_{j=1}^{k} u_{i,j} = 1,\ \ u_{i,j} \in (0,1],\ 1 \le i \le n \tag{3.2}$$

Where U = $[u_{i,j}]$ is an n × k partition matrix and$u_{i,j}$ corresponds to the association degree of membership that i^th object of the given data is having with j^th cluster center c_j.

To update the fuzzy membership matrix equation (3.3) is used

$$u_{i,j} = \frac{1}{\sum_{l=1}^{k}\left(\frac{D_{i,j}}{D_{l,j}}\right)^{2/(m-1)}};\qquad 1 \le i \le n,\ 1 \le j \le k \tag{3.3}$$

And to update cluster centers following equation (3.4) is used

$$c_{j,l} = \frac{\sum_{i=1}^{n}(u_{i,j})^m x_{i,l}}{\sum_{i=1}^{n}(u_{i,j})^m},\ \ 1 \le j \le k,\ 1 \le l \le m \tag{3.4}$$

And biasing factor used here is mentioned below

$$B_{i,j} = \frac{1}{k-1}\sum_{p=1,p\neq j}^{k}\sum_{l=1}^{m}\left|x_{i,l} - c_{p,l}\right|,\ 1 \leq i \leq n, 1 \leq j \leq k \quad (3.5)$$

Where $B_{i,j}$ is the biasing factor for fuzzy membership matrix value of ith object with jth cluster and $k \geq 2$.

Improved Fuzzy c-means Clustering Algorithm:

```
1) Initialize cluster centers
2) Repeat
    a. Compute U using equation (3.3)
    b. Compute biasing factor using (3.5)
    c. Multiply calculated biasing factor with U
    d. Normalize U to fulfill condition (3.2)
    e. Compute C using equation (3.4)
3) Until small change in U
4) Output results.
```

3.4 Implementation Flow Chart for Improved Fuzzy c-means

The overall algorithm is implemented in a framework shown in Figure 3.1, which runs the improved fuzzy c-means algorithm to discover the best cluster centers. In the implementation there are three input parameters, first the data points itself that are to be clustered second the number of clusters present in the data, and finally the stopping criteria. The stopping criteria defined here is ε, which is the minimum change occurring in the value of objective function during successive iterations. Ideally the value of ε is needed to be zero, which is practically difficult to attain so we fixed it at a minimum value of 0.0001. Stopping criteria can also be defined as a fixed number of iterations for which the proposed algorithm is executed that number of times.

The improved fuzzy c-means like other k-means type algorithm starts with the initialization of clusters. Here it is done through initializing fuzzy membership matrix; other already developed advanced methods can be used to initialize the clusters to improve more the performance of algorithm.

Next a loop executes until the stopping criteria is met. In the loop iteratively the cluster centers and biased fuzzy membership matrix are calculated to reach the true centers of the clusters. Upon termination of the loop best results are produced in the output.

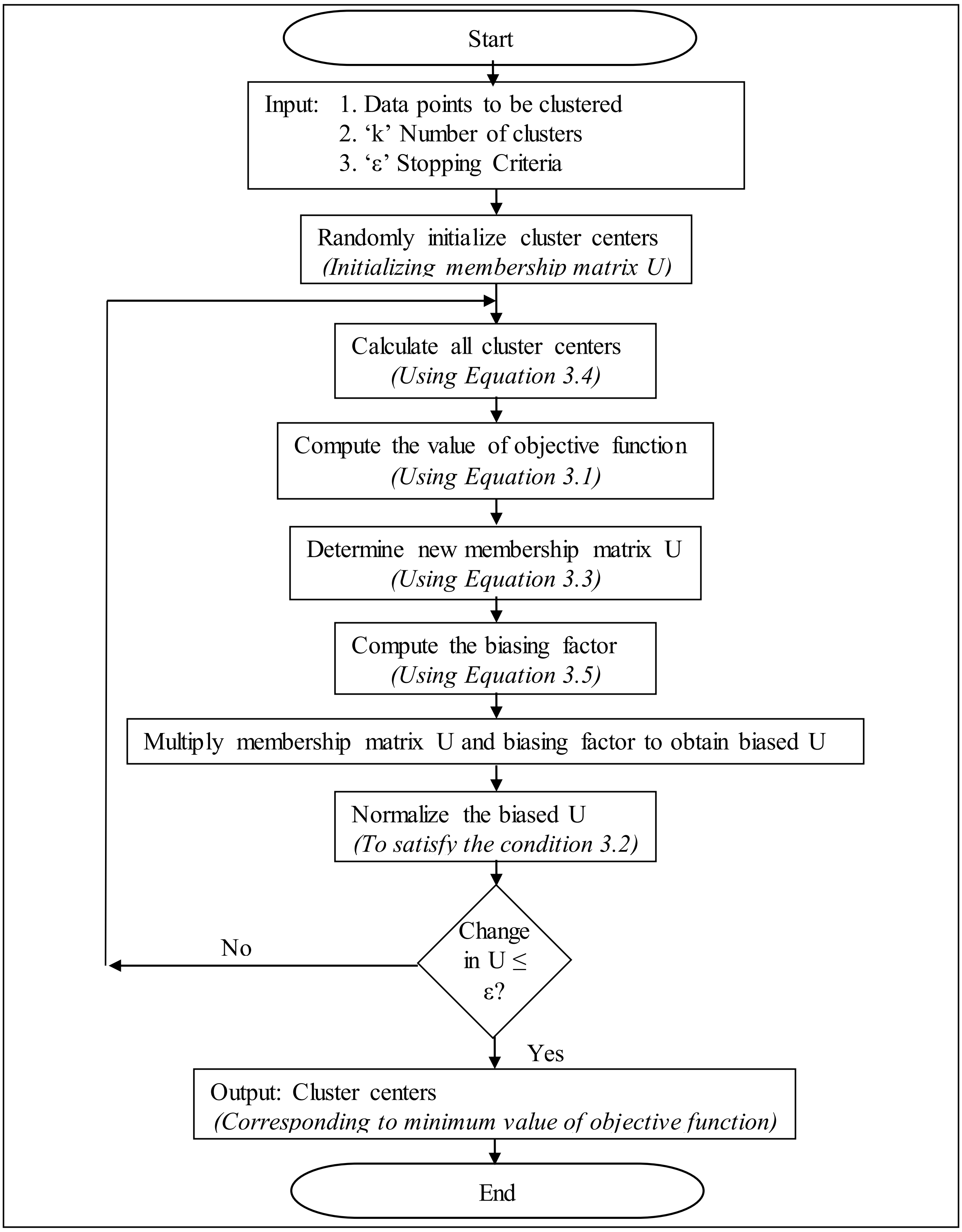

Start
Input: 1. Data points to be clustered
2. 'k' Number of clusters
3. 'ε' Stopping Criteria
Randomly initialize cluster centers
(Initializing membership matrix U)
Calculate all cluster centers
(Using Equation 3.4)
Compute the value of objective function
(Using Equation 3.1)
Determine new membership matrix U
(Using Equation 3.3)
Compute the biasing factor
(Using Equation 3.5)
Multiply membership matrix U and biasing factor to obtain biased U
Normalize the biased U
(To satisfy the condition 3.2)
Change in U ≤ ε?
No
Yes
Output: Cluster centers
(Corresponding to minimum value of objective function)
End

Figure 3.1 Implementation Flow Chart for Improved FCM

3.5 Working

To explain the working of proposed improvement an example is presented. Here a dataset of nine data points are taken in three clusters with three data points in each cluster. All data points with initial cluster centers position are shown in Figure 3.2(a). In this figure nine dots represents data points and three starts indicates initial center positions of three clusters C1, C2 and C3. Forgy initialization method is used here to initialize clusters centers.

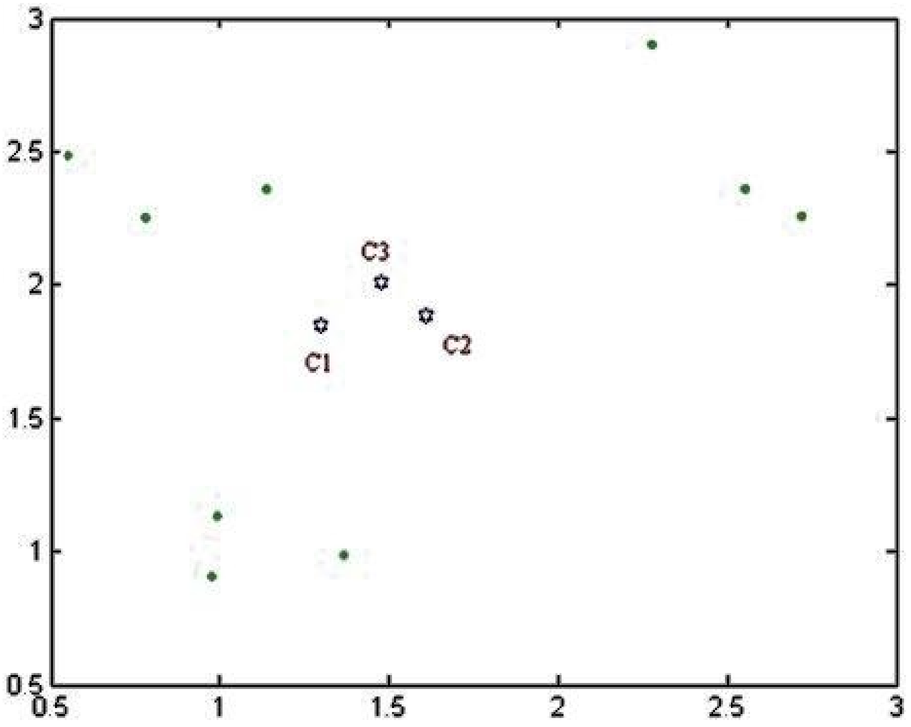
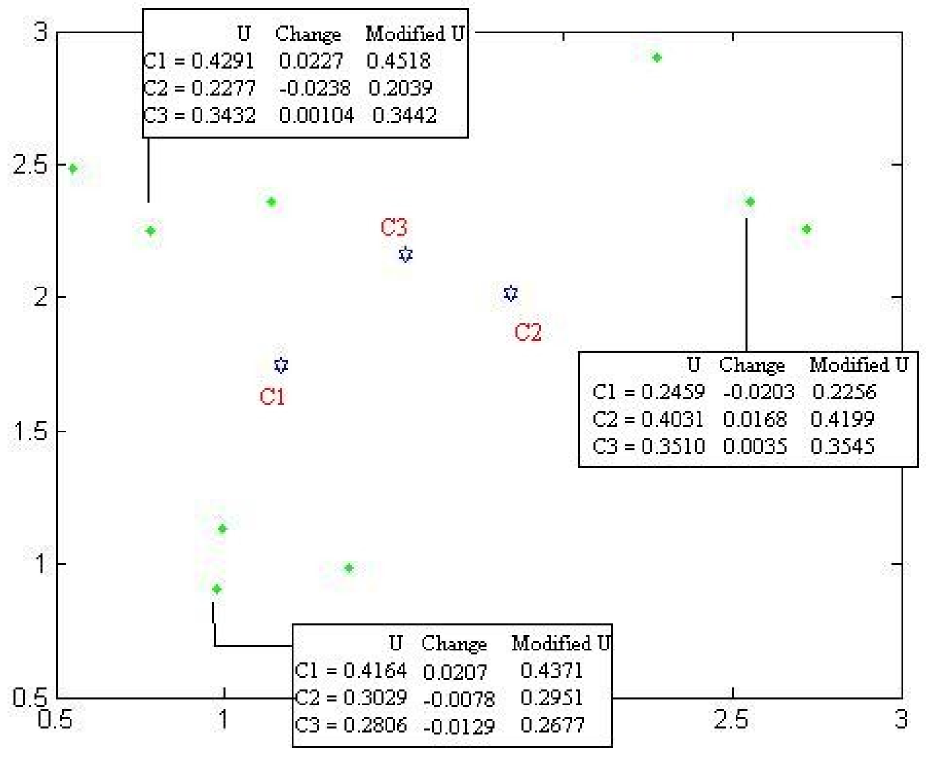


Figure 3.2 (a) Initial cluster centers position

Next three figures i.e. Figure 3.2(b), 3.2(c) and 3.2(d) shows the fuzzy membership values for all data points after 1st, 2nd and 3rd iteration respectively before biasing and normalized values after biasing. For a fuzzy membership matrix crisp values are desired i.e. the membership values should approach to one if it a part of that cluster or should be nearer zero in the case that the data point is not a part of that cluster. Looking into Figure 3.2 (b) we can see that the first data point seems to be a member of C1 or C3 but more likely C1. So we can notice that the change in values is according to the desire i.e. the value for C1 and C3 are increased. Increase in the value of C1 is more that that of C3. As the first data point is at extreme distance from C2 so sum of membership values added in C1 and C3 is the value subtracted from membership value of C2. Similar pattern is seen for all data points in Figures

3.2(c) and Figure 3.2(d). All the figures shows that fuzzy membership matrix values are biased as desired in an ideal fuzzy membership matrix U.

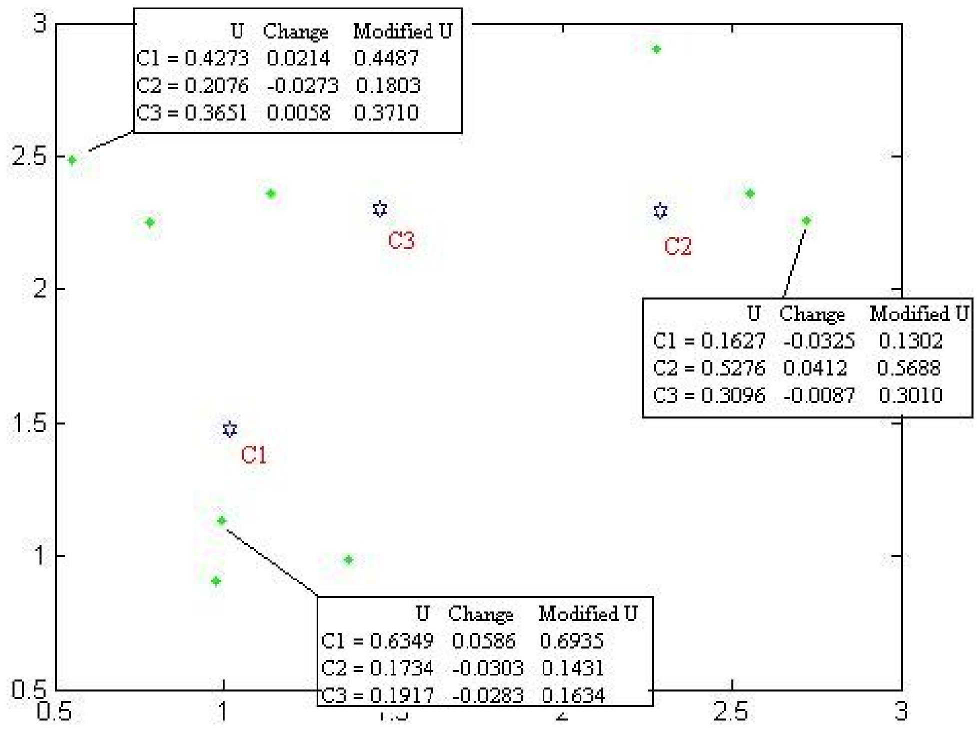


	U	Change	Modified U
C1 =	0.4291	0.0227	0.4518
C2 =	0.2277	-0.0238	0.2039
C3 =	0.3432	0.00104	0.3442

	U	Change	Modified U
C1 =	0.2459	-0.0203	0.2256
C2 =	0.4031	0.0168	0.4199
C3 =	0.3510	0.0035	0.3545

	U	Change	Modified U
C1 =	0.4164	0.0207	0.4371
C2 =	0.3029	-0.0078	0.2951
C3 =	0.2806	-0.0129	0.2677

Figure 3.2 (b) After Iteration 1 of modified FCM

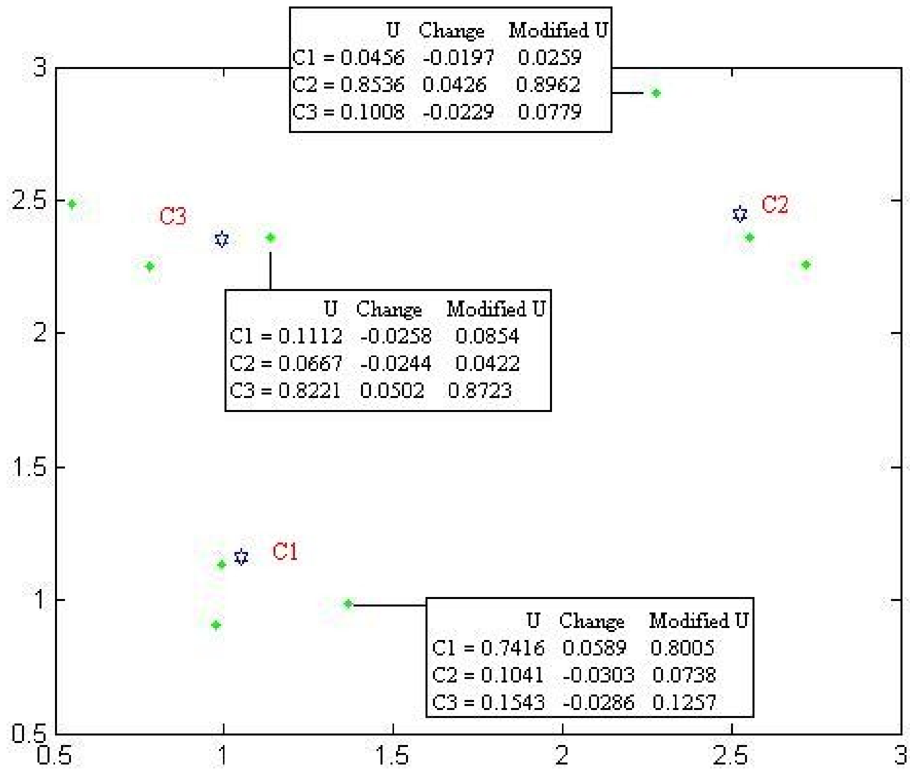


	U	Change	Modified U
C1 =	0.4273	0.0214	0.4487
C2 =	0.2076	-0.0273	0.1803
C3 =	0.3651	0.0058	0.3710

	U	Change	Modified U
C1 =	0.1627	-0.0325	0.1302
C2 =	0.5276	0.0412	0.5688
C3 =	0.3096	-0.0087	0.3010

	U	Change	Modified U
C1 =	0.6349	0.0586	0.6935
C2 =	0.1734	-0.0303	0.1431
C3 =	0.1917	-0.0283	0.1634

Figure 3.2 (c) After Iteration 2 of modified FCM

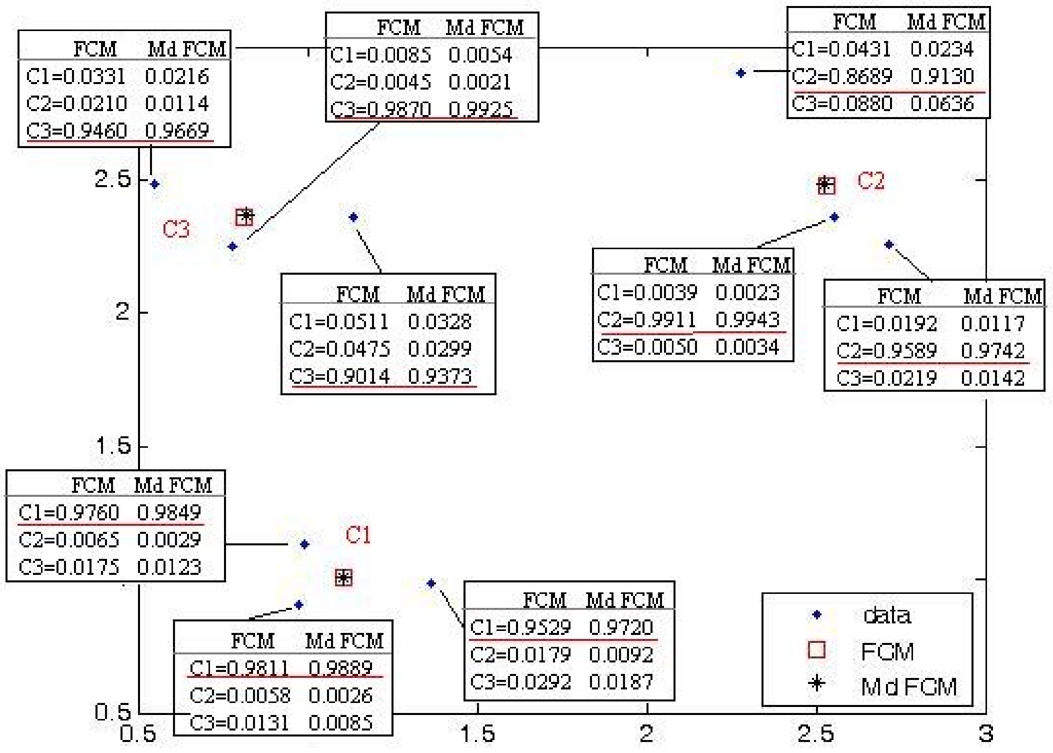


	U	Change	Modified U
C1 =	0.0456	-0.0197	0.0259
C2 =	0.8536	0.0426	0.8962
C3 =	0.1008	-0.0229	0.0779

	U	Change	Modified U
C1 =	0.1112	-0.0258	0.0854
C2 =	0.0667	-0.0244	0.0422
C3 =	0.8221	0.0502	0.8723

	U	Change	Modified U
C1 =	0.7416	0.0589	0.8005
C2 =	0.1041	-0.0303	0.0738
C3 =	0.1543	-0.0286	0.1257

Figure 3.2 (d) After Iteration 3 of modified FCM



FCM	Md FCM
C1=0.0331	0.0216
C2=0.0210	0.0114
C3=0.9460	0.9669

FCM	Md FCM
C1=0.0085	0.0054
C2=0.0045	0.0021
C3=0.9870	0.9925

FCM	Md FCM
C1=0.0431	0.0234
C2=0.8689	0.9130
C3=0.0880	0.0636

FCM	Md FCM
C1=0.0511	0.0328
C2=0.0475	0.0299
C3=0.9014	0.9373

FCM	Md FCM
C1=0.0039	0.0023
C2=0.9911	0.9943
C3=0.0050	0.0034

FCM	Md FCM
C1=0.0192	0.0117
C2=0.9589	0.9742
C3=0.0219	0.0142

FCM	Md FCM
C1=0.9760	0.9849
C2=0.0065	0.0029
C3=0.0175	0.0123

FCM	Md FCM
C1=0.9811	0.9889
C2=0.0058	0.0026
C3=0.0131	0.0085

FCM	Md FCM
C1=0.9529	0.9720
C2=0.0179	0.0092
C3=0.0292	0.0187

Figure 3.2 (e) Iteration 8 of modified FCM and Iteration 9 of FCM

In the last Figure 3.2(e) the final cluster centers position and fuzzy membership values of all data points for both FCM and modified FCM algorithms are shown. It is clear from final results that biased membership values calculated using modified FCM are more crisp and determined in less number of iterations as compared to traditional FCM.

3.6 Implementation in MATLAB

Clustering is a primary tool in data mining. It has been implemented as packages in many software, such SAS, S-PLUS, MATLAB, and other data analysis software. MATLAB (MATrix LABoratory) is an integrated environment that provides mathematical computing, visualization, and a powerful technical language. Using MATLAB to implement a clustering algorithm is convenient and easy. The MATLAB environment enables us to plot clustering results on the fly. Some clustering-related toolboxes for MATLAB have been developed, such as the SOM toolbox and the Netlab toolbox. To implement and analyze the results of the algorithm here MATLAB 7.1 is used as a tool.

For implementation in MATLAB the execution of algorithm is distributed in three parts.

First part contains the generation/loading of test data, calling of main algorithm and passing that test data as parameter and in the end presenting the results. For each experiment a separate version of this part is developed.

The second part is implemented as a module named ***fcmmodified*** containing the source code for overall algorithm. That is initialization of clusters, then a loop in which the third part/module is iteratively executed and terminated when the stopping criteria is met. Then the results obtained are returned to the first part.

The third part implemented as a module named ***fcmstepmodified*** containing the source code for iterative part of the algorithm. It determines the cluster centers, calculates the new normalized biased fuzzy membership matrix and calculates the value of objective function. And finally returns the results.

The MATLAB source code used is given here. It is self-explanatory by the comments given in the source code. Comments are indicated by the '%' sign describing the step of the algorithm implemented by that line of source code.

function [center, U, obj_fcn] = fcmmodified(data, cluster_n, options)

```
if nargin ~= 2 & nargin ~= 3,
   error('Too many or too few input arguments!');
end

% Change the following to set default options
default_options = [2;  % exponent for the partition matrix U
      100;  % max. number of iteration
      1e-5;  % min. amount of improvement in objective function
      1]; % info display during iteration

if nargin == 2,
   options = default_options;
else
   % If "options" is not fully specified, pad it with default values.
   if length(options) < 4,
      tmp = default_options;
      tmp(1:length(options)) = options;
      options = tmp;
   end
   % If some entries of "options" are nan's, replace them with defaults.
   nan_index = find(isnan(options)==1);
   options(nan_index) = default_options(nan_index);
   if options(1) <= 1,
      error('The exponent should be greater than 1!');
   end
end

expo = options(1);      % Exponent for U
max_iter = options(2);      % Max. iteration
min_impro = options(3);      % Min. improvement
display = options(4);       % Display info or not

obj_fcn = zeros(max_iter, 1);  % Array for objective function
U = initfcm(cluster_n, data_n);          % Initial fuzzy partition

% Main loop
for i = 1:max_iter,

   [U, center, obj_fcn(i)] = stepfcmmodified(data, U, cluster_n, expo);

   if display,
      fprintf('Iteration count = %d, obj. fcn = %f\n', i, obj_fcn(i));
   end

   if i > 1,
      if abs(obj_fcn(i) - obj_fcn(i-1)) < min_impro
         break; % Check stopping criteria if met then exit
      end
   end
end

iter_n = i; % Actual number of iterations
obj_fcn(iter_n+1:max_iter) = [];
```

function [U_new, center, obj_fcn] = stepfcmmodified(data, U, cluster_n, expo)

```
mf = U.^expo;       % MF matrix after exponential modification
center = mf*data./((ones(size(data, 2), 1)*sum(mf'))'); % new center

dist = distfcm(center, data);       % fill the distance matrix

distcenter = dist;
for a = 1:size(dist,1)
   temp_dist = dist;
   temp_dist(a,:) = [];
   distcenter(a,:) = mean(temp_dist);
end

obj_fcn = sum(sum((dist.^2).*mf));  % objective function
tmp = dist.^(-2/(expo-1));      % calculate new U, suppose expo != 1
U_new = tmp./(ones(cluster_n, 1)*sum(tmp));
%Bias Calculated U obtained by multiplying with Euclidian distance
U_new = U_new.*distcenter;
% Normalize Biased U
col_sum = sum(U_new);
U_new = U_new./col_sum(ones(cluster_n, 1), :);
```

CHAPTER 04: RESULTS AND DISCUSSIONS

To support the suggested work and confirm the resulting improvements, the algorithm is applied on real and synthetic datasets. In this chapter, first the criterion for evaluating the clustering results is described followed by three experiments that are presented to show the effectiveness of the proposed algorithm.

4.1 Evaluation of Clustering Results

Evaluation of clustering results sometimes referred to as cluster validation. There have been several suggestions for a measure of similarity between two clusters. Such a measure can be used to compare how well different data clustering algorithms perform on a set of data. These measures are usually tied to the type of criterion being considered in assessing the quality of a clustering method.

4.1.1 Internal Evaluation

When a clustering result is evaluated based on the data that was clustered itself, this is called internal evaluation. These methods usually assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters. One drawback of using internal criteria in cluster evaluation is that high scores on an internal measure do not necessarily result in effective information retrieval applications. Additionally, this evaluation is biased towards algorithms that use the same cluster model. For example k-Means clustering naturally optimizes object distances, and a distance-based internal criterion will likely overrate the resulting clustering.

A list of fifteen validation indices used to assess the quality of clustering algorithms based on internal criterion are presented in Table 4.1.

Table 4.1: Cluster Validity Indices

	Index	Definition	Optimal Value
1	Calinski-Harabasz (CH)[34]	$\frac{\sum_i n_i d^2(c_i,c)/(k-1)}{\sum_i \sum_{x \in C_i} d^2(x,c_i)/(n-k)}$	Max
2	Dunn's index[35]	$min_i \left\{ min_j \left(\frac{min_{x \in C_i, y \in C_j} d(x,y)}{max_l \left\{ max_{x,y \in C_l} d(x,y) \right\}} \right) \right\}$	Max
3	I index[36]	$\left(\frac{1}{k} . \frac{\sum_{x \in X} d(x,c)}{\sum_i \sum_{x \in C_i} d(x,c_i)} . max_{i,j} d(c_i,c_j) \right)^p$	Max
4	PCAES[37]	$\sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^2 / \left[min_{1 \le j \le k} \sum_{i=1}^{n} u_{ij}^2 \right] - \sum_{j=1}^{k} exp \left(-min_{k \ne i} \{ \lVert c_i - c_k \rVert^2 \} / \left[\frac{\sum_{l=1}^{k} \lVert c_l - \bar{c} \rVert^2}{k} \right] \right)$	Max
5	Partition Coefficient (PC)[24]	$\frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n} {u_{i,j}}^2$	Max
6	Silhouette[38]	$\frac{1}{k} \sum_i \left\{ \frac{1}{n_i} \sum_{x \in C_i} \frac{b(x)-a(x)}{\max[b(x),a(x)]} \right\},$ $a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \ne x} d(x,y),\ b(x) = min_{j, j \ne i} \left[\frac{1}{n_j} \sum_{y \in C_j} d(x,y) \right]$	Max
7	SD[39]	$Dis(k_{max}) Scat(k) + Dis(k),$ $Scat(k) = \frac{1}{k} \sum_i \left\lVert \frac{\sigma(C_i)}{\sigma(x)} \right\rVert,\ Dis(k) = \frac{max_{i,j} d(c_i,c_j)}{min_{i,j} d(c_i,c_j)} \sum_i \left(\sum_j d(c_i,c_j) \right)^{-1}$	Min
8	Davies-Bouldin (DB)[40]	$\frac{1}{k} \sum_i max_{j, j \ne i} \left\{ \left[\frac{1}{n_i} \sum_{x \in C_i} d(x,c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x,c_j) \right] / d(c_i,c_j) \right\}$	Min
9	Davies-Bouldin (DB*)[41]	$\frac{1}{k} \sum_i \frac{max_{j, j \ne i} \left\{ \frac{1}{n_i} \sum_{x \in C_i} d(x,c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x,c_j) \right\}}{min_{l, l \ne i} d(c_i,c_l)}$	Min
10	Separation Index (S)[42]	$\frac{\sum_{j=1}^{k} \sum_{i=1}^{n} (u_{ij})^2 \lVert x_i - c_j \rVert^2}{N min_{j,l} \lVert c_l - c_j \rVert^2}$	Min
11	Partition Index (SC)[42]	$\sum_{j=1}^{k} \frac{\sum_{i=1}^{n} (u_{ij})^m \lVert x_i - c_j \rVert^2}{N_j \sum_{l=1}^{k} \lVert c_l - c_j \rVert^2}$	Min
12	Xie-Beni (XB)[43]	$\left[\sum_i \sum_{x \in C_i} d^2(x,c_i) \right] / \left[n . min_{i, j \ne i} d^2(c_i,c_j) \right]$	Min
13	Xie-Beni (XB*)[41]	$\left[max_{1 \le i \le k} \left\{ \sum_{x \in C_i} d^2(x,c_i) \right\} \right] / \left[min_{i, j \ne i} d^2(c_i,c_j) \right]$	Min
14	Coefficient Entropy (CE)[44]	$\frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n} u_{i,j} \log_a (u_{i,j})$	Min
15	SDbw[45]	$Scat(k) + \frac{1}{k(k-1)} \sum_i \left[\sum_{j, j \ne i} \frac{\sum_{x \in C_i U C_j} f(x,u_{ij})}{max \left\{ \sum_{x \in C_i} f(x,c_i), \sum_{x \in C_j} f(x,c_j) \right\}} \right]$	Min

4.1.2 External Evaluation

In external evaluation, clustering results are evaluated based on data that was not used for clustering, such as known class labels and external benchmarks. Such benchmarks consist of a set of pre-classified items, and these sets are often created by humans (experts). Thus, the benchmark sets can be thought of as a gold standard for evaluation. These types of evaluation methods measure how close the clustering is to the predetermined benchmark classes. However, it has recently been discussed whether this is adequate for real data, or only synthetic data sets with a factual ground truth, since classes can contain internal structure, the attributes present may not allow separation of clusters or the classes may contain anomalies. Additionally, from a knowledge discovery point of view, the reproduction of known knowledge may not necessarily be the intended result.

Some of the measures of quality of a cluster algorithm using external criterion include:

4.1.2.1 Rand measure (William M. Rand 1971)

The Rand index computes how similar the clusters (returned by the clustering algorithm) are to the benchmark classifications. One can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

4.1.2.2 Adjusted Rand measure

One issue with the Rand index is that false positives and false negatives are equally weighted. This may be an undesirable characteristic for some clustering applications. This issue is addressed in adjusted rand index

$$ARI = \frac{Index - Expected\ Index}{Max\ Index - Expected\ Index}$$

4.2 Experiment No. 1

In this experiment, a dataset containing 1320 synthetic data points from a mixture of four Gaussian distribution having uniform densities given below is tested against changing variance from Σ to 10 Σ. The experiment is repeated 100 times for all variance values and average values are taken to plot the results. Figure 4.1(a) shows an overview of synthetic data with variance Σ = 1data here is compact and there is no overlap among cluster data. While in Figure 4.1(b) data with variance Σ = 10 is shown having sparse and overlapped clusters. So as the variance of data is increased from Σ to 10 Σ the clusters get sparse and overlapped from compact and well separated clusters.

$$0.33\ Guassian\ \left[\begin{pmatrix}1.0\\1.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right] + 0.33\ Guassian\ \left[\begin{pmatrix}2.5\\2.5\end{pmatrix},\begin{pmatrix}0.02 & 0.0\\0.0 & 0.02\end{pmatrix}\right] +$$
$$0.33\ Guassian\ \left[\begin{pmatrix}2.0\\4.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right] + 0.33\ Guassian\ \left[\begin{pmatrix}4.0\\2.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right]$$

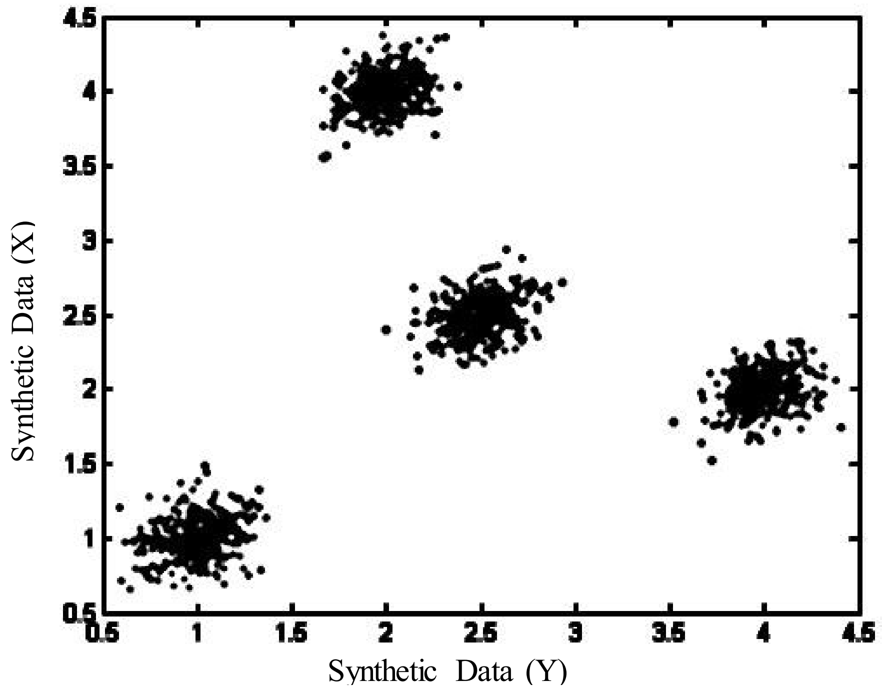


Figure 4.1(a) Outlook of data having uniform density clusters having variance Σ = 1

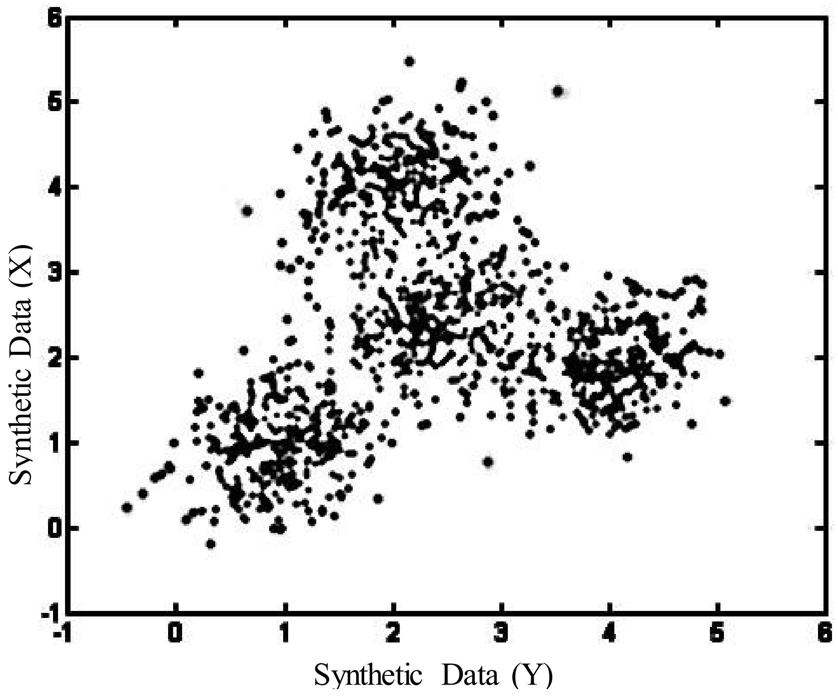


Figure 4.1(b) Outlook of data having uniform density clusters having variance Σ = 10

The results obtained are shown in Figure 4.2 (a) to 4.2 (q). Figure 4.2 (a) and 4.2 (b) shows the external validity indices Rand Index and Adjusted Rand Index. The result accuracies of both algorithms FCM and modified FCM are almost the same for changing variance of data.

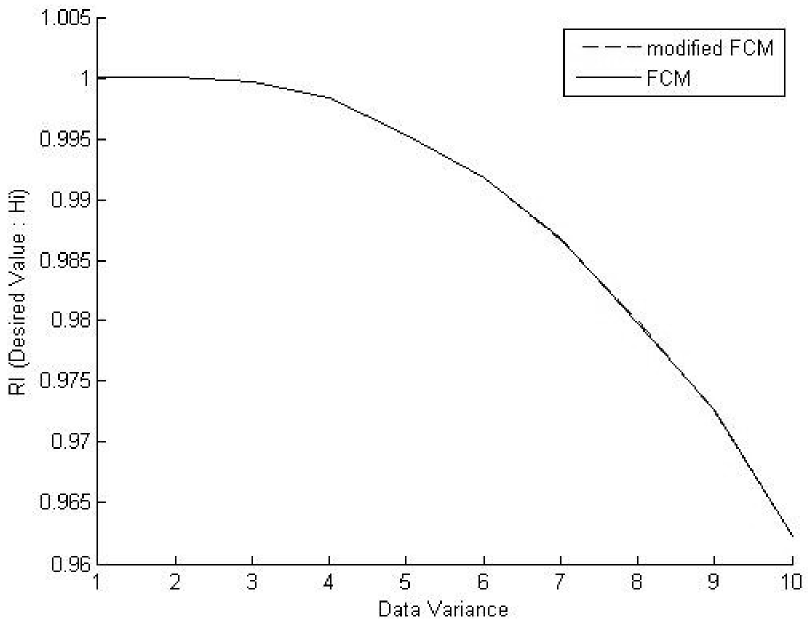


Figure 4.2 (a): Rand index vs Variance of Data

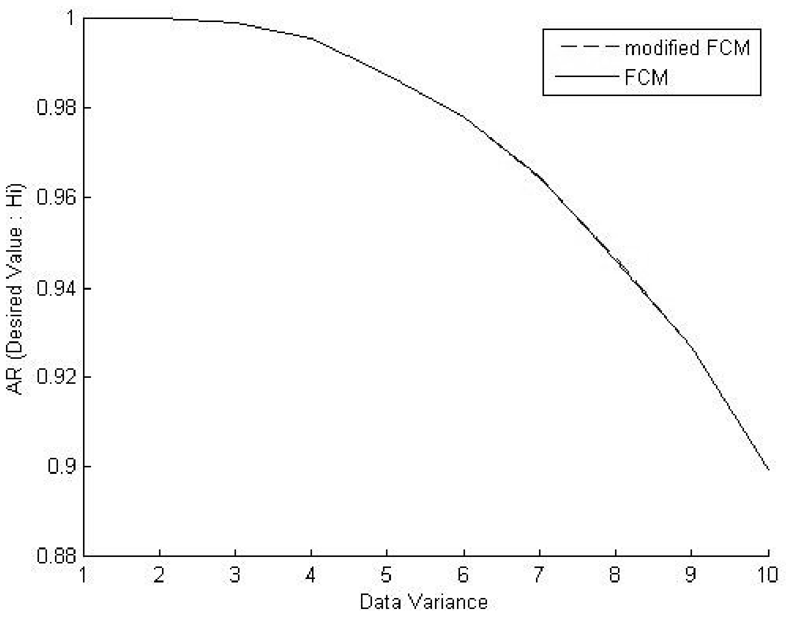


Figure 4.2 (b): Adjusted Rand Index vs Variance of Data

Next we observe a set of six internal validity indices shown in Figure 4.2 (c) to Figure 4.2 (h). For these six indices i.e. CH, DB, DB*, Dunn's, Silhoutte and SDbw results obtained show no improvement by either of the algorithm over other. This may be due to the information used in calculating the validity index. All these indices use either distance from data points to its cluster center or distance from one cluster center to other cluster center, and in some indices both distances, to measure the suitability of clustering result. While no information from fuzzy membership matrix is considered for calculating the index value.

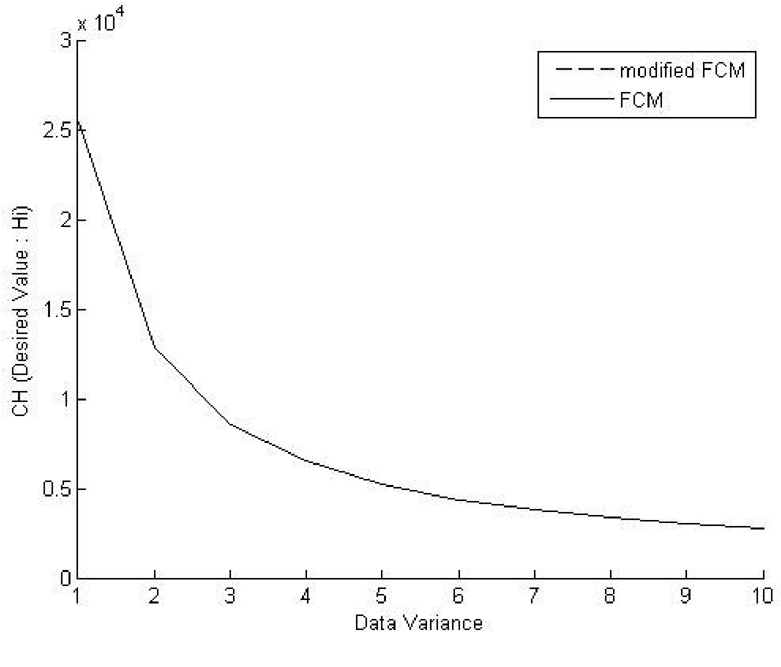


Figure 4.2 (c): CH vs Variance of Data

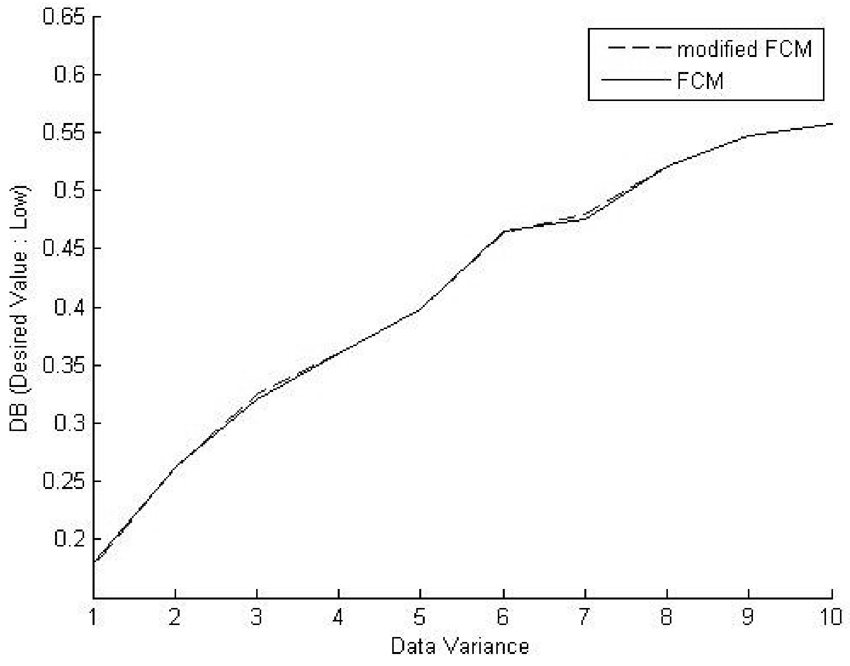


Figure 4.2 (d): DB vs Variance of Data

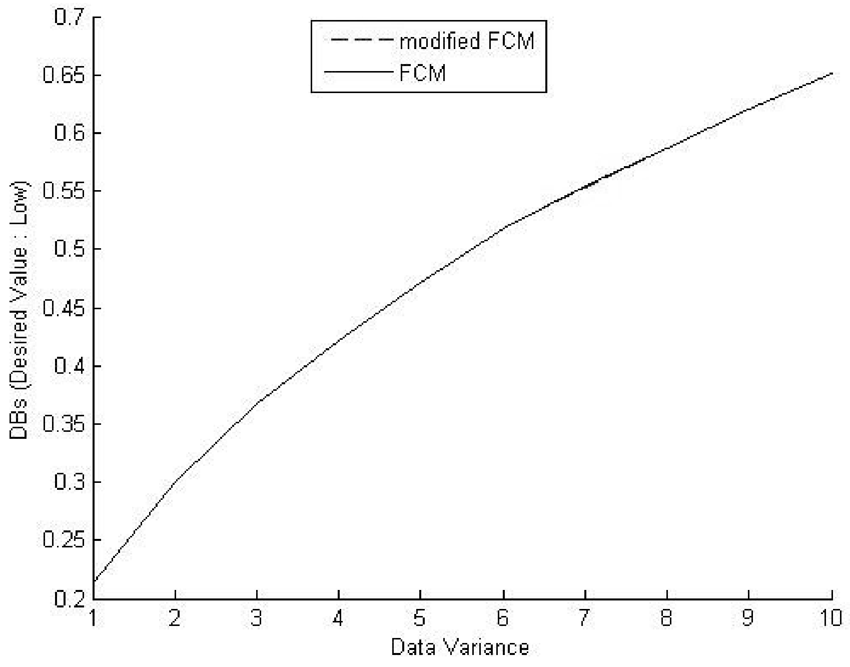


Figure 4.2 (e): DB* vs Variance of Data

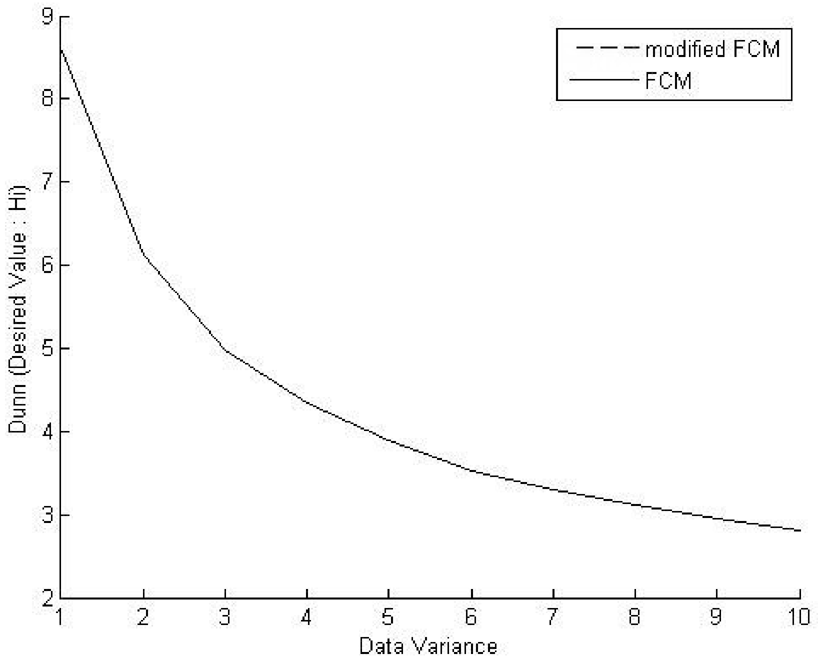


Figure 4.2 (f): Dunn’s vs Variance of Data

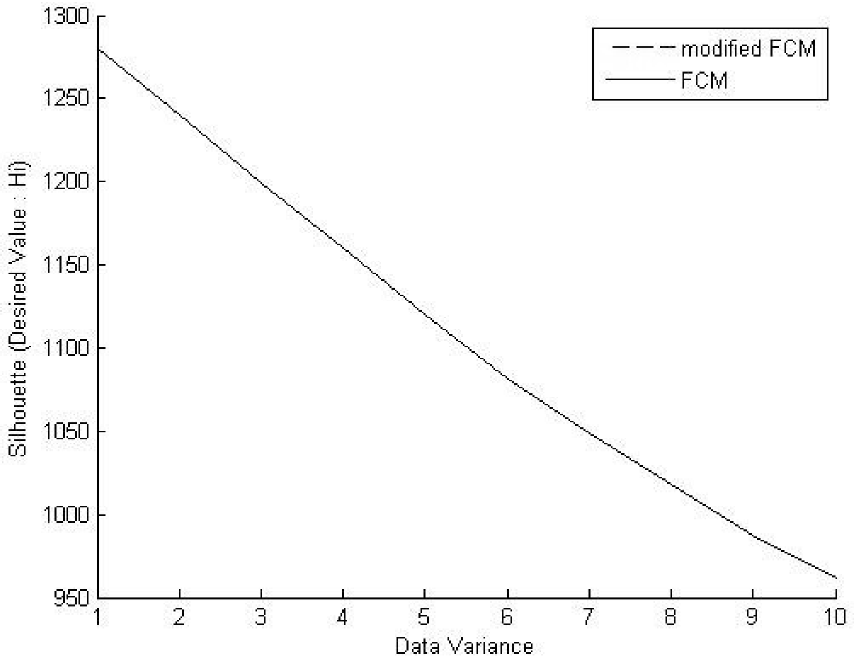


Figure 4.2 (g): Silhouette vs Variance of Data

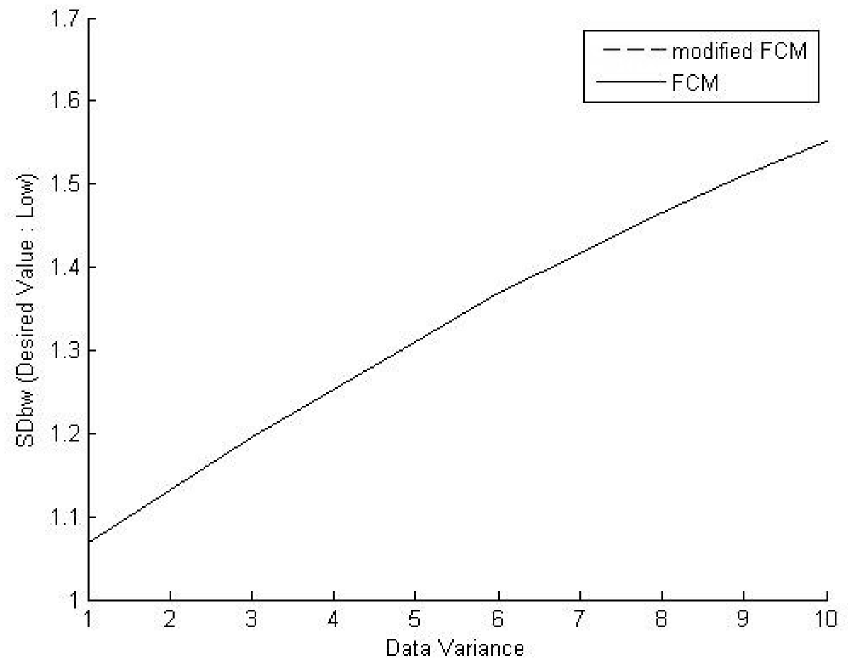


Figure 4.2 (h): SDbw vs Variance of Data

Now a group of seven internal validity indices are presented in Figure 4.2 (i) to Figure 4.2 (o). These seven indices shows a rich improvement in finding crisp clustering results by modified FCM in comparison with traditional FCM. The calculation of first two indices PC and CE, shown in Figure 4.2 (i) and Figure 4.2 (j), is purely based on fuzzy membership matrix U. The proposed modification is also purely based on calculating a desired biased U. Hence a clear improvement is shown by modified FCM over FCM using these indices. Three more indices S, SC and PCAES consider data to center and center to center distances as well as fuzzy membership U for their calculation. So a significant improvement got by modified FCM using S, SC and PCAES indices can be seen in Figure 4.2 (m), Figure 4.2 (n) and Figure 4.2 (l) respectively. As I and SD indices uses distances and variance for confirming clustering results. So the edge got by modified FCM using these two indices is slightly less than the edge got using above five indices. Observing the results obtained for the above seven indices we can see that the distance between the values of validity index for FCM and modified FCM is increased as the data variance is increased. This tendency of graph exhibit that with increase in the variance of data, accuracy obtained in finding better clustering outcome is increased more by modified FCM over FCM.

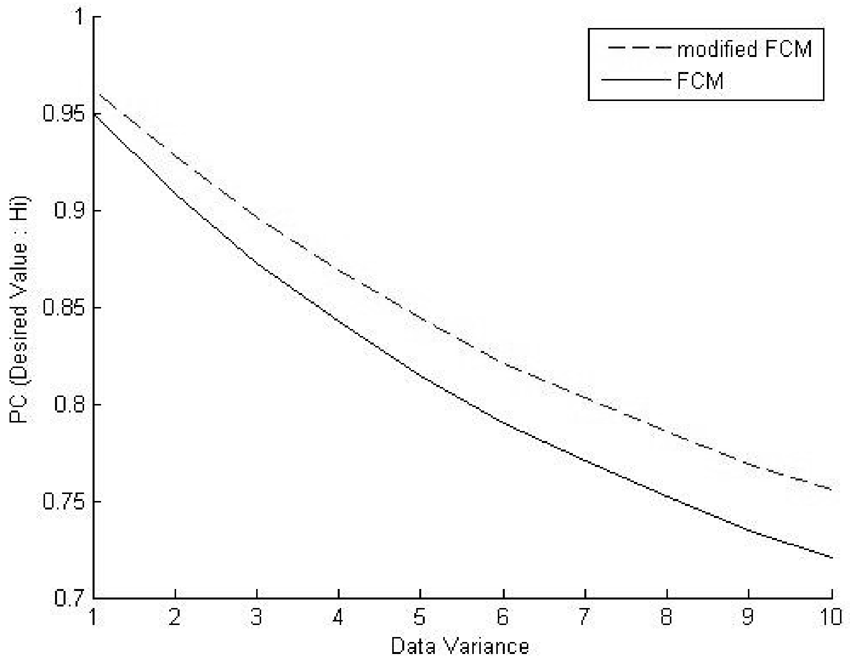


Figure 4.2 (i): PC vs Variance of Data

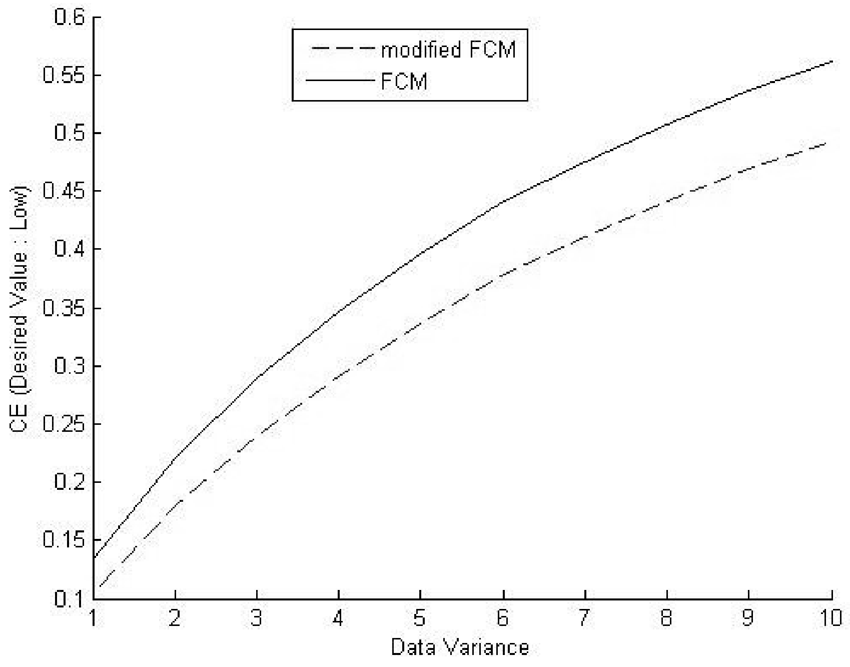


Figure 4.2 (j): CE vs Variance of Data

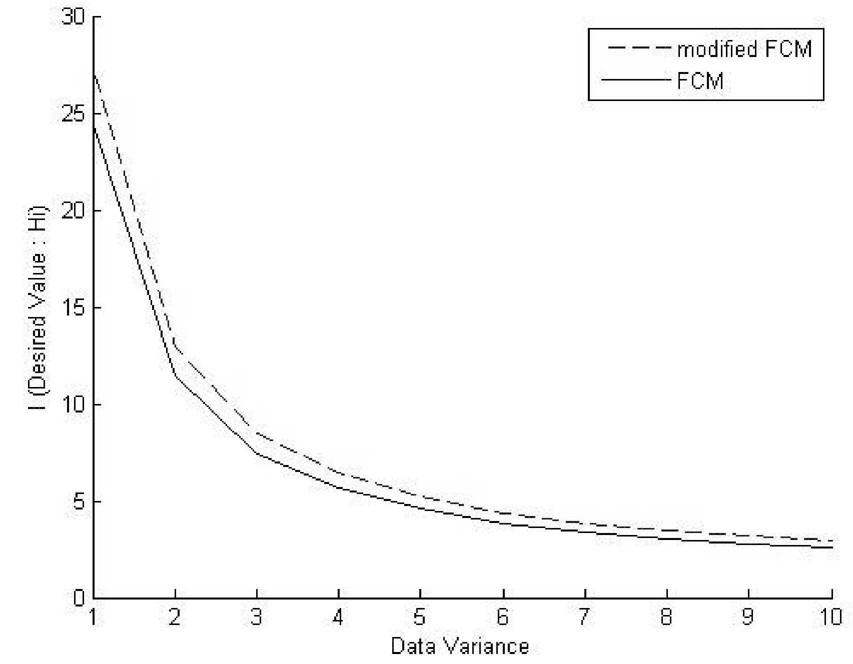


Figure 4.2 (k): I vs Variance of Data

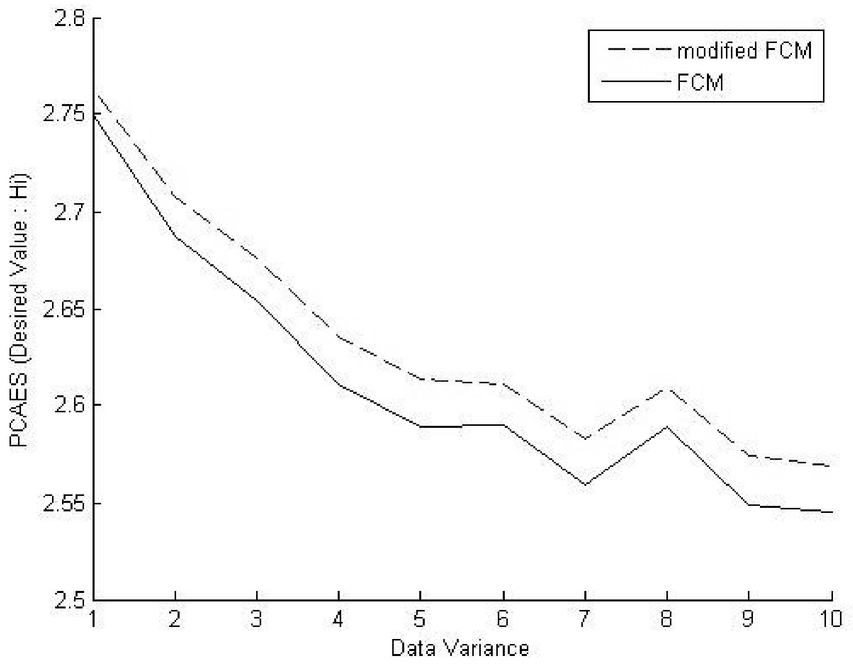


Figure 4.2 (l): PCAES vs Variance of Data

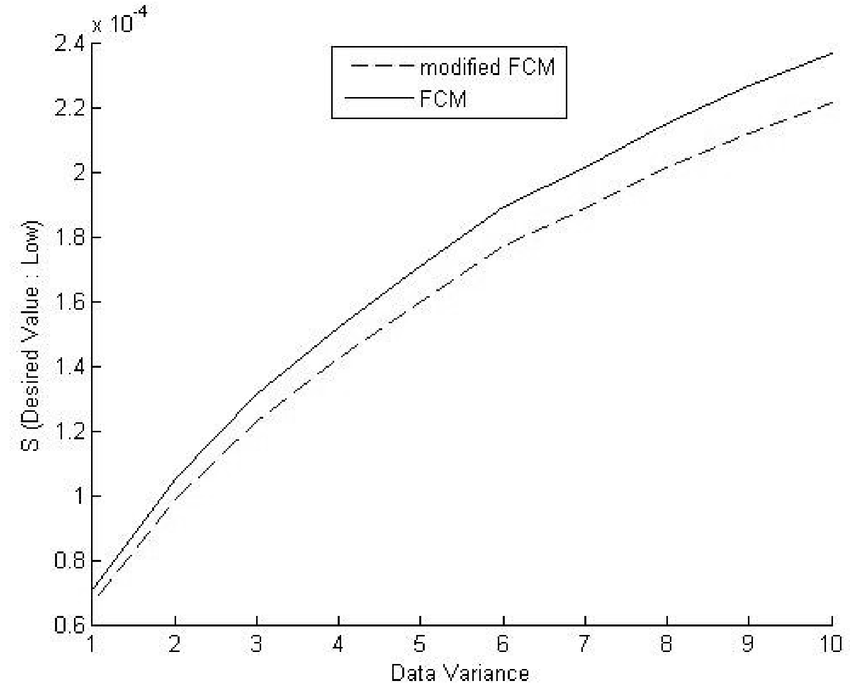


Figure 4.2 (m): S vs Variance of Data

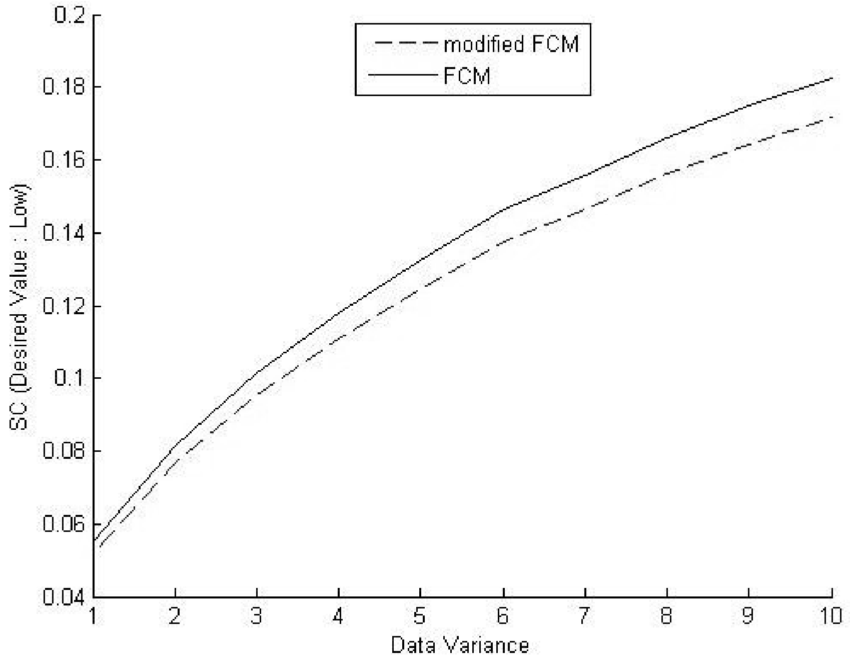


Figure 4.2 (n): SC vs Variance of Data

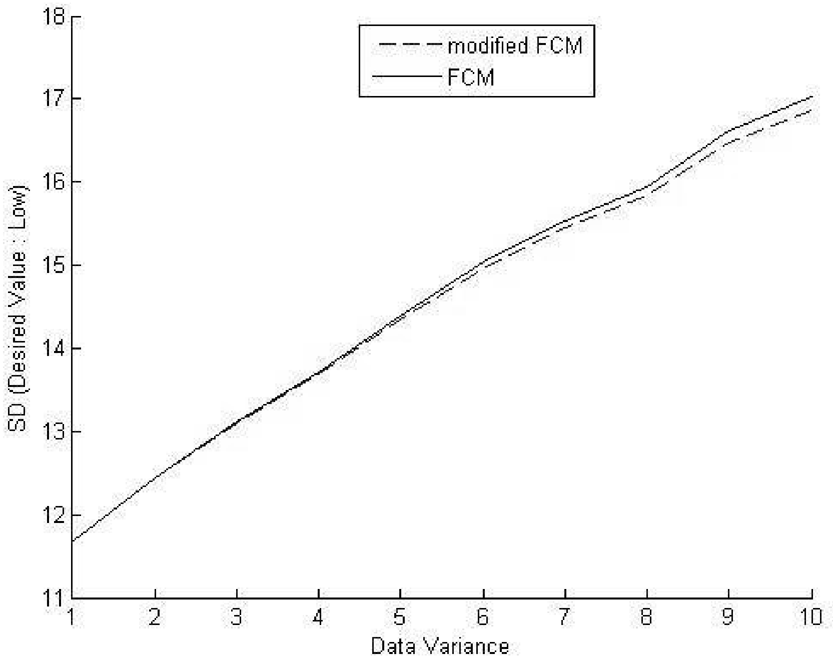


Figure 4.2 (o): SD vs Variance of Data

And in the last two of the validity indices XB and XB* shown in Figure 4.2 (p) and Figure 4.2 (q) indicates the degradation in the performance of modified FCM. Reason behind this is that these two indices do not consider fuzzy membership values in calculation. Also because these indices use data to center distance and only minimum center to center distance for index calculation.

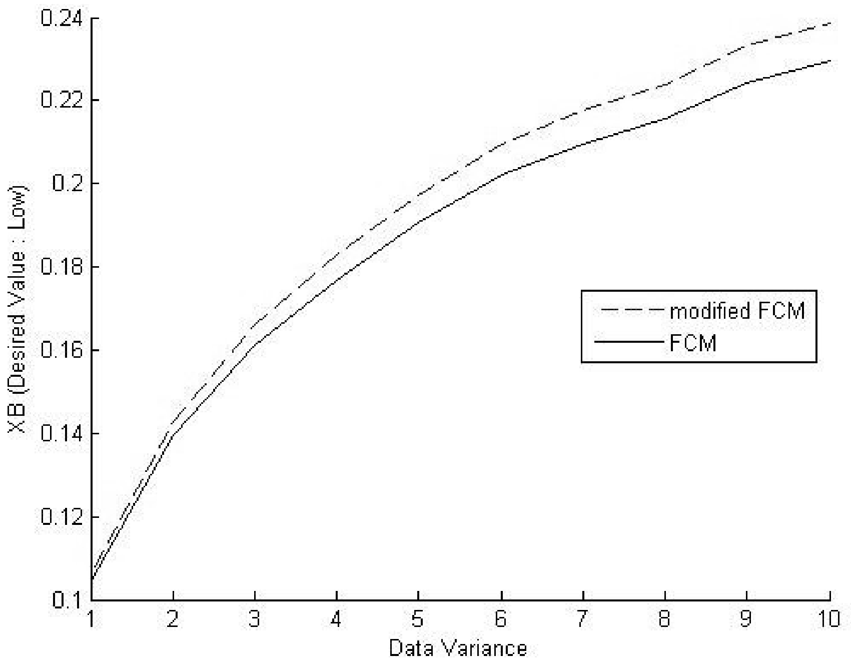


Figure 4.2 (p): XB vs Variance of Data

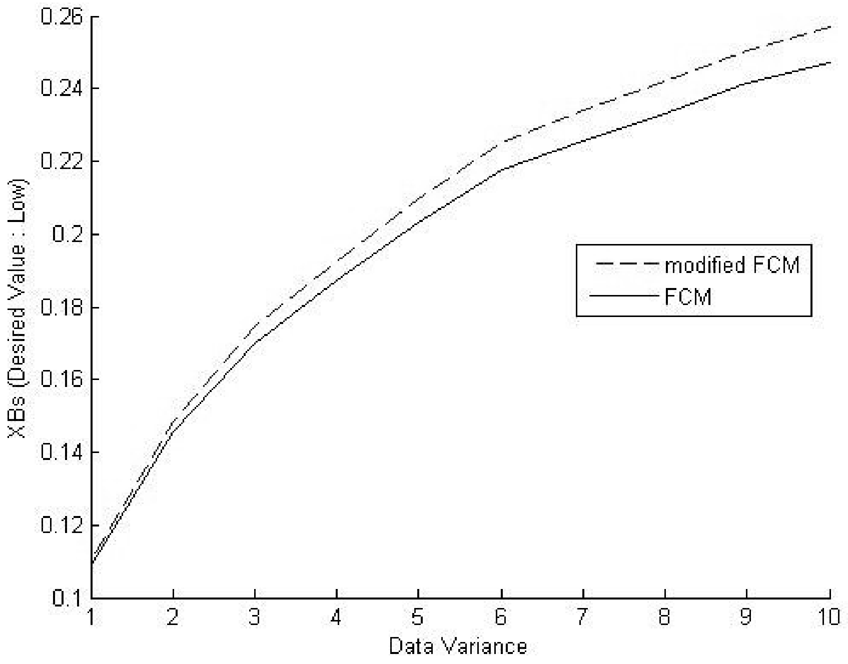


Figure 4.2 (q): XB* vs Variance of Data

4.3 Experiment No. 2

Here the first experiment is repeated with a dataset of 1100 synthetic data points having multi density clusters given by

$$0.11\ Guassian\ \left[\begin{pmatrix}1.0\\1.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right] + 0.22\ Guassian\ \left[\begin{pmatrix}2.5\\2.5\end{pmatrix},\begin{pmatrix}0.02 & 0.0\\0.0 & 0.02\end{pmatrix}\right] +$$

$$0.33\ Guassian\ \left[\begin{pmatrix}2.0\\4.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right] + 0.44\ Guassian\ \left[\begin{pmatrix}4.0\\2.0\end{pmatrix},\begin{pmatrix}0.02 & 0.005\\0.005 & 0.02\end{pmatrix}\right]$$

The outlook of data is shown in Figure 4.3 (a) having variance $\Sigma = 1$ and Figure 4.3 (b) having variance $\Sigma = 10$. It is clear from figures that data spread changes from compact to sparse and clusters getting more overlapped with increase in variance. The results from the experiment are shown in Figure 4.4 (a) to Figure 4.4(q).

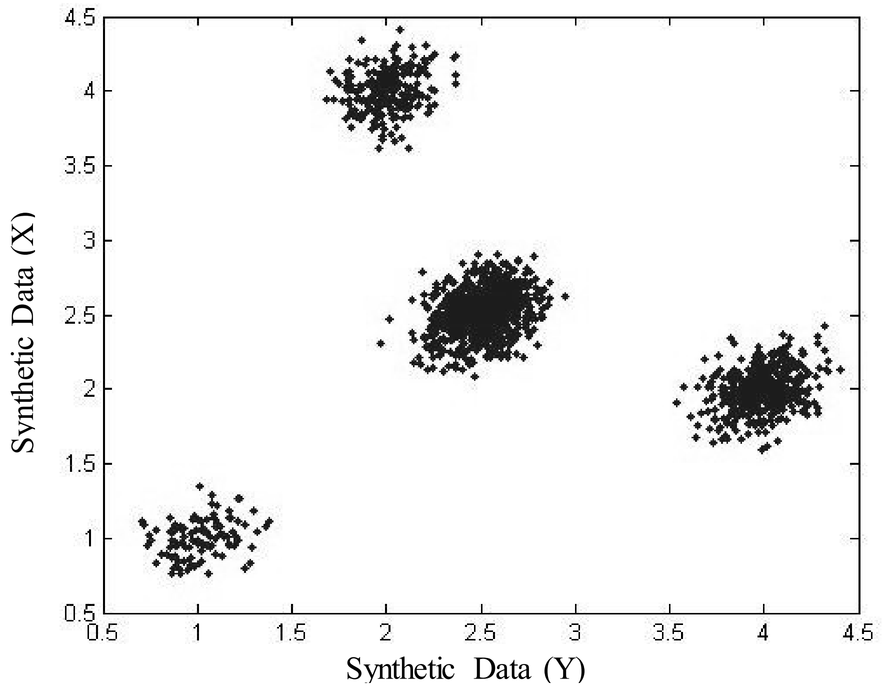


Figure 4.3(a) Outlook of data having multi density clusters with variance Σ = 1

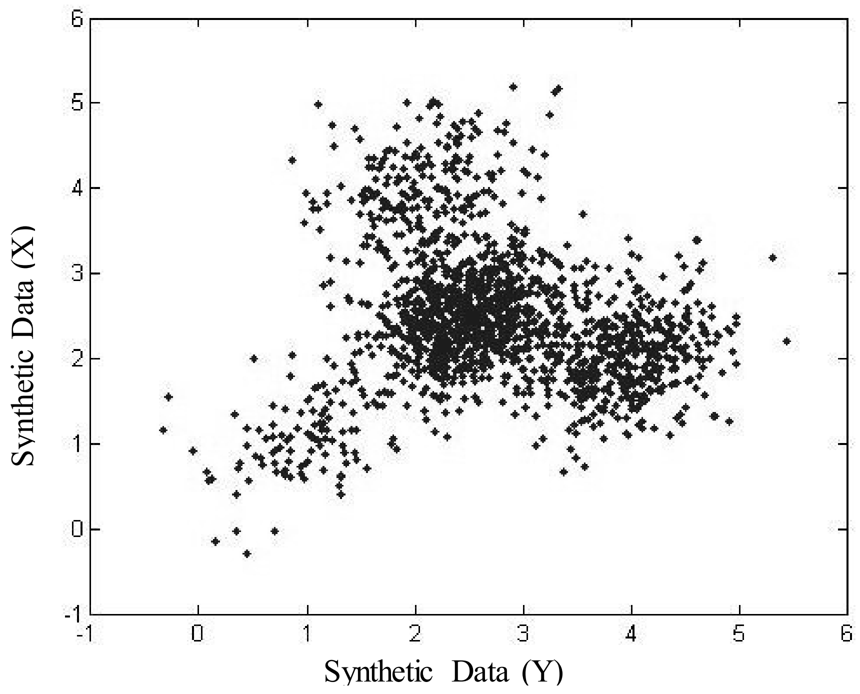


Figure 4.3(b) Outlook of data having multi density clusters with variance Σ = 10

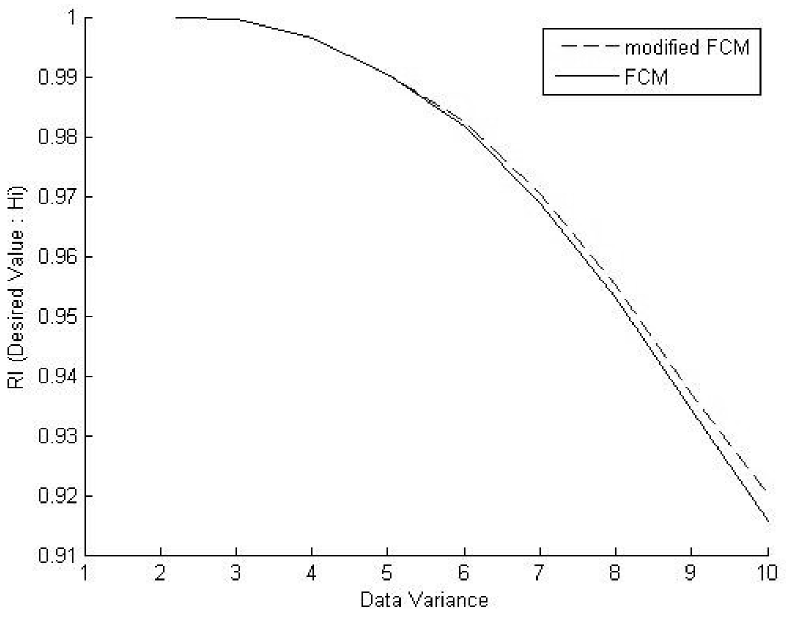


Figure 4.4 (a): Rand index vs Variance of Data

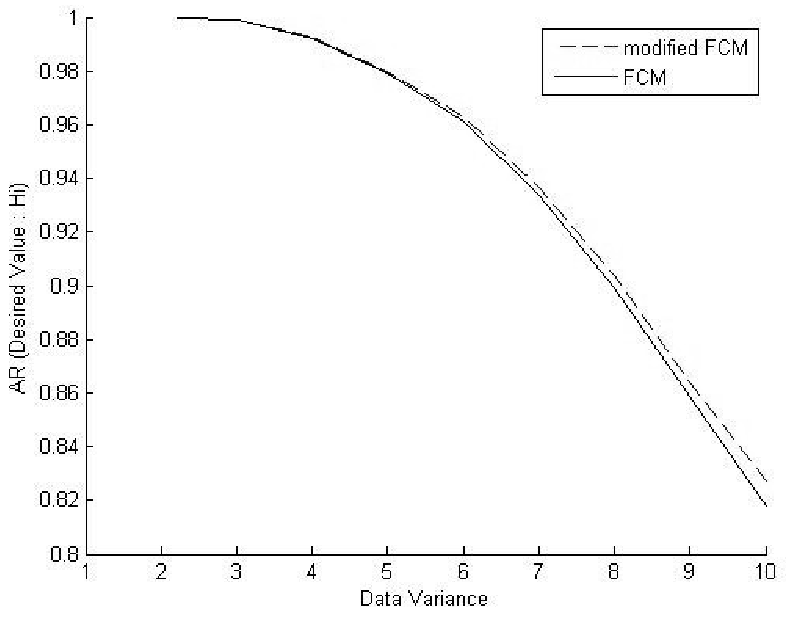


Figure 4.4 (b): Adjusted Rand Index vs Variance of Data

The above two figures i.e. Figure 4.4 (a) and Figure 4.4 (b) shows the results of external validity indices Rand Index and Adjusted Rand Index. In case of multi density clusters the a slight improvement is achieved, as the data get sparse, in the accuracy of labeling data points in their original clusters by modified FCM. When the clusters are well separated and compact accuracy of both algorithms is equal and is 100%. As soon as the clusters get sparse and overlapped the accuracy in overall gets down. But accuracy of modified FCM gets prominent in comparison to FCM with sparse and overlapped clusters.

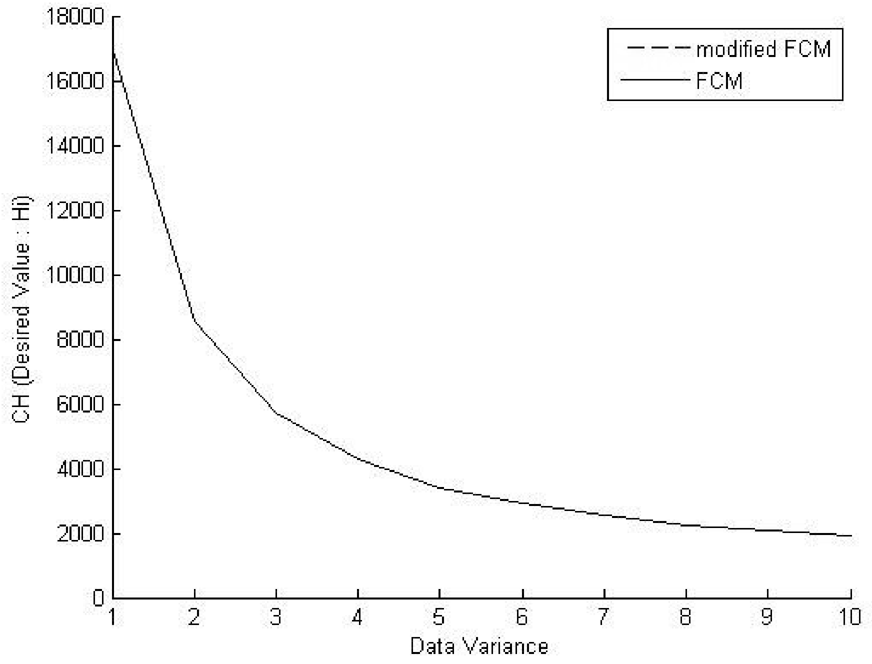


Figure 4.4 (c): CH vs Variance of Data

Out of fifteen internal validity indices only one index CH showed no improvement by either of the algorithm in the above Figure 4.4 (c).

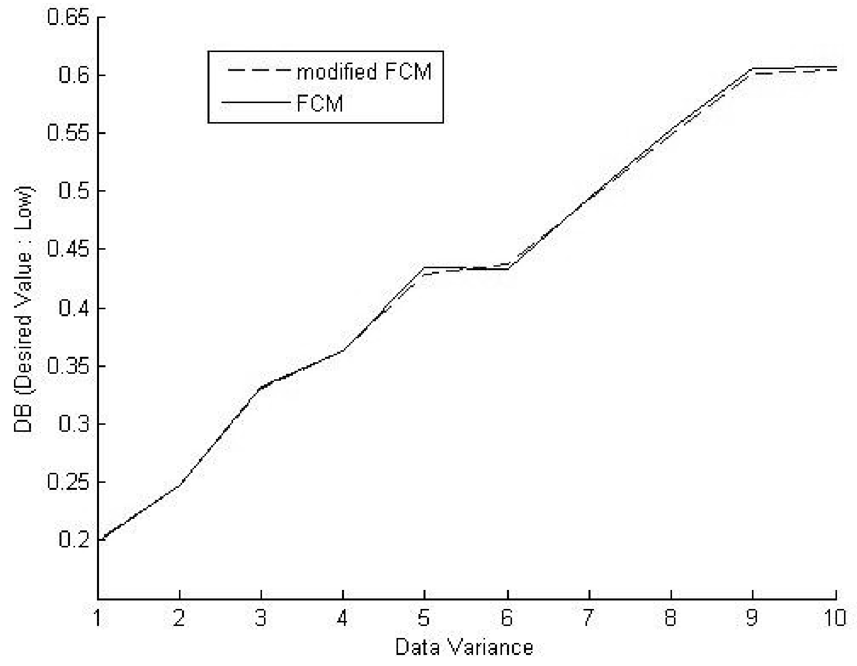


Figure 4.4 (d): DB vs Variance of Data

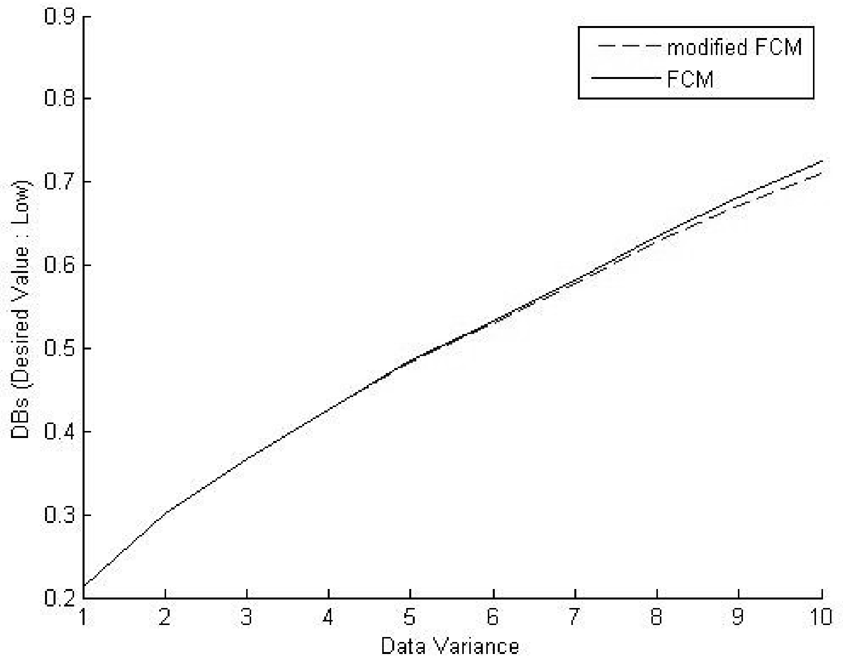


Figure 4.4 (e): DB* vs Variance of Data

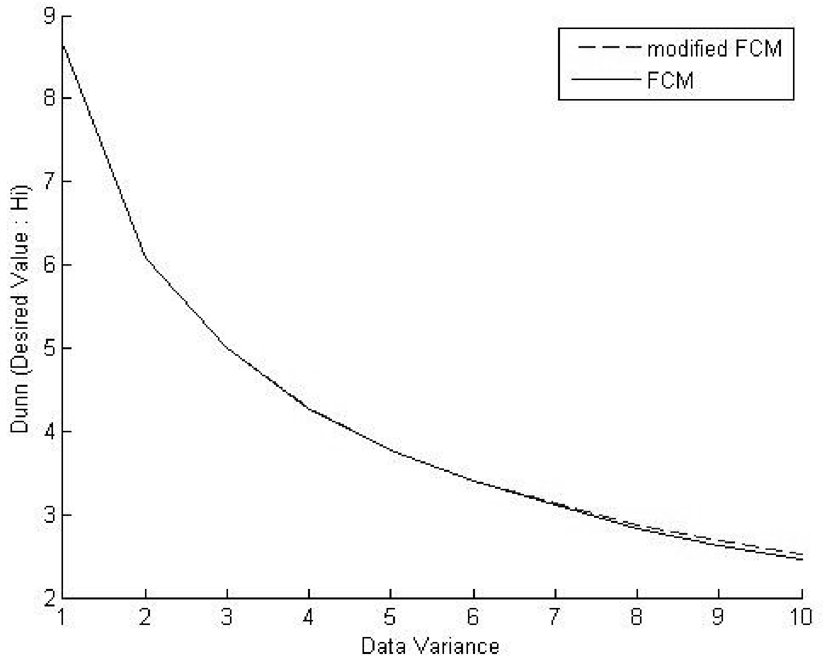


Figure 4.4 (f): Dunn's vs Variance of Data

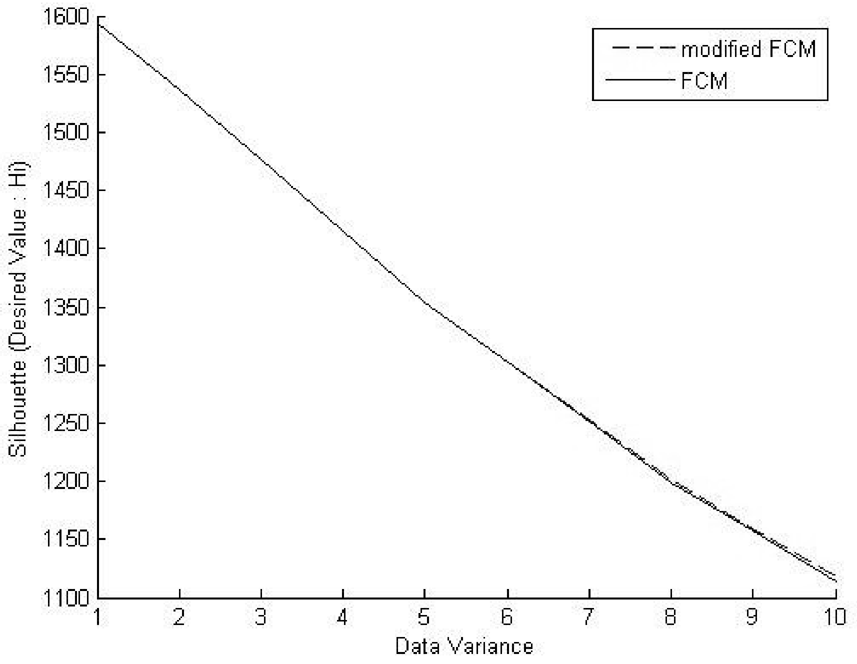


Figure 4.4 (g): Silhouette vs Variance of Data

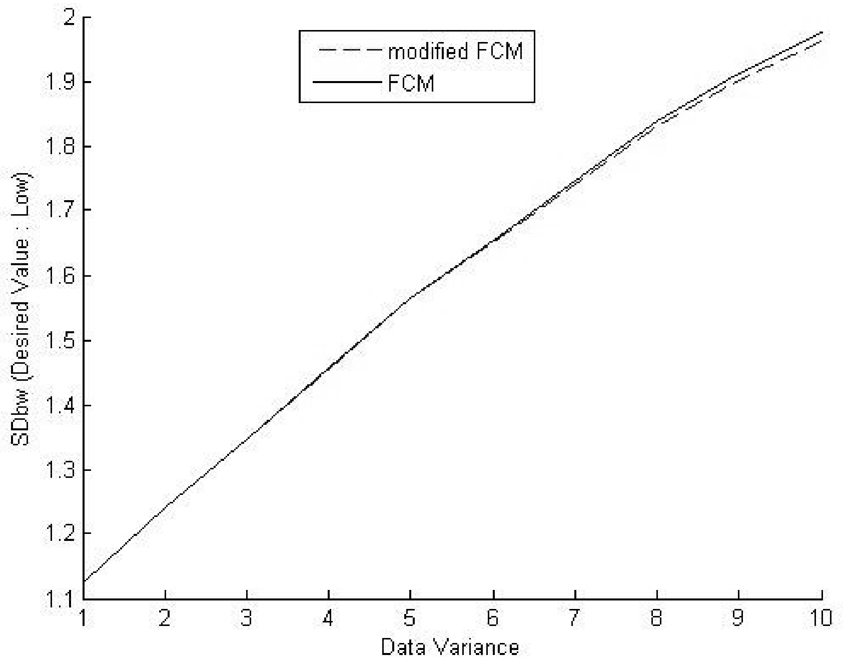


Figure 4.4 (h): SDbw vs Variance of Data

Above in Figure 4.4 (d) to Figure 4.4 (h) we present a group of five validity indices showing similar improvement achieved by modified FCM over FCM. For all the five validity indices i.e. DB, DB*, Dunn's, Silhouette and SDbw a slight improvement is obtained by modified FCM after $\sum$ get greater than six times. In case of uniform density clusters the same validity indices showed no improvement of modified FCM over FCM. This determines that for multi density clusters the performance is increased further.

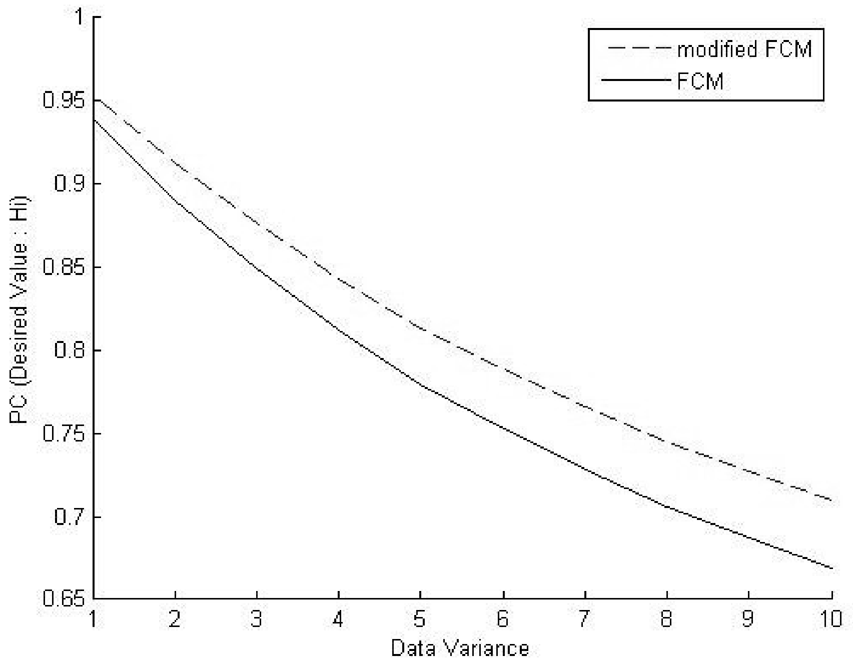


Figure 4.4 (i): PC vs Variance of Data

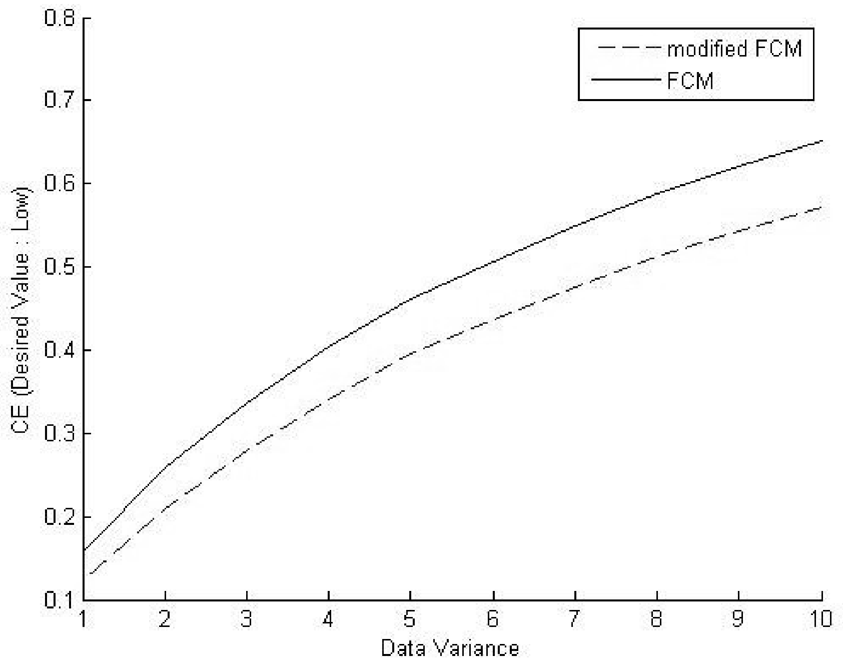


Figure 4.4 (j): CE vs Variance of Data

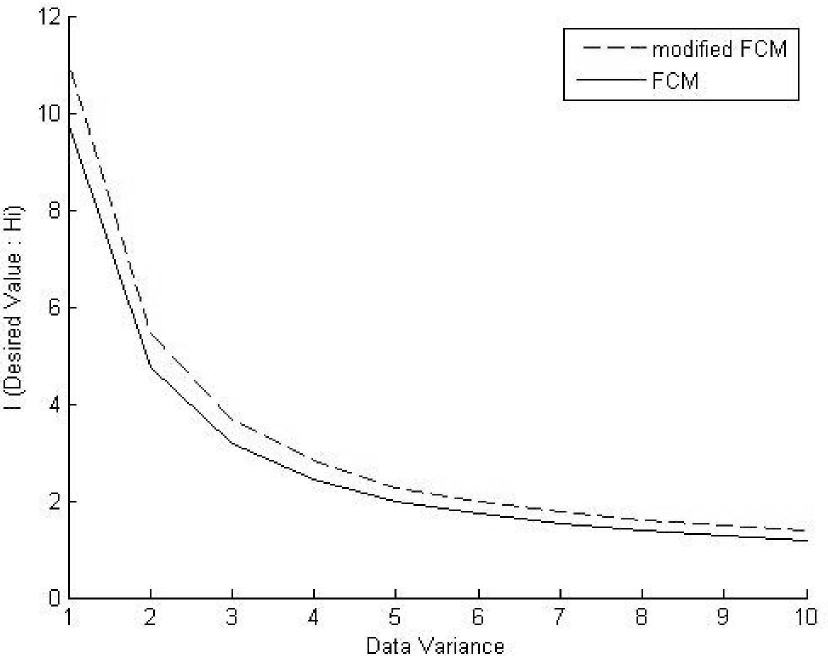


Figure 4.4 (k): I vs Variance of Data

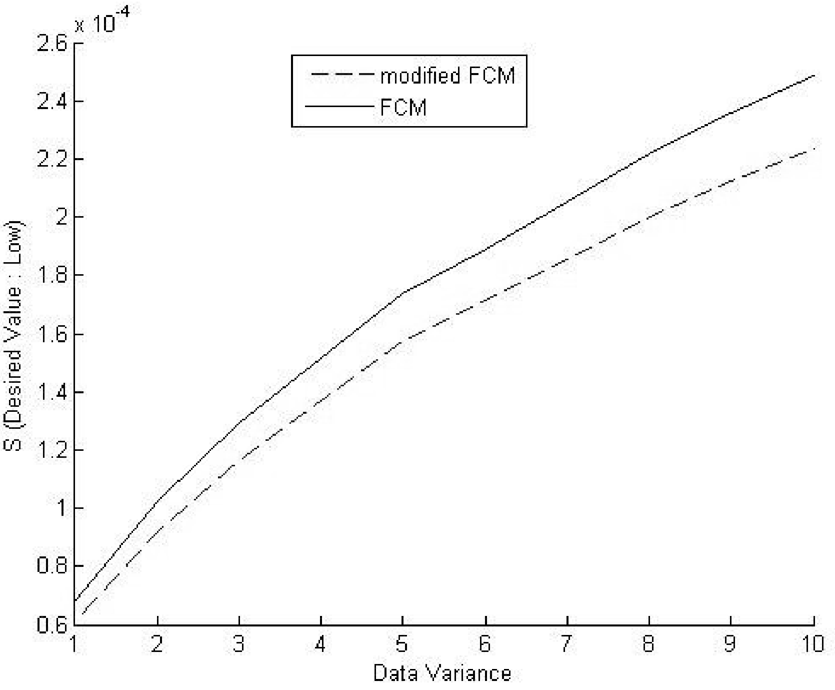


Figure 4.4 (l): S vs Variance of Data

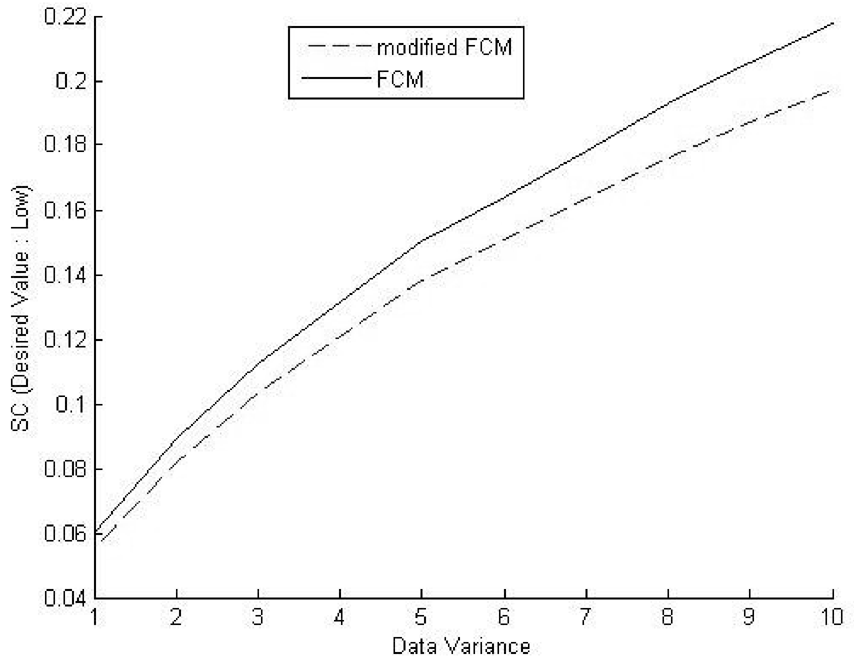


Figure 4.4 (m): SC vs Variance of Data

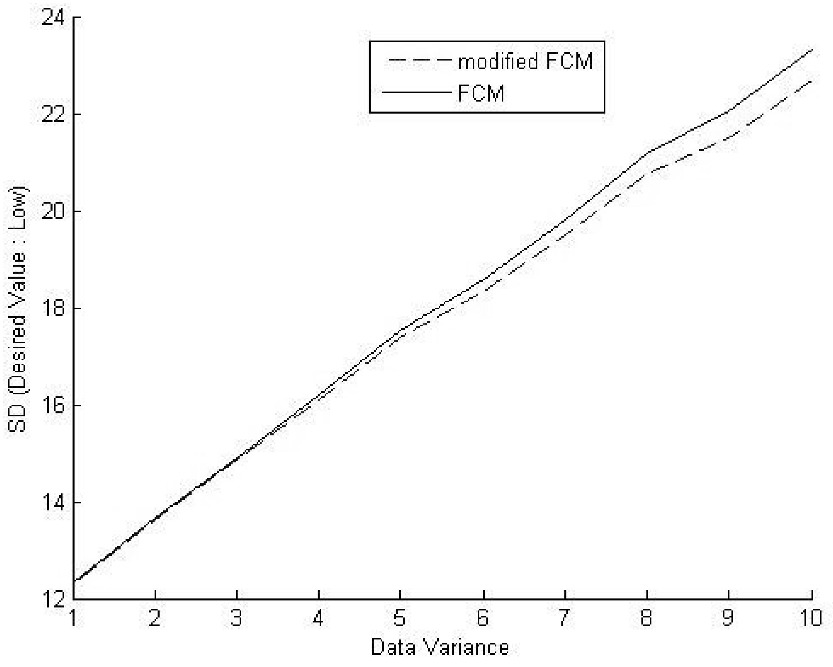


Figure 4.4 (n): SD vs Variance of Data

In the above six validity indices four i.e. PC, CE, S and SC shown in Figure 4.4 (i), Figure 4.4 (j), Figure 4.4 (l) and Figure 4.4 (m) clearly indicates a better performance of modified FCM. While two indices I and SD shown in Figure 4.4 (k) and Figure 4.4 (n) also shows the better performance but less than the other four indices. All six indices also confirm further increased improvement attained by modified FCM in case of multi density clusters as compared to the case of uniform density clusters in experiment no.1.

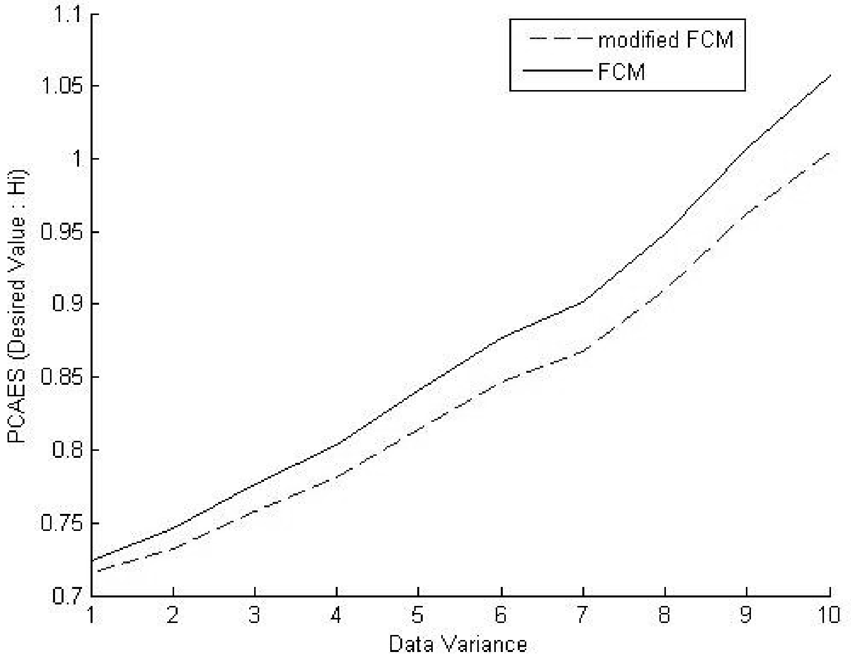


Figure 4.4 (o): PCAES vs Variance of Data

PCAES in Figure 4.4 (o) indicates the degradation in the performance of modified FCM. This is due to the failure of PCAES to measure the fitness of clustering results in case of multi density clusters. In contrast to all other validity indices PCAES indicates improvement in the accuracy of results for both algorithms as the clusters gets sparse and overlapped.

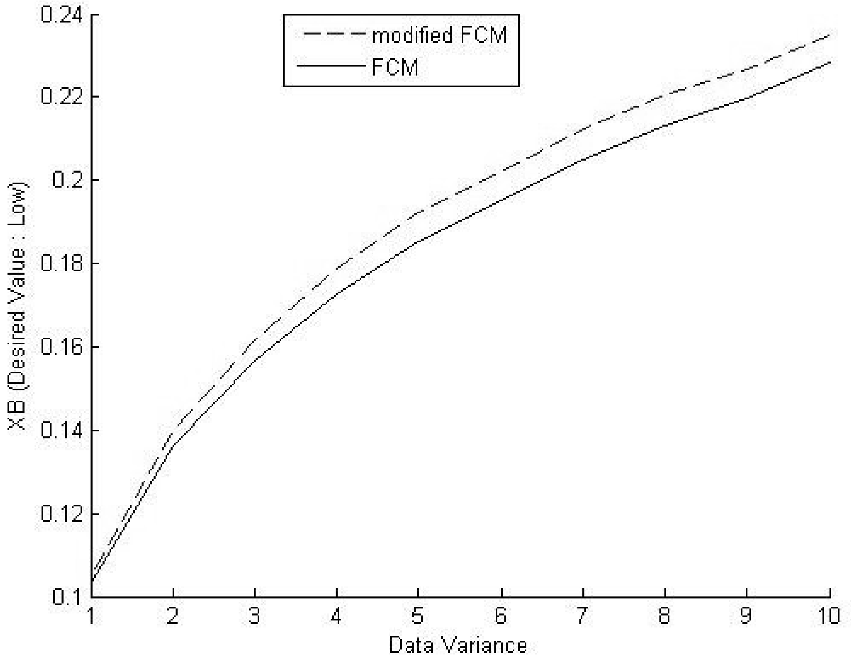


Figure 4.4 (p): XB vs Variance of Data

In case of multi density clusters XB is the only one index that shows the better performance of FCM over modified FCM as shown in Figure 4.4 (p). But still this supports the better performance of modified FCM in case of multi density as compared to uniform density clusters. We can see that the degradation in modified FCM results indicated here is lesser than the case of uniform density clusters in experiment no.1

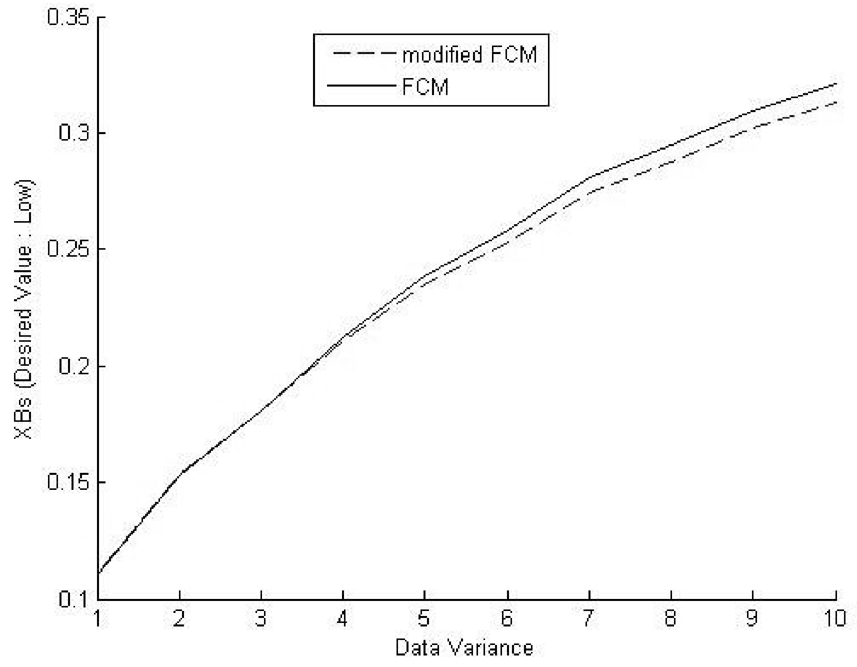


Figure 4.4 (q): XB* vs Variance of Data

In last Figure 4.4 (q) results for XB* are shown. In contrast to experiment no.1 here this validity index now indicates the better performance of modified FCM over FCM. And the performance of modified FCM gets more increased as the clusters becomes spread and overlapped.

4.4 Experiment No. 3

The most popular problem in clustering is to find the right number of clusters in data. In this experiment the modified FCM is compared with FCM in different aspects of input data for finding optimal number of clusters. Results obtained from this experiment are summarized in Table 4.2.

4.4.1 Well Separated Compact Clusters

First we took synthetic dataset of five well separated clusters as shown in Figure 4.5. Starting from initial guess of ten clusters merging down to two clusters, values of fifteen internal validity indices were observed for both FCM and modified FCM. The data was well separated and contained compacted clusters so both algorithms FCM and modified FCM easily determined the true number of clusters for all fifteen validity indices.

4.4.2 Well Separated Compact Clusters with Noise

Another synthetic dataset of five well separated compact clusters but this time with noise is shown in Figure 4.6. Here also both algorithms produce same results i.e. out of fifteen validity indices thirteen validity indices determines true number of clusters.

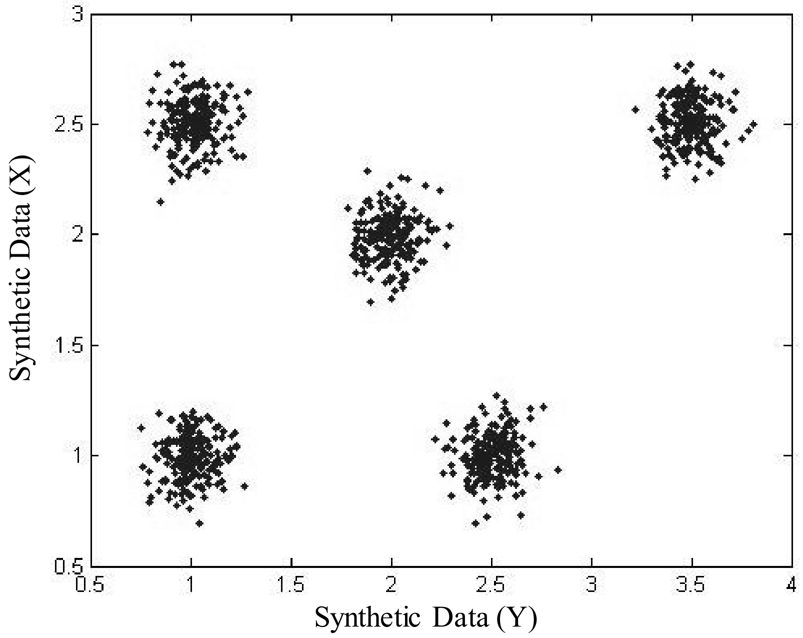


Figure 4.5: Well Separated Compact Clusters

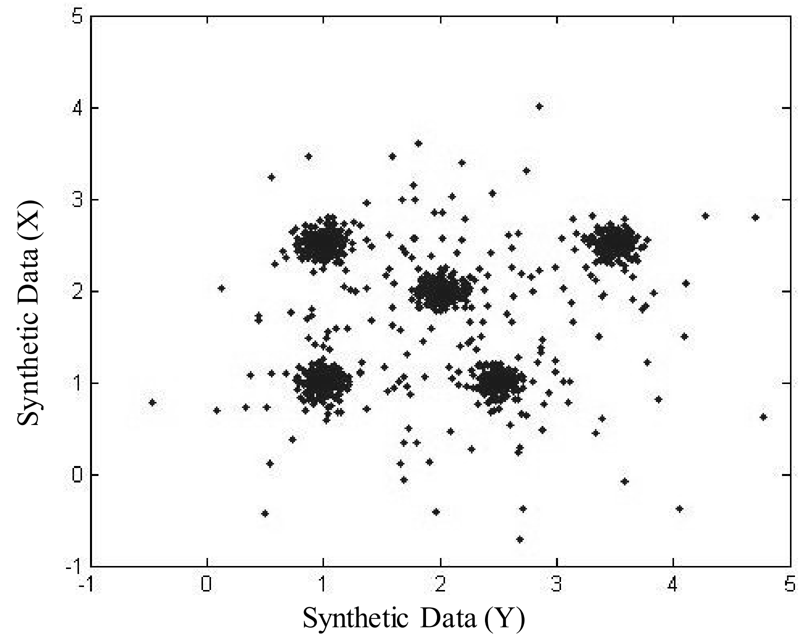


Figure 4.6: Well Separated Compact Clusters with Noise

4.4.3 Multi Density Clusters

Here we took a dataset of three multi density clusters shown in Figure 4.7. Results of this experiment indicate the improvement of modified FCM of FCM. In this case FCM determines true number of clusters for eleven out of fifteen indices while modified FCM determines true number of clusters for twelve out fifteen indices i.e. an edge of one validity index.

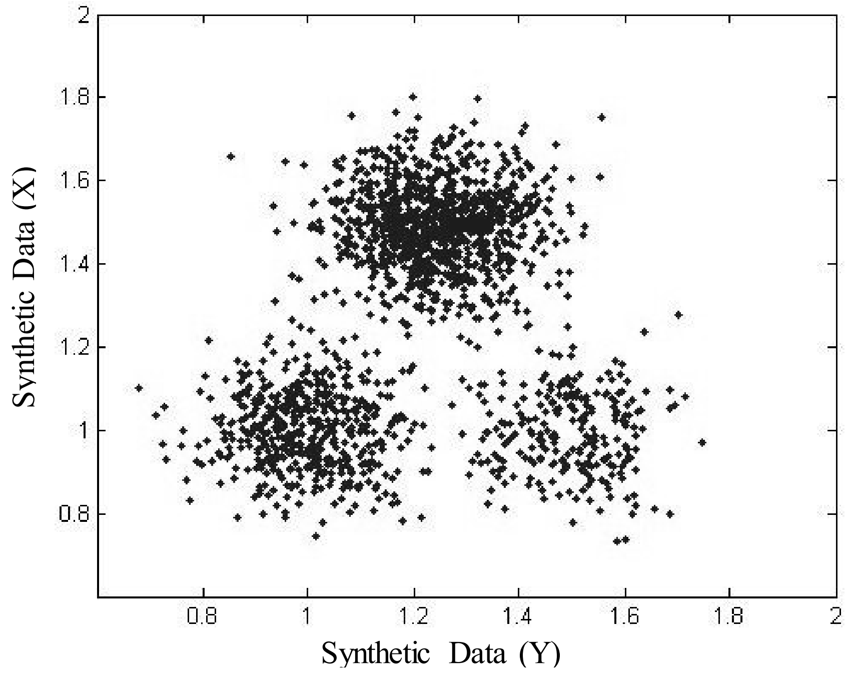


Figure 4.7: Multi Density Clusters

4.4.4 Sub-Clusters

In this we generate a synthetic dataset of five clusters with four of them as sub-clusters since they form two pairs of clusters respectively shown in Figure 4.8. Here both algorithms produce same results i.e. six validity indices out of fifteen determines true number of clusters (five) while rest of nine validity indices indicates presence of three clusters in the data.

4.4.5 Skew Distributed Clusters

In real world problems clusters are not always of same size, he we consider another dataset containing three clusters with one of them having much larger size than the other two as shown in Figure 4.9. This time results shows a more clear edge got by modified FCM over

FCM. For FCM six while for modified FCM eight out of fifteen indices determines true number of clusters i.e. an edge of two validity indices.

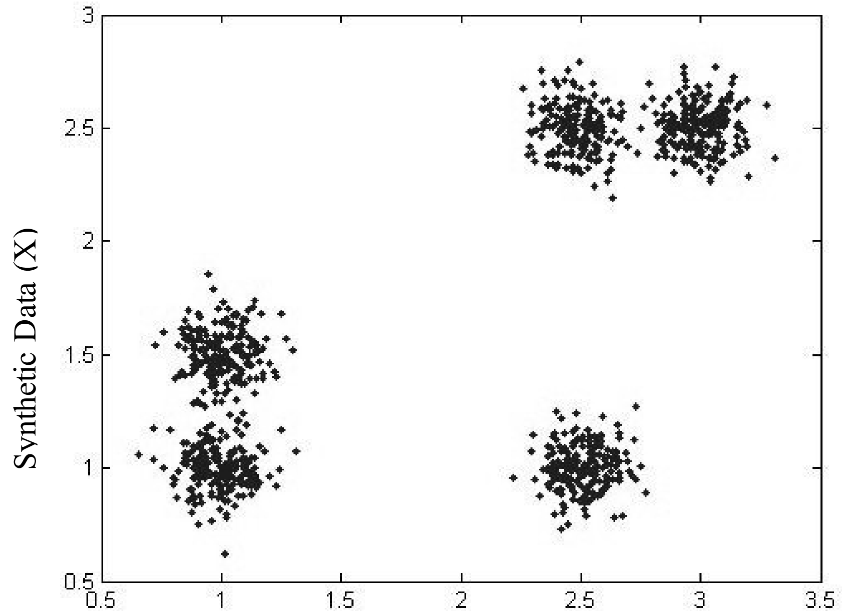


Figure 4.8: Sub-Clusters

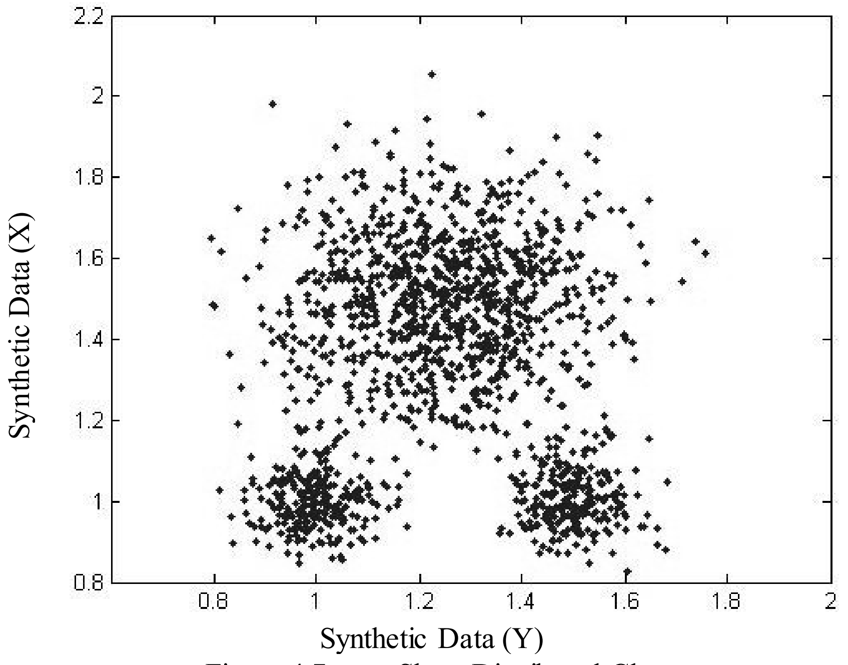


Figure 4.7: Skew Distributed Clusters

4.4.6 Iris Dataset

Next we apply the algorithm to the real world dataset Iris containing three true clusters of 50 samples in each cluster. Results observed from the experiment shows that out of fifteen two indices for FCM while three indices for modified FCM determines true number of clusters. Here again modified FCM acquires an improvement of one validity index over FCM.

4.4.7 Wine Dataset

Another real world dataset, mostly used for analyzing clustering algorithm is Wine dataset. Modified FCM determines the true number of clusters for seven validity indices while FCM determine true number of clusters for six indices. Again a lead of one validity index by modified FCM.

For all of the above dataset summarized results are shown in Table 4.2. Table consists of dataset name along with true number of clusters indicated in parenthesis. In results the determined number of clusters using each validity index is shown in their respective columns for both FCM and modified FCM. Numbers in bold shows that correct number of true clusters is determined by the respective algorithm is that validity index. The numbers having grey color background indicates the improvement attained by modified FCM over FCM.

From the table we can see that the modified FCM determines the actual number of clusters using more validity indices as compared to FCM in case of multi density, skew distributed clusters, Iris dataset and Wine dataset. While in the case of well separated compact, well separated compact with noise and sub-clusters the results are same for both algorithms.

It can also be noted from the results that the improvement got by modified FCM in determining actual number of clusters is in the case of just two indices PC and I index. For rest of validity indices the results are almost same for both algorithms. This is due to the fact that determining true number of clusters depends on the combined effort of both clustering algorithm and a validity index.

4.5 Comparison with Projected Rough Fuzzy c-means Clustering

For further confirming the performance improvement of modified FCM it is compared with similar type of work already done. For this, the results from a research paper of Projected Rough Fuzzy c-means [46] (PRFCM) clustering are used. In these results comparison of PRFCM is done with FCM and Rough FCM (RFCM) on five real world datasets from UCI Machine Repository. Now these results are compared with proposed algorithm in Table 4.3. It is clear from results that for all five datasets modified FCM outperforms over all three algorithms FCM, RFCM and PRFCM.

TABLE 4.2: Determining Number of Clusters using FCM and Modified FCM (MFCM)

Dataset (True # of Clusters)	**Algorithm**	**I**	**PC**	**CE**	**CH**	**Dunn**	**PCAES**	**Silhouette**	**SD**	**DB**	**DB***	**S**	**SC**	**XB**	**XB***	**SDbw**
Well Separated Compact **(5)**	FCM	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
	MFCM	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
Well Separated Compact with Noise **(5)**	FCM	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	10	10	**5**	**5**	**5**
	MFCM	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	10	10	**5**	**5**	**5**
Multi-Density **(3)**	FCM	2	2	2	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	10	**3**	**3**	**3**
	MFCM	2	**3**	2	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	10	**3**	**3**	**3**
Sub-Cluster **(5)**	FCM	**5**	3	3	**5**	3	3	3	**5**	3	3	**5**	**5**	3	3	**5**
	MFCM	**5**	3	3	**5**	3	3	3	**5**	3	3	**5**	**5**	3	3	**5**
Skew Distributed **(3)**	FCM	2	2	2	6	**3**	5	**3**	**3**	**3**	**3**	4	4	**3**	**3**	5
	MFCM	**3**	**3**	2	6	**3**	5	**3**	**3**	**3**	**3**	4	4	**3**	**3**	5
Iris **(3)**	FCM	2	2	2	**3**	2	2	2	**3**	2	2	2	6	2	2	6
	MFCM	**3**	2	2	**3**	2	2	2	**3**	2	2	2	6	2	2	6
Wine **(3)**	FCM	2	2	2	**3**	2	**3**	2	2	**3**	**3**	5	5	**3**	**3**	2
	MFCM	**3**	2	2	**3**	2	**3**	2	2	**3**	**3**	5	8	**3**	**3**	2

Table 4.3: Comparison of Modified FCM with FCM, RFCM, PRFCM

Datasets	**PRFCM**		**FCM**		**RFCM**		**Modified FCM**	
	PC	CE	PC	CE	PC	CE	PC	CE
Breast Cancer	0.5745	0.6153	0.5745	0.6156	0.5399	0.6524	**0.6939**	**0.4783**
Spambase	0.5448	0.6472	0.6238	0.5609	0.6062	0.5765	**0.8555**	**0.2699**
Wine	0.3562	1.0654	0.7912	0.3800	0.3988	1.006	**0.8228**	**0.318665**
Diabetes	0.8242	0.2968	0.8242	0.2968	0.8463	0.2611	**0.8966**	**0.1825**
Magic	0.1524	1.9139	0.2913	1.5159	0.3296	1.4342	**0.6807**	**0.4875**

CHAPTER 05: CONCLUSION AND FUTURE WORK

5.1 Conclusion

Clustering is the process of grouping objects with similar properties. Any cluster should exhibit two main properties; high intra-class similarity and low inter-class similarity. This thesis mainly focuses on the second property of cluster.

First chapter starts with definition of a cluster then know-how of data clustering and the similarity and dissimilarity measure for clustering was introduced. This chapter also included an overview of different types of clustering algorithms that are currently used for different types of data. In last summary of cluster validity and some of the application of data clustering were also discussed.

In the second chapter, some of the clustering techniques were discussed in detail. The main focus was on k-means type clustering techniques. Chapter starts with discussion about classical k-means clustering followed by fuzzy k-means, which adds the advantage of fuzziness to classical k-means by incorporating the fuzzy membership matrix in its objective function. Improvements were made to fuzzy k-means by adding the negative entropy term in competitive agglomerative clustering. Additional improvement was made to competitive agglomerative clustering by changing negative entropy term in agglomerative fuzzy k-means clustering. Another aspect through which improvements are made to classical k-means is by entropy weighting. This aspect is discussed in entropy weighting k-means (EWKM) clustering and improved EWKM clustering. At the end a grid based clustering technique STING and density based clustering algorithms DBSCAN were discussed.

In third chapter the design and implementation of proposed modification was presented. Chapter starts with motivation and work related to proposed modification. The main focus was on the contribution of effort toward a method that addresses the low inter-class similarity. To fulfill the objective of this thesis, a modification to existing fuzzy c-means was suggested i.e. to bias the fuzzy membership matrix with inter-class dissimilarity value (Euclidean distance used here). An algorithm and a flow chart describing the working of the algorithm were included. And then the implementation of the algorithm in MATLAB was explained.

Chapter four contains a summary of validation indices used to validate results and three comprehensive experiments. In first experiment (validity indices vs. changing data variance of uniform density clusters) results showed the improvement in finding crisp clusters by the proposed algorithm over fuzzy c-means. Second experiment (validity indices vs. changing data variance of uniform density clusters) shows that proposed algorithm performs better in case of multi-density clusters than uniform density clusters. These two experiments also confirmed that the accuracy of proposed algorithm increase as data variance is increased. In the third experiment a well-known problem of clustering i.e. finding optimal number of clusters in data is addressed for different aspects of input data. The results showed that proposed algorithm performs better than traditional FCM in case of multi-density and skew distributed clusters. Results of real world dataset Iris and Wine also confirms the improvement achieved by proposed algorithm over FCM.

From all the discussions and the results obtained it can be concluded that the suggested modification showed the importance of considering both the properties of a cluster i.e. high intra-class similarity and high inter-class dissimilarity in designing clustering algorithm. All the results showed an improvement of proposed algorithm over the existing fuzzy c-means clustering.

5.2 Future work

In future more efforts can be added to consider high inter-class dissimilarity among clusters in the process of clustering. This effort can be in the form of introducing some new technique that can be incorporating the second objective of clustering in the objective function and solving the minimization problem or can be in the form of applying the same technique of biasing fuzzy membership matrix to other clustering algorithms. Here in this thesis emphasis was on numeric type data. This technique can be further extended to other types of data as well. Here the effect of proposed improvement has been tested against a limited number of aspects of input data. Performance of proposed improvement can be explored further for more aspects of data.

REFERENCES

[1] Jain, Data clustering: 50 years beyond k-means. *Pattern Recognition Letters,* 2010, volume 31(Issue 8): Pages 651–666.

[2] Everitt, *Cluster Analysis.* 4th edition. 2001, Oxford University Press, New York,

[3] Bock, Probabilistic aspects in cluster analysis. *Conceptual and Numerical Analysis of Data,* 1989, pages 12–44, Augsburg, FRG. Springer-Verlag.

[4] Carmichael, Finding natural clusters. *Systematic Zoology,* 1968, Volume 17(Issue 2): Pages 144–150.

[5] Everitt, *Cluster Analysis.* 3rd edition. 1993, Halsted Press, New York, Toronto.

[6] Jain, Data clustering: A review. *ACM Computing Surveys*, 1999, Volume 31(Issue 3): Pages 264–323.

[7] Gan, Data Clustering: Theory, Algorithms, and Applications, *Series on Statistics and Applied Probability.* 2007, volume 20 of ASA-SIAM

[8] Jain and Dubes, *Algorithms for Clustering Data*. 1988, Prentice Hall, Englewood Cliffs, New Jersey.

[9] Xu and Wunsch, II, *Clustering.* 2009, Wiley-IEEE Press, Hoboken, New Jersey.

[10] Duda, Pattern Classification. *2nd edition.* 2001, John Wiley & Sons, New York, NY,

[11] Mao and Jain, A self-organizing network for hyper ellipsoidal clustering (hec). *IEEE Transactions on Neural Networks,* 1996, Volume 7(Issue 1): Pages 16–29.

[12] Kaufman & Rousseeuw, Finding Groups in Data—An Introduction to Cluster Analysis. *Wiley series in probability and mathematical statistics.* 1990, John Wiley & Sons, Inc., New York.

[13] Grabusts and Borisov, Using grid-clustering methods in data classification. *International Conference on Parallel Computing in Electrical Engineering,* PARELEC '02. Proceedings pages 425–426, Latvia. IEEE.

[14] El-Sonbaty, An efficient density based clustering algorithm for large databases. *16th IEEE International Conference on Tools with Artificial Intelligence, 2004.* ICTAI 2004, pages 673–677.

[15] Beyer, When is "nearest neighbor" meaningful? In Beeri, C. and Buneman, P., editors, *Proceedings of 7th International Conference on Database Theory,* 1999, volume 1540 of Lecture Notes in Computer Science, pages 217–235. Springer, New York.

[16] Agrawal, Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Record ACM Special Interest Group on Management of Data,* 1998, pages 94–105, New York, NY, ACM Press.

[17] Parsons, Subspace clustering for high dimensional data: A review. *SIGKDD, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining,* 2004, Volume 6(Issue 1): Pages 90–105.

[18] Cao and Wu, Projective ART for clustering data sets in high dimensional spaces. *Neural Networks,* 2002, Volume 15(Issue 1): Pages 105–120.

[19] Aggarwal and Yu, Finding generalized projected clusters in high dimensional spaces. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data,* May 16-18, 2000, Dallas, TX, volume 29, pages 70–81. ACM.

[20] Halkidi, Clustering validity checking methods: part II. *ACM SIGMOD Record,* 2002, Volume 31(Issue 3).

[21] Anderberg, *Cluster Analysis for Applications.* 1973, Academic Press, New York.

[22] Hartigan, *Clustering Algorithms.* 1975, John Wiley & Sons, Toronto.

[23] Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics,* 1974, Volume 3(Issue 3): Pages 32–57.

[24] Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* 1981, Kluwer Academic Publishers, Norwell, MA.

[25] Frigui, Clustering by Competitive Agglomeration, *Pattern Recognition,* 1997, Volume 13 (Issue 7), Pages 86-92.

[26] Mark J, Agglomerative Fuzzy K-Means Clustering Algorithm with Selection of Number of Clusters, *IEEE Transactions on Knowledge and Data Engineering,* 2008, Volume 20 (Issue 11).

[27] Elaine Y. Chan, An optimization algorithm for clustering using weighted dissimilarity measures, *Pattern Recognition*, 2003, Volume 37 (Issue 5): Pages 943-952

[28] Liping Jing, An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data, *IEEE Transactions On Knowledge And Data Engineering,* 2007, Volume 19 (Issue 8): Pages 1026-1041

[29] Taoying, Chen, An Improved k-means Algorithm for Clustering Using Entropy Weighting Measures, Taoying Li and Yan Chen, *Proceedings of the 7th World Congress on Intelligent Control and Automation,* June 25 - 27, 2008, Chongqing, China.

[30] Wang, STING: A statistical information grid approach to spatial data mining (1997), Wei Wang , Jiong Yang , Richard Muntz, *http://citeseer.ist.psu.edu*

[31] Wang, STING+: An Approach to Active Spatial Data Mining (1999), Wei Wang , Jiong Yang , Richard Muntz, *Proceedings 15th International Conference on Data Engineering*

[32] Ester, A density-based algorithm for discovering clusters in large spatial databases with noise, Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei

Xu, *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*

[33] Frank Hoeppner, *Fuzzy cluster analysis: methods for classification, data analysis, and image recognition.* 1999, John Wiley and Sons.

[34] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," Comm. in Statistics, vol. 3, no. 1, pp. 1–27, 1974.

[35] J. Dunn, "Well separated clusters and optimal fuzzy partitions," J. Cybern., vol. 4, no. 1, pp. 95–104, 1974.

[36] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," IEEE PAMI, vol. 24, pp. 1650–1654, 2002.

[37] Kuo-Lung Wu, Miin-Shen Yang, "A cluster validity index for fuzzy clustering", Pattern Recognition Letters 26 (2005) 1275–1291

[38] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," J. Comput. Appl. Math., vol. 20, no. 1, pp. 53–65, 1987.

[39] M. Halkidi, M. Vazirgiannis, and Y. Batistakis, "Quality scheme assessment in the clustering process," in PKDD, London, UK, 2000, pp. 265–276.

[40] D. Davies and D. Bouldin, "A cluster separation measure," IEEE PAMI, vol. 1, no. 2, pp. 224–227, 1979.

[41] M. Kim and R. S. Ramakrishna, "New indices for cluster validity assessment," Pattern Recogn. Lett., vol. 26, no. 15, pp. 2353–2363, 2005.

[42] Bensaid, A.M.; Hall, L.O.; Bezdek, J.C.; Clarke, L.P.; Silbiger, M.L.; Arrington, J.A.; Murtagh, R.F.; , "Validity-guided (re)clustering with applications to image segmentation," Fuzzy Systems, IEEE Transactions on , vol.4, no.2, pp.112-123, May 1996

[43] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," IEEE PAMI, vol. 13, no. 8, pp. 841–847, 1991.

[44] Bezdek, J.C., 1974a. Cluster validity with fuzzy sets. J. Cybernet. 3, 58–73.

[45] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," in ICDM, Washington, DC, USA, 2001, pp. 187–194.

[46] Charu and Naveen, "Project Rough Fuzzy c-means Clustering", in IEEE 11th International Conference on Intelligent Systems Design and Applications, 2011

[47] http://www.lysator.liu.se/~jc/mthesis/4_Entropy.html

[48] C. E. Shannon. A mathematical theory of communication. Bell Sys. Tech. J., 27:379-423, 623-656, 1948.