CHAPTER 4

Clustering Algorithms and Validity Indices

This chapter presents a brief overview on clustering, similarity measures and different taxonomical representations of clustering. The chapter also presents discussions on some prominent clustering algorithms and some popular clustering validity assessment indices. Further some experiments on the applications of clustering algorithms for pattern recognition on numeric, image and text data are also presented.

4.1 Overview

Clustering is a fundamental technique for pattern recognition and machine learning. It is one of the vital means of dealing with the data that is encountered in our day-to-day life. It is a key technique for discovering the inherent structure of any given data set. Formally, clustering is defined as the unsupervised classification or partitioning of input patterns (observations, data items, data points, or feature vectors) into clusters (classes, groups) such that data items within a cluster are similar in some respect (often based on proximity according to certain defined distance measure) and dissimilar from those in other clusters. Since clustering is unsupervised, the patterns do not come with prior class label information and so the classification is based on inherent statistical structure of the overall collection of input patterns.

Besides the term clustering, there are a number of other terms with similar meanings which include cluster analysis, automatic classification, numerical taxonomy, botryology and typological analysis, that are also in use [55].

Technically stated, the aim of any clustering technique is to obtain a $K \times n$ partition matrix $U(X)$ of a given data set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ in $\Re^d$ (each point in X is d-dimensional) representing its partitioning into K clusters (C_1, C_2,...,C_K) by optimizing a criterion function defined globally over all of the patterns in the data set. The partition matrix $U(X)$ may be represented as $U = [\mu_{kj}]$, $k = 1,..., K$, and $j = 1,..., n$ where μ_{kj} is the membership degree of a pattern $\mathbf{x}_j$ in cluster C_k. For crisp or hard partitioning of the data, $\mu_{kj} = 1$ if $\mathbf{x}_j \in C_k$, otherwise $\mu_{kj} = 0$ [56].

A graphical representation of clusters and some possible cluster shapes is shown below:

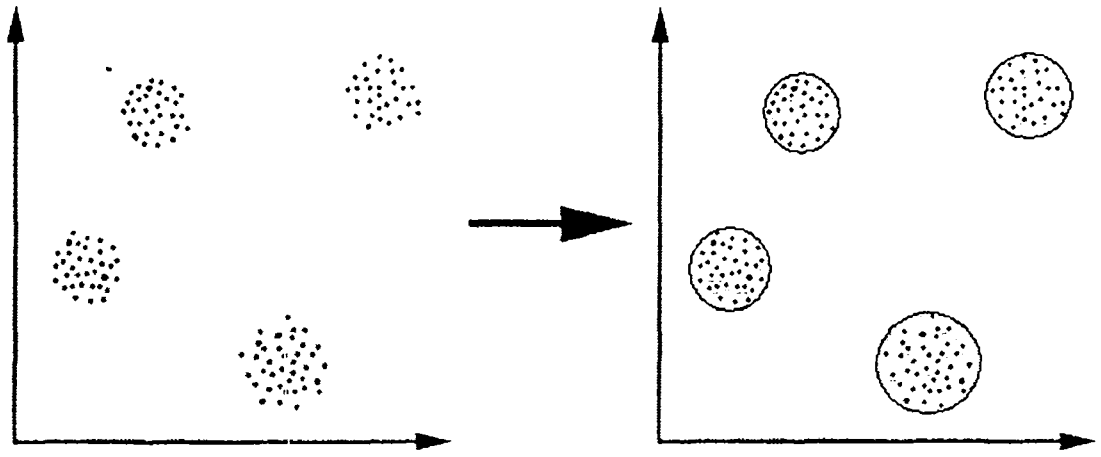
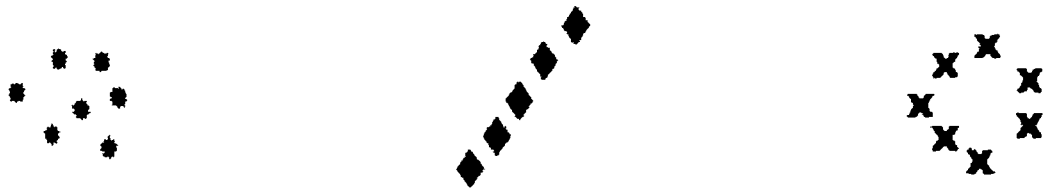


Figure 4.1: Graphical representation of clusters

When depicted graphically, clusters may come in several different shapes, not necessarily circular, as shown below:



Compact Linear Circular

Figure 4.2: Clusters of different shapes

4.2 Proximity Measures

One of the oldest and most influential theoretical assumptions is that the similarity between two patterns is inversely related to psychological distance between the perceptions of the two patterns. Thus an important step in any clustering process is to select a proximity measure that determines how the similarity or dissimilarity of two data points is to be calculated. This influences the shape of the clusters, as some elements may be close to one another according to one proximity measure and farther away according to another. It is thus a most common practice to calculate the dissimilarity using a distance measure defined on the feature space of patterns. Some common *proximity measures* [57] [58] (also called *similarity measures* or *distance measures*) used in literature are described below:

(i) The Euclidean distance: It is the most common distance measure used and preferred by researchers in the field of clustering as it takes the magnitude of the data into account. It is also called 2-norm distance. The Euclidean distance is a true metric, and is defined as:

$$D(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$

(ii) The Manhattan distance: The Manhattan distance is also known by the names city-block distance or 1-norm distance. It is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. It is defined as:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d}|x_i - y_i|$$

(iii) The Minkowski distance: The Minkowski distance of order p between the two points $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$D(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d}|x_i - y_i|^p\right)^{1/p}$$

It is the generalization of both the Euclidean distance and the Manhattan distance. When $p = 2$, the Euclidean distance is obtained, and when $p = 1$ the Manhattan distance is obtained. Minkowski distance is a popular

distance measure for higher dimensional data.

In the above three cases **x** and **y** denote two d-dimensional points such that $\mathbf{x} = (x_1, x_2, ..., x_d)$ and $\mathbf{y} = (y_1, y_2, ..., y_d)$ and $D(\mathbf{x}, \mathbf{y})$ denotes the distance between them.

(iv) The cosine distance: The cosine distance gives the angular cosine distance between two vectors of d dimensions. Given two vectors P and Q the cosine distance, $\cos(\theta)$, is defined as:

$$\cos(\theta) = \frac{P.Q}{\|P\| \|Q\|}$$

The cosine distance is a popular distance measure for clustering a collection of documents represented using the vector space model [35].

Other than the above mentioned proximity measures, many other proximity measures are also found in literature like the Mahalanobis distance and the Hamming distance. A detailed discussion on proximity measures can be found in [58].

4.3 Taxonomy of Clustering

Data clustering can be done using a number of different approaches. Several taxonometric representations of clustering methodology are possible [3] [59] [60] which are described below:

(i) Agglomerative vs. Divisive: An agglomerative (also sometimes called hierarchical) approach begins with each pattern in a distinct (singleton) cluster, and successively merges clusters together until a stopping criterion is satisfied. In the end, a hierarchy of clusters is found. A divisive (also called partitional) method begins with all patterns in a single cluster and performs splitting of patterns into mutually exclusive (hard) or overlapping (fuzzy) clusters until a stopping criterion is met. The partitional clustering algorithms usually adopt an iterative optimization paradigm. It starts with an initial partitioning and requires several iterations to obtain the final

partitioning by optimizing the criterion function. Along with this, in all iterations, any partitional clustering algorithm also computes a representative point, called a centroid or center, using different measures of central tendency (mean, median, mode etc) of all points in the cluster, for all the clusters obtained thus far. The centroids are used a representatives of the clusters obtained.

(ii) Monothetic vs. Polythetic: Monothetic methods use single-feature-based assignment to divide objects into clusters. Polythetic algorithms consider multiple-feature-based assignment. Most algorithms are polythetic; that is, all features enter into the computation of distances between patterns, and decisions are based on those distances. A monothetic algorithm considers features sequentially to divide the given collection of patterns. An example of a monothetic clustering algorithm is presented by Anderberg in [61].

(iii) Hard vs. Fuzzy: A non-fuzzy, crisp or hard clustering algorithm assigns each pattern to exactly one cluster during its operation and in its output. A fuzzy clustering method assigns degrees of membership in several clusters to each input pattern; thus a pattern can belong to more than one cluster. A fuzzy clustering algorithm can be converted to a hard clustering algorithm by assigning each pattern to the cluster for which the pattern has the largest degree of membership.

(iv) Deterministic vs. Stochastic: Deterministic clustering methods always arrive at the same clustering for a given a data set. Stochastic clustering methods employ stochastic elements (e.g., random numbers) to find a good clustering and may not necessarily obtain the same clustering for a given a data set, always.

(v) Incremental vs. Non-incremental: Non-incremental clustering methods mainly rely on having the whole data set ready before applying the algorithm. A hierarchical agglomerative clustering algorithm belongs to this class. Incremental clustering algorithms work by assigning objects to their respective clusters as they are encountered.

4.4 K-means Algorithm

The K-means clustering algorithm is the simplest and most commonly used clustering algorithm that usually employs the sum of squared error (*SSE*) criterion or objective function [62]. It is hard, partitional, non-deterministic, polythetic, non-incremental clustering algorithm that partitions a data set X of n points, into a pre-specified number (K) of mutually exclusive clusters (C_1, C_2,.....,C_K) by optimizing the *SSE* criterion function defined globally over all of the patterns in the data set. The data set X may be represented as a matrix $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots\ldots, \mathbf{x}_n)$ in $\Re^d$ where each row of X, $\mathbf{x}_i$, represents a d-dimensional pattern.

Some important points in K-means clustering algorithm are:

1. None of the clusters is empty; $C_k \neq \emptyset$ for any k.
2. Every sample belongs to a single cluster; $C_i \cap C_j = \emptyset$, $\forall\, i \neq j$.
3. The centroids of clusters are to be found in such a way that they are, as much as possible, far away from each other.

The sum of squared error (*SSE*) criterion function is defined as:

$$J = \sum_{j=1}^{n} \sum_{k=1}^{K} \mu_{kj} \left\| \mathbf{x}_j - \mathbf{z}_k \right\|^2$$

where, $\left\| \mathbf{x}_j - \mathbf{z}_k \right\|^2$ is a chosen distance measure between a data point $\mathbf{x}_j$ in cluster C_k and its cluster centroid $\mathbf{z}_k$. $\mu_{kj} = 1$ if $\mathbf{x}_j \in C_k$, otherwise $\mu_{kj} = 0$. The index J is an indicator of the distance of the n data points from their respective cluster centroids.

The K-means algorithm works as follows:

Step 1): Randomly choose a user specified number of K initial centroids (also called seeds) inside the multidimensional feature space containing the pattern set.

Step 2): Assign each data point to the nearest centroid.

Step 3): Re-calculate the cluster centroids by taking the mean (thus the name) of all points in each cluster.

Step 4): Repeat the process until the K centroids no longer change or negligible changes occur in two successive steps.

The K-means clustering algorithm aims to minimize the value of the *SSE* objective function in each iteration. The algorithm stops when the values of this objective function, and as a result the centroids do not change or negligible changes occur in two successive iterations.

Advantages of K-means algorithm:

1) It is very easy to implement, and has been adapted to many problem domains.
2) Its time complexity is $O(nKdt)$, where n is the number of patterns, K is the number of clusters, d is the number of dimensions in a pattern and t is the number of iterations. Since K, d, and t are usually much less than n, the time complexity of K-means is approximately linear. Therefore, K-means is a good selection for clustering medium and large-sized data sets [63].
3) It does not require the user to specify too many parameters.
4) The algorithm is deterministic after the initial centroids are determined.

Based on the use of different measures of central tendency, the K-means clustering algorithm has three other popular hard clustering derivatives also, namely K-medoids, K-medians and K-modes. This is because the K-means algorithm requires the concept of a mean to be definable for all kinds of data set but which is not always the case [5]. Also for some data sets a mean is more influenced by outliers (*data points lying far apart from all other points in the feature space*) or other extreme values. The derivative K-medoids method can be used in such cases. Compared to means, medoids are less sensitive to such points. The medoids are the nearest objects to the mean of data in one cluster. Calculation of medoids is however costlier than calculation of the straightforward means, as after the calculation of the means, the points in a cluster again need to be checked for their vicinity from their mean. Use of median results in the K-medians method, where the median or 'middle value' is taken for each ordered attribute. Alternatively, in the K-modes method, the most frequent value for each attribute is used. All methods require the user to specify K, the number of clusters.

4.5 Fuzzy c-means Algorithm

The Fuzzy c-means clustering algorithm was proposed by Dunn [64] and was later extended by Bezdek [65] as a fuzzy version of the K-means algorithm. Thus the Fuzzy c-means (FCM) algorithm is sometimes also called as fuzzy K-means algorithm. The algorithm uses the soft computing technique, fuzzy logic (discussed in chapter 3) to obtain a fuzzy partitioning of the input data set X. The algorithm has been the dominant approach in both theoretical and practical applications of fuzzy techniques to unsupervised pattern classification or recognition [65] [66].

In traditional hard clustering approaches the partitioning of input data set X is done in a mutually exclusive manner, thus each pattern belongs to one and only one cluster. The clustering obtained in hard clustering is hence disjoint. Fuzzy clustering extends this notion and associates each pattern with every cluster using a membership function. The design of the membership function is the most important problem in fuzzy clustering. The aim of a fuzzy clustering algorithm is to obtain a fuzzy *pseudopartition*. Hard clustering can be obtained from a fuzzy pseudopartition by thresholding the membership degrees.

A fuzzy pseudopartition or fuzzy c-partition of the data set X of n points is a family of fuzzy subsets of X, denoted by $\mathcal{P} = \{A_1, A_2, ..., A_c\}$, which satisfies

$$\sum_{k=1}^{c} A_k(\mathbf{x}_j) = 1,$$

for all $j \in \{1, 2, ..., n\}$ and,

$$0 < \sum_{j=1}^{n} A_k(\mathbf{x}_j) < n,$$

for all $k \in \{1, 2, ..., c\}$, where c is the number of clusters (a positive number) and $\mathbf{x}_j$ is the j^{th} data point or pattern in X. Comparing $\mathcal{P}$ with $U(X)$, it can be easily deduced that:

$$A_k = \{\mu_{k1}, \mu_{k2}, ..., \mu_{kn}\}.$$

The FCM algorithm uses the fuzzy extension of the *SSE* criterion function as the objective function. It is defined as: $J_m(\mathcal{P}) = \sum_{j=1}^{n}\sum_{k=1}^{c}(\mu_{kj})^m \left\|\mathbf{x}_j - \mathbf{z}_k\right\|^2$, where $\left\|\mathbf{x}_j - \mathbf{z}_k\right\|^2$

represents the distance between $\mathbf{x}_j$ and $\mathbf{z}_k$ (the k^{th} cluster centroid). Like, K-means, in the case of FCM also the distance calculation is usually done using the Euclidean distance. The performance index $J_m(\mathcal{P})$ measures the weighted sum of distances between clusters and points in the corresponding fuzzy clusters. Thus, the smaller the value of $J_m(\mathcal{P})$ the better the clustering is. Therefore, the goal of the FCM clustering algorithm is to find a fuzzy pseudopartition $\mathcal{P}$ that minimizes the value of $J_m(\mathcal{P})$.

The FCM algorithm is now presented below. The algorithm is based on the assumption that the desired number of clusters c is given and, in addition, a particular distance measure, a real number $m \in (1,\infty)$, and a small positive number ε, serving as a stopping criterion, are also chosen.

Step 1): Let $t = 0$. Select an initial partition matrix $U^{(0)}$ of the original data set using a random generator.

Step 2): Calculate the c cluster centroids $\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, ..., \mathbf{z}_k^{(t)}$ for $U^{(t)}$, for a chosen value of m.

$$\mathbf{z}_k^{(t)} = \frac{\sum_{j=1}^{n} (\mu_{kj})^m \mathbf{x}_j}{\sum_{j=1}^{n} (\mu_{kj})^m}$$

for all $k \in \{1,2,...,c\}$

Step 3): Update $U^{(t+1)}$ by the following procedure: For each $\mathbf{x}_j \in X$, if $\left\|\mathbf{x}_j - \mathbf{z}_k^{(t)}\right\|^2 > 0$ for all $k \in \{1,2,...,c\}$, then define

$$\mu_{kj}^{(t+1)} = \left[\sum_{i=1}^{c} \left(\frac{\left\|\mathbf{x}_j - \mathbf{z}_k^{(t)}\right\|^2}{\left\|\mathbf{x}_j - \mathbf{z}_i^{(t)}\right\|^2} \right)^{\frac{1}{m-1}} \right]^{-1};$$

if $\left\|\mathbf{x}_j - \mathbf{z}_k^{(t)}\right\|^2 = 0$ for some $k \in I \subseteq \{1,2,...,c\}$, then define $\mu_{kj}^{(t+1)}$ for $k \in I$ by any non-negative real numbers satisfying:

$$\sum_{k \in I} \mu_{kj}^{(t+1)} = 1,$$

and define $\mu_{kj}^{(t+1)} = 0$ for all $k \in \{1,2,...,c\} - I$.

Step 4): Compare $U^{(t)}$ and $U^{(t+1)}$. If $\left|U^{(t+1)} - U^{(t)}\right| \leq \varepsilon$, then stop; otherwise, increase t by one and return to Step 2.

$\left|U^{(t+1)} - U^{(t)}\right|$ denotes a distance between the fuzzy pseudopartitions $U^{(t+1)}$ and $U^{(t)}$ in the space $\Re^{n \times c}$. An example of this distance is

$$\left|U^{(t+1)} - U^{(t)}\right| = \max_{1 \leq k \leq c, 1 \leq j \leq n} \left|\mu_{kj}^{(t+1)} - \mu_{kj}^{(t)}\right|.$$

The parameter m is selected according to the problem under consideration. When $m \to 1$, the membership degrees converge to 0 or 1 and the algorithm converges to the crisp K-means algorithm. When $m \to \infty$, all clusters tend towards the centroid of the data set X. That is, the partition becomes fuzzier with increasing m. Presently, there is no theoretical basis for an optimal choice for the value of m. However, it is established that the algorithm converges for any $m \in (1,\infty)$ [14].

The advantages of the FCM algorithm are similar to those of its hard counterpart i.e., K-means and the time complexity of FCM is also reported to be similar to that of K-means [67]. Further, the FCM algorithm is more suitable than K-means for clustering fuzzy data sets, i.e., data sets whose clusters could not be distinctly separated in a mutually exclusive manner, easily.

4.6 Validity Indices

In most clustering algorithms, the number of clusters K is specified by the user. Once K is specified any typical clustering algorithm always tries to find the best fit partitioning (hard or fuzzy) for the specified number of clusters. However this does not mean that even the best fit is meaningful at all. Either the number of clusters might be wrong or the cluster shapes obtained might not correspond to the actual groups in the data if the data can be grouped in a meaningful way at all. Further in many cases a prior knowledge of the actual number of clusters is not available and the optimal number of clusters in a data set cannot be easily and intuitively estimated beforehand and requires evaluation of some metric for its determination. Thus the

determination of the optimal number of clusters in a given pattern set is also an important problem in cluster analysis [68] [69] [70]. Moreover, since clustering algorithms are unsupervised, irrespective of the clustering method (hard or fuzzy) the final partitions of data do require some kind of validation in most applications. This validation is done by evaluating some goodness measures called validity assessment indices or validity indices for short.

The validity indices should be such that they should be able to impose an ordering of the clusters in terms of their goodness. In other words, if $U_1,U_2,...,U_m$ be m partitions of X, and the corresponding values of a validity index be $V_1,V_2,...,V_m$, then $V_{k1} \geq V_{k2} \geq ... \geq V_{km}$ will indicate that $U_{k1} \uparrow U_{k2} \uparrow ... \uparrow U_{km}$, for some permutation $k1$, $k2$,..., km of $\{1, 2,...,m\}$. Here '$U_i \uparrow U_j$' indicates that partition U_i is a better clustering than U_j [56].

There are two criteria [71] [72] for evaluation and selection of the optimal clustering from a given set of clustering results of a clustering algorithm:

1) *Compactness*: Members of each cluster should be as close to each other as possible.
2) *Separation*: The clusters should be widely spaced from each other.

A good clustering obtained using a clustering algorithm should have both high compactness and high separation. Further the validity indices should be so defined that they should focus on the checking and measurement of these two criteria for every clustering they evaluate.

Various validity indices have been proposed to quantitatively evaluate the correctness and goodness of a clustering structure to solve the cluster validity problem. Once an appropriate cluster validity index is selected or defined, the general solution to the cluster validity problem is to execute a clustering algorithm for all possible numbers of clusters in sequence and evaluate the cluster validity index value for each clustering structure obtained and select the clustering structure with the best or optimal validity value.

Some representative validity indices for hard and fuzzy clustering are now presented in the following sections. A thorough survey of validity indices can be found in [72].

4.6.1 Validity Indices for Hard Clustering

In this subsection, some of the popular validity indices, used for evaluating the goodness of partitioning obtained by any hard clustering algorithm, are discussed.

***Pakhira Bandyopadhyay Maulik (PBM) Index*:**

This index was proposed by Pakhira *et al.* [68]. The validity index is the square of product of three quantities, $1/K$, E_1/E_K and D_K. Mathematically it is defined as under:

$$PBM(K) = \left[\frac{1}{K} \times \frac{E_1}{E_K} \times D_K \right]^2$$

where, K is the number of clusters. The factor E_1 is the sum of the distances of each pattern $\mathbf{x}_j$ from the geometric center of all patterns, w_0. This factor is independent of the number of clusters and is computed as:

$$E_1 = \sum_{j=1}^{n} \left\| \mathbf{x}_j - w_0 \right\|$$

The factor E_K is the sum of within cluster distances of K clusters, weighted by the corresponding membership degree μ_{kj}:

$$E_K = \sum_{k=1}^{K} \sum_{j=1}^{n} \mu_{kj} \left\| \mathbf{x}_j - \mathbf{z}_k \right\|$$

where, $\mathbf{z}_k$'s are the centroids of the corresponding clusters C_k.

The factor D_K measures the maximum separation between two clusters over all possible pairs of clusters:

$$D_K = \max_{1 \le i \le K, 1 \le j \le K} \left\| \mathbf{z}_i - \mathbf{z}_j \right\|$$

The maximum value of $PBM(K)$ across a hierarchy of m partitions indicates the best partitioning and ensures the formation of a small number of compact clusters with large separation between at least two clusters.

***Xie-Beni (XB) Index*:**

The Xie and Beni index was proposed by Xuan-Li Xie and Gerardo Beni in the year 1991 [73]. The index focuses on the compactness and separation of clusters of

patterns. The index was originally proposed for validation of fuzzy clustering but it can also used to validate hard clustering results. The index is defined by:

$$XB(K) = \frac{\sum_{k=1}^{K}\sum_{j=1}^{n}(\mu_{kj})^m \left\|\mathbf{x}_j - \mathbf{z}_k\right\|^2}{n \times \min_{1\le i\le K, 1\le j\le K}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|^2}$$

Comparing the index with the objective function of the FCM clustering algorithm

$$J_m(\mathcal{P}) = \sum_{j=1}^{n}\sum_{k=1}^{K}(\mu_{kj})^m \left\|\mathbf{x}_j - \mathbf{z}_k\right\|^2$$

it can be inferred that

$$XB(K) = \frac{J_m(\mathcal{P})}{n \times \min_{1\le i\le K, 1\le j\le K}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|^2}$$

where, $J_m(\mathcal{P})$ gives the measure of compactness of clusters and $\min_{1\le i\le K, 1\le j\le K}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|^2$ gives the measure of separation between different cluster centers. The best partitioning across a hierarchy of partitions is indicated by the minimum value of $XB(K)$. The parameter m is called as *fuzzifier* and it is usually chosen between 1 and 2.

Dunn's Index (DI):

The Dunn's Index was proposed by J. C. Dunn [74] in the year 1974 as a validity index for the identification of compact and well separated clusters. The index is thus not suitable for use with fuzzy clusters and is usually used to validate the partitioning obtained by a hard clustering algorithm.

The *DI* index is defined as:

$$DI(K) = \min_{1\le i\le K}\left\{\min_{1\le j\le K, i\ne j}\left\{\frac{\min_{\mathbf{x}\in C_i, \mathbf{y}\in C_j} D(\mathbf{x},\mathbf{y})}{\max_{1\le k\le K}\left\{\max_{\mathbf{x},\mathbf{y}\in C_k} D(\mathbf{x},\mathbf{y})\right\}}\right\}\right\}$$

The compactness measure of a cluster C_k is given by $\max_{\mathbf{x},\mathbf{y}\in C_k} D(\mathbf{x},\mathbf{y})$ and the separation measure between two clusters C_i and C_j is given by $\min_{\mathbf{x}\in C_i, \mathbf{y}\in C_j} D(\mathbf{x},\mathbf{y})$. $D(\mathbf{x},\mathbf{y})$ denotes

the distance between **x** and **y** as per some chosen distance norm.

The best partitioning across a hierarchy of partitions is indicated by the maximum value of $DI(K)$.

The Dunn's Index suffers from the following problems:

- It is computationally very expensive.
- It is sensitive to the presence of noise.

Separation and Compactness (SC) index:

It is the ratio of the sum of compactness and separation of the clusters in a given partitioning. It is defined as:

$$SC(K) = \sum_{k=1}^{K} \frac{\sum_{j=1}^{n} (\mu_{kj})^m \left\| \mathbf{x}_j - \mathbf{z}_k \right\|^2}{n_k \times \sum_{i=1}^{K} \left\| \mathbf{z}_i - \mathbf{z}_k \right\|^2}$$

where, m is usually chosen between 1 and 2.

From the above equation it can be observed that the SC index can also be viewed as a sum of individual cluster validity indices normalized through division by the cardinality n_k (the number of points in a cluster C_k) of each cluster [75].

The SC index is useful when comparing different partitions having equal number of clusters. A lower value of $SC(K)$ indicates a better partitioning across a hierarchy of partitions. The SC index is also called Partition Index.

Davies-Bouldin (DB) Index:

This index [76] is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within the i^{th} cluster is computed as $S_{i,q} = \left(\frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \left\{ \left\| \mathbf{x} - \mathbf{z}_i \right\|^q \right\} \right)^{1/q}$ and the distance between cluster C_i and C_j, denoted by d_{ij}, is defined as $d_{ij,t} = \left\{ \sum_{s=1}^{d} \left| \mathbf{z}_{is} - \mathbf{z}_{js} \right|^t \right\}^{1/t}$. The Davies-Bouldin ($DB$) index is then defined as:

$$DB(K) = \frac{1}{K}\sum_{i=1}^{K} \max_{j, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{d_{ij,t}} \right\}$$

The values of both t and q are usually chosen as 2 for Euclidean distance. A lower value of $DB(K)$ indicates a better partitioning across a hierarchy of partitions.

4.6.2 Validity Indices for Fuzzy Clustering

This subsection presents a discussion on some popularly used validity indices for evaluating the goodness of partitioning obtained by the fuzzy clustering algorithms.

The Fuzzy PBM (PBMF) Index:

The fuzzy version of the *PBM* index called *PBMF* index is obtained by incorporating fuzzy distances. It was also introduced by Pakhira *et al.* [68]. It is defined as follows:

$$PBMF(c) = \left[\frac{1}{c} \times \frac{E_1}{J_m(U,Z)} \times D_c \right]^2$$

where,

$$J_m(U,Z) = \sum_{j=1}^{n}\sum_{k=1}^{K} (\mu_{kj})^m \left\| \mathbf{x}_j - \mathbf{z}_k \right\|$$

U is the partition matrix and Z is the set of cluster centroids. $m \in (1, \infty)$.

Like the hard version, i.e., the *PBM* index, in this case also the best pseudopartitioning across a hierarchy of fuzzy pseudopartitions is determined by the maximum value of the *PBMF* index. It should be noted that D_c is same as D_K, only the notation is different; as in case of fuzzy clustering it is customary to use 'c' to denote the number of clusters, instead of K, which is used in case of hard clustering.

The Xie-Beni (XB) Index:

As already mentioned in subsection 4.6.1, the Xie-Beni Index *XB* was originally proposed for validating the results of fuzzy clustering. The performance of the index heavily depends on the value of the fuzzifier m. For m = 1.5, the index is given by:

$$XB(c) = \frac{\sum_{k=1}^{c}\sum_{j=1}^{n}(\mu_{kj})^{1.5}\left\|\mathbf{x}_j - \mathbf{z}_k\right\|^2}{n \times \min_{1\le i\le c, 1\le j\le c}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|^2}$$

The number of clusters is denoted by c instead of K. Like the hard version the best fuzzy pseudopartitioning across a hierarchy of fuzzy pseudopartitions is indicated by the minimum value of $XB(c)$. The Xie-Beni index XB, has some drawbacks [77]:

- It monotonically decreases when the number of clusters gets close to n.
- The behaviour of XB index becomes unpredictable for very low or high value of the fuzzifier m.

The Partition Coefficient (PC) Index:

This validity index was the first validity index associated with the FCM clustering algorithm. The PC index was proposed by Bezdek [78] as follows:

$$PC(c) = \frac{1}{n}\sum_{k=1}^{c}\sum_{j=1}^{n}(\mu_{kj})^m$$

where, $1/c \le PC(c) \le 1$. The closer to unity the value of PC index is the 'crisper' is the clustering. If all the membership degree values to a fuzzy pseudopartition are equal, i.e., $\mu_{kj} = 1/c$, the PC index obtains its lowest value. Thus a value close to $1/c$ indicates that there is no clustering tendency in the data set or the clustering algorithm failed to reveal it. The best fuzzy pseudopartitioning (or the optimal value of c) is indicated by the maximum value of the PC index. The index provides compact clusters with higher values of μ_{kj}. There are some drawbacks of the PC index:

- The index is monotonically dependant on the number of clusters c.
- The index is sensitive to the value of fuzzifier m.
- The index has no direct connection to any property of the data set.

The Monotonic Partition Coefficient (MPC) Index:

The MPC validity index was proposed by Dave [79] as a modification over the PC index in order to reduce the monotonic dependency of the PC index on c.

The index is defined as:

$$MPC(c) = 1 - \frac{c}{c-1}\left(1 - PC(c)\right),$$

where, $0 \le MPC(c) \le 1$. Like the *PC* index, the optimal pseudopartitioning (or optimal value of c) is obtained by maximizing the value of the *MPC* index.

The Classification Entropy (CE) Index:

The Classification Entropy index *CE* measures the fuzziness in the partition matrix *U*, which is similar to the Partition Coefficient. It was proposed by Bezdek [80]. It is defined as:

$$CE(c) = -\frac{1}{n}\sum_{k=1}^{c}\sum_{j=1}^{n}(\mu_{kj}).\log_a(\mu_{kj}), \text{ where, } a \text{ is the base of the logarithm.}$$

The *CE* index is computed for values of c greater than 1 and the values of the index are obtained in the range $\left[0, \log_a c\right]$. The closer the value of the index is to 0, the harder is the clustering. Thus the optimal number of clusters is indicated by the minimum value. Like the *PC* index, the *CE* index also suffers from the drawbacks of monotonous dependency on the number of clusters c and of lack of direct connection to any property of the data set, because both the indices do not use the data set itself.

4.7 Experiments

In this section some experiments of pattern recognition using clustering techniques on numeric, image and text data, are presented. As clustering techniques, the K-means and FCM clustering algorithms were used.

4.7.1 Clustering of Numeric Data

In this subsection, the experimental results of clustering of numeric data, obtained using the K-means and FCM clustering algorithms, are presented. For the validation of clustering results obtained by K-means, the validity indices *PBM*, *XB*, *DI*, *SC*, *DB*

(discussed in subsection 4.6.1) were used, and for the validation of clustering results obtained by FCM, the validity indices *PBMF*, *XB*, *PC*, *MPC*, *CE* (discussed in subsection 4.6.2) were used. To illustrate the wide applicability of the clustering algorithms they were tested on a number of benchmark numeric data sets obtained from the famous UCI Machine Learning Repository [81] and other sources. The data sets are now described next.

Benchmark real-life data sets from UCI Machine Learning Repository

Iris data set: This is a very well known data set about three species of the Iris flower, namely, Iris Setosa, Iris Versicolor, and Iris Virginica [82]. This data set is in 4-dimensional space, and has 3 clusters. The total number of observations is 150 with the 3 clusters having 50 instances each. The four features are: *sepal length* in cm, *sepal width* in cm, *petal length* in cm, *petal width* in cm. Out of the 3 clusters, 2 clusters are overlapping (Iris Versicolor, and Iris Virginica), and hence the classification of the data set into the optimal number of clusters (i.e., 3) is a difficult problem with this data set. The data set has often been used as a standard for testing clustering algorithms [77].

Cancer data set: This is the original Breast Cancer data obtained from the University of Wisconsin Hospitals. This data set is in 9-dimensional space, and has 2 classes: benign, malignant. The benign class has 458 observations whereas the malignant class has 241 observations. The two clusters are known to be linearly inseparable i.e. overlapping. The total number of observations is 699 (as of 15 July 1992). The nine features are: *clump thickness*, *uniformity of cell size*, *uniformity of cell shape*, *marginal adhesion*, *single epithelial cell size*, *bare nuclei*, *bland chromatin*, *normal nucleoli*, and *mitoses*.

Wine data set: This is the wine recognition data obtained from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents, namely, *Alcohol*, *Malic acid*, *Ash*, *Alkalinity of ash*, *Magnesium*, *Total phenols*, *Flavanoids*, *Nonflavanoid phenols*,

Proanthocyanins, *Color intensity*, *Hue*, *OD280/OD315 of diluted wines*, and *Proline*; found in each of the three types of wines. The data set is thus in 13-dimensional space. It has 3 clusters. The total number of observations is 178. The first class has 59 observations, the second class has 71 observations and the third class has 48 observations. The three clusters are not linearly separable i.e. the clusters are overlapping.

Other real-life data sets

SODAR1 Data Set: This data set is in 2-dimensional space. The total number of observations is 90 and has 3 clusters with 30 observations each. Each point describes the convective boundary layer height return of backscattered SODAR signal (in meters), measured at the interval of every 2 minutes in the time period from 1200 hours to 1500 hours [83].

SODAR2 Data Set: This data is in 2-dimensional space. The total number of observations is 90 and has 3 clusters with 30 observations each. Each point describes the temperature inversion layer with flat top height return of backscattered SODAR signal (in meters), measured at the interval of every 2 minutes in the time period from 2300 hours to 0200 hours [83].

Other numeric data sets

Data_3_2: This data is in 2-dimensional space and has 3 clusters. The total number of points is 76 [84] [85] [86]. The data set is not provided with any class labels.

Data_5_2: This data is in 2-dimensional space, and has 5 clusters. The total number of points is 250 with 50 points in each cluster. It is sometimes also referred to as AD_5_2 [84] [86] [87]. The clusters in this data set are not very well separated.

Data_6_2: This data is in 2-dimensional space, and has 6 clusters. The total number of points is 300 with 50 points in each cluster [84].

Data_9_2: This data is in 2-dimensional space. The total number of points is 900 with 87, 116, 95, 102, 90, 99, 105, 111 and 95 points in 9 different clusters. It is sometimes also referred to as st900_2_9 [86] [88]. The clusters in this data set are quite fuzzy and not very well separated.

Data_10_2: This data is in 2-dimensional space, and has 10 clusters. The total number of points is 500 with 50 points in each cluster. It is sometimes also referred to as AD_10_2 [56] [84] [86] [87]. The data set is not provided with any class labels. Some of the clusters in this data set are not very well separated.

Data_4_3: This data is in 3-dimensional space, and has 4 clusters. The total number of points is 400 with 100 points in each cluster. It is sometimes also referred to as AD_4_3 [84] [86] [87].

Ruspini Data: The famous Ruspini data set is in 2-dimensional space and it has 4 different clusters. The total number of points is 75 [89]. The data set is not provided with any class labels.

The general format of each line of the data files, for all the above data sets, is:

feature1, feature2, feature3, ..., class label

Each line in a data file represents one pattern or feature vector of the data set where feature1, feature2, feature3 denote the values of different features of the pattern and the optional class label provides the class information of the pattern. Data files of data sets Data_3_2, Data_5_2, Data_6_2, Data_9_2, Data_10_2 and Data_4_3 also contain additional information about the number of observations, number of dimensions and the number of clusters in the very first line. So before applying any clustering algorithm on them, they were preprocessed by the removal of this additional first line. These three pieces of deleted information can be easily obtained from the actual feature vectors and class labels. The data files of Data_3_2, Data_10_2 and Ruspini data sets do not contain any class labels.

The graphical representations of all the numeric data sets considered here for experimentation are now shown next in Figures 4.3 through 4.14.

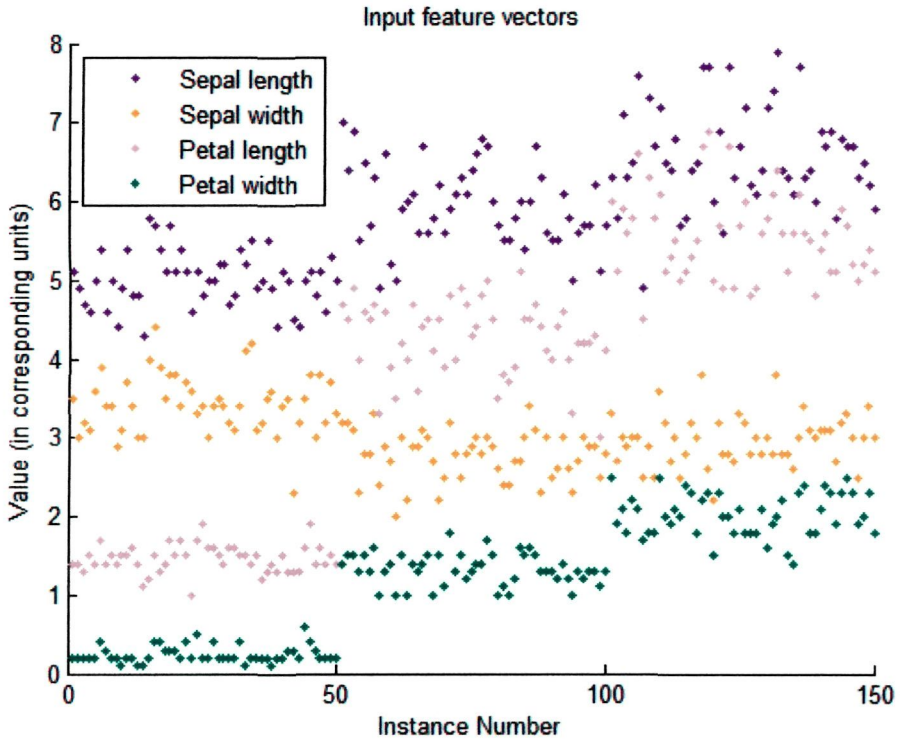


Figure 4.3: Instance number vs. input feature vector values' plot for the Iris data set. $K = 3$, $n = 150$, $d = 4$

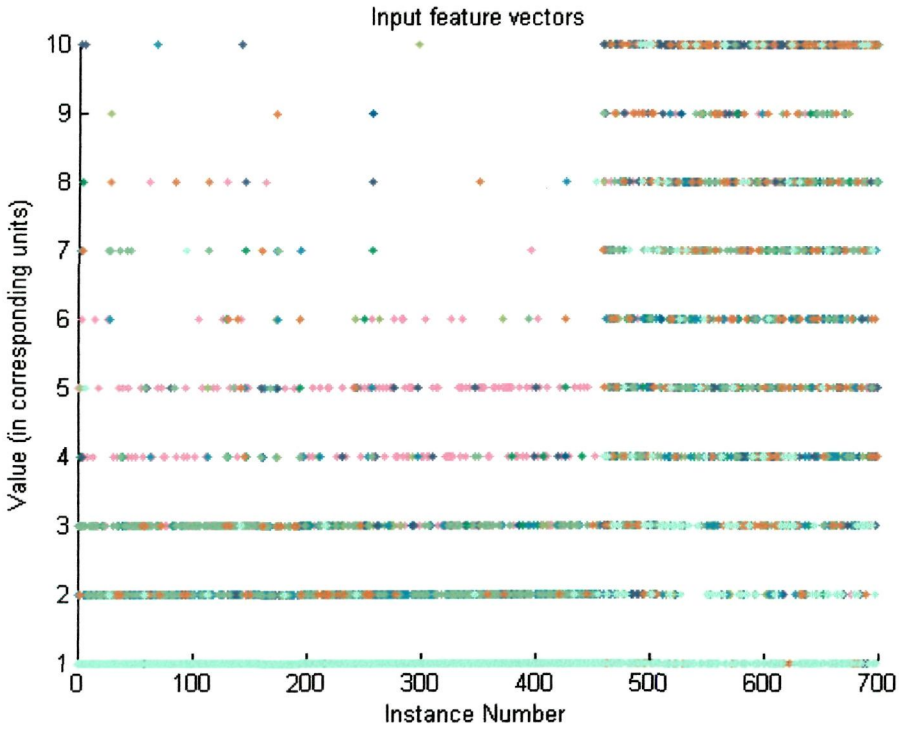


Figure 4.4: Instance number vs. input feature vector values' plot for the Cancer data set. $K = 2$, $n = 699$, $d = 9$

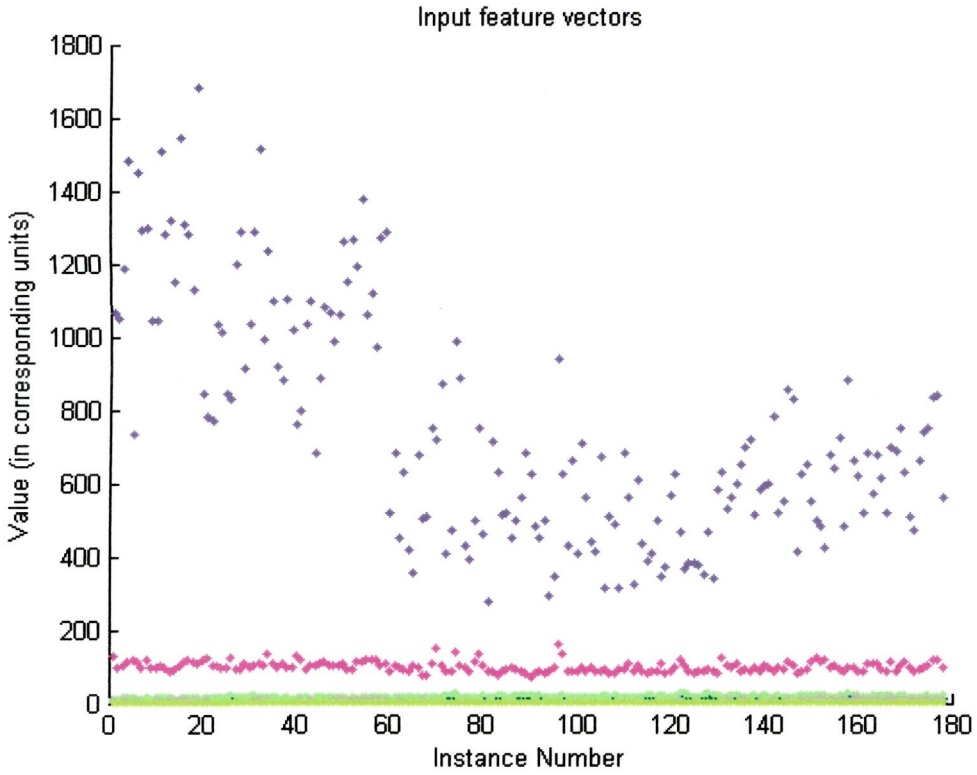


Figure 4.5: Instance number vs. input feature vector values' plot for the Wine data set. $K = 3$, $n = 178$, $d = 13$

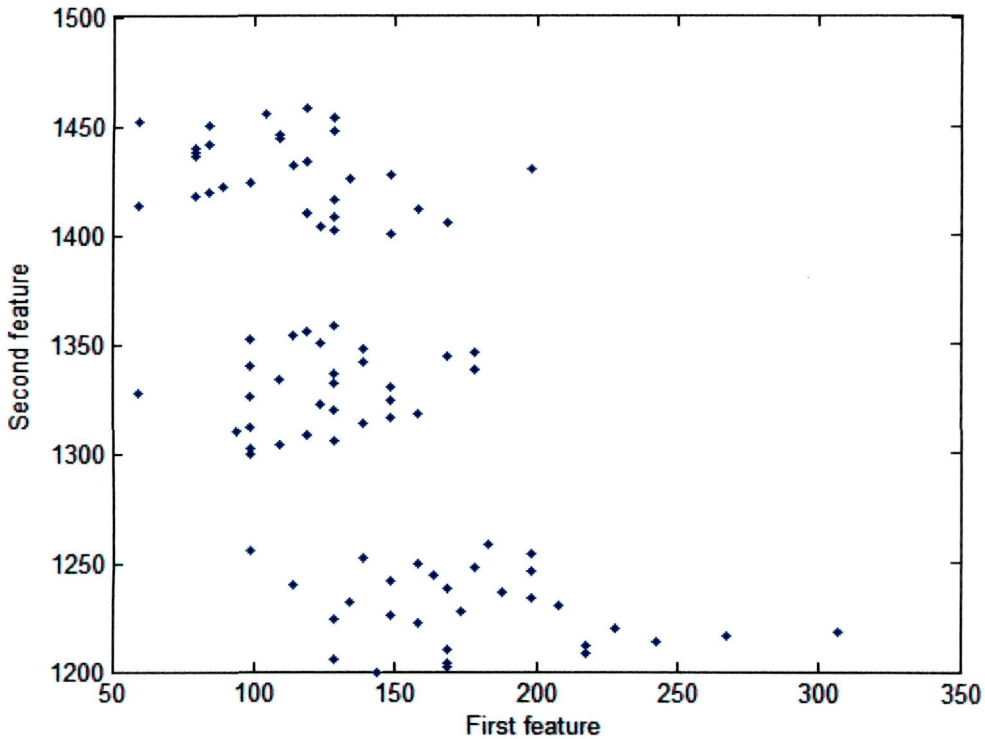


Figure 4.6: Scatter plot for SODAR1 data set. $K = 3$, $n = 90$, $d = 2$

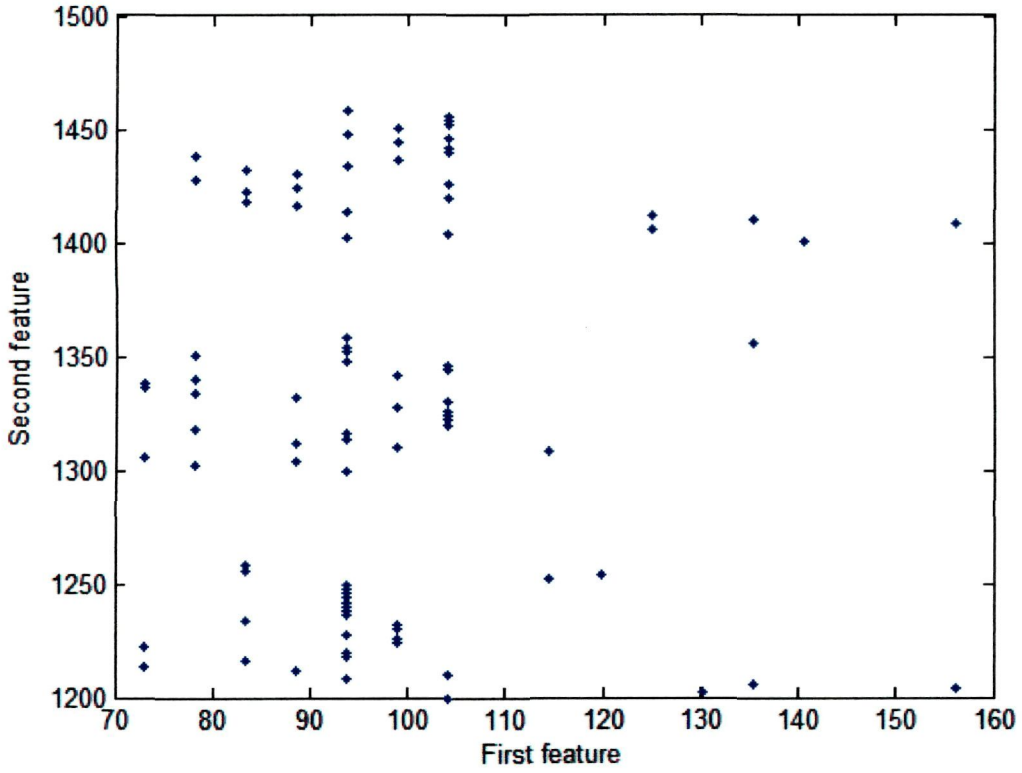


Figure 4.7: Scatter plot for SODAR2 data set. $K = 3$, $n = 90$, $d = 2$

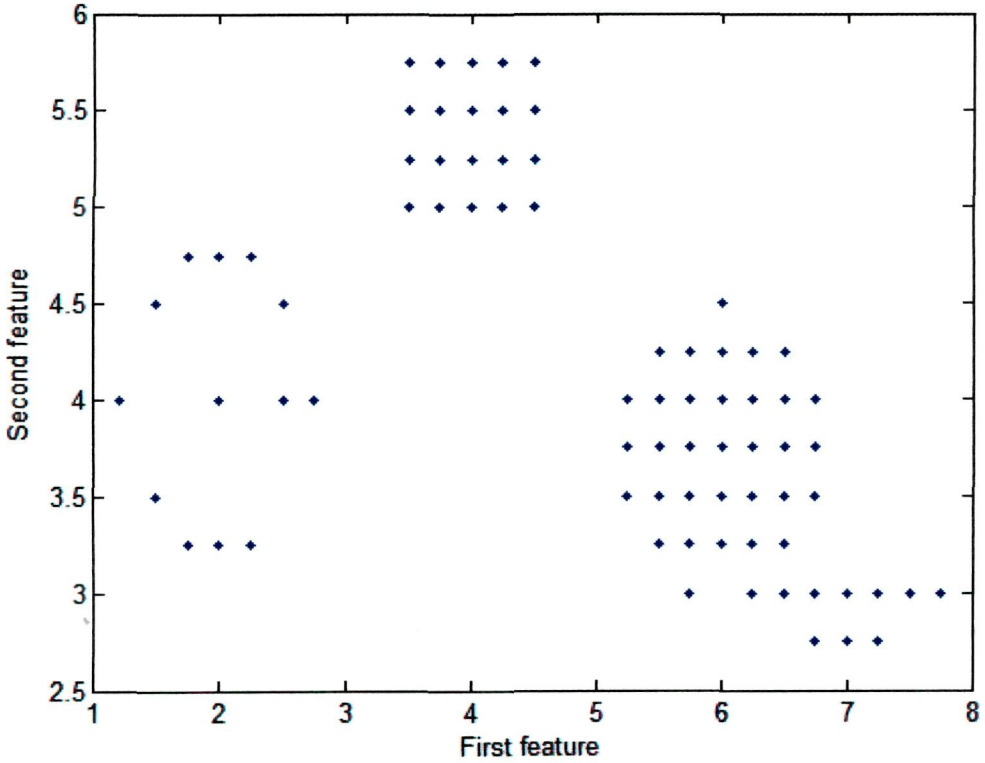


Figure 4.8: Scatter plot for Data_3_2 data set. $K = 3$, $n = 76$, $d = 2$

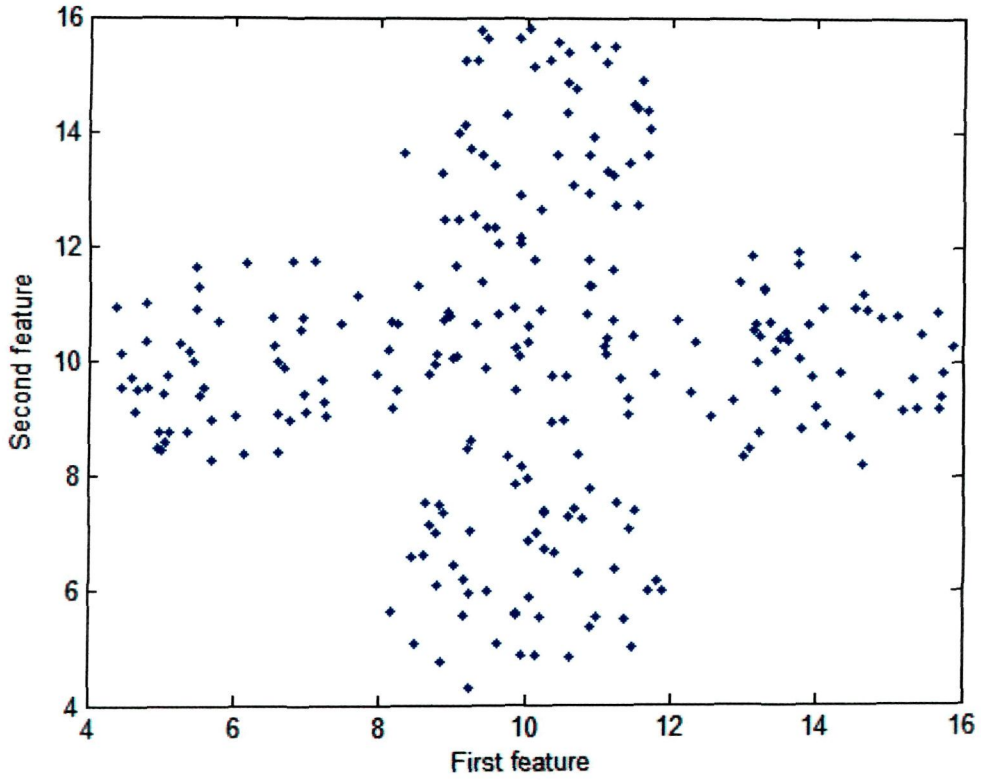


Figure 4.9: Scatter plot for Data_5_2 data set. $K = 5$, $n = 250$, $d = 2$

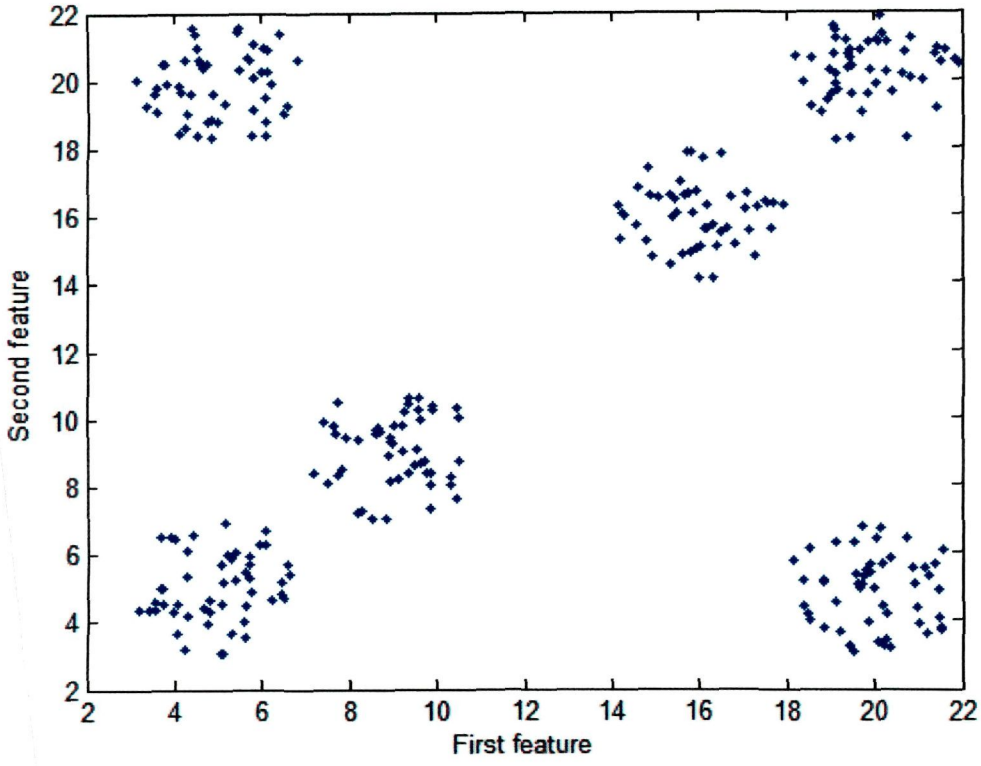


Figure 4.10: Scatter plot for Data_6_2 data set. $K = 6$, $n = 300$, $d = 2$

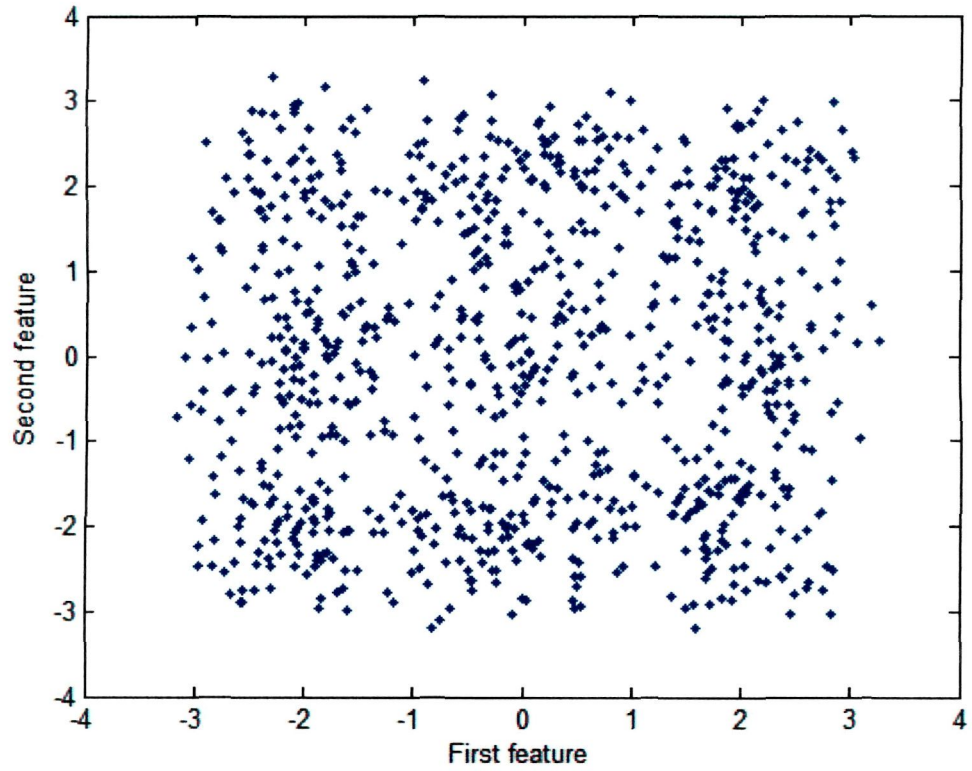


Figure 4.11: Scatter plot for Data_9_2 data set. $K = 9$, $n = 900$, $d = 2$

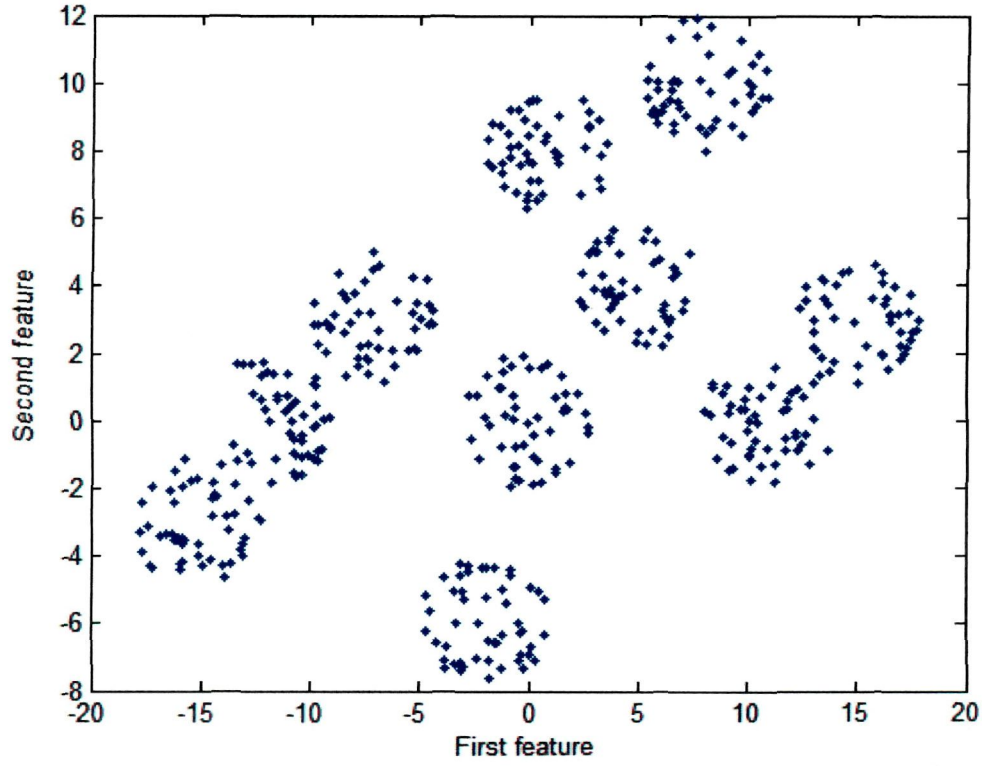


Figure 4.12: Scatter plot for Data_10_2 data set. $K = 10$, $n = 500$, $d = 2$

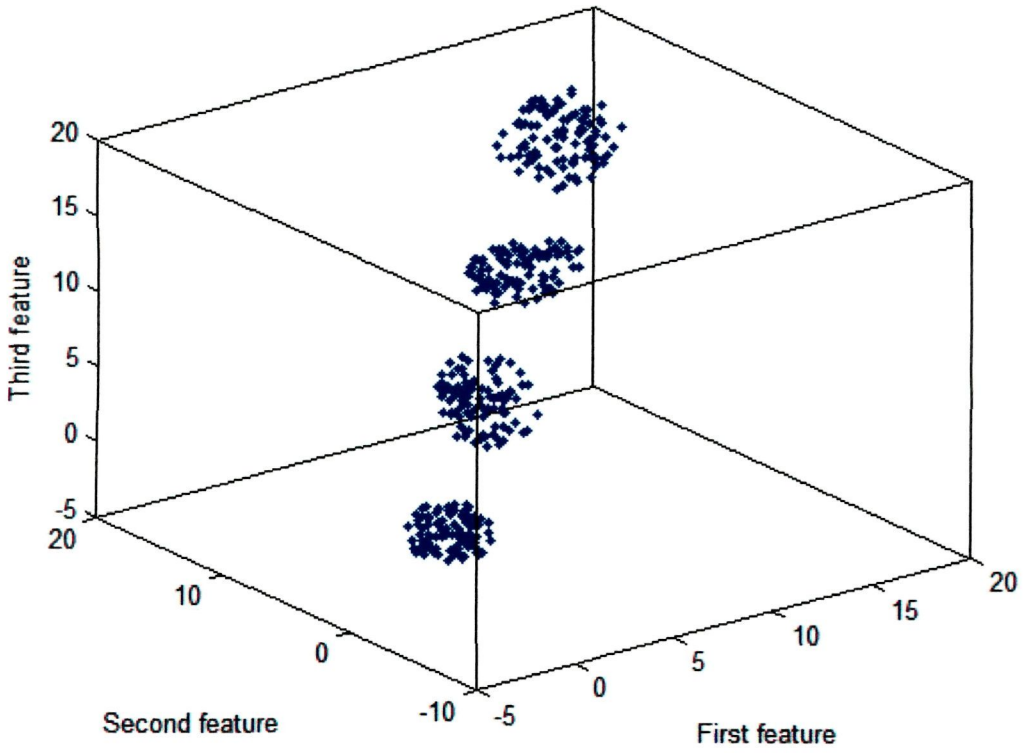


Figure 4.13: Scatter plot for Data_4_3 data set. $K = 4$, $n = 400$, $d = 3$

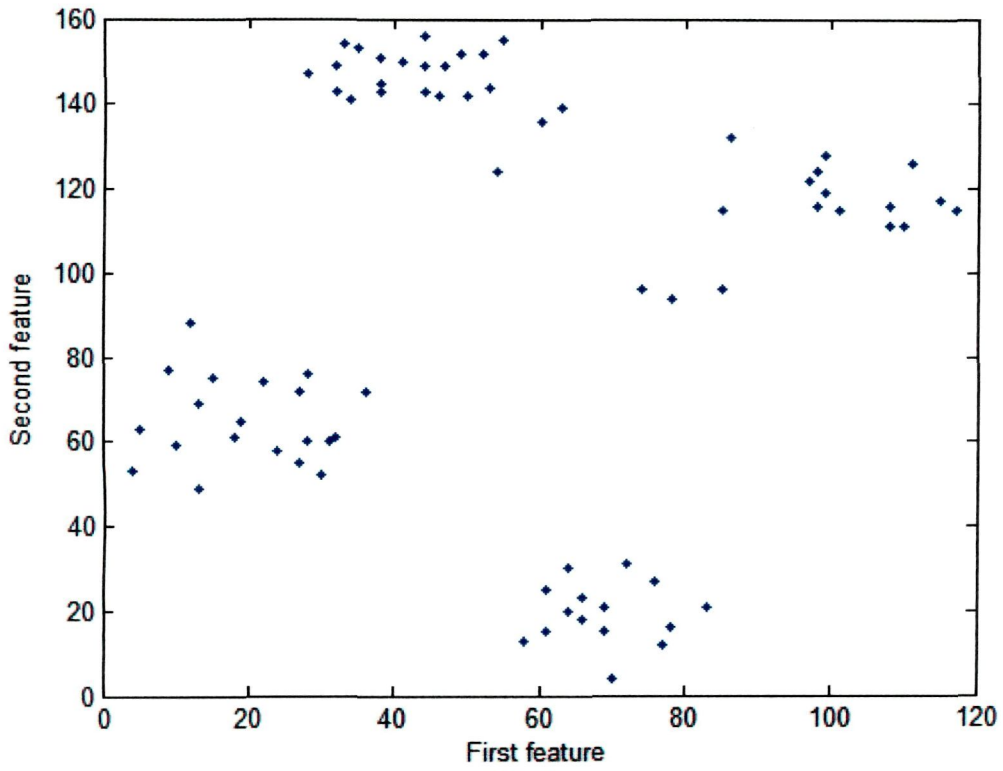


Figure 4.14: Scatter plot for Ruspini data set. $K = 4$, $n = 75$, $d = 2$

Some of the experimental results obtained by the application of the K-means and FCM clustering algorithms on the considered data sets are now presented next in Tables 4.1 - 4.13. For obtaining the results presented in Table 4.1 and Table 4.2, the two algorithms were executed on all the data sets for 50 trials (because of their non-deterministic nature), using the 'Single Clustering' mode (discussed in chapter 8) of the software package PatternWiz. Table 4.1 evaluates the two algorithms based on the similarity of cluster sizes obtained by them compared to the original cluster sizes, and based on the misclassification error percentages in their clustering results. For this evaluation the data sets Iris, Cancer, Wine, SODAR1, Data_5_2 and Data_9_2 were chosen because the clusters in these data sets are not so well separated, i.e. the clusters are overlapping and fuzzy, and so the chances of misclassification error in clustering are more. Moreover for all these data sets, the information about the number of

Table 4.1: Best cluster sizes and lowest misclassification error percentages attained by the K-means and FCM clustering algorithms for the Iris, Cancer, Wine, SODAR1, Data_5_2 and Data_9_2 data sets. For FCM algorithm $m = 1.5$

Data set	Actual no. of clusters	Actual cluster sizes	Best cluster sizes obtained by		Lowest misclassification error percentage	
			K-means	FCM	K-means	FCM
Iris	3	(50, 50, 50)	(38, 50, 62)	(39, 50, 61)	10.6667	11.3333
Cancer	2	(241, 458)	(234, 465)	(231, 468)	4.1488	4.5780
Wine	3	(48, 59, 71)	(47, 62, 69)	(46, 63, 69)	52.2472	52.8090
SODAR1	3	(30, 30, 30)	(29, 30, 31)	(29, 30, 31)	1.1111	1.1111
Data_5_2	5	(50, 50, 50, 50, 50)	(48, 49, 50, 51, 52)	(43, 48, 50, 53, 56)	1.6000	4.8000
Data_9_2	9	(87, 90, 95, 95, 99, 102, 105, 111, 116)	(83, 93, 93, 95, 95, 103, 105, 114, 119)	(84, 92, 92, 95, 96, 104, 106, 113, 118)	7.7778	7.5556

clusters, the sizes of individual clusters in each data set and the class labels of each observation in each data set, are also available. The values in Table 4.1 are the best values obtained from the 50 trials. From Table 4.1 it may seem that the performance of K-means algorithm is better than the FCM algorithm. However, it should be noted that the FCM algorithm is very much dependant on the value of the fuzzifier m which affects the fuzzy belongingness of the observations in the different clusters. As already mentioned in section 4.5, there is no theoretical basis for an optimal choice for the value of m, and the choice of a good value of m is dependant upon the problem under consideration. Thus in the case of Iris, Cancer and SODAR1 data sets, when the value of m was chosen as 12.5, the corresponding cluster sizes and misclassification error percentages obtained were (49, 50, 51) and 7.3333 %; (238, 461) and 3.5765 %; and (30, 30, 30) and 0 % respectively, which are much better than those for K-means. Thus the selection of a good value of m is a tricky task for the FCM algorithm. In general, the value of m is chosen in the interval (1, 2.5] [90] [77]. As such, in this thesis, for all experiments on FCM, m was taken as 1.5.

In Table 4.2 the numbers of iterations and average execution times of the first best trials with optimal clustering (indicated by the lowest value of misclassification error percentage or highest value of the *PBM* index), encountered in a series of 50 trials of each of the algorithms on each of the data sets, are presented. The total execution times of all trials are also presented. The average execution time for a '*first best trial*' is calculated using the following:

$$Average\ execution\ time = \frac{Total\ execution\ time\ of\ all\ trials \times no.of\ iterations\ in\ first\ best\ trial}{Total\ no.of\ iterations\ in\ all\ trials}$$

From Table 4.2 it can be inferred that FCM is computationally costlier than K-means.

Table 4.3 shows the objective function values obtained by K-means and FCM for the Iris and Cancer data sets for varying values for number of clusters (2 to 10). In general, the value of the objective function decreases with increase in the number of clusters. The function has the maximum value when all the features are lumped together into a single cluster and has the minimum value of 0 when the number of clusters equals the total number of observations n [91] [92]. This decreasing trend in the values of the objective function can be easily verified from Table 4.3. It can also be noted that FCM yields comparatively smaller values for the fuzzy *SSE* objective

function compared to the hard *SSE* objective function for K-means.

Table 4.2: Iteration counts and execution times (in seconds) for the K-means and FCM clustering algorithms. For FCM algorithm $m = 1.5$

Data set	No. of clusters	No. of iterations (First best trial)	Average execution time (First best trial)	Total execution time of 50 trials
Algorithm: K-means				
Iris	3	7	0.0014647	0.081602
Cancer	2	6	0.0094869	0.39529
Wine	3	8	0.0031219	0.1518
SODAR1	3	8	0.0018439	0.061308
SODAR2	3	4	0.00094103	0.052462
Data_3_2	3	4	0.00092164	0.05023
Data_5_2	5	8	0.0021571	0.13428
Data_6_2	6	6	0.0027303	0.1397
Data_9_2	9	16	0.016875	1.042
Data_10_2	10	9	0.0064658	0.44327
Data_4_3	4	6	0.002571	0.11527
Ruspini	4	4	0.0011199	0.054594
Algorithm: FCM				
Iris	3	27	0.011445	0.57225
Cancer	2	11	0.014367	0.77189
Wine	3	26	0.013246	0.78153
SODAR1	3	14	0.0047126	0.21005
SODAR2	3	10	0.0036931	0.17542
Data_3_2	3	11	0.0028061	0.28596
Data_5_2	5	25	0.023696	1.3952
Data_6_2	6	11	0.014881	1.6639
Data_9_2	9	38	0.2169	11.8722
Data_10_2	10	27	0.095032	6.1877
Data_4_3	4	9	0.011454	0.81325
Ruspini	4	7	0.0029079	0.25091

Table 4.4 shows a comparison of the actual number of clusters present in the data sets considered, to the number of clusters obtained by the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, using the K-means algorithm. Tables 4.5 through 4.8 provide the actual values of the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, obtained using the K-means clustering algorithm for $K = 2, 3, \ldots, 10$ on all the data sets considered.

Table 4.3: Comparison of objective function values obtained by the K-means and FCM algorithms for the Iris and Cancer data sets for varying number of clusters. For FCM algorithm $m = 1.5$

No. of clusters	Objective function values obtained by K-means		Objective function values obtained by FCM	
	Iris	Cancer	Iris	Cancer
2	152.348	19685.2463	146.2985	18434.8385
3	78.8514	16604.7832	74.3822	14013.8382
4	71.4452	15487.2268	53.736	12021.4487
5	70.8908	13987.1797	45.8452	10597.0034
6	42.4215	13144.7367	38.9158	9487.6806
7	40.5461	12397.126	31.1802	8707.2165
8	36.4972	12901.7124	26.8487	8106.3123
9	31.4269	11591.4448	24.5957	7598.1752
10	33.6003	11678.6978	22.6384	7189.015

Table 4.4: Number of clusters obtained by the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, using the K-means algorithm (entries in bold face indicate the correct determination of number of clusters in a data set by the respective indices)

Data set	Actual no. of clusters	No. of clusters obtained				
		PBM	*XB*	*DI*	*SC*	*DB*
Iris	3	**3**	2	4	9	2
Cancer	2	**2**	**2**	**2**	8	**2**
Wine	3	6	2	4	10	2
SODAR1	3	**3**	**3**	**3**	10	**3**
SODAR2	3	**3**	**3**	**3**	9	**3**
Data_3_2	3	4	**3**	**3**	10	**3**
Data_5_2	5	**5**	4	7	10	4
Data_6_2	6	**6**	4	4	9	4
Data_9_2	9	**9**	**9**	**9**	10	**9**
Data_10_2	10	**10**	**10**	5	**10**	2
Data_4_3	4	**4**	**4**	**4**	9	2
Ruspini	4	**4**	**4**	**4**	10	**4**

Table 4.5: Values of the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, for K = 2, 3, ..., 10 for the Iris, Cancer and Wine data sets using the K-means algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Iris				
	PBM	***XB***	***DI***	***SC***	***DB***
2	19.9233	**0.0658002**	0.0765063	0.117585	**0.237183**
3	**25.1754**	0.162755	0.0988074	0.0747351	0.384907
4	20.7486	0.231735	**0.100844**	0.0617575	0.555922
5	11.8664	1.18882	0.0373457	0.0338432	0.659995
6	12.4381	0.818217	0.0482243	0.0301706	0.707144
7	10.7152	0.712178	0.0522708	0.0310004	0.736929
8	9.21186	0.63098	0.0739221	0.0359406	0.752915
9	10.3237	0.529041	0.058621	**0.0237124**	0.840559
10	8.7435	0.528155	0.0973585	0.0298985	0.832532
	Data set: Cancer				
2	**142.822**	**0.148954**	**0.155446**	0.394815	**0.40595**
3	103.126	0.436167	0.144338	0.470394	0.946671
4	75.9747	0.384287	0.144905	0.475841	1.01249
5	61.5145	2.11989	0.0512316	0.288958	1.22385
6	45.927	2.29214	0.0558146	0.327916	1.28851
7	35.0487	6.3263	0.0558146	0.254836	1.38493
8	27.6969	6.58208	0.0558146	**0.239468**	1.57122
9	24.3976	4.752	0.0585206	0.261513	1.40588
10	23.1193	4.33901	0.0581238	0.248454	1.48279
	Data set: Wine				
2	325954	**0.0743498**	0.0228207	0.165497	**0.285064**
3	473716	0.182226	0.0162604	0.0798641	0.406442
4	640729	0.124934	**0.0339871**	0.0414293	0.379872
5	677339	0.265377	0.0161491	0.0282448	0.441874
6	**759849**	0.207747	0.0280939	0.0196609	0.449514
7	672397	0.258884	0.0190524	0.015582	0.403681
8	640951	0.456965	0.0216495	0.0122152	0.427851
9	582632	0.667629	0.0185358	0.0111505	0.485289
10	530515	0.886174	0.01341	**0.00906197**	0.479391

Table 4.6: Values of the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, for K = 2, 3, ..., 10 for the SODAR1, SODAR2 and Data_3_2 data sets using the K-means algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: SODAR1				
	PBM	***XB***	***DI***	***SC***	***DB***
2	14038	0.151336	0.0344976	0.302672	0.387663
3	**31105.1**	**0.14322**	**0.112338**	0.109346	**0.36403**
4	27603.7	0.195303	0.0719655	0.0677899	0.468386
5	23446.3	0.490971	0.090693	0.0517513	0.538331
6	21379.6	0.516166	0.0840505	0.0383398	0.554853
7	12565.1	0.609603	0.0648082	0.0499664	0.625845
8	19883.9	0.339092	0.104914	0.0295897	0.645206
9	12909.9	0.603838	0.0880515	0.0353606	0.602753
10	12782	0.992903	0.0880515	**0.0231405**	0.572244
	Data set: SODAR2				
2	16053.6	0.102822	0.0289439	0.205644	0.320648
3	**56229.6**	**0.0569907**	**0.476905**	0.0487374	**0.311482**
4	38931.4	0.54479	0.123007	0.0448373	0.467405
5	26739.6	0.80103	0.0875747	0.0417983	0.579622
6	38563.1	0.401533	0.0905689	0.0210194	0.472651
7	34684.1	0.31038	0.0304776	0.0173961	0.517548
8	21785.8	1.30911	0.0440831	0.0181143	0.574191
9	32497.2	0.475229	0.0850429	**0.0117739**	0.511788
10	30933.5	0.289406	0.0467004	0.0128716	0.568759
	Data set: Data_3_2				
2	9.80666	0.100759	0.333333	0.214972	0.317832
3	16.0331	**0.0832373**	**0.415227**	0.0585122	**0.303267**
4	**18.1554**	0.218963	0.158114	0.0395386	0.412564
5	13.1092	0.707726	0.158114	0.0339857	0.517189
6	13.8253	0.369541	0.158114	0.0281657	0.494736
7	12.4554	0.36538	0.185695	0.021115	0.562219
8	10.0363	0.696626	0.16129	0.0195892	0.657952
9	11.1287	0.327264	0.193746	0.0153449	0.625656
10	10.3158	0.299878	0.193746	**0.014048**	0.640545

Table 4.7: Values of the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, for K = 2, 3, ..., 10 for the Data_5_2, Data_6_2 and Data_9_2 data sets using the K-means algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Data_5_2				
	PBM	***XB***	***DI***	***SC***	***DB***
2	10.1531	0.341318	0.0154519	0.682636	0.584221
3	8.35199	0.233192	0.0320038	0.290604	0.595959
4	15.4613	**0.13752**	0.0644697	0.126212	**0.499374**
5	**18.7947**	0.137765	0.0259231	0.0870959	0.567185
6	15.2806	0.240247	0.0118684	0.0817806	0.605405
7	13.1366	0.573653	**0.0887616**	0.0653362	0.746764
8	12.5845	0.5203	0.0387182	0.056068	0.69231
9	11.2389	0.432838	0.0670984	0.0515245	0.768607
10	12.8788	0.407901	0.057631	**0.0470591**	0.820815
	Data set: Data_6_2				
2	76.2417	0.285924	0.3318	0.571848	0.534715
3	103.703	0.151669	0.3318	0.142703	0.36152
4	360.856	**0.0434775**	**0.657568**	0.039085	**0.221265**
5	399.975	0.140488	0.219233	0.0219008	0.258787
6	**626.418**	0.0549314	0.515015	0.0117286	0.27691
7	503.456	0.55416	0.09683	0.0111395	0.393821
8	451.743	0.568062	0.0809665	0.00878719	0.470676
9	405.613	0.521981	0.0809665	**0.00779742**	0.539387
10	380.223	0.580432	0.0691405	0.00779945	0.625711
	Data set: Data_9_2				
2	3.95366	0.39126	0.00579991	0.782513	0.625505
3	3.58962	0.240239	0.016931	0.315337	0.619418
4	4.75711	0.180899	0.0115062	0.169923	0.578885
5	4.32354	0.269542	0.0172067	0.144976	0.700189
6	4.50529	0.260109	0.00593429	0.104958	0.730505
7	4.563	0.215312	0.0174149	0.0801909	0.726383
8	4.45299	0.166231	0.0205335	0.0653071	0.606485
9	**4.97797**	**0.144973**	**0.0383366**	0.0514353	**0.560781**
10	4.37909	0.28042	0.0279794	**0.0506244**	0.64241

Table 4.8: Values of the *PBM*, *XB*, *DI*, *SC* and *DB* validity indices, for K = 2, 3, ..., 10 for the Data_10_2, Data_4_3 and Ruspini data sets using the K-means algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Data_10_2				
	PBM	***XB***	***DI***	***SC***	***DB***
2	155.421	0.159742	0.0270727	0.318775	**0.399191**
3	192.331	0.26049	0.0190878	0.161286	0.417247
4	249.283	0.148284	0.0351633	0.0987887	0.512975
5	269.827	0.217147	**0.0662246**	0.0605545	0.516849
6	233.684	0.246841	0.051412	0.0458703	0.45471
7	214.875	0.352033	0.0397119	0.0396538	0.520334
8	216.269	0.273105	0.0400711	0.0302446	0.523658
9	285.089	0.137057	0.0410937	0.0217647	0.477216
10	**300.15**	**0.136089**	0.0656321	**0.0180894**	0.458114
	Data set: Data_4_3				
2	274.903	0.0755651	0.326609	0.15113	**0.274846**
3	388.445	0.152236	0.0415338	0.0689785	0.292048
4	**941.875**	**0.0520016**	**0.648233**	0.0244999	0.286967
5	180.142	1.715	0.0415338	0.0420537	0.657471
6	506.692	1.32732	0.0365537	0.0204898	0.689287
7	98.8002	2.29986	0.0415338	0.0329479	0.697946
8	358.767	0.811093	0.0338636	0.0203799	0.799155
9	344.214	0.928837	0.0318273	**0.0113638**	0.805329
10	284.893	1.0361	0.0584998	0.0142051	0.866336
	Data set: Ruspini				
2	5848.23	0.14342	0.440293	0.287543	0.379108
3	7018.97	0.173883	0.235521	0.100126	0.346559
4	**24005**	**0.0438631**	**0.504716**	0.029182	**0.265825**
5	19170.7	0.558863	0.0629802	0.0241887	0.401386
6	15592.3	0.489	0.0629802	0.0214562	0.527012
7	14248.4	0.351515	0.0848528	0.0187636	0.536729
8	13156.6	0.305388	0.0768978	0.0133081	0.533993
9	12087	0.571285	0.0768978	0.0115953	0.619698
10	11699	0.470023	0.0967239	**0.00995556**	0.600024

Table 4.9 shows a comparison of the actual number of clusters present in all the benchmark data sets considered to the number of clusters obtained by the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, using the FCM clustering algorithm. Tables 4.10 - 4.13 provide the actual values of the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices obtained using the FCM clustering algorithm, for $c = 2, 3, \ldots, 10$ for all the data sets.

For obtaining the results presented in Tables 4.3, 4.5, 4.6, 4.7, 4.8, 4.10, 4.11, 4.12, 4.13, the 'Multiple Clustering' mode (described in chapter 8) of the software package PatternWiz was used and the best values obtained are presented in the tables.

Table 4.9: Number of clusters obtained by the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, using the FCM algorithm (entries in bold face indicate the correct determination of number of clusters in a data set by the respective indices). For FCM algorithm $m = 1.5$

Data set	Actual no. of clusters	No. of clusters obtained				
		PBMF	*XB*	*PC*	*MPC*	*CE*
Iris	3	**3**	2	2	2	2
Cancer	2	**2**	**2**	**2**	**2**	**2**
Wine	3	7	2	2	2	2
SODAR1	3	**3**	2	**3**	**3**	**3**
SODAR2	3	**3**	**3**	**3**	**3**	**3**
Data_3_2	3	4	**3**	**3**	**3**	**3**
Data_5_2	5	**5**	4	**5**	**5**	**5**
Data_6_2	6	**6**	4	**6**	**6**	**6**
Data_9_2	9	**9**	**9**	2	**9**	2
Data_10_2	10	**10**	**10**	**10**	**10**	2
Data_4_3	4	**4**	**4**	**4**	**4**	**4**
Ruspini	4	**4**	**4**	**4**	**4**	**4**

Table 4.10: Values of the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, for $c = 2, 3, \ldots, 10$ for the Iris, Cancer and Wine data sets using the FCM algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Iris				
	PBMF	***XB***	***PC***	***MPC***	***CE***
2	21.8314	**0.0620203**	**0.979713**	**0.959425**	**0.0591903**
3	**28.2204**	0.156352	0.949132	0.923698	0.145878
4	24.7414	0.219783	0.928866	0.905155	0.207842
5	20.7735	0.507059	0.901455	0.876819	0.283352
6	18.5807	0.394045	0.899112	0.878935	0.303569
7	17.7894	0.63782	0.884802	0.865602	0.343091
8	15.941	0.548355	0.862657	0.843037	0.406182
9	14.8778	0.466376	0.861481	0.844167	0.417872
10	13.8462	0.694578	0.852829	0.836477	0.446799
	Data set: Cancer				
2	**167.963**	**0.135267**	**0.958273**	**0.916547**	**0.116559**
3	143.575	0.524358	0.903204	0.854806	0.265562
4	120.485	2.11283	0.821695	0.762261	0.495875
5	104.341	1.88657	0.795454	0.744317	0.580622
6	93.2341	1.72108	0.772855	0.727426	0.657373
7	84.0972	2.27286	0.756395	0.715794	0.718132
8	74.8518	1.52496	0.747244	0.711136	0.761648
9	66.6795	1.4533	0.742303	0.710091	0.792195
10	64.6411	3.8884	0.697791	0.664212	0.929737
	Data set: Wine				
2	372177	**0.0684558**	**0.969464**	**0.938928**	**0.0839274**
3	553537	0.16076	0.949492	0.924237	0.144544
4	788064	0.119303	0.945154	0.926872	0.153577
5	897532	0.0969246	0.944545	0.930681	0.158683
6	1.00171e+6	0.097282	0.942468	0.930962	0.165287
7	**1.34749e+6**	0.111672	0.943863	0.934507	0.162565
8	1.29341e+6	0.14526	0.941334	0.932953	0.171758
9	1.18169e+6	0.123471	0.941245	0.933901	0.173618
10	1.30725e+6	0.110358	0.941135	0.934595	0.173518

Table 4.11: Values of the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, for c = 2, 3, ..., 10 for the SODAR1, SODAR2 and Data_3_2 data sets using the FCM algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: SODAR1				
	PBMF	***XB***	***PC***	***MPC***	***CE***
2	19402.5	**0.11969**	0.939358	0.878717	0.161218
3	**34745.1**	0.137554	**0.9639**	**0.94585**	**0.115619**
4	32973.3	0.176459	0.950936	0.934582	0.154716
5	24010.8	0.48083	0.91346	0.891826	0.261289
6	24976.4	0.373252	0.904747	0.885696	0.283045
7	25418	0.362437	0.903734	0.887689	0.293598
8	23302.9	0.327553	0.907729	0.894547	0.290425
9	23097.8	0.375463	0.893127	0.879767	0.325711
10	22393.4	0.253546	0.897694	0.886326	0.315037
	Data set: SODAR2				
2	21334.6	0.0870571	0.939116	0.878232	0.159847
3	**58914.4**	**0.0560778**	**0.986798**	**0.980196**	**0.0473207**
4	45523.6	0.187689	0.974161	0.965547	0.0854064
5	39065.3	0.392895	0.949663	0.937079	0.154049
6	31187.3	0.533354	0.940988	0.929186	0.185466
7	45525.2	0.283838	0.932241	0.920948	0.201596
8	41183.4	0.311369	0.92649	0.915988	0.22212
9	41530.4	0.494951	0.916756	0.906351	0.248813
10	38569.2	0.256546	0.921322	0.91258	0.237547
	Data set: Data_3_2				
2	10.2777	0.0991942	0.980581	0.961163	0.0655075
3	16.6513	**0.0820799**	**0.986119**	**0.979178**	**0.0521428**
4	**20.3442**	0.198198	0.952736	0.936981	0.140287
5	18.5852	0.408041	0.929793	0.912241	0.206435
6	16.5135	0.327192	0.91689	0.900268	0.240078
7	14.6465	0.447889	0.902874	0.886687	0.290466
8	12.914	0.408889	0.896624	0.881857	0.311045
9	13.6835	0.297675	0.887109	0.872998	0.332147
10	14.1117	0.254865	0.886583	0.873981	0.334677

Table 4.12: Values of the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, for c = 2, 3, ..., 10 for the Data_5_2, Data_6_2 and Data_9_2 data sets using the FCM algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Data_5_2				
	PBMF	***XB***	***PC***	***MPC***	***CE***
2	9.66792	0.427427	0.892912	0.785824	0.290434
3	11.1323	0.199091	0.881241	0.821862	0.339221
4	20.6979	**0.11544**	0.917881	0.890508	0.255557
5	**22.9332**	0.132255	**0.925252**	**0.906565**	**0.238663**
6	21.1297	0.337375	0.900893	0.881071	0.309039
7	19.0783	0.266367	0.888878	0.870358	0.346722
8	17.9805	0.350945	0.877278	0.859746	0.385514
9	18.1967	0.312778	0.867334	0.85075	0.410412
10	18.1206	0.308445	0.864894	0.849882	0.420045
	Data set: Data_6_2				
2	90.9368	0.307837	0.921979	0.843957	0.210224
3	113.558	0.136096	0.936966	0.905449	0.180614
4	372.346	**0.0429946**	0.986684	0.982245	0.0545747
5	429.522	0.130485	0.986435	0.983044	0.053018
6	**638.347**	0.0543478	**0.992801**	**0.991361**	**0.0288779**
7	527.582	0.572068	0.978099	0.974448	0.0701231
8	492.956	0.550921	0.960657	0.955037	0.115773
9	491.928	0.489647	0.947522	0.940963	0.152752
10	390.247	0.437004	0.948617	0.942908	0.153619
	Data set: Data_9_2				
2	4.97341	0.35803	**0.894294**	0.788587	**0.280409**
3	4.81928	0.191838	0.880133	0.8202	0.339713
4	6.70596	0.152081	0.874517	0.83269	0.366763
5	6.72629	0.237343	0.856118	0.820148	0.429569
6	6.15833	0.176372	0.861123	0.833348	0.42562
7	6.00433	0.18247	0.861442	0.838349	0.430944
8	6.07322	0.147432	0.871677	0.853345	0.406284
9	**6.78326**	**0.120628**	0.885222	**0.870875**	0.367012
10	6.24479	0.309694	0.877447	0.86383	0.393388

Table 4.13: Values of the *PBMF*, *XB*, *PC*, *MPC* and *CE* validity indices, for $c = 2, 3, \ldots, 10$ for the Data_10_2, Data_4_3 and Ruspini data sets using the FCM algorithm (entries in bold face indicate the optimal values for the respective indices)

No. of clusters	Data set: Data_10_2				
	PBMF	***XB***	***PC***	***MPC***	***CE***
2	187.067	0.144337	0.941036	0.882072	**0.163118**
3	282.279	0.219259	0.899258	0.848887	0.280982
4	298.683	0.135128	0.928515	0.904687	0.221545
5	256.374	0.22928	0.908961	0.886202	0.27871
6	297.758	0.187855	0.914709	0.897651	0.26662
7	333.641	0.194382	0.908737	0.893527	0.284127
8	343.843	0.149256	0.917017	0.905162	0.261649
9	260.966	0.643732	0.907537	0.89598	0.295511
10	**345.917**	**0.12599**	**0.950802**	**0.945336**	0.167801
	Data set: Data_4_3				
2	290.283	0.0735712	0.980701	0.961401	0.0648827
3	598.351	0.0838747	0.934147	0.90122	0.181087
4	**960.79**	**0.051736**	**0.991145**	**0.988193**	**0.037491**
5	720.063	0.646528	0.95929	0.949113	0.125813
6	581.335	1.4417	0.919534	0.90344	0.230634
7	575.107	1.2004	0.884122	0.864809	0.317336
8	333.336	0.768899	0.940963	0.932529	0.198867
9	466.378	0.948232	0.853353	0.835022	0.406954
10	413.704	0.893127	0.839023	0.821137	0.453226
	Data set: Ruspini				
2	6280.37	0.140302	0.967713	0.935427	0.103642
3	7675.75	0.167926	0.966268	0.949402	0.114036
4	**25196.7**	**0.0426941**	**0.988394**	**0.984525**	**0.0429053**
5	20022.5	0.135182	0.982153	0.977692	0.0603392
6	17978.9	0.396504	0.960694	0.952833	0.118587
7	15420.2	0.365011	0.95686	0.949671	0.132029
8	13948.6	0.569625	0.937847	0.928968	0.182045
9	15160.2	0.435647	0.917591	0.90729	0.235139
10	10462.6	0.587345	0.938273	0.931415	0.191991

Results presented in Tables 4.4 – 4.8 show that the best validation result for crisp clustering is obtained by the *PBM* validity index among all the tested validity indices. Other than the *PBM* validity index, the *XB* and *DI* validity indices also show good performances. The *DB* validity index also shows somewhat satisfactory performance for some of the data sets but the *SC* validity index shows poor performance for almost all the data sets. Results presented in Tables 4.9 – 4.13 show that the best validation result for fuzzy clustering is obtained by the *PBMF* validity index among all the tested validity indices. Other than *PBMF* index the *MPC*, *PC* and *CE* validity indices also depict good performances in that order, but the *XB* index does not perform much well. For *CE* index the natural logarithm (logarithm to the base $e \approx 2.71828183$) was used.

By studying all the Tables 4.4 – 4.13 it can be inferred that both the hard and fuzzy versions of the *PBM* validity index in general outperform all the other validity indices for almost all the data sets considered. This reinforces the results presented by Maulik and Bandyopadhyay in [56] and Pakhira et al. in [68].

4.7.2 Image Clustering

The classification of a digital image into distinct regions of interest is a fundamental step in computer vision. Technically, it is called as image segmentation. The goal of image segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. It is typically used to locate meaningful objects in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share some common visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture [93].

Clustering is a key tool in image segmentation [94]. To demonstrate the effectiveness of clustering in pattern recognition on images, experiments on image clustering were conducted using the K-means and FCM clustering algorithms on several well known natural images. The results of some of the experiments are presented in this subsection. To begin with the presentation of results, the natural

images used in the experiments are first presented:

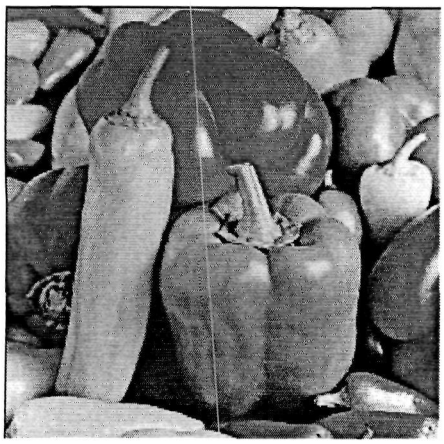


Figure 4.15: ***Peppers image.*** **Courtesy of the Signal and Image Processing Institute at the University of Southern California. Dimension of image: 256×256 pixels**

Figure 4.16: ***Brandy rose image.*** **Copyright photo courtesy of Toni Lankerd, 18347 Woodland Ridge Dr. Apt #7, Spring Lake, MI 49456, U.S.A. (tlankerd@charter.net). Dimension of image: 200×287 pixels**

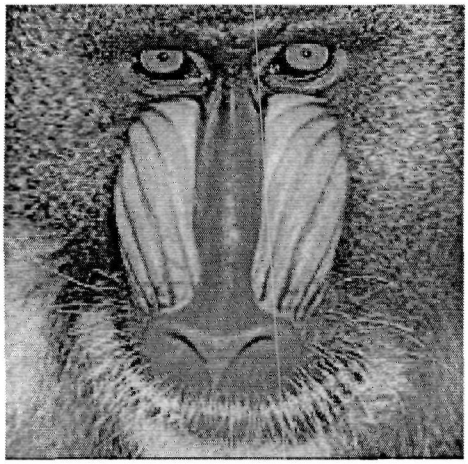


Figure 4.17: ***Baboon image.*** **Courtesy of the Signal and Image Processing Institute at the University of Southern California. Dimension of image: 200×200 pixels**

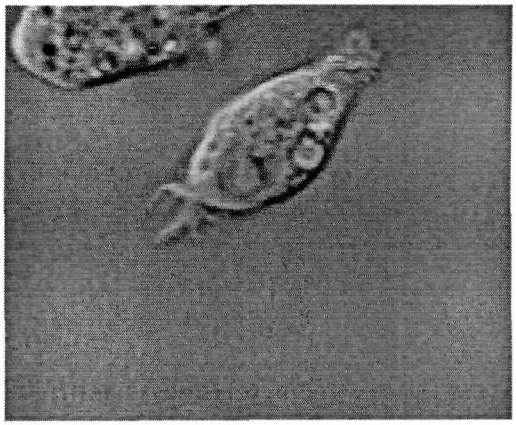


Figure 4.18: ***Cell image.*** **Courtesy of Alan W. Partin, Johns Hopkins University. Dimension of image: 191×159 pixels**

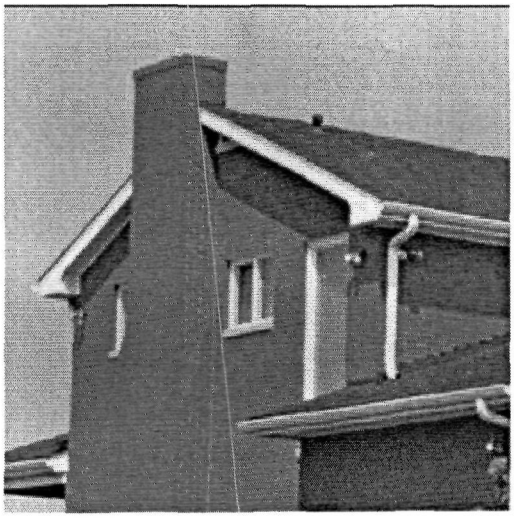


Figure 4.19: *House image.*
Courtesy of the Signal and Image Processing Institute at the University of Southern California.
Dimension of image: 256×256 pixels

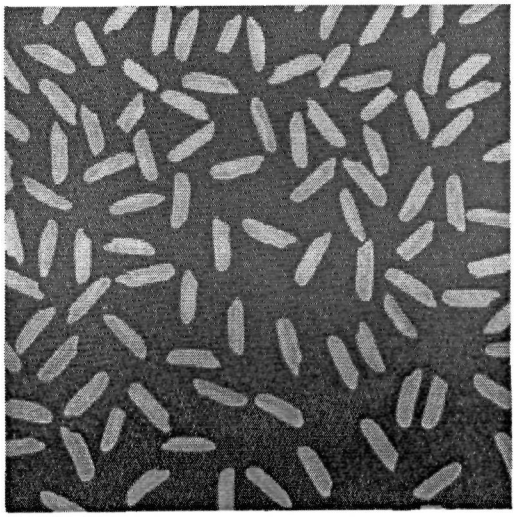


Figure 4.20: *Rice image.*
Courtesy of MATLAB 7 Image Processing Toolbox, © The MathWorks, Inc. 1984-2006.
Dimension of image: 256×256 pixels

Figure 4.21: *Pout image.*
Courtesy of MATLAB 7 Image Processing Toolbox, © The MathWorks, Inc. 1984-2006.
Dimension of image: 240×291 pixels

Figure 4.22: *Coins image.*
Courtesy of MATLAB 7 Image Processing Toolbox, © The MathWorks, Inc. 1984-2006.
Dimension of image: 300×246 pixels

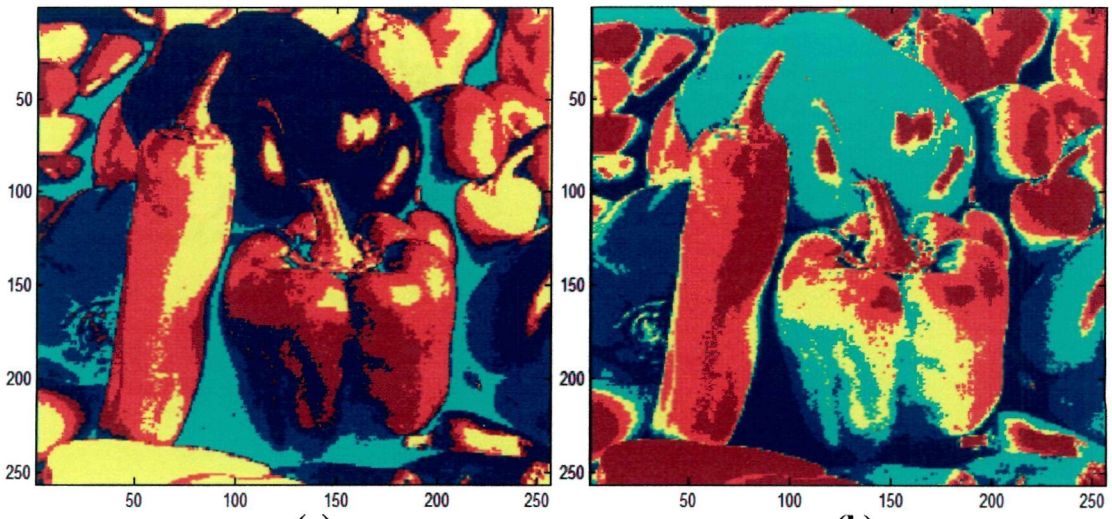
As can be seen from the images, the determination of the optimal number of clusters for the images may be quite complicated, because, the selection of the number of clusters for clustering an image is usually dependant on the number of 'objects of interest' to be considered. Thus, if for the *Coins* image, only the circular coins are considered as 'objects of interest' then the *Coins* image can be clustered by considering the number of clusters as 2 only, however, if somebody is also interested in a detailed segmentation of the coins also, so that the designs carved on the circular coins also get demarcated, then the number of clusters will vary.

For the experiments presented in this subsection, the optimal number of clusters for the images *Baboon* and *Peppers* were obtained from [95]. For all other images, the optimal number of clusters for each of them was estimated through a visual analysis survey by a group of five people. In a visual analysis survey, several clusterings of each image for a varying number of clusters are presented to a group of people and a common consensus is achieved on the optimal segmentation and consequently on the optimal number of clusters for that image. This approach of determination of number of clusters has been used by other researchers also [95] [96].

Table 4.14: Optimal number of clusters considered for the natural images

Image name	No. of clusters considered
Peppers	6
Brandy rose	3
Baboon	6
Cell	3
House	9
Rice	3
Pout	9
Coins	6

Some experimental results of clustering obtained by K-means and FCM on the images considered are now presented next. The values for the number of iterations and average execution time are provided for the first optimal clustering encountered in a series of 50 trials. For FCM, $m = 1.5$. From the results it can be seen that FCM is quite costlier than K-means, however the results obtained by both the algorithms are comparable. For validation, the *PBM* and *PBMF* validity indices were used.



(a) **(b)**

Figure 4.23(a): Peppers image clustered using K-means for $K = 6$.

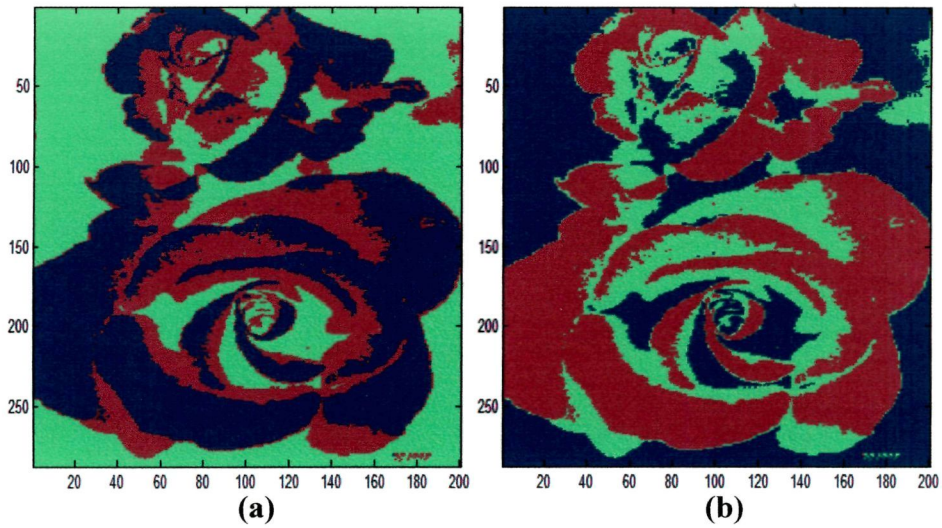
No. of iterations = 12. Average execution time = 0.90688 seconds.

Cumulative execution time = 81.6258. Best *PBM* index value = 83062.4

Figure 4.23(b): Peppers image clustered using FCM for $c = 6$.

No. of iterations = 87. Average execution time = 26.746 seconds.

Cumulative execution time = 1191.7891. Best *PBMF* index value = 99914.9



(a) **(b)**

Figure 4.24(a): Brandy rose image clustered using K-means for $K = 3$.

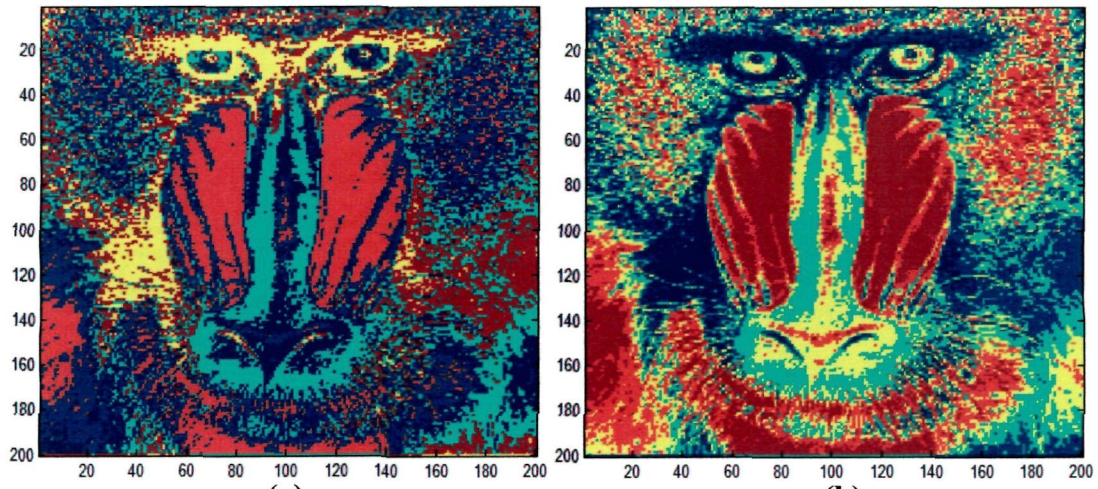
No. of iterations = 11. Average execution time = 0.34096 seconds.

Cumulative execution time = 19.3949. Best *PBM* index value = 51390.5

Figure 4.24(b): Brandy rose image clustered using FCM for $c = 3$.

No. of iterations = 35. Average execution time = 4.6812 seconds.

Cumulative execution time = 247.2499. Best *PBMF* index value = 59738.7



(a) **(b)**

Figure 4.25(a): Baboon image clustered using K-means for $K = 6$.

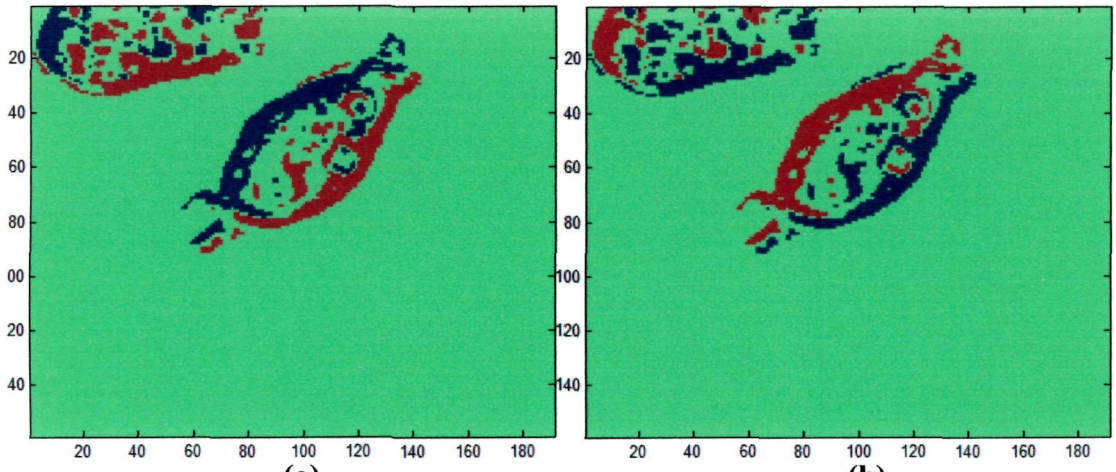
No. of iterations = 8. Average execution time = 0.32334 seconds.

Cumulative execution time = 37.777. Best *PBM* index value = 30190.5

Figure 4.25(b): Baboon image clustered using FCM for $c = 6$.

No. of iterations = 130. Average execution time = 23.2572 seconds.

Cumulative execution time = 1151.1049. Best *PBMF* index value = 37068.5



(a) **(b)**

Figure 4.26(a): Cell image clustered using K-means for $K = 3$.

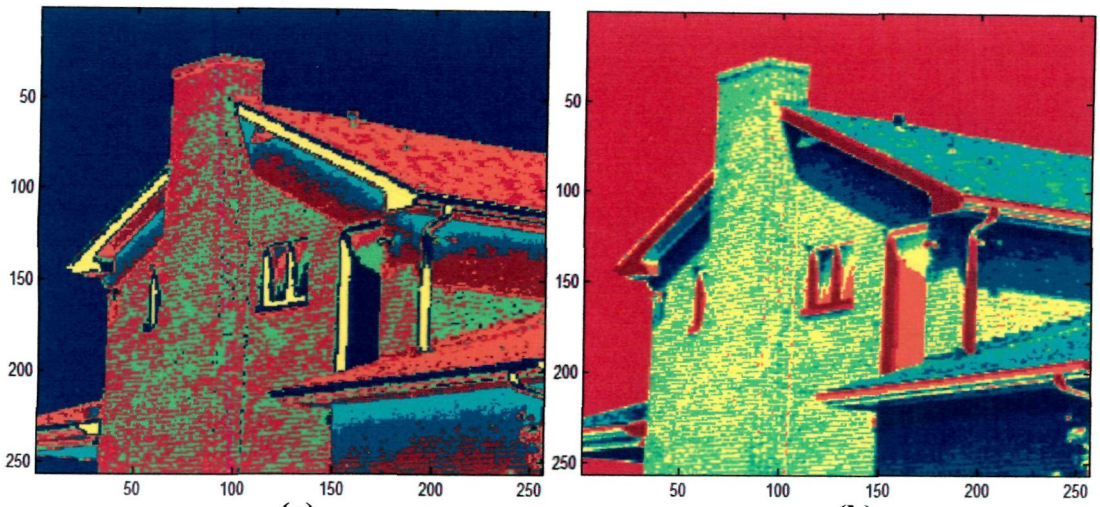
No. of iterations = 9. Average execution time = 0.14264 seconds.

Cumulative execution time = 6.6636. Best *PBM* index value = 5811.83

Figure 4.26(b): Cell image clustered using FCM for $c = 3$.

No. of iterations = 27. Average execution time = 1.8842 seconds.

Cumulative execution time = 87.0683. Best *PBMF* index value = 6595.66



(a) (b)

Figure 4.27(a): House image clustered using K-means for $K = 9$.

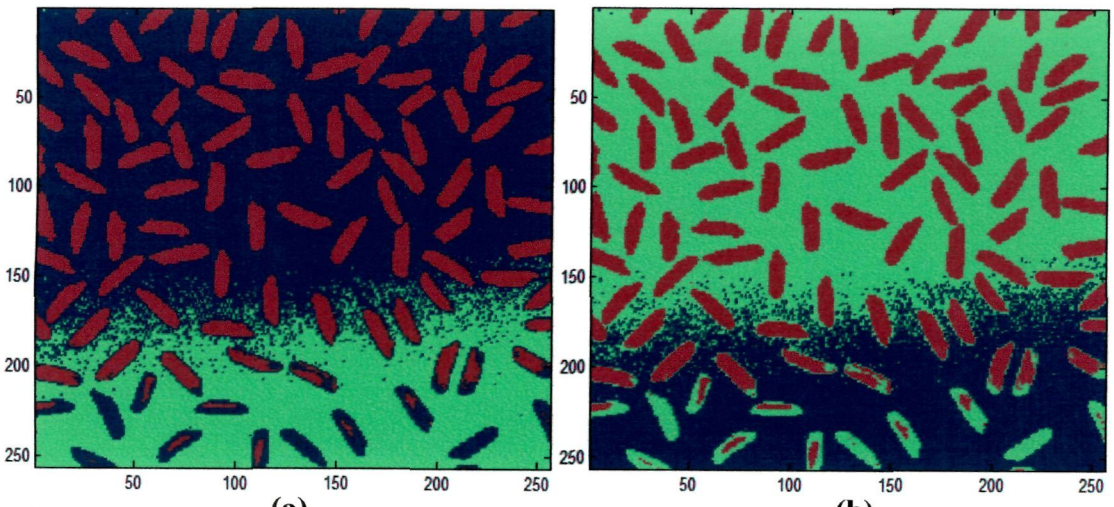
No. of iterations = 8. Average execution time = 0.91052 seconds.

Cumulative execution time = 104.6877. Best *PBM* index value = 167654

Figure 4.27(b): House image clustered using FCM for $c = 9$.

No. of iterations = 85. Average execution time = 41.186 seconds.

Cumulative execution time = 2621.2945. Best *PBMF* index value = 203598



(a) (b)

Figure 4.28(a): Rice image clustered using K-means for $K = 3$.

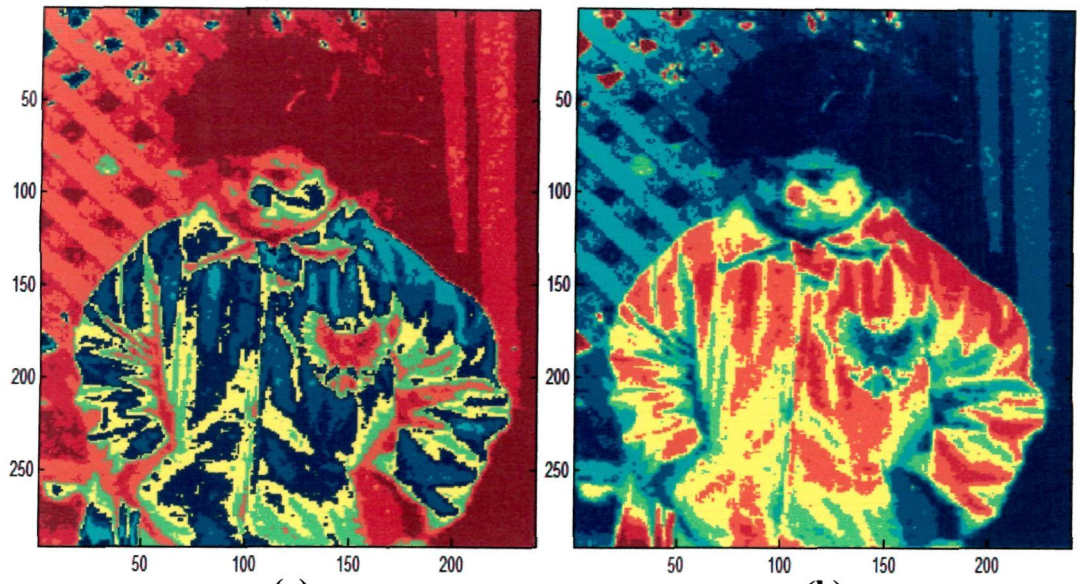
No. of iterations = 5. Average execution time = 0.22381 seconds.

Cumulative execution time = 16.9948. Best *PBM* index value = 46175.6

Figure 4.28(b): Rice image clustered using FCM for $c = 3$.

No. of iterations = 22. Average execution time = 3.4222 seconds.

Cumulative execution time = 164.1814. Best *PBMF* index value = 54430.1



(a) (b)

Figure 4.29(a): Pout image clustered using K-means for $K = 9$.

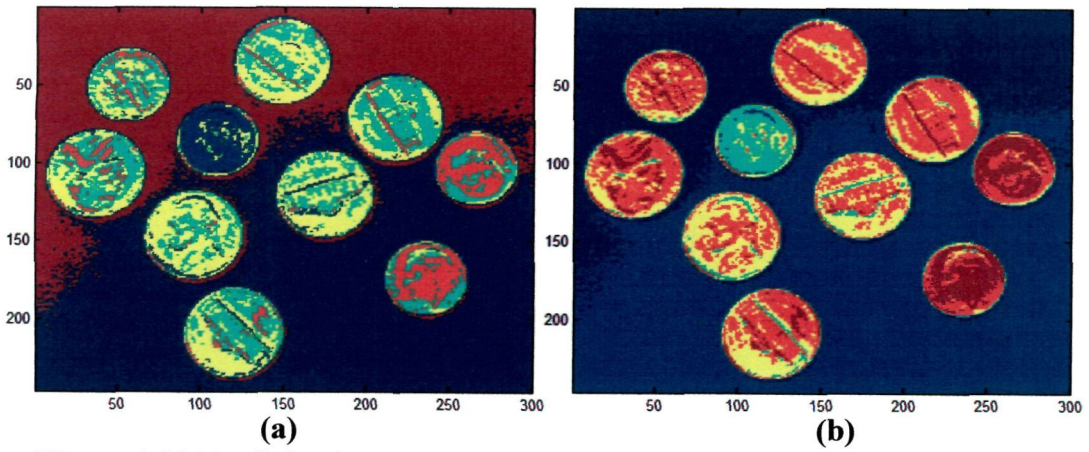
No. of iterations = 19. Average execution time = 2.1599 seconds.

Cumulative execution time = 74.5961. Best *PBM* index value = 32083.5

Figure 4.29(b): Pout image clustered using FCM for $c = 9$.

No. of iterations = 144. Average execution time = 72.3401 seconds.

Cumulative execution time = 2243.5209. Best *PBMF* index value = 40707



(a) (b)

Figure 4.30(a): Coins image clustered using K-means for $K = 6$.

No. of iterations = 14. Average execution time = 1.1461 seconds.

Cumulative execution time = 78.2023. Best *PBM* index value = 260387

Figure 4.30(b): Coins image clustered using FCM for $c = 6$.

No. of iterations = 113. Average execution time = 38.6004 seconds.

Cumulative execution time = 1732.0642. Best *PBMF* index value = 308375

From all the clustering results on the images considered, it can be clearly comprehended that clustering is indeed a very effective tool for image segmentation.

4.7.3 Text Clustering

In this subsection, experimental results of text document clustering using the K-means clustering algorithm are presented. For the representation of the text documents in the collection of text documents considered, the Vector Space Model representation of the documents [35] using the TF-IDF weighting scheme [97] [98] was used. Since the TF-IDF matrix of text documents can grow very large with thousands of dimensions for each pattern vector representing a text document in the text collection, the FCM algorithm was not used in this case, as it has already been seen that FCM is computationally costly. Moreover the FCM algorithm is also heavily dependant on the fuzzifier parameter m and the choice of a good value of m is based on the problem under consideration. So, deciding for a good value of m in every case, for different types of text collections can be a difficult task.

For the experiments on text clustering a simplified version of the 20 Newsgroups text data collection [99] obtained from the UCI machine learning repository [81] was used.

The 20 Newsgroups text collection data set consists of 20000 messages taken from 20 Usenet newsgroups each corresponding to a different topic. The twenty newsgroups are:

comp.graphics *comp.os.ms-windows.misc* *comp.sys.ibm.pc.hardware* *comp.sys.mac.hardware* *comp.windows.x*	*rec.autos* *rec.motorcycles* *rec.sport.baseball* *rec.sport.hockey*	*sci.crypt* *sci.electronics* *sci.med* *sci.space*
misc.forsale	*talk.politics.misc* *talk.politics.guns* *talk.politics.mideast*	*talk.religion.misc* *alt.atheism* *soc.religion.christian*

The simplified subset used in the experiment is composed of 10 articles each from each of the above 20 newsgroups. That is a total of 200 text documents are there in

the set. The preprocessing of the set of documents was done using a set of 700 stop-words and 39 special characters to be filtered out from the documents. After that the set of text documents, when converted using the vector space model technique and TF-IDF weighting scheme resulted in a 200×10602 sized 2-dimensional data matrix which when saved on disk consumed 18.1 Mega Bytes of disk space. Thus the resulting data set that was used in the experiment is quite a large data set. Hereafter it is referred to as *20-mini-newsgroups* data set.

A plot of the data set is shown below:

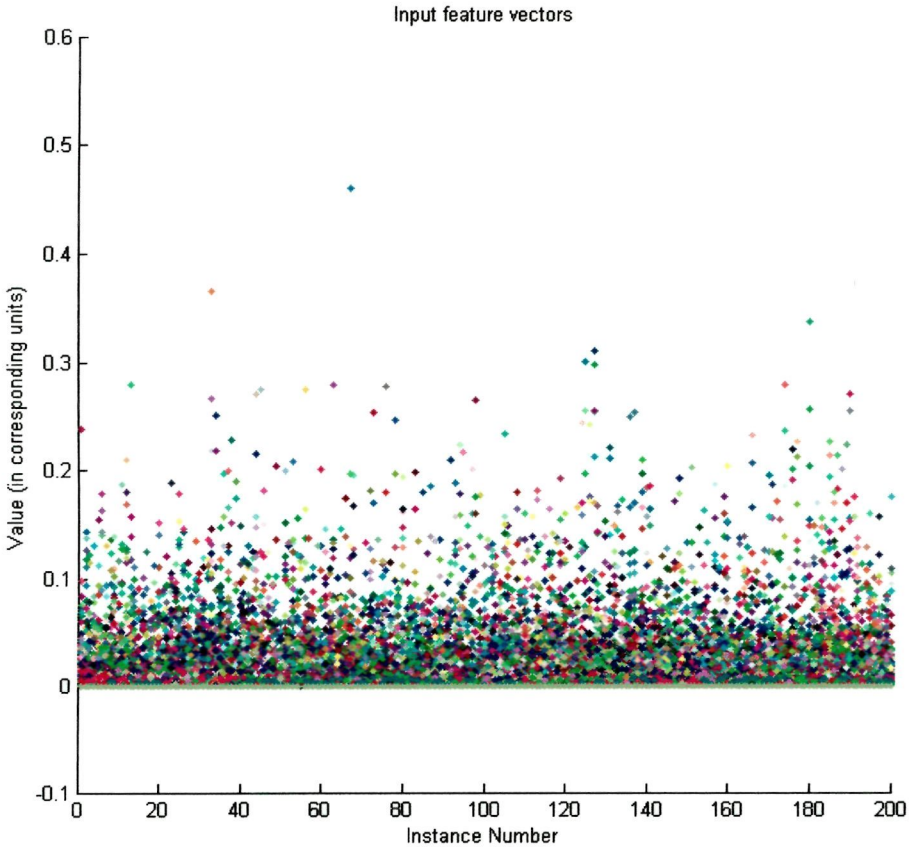


Figure 4.31: Instance number vs. input feature vector values' plot for the 20-mini-newsgroups data set

From the plot it can be easily seen that the 20 clusters are quite overlapping and linearly inseparable. The results of application of K-means algorithm on this data set are now presented next. As there are 20 newsgroups so the value of K was taken as 20. Figure 4.20 shows the summary of the best clustering (as decided by the optimum value of *PBM*) obtained in a series of 50 trials of the K-means algorithm on the 20-mini-newsgroups data set. The cosine distance was used as the proximity measure for

the K-means algorithm and also for the evaluation of value for the *PBM* index.

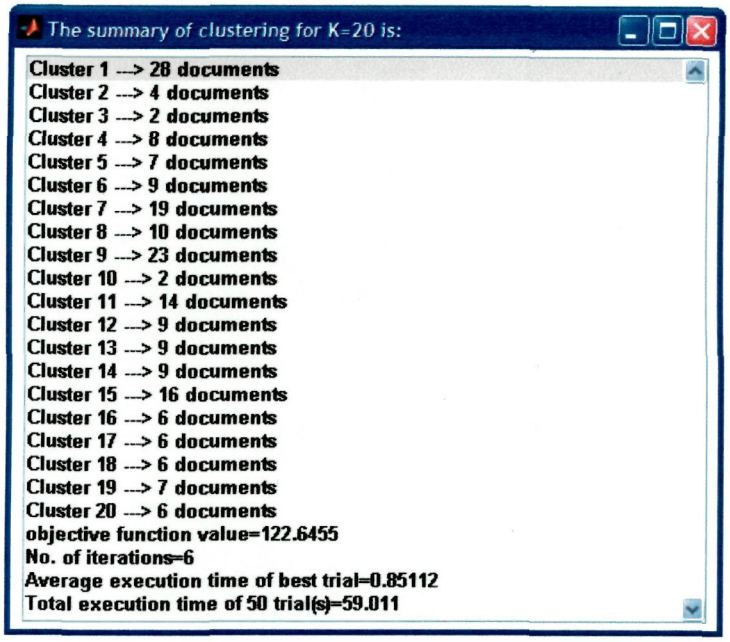


Figure 4.32: Summary output window (from PatternWiz) for K = 20 for the 20-mini-newsgroups data set. Best *PBM* index value = 0.00272557

As can be seen from the results, the sizes of most of the clusters are closer to the ideal value of 10. Moreover the iteration count and execution time (in seconds) required by the best trial of the K-means algorithm is also quite less.

The silhouette plot for the 20 clusters is shown below:

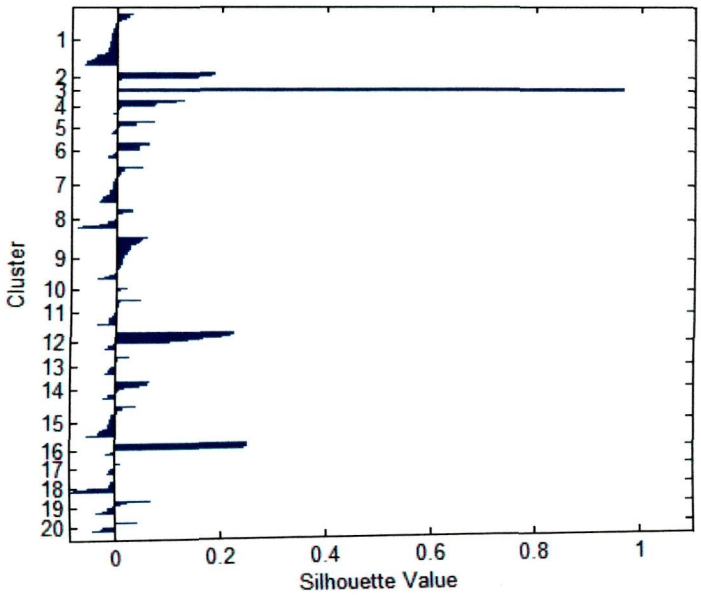


Figure 4.33: Silhouette plot for clustering result obtained by K-means for K = 20 for the 20-mini-newsgroups data set

From the silhouette plot, however, it can be inferred that the clusters formed are not quite well separated, as indicated by the negative silhouette values, and the degree of similarity between the individual text documents in each cluster is also low as indicated by a low silhouette value of 0 to 0.4 for almost all the clusters. A glimpse of the cluster assignments for the individual documents is shown below:

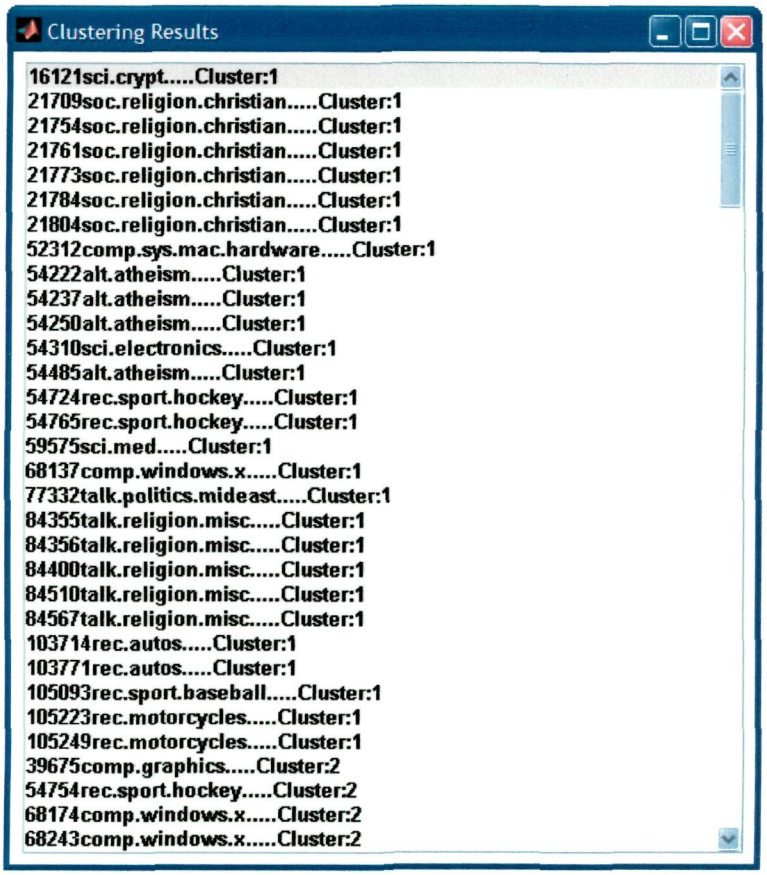


Figure 4.34: Glimpse of the cluster assignments obtained by K-means for $K = 20$ for the 20-mini-newsgroups data set

From this experiment on text clustering conducted using K-means it can thus be inferred that the usual clustering methodology is not very much suitable for text clustering. For the classification of text documents in a more meaningful way, text document classification should involve the discovery of semantic, lexical and ontological relations in the texts of a given sets of documents. The vector space model representation of text documents do not consider such important relations in the text documents which leads to a very low relevance score for relevant documents. Moreover, the vector representations of documents tend to use all the words in the

text documents after removing the stop-words, regardless of their meaning and semantic relations with other words in the collection of documents. This leads to thousands of dimensions in the vector representation of documents. It is however well known that only a very small number of words/terms in documents have distinguishable power to classify documents. Thus these problems need to be solved in an efficient manner for better clustering results for text documents. As already mentioned in subsection 2.6.3, these problems can be addressed very well by using WordNet lexical categories and WordNet ontology as a result of which a well structured document vector space of low dimensionality can be created so that better performances by the clustering algorithms can be attained.

To summarize this section on experimental results of clustering, a comparison of K-means and FCM clustering algorithms is presented below:

Table 4.15: Comparison of K-means and Fuzzy c-means clustering algorithms

Name of algorithm	K-means	Fuzzy c-means
Type of algorithm	Hard, partitional, non-deterministic, polythetic, non-incremental	Fuzzy, pseudo-partitional, non-deterministic, polythetic, non-incremental
Inputs	No. of clusters K	No. of clusters c, fuzzifier parameter m
Shape of clusters	Hyper-spherical	Non-convex
Search Technique	Localized	Localized
Nature of convergence	Convergence to local minima of sum of squared error objective function.	Convergence to local minima of sum of squared error objective function.
Time complexity*	$O(nKdt)$	$O(ncdt)$
Advantages	Fast and easy to implement, suitable for medium to large data sets	Good at dealing with fuzziness of clusters, applicable to medium and large data sets
Disadvantages	Dependent on initial centroids, Not well suited for data with fuzzy overlapping clusters.	Dependent on initial centroids and m, Computationally a little bit costly

*n = no. of points, K, c = no. of clusters, d = no. of dimensions, t = no. of iterations

4.8 Chapter Summary

In this chapter first a brief overview on clustering, similarity measures and different taxonomical representations of clustering are presented. Then discussions on two prominent clustering algorithms namely the K-means hard clustering algorithm and the Fuzzy c-means (FCM) fuzzy clustering algorithm are presented. After that, some popular clustering validity assessment indices for hard and fuzzy clustering are discussed. Further, some experiments on the applications of the K-means and FCM clustering algorithms for pattern recognition on numeric, image and text data are presented. Experiments on numeric and image data showed good performances by both the K-means and FCM clustering algorithms with the K-means algorithm being less costly than FCM. Experimental results on validation of clustering obtained by the two algorithms showed that the *PBM* validity index and its fuzzy version *PBMF* gives the best performance among all the validity indices experimented. Experiments on text clustering revealed the inability of conventional clustering approaches in efficiently utilizing the inherent semantic relationships of words/terms in text documents, for better meaningful results. In general, all the experiments showed that both the algorithms are very well suited for solving a wide array of clustering problems, from various problem and data domains.