B. Sc. Engineering Thesis

Minimal Parameter Clustering of Complex Shaped & Different Sized Dataset with Noise & Outlier Handling

MPCACS

by

Ahmed Al Muzaddid

K. M. A. Solaiman

(In Alphabetical Order)

Submitted to

Department of Computer Science and Engineering

in partial fulfilment of the requirements for the degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

Department of Computer Science and Engineering

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Dhaka-1000, Bangladesh

June, 2014

Declaration

This is to certify that the work presented in this thesis entitled "**Minimal Parameter Clustering of Complex Shaped & Different Sized Dataset with Noise & Outlier Handling (MPCACS)** " is the outcome of the investigation carried out by us under the supervision of Dr. Md. Monirul Islam, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. It is also declared that neither this thesis nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

(Supervisor)

Dr. Md. Monirul Islam

Professor

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka-1000, Bangladesh

(Author)

Ahmed Al Muzaddid

Student No: 0805053

(Author)

K. M. A. Solaiman

Student No: 0805116

Abstract

Clustering is a popular technique used for data analysis and as a pre processing step. In real life, shape of clusters of underlying datasets are unknown and in most of the cases they are irregular. In this dissertation, we have proposed and evaluated a new algorithm for clustering which performs best for complex shaped datasets. We have also concentrated on the use of minimal number of parameters to ensure a stability in the clustering performance. The purpose of this thesis was to investigate the problems and drawbacks of existing clustering algorithms and to find a clustering method capable of solving those problems and which is suitable for irregular shape datasets.

We have used fourier transformation and filtering technique to represent the data set in a real valued functional form. Then we formulated a technique to form clusters from the derived functional form. Conventional clustering algorithms having problem of parameter selection show inconsistency in performance. Our work presents a dynamic parameter selection strategy for our algorithm which gives optimum clustering performance.

Different cluster evaluation metrics are used to compare performance of our algorithm with other existing algorithms which have shown promising improvement. The MPCACS algorithm was applied on the same datasets as the other algorithms and the resulting affects were investigated. The results show that, our algorithm shows significant improvement in clustering complex shaped datasets and far better performance than the traditional algorithms.

Keywords: Clustering, Density Based Clustering, Unsupervised Learning, Irregular Shapes, Complex Shapes, Minimal Parameter Clustering.

Acknowledgements

First of all we would like to thank our supervisor, Dr. Md. Monirul Islam, for introducing us to the fascinating field of unsupervised learning & clustering and guiding us on how to perform our research work. Without his continuous supervision, guidance and valuable advice, it would have been impossible to complete the thesis. We are especially grateful to him for allowing us to chose our course of actions with freedom, for enlightening us during the moments of frustration, for his patience throughout the entire research work and for his moral support.

We are grateful to our friends for their continuous encouragement and for helping us in thesis writing. They have been there in the moment of need and supported us wholeheartedly.

We would like to express our gratitude to all our teachers. Their motivation and encouragement in addition to the background knowledge about our thesis subject they provided meant a lot to us.

Last but not the least, we are grateful to our parents and to our families for their patience, interest, and support during our studies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Introduction

In the world of data mining, clustering is a very popular word as a powerful means of data analysis and information retrieval technique. Clustering can be used to discover the underlying pattern and distribution of datasets. In the sense that cluster analysis does not use category labels that tag objects with prior identifiers or any examples showing the desirable relations among data, makes it possible to consider it as an unsupervised process.
Clustering Analysis is defined as a way to group relatively homogeneous cases or observations in a separate set which is different from other sets or objects formed outside this particular group [JD88, KR90]. Clustering is a form of learning by observation, rather than learning by examples. Everitt [Eve74] describes 'cluster' in the following manner:

- 'A cluster is a set of entities which are alike, and entities from different clusters are not alike.'
- 'A cluster is an aggregation of points in the test space 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.'

- 'Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.'

We focus on the last two definitions of cluster and formulate a method to find clusters based on distance and density. A basic clustering method will produce clusters as illustrated in figure 1.1. Although clustering have been the subject of plenty of studies and a large number of clustering algorithms have been developed, considerable amount of challenges still remain unanswered. However, dealing with high dimensional datasets is rather new in the field of clustering. On the other hand, finding clusters in data is a challenging job when the clusters are of widely differing shapes, sizes, and densities. In most of the cases the data also contains noise and outliers and makes clustering even harder. So solving these 'quality' and 'scalability' issues along with finding methods for optimality have been an issue in the field of clustering.

1.2 Previous Work

In this section we will discuss about the previous works done on clustering. And later on chapter 2, we will discuss the various weaknesses of these works. Till today many clustering algorithms have been developed to address different clustering issues. First in section 1.2.1 we will discuss the traditional clustering algorithms. Then in section 1.2.2 we will discuss the more recent algorithms that have overcome some of the drawbacks of the traditional algorithms.

1.2.1 Traditional Clustering Algorithms

K-means [JD88] is a well known partitioning method. Objects are classified as belonging to one of k groups, k chosen as priori. Cluster is selected by calculating the centroid for each group and assigning object to the group with closest centroid as illustrated in figure 1.2. The mean of each cluster is marked by a '+'. Partitioning algorithms attempt to determine those

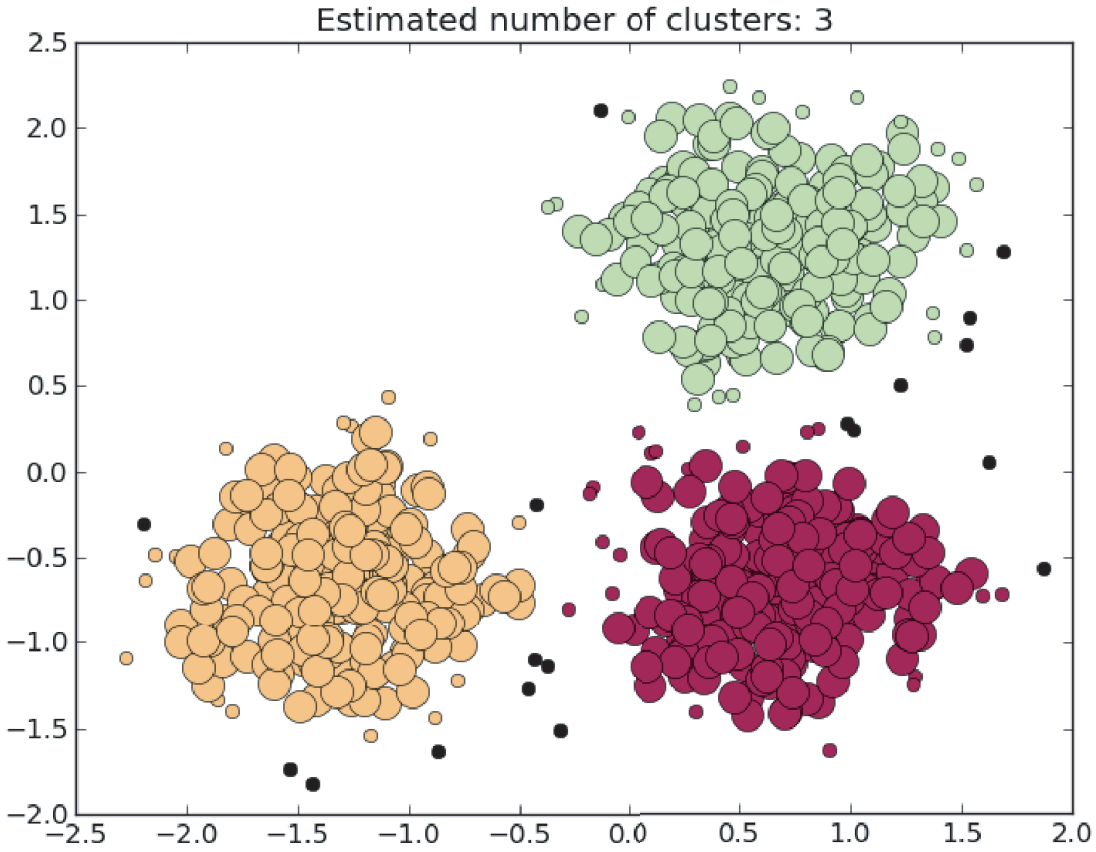


Figure 1.1: Clustering Method

k partitions that optimize a certain criterion function. Typically, the square-error criterion is used, defined as shown in equation 1.1.

$$E = \sum_{i=1}^{k} \sum_{p \epsilon C_i} |p - m_i|^2 \tag{1.1}$$

where E is the sum of the square error for all objects in the data set, p is the point in space representing a given object and m_i is the mean of cluster C_i. This criterion tries to make the resulting k clusters as compact and as separate as possible and worked well when clusters are compact clouds that are rather well separated from one another. The method is relatively scalable and efficient in processing large data sets because the computational complexity of the algorithm is linear and it is order independent. So it can be used for clustering high dimensional

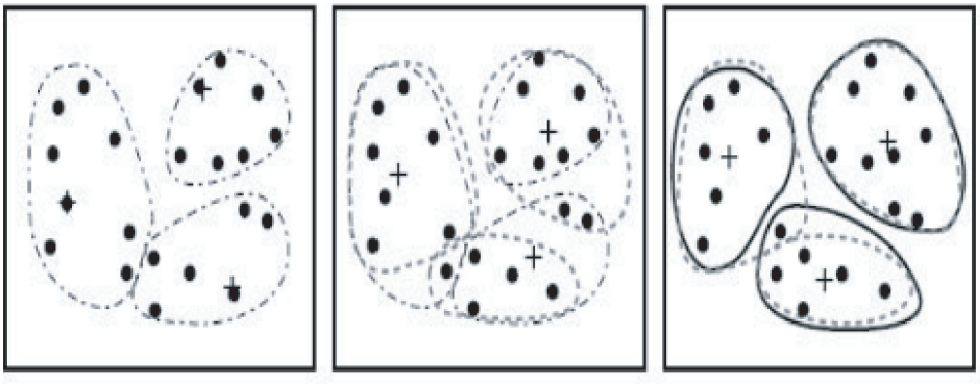


Figure 1.2: Clustering of a set of objects based on the k-means method.

datasets but it has a great number of quality issues discussed later in section 2.3.1.

Hierarchical clustering algorithms combine or divide existing groups, resulting in a hierarchical structure that resembles the merge or divide order. Agglomerative method uses the merging approach where initially the objects belong to a list of singleton sets $S_1, S_2, \ldots, S_n$. Then a cost function is used to find the 'lowest-cost' pair of sets S_i, S_j. Once merged, S_i and S_j are removed from the list of sets and is replaced with the new set $S_i \cup S_j$. This process continues until all objects are in a single group. For this process they have to compute the proximity matrix containing the distance between each pair patterns. This proximity measure is used to find the most similar pair of clusters. Different variants of agglomerative hierarchical clustering algorithms use different cost functions. Complete, average and single linkage respectively use maximum, average and minimum distances between members of two clusters as cost function. Three widely used measures for distance between clusters are as follows, where $|p - p'|$ is the distance between two objects or points, p and p', m_i is the mean for cluster, C_i and n_i is the number of objects in C_i. Single, Complete and average linkage respectively use the distance measures shown in equation 1.2, 1.3 and 1.4.

$$d_{min}(C_i, C_j) = min_{p \epsilon C_i, p' \epsilon C_j} |p - p'| \tag{1.2}$$

$$d_{max}(C_i, C_j) = max_{p \epsilon C_i, p' \epsilon C_j} |p - p'| \tag{1.3}$$

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \epsilon C_i} \sum_{p' \epsilon C_j} |p - p'| \tag{1.4}$$

All of the above measures have a minimum variance and they usually yield the same results of the clusters are compact and well separated. But if the clusters are close to one another(even by outliers), or if their shapes and sizes are not hyper-spherical and uniform, the results can vary to a large extent.

Self Organizing Maps(SOM) [Abb08] is inspired by neural networks and uses a competition and cooperation mechanism to achieve unsupervised learning. A set of nodes is arranged in a geometric pattern, typically 2-dimensional lattice. Each node is associated with a weight vector with the same dimension as the input space. During training, each object in the input is presented to the map and the best matching node is identified. The dimension and size of the map(k) has to be given priori.

Expectation Maximization (EM) is a distance based algorithm that assumes the data set can be modelled as a linear combination of multivariate normal distributions and the algorithm finds the distribution parameters that maximize model quality measure, log likelihood. EM is linear in database size and robust to noisy data. It can be applied to high dimensionality datasets.

1.2.2 Density Based Algorithms

Finding clusters of different shapes and sizes when data contains noise and outlier is a problem that many recent clustering algorithms, have addressed. For low dimensional data DBSCAN [EKSX96], CURE [GRS98], and Chameleon [KHK99] have shown good performance.

In case of density based algorithm, traditionally real valued data with varying number of attributes are handled. They work on relative density of example instance in feature space. According to the density based clustering definition [TK08], clusters are dense regions in the data space, separated by regions of lower sample density and A cluster is defined as a maximal set of density-connected points.

DBSCAN is considered as one of the main success in density based clustering. It uses euclidean distance measure around a sample point called '$\epsilon - neighbourhood'$ and $minpts$ defined as the minimum number of points required in '$\epsilon - neighbourhood'$ for it to be a dense space.

Chameleon builds a list of nearest neighbours of each point and then constructs a weighted similarity graph using those nearest neighbourhood list, and then partitions the graph using a graph partitioning algorithm, METIS [KK98] to obtain cluster fragments which are merged into clusters with a hierarchical agglomerative clustering technique.

CURE also uses the concept of representative points to find many types of non-globular clusters. This method represents a cluster by using multiple representative points from the cluster. The first point is chosen to furthest from the cluster centre, while remaining points are chosen so that they are farthest from all previously chosen points, ensuring the points to be well distributed. This is a result of the way the CURE algorithm finds representative points, i.e., it finds points along the boundary, and then shrinks those points towards the centre of the cluster by a factor, α. It uses an agglomerative hierarchical scheme to perform the actual clustering. However, cure is still biased towards finding globular clusters as it still has the notion of cluster centre. To handle the high dimensionality issue, a shared nearest neighbour approach to similarity was proposed in ROCK [GRS98] and previously by Jarvis and Patrick in [JP73]. At first the nearest neighbours of each point are found, and then a new similarity between points is defined in terms of the shared number of neighbours. Given the new similarity, ROCK uses agglomerative hierarchical clustering , but the Jarvis-Patrick method only finds connected components by grouping all points with non-zero similarity.

So we have almost summarized most of the recent clustering approaches that handle some of the important challenges of clustering. But there are plenty of other works similar to those discussed. For example, DENCLUE [HK98] and OptiGrid [HK99] are more recent density based schemes that are likely to match the performance of DBSCAN or even go beyond it.

1.3 Objective of Thesis

As we have seen, different approaches have been explored previously for finding robust algorithm which can handle clusters of arbitrary shapes and sizes. Among them density based algorithms are most successful in terms of finding arbitrary shaped cluster. So we have taken a density based approach for finding an optimal clustering algorithm. Our method addresses the following issues:

- **Minimal External Parameter for Clustering Analysis**

Clustering serves as an important branch of unsupervised learning. In case of unsupervised learning, as less as possible supervision is desirable. Popular clustering algorithms like Dbscan which deals with arbitrary shape clusters is sensitive to parameter. So firstly we have tried to make our algorithm less sensitive to external parameters. As based on the parameters clustering results may vary from time to time and from user to user. Secondly we tried formulating a method to find those parameters dynamically for optimum clustering. However, we weren't able to totally eliminate the need of it. But we were able to reduce it to one single parameter, 'filterWidth'. This 'filterWidth' is used when calculating the "augmented" density function $f(x)$ which is developed using Fourier transformation. As minimal parameters are needed, so no fluctuations in the cluster results.

- **Improving scalability of online clustering methods**

Clustering is one of the prime techniques to handle the large amount of information present on the web. Clustering can be performed either offline, independent of search queries, or performed online on the basis of the results of search queries. Most traditional clustering algorithms are limited to offline clustering. Besides, many clustering algorithms work well on small data sets containing fewer than several hundred data objects, but a large database may contain millions of objects. Clustering on a sample of a given large data set may lead to biased results. So highly scalable clustering algorithms are needed. Our algorithm aims at improving the performance of online clustering method and also achieve a better scalability.

- **Incremental clustering**

Using some of the advanced algorithms for incremental Fourier algorithm we have tried to handle new data instance without invoking the whole process. This makes the algorithm more versatile and suitable for processing continuous stream of data.

- **Clustering in High dimensional data space**

Clustering high dimensional data has to deal with the "curse of dimensionality" [Bel19]. It is the most widely seen problem which is data analysis techniques such as clustering ,which work well at lower dimensions, often work poorly as the dimensionality of the analysing data increases. We want to solve this problem by using a technique which is generalized for all dimensions.

- **Insensitivity to the order of input records**

Most of the current algorithms performance largely depends on the order of the input records. But this is not an ideal situation. In whatever order data arrives cluster should be the same. We want to eliminate this problem with our algorithm. We have tried to develop a order independent clustering algorithm. Our clustering performance will not deteriorate with the order of data input.

- **Finding methods for clustering complex shapes and types of data**

Most of the existing clustering algorithms are unable to handle clusters of different sizes and non-globular or any kind of irregular shapes. These irregular sizes and shapes is a big challenge to the existing clustering techniques. No single algorithm can give an optimal result for all kinds of sizes and shapes. Our technique aims at finding a solution to it.

- **Handling outlier and noisy data**

In clustering, outlier and noisy data are the irregular instances in the dataset that do not follow any underlying distribution function. Taking them into consideration may lead to bad cluster outputs. Our algorithm aims at handling them efficiently.

1.4 Thesis Organization

Our dissertation is organized as follows. Chapter 2 discusses the motivation behind our work and any necessary literature is also discussed there. The actual clustering algorithm is described in chapter 3. Our total effort is categorized into two main sections. In the first section, we show how we create an augmented data distribution function using sample data instances. It is the core data representation model which is used by all the further steps. Specific data preprocessing and difficulty in handling some types of data are also highlighted in this section. A brief idea of scalability of data size and the treatment for the outlier or the noisy data can be found in this section. In the second section, main algorithm is described with its mathematical formulations and explanations. It consists of the basic idea behind the algorithm and approaches taken to overcome the difficulties regarding arbitrary shaped cluster. We show all possible difficulties which may be faced by the algorithm while clustering the data instance. Run time complexity and memory requirement for the algorithm is also described at the end of this section. Moreover, we describe how to choose the only parameter our algorithm needs dynamically to an optimal level in this section. Strategies for improving performance of some specific data types is also described here. Chapter 4 comprises of the performance analysis with different case studies. Synthetic as well as historical bench mark data sets are used to evaluate performance of our algorithm. We experimentally show that our algorithm performs better than mostly used clustering methods such as K-means, EM, Hierarchical methods, DBSCAN, CURE on a variety of datasets: Spiral data, Jain data, Aggregation data, Compound data and many others. We also show comparison with rarely used algorithms like Cobweb & FarthestFirst and demonstrate the better performance of our method. Chapter 5 includes the conclusion and directions for future work.

Chapter 2

Background

2.1 Introduction

Cluster analysis is an important technique in the rapidly growing field of exploratory data analysis and is being applied in a variety of engineering and scientific disciplines. The purpose of cluster analysis is to place objects into groups, or clusters, suggested by the data, not defined a priori, such that objects in a given cluster tend to be similar to each other in some sense, and objects in different clusters tend to be dissimilar. According to the definition of everitt [Eve74] discussed in chapter 1, clusters can be defined by means of distance and density. Clusters of same kind have less distance in between than clusters of different kinds. This results in high intra-class similarity & low inter-class similarity as shown in figure 2.1. Similarly, density based clusters can be defined as highly dense points, separated from other such regions by lower dense points. And cluster analysis can be described as the method of obtaining these clusters. The interest in cluster analysis is derived from the complexity and scalability issues which are eminent in real life applications of clustering.

Whether for understanding or utility, cluster analysis has long played an important role in a wide variety of fields: psychology and other social sciences, biology, statistics, pattern recognition,

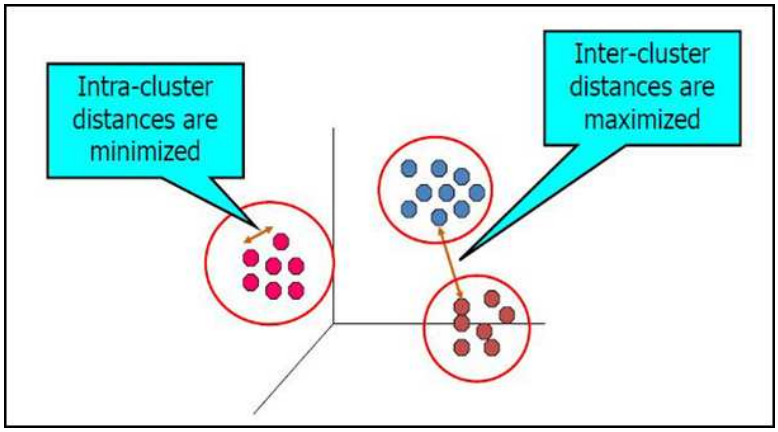


Figure 2.1: Clustering Method

information retrieval, machine learning, and data mining. In the context of understanding data, clusters are potential classes and cluster analysis is the study of techniques for automatically finding classes. Some important fields are biology for creating taxonomy or finding group of genes, information retrieval i.e., understanding www & analysing search queries, climate i.e., finding patterns in atmosphere and ocean, psychology and medicine i.e., identifying different categorise or patterns of a disease & business for segmenting customers in groups for analysis and marketing. Cluster prototypes can be used as the basis for a number of data analysis or data processing techniques. Therefore, in the context of utility, cluster analysis is the study of techniques for finding the most representative cluster prototypes. Clustering can be used as a means of summarization. Many data analysis techniques which are not practical for large data sets, can be used by applying to a reduced data set consisting of cluster prototypes and then using those prototypes for entire data set. Clustering prototypes can also be used for data compression and for finding nearest neighbours. Data compression is known as known as vector quantization and is often applied to image, sound, and video data.

We are going to discuss the literature & incentive behind our thesis work in this section. We will discuss about the drawbacks of the algorithms discussed in chapter 1.2 and we will point out the problems that we are trying to solve.

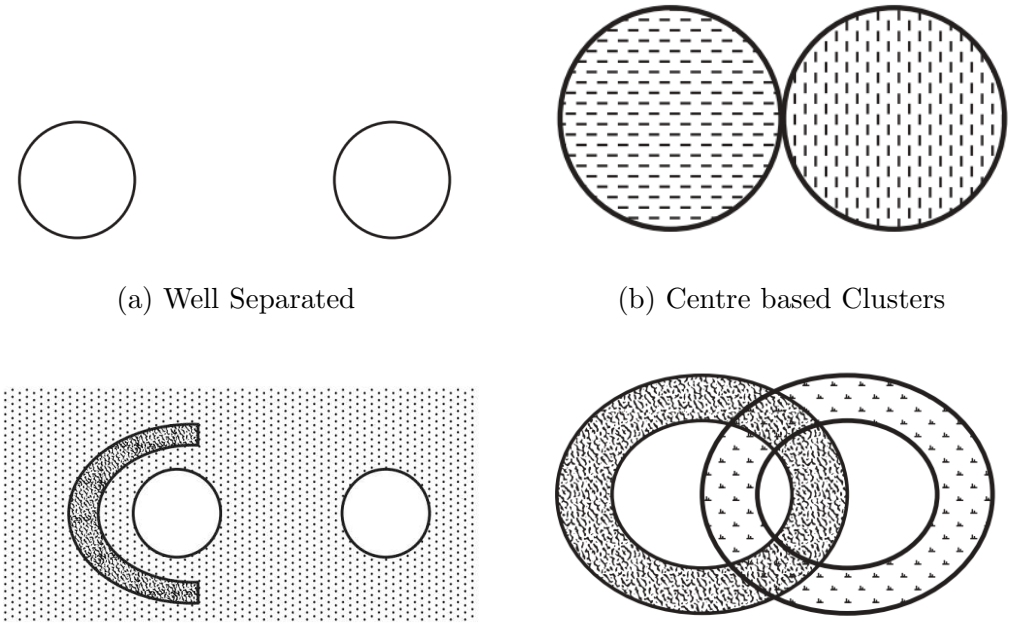
2.2 Literature Review

Cluster analysis groups data objects based on underlying information that describes the objects and their relationships. The superiority of a clustering technique depends on the quality of the clusters. Clustering is different from classification. It creates a labelling of objects with cluster labels but it derives these labels only from the data where classification assigns a class label by developing a model from other labelled objects.

Cluster can be different types. Well separated clusters are those clusters in which each object is closer to every other object in the cluster than to any object not in the cluster. Well separated clusters do not need to be globular, they can be of any shape. Prototype based clusters are those clusters in which each object is closer to its own prototype than to others prototype. The prototype can be centroid, medoid or any central point. Density based cluster is a dense region of objects that is surrounded by a region of low density. This definition is often employed when clusters are irregular or intertwined and when noise and outliers are present. Conceptual clusters do not work well for these types of data. Conceptual cluster is defined as a set of objects that share some property. Figure 2.2 shows different types of clusters.

Clustering algorithms can be categorized based on their cluster model. They vary on measuring criterion and proximity measures. Some of them works with similarity measures and some of them with dissimilarity measures. Normally similarity is expressed in terms of a distance function, typically metric: $d(i, j)$. They also vary in data types they handle. Data can be discrete valued, continuous or categorical. They can be interval-scaled variables, binary variables, nominal, ordinal or ratio variables & variables of mixed types. Different similarity metrics for different types can be found in [JD88, KR90, TK08]. In general, the major clustering methods can be classified into the following categories: Partitioning methods, Hierarchical methods and Density based methods. The most important algorithms are discussed in section 1.2.

Connectivity based clustering is based on the core idea of objects being more related to nearby



(a) Well Separated

(b) Centre based Clusters

(c) Density Based Clusters

(d) Conceptual Clusters

Figure 2.2: Cluster Types

objects than to objects farther away. These algorithms connect 'objects' to form 'clusters' based on their distance. At different distances, different clusters will form, which can be represented using a dendrogram. Hierarchical clustering is an example of this type. In prototype-based partitional clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering treats it as an optimization problem. Using distribution models, clusters can be defined as objects belonging most likely to the same distribution. One prominent method is known as Gaussian mixture models which uses the EM algorithm. In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise and border points. The most popular density based clustering method is DBSCAN.

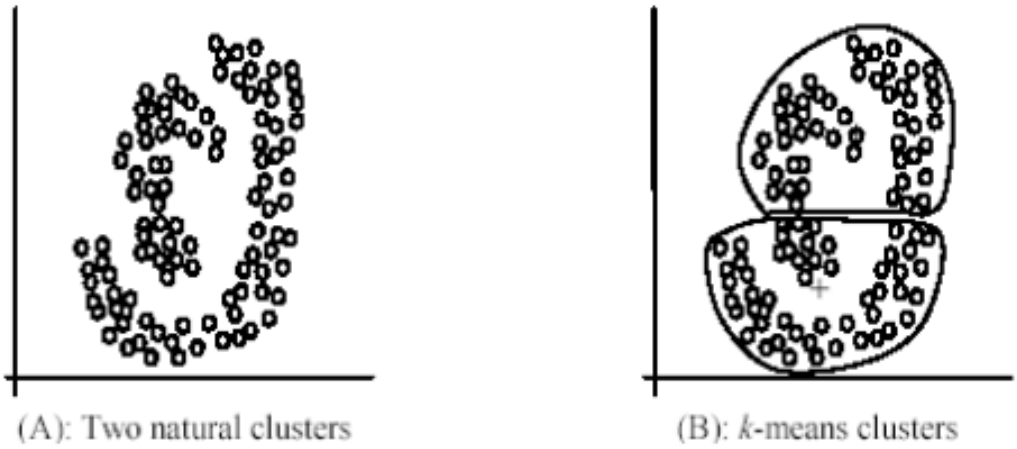


Figure 2.3: Bad results for spherical shape clusters using K-means.

2.3 Problem Review

2.3.1 Partitional Clustering Algorithm

K-means is the mostly used partitional algorithm. The k-means method can be applied only when the mean of a cluster is defined. This may not be the case in some applications, such as when data with categorical attributes are involved. The necessity for users to specify k, the number of clusters, is a huge disadvantage. In case of unsupervised or online learning it might be quite impossible to know the value in advance.

The k-means method is not suitable for discovering clusters with non-globular shapes or clusters of very different size. The centroid representation alone works well if the clusters are of the hyper-spherical shape. If clusters are elongated or are of other shapes such as shown in figure 2.3, centroids are not sufficient.

We have seen in section 1.2, k-means tries to minimize the square error. As shown in figure 2.4 illustrated in [GRS97], the square-error method could split large clusters to minimize the square error. In the figure, the square-error is larger for the three separate clusters in *(a)* than for the three clusters in *(b)* where the big cluster is split into three portions, one of which is merged with the two smaller clusters. The reduction in square error due to splitting the large

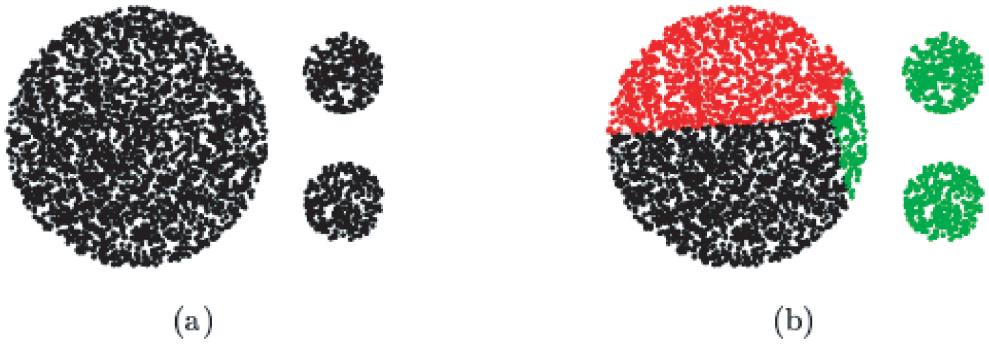


Figure 2.4: Splitting of a large cluster by partitional algorithms

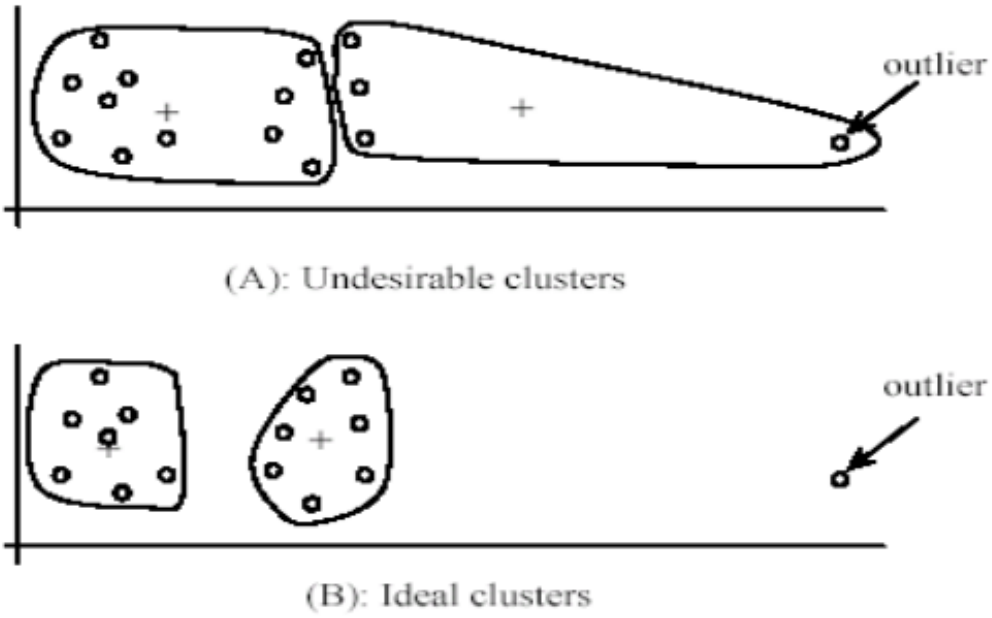
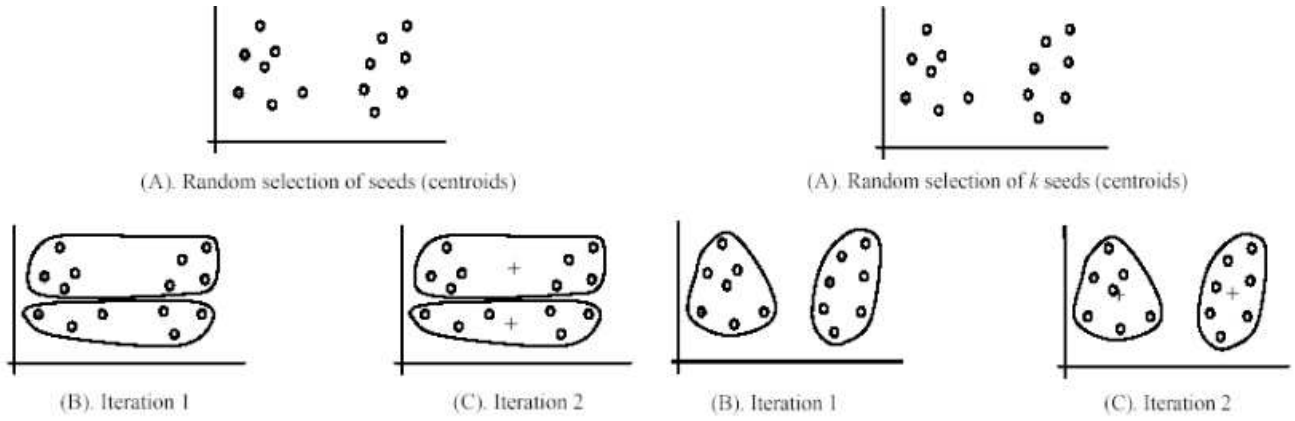


Figure 2.5: Sensitivity of K-means towards noise and outlier

cluster is weighted by many data points in the large cluster.

Moreover, it is sensitive to noise and outlier data points because a small number of such data can substantially influence the mean value. This problem is illustrated in figure 2.5. K-means method is sensitive to initial seeds. Different seeds may result into different clusters as shown in figure 2.6. So based on the user choice cluster shape and result will be different.



(a) Selection 1 (b) Selection 2

Figure 2.6: Different results of K-means using different seeds.

2.3.2 Hierarchical Clustering Algorithm

Single link agglomerative clustering is suitable for clusters with non-globular shapes. But it is sensitive to noise and can't handle cluster of varying density. It may cause the undesirable "chain effect" by noisy points. Considering the example data points in the following figure demonstrated in [GRS97], the desired elongated clusters are shown in figure 2.7a. Single link clustering causes the two elongated clusters to be merged as a single cluster. This is the 'chain effect' as shown in figure 2.7b.

Complete Link and Group Average Clustering are not affected by noise. But they do have a bias towards finding globular clusters and they are sensitive to outliers. Average clustering causes the elongated clusters to be split and portions belonging to neighbouring elongated clusters to be merged. The resulting clusters are shown in figure 2.7c.

When the number N of input data points is large, hierarchical algorithms break down due to their non-linear time complexity and huge I/O costs. Another drawback is that user has to specify the initial number of groups. In hierarchical clustering, proximity matrix or similarity matrix is an essential part. It has to computed before clustering begins.

(a) Desired Clusters (b) Single Link (c) Complete Link

Figure 2.7: Clusters generated by hierarchical algorithms

2.3.3 Distribution Based Clustering Algorithm

While the theoretical foundation of these methods is excellent, they suffer from one key problem known as over fitting, unless constraints are put on the model complexity. A more complex model will usually always be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult. EM will converge to a local optimum, so multiple runs may produce different results. Using these algorithms puts an extra burden on the user: to choose appropriate data models to optimize, and for many real data sets, there may be no mathematical model available the algorithm is able to optimize.

2.3.4 Density Based Clustering Algorithm

DBSCAN uses some key terms: core point, border point and noise point. A point is a core point if it has more than a specified number of points (MinPts) within Eps which are the points that are at the interior of a cluster. A border point has fewer than MinPts within Eps, but is in the neighbourhood of a core point. A noise point is any point that is neither a core point nor a border point. Noise points are discarded, while clusters are formed around the core points.

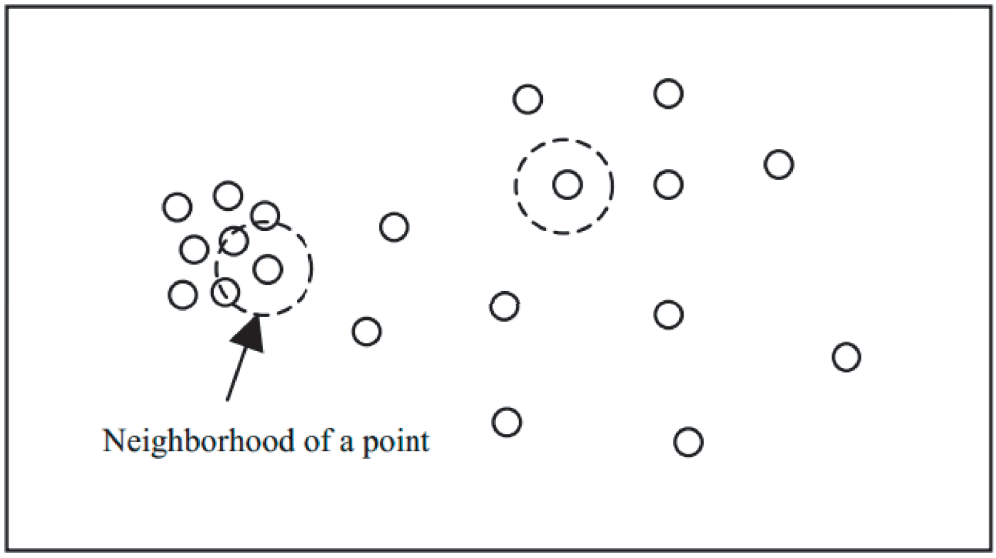


Figure 2.8: Density based Neighbourhoods

An object q is 'directly density-reachable' from object p if p is a core object and q is in p's e-neighbourhood.

While DBSCAN can find clusters of arbitrary shapes, it cannot handle data containing clusters of differing densities, since its density based definition of core points cannot identify the core points of varying density clusters. As in figure 2.8, if the user defines a neighbourhood of a point by specifying a particular radius and looks for 'minpts' within that radius, then either the tight left cluster will be selected as one cluster and the rest will be considered as noise, or else all points will belong to one single cluster, neither of which is desirable.

We see from the above definitions that dbscan largely depends on the parameter value ϵ and minpts. A small change in parameter value can lead to totally different results. Some techniques are adapted to approximate those parameters but they only work in global context. Clusters with different density will suffer from misleading parameter value hence leads to poor clustering. Chameleon does not directly apply the idea of core points. All three methods – DBSCAN, Chameleon and CURE tackles the problem of finding clusters of different sizes and shpaes by finding points or small subsets of points and building clusters around them. This method is especially suitable for spatial data, since non-globular clusters are not represented by their centroid. And so they cannot be handled by centroid based ideas. Single link agglomerative

clustering methods are most suitable for finding clusters with non-globular shapes, but these methods are very fragile and cannot handle noise properly.

Both CURE and DBSCAN have problems with clusters of different density. Chameleon can partly handle clusters of varying density as it uses the nearest neighbour approach, but it does not work well for high dimensional data, e.g., documents.

The key drawback of DBSCAN and OPTICS is that they expect some kind of density drop to detect cluster borders. Moreover, they cannot detect intrinsic cluster structures which are prevalent in the majority of real life data. DENCLUE is better & faster than DBSCAN, but needs a large number of parameters.

So we believe we have pinpointed the key issues in density based clustering: using representative points to deal with differing shapes and sizes, the difficulty of dealing with clusters of differing densities, the importance of eliminating outliers and noise, and the problems with similarity measurement which arises mainly in higher dimensions. And the important issue of depending on external parameters still follows the density based algorithms.

2.4 Summary

K-means and hierarchical method does not work well with outliers, clusters of different sizes and non-globular shapes. These two methods are sensitive to noise and initial seed choices. Performance of SOM becomes lower as value of K (number of clusters) becomes greater. SOM gives better accuracy than hierarchical. And hierarchical gives better accuracy than K-means and EM. All of them have some ambiguity in noisy data. DBSCAN,CURE and chameleon can find clusters of different sizes and shapes, but they face problem in finding globular clusters and in finding clusters of differing densities. Also, all of these methods have a problem of defining a number of external parameters which can effect the clustering result according to the choice of those parameters.

Chapter 3

Our Method

3.1 Introduction

Our method mainly consists of two parts: Data representation and cluster formation. We represent the input data into a functional form so that all necessary information are easy to access and are noise free. This representation is also the basis of our main algorithm. It has a probability density function like interpretation, and for this reason in rest of our paper we refer this as 'augmented density function'.

Main algorithm uses a 'linear density reachability' measure [EKSX96] but the definition is quite different here. For cluster formation step, we maintain two data structures – one of them contains all current clusters and other contains the cluster representatives.

3.2 Data Representation

For data compression and signal & image processing, Fourier transformation is an efficient technique. We adopted fourier transformation technique to represent input sample instance

density in a functional form.

For the sample point (x_1, y_1), (x_2, y_2) $\ldots\ldots\ldots\ldots(x_M, y_N)$, at first we transfer data into frequency domain with equation 3.1. Then we use Gaussian low pass filter in frequency domain. Gaussian filter equation is illustrated in equation 3.2. Applying gaussian filter we get equation 3.4.

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(x,y) e^{-2\pi j(\frac{ux}{M} + \frac{vy}{N})} \tag{3.1}$$

$$H\left(u,v\right) = e^{-D^2(u,v)/2\sigma^2} \tag{3.2}$$

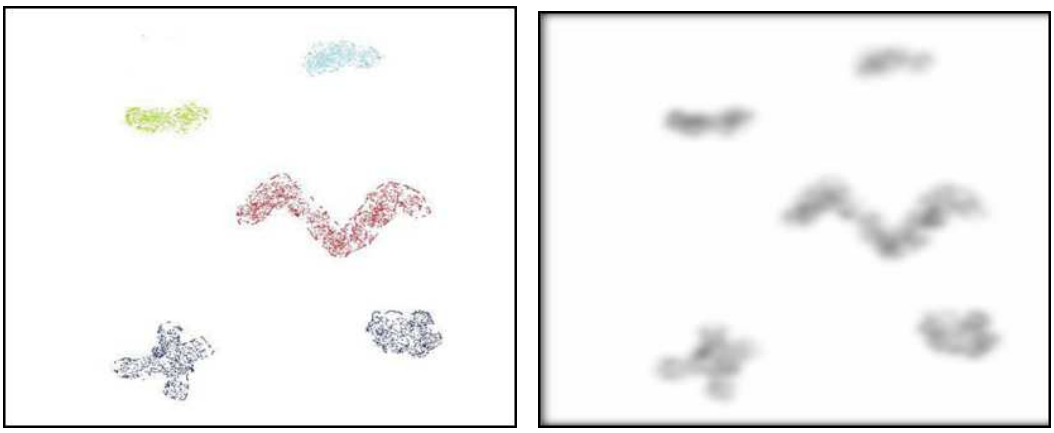
$$D(u,v) = \sqrt{\left(u - \varphi_1/2\right)^2 + \left(u - \varphi_2/2\right)^2} \tag{3.3}$$

where φ_i is the filter width along different dimensions.

$$G\left(u,v\right) = H\left(u,v\right) * F(u,v) \tag{3.4}$$

Low pass filter is used to smooth out any outlier and any abrupt change in input domain as we know any cluster will be continuous in attribute space. As specific instance of low pass filter we use Gaussian low pass filter [BZ08]. Other low pass filter like butterworth and laplacian filter are also tested but the Gaussian low pass filter serves best for our purpose. Gaussian low pass filter requires filter width as its input parameter. Finding this parameter efficiently and effectively will be described later. After that we have to use inverse fourier transformation to get back our required functional form in attribute domain.

$$f(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(u,v) e^{2\pi j(\frac{ux}{M} + \frac{vy}{N})} \tag{3.5}$$



(a) Before Filtering (b) After Filtering

Figure 3.1: The Filtering Process

The main reason behind choosing fourier transformation rather than using other well-known density function is that it requires less parameter and can be represented into functional form with any arbitrary level of accuracy which will serves us to handle clusters of different density. For illustration we can compare it with multivariate Gaussian distribution, in that case we have to find more parameter (σ, μ) and in spite of that it will be lacking representational power.

Some techniques has been already developed to incorporate new data in fourier representation. That is a huge advantage for any incremental stream data processing with less computational effort. And we know that order of data is immaterial in case of fourier transformation as long as the attribute values are only focused in time domain representation. Also higher dimensional fourier transformation is available for higher dimensional data.

Another reason for selecting fourier representation is that efficient algorithms have already been developed for finding fourier transformation like fft [Loa92, HIKP12, EMT99, 32]. Dedicated hardware processors are also available for faster fourier transformation and analysis.

We can easily show that fourier representation of input data set serves as a density function such that

$$p\left(x_i, y_i\right) > \ p\left(x_j, y_j\right) \ \leftrightarrow \ f\left(x_i, y_i\right) > \ f\left(x_j, y_j\right)$$

where $p\left(x_i, y_i\right)$ is the joint probability at point (x_i, y_i). The filtering process is shown in figure 3.1.

3.3 The Algorithm

Algorithm 1 MPCACS

Input: *Dataset to be clustered*
Output: *Number of clusters and their cluster centre*

```
1: x ⇐ next sample data
2: p ⇐ f(x)
3: if p < threshold then
4:    if x from any of the previous cluster then
5:       Check for possibility of merging
6:    end if
7: else
8:    create a new cluster with x as the cluster centre
9: end if
```

Algorithm 2 In Previous Cluster

Input: *sample instance x*
Output: *cluster number which x belongs to*

```
1: belonging_set ⇐ initialize belonging_set to empty
2: for all neighbouring cluster n_i of x do
3:    if current cluster center n_i.c linear density reachable to x then
4:       belonging_set ⇐ belonging_set ∪n_i
5:    end if
6: end for
7: if size of belonging_set is one then
8:    return single cluster
9: else
10:   merged_cluster ⇐ merge(belonging_set)
11: end if
12: return merged_cluster
```

Linear density reachability:

A point X is linear density reachable to Y if for all points p_i between X & Y, $f(p_i)>0$

3.3.1 Description of our Algorithm

When a new sample comes first we find out whether it in a dense zone or not. If it is in a dense zone that means it is member of any one of the previous clusters. If this data instance

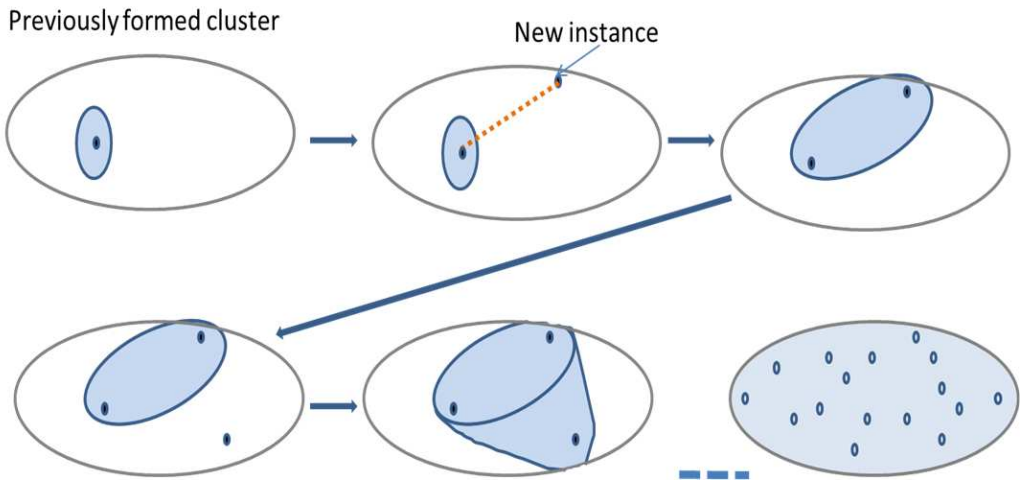


Figure 3.2: Work Flow for Linearly Density Reachable Case

is at boundary of two or more clusters then those clusters should be merged into a single one. If current sample is in dense zone but previously no instance come from that zone then it will form a new cluster.

Algorithm "in previous cluster" finds from which cluster is the sample from, for this it calculates linear density reachability with all neighbouring clusters using cluster centres and cluster representatives.

3.3.2 Intuition behind merging cluster

Linearly density reachable case

For simpler shape clusters like globular or elliptical shapes, new instance will be linearly density reachable from the initial cluster centre. In first step of figure 3.2, a single cluster is initially formed which is the smaller elliptical shape with a centre. When a new instance arrives, algorithm checks whether it is linearly density reachable from previous cluster centre or not. In this case the answer is positive. So new instance is subsumed into the previous cluster. The next step shows the new cluster shape. In similar way, when other new instances arrive inside the simple shape they all will be linearly density reachable and hence form a single cluster without requiring any merging, or small number of merging.

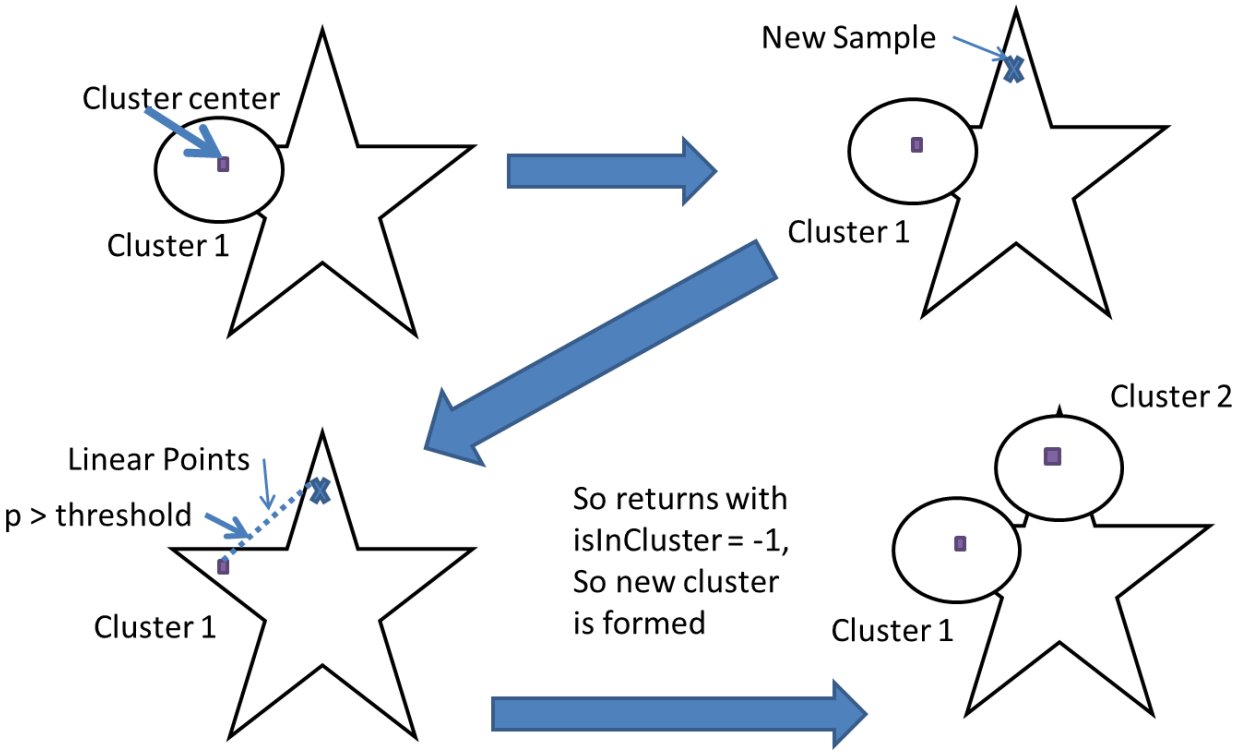


Figure 3.3: Non Linearly Density Reachable Case – Work Flow 1

Non linearly density reachable case

Let augmented density function show a star shape region as a dense zone according to input samples and after some iteration of main algorithm a cluster is formed at certain region like in the first shape of figure 3.3. Let us name this cluster as 'Cluster 1'. Again after some iteration say, another sample have arrived. The centre of 'Cluster 1' and new sample is not linearly density reachable as shown in third step of figure 3.3, so they will not merge. As a result new sample alone will create a new cluster as in the 4th step of figure 3.3. Let it be 'Cluster 2'. After some iteration when a new sample arrives at region such as in first step of figure 3.4, it will be linearly density reachable from both 'Cluster 1' and 'Cluster 2' as shown is second and third step of figure 3.4. So it belongs to both clusters hence there will be two elements in belonging_set . Those two clusters will combine to form a new cluster after merging and cluster centre will be selected among the two cluster centres where augmented density function will give higher numeric value.

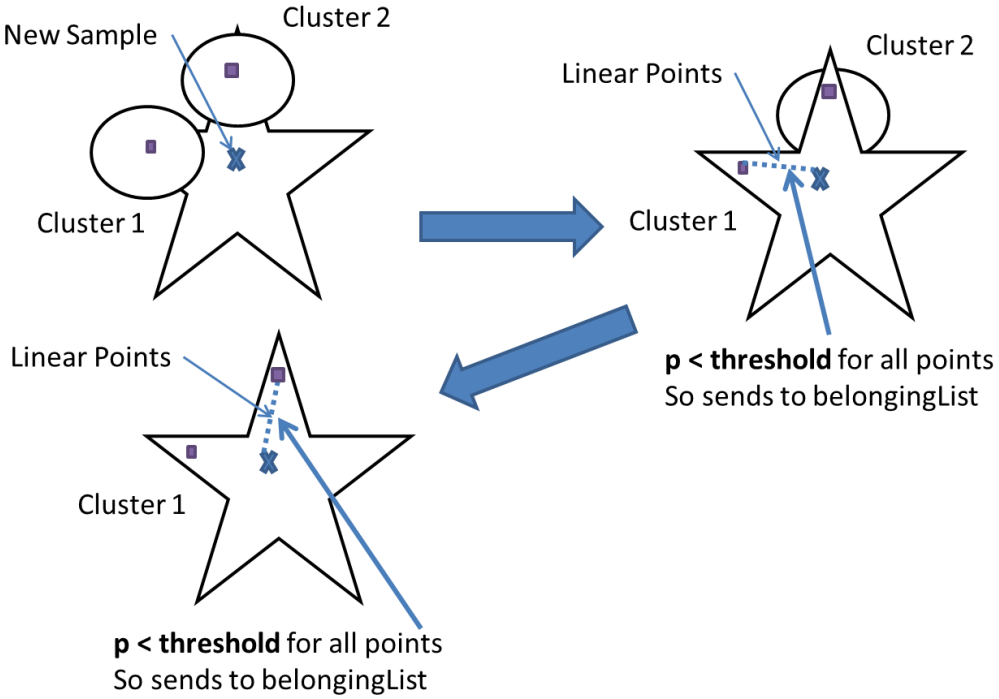


Figure 3.4: Non Linearly Density Reachable Case – Work Flow 2

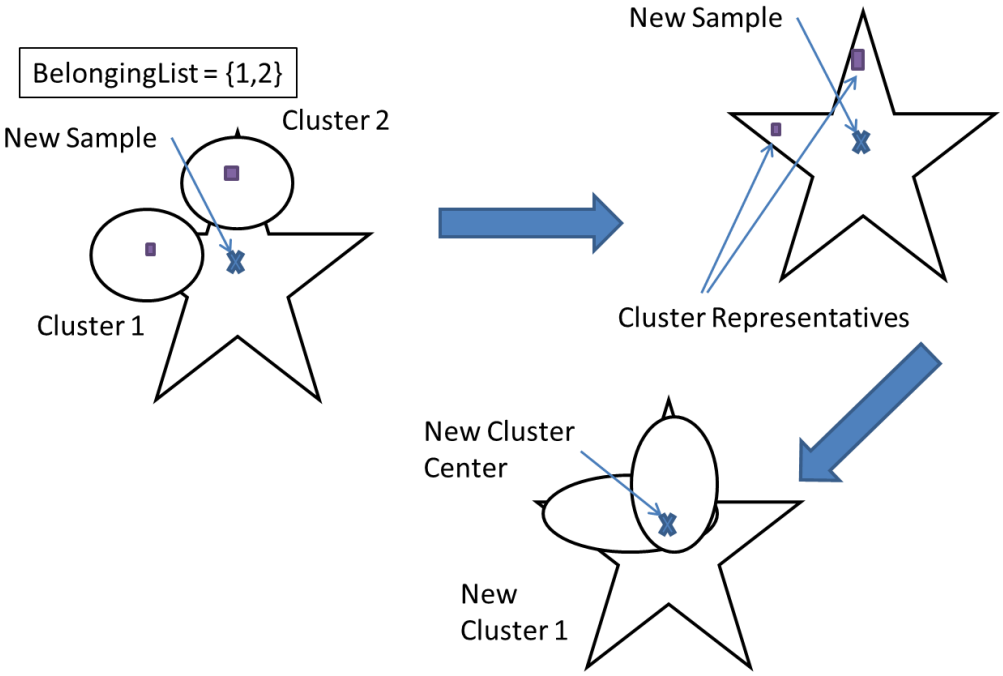


Figure 3.5: Non Linearly Density Reachable Case – Work Flow 3

Table 3.1: Parameter Selection Data for Aggregation Data Set

Filter Width	No. of Clusters
2	1
4	1
8	1
16	4
32	5
48	7
64	8
128	25
256	72

Table 3.2: Performance at different filter widths(Aggregation data)

Filter width	Precision	Recall	Accuracy	F-measure	G-mean
5	12.50%	4.33%	34.64%	6.43%	2.02%
10	49.08%	40.24%	75.89%	44.22%	11.22%
15	50.00%	42.01%	78.43%	45.66%	11.33%
30	62.50%	55.62%	82.74%	58.85%	17.61%
48	**79.73%**	**77.42%**	**88.07%**	**78.55%**	**87.02%**
55	68.14%	69.85%	87.56%	68.98%	28.24%
60	68.14%	69.85%	87.06%	68.81%	28.15%
64	**79.89%**	**77.00%**	**87.18%**	**78.42%**	**86.81%**
70	68.26%	69.84%	85.28%	69.04%	27.87%

3.3.3 Parameter Selection

In our data representation step, we used filter-width (fw) parameter for low pass filtering. From table 3.1 & 3.3 we can see that the number of cluster and performance of the clustering depends on the filter width. But from parameter selection curve in figure 3.6 & 3.7, we can see that performance of the algorithm follow a certain trend with respect to the filter width. By following a simple heuristic we can find the optimum value for this parameter. We choose to increment the filter width multiplying the width each time with factor of two, find the number of cluster. We can see that number of cluster is increasing with the increasing filter width. In some early increments number of cluster increases gradually, but after that it jumps to some big

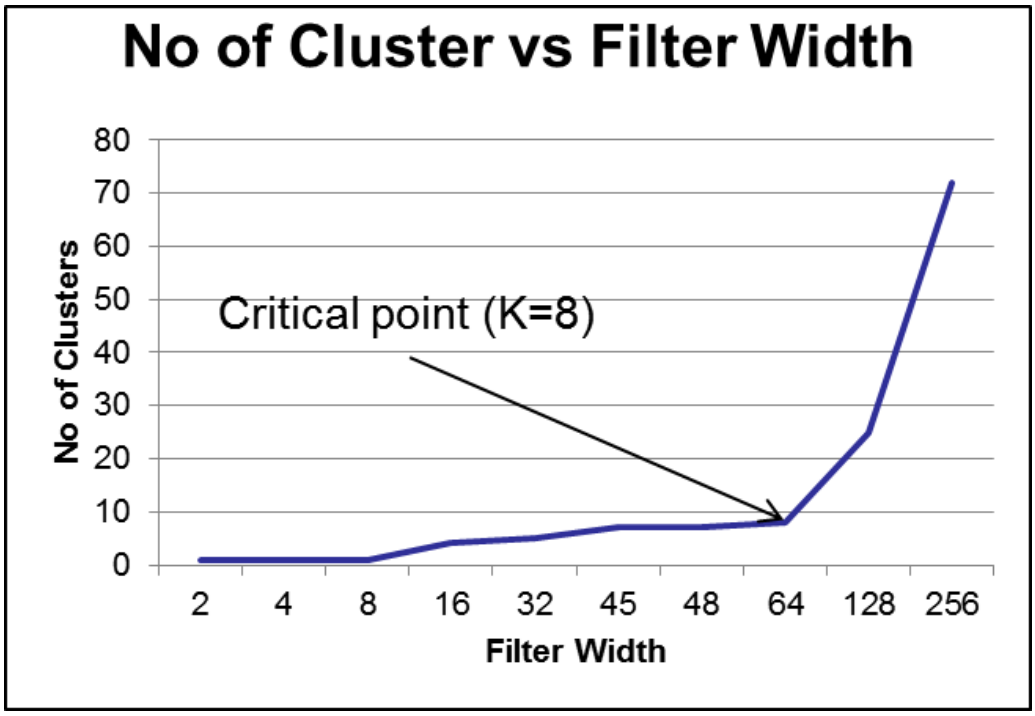


Figure 3.6: Parameter Selection Curve for Aggregation Data

Table 3.3: Parameter Selection Data for Spiral Data Set

Filter Width	No. of Clusters
2	1
4	1
8	2
16	5
32	5
64	5
128	9
256	33

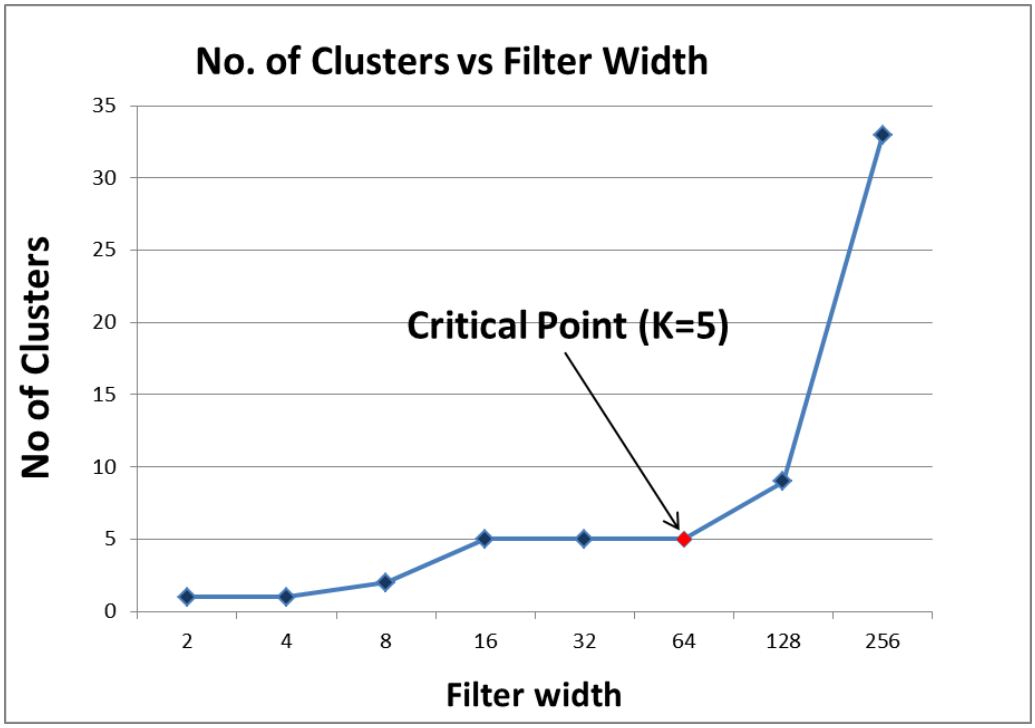


Figure 3.7: Parameter Selection Curve for Spiral Data

Table 3.4: Performance at different filter widths(Spiral data)

Filter width	Precision	Recall	Accuracy	F-measure	G-mean	No of Clusters
8	9.24%	25%	45.83%	13.49%	43.30%	2
10	24.51%	32.58%	47.12%	27.97%	50.62%	2
14	61.30%	44.48%	59.62%	51.55%	61.35%	3
16	100%	77.36%	78.21%	87.24%	87.96%	5
32	100%	77.36%	77.24%	87.24%	87.96%	5
48	**100%**	**77.36%**	**77.56%**	**87.24%**	**87.96%**	**5**
64	**100%**	**77.36%**	**77.56%**	**87.24%**	**87.96%**	**5**
96	100%	75%	77.24%	75%	86.60%	6

number. We called this the 'critical point'. The optimum point can be taken anything between critical point and point before it. But we suggest average of them for better performance.
In figure 3.7, the critical point is at fw = 64, so we can take any value between 64 & 32, or we can take the average $(64 + 32)/2 = 48$ as filter width. And we see from table 3.4 that either fw = 64 or fw = 48 gives optimal performance.

3.3.4 Time & Space Complexity

Complexity of the data representation is dependent on fourier transformation. Using fast fourier transformation, the complexity of transformation results into O(n log$n$) for sample instance with single attribute. If we assume attributes are independent for sample with multiple attributes, then the complexity becomes O(d n log$n$) . Where n is the size of sample and d is the dimension of attribute.
In main algorithm every sample is passed once to fourier transform to get the augmented density. In worst case all clusters can be neighbour of a sample instance. In that case we need O(c) time to find *linear density reachability* for that sample instance. So for all the samples time complexity will be O(nc).
We need memory only to store fourier coefficient for functional representation and a data structure of size c for each cluster. In both cases memory requirement is O(n) in worst case.

3.4 Difference with existing works

In most of the previous clustering technique direct distance measure like euclidean norm is used, but in our approach there is no requirement of such direct measurement. To define a dense area, we have used the value of our augmented density function but in other cases most definition needs parameter specification like minimum points within a certain space. To overcome dependency regarding the order of arrival of sample data, most of the algorithm uses

multiple pass of data set. We have used single pass for each sample when it arrives. While our algorithm can find the 'dense' clusters like other algorithms find, it also finds clusters that methods may overlook, i.e, clusters of low or medium density which may be considered as clusters or outliers.

3.5 Summary

Our algorithm first finds an augmented density function which is developed using fourier transformation. We have only used some low frequency component of fourier transformation which as a result solves the problem of outliers and noises. Then we find the neighbours of each data point using a distance based definition for similarity and the augmented density. The use of density based definition of similarity alleviates problems with varying densities and high dimensionality, while the use of distance based measurement handles problem with shapes and size.

Chapter 4

Experimental Studies

4.1 Introduction

In this section, we study the performance of MPCACS and demonstrate its effectiveness for clustering compared to K-means, EM, Cobweb, FarthestFirst, HierchicalClusterer & MakeDensityBasedClusterer algorithms implemented in weka. From our experimental results, we establish the following decisions –

- MPCACS needs the least number of external parameters
- Gives highest accuracy among all methods for the benchmark datasets Spiral and Jain shapes
- Gives maximum accuracy for different datasets
- Works uniformly throughout all type of datasets

Table 4.1: Characteristics of data sets considered

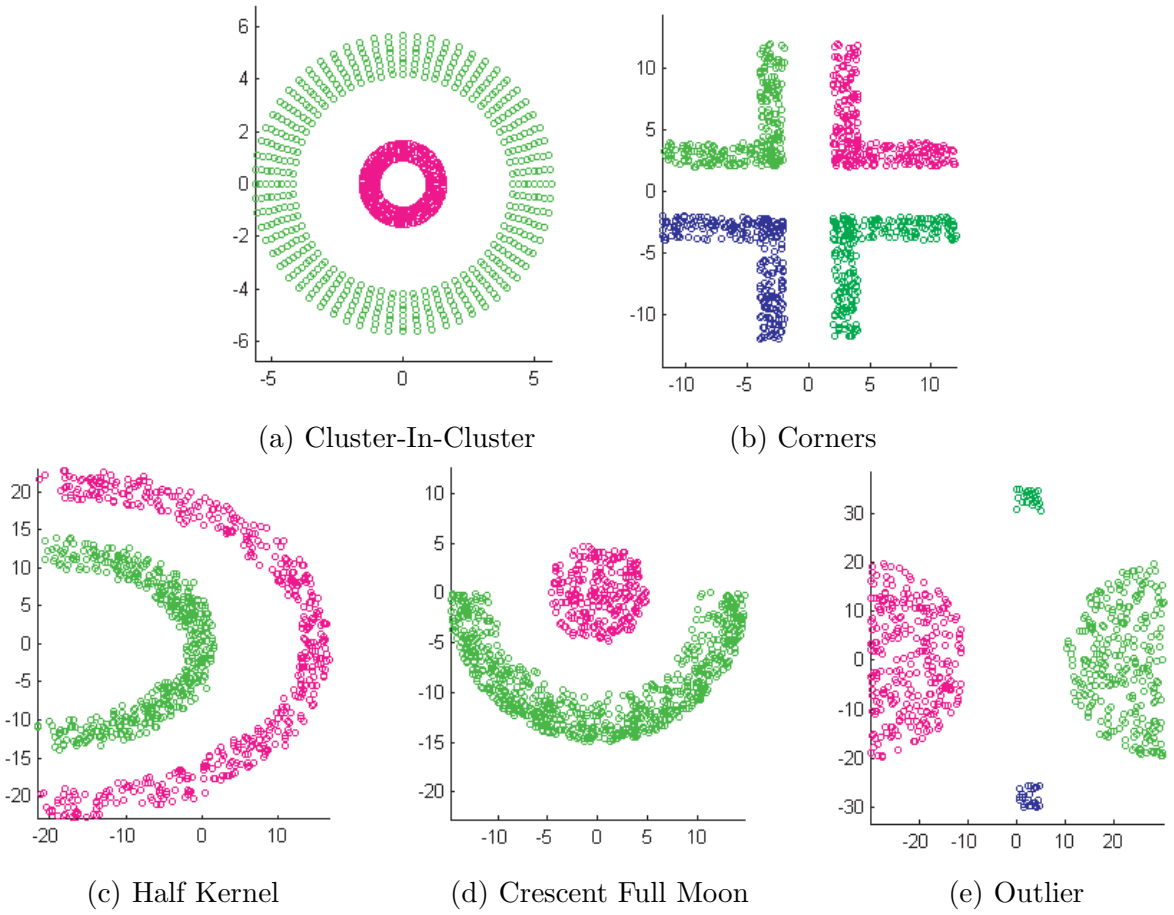
Dataset	**Classes**	**Size**
Cluster–in–Cluster	2	1012(500,512)
Corners	4	1000 (250,250,250,250)
Half Kernel	2	1000 (500,500)
Crescent Full Moon	2	1000 (250,750)
Outlier	4	600 (276, 24, 24, 276)
Aggregation	7	788 (45,170,102,273,34,130,34)
Compound	6	399 (50,92,38,45,158,16)
Jain	2	373 (275,97)
Spiral	3	312 (101,105,106)

4.2 Data Sets

We have used five 2D artificial data sets Cluster-in-Cluster, Corners, Half-kernel, Crescent & Full Moon, Outlier as well as Aggregation [GMT07], Compound [Zah71], Jain [JL05] and Spiral [CY08] data sets to illustrate the experimental studies. Applied data sets consist of different unified densities such as, clusters inside clusters, multi-density, connected clusters, and well-separated densities. It allows to evaluate the effectiveness and efficiency of the MPCACS algorithm over different shapes of data sets. These data sets are summarized in table 4.1 and depicted in figure 4.1 & figure 4.2. We have implemented the artificial data sets using MATLAB software.

4.3 Experimental Setup

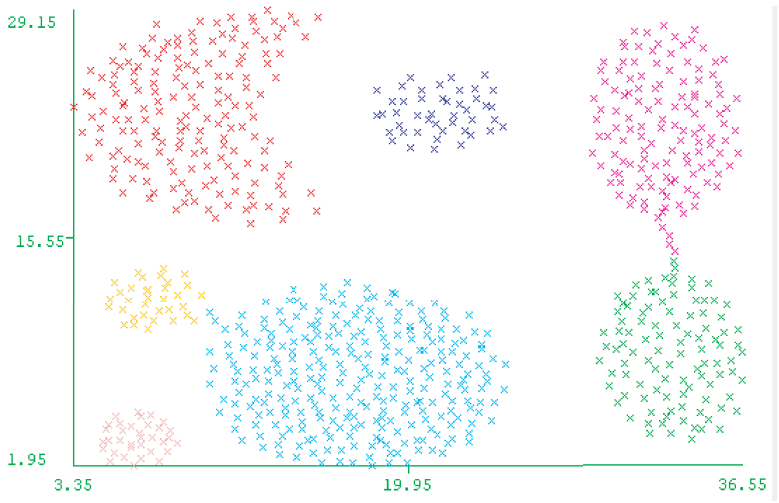
For implementing our own MPCACS algorithm, we have coded in netbeans IDE in Java. We have also implemented the performance criteria in the codes. For the implementations of the built in algorithms we have used the software WEKA. Weka have their own implementation and interpretation of the algorithms and the accuracy of the algorithms are also calculated in their code. But we have implemented other performance criteria like precision, recall, F-measure



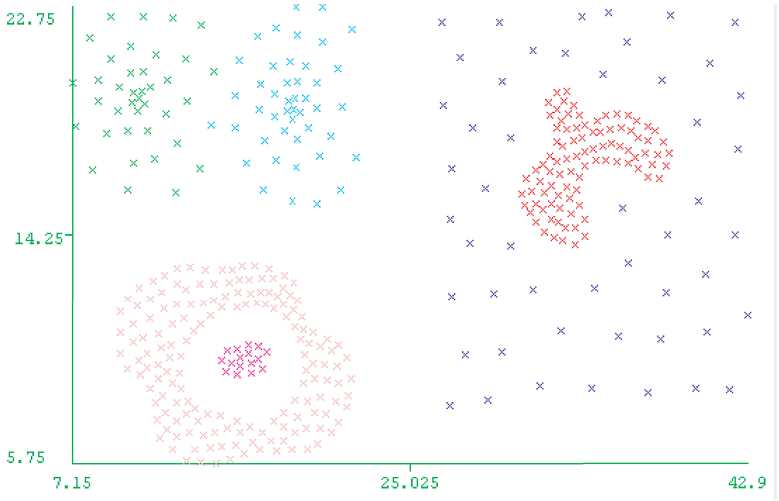
(a) Cluster-In-Cluster (b) Corners

(c) Half Kernel (d) Crescent Full Moon (e) Outlier

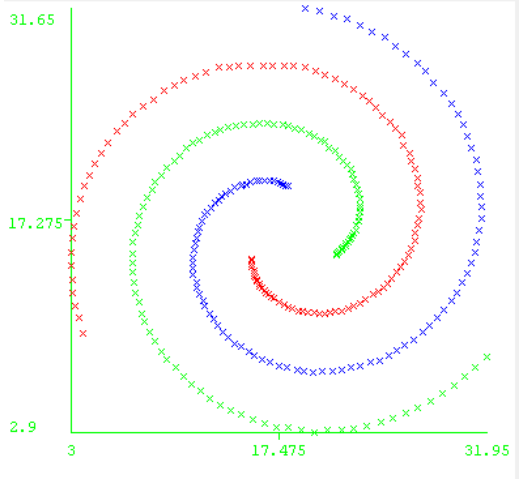
Figure 4.1: Different types of Artificial Datasets Generated by Matlab



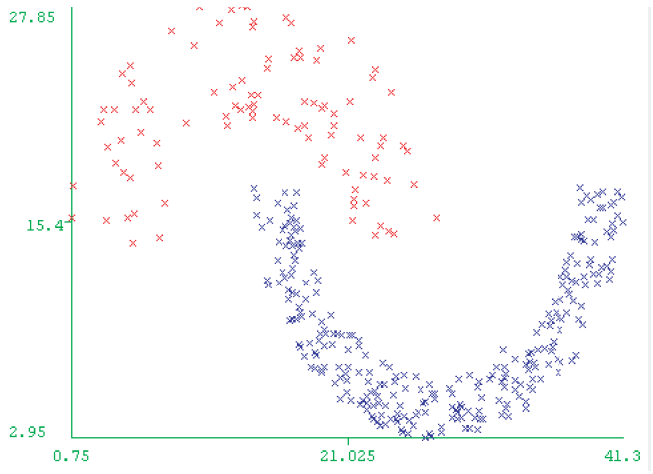
(a) Aggregation



(b) Compound



(c) Spiral



(d) Jain

Figure 4.2: Different types of Shape Datasets

and G-mean inside the weka interface.

4.3.1 Performance Evaluation Criteria

We have established five criteria for performance evaluation – Accuracy or purity, Precision, Recall, F-measure and G-mean. Accuracy is a measurement of overall effectiveness of a classifier. Precision is a measure of the accuracy provided that a specific class has been predicted. Recall is a measure of the ability of a prediction model to select instances of a certain class from a data set. F-measure is a combination of precision and recall. Specificity corresponds to the true-negative rate. Geometric mean (g-mean) is used to quantify the classifier performance in the class imbalance problems by several researchers [SH07, KAKP07]. G-mean is used to evaluate performance of imbalanced data set [NBP09]. G-mean indicates the balance between classification performances on the majority and minority class.

We have used the confusion matrix approach to implement our performance evaluations. A confusion matrix [PFK98] contains information about actual and predicted classifications done by a classification system. The confusion matrix shows how the predictions are made by the model. The rows correspond to the known class of the data, i.e. the labels in the data. The columns correspond to the predictions made by the model. The value of each of element in the matrix is the number of predictions made with the class corresponding to the column for examples with the correct value as represented by the row. Thus, the diagonal elements show the number of correct classifications made for each class, and the off-diagonal elements show

		Predicted class		
		A	B	C
Known class (class label in data)	A	25	5	2
	B	3	32	4
	C	1	0	15

Figure 4.3: Confusion Matrix for a 3 Class dataset

the errors made. So the accuracy can be defined as:

$$Accuracy = \frac{\sum n_{ii}}{n}$$

where n_{ii} is the number of correct classifications for each class which is the diagonal element in confusion matrix and n is total number of classes.

Let us consider a three class problem with the classes A, B, and C. A predictive model may result in the following confusion matrix when tested on independent data. So in the calculations below, we will use this abstract confusion matrix in figure 4.3 for notation from [19].

Here tp_{i} are true positive for C_{i}, and e_{ij} is misclassified classes which are predicted as class C_{j} when actual class is C_{i}.

Precision for class A, $\quad Precision_{\mathrm{A}} = \frac{tp_A}{tp_A + e_{BA} + e_{CA}}$

Recall for class A, $\quad Recall_{\mathrm{A}} = Sensitivity_{\mathrm{A}} = \frac{tp_A}{tp_A + e_{AB} + e_{AC}}$

Specificity for class A, $\quad Specificity_{\mathrm{A}} = \frac{tn_A}{tn_A + e_{BA} + e_{CA}}$

where $tn_A = \ (tp_b + e_{BC} + e_{CB} + tp_c)$

Measures for multi-class classification [SL09] for many classes C_{i} are as follows:

$$Precision_{\mathrm{M}} = \frac{\sum Precision_{\mathrm{i}}}{l}$$

$$Precision_{\mathrm{M}} = \frac{\sum Precision_{\mathrm{i}}}{l}$$

$$Recall_{\mathrm{M}} = \frac{\sum Recall_{\mathrm{i}}}{l}$$

$$Specificity_{\mathrm{M}} = \frac{\sum Specificity_{\mathrm{i}}}{l}$$

$$F - score_{\mathrm{M}} = \frac{2 * Precision_{\mathrm{M}} * \ Recall_{\mathrm{M}}}{Precision_{\mathrm{M}} + \ Recall_{\mathrm{M}}}$$

$$G - mean = \ \sqrt{Sensitivity * Specificity}$$

Here l is the number of clustered instances & M denotes macro-averaging.

4.4 Result

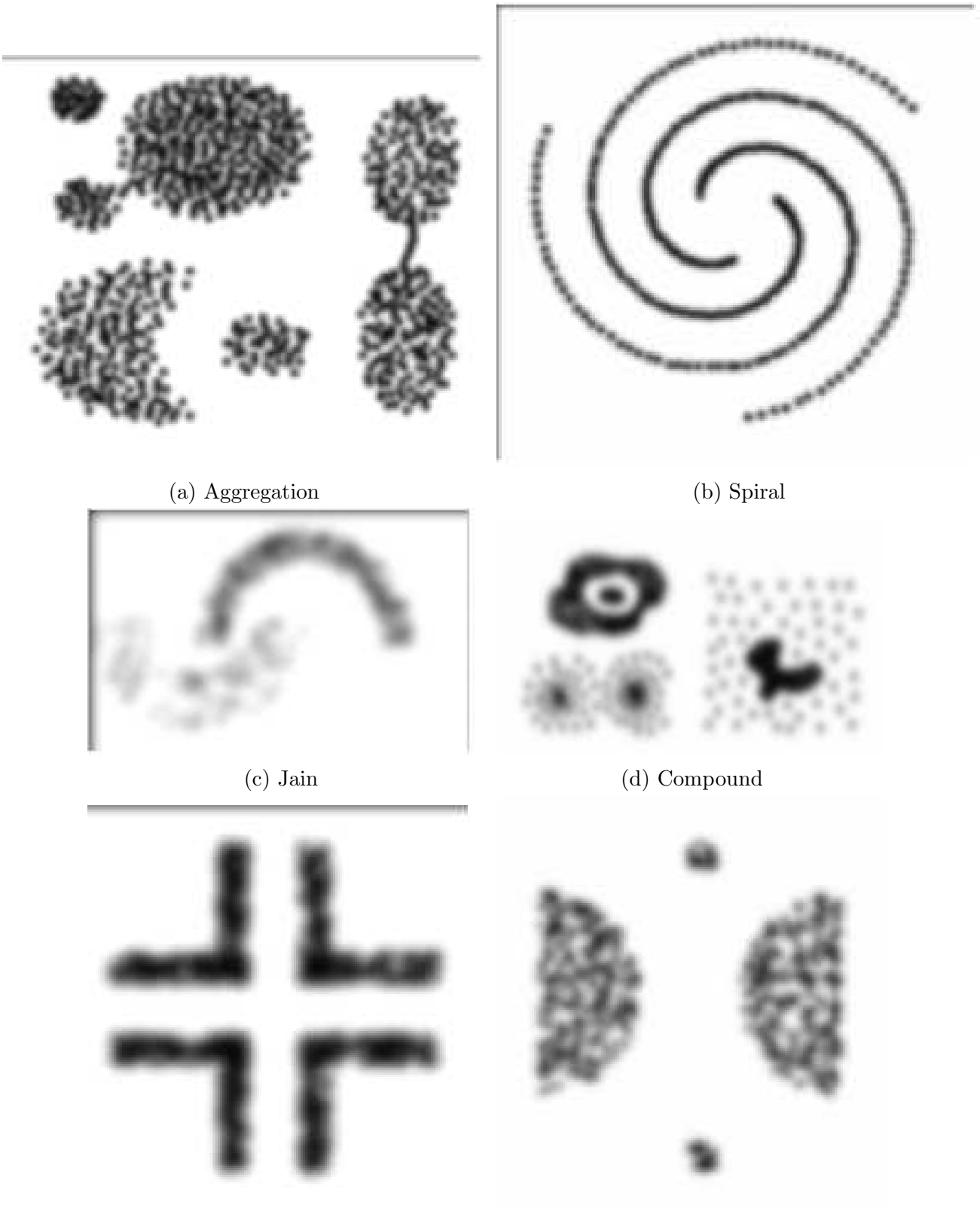
Table 4.2 shows the parameters which has to be given priori to different methods. Then table 4.3 to 4.11 shows different dataset with comparisons and in attached figures, we show the clusters produced by MPCACS for different datasets. Figure 4.4 shows some of the different augmented densities that is produced in the process . Figure 4.5 to figure 4.13 shows the actual clusters by MPCACS with the red rectangles as centres.

Table 4.2: External Parameters used by different algorithms

Algorithm	**External Parameters**
MPUCHD	Filterwidth
K-means	numClusters maxIterations seed
EM	maxIterations maximumNumberOfClusters minLogLikelihoodImprovementCV minLogLikelihoodImprovementIterating minStdDev numClusters numFolds seed
Cobweb	Seed Cutoff acuity
FarthestFirst	numClusters seed
HierarchicalClusterer	distanceFunction linkType numClusters
MakeDensityBasedClusterer	minStdDev clusterer

Table 4.3: Aggregation Results

Aggregation (Filter width = 64)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	79.89%	77.00%	87.18%	78.42%	86.81%
K-means (K = 2)	14.39%	28.57%	51.14%	19.14%	50.76%
K-means (K = 5)	66.38%	70.52%	89.59%	68.39%	83.06%
K-means (K = 7)	74.37%	74.65%	78.30%	74.51%	85.53%
EM	99.73%	83.26%	69.16%	90.75%	91.22%
Cobweb	100%	31.28%	17.39%	47.66%	55.93%
FarthestFirst	14.76%	28.57%	51.14%	19.47%	50.71%
HierarchicalClusterer	19.53%	28.57%	40.36%	23.20%	49.85%
MakeDensityBasedClusterer	14.77%	28.52%	51.02%	19.46%	50.66%



(a) Aggregation

(b) Spiral

(c) Jain

(d) Compound

(e) Corners

(f) Outlier

Figure 4.4: Augmented Density of Different Datasets

Table 4.4: Compound Results

Compound (Filter width = 64)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	68.14%	99.58%	70.43%	80.91%	82.53%
K-means (K = 2)	21.36%	33.33%	62.66%	26.03%	55.23%
K-means (K = 5)	69.60%	71.52%	84.96%	70.55%	83.14%
K-means (K = 7)	70.96%	64.59%	63.41%	67.60%	77.84%
EM	78.97%	55.52%	56.14%	65.20%	74.02%
Cobweb	100.00%	33.44%	24.31%	50.12%	57.83%
FarthestFirst	20.82%	33.33%	62.66%	25.63%	54.94%
HierarchicalClusterer	21.95%	33.33%	62.66%	26.47%	55.28%
MakeDensityBasedClusterer	21.68%	33.33%	62.66%	26.27%	55.26%

Table 4.5: Jain Results

Jain (Filter width = 16)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100.00%	86.47 %	90.61%	92.74 %	92.98%
K-means (K = 2)	84.32%	91.69%	88.20%	87.85%	91.69%
K-means (K = 5)	100.00%	53.05%	42.90%	69.32%	72.83%
K-means (K = 7)	100.00%	39.51%	33.24%	56.64%	62.85%
EM	95.75%	69.20%	54.42%	80.34%	82.50%
Cobweb	100.00%	10.56%	10.19%	19.11%	32.49%
FarthestFirst	84.15%	91.85%	87.94%	87.83%	91.85%
HierarchicalClusterer	87.10%	50.52%	74.26%	63.94%	50.52%
MakeDensityBasedClusterer	84.98%	91.57%	89.01%	88.15%	91.57%

Table 4.6: Spiral Results

Spiral (Filter width = 64)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	77.36%	77.56%	87.24%	87.96%
K-means (K = 2)	23.51%	34.76%	35.26%	28.05%	48.40%
K-means (K = 5)	23.51%	34.76%	35.26%	28.05%	48.40%
K-means (K = 7)	54.02%	25.55%	25.64%	34.7%	47.43%
EM	23.49%	34.75%	35.26%	28.03%	48.39%
Cobweb	79.18%	43.72%	42.31%	55.84%	56.28%
FarthestFirst	23.72%	35.09%	35.58%	28.31%	48.69%
HierarchicalClusterer	50.08%	66.67%	66.35%	57.19%	74.39%
MakeDensityBasedClusterer	23.72%	35.07%	35.58%	28.30%	48.68%

Table 4.7: Corners Results

Corners (Filter width = 32)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	100%	100%	100%	100%
K-means (K = 2)	25.07%	50%	50%	33.39%	64.54%
K-means (K = 5)	91.56%	74.4%	74.4%	82.10%	85.10%
K-means (K = 7)	100%	62.5%	62.5%	76.92%	79.06%
EM	100%	57.0%	57.0%	72.61%	75.49%
Cobweb	100%	34.2%	34.2%	50.97%	58.48%
FarthestFirst	26.56%	50%	50%	34.68%	64.55%
HierarchicalClusterer	25%	50%	50%	33.33%	64.54%
MakeDensityBasedClusterer	25.11%	50%	50%	33.43%	64.54%

Table 4.8: Outlier Results

Outlier (Filter width = 32)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	100%	100%	100%	100%
K-means (K = 2)	46.29%	50.0 %	92.0 %	48.07%	69.38%
K-means (K = 5)	50.32%	46.82%	44.16%	48.51%	65.80%
K-means (K = 7)	65.89%	72.37%	49.16%	68.98%	83.07%
EM	100%	71.28%	47.16%	83.23%	84.43%
Cobweb	100%	54.34%	16.0%	70.42%	73.72%
FarthestFirst	23.02%	25.18%	46.33%	24.05%	43.5%
HierarchicalClusterer	36.98%	50.0 %	50.0 %	42.51%	61.99%
MakeDensityBasedClusterer	46.29%	50.0 %	92.0 %	48.07%	69.38%

Table 4.9: Half Kernel Results

Half Kernel (Filter width = 32)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	100%	100%	100%	100%
K-means (K = 2)	52.1%	52.1%	52.1%	52.1%	52.1%
K-means (K = 5)	83.22%	39.4%	39.4%	53.48%	59.58%
K-means (K = 7)	100%	32.1%	32.1%	48.6%	56.65%
EM	100%	19.5%	19.5%	32.64%	44.16%
Cobweb	100%	6.5%	6.5%	12.21%	25.5%
FarthestFirst	53.4%	52.6%	52.6%	52.99%	52.6%
HierarchicalClusterer	100%	100%	100%	100%	100%
MakeDensityBasedClusterer	51.71%	51.7%	51.7%	51.71%	51.7%

Table 4.10: Crescent Full Moon Results

Crescent Full Moon (Filter width = 32)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	100%	100%	100%	100%
K-means (K = 2)	74.18%	82.2%	73.3%	77..98%	82.2%
K-means (K = 5)	100%	65.07%	47.60%	78.84%	80.66%
K-means (K = 7)	100%	40.47%	32.5%	57.62%	63.61%
EM	100%	20.07%	16.3%	33.43%	44.8%
Cobweb	100%	12.67%	11.4%	22.49%	35.59%
FarthestFirst	64.4%	68.0%	69.8%	66.16%	68.0%
HierarchicalClusterer	100%	100%	100%	100%	100%
MakeDensityBasedClusterer	72.85%	80.2%	70.3%	76.35%	80.2%

Table 4.11: Cluster-In-Cluster Results

Cluster-In-Cluster (Filter width = 32)					
Algorithm	Precision	Recall	Accuracy/Purity	F-measure	G-mean
MPCACS	100%	100%	100%	100%	100%
K-means (K = 2)	52.2%	52.2%	52.17%	52.2%	52.2%
K-means (K = 5)	100%	63.28%	62.85%	77.51%	79.55%
K-means (K = 7)	99.71%	30.38%	30.33%	46.57%	55.09%
EM	25.29%	50%	50.59%	33.59%	50%
Cobweb	97.7%	57.91%	57.41%	72.72%	75.2%
FarthestFirst	55.42%	55.24%	55.14%	55.34%	55.25%
HierarchicalClusterer	100%	100%	100%	100%	100%
MakeDensityBasedClusterer	51.91%	51.9%	51.87%	51.91%	51.9%

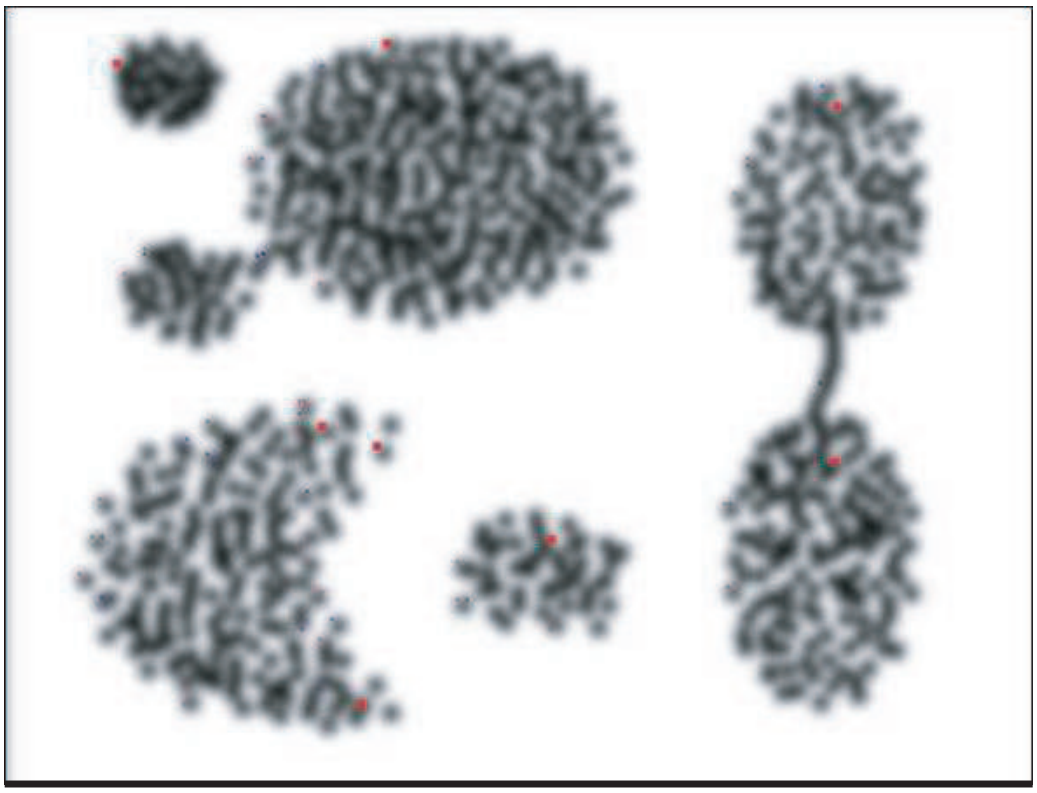


Figure 4.5: Aggregation Cluster by MPCACS

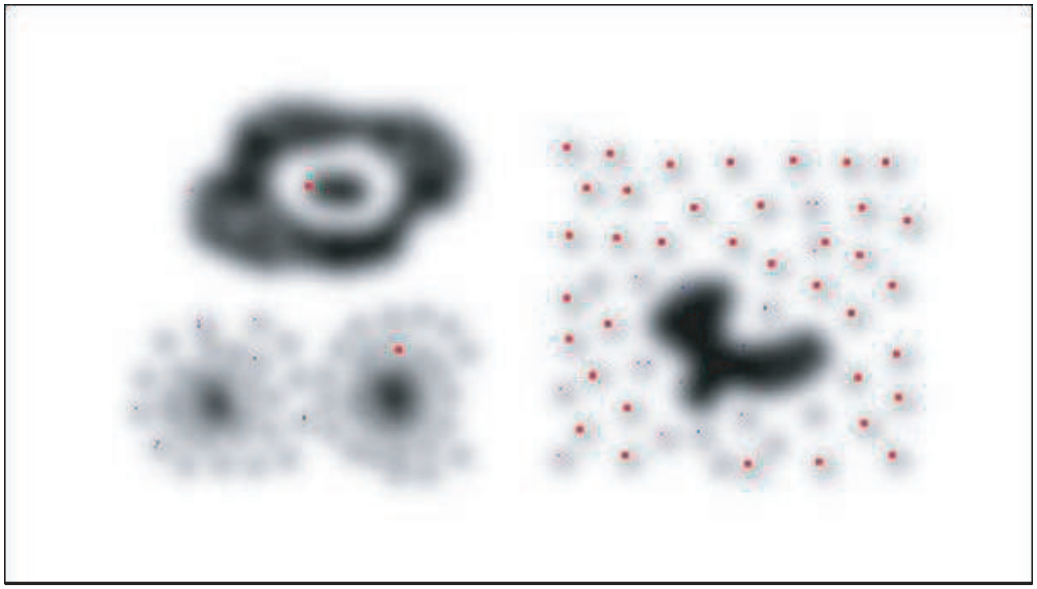


Figure 4.6: Compound Cluster by MPCACS

Figure 4.7: Jain Cluster by MPCACS

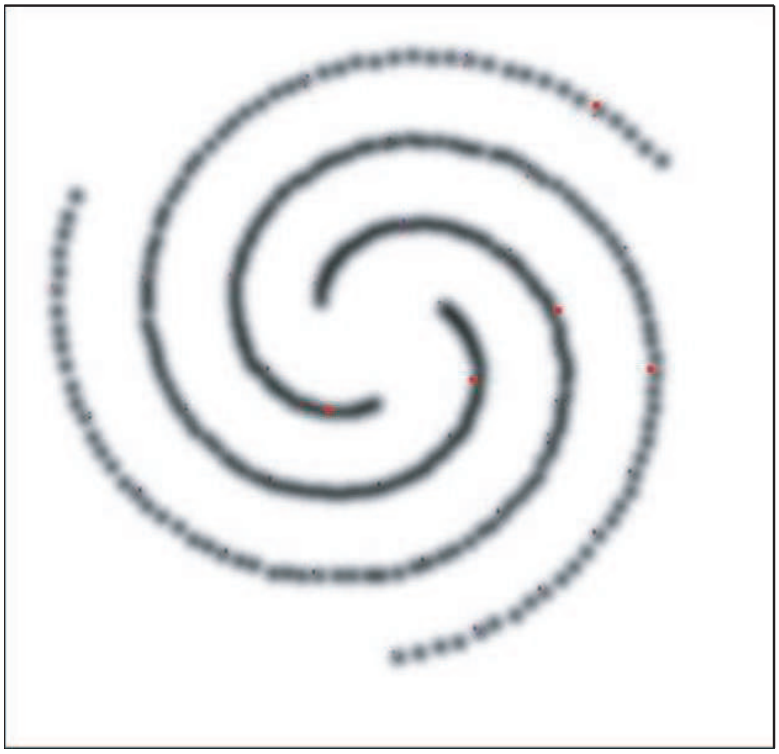


Figure 4.8: Spiral Cluster by MPCACS

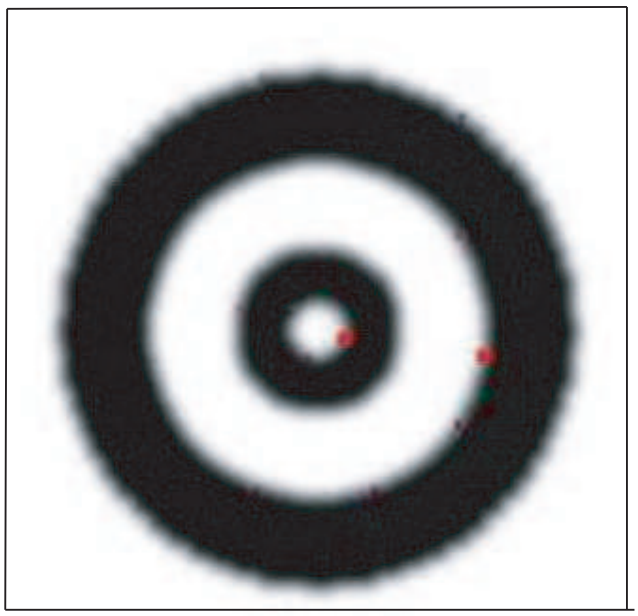


Figure 4.9: Cluster-In-Cluster Cluster by MPCACS

Figure 4.10: Corners Cluster by MPCACS

Figure 4.11: Outlier Cluster by MPCACS

4.5 Analysis

Some of the datasets worked perfectly in a single run. With the default value of filter width — 32 they give the maximum performance possible. Experiments on some specific data sets were repeated several times independently to find the optimal 'filterWidth' value which in return gives the best result. The k-means portion of tables from 4.3 to 4.11 shows the fluctuations in cluster results due to different parameter choice of K-means algorithm and the total table summarizes the results of MPCACS algorithm over nine data sets based on any improvement in the purity and g-means. Some data sets such as Aggregation and Compound have been initialized with a lower purity, then with the parameter selection technique we get the optimal result. This is shown in table 3.2. After some iterations they reached to the optimal results. In contrast, some data sets reaches the optimal results in a single run with default value of filterWidth.

Figure 4.14 & 4.16 shows the macro F-measure values and Accuracy of the compared methods.

Figure 4.12: Half Kernel Cluster by MPCACS

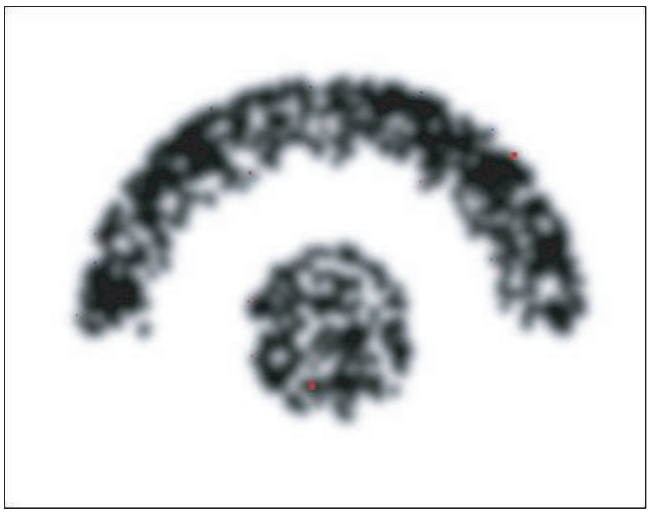


Figure 4.13: Crescent Full Moon Cluster by MPCACS

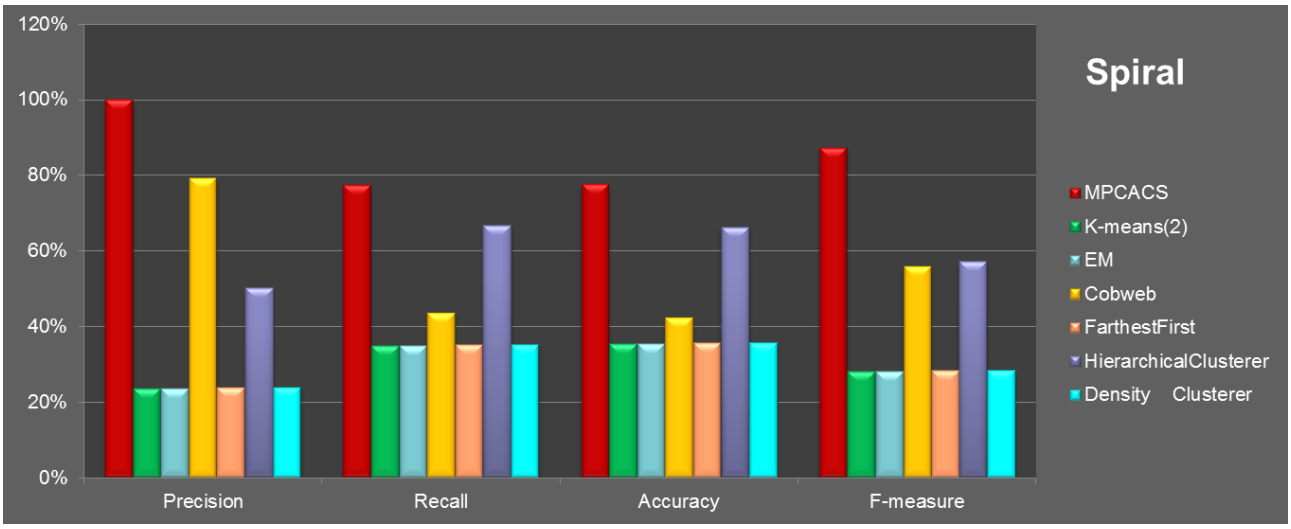


Figure 4.14: F-measure, Precision, Recall and Accuracy of the compared methods on Spiral Dataset

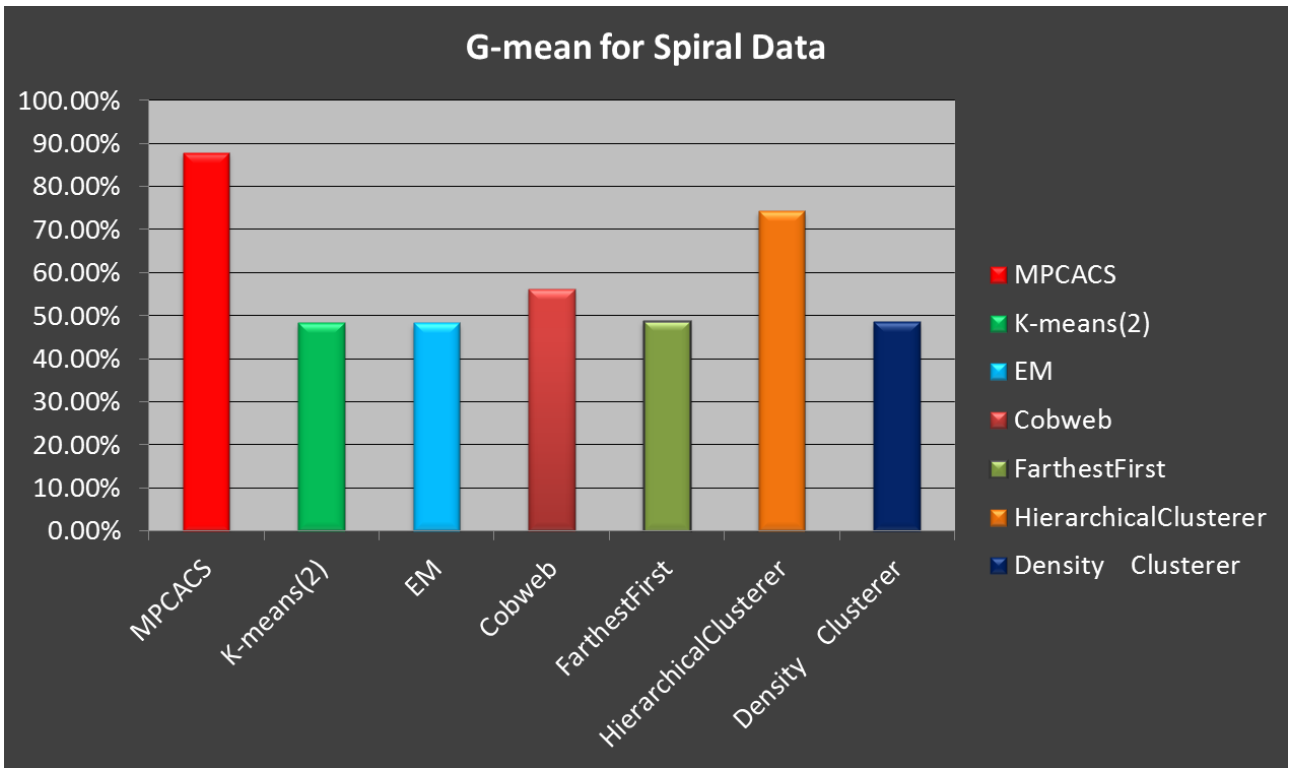


Figure 4.15: G-means of the compared methods on Spiral Dataset

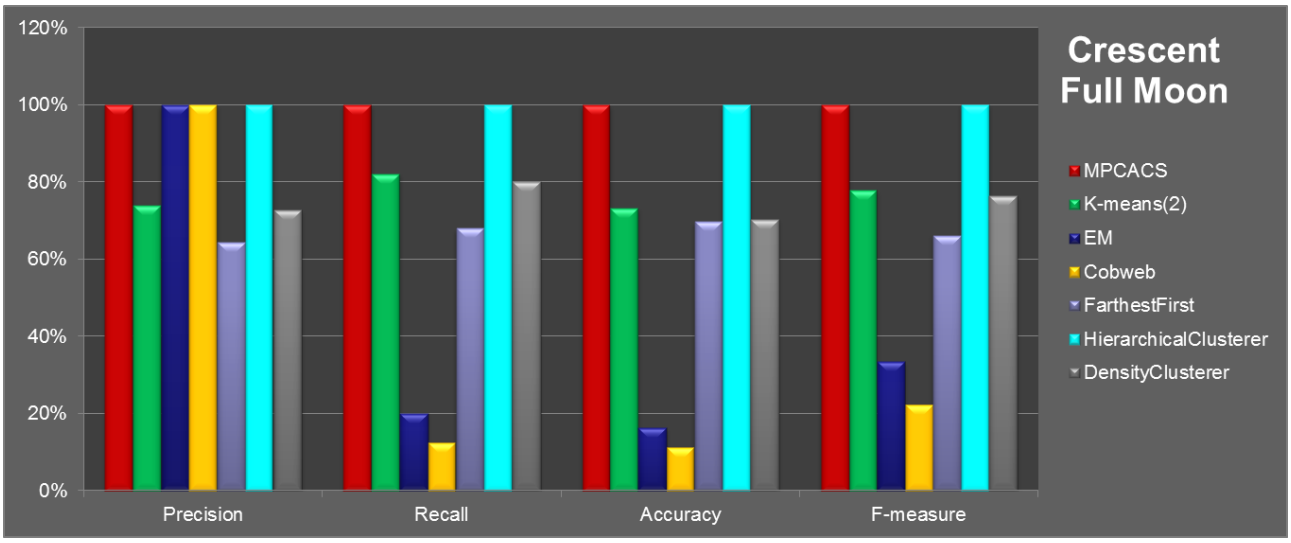


Figure 4.16: F-measure, Precision, Recall and Accuracy of the compared methods on Crescent Full Moon Dataset

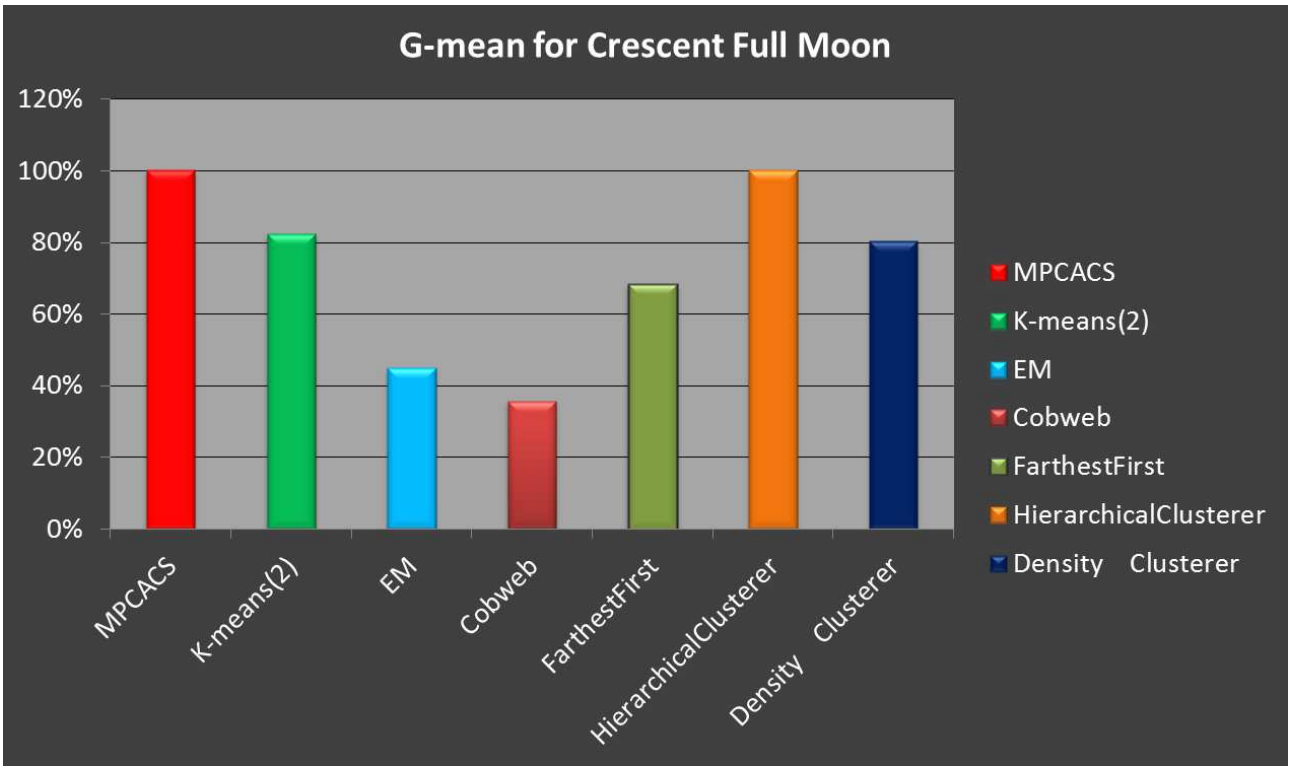


Figure 4.17: G-means of the compared methods on Crescent Full Moon Dataset

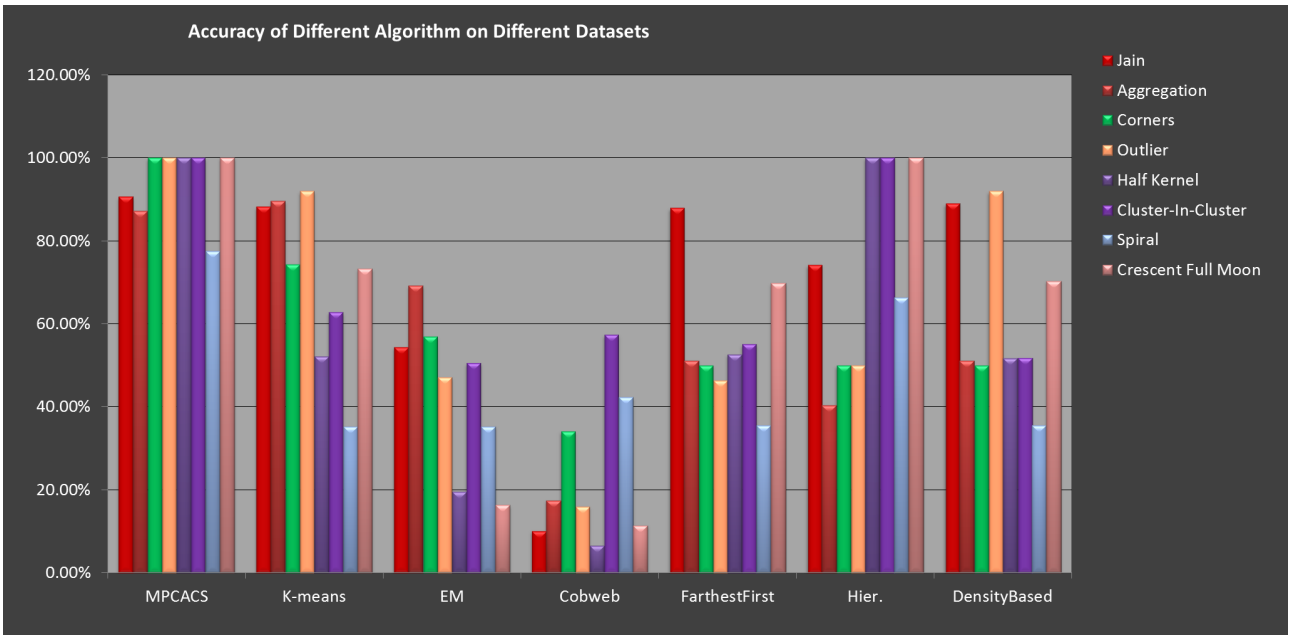


Figure 4.18: Accuracy of different methods on different datasets

The results show that MPCACS method has higher F-measure and accuracy than other compared methods on spiral & crescent full moon datasets. Figure 4.15 & 4.17 shows the G-mean values of the compared methods. The results show that MPCACS has higher G-mean than other compared methods on spiral & crescent full moon datasets. This is true for all our tested datasets.

Figure 4.18 shows that MPCACS gives maximum accuracy which is 100% for almost all data sets. And MPCACS gives almost uniform result for all type of dataset. In the case of Jain dataset 12 un-clustered instances are considered as outlier & have been dumped which shows that our algorithm also handles the outlier detection part successfully.

4.6 Comparison

The comparison results of different clustering algorithms over all data sets are shown in Figure 4.18, 4.19 & 4.20. The MPCACS have the highest accuracy for each data set than other algorithms. For examples, in the Corners data set from Table 4.7, accuracy of MPCACS is

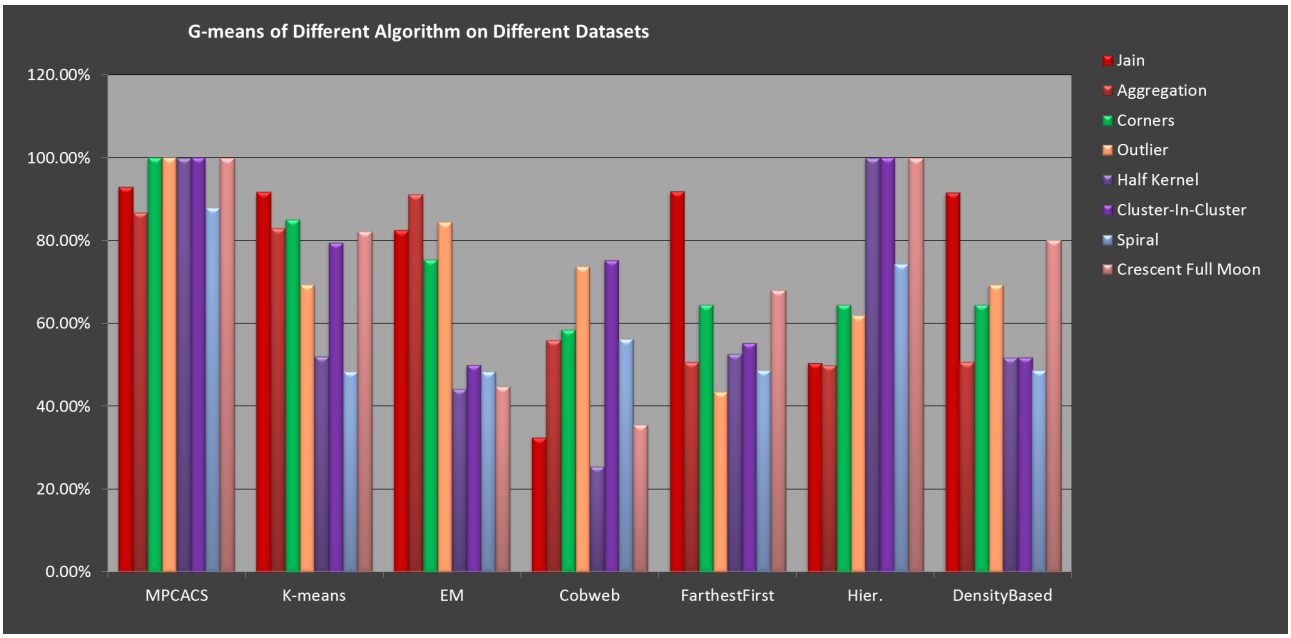


Figure 4.19: G-means of different methods on different datasets

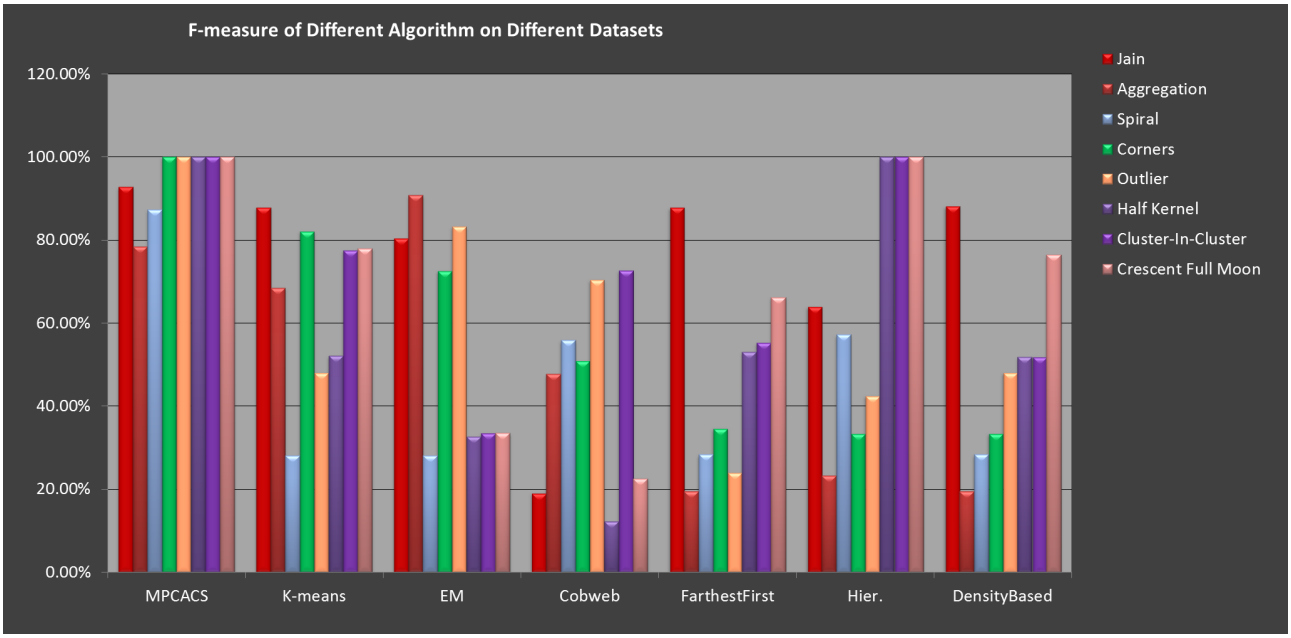


Figure 4.20: F-measure of different methods on different datasets

100% and the second best accuracy is 74.4% which is of K-means provided that K=5. Also MPCACS has higher values of G-mean and F-measure than its counterpart. In the Corners data set, the mean values of MPCACS & K-means are , 100% & 85.10% receptively. This finding suggests that the MPCACS exhibits good classification rates for all clusters. Other mentionable comparison points are given as follows:

- MPCACS is the only method which needs a single external parameter where others need to have at least more than one.
- MPCACS gives uniform accuracy over all data sets where other data sets show fluctuations over different dataset.
- MPCACS have better g-means over imbalanced dataset which means it gives good performance in both positive and negative example.
- In case of separable datasets MPCACS gives maximum accuracy where other datasets may give near maximum accuracy at most.
- In case of data set which is connected is a bad case for MPCACS i.e., compound dataset. In case of those datasets it gives slightly bad performance.

4.7 Summary

From the above experimental studies, we can come to a conclusion that MPCACS is a better algorithm than the existing clustering algorithms. In case of most type of data our algorithm gives better performance than others. Even when the accuracy is less than others it is in tolerable range. So MPCACS can be a consistent solution to the unsupervised learning problem.

Chapter 5

Conclusion

5.1 Summary of Thesis Achievements

We have proposed a novel clustering algorithm for unsupervised learning which is capable of handling outliers and noises with the distinct feature of using minimal external parameter. Our algorithm can handle data of different sizes and shapes and data of different densities also. The only parameter our algorithm needs is not a mandatory user input. It can be dynamically calculated by MPCACS easily. And it is not necessary to bracket filter-width every time. Most of the times we will get maximum performance with the default value. So MPCACS can be a better approach to unsupervised learning method via clustering for almost all possible situations and applications.

5.2 Future Work

For the connected regions which are joined by elongated shape of density sometimes gives a bad approximation. In some cases we consider the connected clusters as a single one. We would like to use 'partition' on those clusters to approximate them more accurately. We will try to

improve performance by dividing the attribute space on the basis of Fourier spectrum energy to identify the different density zone and treat them separately. We are also trying other data compression methods like single valued decomposition for data representation.

Bibliography

[19] http://www.cs.odu.edu/ mukka/cs495s13/lecturenotes/chapter5/recallprecision.pdf. Lecturenotes/Chapter5/recallprecision.

[32] http://en.wikipedia.org/wiki/fast_fourier_transform. Fast Fourier Transformation.

[Abb08] Osama Abu Abbas. Comparisons between data clustering algorithms. *International Arab Journal of Information Technology*, Vol. 5(No. 3), July 2008.

[Bel19] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey, 19.

[BZ08] Herman J. Blinchikoff and Anatol I. Zverev. Filtering in the time and frequency domains. Video Lectures by MIT, Nov. 24 2008.

[CY08] H. Chang and D.Y. Yeung. Robust path-based spectral clustering. *Pattern Recognition*, 41(1):191–203, 2008.

[Das02] Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In *15th Annual Conference on Computational Learning Theory*, volume 40, pages 351–363, 2002.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD 96*, pages 226–231. AAAI Press, 1996.

[EMT99] A. Edelman, P. McCorquodale, and S. Toledo. The future fast fourier transform. *SIAM J. Sci. Computing*, 20:1094–1114, 1999.

[Eve74] B.S. Everitt. *Cluster Analysis*. John Wiley & Sons, Inc., New York, 1974.

[Fis87] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[GLF90] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. volume 40, pages 11–61. Artificial Intelligence, 1990.

[GMT07] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):1–30, 2007.

[GRS97] Sudipto Guha, R. Rastogi, and K. Shim. Cure: A clustering algorithm for large databases. Technical report, Bell Laboratories, Murray Hill, 1997.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *In Proceedings of the 15th International Conference on Data Engineering*, 1998.

[HIKP12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. *ACM-SIAM Symposium On Discrete Algorithms (SODA), Kyoto*, January 2012.

[HK98] Alexander Hinneburg and Daniel. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1998.

[HK99] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *VLDB 99*. Edinburgh, Scotland, 1999.

[Hoc85] Shmoys Hochbaum. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JL05] A. Jain and M. Law. *Data clustering: A user's dilemma*, volume 3776. Lecture Notes in Computer Science, 2005.

[JP73] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, Vol. C-22(No. 11), November 1973.

[KAKP07] M.G. Karagiannopoulos, D.S. Anyfantis, S.B. Kotsiantis, and P.E. Pintelas. Local cost sensitive learning for handling imbalanced data sets. In *Mediterranean Conference on Control & Automation, 2007*, pages 1–6, 2007.

[KHK99] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, Vol. 32(No. 8):68–75, August 1999.

[KK98] G. Karypis and V. Kumar. Mulitlevel algorithms for multi-constraint graph partitioning. Technical report, University of Minnesota, Minneapolis, MN, 1998.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1990.

[Loa92] Charles Van Loan. Computational frameworks for the fast fourier transform. *SIAM*, 1992.

[NBP09] G. Hoang. Nguyen, A. Bouzerdoum, and S. Phung. *Learning pattern classification tasks with imbalanced data sets*, volume 3776, chapter 10, pages 193–208. P. Yin (Eds.), Pattern recognition, Vukovar, Croatia: In-Teh, 2009.

[PFK98] Foster Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 40, pages 351–363, 1998.

[SH07] Chao-Ton Su and Yu-Hsiang Hsiao. An evaluation of the robustness of mts for imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 19:1321–1332, 2007.

[SL09] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[TK08] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition, 2008.

[Zah71] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 100(1):68–86, 1971.