Mixture Models and EM Algorithm

Unnikrishnan Muraleedharan, B.Tech

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Jason Wyse

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Unnikrishnan Muraleedharan

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Unnikrishnan Muraleedharan

August 19, 2022

Mixture Models and EM Algorithm

Unnikrishnan Muraleedharan, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Jason Wyse

The main idea of this work is to understand the importance of clustering. Unsupervised Learning is essential as most of the data available in the real world are not labelled. This dissertation mainly focuses on the functioning of the EM Algorithm on Mixture Models. It also discusses the various clustering techniques like KMeans and KMeans++ algorithm and their main differences. It also establishes how KMeans can be said to be a rudimentary form of EM Algorithm. Further, the importance of initialisation in the EM Algorithm is shown by implementing the algorithm on the same dataset using different initialisations. Performance is measured with the help of the number of iterations for convergence and agreeability using a synthetic data generated. KMeans++ was observed to have better performance. The actual usage of clustering is demonstrated by applying it to a real-world dataset and observing the results. The BIC(Bayesian Information Criteria) value was used to find the ideal clusters in the data.

Acknowledgments

I would like to express my heartfelt gratitude to Prof. Jason Wyse for his constant support and guidance throughout this dissertation. The positivity and encouragement received during our one-on-one discussion have been a critical factor in helping me develop an interest in the data science field and developing an interest in statistics. I am very grateful for the opportunity to learn from him, and my sincere thanks for his feedback and guidance. I am confident that these learnings will help me in future endeavours also.

I would also like to thank the institution for its support and providing vast resources and an environment to conduct my study.

Last but not least, all these wouldn't have been possible without the unwavering support from my family and friends. My father and mother's endless support and encouragement helped me achieve this important milestone in my life.

Unnikrishnan Muraleedharan

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Machine Learning (ML) is not just a buzzword anymore and has captured the imagination of scientists and policymakers, alike, making central to the digital transformation initiatives across the world today.

ML has gained prominence in recent years for the primary reason that it generates vast amounts of data from different applications resulting from an increase in accessibility. It is important to analyse this data and discover interesting insights that will be useful in the future not only to enhance current operational efficiency but also identify predictive trends to enhance productivity.

This process can be quickly accomplished with the help of machines. The machine "learns" from the vast pool of data and improves itself over the process. The increased processing and computational powers, along with the improved algorithms, has today enabled what is referred to as "Machine Learning" to scale new heights. Any activity involving repetitive tasks that could be automated also comes under its purview. ML thus reduces human effort and increases the overall efficiency of the process.

Machine Learning evolved tremendously after the 1990s with the scientist and technologists realising that statistics and computer technology can be combined, leading to probabilistic results, which, in turn, could lead to more precise and quantified results. Machine Learning finds various applications in different fields, including Image Recognition (identifying features in an Image), Speech Recognition (converting speech into text), and predicting trends, and has enormous usage in sectors such as financial services as well as in healthcare to achieve better patient outcomes through early detection of diseases.

At a very high level, ML can be regarded as the process of generating a model based on available data. This model which is build using an algorithm can be fed with input

to predict the output. Its performance depends on various factors such as accuracy, complexity, and computational time, among others.

ML can be divided into three classes based on the process involved in generating the model. This is represented in the form of a flow chart in the paper (1) as shown in Figure 1.1.

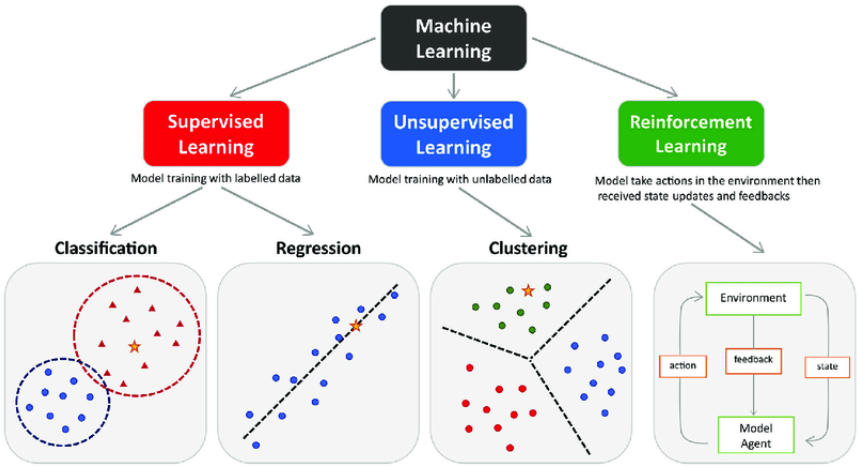

Figure 1.1: ML can be divided into Unsupervised, Supervised and Reinforcement Learning(1)

To discuss the illustration (Figure 1.1) further, Supervised Learning involves the generation of a model from labelled data. The model is created by using data and learning from it, which closely maps the data points with the label so that an input given can predict what the label could be in the future. This is usually tested by dividing the data into a train-test split of 70-30 per cent. The model is generated using the training set and then tested on remaining 30 percent data. The accuracy of the model can be understood using this. The output of Supervised learning can be Categorical or Real-Valued, and based on this, it is divided into Classification and Regression problems, respectively.

Unsupervised Learning is another crucial Machine Learning technique involving training a model from unlabeled data. This technique requires coded algorithms that act on the data without prior knowledge and finds and learns from the data. The main type of Unsupervised Learning is Clustering. The model tries to find a relationship in the subset of the parent group, which is unique to the subset and helps to separate it from the others. Association is another type of unsupervised learning that helps establish a rule that can describe a significant portion of the data. Different techniques can be used for implementing clustering and association. A working clustering approach need not work the same way in two different datasets, i.e., it varies with the data and performance

evaluation methods used.

Reinforcement Learning is a machine learning technique that generates a model based on maximising and minimising the rewards and penalties by placing an agent in an environment it has not seen before. The behaviour of the agent controls the reward and penalty.

1.2 Motivation

Technology advancement has led to applying these methods directly using numerous libraries in languages such as R (2) and Python (3) etc. This helped to create models by feeding in the data to these inbuilt functions without knowing much of the inner workings. Scikit Learn (4) is a popular ML Python library that is simple and provides different functions for implementing various machine learning tasks. It also helps us to pre-process, visualise and execute machine learning tasks. Implementing clustering and generating a model with this is straightforward. The model will work perfectly, but if there is a slight variation in the expected results, it is difficult to understand the root cause. The reasons could be numerous - from the initialisation techniques used to the hidden approximations. It is important to have in-depth knowledge of these techniques from the ground up to prevent this.

This dissertation's primary focus is on understanding the concept of Clustering, which falls under Unsupervised Learning. The reason for choosing Unsupervised learning is because most of the data generated in the present world are not labelled. A massive amount of human effort and time is involved in labelling the data. In addition, implementing this ML technique will help us to identify any unknown patterns in a simple, time-efficient and less costly manner. The manual intervention involved is also reduced. This method is considered a type of exploratory data analysis in which one is unaware of the results beforehand. Implementing this would help to segregate the data without any biases such as prior knowledge about the data types that would have come, if a human was involved in the process.

The collection of individual pieces of data that are related to a parent subject and are grouped is called Dataset. It could be readings or observations of a particular subject over a period of time under certain circumstances. When the dataset is analysed, it can be observed that they could be drawn from different probability distributions such as Uniform, Normal and Binomial etc. In addition, there can be cases wherein it is a distribution formed by the superposition of multiple distributions, and such a distribution is called Mixture Models. The majority of the datasets follow such behaviour. This can be found only by exploring the dataset and running various tests to establish the same.

Implementing clustering techniques would help us to understand or separate the different clusters from such data. This has multiple applications in fields including advertising, healthcare image compression and data mining, among others.

EM Algorithm (5) is a type of clustering technique that can be applied to a dataset. It is an important probabilistic clustering technique. Once clustering is done using this technique, we can define a probability value for each point belonging to a particular cluster. This technique is critical when the data under consideration involves heterogeneous data, i.e. data with missing values and low redundancy. One application in which the EM algorithm plays a vital role is in survival analysis. This has emerged from actuarial statistics, spanning hundreds of years of use in evaluating annuities, insurance rates and life expectancy (6). This data can be considered at a high level as a dataset with missing data because it involves collecting data from people over time at regular intervals, which would include challenges such as failure to collect data during a period. In this scenario, EM algorithm can be applied such that the missing data is filled with sensible values and the EM is run, which would make these values more accurate over the iterations. In terms of clustering, this technique also gives us the probability with which a particular data point belongs to a cluster, which offers a more detailed clustering information. Knowing the group's details would enable us to define a function with the data provided, find missing values and even predict to which cluster a new data would fall.

At a deep level, the dissertation focuses on the Mixture Models and EM algorithms. Scikit Learn library has a inbuilt function to implement the EM Algorithm (4). However, it is essential to understand the inner working of the inbuilt library functions and clearly understand the black box mystery involved in generating the model.
This dissertation tries to find answers to the following questions:

- The different types of techniques involved in clustering.
- The importance of initialisation and different types of techniques involved.
- Are we fully aware of the assumptions used by functions defined in libraries?
- Do we know the underlying concepts that are applied to get the output?
- If we write a similar function, will the output be the same?

1.3 Contribution

The dissertation's primary objective is to understand the concept of clustering. To understand this, we need to generate a well-defined dataset and implement clustering techniques

and study how well the method will be able to define the parameters. KMeans (7) (8) is one of the most essential and efficient clustering techniques. However, KMeans has a non-probabilistic approach for clustering. EM algorithm gives a probabilistic approach toward clustering, and KMeans is considered a particular case of EM algorithm (9). There are applications in which KMeans are needed, such as image compression, in which we can compress the image size by reducing the number of colour combinations, as shown in Figure 1.2. KMeans and EM algorithm is developed from the root level with a clear understanding of each step involved. The developed algorithms are applied to the synthetic dataset generated, and the results are studied. EM Algorithm consists of initialisation in the beginning. This has a significant impact on the results. Usually, KMeans is done on the dataset, and the parameters obtained are used for this initialisation. KMeans++ (10) is another critical approach discussed in this dissertation. In simple terms, it is an intelligent initialisation technique for the initialisation of centroids, which guarantees that the centroids selected initially are widely separated. This algorithm is also developed and studied in this dissertation.

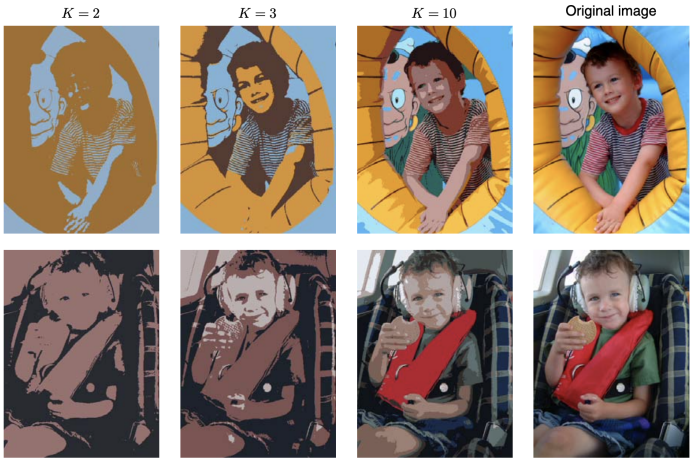

Figure 1.2: Application of KMeans Algorithm to Images. It shows how varying K can help in the compression of the image. The original image has a palette of K colours. Applying the KMeans segments the image based on the value of K. On increasing, K decreases the quality and size of the original image. .(9)

The final objective of the dissertation is to apply these developed algorithms to a real-world dataset and observe the results. The Galaxy dataset (11) is chosen for this, which remains a debatable dataset across the scientific community. The aim is to determine the clusters from the data points. The data points represent the velocities with which the galaxies shift away from our galaxy. The clusters found by running the EM algorithm would give us an idea about the number of galaxies in the universe.

1.4 Dissertation Structure

Chapter 1: : This chapter gives an overview and clear objectives on what the dissertation is trying to achieve.

Chapter 2: : This chapter covers a detailed study of the background of this dissertation. Different approaches such as the KMeans, KMeans++ and the EM Algorithm are discussed in detail.

Chapter 3: This chapter covers the methodology and the process involved in implementing the clustering techniques in this dissertation.

Chapter 4: This chapter discusses the findings and observations obtained by implementing the techniques on a synthetic dataset. It studies and observes the results obtained on the same dataset using the inbuilt library functions. It also shows the difference in the parameters obtained by choosing different initialisation techniques in EM.

Chapter 5: This chapter discusses the limitations of the current algorithms developed and also throws light into the future works that could be carried out, leading to more understanding of the clustering techniques.

High level structure of the dissertation is shown in Figure 1.3

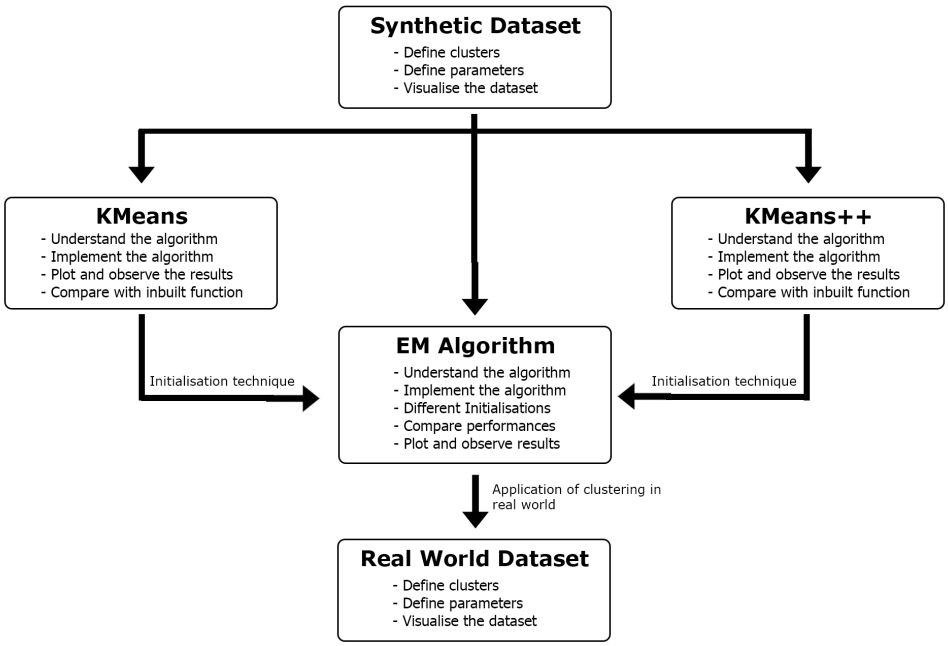

Figure 1.3: Structure of Dissertation

Chapter 2

State of the Art

This chapter gives a deeper understanding of the background of this dissertation. The different clustering techniques used and the importance of each of them are discussed. It dives into the basics of each step during the algorithm, and how the individual clusters are separated and the parameters are estimated. Since we are dealing with mixture models, we need to understand what they are and how they differ from other models. Later we will discuss the KMeans and KMeans++ algorithms, which are simple clustering techniques. It also discusses the shortcomings of these algorithms and how it is used as an initialisation technique in the EM Algorithm. And finally, the EM Algorithm is discussed.

2.1 Mixture Models

Suppose we have a random variable x, with $p(x)$ as the probability distribution . If we decide to study the random variable, we need to be in a place to define a model that will have a similar probability distribution. This model can be used to understand the parameters of the distribution. However, this is not simple, as there could be an infinite number of probability distributions that would lead to the generation of the data under observation. The distribution could be binomial or multinomial if the random variables under consideration are discrete and independent. On the other hand, if the data under consideration is continuous, it could be from a Gaussian or Normal Distribution.

The parameters such as mean, variance and weights can be used to define the distribution. It is a parametric method of modelling. Fitting a model to a dataset means that we are adjusting the parameters of a model such that they will represent the probability distribution function of the data we are exploring. A frequentist approach is chosen for fitting the model by selecting the parameters and then optimising them based on some criteria. The limitation of this approach is that the accuracy of the model is critical. It

means that a poor model will generate poor predictive performance.

Mixture models are used to deal with real-world data. These models are frequently utilised in machine learning, statistical analysis, pattern recognition, and data mining. In theory, we can create a dataset that can be easily fitted using single distribution functions. But in a real-world scenario, the case would be such that the data would be represented more accurately by fitting multiple individual distributions. This would represent the probability distribution function of the dataset more accurately.

For example, consider the "Old Faithful" dataset (12), which comprises 272 measurements in 2 variables. Each measurement represents the waiting and eruption times (mins) for an Old faithful geyser. Figure 2.1 represents the distribution plot of the data. It can be seen that fitting three Gaussian distributions model fits the data more accurately than a single Gaussian distribution model. These super positions, which are created by linearly combining more fundamental distributions like Gaussians, can be represented as probabilistic models called mixture distributions. (9). In summary, we can say that if we can properly select the number of Gaussians, mean, variance and mixing coefficients (the parameters of the combination), the probability distribution of most of the data can be represented to a reasonable level of accuracy.

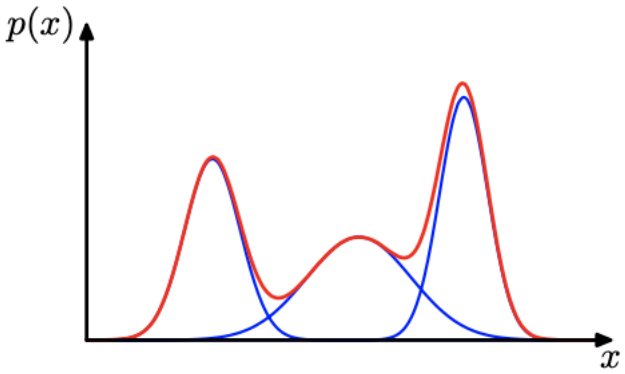

Figure 2.1: The Gaussian distribution function plot in the Old Faithful Dataset(1-D). The blue line shows three individual clusters, and the red shows their superposition. The aim of doing clustering is to fit a model with a distribution function similar to the red line. (9)

The normal distribution in a single dimension is shown in Equation 2.1.

$$N\left(y_i;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-\left(y_i-\mu\right)^2}{2\sigma^2}\right) \tag{2.1}$$

where μ and σ^2 represents the mean and variance of the data.

A representation of mixture of Gaussian's which is a superposition of individual Gaussian is shown in Equation 2.2:

$$p(x) = \sum_{g=1}^{G} \pi_g . N\left(y_i \, ; \mu_g \, , \sigma_g^2\right) \tag{2.2}$$

Here π_g is called mixing coefficients or weights, and it satisfies the following condition:

$$\sum_{g=1}^{G} \pi_g = 1 \tag{2.3}$$

We also know that $N(y_i \, ; \mu_g \, , \sigma_g^2) \ \geq 0$, which is a condition needed for the probability $p(x) \ \geq 0$, for all G (groups)
Looking at this condition and Equation 2.3, we can see that the value of the mixing coefficient lies between 0 and 1, and the sum of all will be equal to 1.

The parameters represent the shape of the Gaussian. Let $\{\pi_1, \pi_2 ... \pi_G\}$, $\{\mu_1, \mu_2 ... \mu_G\}$ and $\{\Sigma_1, \Sigma_2 ... \Sigma_G\}$ be the parameters. To assign values to this, we can use maximum likelihood. From Equation 2.2, we compute the log-likelihood using:

$$\begin{aligned} \log\left(p\left(x|\pi, \mu, \sigma^2\right)\right) &= \sum_{i=1}^{n} \log\left[\sum_{g=1}^{G} \pi_g . N\left(y_i \, ; \mu_g \, , \sigma_g^2\right)\right] \\ &= \sum_{i=1}^{n} \log\left[\sum_{g=1}^{G} \pi_g . \frac{1}{\sqrt{2\pi\sigma_g^2}} \exp\left(\frac{-\left(y_i - \mu_g\right)^2}{2\sigma_g^2}\right)\right] \end{aligned} \tag{2.4}$$

This is a complex calculation as it involves summing the natural logarithm values over g. This will prevent us from getting a closed-form solution. One of the methods that could be employed to solve this would be using the EM algorithm. In a nutshell, what it does is that the parameters are initialised. With these the log-likelihood is calculated. The parameters are again re-estimated. This process is repeated till the convergence of log-likelihood value. Over each iteration, the likelihood value increases. The EM algorithm is discussed in detail in the coming sections.

2.2 KMeans

The main objective of the clustering technique, which comes under Unsupervised Learning, is to locate clusters within data. Clustering is defined as the process of finding homogeneity in a subset within the whole dataset. The data points within a cluster will

have some similarities that will help to separate them from the others in the dataset. KMeans (7) (8) is a well-known simple clustering technique still used for several applications and is based on the work of Llyod in 1982. It is a non-probabilistic clustering technique, i.e., once clustering is done, every data-point is allocated to a cluster, and no details about the degree of the association to the clusters are given. The goal of KMeans is to assign each data point in the dataset to a cluster such that the sum of the squares of the distances to the centroids of the clusters is as small as possible.

Mathematically, it can be expressed as follows:
Consider a dataset having N number of observations in the dimensional space, D which is represented as $\{x_1, x_2...x_N\}$. We aim to find clusters within this group satisfying the condition of KMeans. Suppose we have K clusters. μ_k is a parameter that represents the centroid of the k^{th} cluster. We define a variable called responsibility represented by r_{nk}, which can take values of either 0 or 1. It takes the value 1 when a point comes from that cluster and 0 otherwise.
The objective function can be defined represented by Equation 2.5.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \tau_{nk} \|x_n - \mu_k\|^2 \tag{2.5}$$

In Equation 2.5, r_{nk} and μ_k are the unknowns. We need to compute the parameters in order to minimise the objective function. There is a need for an iterative process such that in one of the steps, the μ_k is kept constant, the r_{nk} is computed to minimise J, and in the next step, the r_{nk} is kept constant, and the μ_k is computed to minimise J.

In simple terms, the algorithm first initialises the number of clusters. Then the centroids of these clusters are defined. We then compute the euclidean distances from each data point to all the centroids. These points are allocated to a cluster which has the least distance. Once this is done, the centroids are calculated with the allocated points. Again, the distance from the points to these new centroids is computed. The points are again assigned to the clusters similarly as above. The procedure is repeated till the values of the determined centroids remains unchanged or falls within a threshold selected.

Mathematically, it can be represented as below:

$$\tau_{nk} = \begin{cases} 1 & \text{if } k = \arg min_j \|x_n - \mu_j\|^2 \\ 0 & otherwise. \end{cases} \tag{2.6}$$

$$\mu_k = \frac{\sum_n \tau_{nk} x_n}{\sum_n \tau_{nk}} \tag{2.7}$$

Consider a cluster K=1. Then, the $\sum_n r_{nk}$ equals the number of points assigned to a cluster, and $\sum_n r_{nk}x_n$ will be the sum of the points in that cluster. With this, we can say that this expression represents the cluster's mean.

Figure 2.2 represents the KMeans clustering done for the famous Old Faithful Dataset (12). The data has been standardised such that the mean=0 and the standard deviation=1.

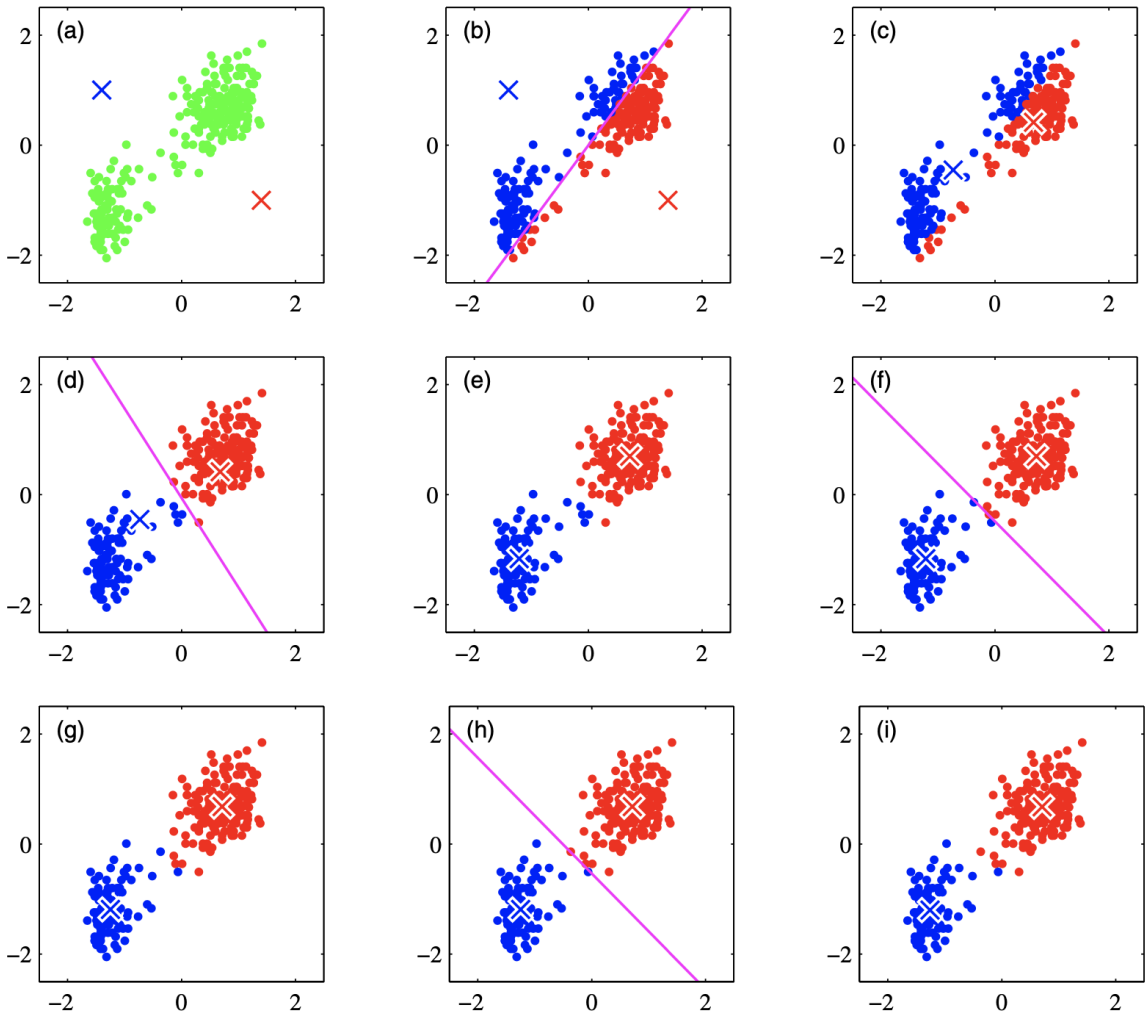

Figure 2.2: KMeans Algorithm working in the Old Faithful Dataset. The data is rescaled with a mean=0 and a standard deviation=1. Data points are plotted in green and shown in (a). Centroids are shown using the cross. (b), (c), (d), (e), (f), (g), and (h) represents the change in the centroids during each of the iterations in the KMeans Algorithm. (i) shows the final clustering obtained when the centroids converge.(9)

The initialisation of the initial parameters such as the clusters present and the centroids plays a crucial role. The accuracy and number of iterations taken to converge depend on these initialisations. Additionally, there is a chance that the convergence will occur to the

local minima rather than the global minima if the initialisation is incorrect. The ideal choice is to choose centroids within the dataset. This prevents choosing centroids that lie outside the dataset's scope, resulting in incorrect clustering. It can also be noted that the shape of the clusters formed in KMeans is spherical. It doesn't give the flexibility of changing the shape of the clusters to fit the points accurately. Consider a case where a point lies almost close to the centre of the two clusters; in this case, point will be allocated to one cluster, which is inappropriate. This is why this algorithm is also called a hard clustering technique.

Despite being widely used for generic clustering, KMeans has three significant drawbacks: it does not scale well, it is has a tendency to converge to local minima and the number of clusters(K) is an input given by the user. (13). Having an idea about the dataset and placing close points in the same subtree as discussed by Ramasubramanian, V. and Kuldip K. Paliwal in the (14) can improve the computational speed.

One simple method to determine the cluster number (K) is plotting the sum of the squared distance among the centroids and points. This method is called as Elbow method. Figure 2.3 shows the plot of the same for a synthetic dataset generated having 3 clusters. It can be seen that at K=3, there is a bend. For that particular dataset, that is the optimum value of K. At K=3, the sum of the squares of the distances between the points, and the centroids within the clusters are ideal, beyond which it slowly converges. Since we know that there are 3 clusters, the result we got from the Elbow plot supports, and hence it can be used as a technique to initialise the value of K.

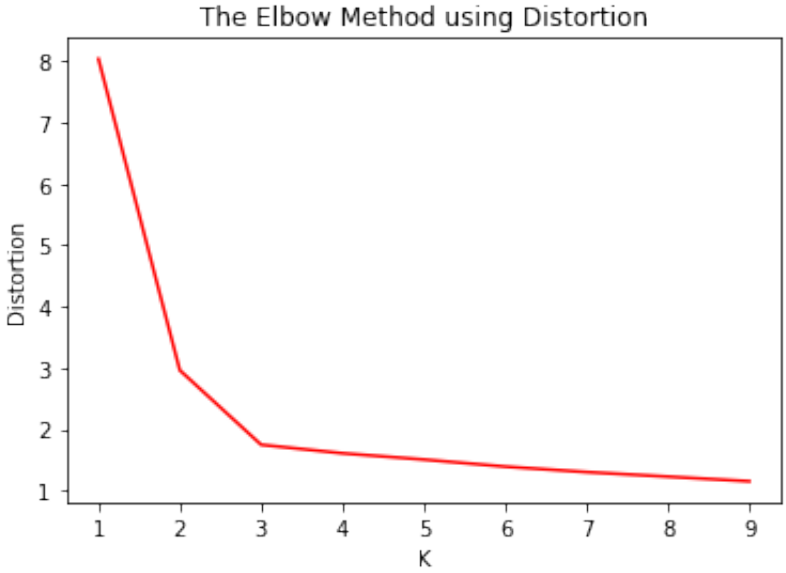

Figure 2.3: Elbow Plot: Distortion vs K

To have a good clustering using KMeans, we must ensure that the centroids of the

initial clusters chosen are well spread within the dataset scope. This leads to the requirement for new techniques for selecting the centroids, and KMeans++ (10) is an excellent method to achieve the same. The following section discusses in detail the KMeans++ algorithm.

2.3 KMeans++

To overcome the shortcomings of KMeans, the initial centroids must be carefully selected. Arthur, David and Vassilvitskii, Sergei in (10) discuss the advantages of careful seeding in the KMeans algorithm. KMeans++ is an algorithm built as an initial step to be done before doing KMeans. It is an intelligent initialisation technique for the initial centroids. Unlike the standard method, in this, the centroids are chosen based on a probability distribution of the weighted distances. This ensures that the centroids are widely separated, which helps to achieve accurate clustering. KMeans++ improves the computational speed by reducing the iterations required for the centroids to converge.

Consider a case where the dataset has N number of observations in the D dimensional space. The user defines the clusters present. One data point is chosen at random as the cluster's centroid. D(x) is defined as the euclidean distance from a point to the closest centroid. Now, determine the next centroid as a point in the dataset in such a way that it has a probability of square of the distance from that point divided by sum of the square of the distance of all points.

Mathematically, it can be expressed as follows: We have the dataset represented by X
First centroid c_1 is x, such that $x \in X$
Next centroid c_2 is x', such that $x' \in X$ and has a probability $\frac{D(x')}{\sum_{x \in X} D(x)^2}$, where $D(x)$ represents the shortest distance between points and previously selected centroid, c_1. This process is repeated till we have the centroids for all the clusters.

Once we get the required centroids, the KMeans algorithm is done, and the clustering is achieved. The advantage of KMeans++ is that it reduces the chances of outliers being selected as the initial centroids. Since it involves the selection of centroids with a probability, there is more clarity on the initialisations. It also increases the chances that these points could be the centroids or lies closer to the centroid, which helps quicker convergence.

Arthur, David & Vassilvitskii, Sergei in (10) also study the running time and the accuracy of KMeans++ applied to four different datasets, out of which three are real-world datasets (Cloud Dataset (15), Intrusion Dataset (15) and Spam Dataset (15)). They studied this by choosing the value of clusters as 10, 25 and 50, and in each K, the

algorithm was run 20 times since the first centroid was randomly selected. The minimum values of the centroids, average potential and mean running time was reported. The results show KMeans++ performs much better compared to KMeans in all the cases. The running time was also reduced when KMeans++ was used. In the Synthetic dataset, KMeans failed to cluster the dataset as it merged some clusters incorrectly. On the other hand, the KMeans++ showed clear separated clusters in the same dataset. In the real-world dataset, it was seen that KMeans++ gave a minimum of 10% more accuracy than the KMeans algorithm. Notable improvement was observed in the Spam and Intrusion dataset. Significant improvement in the convergence time was also observed by using this careful seeding technique.

Due to its simplicity, it is used as a default method for initialisation in the EM Algorithm, which will be discussed in detail in the coming section.

2.4 EM Algorithm

We had briefly discussed the need for the Expectation Maximisation algorithm when we discussed the Mixture Models. Here we will see, in detail, how the algorithm works. The EM algorithm (5) can be called a probabilistic clustering approach. It is used in Mixture Models to estimate the distribution parameters with the help of finding the maximum likelihood. One of the methods that can be employed to find the maximum likelihood of distribution is using the EM algorithm.

The following shows how the EM algorithm can determine the parameters of the Mixture Model.

First, consider a dataset whose labels are represented by $Z_i \sim$ Multinominal Distribution having mixing coefficients or weights as: $\{\pi_1, \pi_2 ... \pi_G\}$, here G represents the group number and it satisfies the conditions $\sum_{g=1}^{G} \pi_g = 1$.

Z_i represents the group, $Z_{ig} = \begin{cases} 1 & \text{if the } y_i \text{ belongs to group } g \\ 0 & Otherwise \end{cases}$

We consider that the dataset under observation is formed from superpositions of multiple Gaussian distributions.

A data point in the dataset can be defined as:

$y_i | Z_i \, ; \theta_{zi} \sim N(\mu_{zi}, \Sigma_{zi})$, where N represents a Gaussian distribution.

Now we will consider a particular case, i.e., the dataset under consideration is a univariate

$$\begin{aligned} f(y_i) &= \sum_{z_i \in Z} f(y_i | z_i) \, p(z_i) \\ &= \sum_{z_i \in Z} f(y_i | z_i) \, \pi_{z_i} \end{aligned} \tag{2.8}$$

$$f(y_i) = \sum_{g=1}^{G} \pi_g N\left(y_i\,;\mu_g\,,\sigma_g^2\right) \tag{2.9}$$

Equation 2.9 shows Gaussian Mixture model represented by linear superposition of individual Gaussian distributions. Naturally, the thought would be to estimate the parameters for the distribution.

Let $\hat{\pi_g}, \hat{\mu_g}, \hat{\sigma_g}$ represent the estimated parameters. To estimate the parameters, the Maximum Likelihood is computed.

Likelihood can be defined as the product of individual functions by independence. It can be written as follows:

$$\begin{aligned} L\left(\pi\,,\mu\,,\sigma^2\right) &= f\left(y_1\right)\cdot f\left(y_2\right)\cdot\cdots f\left(y_n\right) \\ &= \prod_{i=1}^{n} f\left(y_i\right) \\ &= \prod_{i=1}^{n}\left[\sum_{g=1}^{G}\pi_g \cdot N\left(y_i\,;\mu_g\,,\sigma_g^2\right)\right] \end{aligned} \tag{2.10}$$

Since it is Gaussian Distribution,

$$N\left(y_i;\mu_g,\sigma_g^2\right) = \frac{1}{\sqrt{2\pi\sigma_g^2}}\exp\left(\frac{-\left(y_i-\mu_g\right)^2}{2\sigma_g^2}\right) \tag{2.11}$$

Combining Equation 2.10 and Equation 2.11, we get:

$$L\left(\pi\,,\mu\,,\sigma^2\right) = \prod_{i=1}^{n}\left[\sum_{g=1}^{G}\pi_g\cdot\frac{1}{\sqrt{2\pi\sigma_g^2}}\exp\left(\frac{-\left(y_i-\mu_g\right)^2}{2\sigma_g^2}\right)\right] \tag{2.12}$$

To maximise the likelihood, we need to find the natural log and equate it to 0.

$$\log\left(L\left(\pi\,,\mu\,,\sigma^2\right)\right) = \sum_{i=1}^{n}\log\left[\sum_{g=1}^{G}\pi_g\cdot\frac{1}{\sqrt{2\pi\sigma_g^2}}\exp\left(\frac{-\left(y_i-\mu_g\right)^2}{2\sigma_g^2}\right)\right] \tag{2.13}$$

This Equation 2.13, we have seen when we had discussed the Mixture Models. However, this does not have a straightforward solution due to the presence of singularities in the distribution. To explain this consider a case when one of the points in the dataset is equal to the mean of the distribution. Substituting $y_i = \mu_g$, in Equation 2.11, we get the normal

distribution value as:

$$N\left(y_i\,;\mu_g\,,\sigma_g^2\right)=\frac{1}{\sigma_g\sqrt{2\pi}} \tag{2.14}$$

Now, if $\sigma_g \to 0$ in Equation 2.14, then Equation 2.13 would go to infinity. As a result, maximisation of log-likelihood for finding the model parameters cannot be a solution as there will always be the presence of such singularities in a mixture model. In this scenario, the approach that can be used is Expectation Maximisation (5). It provides a simple solution by initialising the parameters and then doing the expectation step, which finds the posterior probabilities. Later in the maximisation step, the parameters are again computed. These actions are repeated until the convergence of log-likelihood.

The motivation for EM Algorithm is - if we know the labels Z_i, then the problem is straightforward, and the following can be used to determine the parameters:
For a group g:

$$\hat{\pi_g}=\frac{(Number\ of\ Z_i\ in\ group\ g)}{n} \tag{2.15}$$

$$\hat{\mu_g}=\frac{\sum_{Z_i\in g} y_i}{(Number\ of\ Z_i\ in\ group\ g)} \tag{2.16}$$

$$\hat{\sigma_g^2}=\frac{\sum_{Z_i\in g}\left(y_i\ -\ \mu_g\right)^2}{(Number\ of\ Z_i\ in\ group\ g)} \tag{2.17}$$

If we assume that the Z_i are known, therefore the Completed likelihood can be written as:

$$f\left(y_i\,,Z_i\right)=N\left(y_i\,;\mu_{z_i}\,,\sigma {z_i}^2\right)\pi_{z_i} \tag{2.18}$$

$$\begin{aligned} L\left(y\,,Z\,;\theta\right) &= \Pi_{i=1}^{n}\Pi_{g=1}^{G}\left[\pi_g\,N\left(y_i\,;\mu_g\,,\sigma_g^2\right)\right]^{z_{ig}} \\ &= \Pi_{g=1}^{G}\Pi_{i=1}^{n}\pi_g^{z_{ig}}\,N\left(y_i\,;\mu_g\,,\sigma_g^2\right)^{z_{ig}} \\ &= \Pi_{g=1}^{G}\pi_g^{\Sigma_{i=1}^{n}z_{ig}}\Pi_{i=1}^{n}N\left(y_i\,;\mu_g\,,\sigma_g^2\right)^{z_{ig}} \end{aligned} \tag{2.19}$$

Log Likelihood,

$$l\left(y,Z;\theta\right)=log\left(L\left(y,Z;\theta\right)\right) \tag{2.20}$$

Where θ is the parameters of the distribution represented by (π,μ,σ^2).
Substituting 2.19 in 2.20, we get,

$$
\begin{aligned}
l(y, Z; \theta) &= \sum_{g=1}^{G} \left\{ \left(\sum_{i=1}^{n} Z_{ig} \right) \log(\pi_g) + \sum_{i=1}^{n} Z_{ig} \log\left[N\left(y_g, \mu_g, \sigma_g^2\right) \right] \right\} \\
&= \sum_{g=1}^{G} \sum_{i=1}^{n} Z_{ig} \log(\pi_g) + \sum_{g=1}^{G} \sum_{i=1}^{n} Z_{ig} \log\left[N\left(y_g, \mu_g, \sigma_g^2\right) \right] \\
&= \sum_{g=1}^{G} \sum_{i=1}^{n} Z_{ig} \log(\pi_g) + \sum_{g=1}^{G} \sum_{i=1}^{n} Z_{ig} \left[\frac{-1}{2} \log\left(\sigma_g^2\right) - \frac{-(y_i - \mu_g)^2}{2\sigma_g^2} - \frac{-1}{2} \log(2\pi) \right]
\end{aligned}
\tag{2.21}
$$

Optimising Equation 2.21 with μ_g , we find the derivative with respect to μ_g and equate it to 0

$$
\begin{aligned}
\frac{\partial l}{\partial \hat{\mu}_g} &= \sum_{i=1}^{n} Z_{ig} \frac{\partial}{\partial \hat{\mu}_g} \left[\frac{-(y_i - \hat{\mu}_g)^2}{2\hat{\sigma}_g^2} \right] \\
&= \sum_{i=1}^{n} \frac{Z_{ig}}{\sigma_g^2} (y_i - \hat{\mu}_g) \\
&= \frac{1}{\sigma_g^2} \left[\sum_{i=1}^{n} Z_{ig} y_i - \hat{\mu}_g \sum_{i=1}^{n} Z_{ig} \right]
\end{aligned}
\tag{2.22}
$$

Equating Equation 2.22 to 0, we get:

$$
\hat{\mu}_g = \frac{\sum_{i=1}^{n} z_{ig} y_i}{\sum_{i=1}^{n} z_{ig}} \tag{2.23}
$$

Similarly, optimising the Equation 2.21 with σ_g^2

$$
\begin{aligned}
\frac{\partial l}{\partial \hat{\sigma}_g^2} &= \sum_{i=1}^{n} z_{ig} \frac{\partial}{\partial \hat{\sigma}_g^2} \left[\frac{-1}{2} \log\left(\hat{\sigma}_g^2\right) - \frac{(y_i - \hat{\mu}_g)^2}{2\hat{\sigma}_g^2} \right] \\
&= \sum_{i=1}^{n} z_{ig} \left[-\frac{1}{2\hat{\sigma}_g^2} + \frac{(y_i - \hat{\mu}_g)^2}{2\hat{\sigma}_g^4} \right] \\
&= \frac{1}{\sigma_g^2} \left[-\frac{\sum_{i=1}^{n} z_{ig}}{2} + \frac{\sum_{i=1}^{n} z_{ig} (y_i - \hat{\mu}_g)^2}{2\hat{\sigma}_g^2} \right]
\end{aligned}
\tag{2.24}
$$

Equating Equation 2.24 to 0, we get:

$$
\hat{\sigma}_g^2 = \frac{\sum_{i=1}^{n} z_{ig} (y_i - \hat{\mu}_g)^2}{\sum_{i=1}^{n} z_{ig}} \tag{2.25}
$$

Similarly , by optimising the equation with π_g and equating the derivative to 0, we get,

$$\hat{\pi}_g = \frac{\sum_{i=1}^{n} z_{ig}}{n} \tag{2.26}$$

With these parameters in hand, we can:

1. Make an educated guess for the values of Z_{ig}, which can be represented as s_{ig}
2. Maximise the values for mean, variance and weights

We iterate through these two steps. In each iteration, it can be seen that the likelihood value increases and finally converges.
For this to work effectively, it depends on the educated guess we make for the Z_{ig}. Once we have the value of Z_{ig}, using the completed log-likelihood, we can compute the values of $\hat{\pi}_g$'s, $\hat{\sigma}_g^2$'s and $\hat{\mu}_g$'s.
It is also possible to initially guess the values of $\hat{\pi}_g$'s, $\hat{\sigma}_g^2$'s and $\hat{\mu}_g$'s and then find values of Z_{ig}'s. This is what is done in the EM Algorithm. The values of Z_{ig}'s are estimated by their posterior expectation. It can be represented as below:

$$S_{ig} = E_{z_{ig}|\pi_g,\mu_g,\sigma_g^2}\{Z_{ig}\} \tag{2.27}$$

Conditional Probability can be written as:

$$\begin{aligned} P\{Z_{ig} = 1|\pi_g, \mu_g, \sigma_g^2\} &= \frac{P\{Z_{ig} = 1|\pi_g, \mu_g, \sigma_g^2\}}{P\{Z_{ig} = 1\}} \\ &= \frac{\pi_g N\left(y_i; \mu_g, \sigma_g^2\right)}{\sum_{k=1}^{G} \pi_k N\left(y_i; \mu_k, \sigma_k^2\right)} \end{aligned} \tag{2.28}$$

$$s_{ig} = E_{z_{ig}|\pi_g,\mu_g,\sigma_g^2}\{Z_{ig}\} = \sum_{z_{ig}\in\{0,1\}} Z_{ig} P\left(Z_{ig}\right) \tag{2.29}$$

Only one term in the above expression would be there as others will be 0. Therefore:

$$s_{ig} = \frac{\pi_g N\left(y_i; \mu_g, \sigma_g^2\right)}{\sum_{k=1}^{G} \pi_k N\left(y_i; \mu_k, \sigma_k^2\right)} \tag{2.30}$$

Now we will see how the Expectation(E) and the Maximisation(M) step are worked out.

E step: The value is computed for s_{ig} over all the data points and groups, i.e., i varying from 1 to n (Datapoints) and g ranging from 1 to G(Groups)

M Step: The values of the $\hat{\pi}_g$, $\hat{\mu}_g$, and $\hat{\sigma}_g^2$ are computed for g varying from 1 to G(Groups) as below:

$$\hat{\pi}_g = \frac{\sum_{i=1}^{n} s_{ig}}{n}$$
$$\hat{\mu}_g = \frac{\sum_{i=1}^{n} s_{ig} y_i}{\sum_{i=1}^{n} s_{ig}}$$
$$\hat{\sigma}_g^2 = \frac{\sum_{i=1}^{n} s_{ig} \left(y_i - \hat{\mu}_g\right)^2}{\sum_{i=1}^{n} s_{ig}}$$

The log-likelihood is evaluated using the below equation:

$$\log\left(L\left(\pi, \mu, \sigma^2\right)\right) = \sum_{i=1}^{n} \log\left[\sum_{g=1}^{G} \pi_g \cdot N\left(y_i; \mu_g, \sigma_g^2\right)\right] \tag{2.31}$$

This value is checked for convergence, i.e., if it falls under a threshold value. If it does not, then the E and M step is again repeated.

Another view of the EM algorithm is when the dataset contains missing values. The parameters for the cluster having the missing data cannot be computed. What the EM algorithm does is it fills them with some values such as 0's in all the missing values. These values can be called Artificial values as they are filled by us to implement the algorithm. With these values, the parameters are calculated. This is the M step. The missing data is replaced with the conditional expectation values, which forms the E step. Now the E and M steps are repeated. In each iteration, the missing data is populated with new artificial data. The likelihood is guaranteed to increase in each iteration, and once a threshold is achieved for the difference between the populated missing values, the iteration is stopped. The maximum likelihood estimation is self-consistent (6), i.e., if we generate artificial data from the Maximum log-likelihood density function, it does not change the maximum likelihood estimation.

Another important consideration is the initialisation of parameters. Compared with KMeans, the EM algorithm is complex as it involves more computation, and the iterations required to reach convergence are also more. The most common method chosen for the initialisation of parameters is by running the KMeans algorithm and taking the means values from that. Covariance is initialised as the covariance obtained by the clusters, and weights can be obtained by the ratio of points assigned to each cluster. With these initialisations the EM algorithm is run.

It can be observed that KMeans is a particular case of EM. s_{ig} that we compute during the EM algorithm can also be called as responsibility. It shows the probability of a point falling into a particular cluster. In KMeans, as we know, it is a hard clustering

technique. A data point is allocated a cluster even if the position of the data point is roughly in between or close to the centre of two clusters. This means that the value of the responsibility will be either 0 or 1, whereas, in EM algorithm, the value will be between 0 and 1.

The above algorithms can be summarised as shown in Table 2.1:

Table 2.1: Clustering Algorithms

KMeans	**KMeans++**	**EM**
1. Initialise number of clusters and cluster centroids	1. An initial centre is randomly chosen c_1 uniformly from X	1. Initialise number of clusters and cluster centroids
2. E Step - data point is assigned to the closest centroids	2. The next centre is chosen $c_i = x' \in X$ having probability $\frac{D(x')}{\sum_{x \in X} D(x)^2}$	2. E Step – Find the responsibility that of each point belonging to a cluster
3. M Step - Find the new centroids of the clusters	3. Repeat the above 2 steps till all the centre are initialised	3. M Step – Find the new means, variance and weights with the responsibility
4. Repeat until it the centroids converge	4. Proceed with the KMeans algorithm after the initialisation step	4. Repeat until it the parameters or the log likelihood converges

Chapter 3

Methodology

This chapter discusses the approach chosen to study and build the algorithms from scratch. It details the steps involved in the algorithm and its implementation in Python. In addition, it discusses how the different algorithms were tested with the help of a synthetic dataset and later applied the algorithms to a real-world dataset.

3.1 Dataset

The most important part of the dissertation is to study the algorithm's behaviour by applying it to a real-world dataset. The algorithm needs to be first applied to a known dataset to understand the performance, and later it needs to be used on the unknown dataset so that we have a comparative study. For this purpose, there is a need to generate a synthetic dataset.

The synthetic dataset is one that is generated by the machine with user-defined parameters. Initially, while doing the dissertation, due to lack of knowledge, the approach chosen for generating the dataset was by selecting a random point and then generating more points around it falling within a fixed radius. The equation of the circle was used for generating the points.
Consider a circle with a radius r
Let (x_c, y_c) be the centres of the circle.
Generate a random angle represented by α defined as:

$\alpha = 2\pi *$ Random Number

We need to find a point (x, y) on the circle such that it is the defined angle from the centre having the radius defined. The points can be defined as:

$[x, y] = [r\cos(\alpha) + x_c\,, r\sin(\alpha) + y_c]$

This was one of the approaches in the beginning, failing to account for several details

about the dataset. One of the issues with this approach is that it did not account for any clear definition of the parameters such as mean, standard deviations or the distribution followed by the points. As a result, even if we implemented clustering, we will not have a known outcome to compare the results we got after clustering.

Since our study deals with the Mixture Models, it was more suitable to generate random points from a mixture distribution and later study how well the algorithm separates the individual distributions. The dissertation is trying to implement EM Algorithm to a univariate dataset as a result, we need to generate a synthetic dataset univariate by nature.

Initially, the clusters and the data points in them are defined. Once the clusters are defined, the mean and variances of each cluster need to be defined, after which we define the centroids for each cluster. The centroids are randomly picked, having a uniform distribution. Now weights are defined for each of the clusters. The weight is also known as mixing coefficients. For instance, if we randomly pick up a data point from the dataset, the probability that it will be from a particular cluster is defined by Mixing coefficients. An array of labels with the sample size is generated, having a probability distribution of mixing coefficients. Now the labels array is looped, and points are randomly picked from a normal distribution based on the cluster parameters.

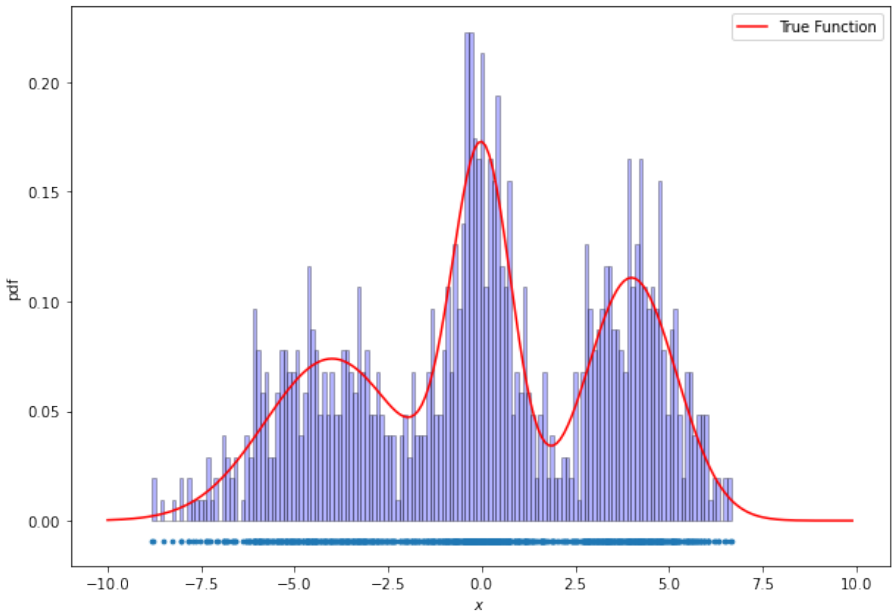

Figure 3.1: Histogram plot of the generated Synthetic Data. Red line represents the probability distribution function of the same.

Now on this dataset, the clustering techniques - KMeans, KMeans++ and EM algorithms are applied, and the results are studied. Generating points based on this approach

will help us define a dataset in more detail enabling us to compare the results once the clustering algorithm is implemented.

We aim to fit a model with a similar probability distribution function represented by the red line in Figure 3.1. This needs to be achieved by the EM Algorithm function developed by feeding in the data points as input.

An example of the creation of a synthetic dataset and the probability distribution of a randomly generated sample is shown in Figure 3.1. Table 3.1 shows the configuration used to generate the dataset.

Table 3.1: Configuration of synthetic dataset generated

Sample	**Cluster**	**Mean**			**Variance**			**Weights**		
1000	3	4	-4	0	1.2	1.8	0.8	0.33	0.33	0.33

3.2 KMeans

The aim is to create a KMeans function from scratch and run it on a synthetic dataset. The first step while doing KMeans is to define the clusters and then the initial centroids. There are numerous ways in which this could be done. One could be to check the minimum and maximum values present in the dataset and then generate the centroids within these limits for the clusters. Another way is to randomly pick points from the dataset as centroids. Once this initialisation is done, the next step is to iterate through the datapoints, compute the distance between each point and the centroids, and assign the points to a cluster whose distance is the shortest. After this, we can see that all the data points in the dataset will be assigned to one of the clusters defined. The cluster centroids are now calculated from the points assigned to a cluster. With these new centroids, the distance is again computed, and the points are assigned to clusters. This process is repeated until the centroid value converge or falls under a threshold.

The number of clusters is selected with the help of plotting the sum of squares of distances between the centroids and points vs K. It can be noted that there will be an elbow after which there is not much variation with the sum of the square of the distances with an increase in K. Running the algorithm is straightforward; however, the value of the chosen initial centroids plays a crucial role, as discussed in the previous chapter. The approaches of selecting random centroids depending on the data points scope and selecting random points within the cluster have been implemented in the function created.

Figure 3.2 shows the KMeans done on a synthetic 2-D dataset having 3 clusters. The dataset is generated with the *make_blob* function inside the Scikit Learn dataset library.

It will return the dataset as well as the labels. One thousand data points with 3 clusters are defined. The KMeans function is run on the dataset. It was run ten times to get a more accurate clustering. Figure 3.2 shows the output of the same runs on the synthetic dataset.
The functions in the implementation can be summarised as shown in Table 3.2 and the parameters are described in Table 3.3

Table 3.2: Functions in the developed KMeans Algorithm

Function Name	**Parameters**	**Purpose**	**Return**
get_centroids_from_points	no_centroids points features	Choose centroids from the dataset	Array of Centroids
get_random_centroids	no_centroids points limit features	Select the centroids randomly from the limits set	An array with the Centroids
cal_dist	p1 p2	Calculates the distance between 2 points	Distance
map_clusters	points centroids	Create an array having labels of the cluster having the shortest distance	A Mapped array of labels having the same dimension as the points
update_cluster_centroids	clusters points no_centroids centroids	Calculates the centroid values based on the cluster passed	An array with the Centroids
implement_kmeans	clusters points features	Implement the KMeans Algorithm	An array with the labelling is returned

Scikit Learn Library (4) has an inbuilt kmeans function. One of the important considerations of the dissertation is to draw a comparison between the inbuilt functions and the developed function. This will give us an idea of how the implementations differ and what can be improved. Table 3.4 shows the parameters passed as input to the KMeans inbuilt function. Figure 3.3 shows the output obtained by passing the same dataset to the inbuilt function.

Table 3.3: Parameters in developed KMeans Algorithm

Parameters	Description
no_centroids	Number of centroids
points	Data Points
features	Dimension
p1	Point 1
p2	Point 2
clusters	Cluster array
centroids	Centroid array
limit	Array with limit

Table 3.4: Parameters in kmeans function in Scikit Learn Library

Parameter	Value
n_clusters	3
init	random
n_init	10
max_iter	300
tol	1e-04
verbose	0
random_state	None
copy_x	None
algorithm	llyod

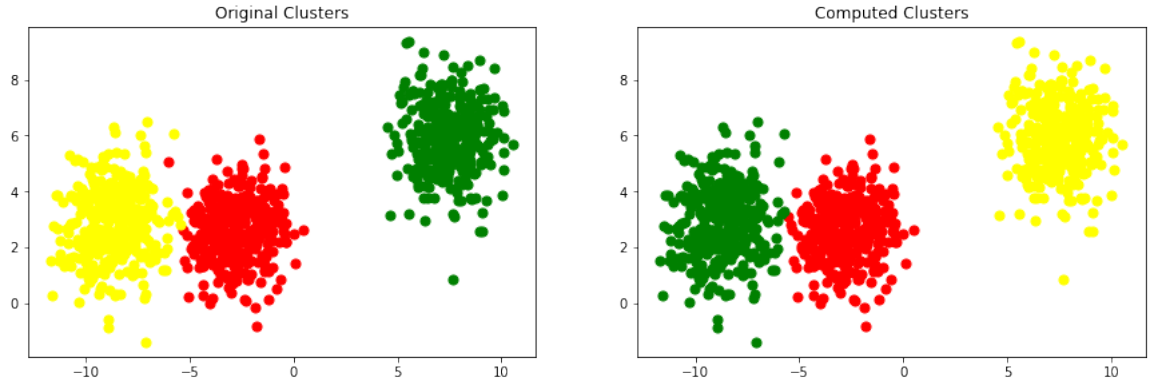

Figure 3.3: Comparison of the original and the output clusters obtained passing the same dataset used in Figure 3.2 to the inbuilt kmeans function.

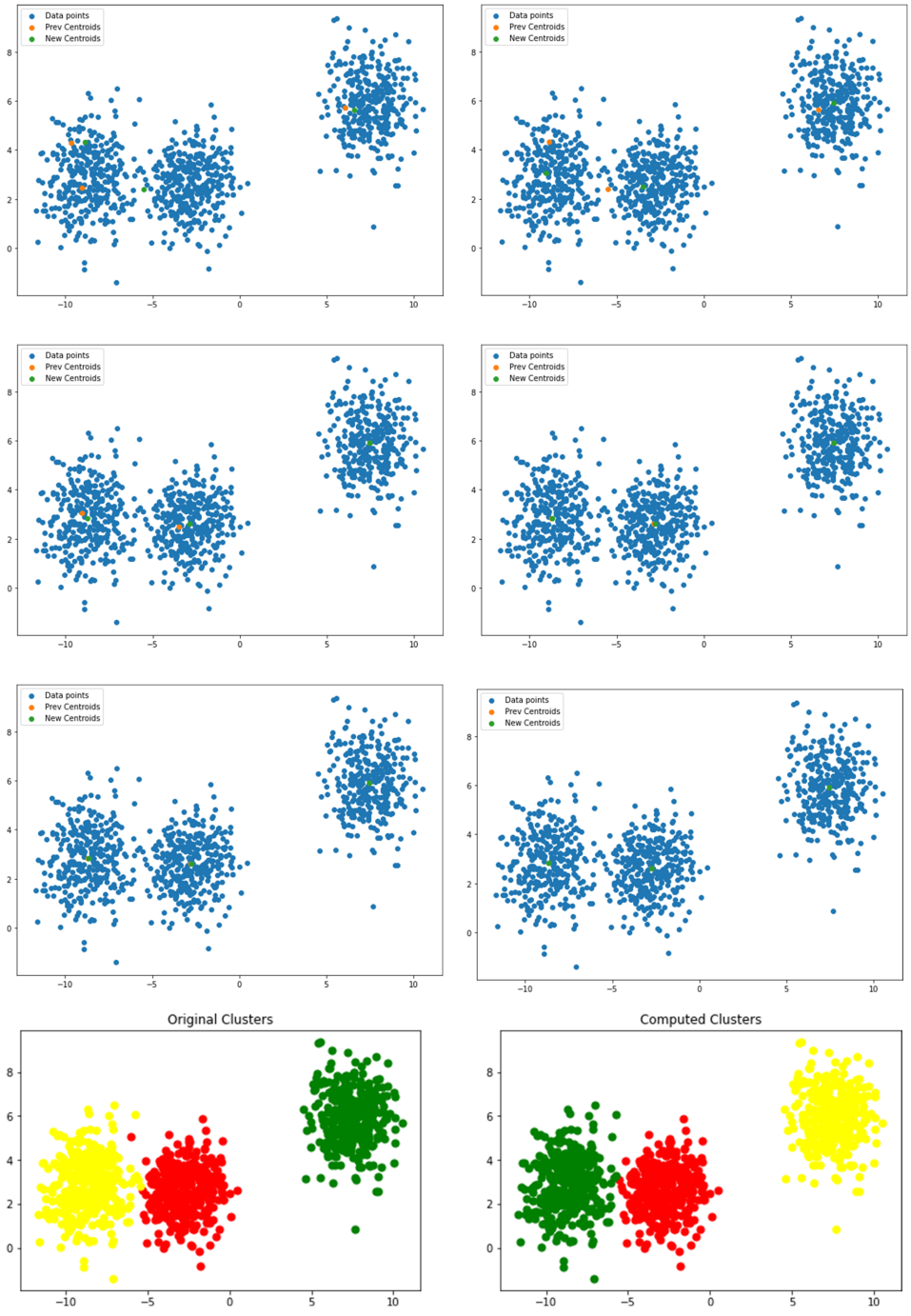

Figure 3.2: KMeans implemented to the synthetic dataset. The shift in the centroids during each iteration is shown in the figures. The final image shows the comparison of the original and the computed clusters obtained when the centroids converged.

3.3 KMeans++

As discussed in the previous chapter, initialisation plays an essential role in convergence and proper data clustering. KMeans++, as discussed before, is a smart initialisation technique. It selects the initial centroids from a probability distribution. This guarantees that the chosen centroids are not close to each other. Once the centroids are set, the process is the same as KMeans.

The aim is to create a KMeans++ algorithm from scratch to understand the concept involved. One of the points is randomly selected from the dataset. An array with the probability distribution is created. The probability distribution is defined as the shortest distance among the centroid chosen and the point divided by the sum of the distances of each point from the centroid. With this probability distribution, the next centroid is selected from the dataset. Once centroids for all the clusters are chosen, the same KMeans algorithm we defined earlier was run.

Figure 3.4 shows the KMeans++ algorithm implemented on a 2D dataset randomly generated in the same way we did for the KMeans case.

The functions in the implementation can be summarised as shown in Table 3.5 and the parameters are described in Table 3.6.

Table 3.5: Developed KMeans++ functions

Function Name	Parameters	Purpose	Return
get_centroids	no_centroids points features	Choose centroids from the dataset	Array of Centroids
calculate_prob_dist	centroid points idx	To calculate the probablity distribution of points for selection of the remaining centroids as per the Kmeans++ algorithm	An array of Probability distribution having the same size as the points
calculate_distance	p1 p2	Calculates the distance between 2 points	Distance
get_centroids_with_prob	clusters points features	Select the remaining centroids from a probablity distribution	An array with the Centroids
map_clusters	points centroids	Create an array having labels of the cluster having the shortest distance	A Mapped array of labels having the same dimension as the points
update_cluster_centroids	clusters points no_centroids centroids	Calculates the centroid values based on the cluster passed	An array with the Centroids
implement_kmeans_pp	clusters points features	Implement the KMeans++ Algorithm	An array with the labelling is returned

Scikit Learn library (4) has also an inbuilt kmeansplusplus function. It is generally used in machine learning for finding the centroids of the clusters. Table 3.7 shows the parameters passed as input to the inbuilt function.

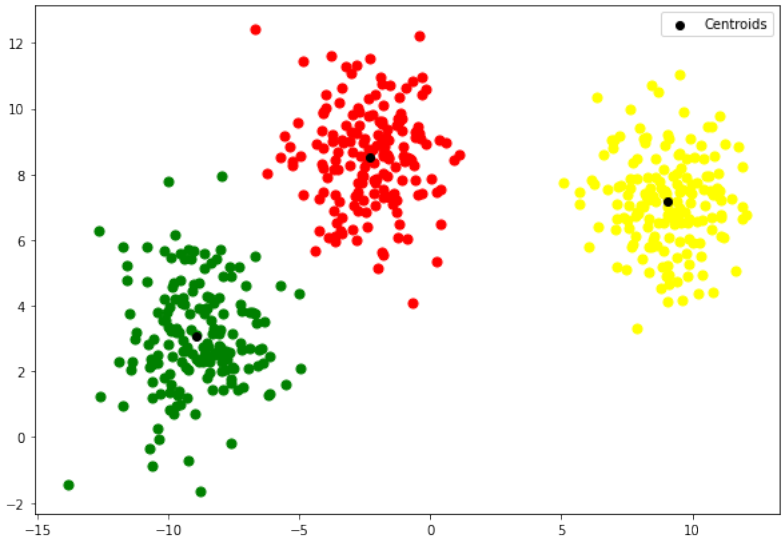

Figure 3.5: Centroids obtained from the inbuilt KMeans++

The plot (Figure 3.5) below shows the centroids of the clusters given by running the library function on the same dataset.No other details about the clustering are obtained other than the centroids and the index of the centroids from the dataset. The centroids are shown in black in Figure 3.5.

Table 3.6: Parameters in the the developed KMeans++ Algorithm

Parameters	**Description**
no_centroids	Number of centroids
points	Data Points
features	Dimension
p1	Point 1
p2	Point 2
clusters	Cluster array
centroids	Centroid array
limit	Array with limit

Table 3.7: Parameters in kmeansplusplus function in Scikit Learn Library

Parameter	**Value**
X	Data points
n_clusters	3
x_squared_norms	None
random_state	None
n_local_trials	None

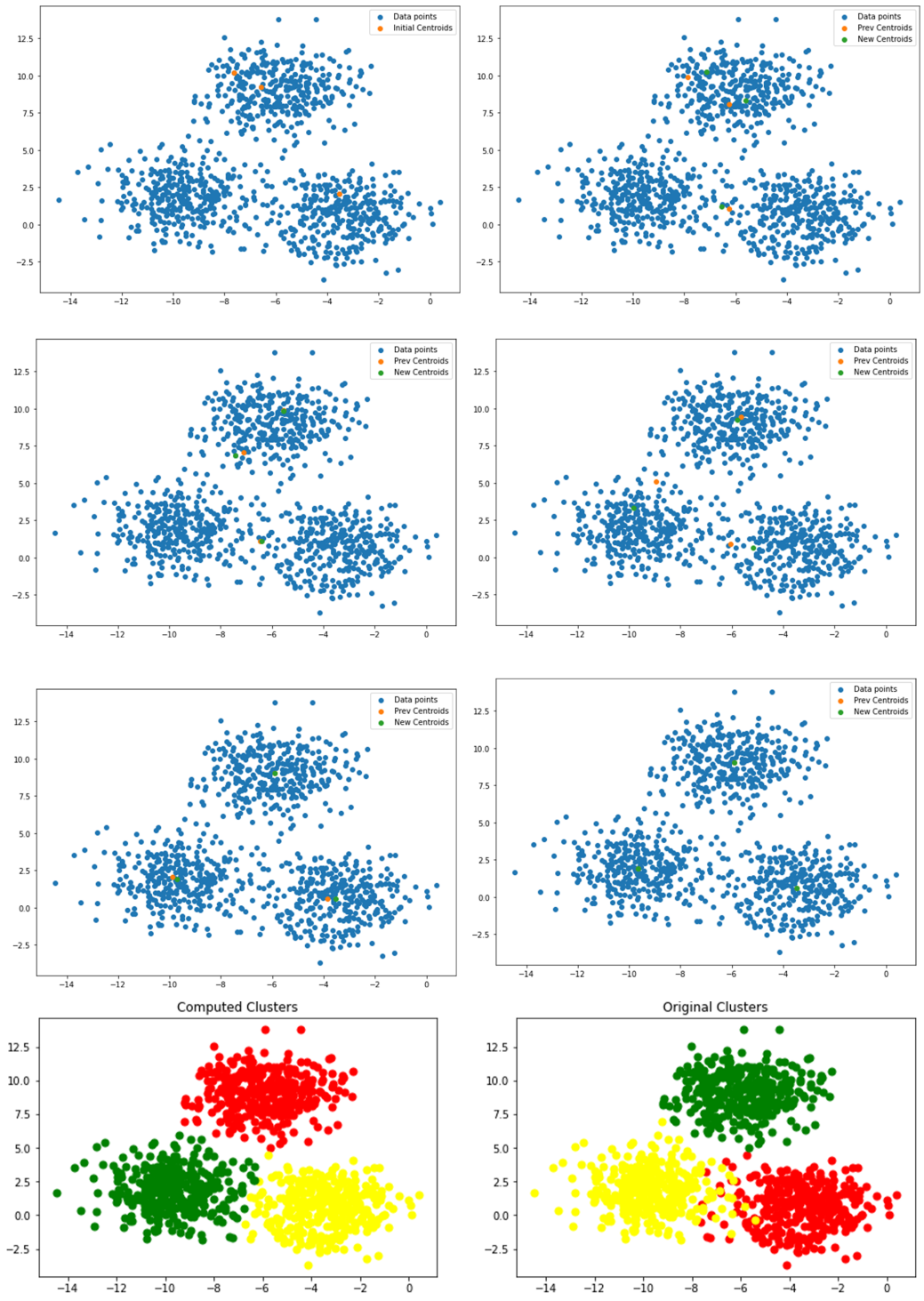

Figure 3.4: KMeans++ implemented to the synthetic dataset. The shift in the centroids during each iteration is shown in the figures. The final image shows the comparison of the original and the computed clusters obtained when the centroids converged.

3.4 EM Algorithm

The aim is to create this technique from the ground up by understanding the concept. The synthetic data generated in the first section is the input. We are trying to cluster the developed 1-D dataset and compute a function which would fit the data. The first step is to initialise the centroids, variance and weights.
The initial centroids can be determined using the following techniques:

- Randomly selecting points from the dataset
- Centroids were obtained by running the KMeans algorithm
- Centroids were obtained by running the KMeans++ algorithm

The variance is initialised as a minimal value, i.e., 10^{-4}. The weights are initialised as the probability of each point coming from a cluster, i.e., in our case, it is defined as having an equal probability of 1/3 for each cluster. Now with these values, the likelihood is calculated using the probability distribution function, which takes in the points, mean and variance. Now the E step is done where this likelihood value is used to calculate the responsibility, which denotes the probability of a point falling into a particular cluster. With this responsibility, the M step is done where the new mean, variance and weights are computed. The log-likelihood is calculated using these values, and the value is stored. The E and M steps are repeated until the relative tolerance of log-likelihood falls under a threshold of 10^{-5}.
The relative tolerance is calculated as follows:

$$\left|\frac{ll_{new} - ll_{old}}{ll_{old}}\right| = \left|\frac{ll_{new}}{ll_{old}} - 1\right|$$

Where ll represents the log-likelihood. It can also be said that at convergence, $\frac{ll_{new}}{ll_{old}}$ will tend to approach 1 making the relative tolerance tending to 0.

Once the threshold is met, the iterations, means, variances and responsibilities are returned. The agreeability between the known and computed labels is calculated using the *rand_score*, a function that comes under the metrics library in Scikit Learn. This will help us determine how well the model has predicted the labels. Once the parameters are known, we can plot the function using the normal distribution equation by defining a variable x varying from -10 to $+10$. The original function and the computed functions are plotted. The same function is run using different initialisation techniques.
The synthetic dataset was generated having the parameters shown in Table 3.8.

Figure 3.7 shows the distribution function's plot during each iteration, and the final function obtained.

Table 3.8: Configuration of synthetic dataset generated

Sample	Cluster	Mean			Variance			Weights		
1000	3	4	0	4	1.6	1.8	1.2	0.33	0.33	0.33

Table 3.9: Functions in the EM Algorithm developed

Function Name	Parameters	Purpose	Return
logsumexp	arr	To prevent underflow and overflow when exponential of large values come	Array
calculate_loglikelihood		Calculates the log-likelihood	Log-likelihood
implement_em	means variances weights	Calculates the responsibilities. With this responsibility calculates the mean, variance and weights. Log-likelihood is calculated. If it within a threshold from the previous log-likelihood the loop is exited	Iteration_number means variances weights log_likelihood responsibilities

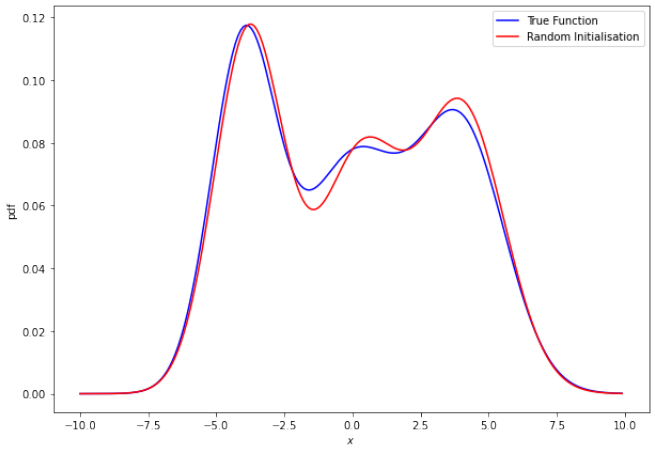

Figure 3.6: GuassianMixture function in the Scikit Learn Library is used to estimate the parameters. The distribution function with the actual and estimated parameters from the inbuilt function is plotted.

The function and the parameters used to implement the same are summarised in Table 3.9.

Scikit Learn Library (4) has an inbuilt function called GuassianMixture, which can estimate the parameters of a Gaussian Mixture distribution. If the same input is passed to the function, and the parameters can be determined, the function is plotted as above along with the true parameters. The same is shown by Figure 3.6.

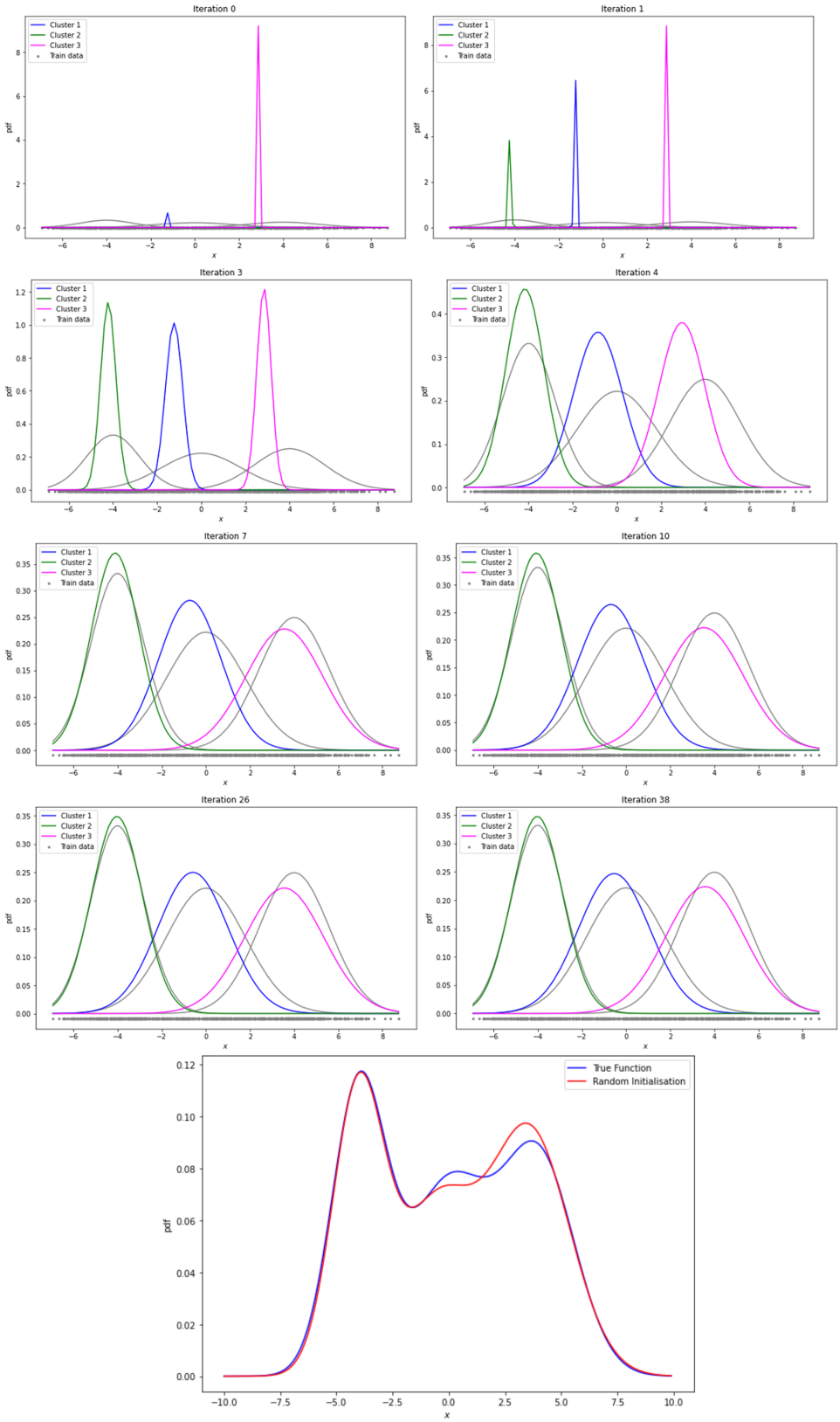

Figure 3.7: EM Algorithm applied to the Synthetic dataset. The distribution function during each iteration is shown in the figures. The final plot shows the true and computed distribution function obtained when the log-likelihood converged.

3.5 Evaluation Criteria

3.5.1 Rand Index

To understand how well the clustering was performed, we need to have certain criteria. Rand Index (16) is one that can be used to evaluate the algorithm's performance. This value gives the value of similarity between labels we have and the ones we computed from the algorithm.
It is calculated as:

$$R = \frac{a+b}{nC_2} \tag{3.1}$$

a - No of times a pair of elements belong to one cluster
b - No of times a pair of elements do not belong to one cluster
nC_2 - Unordered pairs in the unordered list

It can take a value between 0 and 1. When the value of R is 0, the labels under consideration do not agree on the clustering of any pair. On the other hand, when the value of R is 1, it means that the labels under consideration perfectly match.
Example to show how the Rand index is calculated:
Suppose we have 5 data points coming from 2 clusters – $\{a, b, c, d, e\}$
Known cluster labels - $\{0, 0, 0, 1, 1\}$
Now we implement one of the clustering techniques and ended up getting 3 clusters.
Computed cluster labels - $\{0, 0, 1, 1, 2\}$
The first step is to write all the possible unordered lists that can be formed using the data points. A total of 10 combinations of unordered lists can be formed, and they are as follows:
$\{d, e\}, \{c, e\}, \{c, d\}, \{b, e\}, \{b, d\}, \{b, c\}, \{a, e\}, \{a, d\}, \{a, c\}, \{a, b\}$
Next, we need to calculate the value for a and b,
a = 1 because it can be seen that the $\{a, b\}$ pair belongs to the same cluster in both the known and computed clusters.
b = 5 because it can be seen that the $\{a, d\}, \{a, e\}, \{b, d\}, \{b, e\}, \{c, e\}$ pairs belong to different clusters in both the known and computed clusters. Substituting the values in the equation of the Rand Index, we get the value of 0.6.

Scikit Learn Library (4) has a function *rand_score* which takes in the true and predicted labels as input parameters and gives a float value between 0 and 1 with both these values inclusive.

3.5.2 Bayesian Information Criteria

The models generated only try to fit the data. There are multiple methods by which a model can be created. It is necessary to determine which model gave the best performance. Using the available data, the aim is to determine the best model approximating or capable of generating the same data – or in other words, identify a model that reduces the information loss. The two most often used criteria for selecting the model is—the Bayesian Information Criterion (BIC) and the Akaike information criterion (AIC) are derived from the study of information theory as it is used to estimate the expected value of any information. (17). BIC varies from AIC as it gives more importance to the number of parameters.
The general equation for information criteria is $-2ll_{max} + Penalty$.
The Penalty value for AIC is $2p$ and for BIC is $p\log(n)$. Here p represents the free parameters in the mixture. Suppose we have a Gaussian Distribution having k groups, free parameters are weight, means and variance, then: $p = (k-1) + k + k = 3k - 1$

It can be seen that BIC is consistent because it gives more weightage to the number of free parameters when compared to AIC. Yang (18) tries to study the strengths of AIC and BIC criteria for multivariate regression analysis. As per (18), the BIC is considered to be consistent, and AIC accounts for the minimax-rate optimal aspect. These two cannot be combined, and one of them needs to be sacrificed. Since we are studying the performance of the model created with EM algorithm with different initialisation techniques, we choose BIC because it is more consistent.

3.6 Applying the EM to a Real-world dataset

The real-world dataset chosen is the univariate Galaxy dataset (11). It consists of 82 data points which show the velocities with which the other galaxies are shifting away from our galaxy. The number of galaxies in the universe could be found if we can determine the clusters present in the data. The Galaxy data set is frequently used in model-based clustering as a benchmark data set to compare the effectiveness of various modelling strategies. (19). The dataset has been debatable in the scientific community for quite a good number of years because the actual number of galaxies is unknown. In this dissertation, the K values are varied from 1 to 20. For each k, the algorithm is run 20 times. The BIC value is calculated during each run, and the run giving the minimum BIC value is considered the best run. Once this is done, the best BIC value plot for all the clusters (1 to 20) is plotted. From the plot, the K for the minimum BIC value is taken as the clusters. The dataset is plotted with the help of a histogram. The function is fitted

to the dataset with the value of K with a minimum BIC value. Figure 3.8 shows the function plot fitted with the parameters obtained by running the EM algorithm.

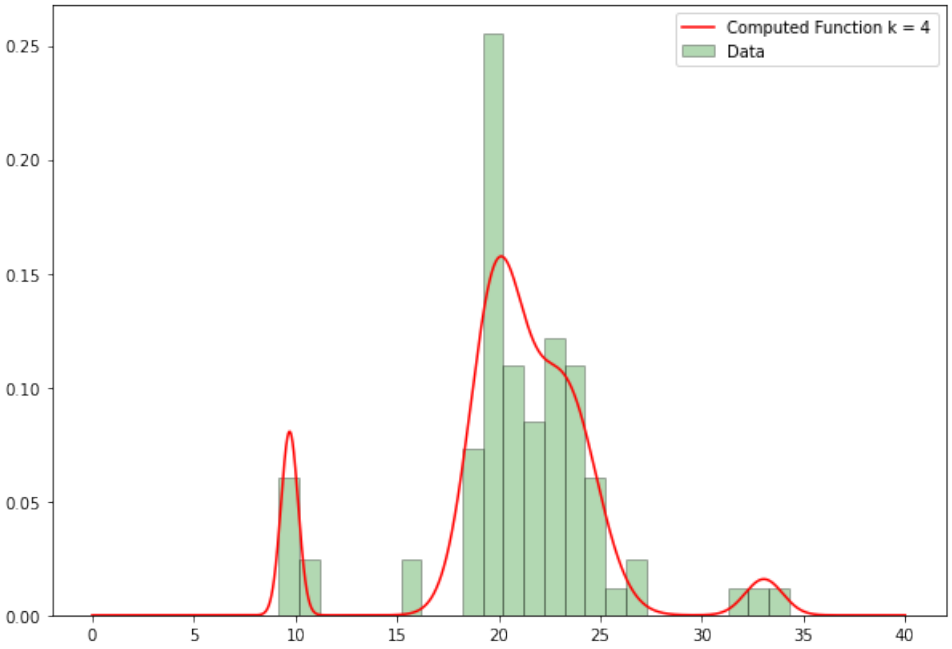

Figure 3.8: Histogram plot of Galaxy Dataset. The red line shows the probability distribution function with the parameters determined by running EM Algorithm.

Chapter 4

Results and Observations

The KMeans and KMeans++ algorithm was built from scratch, carefully applying the theory behind each step. A synthetic dataset was generated, and the same was passed as input to these algorithms.

For KMeans and KMeans++, a multivariate dataset was created using the *make_blobs* function to generate the dataset. Table 4.1 shows the configuration of the dataset generated. The clusters and the centroids of the dataset, were returned. Table 4.2 shows the centroids of the dataset. The scatter plot of the points is shown in the Figure 4.1.

Table 4.1: Dataset configuration for running KMeans and KMeans++

Sample	1000
Features	3
Cluster Std	1.2
Centroids number	3

Table 4.2: Centroids of the generated dataset with 3 clusters

Cluster 1		**Cluster 2**		**Cluster 3**	
1.9089	8.7722	8.0516	-3.2241	-0.4267	5.5773

Through visualisation, we can easily identify the clusters since it is coloured differently. Also, it can be seen that the clusters overlap. Our aim is to implement a clustering technique that would separate the data points and generate an output like in Figure 4.1. Additionally, we need to obtain the value of centroids such that the sum of the square of the distances is minimum. To achieve this, we use KMeans and KMeans++ in the first phase. Later we use the inbuilt functions from the Scikit Learn Library (4) and compare

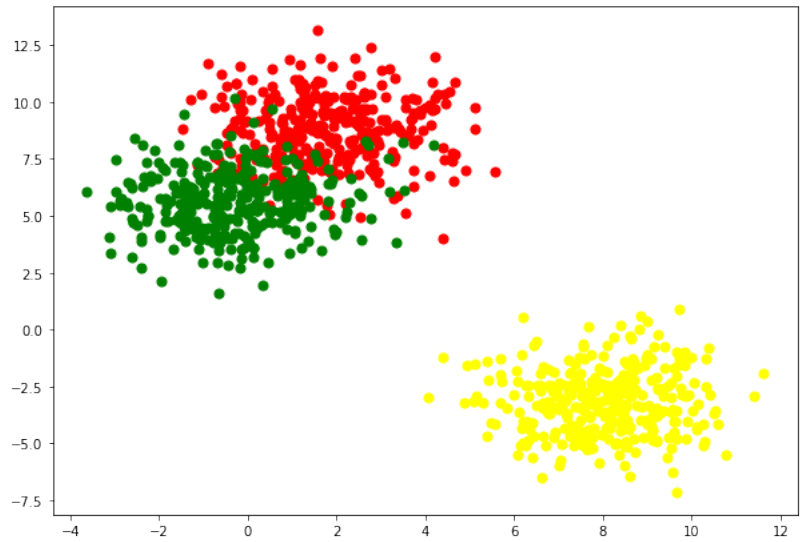

Figure 4.1: The plot of Synthetic Dataset for running KMeans and KMeans++. Red, Green and Yellow show the clusters present in the dataset.

the results. As experimented with and documented by Sieranoja, Sami in (20), repeating the algorithm 100 times reduced the number of enormous clusters by a small percentage. In our case, each algorithm is run ten times to reduce the error, and the results of all the observations are tabulated.

4.1 KMeans

From Table 4.3, it can be seen that the best agreeability observed was 0.92, and the corresponding centroids were: $[8.0221, -3.1298]$, $[1.9031, 8.9771]$, $[-0.3685, 5.6058]$. This was attained in 9 iterations. Figure 4.2 shows the plot of the actual centroids and the centroids we got from the best run of KMeans. It can be seen that it almost coincides, and the degree of difference is very small.

Once the algorithm runs and the centroids converge, the output image of clustering is as shown in Figure 4.3.

Table 4.3: Results of developed KMeans for 10 iterations

Runs	Centroids						Iterations	Agreeability
	Centroid 1		Centroid 2		Centroid 3			
1	8.0221	-3.1298	1.9136	8.9776	-0.3717	5.6151	7	0.9282
2	8.0221	-3.1298	1.9031	8.9771	-0.3685	5.6058	9	0.9293
3	1.9136	8.97768	-0.3717	5.6151	8.0221	-3.1298	7	0.9293
4	1.9031	8.9771	-0.3685	5.6058	8.0221	-3.1298	10	0.9282
5	0.7349	7.2435	7.9322	-4.2026	8.1194	-1.9698	8	0.7219
6	8.8743	-2.9382	6.818	-3.4006	0.7349	7.2435	11	0.9282
7	0.7349	7.2435	8.6411	-2.1832	7.4565	-3.9948	9	0.9282
8	7.9282	-2.1832	8.1262	-4.304	0.7349	7.2435	10	0.9282
9	8.0221	-3.1298	-0.3685	5.6058	1.9031	8.9771	9	0.9293
10	0.3717	5.6151	1.9136	8.9776	8.0221	-3.129	10	0.9282

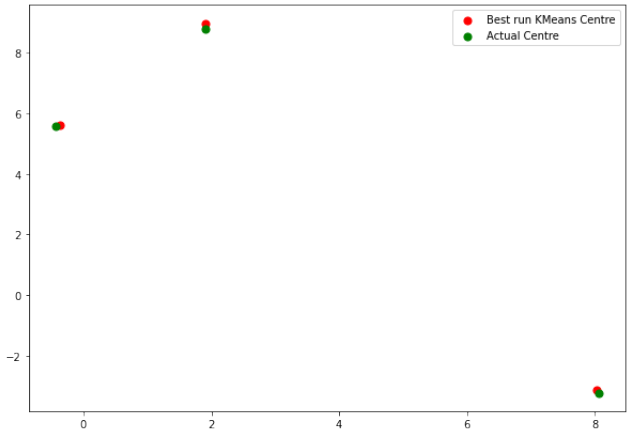

Figure 4.2: Actual Centroids and KMeans estimated Centroids.

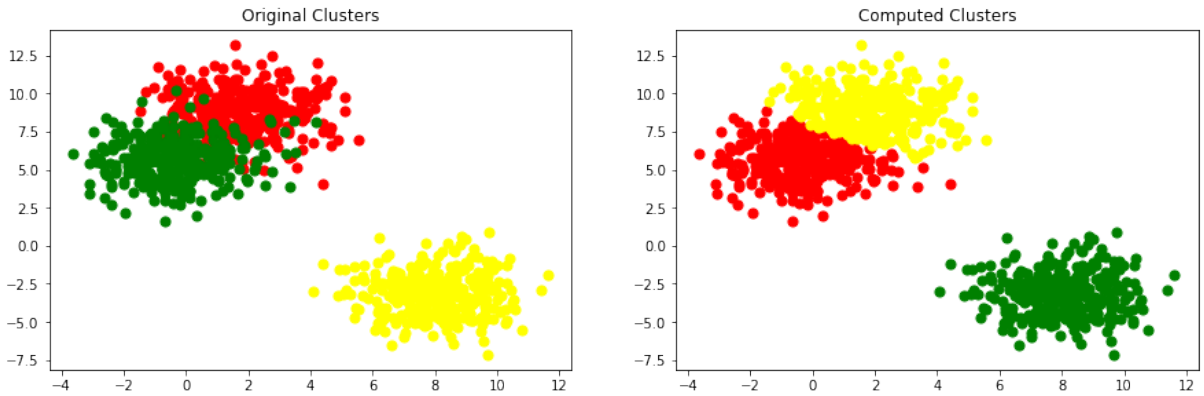

Figure 4.3: Comparison of clusters obtained after running KMeans with the original clusters

4.2 KMeans++

Table 4.4: Results of developed KMeans++ for 10 iterations

Runs	Centroids						Iterations	Agreeability
	Centroid 1		Centroid 2		Centroid 3			
1	-0.3685	5.6058	1.9031	8.9771	8.0221	-3.1298	8	0.9293
2	1.9136	8.9776	8.0221	-3.1298	-0.3717	5.6151	9	0.9282
3	1.9136	8.9776	8.0221	-3.1298	-0.3717	5.6151	7	0.9282
4	-0.3685	5.6058	8.0221	-3.1298	0.3717	5.6151	12	0.9293
5	1.9031	8.9771	8.0221	-3.1298	-0.3685	5.6058	9	0.9293
6	1.9136	8.9776	8.0221	-3.1298	-0.3717	5.6151	7	0.9282
7	-0.3717	5.6151	8.0221	-3.1298	1.9136	8.9776	8	0.9282
8	1.9031	8.9771	-0.3685	5.6058	8.0221	-3.1298	9	0.9293
9	7.9322	-4.2026	0.7349	7.2435	8.1194	-1.9698	8	0.7219
10	1.9031	8.9771	8.0221	-3.1298	-0.3685	5.6058	9	0.9293

From Table 4.4, it can be seen that the best agreeability observed was 0.92 and the corresponding centroids were: [1.9031, 8.9771], [8.0221, -3.1298], [-0.3685, 5.6058] This was attained in 8 iterations. Figure 4.4 shows the clustered output after running the KMeans++ algorithm.

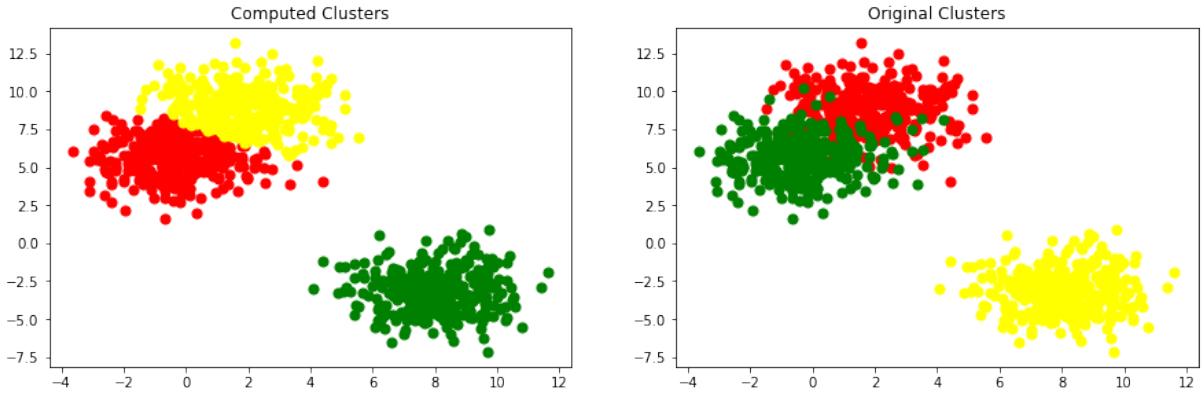

Figure 4.4: Comparison of clusters obtained after running KMeans++ with the original clusters

4.3 KMeans Inbuilt Function

Using the inbuilt KMeans function defined in the Scikit Learn Library (4). The function was executed 10 times and the results were noted. Table 4.5 shows the observations

obtained.

Table 4.5: Results of kmeans function in Scikit Learn Library for 10 iterations

Runs	**Centroids**						**Agreeability**
	Centroids 1		**Centroids 2**		**Centroids 3**		
1	1.9031	8.9771	8.0221	-3.1298	-0.3685	5.6058	0.9293
2	1.9136	8.9776	8.0221	-3.1298	-0.3717	5.6151	0.9282
3	1.9198	8.9809	8.0221	-3.1298	-0.3717	5.6151	0.9282
4	1.9198	8.9809	8.0221	-3.1298	-0.371	5.6218	0.9282
5	-0.3717	5.6151	8.0221	-3.1298	1.9136	8.9776	0.9282
6	8.0221	-3.1298	-0.3717	5.6151	1.9136	8.9776	0.9282
7	-0.3717	5.6125	8.0221	-3.1298	1.9136	8.9776	0.9282
8	1.9136	8.9776	8.0221	-3.1298	-0.3717	5.6151	0.9282
9	-0.3693	5.6279	8.0221	-3.1298	1.9252	8.9848	0.9282
10	-0.3652	5.6180	8.0221	-3.1298	1.9137	8.9851	0.9282

It can be seen by comparing Table 4.2 and Table 4.4 that the best agreeability observed is almost the same for the KMeans function we implemented from scratch and the built-in functions. It can also be noted that the centroids are also identical.

4.4 KMeans++ Inbuilt Function

The built-in KMeans++ function in Scikit Learn only returns the centroids and the indices of the centroids in the dataset. It was observed that one of the parameters in the function was the random state. When passed as None, it generates random centroids during each iteration while doing the KMeans++ algorithm. Figure 4.5 shows the plot of the centroids returned from the KMeans++ inbuilt function over ten iterations.

KMeans algorithm can be used for initial initializations of these centroids generated, leading to fewer iterations and more accurate results.

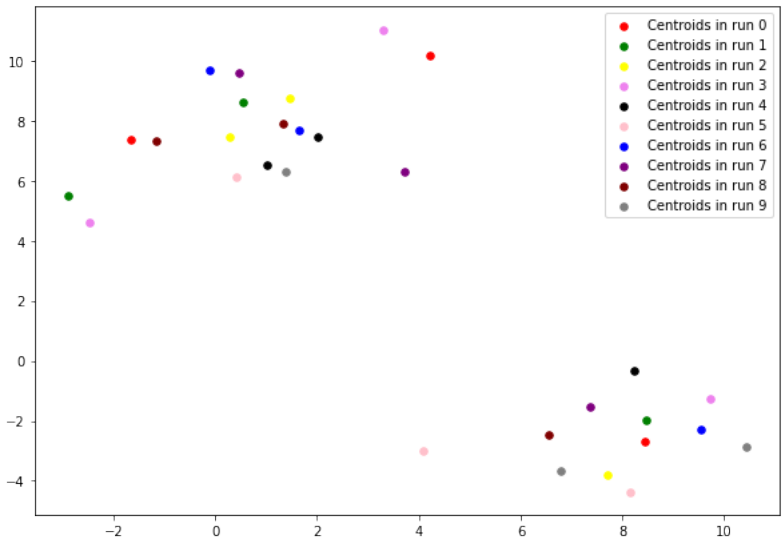

Figure 4.5: Centroids from each iteration generated while using kmeansplusplus function in the Scikit Learn Library

4.5 Comparing results of KMeans and KMeans++ functions developed

Visually observing Figure 4.3 and Figure 4.4, we can see that the input synthetic data we generated has overlapping clusters. As a result, the output will have less accuracy as KMeans and KMeans++ will do hard clustering and assign points to either of the clusters. Both KMeans and KMeans++ gave similar clustering, but it can be observed that the iteration was less for KMeans++ compared to KMeans. Comparing both the implementations, which were built from scratch, the plot of centroids with the best agreeability value is shown in Figure 4.6.

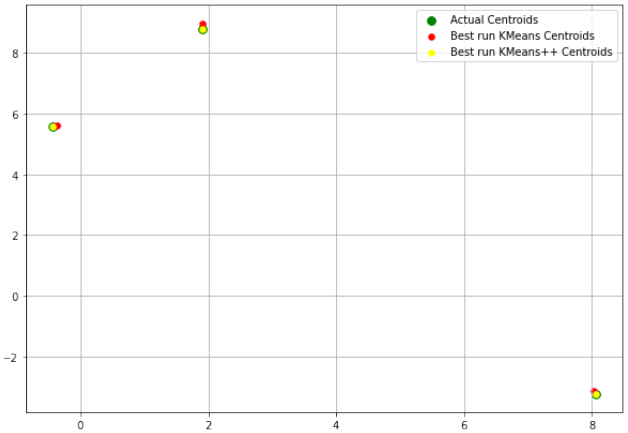

Figure 4.6: Plot of original centroids and the centroids obtained from developed KMeans and Kmeans++

From this Figure 4.6, it can be seen that KMeans++ gave better accuracy as the centroids coincide. From Table 4.3 and Table 4.4, we can see that the KMeans++ took 7 iterations and KMeans took 12 to give the same best agreeability value observed. In summary, we can say that number of iterations taken by KMeans++ to provide the best results is less than KMeans.

Considering all these observations, it can be said that the functions we built from scratch are at par with the inbuilt functions, with the advantage of knowing what happens during each step to give a clear understanding of the algorithm. It also provides the flexibility to change the working as per the requirement.

4.6 EM Algorithm

KMeans and KMeans++ were discussed because they can be used as a logical initialisation technique in EM Algorithm. The main area of focus for this dissertation is to study how the EM Algorithm is applied in Mixture models for clustering. The algorithm was implemented for a univariate distribution. The dataset was generated from a mixture model. Table 4.6 shows the configuration of the dataset generated.

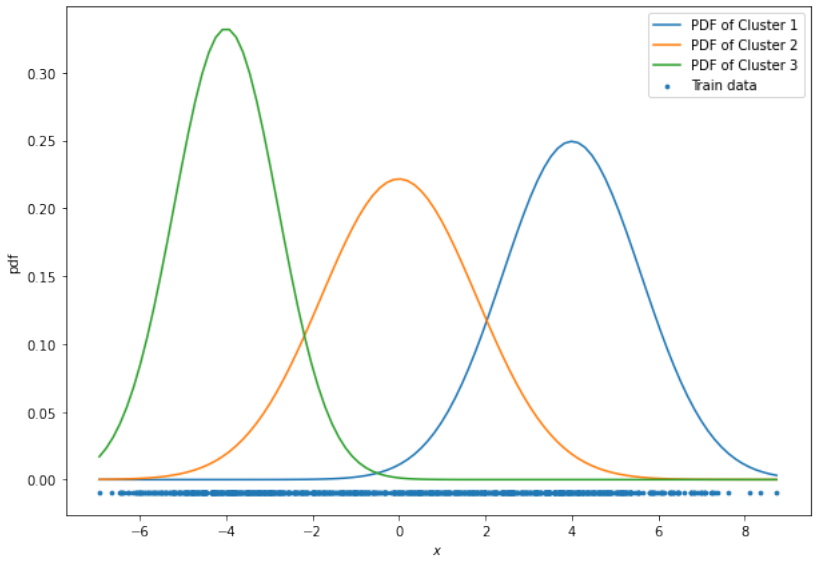

Figure 4.7: Plot of the dataset and the probability distribution function

Table 4.6: Dataset configuration for running EM Algorithm

Sample	Cluster	Mean			Variance			Weights		
1000	3	4	0	-4	1.6	1.8	1.2	0.33	0.33	0.33

An array with the labels and the dataset was generated using the synthetic generator function we had discussed before. This label array will be used to compare the agreeability once the clustering is done by the algorithm. The dataset was plotted. The Probability distribution was also plotted to understand more about the dataset. Figure 4.7 shows the plot of the dataset created.

The algorithm aims to fit a function similar to the probability distribution function shown in Figure 4.7. The EM algorithm is run with different initialisation techniques, and the results are tabulated in Table 4.7, Table 4.8 and Table 4.9.

4.6.1 EM Algorithm with Random Initialization

Table 4.7: Results of developed EM Algorithm run for 10 times(Random Initialisation)

Run	**Parameter**									**Log Likelihood**	**Iteration**	**Agreeability**
	Mean			**Standard Deviation**			**Weights**					
0	-5.9740	-0.2025	6.4267	0.0001	0.0001	0.0001	0.3333	0.3333	0.3333			
1	**-4.2353**	**0.6246**	**4.3705**	**1.0252**	**2.8280**	**1.0078**	**0.2761**	**0.5767**	**0.1472**	**-2044.3782**	**40**	**0.7417**
2	-4.2329	0.6374	4.3725	1.0274	2.8259	1.0050	0.2771	0.5766	0.1463	-2045.9363	6	0.7410
3	-4.2313	0.6472	4.3742	1.0289	2.8247	1.0023	0.2778	0.5767	0.1455	-2047.2926	11	0.7397
4	-4.2311	0.6480	4.3744	1.0290	2.8246	1.0020	0.2779	0.5767	0.1454	-2047.7507	1	0.7397
5	-4.2310	0.6491	4.3745	1.0291	2.8245	1.0017	0.2779	0.5767	0.1454	-2046.7507	2	0.7397
6	-4.2309	0.6496	4.3746	1.0292	2.8245	1.0015	0.2780	0.5767	0.1453	-2046.9371	1	0.7397
7	-4.2308	0.6501	4.3747	1.0293	2.8245	1.0014	0.2780	0.5767	0.1453	-2046.9485	1	0.7397
8	-4.2307	0.6505	4.3748	1.0293	2.8244	1.0013	0.2780	0.5767	0.1452	-2046.9485	1	0.7397
9	-4.2307	0.6509	4.3749	1.0294	2.8244	1.0012	0.2781	0.5767	0.1452	-2046.9485	1	0.7397
10	-4.2306	0.6511	4.3749	1.0294	2.8244	1.0011	0.2781	0.5768	0.1452	-2046.9485	1	0.7397

4.6.2 EM Algorithm with KMeans Initialization

Table 4.8: Results of developed EM Algorithm run for 10 times(KMeans Initialisation)

Run	Parameter									Log Likelihood	Iteration	Agreeability
	Mean			Standard Deviation			Weights					
0	4.1521	-0.0163	-4.0809	0.0001	0.0001	0.0001	0.3333	0.3333	0.3333			
1	3.7387	-0.3544	-4.0986	1.6465	1.2911	1.1115 -	0.3696	0.2768	0.3536	-2119.6463	44	0.8489
2	3.7347	-0.3570	-4.0981	1.6487	1.2881	1.1117	0.3702	0.2761	0.3538	-2118.9985	2	0.8480
3	3.7271	-0.3611	-4.0967	1.6528	1.2817	1.1123	0.3713	0.2745	0.3541	-2121.0121	6	0.8473
4	3.7222	-0.3633	-4.0954	1.6555	1.2770	1.1130	0.3721	0.2734	0.3544	-2120.7214	5	0.8482
5	**3.7170**	**-0.3651**	**-4.0938**	**1.6582**	**1.2717**	**1.1139**	**0.3729**	**0.2722**	**0.3548**	**-2120.8040**	**7**	**0.8493**
6	3.7154	-0.3655	-4.0932	1.6590	1.2700	1.1142	0.3732	0.2719	0.3550	-2120.1452	2	0.8484
7	3.7127	-0.3662	-4.0921	1.6604	1.2669	1.1147	0.3736	0.2712	0.3552	-2120.3806	5	0.8484
8	3.7119	-0.3664	-4.0918	1.6609	1.2660	1.1149	0.3737	0.2710	0.3553	-2120.8435	1	0.8475
9	3.7105	-0.3667	-4.0912	1.6616	1.2643	1.1153	0.3740	0.2706	0.3554	-2119.9777	3	0.8475
10	3.7098	-0.3668	-4.0909	1.6620	1.2635	1.1154	0.3741	0.2704	0.3555	-2119.9681	1	0.8475

4.6.3 EM Algorithm with KMeans++ Initialization

Table 4.9: Results of developed EM Algorithm run for 10 times(KMeans++ Initialisation)

Run	Parameter									Log Likelihood	Iteration	Agreeability
	Mean			Standard Deviation			Weights					
0	0.4448	-4.6772	5.8569	0.0001	0.0001	0.0001	0.3333	0.3333	0.3333			
1	-0.3597	-4.1429	3.8073	1.4043	1.0883	1.6146	0.2992	0.3429	0.3579	-2106.4551	75	0.8566
2	**-0.3620**	**-4.1314**	**3.7850**	**1.3702**	**1.0940**	**1.6251**	**0.2925**	**0.3459**	**0.3617**	**-2110.9189**	**9**	**0.8580**
3	-0.3646	-4.1106	3.7465	1.3137	1.1047	1.6436	0.2810	0.3509	0.3681	-2119.8094	23	0.8509
4	-0.3660	-4.0994	3.7255	1.2847	1.1108	1.6540	0.2749	0.3536	0.3715	-2120.8462	21	0.8482
5	-0.3663	-4.0973	3.7215	1.2793	1.1119	1.6560	0.2738	0.3540	0.3722	-2121.1300	5	0.8482
6	-0.3664	-4.0963	3.7197	1.2769	1.1124	1.6569	0.2733	0.3543	0.3725	-2121.8607	2	0.8482
7	-0.3666	-4.0952	3.7176	1.2741	1.1130	1.6580	0.2727	0.3545	0.3728	-2121.2983	3	0.8482
8	-0.3668	-4.0939	3.7152	1.2709	1.1137	1.6593	0.2720	0.3548	0.3732	-2120.7549	4	0.8484
9	-0.3668	-4.0935	3.7143	1.2697	1.1140	1.6597	0.2717	0.3549	0.3733	-2120.7530	1	0.8484
10	-0.3669	-4.0930	3.7134	1.2686	1.1142	1.6601	0.2715	0.3550	0.3735	-2120.1715	1	0.8484

From Table 4.7, Table 4.8 and Table 4.9 it can be seen that the KMeans and KMeans++ gave faster clustering and more agreeability compared to the random initialisation. It can also be noted that KMeans and KMeans++ gave similar results. KMeans++ gave more agreeability because the initialisations consider the probability distribution of the weighted distances for the initialisation of centroids. It increases the chances of these initial means chosen lying far from each other and closer to the centroids. The parameters that gave the best agreeability are highlighted in bold in Table 4.7, Table 4.8 and Table 4.9.

GaussianMixture is an inbuilt function in the Scikit Learn Library (4) for implementing EM Algorithm. The same dataset is fed into the function, and the parameters are

determined. It is run ten times, and the parameters that gave the best BIC values were taken.

Figure 4.8 displays the fitting of a model generated with the EM algorithm with different initialisation techniques along with the inbuilt GaussianMixture built-in function.

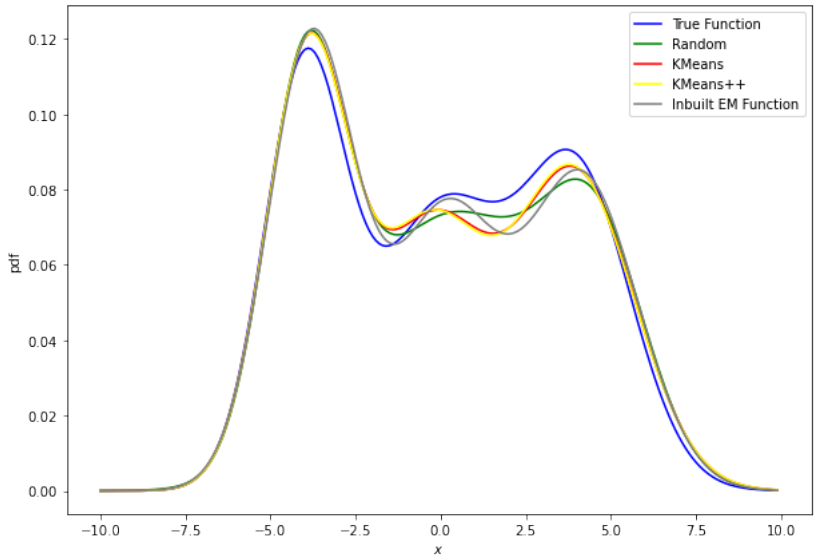

Figure 4.8: Plots of Distribution function using different initialisation techniques in EM

It can be seen that the algorithm built from scratch can fit a model similar to the original function and give a good agreeability value. One of the drawbacks of the built-in function is that the value of the responsibilities of the data points cannot be retrieved. This implementation gives us more clarity of working like the responsibility values, iterations taken to converge and the log-likelihood values. These parameters are critical when we study the algorithm. This provides a more flexible approach to clustering.

4.7 Defining the Degree of separations between clusters

In general, the difficulty in separating the clusters lies in the degree of separation they have in between. The EM algorithm is guaranteed to work with clusters that are well separated. KMeans and KMeans++ will also work well with well-separated clusters, but they will do a hard clustering of data points.

Consider a case when there are 2 clusters present in a univariate distribution having means as μ_1 and μ_2 and variance as σ_1 and σ_2. Let the distance between the means be

$\triangle$. Entropy can be defined as the measure of disorder present in the dataset. It can also be said that when the value of entropy is minimum, the clusters can be easily separated. Mathematically it can be written as:

$$E = -\sum_{k=1}^{K} P_j \log(P_j)$$

Where P_j represents the responsibility value which gives us an idea of the probability of a datapoint belonging to a particular cluster.
Suppose we have 2 clusters, then the value of entropy is maximised when the value P_1 and P_2 are equal. We also know that $P_1 + P_2 = 1$.
So therefore $P_1 = P_2 = 0.5$. This means there is an equal chance for the point to be in both clusters which makes it difficult to cluster.
Consider a case when the mean is defined as:

$\mu_1 = 0$ and $\mu_2 = \mu_1 + \triangle$

Similarly, consider variance is defined as:

$\sigma_1^2 = 1$ and $\sigma_2^2 = \tau^2\sigma_1^2 = \sigma_1^2$

The relationship between the $\mu_1, \mu_2, \triangle, P_j$ and E can be found by simulating Equation 4.1.

$$P_j = \frac{\pi_g \phi\left(x; \mu_j, \sigma_j^2\right)}{\sum_{k=1}^{K} \pi_k \phi\left(x; \mu_k, \sigma_k^2\right)} \tag{4.1}$$

If the value of $\triangle = 0$, it means that the 2 clusters are overlapping. This would be difficult for our algorithm to cluster. We need to use clustering techniques that would consider the entropy factor and a method which would require a subspace sampling like the Gustafson-Kessel Subspace Clustering (21).

It was observed that when the mean of the two clusters was identical, the clustering gave poor results, i.e., agreeability was less than 0.5, which means that it failed to separate the clusters. When the value of $\triangle = \tau^2$, it was observed that the agreeability improved better than the clusters having the same mean. However, the value was between 0.5 and 0.7, which is still very low for a good clustering. Further on increasing this value, it was seen that the agreeability increased.

4.8 Galaxy Dataset

The dataset is converted to a Data frame using the Pandas (22) library in Python. The aim of the experiment is to use the clustering algorithm to find the clusters present in the dataset. The procedure followed is that the value of the cluster is varied from 1 to

20, and in each K value, the EM algorithm is run ten times, and the model that gave the minimum BIC value was taken. Now BIC value vs K is plotted, and we get a plot as shown in Figure 4.9.

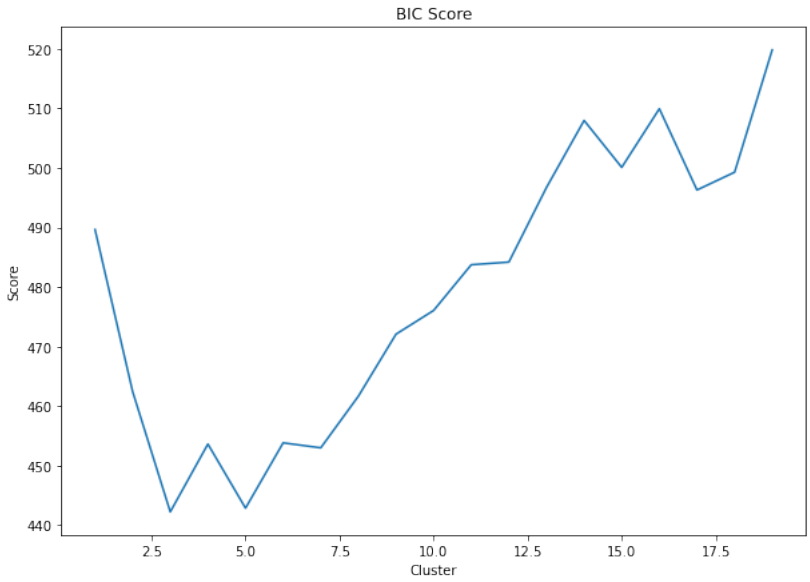

Figure 4.9: BIC Score for different K

From the plot (Figure 4.9), the value for K for which the model gave the lowest BIC value is chosen as the clusters. It can be seen that the lowest BIC value is observed at K=4, and corresponding parameters are noted. With these parameters, the probability distribution function is plotted, as shown in Figure 3.8.

This dataset is controversial, and different models can be fitted based on the cluster number. Here we have chosen BIC as a criterion for the selection of the clusters. Clustering this data, we would get the number of galaxies in the universe. It isn't easy to have a clear answer to this question. It varies with the criteria chosen for selection and the method used for clustering.

Chapter 5

Conclusion and Future Works

Programming in Python has evolved drastically due to its ease of use and availability of direct functions for most tasks. Scikit Learn Library (4) is the most famous one for executing machine learning functions in Python. Most algorithms are readily available in the Scikit Learn library to do majority of the machine learning tasks such as different types of Regression, Clustering, and Preprocessing. Clustering is an essential technique under Unsupervised Machine Learning. The dissertation aimed to implement the EM Algorithm from the root up and apply it to a real-world dataset. This would give us a clear understanding of how the algorithm works and provides flexibility to the whole process, which the inbuilt function fails to do. In implementing the EM Algorithm, initialising the parameters plays a critical role. Different techniques are used, out of which KMeans and KMeans++ are very important. KMeans and KMeans++ algorithms are also implemented from scratch. The similarities and differences between them were studied and applied to different synthetic data, and the results were observed.

An improvement in the accuracy and execution time was observed when the initialisation methods changed. It showed faster convergence and more accurate clustering when the initialisation was good. Using the inbuilt functions for the same purpose directly is like giving input to a black box without understanding how the algorithm works. If the performance of the function is not accurate, there is no flexibility to understand how the function is defined and make necessary corrections. However, if we define a function, we will have the confidence to explain with a clear understanding of the functioning of the inner statements. It also allows the flexibility to change the output and input in favour of the requirement.

EM Algorithm finds its main application when the data under consideration has missing values. It also has a good application in Pattern Recognition, Image segmentation, and Data mining problems. It can also be seen that KMeans and EM algorithms are

closely related. It can be summarized that the EM is a two-step process to find the maximum log-likelihood solution. It is an iterative process of 2 steps, the E step in which the responsibility of a datapoint belonging to a cluster is calculated using the likelihood and then the M step, with the parameters of the distribution like the mean, variance and weights are calculated with this responsibility. After the M step, the log-likelihood is calculated and checked for convergence. The process is repeated if it is not converged or falls under a threshold set.

Currently, the limitation of the EM Algorithm function implemented is that it only supports the univariate distribution. Most real-world datasets are multivariate and are not supported by the implementation. It can be extended to multivariate by making changes to the equations. Table 5.1 shows the changes to be done to implement it for Multivariate Distributions.

Table 5.1: EM in Univariate and Multivariate cases

Univariate Distribution	Multivariate Distribution
Let $x_i \in$ Univariate Distribution	Let $x_i = x_{i1}, x_{i2}, \cdots x_{id} \in \mathrm{R}^d$ (Belongs to a Multivariate Distribution in d dimensions)
E step	E step
$S_{ik} = \frac{\pi_k \phi\left(x_i;\mu_k,\sigma_k^2\right)}{\sum_{g=1}^{G} \pi_g \phi\left(x_i;\mu_g,\sigma_g^2\right)}$	$S_{ik} = \frac{\pi_k \phi_d\left(\underline{X_i};\underline{\mu_k},\sum_k\right)}{\sum_{g=1}^{G} \pi_g \phi_d\left(\underline{X_i};\underline{\mu_g},\sum_g\right)}$
M step	
$\pi_k = \frac{\sum_{i=1}^{n} S_{ik}}{n}$	$\pi_k = \frac{\sum_{i=1}^{n} S_{ik}}{n}$
$\mu_k = \frac{\sum_{i=1}^{n} S_{ik} X_i}{\sum_{i=1}^{n} S_{ik}}$	$\underline{\mu_k} = \frac{\sum_{i=1}^{n} S_{ik} \underline{x_i}}{\sum_{i=1}^{n} S_{ik}} \in \mathrm{R}^d$
$\sigma_k^2 = \frac{\sum_{i=1}^{n} S_{ik} (X_i - \mu_k)^2}{n}$	$\sum_k = \frac{\sum_{i=1}^{n} S_{ik} \left(\underline{x_i} - \underline{\mu_k}\right)\left(\underline{x_i} - \underline{\mu_k}\right)^{\tau}}{n}$

If the current implementation is changed a little, it can accommodate the Multivariate case, giving the function implemented an increased application in real-world data. Additionally, the mixture distributions are currently coming from the superposition of multiple Gaussian Distributions. This could also be expanded into various distributions like Mixtures of Bernoulli distributions. This could also lead to increased applications in the present world datasets.

We have discussed and implemented mainly three clustering techniques. It can be seen that different clustering techniques are suitable for different datasets, and there is no one clustering technique that fits all datasets. This can only be found out by studying the dataset, statistically implementing various procedures to understand them, and later applying a suitable clustering algorithm and then studying the results. The current implementation can be extended and published as a library having different clustering algorithms like BIRCH (23), DBSCAN (24), Affinity Propagation Clustering Algorithm (25), and OPTICS algorithm (26), which can be configured based on the dataset input.

Bibliography

[1] J. Peng, E. C. Jury, P. Dönnes, and C. Ciurtin, "Machine learning techniques for personalised medicine approaches in immune-mediated chronic inflammatory diseases: Applications and challenges," *Frontiers in Pharmacology*, vol. 12, 2021.

[2] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022.

[3] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[6] B. Efron and T. Hastie, *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Institute of Mathematical Statistics Monographs, Cambridge University Press, 2016.

[7] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[8] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, 1967.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[10] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *K-Means++: The Advantages of Careful Seeding*, vol. 8, pp. 1027–1035, 01 2007.

[11] K. Roeder, "Density estimation with confidence sets exemplified by superclusters and voids in the galaxies," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 617–624, 1990.

[12] A. Azzalini and A. W. Bowman, "A look at some data on the old faithful geyser," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 39, no. 3, pp. 357–365, 1990.

[13] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," *Machine Learning, p*, 01 2002.

[14] V. Ramasubramanian and K. Paliwal, "Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding," *IEEE Transactions on Signal Processing*, vol. 40, no. 3, pp. 518–531, 1992.

[15] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[16] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[17] F. J. Fabozzi, S. M. Focardi, S. T. Rachev, and B. G. Arshanapalli, *The basics of financial econometrics: Tools, concepts, and asset management applications*. John Wiley & Sons, 2014.

[18] Y. Yang, "Can the strengths of aic and bic be shared? a conflict between model indentification and regression estimation," *Biometrika*, vol. 92, no. 4, pp. 937–950, 2005.

[19] B. Grün, G. Malsiner-Walli, and S. Frühwirth-Schnatter, "How many data clusters are in the galaxy data set?," *Advances in Data Analysis and Classification*, vol. 16, no. 2, pp. 325–349, 2022.

[20] P. Fränti and S. Sieranoja, "How much can k-means be improved by using better initialization and repeats?," *Pattern Recognition*, vol. 93, pp. 95–112, 2019.

[21] B. Wiswedel and M. R. Berthold, "Fuzzy clustering in parallel universes," *International Journal of Approximate Reasoning*, vol. 45, no. 3, pp. 439–454, 2007.

[22] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[23] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[25] L. Sun, R. Liu, J. Xu, S. Zhang, and Y. Tian, "An affinity propagation clustering method using hybrid kernel function with lle," *IEEE Access*, vol. 6, pp. 68892–68909, 2018.

[26] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "Acm sigmod record," *New York: ACM*, pp. 49–60, 1999.

Appendix

Dataset Generated for testing EM Algorithm

The function returns the array of datapoints and labels.

Sample size - 100

Cluster 1 - [Mean = 4, Variance = 1.6]

Cluster 2 - [Mean = 0, Variance = 1.8]

Cluster 3 - [Mean = -4, Variance = 1.2]

Sample data points generated for testing the EM algorithm is as follows:
[-4.7992296 , -4.01709602, 4.0148895 , 0.04519512, -5.06339894, 2.87541568, 0.86967226, -2.8796678 , 4.92469682, -2.95873976, 1.71153414, -3.49728092, 1.45861897, -6.50318026, 4.38054232, 2.94354019, 0.78948193, 1.68393786, -3.7312411 , 1.23877229, -4.62013442, -0.40995411, -3.23438263, 5.24351688, 5.03148761, -3.56237757, 3.70982308, 5.57707522, 0.17756134, 2.76037168, -2.83507323, 3.07100039, 4.7106732 , -3.66989446, -4.17089294, -7.41633544, -5.73261326, 6.44504413, -0.15481344, -6.15444619, -6.06955754, 2.63466314, -4.37506931, 6.67254713, 3.92092281, -4.47653055, 3.10976208, -1.04346032, -4.81169808, 0.35476144, 4.06590366, -2.87029709, 4.93102187, 0.38682194, 3.66211849, 0.44011338, -4.60559172, -2.5636032 , 5.32806023, 4.46902767, 7.0768629 , -3.80096226, 2.9147181 , 1.36140363, -3.88296488, 0.23244367, 5.56871543, 3.24084374, 2.39252758, 3.06323084, 2.21711782, -3.78864429, -6.23138021, -3.54990649, 3.19599762, -0.42253938, 3.36608011, -0.10810009, 6.51525922, 4.98294599, -4.60712893, 3.0074766 , -1.57060911, -3.49931299, -4.2487759 , 3.14114963, 3.42790069, -4.77070404, -3.80537902, -3.18876962, -6.21167251, 3.97352117, 2.03608312, -2.88482193, -5.77993548, 0.95564781, 4.14318117, -1.66143309, 1.93051832, 0.47002926]

The Mixture model label for the same is as follows:
[2, 2, 0, 1, 2, 0, 1, 2, 0, 2, 1, 2, 1, 2, 0, 0, 1, 1, 2, 0, 2, 1, 2, 0, 0, 2, 0, 0, 1, 0, 2, 0, 0, 2, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 1, 0, 1, 2, 1, 0, 2, 0, 1, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0, 2, 1, 0, 0, 1, 0, 0, 2, 2, 2, 0, 1, 1, 1, 0, 0, 2, 0, 1, 1, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2, 1, 0, 0, 1, 1, 1]

All the clusters are given equal weightage i.e. 1/3. The number of datapoints belonging to each cluster is: **Cluster 1 - 39, Cluster 2 - 26 and Cluster 3 - 35**. The plot of the data points and the distribution is shown in Figure 1.

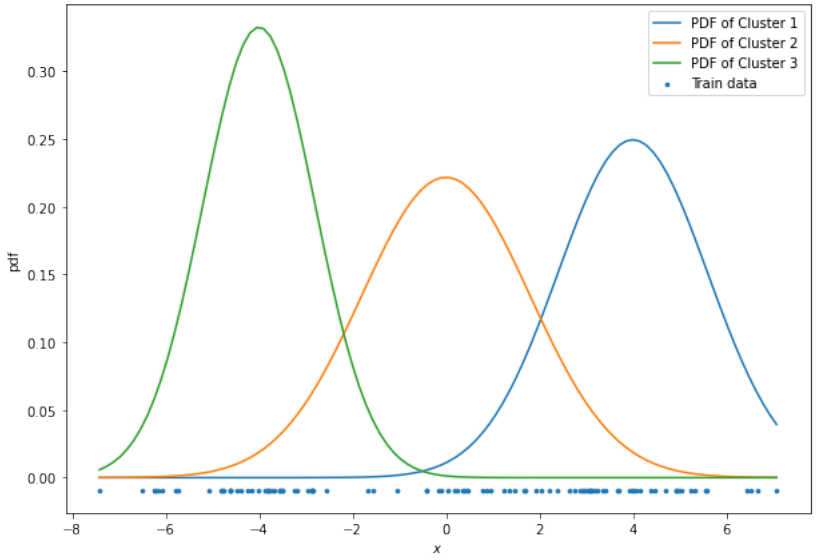

Figure 1: Plot of the dataset and the probability distribution function