EE 525x Data Analytics
Project Report: Gaussian Mixture Model based Clustering

Soumyabrata Talukder

Date: 30th Apr'18

1. Introduction

The course takes us through a couple of cardinal methods (e.g. K-means, hierarchical clustering) of unsupervised learning of a given data set. Such methods offer ways to visualize an unlabelled data set by segregating the data samples into user-defined number of clusters, based on their underlying properties, that are not intuitive otherwise. The limitation of the said methods has also been emphasized in the course lectures. The K-means method segregates the data into a defined number of clusters and also returns their respective centroids. However, if we want to add a set of fresh data samples into the learned model, membership of the fresh data is decided based on their respective nearest centroid only, irrespective of the distribution the population of the already formed clusters. Unlike K-means, hierarchical clustering method partitions the data set hierarchically depending upon the user-defined linkage of the data points/sets. In this method, the shape of the formed clusters is primarily governed by the definition of linkage, which equips this method with a degree of freedom for course control of the shape of the clusters. In both the methods, the resulting clusters and their members also depend upon the *distance* function. Although the said methods are popular, they cannot be directly applied in cases where the data samples are drawn from a (or a group of) specific distribution(s) and are periodically added into the model, since none of the methods models the underlying distribution of the data samples. The objective of this project is to review existing literature to understand and implement Gaussian mixture model based clustering method, which supposedly addresses the aforementioned shortcomings of K-means and hierarchical clustering methods, where the data set is drawn from a Gaussian mixture model.

2. Background (Gaussian Mixture Model)

Consider G numbers of p dimensional Gaussian distributions (hereinafter referred as *component distributions*), where the mean and covariance of the g^{th} distribution is denoted by μ_g and Σ_g respectively. Assume that n data samples are collected, by drawing one sample from a distribution at a time, where the probability of drawing from the g^{th} distribution in every turn, is given by a discrete probability density π_g ($\pi_g > 0, \sum_{g=1}^{G} \pi_g = 1$). The collected set of data forms a data matrix X ($\mathrm{X} \in \mathbb{R}^{n \times p}$). Such model of random sample generator is termed as **Gaussian Mixture Model** (GMM), where the model parameters are denoted by $\vartheta = (\pi_1, \pi_2, ..\pi_G, \mu_1, \mu_2, ..\mu_G, \Sigma_1, \Sigma_2, ..\Sigma_G)$. For all sampled data $\mathrm{x} \subset \mathrm{X}$, its density can be written as

$$f(x|\vartheta) = \sum_{g=1}^{G} \pi_g \varphi(x|\mu_g, \Sigma_g)$$

where

$$\varphi(x|\mu_g, \Sigma_g) = \frac{1}{\sqrt{(2\pi)^p|\Sigma_g|}} exp\{\frac{1}{2}(x-\mu_g)'\Sigma_g^{-1}(x-\mu_g)\}$$

3. Problem formulation

A two dimensional ($p = 2$) synthetic data set, containing 300 samples ($n = 300$) drawn from a GMM of three distributions ($G = 3$), defined by the model parameters as in Table 1, has been used for this project. A scatter diagram of the synthetic data is shown in Figure 1. Since the samples are drawn randomly from the distribution, we do not have *a priori* knowledge of the distribution from where the respective data samples are sourced. The problem here is to cluster the data set such that the clusters depict a best possible segregation of the data according to their membership in the original component distributions. The generation and analysis of the synthetic data as explained hereinafter, are performed in MATLAB.

Table 1 : GMM parameters

$Distribution No.(g)$	$Mean(\mu_g)$	$Covariance(\Sigma_g)$	$Mixing Proportion(\pi_g)$
1	`[5, 1]`	`[0.8 0.75; 0.75 0.8]`	0.4
2	`[6, 2]`	`[1 -0.75; -0.75 1]`	0.3
3	`[3, -4]`	`[1 -0.55; -0.55 1]`	0.3

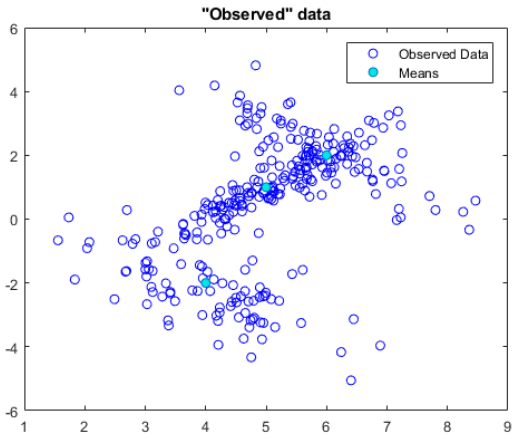

Figure 1: The synthetic data

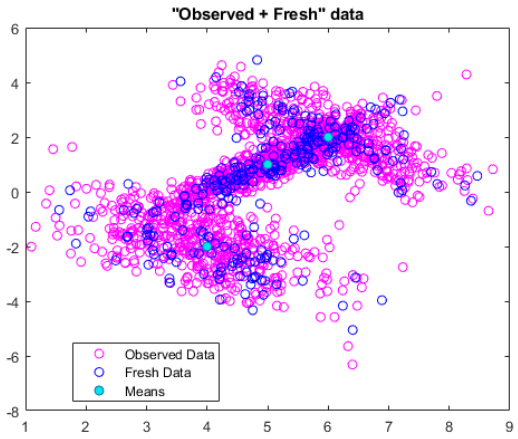

Figure 2: Data analysis by conventional approaches

First, K-means and hierarchical clustering methods are applied on the synthetic data set. The data set has been analyzed using various types of distance functions, while implementing K-means algorithm. Trials with different linkages (e.g. average, complete, ward and centroid) has also been performed, in case of hierarchical clustering. Finally, the data set has been fitted into a Gaussian Mixture Model (GMM), by learning the mean, variances and the membership probabilities of the original GMM, from where the data samples are drawn. Subsequently, 1500 fresh data samples are randomly drawn from the original GMM population, and are fitted in the learned models to observe how the clusters, formed using the different methods, grow in addition of fresh data. The total population of the initially observed data for clustering and the freshly drawn data is shown in Figure 2.

4. Clustering by conventional approaches

4.1. K-Means

For implementing K-Means algorithm MATLAB built-in function has been used, with K-Means ++ initialization method. Among the various distance functions, *Euclidean* (L-2), *Manhattan* (L-1) and *Cosine Similarities* are tried and it is observed that L-2 norm generates the most meaningful clustering. K-Means with *Cosine Similarities* fails to generate a relevant result in this case, since its capability of clustering data is limited to the cases where the actual clusters are oriented radially. The result obtained by K-Means with L-1 norm, is close to what has been obtained using L-2 norm, barring few spurious mixing of clusters. The obtained clusters, using L1 and L2 distance functions are depicted in Figure 3,4.

Figure 3: K-Means clustering of initial data using L2 norm

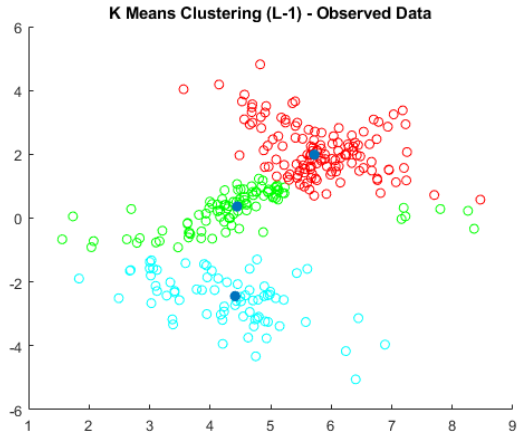

Figure 4: K-Means clustering of initial data using L1 norm

After clustering of the initial data using K-Means, the fresh data are also clustered according to their shortest distance from the already obtained cluster centers. It is observed that the growth of the clusters, on addition of fresh data, is forming a Voronoi partition around the cluster centers, in case of L2 distance function, while that for L1 distance is similar but with spurious mixing of clusters. The clustered complete data population by K-Means with L1 and L2 norm are captured in Figure 5, 6.

4.2. Hierarchical clustering

Hierarchical clustering is performed on the union of initial and fresh data samples, using different linkages e.g. *ward, complete, average* and *single*. Out of the said four trials, only *ward* and *complete* linkages generated somewhat meaningful result, while both *single* and *average* linkages, pulling almost the complete data set into a single cluster greedily. The clusters obtained using *ward* and *complete* are depicted in Figure 7, 8.

5. GMM based clustering

In contrast to the previously discussed conventional methods, GMM based clustering is performed, estimating the GMM parameters of the source distributions by analyzing the observed data samples. First we define a set of membership vectors $z_1, z_2, ..., z_n$ corresponding to the respective initially observed data

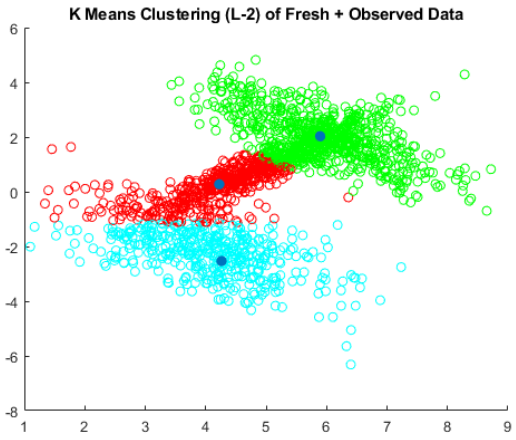

Figure 5: K-Means clustering of complete data using L2 norm

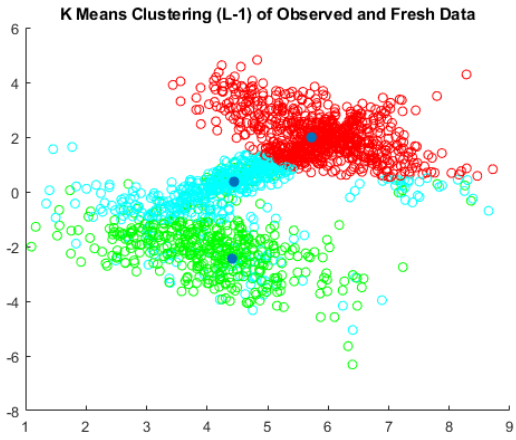

Figure 6: K-Means clustering of complete data using L1 norm

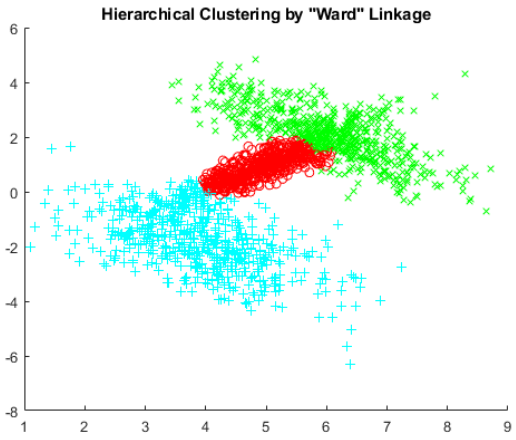

Figure 7: Hierarchical clustering of complete data using *ward* linkage

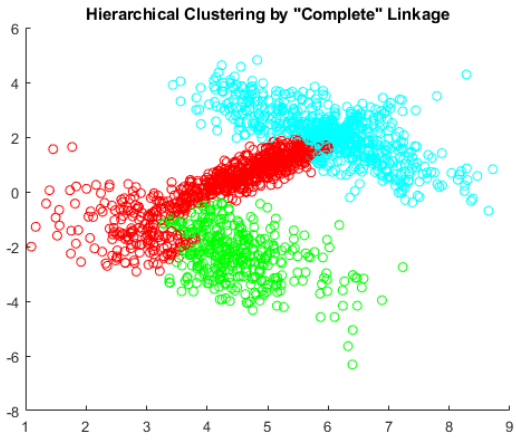

Figure 8: Hierarchical clustering of complete data using *complete* linkage

samples $x_1, x_2, ..., x_n$, where every membership vector $z_i = (z_{i1}, z_{i2}, ..., z_{iG})$ has G elements with the element z_{ig} indicating the g^{th} component membership of the i^{th} sample. Since source of every data sample is one of the distinct distributions, ideally $z_{ig} = 0$ for all g except g = k, for which $z_{ik} = 1$, indicating that the i^{th} sample has been drawn from the k^{th} component distribution. With such configuration, the complete data likelihood can be written as

$$L_c(\vartheta) = \prod_{i=1}^{n} \prod_{g=1}^{G} [\pi_g \varphi(x_i|\mu_g, \Sigma_g)]^{z_{ig}}$$

Taking natural logarithm of the above likelihood gives the complete data log likelihood

$$l_c(\vartheta) = \sum_{i=1}^{n} \sum_{g=1}^{G} z_{ig} [\log \pi_g + \log \varphi(x_i|\mu_g, \Sigma_g)]$$

The problem of estimating the GMM parameters can now be formally defined as finding the combination of model parameters and component membership vectors, such that the expected value of the complete data log likelihood is maximized. The expected value of the log likelihood function can be written as

$$Q(\vartheta) = \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z_{ig}} [\log \pi_g - \frac{p}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_g| - \frac{1}{2} \mathrm{Tr}(x_i - \mu_g)(x_i - \mu_g)' \Sigma_g^{-1}]$$

equivalently,

$$Q(\vartheta) = \sum_{g=1}^{G} n_g \log \pi_g - \frac{np}{2} \log 2\pi - \sum_{g=1}^{G} \frac{n_g}{2} \mathrm{Tr}(S_g \Sigma_g^{-1})$$

where $n_g = \sum_{i=1}^{n} \hat{z_{ig}}$, $S_g = \frac{1}{n_g} \sum_{i=1}^{n} \hat{z_{ig}} (x_i - \mu_g)(x_i - \mu_g)'$, $\hat{z_{ig}} = \frac{\hat{\pi_g} \varphi(x_i|\hat{\mu_g}, \hat{\Sigma_g})}{\sum_{h=1}^{G} \hat{\pi_h} f(x_i|\hat{\mu_g}, \hat{\Sigma_g})}$ and $\hat{\mu_g}, \hat{\pi_g}$ and $\hat{\Sigma_g}$ are the global maximizer of the expected log likelihood for a given z_{ig}.

It can be noted that the expected log-likelihood is a concave function, for a given value of z_{ig} and hence one can find a global maximizer of the expected log-likelihood function for a known z_{ig}. However, the elements of z_{ig} are random and therefore there is no direct approach to find a global maximum of the expected log-likelihood. Instead we embrace a well known iterative algorithm called **expectation maximization** (EM) algorithm to solve the problem. The EM algorithm is sequenced as following:

initialize $\hat{z_{ig}}$
while convergence criterion not met
  update $\hat{\pi_g} = \frac{n_g}{n}$
  update $\hat{\mu_g} = \frac{1}{n_g} \sum_{i=1}^{n} \hat{z_{ig}} x_i$
  update $\hat{\Sigma_g} = \frac{1}{n_g} \sum_{i=1}^{n} \hat{z_{ig}} (x_i - \hat{\mu_g})(x_i - \hat{\mu_g})'$
  update $\hat{z_{ig}}$
  chck convergence criterion
end while

The update rules of the above algorithm are found by analytically finding the global maximizers of expected value of the log-likelihood, for a given z_{ig}. MATLAB built-in function for fitting GMM, which implements EM algorithm, has been used for analyzing the initial observed data in this case. The model parameters estimated by this algorithm are shown in Table 2 below. Comparing the estimated parameter values with the actual values of Table 1, it is understood that the algorithm indeed generates a superlatively close estimation of the actual model parameters. The component distribution contours based on the estimated parameters, superimposed on the initial observed data is captured in Figure 9, to visualize the orientation of the components with respect to the actual distribution of the data samples.

Table 2 : GMM estimated parameters

$Distribution No.(g)$	$Mean(\mu_g)$	$Covariance(\Sigma_g)$	$Mixing Proportion(\pi_g)$
1	[4.94 0.9]	[0.93 0.81; 0.81 0.79]	0.46
2	[6.07 1.98]	[1.2 -0.93; -0.93 1.3]	0.27
3	[3.89 -2.1]	[0.9 -0.54; -0.54 1.07]	0.27

The final step of EM algorithm provides the estimated values of all z_i vectors, where the set $(z_{1k}, z_{2k}, ..., z_{nk})$ depicts the *a posteriori* membership of all data samples for k^{th} component distribution, given the finally estimated model parameters ϑ. Such *a posteriori* membership values can be exploited to visualize **soft cluster** of the data set with respect to a given estimated component distribution. The soft clustering of the synthetic data set with respect to *Component 1* is given in Figure 10.

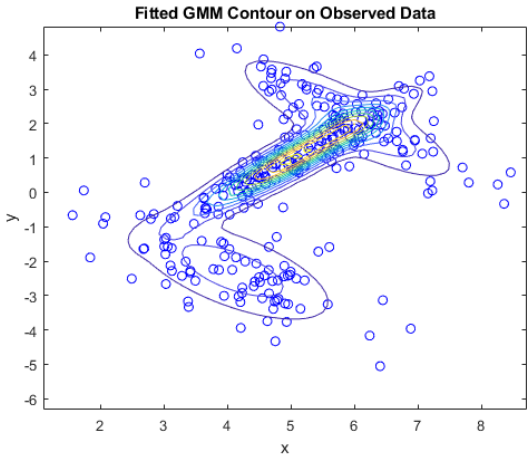

Figure 9: Estimated GMM Contour

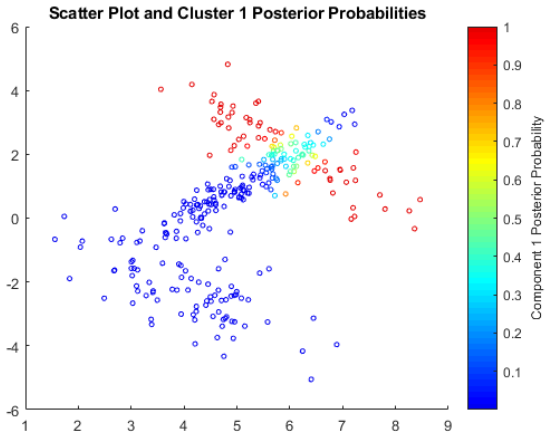

Figure 10: GMM soft clustering with respect to Cluster 1

The cluster, for which *a posteriori* membership of a particular sample is maximum, can be labelled as the cluster of that sample. Similarly all the samples can be allocated to one of the estimated component clusters to visualize a hard clusters of the data set. Such hard clustering of the synthetic data, obtained according to the maximum *a posteriori* membership, is captured in Figure 11. Now that the initially observed synthetic data is clustered based on the learned GMM parameters, we can also determine the expected membership of the fresh data samples by estimating their *a posteriori* membership, given the estimated model parameters. The clustering of the complete data set, based on this method, has been provided in Figure 12.

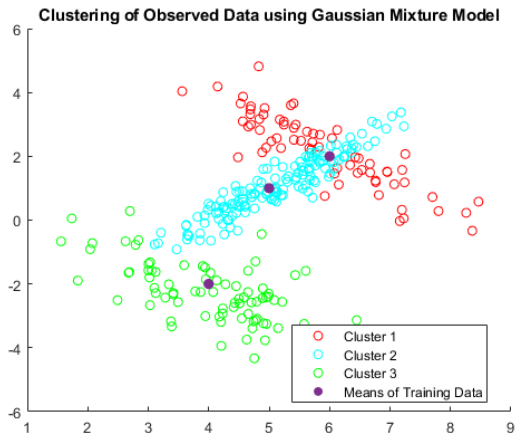

Figure 11: Hard clustering based on estimated GMM

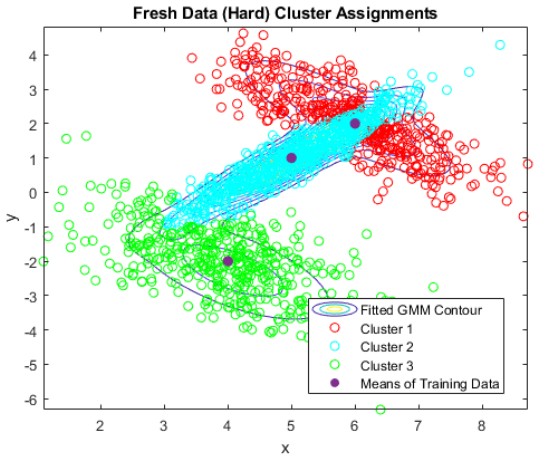

Figure 12: Hard clustering of the complete data set using estimated GMM

6. Challenges in implimentation

Although GMM based clustering method turns out to be an extremely powerful method in unsupervised learning, the implementation of the method is not straight forward. Time and again, challenges faced in implementation of this method, have been highlighted in literature and various solutions are also proposed. Few of such key challenges are enumerated below:

6.1. Parameter Initialization

GMM based clustering is highly sensitive to initialization of parameters, owing to the singularity riddled surface of the likelihood function, making the estimated parameters unreliable. Zhou et al. [1] proposes a **deterministic annealing** algorithm that alleviates this concern, by making the likelihood surface flatter. An auxiliary variable m $(m \in (0,1))$ is introduced, drawing from an increasing sequence of user-specified values. The number of deterministic iterations to be performed before commencement of EM algorithm, depends upon the length of this sequence. Deterministic annealing iteration is similar to EM algorithm except that estimation step is performed by the following

$$\hat{z^*_{ig}} = \frac{\hat{\pi_g}\varphi(x_i|\hat{\mu_g}, \hat{\Sigma_g})^m}{\sum_{h=1}^{G} \hat{\pi_h} f(x_i|\hat{\mu_g}, \hat{\Sigma_g})^m}$$

where $\hat{\pi_h}, \hat{\mu_g}$ and $\hat{\Sigma_g}$ are chosen arbitrarily in the first iteration.

6.2. Stopping criterion

One of the popular stopping criteria of EM algorithm is lack of progress of the log-likelihood, mathematically expressed by $l^{k+1} - l^k < \varepsilon$ $(\varepsilon \geq 0)$. While this approach works for clustering problem with smooth increment of log-likelihood (refer Figure 13), it causes premature stopping of the algorithm if log-likelihood increment is not enough smooth (refer Figure 14). In such cases, McNicholas et al.[2, 3] considers convergence criterion based on Aitken's acceleration, given by

$$l_{\infty}^{k+1} - l^k = \frac{l^{k+1} - l^k}{1 - a^k} < \varepsilon$$

where

$$a^k = \frac{l^{k+1} - l^k}{l^k - l^{k-1}}$$

is an acceleration parameter, used to obtain l_{∞}^{k+1}, the asymptotic log-likelihood value estimated at $(k+1)^{th}$ iteration.

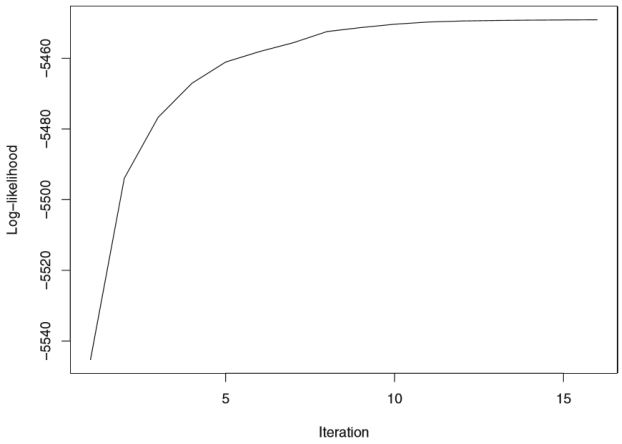

Figure 13: Smoothly evolving log-likelihood

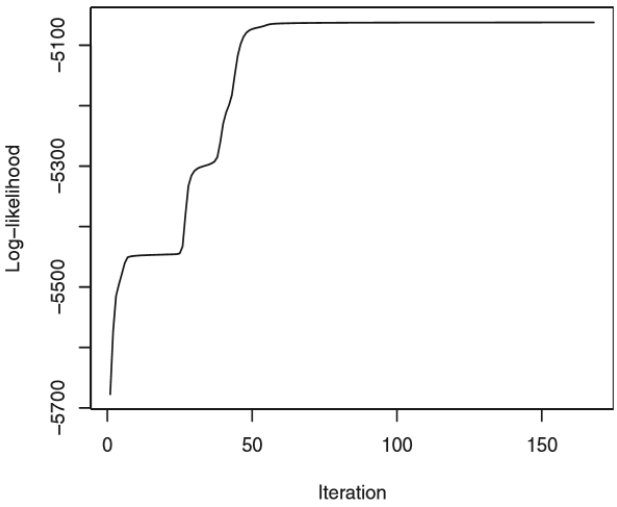

Figure 14: Roughly evolving log-likelihood

6.3. Limited scalability

For a p dimensional GMM with G component distributions, the number of model parameters is given by $G - 1 + Gp + Gp\frac{p+1}{2}$ where there are $G - 1$ numbers of mixing proportions, Gp numbers of parameters related to the means and the rest $Gp\frac{p+1}{2}$ numbers of parameters are attributed to the covariance matrices. It can be noted that the number of covariance related parameters grows exponentially with dimension of the data, which can make the algorithm computationally unfeasible while clustering very high-dimensional data. In such applications, wherever pertinent, one way to harness computational complexity is by constraining the structure of the covariance matrix while performing the *maximization* step of EM algorithm. The constraints can be of various types e.g. $\Sigma_g = \lambda_g I_p$, $\Sigma_g =_p$ or $\Sigma_g = \Sigma$. This technique is termed as *Gaussian Parsimonious Clustering Method* (GPCM) [4].

6.4. Model Selection

Often it is found that the exact number of component distributions of GMM corresponding to a data set in hand, is not *a priori* known. Even in some cases, where the number of components is known, there is a possibility that the samples are contaminated with Gaussian noise(s) with mean and variance both reasonably

high, causing existence of outliers in the data set. If the samples are high dimensional, unavailability of *a priori* knowledge of structure of the covariance matrix, may cause GPCM to generate unreliable result. To overcome such issues, it is customary to fit each GMM, with different structures of covariance matrices and various number of components, using EM algorithm. The best of these family of models is selected using some criterion and the associated clusters are reported. The most popular criterion for this purpose is the *Bayesian information criterion* (BIC) given by $BIC = 2l(\hat{\vartheta}) - \rho \log n$. Application of model selection using BIC is found in literature ([7]by Andrews et al. [6] by Vrbik et al.) over the last decade. Despite its popularity, BIC based model selection doesn't always generate reliable result and the method, being a brute force technique, comes with painfully complex computational burden at times.

7. Conclusion

As depicted by comparative study of Figure 5, 6, 8 and 12, it is obvious that performance of GMM based clustering is much superior as compared to conventional *K-Means* and *Hierarchical clustering* algorithms, when the samples come from a GMM population. The success of GMM based technique, in this case, is attributed to the efficient technique of learning the source model parameters. Despite several implementational challenges, GMM based learning technique finds a broad spectrum of application areas like bio-informatics [3], matrix completion [5], facial recognition, just to name a few. A key strength of GMM based clustering is, off course, its ability to offer a visualization of soft clusters, depicting the likelihood of each point to be a member of a cluster, which is not possible using *K-Means* and *Hierarchical clustering*.

The implementational issues of GMM based clustering opens up a wide area of research. Cognizing limitation of BIC based model selection, research efforts are observed for finding more meaningful alternative of BIC (*Integrated completed likelihood* (ICL) by Biernacki et al.). Recent research also shows efforts towards overcoming computational complexities of model selection and finding optimal estimation of GMM parameters ([8] Yihong Wu et al.).

8. Bibliography

[1] Zhou, H. and K. L. Lange (2010). On the bumpy road to the dominant mode. Scandinavian Journal of Statistics 37(4), 61263

[2] McNicholas, P. D. and T. B. Murphy (2010a). Model-based clustering of longitudinal data. The Canadian Journal of Statistics 38(1), 153168.

[3] McNicholas, P. D. and T. B. Murphy (2010b). Model-based clustering of microarray expression data via latent Gaussian mixture models. Bioinformatics 26(21), 27052712.

[4] McNicholas, P. D. and T. B. Murphy (2005). Parsimonious Gaussian mixture models. Technical Report 05/11, Department of Statistics, Trinity College Dublin, Dublin, Ireland.

[5] F. Lger, G. Yu, and G. Sapiro. Efficient matrix completion with gaussian models. arXiv preprint arXiv:1010.4050, 2010.

[6] Vrbik, I. and P. D. McNicholas (2014). Parsimonious skew mixture models for model-based clustering and classication. Computational Statistics and Data Analysis 71, 196210.

[7] Andrews, J. L. and P. D. McNicholas (2011a). Extending mixtures of multivariate t-factor analyzers. Statistics and Computing 21(3), 361373.

[8] Yihong Wu and Pengkun Yang (2018). Optimal estimation of Gaussian mixtures via denoised method of moments. Working Paper. Department of Statistics and Data Science, Yale University

[9] Andrew Ng Lecture 13, Machine Learning, Stanford University ($https://www.youtube.com/watch?v = LBtuYU - HfUg$)

[10] MATLAB Help *https* : *//www.mathworks.com/help/stats/fitgmdist.html*

Annexure

MATLAB code for the project:

```
clear all
nos = 100;
test_nos = 500;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Generate Observed Data and Fresh Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mu1 = [5 1];
sigma1 = [0.8 .75; .75 0.8];
mu2 = [6 2];
sigma2 = [1 -0.75; -0.75 1];
mu3 = [4 -2];
sigma3 = [1 -.55; -.55 1];
Mu = [mu1; mu2; mu3];
Sigma = cat(3,sigma1,sigma2,sigma3);

p = [0.4 0.3 0.3]; % Mixing proportions

gmTrue = gmdistribution(Mu,Sigma,p);
R = random(gmTrue,3*nos);
R_test = random(gmTrue,3*test_nos);
R_total = [R;R_test];

figure
plot(R(:,1),R(:,2),'bo');
hold on
s = scatter(Mu(:,1),Mu(:,2))
s.MarkerFaceColor = [0 0.9 0.9];
legend('Observed Data','Means')
title('"Observed" data');

figure
plot(R_test(:,1),R_test(:,2),'mo')
hold on
plot(R(:,1),R(:,2),'bo');
s = scatter(Mu(:,1),Mu(:,2))
s.MarkerFaceColor = [0 0.9 0.9];
legend('Observed Data','Fresh Data','Means','Location','best');
title('"Observed + Fresh" data');
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       K Means by Euclidean Norm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[idx,C] = kmeans(R,3);

figure
hold on

for i =1:nos*3
    if idx(i) == 1
```

```
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
        plot(R(i,1),R(i,2),'go')
    else
        plot(R(i,1),R(i,2),'co')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (L2) - Observed Data')

hold off

figure
hold on
for i =1:nos*3
    if idx(i) == 1
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
        plot(R(i,1),R(i,2),'go')
    else
        plot(R(i,1),R(i,2),'co')
    end
end
for i = 1:length(R_test(:,1))
    d = repmat(R_test(i,:),[3,1]);
    k = d - C;
    k_norm = vecnorm(k');
    idx_f = find(k_norm == min(k_norm));
    if idx_f == 1
        plot(R_test(i,1),R_test(i,2),'ro')
    elseif idx_f == 2
        plot(R_test(i,1),R_test(i,2),'go')
    else
        plot(R_test(i,1),R_test(i,2),'co')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (L-2) of Fresh + Observed Data')

hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          K Means by Manhattan Norm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[idx,C] = kmeans(R,3,'Distance','cityblock');
% scatter(C(:,1),C(:,2),50,'filled')
figure
hold on

for i =1:nos*3
    if idx(i) == 1
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
```

```
        plot(R(i,1),R(i,2),'co')
    else
        plot(R(i,1),R(i,2),'go')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (L-1) - Observed Data')

hold off

figure

hold on
for i =1:nos*3
    if idx(i) == 1
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
        plot(R(i,1),R(i,2),'co')
    else
        plot(R(i,1),R(i,2),'go')
    end
end
for i = 1:length(R_test(:,1))
    d = repmat(R_test(i,:),[3,1]);
    k = d - C;
    k_norm = vecnorm(k',1);
    idx_f = find(k_norm == min(k_norm));
    if idx_f == 1
        plot(R_test(i,1),R_test(i,2),'ro')
    elseif idx_f == 2
        plot(R_test(i,1),R_test(i,2),'go')
    else
        plot(R_test(i,1),R_test(i,2),'co')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (L-1) of Observed and Fresh Data')
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       K Means by Cosine Similarity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[idx,C] = kmeans(R,3,'Distance','cosine');
% scatter(C(:,1),C(:,2),50,'filled')
figure
hold on
for i =1:nos*3
    if idx(i) == 1
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
        plot(R(i,1),R(i,2),'co')
    else
```

```
        plot(R(i,1),R(i,2),'go')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (Cosine) - Observed Data')

hold off

figure

hold on
for i =1:nos*3
    if idx(i) == 1
        plot(R(i,1),R(i,2),'ro')
    elseif idx(i) == 2
        plot(R(i,1),R(i,2),'co')
    else
        plot(R(i,1),R(i,2),'go')
    end
end
for i = 1:length(R_test(:,1))
    d = repmat(R_test(i,:),[3,1]);
    k = d - C;
    k_norm = d.*C;
    idx_f = find(k_norm == min(k_norm));
    if idx_f == 1
        plot(R_test(i,1),R_test(i,2),'ro')
    elseif idx_f == 2
        plot(R_test(i,1),R_test(i,2),'go')
    else
        plot(R_test(i,1),R_test(i,2),'co')
    end
end

scatter(C(:,1),C(:,2),50,'filled')
title('K Means Clustering (Cosine) - Fresh and Observed data')

hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Hierarchical Clustering by 'Ward' Linkage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z = linkage(R_total,'ward');
T = cluster(Z,'maxclust',3);
figure
hold on
for i = 1:length(R_total(:,1))
    if T(i) == 1
        plot(R_total(i,1),R_total(i,2),'ro')
    elseif T(i) == 2
        plot(R_total(i,1),R_total(i,2),'gx')
    else
        plot(R_total(i,1),R_total(i,2),'c+')
    end
end
title('Hierarchical Clustering by "Ward" Linkage')
```

```
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Hierarchical Clustering by 'Single' Linkage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z = linkage(R_total,'single');
T = cluster(Z,'maxclust',3);
figure
hold on
for i = 1:length(R_total(:,1))
    if T(i) == 1
        plot(R_total(i,1),R_total(i,2),'ro')
    elseif T(i) == 2
        plot(R_total(i,1),R_total(i,2),'go')
    else
        plot(R_total(i,1),R_total(i,2),'co')
    end
end
title('Hierarchical Clustering by "Single" Linkage')
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Hierarchical Clustering by 'Complete' Linkage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z = linkage(R_total,'complete');
T = cluster(Z,'maxclust',3);
figure
hold on
for i = 1:length(R_total(:,1))
    if T(i) == 1
        plot(R_total(i,1),R_total(i,2),'ro')
    elseif T(i) == 2
        plot(R_total(i,1),R_total(i,2),'go')
    else
        plot(R_total(i,1),R_total(i,2),'co')
    end
end
title('Hierarchical Clustering by "Complete" Linkage')
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Hierarchical Clustering by 'Average' Linkage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Z = linkage(R_total,'average');
T = cluster(Z,'maxclust',3);
figure
hold on
for i = 1:length(R_total(:,1))
    if T(i) == 1
        plot(R_total(i,1),R_total(i,2),'ro')
    elseif T(i) == 2
        plot(R_total(i,1),R_total(i,2),'go')
    else
        plot(R_total(i,1),R_total(i,2),'co')
    end
end
title('Hierarchical Clustering by "Average" Linkage')
hold off
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Gaussian Mixture Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

options = statset('Display','final');
gm = fitgmdist(R,3,'Options',options)

figure
plot(R(:,1),R(:,2),'bo')
hold on
ezcontour(@(x,y)pdf(gm,[x y]),[min(R_total(:,1))
max(R_total(:,1))],[min(R_total(:,2)) max(R_total(:,2))])
title('Fitted GMM Contour on Observed Data')
hold off

%%%%%%%%%%%%%%%%%%%%%%%%      Hard Clustering       %%%%%%%%%%%%%%%%%%%%%%%%

idx = cluster(gm,R);
cluster1 = find(idx == 1); % |1| for cluster 1 membership
cluster2 = find(idx == 2); % |2| for cluster 2 membership
cluster3 = find(idx == 3); % |3| for cluster 3 membership

figure
scatter(R(cluster1,1),R(cluster1,2),'ro');
hold on
scatter(R(cluster2,1),R(cluster2,2),'co');
scatter(R(cluster3,1),R(cluster3,2),'go');
scatter(Mu(:,1),Mu(:,2),50,'filled')
legend('Cluster 1','Cluster 2','Cluster 3','Means of Training
Data','Location','best')
title('Clustering of Observed Data using Gaussian Mixture Model')
hold off

[idx_test,~,P_test] = cluster(gm,R_test);
cluster1 = find(idx_test == 1); % |1| for cluster 1 membership
cluster2 = find(idx_test == 2); % |2| for cluster 2 membership
cluster3 = find(idx_test == 3); % |3| for cluster 3 membership
figure
ezcontour(@(x,y)pdf(gm,[x y]),[min(R_total(:,1))
max(R_total(:,1))],[min(R_total(:,2)) max(R_total(:,2))])
hold on
scatter(R_test(cluster1,1),R_test(cluster1,2),'ro');
scatter(R_test(cluster2,1),R_test(cluster2,2),'co');
scatter(R_test(cluster3,1),R_test(cluster3,2),'go');
title('Fresh Data (Hard) Cluster Assignments')
scatter(Mu(:,1),Mu(:,2),50,'filled')
legend('Fitted GMM Contour','Cluster 1','Cluster 2','Cluster 3','Means of
Training Data','Location','best')
hold off

c1 = (idx == 1); % |1| for cluster 1 membership
c2 = (idx == 2); % |2| for cluster 2 membership
c3 = (idx == 3); % |3| for cluster 3 membership
P = posterior(gm,R);
```

```
figure
scatter(R(c1,1),R(c1,2),10,P(c1,1),'o')
hold on
scatter(R(c2,1),R(c2,2),10,P(c2,1),'o')
scatter(R(c3,1),R(c3,2),10,P(c3,1),'o')
hold off
clrmap = jet(80);
colormap(clrmap(9:72,:))
ylabel(colorbar,'Component 1 Posterior Probability')
title('Scatter Plot and Cluster 1 Posterior Probabilities')

%%%%%%%%%%%%%%%%%%%%%%%%      Soft Clustering       %%%%%%%%%%%%%%%%%%%%%%%%

threshold = [0.4 0.6];
idx_2 = find(P(:,1)>=threshold(1) & P(:,1)<=threshold(2));
numInBoth = numel(idx_2);
figure
scatter(R_test(cluster1,1),R_test(cluster1,2),'ro');
hold on
scatter(R_test(cluster2,1),R_test(cluster2,2),'co');
scatter(R_test(cluster3,1),R_test(cluster3,2),'go');
scatter(Mu(:,1),Mu(:,2),50,'filled');
plot(R_test(idx_2,1),R_test(idx_2,2),'ko','MarkerSize',10);
title('Fresh Data (Soft) Cluster Assignments');
legend('Cluster 1','Cluster 2','Cluster 3','Means of Training Data','Data
hard to distinguish between two clusters','Location','best');
hold off
```