Assignment 1

Name: Mousaalreza Dastmard

Course: Advances in data analysis and statistical modelling

1) Using “MixSampling.m” function, generate 300 observations by a bivariate Gaussian mixture with a structure of three groups.
In order to have a matrix of 300 observations with 2 variables which are categorized in 3 groups I set the arguments of the function MixSampling.m as follow:

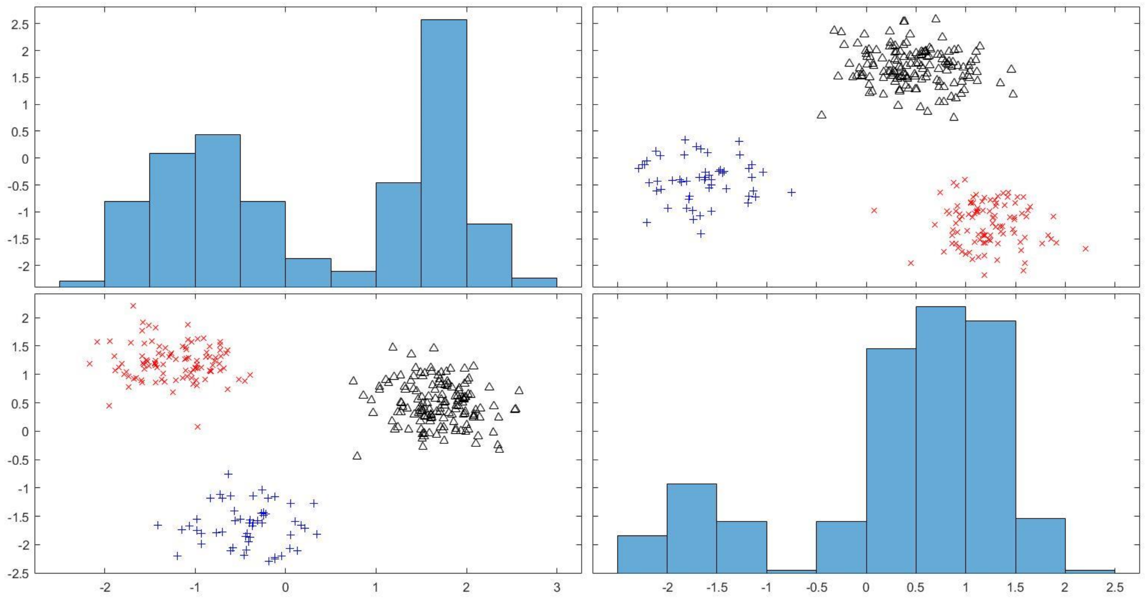
Table 1: Inputs of MixSampling.m

Argument	value	Description
n	300	number of Objects/Observations
Pr	[0.2 0.3 0.5]	vector of a priori probabilities (3 x 1) K=3, The summation must equal to 1
dist	9	constant distance in symmetric matrix
csifma	0	error to perturbate centroids
esigma	1.5	standard deviation of error

```
[X,U,u]=MixSampling(300,[.2;.3;.5], 9,0,1.5);
```

The final results are stored on X and the membership matrix is U and u is categorical variable shows the cluster index.
Based on the inputs defined in Table 1 the results of bivariate observations show on figure 1

Figure 1



2) Apply the Well Structured Partition (WSP) and Well Structured Perfect Partition (WSPP) models on data generated in the step 1) fixing the number of clusters equal to 3. Compare the membership matrices obtained by the two models with the “real” memberships values. Comment the results.

First, to apply WSP and WSPP it is highly needed to have distance matrix of observations, to do so function pdist.m output the distance as a vector and to convert it as a pairwise distance matrix squaredorm.m can be used.

```
D = pdist(X);
D = squareform(D);
```

The matrix D is 300x300 shows the distance between observations that can be inputted in WSP.m/WSPP.m (K=3, Rndstart=10) and to make comparison in form of confusion matrix Cm_P and Cm_PP we can use UOtt_P’U and UOtt_PP’U

```
[UOtt_P,DbOtt_P, DwOtt_P, fOtt_P,iterOtt_P]=WSP(D, K, Rndstart);
Cm_P = UOtt_P'*U;

[UOtt_PP,DbOtt_PP, DwOtt_PP, fOtt_PP,iterOtt_PP]=WSPP(D, K, Rndstart);
Cm_PP = UOtt_PP'*U;
```

Processing Results for WSP:

```
Well Structured Partition: loop=1, f=0.0338153 itr=5
Well Structured Partition: loop=2, f=0.0338153 itr=5
Well Structured Partition: loop=3, f=0.0338153 itr=5
Well Structured Partition: loop=4, f=0.0338153 itr=5
Well Structured Partition: loop=5, f=0.0338153 itr=5
Well Structured Partition: loop=6, f=0.0338153 itr=5
Well Structured Partition: loop=7, f=0.0338153 itr=5
Well Structured Partition: loop=8, f=0.0338153 itr=5
Well Structured Partition: loop=9, f=0.0338153 itr=5
Well Structured Partition: loop=10, f=0.0338153 itr=5
Well Structured Partition (Final):  loopOtt=5, fOtt=0.0338153, iter=5
Cm_P =

     0     0   160
    52     0     0
     0    88     0
```

Processing Results for WSPP:

```
Well Structured Perfect Partition: loop=2, f=0.12713 itr=4
Well Structured Perfect Partition: loop=3, f=0.368872 itr=18
Well Structured Perfect Partition: loop=4, f=0.368874 itr=15
Well Structured Perfect Partition: loop=5, f=0.368872 itr=18
Well Structured Perfect Partition: loop=6, f=0.368869 itr=19
Well Structured Perfect Partition: loop=7, f=0.368874 itr=19
Well Structured Perfect Partition: loop=8, f=0.36887 itr=19
Well Structured Perfect Partition: loop=9, f=0.12713 itr=4
Well Structured Perfect Partition: loop=10, f=0.12713 itr=4
Well Structured Perfect Partition (Final):  loopOtt=2, fOtt=0.12713, iter=4
Cm_PP =

     0    88     0
    52     0     0
     0     0   160
```

The both procedures work perfect to find the clusters. It might because of the parameters that we set to generate the observations and the distinction between clusters is high so the 2 procedures work well on this data.

3) Generate 4 noise variables of 300 observations by a Gaussian distribution with μ=0 and σ=2. Add these 4 variables to the data generated in the step 1). Now, you have a new 300×6 data matrix.

Gaussian variables can be produced by randn.m function in Matlab, since the default results are based on mu=0, sigma=1 we should add a statistically trick as follow:

```
Y = randn(300,4)*2;
```

Multiplying by 2 convert the data to have sigma = 2
Lets name the new observations matrix X_new which is 300x6, so

```
X_new = [X Y];
```

4) Apply the k-means model on this new data set fixing the number of clusters equal to 3. Compare the membership matrix obtained by k-means with the "real" memberships values. Comment the results.

In order to cluster the X_new with 3 clusters we would use kmeans.m as below:

```
[loopOtt_KM,UOtt_KM,fOtt_KM,iterOtt_KM] = kmeansVICHI(X_new,K,Rndstart);
Cm_KM = UOtt_KM'*U;
```

Result of comparison is:

```
Cm_KM =

    17    37     0
    21    71     0
    40     0   114
```

It seems that by adding noise to the data kmeans algorithm get confused to find the clusters very well.

5) "Using a dimensionality reduction technique, such as principal components analysis (PCA), to create new variables and then using cluster analysis, such as k-means, to form groups using these new variables". This technique is called as "Tandem Analysis". Use this approach on the new data set generated in the step 3) fixing the number of clusters equal to 3. Compare the membership matrix obtained by k-means with the "real" memberships values. Comment the results

Matlab 2018 support PCA with the function PCA.m. with the below simple code we are able to setup PCA.

```
[coeff,score,latent,tsquared,explained,mu] = pca(Z);
```

Take a look at ratio of total variance explained by each component

```
explained =

   26.2423
   22.6805
   17.9587
   16.4826
    8.7787
    7.8573
```

And calculating the cumulative summation of the explained. About 83% of total variance explained by the 4 first component.

```
cumsum(explained)

ans =

   26.2423
   48.9228
   66.8814
   83.3640
   92.1427
  100.0000
```

So, we take the 4 first component and use the kmean on them.

```
Factors = score(:,1:4);
[loopOtt_TN,UOtt_TN,fOtt_TN,iterOtt_TN] = kmeansVICHI(Factor, K,Rndstart);
Cm_TN = UOtt_TN'*U;

Cm_TN =

    44    24    43
    35    33    33
    27    39    22
```

6) Apply Reduced k-means (REDKM) on data generated in the step 3) fixing the number of clusters equal to 3. Compare the membership matrix obtained by the REDKM with the "real" memberships values. Comment the results underling the simultaneous approach advantages with respect to the sequential approach.

The code to use Reduced k-means is as follow:

```
[Urkm_RKM,Arkm_RKM, Yrkm_RKM,frkm_RKM,inrkm_RKM]=REDKM(X_new, K, Q, Rndstart)
Cm_RKM = Urkm_RKM'*U;
```

Q is the parameter of number of factors which is set to 2. The Final results is as below:

```
Cm_RKM =
     0     0   160
     0    88     0
    52     0     0
```
Reduced k-means work better than regular kmeans when it faces noise on data as the results proof it. It should be mentioned that reason that non-zero numbers are not on diagonal is that the algorithm find the clusters very well but it assign different index to clusters which is not important and by permutation (it means change the index of clusters) it can be solved:

```
Cm_RKM =

   160     0     0
     0    88     0
     0     0    52
```

7) Load MatLab workspace named “data02.mat”. Repeat the steps 5) and 6) on the ECSI data set. Define the partition obtained by both approaches and comment the results.

For the new data set we set number of factors Q=2 and Rndstar = 25. Number of cluster as previous is 3. Finally the geraphical results is shown as below using gplotmatrix.m function

```
load data02.mat
[Urkm_RKMND,Arkm_RKMND, Yrkm_RKMND,frkm_RKMND,inrkm_RKMND] = REDKM(X, K, Q, Rndstart);
figure
gplotmatrix(Yrkm_RKMND,[],Urkm_RKMND*[1 2 3]','brkmcywv','+x^d.*os',[],'off','hist')
```

Figure 2