Q1: Given the data file "Q1-Data.xlsx"

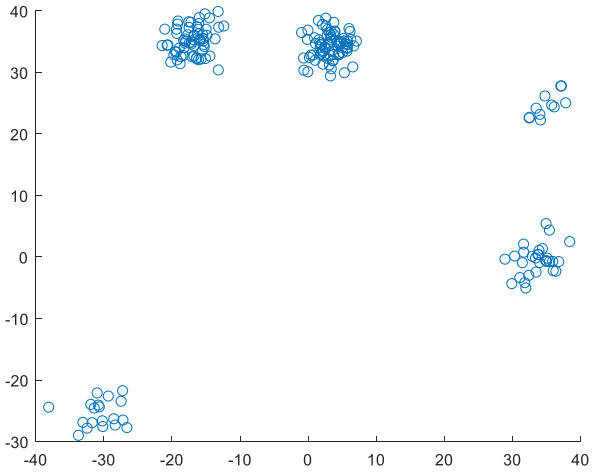
a) Implement the K-means clustering algorithm for K=5.
b) Implement the GMM algorithm with 5 Gaussians.

Report the clusters centers and determine the Gaussian parameters.

Solution:

File-Reading

```
Data = xlsread('Q1-DATA.xlsx');
```



Initialization

```
K=5;
min_data = min(Data);
max_data = max(Data);
centers_x = (max_data(1)-min_data(1)).*rand([K, 1])+ min_data(1);
centers_y = (max_data(2)-min_data(2)).*rand([K, 1])+ min_data(2);
```

Part a. K-means Clustering

In this part, we are asked to use K-means clustering algorithm with 5 clusters to cluster the data. In this algorithm, we start off with randomly initial cluster centers (initialized as in the above code snippet). We then assign membership to each point in the dataset based on its proximity to each cluster center. The algorithm runs for 100 iterations or until the cost function does not decrease. The code is in the appendix. The cost function used is,

$$C = \sum_{n=1}^{N} x_n - \mu_{kn}^{i}$$

where:
x_n is the n^{th} data point
$\mu_{kn}^{i} \coloneqq argmin_k(dist(\mu_k^i, x_n))$ is the cluster center to which x_n has been assigned at the previous step. The distance metric is the Euclidean norm. i is the step number (iteration).

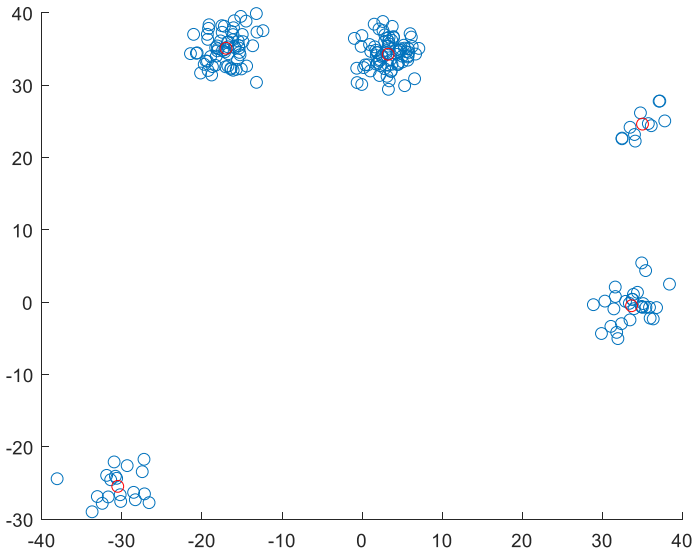
Results

Since the objective (cost) function is not convex, the algorithm suffered from poor convergence most of the time. Below are some successful and poor results:

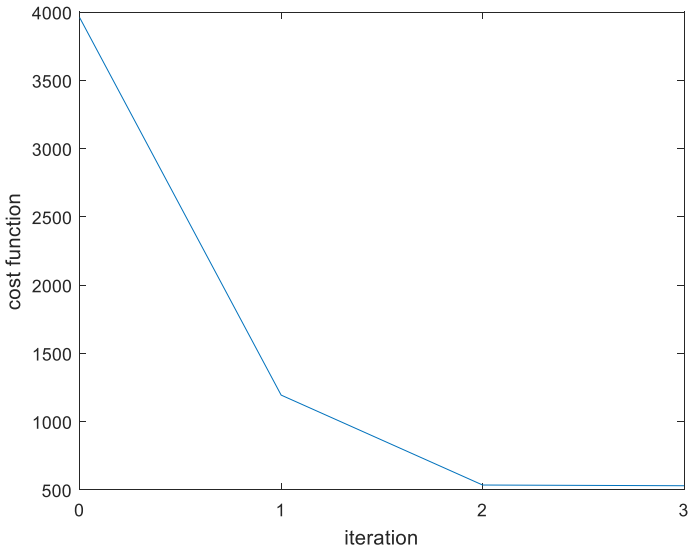
Successful Results:

The following figure shows the initial clusters locations (in red) w.r.t to the data points (in blue).

After 3 steps, the algorithm converged to the following cluster centers:



The convergence of the cost function is:

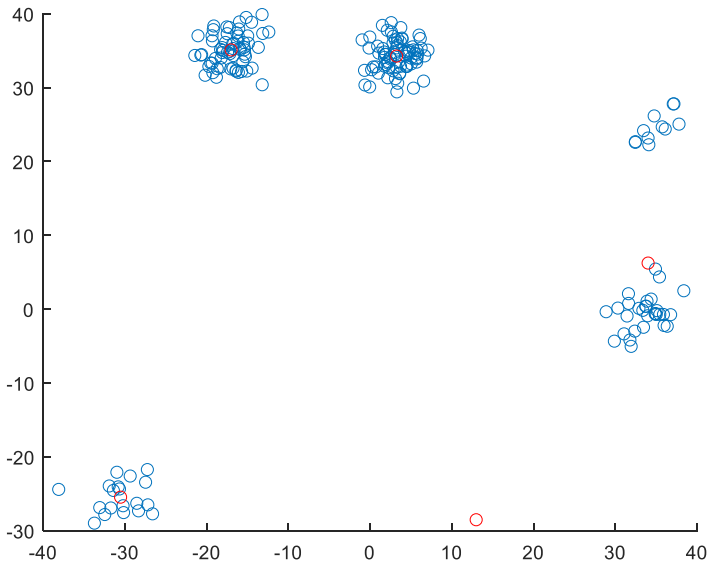


Where the initial cost was 3960 and the final ("optimal") cost was 530. Note that it only took 4 steps for the algorithm to converge.

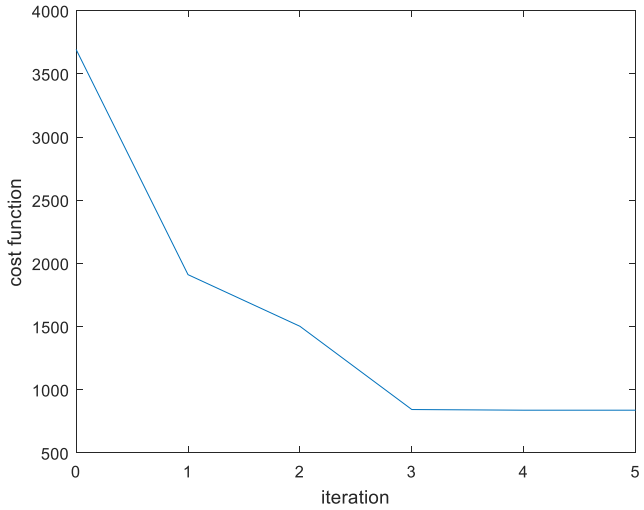
The cluster centers are

33.7	-0.6
-16.9	35.0
3.34	34.2
35.1	24.5
-30.4	-25.6

The final results after convergence after 6 steps are in the following figure. As it can be seen, one centroid is far off close to the x-axis, while one cluster (top-right) is not properly assign a centroid.



The convergence w.r.t iteration is:



The final cost is around 840, which is remarkably lower than in the previous attempt.

Part b. Gaussian Mixture Model

In this part, we use GMM iterative algorithm to cluster the data with $K = 5$ as the number of Gaussians. Since there are 5 clusters, I initialized the Gaussian weights $\pi_k = \frac{1}{5}$ for each cluster.

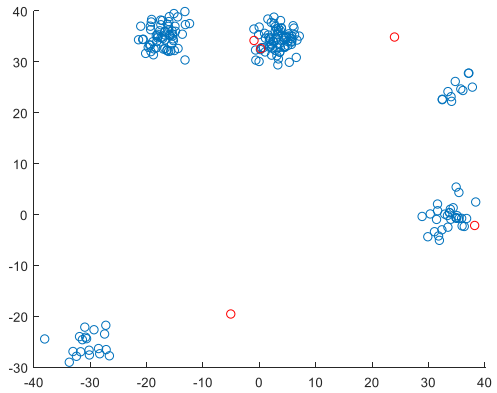
The initialization of the centers μ_k's is carried out in the same way as in part a. The initialization of the covariance matrices is $\Sigma_k = \sigma I$, I chose σ to be 2.
In order to calculate the Gaussian response for the data given a center μ_k and Σ_k, I used "mvnpdf" function in Matlab, as follows:

```
mvnpdf(Data, MUS(k, :), SIGMAS(:, :, k)+ 1e-10*eye(2,2));
```

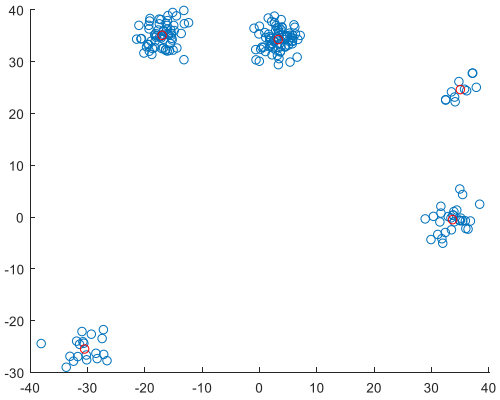
Where it can calculate the pdf for the 200 data points at once for each center-covariance matrix pair. I had to add "`1e-10*eye(2,2)`", so as to make sure that the covariance matrix is non-zero.

Results

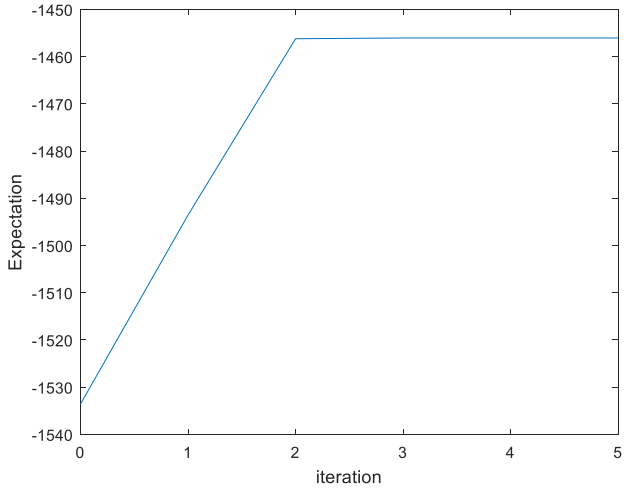
The following figure shows the initial centers locations (in red) w.r.t to the data points (in blue).



After 5 steps, the algorithm converges to the following centers.



The expectation of the log-likelihood w.r.t iteration is shown below.



The Gaussian parameters are as follows:

mean vectors	x-coordinate	y-coordinate
μ_1	3.34	34.16
μ_2	-30.42	-25.57
μ_3	33.74	-0.61
μ_4	35.10	24.51
μ_5	-16.89	34.99

The covariance matrices obtained are as follows:

Σ_1

3.6487	0.3241
0.3241	4.2056

Σ_2

7.3213	0.5728
0.5728	4.3738

Σ_3

4.7308	1.6956
1.6956	5.2747

Σ_4

3.2587	2.5919
2.5919	3.5009

Σ_5

4.2214	0.9101
0.9101	4.9011

The weights $\pi_k{}'s$ are as follows:

π_1	π_2	π_3	π_4	π_5
0.077	0.019	0.030	0.011	0.063

Q1-a Code:

```
clear all
close all
clc;
Data = xlsread('Q1-DATA.xlsx');

%K-mean
K=5;
min_data = min(Data);
max_data = max(Data);

centers_x = (max_data(1)-min_data(1)).*rand([K, 1])+ min_data(1);
centers_y = (max_data(2)-min_data(2)).*rand([K, 1])+ min_data(2);
centers = [centers_x centers_y];
scatter(Data(:, 1), Data(:,2));
hold on
scatter(centers(:, 1), centers(:, 2), 'r');
membership_map = zeros(200, 1);
N=10;
cost_current = zeros(N, 1);
iter = 0;
while iter<N
    %assign membership
    for i=1:200
        dist_min = Inf;
        for k=1:K
            distance = sqrt((Data(i,1)-centers(k,1)).^2+...
                (Data(i,2)-centers(k,2)).^2);
            if distance <dist_min
               dist_min = distance;
               membership_map(i) = k;
            end
        end
        cost_current(iter+1) = cost_current(iter+1) + dist_min;
    end
    %update centers
    for k=1:5
        clus_memb_idx = find(membership_map==k);
        if numel(clus_memb_idx)~=0
            centers(k, :) = sum(Data(clus_memb_idx, :), 1)...
                /numel(clus_memb_idx);
        end
    end
    if (iter > 1) && (cost_current(iter+1) >=cost_current(iter))
        break
    end
    iter=iter +1;

end
```

```
figure;
scatter(Data(:, 1), Data(:,2));
hold on
scatter(centers(:, 1), centers(:, 2), 'r');
membership_map = zeros(200, 1);
```

Q1-a Code:

```
clear all
close all
clc;
Data = xlsread('Q1-DATA.xlsx');

K=5;
min_data = min(Data);
max_data = max(Data);
centers_x = (max_data(1)-min_data(1)).*rand([K, 1])+ min_data(1);
centers_y = (max_data(2)-min_data(2)).*rand([K, 1])+ min_data(2);

MUS = [centers_x centers_y];

scatter(Data(:, 1), Data(:, 2))
hold on
scatter(MUS(:, 1), MUS(:, 2), 'r')

gauss_weights = 1/K*rand([1, K]);
r = zeros(200, 5);
sigma_0 = 2;
SIGMAS = repmat(sigma_0*eye(2), [1, 1, 5]);
iter = 1;
N = 1000;

while iter<=N
    expecation_vec(iter) = 0;
    for k=1:K
        r(:,k) = gauss_weights(k).*mvnpdf(Data, MUS(k, :), SIGMAS(:, :, k)+
1e-10*eye(2,2));
    end
    %normalize r
    r = r./sum(r, 2);
    %update MUS and SIGMAS
    gauss_weights = sum(r, 1)/N;
    for k=1:K
        MUS(k,:) = sum(r(:,k).*Data, 1)./(sum(r(:, k), 1));
        SIGMAS(:, :, k) = (Data-MUS(k,:))'*diag(r(:, k))*(Data-
MUS(k,:))/sum(r(:, k), 1);
    end
    for k=1:K
        log_likelihood = log(mvnpdf(Data, MUS(k, :), SIGMAS(:, :, k)+1e-
10*eye(2)));
        log_likelihood = max(log_likelihood, -1000);
        expecation_vec(iter) = expecation_vec(iter) +
sum(log_likelihood.*r(:, k),1);
```

```
        expecation_vec(iter) = expecation_vec(iter) +
log(gauss_weights(k)+1e-10).*sum(r(:,k), 1);
    end
    if (iter ~=1) && (expecation_vec(iter)<=expecation_vec(iter-1))
      break
    end
    iter = iter + 1;
end

figure;
scatter(Data(:, 1), Data(:, 2))
hold on
scatter(MUS(:, 1), MUS(:, 2), 'r')
```

Q2: In this problem, you will perform K-means clustering manually, with K = 2, on a small example with n = 6 observations and p = 2 features. The observations are as follows.

Obs.	X1	X2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

(a) Plot the observations using MATLAB or any other software package.

(b) Randomly assign a cluster label to each observation. Report the cluster labels for each observation.

(c) Compute the centroid for each cluster.

(d) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

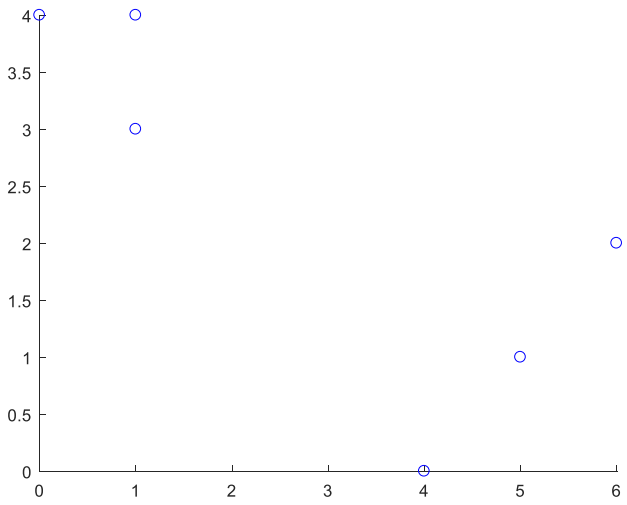
(e) Repeat (c) and (d) until the answers obtained stop changing.

(f) In your plot from (a), color the observations according to the cluster labels obtained.

Solution:

a.

The data points are shown as follows:



b.

In this part, I randomly assigned labels to the data points as follows:

Data point		Label
1	4	1
1	3	2
0	4	1
5	1	1
6	2	2
4	0	2

c.

Based on the above assignment, the centroids are:

$$C_1 = < \frac{1+0+5}{3}, \frac{4+4+1}{3} > = < 2, 3 >$$

$$C2 = < \frac{1+6+4}{3}, \frac{3+2+0}{3} > = < 3.67, 1.67 >$$

d.

The distances (squared) between each data point and the centroids are as follows:

Data point		Distance to C1	Distance to C2
1	4	2	12.56
1	3	1	8.89
0	4	5	18.89
5	1	13	2.22
6	2	17	5.56
4	0	13	2.89

According to the distance rule, the new cluster labels are:

Data point		Label
1	4	1
1	3	1
0	4	1
5	1	2
6	2	2
4	0	2

e.

Here we re-evaluate the centroids based on the new memberships of the data points:

$$C_1 = < \frac{1+1+0}{3}, \frac{4+3+4}{3} > = < 0.67, 3.67 >$$

$$C_2 = < \frac{5+6+4}{3}, \frac{1+2+0}{3} > = < 5, 1 >$$

The new Euclidean distances (squared) between the centroids and the data points are:

Data point		Distance to C1	Distance to C2
1	4	0.22	25
1	3	0.56	20
0	4	0.55	34
5	1	25.89	0
6	2	31.22	2
4	0	24.56	2

Data point		Label
1	4	1
1	3	1
0	4	1

The new labels are:

5	1	2
6	2	2
4	0	2

Note that the labels did not change from the previous step. Therefore the algorithm converged in 2 steps.

f.

The following figure shows the data points and their labels. Red corresponds to class 1 and Blue corresponds to class 2.