ECE175 HW5 Report

Di Guan (A91041815)

Problem 1.

Consider a random variable X whose distribution according to a Gaussian mixture

$$P_X(\mathrm{x}) = \sum_{c=1}^{C} \pi_c G(\mathrm{x}; \mu_c, \Sigma_c)$$

Show that X has mean

$$\mu_x = E_X[\mathrm{x}] = \sum_{c=1}^{C} \pi_c \mu_c$$

and covariance

$$\Sigma_x = E_X[(\mathrm{x} - \mu_x)(\mathrm{x} - \mu_x)^T] = \sum_{c=1}^{C} \pi_c [\Sigma_c + (\mu_c - \mu_x)(\mu_c - \mu_x)^T]$$

Proof:

$$\begin{aligned} with\ P_X(\mathrm{x}) &= \sum_{c=1}^{C} \pi_c G(\mathrm{x}; \mu_c, \Sigma_c) \\ \mu_x = E_X[\mathrm{x}] &= E_Z[E_{X|Z}[x|c]] \\ &= \sum_{c=1}^{C} E_{X|Z}[x|z=c] P_Z(z=c) \\ &= \sum_{c=1}^{C} E[G(x; \mu_c, \Sigma_c)] \pi_c \\ &= \sum_{c=1}^{C} \mu_c \pi_c \end{aligned}$$

$$
\begin{aligned}
\Sigma_x =& E_X[(\mathrm{x}-\mu_x)(\mathrm{x}-\mu_x)^T] \\
=& E_Z[E_{X|Z}[(\mathrm{x}-\mu_x)(\mathrm{x}-\mu_x)^T|c]] \\
=& E_Z[E_{X|Z}[(\mathrm{x}-\mu_c+\mu_c-\mu_x)(\mathrm{x}-\mu_c+\mu_c-\mu_x)^T|c]] \\
=& E_Z[E_{X|Z}[(\mathrm{x}-\mu_c)(\mathrm{x}-\mu_c)^T+(\mathrm{x}-\mu_c)(\mu_c-\mu_x)^T \\
& +(\mu_c-\mu_x)(\mathrm{x}-\mu_c)^T+(\mu_c-\mu_x)(\mu_c-\mu_x)^T|c]] \\
=& E_Z[E_{X|Z}[(\mathrm{x}-\mu_c)(\mathrm{x}-\mu_c)^T|c]+E_{X|Z}[(\mathrm{x}-\mu_c)(\mu_c-\mu_x)^T|c] \\
& +E_{X|Z}[(\mu_c-\mu_x)(\mathrm{x}-\mu_c)^T|c]+E_{X|Z}[(\mu_c-\mu_x)(\mu_c-\mu_x)^T|c]] \\
=& E_Z[\Sigma_c+E_{X|Z}[(\mathrm{x}-\mu_c)|c](\mu_c-\mu_x)^T+(\mu_c-\mu_x)E_{X|Z}[(\mathrm{x}-\mu_c)^T|c]+(\mu_c-\mu_x)(\mu_c-\mu_x)^T] \\
=& E_Z[\Sigma_c+(E_{X|Z}[\mathrm{x}|c]-\mu_c)(\mu_c-\mu_x)^T+(\mu_c-\mu_x)(E_{X|Z}[\mathrm{x}^{\mathrm{T}}|c]-\mu_c^T)+(\mu_c-\mu_x)(\mu_c-\mu_x)^T] \\
=& E_Z[\Sigma_c+(\mu_c-\mu_c)(\mu_c-\mu_x)^T+(\mu_c-\mu_x)(\mu_c^T-\mu_c^T)+(\mu_c-\mu_x)(\mu_c-\mu_x)^T] \\
=& E_Z[\Sigma_c+(\mu_c-\mu_x)(\mu_c-\mu_x)^T] \\
=& \sum_{c=1}^{C}[\Sigma_c+(\mu_c-\mu_x)(\mu_c-\mu_x)^T]P_Z(c) \\
=& \sum_{c=1}^{C}\pi_c[\Sigma_c+(\mu_c-\mu_x)(\mu_c-\mu_x)^T]
\end{aligned}
$$

Problem 2.

Consider a mixture of two Gaussian

$$P_X(\mathrm{x})=\sum_{c=1}^{2}\pi_c G(\mathrm{x};\mu_c,\Sigma_c)$$

where the covariance matrices are diagonal,

$$\Sigma_c = diag(\sigma_{c,1}^2,\sigma_{c,2}^2)$$

and a training sample of five points

$$D=[(-2.5,-1),(-2,0.5),(-1,0),(2.5,-1),(2,1)]$$

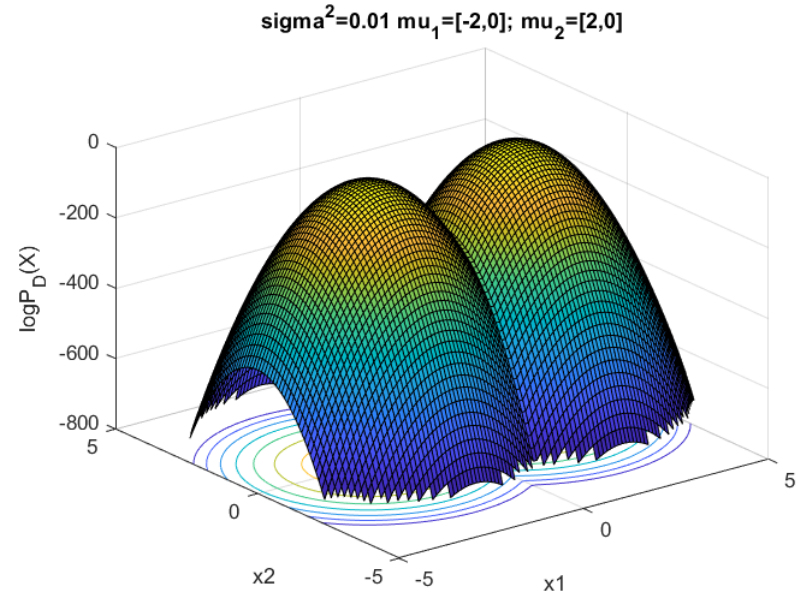
(a).

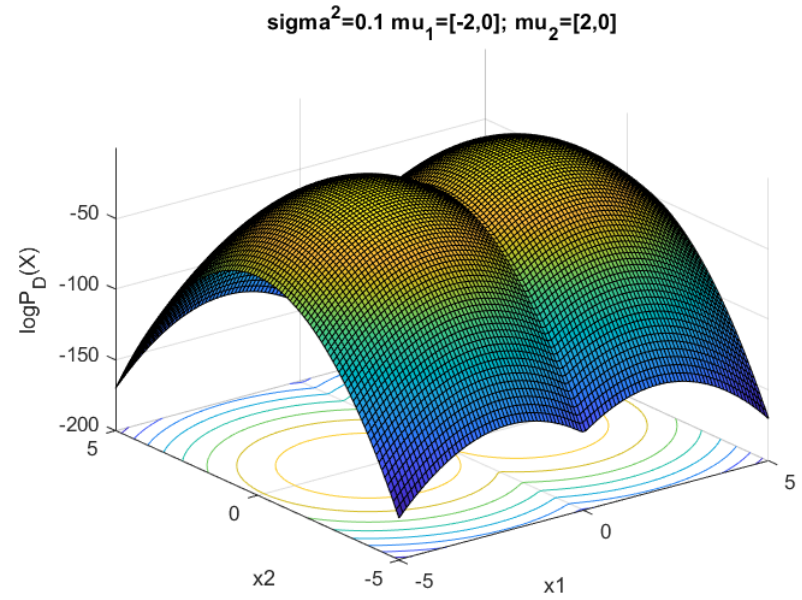
assume the following hold

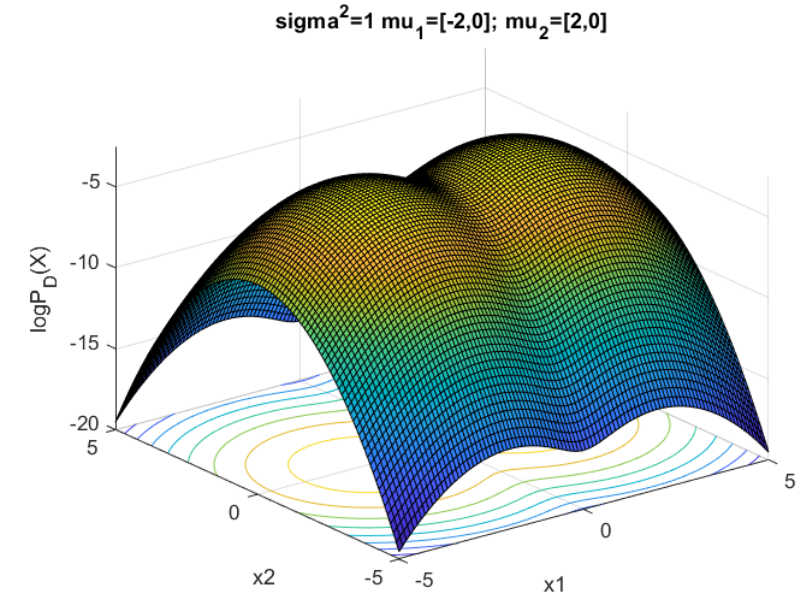
$$
\begin{aligned}
\mu_1 &= -\mu_2 \\
\Sigma_1 &= \Sigma_2 = \sigma^2 I \\
\pi_1 &= \pi_2 = \frac{1}{2}
\end{aligned}
$$

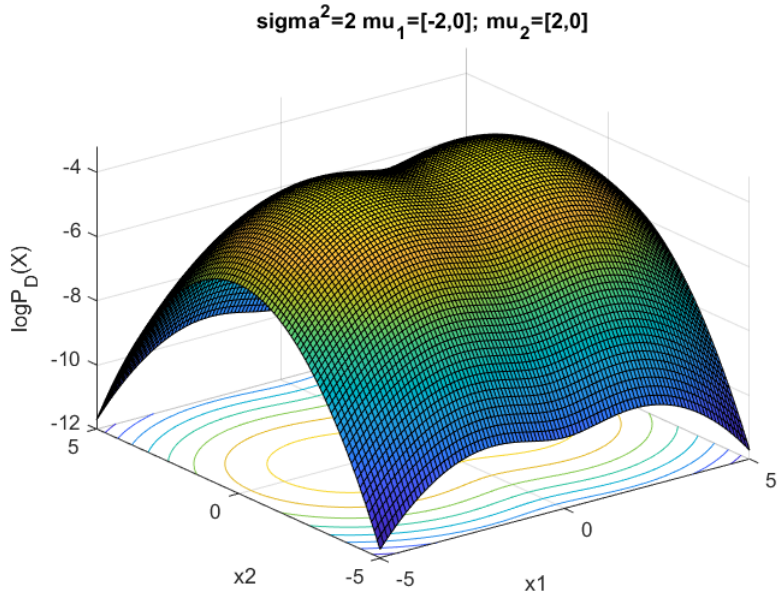
- plot the log-likelihood surface log Px(D) as a function of the mean parameter (entries for mu_1) for sigma^2={0.1, 1, 2}. Let the coordinate axis cover the range ([-5,5]).

  I chose the the mean parameter for mu_1=[-2,0] and mu_2=[2,0]


sigma^2=0.01 mu$_1$=[-2,0]; mu$_2$=[2,0]
logP$_D$(X)
0
-200
-400
-600
-800
x2
x1
5
0
-5


sigma^2=0.1 mu$_1$=[-2,0]; mu$_2$=[2,0]
logP$_D$(X)
-50
-100
-150
-200
x2
x1
5
0
-5


sigma^2=1 mu$_1$=[-2,0]; mu$_2$=[2,0]
logP$_D$(X)
-5
-10
-15
-20
x2
x1
5
0
-5



- What can you say about the local maxima of the likelihood surface and how it changes with sigma^2?

  With the fact of mu_1 = - mu_2 and equal weights on both component, thus we can see the local maxima of the likelihood surface for Px(x) is symmetric and with the same value of maximum value.

  When sigma is relatively small, we can clearly see there were two bell shapes. While the sigma increases, the two bell shapes seem to converge to almost one bel shape.

- How does the convergence to the optimal depend on the location of the initial parameter guess?

  If we started the initial parameter guess that are really close to the saddle point, then things might not work well. Therefore a good initialization is really important.

code:

```
%problem 2
clear
clc
%part a
x1=linspace(-5,5);
x2=x1;
[X1,X2]=meshgrid(x1,x2);
sigma=2;
cov=[sigma 0;0 sigma];
inverse=inv(cov);
constant=1/(2*pi*sqrt(det(cov)));
z1=constant*exp(-0.5*((X1-2).^2*inverse(1,1)+(X2).^2*inverse(2,2)));
z2=constant*exp(-0.5*((X1+2).^2*inverse(1,1)+(X2).^2*inverse(2,2)));
z=log(0.5*z1+0.5*z2);
figure(1)
surfc(X1,X2,z);
xlabel('x1');
ylabel('x2');
zlabel('logP_D(X)');
title(['sigma^2=',num2str(sigma),' mu_1=[-2,0]; mu_2=[2,0]']);
```

(b).

Starting from the initial parameter estimate

$$\mu_1^{(0)} = -\mu_2^{(0)} = (-0.1, 0)$$
$$\Sigma_1^{(0)} = \Sigma_2^{(0)} = I$$
$$\pi_1^{(0)} = \pi_2^{(0)} = \frac{1}{2}$$

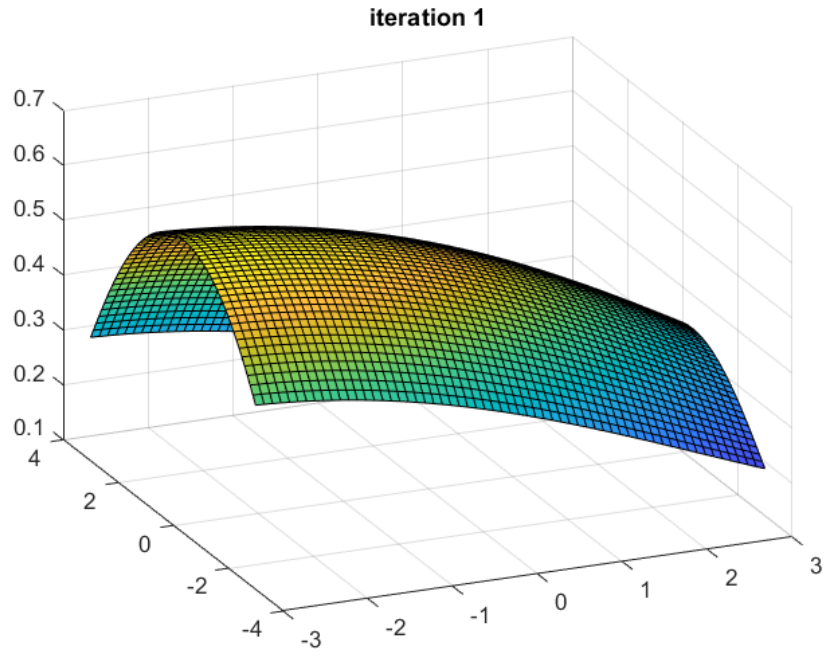
compute all the quantities involved in the first 3 iterations of the EM algorithm. For each iteration, produce

- plot 1: the posterior surface Pz|x(1|x) for the first class as a function of x
- plot 2: the mean of each Gaussian, the contour where the Mahalanobis distance associated with it becomes 1, the points in D, and the means of the solutions obtained the previous steps

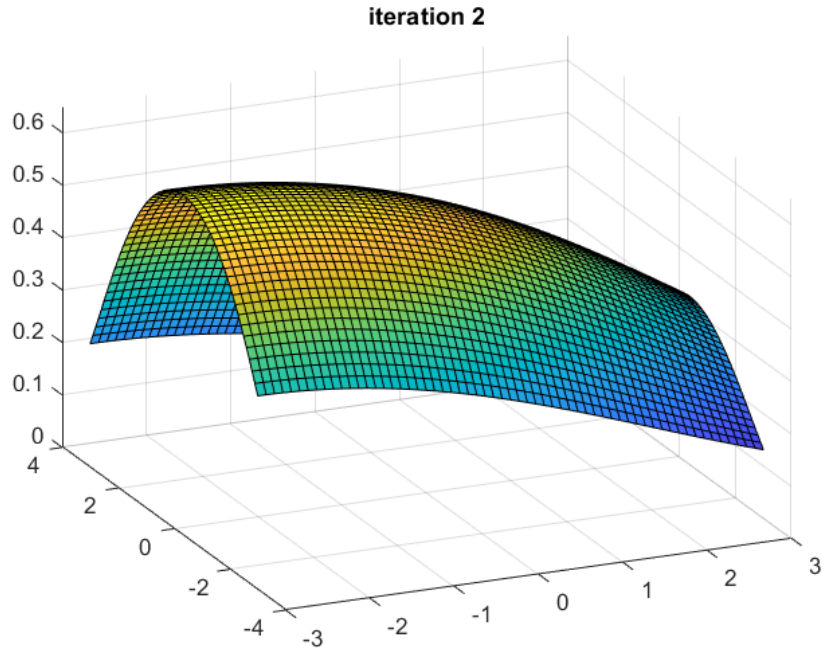
  Let EM run until convergence, storing the mean estimates at each iteration. Produce the two plots above for the final solution. In plot 2, plot the values of the mean as they progress from the initial to the final estimate.

plot 1:

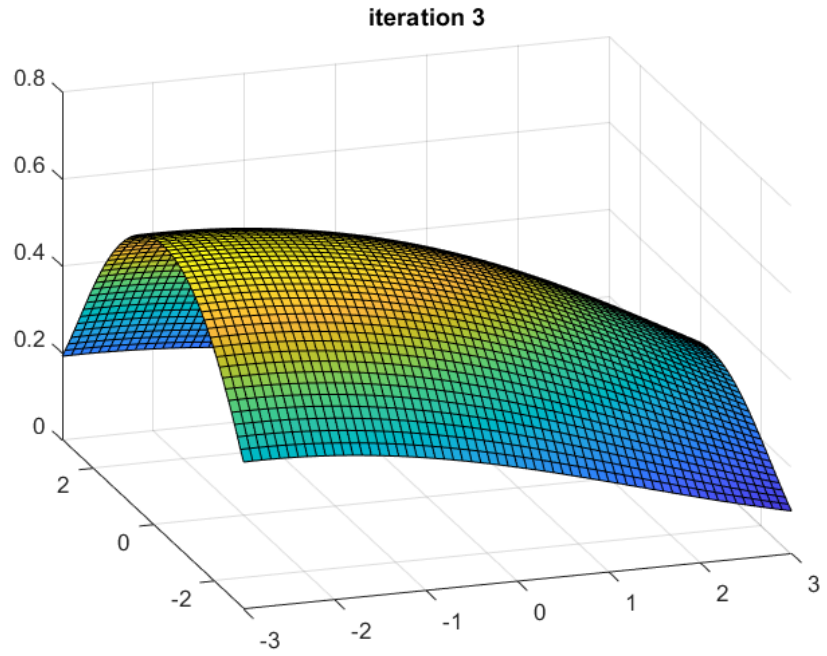
iteration 1:



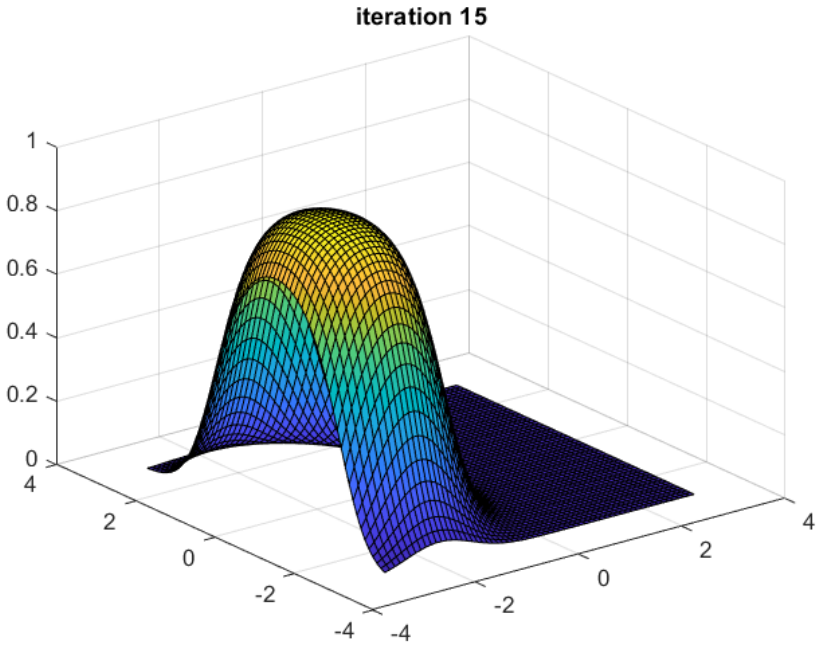
iteration 2:



iteration 3:

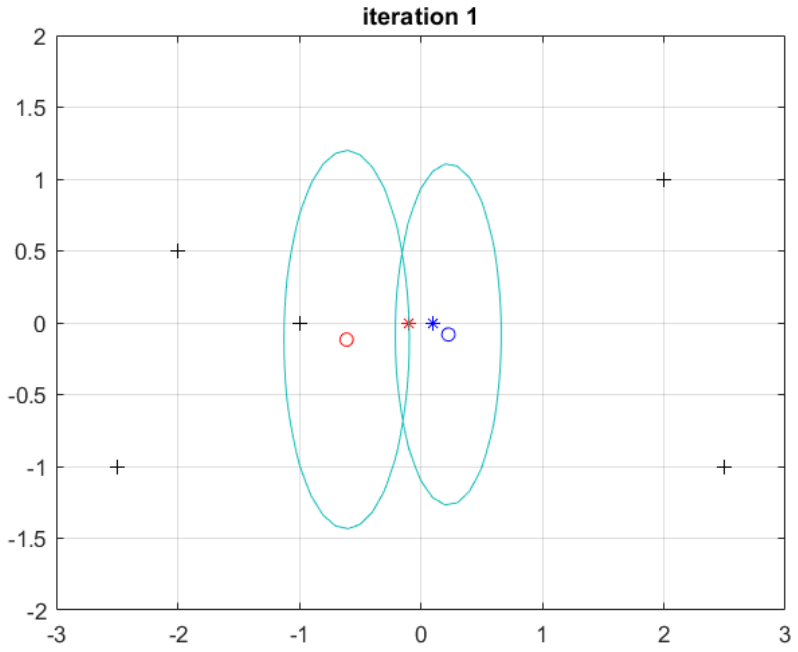


last iteration:

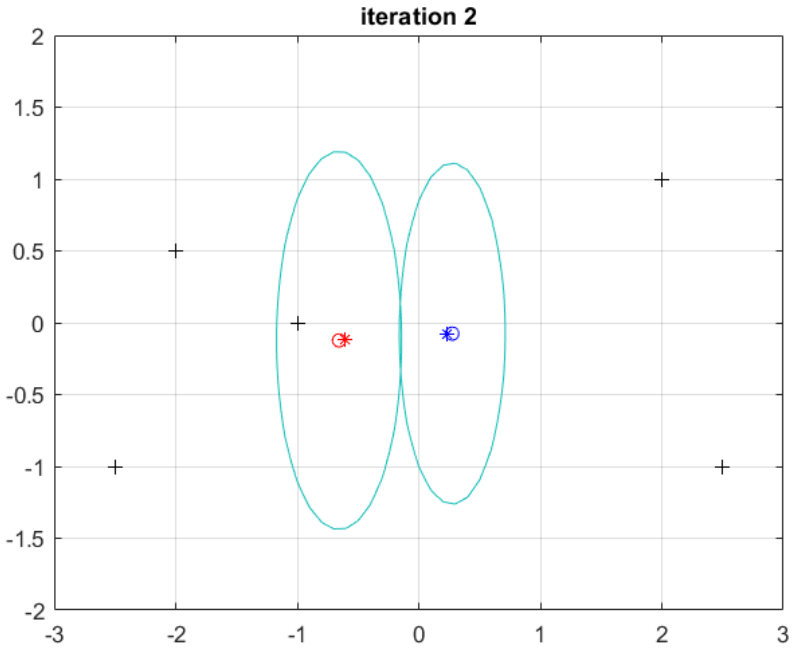


plot 2:

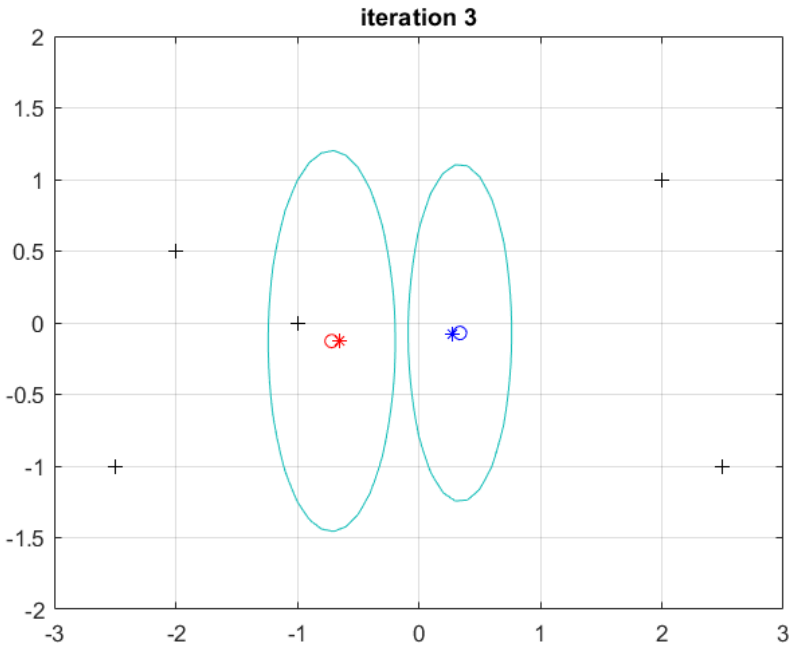
iteration 1:



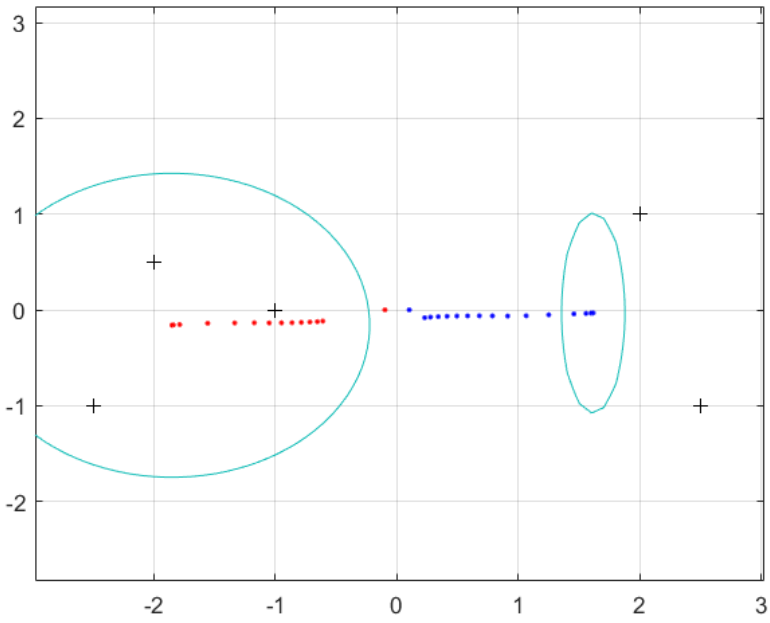
iteration 2:



iteration 3:



last iteration:



code:

```
%problem 2
%part b
clear
clc
%data
D=[-2.5,-1;-2,0.5;-1,0;2.5,-1;2,1];
figure
x1=D(:,1);
x2=D(:,2);
plot(x1,x2,'black+')
xlim([-3,3])
ylim([-2,2])
grid on
hold on
%initialize the parameters, mu,cov,pi
mu_1_updated=zeros(20,2);
mu_2_updated=zeros(20,2);
weight=[0.5,0.5];
mu_1=[-0.1,0];
mu_2=[0.1,0];
mu_1_updated(1,:)=mu_1;
mu_2_updated(1,:)=mu_2;
cov_1=[1,0;0,1];
cov_2=cov_1;
plot(mu_1(1),mu_1(2),'ro')
hold on
plot(mu_2(1),mu_2(2),'bo')
hold on
%E-step
h_kj=zeros(5,2);
n_j_new=zeros(1,2);

%5 data points
```

```
for i = 1:13
%while 1
    for k = 1:5
        pro_k1=weight(1)*gaussian_k(D,mu_1,cov_1,k);

 sum_pro_kj=weight(1)*gaussian_k(D,mu_1,cov_1,k)+weight(2)*gaussian_k(D,mu_2,cov
_2,k);
        h_k1=pro_k1/sum_pro_kj;
        h_kj(k,1)=h_k1;

        pro_k2=weight(2)*gaussian_k(D,mu_2,cov_2,k);

 sum_pro_kj=weight(1)*gaussian_k(D,mu_1,cov_1,k)+weight(2)*gaussian_k(D,mu_2,cov
_2,k);
        h_k2=pro_k2/sum_pro_kj;
        h_kj(k,2)=h_k2;

    end
    %%%%%M-step

    %calculate new n_j
    n_j_new(1,1)=sum(h_kj(:,1));
    n_j_new(1,2)=sum(h_kj(:,2));
    %calculate new mean
    mu_1=h_kj(:,1)'*D/n_j_new(1,1);
    mu_2=h_kj(:,2)'*D/n_j_new(1,2);
    mu_1_updated(i+1,:)=mu_1;
    mu_2_updated(i+1,:)=mu_2;
    dif= sqrt(sum((mu_2_updated(i+1,:)-mu_2_updated(i,:)).^2));
%     if dif<0.001
%         break
%     end
    %calculate new covariance
    %cov_1
    sigma1_11=0;
    sigma1_22=0;
    for index=1:5
        sigma1_11=sigma1_11+h_kj(index,1)*(D(index,1)-mu_1(1))^2;
        sigma1_22=sigma1_22+h_kj(index,1)*(D(index,2)-mu_1(2))^2;
    end
    cov_1=[sigma1_11,0;0,sigma1_22];
    cov_1=cov_1/n_j_new(1,1);

    %cov_2
    sigma2_11=0;
    sigma2_22=0;
    for index=1:5
        sigma2_11=sigma2_11+h_kj(index,2)*(D(index,1)-mu_1(1))^2;
        sigma2_22=sigma2_22+h_kj(index,2)*(D(index,2)-mu_1(2))^2;
    end
    cov_2=[sigma2_11,0;0,sigma2_22];
    cov_2=cov_2/n_j_new(1,2);

    %calculate new pi
    weight(1)=n_j_new(1,1)/5;
    weight(2)=n_j_new(1,2)/5;
```

```
%     plot(mu_1_updated(i+1,1),mu_1(i+1,2),'r.')
%     hold on
%     plot(mu_2_updated(i+1,1),mu_2_updated(i+1,2),'b.')
%     hold on

%     [fx1,fx2]=meshgrid(-3:0.1:3);
%     z=cov_1(1,1).*(fx1-mu_1_updated(i+1,1)).^2+cov_1(2,2).*(fx2-
mu_1_updated(i+1,2)).^2;
%     contour(fx1,fx2,z,[1,1]);
%     hold on
%     z=cov_2(1,1).*(fx1-mu_2_updated(i+1,1)).^2+cov_2(2,2).*(fx2-
mu_2_updated(i+1,2)).^2;
%     contour(fx1,fx2,z,[1,1])
%     title(['iteration',num2str(1)]);
    %mu_new=h_kj'*D./n_j_new';
end

syms x1 x2
constant_1=1/(2*pi*sqrt(det(cov_1)));
constant_2=1/(2*pi*sqrt(det(cov_2)));

%gaussian distribution
p_x1=weight(1)*constant_1*exp(-0.5.*([x1;x2]-mu_1')'*inv(cov_1)*([x1;x2]-
mu_1'));
p_x2=weight(2)*constant_2*exp(-0.5.*([x1;x2]-mu_2')'*inv(cov_2)*([x1;x2]-
mu_2'));

p=p_x1./(p_x1+p_x2);
[x1,x2]=meshgrid(-3:0.1:3);
z0=eval(p);
figure
surf(x1,x2,z0)
```

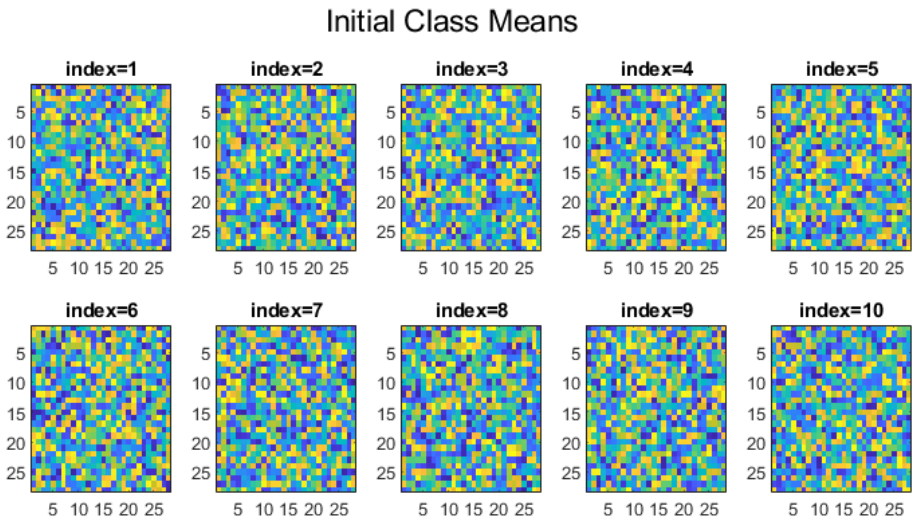
Problem 3.

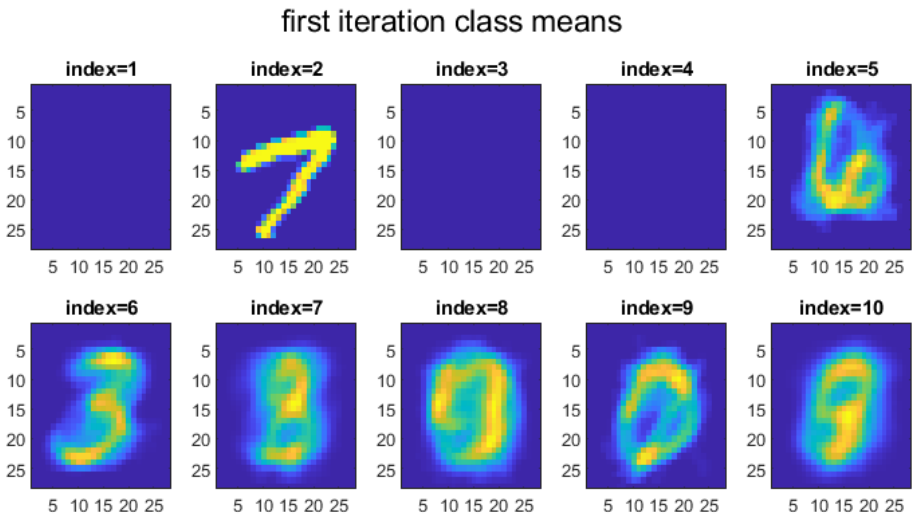
In this problem, we try to solve digit classification in an unsupervised manner, using K-means clustering. We assume that we only have the training images, without labels, and that there are 10 digit classes (as before). Hence, there are 10 clusters to learn. Each cluster has the same prior probability and gaussian distribution with identity covariances. Implement a K-means algorithm to learn the means of these clusters. A good stopping rule can be when the assignments of points to clusters do not change much in an iteration, say 0.2% (10 changes for a set of 5000 images).

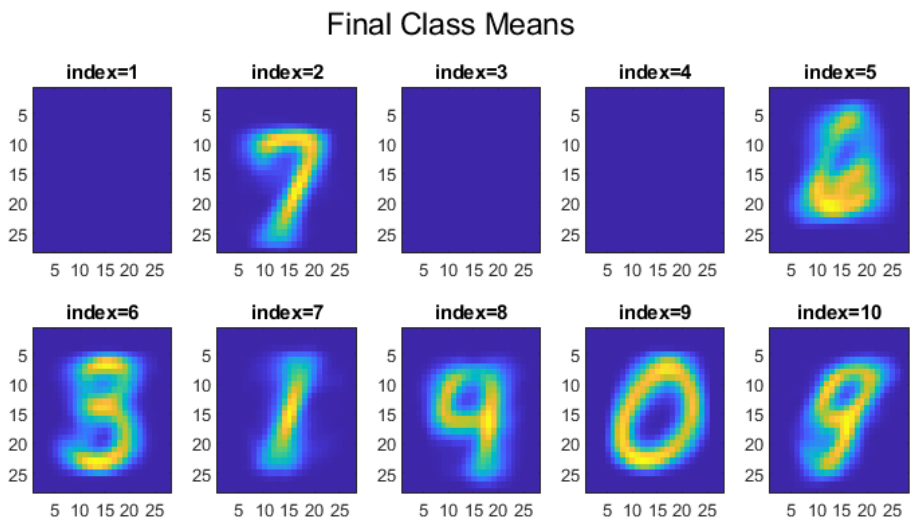
part 1.

First consider 10 random initializations suitably scaled to match the image intensity range. Run a K-means algorithm using these random initializations. Is there any problem that you encounter while running the algorithm? If yes, what is it and how can you tackle it (you need not implement the part of how to tackle it)? If not, submit the final class means as 28×28 images.

- 10 random image was initialized as the initial class mean. When I run K-means algorithm, I did encounter the problem illustrated below such that some of the class means became the empty clusters, which also led to the final class means appearing the empty clusters(I set the max iteration of 50 as the final class means). To tackle it, we could re-initialize the class means of those empty clusters.

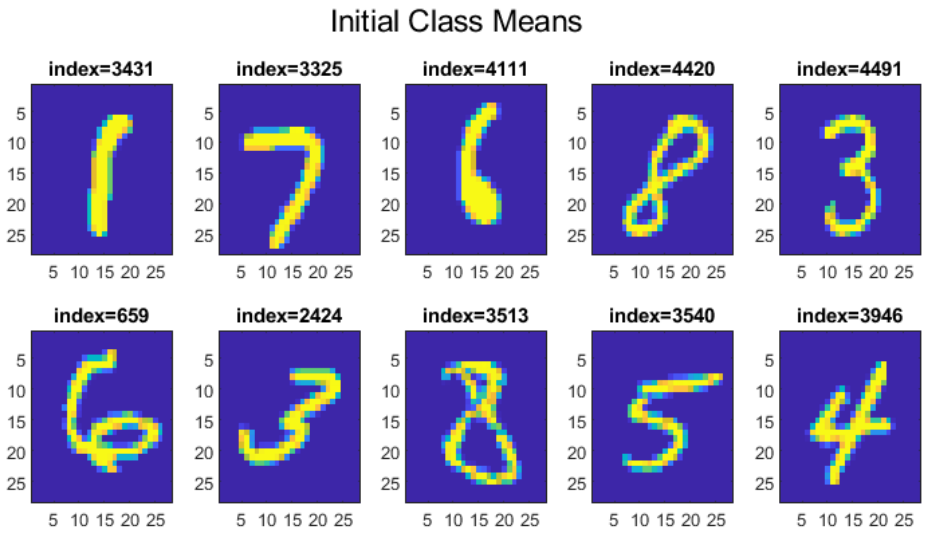


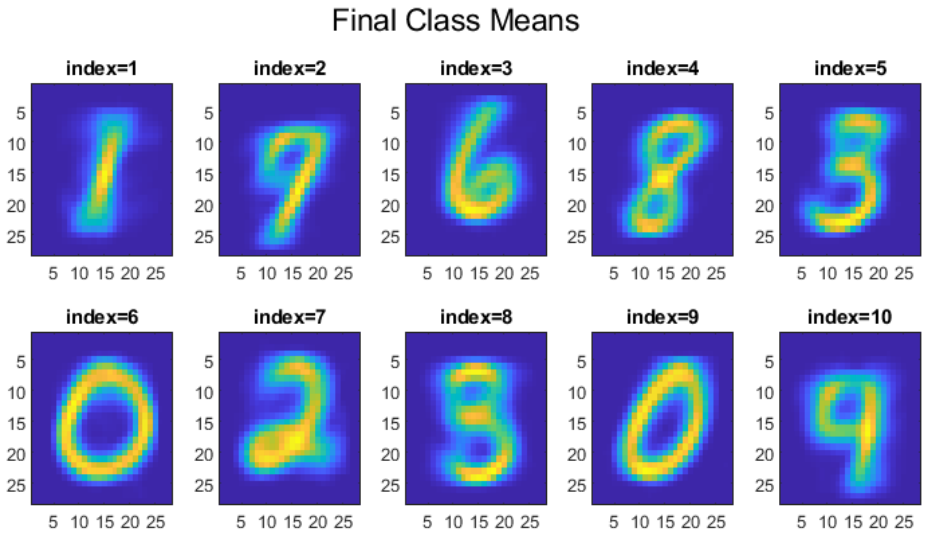




part 2.

Now instead of choosing random initializations for the class means, choose 10 random images from the training data itself and assume it to be the initial class means. Run the K-means algorithm and display the final class means as grayscale images. Also submit the image number of the random image chosen for initialization.

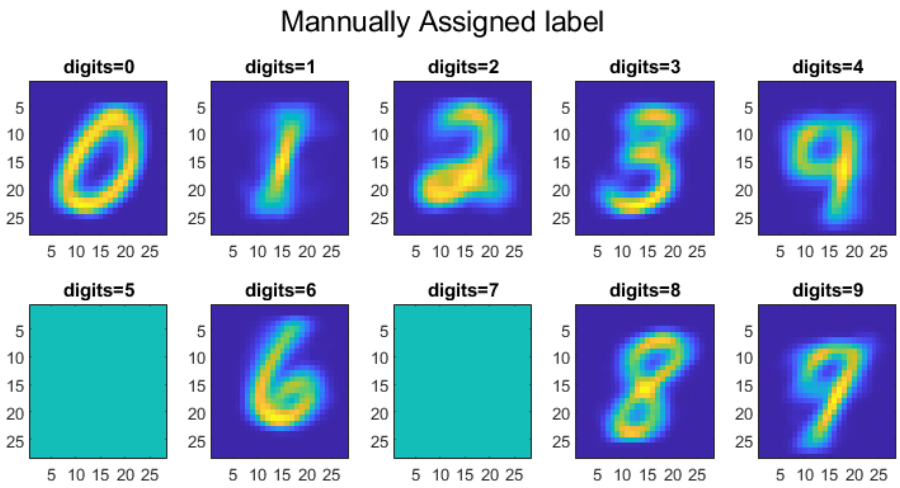




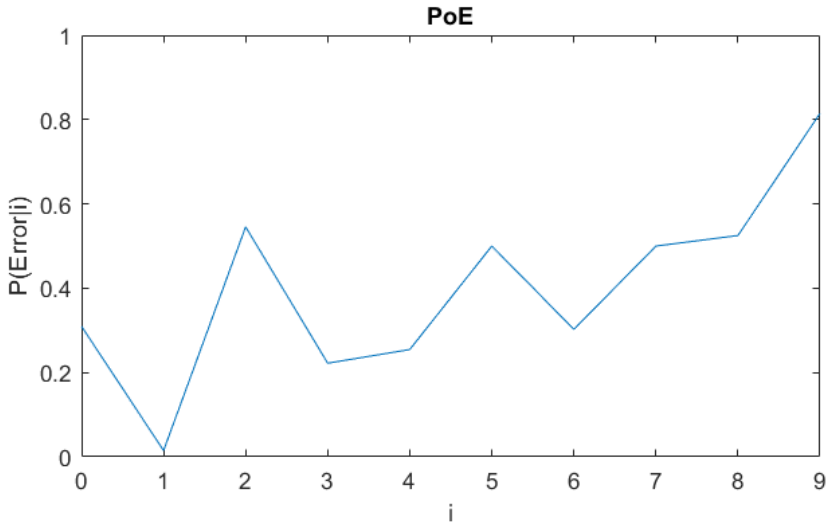
part 3.

Manually assign labels (0,1,2 ... 9) to the class means obtained in part 2. It is possible that some of the digit labels do not have a representation in the means obtained above, ignore those labels. Also some labels will have more than one representation, choose the one you feel the best. Now using these means perform a classification using gaussian classifier of HW3 on the test data. As before compute and display the error rates per class and total error rates. (For the digits that do not have a representation, consider the error rate to be the 50%)

- from the results of part 2 above, we obtained the class means for the class of 0,1,2,3,4,6,8,9. Digits 5 and 7 does not have representation in the final class means and digits 0 and 3 have two representations each. We manually assigned the labels for those digits that have representations and they are illustrated as below:



- by using those final class means, we conducted BDR to predict the labels for each image in testing data and the error rate per class was illustrated as below. Note that we manually assign the error rate for digits 5 and 7 to 0.5 since they do not have a representation in the final class means.



- the table of the error rate per class was demonstrated below:

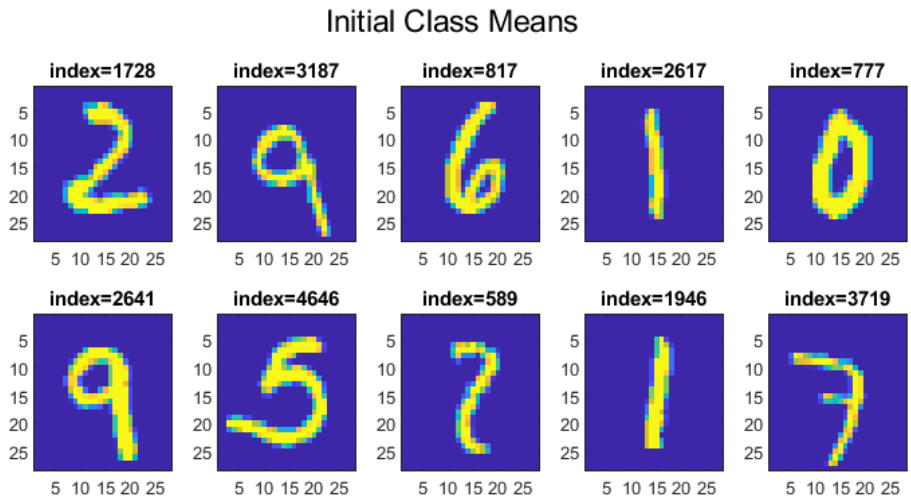
Class	PoE	# error images	# images
0	0.31	13	42
1	0.015	1	67
2	0.55	30	55
3	0.22	10	45
4	0.26	14	55
5	***0.5***	***48***	***50***
6	0.3	13	43
7	***0.5***	***49***	***49***
8	0.53	21	40
9	0.82	44	54

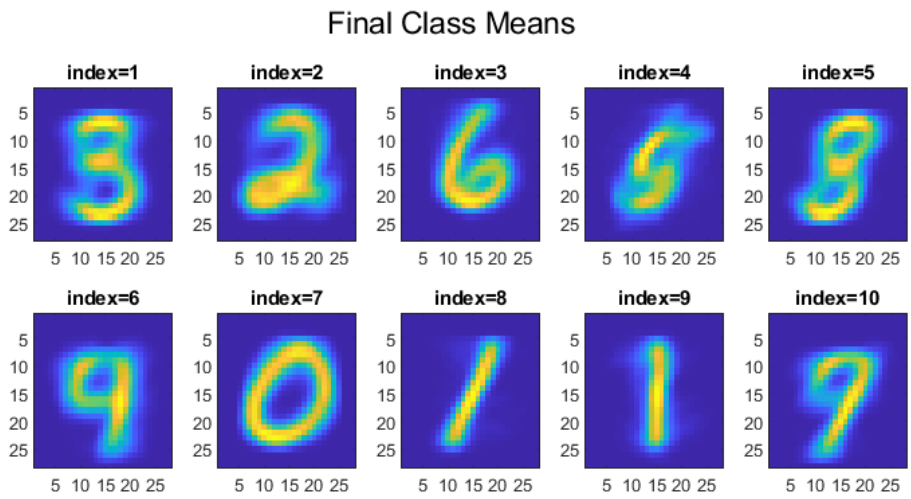
- the total error rate is **0.486**(which is obtained from 243/500).

part 4.

Repeat part 2 for another set of random images. Are these means different from the ones you obtained above? What can you say about the sensitivity to the initialization from the above experiment.

- after repeating part 2 for another set of random images, the final class means obtained in part 4 are different from the ones in part 2. Thus, the final class means of conducting K-means algorithm for this specific problem is sensitive to the initialization. From the intuitive ways, no matter how we assign the initial class means, the final class means after K-means algorithm should perform the same, but it just does not work for this specific problem. As you can see here, the final class means for both parts represented some repeated digits such as 0, 3 in part 2 and 1, 3 in part 4.





code:

```
%problem 3
clear
clc
data=load('HW5_Data\data.mat');
imageTest=data.imageTest;
imageTrain=data.imageTrain;
label=load('HW5_Data\label.mat');
labelTest=label.labelTest;
labelTrain=label.labelTrain;

%part 1&2&4

%choose those 10 random images as the initial class means
figure
sgtitle('Initial Class Means')
%initialize the class means
initial_means=zeros(28*28,10);

%%%part 1
%initialize random class means
% for i =1:10
%     single_initial_means=200*rand(28)+50;
%     subplot(2,5,i)
%     imagesc(single_initial_means)
%     title(['index=',num2str(i)])
%     %convert to (28*28)*10 vectors and store into initial_means
%     initial_means(:,i)=reshape(single_initial_means,[28*28,1]);
% end

%%%%%part 2&4
%generate random values from 1 to 5000
random_idx=round(10000*rand(10,1)/2);
%random_idx=[3431,3325,4111,4420,4491,659,2424,3513,3540,3946];
for i =1:10
    subplot(2,5,i)
    imagesc(imageTrain(:,:,random_idx(i)))
    title(['index=',num2str(random_idx(i))])
    %convert to (28*28)*10 vectors and store into initial_means
    initial_means(:,i)=reshape(imageTrain(:,:,random_idx(i)),[28*28,1]);
end
```

```
%convert imgaeTrain to the size of 784*5000
imageTrain_reshape=reshape(imageTrain,[784,5000]);
%K-means algorithm
image_means_distance=zeros(5000,10);
image_class=zeros(5000,1);
iteration=1;
%for iteration=1:30
while 1
    image_class_previous=image_class;
    for i =1:5000
        single_train_image=repmat(imageTrain_reshape(:,i),1,10);
        dif=single_train_image-initial_means;
        image_means_distance(i,:)=sqrt(sum(dif.^2));
        [~,index]=min(image_means_distance(i,:));
        image_class(i,1)=index;
    end
    %check the error
    count=find((image_class_previous==image_class)==0);
    count_size=size(count);
    count=count_size(1);
    if count<20
        break
    end
    for label=1:10
        x=find(image_class==label);
        y=imageTrain_reshape(:,x);
        initial_means(:,label)=mean(y,2);
    end

    iteration=iteration+1;
%     if iteration==1
%         initial_means_fisrt=reshape(initial_means,[28,28,10]);
%         figure
%         sgtitle('first iteration class means')
%         for first_iterate_i =1:10
%             subplot(2,5,first_iterate_i)
%             imagesc(initial_means_fisrt(:,:,first_iterate_i))
%             title(['index=',num2str(first_iterate_i)])
%         end
%     end
end

initial_means=reshape(initial_means,[28,28,10]);
figure
sgtitle('Final Class Means')
for i =1:10
    subplot(2,5,i)
    imagesc(initial_means(:,:,i))
    title(['index=',num2str(i)])
end



%part 3
assign_means=zeros(28,28,10);
assign_means(:,:,1)=initial_means(:,:,9);
assign_means(:,:,2)=initial_means(:,:,1);
```

```
assign_means(:,:,3)=initial_means(:,:,7);
assign_means(:,:,4)=initial_means(:,:,5);
assign_means(:,:,5)=initial_means(:,:,10);
assign_means(:,:,7)=initial_means(:,:,3);
assign_means(:,:,9)=initial_means(:,:,4);
assign_means(:,:,10)=initial_means(:,:,2);
figure
sgtitle('Mannually Assigned label')
for i =1:10
    subplot(2,5,i)
    imagesc(assign_means(:,:,i))
    title(['digits=',num2str(i-1)])
end
%initialization
stat_test=tabulate(labelTest);
i_x=zeros(1,10);
predict_label=zeros(500,1);
error_class=zeros(10,1);
assign_means=reshape(assign_means,[28*28,10]);
for i=1:500
    for j=0:9
        %reshape each image in testing image to the size of 784*1
        image=reshape(imageTest(:,:,i), [28*28,1]);
        %each class mean from class 0 to 9
        class_mean=assign_means(:,j+1);
        %calculate the value of i_x, each tesing image wrt 10 class sample
        %mean and restore to variable i_x
        i_x(1,j+1)=(image-class_mean)'*(image-class_mean);
        %find the minimum value and location within i_x for each testImage
        [val,loc]=min(i_x);
        %restore the value of class to variable predict_label(500*1)
        predict_label(i,1)=loc-1;
    end
    %calculate the error
    if labelTest(i) ~= predict_label(i)
        error_class(labelTest(i)+1)=error_class(labelTest(i)+1)+1;
    end
end
%calculate error rate for each class
error_rate_class=error_class./stat_test(:,2);
%plot error rate for each class
figure
x=(0:9);
plot(x,error_rate_class);
ylim([0,1])
xlabel('i')
ylabel('P(Error|i)')
title('PoE')
```