Assignment 2

DANDAN LIN

NUID:001093902

Question 1:

We can know that:

Class 0

$$m_{01} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad C_{01} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \qquad m_{02} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \quad C_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

$$w_1 = w_2 = \frac{1}{2}$$

$$P(L = 0) = 0.6$$

Class 1

$$m_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad C_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$P(L = 1) = 0.4$$

Plots for the datasets used are shown below in Figure 1.

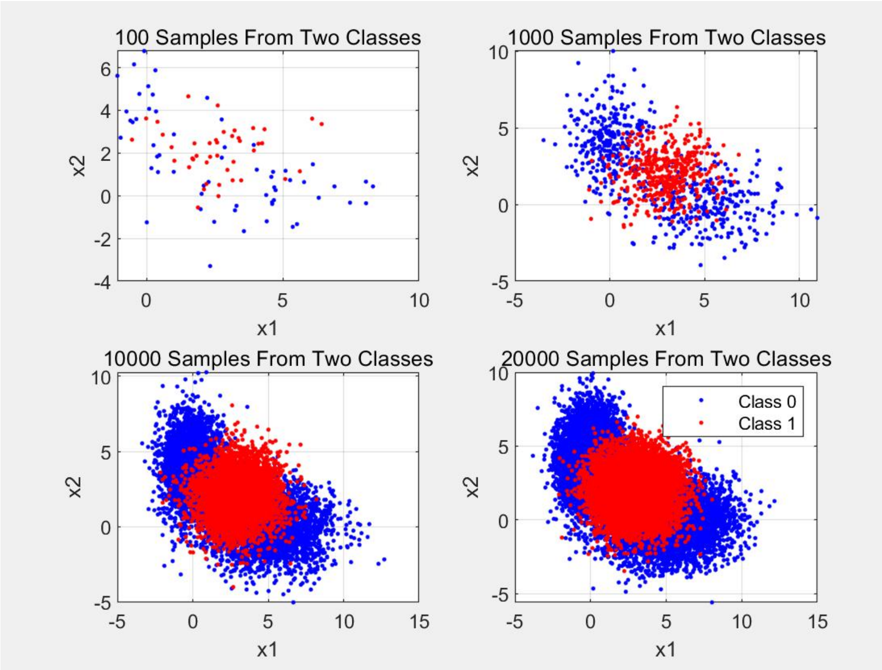

Figure 1 Problem 1 Training and Validation Data

Part 1: Theoretically optimal classifier with given datasets

The minimum expected risk classification rule:

$$(D=1)\left[\frac{g(x|L=1)}{g(x|L=0)}\right] \mathop{\gtrless} \left[\frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} \bullet \frac{P(L=0)}{P(L=1)}\right] = \gamma (D=0)$$

Where λ_{00} means the risk of true negative classification, λ_{01} means the risk of false negative classification, λ_{10} means the risk of false positive classification, λ_{11} means the risk of true positive classification. To minimize the probability of wrong classification in 0-1 loss function, the loss of false classification should be 1 and the loss of true classification should be 0, so the rule can be simplified into:

$$(D=1)[\frac{g(x|L=1)}{g(x|L=0)}] \mathop{\gtrless} [\frac{1-0}{1-0}] \bullet \frac{P(L=0)}{P(L=1)} = \gamma(D=0)$$

$$(D=1)[\frac{g(x|L=1)}{g(x|L=0)}] \mathop{\gtrless} [\frac{0.6}{0.4}] = 1.5 = \gamma(D=0)$$

The following figure 2 shows the ROC curve with the calculated ideal minimum error point as well as the minimum error point estimated from the generated validation data:

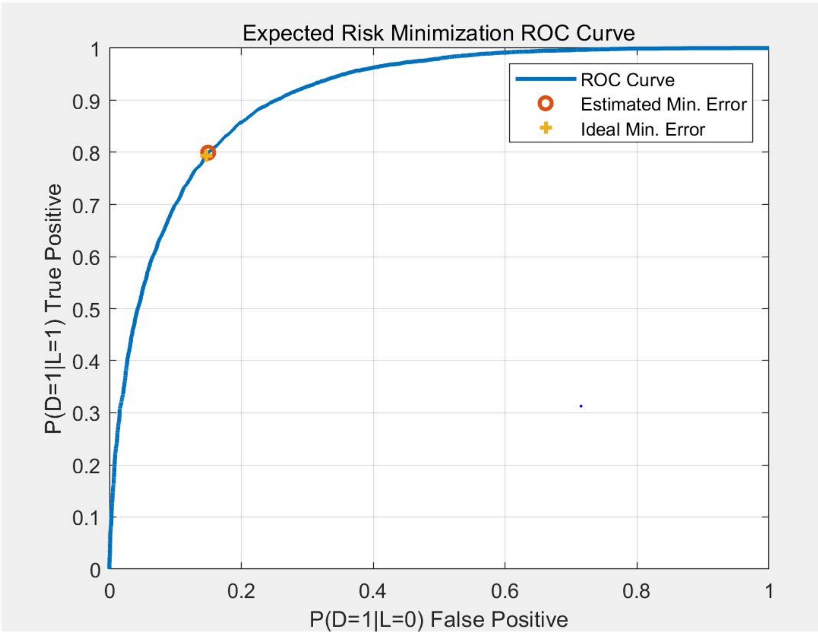

Figure 2   ROC Curve for Known Ideal Classification Case

The probability of error versus Gamma with the calculated and estimated minimum error points marked are shown in Figure 3.

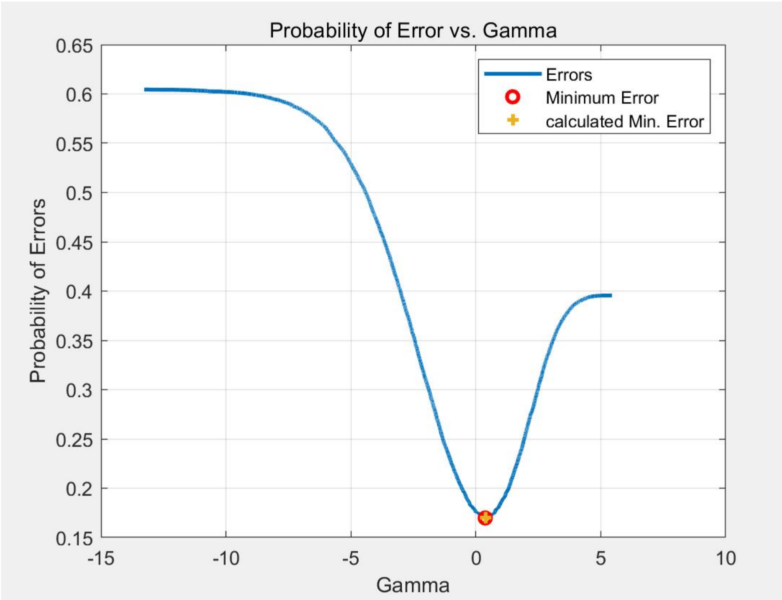

Figure 3 Probability of Error Curve for Ideal Classification

The below table shows minimum errors associated with the Theoretical calculated and estimated cases along with their associated threshold:

Table 1: Theoretical and Estimated MInimum Errors and threshold

	γ	min p_e
Theoretical	1.5	0.1746
From data	1.57	0.1744

Figure 4 shows the decision space for each distribution along with equilevel contours of the discriminant function.

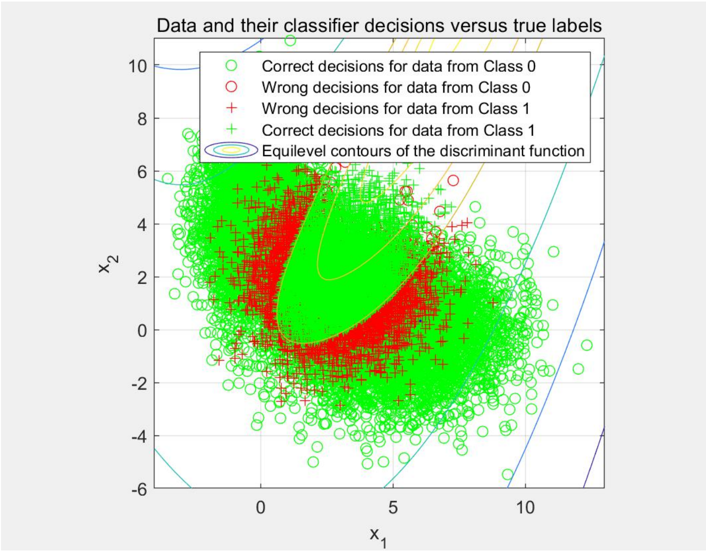

Figure 4: Decision Boundary of Ideal Classifier

Part 2:

For this part classification was performed with estimated knowledge of the underlying distributions of the data. Class 0 was modeled as a GMM with 2 components and Class 1 was modeled as a single Gaussian. Parameters were estimated using each of the 3 training sets of data and then the parameters estimates were used to classify the data.

In class 0,EM estimation is based on the Maximum Likelihood Estimation formulas.In the following formulas,it depends on class prior, mean and covariance of the conditional pdf. Also,these parameter estimation for Class 0 was performed using the built in Matlab function fitgmdist.

$$\arg\max Q(\theta,\theta^{g}) = \sum_{l=1}^{M}\sum_{i=1}^{N}\ln(\alpha_l)\,p(l|x_i\theta^{g}) + \sum_{l=1}^{M}\sum_{i=1}^{N}p(l|x_i\theta^{g})\ln\left(p_l(x_i|\theta_l)\right)$$

As we known,the α , μ and Σ :

$$\widehat{\alpha}_l = \frac{1}{N}\sum_{i=1}^{N}P_l(x_i|x_i,\theta^{g})$$

$$\widehat{u}_l = \frac{\sum_{i=1}^{N}x_i p(l|x_i\theta)}{\sum_{i=1}^{N}p\left(l|x_i\theta\right)}$$

$$\widehat{\Sigma}_l = \frac{\sum_{i=1}^{N}p(l|x_i\theta^{g})(x_i-u_l)(x_i-u_l)^{T}}{\sum_{i=1}^{N}p(l|x_i\theta^{g})}$$

In class 1,since there is only a single Gaussian the maximum likelihood, estimates are the sample average and covariance.The following formulas is for MLE of the model parameters.and parameter estimation for Class 1 was also performed using the fitgmdist Matlab function.

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N}x_i$$

$$\widehat{\Sigma} = \frac{1}{N}(x_i-\widehat{\mu})(x_i-\widehat{\mu})^{T}$$

Table 2, Table 3, and Table 4 show the parameter estimates obtained from analysis of each of sets of training data.

Table 2: Estimated Means

	$D^{train}{}_{100}$	$D^{train}{}_{1000}$	$D^{train}{}_{10000}$
$\widehat{m}_{01}$	$\begin{bmatrix} 4.22 \\ 0.36 \end{bmatrix}$	$\begin{bmatrix} 5.04 \\ 0.04 \end{bmatrix}$	$\begin{bmatrix} 4.96 \\ -0.01 \end{bmatrix}$
$\widehat{m}_{02}$	$\begin{bmatrix} -0.07 \\ 3.78 \end{bmatrix}$	$\begin{bmatrix} 0.02 \\ 4.10 \end{bmatrix}$	$\begin{bmatrix} 0.02 \\ 4.02 \end{bmatrix}$
$\widehat{m}_{10}$	$\begin{bmatrix} 2.71 \\ 2.11 \end{bmatrix}$	$\begin{bmatrix} 2.93 \\ 2.05 \end{bmatrix}$	$\begin{bmatrix} 2.98 \\ 1.99 \end{bmatrix}$

Table 3: Estimated Covariances

	D^{100}_{train}	D^{1000}_{train}	D^{10k}_{train}
$\widehat{C}_{01}$	$\begin{bmatrix} 4.01 & -0.34 \\ -0.34 & 2.13 \end{bmatrix}$	$\begin{bmatrix} 3.68 & -0.06 \\ -0.06 & 1.97 \end{bmatrix}$	$\begin{bmatrix} 4.02 & -0.01 \\ -0.01 & 2.00 \end{bmatrix}$
$\widehat{C}_{02}$	$\begin{bmatrix} 0.28 & -0.30 \\ -0.30 & 2.85 \end{bmatrix}$	$\begin{bmatrix} 1.13 & -0.03 \\ -0.03 & 3.11 \end{bmatrix}$	$\begin{bmatrix} 1.02 & -0.05 \\ -0.05 & 2.91 \end{bmatrix}$
$\widehat{C}_{10}$	$\begin{bmatrix} 2.24 & -0.003 \\ -0.003 & 1.30 \end{bmatrix}$	$\begin{bmatrix} 1.98 & 0.16 \\ 0.16 & 2.03 \end{bmatrix}$	$\begin{bmatrix} 1.99 & 0.01 \\ 0.01 & 2.07 \end{bmatrix}$

Table 4: Estimated Alphas for Class 0 GMM

	D^{100}_{train}	D^{1000}_{train}	D^{10k}_{train}
$\widehat{\alpha}_{01}$	0.66	0.501	0.503
$\widehat{\alpha}_{02}$	0.34	0.499	0.497

Table 5: Sample Class Priors

	D^{100}_{train}	D^{1000}_{train}	D^{10k}_{train}
$\widehat{P}(L=0)$	0.58	0.59	0.60
$\widehat{P}(L=1)$	0.42	0.41	0.40

Figure 5(for D_{train}^{100}), Figure 6(for D_{train}^{1000}), and Figure 7(for D_{train}^{10k}) below show the resulting estimated distributions for the Class 0 GMM .

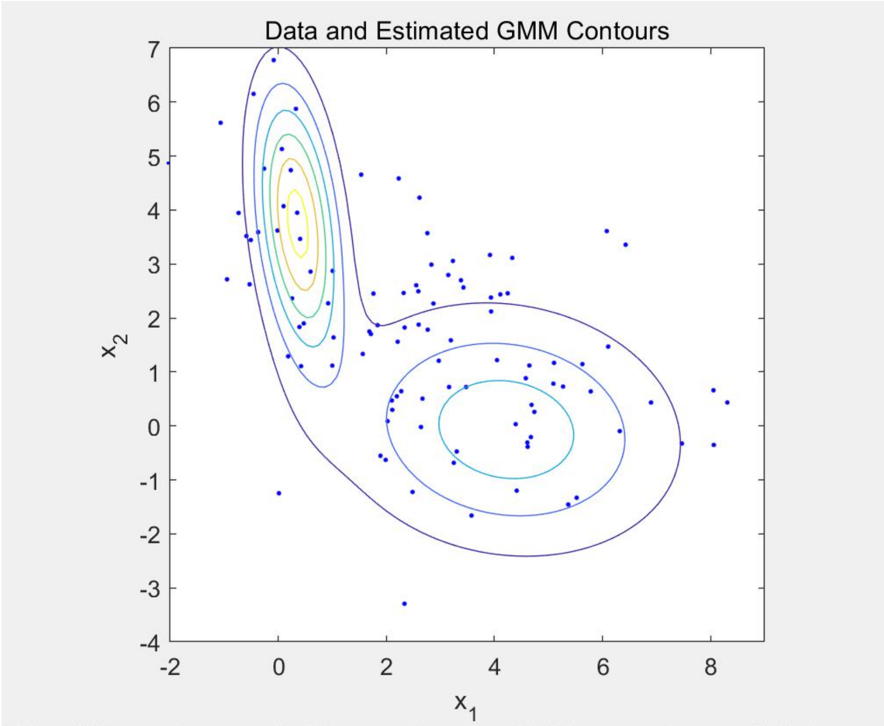

Figure 5: Contour Plot of Estimated Distributions for Class 0 GMM for D100 Training Data

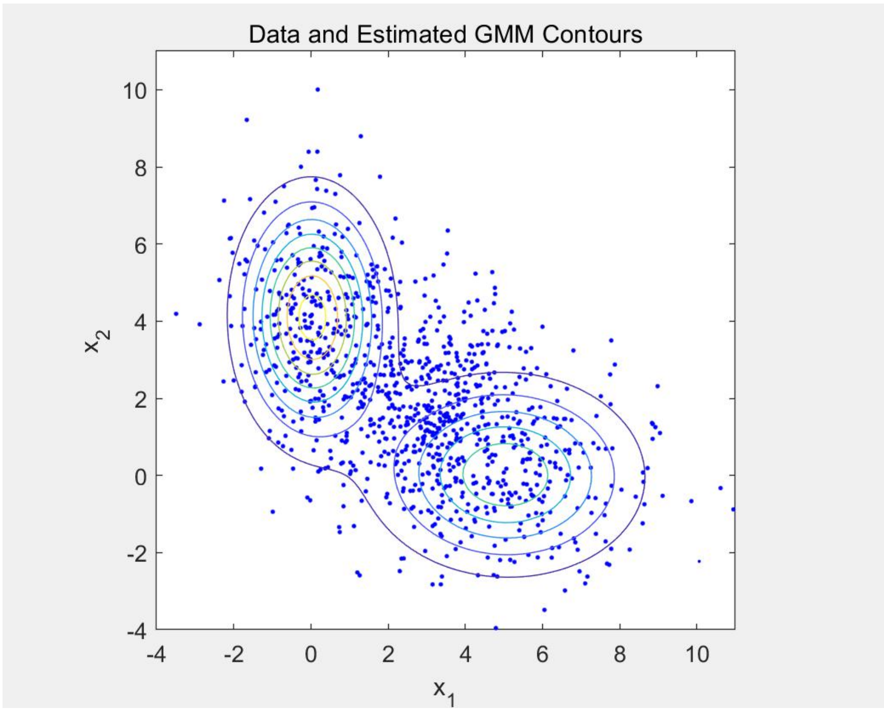

Figure 6: Contour Plot of Estimated Distributions for Class 0 GMM for D1000 Training Data

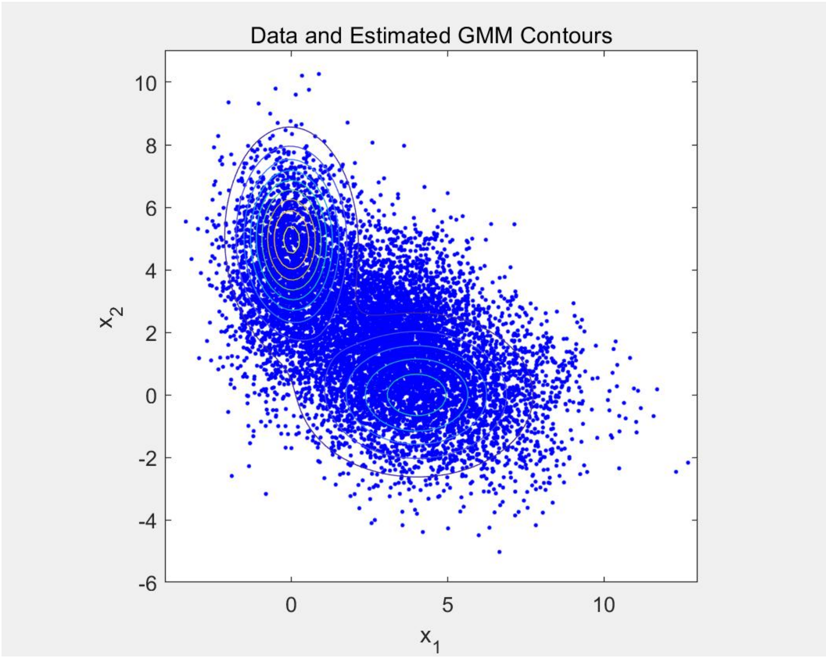

Figure 7: Contour Plot of Estimated Distributions for Class 0 GMM for D10K Training Data

Here is a summary of the minimum estimated probability of errors associated with the parameters estimated from the three training datasets which shown in Table 5. Overall, The minimum probability of error decrease as the number of training samples increases.

Table 5: Minimum Probability of Error for all 3 Training Datasets

Training Samples	$\lambda_{\mathrm{minErr}}$	min P_e
100	1.44	0.1728
1000	1.72	0.1737
10K	1.63	0.1732
Known PDF(Part 1)	1.5	0.1746

Perhaps because of the randomness of the data, the probability of error fluctuates a bit.But we still can find that the minimum probability of error decrease as the number of training samples increases, which is the same as theoretical trends.

The ROC curves for training data is shown in Figure 8.

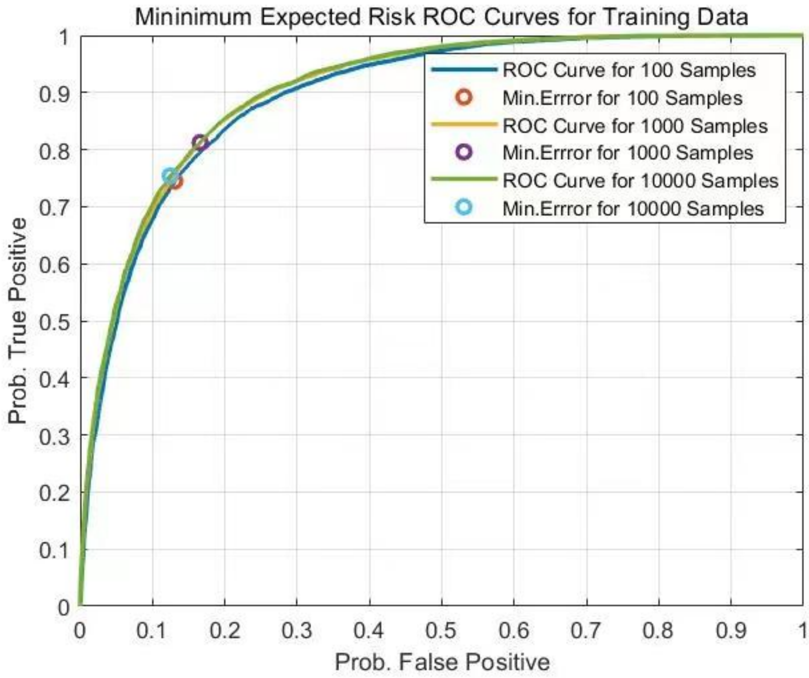

Figure 8: ROC Curves for All Sets of Training Data

Part 3:

In part3 maximum likelihood parameter estimation techniques were used to train logistic-linear-function-based and logistic-quadratic-function-based approximation of class label posterior functions given a sample. This training was performed on each of training data sets which consists of 100, 1000, and 10000 samples Respectively.Then it was used to classify samples from the 20000 sample validation data set.

The logistic function is defined as below:

$$h(x, w) = \frac{1}{1 + e^{W^T z(x)}}$$

The linear logistic function:

$$z(x) = [1 \quad x_1 \quad x_2]^T$$

The quadratic logistic function:

$$z(x) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_1 \bullet x_2 \quad x_2^2]^T$$

The cost function for estimating w vectors:

$$\hat{\theta}_{ML} = -\frac{1}{N}\sum_{1}^{N} l_n \ln(h(x_n, \theta)) + (1 - l_n)\ln(1 - h(x_n, \theta))$$

The minimum expected risk classification criteria:

$$(l_n = 1)\hat{w}^T z(x) \underset{<}{\overset{>}{}} 0(l_n = 0)$$

(a)

The validation dataset was classified with each of the three different approximations based on the decision rule. Plots of the correct vs incorrect classification for the logistic linear classifier trained on100, 1000, and 10000 samples are shown in Figures 9, 10, and 11. Green and red points indicate whether the point was classified correctly, and the black lines on the plots is the decision boundary.

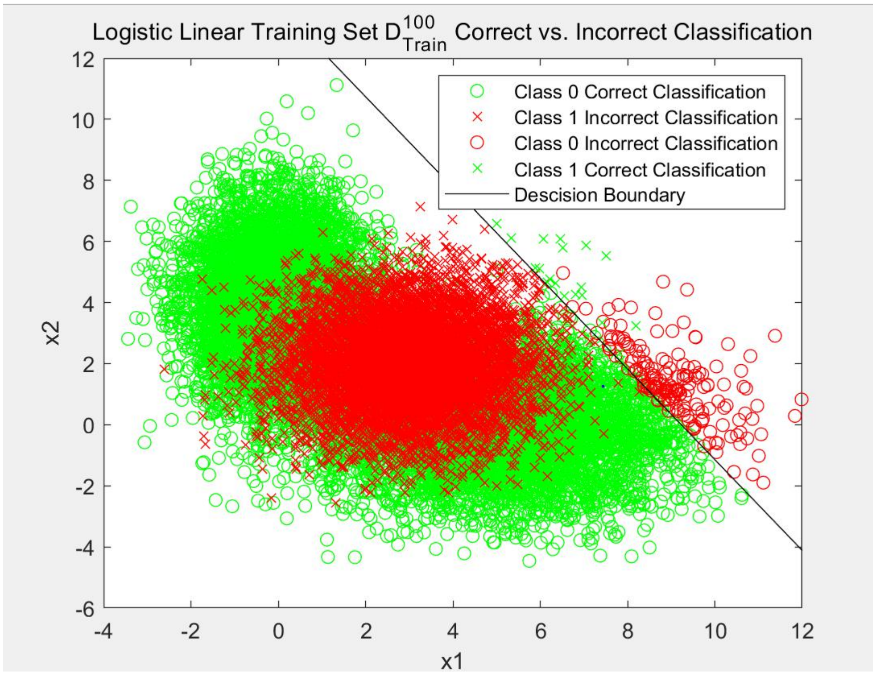

Figure 9: Classifier for Linear Logistic Fit on D100 Training Data

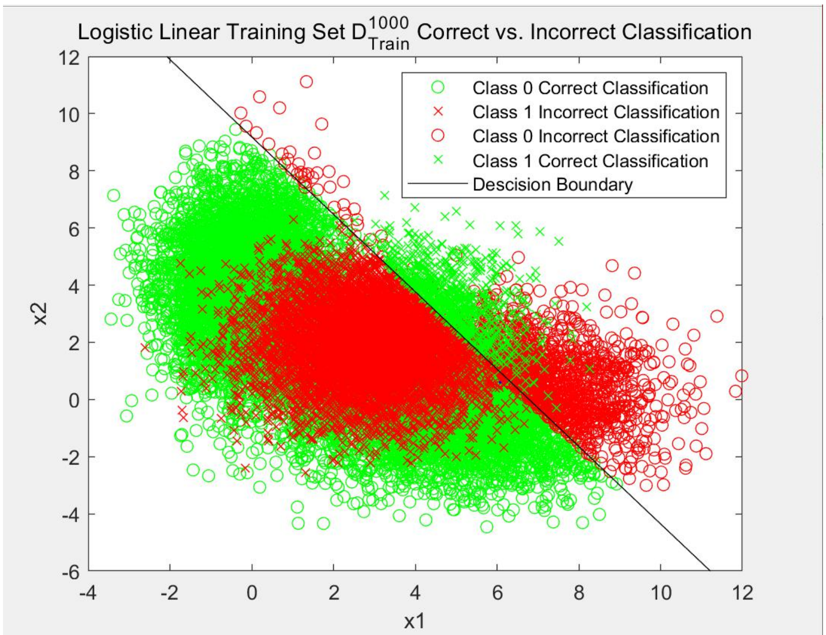

Figure 10: Classifier for Linear Logistic Fit on D1000 Training Data

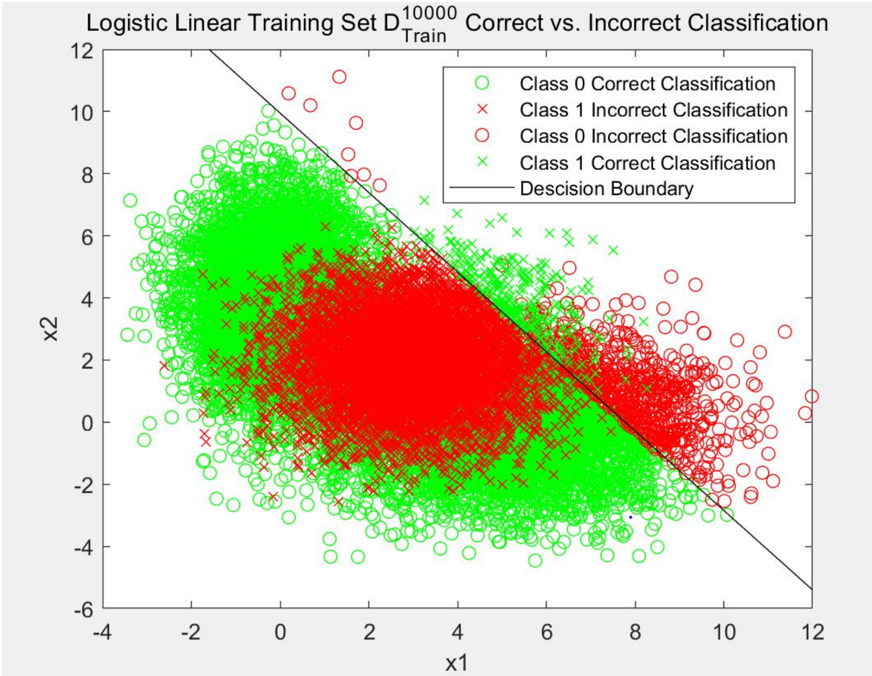

Figure 11: Classifier for Linear Logistic Fit on D10000 Training Data

(b)

Plots of the correct vs incorrect classification for the logistic quadratic classifier trained on100, 1000, and 10000 samples are shown in Figures 12, 13, and 14. Green and red points indicate whether the point was classified correctly, and the black lines on the plots is the decision boundary.

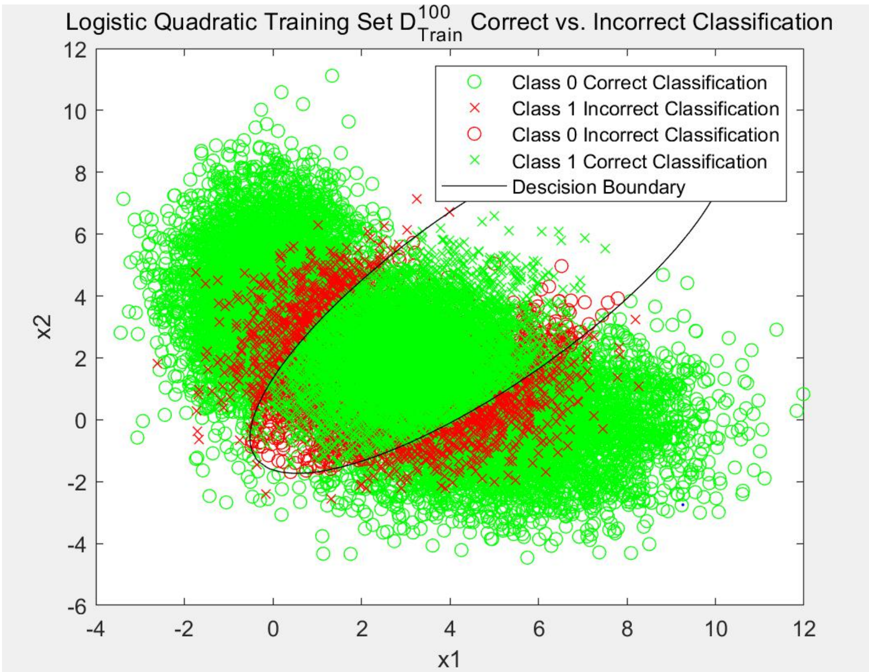

Figure 12: Classifier for Quadratic Logistic Fit on D100 Training Data

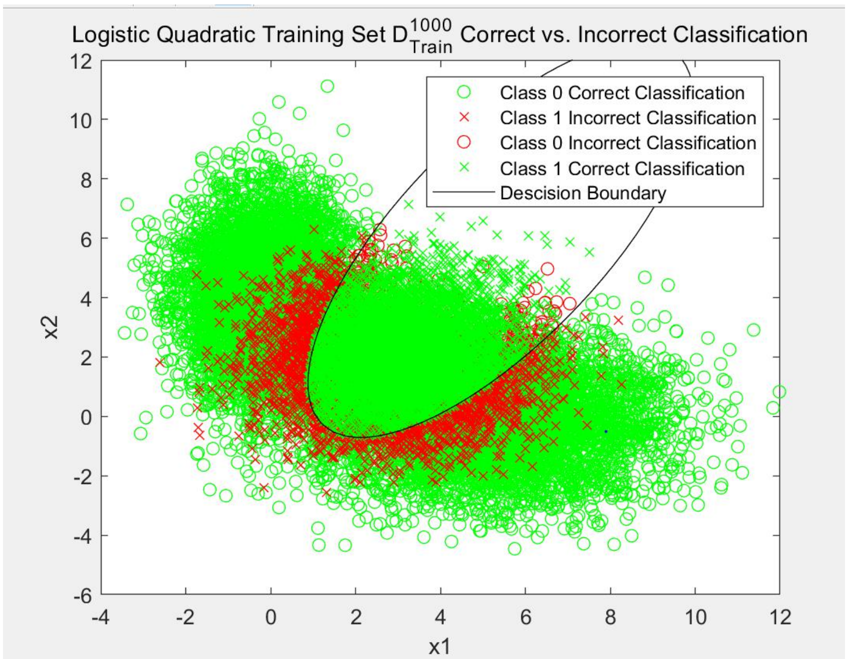

Figure 13: Classifier for Quadratic Logistic Fit on D1000 Training Data

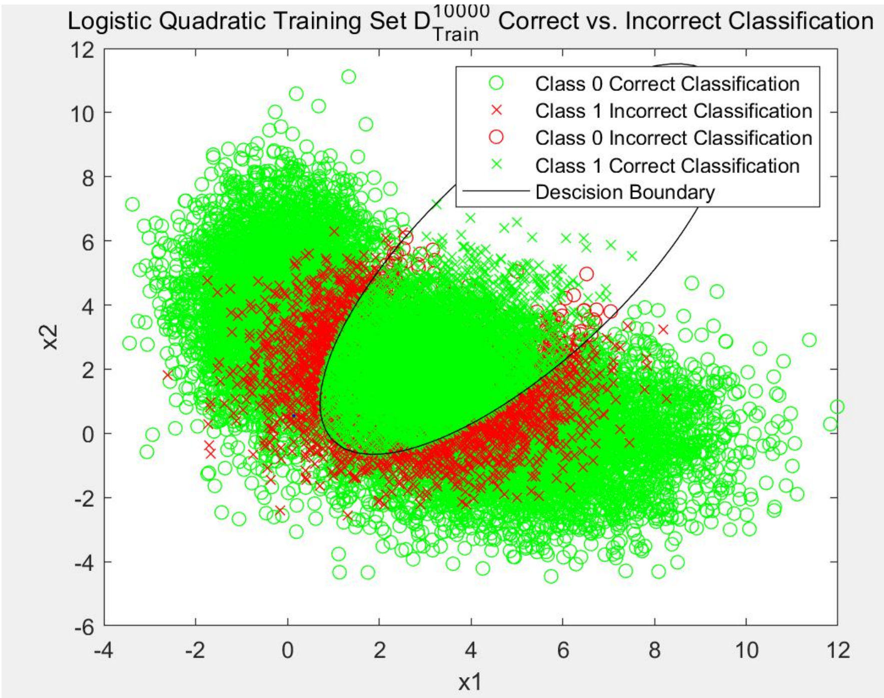

Figure 14: Classifier for Quadratic Logistic Fit on D10000 Training Data

As shown in table 6, there is a summary of the resulting probability of errors from classifying the 20000 sample validation data set using each of the 3 training data sets. The data shows that for both the linear and quadratic estimation functions the probabilities of error decrease when the number of points in the training datasets increase. Also, we can find the quadratic logistic function is better than the linear logistic function in all cases and for the 10000 sample training data set even approached the theoretical optimal probability of error of 0.174 obtained in Part 1.

Table 6：Logistic Function Probabilities of Error

Training Dataset	Linear	Quadratic

100	**0.47**	**0.20**
1000	**0.423**	**0.18**
10000	**0.428**	**0.178**

Perhaps because of the randomness of the data, the probability of error fluctuates a bit.However, we can still see the general trend that the probabilities of error decrease when the number of points in the training datasets increase and the quadratic logistic function is better.

Question 2:

The objective is to find the $[x, y]^T$ coordinate position with the highest probability given the prior distribution as well as the range measurements from each of the K reference coordinates.

$$\begin{bmatrix} x_{MAP} \\ y_{MAP} \end{bmatrix} = \arg\max_{\begin{bmatrix} x \\ y \end{bmatrix}} p\left(\begin{bmatrix} x \\ y \end{bmatrix} \middle| \{r_1 \cdots r_k\}\right) \tag{1}$$

$$= \arg\max((2\pi\sigma_x\sigma_y)^{-1} e^{-\frac{1}{2}[x \quad y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}}) \prod_{i=1}^{K} p(\begin{bmatrix} x \\ y \end{bmatrix} | r_i) \tag{2}$$

$$= \arg\max_{\begin{bmatrix} x \\ y \end{bmatrix}} \ln((2\pi\sigma_x\sigma_y)^{-1}) + \ln(e^{-\frac{1}{2}[x \quad y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}}) + \sum_{i=1}^{K} \ln p\left(\begin{bmatrix} x \\ y \end{bmatrix} \middle| r_i\right) \tag{3}$$

$$= \arg\max_{\begin{bmatrix} x \\ y \end{bmatrix}} -\frac{1}{2}[x \quad y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \ln \mathrm{N}(n_i | 0, \sigma_i^2) \tag{4}$$

$$= \arg\max -\frac{1}{2}[x \quad y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \ln((2\pi\sigma_i^2)^{-\frac{1}{2}} e^{-\frac{((r_i - d_i) - 0)^2}{2\sigma_i^2}}) \tag{5}$$

$$= \arg\max -\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x{}^2 & 0 \\ 0 & \sigma_y{}^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \ln((2\pi\sigma_i{}^2)^{-\frac{1}{2}}) + \ln(e^{-\frac{(r_i - d_i)^2}{2\sigma_i{}^2}}) \quad (6)$$

$$= \arg\max -\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x{}^2 & 0 \\ 0 & \sigma_y{}^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} -\frac{(r_i - d_i)^2}{2\sigma_i{}^2} \quad (7)$$

$$= \arg\min \begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x{}^2 & 0 \\ 0 & \sigma_y{}^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \frac{(r_i - d_i)^2}{\sigma_i{}^2} \quad (8)$$

$$= \arg\min \begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x{}^2 & 0 \\ 0 & \sigma_y{}^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{K} \frac{\left(r_i - \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right\|\right)^2}{\sigma_i{}^2} \quad (9)$$

$$= \arg\min \left[\frac{x^2}{\sigma_x{}^2} + \frac{y^2}{\sigma_y{}^2} \right] + \sum_{i=1}^{K} \frac{\left(r_i - \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right\|\right)^2}{\sigma_i{}^2} \quad (10)$$

Using $\sigma_x = \sigma_y = 0.25$,and $\sigma_i = 0.3$

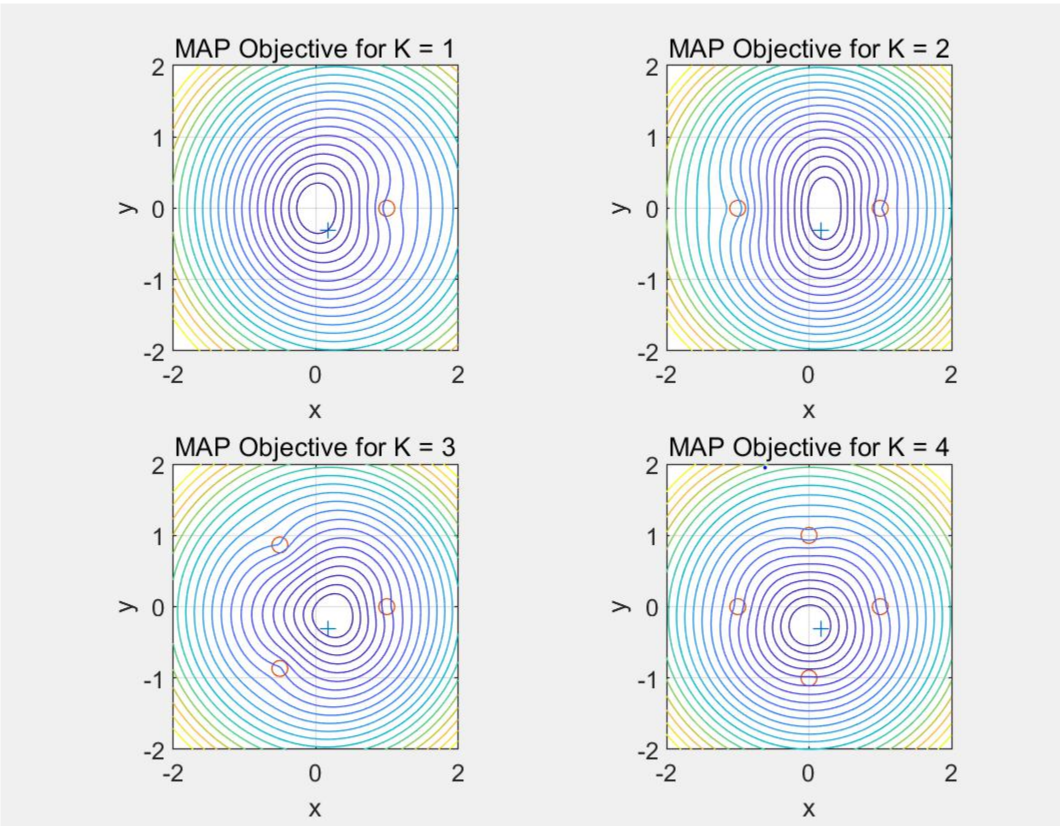

Figure : MAP Objective for K=1,2,3,4

The code generates a random coordinate pair within true position inside unit circle. Then, for each value of K,the code lets the landmark positions evenly spaces on the unit circle, generates range measurements and evaluates the MAP estimation objective function on a grid.Finally,it displays true position and landmark positions and the MAP objective contours for the range of horizontal and vertical coordinates from -2 to 2 .Additionally, it plots the true location (shown as a blue '+')the landmark locations (shown as red circles).

For K < 3,the MAP estimate of position is not very accurate, where all estimates are symmetric around the x-axis because all landmarks and the prior bias have a y-coordinate of 0. However, for K = 3 and K = 4, the estimator is much more accurate.This can be seen from the above four graph, in which the true location lies more closer to central estimate contour, respectively. In general, the MAP estimate gets more accurate when K increases. While this is not always true, since the estimator's accuracy can be also determined from the contour graph based on the distance from the true location to the point with the lowest contour, roughly around the center of the innermost contour.

In conclusion, as K increases, the certainty of the estimator increases,which is more visible on the figure above. A shrinkage of the area of locations with a high probability on the figure could represent the certainty of the estimator.If the value of K gets a larger number,we can see this phenomenon more clearly.

Question 3:

If we choose the ω_i from which we can get our average risk at a point x is:

$$R(\omega_i|x) = \sum_{j=1}^{c} \lambda(\omega_i|\omega_j)P(\omega_j|x) = 0 \times P(\omega_i|x) + \sum_{j=1, j\neq i}^{c} \lambda_s P(\omega_j|x)$$

the cost of choosing class ω_i:

$$\lambda(\omega_i|\omega_j)$$

Where the truth class is ω_j.

Hence:

$$R(\omega_i|x) = \lambda_s(1 - P(\omega_i|x))$$

Associate x with the ω_i if the max posterior class probability and the average risk is less than the cost of rejection:

$$\lambda_s(1 - P(\omega_i|x)) \leq \lambda_r$$

$$P(\omega_i|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$$

Appendix:

Q1:

```
%%=========================Question1=========================
%%

% Dandan lin/001093902

% Code help and example from Prof.Deniz

clear all;

close all;

%Switches to bypass parts 1 and 2 for debugging

Part1=1;Part2=1;

dimension=2; %Dimension of data

%Define data

D.d100.N=100;

D.d1000.N=1000;

D.d10k.N=10000;

D.d20k.N=20000;

dTypes=fieldnames(D);

%Define Statistics

p=[0.6 0.4]; %Prior

%Label 0 GMM Stats

mu0=[5 0;0 4]';

Sigma0(:,:,1)=[4 0;0 2];

Sigma0(:,:,2)=[1 0;0 3];

alpha0=[0.5 0.5];

%Label 1 Single Gaussian Stats

mu1=[3 2]';
```

```
Sigma1=[2 0;0 2];

alpha1=1;

figure;

%Generate Data

for ind=1:length(dTypes)

	D.(dTypes{ind}).x=zeros(dimension,D.(dTypes{ind}).N); %Initialize Data


	%Determine Posteriors

	D.(dTypes{ind}).labels = rand(1,D.(dTypes{ind}).N)>=p(1);

	D.(dTypes{ind}).N0=sum(~D.(dTypes{ind}).labels);

	D.(dTypes{ind}).N1=sum(D.(dTypes{ind}).labels);


	D.(dTypes{ind}).phat(1)=D.(dTypes{ind}).N0/D.(dTypes{ind}).N;

	D.(dTypes{ind}).phat(2)=D.(dTypes{ind}).N1/D.(dTypes{ind}).N;


	[D.(dTypes{ind}).x(:,~D.(dTypes{ind}).labels),...

	D.(dTypes{ind}).dist(:,~D.(dTypes{ind}).labels)]=...

	randGMM(D.(dTypes{ind}).N0,alpha0,mu0,Sigma0);

	[D.(dTypes{ind}).x(:,D.(dTypes{ind}).labels),...

		D.(dTypes{ind}).dist(:,D.(dTypes{ind}).labels)]=...

		randGMM(D.(dTypes{ind}).N1,alpha1,mu1,Sigma1);

	subplot(2,2,ind);

	plot(D.(dTypes{ind}).x(1,~D.(dTypes{ind}).labels),...

D.(dTypes{ind}).x(2,~D.(dTypes{ind}).labels),'b.','DisplayName','Class 0');

	hold all;

	plot(D.(dTypes{ind}).x(1,D.(dTypes{ind}).labels),...

		D.(dTypes{ind}).x(2,D.(dTypes{ind}).labels),'r.','DisplayName','Class 1');

	grid on;
```

```
    xlabel('x1');ylabel('x2');

    title([num2str(D.(dTypes{ind}).N) ' Samples From Two Classes']);


end

legend 'show';

%Part 1: Optimal Classifier with Knowledge of PDFs%

if Part1


px0=evalGMM(D.d20k.x,alpha0,mu0,Sigma0);

px1=evalGaussian(D.d20k.x ,mu1,Sigma1);

discScore=log(px1./px0);

sortDS=sort(discScore);

%Generate vector of gammas for parametric sweep

logGamma=[min(discScore)-eps sort(discScore)+eps];


prob=CalcProb(discScore,logGamma,D.d20k.labels,D.d20k.N0,D.d20k.N1,D.d20k.p
hat);

logGamma_ideal=log(p(1)/p(2));

decision_ideal=discScore>logGamma_ideal;

p10_ideal=sum(decision_ideal==1 & D.d20k.labels==0)/D.d20k.N0;

p11_ideal=sum(decision_ideal==1 & D.d20k.labels==1)/D.d20k.N1;

pFE_ideal=(p10_ideal*D.d20k.N0+(1-p11_ideal)*D.d20k.N1)/(D.d20k.N0+D.d20k.
N1);

% %Estimate Minimum Error

%If multiple minimums are found choose the one closest to the theoretical

%minimum

[prob.min_pFE, prob.min_pFE_ind]=min(prob.pFE);

if length(prob.min_pFE_ind)>1
```

```
[~,minDistTheory_ind]=min(abs(logGamma(prob.min_pFE_ind)-logGamma_ideal));
    prob.min_pFE_ind=prob.min_pFE_ind(minDistTheory_ind);
end
%Find minimum gamma and corresponding false and true positive rates
minGAMMA=exp(logGamma(prob.min_pFE_ind));
prob.min_FP=prob.p10(prob.min_pFE_ind);
prob.min_TP=prob.p11(prob.min_pFE_ind);
fprintf('part1:Estimated: Gamma=%1.2f,
Error=%1.2f%%\n',minGAMMA,100*prob.min_pFE);%

%Plot
plotROC(prob.p10,prob.p11,prob.min_FP,prob.min_TP,p10_ideal,p11_ideal);
plotMinPFE(logGamma,prob.pFE,prob.min_pFE_ind,logGamma_ideal,pFE_ideal);
plotDecisions(D.d20k.x,D.d20k.labels,decision_ideal);
plotERMContours(D.d20k.x,alpha0,mu0,Sigma0,mu1,Sigma1,logGamma_ideal);
end
%Part 2: Classificiation with Maximumlikelihood Parameter Estimation%
if Part2
roc=zeros(4,20001,3);
for ind=1:length(dTypes)-1
    %Estimate Parameters using matlab built in function
    D.(dTypes{ind}).DMM_Est0=...
        fitgmdist(D.(dTypes{ind}).x(:,~D.(dTypes{ind}).labels)',2,'Replicates',10);
    D.(dTypes{ind}).DMM_Est1=...
            fitgmdist(D.(dTypes{ind}).x(:,D.(dTypes{ind}).labels)',1);
    plotContours(D.(dTypes{ind}).x,...
        D.(dTypes{ind}).DMM_Est0.ComponentProportion,...
        D.(dTypes{ind}).DMM_Est0.mu,D.(dTypes{ind}).DMM_Est0.Sigma);
```

```
%Calculate discriminate score
px0=pdf(D.(dTypes{ind}).DMM_Est0,D.d10k.x');
px1=pdf(D.(dTypes{ind}).DMM_Est1,D.d10k.x');
discScore=log(px1'./px0');

sortDS=sort(discScore);
%Generate vector of gammas for parametric sweep
logGamma=[min(discScore)-eps sort(discScore)+eps];
prob=CalcProb(discScore,logGamma,D.d20k.labels,...
D.d20k.N0,D.d20k.N1,D.(dTypes{ind}).phat);

%Estimate Minimum Error
%If multiple minimums are found choose the one closest to the theoretical
%minimum
[prob.min_pFE, prob.min_pFE_ind]=min(prob.pFE);
if length(prob.min_pFE_ind) > 1
    [~,minDistTheory_ind]=...
        min(abs(logGamma(prob.min_pFE_ind)-logGamma_ideal));
    prob.min_pFE_ind=min_pFE_ind(minDistTheory_ind);
end

%Find minimum gamma and corresponding false and true positive rates
minGAMMA=exp(logGamma(prob.min_pFE_ind));
prob.min_FP=prob.p10(prob.min_pFE_ind);
prob.min_TP=prob.p11(prob.min_pFE_ind);

%Plot
```

```
        %plotROC_3(prob.p10,prob.p11,prob.min_FP,prob.min_TP,dTypes);
        roc(1,:,ind)=prob.p10;
        roc(2,:,ind)=prob.p11;
        roc(3,:,ind)=prob.min_FP;
        roc(4,:,ind)=prob.min_TP;


    plotMinPFE(logGamma,prob.pFE,prob.min_pFE_ind,logGamma_ideal,pFE_ideal);
        fprintf('Estimated: Gamma=%1.2f, Error=%1.2f%%\n',...
            minGAMMA,100*prob.min_pFE);
    end
    figure;
    sets=[100,1000,10000,20000];
    for ind=1:length(dTypes)-1
        nameR=('ROC Curve for '+string(sets(ind))+' Samples');
        nameM=('Min.Errror for '+string(sets(ind))+' Samples');
        plot(roc(1,:,ind),roc(2,:,ind),'DisplayName',nameR,'LineWidth',2);
        hold on;
        plot(roc(3,:,ind),roc(4,:,ind),'o','DisplayName',nameM,'LineWidth',2);
        hold on;
    end
    xlabel('Prob. False Positive');
    ylabel('Prob. True Positive');
    title('Mininimum Expected Risk ROC Curves for Training Data');
    legend 'show';
    grid on; box on;


    end
```

```
%Part 3: Classificiation with Maximumlikelihood Parameter Estimation%
options=optimset('MaxFunEvals',3000,'MaxIter',10000);
for ind=1:length(dTypes)

    lin.x=[ones(1,D.(dTypes{ind}).N); D.(dTypes{ind}).x];
    lin.init=zeros(dimension+1,1);

    %[lin.theta,lin.cost]=thetaEst(lin.x,lin.init,D.(dTypes{ind}).labels);
    [lin.theta,lin.cost]=...
        fminsearch(@(theta)(costFun(theta,lin.x,D.(dTypes{ind}).labels)),...
        lin.init,options);
    lin.discScore=lin.theta'*[ones(1,D.d20k.N); D.d20k.x];
    gamma=0;

    lin.prob=CalcProb(lin.discScore,gamma,D.d20k.labels,...
        D.d20k.N0,D.d20k.N1,D.d20k.phat);

% quad.decision=[ones(D.d20k.N,1) D.d20k.x]*quad.theta>0;
  %h=linspace(min(lin.x(:,2))-6,max(lin.x(:,2))+6);
 %v=linspace(min(lin.x(:,3))-6,max(lin.x(:,3))+6);
 %s=getBoundry(h,v,lin.theta);
    plotDecisions(D.d20k.x,D.d20k.labels,lin.prob.decisions);
    title(sprintf('Data and Classifier Decisions Against True Label for Linear
Logistic Fit\nProbability of Error=%1.1f%%',100*lin.prob.pFE));
    fprintf('linear for %d samples is %f\n',sets(ind),lin.prob.pFE)
% plotDecisions(D.d20k.x,D.d20k.labels,quad.decision);

    quad.x=[ones(1,D.(dTypes{ind}).N); D.(dTypes{ind}).x;...
```

```
        D.(dTypes{ind}).x(1,:).^2;...
        D.(dTypes{ind}).x(1,:).*D.(dTypes{ind}).x(2,:);...
        D.(dTypes{ind}).x(2,:).^2];
    quad.init= zeros(2*(dimension+1),1);

    [quad.theta,quad.cost]=...
        fminsearch(@(theta)(costFun(theta,quad.x,D.(dTypes{ind}).labels)),...
        quad.init,options);

    quad.xScore=[ones(1,D.d20k.N); D.d20k.x; D.d20k.x(1,:).^2;...
        D.d20k.x(1,:).*D.d20k.x(2,:); D.d20k.x(2,:).^2];

    quad.discScore=quad.theta'*quad.xScore;
    gamma=0;
    quad.prob=CalcProb(quad.discScore,gamma,D.d10k.labels,...
        D.d20k.N0,D.d20k.N1,D.d20k.phat);

    fprintf('quar for %d samples is %f\n', sets(ind), quad.prob.pFE)
    plotDecisions(D.d20k.x,D.d20k.labels,quad.prob.decisions);
    title(sprintf('Data and Classifier Decisions Against True Label for Linear
Logistic Fit\nProbability of Error=%1.1f%%',100*quad.prob.pFE));

end
%Function Definitions%
function cost=costFun(theta,x,labels)
h=1./(1+exp(-x'*theta));
cost=-1/length(h)*sum((labels'.*log(h)+(1-labels)'.*(log(1-h))));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,labels] = randGMM(N,alpha,mu,Sigma)
d = size(mu,1); % dimensionality of samples
cum_alpha = [0,cumsum(alpha)];
u = rand(1,N); x = zeros(d,N); labels = zeros(1,N);
for m = 1:length(alpha)
    ind = find(cum_alpha(m)<u & u<=cum_alpha(m+1));
    x(:,ind) = randGaussian(length(ind),mu(:,m),Sigma(:,:,m));
    labels(ind)=m-1;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x = randGaussian(N,mu,Sigma)
% Generates N samples from a Gaussian pdf with mean mu covariance Sigma
n = length(mu);
z = randn(n,N);
A = Sigma^(1/2);
x = A*z + repmat(mu,1,N);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gmm = evalGMM(x,alpha,mu,Sigma)
gmm = zeros(1,size(x,2));
for m = 1:length(alpha) % evaluate the GMM on the grid
    gmm = gmm + alpha(m)*evalGaussian(x,mu(:,m),Sigma(:,:,m));
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function g = evalGaussian(x,mu,Sigma)
```

```
% Evaluates the Gaussian pdf N(mu,Sigma) at each coumn of X
[n,N] = size(x);
invSigma = inv(Sigma);
C = (2*pi)^(-n/2) * det(invSigma)^(1/2);
E = -0.5*sum((x-repmat(mu,1,N)).*(invSigma*(x-repmat(mu,1,N))),1);
g = C*exp(E);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function prob=CalcProb(discScore,logGamma,labels,N0,N1,phat)
for ind=1:length(logGamma)
    prob.decisions=discScore>=logGamma(ind);
    Num_pos(ind)=sum(prob.decisions);
    prob.p10(ind)=sum(prob.decisions==1 & labels==0)/N0;
    prob.p11(ind)=sum(prob.decisions==1 & labels==1)/N1;
    prob.p01(ind)=sum(prob.decisions==0 & labels==1)/N1;
    prob.p00(ind)=sum(prob.decisions==0 & labels==0)/N0;

  prob.pFE(ind)=prob.p10(ind)*phat(1) + prob.p01(ind)*phat(2);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotContours(x,alpha,mu,Sigma)
figure
if size(x,1)==2
    plot(x(1,:),x(2,:),'b.');
    xlabel('x_1'), ylabel('x_2'), title('Data and Estimated GMM Contours'),
    axis equal, hold on;
    rangex1 = [min(x(1,:)),max(x(1,:))];
```

```
    rangex2 = [min(x(2,:)),max(x(2,:))];
    [x1Grid,x2Grid,zGMM] = contourGMM(alpha,mu,Sigma,rangex1,rangex2);
    contour(x1Grid,x2Grid,zGMM); axis equal,
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1Grid,x2Grid,zGMM] = contourGMM(alpha,mu,Sigma,rangex1,rangex2)
x1Grid = linspace(floor(rangex1(1)),ceil(rangex1(2)),101);
x2Grid = linspace(floor(rangex2(1)),ceil(rangex2(2)),91);
[h,v] = meshgrid(x1Grid,x2Grid);
GMM = evalGMM([h(:)';v(:)'],alpha, mu, Sigma);
zGMM = reshape(GMM,91,101);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotROC(p10,p11,min_FP,min_TP,p10_ideal,p11_ideal)
figure;
plot(p10,p11,'DisplayName','ROC Curve','LineWidth',2);
hold all;
plot(min_FP,min_TP,'o','DisplayName','Estimated Min. Error','LineWidth',2);
hold all;
plot(p10_ideal,p11_ideal,'*','DisplayName','Ideal Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve');
legend 'show';
grid on; box on;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function plotMinPFE(logGamma,pFE,min_pFE_ind,logGamma_ideal,pFE_ideal)
figure;
plot(logGamma,pFE,'DisplayName','Errors','LineWidth',2);
hold on;
plot(logGamma(min_pFE_ind),pFE(min_pFE_ind),...
    'ro','DisplayName','Minimum Error','LineWidth',2);
hold on;
plot(logGamma_ideal,pFE_ideal,...
    '+g','DisplayName','Calculated Min.Error','LineWidth',2);
xlabel('Gamma');
ylabel('Proportion of Errors');
title('Probability of Error vs. Gamma')
grid on;
legend 'show';
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotROC_3(p10,p11,min_FP,min_TP,dTypes)
figure;
sets=[100,1000,10000,20000];
for ind=1:length(dTypes)-1
    nameR=('ROC Curve for '+string(sets(ind))+' Samples');
    nameM=('Min.Errror for '+string(sets(ind))+' Samples');
    plot(p10,p11,'DisplayName',nameR,'LineWidth',2);
    hold on;
    plot(min_FP,min_TP,'o','DisplayName',nameM,'LineWidth',2);
    hold on;
end
xlabel('Prob. False Positive');
```

```
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curves for Training Data');
legend 'show';
grid on; box on;
% plot(p10,p11,'DisplayName','ROC Curve','LineWidth',2);
% hold all;
% plot(min_FP,min_TP,'o','DisplayName','Estimated Min. Error','LineWidth',2);
% hold all;
% plot(p10_ideal,p11_ideal,'*','DisplayName','Ideal Min. Error','LineWidth',2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotDecisions(x,labels,decisions)
ind00 = find(decisions==0 & labels==0);
ind10 = find(decisions==1 & labels==0);
ind01 = find(decisions==0 & labels==1);
ind11 = find(decisions==1 & labels==1);
figure; % class 0 circle, class 1 +, correct green, incorrect red
plot(x(1,ind00),x(2,ind00),'og','DisplayName','Class 0, Correct'); hold on,
plot(x(1,ind10),x(2,ind10),'or','DisplayName','Class 0, Incorrect'); hold on,
plot(x(1,ind01),x(2,ind01),'+r','DisplayName','Class 1, Correct'); hold on,
plot(x(1,ind11),x(2,ind11),'+g','DisplayName','Class 1, Incorrect'); hold on,
axis equal,
grid on;
title('Data and their classifier decisions versus true labels');
xlabel('x_1'), ylabel('x_2');
legend('Correct decisions for data from Class 0',...
 'Wrong decisions for data from Class 0',...
 'Wrong decisions for data from Class 1',...
```

```
 'Correct decisions for data from Class 1');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotERMContours(x,alpha0,mu0,Sigma0,mu1,Sigma1,logGamma_ideal)
horizontalGrid = linspace(floor(min(x(1,:))),ceil(max(x(1,:))),101);
verticalGrid = linspace(floor(min(x(2,:))),ceil(max(x(2,:))),91);
[h,v] = meshgrid(horizontalGrid,verticalGrid);
discriminantScoreGridValues =...
    log(evalGaussian([h(:)';v(:)'],mu1,Sigma1))-log(evalGMM([h(:)';v(:)'],...
    alpha0,mu0,Sigma0)) - logGamma_ideal;
minDSGV = min(discriminantScoreGridValues);
maxDSGV = max(discriminantScoreGridValues);
discriminantScoreGrid = reshape(discriminantScoreGridValues,91,101);
contour(horizontalGrid,verticalGrid,...
    discriminantScoreGrid,[minDSGV*[0.9,0.6,0.3],0,[0.3,0.6,0.9]*maxDSGV]);
% plot equilevel contours of the discriminant function
% including the contour at level 0 which is the decision boundary
legend('Correct decisions for data from Class 0',...
    'Wrong decisions for data from Class 0',...
    'Wrong decisions for data from Class 1',...
    'Correct decisions for data from Class 1',...
    'Equilevel contours of the discriminant function' ),
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function s = getBoundry(h, v, theta)
z = zeros(length(h), length(v));
for i = 1:length(h)
for j = 1:length(v)
```

```
x_bound = [1 h(i) v(j) h(i)^2 h(i)*v(j) v(j)^2];
z(i, j) = x_bound.*theta;
end
end
s = z';
end
%End Script
```

Q2:

```
%%=======================Question2=========================%%
% Code help and example from Prof.Deniz Nightsnack from GitHub
  clear all, close all,
% True position inside unit circle
radius = rand; theta = 2*pi*rand;
pTrue = [radius*cos(theta);radius*sin(theta)];

for K = 1:4 % for each specified number of landmarks

    % Landmark positions evenly spaces on the unit circle
    radius = 1; theta = [0,2*pi/K*[1:(K-1)]];
    pLandmarks = [radius*cos(theta);radius*sin(theta)];

    % Generate range measurements
    sigma = 3e-1*ones(1,K);
    r = sqrt(sum((repmat(pTrue,1,K)-pLandmarks).^2,1)) + sigma.*randn(1,K);

    % Parameters of the prior
```

```
        sigmax = 25e-2; sigmay = sigmax;

        % Evaluate the MAP estimation objective function on a grid
        Nx = 101; Ny = 99;
        xGrid = linspace(-2,2,Nx); yGrid = linspace(-2,2,Ny);
        [h,v] = meshgrid(xGrid,yGrid);
        MAPobjective = (h(:)/sigmax).^2 + (v(:)/sigmay).^2;
        for i = 1:K
            di = sqrt((h(:)-pLandmarks(1,i)).^2+(v(:)-pLandmarks(2,i)).^2);
            MAPobjective = MAPobjective + ((r(i)-di)/sigma(i)).^2;
        end
        zGrid = reshape(MAPobjective,Ny,Nx);

        % Display true position and landmark positions
        figure(ceil(K/4)), subplot(2,2,mod(K-1,4)+1),
        plot(pTrue(1),pTrue(2),'+'); hold on,
        plot(pLandmarks(1,:),pLandmarks(2,:),'o'); axis([-2 2 -2 2]),
        % Display the MAP objective contours
        minV = min(MAPobjective); maxV = max(MAPobjective);
        values = minV + (sqrt(maxV-minV)*linspace(0.1,0.9,21)).^2;
        contour(xGrid,yGrid,zGrid,values); xlabel('x'), ylabel('y'),
        title(strcat({'MAP Objective for K = '},num2str(K)));
        grid on, axis equal,
    end
```