Assignment 2

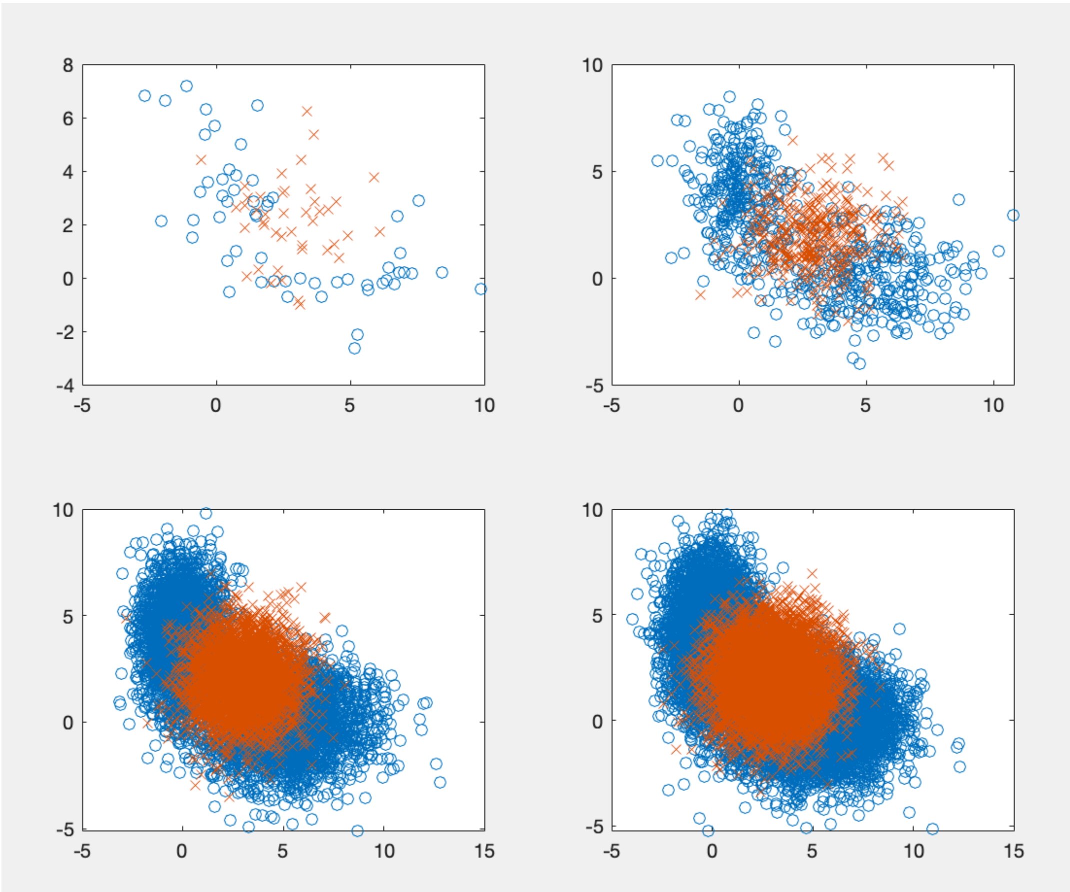
by Jiabo Cheng

Question1:

Generate training dataset and validation dataset from class-conditional Gaussian pdfs below:

$$\mathbf{m}_{01} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \quad \mathbf{m}_{02} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \mathbf{m}_{1} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \mathbf{C}_{1} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

The visualization of generated dataset is shown below:



Part 1:

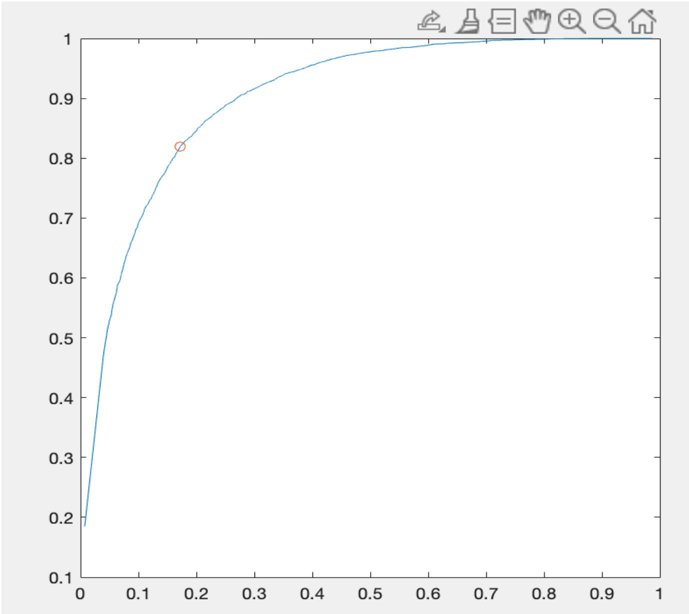
The following equation represents the expression for classifier:

Loss Matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$\frac{P(x \mid L=1)}{P(x \mid L=0)} \underset{D=0}{\overset{D=1}{\gtrless}} \frac{Prior(0)}{Prior(1)} \cdot \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}}$$

$$(D=1) \quad \frac{g(x \mid m_{01}, m_{02}, C_{01}, C_{02}, \alpha)}{g(x \mid m_1, C_1)} \gtrless \frac{0.6}{0.4} \quad (D=0)$$

The ROC curve with the point achieved the minimum P-error:



The minimum P-error is 0.1735.

Part 2:

Using the maximum likelihood parameter estimation technique:

Parameter Estimation

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta) = L(\theta|X)$$

$$\theta^* = \arg\max L(\theta|X)$$

$L=1$:

Estimation of μ:

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

Estimation of Σ:

$$\hat{\delta}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i-\mu)^2$$

$L=0$:

Estimation of α, μ, σ^2

$$\alpha, \mu, \sigma^2 = \arg\max_{(\alpha,\mu,\sigma^2)} \sum_{i=1}^{M} \log \sum_{j=1}^{k} P(x_i|z=j;\mu,\sigma)P(z=j;\theta)$$

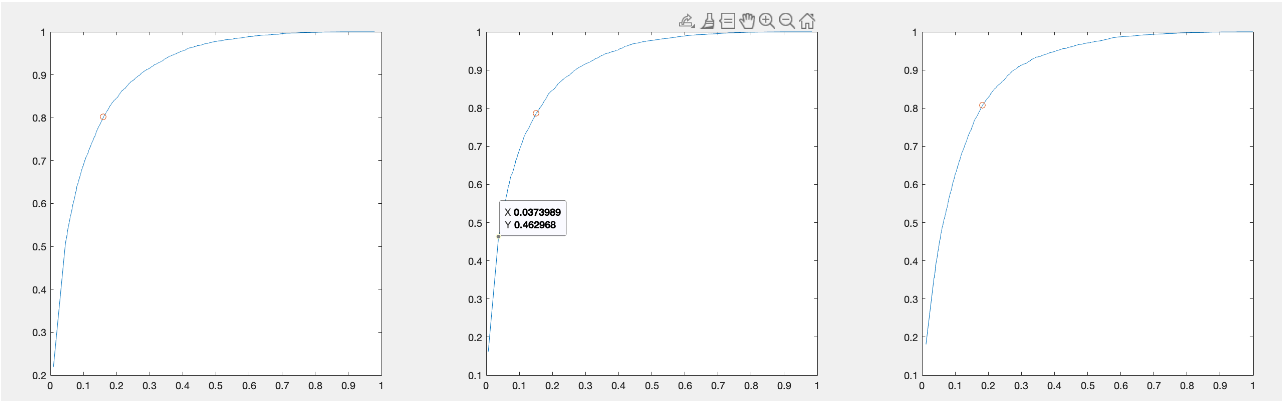
$$\alpha, \mu, \sigma^2 = \arg\max \sum_{i=1}^{m} \log \phi N(a_i;\mu_1,\sigma_1^2) + \phi_2 N(a_2,\mu_2,\sigma_2^2)$$

The parameter estimation result is (Using 10k training set):

P	Mu1	Sigma1	Alpha	Mu0	Sigma0
[0.5944, 0.4056]	[2.9934, 1.9878]	[1.9670, -0.0376; -0.0376, 2.0844]	[0.4956, 0.5044]	[4.9997,-0.0308; 0.0114, 4.0407]	[3,9041, -0.0014; -0.0014, 2.0701] [0.9854, -0.0439; -0.0430, 2.9155]

The parameters estimated above are close to the true parameters.

Generate the ROC curve using the parameter estimated by 10k, 1k, and 100 training dataset. (From left to right, the dataset size is 10k, 1k, 100k)



The minimum P_error is:

	10k	1k	100
Min_P_Error	0.1749	0.1726	0.1847

From the ROC curved and the P_error, we could found the accuracy will be slightly higher when the size of training data increases.

Part 3

The transformation function applied to logistic-linear-function classifier is shown below:

Linear Transformation

$$z(x) = [1, x^T]^T$$

$$w = [\theta_0, \theta_1, \theta_2]^T$$

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

$$\text{cost} \begin{cases} -\log[h(x,w)] & \text{if } y=1 \\ -\log[1-h(x,w)] & \text{if } y=0 \end{cases}$$

$$\Rightarrow \text{cost} = -y\log[h(x,w)] - (1-y)\log[1-h(x,w)]$$

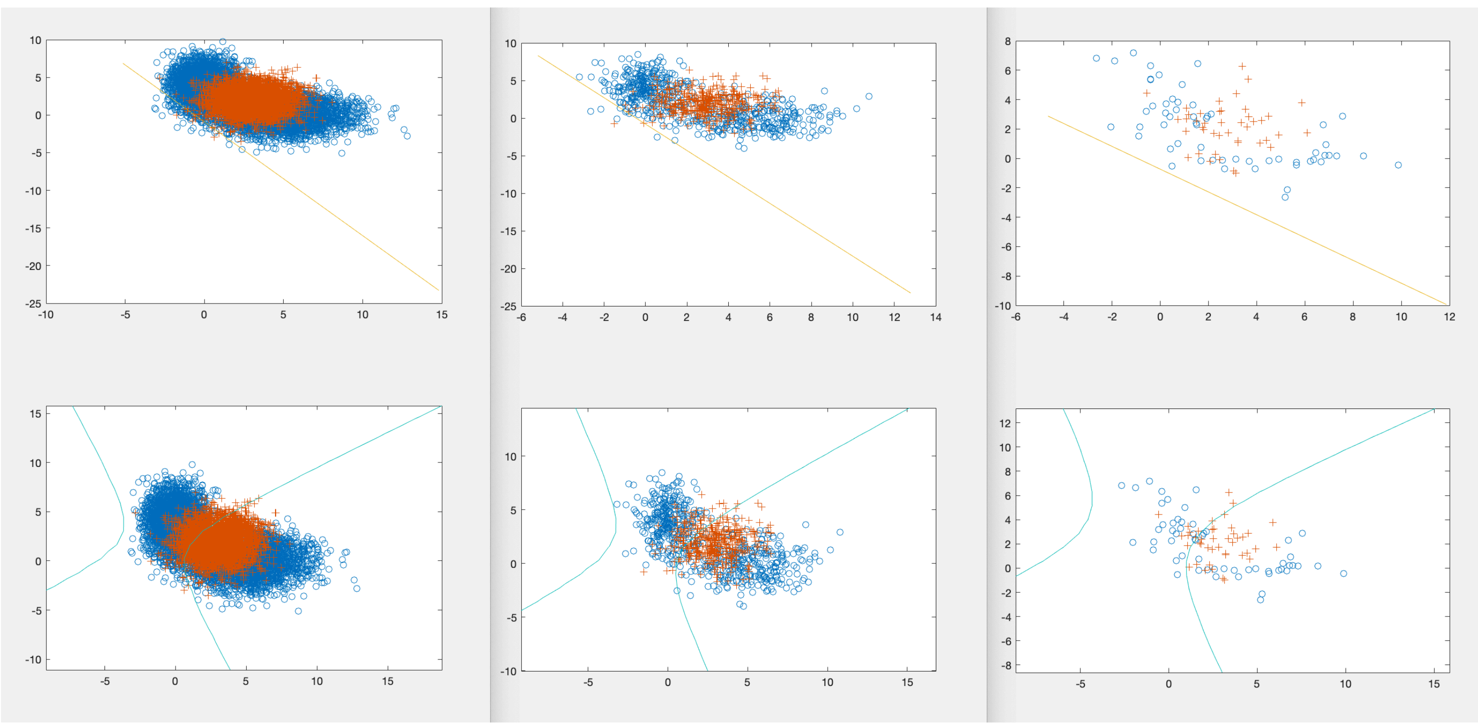
Quadratic Transformation:

$$z(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$$

$$w = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$$

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

The visualization of the training dataset and the decision boundary is shown below:



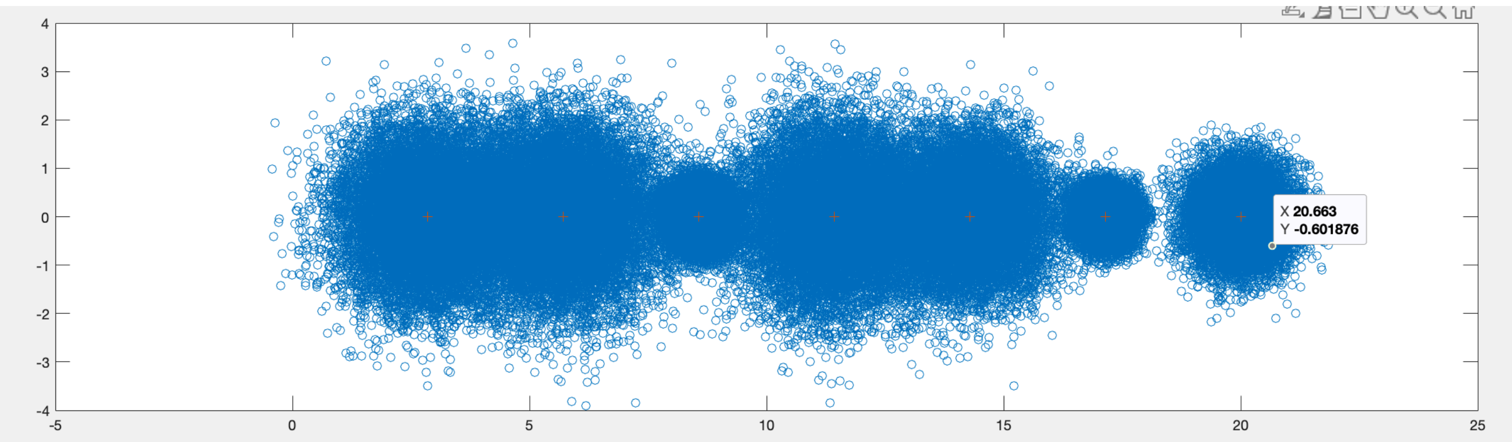
The classification result is:

	10k	1k	100
Logistic-Linear-function minimum_P_Error	0.4312	0.4162	0.4133
Logistic-Quadratic-function minimum_P_Error	0.1779	0.1787	0.1858

The result shows that with a larger training set, the minimum_P_Error will be slightly lower. The reason for the abnormal liner-function P_Error result might be the maximum iteration times in fminsearch limit the accuracy of parameter estimation.
However, obviously, based on the visualization above, the logistic-quadratic-function has a significant advantage in classifying this dataset.

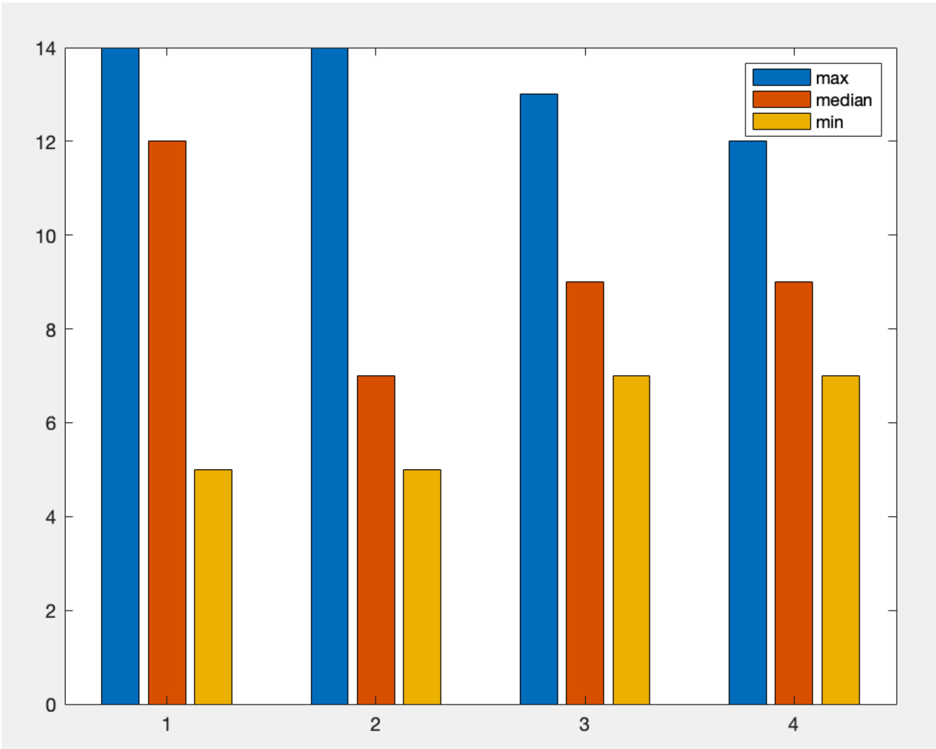
Question 2:

Generate a 2-dimensional C(C=7) Gaussian components with mean vectors spaced on a line. The generated dataset shows below:

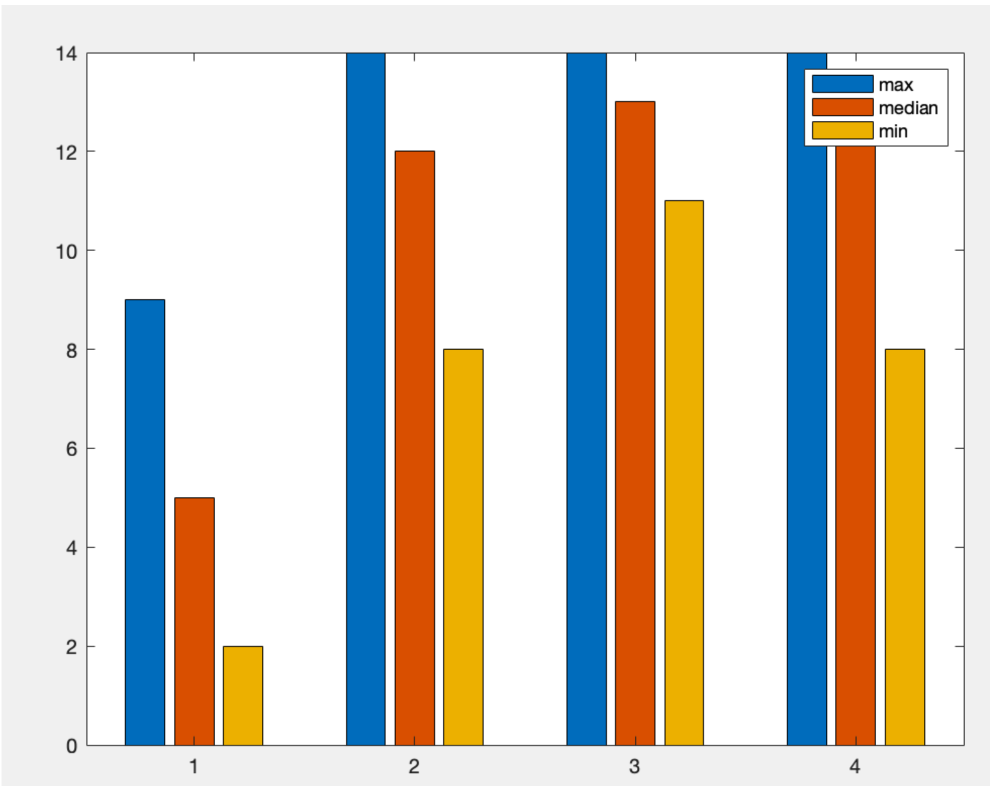


The mean is evenly distributed from (1,0) to(20,0), sigma is identical matrix.

The Bar plot below shows the maximum, median, minimum M selected using BIC verses the size of dataset. (From left to right represents 100, 1k, 10k, 100k samples)
From the bar plot above, we could found with more training data, the M selected by BIC will be smaller, the range between maximum M and minimum M also becomes smaller. It shows the selected Ms are closing to the true C(C=7), which means the parameters estimation is more accurate.



The bar plot shows the maximum, median, and minimum M selected using K-fold(K=10) method verses the size of dataset(From left to right represents 100, 1k, 10k, 100k samples)



From the bar plot above, as increase of training samples, the M selected by K-fold also increase. The reason of the continuous increment of selected Ms might be the K-fold method doesn't have any penalty for the number of parameters, which might lead to overfitting. When M>C(C=7), the estimated distribution is able to cover the data generated from GMM with smaller number of components, all the M above actual C(C=7) would have reasonable accuracy.

Appendix-Question1

```
% Question 1
% Init
clear all;
close all;
n=2;
N_train = [100,1000,10000];
N_valid = 20000;
P = [0.6,0.4];
mu1 = [3;2];
C1 = [2 0; 0 2];

mu0 = [5 0; 0 4];
C0(:,:,1) = [4 0; 0 2];
C0(:,:,2) = [1 0; 0 3];

alpha = [0.5,0.5];
figure(1)
[X_train_100, label100] = generate_data(n, N_train(1),P,mu1,C1,alpha,mu0,C0,1);
[X_train_1000, label1000] = generate_data(n, N_train(2),P,mu1,C1,alpha,mu0,C0,2);
[X_train_10000, label10000] = generate_data(n, N_train(3),P,mu1,C1,alpha,mu0,C0,3);

[X_valid, vlabel] = generate_data(n,N_valid,P,mu1,C1,alpha,mu0,C0,4);

%plot(X_train_100(1,:),X_train_100(2,:),'o')

% Part 1
%score = eval_g(X_train_100,mu1,C1)
figure_n = 1;
figure(2)
min_error = PClassifier(X_valid,vlabel,P,mu1,C1,alpha,mu0,C0,figure_n)

%-------------------_????_------------------
%create grid?
%h_grid = linspace(floor(min(X_valid(1,:)))-2,ceil(max(X_valid(1,:)))+2);
%v_grid = linspace(floor(min(X_valid(2,:)))-2,ceil(max(X_valid(2,:)))+2);
%[h,v]= meshgrid(h_grid,v_grid);

%calculate score for gird point
%Score_grid = log(eval_g([h(:)';v(:)'],mu1,C1))-log(eval_gmm([h(:)';v(:)'],alpha,mu0,C0));
%S_grid = reshape(Score_grid,length(h_grid),length(v_grid));

%figure(2)
%plot_classified_data
```

```
% Part2
%Train10000
[P_est,mu1_est,C1_est,alpha_e,mu0_est,C0_est]= Param_est(X_train_10000,label10000);
figure(3)
figure_n = 1
alphaT = alpha_e';
min_error =
PClassifier(X_valid,vlabel,P_est,mu1_est,C1_est,alphaT,mu0_est,C0_est,figure_n)
%train1000
[P_est,mu1_est,C1_est,alpha_e,mu0_est,C0_est]= Param_est(X_train_1000,label1000)

figure_n = 2
alphaT = alpha_e';
min_error =
PClassifier(X_valid,vlabel,P_est,mu1_est,C1_est,alphaT,mu0_est,C0_est,figure_n)
%train100
[P_est,mu1_est,C1_est,alpha_e,mu0_est,C0_est]= Param_est(X_train_100,label100)

figure_n = 3
alphaT = alpha_e';
min_error =
PClassifier(X_valid,vlabel,P_est,mu1_est,C1_est,alphaT,mu0_est,C0_est,figure_n)

%-------------
%part 3
%Linear
%figure(4)
[error_L10k,error_Q10k] = logistic_classifier(X_train_10000,label10000,N_valid, X_valid,
vlabel,4,P)

[error_L1k,error_Q1k] = logistic_classifier(X_train_1000,label1000,N_valid, X_valid,
vlabel,5,P)

[error_L,error_Q] = logistic_classifier(X_train_100,label100,N_valid, X_valid, vlabel,6,P)


function cost = cost_func(theta, x, label, N)
h=1 ./ (1+exp(-x*theta));
cost = (-1/N)*((sum(label'*log(h)))+(sum((1-label)'*log(1-h))));

end

%---------------------------
function [P_est,mu1_est,C1_est,alpha_e,mu0_est,C0_est] = Param_est(X,label)
P_est = [1-(sum(label)/length(label)), sum(label)/length(label)];
M = 1;

x= X(:,label==1);
```

```
N= length(x);
[aa,mu1_est,C1_est] = EMforGmm(M,N,x)

M = 2;

x= X(:,label==0);
N= length(x);
[alpha_e,mu0_est,C0_est]=EMforGmm(M,N,x)

end



%---------------------------------
% Generate true labels
function [alpha_e,mu,Sigma] = EMforGmm(M,N,x)

regWeight = 1e-10;
delta = 1e-2;
alpha_e = ones(1,M)/M;
shuffle = randperm(N);
mu= x(:,shuffle(1:M));
[~, centroidlabel] = min(pdist2(mu',x'),[],1);
for m= 1:M
    Sigma(:,:,m) = cov(x(:,find(centroidlabel==m))')+regWeight*eye(2,2);

end

%maximum
t=0;
converged = 0;
while ~converged
  for l = 1:M
      temp(l,:) = repmat(alpha_e(l),1,N).*eval_g(x,mu(:,l),Sigma(:,:,l));
  end
  plgivenx = temp./sum(temp,1);
  alphaNew = mean(plgivenx,2);
  w = plgivenx ./repmat(sum(plgivenx,2),1,N);
  muNew = x*w';

  for l=1:M
    v= x-repmat(muNew(:,l),1,N);
    u = repmat(w(l,:),2,1).*v;
    SigmaNew(:,:,l) = u*v' +regWeight*eye(2,2);
  end
  Dalpha = sum(abs(alphaNew-alpha_e));
  Dmu = sum(sum(abs(muNew-mu)));
  DSigma = sum(sum(abs(abs(SigmaNew-Sigma))));
  converged = ((Dalpha+Dmu+DSigma)<delta);
```

```
  alpha_e = alphaNew;
  mu = muNew;
  Sigma = SigmaNew;
  t=t+1
end

end

%draw plot
function min_error = PClassifier(X,label,P,mu1,C1,alpha,mu0,C0,figure_n)
Score = log(eval_g(X,mu1,C1))-log(eval_gmm(X,alpha,mu0,C0));
tau = log(sort(Score(Score >=0)));
mid_tau =[tau(1)-1, tau(1:end-1)+diff(tau)./2 tau(end)+1];
for i = 1:length(mid_tau)
   decision = (Score>=mid_tau(i));
   pFA(i) = sum(decision==1 & label== 0)/length(label(label==0));
   pCD(i) = sum(decision==1 & label== 1)/length(label(label==1));
   pE(i) = pFA(i)*P(1)+(1-pCD(i))*P(2);

end
%find the minimal pe
[min_error,min_index] = min(pE);
min_decision = (Score >=mid_tau(min_index));
min_FA = pFA(min_index);
min_CD = pCD(min_index);

subplot(1,3,figure_n)
plot(pFA,pCD,'-',min_FA,min_CD,'o')
end

%part3
function [error_L,error_Q] = logistic_classifier(X_train,label_t,N_valid, X_valid,
vlabel,figure_n,P)
n=2;

x_L = [ones(length(X_train),1) X_train'];
initial_theta_L = zeros(n+1,1);
label = double(label_t)';
[theta_L, cost_L] = fminsearch(@(t)(cost_func(t,x_L,label,length(X_train))),initial_theta_L);
%validate
valid_L = [ones(N_valid,1) X_valid'];
decision_L = valid_L*theta_L>=0;
pFA=length(find(decision_L'==1 & vlabel==0))/length(find(vlabel ==0));
pMD=length(find(decision_L'==0 & vlabel==1))/length(find(vlabel ==1));
error_L = pFA*P(1)+pMD*P(2);
%plot
plot_x1 = [min(x_L(:,2))-2, max(x_L(:,2))+2];
plot_x2 = (-1./theta_L(3).* theta_L(2).*plot_x1+theta_L(1));
```

```
bound = [plot_x1;plot_x2];

figure(figure_n);
subplot(2,1,1)
plot(x_L(label==0,2),x_L(label==0,3),'o',x_L(label==1,2),x_L(label==1,3),'+');hold on;
plot(bound(1,:),bound(2,:));

%Q
x_Q = [ones(length(X_train),1) X_train',(X_train(1,:).*X_train(2,:))', (X_train.^2)'];
initial_theta_Q = zeros(6,1);

[theta_Q, cost_Q] = fminsearch(@(t)(cost_func(t,x_Q,label,length(X_train))),initial_theta_Q);
%validate
valid_Q =[ones(length(X_valid),1) X_valid',(X_valid(1,:).*X_valid(2,:))', (X_valid.^2)'];
decision_Q = valid_Q *theta_Q >=0;
pFA=length(find(decision_Q'==1 & vlabel==0))/length(find(vlabel ==0));
pMD=length(find(decision_Q'==0 & vlabel==1))/length(find(vlabel ==1));
error_Q = pFA*P(1)+pMD*P(2);

hgrid = linspace(min(x_Q(:,2))-6, max(x_Q(:,2))+6,20);
vgrid = linspace(min(x_Q(:,3))-6, max(x_Q(:,3))+6,20);
z= zeros(length(hgrid),length(vgrid));
for i=1:length(hgrid)
  for j=1:length(vgrid)
     xbound=[1, hgrid(i) vgrid(j) hgrid(i)^2 hgrid(i)*vgrid(j) vgrid(j)^2];
     z(i,j) = xbound* theta_Q;
  end
end
gridscore = z';

bound = [hgrid;vgrid;gridscore];
subplot(2,1,2)
plot(x_Q(label==0,2),x_Q(label==0,3),'o',x_Q(label==1,2),x_Q(label==1,3),'+');hold on;
contour(bound(1,:),bound(2,:),bound(3:end,:),[0,0]);
end




function [x, label] = generate_data(n, N, P, mu1, sigma1,alpha,mu0,sigma0,i)
x = zeros(n,N);
label = (rand(1,N)>=P(1));
Nc = [length(find(label ==0)),length(find(label ==1))];
x(:,label ==1) = mvnrnd(mu1,sigma1,Nc(2))';
x(:,label ==0) = gmm_gen (n,Nc(1),alpha,mu0,sigma0);
subplot(2,2,i);
plot(x(1,label==0),x(2,label==0),'o',x(1,label==1),x(2,label==1),'x')
end
```

```
function xgmm = gmm_gen(n,N,alpha, mu0, sigma0)
cum_alpha = [0,cumsum(alpha)];
u = rand(1,N);
xgmm = zeros(n,N);
for m = 1:length(alpha)
    ind = find((cum_alpha(m)<u)&(u <=cum_alpha(m+1)));
    xgmm(:,ind)=mvnrnd(mu0(:,m),sigma0(:,:,m),length(ind))';

end
end
%function g=eval_g(x,mu,Sigma)
%[n,N] = size(x);
%C = ((2*pi)^n * det(Sigma))^(-1/2);
%g = C*exp(E);
%end
function g = eval_g(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 s=eval_gmm(x,alpha,mu0,Sigma0)
[n,N] = size(x);
g = zeros(length(alpha),N);
for i= 1:length(alpha)

    C = ((2*pi)^n * det(Sigma0(:,:,i)))^(-1/2);
    E = -0.5*sum((x-repmat(mu0(:,i),1,N)).*(inv(Sigma0(:,:,i))*(x-repmat(mu0(:,i),1,N))),1);
    g(i,:)= C*exp(E);
end
g;
s = alpha* g;

end

%------------
```

Appendix-Question2

```
clear all;
close all;
warning('off')
parfevalOnAll(gcp(), @warning, 0, 'off', 'MATLAB:singularMatrix');
n=2;
C=7;
```

```
N_o=10^5;
[N_train] = [10^2, 10^3, 10^4, 10^5];
alpha = (ones(C,1)/C)';
    BIC_100 = [];
    BIC_1k = [];
    BIC_10k = [];
    BIC_100k = [];
    KF_100 = [];
    KF_1k = [];
    KF_10k = [];
    KF_100k = [];
miu = cumsum(ones(1,C)/C);
for i=1:C
    mu(:,i)= [20*miu(i) 0]; %adjustable
    sigma(:,:,i) = 1*rand(1)*eye(n);
end

X_o = gmm_gen(n,N_o, alpha, mu,sigma);
figure(1)
plot(X_o(1,:),X_o(2,:),'o',mu(1,:),mu(2,:),'+')

E=100;
hist_BIC = zeros(E,length(N_train));
hist_KF = zeros(E,length(N_train));

for e =1:E
    e
    clearvars sigma_e;
    clearvars mu_e;
    clearvars alpha_e;
    level = ceil(4*rand(1));
    %level=1
    N = N_train(level)
    X = X_o(:,ceil(N*rand(1,N)));

    %-------------BIC
    %kfold
    npercept = 5;
    K=10; %---------

    maxM=14; %----------
    tic
    for M = 1:maxM

        nParams(1,M) = (M-1)+ n*M + M*(n+nchoosek(2,2));
       [alpha_e,mu_e,sigma_e] = EMforGmm(M,N,X);
        neg2loglike(1,M) = -2*sum(log(eval_gmm(X,alpha_e,mu_e,sigma_e)));
        BIC(1,M) = neg2loglike(1,M) + nParams(1,M)*log(N);
```

```
    %kfold
    clearvars dummy;
    dummy = ceil(linspace(0,N,K+1));

    parfor k = 1:K %enable parrallel

      %part_ind(k,:)=[dummy(k)+1, dummy(k+1)];
      K_valid = X(:,[dummy(k)+1:dummy(k+1)]);
      train_ind = setdiff([1:N],[dummy(k)+1:dummy(k+1)]);
      K_train = X(:,train_ind);
  %estimate parameter

      [alpha_e,mu_e,sigma_e] = EMforGmm(M,length(K_train),K_train);
      loglike(k,:) = sum(log(eval_gmm(K_valid,alpha_e,mu_e,sigma_e)));
    end

    aveloglike(1,M) = mean(loglike);
  end
  toc

  [~, BIC_M] = min(BIC);
  [~, KF_M] = max(aveloglike);



  if level==1
    BIC_100 = [BIC_100, BIC_M]
    KF_100 = [KF_100, KF_M]
  elseif  level==2
    BIC_1k = [BIC_1k, BIC_M]
    KF_1k = [KF_1k, KF_M]
  elseif level==3
    BIC_10k = [BIC_10k, BIC_M]
    KF_10k = [KF_10k, KF_M]
  elseif level==4
    BIC_100k = [BIC_100k, BIC_M]
    KF_100k = [KF_100k, KF_M]
  end

  %hist_BIC(e,level)= BIC_M
  %hist_KF(e,level) = KF_M



end
```

```
max1 = [max(BIC_100) max(BIC_1k) max(BIC_10k) max(BIC_100k)];
avg1 = [median(BIC_100) median(BIC_1k) median(BIC_10k) median(BIC_100k)];
low1 = [min(BIC_100) min(BIC_1k) min(BIC_10k) min(BIC_100k)];
figure(2)
bar([1:4],[max1; avg1;low1]);
legend('max','median','min');

max2 = [max(KF_100) max(KF_1k) max(KF_10k) max(KF_100k)];
avg2 = [median(KF_100) median(KF_1k) median(KF_10k) median(KF_100k)];
low2 = [min(KF_100) min(KF_1k) min(KF_10k) min(KF_100k)];
figure(3)
bar([1:4],[max2; avg2;low2]);
legend('max','median','min');

function [alpha_e,mu,Sigma] = EMforGmm(M,N,x)

regWeight = 1e-10;
delta = 1e-2;
alpha_e = ones(1,M)/M;
shuffle = randperm(N);
mu= x(:,shuffle(1:M));
[~, centroidlabel] = min(pdist2(mu',x'),[],1);
for m= 1:M
    Sigma(:,:,m) = cov(x(:,find(centroidlabel==m))')+regWeight*eye(2,2);

end

%maximum
t=0;
converged = 0;
while ~converged


   for l = 1:M
       temp(l,:) = repmat(alpha_e(l),1,N).*eval_g(x,mu(:,l),Sigma(:,:,l));
   end



   plgivenx = temp./sum(temp,1);
   alphaNew = mean(plgivenx,2);
   w = plgivenx ./repmat(sum(plgivenx,2),1,N);
   muNew = x*w';

   for l=1:M
     v= x-repmat(muNew(:,l),1,N);
     u = repmat(w(l,:),2,1).*v;
     SigmaNew(:,:,l) = u*v' +regWeight*eye(2,2);
   end
```

```
   Dalpha = sum(abs(alphaNew-alpha_e));
   Dmu = sum(sum(abs(muNew-mu)));
   DSigma = sum(sum(abs(abs(SigmaNew-Sigma))));
   converged = ((Dalpha+Dmu+DSigma)<delta);
   alpha_e = alphaNew;
   mu = muNew;
   Sigma = SigmaNew;
   t=t+1;
   if t == 150
      converged = 1;
   end
end

end

function s=eval_gmm(x,alpha,mu0,Sigma0)
[n,N] = size(x);
g = zeros(length(alpha),N);

for i= 1:length(alpha)
    C = ((2*pi)^n * det(Sigma0(:,:,i)))^(-1/2);
    E = -0.5*sum((x-repmat(mu0(:,i),1,N)).*(inv(Sigma0(:,:,i))*(x-repmat(mu0(:,i),1,N))),1);
    g(i,:)= C*exp(E);
end

g;
s = alpha'* g;

end


function xgmm = gmm_gen(n,N,alpha, mu0, sigma0)
cum_alpha = [0,cumsum(alpha)];
u = rand(1,N);
xgmm = zeros(n,N);
for m = 1:length(alpha)
    ind = find((cum_alpha(m)<u)&(u <=cum_alpha(m+1)));
    xgmm(:,ind)=mvnrnd(mu0(:,m),sigma0(:,:,m),length(ind))';

end
end


function g = eval_g(x,mu,Sigma)
% Evaluates the Gaussian pdf N(mu,Sigma) at each coumn of X
[n,N] = size(x);
%invSigma = Sigma\eye(size(Sigma)) ;
```

```
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
```