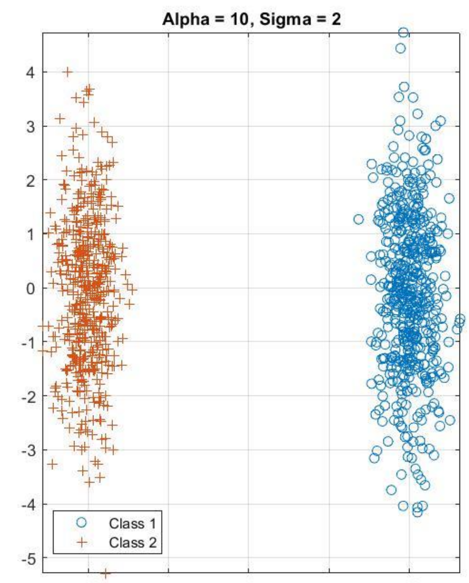
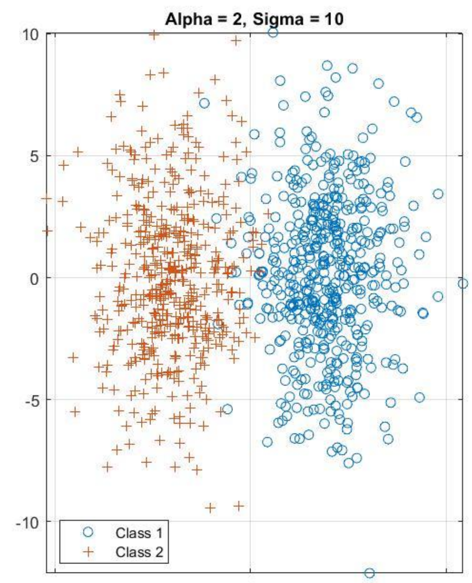
Problem 4.

(b)

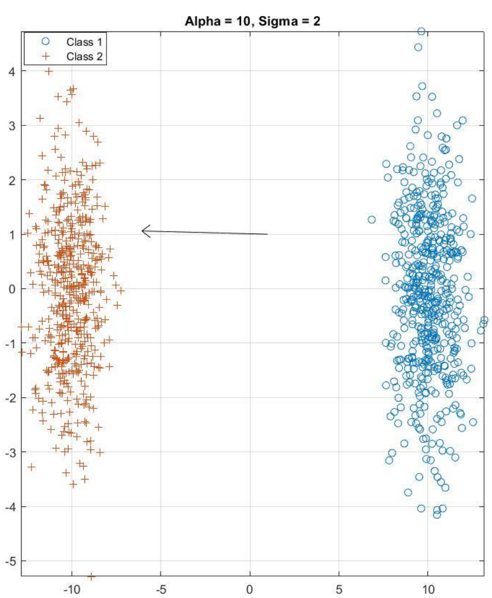


Alpha = 10, Sigma = 2
Class 1
Class 2

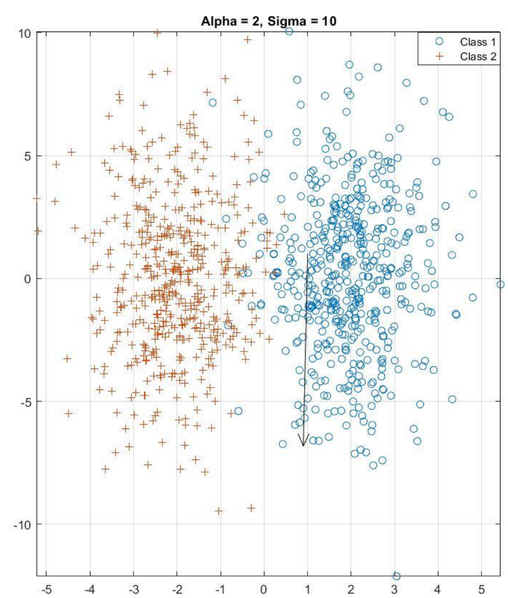


Alpha = 2, Sigma = 10
Class 1
Class 2

(c)

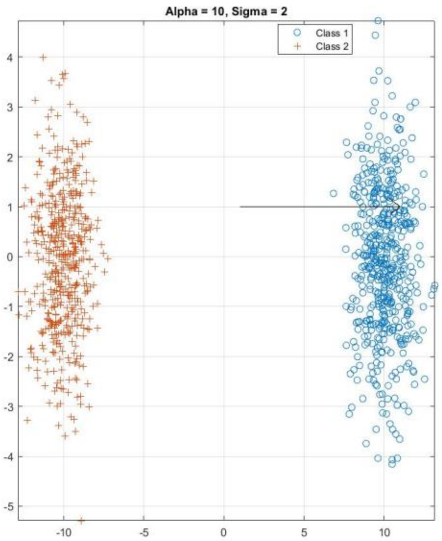


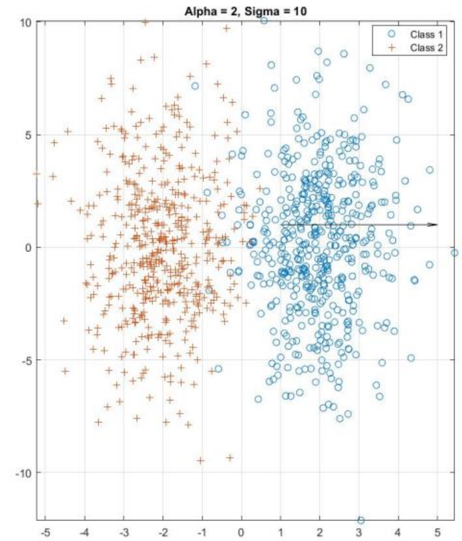
Alpha = 10, Sigma = 2
Class 1
Class 2



Alpha = 2, Sigma = 10
Class 1
Class 2

(d)



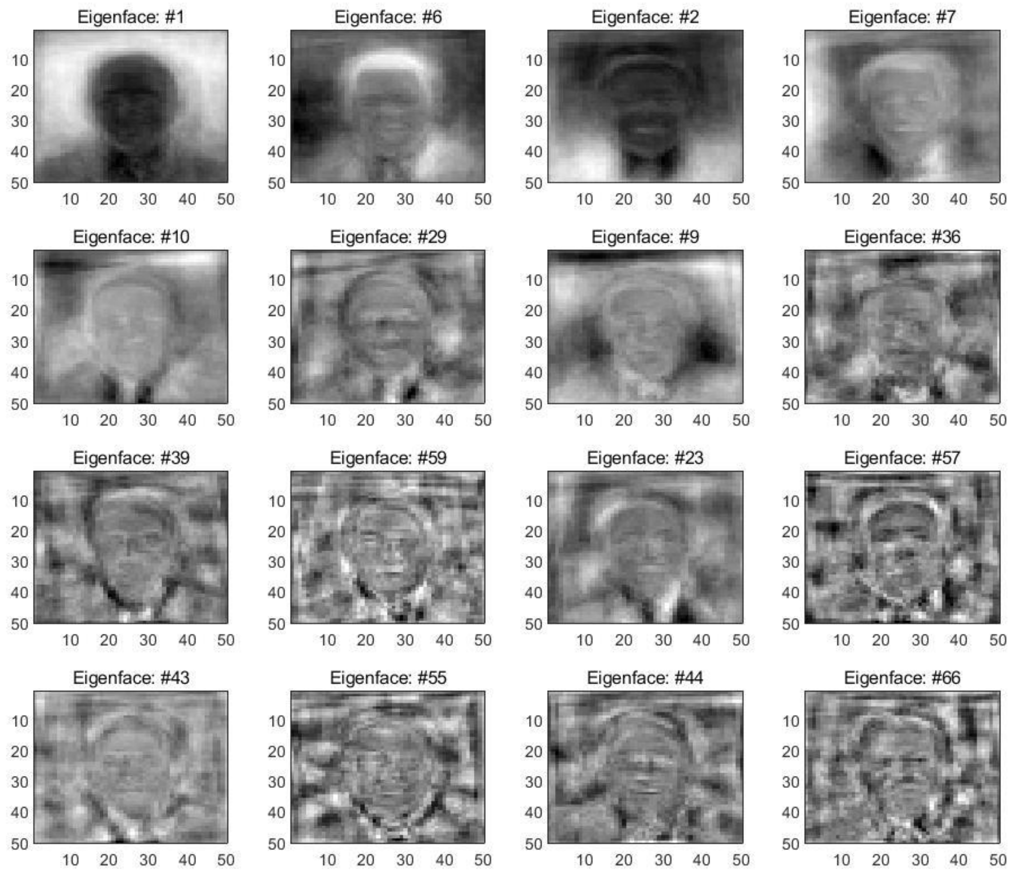


(e) From the results shown above, the PCA is not always a good approach to dimensionality reduction. The direction of the largest variance is not always the direction of discriminant which can be observed from condition B. However, the LDA provides a better direction of discrimination. Therefore, for the classification of these classes, the LDA is the better approach.

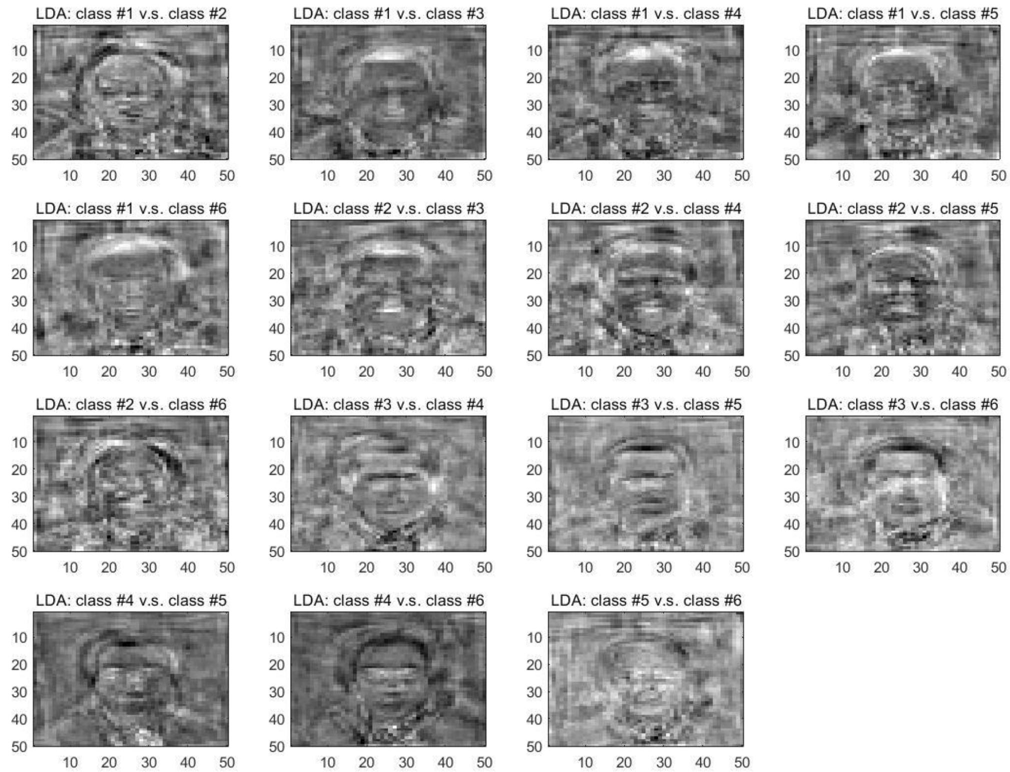
Problem 5.

(a)

The Most 16 Representative Eigenfaces in Training Set



(b)



(d) The probability of error of each class is: 40%, 30%, 60% 40%,70%, 50 %

The # of error of each class are: 4, 3, 6, 4, 7, 5

The overall error rate of all classes is: 48.333%

(e) The probability of error of each class is: 30%, 20%, 20%, 10%, 0%, 30%,

The # of error of each class are: 3, 2, 2, 1, 0, 3,

The overall error rate of all classes is: 18.333%

(f) The probability of error of each class is: 60%, 30%, 80%, 50%, 50%, 40%

The # of error of each class are: 6, 3, 8, 5, 5, 4

The overall error rate of all classes is: 51.667%

Code:

```
clear;
%% 1.4 (b)
% Condition A: alpha = 10, var = 2
% Condition B: alpha = 2, var = 10
Alpha = [10,2];
Var = [2,10];
Gaussian = [];
figure(1)
for i = 1:2
    subplot(1,2,i)
    mu = [Alpha(i);0];
    Sigma = [1,0;0,Var(i)];
    Gaussian_1 = mvnrnd(mu,Sigma,500);
    Gaussian_2 = mvnrnd(-1*mu,Sigma,500);
    Gaussian = [Gaussian, Gaussian_1, Gaussian_2];

    plot(Gaussian_1(:,1),Gaussian_1(:,2),'o'); hold on;
    plot(Gaussian_2(:,1),Gaussian_2(:,2),'+'); hold off;
    grid on;
    axis tight;
    legend('Class 1','Class 2');
    title("Alpha = " + Alpha(i)+ ", Sigma = " + Var(i));
end
%%
figure(2)
subplot(1,2,1)
[~,~,V_1] = svd(Gaussian(:,1:4));
PCs = Gaussian(:,1:4)* V_1;

plot(Gaussian(:,1),Gaussian(:,2),'o');hold on;
plot(Gaussian(:,3),Gaussian(:,4),'+');
quiver(V_1(1,1),V_1(2,1),10,'black');hold off;
grid on;
axis tight;
legend('Class 1','Class 2');
title("Alpha = " + Alpha(1)+ ", Sigma = " + Var(1));

subplot(1,2,2)
[~,~,V_2] = svd(Gaussian(:,5:8));

plot(Gaussian(:,5),Gaussian(:,6),'o');hold on;
plot(Gaussian(:,7),Gaussian(:,8),'+');
quiver(V_2(1,1),V_2(2,1),10,'black');hold off;
grid on;
axis tight;
legend('Class 1','Class 2');
title("Alpha = " + Alpha(2)+ ", Sigma = " + Var(2));
%%
figure(3)
subplot(1,2,1)
mu = [Alpha(1);0];
Sigma = [1,0;0,Var(1)];
w_1 = inv(Sigma)*(2*mu);
```

```
plot(Gaussian(:,1),Gaussian(:,2),'o');hold on;
plot(Gaussian(:,3),Gaussian(:,4),'+');
quiver(1/20*w_1(1),1/20*w_1(2),10,'black'); hold off;
grid on;
axis tight;
legend('Class 1','Class 2');
title("Alpha = " + Alpha(1)+ ", Sigma = " + Var(1));

subplot(1,2,2)
mu = [Alpha(2);0];
Sigma = [1,0;0,Var(2)];
w_2 = inv(Sigma)*(2*mu);
plot(Gaussian(:,5),Gaussian(:,6),'o');hold on;
plot(Gaussian(:,7),Gaussian(:,8),'+');
quiver(1/10*w_2(1),1/10*w_2(2),10,'black'); hold off;
grid on;
axis tight;
legend('Class 1','Class 2');
title("Alpha = " + Alpha(2)+ ", Sigma = " + Var(2));
```

```
clear;
% Load images
img = imread("trainset\subset0\person_1_1.jpg");
img_size = size(img);
imgs = zeros([img_size(1)*img_size(2),240]);
idx = 1;
for i = 0:5
    for j = 1:40
        train_name =
"trainset\subset"+int2str(i)+"\person_"+int2str(i+1)+"_"+int2str(j)+".jpg";
        if isfile(train_name)
            imgs(:,idx) = reshape(imread(train_name),[img_size(1)*img_size(2),1]);
            idx = idx+1;
        end
    end
end

idx = 1;
tests = zeros([img_size(1)*img_size(2),60]);
for i = 0:5
    for j = 1:10
        test_name =
"testset\subset"+int2str(i+6)+"\person_"+int2str(i+1)+"_"+int2str(j)+".jpg";
        if isfile(train_name)
            tests(:,idx) = reshape(imread(test_name),[img_size(1)*img_size(2),1]);
            idx = idx+1;
        end
    end
```

```
end
%%
% Calculate the average of all images in training set
Psi = sum(imgs,2)/size(imgs,2);
% Normalization
trains = imgs - Psi;
tests = tests - Psi;
% Calculate C' instad of C to reduce computation, C' has the same
% eigenvalue as C has
C_p = trains'*trains;

% Implement SVD on C'
[U,S,V] = svd(C_p);
%% (a)
% Find the singular value of data in training set
singulars = diag(S);

% Calculate the eigenfaces by U_i = A*v_i
eig_faces = trains*V;
% Find the 16 most representative eigenfaces
[~,idx] = sort(sum(eig_faces),'descend');
eig_faces_16 = eig_faces(:,idx(1:16));
% Find the 30 most representative eigenfaces
eig_faces_30 = eig_faces(:,idx(1:30));

% Normalize
eig_faces_16 = eig_faces_16./norm(eig_faces_16);

figure
for i = 1:16
    subplot(4,4,i)
    imagesc(reshape(eig_faces_16(:,i),img_size));
    colormap(gray(255));
    subtitle("Eigenface: #"+int2str(idx(i)));
end
sgtitle(['The Most 16 Representative Eigenfaces in ' ...
    'Training Set']);
%% (b)
% Calculate the mean and covariance of each class
mu = zeros(size(trains,1),6);
sigma = zeros(size(trains,1),size(trains,1),6);
for i = 0:5
    mu(:,i+1) = mean(trains(:,1+i*40:40+i*40),2);
    sigma(:,:,i+1) = cov(trains(:,1+i*40:40+i*40)');
end

% Calculate the LDA of each two classes among the six face classes
w = [];
for i = 1:5
    for j = i+1:6
        % The LDA matrix is singular, adding gamma = 1 when calculating
        % sigma, i.e. implementing the RDA instead of LDA
        LDA = inv(sigma(:,:,i)+sigma(:,:,j)+eye(2500))*(mu(:,i)-mu(:,j));
        w = [w, LDA];
    end
```

```
end

% Plot the LDA of each two classes among the six face classes
i = 1;
j = i+1;
figure
for n = 1:15
    subplot(4,4,n)
    imagesc(reshape(w(:,n),img_size));
    colormap(gray(255));
    subtitle("LDA: class #"+ i +" v.s. class #" + j);
    if j == 6 && i <= 5
        i = i+1;
        j = i+1;
    else
        j = j+1;
    end
end

%% (c)
train_PCA = eig_faces_16(:,1:15)'*trains;
mu = zeros(15,6);
sigma = zeros(15,15,6);
test_PCA = eig_faces_16(:,1:15)'*tests;

[err_PCA,counts_PCA] = PoE(train_PCA,test_PCA);
disp("The probability of error of each class is: "+int2str(err_PCA));
disp("The # of error of each class are: "+int2str(counts_PCA));
disp("The overall error rate of all classes is: " + sum(counts_PCA)/60);

% (d)
train_LDA = w'*trains;
mu = zeros(15,6);
sigma = zeros(15,15,6);
test_LDA = w'*tests;

[err_LDA,counts_LDA] = PoE(train_LDA,test_LDA);
disp("The probability of error of each class is: "+int2str(err_LDA));
disp("The # of error of each class are: "+int2str(counts_LDA));
disp("The overall error rate of all classes is: " + sum(counts_LDA)/60);

%% (e)
train_P_L = eig_faces_30(:,1:30)'*trains;
test_P_L = eig_faces_30(:,1:30)'*tests;
mu_P_L = zeros(30,6);
sigma_P_L = zeros(30,30,6);
for i = 0:5
    mu_P_L(:,i+1) = mean(train_P_L(:,1+i*40:40+i*40),2);
    sigma_P_L(:,:,i+1) = cov(train_P_L(:,1+i*40:40+i*40)');
end
w_P_L = [];
for i =1:5
    for j = i+1:6
        LDA = inv(sigma_P_L(:,:,i)+sigma_P_L(:,:,j)+eye(30))*(mu_P_L(:,i)-
mu_P_L(:,j));
```

```
        w_P_L =[w_P_L, LDA];
    end
end

[err_P_L,counts_P_L] = PoE(w_P_L'*train_P_L, w_P_L'*test_P_L);

disp("The probability of error of each class is: "+int2str(err_P_L));
disp("The # of error of each class are: "+int2str(counts_P_L));
disp("The overall error rate of all classes is: " + sum(counts_P_L)/60);
%%
% PoE contains a guassian classifier
function [errors,counts] = PoE(train, test)
    mu = zeros(size(train,1),6);
    sigma = zeros(size(train,1),size(train,1),6);

    % calculate the mu and sigma for mvnpdf
    for i = 0:5
        mu(:,i+1) = mean(train(:,1+i*40:40+i*40),2);
        sigma(:,:,i+1) = cov(train(:,1+i*40:40+i*40)');
    end

    % implement multivariate guassian distribution
    p = zeros(6,60);
    for i = 1:6
        for j = 1:60
            p(i,j) = (test(:,j)-mu(:,i))'*((sigma(:,:,i))\(test(:,j)-
mu(:,i)))+log(det(sigma(:,:,i)));
        end
    end

    [~,ind] = min(p);
    counts = zeros(1,6);
    for i = 1:60
        % count the missed classification
        if ind(i) ~= ceil(i/10)
            counts(ceil(i/10)) = counts(ceil(i/10))+1;
        end
    end
    errors = counts*10;
end
```