
Table of Contents

.....	1
Setup	1
Part 4b, c, and d:	1
Part 4e:	3
Part 5a: compute PCA of the data and make a 4x4 plot showing 16 principal components of largest variance	3
Part 5b	4
Part C	6
Part 5d	8
Part 5e	8
functions	9

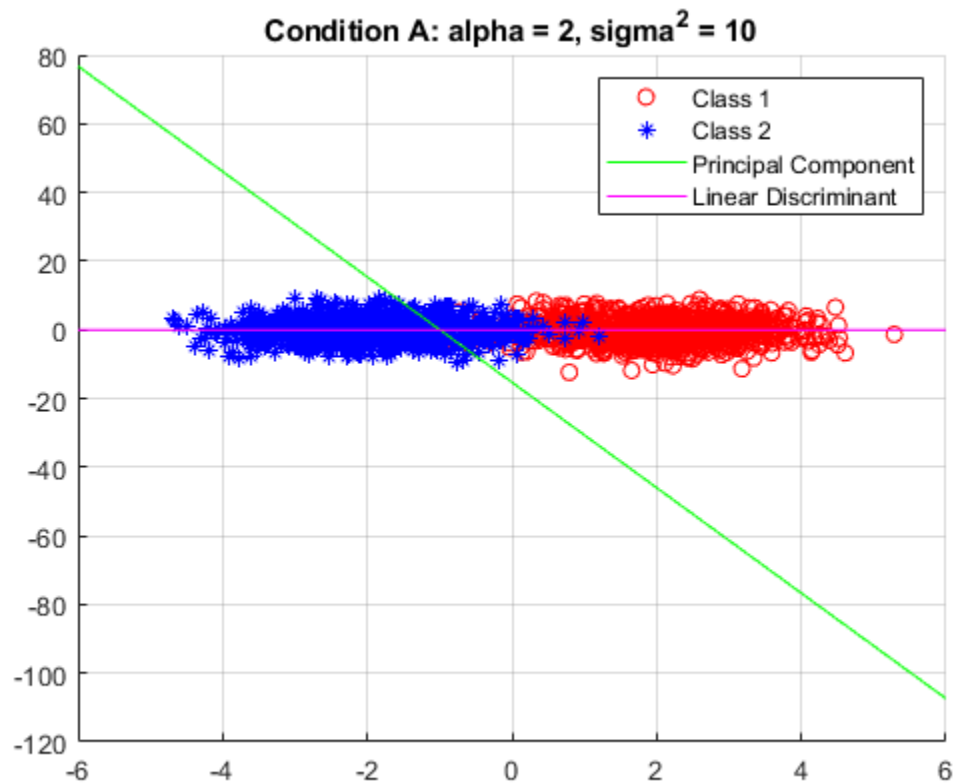
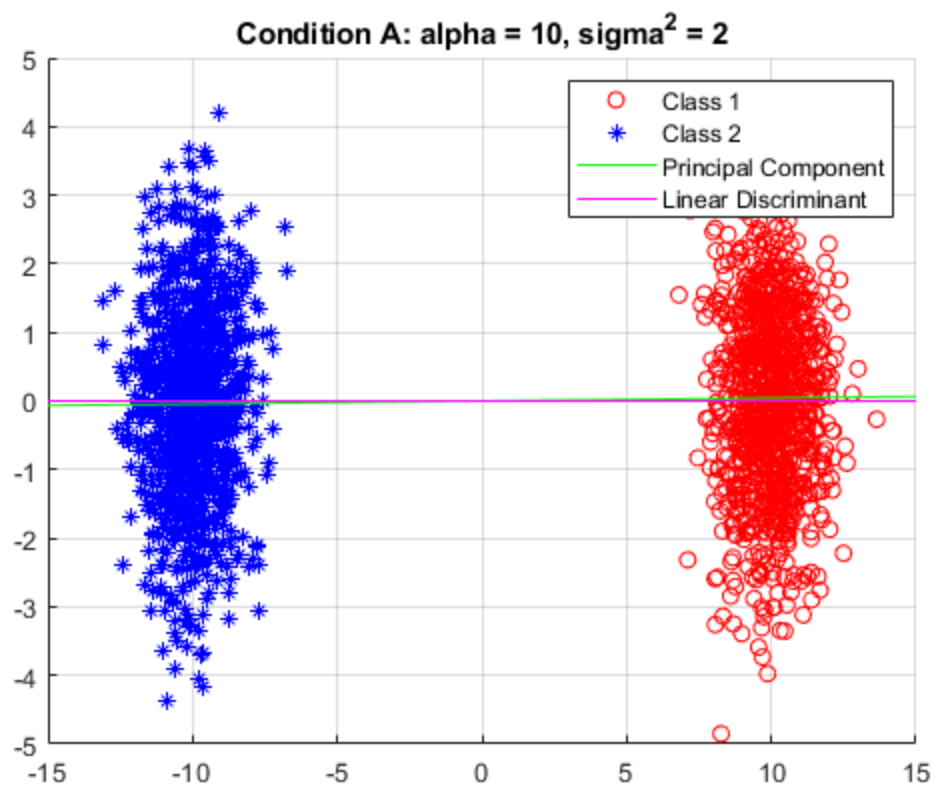
%Andy Nguyen
%1/18/2024
%ECE 271B

Setup

```
close all; clear; clc;
```

Part 4b, c, and d:

```
[M1, E1, N1, w1] = part4bcd(10, 2, 'Condition A: alpha = 10, sigma^2 = 2');  
[M2, E2, N2, w2] = part4bcd(2, 10, 'Condition A: alpha = 2, sigma^2 = 10');
```



Part 4e:

% approach to dimensionality reduction is not always a good one. It is possible for the PCA to throw away all the discriminant info. Although we are keeping the dimensions of largest variance, we might run the risk of throwing away important information stored in the other dimensions. In terms of classification, LDA seems to perform better because it projects the data in a way that maximizes the distance between classes. In the example plots, we can see that the PCA and LDA lines up with the sigma being less than the means (condition A). This shows that PCA and LDA performance are similar here. However, in condition B, when the sigma was greatly larger than the means, we can see that LDA was better at showing the dimension that better separated the data.

Part 5a: compute PCA of the data and make a 4x4 plot showing 16 principal components of largest variance

Get training set

```
mainDir = 'X:\ECE\ECE271B\hw1\trainset';
[folderList, ~] = extractImages(mainDir);
numImages = sum(cellfun(@length, folderList));
%reshape 50x50 matrix into 2500 dimensional vector and add to final matrix
X = zeros(2500, numImages);
imageIndex = 1;
for j = 1:length(folderList)
    for i = 1:length(folderList{j})
        imgVector = reshape(folderList{j}{i}, [2500, 1]);
        X(:, imageIndex) = imgVector;
        imageIndex = imageIndex + 1;
    end
end
%make values of A from 0 to 1
X = X/255.0;
X_mean = sum(X, 2)/size(X,2);
X_centered = X-X_mean;
[M, E, N] = svd(X_centered');

top16PC = diag(E(1:16, 1:16));
%create 4x4 plot
figure();
for i = 1:16
    subplot(4,4,i)
    eigenface = mat2gray(reshape(N(:,i), [50,50]));
    imshow(eigenface)
    title(sprintf('PC %i', i))
end
```



Part 5b

```
figure();
plotIndex = 1;
phi30 = N(:, 1:30); %PCA of 30 principal components (2500 x 30)
for class1Index = 1:length(folderList)-1
    for class2Index = class1Index+1:length(folderList)
        A = zeros(2500, length(folderList{class1Index}));
        B = zeros(2500, length(folderList{class2Index}));

        for i = 1:length(folderList{class1Index})
            imgVectorA = reshape(folderList{class1Index}{i}, [2500, 1]);
            A(:, i) = imgVectorA;
            imgVectorB = reshape(folderList{class2Index}{i}, [2500, 1]);
            B(:, i) = imgVectorB;
        end
        A = A/255.0; % 2500 x 40
        B = B/255.0;
        A_mean = sum(A, 2) / size(A, 2);
        B_mean = sum(B, 2) / size(B, 2);
        A_cov = A*A';
        B_cov = B*B';
        %RDA where gamma =1
        gamma = 1;
        S_w = A_cov + B_cov + gamma*eye(size(A_cov));
```

```

%      S_b = (A_mean - B_mean) * (A_mean - B_mean)';
w(:, plotIndex) = inv(S_w)*(A_mean - B_mean);
subplot(4,4, plotIndex)
eigenface_b = mat2gray(reshape(w(:, plotIndex), [50, 50]));
imshow(eigenface_b)
hold on;

title(sprintf("Class %i vs. Class %i", class1Index, class2Index))
%      legend(sprintf("Class %i", class1Index), sprintf("Class %i",
class2Index))

%Part 5e code:
A_PCA = phi30'*A; %30 dimensions x 40 features
B_PCA = phi30'*B; %30 x 40
A_PCA_mean = sum(A_PCA, 2) / size(A_PCA,2); % 30 x 1
B_PCA_mean = sum(B_PCA, 2) / size(B_PCA,2);
A_PCA_cov = A_PCA*A_PCA'; %30 x 30
B_PCA_cov = B_PCA*B_PCA';
S_w_PCA = A_PCA_cov + B_PCA_cov; %30 x 30
w_PCA(:, plotIndex) = inv(S_w_PCA)*(A_PCA_mean-B_PCA_mean); %30 x 15
(when loop is done)

plotIndex = plotIndex + 1;
end
end

```

Class 1 vs. Class 2 Class 1 vs. Class 3 Class 1 vs. Class 4 Class 1 vs. Class 5



Class 1 vs. Class 6 Class 2 vs. Class 3 Class 2 vs. Class 4 Class 2 vs. Class 5



Class 2 vs. Class 6 Class 3 vs. Class 4 Class 3 vs. Class 5 Class 3 vs. Class 6



Class 4 vs. Class 5 Class 4 vs. Class 6 Class 5 vs. Class 6



Part C

```
phi = N(:, 1:16);

for class1Index = 1:length(folderList)
    X = zeros(2500, length(folderList{class1Index}));
    for i = 1:length(folderList{class1Index})
        imgVectorX = reshape(folderList{class1Index}{i}, [2500, 1]);
        X(:, i) = imgVectorX;
    end
    X = double(X) / 255.0;
    z_PCA = phi'*X;
    z_LDA = w'*X;
    %for PCA
    mu_classPCA{class1Index} = mean(z_PCA,2);
    z0_PCA = z_PCA - mu_classPCA{class1Index};
    sigma_classPCA{class1Index} = (z0_PCA*z0_PCA') / (size(z_PCA,2)-1);

    %for LDA
    mu_classLDA{class1Index} = mean(z_LDA, 2);
    z0_LDA = z_LDA - mu_classLDA{class1Index};
    sigma_classLDA{class1Index} = (z0_LDA*z0_LDA') / (size(z_LDA,2)-1);

    %for PCA + LDA (k = 30)
    z_PCA_LDA = w_PCA'*phi30'*X;
```

```

mu_class_PCA_LDA{class1Index} = mean(z_PCA_LDA, 2);
z0_PCA_LDA = z_PCA_LDA - mu_class_PCA_LDA{class1Index};
sigma_class_PCA_LDA{class1Index} = (z0_PCA_LDA*z0_PCA_LDA') /
(size(z0_PCA_LDA, 2)-1);

end

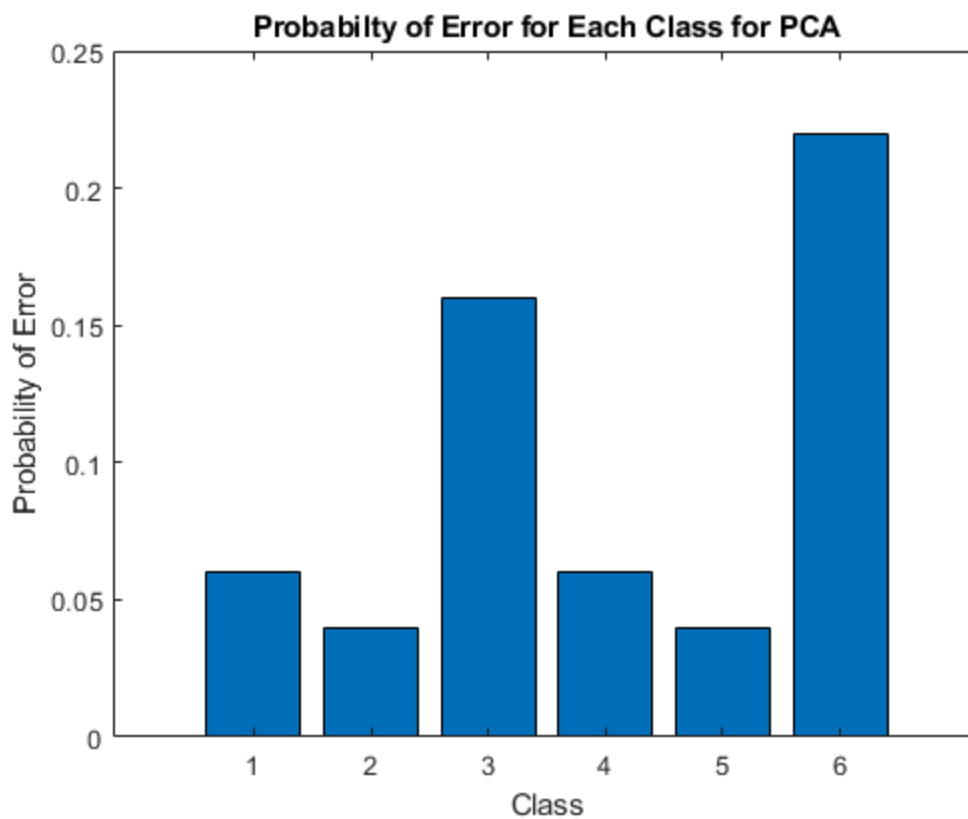
%iterate through test folder list
testDir = 'X:\ECE\ECE271B\hw1\testset';
[testFolderList, testImgNames] = extractImages(testDir);

[averageProbErrorPCA, probErrorPCA] =
calc_prob_error_gauss_classifier(mu_classPCA, sigma_classPCA, phi,
testFolderList, testImgNames);
title('Probabilty of Error for Each Class for PCA')
averageProbErrorPCA

```

averageProbErrorPCA =

0.0967

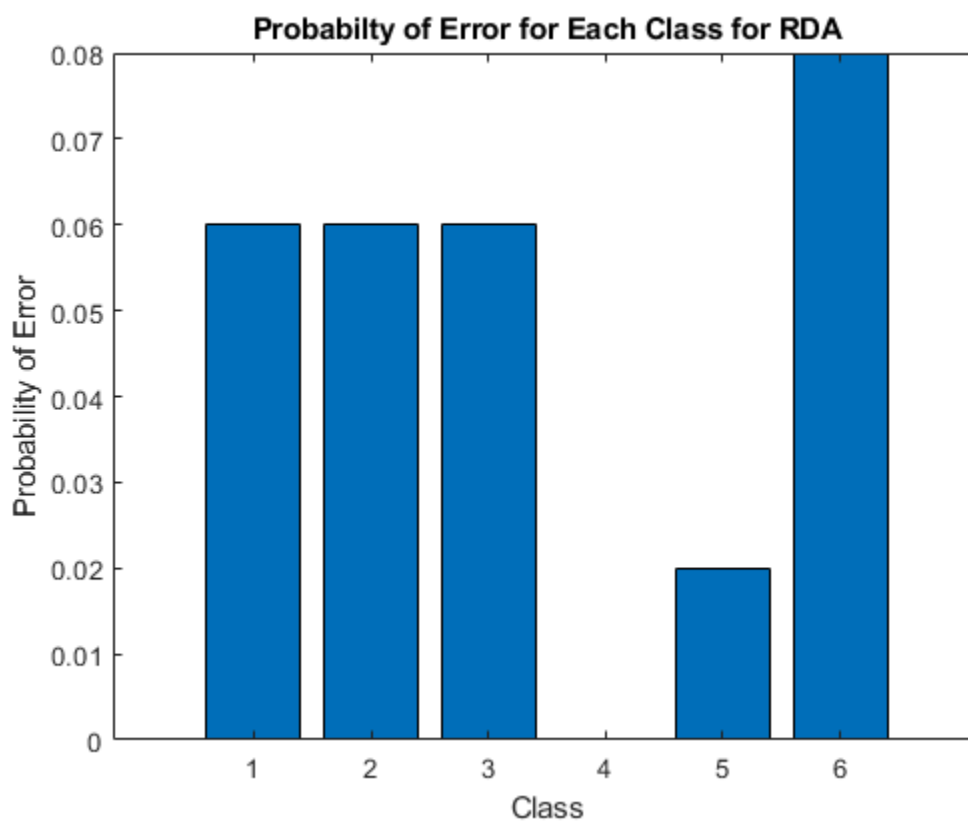


Part 5d

```
[averageProbErrorRDA, probErrorLDA] =  
    calc_prob_error_gauss_classifier(mu_classLDA, sigma_classLDA, w,  
    testFolderList, testImgNames);  
title('Probability of Error for Each Class for RDA')  
averageProbErrorRDA
```

```
averageProbErrorRDA =
```

```
0.0467
```



Part 5e

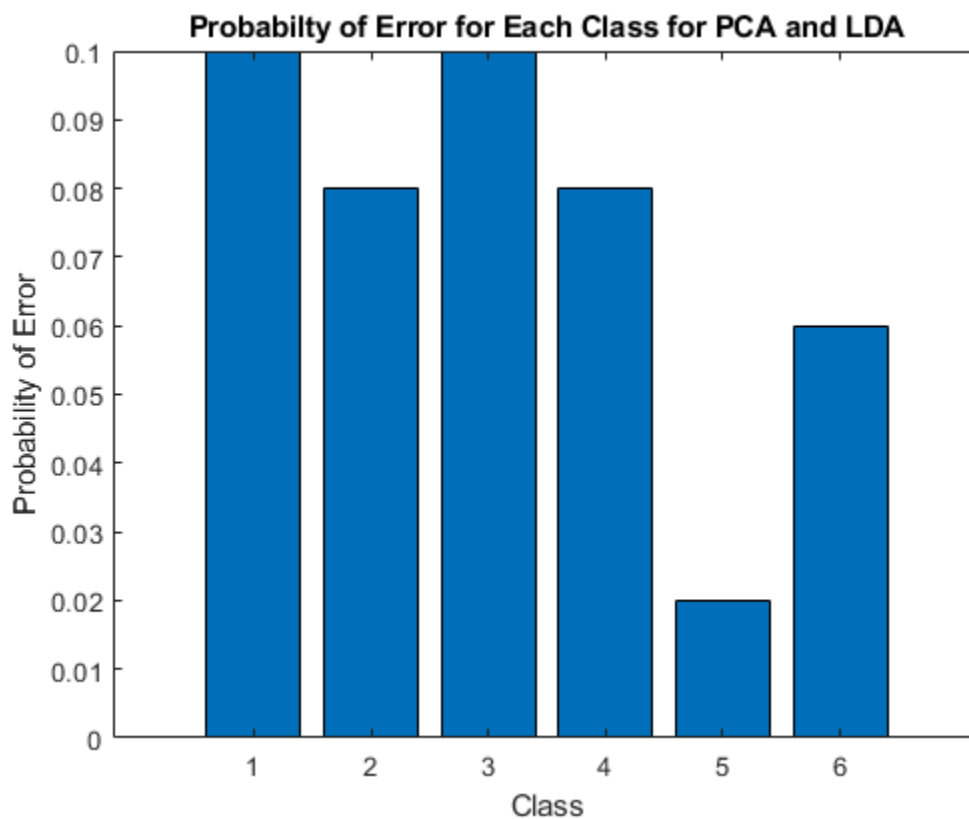
From these three graphs, we can see that just doing PCA performed the worst. RDA seems to have performed the best and PCA + LDA seems to be second. We can see that although PCA can help clean up the data and reduce dimensionality, LDA can be more beneficial in terms of this gaussian classifier example. RDA is good when we need to address singular scatter matrices. PCA + LDA is also a good alternative to address singular covariances. In other examples, PCA might be more beneficial as a technique, however, in the gaussian classifier case, it seems that RDA outperforms the other techniques.

```
[averageProbError_PCA_and_LDA, probError_PCA_and_LDA] =  
    calc_prob_error_gauss_classifier(mu_class_PCA_LDA, sigma_class_PCA_LDA,  
    phi30*w_PCA, testFolderList, testImgNames);
```

```
title('Probability of Error for Each Class for PCA and LDA')
averageProbError_PCA_and_LDA
```

```
averageProbError_PCA_and_LDA =
```

```
0.0733
```



functions

```
function [] = plotLine(pointA, pointB, color)
%plots infinite line from pointA to pointB

xlim = get(gca, 'XLim');
m = (pointB(2)-pointA(2))/(pointB(1)-pointA(1));
n = pointB(2)*m - pointA(2);
y1 = m*xlim(1) + n;
y2 = m*xlim(2) + n;
line([xlim(1) xlim(2)], [y1 y2], 'Color', color)

end

function [M, E, N, w] = part4bcd(alpha, sigma, titleString)

%part 4b
```

```

mu1 = [alpha; 0];
mu2 = -mu1;
Gamma_cov = [1, 0; 0, sigma];
numSamples = 1000;
samplesClass1 = mvnrnd(mu1, Gamma_cov, numSamples);
samplesClass2 = mvnrnd(mu2, Gamma_cov, numSamples);
figure();
hold on;
grid on;
title(titleString)
plot(samplesClass1(:,1), samplesClass1(:,2), 'ro')
plot(samplesClass2(:,1), samplesClass2(:,2), 'b*')
%part 4c
X = [samplesClass1', samplesClass2'];
mean_X = sum(X,2)/size(X,2);
Xc = X - mean_X;
[M, E, N] = svd(Xc');
plotLine([0,0], [N(1,1), N(1,2)], 'g')

%part 4d

w = inv(Gamma_cov + Gamma_cov)*(mu2 - mu1);
plotLine([0,0], w, 'm')
legend('Class 1', 'Class 2', 'Principal Component', 'Linear Discriminant')

end

function min_i = gauss_classify(phi, imageVector, class_means, class_sigmas)
min_i = -1; %not valid
min_prob = inf;
for index = 1:length(class_means)
    prob = (phi'*imageVector -
        class_means{index})'*pinv(class_sigmas{index})*(phi'*imageVector -
        class_means{index})+log(det(class_sigmas{index}));
    if (prob < min_prob)
        min_prob = prob;
        min_i = index;
    end
end
end
end

function [averageProbError, probError] =
    calc_prob_error_gauss_classifier(mu_class, sigma_class, phi, testFolderList,
    testImgNames)
totalClassImg = zeros(6,1);
correctClassCount = zeros(6,1);
for n = 1:length(testFolderList)
    for m = 1:length(testFolderList{n})
        imgVector = double(reshape(testFolderList{n}{m}, [2500, 1])) / 255.0;
        testImgNames{n}{m};
        label = str2double(testImgNames{n}{m}(8));
        min_i = gauss_classify(phi, imgVector, mu_class, sigma_class);
        totalClassImg(label) = totalClassImg(label) + 1;
        if (label == min_i)

```

```
        correctClassCount(label) = correctClassCount(label) + 1;
    end
end
end

probError = zeros(6,1);
figure()
for i = 1:length(totalClassImg)
    probError(i) = 1-correctClassCount(i) / totalClassImg(i);
end
bar([1:6], probError);
title('Probabilty of Error for Each Class')
ylabel('Probability of Error')
xlabel('Class')
averageProbError = 1-sum(correctClassCount) / sum(totalClassImg);
end
```

Published with MATLAB® R2022b