EEEM007 Advanced Signal Processing
LAB EXPERIMENT: PATTERN RECOGNITION

By Maxence Boels

Professor M D Plumbley

May 11, 2020

1 INTRODUCTION

Those experiments are carried out to obtain empirical results to be compared to the theoretical findings in the first Assignment of this module. This assignment is developed in Matlab and the code is available in the appendix. The following experiments are well detailed in 4 chapters. First, we consider the Gaussian and k-Nearest Neighbors classifiers. Secondly, the relation between the reliability of the error and the size of the test set are demonstrated. As third experiment, the dependence of the test error on the dimensionality of the pattern recognition problem are graphically illustrated and commented. Finally, the relation between the class separability and the error probability is discussed.

Contents

2 EXPERIMENT 1

For this experiment 2 classifiers are used, a Gaussian Classifier and a k-Nearest Neighbour Classifier.

2.1 Part A: Gaussian Classifier

2.1.1 Description of experiment, objectives, design choices

The objective is to experiment the effect of the training sample size on the classifier performance and the effect of the size of the test set on the confidence of the test error estimated.

The Gaussian Classifier finds the best classification boundary with the Bayes Error calculated on the training sample. For this experiment, different training sample size have been selected as shown in table 2.1.

TRAINING SAMPLE	N1	N2	N3	N4	N5
SAMPLE SIZE	3	5	10	50	100

Table 2.1 - Training Samples

Error Design Estimate: The testing is performed with the same training samples used to design i.e. train the classifier.

Error Test Estimate: The testing is performed with all the observation in the testing set i.e. all samples.

Note that, for each experiment, 10 independent classifiers are created, and the average error is computed.

2.1.2 Predicted experimental outcome from the theory

Firstly, the *design error* corresponds to the maximum performance the classifier can achieve.

The second intuition is to say that the confidence of *test error* decreases with the test sample size as we can rely on less observations to evaluate the result.

Finally, the *test error* of each classifier will normally converge to the full sample design error as the number of samples increase in the test set.

2.1.3 Presentation of the experimental results obtained

As depicted on the Fig 2.1, the average *design errors* and *test errors* converge as the number of samples increases. After converging at 10 samples, the 2 errors reach a plateau at 0.11 and remain steady.

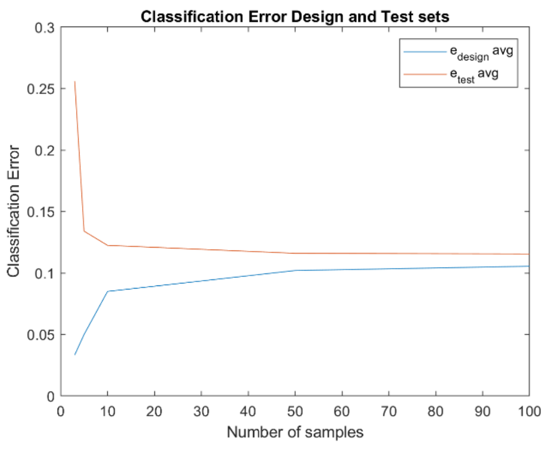


Figure 2.1 - Design and test errors

2.1.4 Analysis of results, discussion, and conclusion

One can conclude that when training a classifier with a small sample (less than 10), and test it with the same sample, the *design error* is significantly smaller than the *test error* with the same sample size. After 10 observations per sample, both errors are significantly not separable.

Training size: Increasing the sample size during training gives a better test error, closer to the optimal Bayes Error.

Class separability: the design error becomes more accurate as the number of samples increases. Thus, the separability between both classes increases slightly with smaller number of samples. This can be explained with the Central Limit Theorem with more probability to have observations close to the mean.

Theoretical error: the Bayes Error calculated theoretically in assignment 1 is 0.11, and like the empiric results illustrated on Figure 2.1.

2.2 Part B: k-Nearest Neighbour Classifier

2.2.1 Description of experiment, objectives, design choices

We want to compare the previous results with another classifier, namely k-Nearest Neighbour which uses the k closest observations from a point to attribute the majority class to the unlabelled point. In this experiment, we use k from 1 to 51.

The objective is to use the same methodology as in Part A but changing the Gaussian Classifier to a KNN Classifier and comparing the results to the Bayes Error. For this experience there is no boundary, but instead using the distance between observations to attribute a class label to new predicted points.

2.2.2 Predicted experimental outcome from the theory

The first intuition is to say that *if k is small*, new observations x from class A on the correct side of the optimal Bayes decision boundary might take the opposite class label B if x is next to a few class B observations (sensitive to outliers). Thus, reinforcing a more *heterogeneous* distribution.

On the other hand, if *k is high, then the classifier will encourage a homogeneous distribution and remove the fine grained "details" near the Bayes boundary.*

Regarding the design, test, and Bayes error, we can expect to have also

2.2.3 Presentation of the experimental results obtained

Comparing the KNN classifier with the gaussian classifier, one can observe the larger spread in design and test errors for small sample sizes i.e. bellow 10 samples per class.

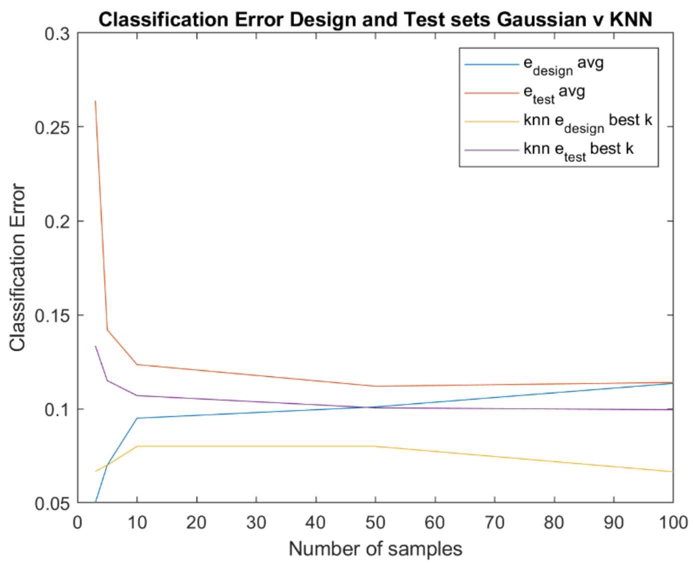


Figure 2.2 – Test and Design Error with Gaussian v KNN

2.2.4 Analysis of results, discussion, and conclusion

Thus, KNN classifiers are less sensitive to small samples than gaussian classifiers and have a better estimation of the Bayes Error with small samples, but are comparable for larger samples, reaching the same plateau as the Gaussian Classifier, both converging around the Bayes Error. This difference may be caused by the fact that KNNs have more parameters for selecting the best error i.e. using the best k value.

3 EXPERIMENT 2

In this experiment, the effect of the size of the test set on the reliability of the error is empirically explored.

3.1 Part A: Mean and Variance

3.1.1 Description of experiment, objectives, design choices

In this experiment, the *test error* is calculated with different training samples as given in Table 3.1.

TRAINING SAMPLE	N1	N2	N3	N4	N5
SAMPLE SIZE	3	5	10	50	100

Table 3.1 - Training samples

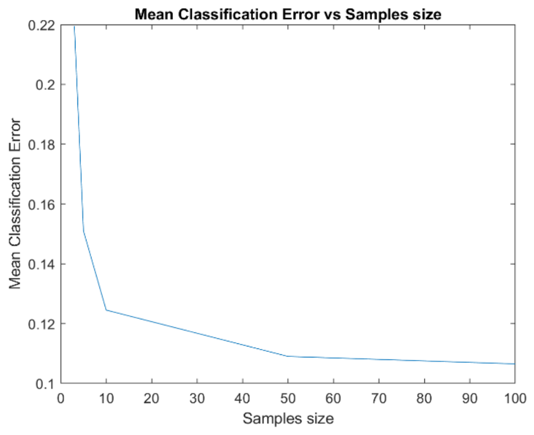
Similarly, to experiment 1, an estimated test error is calculated based on 5 different classifiers. Here, the mean and variance are computed by averaging 10 independent test sets, one for each sample size.

3.1.2 Predicted experimental outcome from the theory

One can expect the reliability of the empirical error and the classification performance to increase with the sample size.

3.1.3 Presentation of the experimental results obtained

Figure 3.1 and 3.2 illustrate the *mean test error* and *mean variance test error* on sample size, respectively. Both figures depict a sharp fall and remain steady with increasing sample sizes.



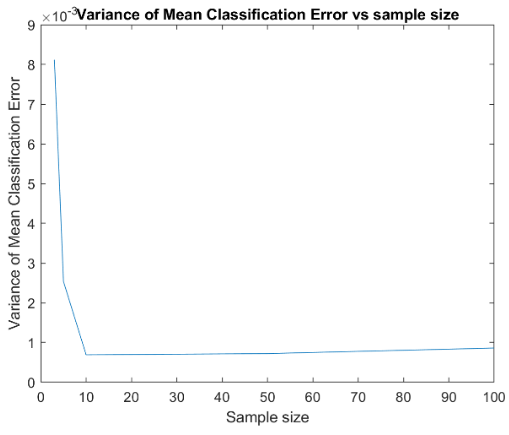


Figure 3.1 - Mean Test Error, Figure 3.2 - Mean Variance of Test Error

3.1.4 Analysis of results, discussion, and conclusion

One can conclude that the classification error decreases with a larger number of samples used for training. Likewise, the variance in the classification error decreases with a growing number of samples. This can be explained by a more accurate prediction when using more observations, they tend to be centred around the mean. There is a clear elbow at sample size equal to ten. The sample size influences the variance and test error, especially before 10 samples.

3.2 Part B: Train and Test sets

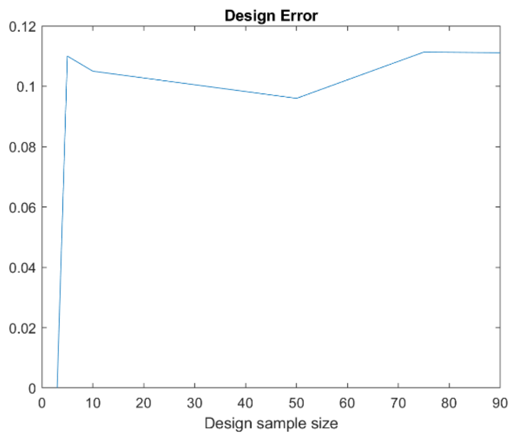
For this experiment, the total number of samples per class used for training and testing sums to 100. The objective is to find which combination performs best for design and test errors.

Train set	*Test set*	*Design Error*	*Test Error*
3	97	0.100	0.190
5	95	0.060	0.132
10	90	0.090	0.126
50	50	0.116	0.114
75	25	0.109	0.116
90	10	0.103	0.125

Table 3.2 - Design and Test Errors

As illustrated in the Table 3.2, the lowest *test error* is 0.114 with 50 training samples and 50 test samples.

In Table 3.3, the design error is underestimated for small samples and reaches the Bayes Error after 5 samples for each class. This is explained by the fact that in small sampling, the probability of picking False Negatives and False Positives within the design set is small. Whereas, the test error is overestimated when training the classifier with few samples and testing with many. The boundary is not optimal when designing with small samples.



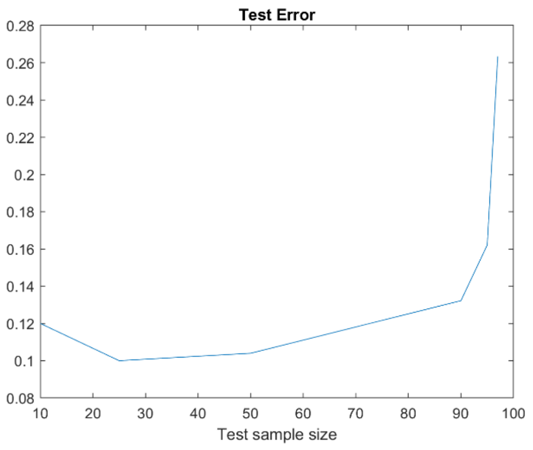


Table 3.3 - Design and Test Errors

4 EXPERIMENT 3

4.1.1 Description of experiment, objectives, design choices

In this experiment, the relation between the test error and dimensionality of the pattern recognition problem is highlighted and compared with the True Error. We consider this error to be estimated with 500 samples for "high" dimensionality problems.

4.1.2 Predicted experimental outcome from the theory

For this experiment, we can expect to see the classification error increase as the dimensionality grow with a given number of samples. The data is spread out and decreases the classifier's predictive ability.

4.1.3 Presentation of the experimental results obtained

The True Classification Error has been approximated with 500 samples and is expected to be close to 0.06%. As depicted on Figure 4.1, smaller dimensionality problems need less samples to reach the True Classification Error. The blue curb with 5 dimensions drops faster to 0.06 than the yellow line which needs 180 samples to obtain the same classification error. The True Error is reached among all dimension sizes with around 180 samples as they reach a plateau after this point.

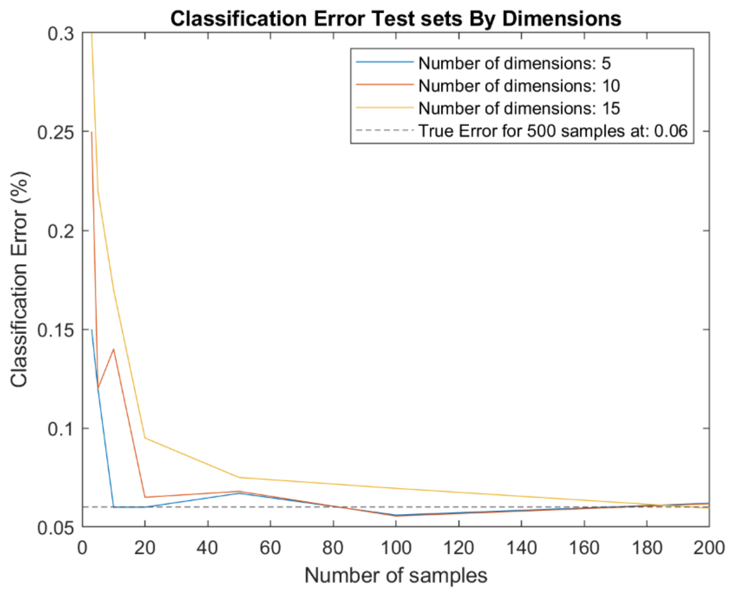


Figure 4.1 - Classification Error by samples and Dimensions

4.1.4 Analysis of results, discussion, and conclusion

One can conclude that the number of samples should increase as the dimensionality of the pattern recognition problem expands. It can be explained with the "curse of dimensionality" principle which refers to a spread of the data among the feature space and makes it more difficult to find class clusters with a fixed dataset.

5 EXPERIMENT 4

5.1.1 Description of experiment, objectives, design choices

As final experiment, we explore the relationship between class separability and error probability by arbitrary choosing the dimensionality and generating a sequence of sets of 2 normally distributed classes. For this experiment, the Mahalanobis distance between the means of both classes in each set increases monotonically. We want to observe how it impacts the Classification Error.

5.1.2 Predicted experimental outcome from the theory

As the distance between both classes' means increases, we can expect the classification error to decrease, since the overlapping area that can lead to false negatives and false positives shrinks. This is also true for high dimensional problems as illustrated in Experiment 3.

5.1.3 Presentation of the experimental results obtained

Figure 5.1 depicts a fall in the classification error as the Mahalanobis distance between class means grows and reaches its best score with a Mahalanobis distance of 3. One can also observe the curvature which means that more distance is needed when the error is close to zero.

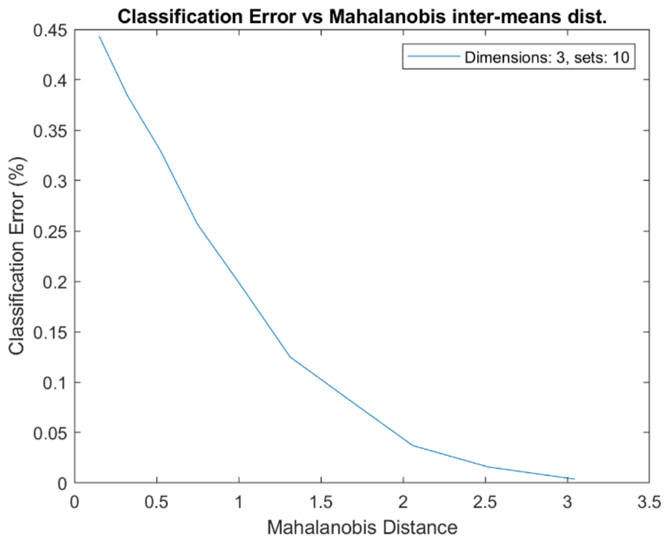


Figure 5.1 - Classification Error vs Mahalanobis dist. between class means

Monotonically increase function: $f(x) = a^x$ with a = 1.15 and x = [1:10].

Sets	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
M. dist.	0.15	0.32	0.52	0.74	1.01	1.31	1.66	2.05	2.51	3.04

Table 5.1 - Mahalanobis distance between sets

Figure 5.2 is another illustration that highlights the relation between the Mahalanobis distance and the sample size which after 30 samples per class in a 3-dimensional space reaches a plateau across all inter-mean Mahalanobis distance.

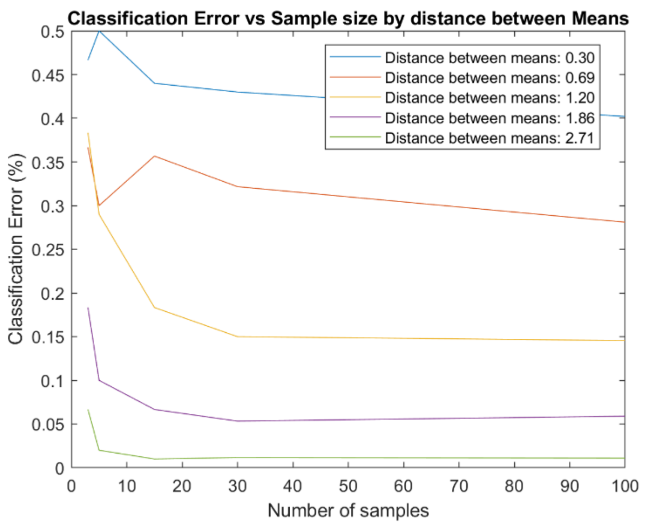


Figure 5.2 - Classification Error vs sample size by distance between means with 5 sets and 3 dimensions

Note that for this experiment, an identity covariance matrix was used, similarly as in the third experiment.

5.1.4 Analysis of results, discussion, and conclusion

To conclude, we can assert that the error probability decreases with the increasing class inter-means Mahalanobis distance in multidimensional feature spaces and that augmenting small samples size can increase the classifier's performance, especially when the dimensionality expands.

6 APPENDIX

6.1 EXP1:

```
sigma = [2 1; 1 2];
u1 = [1; 2]; % class 1
u2 = [4; 5]; % class 2
%number of samples from training set
Nd = [3,5,10,50,100];
for n = 1:length(Nd)
 for sample = Nd(n)
    for i = 1:10
    % create training set
    train1 = mvnrnd(u1, sigma, sample);
    train2 = mvnrnd(u2, sigma, sample);
    trainset = [train1; train2];
    % create test set
    test1 = mvnrnd(u1, sigma, 100);
    test2 = mvnrnd(u2, sigma, 100);
    testset = [test1; test2];
    % Create labels
    class1 = zeros(sample,1);
    class2 = ones(sample,1);
    labels = [class1 ; class2];
    labels_testset = [zeros(100,1); ones(100,1)];
    % Train classifier
    gaussian_classifier = fitcnb(trainset, labels);
    % Test with trainset as Design error
    y_pred_design = predict(gaussian_classifier, trainset);
    % Calculate design error for trial i
    error_design(i) = sum(y_pred_design ~= labels)/numel(y_pred_design);
    % Test with testset as test error
    y_pred_test = predict(gaussian_classifier, testset);
    % Calculate test error for trial i
    error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);
    end
  % Calculate average design error
  error_design_avg(n) = mean(error_design);
  % Calculate average test error
  error_test_avg(n) = mean(error_test);
  end
end
% KNN Classifier
sigma = [2 1; 1 2];
u1 = [1; 2]; % class 1
u2 = [4; 5]; % class 2
%number of samples from training set
Nd = [3,5,10,50,100];
% number of nearest neighbours
K = 1:2:51;
for j = 1:length(K)
for k = K(j)
    for n = 1:length(Nd)
     for sample = Nd(n)
        for i = 1:10
        % create training set
        train1 = mvnrnd(u1, sigma, sample); % class 1
        train2 = mvnrnd(u2, sigma, sample); % class 2
        trainset = [train1; train2];
        % create test set
        test1 = mvnrnd(u1, sigma, 100); % class 1
        test2 = mvnrnd(u2, sigma, 100); % class 2
        testset = [test1; test2];
        % Create labels
        class1 = zeros(sample,1);
        class2 = ones(sample,1);
        labels_trainset = [class1 ; class2];
        labels_testset = [zeros(100,1); ones(100,1)];
        % train classifier KNN
        Mdl = fitcknn(trainset, labels_trainset,'NumNeighbors',k ,...
        'NSMethod','exhaustive','Distance','minkowski',...
        'Standardize',1);
        % Test with trainset as Design error
        y_pred_design = predict(Mdl, trainset);
        % Calculate design error for trial i
        knn_error_design(i) = sum(y_pred_design ~= labels_trainset)/numel(y_pred_design);
        % Test with testset as test error
        y_pred_test = predict(Mdl, testset);
        % Calculate test error for trial i
        knn_error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);
        end
      % Calculate average design error
      knn_error_design_avg(n) = mean(knn_error_design);
      % Calculate average test error
      knn_error_test_avg(n) = mean(knn_error_test);
      end
    end
  % Calculate average design error for k
  all_error_design_avg_K(j,:) = knn_error_design_avg;
  % Calculate average test error for k
  all_error_test_avg_K(j,:) = knn_error_test_avg;
```

```
end
end
%minimum error for all k
error_design_avg_K = min(all_error_design_avg_K(2:end,:))
error_test_avg_K = min(all_error_test_avg_K(2:end,:))

figure()
plot(Nd, error_design_avg)
hold on
plot(Nd, error_test_avg)
hold on
plot(Nd, error_design_avg_K)
hold on
plot(Nd, error_test_avg_K)
legend('e_d_e_s_i_g_n_ avg','e_t_e_s_t_ avg', 'knn e_d_e_s_i_g_n_ best k', 'knn e_t_e_s_t_ best k')
xlabel('Number of samples')
ylabel('Classification Error')
title('Classification Error Design and Test sets Gaussian v KNN')
hold off
```

6.2 EXP2:

6.2.1 Part A:

```
% exp2
% In this experiment we shall investigate the effect of the size of test set on the reliability of the
% empirical error count estimator and we will also learn how to classify test and training samples.

% covariance matrix
sigma = [2 1; 1 2];

% means class 1 and 2
u1 = [1; 2];
u2 = [4; 5];

%number of samples from test patterns
Nt = [3,5,10,50,100];

for s = 1:length(Nt)
 for sample = Nt(s)
    % create 10 independent test sets
    for i = 1:10
     % create training set
     train1 = mvnrnd(u1, sigma, sample);
     train2 = mvnrnd(u2, sigma, sample);
     trainset = [train1; train2];

     % generate 10 independent test sets
     test1 = mvnrnd(u1, sigma, 100);
     test2 = mvnrnd(u2, sigma, 100);
     testset = [test1; test2];

     % Create labels for training (sample size) and test set.
     class1 = zeros(sample,1);
     class2 = ones(sample,1);
     labels_trainset = [class1 ; class2];
     labels_testset = [zeros(100,1); ones(100,1)];

     % Train classifier with trainset
     gaussian_classifier = fitcnb(trainset, labels_trainset);

     % Test with testset as test error
     y_pred_test = predict(gaussian_classifier, testset);
     % Calculate test error for trial i
     error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);
    end

    % Calculate average test error for each sample size.
    error_test_avg(s) = mean(error_test);
    % the variance is normalized by the number of observations-1 by default.
    error_test_var(s) = var(error_test);
  end
end

figure()
plot(Nt,error_test_avg)
xlabel('Samples size')
ylabel('Mean Classification Error')
title('Mean Classification Error vs Samples size')
hold off

figure()
plot(Nt, error_test_var)
xlabel('Sample size')
ylabel('Variance of Mean Classification Error')
title('Variance of Mean Classification Error vs sample size')
hold off
```

6.2.2 Part B:

```
% exp2B
% covariance matrix
sigma = [2 1; 1 2];
% means class 1 and 2
u1 = [1; 2];
u2 = [4; 5];
% Dataset = 100 observations per class?????????
% Nd training and Nt testing sample split.
Nd = [3, 5, 10, 50, 75, 90];
Nt = [97, 95, 90, 50, 25, 10];
for s = 1:length(Nd)
 for sample = Nt(s)
    for i = 1:10
    % create training set
    train1 = mvnrnd(u1, sigma, Nd(s));
    train2 = mvnrnd(u2, sigma, Nd(s));
    trainset = [train1; train2];
    % generate 10 independent test sets
    test1 = mvnrnd(u1, sigma, Nt(s));
    test2 = mvnrnd(u2, sigma, Nt(s));
    testset = [test1; test2];
    % Create labels for train and test sets
    class1_train = zeros(Nd(s),1);
    class2_train = ones(Nd(s),1);
    class1_test = zeros(Nt(s),1);
    class2_test = ones(Nt(s),1);
    labels_trainset = [class1_train ; class2_train];
    labels_testset = [class1_test ; class2_test];
    % Train classifier
    gaussian_classifier = fitcnb(trainset, labels_trainset);
    % Test with trainset as Design error
    y_pred_design = predict(gaussian_classifier, trainset);
    % Calculate design error for trial i
    error_design(i) = sum(y_pred_design ~= labels_trainset)/numel(y_pred_design);
    % Test with testset as test error
    y_pred_test = predict(gaussian_classifier, testset);
    % Calculate test error for trial i
    error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);
    end
% Calculate average design and test error for each sample size split.
error_design_avg(s) = mean(error_design);
error_test_avg(s) = mean(error_test);
end
end
figure()
% create a table with design error and test error:
plot(Nd,error_design_avg)
title('Design Error')
xlabel('Design sample size')
hold off
figure()
% create a table with design error and test error:
plot(Nt,error_test_avg)
title('Test Error')
xlabel('Test sample size')
```

6.3 EXP3

```
% exp3

clear all
% dimensions
dim = [5, 10, 15];

%number of samples from training set
Nd = [3,5,10,20,50,100,200];

% Aconstant per dimension
% Adjust based on trial and error

c1 = [1, 1, 1];
c2 = [2.4, 2, 1.8];

Nd_TrueError = 500;

for n = 1:length(dim)
 for d = dim(n)
     for s = 1:length(Nd)
        for sample = Nd(s)
            for i = 1:10
                % covariance matrix
                sigma = eye(d);

                % means class 1 and 2
                u1 = ones([d, 1]) * c1(n);
                u2 = ones([d, 1]) * c2(n);

                % create train set
                train1 = mvnrnd(u1, sigma, sample);
                train2 = mvnrnd(u2, sigma, sample);
                trainset = [train1; train2];

                % create test set
                test1 = mvnrnd(u1, sigma, sample);
                test2 = mvnrnd(u2, sigma, sample);
                testset = [test1; test2];

                % Create labels train and test sets
                labels_trainset = [zeros(sample,1) ; ones(sample,1)];
                labels_testset = [zeros(sample,1); ones(sample,1)];

                % Train classifier
                gaussian_classifier = fitcnb(trainset, labels_trainset);

                % Test with testset as test error
                y_pred_test = predict(gaussian_classifier, testset);
                % Calculate test error for trial i
                error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);
            end
          % Calculate average test error
          error_test_avg(s) = mean(error_test);
        end
     end
    error_test_avg_dim(n,:) = error_test_avg;
 end
end


figure()
for n = 1:length(dim)
 for d = dim(n)
    plot(Nd, error_test_avg_dim(n,:))
    hold on
 end
end

TrueError = 0.06;
xlabel('Number of samples')
ylabel('Classification Error (%)')
yline(0.06, '--');
title('Classification Error Test sets By Dimensions')
legend(sprintf('Number of dimensions: %d', dim(1)), ...
       sprintf('Number of dimensions: %d', dim(2)), ...
       sprintf('Number of dimensions: %d', dim(3)), ...
       sprintf('True Error for 500 samples at: %.2f', TrueError))
```

6.4 EXP4:

```
%exp4
% Increasing the class separbility
clear all

dim= 3;
sets = 10;

% class separability multiplication factors
c1 = 1;
%c2 = a^x;

% monotonically increasing function f(x) = a^x with a > 1
a = 1.15;

% Sets
Nd = 500;

% covariance matrix
sigma = eye(dim);

for x = 1:sets
            for i = 1:10

            % means class 1 and 2
            u1 = ones([dim, 1]) * c1;
            u2 = ones([dim, 1]) * a^x;

            % create train set
            train1 = mvnrnd(u1, sigma, Nd);
            train2 = mvnrnd(u2, sigma, Nd);
            trainset = [train1; train2];

            % create test set
            test1 = mvnrnd(u1, sigma, Nd);
            test2 = mvnrnd(u2, sigma, Nd);
            testset = [test1; test2];

            % Create labels train and test sets
            labels_trainset = [zeros(Nd,1) ; ones(Nd,1)];
            labels_testset = [zeros(Nd,1); ones(Nd,1)];

            % Train classifier
            gaussian_classifier = fitcnb(trainset, labels_trainset);

            % Test with testset as test error
            y_pred_test = predict(gaussian_classifier, testset);
            % Calculate test error for trial i
            error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);

            end
        % Calculate average test error
        error_test_avg(x) = mean(error_test);
end
```

```
for x = 1:sets
    distance(x) = a^x-c1;
end

figure()
plot(distance, error_test_avg)
xlabel('Mahalanobis Distance')
ylabel('Classification Error (%)')
title('Classification Error vs Mahalanobis inter-means dist.')
legend(sprintf('Dimensions: %d, sets: %d', dim, sets))



%exp4 B
% Increasing the class separability
clear all

dim= 3;
sets = 5;

% class separability multiplication factors
c1 = 1;
%c2 = a^x;

% monotonically increasing function f(x) = a^x with a > 1
a = 1.3;

% Sets
Nd = [3, 5, 15, 30, 100];

% covariance matrix
sigma = eye(dim);

for x = 1:sets
    for s = 1:length(Nd)
        for sample = Nd(s)
            for i = 1:10

            % means class 1 and 2
            u1 = ones([dim, 1]) * c1;
            u2 = ones([dim, 1]) * a^x;

            % create train set
            train1 = mvnrnd(u1, sigma, sample);
            train2 = mvnrnd(u2, sigma, sample);
            trainset = [train1; train2];

            % create test set
            test1 = mvnrnd(u1, sigma, sample);
            test2 = mvnrnd(u2, sigma, sample);
            testset = [test1; test2];

            % Create labels train and test sets
            labels_trainset = [zeros(sample,1) ; ones(sample,1)];
            labels_testset = [zeros(sample,1); ones(sample,1)];

            % Train classifier
            gaussian_classifier = fitcnb(trainset, labels_trainset);

            % Test with testset as test error
            y_pred_test = predict(gaussian_classifier, testset);
            % Calculate test error for trial i
            error_test(i) = sum(y_pred_test ~= labels_testset)/numel(y_pred_test);

            end
        % Calculate average test error
        error_test_avg(s) = mean(error_test);
        end
    end
error_test_avg_dim(x,:) = error_test_avg;
end

figure()
for n = 1:length(Nd)
    plot(Nd, error_test_avg_dim(n,:))
    hold on
end

xlabel('Number of samples')
ylabel('Classification Error (%)')
x = 1:sets;
title('Classification Error vs Sample size by distance between Means')
legend(sprintf('Distance between means: %.2f', a^x(1)-c1), ...
       sprintf('Distance between means: %.2f', a^x(2)-c1), ...
       sprintf('Distance between means: %.2f', a^x(3)-c1), ...
       sprintf('Distance between means: %.2f', a^x(4)-c1), ...
       sprintf('Distance between means: %.2f', a^x(5)-c1))
```