EEEM007: Advanced Signal Processing

Lab Experiments: Pattern Recognition

Author(s):
Kausthub Krishnamurthy

URN:
6562233

Page Count:
18

Due Date:
May 7, 2019

1 Introduction

This report includes the documentation of a number of experiments carried out as stipulated by the handout provided. The six experiments attempted will explore the effect of the following variables on a pattern recognition system.

1. The effect of training sample size on classifier performance.
2. The effect of test set size on classifier performance, as well as dimensionality effects.
3. The effect of the size of test set on the reliability of the empirical error count estimator.
4. To explore the relationship between class separability and error probability.
5. The effect, on the classifier error probability, of discrepancies between the true and assumed class probability distribution models.
6. Comparing kNN with a Gaussian classifier.

Experiments 1-4 and 6 will each be presented in the following sections, and though some attempt was made at Experiment 5, only a partial solution will be presented in this report.

2 Experiment 1

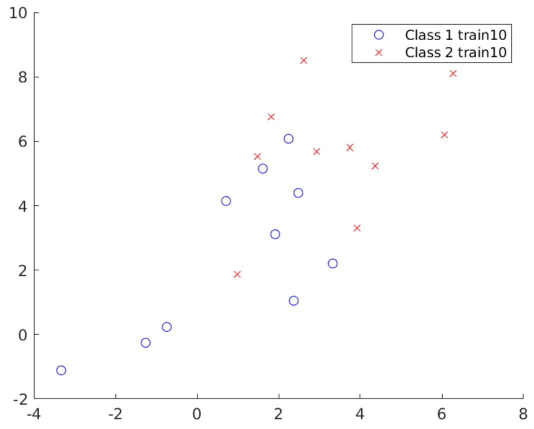
In this experiment the aim is to investigate the effect of the training sample size on the performance of the classifier. For this reason the test dataset will be first generated and will not be changed throughout this experiment. The training set, however, will be regenerated on every iteration of the experiment at a range of sizes. As opposed to re-sampling a static training dataset, this regenerated option was used as it would introduce an additional amount of unpredictability to the system. What this means is that as the training set size increases the test error will generally tend to reduce, but the training error may increase due to this randomness. However in general the mean test and training errors should be seen to converge. In order to account for individual bias in any one training set this experiment was conducted 10 times on each independent training dataset in all sizes $Nd == [3, 5, 10, 50, 100]$ in order to get an average case behaviour of the system with respect to changing training data. In this case the class parameters were used as instructed resulting in a mean of class 1 as μ_1, and the mean of class 2 as μ_1, and both classes sharing identical covariance matrices of $\Sigma_1 = \Sigma_2$:

$$\mu_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

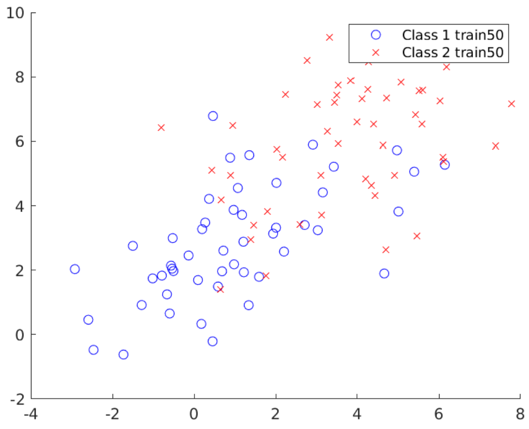
$$\mu_2 = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

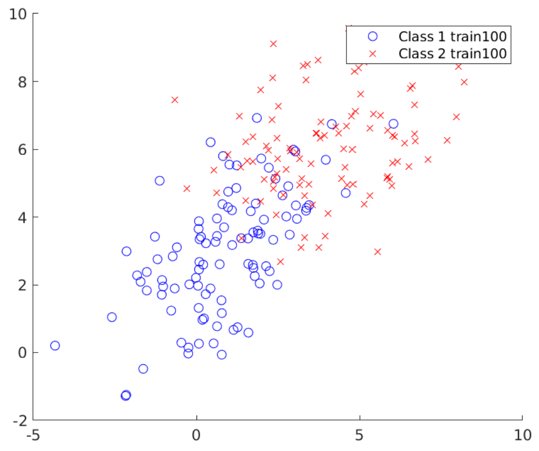
As seen in Figure 2.1, some of the datasets generated in this experiment at various sizes of N_D which allows us to visualize the overlap between the classes, and how different instances can pose different difficulties in the separability of the data-points in each class from the other.



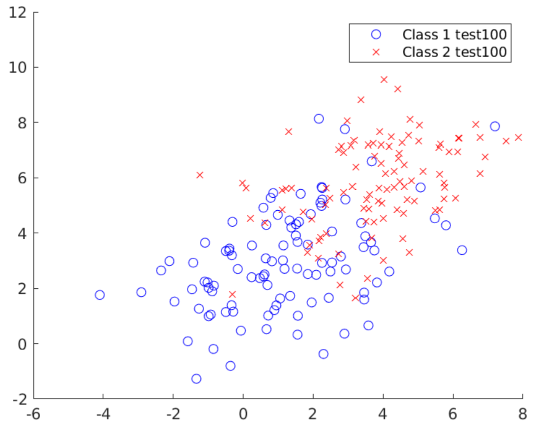
(a) 10 sample training set



(b) 50 sample training set



(c) 100 sample training set



(d) 100 sample test set

Figure 2.1: Visualisation of examples of test and training data generated for Experiment 1.

On each iteration the training data sets are regenerated, labels are configured appropriately, the model is created using `fitcnb()` and a prediction is made with the `predict()` function. These are then compared to the ground truth labelled data and an error calculated.

2.1 Results & Discussion

Seen below, in Figure 2.2 the gradual convergence of the test and training near error percentages. It is important to notice that naturally the test error remains higher to begin with in most cases as the model is more closely fit to the training data when N_D is small. As this value grows the errors converge as predicted.

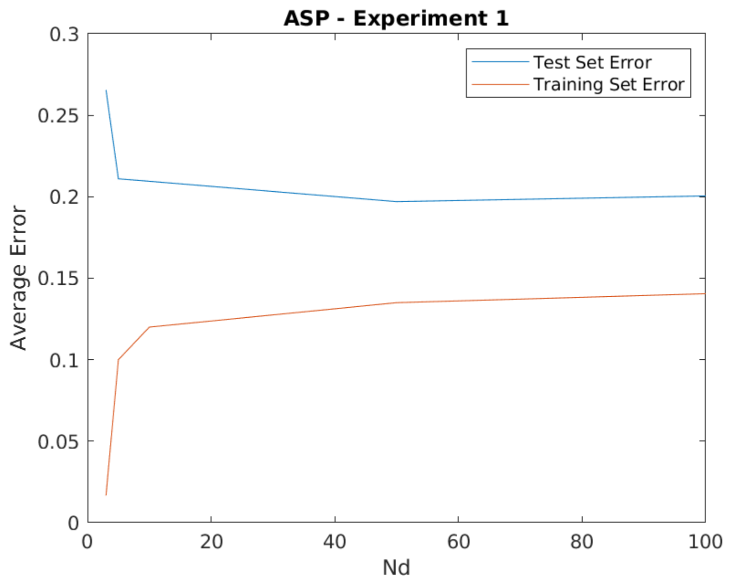


Figure 2.2: Experiment 1 Results

3 Experiment 2

This experiment investigates the effects of pattern recognition in higher dimensional domains. In order to do this the training error would now be regenerated for every new dimension considered (5, 10, 15), but would remain constant in that loop. The test dataset, however, would be regenerated on every iteration of the experiment. Once again following a similar format to the training cycle and repetitions as in Experiment 1, this experiment also has one fundamental difference in that since the training error will effectively remain constant (relative to the dimension). However the testing error should approach the true error value which was approximated by the $E(500)$ error probability approximation. As instructed the covariance matrix was selected to be an identity matrix, and the class means were designed in a way that μ_1 is a random vector of integers, and that μ_2 is offset by a random fractional distance from μ_1. The weights were manually adjusted to ensure that the error probability value remained within the range of 5-10% for all dimensions.

3.1 Results & Discussion

Unlike the expected convergence, the errors were seen to oscillate prior to convergence. This oscillation was more present at lower values of N_D and convergence to reasonable performance needed larger training datasets in lower dimensions. As seen in Figure 3.1 the oscillation lasts longer in the 5 dimensional space as opposed to the 10 and 15 dimensional spaces as seen in Figure 3.2. Though these experiments were conducted in low granularity it's clear that between 100-200 training sample sets are needed at a higher dimension of 10 or 15.

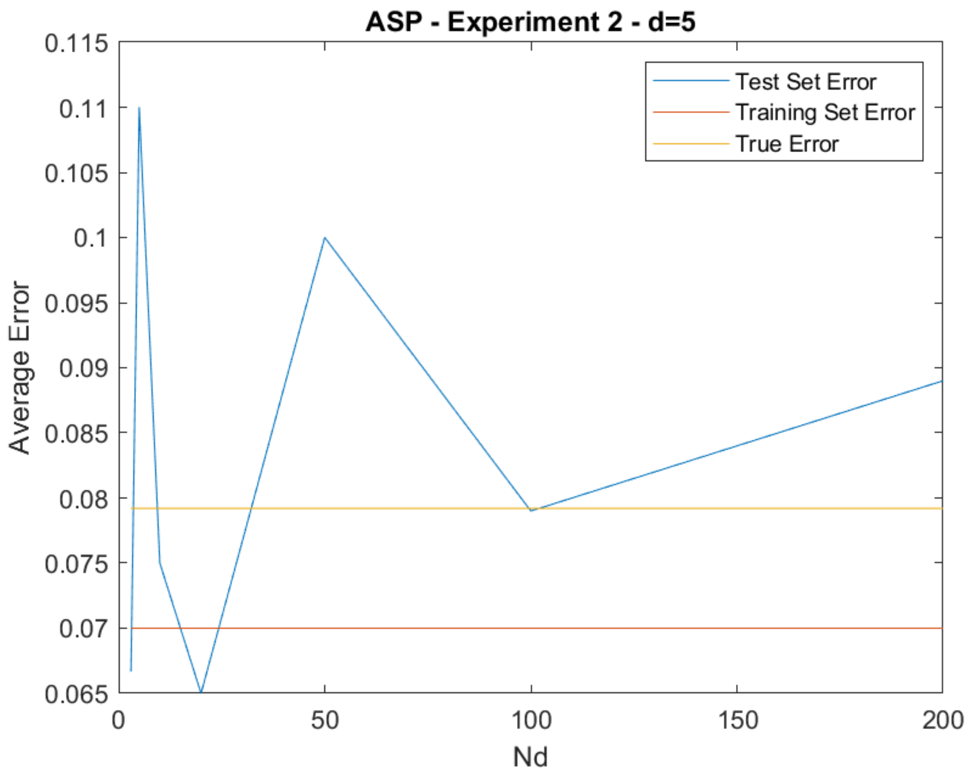
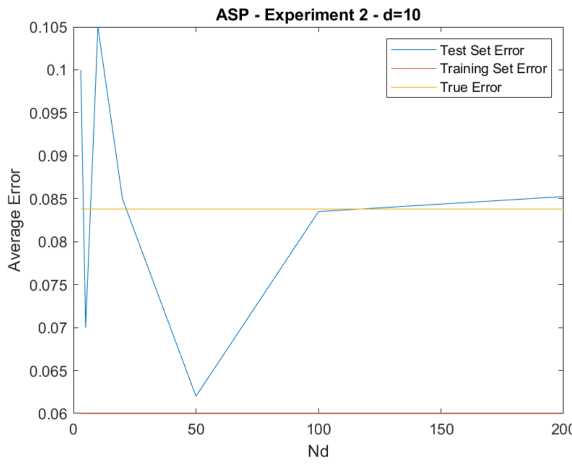
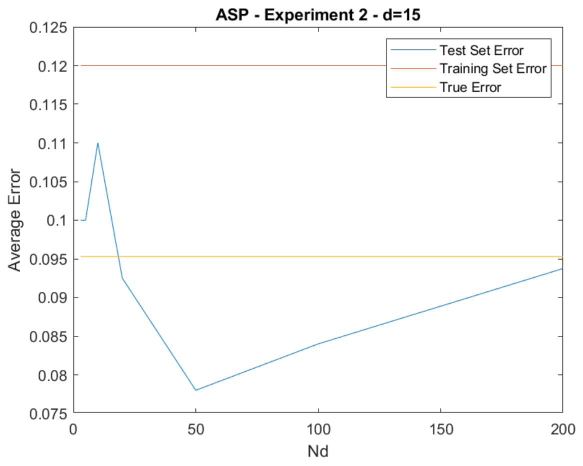


Figure 3.1: 5 Dimensions



(a) 10 Dimensions



(b) 15 Dimensions

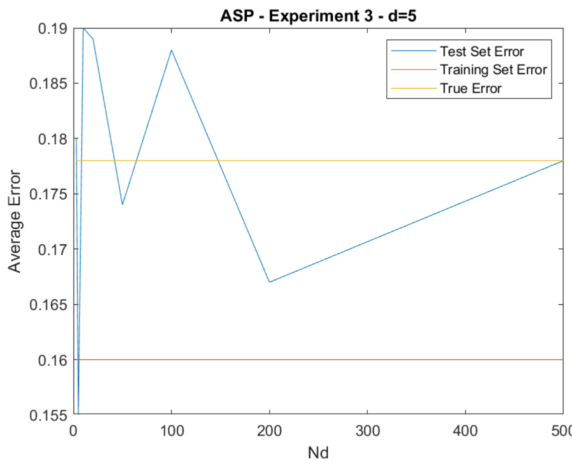
Figure 3.2: Experiment 2 Results - Mean Error Comparison vs Test Set Size

4 Experiment 3

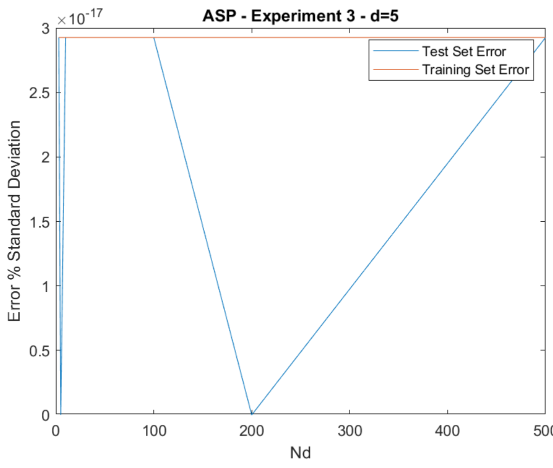
This experiment investigates the size of the test set and the reliability of a system given greater independence in datasets. This time the class means are defined in a similar manner to the process in Experiment 2, however the weighting differs as the error probability is given a wider allowable range of 1-10%. Ten such test sets are created and stored independently, and the process once again iterates 10 times for each sample size $N_D = 5, 10, 20, 50$ and then intervals of 50 until $N_D = 500$. Once again, as in Experiment 2, the Test Set Error should converge to the true error, and the training set error should remain constant as the training set itself is never changed.

4.1 Results and Discussion

As expected the error converges to the true error and, as mentioned in Experiment 2, this occurs at $N_D > 200$ in the 5-dimensional space selected for this experiment. The oscillatory nature observed previously in Experiment 2 presents similar characteristics here as the standard deviation in error varies as seen in 4.1(b). So far this allows us to conclude that greater reliability in approximating the true error can be achieved with larger test data sets. However the standard deviation does not significantly reduce suggesting that an increase in test dataset may not be enough to improve reliability in the system - and that the increase in reliability is more prevalent when an increase in training data is offered (as in Experiment 1).



(a) 5 Dimensions



(b) Standard Deviation of Error % vs Test Set Size

Figure 4.1: Experiment 3 Results - Test Set Size Reliability

5 Experiment 4

In this experiment the relationship between class separability and error probability is explored by creating 10 pairs of 2-class datasets, each separated monotonically such that the Mahalanobis distances for each pair of classes were recorded as: $d_M^2 = 1.45, 1.7, 3.4, 3.7, 4.583, 6.25, 12.5, 12.75, 16.67, 24.05$. These classes were generated randomly and then sorted to ensure that the sequence of class pairs would be ranked in a monotonic fashion (and manually checked so as to not be equal). All the class parameters were randomized in this experiment, and the covariance matrix was some randomized diagonal matrix. The number of test samples was nominally chosen as 100 samples per class to remain consistent with much of the other experimentation in this report.

5.1 Results & Discussion

What can be observed in the results is that in general as the Mahalanobis distance between the two classes increases, its linear separability is increased, naturally leading to reduced errors. This is reflected in Figure 5.1 in which there is a distinct downward trend in the error percentage as a function of the Mahalanobis distance. There do exist some occasional upwards spikes in error which can likely be attributed to individual test case variations, as in this experiment 10 iterations per sequence was not performed (unlike in previous experiments). This makes it more prone to noise and extrapolating from previous experiments (1 and 2), adding this iterative step to aggregate a mean error would reduce the sensitivity to such spikes.

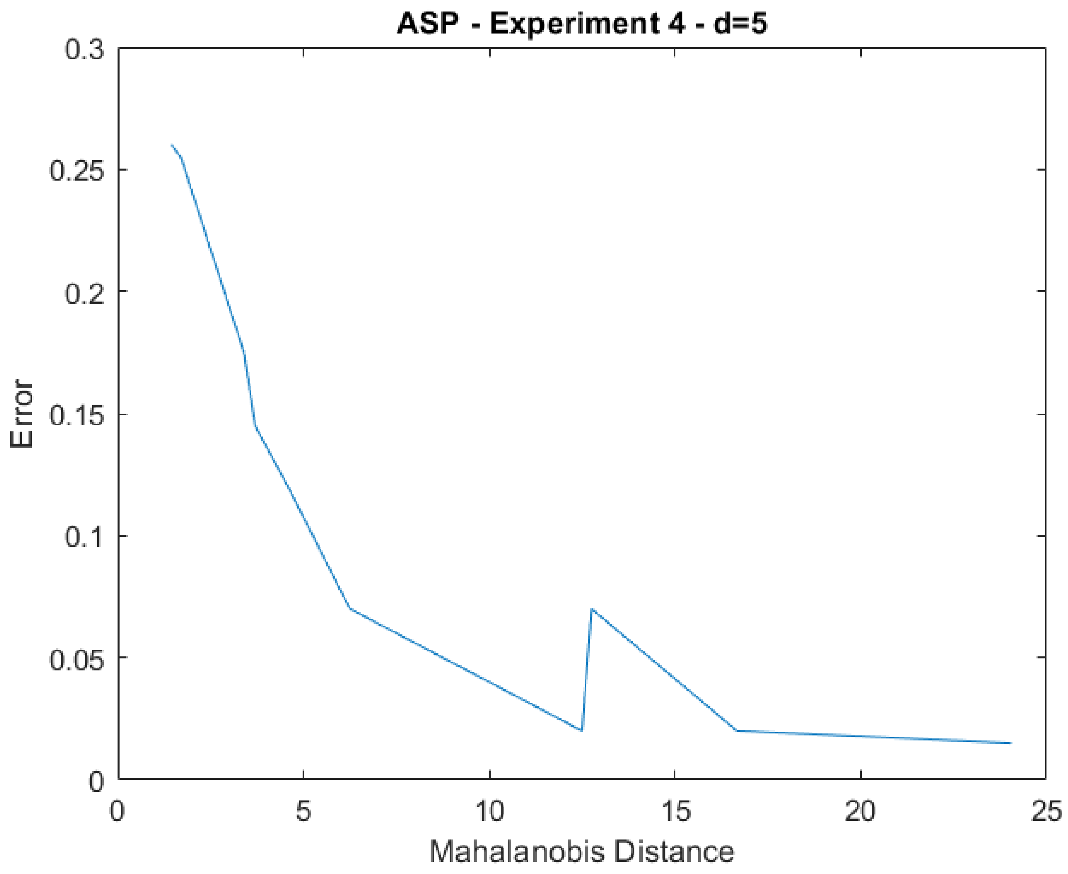


Figure 5.1: Experiment 4 Results

6 Experiment 5

This experiment aims to investigate the effects of discrepancies between true and assumed class probability distributions, and how this may impact the classifier error probability. As classes with differing covariances are predicted by a model that ill-fits their class characteristics, the error probability would worsen. On the other hand when modelling the classifier against a subset of the actual test data, the characteristics between the training and test datasets would be more closely aligned, allowing for a more accurate classification error probability closer to the Bayesian optimal error.

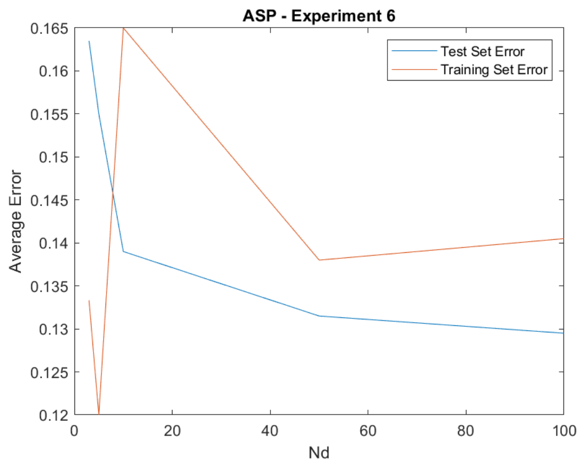
Unfortunately this experiment was left incomplete and the predictions above could not be fully substantiated.

7 Experiment 6

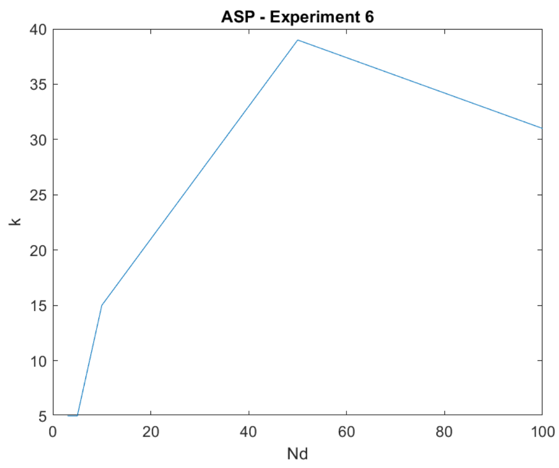
Similar in structure to Experiment 1, this experiment aims to compare the results of a Gaussian classifier `fitcnb` and a k-Nearest Neighbours classifier. This experiment was conducted in an iterative manner, taking into account the average error across 10 iterations for each value of k and for each different sample size. These were then collated and compared to identify, at each value of sample size N_D, the lowest test error, the k-value corresponding to this, and the training error corresponding to that k-value. The k values in this iterative process needed to be limited to the total number of data-points $2N_D$.

7.1 Results & Discussion

Seen in Figure 7.1 the k-value tended to be higher as N_D increased, however this was also prone to variance in datasets and was not a monotonic effect. Ultimately this seems to perform marginally more effectively than the Gaussian classifier in Experiment 1. Comparing to Figure **??** where the error tended to settle around an error of 0.2%, the test set error with a k-Nearest Neighbour classifier seemed to settle at 0.13% when comparing at $N_D = 100$, and with approximately 31-Nearest Neighbours being taken into account for the classification.



(a) Lowest Error % for k-Nearest Neighbour algorithm



(b) Optimum k value for k-Nearest Neighbour algorithm as a function of the size of the test dataset.

Figure 7.1: Experiment 6 Results - kNN comparison

8 Conclusions

All in all 5 out of the 6 stipulated experiments were completed and recorded as above. Primarily the learnings gained through these experiments conclude that the critical features to error probability reduction in pattern recognition and classification algorithms are ensuring an appropriate training data size which contains inside it the right characteristics around the covariance in the classes. Combined with the learnings from the first assignment in this course, it can be assessed that an accurate fit to data models is critical to ensuring accurate classification and that over-fitting or under-fitting can easily be experienced as optimal data sets are rarely realistic in practical applications. The randomized nature of a number of these experiments aided in reaching this conclusion.

It's also noted that different classifiers would have different strengths and weaknesses, as seen in utilizing the benefits of the k-Nearest Neighbour classifier to overcome the lack of training data and still improving the performance when compared to the Gaussian classifier used in most of the experiments here.

A Appendix A: Source Code

A.1 Experiment 1

Listing 1: Matlab Code developed to simulate experiment 1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp1.m
% Date started: 11-Mar-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear everything and setup
clear all
close all
clc

a = 1;
b = 3;
c = 3;
d = 3;
f = 1;

mu1 = [a;b]
mu2 = [a+d; b+d]
cova = [c f;f c] % cova is the covariance matrix. didn't use cov since it's a built
    ↪ in function


%% Effect of training sample size on classifier performance
X1_test = mvnrnd(mu1,cova,100);
X2_test = mvnrnd(mu2,cova,100);
plot2Classes(X1_test, X2_test, 1, "test")

test_labels = [zeros(100,1); ones(100,1)];
sampleSizes = [3 5 10 50 100]

E_train = [];
E_test = [];

for i = 1:length(sampleSizes)
    e_train = [];
    e_test = [];
    for iters = 1:10
        Nd = sampleSizes(i);
        X1_design = mvnrnd(mu1,cova,Nd);
        X2_design = mvnrnd(mu2,cova,Nd);
        X = [X1_design; X2_design];
        plot2Classes(X1_design, X2_design, 1, "train")
        train_labels = [zeros(Nd,1); ones(Nd,1)];
        mdl = fitcnb(X, train_labels);
        train_prediction = predict(mdl, X);
        test_prediction = predict(mdl, [X1_test; X2_test]);

        e_train(iters) = sum(xor(train_prediction, train_labels))/length(
            ↪ train_prediction);
        e_test(iters) = sum(xor(test_prediction, test_labels))/length(
            ↪ test_prediction);
    end
    E_train(i) = sum(e_train)/iters;
    E_test(i) = sum(e_test)/iters;
end

fig = figure
plot(sampleSizes, E_test)
hold on
plot(sampleSizes, E_train)
legend('Test Set Error', 'Training Set Error')
title(sprintf("ASP - Experiment 1"))
xlabel('Nd')
ylabel('Average Error')
% saveas(fig,'./Exp1-results/ErrorComparison.png')
```

A.2 Plot 2 Classes

Listing 2: Matlab Code developed to help plot classes in order to visualize the process. Predominantly used in Experiment 1.to simulate experiment 1.

```
function plot2Classes(w1,w2, exp, name)

    fig = figure
    scatter(w1(:,1),w1(:,2), 'bo')
    hold on
    scatter(w2(:,1),w2(:,2), 'rx')
    legend(sprintf('Class 1 ' + name + length(w1)), sprintf('Class 2 '+ name +
        ↪ length(w2)))
% saveas(fig,sprintf("./Exp%i-results/",exp) + name + length(w1) +'.png')
end
```

A.3 Experiment 2

Listing 3: Matlab Code developed to simulate experiment 2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp2.m
% Date started: 18-Apr-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear everything and setup

clear all
close all
clc

a = 1;
b = 3;
c = 3;
d = 3;
f = 1;

%% Effect of test set size on classifier performance

sampleSizes = [3 5 10 20 50 100 200 500]
dimensionSizes = [5, 10, 15]

E_train = [];
E_test = [];

for k = 1:length(dimensionSizes)
    dims = dimensionSizes(k)
    mu1 = randi(5,dims,1)
    if dims == 5
        mu2 = mu1 + 2*rand(dims,1)
    elseif dims == 10
        mu2 = mu1 + 1.6*rand(dims,1)
    else dims == 15
        mu2 = mu1 + 1.35*rand(dims,1)
    end
    cova = eye(dims)
    X1_design = mvnrnd(mu1,cova,100);
    X2_design = mvnrnd(mu2,cova,100);
    for i = 1:length(sampleSizes)
        e_train = [];
        e_test = [];
        for iters = 1:10
            Nd = sampleSizes(i);
            X1_test = mvnrnd(mu1,cova,Nd);
            X2_test = mvnrnd(mu2,cova,Nd);
            X = [X1_design; X2_design];
            train_labels = [zeros(100,1); ones(100,1)];
            test_labels = [zeros(Nd,1); ones(Nd,1)];
            mdl = fitcnb(X, train_labels);
            train_prediction = predict(mdl, X);
            test_prediction = predict(mdl, [X1_test; X2_test]);

            e_train(iters) = sum(xor(train_prediction, train_labels))/length(
               ↪ train_prediction);
            e_test(iters) = sum(xor(test_prediction, test_labels))/length(
               ↪ test_prediction);
        end
        E_train(i) = sum(e_train)/iters;
        E_test(i) = sum(e_test)/iters;
    end

    fig = figure
    plot(sampleSizes(1:7), E_test(1:7))
    hold on
    plot(sampleSizes(1:7), E_train(1:7))
    hold on
    E_true = [E_test(8), E_test(8), E_test(8), E_test(8), E_test(8), E_test(8),
        ↪ E_test(8)]
    plot(sampleSizes(1:7), E_true)
    legend('Test Set Error', 'Training Set Error', 'True Error')
    title(sprintf("ASP - Experiment 2 -" + " d=" + dims))
    xlabel('Nd')
    ylabel('Average Error')
% saveas(fig,"./Exp2-results/ErrorComparison_" + dims + ".png")
% w = waitforbuttonpress
end
```

A.4 Experiment 3

Listing 4: Matlab Code developed to simulate experiment 3.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp3.m
% Date started: 03-May-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear everything and setup
```

```

clear all
close all
clc

a = 1;
b = 3;
c = 3;
d = 3;
f = 1;

%% effect of the size of test set on the reliability of the empirical error count
    ↪ estimator

sampleSizes = [3 5 10 20 50 100 200 500];

dims = 5;
mu1 = randi(5,dims,1);
mu2 = mu1 + 2*rand(dims,1);
cova = eye(dims);

X1_design = mvnrnd(mu1,cova,100);
X2_design = mvnrnd(mu2,cova,100);
Xd = [X1_design; X2_design];
train_labels = [zeros(100,1); ones(100,1)];

for i = 1:10
    X1_test = mvnrnd(mu1,cova,500);
    X2_test = mvnrnd(mu2,cova,500);
    Xt{i} = [X1_test; X2_test];
end
test_labels = [zeros(500,1); ones(500,1)];

testSizes = [5, 10, 20, 50:50:500];
E_train_mean = [];
E_test_mean = [];
E_train_std = [];
E_test_std = [];

for i = 1:length(sampleSizes)
    e_train = [];
    e_test = [];
    for iters = 1:10
        mdl = fitcnb(Xd, train_labels);
        train_prediction = predict(mdl, Xd);
        test_prediction = predict(mdl, Xt{i});
        e_train(iters) = sum(xor(train_prediction, train_labels))/length(
            ↪ train_prediction);
        e_test(iters) = sum(xor(test_prediction, test_labels))/length(
            ↪ test_prediction);
    end
    E_train_mean(i) = sum(e_train)/iters;
    E_test_mean(i) = sum(e_test)/iters;
% E_train_std(i) = (1/9)*sum((e_train.*E_train_mean(i)).^2);
% E_test_std(i) = (1/9)*sum((e_test.*E_test_mean(i)).^2);
    E_train_std(i) = std(e_train);
    E_test_std(i) = std(e_test);
end

fig = figure;
plot(sampleSizes, E_test_mean)
hold on
plot(sampleSizes, E_train_mean)
hold on
E_true = [E_test_mean(end), E_test_mean(end), E_test_mean(end), E_test_mean(end),
    ↪ E_test_mean(end), E_test_mean(end), E_test_mean(end), E_test_mean(end)];
plot(sampleSizes, E_true)
legend('Test Set Error', 'Training Set Error', 'True Error')
title(sprintf("ASP - Experiment 3 -" + " d=" + dims))
xlabel('Nd')
ylabel('Average Error')
% saveas(fig,"./Exp3-results/ErrorComparison_" + dims + ".png")

fig = figure;
plot(sampleSizes, E_test_std)
hold on
plot(sampleSizes, E_train_std)
legend('Test Set Error', 'Training Set Error')
title(sprintf("ASP - Experiment 3 -" + " d=" + dims))
xlabel('Nd')
ylabel('Error % Standard Deviation')
% saveas(fig,"./Exp3-results/ErrorStandardDeviation_" + dims + ".png")
```

A.5 Experiment 4

Listing 5: Matlab Code developed to simulate experiment 4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp4.m
% Date started: 7-May-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%% Clear everything and setup
clear all
close all
clc

%% create multiple distribution pairs monotonically increasing.
dims = 5;
numSequences = 10;
classSizes = 100;

mu1 = [];
mu2 = [];
cova = [];
dm2 = [];

for i = 1:numSequences
    mu1{i} = randi(5,dims,1);
    mu2{i} = randi(5,dims,1);
    cova{i} = randi(5,dims,1) .* eye(5);
    dm2(1,i) = (mu1{i}-mu2{i})'*inv(cova{i})*(mu1{i}-mu2{i});
    dm2(2,i) = i;
end
ordered_dm2 = sortrows(dm2', 1)';

for i = 1:numSequences
    X1 = mvnrnd(mu1{ordered_dm2(2,i)},cova{ordered_dm2(2,i)},classSizes);
    X2 = mvnrnd(mu2{ordered_dm2(2,i)},cova{ordered_dm2(2,i)},classSizes);
    Xt{i} = [X1; X2];
    ordered_m1{i} = mu1{ordered_dm2(2,i)};
    ordered_m2{i} = mu2{ordered_dm2(2,i)};
    ordered_cova{i} = cova{ordered_dm2(2,i)};
end
labels = [zeros(classSizes,1); ones(classSizes,1)];

for i = 1:numSequences
    mdl = fitcnb(Xt{i}, labels);
    prediction = predict(mdl, Xt{i});
    e(i) = sum(xor(prediction, labels))/length(prediction);
end

fig = figure;
plot(ordered_dm2(1,:), e)
xlabel('Mahalanobis Distance')
ylabel('Error')
title(sprintf("ASP - Experiment 4 -" + " d=" + dims))
% saveas(fig,"./Exp4-results/MahalError_" + dims + ".png")
```

A.6 Experiment 5

Listing 6: Matlab Code developed to simulate experiment 5.

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp5.m
% Date started: 7-May-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a = 1;
b = 3;
c = 3;
d = 3;
f = 1;

mu1 = [a;b]
mu2 = [a+d; b+d]
cova = eye(2)
```

A.7 Experiment 6

Listing 7: Matlab Code developed to simulate experiment 6.

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% @author: Kausthub Krishnamurthy
% URN: 6562233
% EEEM007 Advanced Signal Processing - Lab Experiments
% Filename: asp_exp6.m
% Date started: 6-May-2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear everything and setup
clear all
close all
clc

a = 1;
b = 3;
c = 3;
d = 3;
f = 1;

mu1 = [a;b]
```

```
mu2 = [a+d; b+d]
cova = [c f;f c] % cova is the covariance matrix. didn't use cov since it's a built
    ↪ in function


%% Comparing kNN with a gaussian classifier
X1_test = mvnrnd(mu1,cova,100);
X2_test = mvnrnd(mu2,cova,100);

test_labels = [zeros(100,1); ones(100,1)];
sampleSizes = [3 5 10 50 100];
% sampleSizes = [3, 5:5:100];

kND_best = [];
E_train_best = [];
E_test_best = [];

E_train_mean_k = [];
E_test_mean_k = [];
for i = 1:length(sampleSizes)
    Nd = sampleSizes(i);
    E_train_mean = [];
    E_test_mean = [];
    if Nd*2 < 51
        kVals = 1:2:(2*Nd);
    else
        kVals = 1:2:51;
    end
    for kIter = 1:length(kVals)
        k = kVals(kIter);
        e_train = [];
        e_test = [];
        for iters = 1:10
            X1_design = mvnrnd(mu1,cova,Nd);
            X2_design = mvnrnd(mu2,cova,Nd);
            X = [X1_design; X2_design];
            train_labels = [zeros(Nd,1); ones(Nd,1)];
            mdl = fitcknn(X, train_labels);
            mdl.NumNeighbors = k;
            train_prediction = predict(mdl, X);
            test_prediction = predict(mdl, [X1_test; X2_test]);

            e_train(iters) = sum(xor(train_prediction, train_labels))/length(
                ↪ train_prediction);
            e_test(iters) = sum(xor(test_prediction, test_labels))/length(
                ↪ test_prediction);
        end
        E_train_mean(kIter) = sum(e_train)/iters;
        E_test_mean(kIter) = sum(e_test)/iters;
    end
    E_train_mean_k{i} = E_train_mean;
    E_test_mean_k{i} = E_test_mean;
end
for i = 1:length(E_test_mean_k)
    [val, idx] = min(E_test_mean_k{i});
    kND_best(i) = kVals(idx);
    E_train_best(i) = E_train_mean_k{i}(idx);
    E_test_best(i) = val;
end

fig = figure;
plot(sampleSizes, E_test_best)
hold on
plot(sampleSizes, E_train_best)
legend('Test Set Error', 'Training Set Error')
title(sprintf("ASP - Experiment 6"))
xlabel('Nd')
ylabel('Average Error')
% saveas(fig,'./Exp6-results/ErrorComparison.png')

fig = figure;
plot(sampleSizes, kND_best)
title(sprintf("ASP - Experiment 6"))
xlabel('Nd')
ylabel('k')
% saveas(fig,'./Exp6-results/BestkVals.png')
```