Optimization for Machine Learning

Suseela Pattamatta
ECE 503
December 21, 2019
Under the guidance of Prof Lei Zhao

Contents

Objective

Perform three machine learning algorithms - PCA, K-means clustering and Linear Regression, on the Iris flower dataset. The report includes different observations like calculating the misclassifications, error rates, CPU time and confusion charts.

1 Introduction

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species. Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

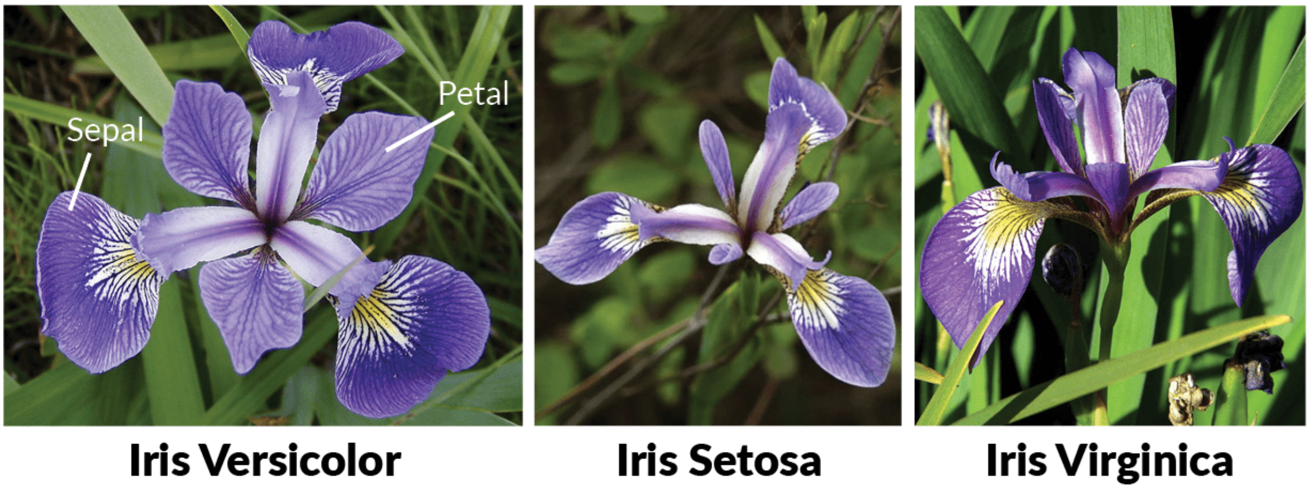


Figure 1: The three species of Iris flower

2 Principal Component Analysis

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. Basically, it is a method of summarizing data.

In this project, I am using the Iris dataset. The dataset consists of 150 samples, 50 each for the three different varieties of Iris flowers. The first 40 samples of each variety are considered as training data, while the other 10 samples of each variety are treated as testing data. So, the training data consists of 120 samples and the testing data consists to 30 samples. The labels are contained in a different file. The first 10 samples are labelled 1, the next 10 are labelled as 2 and the last 10 samples are labelled 3.

In this experiment, we create a model where the machine should be able to understand and recognize the different features of the Iris flowers in an efficient manner. To perform this, we use the training data and a testing data to which we apply the knowledge gained from the trained model. PCA works on the basis of Singular Variable Decomposition (SVD), where the dimensionality of a matrix is reduced

2.1 Implementation

1. The following initial variable are considered:
   - Number of classes: 3
   - Number of samples per class: 40
   - Number of rank approximations: 4 (i=1,2,3,4)
2. Calculate the centralized mean and covariance of each of the three classes in the dataset and put them in a matrix
3. Calculate the eigen vectors of the covariance matrix
4. Calculate the principal components
5. Calculate the PCA approximations
6. Calculate the error between PCA approximation and the test data
7. Identify the target classes of PCA approximation by which classes have the least errors
8. Create a Confusion chart to display the misclassifications in each PCA approximation

The final section of the MATLAB code is used to compute the CPU execution time of the PCA approximations and calculates the number of wrongly labelled data points and plots the data for each rank approximation values.

2.2 MATLAB Code and Results

Contents

```
clc;
clear;
close all;

PLOT = 1;
```

Load Data

```
load D_iris_tr; % Training data
load D_iris_te; % Testing data

Lte28 = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];

Length_Test = length(Lte28);
```

Testing

Calculating PCA for the 4 parameters (Petal length, Petal width, Sepal length and Sepal width)

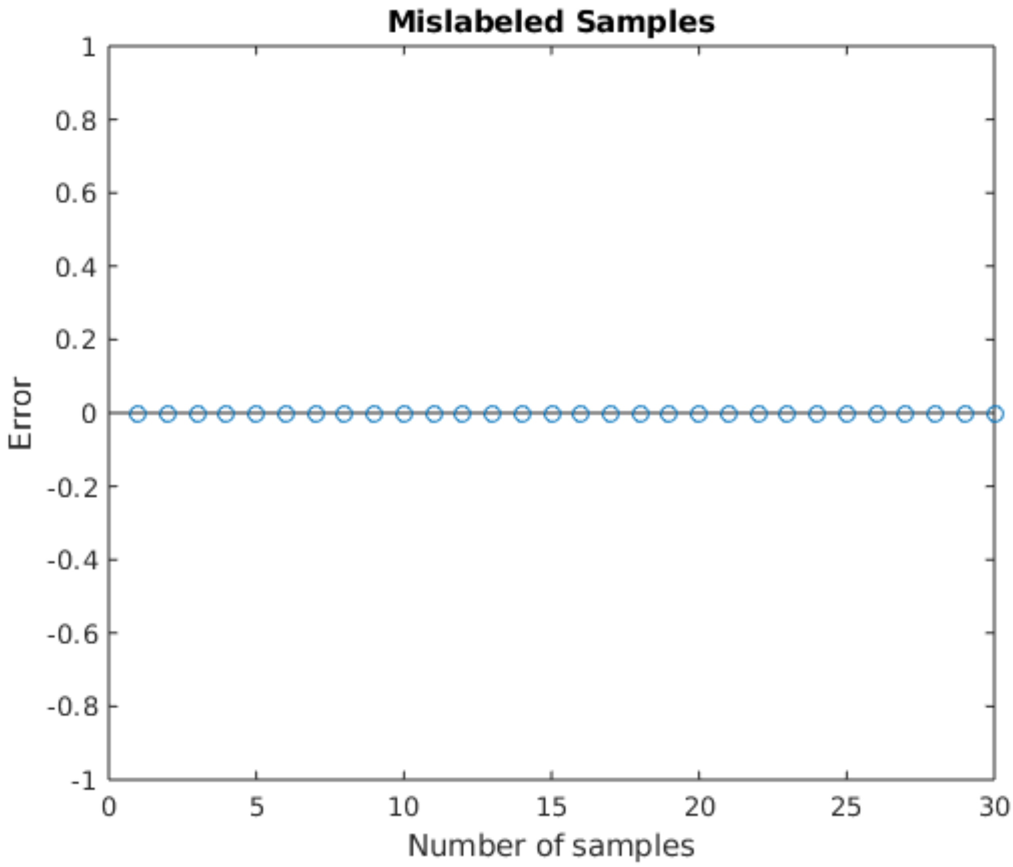
```
for q = 1:4
```

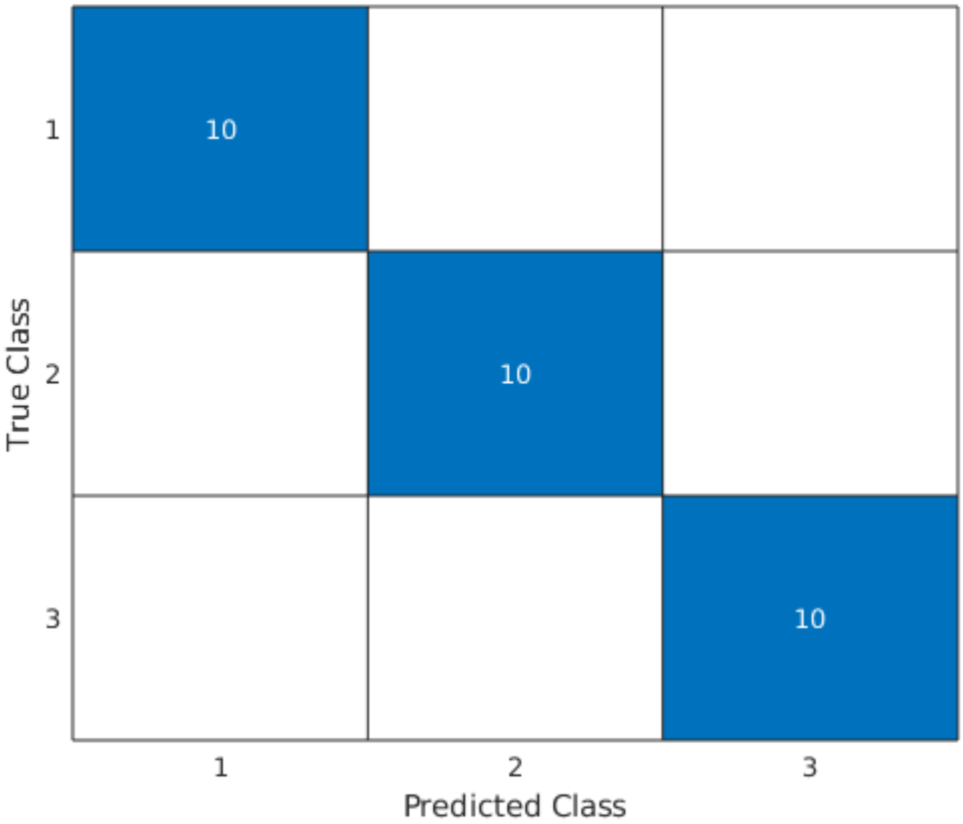
```
    num_feature = 4;
    nj = 40; % Samples per class
    ni = 3; % Classes

    mu = zeros(num_feature,ni);
    U = zeros(num_feature,q*ni);
    for ii = 1:ni
        Xi = D_iris_tr(:,(ii-1)*nj+1:ii*nj);
        mu_j = mean(Xi,2); % Calculate Mean of Class
        Xh = Xi - mu_j*ones(1,nj); % Xh = (Xi - mu_i)
        Cj = Xh*Xh'; % Covariance Matrix

        % STEP 2 ------------------------------
        [Uq,~] = eigs(Cj,q); % Get the eigen values
        % Save the sub space
        U(:,(ii-1)*q+1:ii*q) = Uq;
        mu(:,ii) = mu_j;
end
```

Testing

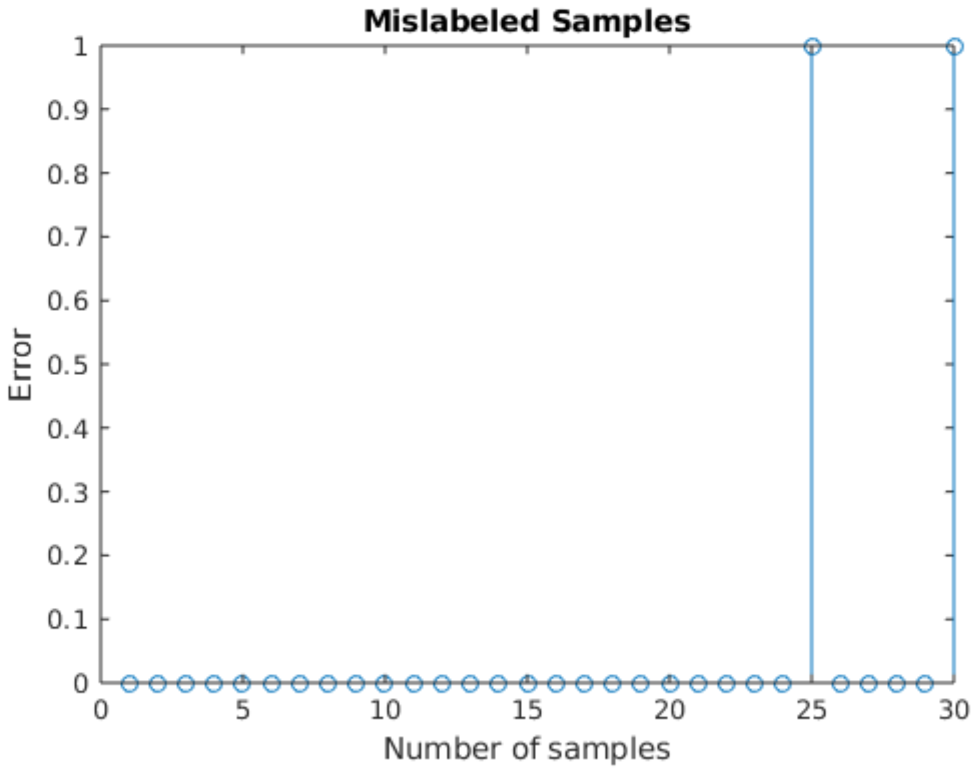
```
t0 = cputime; % Get CPU time
Predicted_Label = zeros(Length_Test,1);
for jj = 1:Length_Test
     Xt = D_iris_te(:,jj);
     e = zeros(1,ni);
     for ii = 1:ni
         % STEP 3 ---------------------
         Cj2 = Xt - mu(:,ii);
         fj = U(:,(ii-1)*q+1:ii*q).'*Cj2;

         % STEP 4 ---------------------
         Xj = U(:,(ii-1)*q+1:ii*q)*fj + mu(:,ii);

         % STEP 5 ---------------------
         e(ii) = norm(Xt-Xj);
     end
     [~,MinIdx] = min(e);
     Predicted_Label(jj) =  MinIdx;
end

cpt = cputime - t0;
E = (Lte28 ~= Predicted_Label);
if PLOT == 1
    figure
    stem(E)
    xlabel('Number of samples');
    ylabel('Error');
    title('Mislabeled Samples');

end
disp('Number of Errors')
n_mis = sum(E)
disp('Error rate is : ');
disp(n_mis/30);
disp('CPU time (s):')
cpt

figure
P = confusionchart(Lte28,Predicted_Label);
```

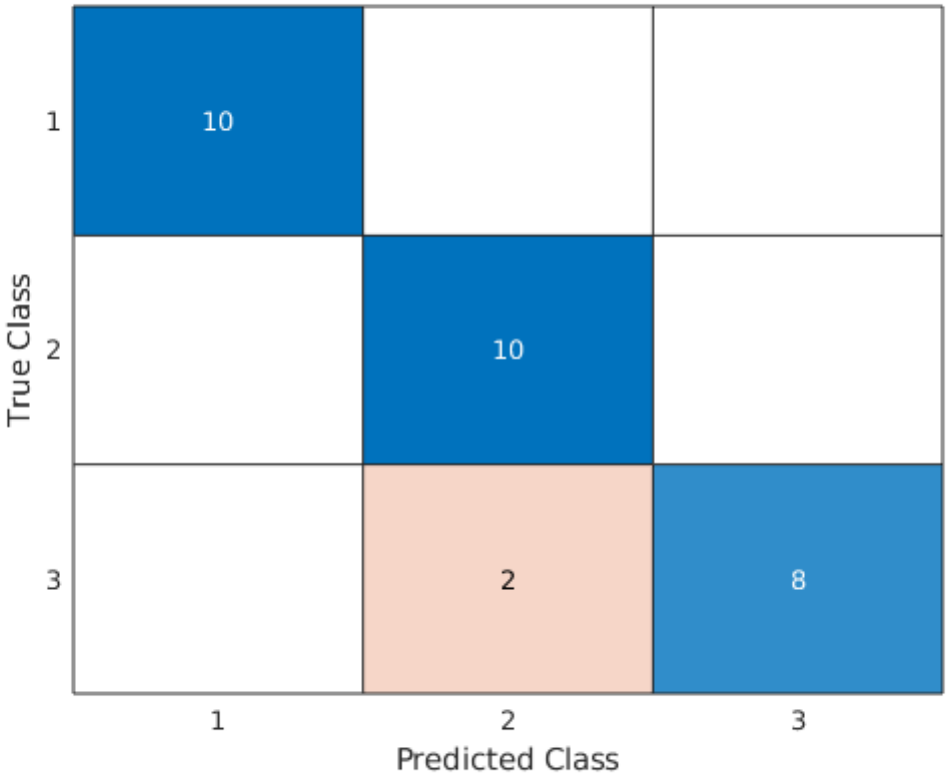
```
Number of Errors

n_mis =

     0

Error rate is :
     0

CPU time (s):

cpt =
    0.0100
```


Mislabeled Samples
Error
Number of samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25
30

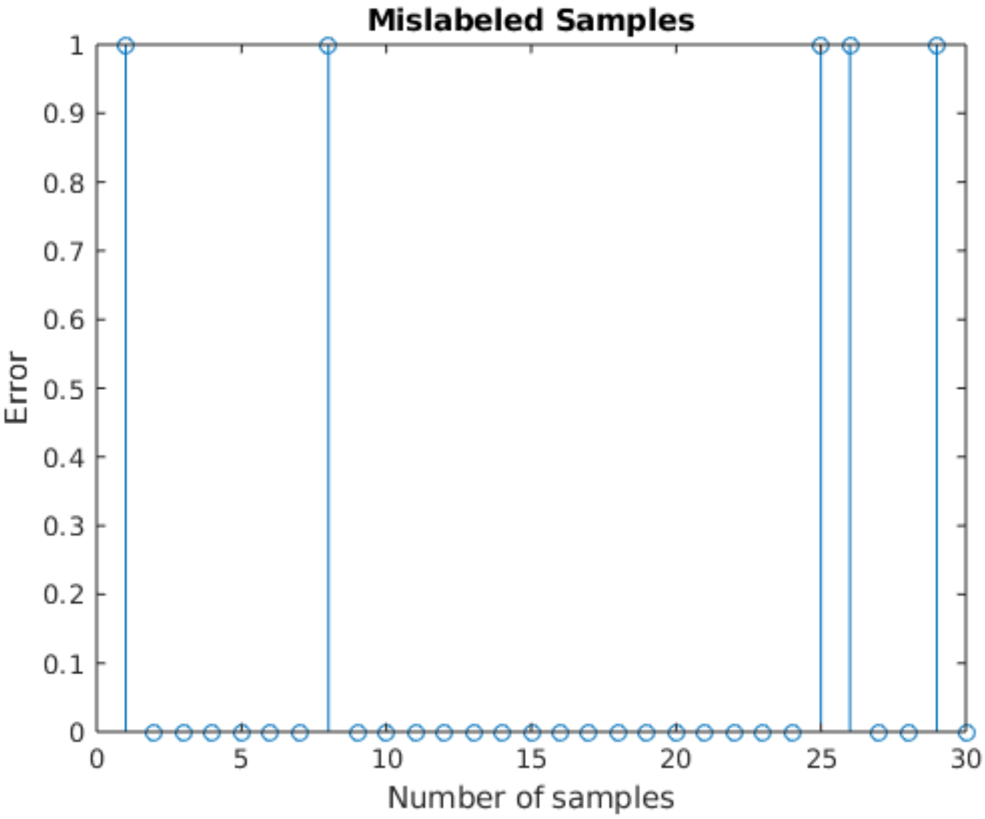


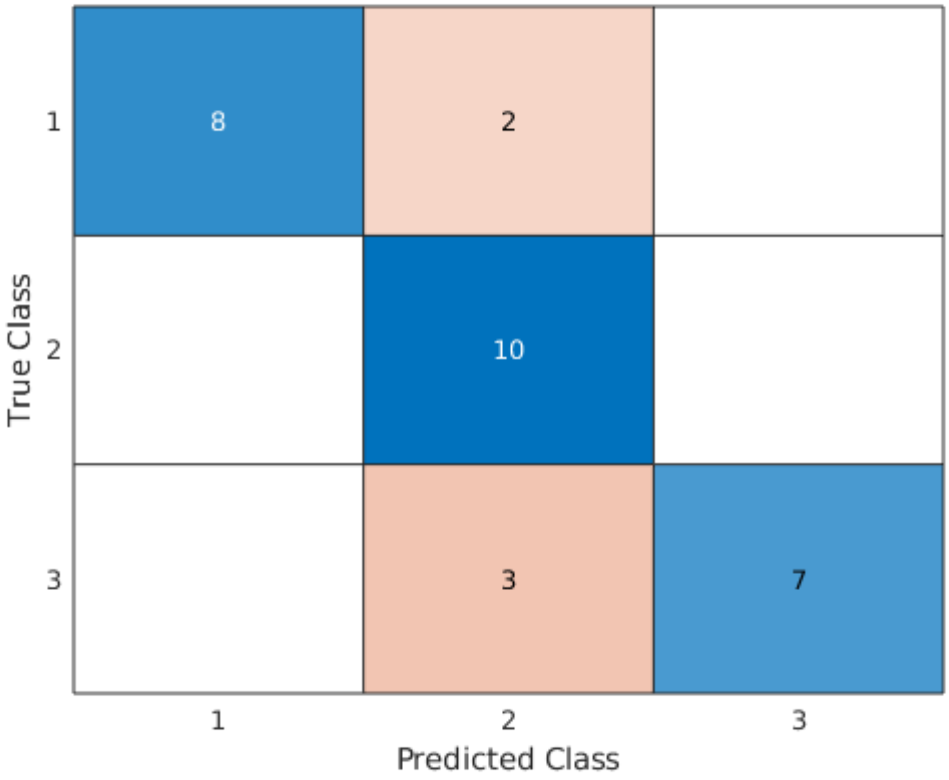
```
Number of Errors

n_mis =

     2

Error rate is :
    0.0667

CPU time (s):

cpt =

     0
```



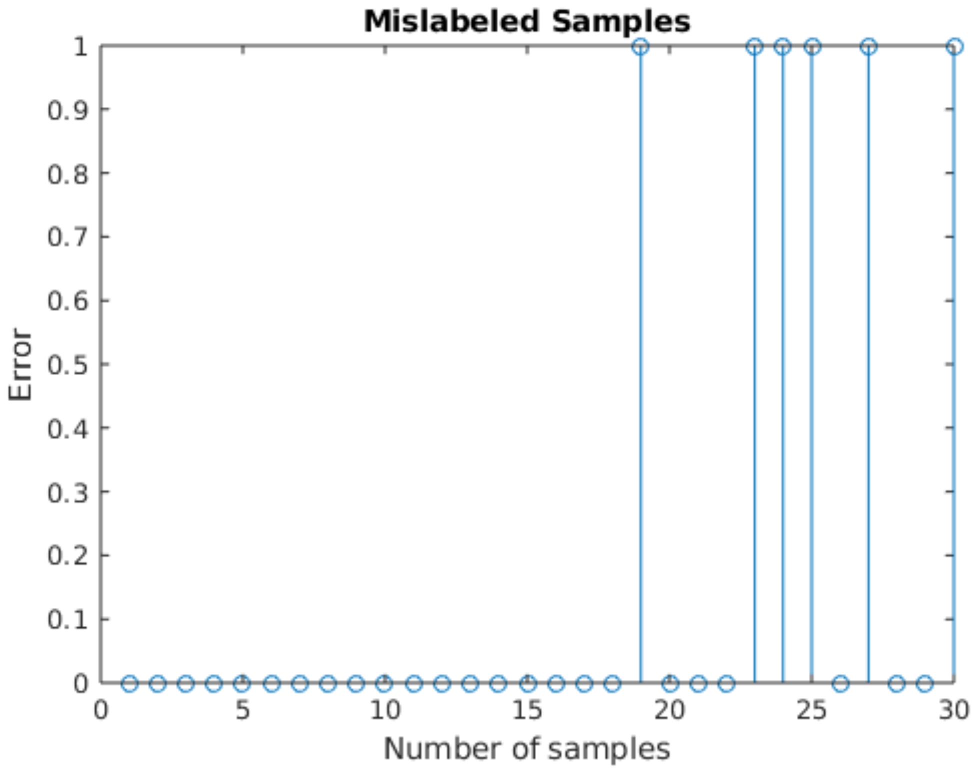


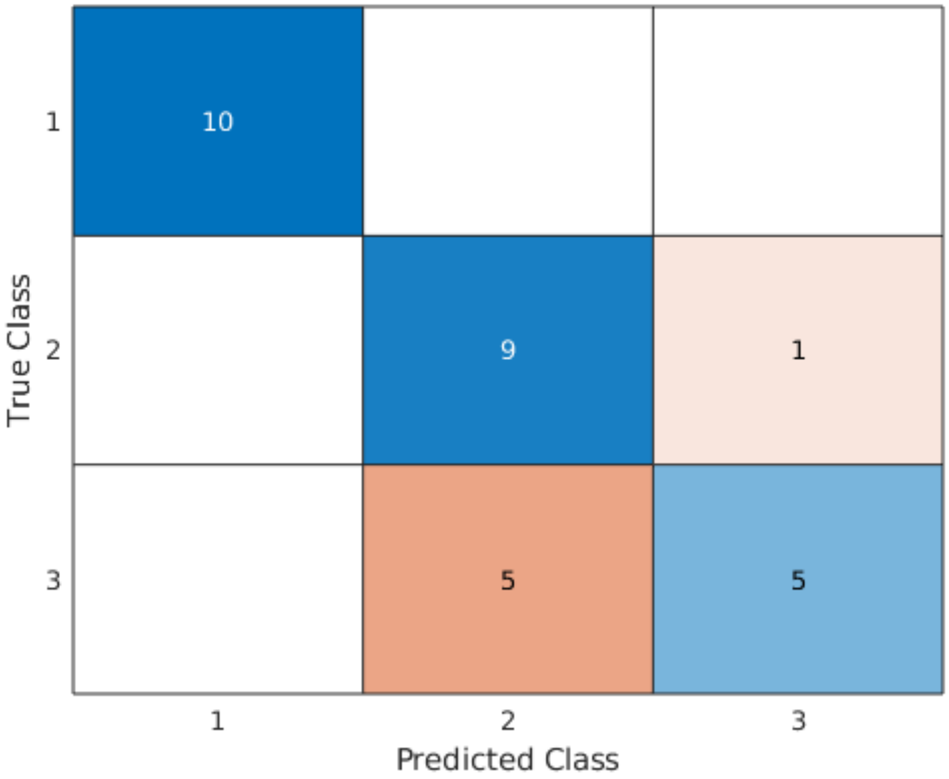
```
Number of Errors

n_mis =
     5

Error rate is :
    0.1667

CPU time (s):

cpt =

     0
```





```
Number of Errors

n_mis =

     6

Error rate is :
    0.2000

CPU time (s):

cpt =

     0
```


Mislabeled Samples
Error
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
Number of samples


True Class
1
2
3
10
9
1
5
5
1
2
3
Predicted Class

```
end
```

2.3 Discussion

This experiment uses the PCA algorithm to classify the three species of Iris flower. The algorithm performed the best when rank approximation was set to 1, which provided 0% error or 0 mislabeled numbers in the dataset when tested on the 30 test samples set. The consecutive images show the mislabeled samples in terms of the % error of each species and the confusion matrix for each rank approximation. Overall, the algorithm performed quite well for not having to modify and be tuned. To achieve better results one could clean the sample data set more to discard any outliers or run another algorithm such a k-means over the classified data to provide better grouping of each sample.

2.4 Conclusions

PCA algorithm has been used to recognize the different species of Iris flowers from a sample of 30 images. When the rank is set to the optimum value (1), the number of mislabeled digits takes the least possible value (0).

3 K-means clustering

In this algorithm, we investigate the clustering technique using the features of the data samples. K-means clustering algorithm segregates the data samples into k number of groups based on the centroids provided to the algorithm. The centroids and the associated points tend to change with each iteration. At a certain point, the clusters are static and that is when K-means clustering is efficiently enforced.

3.1 Implementation

1. A pair of points are assigned as arbitrary centroids.
2. The distance between each point and each centroid is calculated (the second norm).
3. The centroid with the least distance to a particular point is assigned a rank 1
4. Once all the points are assigned a rank according to the distance to the centroid pairs, the points are clustered into 2 groups.
5. The new centroids can be calculated as the average point of all the points in a cluster.
6. Steps 2 to 4 should be repeated until the new centroids do not change in position.

3.2 MATLAB Code and Results

Contents

```
clear all
clc
```

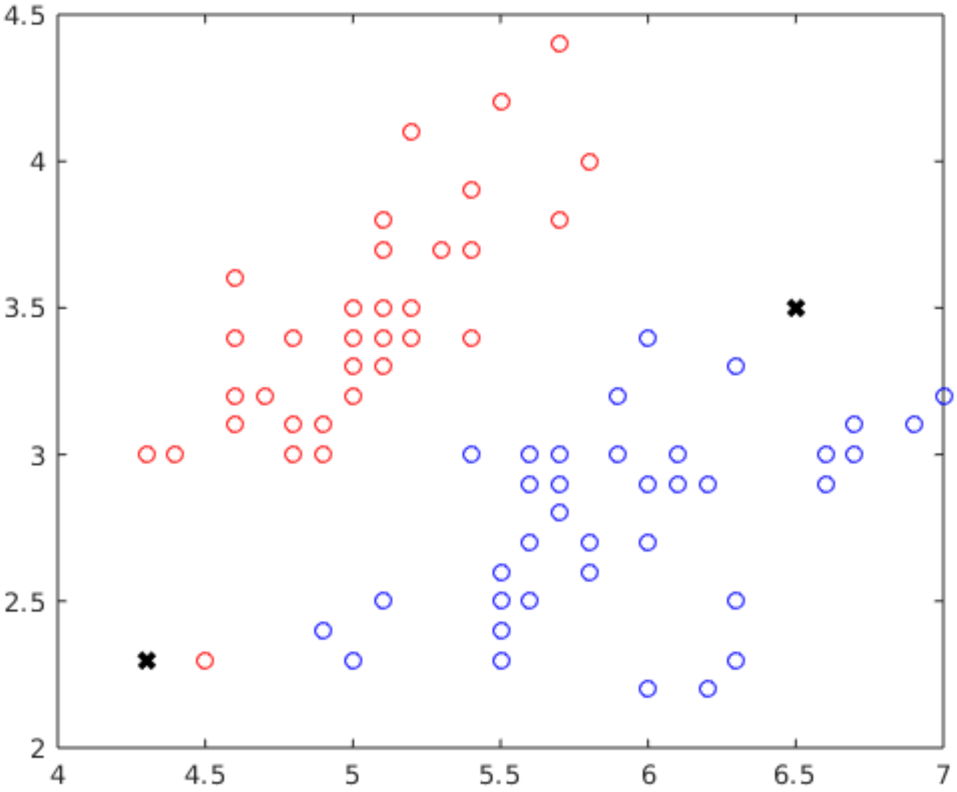
Set data

```
load D_iris_tr
X = D_iris_tr';

D=[X(1:80,1) X(1:80,2)];
```

Initial

```
K=2;
u1_0=[4.3,2.3];
u2_0=[6.5,3.5];
u=[u1_0' u2_0']
disp('Size of cluster 1')
disp(length(D(1:40,1)))
disp('Size of cluster 2')
disp(length(D(41:80,1)))

fig = figure;
plot(D(1:40,1),D(1:40,2),'ro')
hold on
plot(D(41:80,1),D(41:80,2),'bo')
hold on
plot(u(1,:),u(2,:),'kx','LineWidth',3)
hold off

R=zeros(80,2); %rank
```

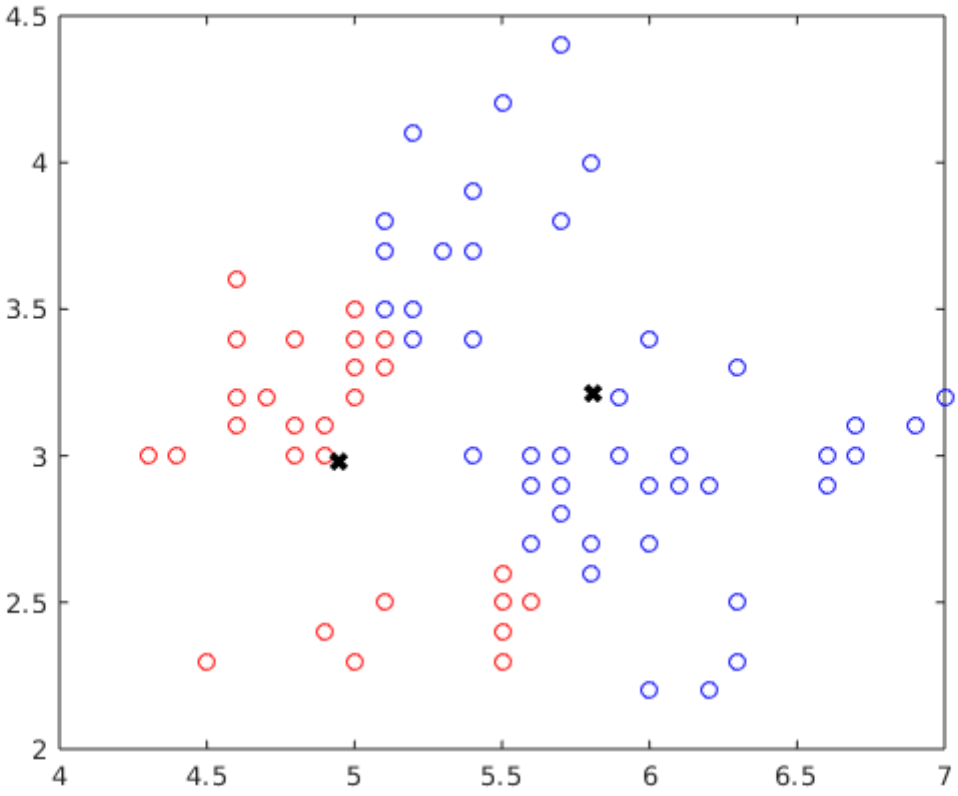
```
u =

    4.3000    6.5000
    2.3000    3.5000

Size of cluster 1
    40

Size of cluster 2
    40
```



First iteration

```
n=0;
m=0;
ii=1;
while ii<=80
    Xi=D(ii,:);
    if norm(Xi-u1_0,'fro')<=norm(Xi-u2_0,'fro')
        R(ii,1)=1;
        n=n+1;
        C11(n,:)=Xi';
    else
        R(ii,2)=1;
        m=m+1;
        C12(m,:)=Xi';
    end
    ii=ii+1;
end
u11= mean(C11,1)';
u12= mean(C12,1)';

u1 = [u11 u12]
disp('Size of cluster 1')
disp(length(C11(:,1)))
disp('Size of cluster 2')
disp(length(C12(:,1)))

figure;
plot(C11(1:end,1),C11(1:end,2),'ro')
hold on
plot(C12(1:end,1),C12(1:end,2),'bo')
hold on
plot(u1(1,:),u1(2,:),'kx','LineWidth',3)
```

```
u1 =

    4.9452    5.8082
    2.9806    3.2102

Size of cluster 1
    31

Size of cluster 2
    49
```

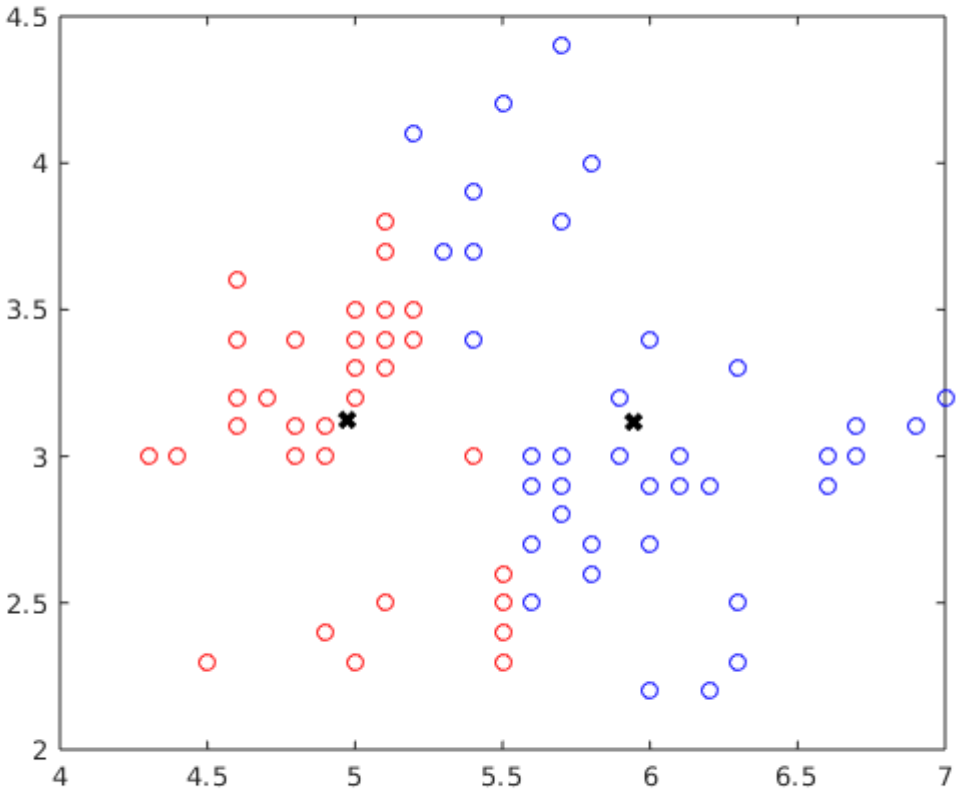


Second iteration

```
n=0;
```

```
m=0;
ii=1;
while ii<=80
    Xi=D(ii,:);
    if norm(Xi-u11','fro')<=norm(Xi-u12','fro')
        R(ii,1)=1;
        n=n+1;
        C21(n,:)=Xi';
    else
        R(ii,2)=1;
        m=m+1;
        C22(m,:)=Xi';
    end
    ii=ii+1;
end
u21= mean(C21,1)';
u22= mean(C22,1)';
u2=[u21 u22]
disp('Size of cluster 1')
disp(length(C21(:,1)))
disp('Size of cluster 2')
disp(length(C22(:,1)))


figure;
plot(C21(1:end,1),C21(1:end,2),'ro')
hold on
plot(C22(1:end,1),C22(1:end,2),'bo')
hold on
plot(u2(1,:),u2(2,:),'kx','LineWidth',3)
```

```
u2 =

    4.9769    5.9463
    3.1256    3.1171

Size of cluster 1
    39

Size of cluster 2
    41
```



Published with MATLAB® R2019b

3.3 Discussion

To adhere to the project report length, only two iterations of K-means clustering have been performed. More iterations can be performed to obtain the converging of the pairs of centroids. A third or fourth iteration could produce a convergence for the initial points.

3.4 Conclusions

The K-means algorithm has been successfully implemented to cluster the samples of 2 species, based on the features of the sample data. The centroid points show a converging trend which means that the points are clustered perfectly. The sample experiment using Iris flower dataset can be extrapolated to different datasets.

4 Linear regression

In this experiment, we investigate a technique for multi-category classification based on binary classifications. The technique is then applied to Fisher's 3-class datasets of *Iris* plants to demonstrate its effectiveness.

4.1 Implementation

The following figures show the MATLAB script used in this lab as well as results with a brief description of how the program is implemented. The program shown was the code used in this lab. It consists of three primary sections. The first section of the code is used to extract the training and testing dataset for the 3 species of flowers.

In the training section, we also assign the **y** and assigning classes such that all data corresponding to Setosa are labeled as the **P**, while the other 2 species are labeled as the **N**. The P and N vectors are reassigned with respective to the other species for the second and third cases. Then the Linear Regression algorithm in which the following weights are computed for the three cases:

$$\hat{w}_1 = (\hat{X}_1 * \hat{X}_1')/(xy_1)$$
$$\hat{w}_2 = (\hat{X}_2 * \hat{X}_2')/(xy_2)$$
$$\hat{w}_3 = (\hat{X}_3 * \hat{X}_3')/(xy_3),$$

where $\hat{\text{w}}$ is the weight and $\hat{\text{X}}$ is the altered input with respect to **P** and **N** vectors in each case.

w_s and b_s are computed from $\hat{\text{w}}$ so as to complete the linear regression equation. In the testing section, the linear regression equation

$$y_{te} = w_s * X_{te} + b_s$$

has been used to compute the mis-classifications that arise while assigning the data to a particular species. A confusion matrix is then computed with the classification data from each species.

4.2 MATLAB Code and Results

Contents

```
clear all
clc

load D_iris_tr
load D_iris_te
```

Training data

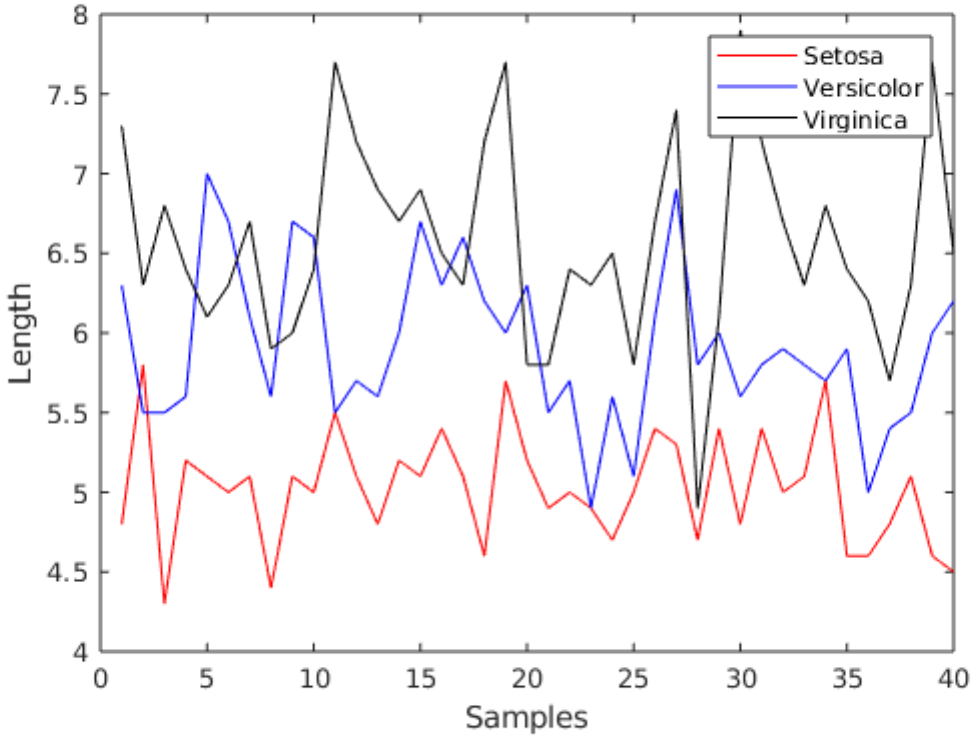
```
X1 = D_iris_tr(:,1:40);
X2 = D_iris_tr(:,41:80);
X3 = D_iris_tr(:,81:120);
```

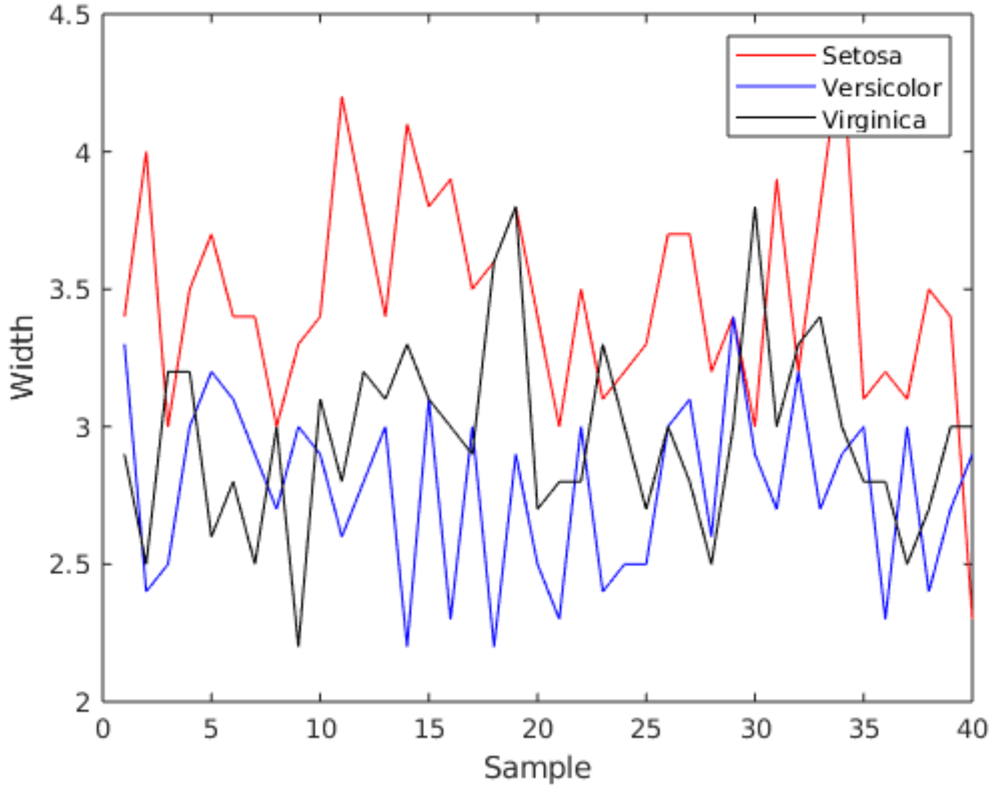
Testing data

```
Test1 = D_iris_te(:,1:10);
Test2 = D_iris_te(:,11:20);
Test3 = D_iris_te(:,21:30);
```

Training

```
PLOT =1;
y = [ones(40,1); -ones(80,1)];

if PLOT == 1
    figure, plot(X1(1,:),'r-');
    hold on;
    plot(X2(1,:),'b-');
    hold on;
    plot(X3(1,:),'k-');
    legend('Setosa','Versicolor','Virginica');
    xlabel('Samples');
    ylabel('Length');

    figure,plot(X1(2,:),'r-');
    hold on;
    plot(X2(2,:),'b-');
    hold on;
    plot(X3(2,:),'k-');
    legend('Setosa','Versicolor','Virginica');
    xlabel('Sample');
    ylabel('Width');
    hold off
end
% First linear model
P = X1;
N = [X2 X3];
X = [P N];
Xh1 = [X' ones(120,1)];
pq1 = Xh1'*y;
wh1 = (Xh1'*Xh1)\pq1;
w1 = wh1(1:4);
b1 = wh1(5);

% Second linear model
P = X2;
N = [X1 X3];
X = [P N];
Xh2 = [X' ones(120,1)];
pq2 = Xh2'*y;
wh2 = (Xh2'*Xh2)\pq2;
w2 = wh2(1:4);
b2 = wh2(5);

%Third linear model
P = X3;
N = [X1 X2];
X = [P N];
Xh3 = [X' ones(120,1)];
pq3 = Xh3'*y;
wh3 = (Xh3'*Xh3)\pq3;
w3 = wh3(1:4);
b3 = wh3(5);
```





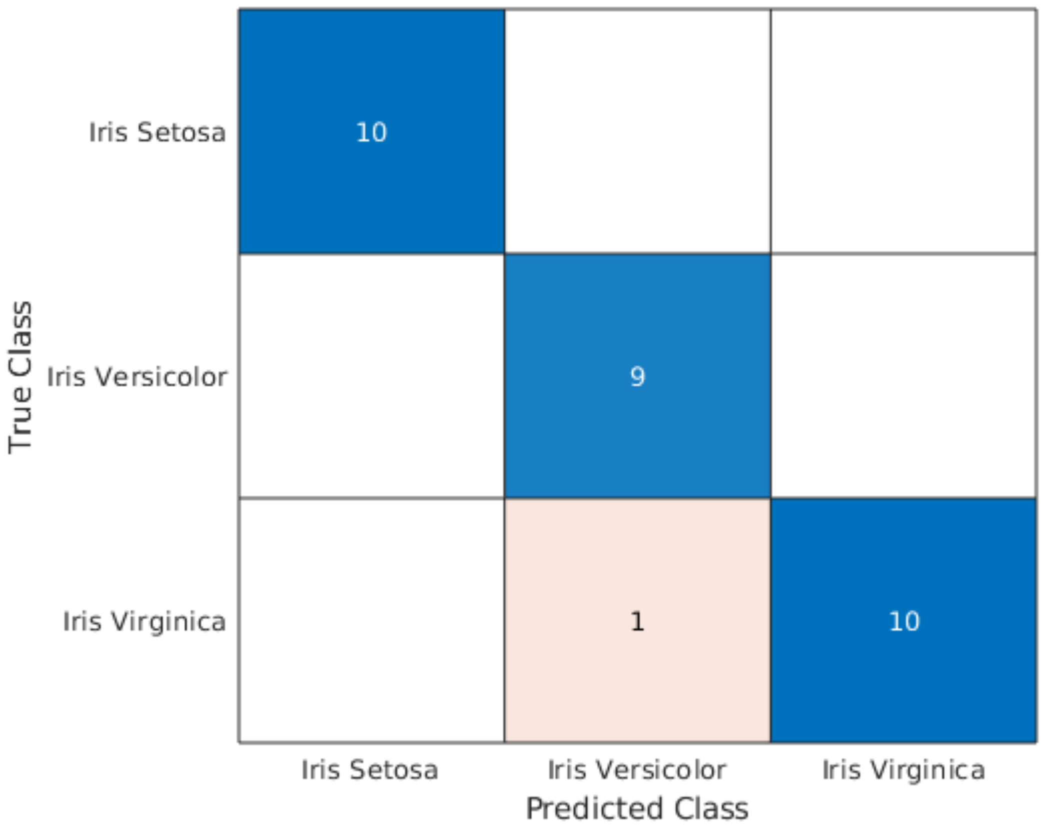
Testing

```
Ws = [w1 w2 w3];
bs = [b1 b2 b3]';
```

```
E = zeros(3,30);
Test = [Test1 Test2 Test3];

%Step 1
miss_class = 0;
yk = zeros(1,30);
yk(1:10) = 1+zeros(1,10);
yk(11:20) = 2+zeros(1,10);
yk(21:30) = 3+zeros(1,10);

t0 = cputime;
%Step 2
for i = 1:30
        xi = Test(:,i);
        fi = Ws'*xi + bs;
       [~,ind] = max(fi);
      % Compare the ground truth with the prediction
        if ind ~= yk(i)
           miss_class = miss_class + 1;
        end
        E(ind,i) = 1;
end

cpt = cputime - t0;

disp('Number of misrepresented classes is : ')
disp(miss_class);
disp('Error rate is : ');
disp(miss_class/30);

%Step 3
E1 = E(:,1:10);
c1 = sum(E1')';
E2 = E(:,11:20);
c2 = sum(E2')';
E3 = E(:,21:30);
c3 = sum(E3')';

disp('Confusion matrix : ');
C = [c1 c2 c3];
display(C);

classLabels = {'Iris Setosa','Iris Versicolor','Iris Virginica'};
figure
Cc = confusionchart(C,classLabels);

disp('CPU time (s):')
cpt
```

```
Number of misrepresented classes is :
     1

Error rate is :
    0.0333
```

```
Confusion matrix :

C =

    10     0     0
     0     9     0
     0     1    10

CPU time (s):

cpt =

     0
```



Published with MATLAB® R2019b

4.3 Discussion

The best classification results was one mislabeled flower which was attained by only using the fourth feature. From the confusion matrix, it has been found that the only flower that was mis-classified was *Iris versicolor*, which was labeled as *Iris virginica.* It is clear that there is quite a lot of correlation between the different flower species features, which is why the one dominant feature is essentially the only feature that is required to classify the different flower species.

4.4 Conclusions

The binary classification model was successful in classifying the different *Iris* flower species in the final test dataset with only one mislabelled flower. One of the disadvantages of using this linear model is the fact that it is linear and therefore can only build models with two inputs. For this experiment we used positive and negative values for inputs to distinguish the different flower species. This worked however it would be disadvantageous if working with more than three classes as you need to compare each case and a different algorithm would be better suited.

5 Summary

The dataset of Iris plants has been studied using three algorithms namely, the PCA, where the species were classified according to their petal length, width or their sepal length and width; with linear regression, where they were classified using all features at once (multi-category classification) based on binary classifications; and the K-means algorithm, where two features were chossen and clustered for two species.