Dimensionality reduction using FDA for Bayes Classifier

19EAC381 Machine Learning Lab with Python Project report

Submitted by

Kuruba Aparna
(AM.EN.U4EAC19032)

Lekshmi Manoj
(AM.EN.U4EAC19033)

Maddineni Vagdevi
(AM.EN.U4EAC19034)

Mallu Srihith Reddy
(AM.EN.U4EAC19035)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
AMRITA SCHOOL OF ENGINEERING AMRITAPURI
AMRITAPURI, KOLLAM
KERALA, INDIA 690525.

Contents

Abstract

In this study, we have implemented dimensionality reduction using Fisher Discriminant Analysis on a two class classification problem for Bayes Classifier. Initially, the accuracy values provided by a bayes classifier without dimensionality reduction were found out. The Multivariate Gaussian approach was taken into account to achieve this, The posterior probability associated to each class was found out which was then classified to either of the classes making use of the maximum likelihood approach.

The same experiment was then performed using FDA where the feature vectors were projected onto a set of new dimensions and the accuracy values were plotted with respect to the changing number of projection vectors.We used around 40 different values of eigenvectors for projection so that the inferences could be more accurate.

Another objective was to compare the accuracy values with the changing covariance matrices. The three types of covariance matrices: full, diagonal and spherical were brought into consideration and the accuracy values with the changing eigenvectors were plotted.The images that got misclassified were displayed with their actual class label and the predicted class labels. Lastly, the result obtained through dimensionality reduction was compared to the normal bayes classifier and the inferences are noted.

Chapter 1

Introduction

1.1 Introduction

Machine Learning is one of the most exciting technologies that one would have ever come across. It is actively being used today, perhaps in many more places than one would expect. As it is evident from the name, it gives the computer that makes it more similar to humans: the ability to learn.Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches in machine learning : supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.The type of algorithm chosen depends on what type of data we wish to predict.

Here for this experiment,the provided dataset contains training images of 2 characters in a folder named TrainCharacters.zip. There are 200 training images of size 128×128 for each character class. For evaluating the classifiers,there are 300 test images of size 128×128 in a separate folder TestCharacters.zip. We use the supervised leaning algothim which is the Bayes Classifier to classifier the two sets of characters 'e' and 'c'. Dimensionality reduction is the main objective of the study. It often helps in data compression, and hence reduced storage space and reduces computation time. I also helps remove redundant features, if any. Dimensionality reduction is planned to be achieved through FDA.

Chapter 2

Background Study

2.1 Relevant theory

Bayes classifier:

A Bayes Classifier is a type of generative classifier in which the samples of training data of a class are assumed to come from a probability density function. A suitable decision making process when all related probability distributions are given. It is a supervised learning paradigm (Labelled training data set). Supervised learning is a machine learning task in which a function maps the input to output data using the provided input-output pairs. Designed with Bayes theorem with appropriate decision surfaces. It assumes that the occurrence of a certain feature is independent of the occurrence of other features. It is mainly used in text classification which includes a high-dimensional training dataset. It is mainly used in spam filtration, sentimental analysis, and classifying articles.

It can be used for Binary as well as Multi-class Classifications. It performs well in Multi-class predictions as compared to the other Algorithms. It is the most popular choice for text classification problems. It is based on the Bayes Rule which states that the conditional probability is the likelihood of an outcome occurring, based on a previous outcome occurring.

$$\Pr(y|x) = \frac{Pr(x|y)Pr(y)}{Pr(x)}$$

wherein, x refers to feature value and y refers to class label.

Pr(y/x) = Posterior: what we know about y after seeing x

Pr(x/y) = Likelihood: propensity for observing a certain value of x given a certain value of y

Pr(y) = Prior: what we know about y before seeing x

Pr(x) = Evidence: constant to ensure lhs is a valid distribution

If we consider a two class classification problem, if we do the classification depending solely on the prior probability, no feature is selected for classification but only prior probability. Therefore, regardless of whichever pattern that comes in, the solution will assign it as the one with higher prior probability.

So the solution to this is to look for additional discriminative features. Select features like length, width, colour, texture etc that distinguish each type. Assume we have d features. Set of features represented by d dimensional feature vector x. The class conditional density is the probability distribution function associated with the d features for each category of patterns. For a set of d features there will be a d dimensional probability distribution. The multivariate gaussian function is considered here.

The multivariate Gaussian distribution generalizes the one-dimensional Gaussian distribution to higher-dimensional data. In the absence of information about the real distribution of a dataset, it is usually a fine choice to assume that data is normally distributed. The posterior probability is found and using the maximum likelihood approach the classification is proceeded. Thus the Bayes decision rule can be interpreted as calling for deciding a particular class if the likelihood ratio exceeds a threshold value that is independent of the observation of the pattern.

Dimensionality reduction:

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically the features. The higher the number of features, the harder it gets to visualize the training set and then work on it and more input features often make a predictive modeling task more challenging to model, more generally referred to as the curse of dimensionality. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play.

It is said that each feature contributes to reducing the probability of error. Naturally, the most useful features are the ones for which the difference between the means is large relative to the standard deviations. However no feature is useless if its means for the two classes differ. An obvious way to reduce the error rate further is to introduce new, independent features. Each new feature need not add much, but if r can be increased without limit, the probability of error can be made arbitrarily small.

Although increasing the number of features increases the cost and complexity of both the feature extractor and the classifier, it is often reasonable to believe that the performance will improve. After all, if the probabilistic structure of the problem were completely known, the Bayes risk could not possibly be increased by adding new features. At worst, the Bayes classifier would ignore the new features, but if the new features provide any additional information, the performance must improve. Unfortunately, it has frequently been observed in practice that, beyond a certain point, the inclusion of additional features leads to worse rather than better performance. This apparent paradox presents a genuine and serious problem for classified design.

Components of Dimensionality Reduction:

There are two components of dimensionality reduction: Feature selection: In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. Feature extraction: This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction:

The various methods used for dimensionality reduction include:
Principal Component Analysis (PCA)
Linear Discriminant Analysis (LDA)
Generalized Discriminant Analysis (GDA)

Linear Discriminant Analysis:

LDA is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space. For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification.

Comparing PCA and LDA:

PCA for short is a commonly used unsupervised linear transformation technique. PCA reduces the number of dimensions by finding the maximum variance in high dimensional data. LDA for short is a supervised method that takes class labels into account when reducing the number of dimensions. The goal of LDA is to find a feature subspace that best optimizes class separability.

Fisher Discriminant Analysis: Fisher discriminant Analysis is one way to view a

linear classification model in terms of dimensionality reduction. FDA, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension. However, the projected data can subsequently be used to construct a discriminant, in a two class classification problem, by choosing a threshold y0 so that we classify a new point as belonging to class 1 if y(x) greater than y0 and classify it as belonging to class two otherwise.

In the case of a two class classification problem, assume we take the D dimensional input vector x and project it down to one dimension using y.

$$y = \ w^T x$$

If we place a threshold on y and classify y:w0 as class C1, and otherwise class C2, then we obtain our standard linear classifier

In general, the projection onto one dimension leads to a considerable loss of information, and classes that are well separated in the original D-dimensional space may become strongly overlapping in one dimension.However, by adjusting the components of the weight vector w, we can select a projection that maximizes the class separation.

Let us consider a two-class problem in which there are N1 points of class C1 and N2 points of class C2, so that the mean vectors of the two classes are given by,

$$m_1 = 1/N_1(\sum_{n\in C_1} x_n)$$

$$m_2 = 1/N_2(\sum_{n\in C_2} x_n)$$

So the simplest measure of the separation of the classes when projected onto W is the separation of the projected class means.

$$m_2 - m_1 = w^T(m_2 - m_1)$$

Where,

$$m_k = w^T m_k$$

The within-class variance of the transformed data from class Ck is therefore given by

$$s_k^2 = \sum_{n\in C_k} (y_n - m_k)^2$$

The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given b

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

We can make the dependence on w to rewrite the Fisher criterion in the form

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Where S_B is the between-class covariance matrix and is given by

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

S_W is the total within-class covariance matrix, given by

$$S_W = \sum_{n\in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n\in C_2} (x_n - m_2)(x_n - m_2)^T$$

By differentiating with respect to W we get,

$$(w^T S_B w) S_W w = (w^T S_W w) S_B w$$

Chapter 3

Block diagram and Work done

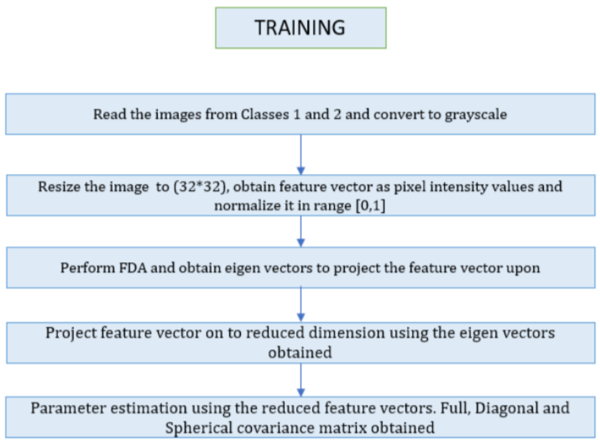

TRAINING
Read the images from Classes 1 and 2 and convert to grayscale
Resize the image to (32*32), obtain feature vector as pixel intensity values and normalize it in range [0,1]
Perform FDA and obtain eigen vectors to project the feature vector upon
Project feature vector on to reduced dimension using the eigen vectors obtained
Parameter estimation using the reduced feature vectors. Full, Diagonal and Spherical covariance matrix obtained

Figure 3.1: The Training Phase

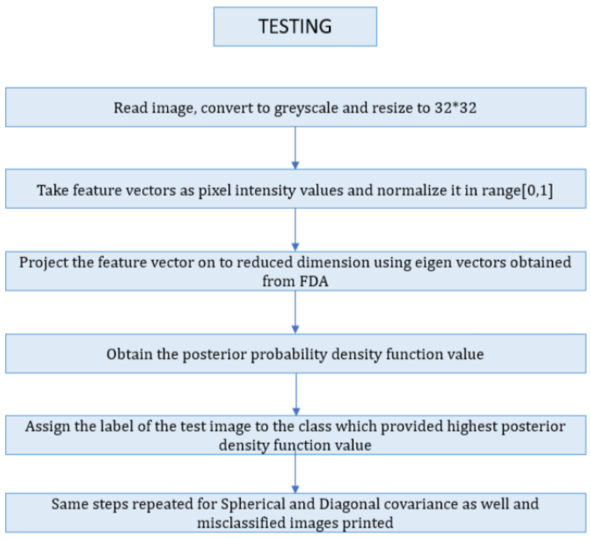

TESTING
Read image, convert to greyscale and resize to 32*32
Take feature vectors as pixel intensity values and normalize it in range[0,1]
Project the feature vector on to reduced dimension using eigen vectors obtained from FDA
Obtain the posterior probability density function value
Assign the label of the test image to the class which provided highest posterior density function value
Same steps repeated for Spherical and Diagonal covariance as well and misclassified images printed

Figure 3.2: The Testing Phase

3.1 Work done

In the training part, there are two classes with 200 images in each of them. The path of the classes were assigned and the images from both the folders were read on greyscale one after the other. The images were resized to 32*32 and reshaped to 1024*1. This is followed by the normalisation of the image pixel intensity value to the range [0,1]. The mean value corresponding to each class was found.

To begin with dimensionality reduction using Fisher Discriminant Analysis, the scatter matrix was found out using the mean value. Then adding up the scatter matrix for both the classes would give the within class scatter matrix. Another value to be found out is the between class scatter matrix, that is also obtained by multiplying the subtracted means of the two classes and its transpose. The inverse value of within class scatter matrix is found out which is then multiplied with the between class scatter matrix value. Hence we get an array of size 1024*1024 and we then find out the eigenvalues and eigenvectors of the same using linalg. The eigenvectors are then sorted in the descending order.

For projecting onto the desired dimension, the feature vector corresponding to each image is found out from the testing set for both classes. The desired number of eigenvectors (N) for projection is being chosen and the feature values are projected to a newer dimension.This is done by subtracting the initial mean value from feature value and finding out the projection vector corresponding to each image. The reduced feature vectors are then used in the parameter estimation part where the mean and covariance with respect to the new dimension is found out. The mean would be of size (N*1) and the covariance would be of size (N*N).

The three sets of covariance values are found out including full covariance, diagonal covariance and spherical covariance. The diagonal covariance is obtained by multiplying the covariance matrix with an identity matrix. The spherical covariance has the diagonal elements the same which is the variance and the off diagonal elements as zero.

In the testing phase, the test folders are mounted and the images read as greyscale, resized, reshaped and normalised. The feature vector corresponding to each image is found out and the dimensionality reduction is achieved using the eigenvectors found before. The feature values are projected to the desired (N) dimension. These projected feature vectors are then passed to the gauss function for finding the posterior probability. The parameters to the function are the mean values found out using reduced feature vectors, covariance and the reduced feature vectors. The posterior probability corresponding to the two classes are appended to an array. The maximum value among that is chosen and the index corresponding to that is checked to classify it to the correct class. The same step is repeated for both the classes and the accuracy corresponding to both the classes are found out.

The number of eigenvectors are changed and the same steps are repeated to achieve the first objective of the experiment. The misclassification is also found out and the misclassified images are printed. The covariance matrices are changed in the gauss function to obtain the second objective of the experiment. A comparison with normal bayes classifier is done with respect to labsheet 3 as the last objective.

Chapter 4

Simulation results and inferences

4.1 Experimental results

Objective 1:Variation of acccuracy with number of eigen vectors for projection

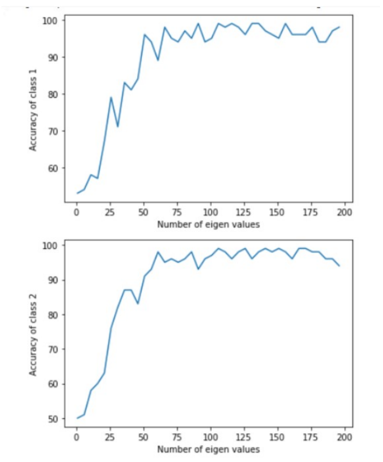


Figure 4.1: Variation of accuracy with the change in eigenvectos for both the classes

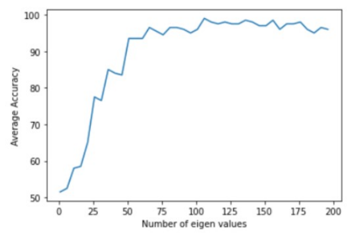


Figure 4.2: Average accuracy

Inference 1:

The highest accuracy obtained at eigenvectors chosen are 110 .The variation of accuracy with respect to the increasing values of eigenvectors seems to be increasing with the increase in eigen vectors for projection. The result is plotted for the first 40 eigenvectors that come in range 1 to 200 with a difference of 5. The bayes classifier hence tends to provide a better accuracy as the information increases with respect to the increasing number of eigenvectors.

Objective 2: Variation of accuracy with type of covarience matrix (full,diagonal,spherical)

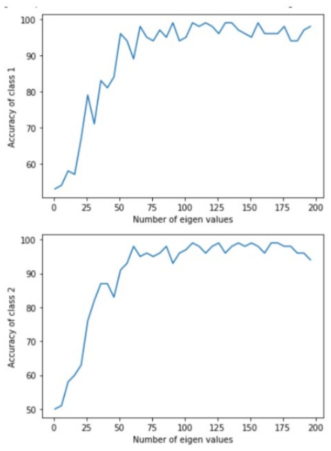


Figure 4.3: Class wise Accuracy of Full Covariance Matrix

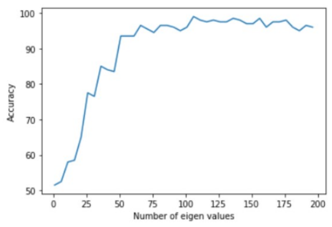


Figure 4.4: Average Accuracy of Full Covariance Matrix

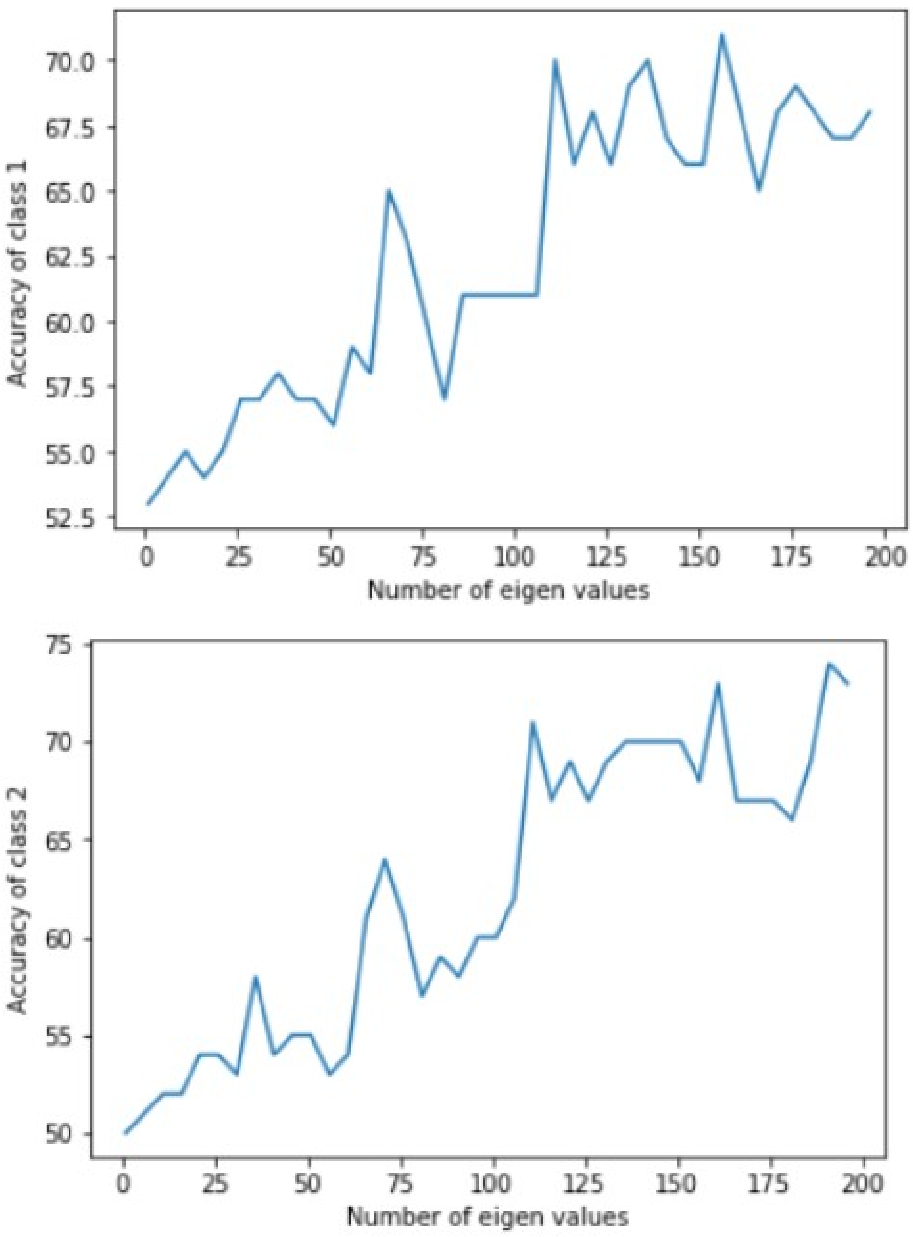


Fig 4.5. Class wise Accuracy of Diagonal Covariance Matrix

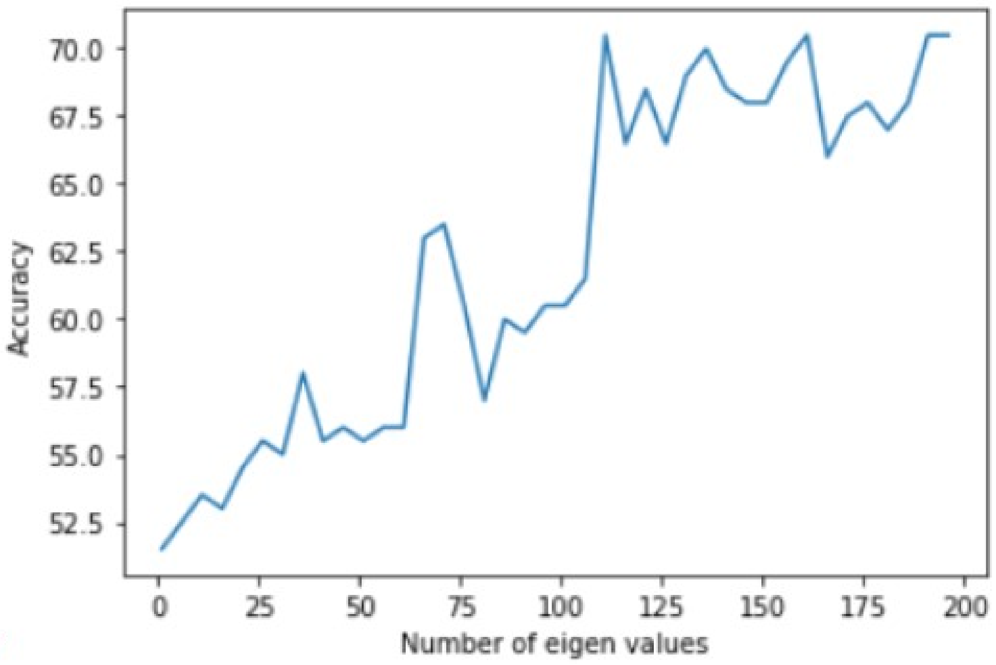
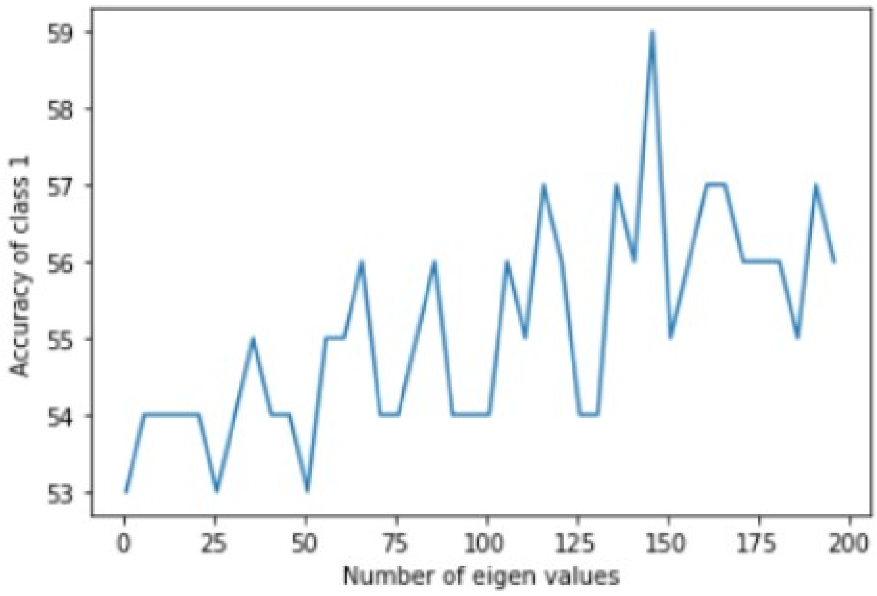


Fig 4.6.Average Accuracy of Diagonal Covariance Matrix



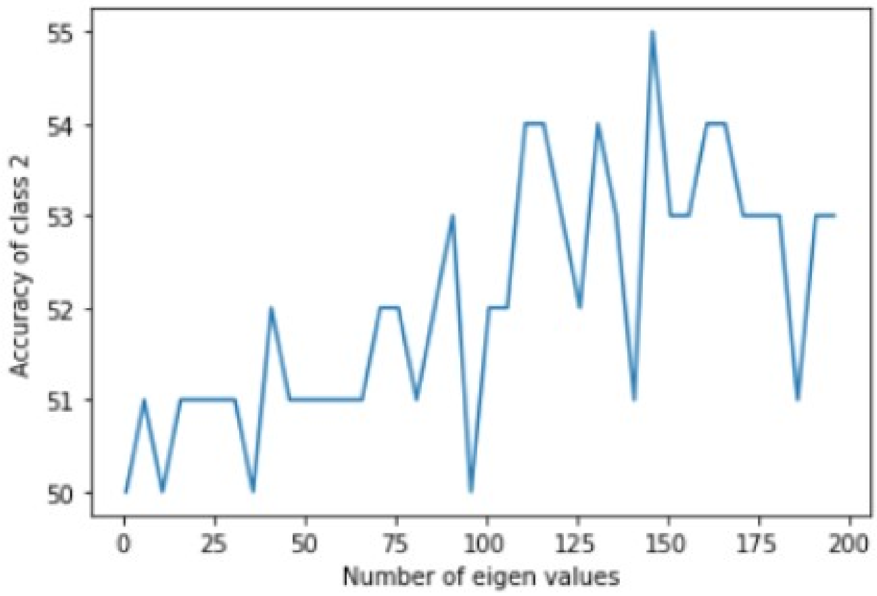


Fig 4.7. Class wise Accuracy of Spherical Covariance Matrix

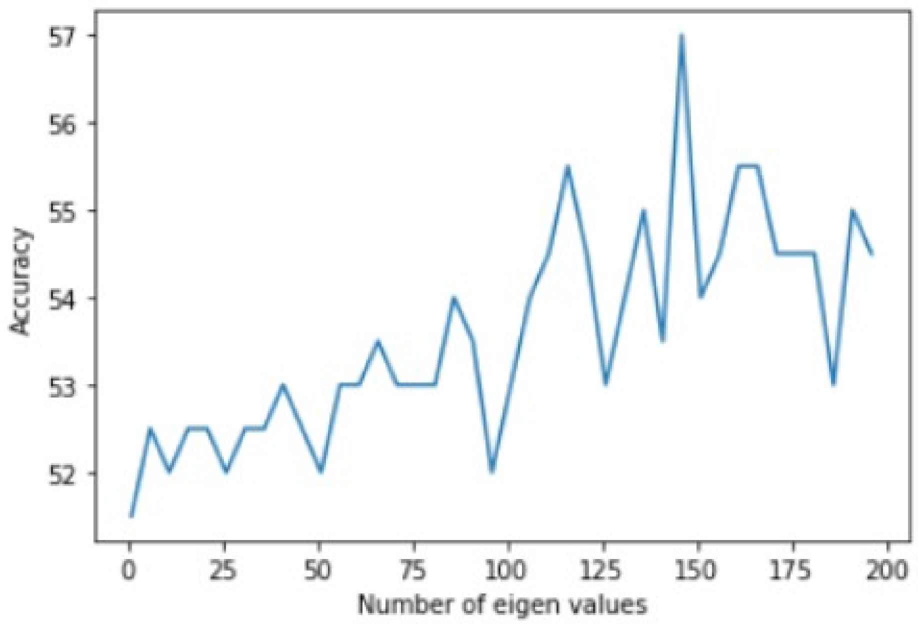


Fig 4.8. Average Accuracy of Spherical Covariance Matrix

Inference 2:

The variation of accuracy with the change in covariance matrix is such that it is highest for full covariance matrix, followed by diagonal covariance matrix and the least accuracy was plotted for spherical covariance matrix. If the features are statistically independent, there are some theoretical results that suggest the possibility of excellent performance. Thus, the probability of error decreases as N increases, approaching zero as N approaches infinity.This shows how each feature contributes to reducing the probability of error.

Objective 3: Some examples of misclassified samples

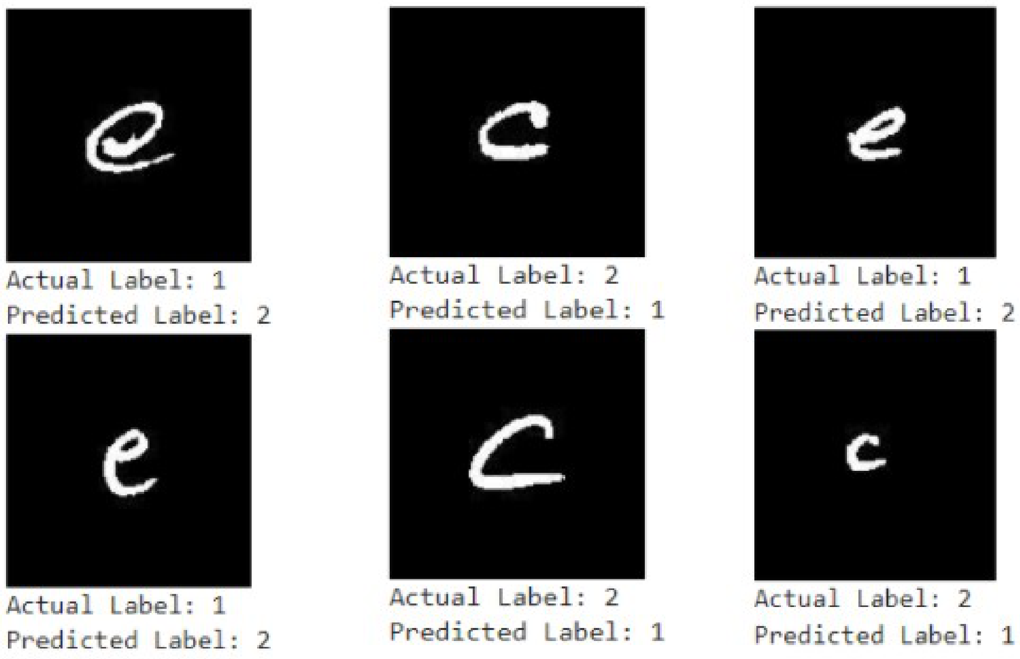


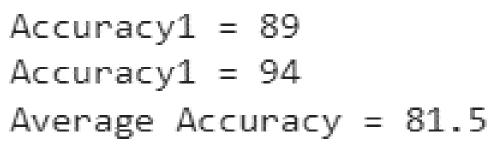
Fig 4.9. Misclassified Images

Inference 3:

The misclassified images are obtained for four cases with regards to the full covariance matrix. The predicted class and the actual class are also mentioned.

Objective 4: Comparision with Bayes Classifier without dimensionality reduction

For Bayes Classifier without Dimensionality Reduction:



```
Accuracy1 = 89
Accuracy1 = 94
Average Accuracy = 81.5
```

Fig 4.10. Accuracy of bayes classifier without dimensionality reduction

Inference 4:

On comparing the output with the bayes classifier without dimensionality reduction, where the accuracy values for normal bayes classifier was 89 and 94 for respectively considering the pooled covariance scenario. The average accuracy of the FDA reduced values seems to be equal at the number of eigenvalues reaching 50 and 55. Overall the accuracy values seem to be increasing with the increasing number of vectors for projection.

4.2 Conclusion

The Bayes classifier chooses the class that has the greatest a posteriori probability of occurrence called maximum a posteriori estimation. It can be shown that of all classifiers, Bayes classifier is the one that will have the lowest probability of miss classifying an observation, i.e. the lowest probability of error. So if we know the posterior distribution, then using the Bayes classifier is as good as it gets.

In general, the projection onto one dimension leads to a considerable loss of information, and classes that are well separated in the original D-dimensional space may become strongly overlapping in one dimension. However, by adjusting the components of the weight vector w, we can select a projection that maximizes the class separation. To be precise, the idea is that it maximizes a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap.

The most useful features are the ones for which the difference between the means is large relative to the standard deviations. However no feature is useless if its means for the two classes differ. An obvious way to reduce the error rate further is to introduce new, independent features. Each new feature need not add much, but if r can be increased without limit, the probability of error can be made arbitrarily small. So as the number of information the feature vector provides which is decided by the number of eigenvectors

Bibliography

[1] Pattern Classification, 2nd Edition (2000),Richard O. Duda, Peter E. Hart, David G. Stork,Wiley

[2] https://sthalles.github.io/fisher-linear-discriminant/

[3] https://scikit-learn.org/stable/auto-examples/mixture/plot-gmm-covariances.htmlsphx-glr-auto-examples-mixture-plot-gmm-covariances-py

Appendix

Python Code

```
from google.colab import drive
drive.mount('/content/gdrive/')
%cd /content/gdrive/MyDrive/Machine Learning/LAB 3
!ls
```

```
Mounted at /content/gdrive/
/content/gdrive/MyDrive/Machine Learning/LAB 3
'Copy of ML.LS.3.19033.ipynb'   ML.PROJECT.ROUGH.ipynb   TrainCharacters
 ML.LS.3.19033.ipynb            TestCharacters
```

```
def gauss(x,m,c):
  w_1 = -0.5*(math.log(np.linalg.det(c),np.e))
  d=x.shape[0]
  u=x-m
  w1=np.linalg.inv(c)
  w_1_1 = -0.5*np.matmul(u.transpose(),np.matmul(w1,u))
  w_1_0 = -0.5*d*math.log(2*np.pi,np.e) + math.log(1/3.0,np.e)
  g = w_1+w_1_0+w_1_1
  return (g[0][0])
```

```
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import glob
```

```
import math
import numpy as np
import cv2
from matplotlib import pyplot as plt

xi_max=255.0
lamda=0.5
I=np.identity(1024)

path1="/content/gdrive/MyDrive/Machine Learning/LAB 3/TrainCharacters/TrainCharacters/1/*.jpg"
path2="/content/gdrive/MyDrive/Machine Learning/LAB 3/TrainCharacters/TrainCharacters/2/*.jpg"

#x1=np.array([])
#x2=np.array([])
#A=np.array([])
s1 = np.zeros((1024,1024))
s2 = np.zeros((1024,1024))

Mu1=np.zeros((1024,1))
for i in glob.glob(path1):
     read_image1=cv2.imread(i,0)
     image_resize1=cv2.resize(read_image1,(32,32))
     reshaped1=np.reshape(image_resize1,(1024,1))
     norm1=reshaped1/xi_max
     #x1=np.append(x1,norm1)
     #A=np.append(A,norm1)
     Mu1=Mu1+norm1

Mu1=Mu1/(200)
```

```
Mu1=Mu1/(200)
print(Mu1.shape)
for i in glob.glob(path1):
     read_image1=cv2.imread(i,0)
     image_resize1=cv2.resize(read_image1,(32,32))
     reshaped1=np.reshape(image_resize1,(1024,1))
     norm1=reshaped1/xi_max
     s1+=np.dot((norm1-Mu1),((norm1-Mu1).T))
s1=s1+lamda*I


Mu2=np.zeros((1024,1))
for i in glob.glob(path2):
     read_image2=cv2.imread(i,0)
     image_resize2=cv2.resize(read_image2,(32,32))
     reshaped2=np.reshape(image_resize2,(1024,1))
     norm2=(reshaped2/xi_max)
     #x2=np.append(x2,norm2)
     #A=np.append(A,norm2)
     Mu2=Mu2+norm2
Mu2=Mu2/(200)
print(Mu2.shape)

for i in glob.glob(path2):
     read_image2=cv2.imread(i,0)
     image_resize2=cv2.resize(read_image2,(32,32))
     reshaped2=np.reshape(image_resize2,(1024,1))
     norm2=(reshaped2/xi_max)
     s2 += np.dot((norm2-Mu2),((norm2-Mu2).T))
s2=s2+lamda*I
```

```
(1024, 1)
(1024, 1)
```

```
S_W = s1+s2
#print(S_W,S_W.shape)
S_B= np.dot((Mu1-Mu2),((Mu1-Mu2).T))
#print(S_B,S_B.shape)
S_Winv = np.linalg.inv(S_W)
Y= np.dot(S_Winv,S_B)
(Y,Y.shape)
```

```
(array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]), (1024, 1024))
```

```
w,v=np.linalg.eig(Y)
#print(w)
#print(v)
v=v.real
#w=w.real
index = w.argsort()[::-1]
```

```
w=w[index]
v=v[:,index]

lek1=[]
lek2=[]
val=255.0
for N in range(1,200,5):
    I=np.identity(N)
    project_array2=np.array([])
    project_array1=np.array([])
    for j in glob.glob(path1):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x3=np.reshape(norm,(1024,1))
        z3=x3-Mu1
        for i in range(N):
            projection1=np.dot((v[:,i]).T,z3)
            project_array1=np.append(project_array1,projection1)
    project_array1=np.reshape(project_array1,(200,N))
    project_array1=project_array1.T
    Mu3=np.sum(project_array1,axis=1)
    mean1=np.reshape(Mu3,(N,1))
    for j in glob.glob(path2):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x4=np.reshape(norm,(1024,1))
        z4=x4-Mu2
        for i in range(N):
            projection2=np.dot((v[:,i]).T,z4)
            project_array2=np.append(project_array2,projection2)
    project_array2=np.reshape(project_array2,(200,N))
    project_array2=project_array2.T
    Mu4=np.sum(project_array2,axis=1)
    mean2=np.reshape(Mu4,(N,1))
    Mu_pooled=(Mu3+Mu4)/400
    Mu_pooled=np.array(Mu_pooled)
    #s3=np.zeros((N,N))
    s3=0
    for i in range(200):
        s=project_array1[:,i]-Mu_pooled
        s3=s3+np.dot(s,s.T)
    for i in range(200):
       s=project_array2[:,i]-Mu_pooled
       s3=s3+np.dot(s,s.T)
    s3=s3/400
    s3=s3+lamda*I
    path3="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/1/*.jpg"
    path4="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/2/*.jpg"

    index1=0
    index2=0

    cnt1=0
    cnt2=0

    l1=[]
```

```
        l2=[]

        actual_label1=[]
        actual_label2=[]

        predicted_class_label1=[]
        predicted_class_label2=[]

        new_array1=np.zeros(100)
        new_array2=np.zeros(100)

        for j in glob.glob(path3):
            read_image=cv2.imread(j,0)
            image_resize=cv2.resize(read_image,(32,32))
            norm=np.divide(image_resize,xi_max)
            x4=np.reshape(norm,(1024,1))
            z5=x4-Mu1
            project_array3=np.array([])
            for i in range(N):
                projection3=np.dot((v[:,i]).T,z5)
                project_array3=np.append(project_array3,projection3)
            arr1=[0,0]
            arr1[0]=gauss(project_array3,mean1,s3)
            arr1[1]=gauss(project_array3,mean2,s3)
            max1=max(arr1)
            for i in range(2):
                  if (max1==arr1[i]):
                     new_array1[index1]=i+1
                     if (new_array1[index1]==1):
                          cnt1=cnt1+1
                     else:
                          l1.append(j)
                          actual_label1.append(1)
                          predicted_class_label1.append(i+1)
            index1=index1+1
        #print("Accuracy1=",cnt1)
        lek1.append(cnt1)
        for j in glob.glob(path4):
          read_image=cv2.imread(j,0)
          image_resize=cv2.resize(read_image,(32,32))
          norm=np.divide(image_resize,val)
          x7=np.reshape(norm,(1024,1))
          z7=x7-Mu2
          project_array6=np.array([])
          for k in range(N):
              projection6=np.dot((v[:,k]).T,z7)
              project_array6=np.append(project_array6,projection6)
          arr2=np.zeros(2)
          arr2[0]=gauss(project_array6,mean1,s3)
          arr2[1]=gauss(project_array6,mean2,s3)
          max2=np.max(arr2)
          for i in range(2):
                 if (max2==arr2[i]):
                     new_array2[index2]=i+1
                     if (new_array2[index2]==2):
                          cnt2=cnt2+1
                     else:
                          l2.append(j)
```

```
                    actual_label2.append(2)
                    predicted_class_label2.append(i+1)
      index2=index2+1
    #print("Accuracy2=",cnt2)
    lek2.append(cnt2)


print("Accuracy of Class 1 : ",lek1)
print("Accuracy of Class 2 : ",lek2)
```

```
Accuracy of Class 1 :  [53, 54, 58, 57, 67, 79, 71, 83, 81, 84, 96, 94, 89, 98, 95, 
Accuracy of Class 2 :  [50, 51, 58, 60, 63, 76, 82, 87, 87, 83, 91, 93, 98, 95, 96, 
```

```
avac=[]
for i in range(len(lek1)):
  p=lek1[i]+lek2[i]
  p=p/2
  avac.append(p)
print("Accuracy of Class 1 : ",lek1)
print("Accuracy of Class 2 : ",lek2)
print("Average Accuracy :",avac)
```

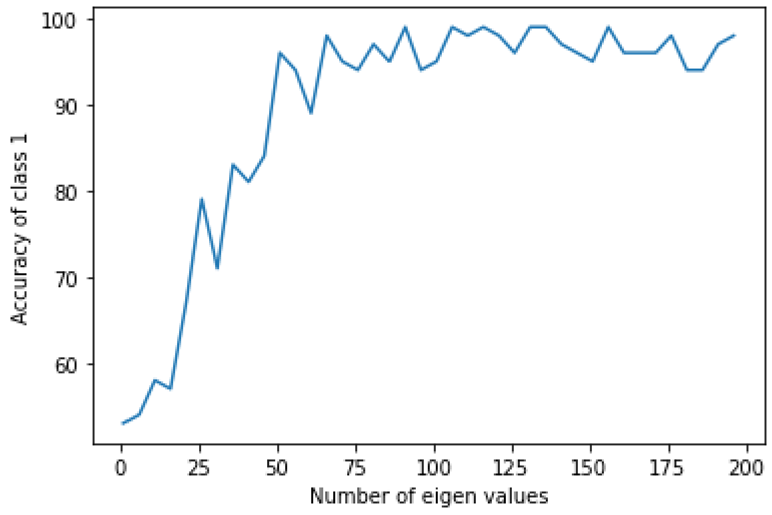
```
Accuracy of Class 1 :  [53, 54, 58, 57, 67, 79, 71, 83, 81, 84, 96, 94, 89, 98, 95, 
Accuracy of Class 2 :  [50, 51, 58, 60, 63, 76, 82, 87, 87, 83, 91, 93, 98, 95, 96, 
Average Accuracy : [51.5, 52.5, 58.0, 58.5, 65.0, 77.5, 76.5, 85.0, 84.0, 83.5, 93.5
```

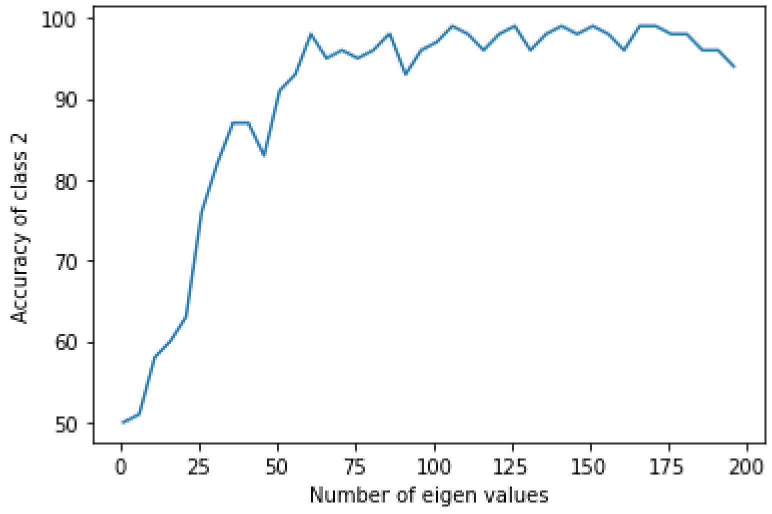
```
print("Accuracy1 = 89")
print("Accuracy1 = 94")
print("Average Accuracy = 81.5")
```

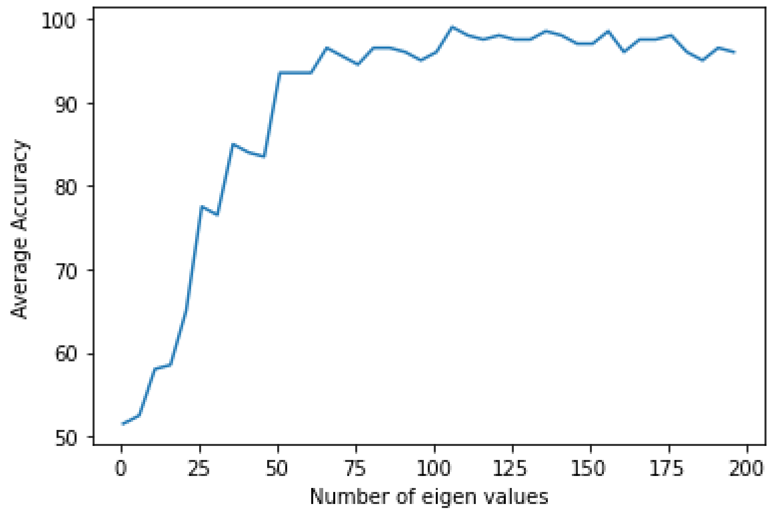
```
Accuracy1 = 89
Accuracy1 = 94
Average Accuracy = 81.5
```

```
import matplotlib.pyplot as plt

x=np.arange(1,200,5)
plt.figure(1)
plt.xlabel("Number of eigen values")
plt.ylabel("Accuracy of class 1")
plt.plot(x,lek1)
plt.figure(2)
plt.xlabel("Number of eigen values")
plt.ylabel("Accuracy of class 2")
plt.plot(x,lek2)
plt.figure(3)
plt.xlabel("Number of eigen values")
plt.ylabel("Average Accuracy")
plt.plot(x,avac)
```

```
[<matplotlib.lines.Line2D at 0x7f5885938250>]
```

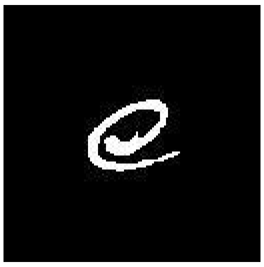




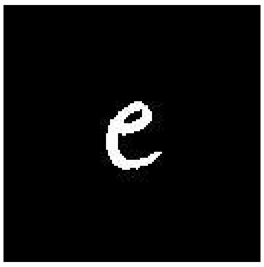


```
from google.colab.patches import cv2_imshow

newimage1=cv2.imread(l1[0],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage1)
print("Actual Label:",actual_label1[0])
```

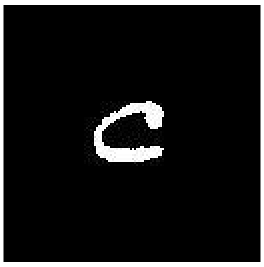
```
    print("Predicted Label:",predicted_class_label1[0])

    newimage2=cv2.imread(l1[1],cv2.IMREAD_UNCHANGED)
    cv2_imshow(newimage2)
    print("Actual Label:",actual_label1[1])
    print("Predicted Label:",predicted_class_label1[1])

    newimage3=cv2.imread(l2[0],cv2.IMREAD_UNCHANGED)
    cv2_imshow(newimage3)
    print("Actual Label:",actual_label2[0])
    print("Predicted Label:",predicted_class_label2[0])

    newimage4=cv2.imread(l2[1],cv2.IMREAD_UNCHANGED)
    cv2_imshow(newimage4)
    print("Actual Label:",actual_label2[1])
    print("Predicted Label:",predicted_class_label2[1])

    newimage2=cv2.imread(l1[3],cv2.IMREAD_UNCHANGED)
    cv2_imshow(newimage2)
    print("Actual Label:",actual_label1[3])
    print("Predicted Label:",predicted_class_label1[3])

    newimage4=cv2.imread(l2[3],cv2.IMREAD_UNCHANGED)
    cv2_imshow(newimage4)
    print("Actual Label:",actual_label2[3])
    print("Predicted Label:",predicted_class_label2[3])
```



```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 2
```

```
lek1=[]
lek2=[]
val=255.0
for N in range(1,200,5):
    I=np.identity(N)
    project_array2=np.array([])
    project_array1=np.array([])
    for j in glob.glob(path1):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x3=np.reshape(norm,(1024,1))
        z3=x3-Mu1
        for i in range(N):
            projection1=np.dot((v[:,i]).T,z3)
            project_array1=np.append(project_array1,projection1)
    project_array1=np.reshape(project_array1,(200,N))
    project_array1=project_array1.T
    Mu3=np.sum(project_array1,axis=1)
    mean1=np.reshape(Mu3,(N,1))
    for j in glob.glob(path2):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x4=np.reshape(norm,(1024,1))
        z4=x4-Mu2
        for i in range(N):
            projection2=np.dot((v[:,i]).T,z4)
            project_array2=np.append(project_array2,projection2)
    project_array2=np.reshape(project_array2,(200,N))
```

```
project_array2=project_array2.T
Mu4=np.sum(project_array2,axis=1)
mean2=np.reshape(Mu4,(N,1))
Mu_pooled=(Mu3+Mu4)/400
Mu_pooled=np.array(Mu_pooled)
#s3=np.zeros((N,N))
s3=0
for i in range(200):
    s=project_array1[:,i]-Mu_pooled
    s3=s3+np.dot(s,s.T)
for i in range(200):
   s=project_array2[:,i]-Mu_pooled
   s3=s3+np.dot(s,s.T)
s3=s3/400
s3=s3+lamda*I
s4=s3*I
path3="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/1/*.jpg"
path4="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/2/*.jpg"

index1=0
index2=0

cnt1=0
cnt2=0

l1=[]
l2=[]

actual_label1=[]
actual_label2=[]

predicted_class_label1=[]
predicted_class_label2=[]

new_array1=np.zeros(100)
new_array2=np.zeros(100)

for j in glob.glob(path3):
    read_image=cv2.imread(j,0)
    image_resize=cv2.resize(read_image,(32,32))
    norm=np.divide(image_resize,xi_max)
    x4=np.reshape(norm,(1024,1))
    z5=x4-Mu1
    project_array3=np.array([])
    for i in range(N):
        projection3=np.dot((v[:,i]).T,z5)
        project_array3=np.append(project_array3,projection3)
    arr1=[0,0]
    arr1[0]=gauss(project_array3,mean1,s4)
    arr1[1]=gauss(project_array3,mean2,s4)
    max1=max(arr1)
    for i in range(2):
          if (max1==arr1[i]):
             new_array1[index1]=i+1
             if (new_array1[index1]==1):
                  cnt1=cnt1+1
             else:
                  l1.append(j)
```

```
                      actual_label1.append(1)
                      predicted_class_label1.append(i+1)
           index1=index1+1
        #print("Accuracy1=",cnt1)
        lek1.append(cnt1)
        for j in glob.glob(path4):
          read_image=cv2.imread(j,0)
          image_resize=cv2.resize(read_image,(32,32))
          norm=np.divide(image_resize,val)
          x7=np.reshape(norm,(1024,1))
          z7=x7-Mu2
          project_array6=np.array([])
          for k in range(N):
              projection6=np.dot((v[:,k]).T,z7)
              project_array6=np.append(project_array6,projection6)
          arr2=np.zeros(2)
          arr2[0]=gauss(project_array6,mean1,s4)
          arr2[1]=gauss(project_array6,mean2,s4)
          max2=np.max(arr2)
          for i in range(2):
                 if (max2==arr2[i]):
                     new_array2[index2]=i+1
                     if (new_array2[index2]==2):
                          cnt2=cnt2+1
                     else:
                          l2.append(j)
                          actual_label2.append(2)
                          predicted_class_label2.append(i+1)
          index2=index2+1
        #print("Accuracy2=",cnt2)
        lek2.append(cnt2)


    print("Accuracy of Class 1 : ",lek1)
    print("Accuracy of Class 2 : ",lek2)
```

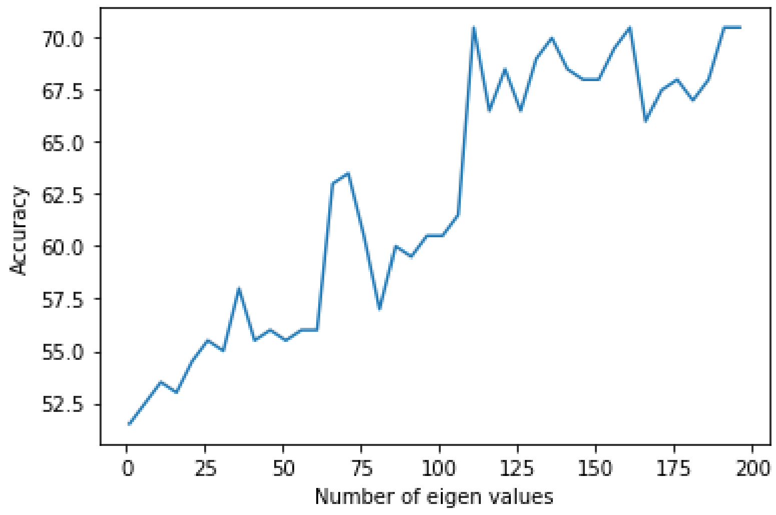
```
Accuracy of Class 1 :  [53, 54, 55, 54, 55, 57, 57, 58, 57, 57, 56, 59, 58, 65, 63, 6
Accuracy of Class 2 :  [50, 51, 52, 52, 54, 54, 53, 58, 54, 55, 55, 53, 54, 61, 64, 6
```

```
avac1=[]
for i in range(len(lek1)):
  p=lek1[i]+lek2[i]
  p=p/2
  avac1.append(p)
print("Accuracy of Class 1 : ",lek1)
print("Accuracy of Class 2 : ",lek2)
print("Average Accuracy : ",avac1)
```

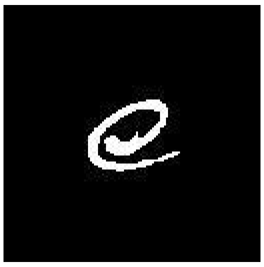
```
Accuracy of Class 1 :  [53, 54, 55, 54, 55, 57, 57, 58, 57, 57, 56, 59, 58, 65, 63, 6
Accuracy of Class 2 :  [50, 51, 52, 52, 54, 54, 53, 58, 54, 55, 55, 53, 54, 61, 64, 6
Average Accuracy :  [51.5, 52.5, 53.5, 53.0, 54.5, 55.5, 55.0, 58.0, 55.5, 56.0, 55.5
```

```
import matplotlib.pyplot as plt

x=np.arange(1,200,5)
plt.figure(1)
plt.xlabel("Number of eigen values")
plt.ylabel("Accuracy")
plt.plot(x,avac1)
```

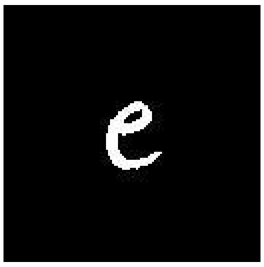
```
[<matplotlib.lines.Line2D at 0x7f5885abd590>]
```



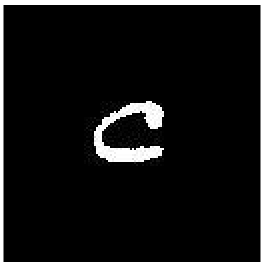
```
from google.colab.patches import cv2_imshow

newimage1=cv2.imread(l1[0],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage1)
print("Actual Label:",actual_label1[0])
print("Predicted Label:",predicted_class_label1[0])

newimage2=cv2.imread(l1[1],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage2)
print("Actual Label:",actual_label1[1])
print("Predicted Label:",predicted_class_label1[1])

newimage3=cv2.imread(l2[0],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage3)
print("Actual Label:",actual_label2[0])
print("Predicted Label:",predicted_class_label2[0])

newimage4=cv2.imread(l2[1],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage4)
print("Actual Label:",actual_label2[1])
print("Predicted Label:",predicted_class_label2[1])
```



```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 2
Predicted Label: 1
```

```
lek1=[]
lek2=[]
val=255.0
for N in range(1,200,5):
    I=np.identity(N)
    project_array2=np.array([])
    project_array1=np.array([])
    for j in glob.glob(path1):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x3=np.reshape(norm,(1024,1))
        z3=x3-Mu1
        for i in range(N):
            projection1=np.dot((v[:,i]).T,z3)
            project_array1=np.append(project_array1,projection1)
    project_array1=np.reshape(project_array1,(200,N))
    project_array1=project_array1.T
    Mu3=np.sum(project_array1,axis=1)
    mean1=np.reshape(Mu3,(N,1))
    for j in glob.glob(path2):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,xi_max)
        x4=np.reshape(norm,(1024,1))
        z4=x4-Mu2
        for i in range(N):
            projection2=np.dot((v[:,i]).T,z4)
            project_array2=np.append(project_array2,projection2)
```

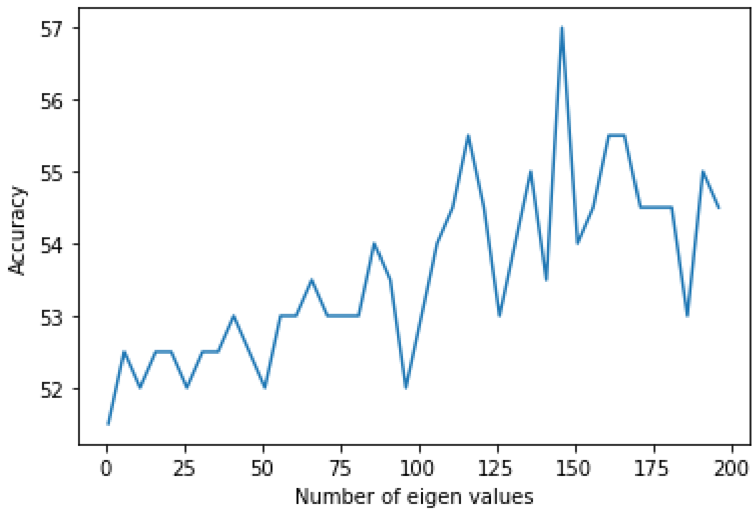
```
project_array2=np.reshape(project_array2,(200,N))
project_array2=project_array2.T
Mu4=np.sum(project_array2,axis=1)
mean2=np.reshape(Mu4,(N,1))
Mu_pooled=(Mu3+Mu4)/400
Mu_pooled=np.array(Mu_pooled)
#s3=np.zeros((N,N))
s3=0
for i in range(200):
    #s=project_array1[:,i]-Mu_pooled
    #s3=s3+np.dot(s,s.T)
    s3=np.var(s)
for i in range(200):
   s=project_array2[:,i]-Mu_pooled
   #s3=s3+np.dot(s,s.T)
   s3=np.var(s)
s3=s3/400
s3=s3+lamda*I
s5=s3*I
path3="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/1/*.jpg"
path4="/content/gdrive/MyDrive/Machine Learning/LAB 3/TestCharacters/TestCharacters/2/*.jpg"

index1=0
index2=0

cnt1=0
cnt2=0

l1=[]
l2=[]

actual_label1=[]
actual_label2=[]

predicted_class_label1=[]
predicted_class_label2=[]

new_array1=np.zeros(100)
new_array2=np.zeros(100)

for j in glob.glob(path3):
    read_image=cv2.imread(j,0)
    image_resize=cv2.resize(read_image,(32,32))
    norm=np.divide(image_resize,xi_max)
    x4=np.reshape(norm,(1024,1))
    z5=x4-Mu1
    project_array3=np.array([])
    for i in range(N):
        projection3=np.dot((v[:,i]).T,z5)
        project_array3=np.append(project_array3,projection3)
    arr1=[0,0]
    arr1[0]=gauss(project_array3,mean1,s5)
    arr1[1]=gauss(project_array3,mean2,s5)
    max1=max(arr1)
    for i in range(2):
          if (max1==arr1[i]):
             new_array1[index1]=i+1
             if (new_array1[index1]==1):
                    cnt1=cnt1+1
                else:
                    l1.append(j)
                    actual_label1.append(1)
                    predicted_class_label1.append(i+1)
          index1=index1+1
      #print("Accuracy1=",cnt1)
      lek1.append(cnt1)
      for j in glob.glob(path4):
        read_image=cv2.imread(j,0)
        image_resize=cv2.resize(read_image,(32,32))
        norm=np.divide(image_resize,val)
        x7=np.reshape(norm,(1024,1))
        z7=x7-Mu2
        project_array6=np.array([])
        for k in range(N):
            projection6=np.dot((v[:,k]).T,z7)
            project_array6=np.append(project_array6,projection6)
        arr2=np.zeros(2)
        arr2[0]=gauss(project_array6,mean1,s5)
        arr2[1]=gauss(project_array6,mean2,s5)
        max2=np.max(arr2)
        for i in range(2):
              if (max2==arr2[i]):
                  new_array2[index2]=i+1
                  if (new_array2[index2]==2):
                      cnt2=cnt2+1
                  else:
                      l2.append(j)
                      actual_label2.append(2)
                      predicted_class_label2.append(i+1)
        index2=index2+1
      #print("Accuracy2=",cnt2)
      lek2.append(cnt2)


  print("Accuracy of Class 1 : ",lek1)
  print("Accuracy of Class 2 : ",lek2)
```

```
Accuracy of Class 1 :  [53, 54, 54, 54, 54, 53, 54, 55, 54, 54, 53, 55, 55, 56, 54, 5
Accuracy of Class 2 :  [50, 51, 50, 51, 51, 51, 51, 50, 52, 51, 51, 51, 51, 51, 52, 5
```

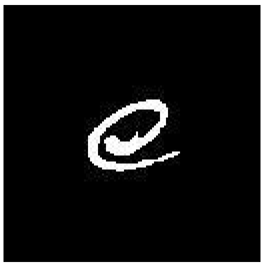
```
avac2=[]
for i in range(len(lek1)):
  p=lek1[i]+lek2[i]
  p=p/2
  avac2.append(p)
print(avac2)
```

```
[51.5, 52.5, 52.0, 52.5, 52.5, 52.0, 52.5, 52.5, 53.0, 52.5, 52.0, 53.0, 53.0, 53.5,
import matplotlib.pyplot as plt

x=np.arange(1,200,5)
plt.figure(1)
plt.xlabel("Number of eigen values")
plt.ylabel("Accuracy")
plt.plot(x,avac2)
```

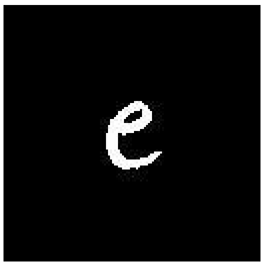
```
[<matplotlib.lines.Line2D at 0x7f5885a00f90>]
```



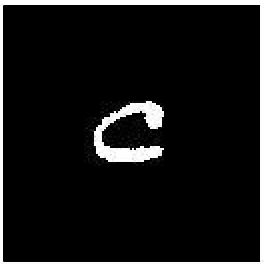
```
from google.colab.patches import cv2_imshow

newimage1=cv2.imread(l1[0],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage1)
print("Actual Label:",actual_label1[0])
print("Predicted Label:",predicted_class_label1[0])

newimage2=cv2.imread(l1[1],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage2)
print("Actual Label:",actual_label1[1])
print("Predicted Label:",predicted_class_label1[1])

newimage3=cv2.imread(l2[0],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage3)
print("Actual Label:",actual_label2[0])
print("Predicted Label:",predicted_class_label2[0])

newimage4=cv2.imread(l2[1],cv2.IMREAD_UNCHANGED)
cv2_imshow(newimage4)
print("Actual Label:",actual_label2[1])
print("Predicted Label:",predicted_class_label2[1])
```



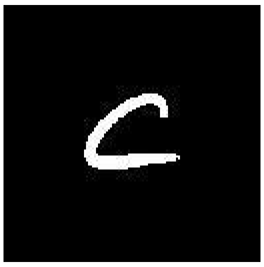
```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 1
Predicted Label: 2
```



```
Actual Label: 2
Predicted Label: 1
```



```
Actual Label: 2
Predicted Label: 1
```

✓ 0s completed at 9:32 PM