Pattern Classification
Assignment 1

Bayes Implementation of Image Recognition

Ritu Ann Roy George - B170106EC

NA Aakaash Kumaran - B171000EC

Priyamvada Yashaswi - B171037EC

Minnu Saji - B170124EC

Vinay Kumar Patnaik - B170396EC

In [1]:
```
import pandas as pd
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
import matplotlib.image as mpimg
import glob
%matplotlib inline
```

Extract and Vectorise Images from Sign MNIST Database

In [2]:
```
tr=pd.read_csv(r'D:\AnacondaProjects\signdb\sign_mnist_train.csv')
tr= tr.sort_values(by=['label'])
img_tr = tr.drop(['label'], axis = 1)
l_tr=tr['label']

te=pd.read_csv(r'D:\AnacondaProjects\signdb\sign_mnist_test.csv')
te= te.sort_values(by=['label'])
img_te = te.drop(['label'], axis = 1)
l_te=te['label']

l_test = np.array(l_te)
img_test = np.array(img_te)
l_test = l_test[0:10]
```

```
img_test = img_test[0:10]
l_train = np.array(l_tr)
img_train = np.array(img_tr)

nc=15 #no.of classes

def divclas(l,img,num):
    val=l[0]
    train=[]
    class_tr_l=[]
    class_tr_img=[]
    class_tr_l.append(l[0])
    for i in range(len(l)):
        if val==l[i]:
            train.append(img[i])
        else:
            train=np.array(train)
            class_tr_img.append(train[0:num])
            val=l[i]
            train=[]
            train.append(img[i])
            class_tr_l.append(l[i])
    train=np.array(train)
    class_tr_img.append(train[0:num])
    class_tr_l=np.array(class_tr_l)
    class_tr_img=np.array(class_tr_img)
    return class_tr_l[0:nc],class_tr_img[0:nc]

class_tr_l,img_train_class=divclas(l_train,img_train,10)
class_te_l,img_test_class=divclas(l_train,img_train,1)
class_te_l,img_test_class = shuffle(class_te_l,img_test_class, random_state=0)
```

PCA and Class Projection Computation (EigenVectors and EigenValues)

In [3]:

```
dif = []
for clas in img_train_class:
    for i in range(len(clas)):
        for j in range(i,len(clas)):
            if i != j:
                dif.append(clas[i] - clas[j])
dif=np.array(dif)
mean_dif=0
for i in range(len(dif)):
    mean_dif += dif[i] / len(dif)
norm_dif = dif - mean_dif
```

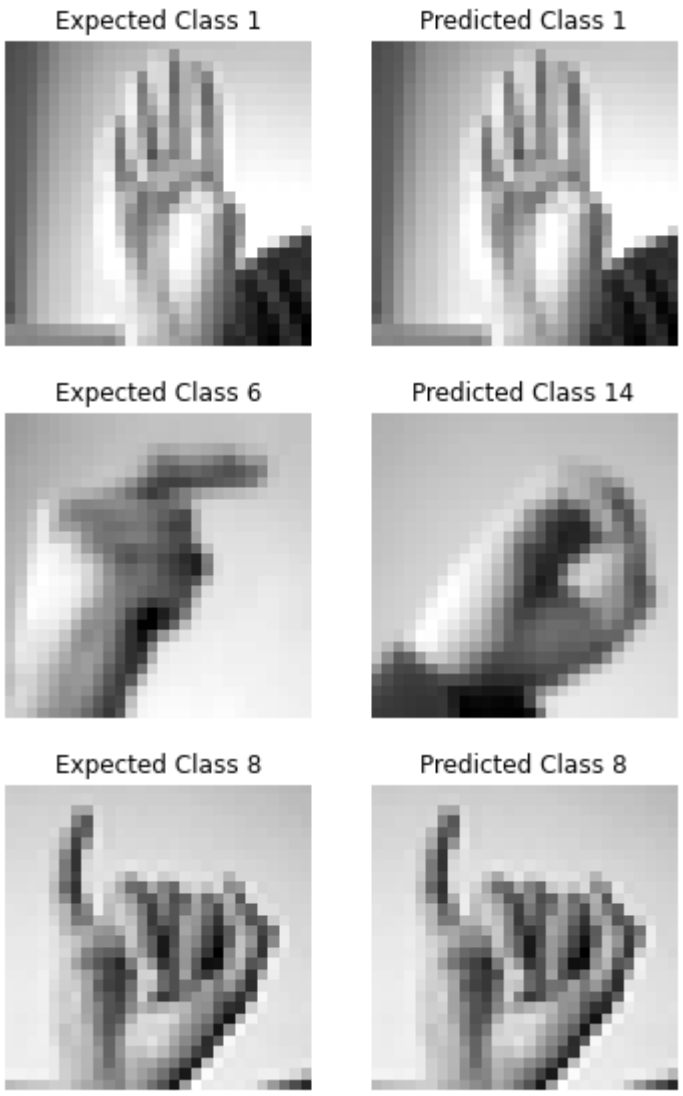
```
C = np.cov(dif)
[evalu , evect] = np.linalg.eigh(C)
evect = np.dot(norm_dif.T, evect )

for i in range (evect.shape[1]):
    evect[:,i] = evect[:,i]/ np.linalg.norm( evect[:,i])

indices = np.argsort (- evalu)
evalu  = evalu[indices]
evect = evect[:, indices]
```

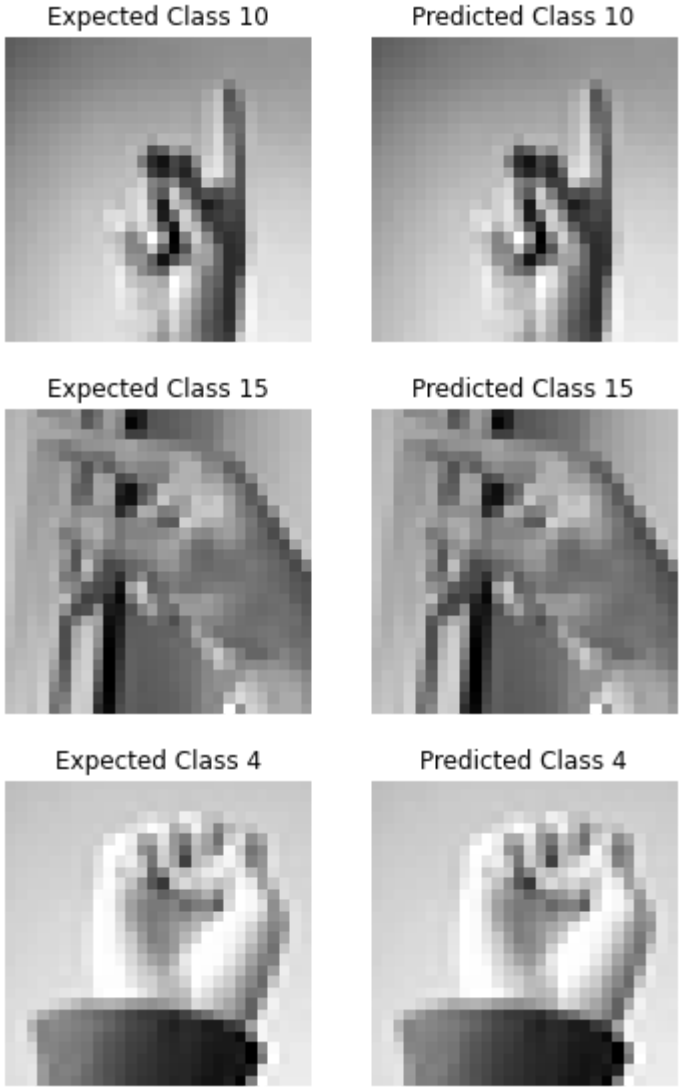
Prediction of Class

In [4]:
```
N = 20

def coef(img):
    return (np.power(evalu[:N],-1/2) * (evect.T[:N] * img).T).T

p_set = []
for clas in img_train_class:
    p = 0
    for i in range(len(clas)):
        p += coef(clas[i]) / len(clas)
    p_set.append(p)
p_set = np.array(p_set)

def predict_class(img):
    allps = []
    for i in range(p_set.shape[0]):
        pr = np.exp(- 0.5 * np.linalg.norm(coef(img)- p_set[i]))
        allps.append(pr)
    pred = np.argmax(allps)
    return pred
```

In [5]:
```
acc=0
for k in range(img_test_class.shape[0]):
    plt.subplot(1,2,1)
    plt.imshow(np.reshape(img_test_class[k],(28,28)),cmap='gray')
    plt.title('Expected Class {}'.format(class_te_l[k]))
    plt.axis('off')
    plt.subplot(1,2,2)
    c=predict_class(img_test_class[k])
    plt.imshow(np.reshape(img_train_class[c][0],(28,28)),cmap='gray')
    plt.title('Predicted Class {}'.format(class_tr_l[c]))
    if(class_te_l[k]==class_tr_l[c]):
```

```
        acc+=1
    plt.axis('off')
    plt.show()
```




Expected Class 10
Predicted Class 10
Expected Class 15
Predicted Class 15
Expected Class 4
Predicted Class 4

Expected Class 2

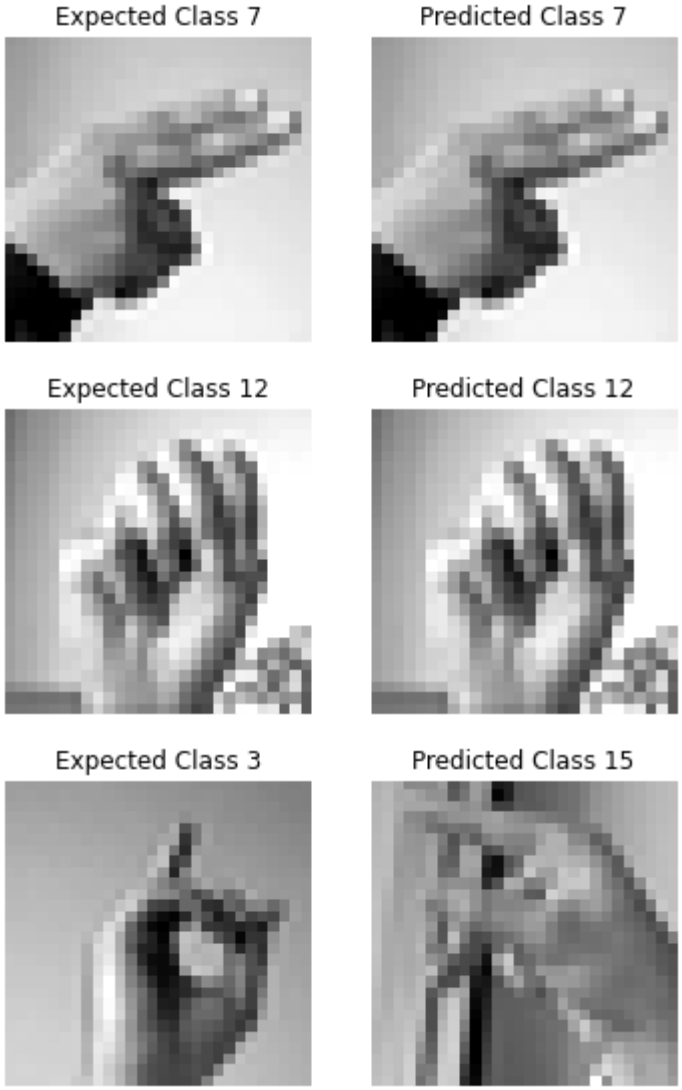
Predicted Class 2

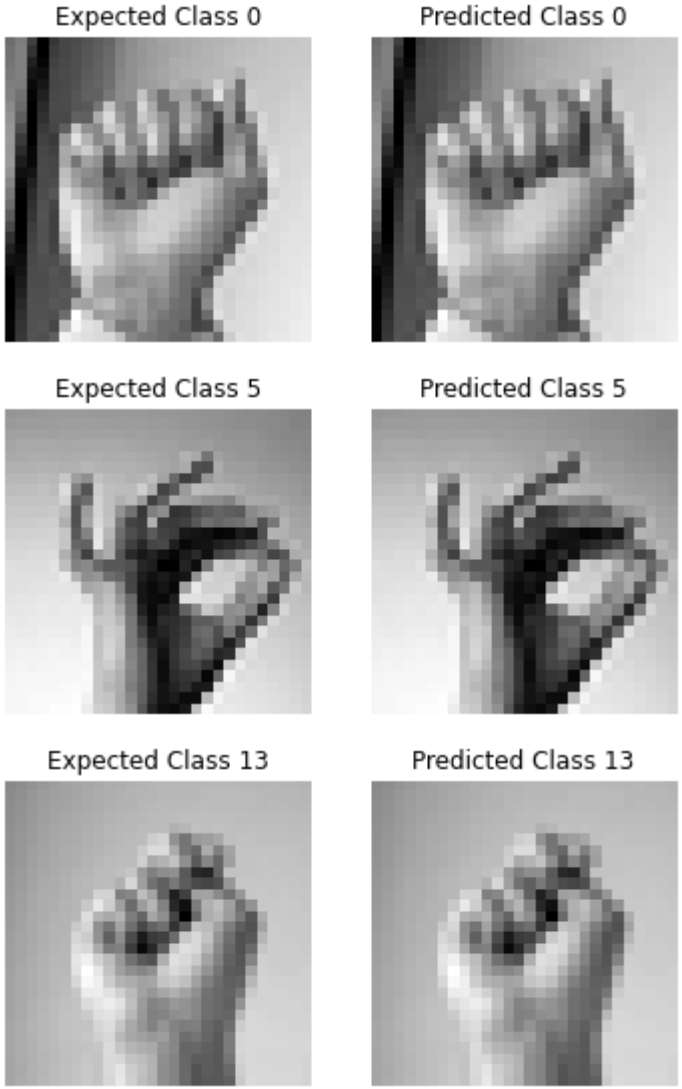
Expected Class 14

Predicted Class 14

Expected Class 11

Predicted Class 11


Expected Class 7
Predicted Class 7
Expected Class 12
Predicted Class 12
Expected Class 3
Predicted Class 15



In [6]:
```
print(100*acc/nc)
```

```
86.66666666666667
```