SML Assignment 3

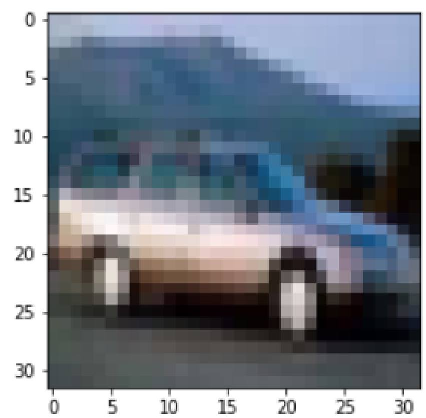
Name - Utkarsh Dubey
Roll no - 2019213

Question 1 -

Visualizing as images -

```
plt.show()
```

```
Printing images of  automobile
(3072,)
```




```
(3072,)
```

5 Images of each class are printed.

Applied LDA on the dataset -

```
]: #LDA
clf = LinearDiscriminantAnalysis()
clf.fit(xTrain,yTrain)
print("Accuracy on testing data - ",clf.score(xTest,yTest))
```

```
Accuracy on testing data -  0.3713
```

And got an accuracy of 37%

Accuracy of each class -

```
Accuracy for class airplane =  0.463
Accuracy for class automobile =  0.415
Accuracy for class bird =  0.255
Accuracy for class cat =  0.245
Accuracy for class deer =  0.271
Accuracy for class dog =  0.329
Accuracy for class frog =  0.413
Accuracy for class horse =  0.404
Accuracy for class ship =  0.494
Accuracy for class truck =  0.424
```

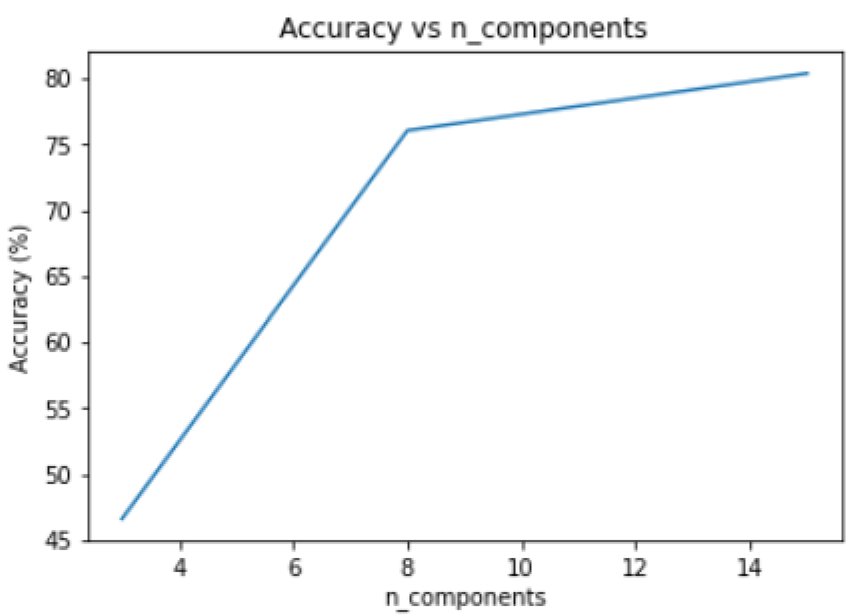
Question 2 -

Applied PCA with given n_components -

```
pca = PCA(n_components = n)
```

here n = [3,8,15]

Accuracy vs n_components graph -



As we can see n_components = 15 is performing the best, as if we reduce the number of n_components, we reduce the dimension, hence losing important data, which would have been useful for classifying into cases.

Question 3 -

Implemented the FDA class -

```
#class for FDA
class FDA:
    W = None
    eigenValues = None
    Sw, Sb = None, None

    def __init__(self):
        W = None
        eigenValues = None
        Sw, Sb = None, None
```

```
def fit(self, x, y):
    Sw,Sb = self.getScatters(x, y)
    self.Sw = Sw
    self.Sb = Sb
    Sw_inv_Sb = np.matmul(np.linalg.inv(Sw), Sb)
    eigValues, eigVectors = np.linalg.eigh(Sw_inv_Sb)
    idx = eigValues.argsort()[::-1]
    eigVectors = eigVectors[:,idx]
    eigValues = eigValues[idx]
    rank = np.linalg.matrix_rank(Sb)
    eigVectors = eigVectors[:,:rank]
    eigValues  = eigValues[:rank]
    self.eigenValues = eigValues
    self.W = eigVectors
```

```python
def getScatters(self, x, y):
    Si = []
    uniqueY = np.unique(y)
    for label in uniqueY:
        indexes = np.where(y == label)[0]
        selectedX = x[indexes]
        selectedXTrans = selectedX.T
        scatterMatrix = np.cov(selectedXTrans, ddof=0) * selectedXTrans.shape[1]
        Si.append(scatterMatrix)
    Sw = np.zeros(Si[0].shape)
    for scatterMatrix in Si:
        Sw += scatterMatrix

    xTrans = x.T
    St = np.cov(xTrans, ddof=0) * xTrans.shape[1]
    Sb = St - Sw

    return Sw,Sb
```

```python
xTrainProjected = np.matmul(W.T,xTrain.T).T
xTestProjected = np.matmul(W.T,xTest.T).T

clf = LinearDiscriminantAnalysis()
clf.fit(xTrainProjected,yTrain)

print("Accuracy by own fda = ",clf.score(xTestProjected,yTest)*100,"%")
```

```
Accuracy by own fda =  72.25 %
```

Got an accuracy of 72.25%

Class wise accuracy as

```
Accuracy for class 0 =  74.0 %
Accuracy for class 1 =  87.0 %
Accuracy for class 2 =  53.900000000000006 %
Accuracy for class 3 =  76.3 %
Accuracy for class 4 =  61.6 %
Accuracy for class 5 =  78.9 %
Accuracy for class 6 =  36.1 %
Accuracy for class 7 =  78.2 %
Accuracy for class 8 =  85.5 %
Accuracy for class 9 =  91.0 %
```

Question 4 -

Applying PCA

```
pca = PCA(n_components=15)
```

```
pca.fit(xTrain)
xTrain = pca.transform(xTrain)
xTest = pca.transform(xTest)
```

Applying FDA

```
fda = FDA()
fda.fit(xTrain,yTrain)

W = fda.W
xTrainProjected = np.matmul(W.T,xTrain.T).T
xTestProjected = np.matmul(W.T,xTest.T).T
```

```
clf = LinearDiscriminantAnalysis()
clf.fit(xTrainProjected,yTrain)

print("Accuracy we get is - ", clf.score(xTestProjected, yTest)*100,"%")
```

```
Accuracy we get is -  79.36999999999999 %
```

Got an accuracy of 79.3%

Class-wise accuracy as -

```
Accuracy for class 0 =  86.3265306122449 %
Accuracy for class 1 =  95.94713656387665 %
Accuracy for class 2 =  74.70930232558139 %
Accuracy for class 3 =  79.50495049504951 %
Accuracy for class 4 =  76.17107942973523 %
Accuracy for class 5 =  66.59192825112108 %
Accuracy for class 6 =  83.08977035490605 %
Accuracy for class 7 =  79.47470817120622 %
Accuracy for class 8 =  74.435318275154 %
Accuracy for class 9 =  74.13280475718534 %
```

Question 5 -

SML Assignment-3

2019213 Utkarsh Dubey

Q5

Ans a) we have

$$d_i = y_i (\beta^T x_i + \beta_0)$$

Update Rule parameter will not change by changing activation function

As we have

$$\beta_{new} = \beta - n \frac{d}{d\beta}(d_i)$$

$$\beta_{0new} = \beta_0 - n \frac{d}{d\beta_0}(d_i)$$

Here update rule is not changing as the distance is independent of of the activation function.

b) $$\phi(\beta, \beta_0) = - \sum_{i=1}^{N} y_i (\beta^T x_i + \beta_0)$$

$$\Rightarrow \beta^T \beta = 1$$

$$g(\beta) = \beta^T \beta - 1$$

$$L(\beta, \beta_0) = \phi(\beta, \beta_0) + \lambda g(\beta)$$
$$= - \sum_{i=1}^{N} y_i (\beta^T x_i + \beta_0) + \lambda(\beta^T \beta + 1)$$

$$\frac{dL}{d\beta} = - \sum_{i=1}^{N} y_i x_i + \lambda(\beta, \beta) = 0$$

$$\frac{dL}{d\beta_0} = - \sum_{i=1}^{N} y_i = 0$$

Hence $$\boxed{\beta_{new} = \beta - n \frac{dL}{d\beta}}$$

$$\beta_{0new} = \beta_0 - n \frac{dL}{d\beta_0}$$

λ is hyperparameter.

Question 6

Q6

A

$x \rightarrow y_i \rightarrow$ Perceptron $\rightarrow y_2$

$y_1 = \sigma(B_{11} x + B_{01})$

$y_2 = \cancel{\sigma}$ Sym $(B_{12} y_i + B_{02})$

$\rightarrow d_i = -y_i (B_{12} y_{i1} + B_{02})$

$\frac{\partial d_i}{\partial B_{11}} = -y_i \left(B_{12} \frac{\partial y_{i1}}{\partial B_{11}} + 0\right)$

$\sigma'(x) = \sigma(x)[1 - \sigma(x)]$

$= -y_i (B_{12} \sigma'(B_{11} x_i + B_{01} x_i)$

$= -y_i x_i B_{12} e^{-v_i} (1 + e^{-v_i})^2$

$\frac{\partial d_i}{\partial B_{01}} = -y_i \left(B_{12} \frac{\partial y_{i1}}{\partial B_{01}}\right)$

$\frac{\partial y_{i1}}{\partial B_{01}} = \sigma'(B_1 x_i + B_{01}) = \left(\frac{e^{-v_i}}{1 + e^{-v_1}}\right)^2$

$$\frac{\partial d_i}{\partial B_{01}} = \frac{-y_i \, B_{12} \, e^{-v_i}}{(1 + e^{-v_i})^2}$$

$$\cancel{\frac{\partial d_i}{\partial B_{12}}} = \cancel{-y_i \, (\sigma \, (B_{11})}$$

$$\frac{\partial d_i}{\partial B_{12}} = -y_i \left[\sigma \, [B_{11} x + B_{01}] \right]$$

$$\frac{\partial d_i}{B_{02}} = -y_i$$

$$\boxed{\begin{aligned} B_{01} &= B_{01} - n \frac{\partial d_i}{\partial B_{01}} \\ B_{12} &= B_{12} - n \frac{\partial d_i}{\partial B_{12}} \\ B_{02} &= B_{02} - n \frac{\partial d_i}{\partial B_{02}} \\ B_{11} &= B_{11} - n \frac{\partial d_i}{\partial B_{11}} \end{aligned}}$$