Assignment Project Report

Linear Discriminant Analysis

Name: Arun Govind
Course: AI and ML
(Batch 4)

- **Problem Statement**

  Perform LDA on Iris Dataset and implement a classifier to test the performance of LDA

- **Prerequisites**

  1. Software:
     - Python 3 (Use anaconda as your python distributor as well)

  2. Tools:
     - Numpy
     - Sklearn
     - KNN
     - Matplot

  3. Dataset used: Iris dataset provided in the Sklearn library.

- **Method Used**

  LDA, Linear Discriminant Analysis is a method used in statistics and other fields, to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

  Using LDA based classification, we can find discriminative features for a given audio segment to achieve the task of Automatic Speech Classification such that speech belonging to the same class are close together, but samples from different classes are far apart from each other.

- **Implementation:**

1. Code for Importing all Libraries to fetch and arrange data:

In [47]:
```
# Importing Datasets From Sklearn

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

In [48]:
```
# Loading IRIS Dataset

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names
```

2. Code fitting data into the LDA model:

In [49]:
```
# fitting the LDA model
lda = LDA(n_components=2)
lda_X = lda.fit(X,y).transform(X)
```

3. Plotting a graph for the LDA model:

In [50]:
```
'''Assigning colors for graph'''

plt.figure()
colors = ['blue', 'red', 'green']
lw = 2
```

```
<Figure size 432x288 with 0 Axes>
```

In [51]:
```
# Plotting the graph for LDA (IRIS dataset)
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(lda_X[y == i, 0], lda_X[y == i, 1], alpha=.8, color=color,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA of IRIS dataset')

plt.show()
```

4. Training the LDA model:

Once dataset is loaded into a pandas data frame object, the first step is to divide dataset into features and corresponding labels and then divide the resultant dataset into training and test sets. The following code divides data into test and train:

```
In [52]: from sklearn.model_selection import train_test_split
         from sklearn import neighbors, datasets, preprocessing
```

```
In [53]: Xtrain, Xtest, y_train, y_test = train_test_split(X, y)
         scaler = preprocessing.StandardScaler().fit(Xtrain)
         Xtrain = scaler.transform(Xtrain)
         Xtest = scaler.transform(Xtest)

         clf = LDA()
         clf.fit(Xtrain,y_train)
         y_pred=clf.predict(Xtest)

         y_pred
```

In the script above the LinearDiscriminantAnalysis class is imported as LDA. Like PCA, we have to pass the value for the n_components parameter of the LDA, which refers to the number of linear discriminates that we want to retrieve. In this case we set the n_components to 1, since we first want to check the performance of our classifier with a single linear discriminant. Finally, we execute the fit and transform methods to actually retrieve the linear discriminants.

5. Applying KNN Classifier:

Since we want to compare the performance of LDA with one linear discriminant to the performance of PCA with one principal component, we will use the K-Nearest Neighbor classifier method.

```
In [55]: #Fitting above to KNN to determine accuracy
         knn = neighbors.KNeighborsClassifier(n_neighbors=5)
         knn.fit(Xtrain, y_train)
         y_pred = knn.predict(Xtest)
         y_pred
```

```
Out[55]: array([2, 2, 2, 0, 1, 1, 2, 0, 2, 2, 2, 2, 1, 0, 1, 2, 1, 1, 1, 2, 2, 2,
                2, 0, 0, 2, 0, 1, 1, 2, 2, 1, 2, 1, 2, 2, 0, 1])
```

```
In [56]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report
         from sklearn.metrics import confusion_matrix

         print('Accuracy Score:', accuracy_score(y_test, y_pred))
         print('Confusion matrix \n',  confusion_matrix(y_test, y_pred))
         print('Classification \n', classification_report(y_test, y_pred))
```

- **Results:**

  Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique which is commonly used for the supervised classification problems.

  1. Classification report:

Classification

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	0.92	1.00	0.96	11
2	1.00	0.95	0.97	20
accuracy			0.97	38
macro avg	0.97	0.98	0.98	38
weighted avg	0.98	0.97	0.97	38

  2. Confusion matrix:

```
Confusion matrix
 [[ 7  0  0]
 [ 0 11  0]
 [ 0  1 19]]
```

  3. Accuracy score:

```
Accuracy Score: 0.9736842105263158
```

Our model achieves an overall accuracy of 97.3%