Havan Patel
03/27/2022
DATA 55100
Spring 2022
Unsupervised Machine Learning
Assignment 2

PCA vs LDA

1. The Technical Description of All Techniques Utilized

In this project we are going to implement and analyze two of the most used techniques in reducing large n * m dimensions in machine learning. One of the techniques is called Principal Component Analysis known as PCA and another on is called Linear Discriminant Analysis known as the LDA. In this assignment I have used python to implement both PCA and LDA dimension reduction techniques. I have also utilized libraries such as pandas, NumPy, pyplot, and sklearn to help me implement and do calculations for PCA and LDA. I also translate the result into a 1d, 2d and 3d plot so we can compare the outputs and analyze it. We will see that in the later section below. First let's analyze the theory and technique in a concise manner as we have a lot to cover.

First let's take look at PCA, it is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It basically rotates the coordinate axes, in a such way that axis captures almost all the information content or the variance. Now let's look at the math theory for PCA in simple step process. First is, to find the mean of every dimension of your dataset which helps centering our data. It is done by subtraction of the variable average from the data and the vector of averages that corresponds to a point in the K-space. Second Step is to find covariance matrix. In mathematics, we find the covariance between X1 and X2 and represent it in a form of a matrix. The formula for finding covariance matrix is σjk=1/(n−1) $\sum_i^1 n$ (xij−xj) (xik−xk), where xj is mean of that column 'j' and xk is mean of column 'k'. That matrix shows us information of how much details is contained between two-dimensional space x1 and x2 in a numerical sense. The diagonal elements in the matrix are variance spread of x1 with itself and x2 with itself telling us how much details are in that variable itself. Therefore, the diagonal element will be close to one if the data is normalized. The next and final step is doing Eigen decomposition, which is a process of transforming the original covariance matrix between x1 and x2 into another matrix. This newly formed matrix will have the diagonal elements as one and elements that are not diagonal will have values that are close to zero. Thus, we get two new outputs, eigenvector and eigenvalue. Where eigenvectors represent the new dimension of the new mathematical space and eigen value is information about content of each of the eigenvectors and spread of the data on eigenvectors.

Second approach is LDA, mostly used for feature extraction in pattern classification problems. The difference from PCA is that LDA tries to maintain as much of the class discriminatory information as possible. LDA creates a linear combination of these features which yields the largest mean differences between the desired classes. The approach is very similar to PCA but here we are introducing new concepts of scatter matrix and there are two types of them. Scatter matrix is nothing but to approximate the covariance matrix. First one is called between class scatter (SB), and another is within class scatter (SW). Formula for SB is $S_b = (\mu 1 - \mu 2)(\mu 1 - \mu 2)^T$ and SW is $S_w = S_1 + S_2 + \ldots + S_b$. SB measures the separation between the classes and SW measures the spread around the means of each class. So, the goal is for the between class measures to be maximized and minimizing the within class measures. 'One way to do this is to maximize the ratio $\frac{det|S_b|}{det|S_w|}$." After computing the scatter matrix, we can compute the eigenvectors and corresponding eigenvalues. To do that we can obtain the linear discriminant by computing $S_w^{-1}S_B$. By computing this, we will finally receive dimensions that is reduce and we can use those matrices to plot it and visualize to understand the correlation and separation of classes.

2. The Design of The Algorithms

2.1 PCA

1. Getting the mean of each variable
   - Deduct the mean of every factor from the dataset so that the dataset should be focused on the origin. I also get the dimension of data X into variable N, M
   - This step is required as it will helps us calculating the covariance of the matrix
   - I utilized numpy to help calculating the mean of the dataset X

```
def PCA(X , num_components):
      N, M = X.shape
      mean = X - np.mean(X , axis = 0)
```

2. Calculate Covariance of matrix
   - Calculate the covariance matrix from the mean matrix
   - Covariance matrix is a square matrix which is a measure of the spread of set of points around their center of mean.
   - To find covariance of the matrix I used two for loops to calculate the mean of column I and column J to find the number of Covariance between column "i" and column "j".

```
# calculate covariance matrix
    cov = np.zeros((M, M))
    for i in range(M):
        mean_col_i = np.sum(X[:, i]) / N
        for j in range(M):
            mean_col_j = np.sum(X[:, j]) / N
            cov[i, j] = np.sum((X[:, i] - mean_col_i) * (X[:, j] -mean_col_j)) / (N - 1)
```

3. Calculate the eigenvectors and eigenvalues of the covariance matrix
   - compute the Eigenvalues and Eigenvectors for the covariance matrix from above step 2.

```
eigen_values , eigen_vectors = np.linalg.eigh(co_var)
```

4. Sorting the eigenvalues in decreasing order
   - For us to get the 1st principal component with the highest value, 2nd principal component with the second highest value, and so one we need to sort the matrix.

```
index = np.argsort(eigen_values)[::-1]
s_eigenvectors = eigen_vectors[:,index]
```

5. Select the eigenvector subset from the reordered eigenvalue
   - Select the number of n eigenvector for our desired dimension.

```
eigenvector_dimension = s_eigenvectors[:,0:num_components]
```

6. Transforming the data
   - By transforming the data with dot product of transpose eigenvector and transpose of mean, we will get the reduced dimension from high dimensions, and we return reduced_dim.

```
reduced_dim = np.dot(eigenvector_subset.transpose(),_mean.transpose()).transpose()
```

2.2 LDA

1. Initialize variables like number of features and classes from dataset. Then, we need to first calculate the mean of the entire dataset. And initialize empty matrix to store the values.

```
def LDA(X, y, num_classes):
        dimensions = X.shape[1]
        classes = np.unique(y)
        mu = np.mean(X, axis=0)
        SW = np.zeros((dimensions, dimensions))
        SB = np.zeros((dimensions, dimensions))
```

2. We then calculate the between class and within class matrix to maximize the separation of classes. For each label in class, we need to find between and within matrix.

```
for label in classes:
        sample = X[y == label]
        sample_mean = np.mean(sample, axis=0)
        SW += (sample - sample_mean).T.dot((sample - sample_mean))

        mean_diff = (sample.shape[0] - mu).values.reshape(dimensions, 1)
        SB += number_of_label * (mean_diff).dot(mean_diff.T)
```

3. We need to calculate the inverse of within class matrix and dot product it with between class matrices. We then calculate the eigenvalue and eigenvector like PCA and then we sort it in decreasing order.

```
    eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(SW).dot(SB))
    eigenvectors = eigenvectors.T
     index = np.argsort(abs(eigenvalues))[::-1]
     eigenvalues = eigenvalues[index]
     eigenvectors = eigenvectors[index]
```

4. We then choose only the first largest n eigenvectors that will give us the reduced dimension

```
ld = eigenvectors[0 : num_components]
```

5. At last, we need to transform the original features or dimensions into the reduced dimension and project the data using the dot product

```
return np.dot(X, ld.T)
```

3 The Results/Analysis of The Algorithms

3.1 PCA

Figure 1

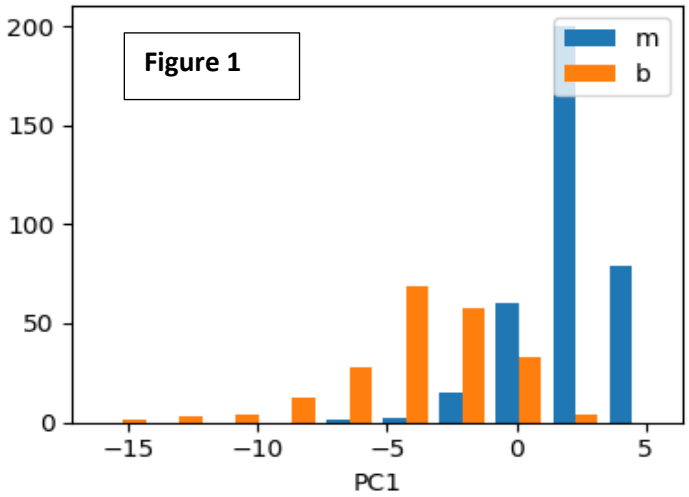


Figure 3

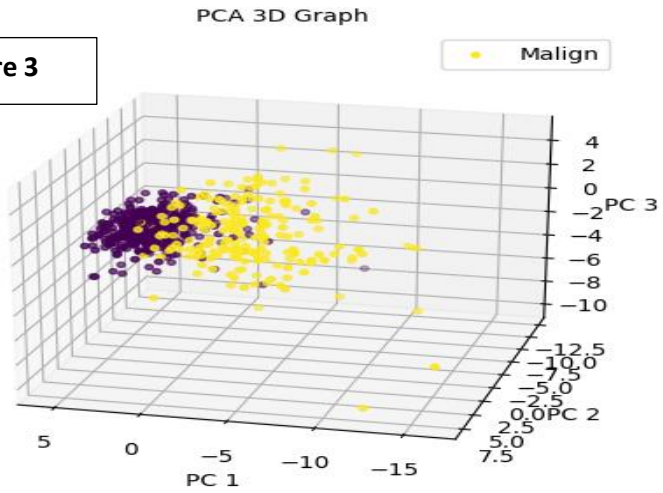
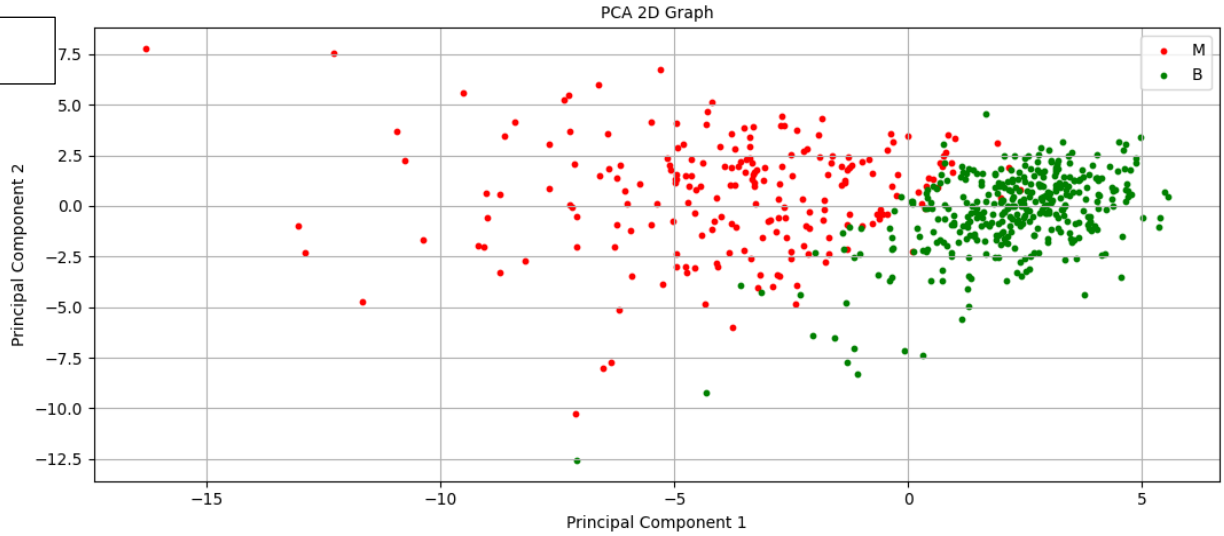


Figure 2



The three graphs that you see above were generated using the PCA algorithm explained in the previous section of PCA. In this project for PCA implementation, I have used Breast Cancer dataset from kaggle site which has more than 25 features, 212 sample of malignant class and 357 sample of benign class. This dataset is a very valued and has two main classes (malignant and benign) which determines if a patient has a breast cancer or not. Now to examine the results, let's just take look at the figure 1 and figure 2 to keep it concise. Based on the number of components you provide in the algorithm; you will get the reduced dimensions. In the 1-dimension histogram graph (Fig. 1), we can see the separation as well some overlap of the information between two classes. Likewise, the 2D scatterplot (Fig. 2) we can see the same separation of sample or two classes in two different formations with two different components. We also see the two classes Malignant representing 'M' and Benign representing 'B', when they are projected on to a two-dimensional space, they can be linearly distinguishable to some degree. We can see that the Malignant class is scattered out more than class Benign. The reason for the graph to be in

the 2-D space is because I provided the PCA algorithm with 2 components. The reason for the separation of the two classes is because there is more information contained in the 1st principal component. Not only that but we can clearly see the splitting of the data, which also indicates that there is not much loss of the data. PCA is a decent tool for investigating a dataset, however it will be unable to observe more unpretentious groupings that may produce better representations for more intricate datasets. We know that PCA is often not the best suited for more large number of datasets.

3.2 LDA

Figure 4

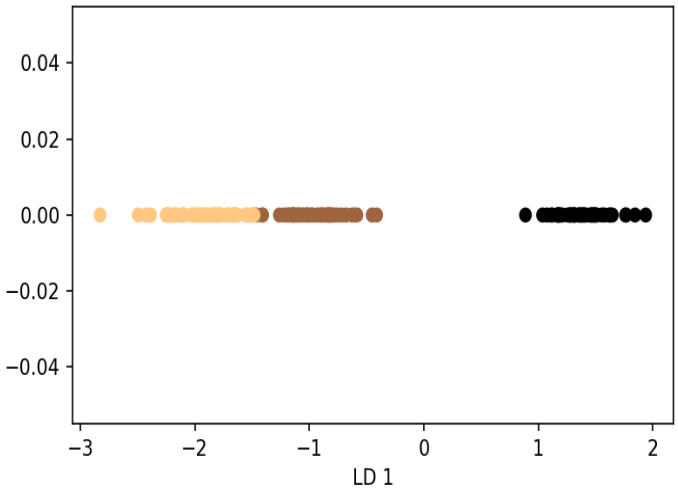
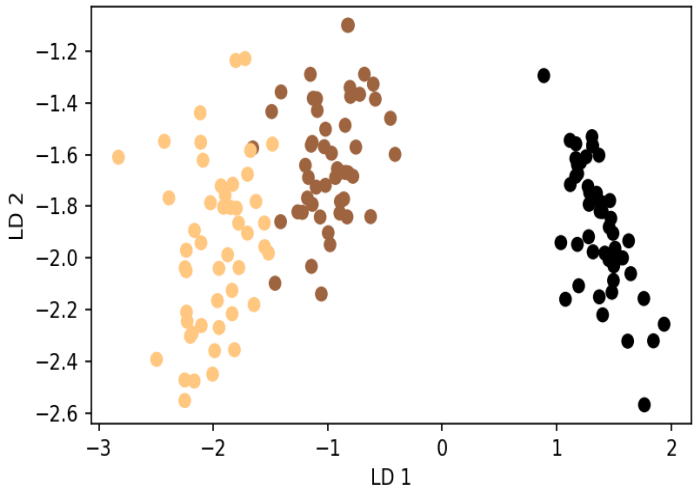


Figure 5



The above 2 graphs (Fig. 4 & 5) were generated using the LDA algorithm. In the implementation of LDA, I have used the iris dataset from Kaggle which has 3 classes, more than 4 features/dimensions and 150 sample of data. This is one of the most used datasets in Data science for many reasons. This dataset contains three classes 'Iris-setosa', 'Iris-versicolor', and 'Iris-virginica'. We will perform LDA on this class/labels and interpret the result above. Let's see Fig. 4, we see it's a 1D representation of the 3 classes or labels presented on a flat plane. We can clearly see the separation between the three classes. Black dots represent Iris-Setosa class, brown dots represent Iris Versicolor and yellow dot represent Iris-Virginica class. Of course, there will be somewhat overlapping but not as much as PCA if we compare it with Fig. 2. This is the one of the differences I see in PCA and LDA that LDA is performing the calculation to maximize the separation between multiple classes rather than finding the component that maximizes our variance of the dataset. Fig. 5 which is 2d representation, gives us clear picture of separation of classes along the axes then Fig. 4. We can see how much of separation and overlapping is happening, and we see that again that there is not much overlapping. However, one question that comes to my mind is how much of separation or overlapping will happen if we have more than 5 to 10 classes because this dataset contained information on a smaller scale (only 3 classes) compared to dataset used in PCA.

4 Conclusion

To wrap up the observation from the results that were performed by PCA and LDA, we see that both algorithms are decreasing the dimensions of the large dataset with multiple classes and features or dimensions. However, for PCA and LDA to perform at high level, there must be some sort of data cleaning that needs to happen because it may produce results that maybe biased or wrong. In real-world, data is extremely large and not always clean, therefore it is important and necessary to apply preprocessing of the data as I do in this project. I had to remove some columns from the dataset where I had no use of it. For example, in the Iris and breast cancer dataset I to remove column called 'ID' which served no purpose in the algorithm. To conclude this, we see that PCA is applied to data identifies the combination of components that has the most variance in the data and LDA identifies classes that account for most variance between/within classes.

Resources:

- UCI Machine Learning. "Breast Cancer Wisconsin (Diagnostic) Data Set" Kaggle, 25 September 2016, https://www.kaggle.com/uciml/breast-cancer-wisconsin-data?select=data.csv

- UCI Machine Learning. "Iris Species" Kaggle, 27 September 2016, https://www.kaggle.com/datasets/uciml/iris
- https://en.wikipedia.org/wiki/Linear_discriminant_analysis
- https://en.wikipedia.org/wiki/Principal_component_analysis
- Jolliffe Ian T. and Cadima Jorge, et al. "Principal Component Analysis: A Review and Recent Developments." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 13 Apr. 2016, https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202#:~:text=Principal%20component%20analysis%20(PCA)%20is,variables%20that%20successively%20maximize%20variance.
- PCA versus LDA article from Week 1