PCA vs LDA: What's the Difference? Both PCA and LDA are linear transformation techniques. However, PCA is an unsupervised while LDA is a supervised dimensionality reduction technique.

PCA has no concern with the class labels. In simple words, PCA summarizes the feature set without relying on the output.

PCA tries to find the directions of the maximum variance in the dataset.

In a large feature set, there are many features that are merely duplicate of the other features or have a high correlation with the other features.

Such features are basically redundant and can be ignored. The role of PCA is to find such highly correlated or duplicate features and to come up with a new feature set where there is minimum correlation between the features or in other words feature set with maximum variance between the features.

Since the variance between the features doesn't depend upon the output, therefore PCA doesn't take the output labels into account.

Importing Libraries

```
import numpy as np
import pandas as pd
```

Importing the Dataset

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv(url, names=names)
```

```
dataset
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Data Preprocessing

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 labels and feature set:

```
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values
```

The above script assigns the first four columns of the dataset i.e. the feature set to X variable while the values in the fifth column (labels) are assigned to the y variable.

The following code divides data into training and test sets:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Feature Scaling

As was the case with PCA, we need to perform feature scaling for LDA too. Execute the following script to do so:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Performing LDA

It requires only four lines of code to perform LDA with Scikit-Learn.

The LinearDiscriminantAnalysis class of the sklearn.discriminant_analysis library can be used to Perform LDA in Python.

Take a look at the following script:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=1)
```

```
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```

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.

Notice, in case of LDA, the transform method takes two parameters: the X_train and the y_train.

However in the case of PCA, the transform method only requires one parameter i.e. X_train.

This reflects the fact that LDA takes the output class labels into account while selecting the linear discriminants, while PCA doesn't depend upon the output labels.

Training and Making Predictions

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 same Random Forest classifier that we used to evaluate performance of PCA-reduced algorithms.

Execute the following code:

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
y_pred
```

```
     array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

Evaluating the Performance

As always, the last step is to evaluate performance of the algorithm with the help of a confusion matrix and find the accuracy of the prediction. Execute the following script:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy1.0
```

We can see that with one linear discriminant, the algorithm achieved an accuracy of 100%, which is greater than the accuracy achieved with one principal component, which was 93.33%.

PCA vs LDA: What to Choose for Dimensionality Reduction?

In case of uniformly distributed data, LDA almost always performs better than PCA. However if the data is highly skewed (irregularly distributed) then it is advised to use PCA since LDA can be biased towards the majority class.

Finally, it is beneficial that PCA can be applied to labeled as well as unlabeled data since it doesn't rely on the output labels.

On the other hand, LDA requires output classes for finding linear discriminants and hence requires labeled data.

Colab paid products - Cancel contracts here