Experiment No. 9
Implementation of Principle Component Analysis as Dimensionality Reduction Technique.
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Implementation of Principle Component Analysis as Dimensionality Reduction Technique.

Objective: Able to perform Singular Value Decomposition for obtaining Dimensionality Reduction that yields Principal Component Analysis results.

Theory:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value

ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.
   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it.

After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

Implementation:

```
from tensorflow.keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 50s 0us/step
```

```
print('Traning data shape:', X_train.shape)

print('Testing data shape:', X_test.shape)
```

```
Traning data shape: (50000, 32, 32, 3)
Testing data shape: (10000, 32, 32, 3)
```

y_train.shape, y_test.shape

```
((50000, 1), (10000, 1))
```

import matplotlib.pyplot as plt

%matplotlib inline

label_dict = {

 0: 'airplane',

 1: 'automobile',

 2: 'bird',

 3: 'cat',

 4: 'deer',

 5: 'dog',

 6: 'frog',

 7: 'horse',

 8: 'ship',

 9: 'truck',

}

import numpy as np

plt.figure(figsize=[5,5])

Display the first image in training data

plt.subplot(121)

curr_img = np.reshape(X_train[0], (32,32,3))

plt.imshow(curr_img)

print(plt.title("(Label: " + str(label_dict[y_train[0][0]]) + ")"))

Display the first image in testing data

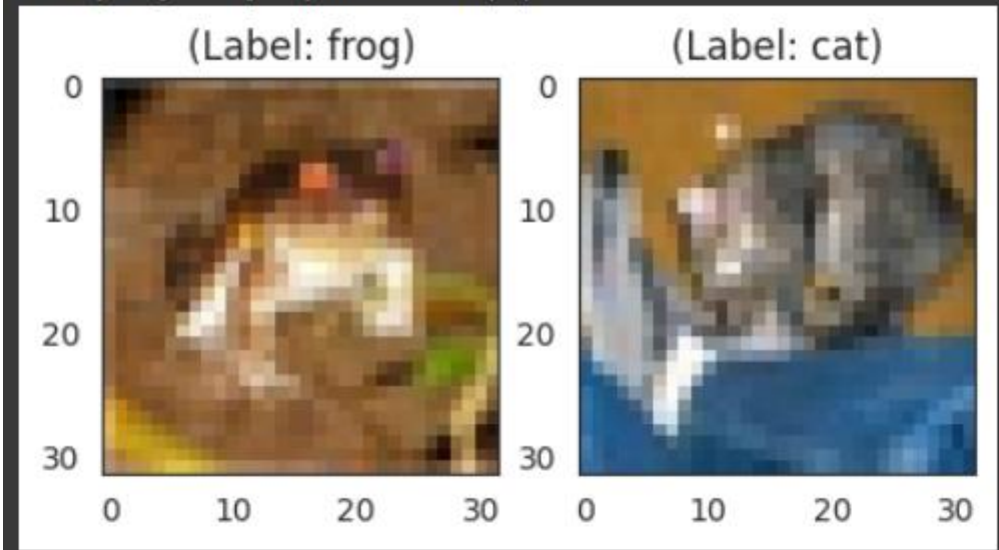
plt.subplot(122)

curr_img = np.reshape(X_test[0],(32,32,3))

plt.imshow(curr_img)

print(plt.title("(Label: " + str(label_dict[y_test[0][0]]) + ")"))

```
Text(0.5, 1.0, '(Label: frog)')
Text(0.5, 1.0, '(Label: cat)')
```



np.min(X_train), np.max(X_train)

```
(0, 255)
```

X_train = X_train / 255.0

np.min(X_train), np.max(X_train)

```
(0.0, 1.0)
```

X_train.shape

```
(50000, 32, 32, 3)
```

x_train_flat = X_train.reshape(-1,3072)

feat_cols = ['pixel'+str(i) for i in range(x_train_flat.shape[1])]

import pandas as pd

df_cifar = pd.DataFrame(x_train_flat, columns=feat_cols)

df_cifar['label'] = y_train

print('Size of the dataframe: {}'.format(df_cifar.shape))

```
Size of the dataframe: (50000, 3073)
```

from sklearn.decomposition import PCA

pca_cifar = PCA(n_components=2)

principalComponents_cifar = pca_cifar.fit_transform(df_cifar.iloc[:,:-1])

principal_cifar_Df = pd.DataFrame(data = principalComponents_cifar

, columns = ['principal component 1', 'principal component 2'])

principal_cifar_Df['y'] = y_train

principal_cifar_Df.head()

	principal component 1	principal component 2	y
0	-6.401018	2.729039	6
1	0.829783	-0.949943	9
2	7.730200	-11.522102	9
3	-10.347817	0.010738	4
4	-2.625651	-4.969240	1

print('Explained variation per principal component: {}'.format(pca_cifar.explained_variance_ratio_))

```
Explained variation per principal component: [0.2907663  0.11253144]
```

import seaborn as sns

plt.figure(figsize=(16,10))

sns.scatterplot(

x="principal component 1", y="principal component 2",

hue="y",

palette=sns.color_palette("hls", 10),

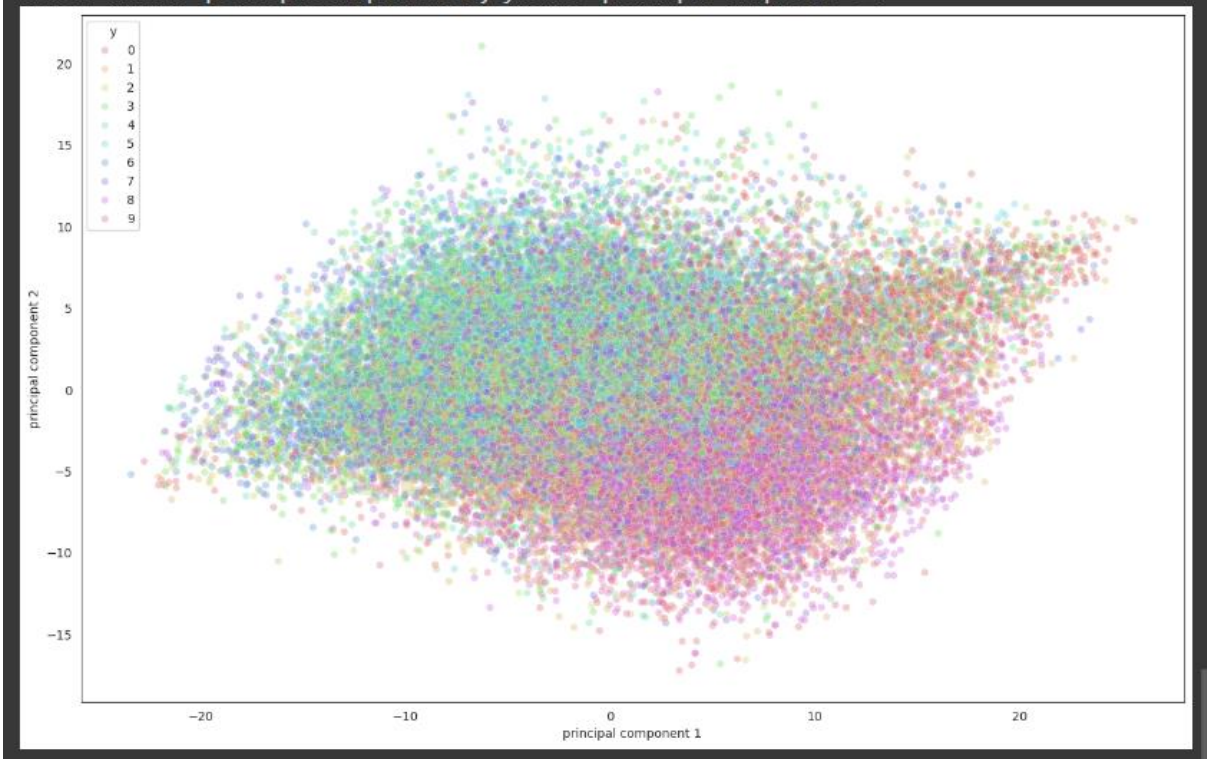
data=principal_cifar_Df,

legend="full",

alpha=0.3

)

<Axes: xlabel='principal component 1', ylabel='principal component 2'>



Conclusion:

1. **What is PCA?**

   PCA, or Principal Component Analysis, is a widely used dimensionality reduction technique in machine learning and data analysis. Its primary objective is to transform high-dimensional data into a lower-dimensional space while preserving as much of the original variance as possible. PCA achieves this by identifying the principal components, which are orthogonal vectors that represent the directions of maximum variance in the data. These components are ordered by the amount of variance they capture, allowing for the retention of the most important information while reducing the dimensionality of the dataset. PCA finds applications in various fields, including image processing, data visualization, and feature extraction, enabling more efficient analysis and interpretation of complex datasets.

2. How it is used?

PCA is utilized in various ways across different domains. In data preprocessing, it aids in reducing the computational complexity of algorithms by decreasing the number of features, thus improving efficiency without significant loss of information. In data exploration and visualization, PCA enables the representation of high-dimensional data in a lower-dimensional space, facilitating easier interpretation and analysis. Moreover, PCA is employed in feature extraction, where it identifies the most relevant features that contribute the most to the variance in the dataset, helping to improve the performance of machine learning models by focusing on the most informative aspects of the data. Overall, PCA serves as a versatile tool for dimensionality reduction, data compression, visualization, and feature extraction, enhancing the efficiency and effectiveness of various data analysis tasks.