**PART A**

(PART A : TO BE REFFERED BY STUDENTS)

**EXPERIMENT NO. 2**

**A.1 AIM: -** Handwritten Digit Recognition System using PCA

**A.2 Prerequisite**

∙ Different programming language (Python or Java), Understanding of

Machine Learning Algorithms, Machine Learning Algorithms

**A.3 Outcome**

After successful completion of this experiment students will be able to understand working of Convolutional Neural Networks (CNN) and apply this algorithm wherever required

**A.4 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.

**HOW DO YOU DO A PRINCIPAL COMPONENT ANALYSIS?**

1. Standardize the range of continuous initial variables

2. Compute the covariance matrix to identify correlations

3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

4. Create a feature vector to decide which principal components to keep 5. Recast the data along the principal components axes

**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, andunimportant features will be removed out.

**A5. Task**

**Given the MNIST data set your goal is to correctly identify digits from a dataset of tens of thousands of handwritten images. Perform Handwriting detection using PCA.**

**Link: http://yann.lecun.com/exdb/mnist/**

**Or**

**Link: https://www.kaggle.com/competitions/digit-recognizer**

**Note: Assume necessary Details. Use Exploratory Data Analysis and show details.**

**You can use any technique for pre-processing if required.**

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No. : C027 | Name: Vishesh Giyanani |
| Class : B | Batch : EB1 |
| Date of Experiment: 05/12/2024 | Date of Submission: 05/12/2024 |
| Grade : |  |

**B.1 Documentation written by student:**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

# %%

train\_df = pd.read\_csv("/Vishesh/MLOA/Exp2/input/digit-recognizer/train.csv")

test\_df = pd.read\_csv("/Vishesh/MLOA/Exp2/input/digit-recognizer/test.csv")

train\_df

# %%

train\_X,train\_y = train\_df.iloc[:,1:].to\_numpy(), train\_df.iloc[:,[0]].to\_numpy()

test\_X,test\_y = test\_df.iloc[:,1:].to\_numpy(), test\_df.iloc[:,[0]].to\_numpy()

# %%

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

train\_X = ss.fit\_transform(train\_X)

test\_X = ss.fit\_transform(test\_X)

# %% [markdown]

# # Before PCA

# %%

print(f"Columns before PCA: {train\_X.shape[1]}")

# %%

import numpy as np

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

find\_pca = PCA()

find\_pca.fit(train\_X)

exp\_var\_PCA = find\_pca.explained\_variance\_ratio\_

cumsum\_exp\_var = np.cumsum(exp\_var\_PCA)

plt.step(

range(0, len(cumsum\_exp\_var)),

cumsum\_exp\_var,

*label*='Cumulative explained variance',

*color*='C1'

)

plt.plot()

# %%

best\_components = np.argmax(cumsum\_exp\_var > 0.90)

best\_components

# %%

best\_pca = PCA(*n\_components*=best\_components)

train\_X\_pca = best\_pca.fit\_transform(train\_X)

test\_X\_pca = best\_pca.fit\_transform(test\_X)

# %%

print(f"Columns before PCA: {train\_X.shape[1]}")

print(f"Columns after PCA: {train\_X\_pca.shape[1]}")

# %% [markdown]

# # Creating model

# %%

from sklearn.model\_selection import GridSearchCV

from sklearn.svm import SVC

param\_grid = {'kernel':['linear', 'poly'],

'degree':[1, 2,],

'gamma': [0.01, 0.1],

'coef0': [0.5, 1]}

grid\_svm = GridSearchCV(*estimator*=SVC(*probability* = True),*param\_grid*=param\_grid,*scoring*='accuracy',*verbose*=3,*return\_train\_score*=True,*cv*=3,*refit*=True)

grid\_svm.fit(train\_X\_pca,train\_y)

# %%

grid\_svm.best\_params\_,grid\_svm.best\_estimator\_

# %%

predictions\_test = grid\_svm.predict(test\_X\_pca)

predictions\_train = grid\_svm.predict(train\_X\_pca)

# %%

from sklearn.metrics import classification\_report,accuracy\_score

print(classification\_report(test\_y,predictions\_test))

accuracy\_score(test\_y,predictions\_test)

**A screenshot of a computer

Description automatically generated**

**A screen shot of a graph

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**B.2 Observations and learning:**

PCA is a very important aspect of preprocessing and reduces the number of dimensions to fit the data in a faster and easier manner to the model. To get a good subset we need to set a variance threshold and check the threshold with the explained\_variance\_ratio for all the difference n\_components.

**B.3 Conclusion:**

PCA is a very important aspect of preprocessing and reduces the number of dimensions to fit the data in a faster and easier manner to the model. It not only reduces the time to fit a particular model to the data but also reduces the risk of overfitting of a model to the data.