**PRACTICAL 6**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Semester:** | VII | **Division:** | 6 |
| **Roll No.:** | 21BCP359 | **Date:** | 03-09-24 | **Batch:** | G11 |
| **Aim:** | Understanding Principal Component Analysis in Datasets. | | | | |

1. **Using sklearn library import the digits datasets (i.e. through load\_digits()). It is a 8 x 8 pixel images dataset and they are 64-dimensional. In order to retrieve some of the intuition behind the relationship between these points, make use of PCA to reduce the dimension to a manageable number of dimensions (i.e., 2). After reducing the dimension plot them using scatter plots with cmap.**
2. **How many number of components will be ideal is an important part of using PCA. Using this data plot the cumulative explained variance ratio as a function of the number of components.**
3. **Try to reconstruct the data using the largest subset of principal components. The idea behind this is that any components with variance much larger than the effect of the noise should be relatively unaffected by the noise. Add some random noise to the dataset and replot it.**

**Code**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_digits

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

*# Load the digits dataset*

digits = load\_digits()

X = digits.data  # Feature matrix

y = digits.target  # Labels

*# Standardize the data (PCA works better on standardized data)*

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

*# Apply PCA to reduce the dimensionality to 2 dimensions*

pca\_2d = PCA(n\_components=2)

X\_pca\_2d = pca\_2d.fit\_transform(X\_scaled)

*# Scatter plot of the 2D PCA-reduced data*

plt.figure(figsize=(8, 6))

scatter = plt.scatter(X\_pca\_2d[:, 0], X\_pca\_2d[:, 1], c=y, cmap='Spectral', edgecolor='k', s=60)

plt.colorbar(scatter)

plt.xlabel('First Principal Component')

plt.ylabel('Second Principal Component')

plt.title('2D PCA of Digits Dataset')

plt.show()

*# Plot the cumulative explained variance as a function of the number of components*

pca\_full = PCA().fit(X\_scaled)

plt.figure(figsize=(8, 6))

plt.plot(np.cumsum(pca\_full.explained\_variance\_ratio\_), marker='o')

plt.xlabel('Number of Components')

plt.ylabel('Cumulative Explained Variance Ratio')

plt.title('Cumulative Explained Variance Ratio by PCA Components')

plt.grid(True)

plt.show()

*# Reconstruct the data using the largest subset of principal components*

n\_components = 30  # Select the top 30 components

pca\_reconstruct = PCA(n\_components=n\_components)

X\_pca\_reduced = pca\_reconstruct.fit\_transform(X\_scaled)

X\_reconstructed = pca\_reconstruct.inverse\_transform(X\_pca\_reduced)

*# Add noise to the dataset*

noise\_factor = 0.5

X\_noisy = X\_scaled + noise\_factor \* np.random.normal(size=X\_scaled.shape)

*# Apply PCA again to the noisy data*

X\_pca\_noisy = pca\_2d.fit\_transform(X\_noisy)

*# Scatter plot of noisy PCA-reduced data*

plt.figure(figsize=(8, 6))

scatter\_noisy = plt.scatter(X\_pca\_noisy[:, 0], X\_pca\_noisy[:, 1], c=y, cmap='Spectral', edgecolor='k', s=60)

plt.colorbar(scatter\_noisy)

plt.xlabel('First Principal Component (Noisy)')

plt.ylabel('Second Principal Component (Noisy)')

plt.title('2D PCA of Noisy Digits Dataset')

plt.show()

**Output**





