BIG DATA ANALYSIS

**Project** :Split the data into 80%-20% ratio for training and testing randomly. 1. Find principal components that ensure 90% of the variation for each class in the training data and find the first two principal components of the whole training data. 2. Make reasonable normality assumption and estimate the parameters of the first two principal components of each class. 3. For a new image following classification rule is used: compute the likelihood using the joint distribution of first two principal components for each group. Whichever group has highest likelihood classify into that group. Test the success rate of this method using test data.

**STEPS IMPLEMENTED IN CODE :**

1. Data is imported and it is split into 80% for training and 20% for testing randomly i.e total 10,951 in it 8757 are for training 2190 for testing.
2. Next no.of principal components of each class[0,1,2,....9] that ensures 90% of the variation
3. And then finding the first two principal components of the training data
4. Estimating the parameters of the first two principal components of each class
5. Next we take test data and classify each data point into the class with maximum likelihood
6. Next the success rate is calculated i.e. Number of correctly classified images / total number of images.

import zipfile import os

import numpy as np from PIL import Image

from sklearn.model\_selection import train\_test\_split from sklearn.decomposition import PCA

from scipy.stats import multivariate\_normal

*# Path to the uploaded ZIP file*

zip\_path = 'C:\\Users\\HP\\Downloads\\Final Dataset-20240627T121942Z- 001.zip'

extract\_path = 'C:\\Users\\HP\\Downloads\\Final\_Dataset'

*# Extract the ZIP file*

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref: zip\_ref.extractall(extract\_path)

*# Function to load images and labels, and resize images*

def load\_data(directory, image\_size=(128, 128)): data = []

labels = []

for label\_dir in os.listdir(directory):

label\_path = os.path.join(directory, label\_dir) if os.path.isdir(label\_path):

for file in os.listdir(label\_path):

file\_path = os.path.join(label\_path, file) if file\_path.endswith('.png') or

file\_path.endswith('.jpg'):

image = Image.open(file\_path).convert('L') image = image.resize(image\_size)

image = np.array(image).flatten() data.append(image) labels.append(label\_dir)

return np.array(data), np.array(labels)

*# Load data with resized images*

data, labels = load\_data(extract\_path)

*# Split the data into training (80%) and testing (20%) sets* X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)

*# Perform PCA to find principal components ensuring 90% of the variation for each class*

class\_pca = {} class\_means = {} class\_covariances = {}

print("PCA Explained Variance Ratios for each class:") for label in np.unique(y\_train):

class\_data = X\_train[y\_train == label] pca = PCA(n\_components=0.9) pca.fit(class\_data)

class\_pca[label] = pca

transformed\_data = pca.transform(class\_data) class\_means[label] = np.mean(transformed\_data, axis=0)

class\_covariances[label] = np.cov(transformed\_data, rowvar=False) print(f"Class {label}: {pca.explained\_variance\_ratio\_}")

*# Find the first two principal components of the whole training data*

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

X\_train\_pca = pca\_all.fit\_transform(X\_train)

print("\nFirst two principal components of the whole training data:") print(pca\_all.components\_)

*# Estimate the parameters of the first two principal components for each class*

class\_pca\_2d\_means = {} class\_pca\_2d\_covariances = {}

print("\nParameters of the first two principal components for each class:")

for label in np.unique(y\_train):

transformed\_data = X\_train\_pca[y\_train == label] class\_pca\_2d\_means[label] = np.mean(transformed\_data, axis=0) class\_pca\_2d\_covariances[label] = np.cov(transformed\_data,

rowvar=False)

print(f"Class {label} Mean: {class\_pca\_2d\_means[label]}") print(f"Class {label} Covariance:

{class\_pca\_2d\_covariances[label]}")

*# Classification function based on the likelihood of the joint distribution of the first two principal components*

def classify\_image(image, class\_pca\_2d\_means, class\_pca\_2d\_covariances):

transformed\_image = pca\_all.transform([image])[0] likelihoods = {}

for label in class\_pca\_2d\_means: mean = class\_pca\_2d\_means[label]

cov = class\_pca\_2d\_covariances[label] likelihoods[label] =

multivariate\_normal.pdf(transformed\_image, mean=mean, cov=cov) return max(likelihoods, key=likelihoods.get)

*# Test the success rate of the classification method using the test data*

correct\_predictions = 0

for image, true\_label in zip(X\_test, y\_test):

predicted\_label = classify\_image(image, class\_pca\_2d\_means,

class\_pca\_2d\_covariances)

if predicted\_label == true\_label: correct\_predictions += 1

success\_rate = correct\_predictions / len(y\_test) print(f'\nSuccess rate: {success\_rate \* 100:.2f}%')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PCA Explained Variance Ratios for each class:  Class Final Dataset: [0.1894123 0.08290179 0.04418831 0.04116238  0.03321517 0.02224378 | | | | | |
| 0.02007158 | 0.01877281 | 0.01448812 | 0.01238556 | 0.01227758 | 0.01186291 |
| 0.0117311 | 0.01109165 | 0.01004302 | 0.00937449 | 0.00898574 | 0.00879577 |
| 0.00829743 | 0.00807912 | 0.00791085 | 0.00726139 | 0.00709967 | 0.00677846 |
| 0.00671349 | 0.00621068 | 0.00589119 | 0.00576133 | 0.00559478 | 0.00542195 |
| 0.00527661 | 0.00522565 | 0.00515228 | 0.004826 | 0.00477322 | 0.00463051 |
| 0.00458022 | 0.00438651 | 0.00418996 | 0.00415843 | 0.00400312 | 0.00390581 |
| 0.00378175 | 0.00364343 | 0.00361653 | 0.00358477 | 0.00337435 | 0.0033532 |
| 0.00328592 | 0.00327463 | 0.00321401 | 0.00314166 | 0.00311051 | 0.00300075 |
| 0.00295508 | 0.00285951 | 0.00279156 | 0.00275579 | 0.0026512 | 0.0026425 |
| 0.00260155 | 0.00257148 | 0.00250667 | 0.00247815 | 0.00243705 | 0.00241943 |
| 0.00233055 | 0.00231326 | 0.00224812 | 0.00221818 | 0.002133 | 0.00212558 |
| 0.0021046 | 0.00206248 | 0.00200812 | 0.0019891 | 0.00197027 | 0.00193677 |
| 0.00190316 | 0.00188738 | 0.00184063 | 0.00183519 | 0.00179812 | 0.00176747 |
| 0.00174261 | 0.00171624 | 0.00166887 | 0.00165127 | 0.00161098 | 0.00159901 |
| 0.00157008 | 0.00155738 | 0.00154183 | 0.00152327 | 0.00152114 | 0.00149739 |
| 0.00145519 | 0.00143255 | 0.00142135 | 0.00139667 | 0.00138203 | 0.00136034 |
| 0.00133244 | 0.00131449 | 0.00129607 | 0.00127772 | 0.00125824 | 0.00124564 |
| 0.00123438 | 0.00122826 | 0.00120912 | 0.00118208 | 0.00117831 | 0.00115336 |
| 0.00114533 | 0.00114191 | 0.00110465 | 0.00109723 | 0.00107463 | 0.00107085 |
| 0.00106051 | 0.00104939 | 0.00103125 | 0.00102702 | 0.00101828 | 0.0010064 |
| 0.0009995 | 0.00098597 | 0.00097117 | 0.00095866 | 0.00095399 | 0.00094766 |
| 0.00093051 | 0.00092358 | 0.00091393 | 0.00089901 | 0.000889 | 0.00088717 |
| 0.00088115 | 0.00087576 | 0.00085714 | 0.00084413 | 0.00084148 | 0.0008339 |
| 0.00082742 | 0.00082044 | 0.00081042 | 0.00079867 | 0.00078391 | 0.00078097 |
| 0.00077497 | 0.00077236 | 0.0007653 | 0.00075776 | 0.00074406 | 0.00073622 |
| 0.00072938 | 0.00072214 | 0.00071398 | 0.00070781 | 0.00070376 | 0.00069478 |
| 0.00068746 | 0.00068339 | 0.00067501 | 0.00067044 | 0.00066343 | 0.00066303 |
| 0.00065867 | 0.00065431 | 0.00064618 | 0.0006341 | 0.00063034] |  |

First two principal components of the whole training data:

[[-0.00212649 -0.00228918 -0.00257557 ... 0.00922898 0.00944659

0.00950645]

[ 0.01897664 0.02148762 0.02772018 ... -0.00489034 -0.00639537

-0.00696587]]

Parameters of the first two principal components for each class: Class Final Dataset Mean: [1.79054823e-15 8.09639200e-14]

Class Final Dataset Covariance: [[5.32243895e+06 8.50425819e-12]

[8.50425819e-12 2.32951976e+06]]

Success rate: 95.96%