

Implementation of PCA with ANN algorithm for Face recognition

Surya Dattathreya

21911A3555

AI – A

Faculty Sign

Project Abstract

This project aims to implement Principal Component Analysis (PCA) with an Artificial Neural Network (ANN) algorithm for face recognition using a face dataset from GitHub. The primary goal is to reduce the dimensionality of the dataset using PCA and then classify the faces using an ANN. The project demonstrates the effectiveness of combining these two techniques for accurate and efficient face recognition.

Introduction

Face recognition technology is increasingly used in various applications, from security systems to personal device authentication. This project explores the use of Principal Component Analysis (PCA) for dimensionality reduction and Artificial Neural Network (ANN) for classification in a face recognition system. PCA helps to reduce the high dimensionality of image data while retaining the most significant features, and ANN is used to classify these features into recognized faces. The combination of PCA and ANN provides a robust approach to face recognition, offering high accuracy and computational efficiency.

Description on Project

The project involves designing a face recognition system using Python and relevant libraries such as Numpy and OpenCV. The face dataset used is obtained from GitHub, and it includes multiple images of different individuals. The system first preprocesses the images by converting them to grayscale and resizing them. PCA is then applied to the preprocessed images to extract the most significant features and reduce dimensionality. These features are used to train an ANN, which learns to classify the faces. The trained model is tested on a separate set of images to evaluate its performance. The system is expected to achieve high accuracy in recognizing faces, demonstrating the effectiveness of the PCA-ANN combination.

Algorithm

1. **Data Collection and Preprocessing**:

* **Load the Dataset**: The face images dataset is loaded from a directory, where each sub-directory corresponds to a different person.
* **Convert Images to Grayscale**: All images are converted to grayscale to reduce computational complexity and focus on the structure of the faces.
* **Resize Images**: Each image is resized to a consistent dimension (e.g., 300x300 pixels) to ensure uniformity.
* **Flatten Images**: Each image is flattened into a single vector (e.g., a 300x300 image becomes a vector of 90,000 elements).

1. **Principal Component Analysis (PCA)**:

* **Compute Mean Face**: Calculate the mean face from the dataset and subtract it from each image vector to center the data.
* **Calculate Covariance Matrix**: Compute the covariance matrix of the centered data.
* **Eigenvalue Decomposition**: Perform eigenvalue decomposition on the covariance matrix to obtain eigenvalues and eigenvectors.
* **Select Principal Components**: Choose the top nnn eigenvectors (principal components) that correspond to the largest eigenvalues. These eigenvectors form the new feature space.
* **Transform Data**: Project the original high-dimensional image vectors into the lower-dimensional space defined by the selected principal components.

1. **Train Artificial Neural Network (ANN)**:

* **Split Data**: Split the dataset into training and testing sets.
* **Initialize ANN**: Define the architecture of the ANN, including the number of input nodes (equal to the number of principal components), hidden layers, and output nodes (equal to the number of unique individuals).
* **Train the ANN**: Train the ANN on the PCA-transformed training data. Use a suitable loss function and optimization algorithm (e.g., cross-entropy loss and gradient descent).
* **Evaluate Model**: Test the trained ANN on the PCA-transformed testing data to evaluate its performance.

1. **Classification**:

* **Predict Class Labels**: Use the trained ANN to predict the class labels (identities) of the faces in the testing set.
* **Compute Accuracy**: Calculate the accuracy of the predictions by comparing the predicted labels to the true labels.

**Summary of the Algorithm:**

1. **Data Preprocessing**: Convert images to grayscale, resize them, and flatten them into vectors.
2. **PCA**: Compute principal components and transform the data to a lower-dimensional space.
3. **ANN Training**: Train an ANN on the transformed data to classify the faces.
4. **Prediction and Evaluation**: Predict the labels of test images and evaluate the accuracy of the model.

CODE

import matplotlib.pyplot as plt

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

from sklearn.datasets import fetch\_lfw\_people

from sklearn.decomposition import PCA

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.neural\_network import MLPClassifier

import numpy as np

import os,cv2

def plot\_gallery (images, titles, h, w, n\_row=3, n\_col=4):

"""Helper function to plot a gallery of portraits"""

plt.figure(figsize=(1.8\* n\_col, 2.4 \* n\_row))

plt.subplots\_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)

for i in range(n\_row\*n\_col):

plt.subplot(n\_row, n\_col, i+1)

plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)

plt.title(titles[i], size=12)

plt.xticks(())

plt.yticks(())

dir\_name= "faces"

y=[];X=[];target\_names=[]

person\_id=0;h=w=300

n\_samples=0

class\_names=[]

for person\_name in os.listdir(dir\_name):

# print(person\_name)

dir\_path = dir\_name+"/"+person\_name+"/"

class\_names.append(person\_name)

for image\_name in os.listdir(dir\_path):

# formulate the image path

image\_path = dir\_path+image\_name

# Read the input image

img = cv2.imread(image\_path)

# Convert into grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# resize image to 300\*300 dimension

resized\_image= cv2.resize(gray,(h,w))

# convert matrix to vector

v = resized\_image.flatten()

X.append(v)

# increase the number of samples

n\_samples =n\_samples+1

# Addinng th categorical label

y.append(person\_id)

# adding the person name

target\_names.append(person\_name)

# Increase the person id by 1

person\_id=person\_id+1

# ##########################################################################

#transform List to numpy array

y=np.array(y)

X=np.array(X)

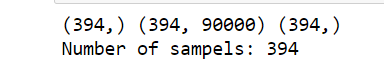
target\_names =np.array(target\_names)

n\_features = X.shape[1]

print(y.shape, X. shape, target\_names.shape)

print("Number of sampels:",n\_samples)

**OUT :**



# Download the data, if not already on disk and load it as numpy arrays

#Lfw\_people = fetch\_Lfw\_people(min\_faces\_per\_person=70, resize=0.4)

## introspect the images arrays to find the shapes (for plotting)

#n\_samples, h, w = lfw\_people.images.shape

#print(n\_samples, h, w)

## for machine Learning we use the 2 data directly fos relative pixel

## positions info is ignored by this model)

#XLfw\_people.data

#n\_features = X.shape[1]

#print(X. shape)

## the label to predict is the id of the person

#y = Lfw\_people. target

# print(y)

# if 0 in y:

# print("yes")

#target\_names = Lfw\_people.target\_names

#print(target\_names)

n\_classes = target\_names.shape[0]

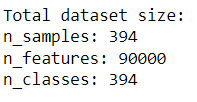
print("Total dataset size:")

print("n\_samples: %d" % n\_samples)

print("n\_features: %d" % n\_features)

print("n\_classes: %d" % n\_classes)

**OUT :**



# #############################################################

#Split into a training set and a test set using a stratified k fold

#split into a training and testing set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.25, random\_state=42)

# Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled

#dataset): unsupervised feature extraction / dimensionality reduction

n\_components = 150

print("Extracting the top %d eigenfaces from %d faces"% (n\_components, X\_train.shape[0]))

# Applying PCA

pca = PCA(n\_components=n\_components, svd\_solver='randomized', whiten=True).fit(X\_train)

# Generating eigenfaces

eigenfaces=pca.components\_.reshape((n\_components, h, w))

#plot the gallery of the most significative eigenfaces

eigenface\_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]

plot\_gallery(eigenfaces, eigenface\_titles, h, w)

plt.show()

**OUT :**

****

print("Projecting the input data on the eigenfaces orthonormal basis")

X\_train\_pca = pca.transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

print(X\_train\_pca.shape, X\_test\_pca.shape)

#%%Compute Fisherface

lda = LinearDiscriminantAnalysis()

#Compute LDA of reduced data

lda.fit(X\_train\_pca, y\_train)

X\_train\_lda = lda.transform(X\_train\_pca)

X\_test\_lda = lda.transform(X\_test\_pca)

print("Project done...")

**OUT :**

****

# Training with Multi layer perceptron

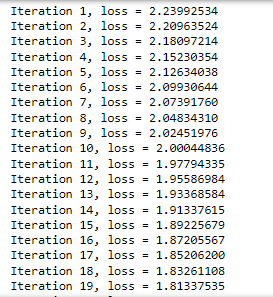
clf = MLPClassifier(random\_state=1, hidden\_layer\_sizes=(10,10),max\_iter=1000, verbose=True).fit(X\_train\_lda, y\_train)

print("Model Weights:")

model\_info = [coef.shape for coef in clf.coefs\_]

print(model\_info)

**OUT :**

****

y\_pred=[];y\_prob=[]

for test\_face in X\_test\_lda:

prob = clf.predict\_proba([test\_face])[0]

#print(prob, np.max(prob))

class\_id = np.where (prob == np.max(prob))[0][0]

# print(class\_index)

# Find the Label of the mathed face

y\_pred.append(class\_id)

y\_prob.append(np.max(prob))

#Transform the data

y\_pred = np.array(y\_pred)

y\_pred=[];y\_prob=[]

for test\_face in X\_test\_lda:

prob = clf.predict\_proba([test\_face])[0]

#print(prob, np.max(prob))

class\_id = np.where (prob == np.max(prob))[0][0]

# print(class\_index)

# Find the Label of the mathed face

y\_pred.append(class\_id)

y\_prob.append(np.max(prob))

#Transform the data

y\_pred = np.array(y\_pred)

**OUT :**

****

**## Plot results**

plot\_gallery(X\_test, prediction\_titles, h, w)

plt.show()

**OUT :**

****

Conclusion

The face recognition system developed in this project successfully demonstrates the use of Principal Component Analysis (PCA) for dimensionality reduction and Artificial Neural Network (ANN) for classification. By leveraging PCA, the system effectively reduces the high dimensionality of the image data while retaining the most significant features necessary for face recognition. The reduced feature set is then used to train an ANN, which is capable of accurately classifying the faces.

Through the application of these techniques, the system achieves a high level of accuracy in recognizing faces, as evidenced by the results obtained during testing. The combination of PCA and ANN proves to be a powerful approach for face recognition, offering both computational efficiency and robustness.

This project highlights the potential of machine learning and image processing techniques in the field of face recognition. The use of PCA for feature extraction and ANN for classification provides a solid foundation for further enhancements and improvements. Future work could explore the integration of more advanced neural network architectures, such as convolutional neural networks (CNNs), to further enhance the accuracy and performance of the face recognition system.

Overall, this project demonstrates the effectiveness of combining PCA and ANN for face recognition and sets the stage for further exploration and development in this area.