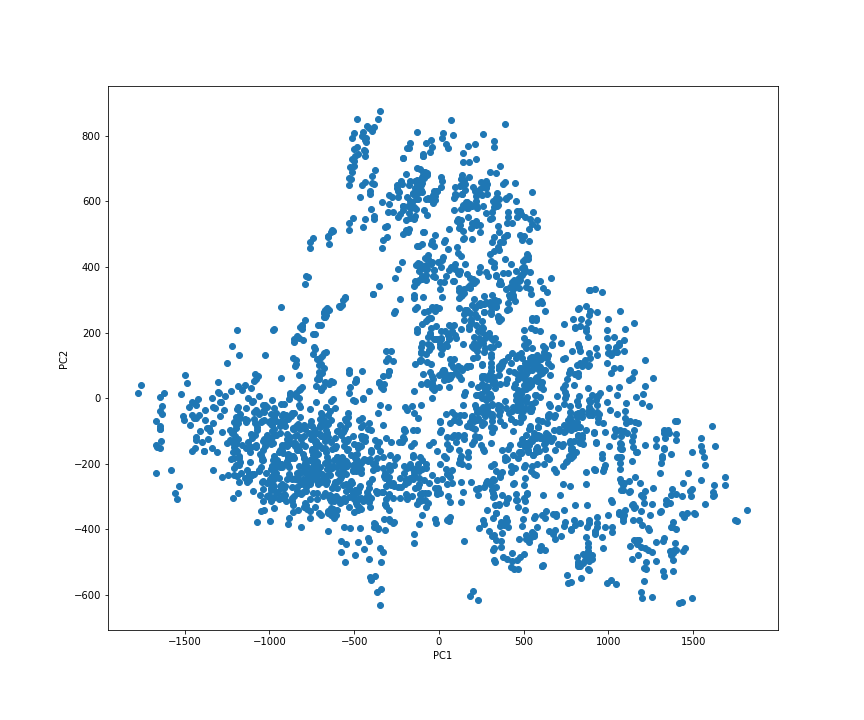
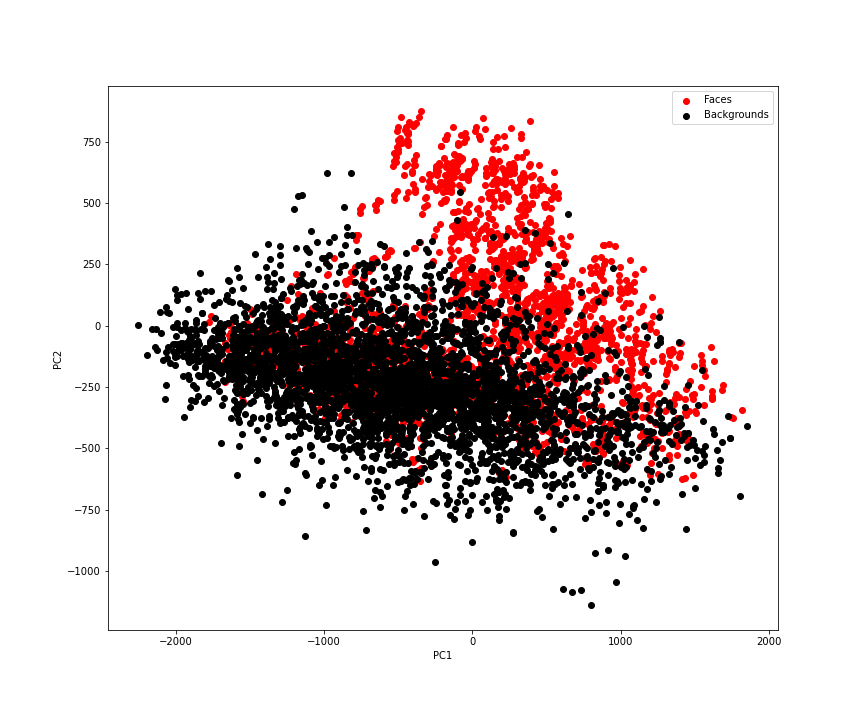
Jarod Klion

Homework 11

STA5635

April 5, 2022

* 1. Shape, square

     Description automatically generatedeigenvalues sorted in decreasing order
  2. Coordinates of the faces projected to the 2D plane generated by PC1 and PC2
  3. Coordinates of faces and backgrounds projected to 2D plane generated by PC1 and PC2
  4. A picture containing logo

     Description automatically generatedDistances to 15-PC plane vs projections of faces and backgrounds on the first PC
  5. Chart, histogram

     Description automatically generatedHistogram of distances from above

#!/usr/bin/env python

# coding: utf-8

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

import os

#read face images into a dict

faces = {}

for filename in os.listdir('../datasets/faces/'):

if not filename.startswith('face0'):

continue #not a faces pic

faces[filename] = plt.imread(os.path.join('../datasets/faces',filename))

#look at a subset of images

fig, axes = plt.subplots(4, 4, sharex = True, sharey = True, figsize = (10,12))

faceimages = list(faces.values())[:16]

for i in range(16):

axes[i % 4][i // 4].imshow(faceimages[i], cmap = 'gray')

plt.show()

faceshape = list(faces.values())[0].shape

print("Face image shape:", faceshape)

#create matrix to perform PCA

facematrix = []

for key, val in faces.items():

facematrix.append(val.flatten())

#(n\_samples, n\_pixels) matrix -> (2429, 19x19)

facematrix = np.array(facematrix)

facematrix\_centered = facematrix - np.mean(facematrix, axis = 0)

print(facematrix.shape)

# # Part A

pca = PCA().fit(facematrix\_centered)

#print/store eigenvectors

#print(pca.components\_)

eigenvectors = pca.components\_[3:]

#print/store eigenvalues

eigenvalues = pca.explained\_variance\_[3:]

#make eigenvalue indices to plot against

eigenvalues\_idx = range(1, len(eigenvalues) + 1)

#plot eigenvalues

plt.figure(figsize = (13, 8))

plt.plot(eigenvalues\_idx, eigenvalues)

plt.xlabel("Indices")

plt.ylabel("Eigenvalues")

plt.title("Eigenvalues vs Indices")

plt.savefig("Eigenvalues.png")

plt.show()

# # Part B

two\_pcas = PCA(n\_components=2).fit(facematrix\_centered)

faces\_two = two\_pcas.transform(facematrix\_centered)

#plot projection onto PC1 and PC2

plt.figure(figsize = (12, 10))

plt.scatter(faces\_two[:,0], faces\_two[:,1])

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.savefig("Faces Projection.png")

plt.show()

# # Part C

bgs = {}

for filename in os.listdir('../datasets/faces/'):

if not filename.startswith('B'):

continue #not a background pic

bgs[filename] = plt.imread(os.path.join('../datasets/faces',filename))

fig, axes = plt.subplots(4, 4, sharex = True, sharey = True, figsize = (10,12))

bgimages = list(bgs.values())[:16]

for i in range(16):

axes[i % 4][i // 4].imshow(bgimages[i], cmap = 'gray')

plt.show()

bgshape = list(bgs.values())[0].shape

print("Background image shape:", bgshape)

#create matrix

bgmatrix = []

bglabel = []

for key, val in bgs.items():

bgmatrix.append(val.flatten())

bglabel.append(key.split("\_")[0])

#(n\_samples, n\_pixels) matrix -> (3699, 19x19)

bgmatrix = np.array(bgmatrix)

bgmatrix\_centered = bgmatrix - np.mean(facematrix, axis = 0)

print(bgmatrix\_centered.shape)

bgs\_two = two\_pcas.transform(bgmatrix\_centered)

plt.figure(figsize = (12, 10))

plt.scatter(faces\_two[:,0], faces\_two[:,1], color = "red", label = "Faces")

plt.scatter(bgs\_two[:,0], bgs\_two[:,1], color = "black", label = "Backgrounds")

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.legend()

plt.savefig("Faces and Backgrounds Projections.png")

plt.show()

# # Part D

fifteen\_pca\_components = pca.components\_[:15]

first\_comp = PCA(n\_components=1).fit(facematrix\_centered)

faces\_one = first\_comp.transform(facematrix\_centered)

bgs\_one = first\_comp.transform(bgmatrix\_centered)

#project of coordinates to PC plane

faceweights = fifteen\_pca\_components @ (facematrix\_centered).T

bgweights = fifteen\_pca\_components @ (bgmatrix\_centered).T

#create list for later storage

facedists = []

bgdists = []

for i in range(facematrix\_centered.shape[0]):

query\_weight = fifteen\_pca\_components @ facematrix\_centered[i, :, np.newaxis]

facedists.append(np.sqrt(np.linalg.norm(faceweights - query\_weight)))

for i in range(bgmatrix\_centered.shape[0]):

query\_weight = fifteen\_pca\_components @ bgmatrix\_centered[i, :, np.newaxis]

bgdists.append(np.sqrt(np.linalg.norm(bgweights - query\_weight)))

plt.figure(figsize = (12, 10))

plt.scatter(faces\_one, facedists, color = "red", label = "Faces")

plt.scatter(bgs\_one, bgdists, color = "black", label = "Backgrounds")

plt.xlabel("Projection to PC1")

plt.ylabel("Distance to plane")

plt.legend()

plt.savefig("Distances vs Projection.png")

plt.show()

# # Part E

#plot histogram of distances

plt.hist(bgdists, color = 'black', label = 'Backgrounds')

plt.hist(facedists, color = 'red', label = 'Faces')

plt.legend()

plt.savefig("Histogram.png")

plt.show()