Eigenfaces - 160300153

February 12, 2021

[23]:
```
from os import walk
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import fetch_lfw_people
from sklearn.decomposition import PCA
import pandas as pd
```

0.1 PCA

La idea detras de PCA es la reduccion de la dimensionalidad de un conjunto de datos y seleccionar el vector que nos permita obtener la mayor cantidad de informacion. Escencialmente, las egienfaces son obtenidas de los vectores utilizados en algebra lineal llamados egienvectores los cuales apuntan la direccion a la que la informacion se maximiza.

En el caso particular de las imagenes, podemos obtener dimensionalidades inmensas en los conjuntos de datos, por ejemplo, en la actualidad una imagen tomada desde una camara de smartphone tiene una tamaño de aproximadamente 1944 * 2592 lo que nos daria una dimensionalidad de: 1944 X 2592 = 5038848

Esta es una dimensionalidad inmensa lo que hace un analisis muy complejo debido a la dimensionalidad de estos, aqui es donde entra PCA, ya que nos permite construir un subespacio de dimensionalidades simples que mejor permita obtener la variacion entre los datos. Partiendo del punto en el que obtenemos la mayor cantidad de variacion en la informacion calculando los componentes principales, seguimos manteniendo los aspectos importantes de nuestros datos y de la misma forma facilitamos el poder clasificar nuestros conjuntos.

0.2 Comenzando

Ahora que entendemos un poco de la idea base de las eigenfaces o eigenvectores podemos comenzar, primero vamos a obtener las eigenfaces de un conjunto de fotos, en este caso utilzare fotos mias para este proyecto, posteriormente, utilizare un conjunto de datos llamado Labeled Faces in the Wild que contiene conjuntos de datos especiales para el estudio del reconocimiento facial

0.2.1 Lectura de imagenes

Primero realizamos la lectura de las imagenes

En esta ocacion, estoy utilizando un conjunto de 10 imagenes todas con una resolucion de: 1944 X 2592

[31]:
```
X = plt.imread('./photos/01.jpg')
#Obtenemos el tamaño de la imagen
shape = X.shape
print(' Resolucion de la imagen', shape  )

faces = pd.DataFrame()
for path,data,files in walk('./photos'):
    for i in files:
        X = plt.imread(path+'/'+i)
        X = X.astype(np.uint8)
        X = X/255
        face = pd.Series(X.flatten(),name=i)
        faces = faces.append(face)

fig, axes = plt.subplots(2,5,
 subplot_kw={'xticks':[], 'yticks':[]},
 gridspec_kw=dict(hspace=0.01, wspace=0.01))

for i, ax in enumerate(axes.flat):
    ax.imshow(faces.iloc[i].values.reshape(shape),cmap="gray")
```

```
Resolucion de la imagen (1944, 2592)
```

0.2.2 Aplicando PCA

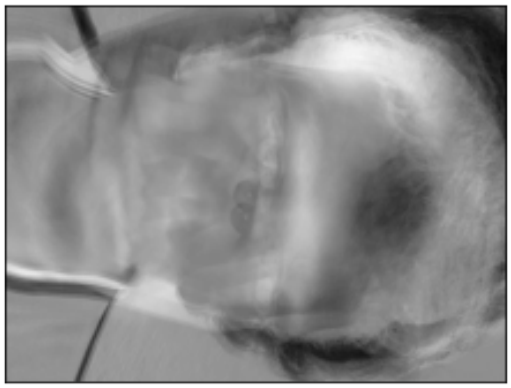
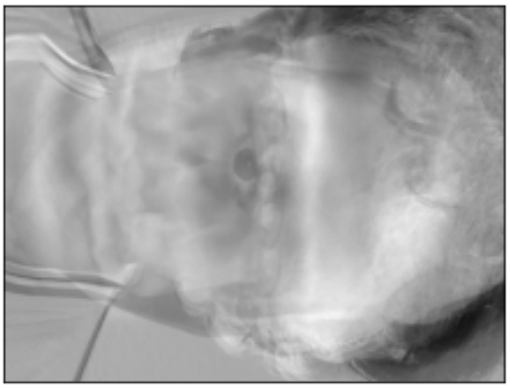
Aplicamos PCA para reducir la dimensionalidad de los datos a 2

[33]:
```
n=2
faces_pca = PCA(n_components=n)
faces_pca.fit(faces)
```

[33]:
```
PCA(n_components=2)
```

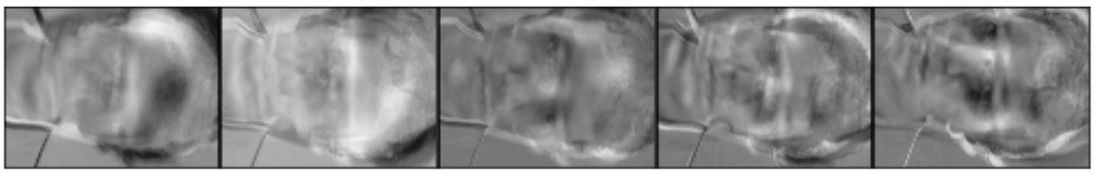
Y finalmente graficamos las eigenfaces obtenidas

[35]:
```
faces_pca.transform(faces)
fig, axes = plt.subplots(1,n,figsize=(9,3),
    subplot_kw={'xticks':[], 'yticks':[]},
 gridspec_kw=dict(hspace=0.01, wspace=0.01))
for i, ax in enumerate(axes.flat):
    ax.imshow(faces_pca.components_[i].reshape(shape),cmap="gray")
```

El resultado en lo personal se me hizo comico ya que parecen fantasmas, ahora realizare el mismo procedimiento pero con una dimensionalidad de 5

[36]:
```
n=5
faces_pca = PCA(n_components=n)
faces_pca.fit(faces)
faces_pca.transform(faces)
fig, axes = plt.subplots(1,n,figsize=(9,3),
    subplot_kw={'xticks':[], 'yticks':[]},
 gridspec_kw=dict(hspace=0.01, wspace=0.01))
for i, ax in enumerate(axes.flat):
    ax.imshow(faces_pca.components_[i].reshape(shape),cmap="gray")
```



0.3 Utilizando Labeled Faces in the Wild

Utilizando el conjunto de datos Labeled Faces in the Wild proyectare mi rostro con los datos obtenidos en las eigenfaces de este conjunto.

0.3.1 Obtenemos el conjunto de datos y le damos una dimensionalidad de 5

[37]:
```
n_components = 5
lfw_people = fetch_lfw_people(min_faces_per_person=10, resize=0.4)
X = lfw_people.data
pca = PCA(n_components=n_components, svd_solver='randomized',whiten=True).fit(X)
```

0.3.2 Lectura de imagen

Se realiza la lectura de la imagen y se proyecta en el conjunto de datos

[42]:
```
image = plt.imread('./01.jpg').reshape(-1,1)
r = pca.transform(image)
```

0.3.3 Obtencion de imagen

Finalmente obtenemos la imagen proyectada mediante la siguiente formula:

$ r1C1+r2C2+r3C3+r4C4+r5C5 $

donde ri son los valores obtenidos mediante la proyeccion y Ci son las componentes principales

[43]:
```
c = pca.components_
X_train_pca = pca.transform(image)
total = np.zeros(X_train_pca.shape[0])
fig, ax = plt.subplots(1,2,
         subplot_kw={'xticks':[], 'yticks':[]},
         gridspec_kw=dict(hspace=0.01, wspace=0.01))

for i in range(0,X_train_pca.shape[1]):
    val = c[i,:] * r[:,i]
    total += val

ax[0].imshow(total.reshape(50,37),cmap='gray')
ax[1].imshow(image.reshape(50,37),cmap='gray')
```

[43]:
```
<matplotlib.image.AxesImage at 0x7fad877b0a00>
```

Como podemos ver, tenemos la imagen proyectada y es bastante similar a la original, ahora probare con diferentes dimensionalidades, cabe mencionar que el maximo de dimensionalidad a utilizar es 1850 debido a que es la dimensionalidad original de los datos.

Para ello realice la siguiente funcion que hace exactamente lo mismo que en las ultimas 3 celdas de codigo por fines de espacio

[44]:
```
def Eigenfaces(n_components):
    image = plt.imread('./01.jpg').reshape(-1,1)
    lfw_people = fetch_lfw_people(min_faces_per_person=10, resize=0.4)
    X = lfw_people.data
    pca = PCA(n_components=n_components, svd_solver='randomized',
              whiten=True).fit(X)
    c = pca.components_
    X_train_pca = pca.transform(image)
    total = np.zeros(X_train_pca.shape[0])
    fig, ax = plt.subplots(1,2,
         subplot_kw={'xticks':[], 'yticks':[]},
         gridspec_kw=dict(hspace=0.01, wspace=0.01))

    for i in range(0,X_train_pca.shape[1]):
        val = c[i,:] * X_train_pca[:,i]
        total += val


    ax[0].imshow(total.reshape(50,37),cmap='gray')
    ax[1].imshow(image.reshape(50,37),cmap='gray')
```

0.4 Dimensionalidad de 20

[45]:
```
Eigenfaces(20)
```

0.5 Dimensionalidad de 50

[46]:
```
Eigenfaces(50)
```

0.6 Dimensionalidad de 300

[47]:
```
Eigenfaces(300)
```

0.7 Dimensionalidad de 1850

[48]:
```
Eigenfaces(1850)
```

0.7.1 Conclusiones

Como podemos ver, gracias a la reduccion de dimensionalidad podemos facilitar la proyeccion de las imagenes, en este caso fue notorio que a menor dimensionalidad de los datos obtenemos una cara mejor reconstruida y que al utilizar la dimensionalidad original obtenemos una imagen poryectada a la cual solo se le puede distinguir la silueta.

Vazquez Pompa Noe - 160300153