Pattern recognition versus Principal component analysis

We will use the PCA (Principal Component Analysis) as a dimension reduction tool on images. The goal is to recognize the gestures of a person's face contained in a series of images.

LIBRARY

To play with PCA : scikit-learn
To play with images : matplotlib

MATERIAL

To understand dimension reduction with PCA go to the section « PCA as Noise Filtering ».
https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Dimension reduction:
https://towardsdatascience.com/principal-component-analysis-your-tutorial-and-code-9719d3d3f376

PCA on images
https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial

TIPS

To display the gray images of the database:

```
img_test = mpimg.imread(PathBaseYes)
img = np.array(img_test)
# Display
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
plt._show()
```

To load all the images from a directory, SizeX and SizeY are the image sizes

```
def PCA_base_creation(Base_dir_name):
    vect_database = []
    for ii in range(500) :
        try:
            File_name = Base_dir_name + "im" + str(ii).zfill(3) + ".bmp"
            print(File_name)
            img = mpimg.imread(File_name)
            size = SizeX*SizeY
            print(size)
            vect = img.reshape(size)
```

```
            vect_database.append(vect)
        except:
            break;
    return vect_database
```

The PCA methods:

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA

TODO

In the data zip file, you will find three folders

- yes: images of a face making a “yes” gesture
- no: images of a face making a “no” gesture
- test; a test image

1) Perform a PCA on the base of « yes », project one image of « yes » database on the eigenvectors, retain three projection coefficients, rebuild the image from those coefficients and analyze the result. Do the same thing retaining all the coefficients.

2) Calculate an average projection error (Euclidean distance, max of the absolute value the error on an image) between the images and the reconstruction of the projected images in the base of the eigenvectors. Use this error to measure the quality of reconstruction when you learn on a basis (yes for example) and you test projection / reconstruction on a different basis (no). Learn on a base and test on the same base. Display some reconstructed images and compare them to the original images to subjectively measure the reconstruction. Conclude.

3) Vary the number of eigenvectors selected and measure the impact on the reconstruction error and the subjective quality of the reconstruction. Display the results as a graph.

4) Download a face database

```
from sklearn.datasets import fetch_olivetti_faces
faces = fetch_olivetti_faces()
```

Display the images

```
images = faces.images # save images
print("image size:", faces.images.shape)
fig = plt.figure() # create a new figure window
for i in range(20): # display 20 images
    # subplot : 4 rows and 5 columns
    img_grid = fig.add_subplot(4, 5, i+1)
    # plot features as image
```

```
    img_grid.imshow(images[i], cmap='gray')

plt.show()
```

Perform a PCA on this database.
Take five new images with the same dimension, one of them must represent a face, the other ones can represent what you want (car, flower…).
Use the projection error in order to detect in an automatic manner, the image with the face.