4 Tutorial 4: Image Processing & Analysis

In this tutorial, we will apply the Principal Component Analysis (PCA) algorithm to a dataset of images to build a simple image recognition software. The dataset is provided to you and consists of 20 images, as shown in Fig(6): 5 images of George Clooney (the famous Hollywood actor), 5 images of Roger Federer (former professional tennis player), 5 images of Barack Obama (the 44th president of the United States), and 5 images of Margret Thatcher (prime minister of the United Kingdom from 1979 to 1990).

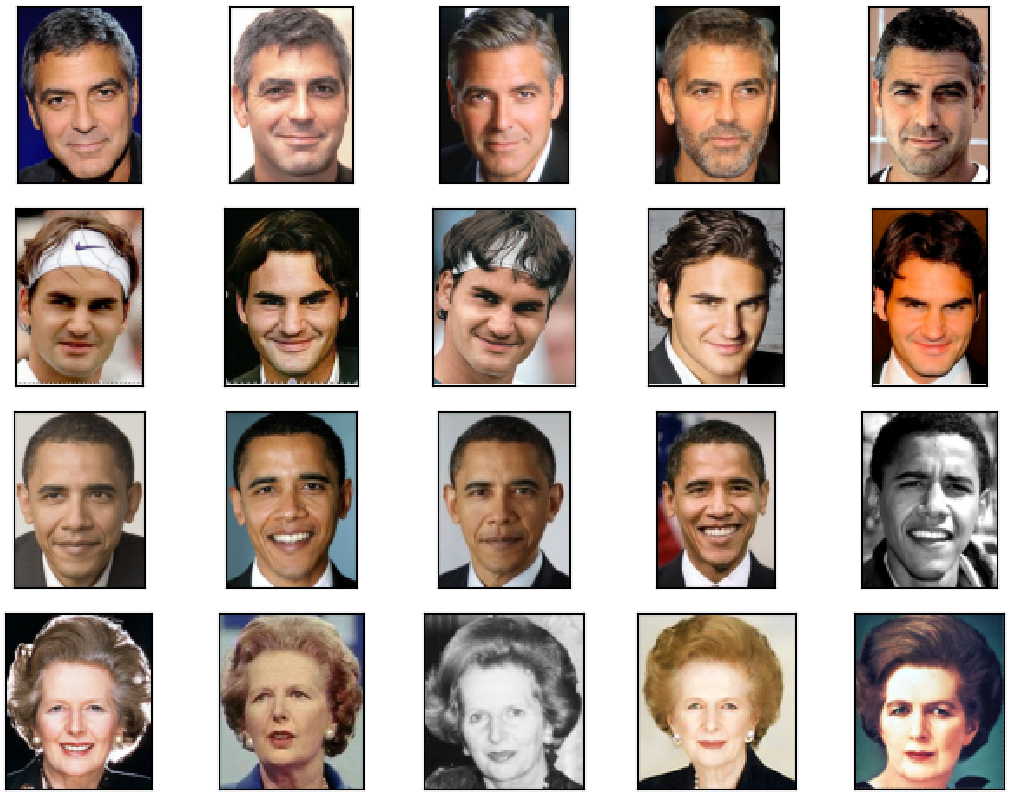

Figure 6: The image dataset that you will be working on during this tutorial. There are 20 images: 5 images of George Clooney (first row), 5 images of Roger Federer (second row), 5 images of Barack Obama (third row), and 5 images of Margret Thatcher (fourth row).

This tutorial focuses on using the Principal Component Analysis (PCA) algorithm to motivate the importance of eigenvalues and eigenvectors to be considered in the context of the computer vision problem of human face recognition. The approach of using so-called eigenfaces for recognition was developed by Sirovich and Kirby [?, ?] and used by Turk and Pentland [?] in face classification. It is considered the first successful example of facial recognition technology. The scheme is based on an information theory approach that decomposes face images into a small set of characteristic feature images called "eigenfaces," which may be thought of as the principal components of the initial training set of face images. Recognition is performed by projecting a new image into the subspace spanned by the eigenfaces ("face space") and then classifying the face by comparing its position in face space with the positions of known individuals.

In the language of information theory, we want to extract the relevant information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of models encoded similarly. A simple approach to extracting the information contained in an image of a face is to somehow capture the variation in a collection of face images, independent of any judgment of features, and use this information to encode and compare individual face images [?].

In mathematical terms, we wish to find the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images, treating an image as a point (or vector) in a very high dimensional space. The eigenvectors are ordered, each one accounting for a different amount of variation among the face images. These eigenvectors can be thought of as a set of features that together characterize the variation between face images [?].

In what follows, some very basic ideas of eigenfaces will be given. To develop this simple software, follow the following steps in order.

1. **Prepare the image dataset**.

   - **Download the "training" set of images.** This will be a highly limited example as we will consider only faces from four different people representing the political, entertainment, and sports arenas, namely George Clooney, Barack Obama, Margaret Thatcher, and Roger Federer. For each celebrity, five images are considered to be cropped the same way. So, in total, you should have 20 colored images. All of the images have a format of ".jpg". This "training" dataset of images will be given to you in a folder titled "faces".
   - **Import and pre-process the images**. Once you have the image dataset on your computer, read all the images, turn them into their grayscaled versions, convert them into "double" precision numbers, and resize them to the same number of pixels so they have a dimension of $m \times n = 120 \times 80$. Can you do all these steps in one line of code for each image? Visualize the images in a grayscale colormap using a subplot figure.

2. **Training**.

- **Calculate the average face**. To start understanding the face recognition process, we can begin by trying to understand the concept of an average face. To do that, compute the average of the five images for each individual divided by the total number of images, i.e., 5. You should end up with four images corresponding to each celebrity's average face. Visualize the averaged faces in a grayscale colormap using a subplot figure.
- **Construct the images dataset**. Arrange (concatenate) the images into a matrix so that:

$$\mathbf{D} = \begin{bmatrix} \text{image}_1 \\ \text{image}_2 \\ \vdots \\ \text{image}_N \end{bmatrix}$$

  where each image is reshaped into a vector with one row and $m \times n$ columns using the command: `image = np.reshape(img,(1,m*n))` where the `img` is the grayscaled, resized image.
- **Compute the covariance matrix**. What the average face emphasizes is the common features among a group of pictures, i.e. the pixels that are correlated. This gives a clue on how to pursue a face recognition algorithm: look for the correlations and common features of faces. Alternatively, two faces that are highly uncorrelated are most likely different. Correlations between data sets can be computed using the covariance. To compute a correlation matrix, it only remains to multiply each row vector by each other row vector which can be computed as $\mathbf{Cov} = \mathbf{D}^T\mathbf{D}$. This results in a very large covariance matrix of the size $(m \times m)$.
- **Compute the eigenvalues and eigenvectors of the covariance matrix**. Our objective is then to compute this matrix's eigenvalues and eigenvectors to gain insight into the face recognition algorithm. The matrix of eigenvectors should be a square matrix of the size $\{(m \times n) \times (m \times n)\}$. Additionally, you should get $(m \times n)$ nonzero positive eigenvalues in the diagonal elements of the eigenvalues matrix.
- **Sort and select**. Sort the eigenvalues and the eigenvectors in ascending order and choose a truncation limit k to slice the eigenvectors columns and the diagonal elements of the eigenvalues. The truncated matrix of the eigenvectors (or the eigenvectors space or the face space) should have the size of $\{(m \times n) \times k\}$ and the eigenvalues matrix should have the size of $k \times k$. There is a quick way to do this (both in Matlab and Python) which is via `eigs` command:

In Matlab, this can be done by: `[V,E] = eigs(covariance,20,'lm');`
In Python, this can be done by:

```
from scipy.sparse.linalg import eigs
E, V = eigs(covariance, 20, which='LM')
```

Here, the E and V are the eigenvalues and the eigenvectors of the covariance matrix, respectively, $k = 20$ is the chosen truncation limit, and `'lm'` or `'LM'` means the eigen largest magnitude.

Visualise the trend of the eigenvalues which should show a decrease in their value. The magnitude of the eigenvalues is directly related to how important its eigenvector is in composing the images. Also, visualize the first four columns of the eigenvectors which should show the most dominant features of the images being trained.

- **Project the average faces along the principal components**. To represent an individual in terms of the eigenvectors, it remains to understand the weighting of each image on the eigenvectors space (or the face space). This can be done by projecting the average faces on the eigenvectors to compute the PCA modes of each face, i.e., the fingerprint of each face and a unique representation of each celebrity on the eigenvectors. These modes can be used for classification tasks. To do that, first, reshape each average face into a row vector of the size $1 \times (m \times n)$. Then, compute the inner product between the vectorized average face and the eigenvectors. The result should be a row vector of a size $1 \times k$. Visualize these projections for each individual as a bar chart. Note that each celebrity has a different set of coefficients in this representation. It is these differences that can be used as the basis for face recognition.

3. **Testing**. To demonstrate how the eigenvectors can be used in practice in the context of face recognition, three new images are introduced, as shown in Fig(7): a new picture of Margaret Thatcher (left), a picture of Meryl Streep as Margret thatcher taken from her role in the Iron Lady movie (middle), and the final one is for Hilary Clinton (the US secretary of state).

   To test these three images, do the following:

   - Read these images, turn them to grayscale, convert them into a double precision, and resize them to the same as before $(m \times n)$. Visualize these test images.
   - Reshape each test image into a vector of a size $(1 \times (m \times n))$.

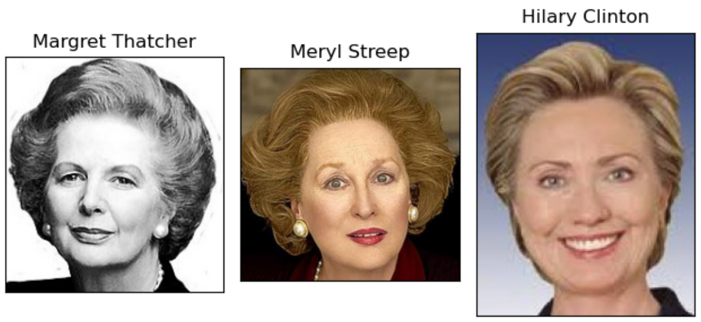

Figure 7: Three testing images: (left) Another image of Margret Thatcher (Prime Minister of the United Kingdom from 1979 to 1990 and Leader of the Conservative Party from 1975 to 1990), (middle) Meryl Streep playing Margret Thatcher in the 2011 Iron Lady movie, and (right) Hilary Clinton (67th United States secretary of state in the administration of Barack Obama from 2009 to 2013, as a U.S. senator).

- Project the vectorized test images into the eigenvectors, i.e., compute the inner product between vectorized test images and the eigenvectors. Visualize the three projections as a bar chart.
- Calculate the error or the difference between each of these test images and all the trained five images of Margret Thatcher to see which of these test images looks the closest to Margret Thatcher. To do that, you need to compute:

$$\text{Error}_j = \frac{\|c_j - c_{\text{new}}\|}{\|c_j\|}$$

  where c_j is the projection of one of the trained images and c_{new} is the projection of one of the test images. Visualize the results as a bar chart. For each test image, you should have five bars that correspond to each of the Margret Thatcher trained images.

4. **Answer the following questions**.

- How would you improve the eigenface methodology?
- Take a selfie and pass it on through your software to test which celebrity you look much alike. For this, you need to calculate the difference between the projection of your selfie on the face space and the projections of each of the trained images.
- If you still have time in the tutorial, apply one of the edge detection methods on the images as a pre-processing step before constructing the image dataset. Would that enhance the accuracy of the eigenface method?