Machine Learning NYUSH
PCA Assignment

May 9, 2017

We use the Olivetti face dataset.[1] The data contains 400 face images of size 64×64. In faces.csv, each line represents a face image. The first 64 values represent the first column of the image, and the next 64 values represent the second column and so on.

We provide some sample code. The sample code shows how to load the data and display the first image. There are also hints in the comments which will aid you in writing your own code.

In the sample code, you will find the lines "Your Code Here", below which you may start writing your code.

Your tasks are as follows:

1. Display a face image chosen randomly from the 400 images.
2. Compute and display the mean of the faces. Subtract the mean from each of the faces to get the "centered faces".
3. For each centered face, compute its covariance matrix. Then compute the average covariance matrix $\mathbf{V}$ by averaging the 400 covariance matrices. Note that $\mathbf{V}$ is a 64×64 $(4,096)$ matrix
4. Calculate the eigenvalues and eigenvectors of the covariance matrix V using the method stated in the eigenface tutorial.
5. Display the first 10 principal components.
6. Reconstruct the first face using the first two principal components.
7. Randomly choose a face, reconstruct it using $5, 10, 25, 50, 100, 200, 300, 399$ principal components, and show the reconstructed images.
8. Recall that in principal component analysis, the total variance of the data is given by the sum of all the eigenvalues, i.e., $\sum_j \lambda_j$. The proportion of variance explained by the ith principal component is given by $\frac{\lambda_i}{\sum_j \lambda_j}$. Plot the proportion of variance explained by all the principal components.

Eigenfaces Tutorial

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The main idea is to represent a face using a linear composition of base features or images (called eigenfaces). For more information on eigenfaces, see [?], [?] or the eigenfaces wikipedia page.

Suppose we have a set of m images $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}$ which is represented using an $n \times n$ matrix, where

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} & p_{12}^{(r)} & \cdots & p_{1n}^{(r)} \\ p_{21}^{(r)} & p_{22}^{(r)} & \cdots & p_{2n}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(r)} & p_{n2}^{(r)} & \cdots & p_{nn}^{(r)} \end{bmatrix}_{n \times n}, \quad \text{for all } r = 1, \cdots, m.$$

[1] http://www.cs.nyu.edu/~roweis/data.html

$0 \leqslant p_{ij}^{(r)} \leqslant 255$ represents the pixel intensity, and $n \times n$ represents the number of pixels in the image. Now we wish to change the representation of our image into a vector of dimension n^2, we do this by concatenating all the columns of the matrix $\mathbf{x}^{(r)}$ as follows

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} \\ p_{21}^{(r)} \\ \vdots \\ p_{n1}^{(r)} \\ p_{12}^{(r)} \\ p_{22}^{(r)} \\ \vdots \\ p_{n2}^{(r)} \\ \vdots \\ p_{1n}^{(r)} \\ p_{2n}^{(r)} \\ \vdots \\ p_{nn}^{(r)} \end{bmatrix}_{n^2 \times 1}$$

where $r = 1, \cdots, m$ and $0 \leqslant p_{ij}^{(r)} \leqslant 255$. Our goal therefore, is to extract a lower dimension set of useful features out of these m n^2-dimensional vectors.

Since we are much more interested in the characteristic features of those faces, let's subtract everything that is common among them, i.e., the average face. The average face of the images can be defined as $\mathbf{m} = \frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)}$. We then redefine:

$$\mathbf{x}^{(r)} \leftarrow \mathbf{x}^{(r)} - \mathbf{m}.$$

So now $\frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)} = \mathbf{0}$. In order to find the principal components, we will attempt to find the eigenvectors $\mathbf{z}_j$ and the corresponding eigenvalues λ_j of the covariance matrix

$$\begin{aligned} \mathbf{V} &= \frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)} \mathbf{x}^{(r)T} \\ &= A^T A \end{aligned}$$

where the matrix $A^T = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}]$. However, because the dimension of the matrix $\mathbf{V}$ is $n^2 \times n^2$, the task of determining the eigenvalues and eigenvectors is intractable (to put things into perspective, suppose our set of images are 64×64, then the matrix $\mathbf{V}$ would have $16,777,216$ elements). Fortunately, we can make this problem more computationally feasible by solving a much smaller $m \times m$ matrix $L = AA^T$. Denote the eigenvectors of matrix L by $\mathbf{v}_j,\ j = 1, \cdots, m$. Observe that for $L\mathbf{v}_j = \lambda_j \mathbf{v}_j$,

$$\begin{aligned} A^T L \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ A^T A A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ \mathbf{V} A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \end{aligned}$$

and hence $\mathbf{z}_j = A^T \mathbf{v}_j$ and j are the eigenvectors and eigenvalues of $\mathbf{V}$, respectively. Thus, we can find the eigenvectors of $\mathbf{V}$ by first finding the eigenvectors of L, then multiplying each eigenvector $\mathbf{v}_j$ by A^T. One final step is that $\mathbf{z}_j$ needs to be normalized, i.e., $\|\mathbf{z}_j\| = 1$.

The eigenvectors $\mathbf{z}_j$ are the eigenfaces. You can view these faces by scaling them to 255 (this can be done automatically in Matlab or R). We can discard the components with the smallest eigenvalues. Only the k ($k \leqslant m$) ones with the largest eigenvalues (i.e., only the ones making the greatest contribution to the variance of the original image set) and chuck them into the matrix

$$U = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k]_{n^2 \times k}.$$

Once we have a new face image $\mathbf{x}$, it can then be transformed into its eigenface components by a simple operation

$$\Omega = U^T(\mathbf{x} - \mathbf{m}) = [\omega_1, \omega_2, \cdots, \omega_k]^T.$$

Notice that we have reduced an image of size $n \times n$ into a vector of length k. To approximate the original image $\mathbf{x}'$, all we have to do is to project it into the face space and adjust for the mean by

$$\mathbf{x}' = \mathbf{m} + U\Omega = \mathbf{m} + \sum_{j=1}^{k} \omega_j \mathbf{z}_j.$$

Eigenfaces Tutorial

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The main idea is to represent a face using a linear composition of base features or images (called eigenfaces). For more information on eigenfaces, see [?], [?] or the eigenfaces wikipedia page.

Suppose we have a set of m images $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ which is represented using an $n \times n$ matrix, where

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} & p_{12}^{(r)} & \cdots & p_{1n}^{(r)} \\ p_{21}^{(r)} & p_{22}^{(r)} & \cdots & p_{2n}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(r)} & p_{n2}^{(r)} & \cdots & p_{nn}^{(r)} \end{bmatrix}_{n \times n}, \quad \text{for all } r = 1, \cdots, m.$$

$0 \leqslant p_{ij}^{(r)} \leqslant 255$ represents the pixel intensity, and $n \times n$ represents the number of pixels in the image. Now we wish to change the representation of our image into a vector of dimension n^2, we do this by concatenating all the columns of the matrix $\mathbf{x}^{(r)}$ as follows

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} \\ p_{21}^{(r)} \\ \vdots \\ p_{n1}^{(r)} \\ p_{12}^{(r)} \\ p_{22}^{(r)} \\ \vdots \\ p_{n2}^{(r)} \\ \vdots \\ p_{1n}^{(r)} \\ p_{2n}^{(r)} \\ \vdots \\ p_{nn}^{(r)} \end{bmatrix}_{n^2 \times 1}$$

where $r = 1, \cdots, m$ and $0 \leqslant p_{ij}^{(r)} \leqslant 255$. Our goal therefore, is to extract a lower dimension set of useful features out of these m n^2-dimensional vectors.

Since we are much more interested in the characteristic features of those faces, let's subtract everything that is common among them, i.e., the average face. The average face of the images can be defined as $\mathbf{m} = \frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)}$. We then redefine:

$$\mathbf{x}^{(r)} \leftarrow \mathbf{x}^{(r)} - \mathbf{m}.$$

So now $\frac{1}{m}\sum_{r=1}^{m} \mathbf{x}^{(r)} = \mathbf{0}$. In order to find the principal components, we will attempt to find the eigenvectors $\mathbf{z}_j$ and the corresponding eigenvalues λ_j of the covariance matrix

$$\begin{aligned}\mathbf{V} &= \frac{1}{m}\sum_{r=1}^{m} \mathbf{x}^{(r)}\mathbf{x}^{(r)^T} \\ &= A^T A\end{aligned}$$

where the matrix $A^T = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}]$. However, because the dimension of the matrix $\mathbf{V}$ is $n^2 \times n^2$, the task of determining the eigenvalues and eigenvectors is intractable (to put things into perspective, suppose our set of images are 64×64, then the matrix $\mathbf{V}$ would have $16,777,216$ elements). Fortunately, we can make this problem more computationally feasible by solving a much smaller $m \times m$ matrix $L = AA^T$. Denote the eigenvectors of matrix L by $\mathbf{v}_j,\ j = 1, \cdots, m$. Observe that for $L\mathbf{v}_j = \lambda_j \mathbf{v}_j$,

$$\begin{aligned}A^T L\mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ A^T A A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ \mathbf{V} A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j\end{aligned}$$

and hence $\mathbf{z}_j = A^T\mathbf{v}_j$ and j are the eigenvectors and eigenvalues of $\mathbf{V}$, respectively. Thus, we can find the eigenvectors of $\mathbf{V}$ by first finding the eigenvectors of L, then multiplying each eigenvector $\mathbf{v}_j$ by A^T. One final step is that $\mathbf{z}_j$ needs to be normalized, i.e., $\|\mathbf{z}_j\| = 1$.

The eigenvectors $\mathbf{z}_j$ are the eigenfaces. You can view these faces by scaling them to 255 (this can be done automatically in Matlab or R). We can discard the components with the smallest eigenvalues. Only the k ($k \leqslant m$) ones with the largest eigenvalues (i.e., only the ones making the greatest contribution to the variance of the original image set) and chuck them into the matrix

$$U = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k]_{n^2 \times k}.$$

Once we have a new face image $\mathbf{x}$, it can then be transformed into its eigenface components by a simple operation

$$\Omega = U^T(\mathbf{x} - \mathbf{m}) = [\omega_1, \omega_2, \cdots, \omega_k]^T.$$

Notice that we have reduced an image of size $n \times n$ into a vector of length k. To approximate the original image $\mathbf{x}'$, all we have to do is to project it into the face space and adjust for the mean by

$$\mathbf{x}' = \mathbf{m} + U\Omega = \mathbf{m} + \sum_{j=1}^{k} \omega_j \mathbf{z}_j.$$