Principal Components Analysis

Santiago Paternain, Aryan Mokhtari and Alejandro Ribeiro

March 23, 2021

At this point we have already seen how the Discrete Fourier Transform and the Discrete Cosine Transform can be written in terms of a matrix product. The main idea is to multiply the signal by a specific *unitary matrix*. Let us remind ourselves of the definition of *unitary matrix*.

Definition 1 (Unitary matrix) *Let* $\mathbf{M} \in \mathbb{C}^{N\times N}$ *be a matrix with complex entries. Denote by* $\mathbf{M}^*$ *the conjugate of* $\mathbf{M}$*, i.e., for each entry* i, j *we have that* $(M^*)_{ij} = M^*_{ij}$*. Then, we say* $\mathbf{M}$ *is unitary if*

$$(\mathbf{M}^*)^T \mathbf{M} = \mathbf{M}^H \mathbf{M} = \mathbf{I}, \tag{1}$$

where $\mathbf{I}$ *is the* $N \times N$ *identity matrix and* $(\cdot)^T$ *denotes the transpose of a matrix. Note that we have defined the superscript symbol* H *to denote conjugate and transposition (this operation is called the* Hermitian *operation).*

For the Discrete Fourier transform we can define the following matrix

$$\mathbf{F} = \begin{bmatrix} \mathbf{e}_{0N}^H \\ \mathbf{e}_{1N}^H \\ \vdots \\ \mathbf{e}_{(N-1)N}^H \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi(1)(1)/N} & \cdots & e^{-j2\pi(1)(N-1)/N} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j2\pi(N-1)(1)/N} & \cdots & e^{-j2\pi(N-1)(N-1)/N} \end{bmatrix}. \tag{2}$$

Then, if we consider a signal $x(n)$ for $n = 0, \ldots, N-1$ as a vector

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}, \tag{3}$$

we can write the DFT as the product between F and the vector $\mathbf{x}$ i.e

$$\mathbf{Fx} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi(1)(1)/N} & \cdots & e^{-j2\pi(1)(N-1)/N} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j2\pi(N-1)(1)/N} & \cdots & e^{-j2\pi(N-1)(N-1)/N} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$
$$= \frac{1}{\sqrt{N}} \begin{bmatrix} \sum_{n=1}^{N-1} x(n) e^{-2\pi j \frac{0}{N} n} \\ \sum_{n=1}^{N-1} x(n) e^{-2\pi j \frac{1}{N} n} \\ \vdots \sum_{n=1}^{N-1} x(n) e^{-2\pi j \frac{N-1}{N} n} \end{bmatrix} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix}. \tag{4}$$

In PCA decomposition we define a new unitary matrix based on the eigenvectors of the covariance matrix of a dataset. Before defining the covariance matrix we need to define the sample mean signal.

Definition 2 (Sample mean) *Let $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_M$ be M different signals in a dataset. Then, the sample mean of the dataset is*

$$\bar{\mathbf{x}} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m. \tag{5}$$

We next define the sample covariance matrix

Definition 3 (Sample covariance matrix) *Let $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_M$ be M different signals in a dataset. Then, the sample covariance matrix of the dataset is*

$$\Sigma = \frac{1}{M} \sum_{m=1}^{M} (\mathbf{x}_m - \bar{\mathbf{x}})(\mathbf{x}_m - \bar{\mathbf{x}})^H. \tag{6}$$

Now that we know how to get the sample covariance matrix, we can define a unitary matrix that is "adapted" to our signals. Let $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N$ be the eigenvectors of the sample covariance matrix of the dataset. Then, as in the discrete Fourier transform, define the unitary matrix

$$\mathbf{P} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N]. \tag{7}$$

The PCA decomposition can be written as a product between $\mathbf{P}$ and the centered signal, i.e., the signal minus its sample mean. Explicitly,

$$\mathbf{X}_{PCA} = \mathbf{P}^H (\mathbf{x} - \bar{\mathbf{x}}). \tag{8}$$

Just as with the DFT and the DCT, we can also define the inverse operation to PCA decomposition, which gives us a signal based on $\mathbf{X}_{PCA}$. Because $\mathbf{P}$ is unitary, its inverse is given by $\mathbf{P}^H$. Hence, the inverse transformation is given by

$$\mathbf{x} = \mathbf{P}\mathbf{X}_{PCA} + \bar{\mathbf{x}}. \tag{9}$$

You can prove this is the case by plugging in (8) into (9).

In this Lab we are going to be taking PCA decomposition of faces, so that we need to deal with two-dimensional signals. A way of doing this is to vectorize the matrices representing the images. Let $\mathbf{Z}$ be a N by N matrix. Let $\mathbf{z}_k$ be the k-th column of the matrix, then we can represent the matrix $\mathbf{Z}$ as a concatenation of column vectors

$$\mathbf{Z} = \left[\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N\right]. \tag{10}$$

The one dimensional representation of the signal $\mathbf{Z}$ can be obtained by stacking its columns as in

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_N \end{bmatrix} \tag{11}$$

With this idea we can treat two dimensional signals as if they were one dimensional and use our original definition of the PCA. In what follows, we are going to use PCA decomposition for image compression. We can compress an image by keeping the eigenvectors associated with the larger eigenvalues of the covariance matrix. Next week, we will use a similar idea to build a face recognition system.

Since we haven't yet covered the concept of covariance matrix we are providing the sample covariance matrix, computed using (6), and the sample mean, computed using (2), of the data set (c.f. Figure 2b).

1 PCA: Decomposition

1.1 Decomposition in principal components. Build a class that, given a covariance matrix and the number of desired principal components K, returns the K eigenvectors of the covariance matrix corresponding to the K largest eigenvalues. Check that the matrix containing the eigenvectors is unitary.

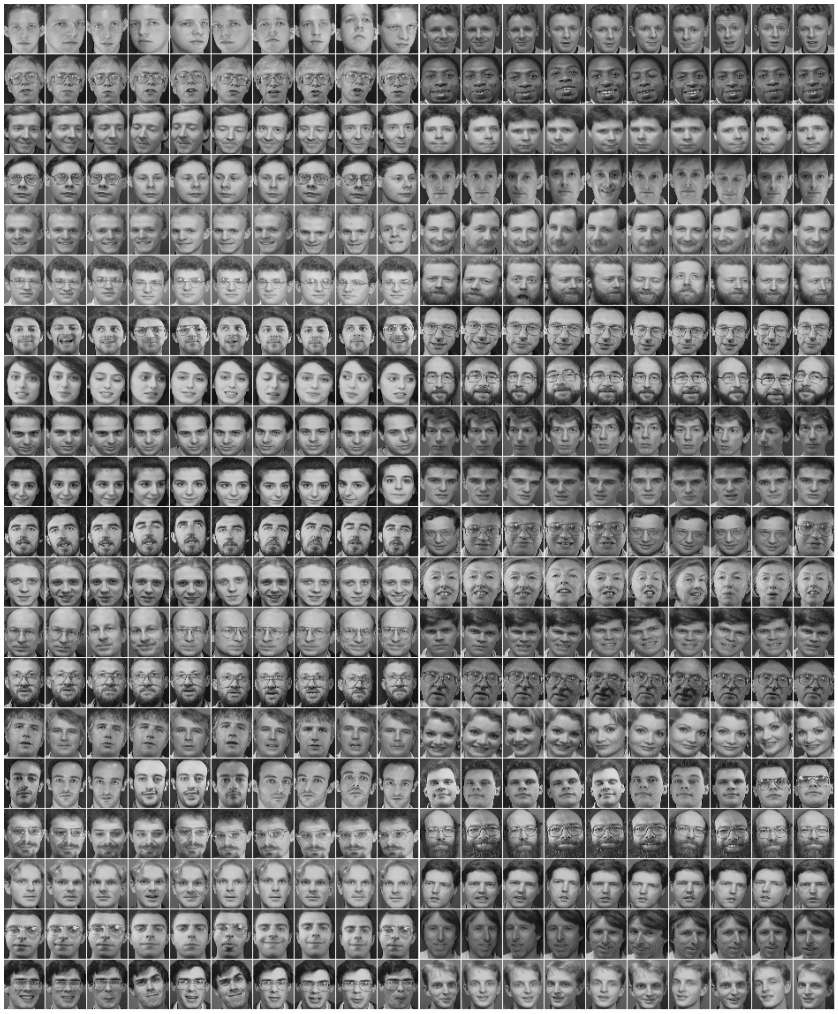

Figure 1: Dataset

(a) An example of the dataset

(b) This is the mean face of the dataset

1.2 Decomposition of a face in the space of eigenfaces. Build a class that takes as inputs an image containing a face, the mean face, and the eigenfaces and returns the projection in the space of eigenfaces. Note that eigenfaces are the eigenvectors represented as images.

2 Reconstruction

2.1 Reconstruction of a face using K principal components. Build a class that takes as inputs the mean face, K eigenfaces, and the K coefficients representing the projection onto those eigenfaces and outputs a reconstructed face. Test this class for any of the faces of the data set for 5, 25, and 50 principal components.

2.2 Reconstruction error. For any given number K of principal components (eigenfaces), we can compute the reconstruction error as $\|\mathbf{X} - \hat{\mathbf{X}}_K\|$, where $\mathbf{X}$ is the original image and $\hat{\mathbf{X}}_K$ is the image reconstructed using K eigenfaces. Compute this error ranging the number of eigenfaces used from $K = 0$ to the maximum number of eigenfaces (size of the image). How many principal components are needed to obtain a 60% reduction in the reconstruction error (as compared to $K = 0$)?