The lab studies the PCA decomposition properties on some visual datasets. [1]

1 Programming the PCA

You are asked to program the following functions:

-
  ```
  function [Lambda,U,meanX]=MyPCA(X)
  %%
  % Performs the extraction of the PCA components given a dataset
  % Input:    X, DxN matrix of N points x of dimension D
  % Output:
  %    Lambda : set of eigenvalues of the covariance matrix ranked in decreasing order
  %    U      : matrix of eigenvectors (ranked in the same order than eigenvalues)
  %    meanX  : average value of the datas in X
  %
  ```

  For this function, you can use for instance (and appropriately) the eig or SVD eigenvalue decomposition function from matlab.

-
  ```
  function [Y]=PCAProjection(Z,meanX,P)
  %%
  % Projects a matrix of data points Z on the first M eigenvectors
  % Input:
  %    Z     : DxN data matrix (N columns of data zi of dimension D)
  %    meanX : mean of data points provided by MyPCA
  %    P     : DxM projection matrix containing the first M eigenvectors obtained from MyPCA
  % Output:
  %    Y     : MxN matrix containing the components in PCA subspace for all data points of Z
  %
  ```

-
  ```
  function [Ztilde]=PCAReconstruction(Y,meanX,P)
  %%
  % Reconstructs data points given their coordinates Y in the space spanned by the M eigenvectors of P
  % Input:
  %    Y     : MxN coordinates of the N points to (re)construct
  %    meanX : mean of data points provided by MyPCA
  %    P     : DxM projection matrix containing the first M eigenvectors obtained from MyPCA
  % Output:
  %    Ztilde     : DxN matrix containing the constructed vectors
  %
  ```

[1]The lab is inspired by a lab from the ASI master program from INSA rouen.

2 Testing the PCA - Digit dataset

- Use the usps digit dataset (usps.mat file). Select one digit (avoid one or zero) and extract the image data corresponding to this character.
- Test the above programs on the dataset of the selected character. In particular, you can:
  1. compute the PCA components, and visualise them[2] Comment on what is captured by each eigenvector (if it make sense).
  2. select a few images from the same digit class, and reconstruct it using the 2, 5 10, 50 or more first eigenvectors. According to you, what is the appropriate number of eigenvectors needed to compress the selected digit ?
  3. select a digit from another class. Reconstruct it with the same number of eigenvectors as above. Comment the result.
- Now use all the digit dataset.
  1. Apply the PCA and visualize the obtained eigenvectors. Comment on the obtained results.
  2. Synthetize new digit images by setting different coordinates (in the eigenvector space) for the few first eigenvectors. Comment on (for instance) the acceptable range of values to obtain images that 'look' like (or not) a real digit image.
     According to this experiment, if you would be given an input image for which the reconstruction error[3] would be 0 when using a few eigenvectors, could you conclude that its content is 'similar' to that of the data used for learning the PCA? explain.

3 Eigenfaces - face recognition

We are given a set of labelled images of size 50×50 representing the faces of 10 people under different illumination conditions. Each image is thus represented by a 1D vector of 2500 dimensions. The goal is to recognize the faces by using the K Nearest Neighboor (kNN) algorithm. To avoid computing euclidian distances in the 2500 dimensional spaces, the face images will be first projected into a smaller dimension space using PCA.

(a) Using the Subset1YaleFaces.mat data, compute the eigenvectors (eigenfaces) representation. Display the set of eigenvalues. How many non-zero eigenvalues do you obtain ? why is it so? Display the 'mean' face as well as the first 9 eigenfaces. Then, for different values of M, take a face, project it on the first M eigenvectors and visualize its reconstruction.

(b) if we denote by $\mathbf{y}$ the coefficients of the PCA decomposition of a data point $\mathbf{x}$ using all components, show or explain why:

$$\|\mathbf{x}_2 - \mathbf{x}_1\|^2 = \|\mathbf{y}_2 - \mathbf{y}_1\|^2. \tag{1}$$

[2] To visualize the impact of the k^{th} eigenvector $\mathbf{u}_k$ on data (and interpret it somehow), you can synthetize images as $\bar{\mathbf{x}} + y_k \mathbf{u}_k$ by varying y_k within the range of the eigenvector variability present in the data. Since the variance of y_k within the training data is λ_k, you could take $y_k = t \times \sqrt{\lambda_k}$, with $t \in \{-3, -1.5, 0, 1.5, 3, \}$.

[3] Defined as $\|\mathbf{x} - \tilde{\mathbf{x}}\|$, where $\tilde{\mathbf{x}}$ denotes the reconstructed image.

If we denote by $\mathbf{y}_i^M$ the M first coordinates of $\mathbf{y}_i$ (corresponding to the M first eigenvectors) associated to an input vector $\mathbf{x}_i$, and by $\tilde{\mathbf{x}}_i$ the reconstructed face of this input image $\mathbf{x}_i$ by using these first M components, show that:

$$\|\tilde{\mathbf{x}}_2 - \tilde{\mathbf{x}}_1\|^2 = \|\mathbf{y}_2^M - \mathbf{y}_1^M\|^2 \tag{2}$$

c) Load the data SameFace.mat, that contains different croppings of a given input face. Viusalize the different images and then, for a given M value (e.g. 100 or 150), compute the reconstruction error for each of the cropped images. Comment the results.

d) We are now going to use a classifier to recognize images of people in the database. As classifier, we will use a knn classifier. In Matlab, you can use the function knnclassify. The training data will be the faces from Subset1YaleFaces.mat (cf above) (i.e. this set will be used to build the PCA projection matrix, and also as trianoing data by using the corresponding identity labels). First, using the validation dataset $\mathbf{X}_{\text{valid}}$ Subset2YaleFaces.mat, program and do the following:

1. project all faces from this dataset in order to obtain their PCA representation $\mathbf{y}$.
2. for different values of M and k (you can loop over these values), classify all faces using knn (and the Subset1 dataset) from $\mathbf{X}_{\text{valid}}$, and compute the recognition error.
3. select the values of M and k that give the best performance on this validation set. Then classify the faces from the Subset3YaleFaces.mat dataset, and compute the recognition error. You can check for the errors whether they make sense or not visually.

1) **Spike Sorting.** Many *in vivo* studies involve recording neural action potentials by extracellular electrodes. Action potentials are large enough events that the voltage changes are detectable by an electrode placed many micrometers away from the cell body. For the same reason, an extracellular electrode often records a superposition of action potentials from multiple cells at the same time. Luckily, the waveforms from individual cells tend to have different shapes at the electrode (due to differing distances to the electrode, neuron geometry, etc.), and we can use this fact to cluster waveforms from individual cells and thus identify the source of each action potential.

   The file 'SpikeSortingData.mat' contains a sample dataset from a typical extracellular recording. The variable 'voltageRecording' is the entire voltage sequence recorded during the experiment. The matrix 'waveforms' contains all the action potential events extracted from the recording: each row of the matrix is the recorded waveform of an action potential.

   a. Begin by plotting a few (or all) of the waveforms on the same plot. How many neurons do you think can be seen in this recording?

   b. Calculate the covariance matrix of the waveforms. Do not use the 'cov' command (even though it will give you the same results); instead, subtract the mean of each recorded waveform from every element of that waveform. (Hints: You can use the command 'mean', and compute the mean along the appropriate dimension. And, for subtraction of the mean across all elements of the waveform, recall that 'ones(5,1)*a' gives you a vector of a's.) Next, take the product of this mean-subtracted matrix with its transpose, so that the result is a 70×70 matrix. This product is the covariance matrix, which represents the similarities of the waveforms in the 70 dimensional space corresponding to the 70 time-bins devoted to each recorded waveform.

Next, use the ‘eigs’ command to find the largest 20 eigenvalues and corresponding eigenvectors of this covariance matrix. Make a plot of all 20 eigenvalues: this is called the eigenvalue spectrum.

c. Project each of the waveforms onto the largest 2 eigenvectors (i.e. the eigenvectors corresponding to the 2 largest eigenvalues) of the covariance matrix, which you found in part a, keeping just the coefficients of the projections and not the entire eigenvectors. In other words, compute the coefficients of each waveform along the first two eigenvectors. Now, you have a 2-d (2 coefficient) representation of each 70-dimensional waveform. Plot this 2-d representation of each waveform in a 2-d plot in Matlab, with each waveform represented as a point. What does this plot imply about the number of neurons recorded in the experiment? Which action potentials seem to belong to which neuron?

d. Use the function ‘kmeans’ to cluster the lower-dimension waveforms (for example try ‘c = kmeans(projectedWaveforms,k);’ where k is the number of neurons you think are there by eye, from part c). Now plot separately the waveforms as you did in part a, but only the waveforms for which c=1 (Matlab hint: ‘plot(1:70,waveforms(c==1,:))’). Make separate plots for c=1 to k.

2) **Dimensionality reduction.** Download the file ‘HandwrittenDigits.mat’ and the function ‘plotImage.m’ from the course website. The ‘.mat’ file contains a matrix of images of handwritten numbers. Each row of the matrix is a 28x28 pixel image. The helper function ‘plotImage’ can take a single row of this matrix as an argument and plot it as an image.

a. As in problem 1, subtract the mean of each image, compute the covariance matrix of the mean-subtracted data, and use the ‘eigs’ command to get the largest 200 eigenvalues and eigenvectors of the covariance matrix of the ’images’ data. Plot the eigenvalues. Use the plotImage function to plot the 1st, 2nd, 5th, 50th, and 100th eigenvectors. What is different about the eigenvectors associated with the large eigenvalues and the eigenvectors that go with the smaller eigenvalues?

b. Project the first image (the first row of the images matrix) onto the first eigenvector. Plot the projection (the projected coefficient times the first eigenvector) using plotImage. Next, project the first image (the original image, not the component along the first eigenvector) onto the first 20 eigenvectors. The resulting vector should be the sum $\hat{x} = \sum_{i=1}^{5} (x \cdot v_i) v_i$ where the v_i are the eigenvectors and x is the image.

c. For $k = 1$ to 200, calculate the projection of the first image onto the first k eigenvectors. Calculate the mean squared error (MSE) of each projection with the original image (the mean of the squared differences between the each pixel in the low-dimensional reconstruction and the original images). Plot the MSE by the number of eigenvectors used. Keeping in mind that the original images are 784 dimensional vectors, what does this plot indicate about the data? Explain keeping in mind the content of this 4000 image dataset.