EE-613: - PCA and Probabilistic PCA

Fall 2017

The lab studies the PCA decomposition properties on some visual datasets, and includes as well some example of Probabilistic PCA.

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, NxD 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 (one column per eigenvector, ranked in the same order than eigenvalues)
  %    meanX  : average value of the datas in X (row vector of size D)
  %
  ```

  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     : NxD data matrix (N rows of data zi of dimension D)
  %    meanX : mean of data points provided by MyPCA (row of D dimensions)
  %    P     : DxM projection matrix containing the first M eigenvectors obtained from MyPCA
  % Output:
  %    Y     : NxM 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     : NxM  coordinates in low M dimensional space of the N points to (re)construct
  %    meanX : mean of data points provided by MyPCA (one row of D values)
  %    P     : DxM projection matrix containing the first M eigenvectors obtained from MyPCA
  % Output:
  %    Ztilde     : NxD matrix containing the constructed vectors
  %
  ```

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[1]. Comment on what is captured by each eigenvector (if it makes 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 an image from a digit from another class. Reconstruct it with the same number of eigenvectors as above (using the eigenvectors found in questions 1). Repeat with other images from other digit classes. Comment the result.

- Now use all the digit dataset.
  1. Apply the PCA and visualize the obtained eigenvectors as above. Comment on the obtained results.
  2. Take digits from every class, and see how well we can reconstruct them using from 1 up to 15 eigenvectors. Are there some digits which are easier to reconstruct? What could be the explanation?
  3. 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[2] 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.

- Probabilistic PCA. The main file comprises code to perform probabilistic PCA (showPPCA) (assuming that the MyPCA function is written). The code proposes to generate samples according to the graphical model. Use it with images of only one digit, and see whether the generated samples match your expectation of a sample. Test different subspace dimensions.
  Do it again using data from 2 digits (e.g. 3 and 7). Comment. You can also consider the data from all digits.
  Finally, one advantage of the PPCA is to provide a principled way to learn subspace projections using input data which have missing values. The code allows you to consider that, by corrupting each input image with a square of missing values[3]. Test different corruption levels, and compare the reconstructed images.

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.

[1] 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 $t \in \{-3, -1.5, 0, 1.5, 3\}$.

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

[3] Warning: the corresponding code takes quite some time.

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 (as mentioned in footnote 1).. 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:

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

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. 50, 100 or 150), compute the reconstruction error for each of the cropped images. Comment the results. What can you conclude regarding cropping? Given an input face image, could you suggest a way to select the best cropping?

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 training 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.