EE-613: - PCA and Probabilistic PCA

Fall 2019

The lab studies the PCA decomposition properties on some visual datasets, and includes as well some example of Probabilistic PCA.You can follow the Jupyter notebook to do the exercices. Note that several functions can be found in ../libs/utils.py. You can look at them to understand what they do, and take inspirations of them to program othe functions to address the lab questions.

1 Programming the PCA

In step 1 to 3 of the notebook, you are shown how to load data, and you are asked to program the following functions (see the course).

- ```
  # PCA.
  def my_pca(X):
      """ This function performs the extraction of the PCA components
          given a dataset (X).

  # PCA projection.
  def pca_projection(X, X_mean, UM):
      """ This function projects a matrix of data points X on the
          first M eigenvectors.

  # PCA reconstruction.
  def pca_reconstruction(Y, X_mean, UM):
      """ This function reconstructs data points given their coordinates Y
          in the space spanned by the M eigenvectors of UM.
  ```

  For the first function, you can use for instance (and appropriately) the eig or SVD eigenvalue decomposition function (or the pca function).

- Leave the Probabilistic PCA formula aside at the beginning.

2 Testing the PCA - Digit dataset

- Use the usps digit dataset. Select one digit (avoid one or zero) and extract the image data corresponding to this character.
- **Exercise 1** Test the above programs on the dataset of the selected character. In particular, you can:
  1. Compute the PCA components, and visualise them. Comment on what is captured by each eigenvector (whether you can interpret what you observe, whether it makes sense to you).
  ```

Note that code is provided to show the eigenvectors. However, 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 $t \in \{-3, -1.5, 0, 1.5, 3\}$. To do so, you can for instance update or modify the code given in section 3 to show the eigenvectors.

2. Select a few images from the same digit class, and reconstruct them using the M=2, 5 10, 50 or more first eigenvectors. What do you observe when M is low (2 or 5) and comparing the reconstructed images of the different images? According to you, what is the appropriate number of eigenvectors needed to compress the selected digit (note Mo this number)? If you would have to select 80% of the variance of the original data, which number Mv would you get? Do you see a difference in the set of eigenvalues, and for Mv when considering different digits (7 or 8 for instance)? can you explain the difference?
3. Select an image from a digit from another class. Reconstruct it with the same number Mo of eigenvectors as above (using the eigenvectors found in questions 1). Repeat with other images from other digit classes. Comment the result.

- **Exercise 2.** 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 (manually) different coordinates (in the eigenvector space) for the few first eigenvectors (M<5). 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[1] 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 final part of the notebook contains code to perform probabilistic PCA (assuming that the my_pca function is written). It also 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.

  What do you notice regarding the noise added when changing dimension (e.g. comparing M=5 and M=50)? Explain (see course on what does the noise in the data space correspond to).
  Do it again using data from 2 digits (e.g. 3 and 7). Comment. You can also consider the data from all digits.

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

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.

The notebook pca_faces.ipynb is provided.

(a) Using the set1 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:

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

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 dataset 'same' (9 images) which contains different croppings of a given input face. Viusalize the different images and then, for a given M value (e.g. 5, 10 or 15), 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 (you can use the one from sklearn). To follow a good experimental protocol, we will use the following. The training data will be the faces from set1 (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). Then, we will use as validation dataset $\mathbf{X}_{\text{valid}}$ the set2 of the Yale data. Given this datset, 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 set3 dataset (the test set), and compute the recognition error. You can check for the errors whether they make sense or not visually.