Middle East Technical University | Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Fall 2017-2018

Take Home Exam 3

Due date: **02.01.2018 - 23:59**

1 Objectives

To familiarize yourselves with K-means clustering and Principle Component Analysis through implementing and applying them on a real data set, for image compression problem.

2 Specifications

This homework consists of two parts. In the first part you will implement the K-means clustering algorithm. In the second, you will implement the Principle Component Analysis. In both parts, you will use the "YALE Face Database" which you can find in

`http://vismod.media.mit.edu/vismod/classes/mas622-00/datasets/`

The centered images are provided as a NumPy array in the file "FaceImages.npy" where each image is a 45045 dimensional vector. A python function for plotting an image in vector form is provided in the homework files.

In the first part of this assignment, your aim is to find K "mean" colors so that we can decrease the number of colors used in the images from 256 to K, which will require less bits to represent the same image leading to a lossy compression.

In the second part, your aim is to find K principle components, so that we can express the images in terms of their linear combination which reduces 231-by-195 images into K coefficients as a lossy compression. The original database consists of 165x231x195 numbers. With the new representation we need only (K+2)x231x195 numbers for mean image, standard deviation and eigenfaces that are the principal components of the database and 165xK numbers to represent images.

Homework file is continued with a review of K-means and PCA algorithms.

2.1 K-means Clustering

K-means clustering is a very intuitive and easy to implement algorithm. It consists of an initialization step and an alternating procedure that stops when there is no change as a result of alternations. At the initialization step K random points are chosen to be cluster means. The alternation process is as follows

- Assign data instances to the closest cluster center using the Euclidean Distance given in (1).
- Update each cluster center as the mean of its assigned data instances.

$$d_{euclidean}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

2.2 Principal Component Analysis

Assume that you are given a dataset $D = \{\mathbf{x}^{(i)} \,|\, i \in [1..m]\}$, in which each $\mathbf{x}^{(i)}$ is an n-dimensional multivariate vector. Before starting the Principal Component Analysis, one should zero center and scale the dataset according to (2).

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j} \tag{2}$$

in which the subscript corresponds to j^{th} feature, superscript corresponds to i^{th} data example, μ and σ are the mean value and the standard deviation.

Principal components are the K set of uncorrelated variables learned from correlated ones representing the data, which have the largest variance. These correspond to the eigenvectors of the covariance matrix of the data related with the K largest eigenvalues. The covariance matrix is given in (3)

$$\Sigma = \frac{1}{m} \sum_{1}^{m} (x^{(i)})(x^{(i)})^T = XX^T \tag{3}$$

where Σ is the covariance matrix, X is data matrix with size nxm.

Having found the principal components, any specific data example $x^{(ex)}$ can be represented as a linear combination of them. The coefficient vector $c^{(ex)}$ is given by (4)

$$c^{(ex)} = P^T x^{(ex)} \tag{4}$$

where P is an n-by-K matrix whose columns are the vectors corresponding to principal components.

The covariance matrix Σ is nxn dimensional which is not practical while working with high dimensional data like images. Instead of computing eigenvectors of Σ directly, we can compute first m principal components (corresponding to m largest eigenvalues) from the correlation between data examples. While working with Σ gives the dependence of pixels on each other, working with correlation between data examples gives the dependence of data examples on each other. Starting from that information, we can construct the correlation between pixels. For detailed explanation see (5). Note that this approach results in loss of eigenvectors that comes after the first m ones.

$$\begin{gathered}
\text{Let } U = \{\mathbf{u}_{(i)} \,|\, i \in [1..m]\} \text{ be the first } m \text{ eigenvectors of } \Sigma \\
\text{Let } V = \{\mathbf{v}_{(i)} \,|\, i \in [1..m]\} \text{ be the first } m \text{ eigenvectors of } m\text{x}m \text{ matrix } X^T X \\
X^T X v_i = \lambda_i v_i \\
X^T X v_i = \lambda_i v_i \Rightarrow XX^T X v_i = \lambda_i X v_i \Rightarrow \Sigma X v_i = \lambda_i X v_i \Rightarrow \Sigma u_i = \lambda_i u_i \\
u_i = X v_i
\end{gathered} \tag{5}$$

2.3 Programming and Interpretation Tasks

First, you need to input the complete dataset. As in the previous homeworks, you must use vectorized implementations.

For the first part of the homework, you should write two Python **functions**. The first function called **findClusterCenters** will take all the images in the database (as a data design matrix) and the number of clusters (K) as input and run the K-means algorithm on all of the pixels as data instances. To be more specific, you will use all the pixels of all the images in the dataset to compute K cluster centers. The function will output these cluster centers. The second one called **kmeansCompress** will take the cluster centers and an image to compress as input and assign each pixel of the image to index of the closest center using Euclidean Distance. Having done that, the image is compressed from 231x195x8 bits to 231x195x($log_2 K$) bits. This function should also reconstruct the image such that each pixel value is replaced with the assigned cluster center. The function returns the compressed and reconstructed images. Here, you are not expected to deal with bit operations, return only the image with new colors as 8 bit unsigned integers.

For the second part of the homework, you should also write two Python **functions**. The first one called **findPrincipalComponents** takes all the images and the number of principal components (K) as input, computes and returns the first K principal components of the dataset as described in the section 2.2. The second one called **pcaCompress** takes the K principal components and an image to compress, and computes the coefficients to represent the image in terms of the principal components. It then reconstructs the image using these and return the coefficients and reconstructed image.

Having implemented the two methods, run the algorithms for K=16,32,64 and test using the first two images provided in **"FaceImages.npy"**. You should prepare a report discussing the methods. The report should contain at least the followings;

- The cluster means as a figure (see Figure 1) for the three different K values in which each color is represented as 8x8 squares, and a **discussion on why and how** these colors are the results.
- The first 16 principle components as 231x195 images (see Figure 1) and the **discussion of their appearances, meanings about the database**.
- Comparison of the original and reconstructed images (using a reconstruction error metric may ease your discussion).
- Comparison of the algorithms in terms of reconstruction and compression, and detailed discussion explaining the reasons behind the results.
- Discussion on the effects of the value of K
- A recommendation for choosing best K
- Possible problems that can be encountered using these algorithms.

3 Restrictions and Tips

- Do not use any available Python repository files without referring to them in your report.
- Toolbox function use is not allowed in this homework. Do not use any ML-related toolbox. However, you may cross-check your results utilizing toolboxes. Your homework submission must not include any high-level toolbox function.

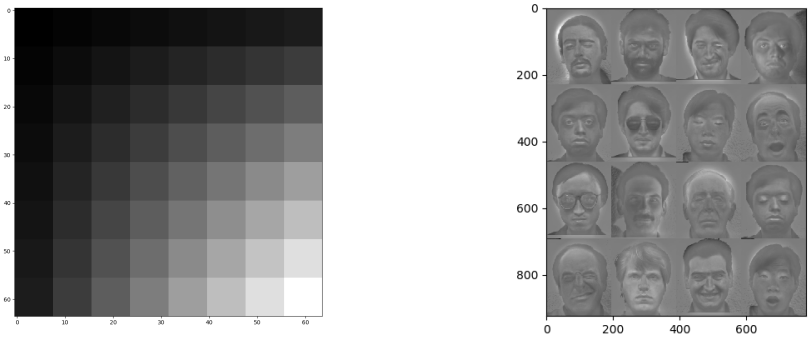

Figure 1: Example figures for representing cluster means and principal components (eigenfaces). Note that your results will be different.

- If you encounter trouble regarding vectorization, first try to implement the tasks by using loops. Vectorization is required, since typical ML projects deal with massive amounts of data. If at the end you fail to vectorize your code, submit the current version. Although vectorization appears to be essential in ML programming, since this is not a Python course, unvectorized versions will only result in a minor decline (at most 10%) in the grade you are going to receive.
- Implementation should be of your own. Readily-used codes should not exceed a reasonable threshold within your total work. In fact you shouldn't need any code on repositories.
- Don't forget that the code you are going to submit will also be subject to manual inspection.

4 Submission

- **Late Submission:** As in the syllabus.
- Implement the task in a file named **the3.py**. The implementation together with a 3-to-4 pages long report focusing on theoretical and practical aspects you observed regarding this task should be uploaded on COW before the specified deadline as a compressed archive file whose name should be <student_id>_the3.tar.gz, e.g. 1234567_the3.tar.gz.
- The archive must contain **no directories** on top of implementation files.
- Do not include the face database in the archive.

5 Regulations

1. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
2. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.