ΕΠ08 Pattern Recognition – Machine Learning

2nd Assignment

This project is **individual** and consists of two questions. It is highly recommended that you spend time to understand the fundamental logic behind the questions of this project and that you avoid searching for solutions on the internet. However, if you read and / or use any material and / or source code that is available on the internet, you have to cite the source and / or the link to the website from which you obtained information correctly. In all cases, copying part of the project or the whole project is unacceptable and in case of suspecting copying, all involved parties will fail the course.

You must submit **only one IPython notebook file (Jupyter notebook)** named: ***Surname_IdentificationNumber.ipynb***

You can use caption cells to further organize your document.

Important: You must make sure that the IPython notebook that you submitted can open and run in the google colab environment.

[Question 1: Face Recognition] [70 points]

In this question you will apply the Eigenfaces method (i.e. combination of PCA for feature extraction and nearest neighbor classifier for face recognition). You will use pictures of faces from the Yale B face database in which there are 10 faces photographed under 64 different lighting conditions. During your implementation, you will evaluate the ability of the Eigenfaces algorithm to handle lighting conditions of the test set images, which will differ from those in the training set images.

The data is available in the file faces.zip in the Documents directory in eclass.

The Eigenfaces method for face recognition includes 3 basic steps:

Step 1: Each image of dimension 50 x 50 pixels, is converted to vector of 2500 elements and saved as a column in the training data matrix X. Then we apply Principal Component Analysis (PCA) to the training data matrix and extract the d principal components. The d eigenvectors, when converted and displayed as images, are called Eigenfaces.

Step 2: We project the training and test set images into d-dimensional space and thus extract low-dimensional features (d-dimensional features). The low-dimensional / d-dimensional space is called eigenspace.

Step 3: Face recognition is done in the eigenspace using a nearest neighbor classifier of with the Euclidean distance as a metric.

You are going to use the following subsets of the Yale B face dataset:

Set_1: person*_01.png to person*_07.png (i.e. the first 7 images of each face)
Set_2: person*_08.png to person*_19.png
Set_3: person*_20.png to person*_31.png
Set_4: person*_32.png to person*_45.png
Set_5: person*_46.png to person*_64.png

Questions:

I. Write a loadImages(path, set_number) function that takes the path of the images folder as an input, e.g. loadImages("C:/Images", "Set_1"), reads the images and returns a data matrix depending on the set_number, where each image is represented as a column vector. The function also returns the labels of the categories in which the different images belong, encoded with integers (e.g. 0 for images that belong to person_0, 1 for images that belong to person_1 etc.).

II. Train the Eigenfaces method with d = 9 and d = 30 using all images in Set_1 (70 images) and recognize the faces in sets Set_1 to Set_5. Write down the classification accuracy for each Set and each value d. We expect 100% classification accuracy in Set_1, as it was ised for training the Eigenfaces method. Comment on the generalization ability of the method in different Sets.

III. Display (in image form) the 9 top eigenvectors that were obtained after training the Eigenfaces method in Set_1. What do you notice? What do the different eigenvectors express?

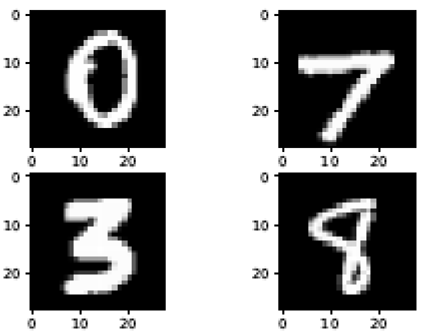
IV. Use d = 9 and d = 30 Eigenfaces obtained from Set_1, to reconstruct a random image from each of the 5 Sets. Display both the original and reconstructed images for different values of d. Comment on the reconstruction quality of each image.

V. Display the top 9 singular vectors obtained after applying SVD to the data matrix of Set_1. Are the singular vectors different than the corresponding eigenvectors. If so, why?

Note: You can either use already implemented versions of PCA or implement it yourself using eigenvector analysis (functions of eig form) in the covariance matrix. It is recommended that you pre-process each image by subtracting its mean value and dividing with the standard deviation of its values.

[Question 2: Image classification using SVMs] [30 points]

In this question, you are asked to look into the performance of Support Vector Machines in recognizing handwritten digits. For this purpose you are going to use the MNIST dataset and algorithm implementations of the scikit-learn library. The MNIST dataset is consisted of 70000 images of handwritten digits and is typically divided into three subsets: training set (50000 images), validation set (10000 images) and test set (10000 images). Each image has dimensions 28 x 28 pixels and depicts a handwritten digit. Examples of such images are displayed in the following figure:



- You are asked to load the data from the MNIST dataset and to convert each image to a vector of size 28 x 28 = 784. After that, normalize the data in the range [0, 1].
- There are various options that may affect the performance of SVMs in classification problems. Examples of such parameters are the kernel type and values of various (hyper) parameters. You are asked to examine the performance of SVMs for linear and RBF kernels and different hyperparameter values in order to determine the parameter / kernel combination that leads to the highest classification accuracy. For this experiment, use the 60000 images for training and the 10000 examples for testing. Write down the parameter values, i.e. kernel type, values of C and gamma that lead to the best performance in both the training and the test set.
- Then, apply PCA to the data choosing 3 different values for the explained variance and execute the SVM method for each covariance value using the parameters that lead to the best performance in the previous question. For each execution, write down the number of components kept, as well as the classification accuracy. Moreover, write down the execution time of each experiment and make conclusions about the trade-off between classification accuracy, dimensionality reduction and execution time of the algorithm.