PCA and Eigen faces

Saman Paidarnia

Faculty of Computer Science

Research group of Visualization and Data Analysis

University of Vienna

The project has been presented in the: https://github.com/saman-nia/PCA-Eigenfaces

Abstract

This assignment focused on the method presented in the paper by M. Turk and A. I use Eigen faces for recognize the face. I employee kNN for the performance evaluation. I did the assignment in the Jupyter Notebook along the results. I have also took the face data sets which has been provided by the University of Vienna.

Datasets

Three different images of each of thirty four distinct subjects. The images were taken at different times, varying the lighting, facial expressions smiling / not smiling.

I created the Convert2PGM.ipynb for converting image from .GIF to .PGM. I have also defined a class with name of class_faceRecog.py for build the Eigen face. I called the class via the PCA_&_Eigenfaces.ipynb.

All the images took in a dark background with the subjects in an upright, frontal position. For first part I only used only one image per subject should be used. Workflow:

a) Load images.

b) Compute and display the mean face.

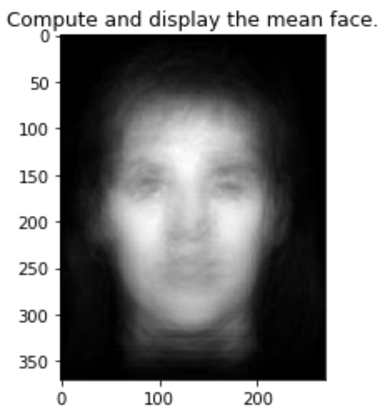


Figure 1: mean of the all faces.

c) Compute the number of principal components we need to capture 80% of face variance which was 15.

d) Compute the normalized images.

e) Compute the eigenvalue and eigenvector.

f) Visualize the first k = 16 Eigen faces in a 4 * 4 grid.

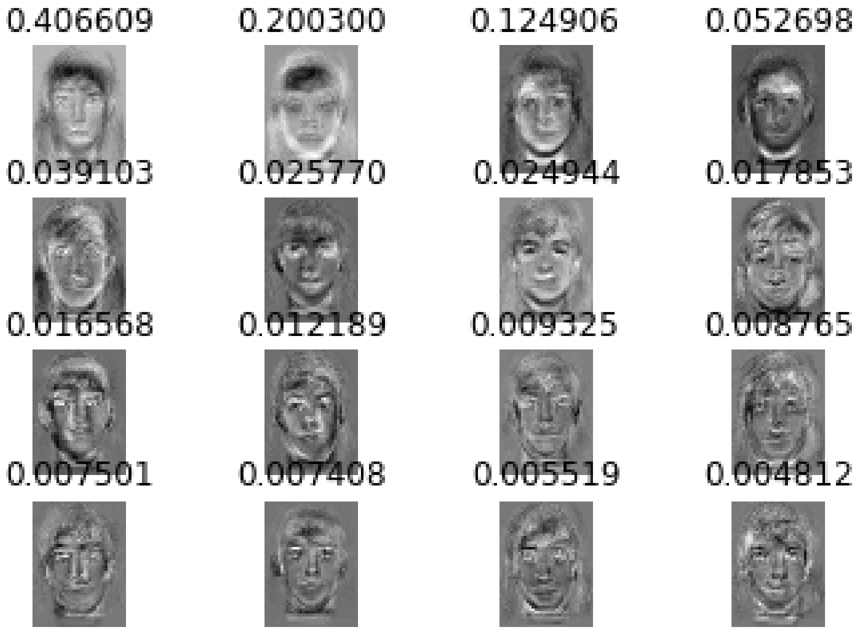


Figure 2: reconstruct each subject's face.

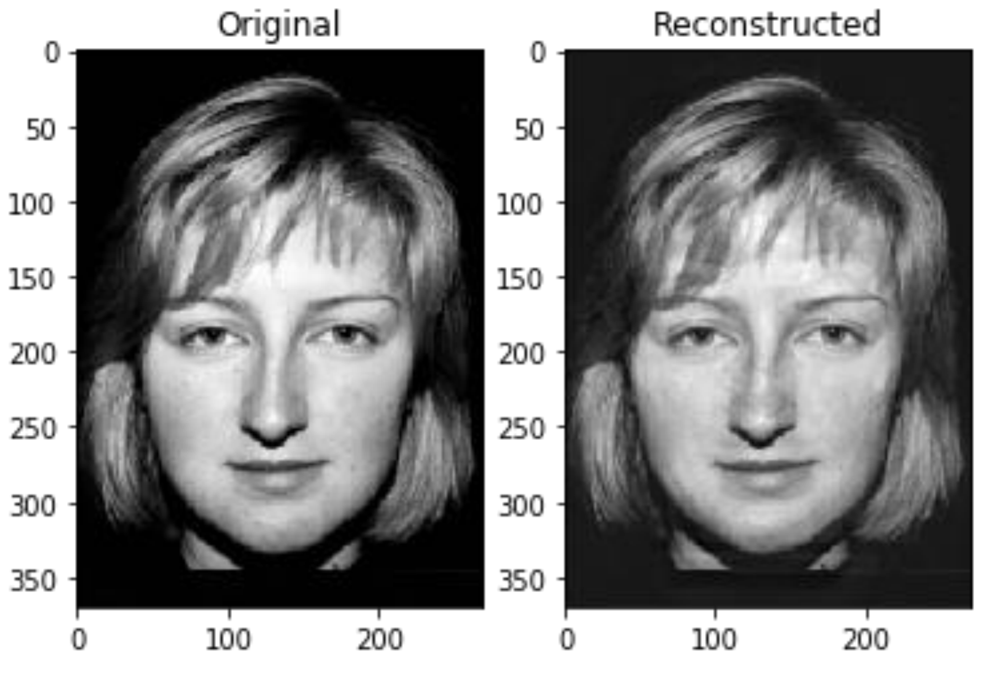


Figure 3: reconstruct subject of 10 with 80% variance and 5 nearest neighbor.

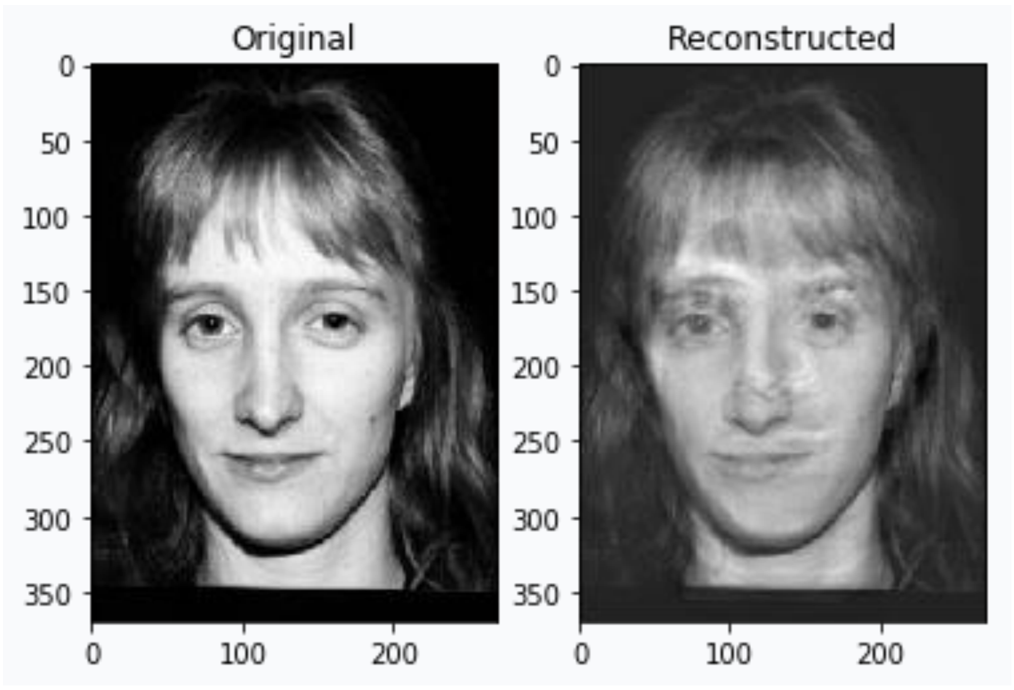


Figure 4: reconstruct subject of 10 with 80% variance and 5 nearest neighbor.

K-Nearest Neighbors

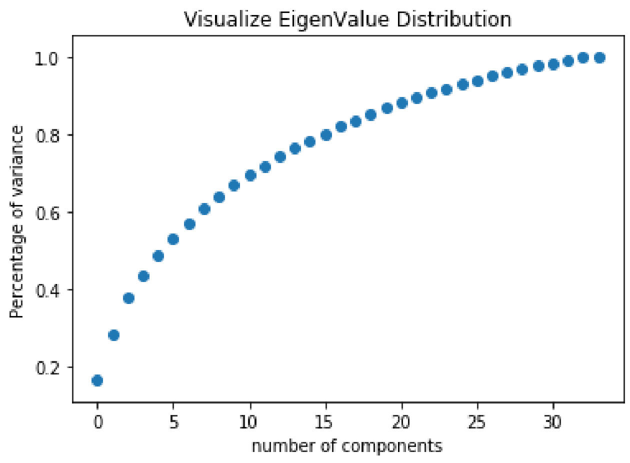
a) kNN is one of the simplest of classification algorithms available for supervised learning. The idea is to search for closest match of the test data in feature space. We will look into it with below image. In the image, there are few families. We call each family as Class.

kNN in OpenCV

b) I label the training family as Class-1 and Test family as Class-2

c) We can specify how many neighbors we want. It returns:

   a. The label given to test data depending upon the kNN theory we saw earlier. If you want Nearest Neighbors algorithm, specify a range of K from 1 to 10 where k is the number of neighbors.
   b. The labels of k-Nearest Neighbors.
   c. Corresponding distances from Test data to each nearest neighbor.



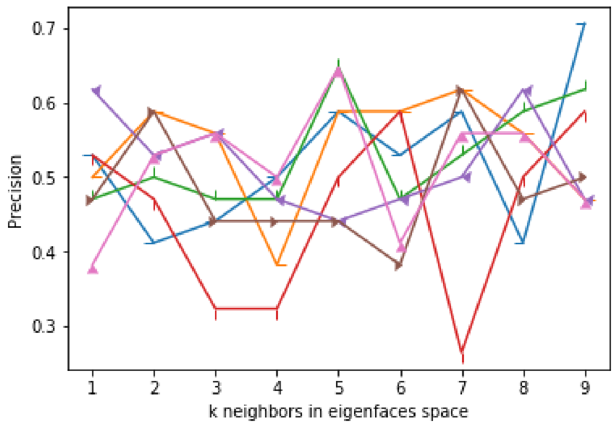


Figure 5: accuracy model

Unfortunately I could not do the accuracy well because I had a problem in "giving a class to per subject". My code gave to all picture in any variances same class and that's why the model would be always 100% correct. I multiplied a random number to "responses" in model accuracy to produce a variable visualization but it would not works well. Because of restriction on the deadline time, I have uploaded it, however I will try to solve it soon.

The Eigen faces is an approaches to represent an image, with the top k component Eigen faces represent as much variance as possible.

I am trying to clarify some concepts for face recognition. According to my understanding, given a training set of images with each image measuring all have a same size, I have also made a matrix of training images.

I used PCA, we would be reducing the high dimensions.

PCA does dimensional reduction by expressing D dimensional vectors on an M dimensional subspace, with $M<D$. The vector itself can be written as a linear combination of M eigenvectors, where the eigenvector is itself a unit vector that lives in the D dimensional space.

Consider, for example, a two dimensional space which I reduce to one dimension using PCA. I find that the principal eigenvector is the unit vector that points equally in the positive *x^* and *y^ direction, i.e.*

v^=12–√(x^+y^)

In general, we express a *D* dimensional vector, *x,* as a reduced *M* dimensional vector *a*, where each component *ai* of a is given by,

ai=∑jxjVij

The accuracy would depend on the classifier you are using once you have the data in the PCA projected space. In the original Turk/Pentland Eigen face paper. They just use kNN / Euclidean distance but a modern implementation might use SVMs.