Face Recognition by using eigenfaces, SVD, and PCA

Leonid Dashko

Face detection

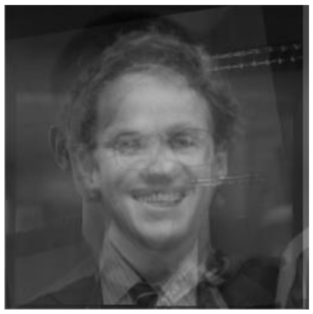
I used already pre-trained classifiers for detecting faces (file is "*haarcascade_frontalface.xml*") based on Haar Cascades. The Haar feature-based cascade classifiers created by Paul Viola and Michael Jones to detect different types of objects that rely on edge and specific features in the images (e.g. each face contains at least by 3 features: nose, mouth, eyes). [1]

Also, the Haar classifier is based on assumption that regions with eyes and mouth are darker than cheeks and forehead. [2]

Face recognition

1) ***Transform the image into a vector.*** For example, we have an image 100x100 pixels. As a result, we will have a vector of size 1x10,000 (flatten image).
2) ***Compute SVD***

- Read all pre-processed train images and flatten them (they must have the same size)
- Compute the mean face. The example of the computed mean face is below (~5 images in the face DB):



- Subtract the mean face from each image before performing SVD and PCA
- Compute the SVD for the matrix from the previous step. As a result, we got

$$U, S, Vt = SVD(A)$$

Where the matrix "U" represents eigenfaces.

To reconstruct our initial matrix "A" we multiply "U", "S", "Vt":

$$A' = U \times S \times V^T$$

On this step, we can reduce the dimensionality of the initial matrix "A" by sorting only the most important features and slicing matrices "U", "S", "Vt" accordingly, then multiplied all these 3 matrices will give an approximation of the initial matrix of images but with reduced dimensionality.

3) ***Project onto PCA space***

In order to select the most similar face to the input face, we will need to project the received features onto the PCA space where each feature will be in a new dimension, so the number of features = the number of dimensions.

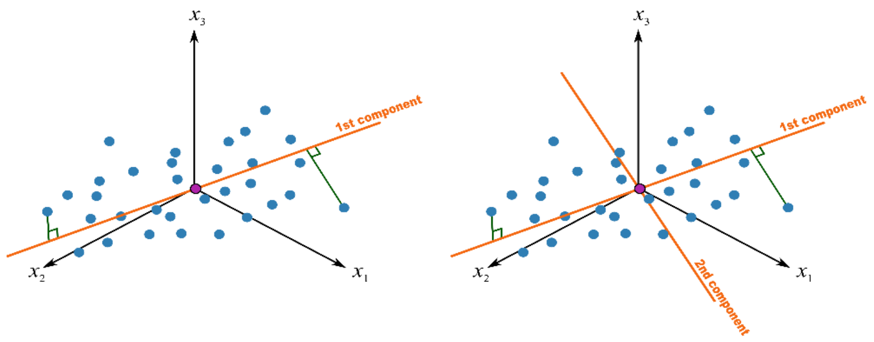


Image source – [3].

To do the PCA projections, we need to obtain weights of these features for each dimension by multiplying the reconstructed array of images A' with subtracted mean face by eigenfaces.

4) Project test image onto PCA space and find the most similar face in the face database

We need to grayscale the target image, flatten it (convert to vector), and project onto PCA space by multiplying the obtained eigenfaces from step #3 by target image (flatten and grayscale).

To find the identical face we compute the Euclidian distance between feature weights of test image and weights of all other images obtained in step #3. This image has the most

Notice: part of the code was taken from the source [4] and extended.

<u>References</u>

1. Face Detection using Haar Cascades
   http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html
2. Mastering OpenCV with Practical Computer Vision Projects (Chapter 8) by D.Baggio, S.Emami, D.Escrivá, K.Ievgen, N.Mahmood, J.Saragih, R.Shilkrot
3. Geometric explanation of PCA
   https://learnche.org/pid/latent-variable-modelling/principal-component-analysis/geometric-explanation-of-pca
4. Eigenfaces and Forms
   https://wellecks.wordpress.com/tag/eigenfaces/