CIS4930/5930 Project 1: PCA for Analyzing Huamn Faces

Due: June 19, 11:59pm

Your submission will be an experimental report, you need to visualize the results in good figures (your grade will be based on the quality of results and analysis), such as mean, eigen-vectors, and plots. Print out the result for submission. Don't print the code.

1 Objectives

Human faces are highly structured visual patterns and have been a major focus of research over the past 30 years across fields such as computer vision, computer graphics, and human-computer interaction. Applications include face recognition, identity verification, expression analysis, and facial animation. A critical step in these tasks is extracting effective representations from face images. Principal Component Analysis (PCA) has proven to be a powerful method for facial representation.

Since PCA is applied to the intensity values of aligned faces, it is important to first align the landmarks. This alignment process-whether geometric or temporal-is also commonly used in other biological data, such as speech signals, where the rate of utterance may vary over time. This project will explore and compare two dimensionality reduction techniques: PCA, a generative method, and Fisher Linear Discriminant (FLD), a discriminative method.

Dataset We provide a dataset of 1,000 faces (./images) and each has 128 x 128 pixels. The face images are pre-processed so that the background and hair are removed. Each face has a number of landmarks, 68 points per image, which are identified by OpenFace (./landmarks). These landmarks should be aligned so that the appearance (i.e. grey level, intensity on pixels) can be compared meaningfully across face. Such alignment (geometry for image or time warping for speech) is quite common in other patterns/data. To study the discriminative method, we divide the dataset into two categories: male (./male_images) and female (./female_images), with the corresponding landmarks (./male_landmarks and ./female_landmarks).

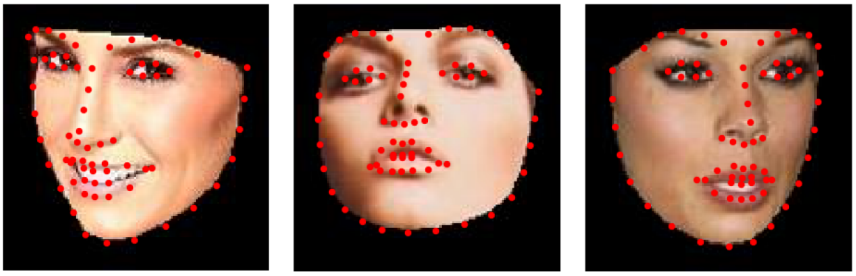

Figure 1: Example faces with 68 landmarks from CelebA. The data set contains 1000 images from CelebA, and they are cropped to 128 x 128 pixels by the OpenFace. These faces have different colors, illuminations, identities, viewing angles, shapes, and expressions.

2 PCA for face reconstruction and synthesis.

Divide the 1000 faces into two parts: the first 800 faces form a training set, and the remaining 200 faces form the test set. In the following steps, you always use the training set to calculate the eigen-vectors and eigen-values, and calculate the reconstruction errors for the test images using the eigen-vectors computed from the training set.

1. Compute the mean and first $K = 50$ eigen-faces for the training images with no landmark alignment, and use them to reconstruct the remaining 200 test faces. i) Display the first 10 eigen-faces; ii) Plot 10 reconstructed faces and the corresponding original faces; and iii) Plot the total reconstruction error (squared intensity difference between the reconstructed images and their original ones) per pixel (i.e. normalize

the error by the pixel number, and average over the testing images) over the number of eigen-faces $K = 1, 5, 10, 15, ..., 50$. Note that, PCA is usually applied on the intensity or gray images. For the color images, you can first transform the color image from the RGB color model to the HSV (hue, saturation, value) model. Then apply PCA on the V channel. In python, you can use skimage.color.rgb2hsv and skimage.color.hsv2rgb to transform the images between RGB and HSV.

2. Compute the mean and first $K = 50$ eigen-warping of the landmarks for the training faces. Here warping means displacement of points on the face images. i) Display the first 10 eigenwarpings (you need to add the mean to make it meaningful), and use them to reconstruct the landmarks for the test faces. ii) Plot the reconstruction error (in terms of distance) over the number of eigen-warpings $K = 1, 5, 10, 15, ..., 50$ (again, the error is averaged over all the testing images).

3. Combine the two steps above. Our objective is to reconstruct images based on top 10 eigen-vectors for the warping and then top K (say 50) eigen-vectors for the appearance. For the training images, we first align the images by warping their landmarks into the mean position, and then compute the eigen-faces (appearance) from these aligned images. For each testing face: i) project its landmarks to the top 10 eigen-warpings, you get the reconstructed landmarks. (here you lose a bit of geometric precision of reconstruction). ii) Warp the face image to the mean position and then project to the top k (say K=50) eigen-faces, you get the reconstructed image at mean position (here you further lose a bit of appearance accuracy). iii) Warp the reconstructed faces in step ii) to the positions reconstructed in step i). Note that this new image is constructed from 60 numbers. Then compare the reconstructed faces against the original testing images (here you have loss in both geometry and appearance). Plot 20 reconstructed faces and their corresponding original faces. Plot the reconstruction errors per pixel against the number of eigen-faces $K = 1, 5, 10, 15, ..., 50$. (mywarper.py provides a function warp(image, original_landmark, warped_landmark) for you to use.)

4. Synthesize random faces by random sampling of the landmarks (based on the top 10 eigenvalues and eigenvectors in the wrapping analysis) and a random sampling of the appearance (based on the top 50 eigenvalues and eigen-vectors in the intensity analysis). Display 50 synthesized face images. As we discussed in class, each axis (i.e. the i-th eigen-vector) has its own unit, that is $\sqrt{\sigma_i}$, the square-root of the i-th eigenvalue. Specifically, the eigen-faces and eigen-warpings for the landmarks can be seen as the basis functions for the generated faces, while we need to sample the coefficients (latent vector) at each eigen-axis.

3 Fisher Linear Discriminant (FLD) for gender discrimination

We have divided the 1000 faces into male (412) and female faces (588). Then you can randomly choose 800 faces as the training set and the remaining 200 faces as testing set.

1. Find the Fisher face that distinguishes male from female using the training sets, and then test it on the 200 testing faces and report the error rate. This Fisher face mixes both geometry and appearance difference between male and female.

2. Compute the Fisher face for the key point (geometric shape) and Fisher face for the appearance (after aligning them to the mean position) respectively. Project all the faces to the 2D-feature space learned by the fisher-faces, and visualize how separable these points are.

[The within-class scatter matrix is very high dimensional, you may compute the Fisher faces over the reduced dimensions in steps (2) and (3) in section 2: i.e. each face is now reduced to 10 dimensional geometric vector + 50 dimensional appearance vector. After the Fisher linear Discriminant analysis, we represent each face by as few as 2 dimensions for discriminative purpose and yet, it can tell apart male from female!]