Machine Learning

Lab 2 - Principal Component Analysis (PCA)
Spring 2025

Introduction

In this lab you'll work on visualizing data and reducing its dimensionality.

You may not use any functions from machine learning library in your code, however you may use statistical functions. For example, if available you **MAY NOT** use functions like

- pca
- entropy
- zscore

however you **MAY** use basic statistical functions like:

- std
- mean
- cov
- svd
- eig

Your task will be to write a *single script* such so that we can run it in the command line and it output (and/or displays) the requested information/figures.

Grading

- +1pt Can run script properly.
- +2pts Parses dataset correctly.
- +2pts Some results for Part 1.
- +2pts Correct results for Part 1.
- +1pt Displays some image for Part 2.
- +2pts Generates correct image for Part 2.

DataSets

Yale Faces Datasaet This dataset consists of 154 images (each of which is 243x320 pixels) taken from 14 people at 11 different viewing conditions (for our purposes, the first person was removed from the official dataset so person ID=2 is the first person).

The filename of each images encode class information:

$$\text{subject} < ID > . < condition >$$

Data obtained from: http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

1 Dimensionality Reduction via PCA

Download and extract the dataset *yalefaces.zip* from Blackboard. This dataset has 154 images ($N = 154$) each of which is a 243x320 image ($D = 77760$). In order to process this data your script will need to:

1. Read in the list of files
2. Create a 154x1600 data matrix such that for each image file
   (a) Read in the image as a 2D array (243x320 pixels)
   (b) Subsample/resize the image to become a 40x40 pixel image (for processing speed). I suggest you use your image processing library to do this for you (like pillow's *resize* method).
   (c) *Flatten* the image to a 1D array (1x1600)
   (d) Concatenate this as a row of your data matrix.

Refer back to Lab1 for code to do this.

Once you have your data matrix, your script should:

1. Reduce the data to 2D using PCA (using the two most relevant eigenvectors). You may use *eig*, *eigh*, or *svd* to do this, but make sure to read the function's documentation for usage. In addition, if necessary, cast imaginary numbers to real.
2. Plot the data as points in 2D space for visualization

Your graph should end up looking similar to Figure 1 (although it may be rotated differently, depending how you ordered things and/or your statistical library).

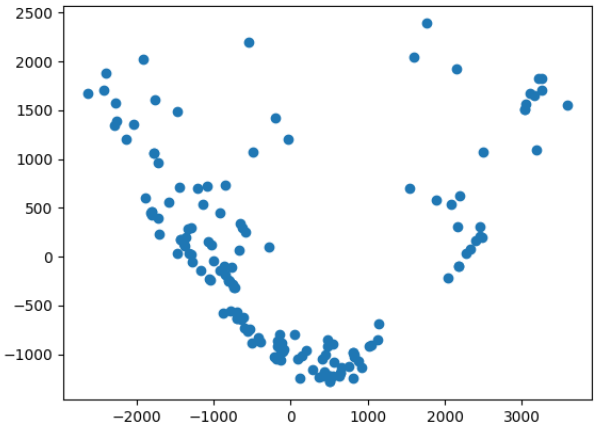

Figure 1: 2D PCA Projection of data

NOTE: Depending on your linear algebra package, the eigenvectors may have the opposite direction. This is fine since technically an eigenvector multiplied by any scalar are equivalent.

2 Eigenfaces

One (cool?) application of PCA is lossy-compression. We can project our data down to k dimenions (where $k < D$), then when we need to reconstruct our data, we can just multiply the k dimensional data by (the transpose of) its k associated principle components in order to return to our original feature space!

Write a script that:

1. Takes your original 154×1600 data matrix from the previous problem and performs PCA on the data to extract its eigenvectors and eigenvalues (again, although you may not use any package ML functions like *pca*, you **may** use statistical functions like *svd, eig*).

2. Determines the smallest k such that the k largest eigenvalues constitute at least 95% of the eigenvalues.

3. Projects *subject02.centerlight* onto the k most important principle components, resulting in a feature vector of length k. **NOTE:** Different OS's parse the directories in different ways. There's no guarantee that this image is the first one.

4. Reconstructs this person, again using the k most significant principle components (your feature vector should now once again be 1×1600).

5. Un-does any standardization that you may have done to get the eigenvectors (zero-meaning, zscoring, etc..).

6. Reshapes this feature vector to be a 40×40 image/matrix.

7. Displays the original and reconstructed image, and outputs to the command line the values of k.