Principal Component Analysis

MAE301 Applied Experimental Statistics

Max Yi Ren

Department of Mechanical Engineering
Arizona State University

October 15, 2015

Outline

high-dimensional data

(Eigenface) Human faces are different but have commonalities. Consider each pixel of a gray-scale face image as a variable, a face can be represented as a data point in a high-dimensional space.

Figure: Sample faces from the modified Olivetti faces dataset

learning a good representation?

Can we extract from high-dimensional data a lower-dimensional but "meaningful" representation?

One "meaningful" representation is a set of directions that "explain" the variances among data points. These directions are called **Principal Components** (PCs). The first PC explains the largest variance, and the second follows. We will show that all PCs are orthogonal.

principal components of faces

A list of PCs for the modified Olivetti faces dataset.

Figure: The first 16 PCs of the modified Olivetti faces dataset

using PCs for face compression

Figure: Dimension reduction: Reconstruction of one face using 5 to 100 top principal components. The first figure is the mean face.

data preparation

The m-by-n data set $\mathbf{X}$ has m observations, $\mathbf{x}_i$ for $i = 1, \cdots, m$, each with n variables. We assume that the data is normalized to have zero mean in each variable, i.e., $\sum_{i=1}^{m} X_{ij} = 0$ for all $j = 1, \cdots, n$. In a matrix form, we have $\mathbf{X}^T \mathbf{1} = \mathbf{0}$.

Why do we need to have zero mean? What will happen to the principal components when the mean is large? What will happen if we scale values of one dimension up or down, e.g., from "inch" to "feet"?

derivation of a principal component (1/3)

Consider an n-by-1 unit vector $\mathbf{v}$. The projection of a data point $\mathbf{x}_i$ to $\mathbf{v}$ is $\mathbf{x}_i^T\mathbf{v}$ (why?).

The variance (under large sample) of all projections $\mathbf{x}_i^T\mathbf{v}$ for $i = 1, \cdots, m$ can be calculated as

$$\sigma^2 \approx s^2 \approx \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i^T\mathbf{v} - \mu)^2, \tag{1}$$

where

$$\mu = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i^T\mathbf{v} \tag{2}$$

is the mean of all projections (what is the mean?).

derivation of a principal component (2/3)

We can also rewrite the variance in a matrix form:

$$\sigma^2 = \frac{1}{m}\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} + \mu^2 - \frac{2}{m}\mu\mathbf{1}^T\mathbf{X}\mathbf{v}, \tag{3}$$

and

$$\mu = \frac{1}{m}\mathbf{1}^T\mathbf{X}\mathbf{v}. \tag{4}$$

Notice that $\mathbf{1}^T\mathbf{X} = \mathbf{0}^T$, we have

$$\sigma^2 = \frac{1}{m}\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v}. \tag{5}$$

derivation of a principal component (3/3)

Recall that the 1st principal component ($\mathbf{v}_1$) is a unit vector that maximizes the variance, i.e.,

$$\mathbf{v}_1 = arg \max_{\mathbf{v} \text{ such that } ||\mathbf{v}||=1} \mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}. \tag{6}$$

It turns out that the solution to this optimization problem satisfies (the KKT conditions):

$$\mathbf{X}^T \mathbf{X} \mathbf{v} = \lambda \mathbf{v}, \tag{7}$$

where λ is an unknown scalar.

interpretation of principal components (1/2)

According to the equation

$$\mathbf{X}^T\mathbf{X}\mathbf{v} = \lambda\mathbf{v}, \tag{8}$$

all principal components are eigenvectors of the matrix $\mathbf{X}^T\mathbf{X}$ (the covariance matrix). Since $\mathbf{X}^T\mathbf{X}$ is symmetric and positive semi-definite (why?), the principal components are all orthogonal (why?).

interpretation of principal components (2/2)

But which eigenvector is the 1st principal component? To answer this, let's put the solution

$$\mathbf{X}^T\mathbf{X}\mathbf{v} = \lambda\mathbf{v} \tag{9}$$

back into the objective of maximizing the variance of projections:

$$\sigma^2 = \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} \tag{10}$$

to get

$$\sigma^2 = \lambda\mathbf{v}^T\mathbf{v} = \lambda. \tag{11}$$

Since the 1st principal component represents the direction where the largest variance occurs, it is the eigenvector of $\mathbf{X}^T\mathbf{X}$ associated with the largest eigenvalue.

things learned so far

- PCA is about finding principal components, i.e., directions where largest variances in the data occurs.
- The principal components are the eigenvectors of the covariance matrix, and the data variances in these directions are captured by the eigenvalues.

singular value decomposition (SVD)

Consider the SVD of the data $\mathbf{X}$ to be $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary and $\mathbf{\Sigma}$ is diagonal. Then we have

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T. \tag{12}$$

In parallel, we know that the eigenvalue decomposition of the covariance matrix is

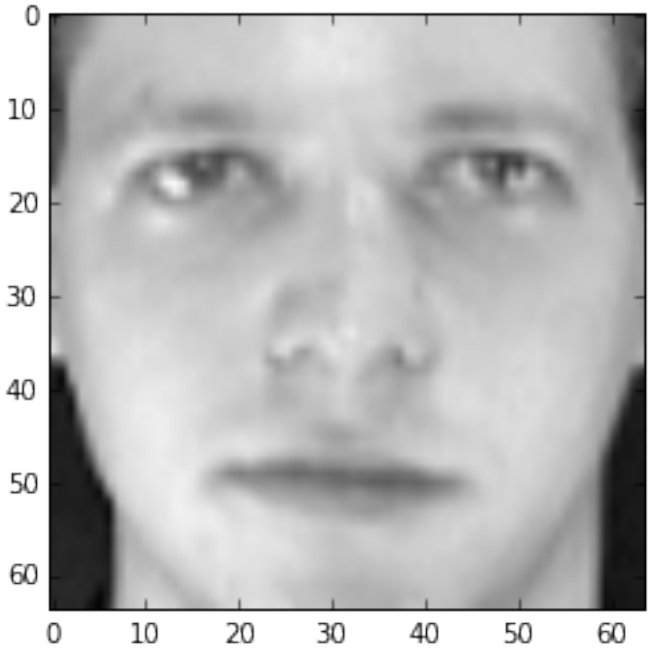
$$\mathbf{X}^T\mathbf{X} = \bar{\mathbf{V}}\mathbf{\Lambda}\bar{\mathbf{V}}^T. \tag{13}$$

So we have $\mathbf{V} = \bar{\mathbf{V}}$ and $\mathbf{\Sigma}^2 = \mathbf{\Lambda}$, i.e., the eigenvalues and eigenvectors of the covariance matrix can be derived by SVD of the data matrix.

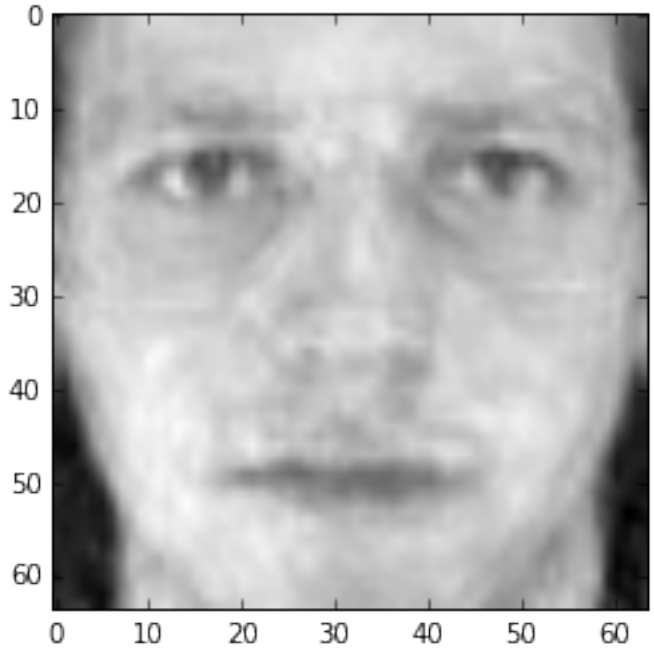
Eigenface reconstruction

Figure: Dimension reduction: Reconstruction of one face using 5 to 100 top principal components. The first figure is the mean face.

Eigenface reconstruction w/ other faces



(a) The target face



(b) Reconstruction

Figure: Reconstruction of a new face: A comparison between the target face and its reconstruction using PCs learned from all other faces.

drawbacks of PCA

PCA works well when data is clustered (e.g., generated from a single-modal distribution). PCs are not meaningful when manifolds exist.

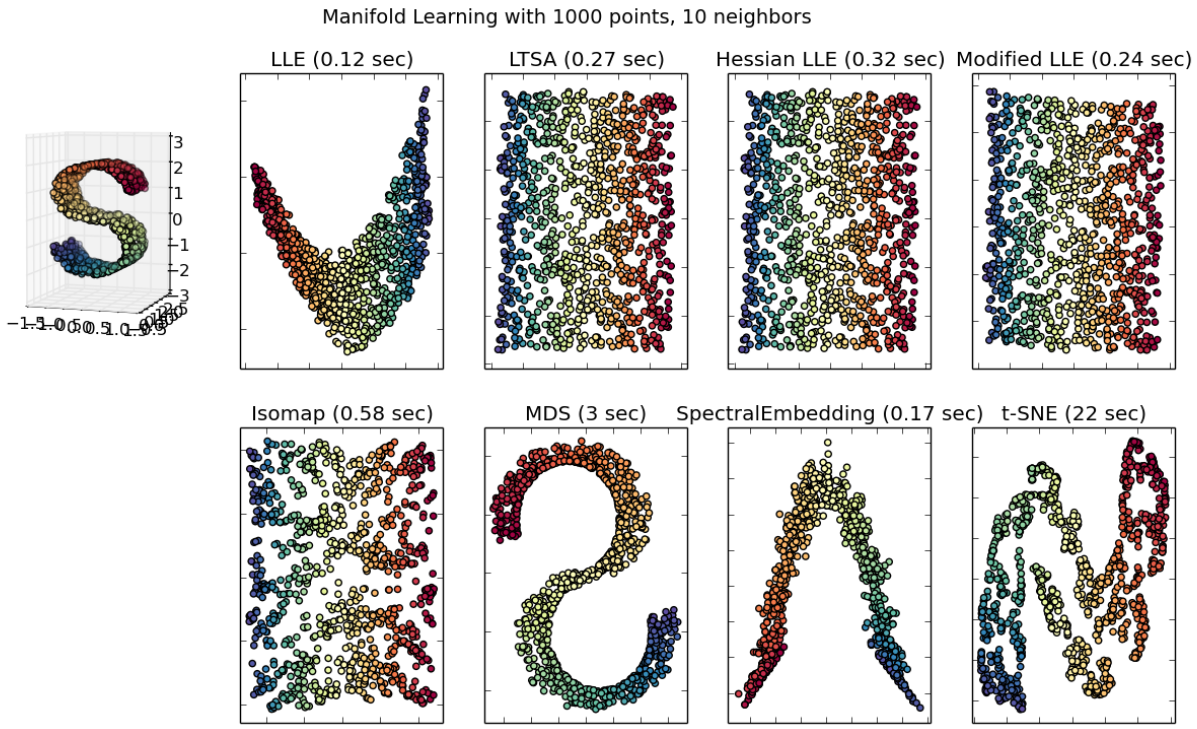


Figure: Comparison between nonlinear dimension reduction methods. Figure from Sciki-learn.

summary of the class

- principal component analysis: data dimension reduction using a linear subspace
- relationship between principal components and eigenvectors of the information matrix
- drawbacks of PCA

Python code for demos in the class

```
###### load a bunch of packages
import itertools
import numpy as np
import matplotlib as mlp
import matplotlib.pyplot as plt
import pandas as pd
import scipy
from sklearn import cluster
from sklearn import datasets
from sklearn import metrics
from sklearn.neighbors import kneighbors_graph
from sklearn.preprocessing import StandardScaler
from sklearn import decomposition # PCA
import time
###### DONE loading

###### import images from Olivetti dataset
faces = datasets.olivetti_faces.fetch_olivetti_faces()
print(faces.DESCR)

###### some preprocess on all the faces
faces_images = faces['images']
faces_data = faces.data
n_samples, H, W = faces_images.shape
n_features = H*W
mean_image = faces_data.mean(axis=0) # averaged face over all faces
faces_data_centered = faces_data - faces_data.mean(axis=0) # global centering
faces_data_centered -= faces_data_centered.mean(axis=1).reshape(n_samples, -1) # local centering
```

Python code for demos in the class

```
##### plot the first 32 faces, just to take a look
fig = plt.figure(figsize=(16, 16))
for ii in range(32):
    plt.subplot(8, 8, ii + 1) # It starts with one
    plt.imshow(np.reshape(faces_data_centered[ii],[H,W]), cmap=plt.cm.gray)
    plt.grid(False);
    plt.xticks([]);
    plt.yticks([]);

##### Creating PCA object
pca = decomposition.RandomizedPCA()
pca.fit(faces_data_centered) # We are applying PCA to the data

##### plot the first 16 PCs
plt.figure(figsize=(16, 16));
for ii in range(16):
    plt.subplot(4, 4, ii + 1) # It starts with one
    plt.imshow(pca.components_[ii].reshape(64, 64), cmap=plt.cm.gray)
    plt.grid(False);
    plt.xticks([]);
    plt.yticks([]);

##### plot Explained Variance Ratio over Component
with plt.style.context('fivethirtyeight'):
    plt.figure(figsize=(16, 12));
    plt.title('Explained Variance Ratio over Component');
    plt.plot(pca.explained_variance_ratio_);

##### plot Cumulative Explained Variance over EigenFace
with plt.style.context('fivethirtyeight'):
    plt.figure(figsize=(16, 12));
    plt.title('Cumulative Explained Variance over EigenFace');
    plt.plot(pca.explained_variance_ratio_.cumsum());
```

Python code for demos in the class

```
##### reconstructed faces
loadings = pca.components_
n_components = loadings.shape[0]
scores = np.dot(faces_data_centered[:,:], loadings[:,:].T)
img_proj = []
img_idx = 0 # pick one image to look at
for n in range(n_components):
    proj = np.dot(scores[img_idx, n], loadings[n,:])
    img_proj.append(proj)
faces = mean_image
face_list = []
face_list.append(mean_image)
for i in range(len(img_proj)):
    faces = np.add(faces, img_proj[i])
    face_list.append(faces)
face_arr = np.asarray(face_list)

##### define a plotting function
def image_grid(D,H,W,cols=10,scale=1):
    """ display a grid of images
        H,W: Height and width of the images
        cols: number of columns = number of images in each row
        scale: 1 to fill screen
    """
    n = np.shape(D)[0]
    rows = int(np.ceil((n+0.0)/cols))
    fig = plt.figure(1,figsize=[scale*20.0/H*W,scale*20.0/cols*rows],dpi=300)
    for i in range(n):
        plt.subplot(rows,cols,i+1)
        fig=plt.imshow(np.reshape(D[i,:],[H,W]), cmap = plt.get_cmap("gray"))
        plt.axis('off')

##### plot faces reconstructed with 5 to 100 PCs, at an interval of 5
image_grid(face_arr[range(0, 100, 5)], H, W, cols=5)
```

Python code for demos in the class

```
##### reconstruct a face that does not exist in the dataset
new_faces = faces_data[10:,:] # take out the first 10 faces from the same person
target_face = faces_data[0,:] # pick the first face for reconstruction
mean_face = new_faces.mean(axis=0)
centered_face = target_face - mean_face
plt.imshow(np.reshape(centered_face,[H,W]), cmap = plt.get_cmap("gray"))
faces_data_centered = new_faces - mean_face
pca.fit(faces_data_centered)

loadings = pca.components_
n_components = loadings.shape[0]
scores = np.dot(centered_face, loadings.T)
reconstruct = np.dot(scores, loadings)
plt.imshow(np.reshape(reconstruct+mean_face, [H,W]), cmap = plt.get_cmap("gray"))
plt.figure()
```