Principal Component Analysis

Mohamed Ali Ashfaq Ahamed[†]

[†] *Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's (NL) A1C 5S7, Canada*

E-mails: aamohamedali@mun.ca

Abstract

Principal componenet analysis (PCA) is decomposing a matrix in a professional way with the highest details in one part of the matrix and the noisy redundant details on the remaining part of the matrix (where we can omit the details). PCA is a method for compressing a lot of data into something that captures the essence of the original data. The goal of PCA is to convert a high-dimensional matrix into low dimensional matrix with the same features. There are many applications like mathematical modeling, scientific computing, machine learning, fluid dynamics and data science. The best example that i can present is a movie, where the 3D subjects are turned into 2D subjects where story is conveyed to the audience by omitting the 3rd dimension which doesn't add much to the story.

1 Introduction

1.1 Background

Suppose there is a measurement of temperature T and pressure P given at different times at constant volume V. The data set represents a set of N measurements. With the knowledge of PCA related to covariance we try to determine the dependence of P on T. Suppose P and T are independent of each other then the two variables would be uncorrelated. it is known that P and V are correlated by the ideal gas law pv=nrt. The tool for quantifying correlations between random variables is covariance.

1.2 Context

Let W be a subspace for V. For any $v \in V$, consider the problem of finding a vector $w \in W$ such that $||v - W||$ is as small as possible. The vector $w \in W$ is the best approximation of $v \in V$ by vectors in W if

$$||v - W|| < ||v - w||.$$

for all $w \in W$.

Finding the best approximation is, in general, rather difficult. However, if the norm $||.||$ corresponds to the induced norm of an inner product, then one can use orthogonal projection and the problem is greatly simplified.

1.3 Outline

1. Based on the standard covariance matrix and singular value decomposition (SVD), finding the principal components and principal directions for figure 1.

Height (cm)	Weight (kg)
170	65
175	70
180	72
182	75
185	80

Figure 1: First data set.

2. Based on the singular value decomposition (SVD) finding the principal components and the corresponding directions of data for figure 2.

Mathematics	Physics	English
80	75	70
85	80	75
90	85	80
95	90	85
100	95	90
95	90	85
90	85	80
85	80	75
80	75	70
75	70	65

Figure 2: Second data set.

3. Suppose that $U \in R^{NXM}$ represents M measurements of a N-dimensional data,where N is much larger than M. U is the satellite measurement of a winter storm passing over Newfoundland. Deriving the following expression.

$$\psi_i = \frac{1}{\sqrt{MU}} U^T \phi_i.$$

where ϕ_i denote the principal directions, λ_i denote the principal values of the given data U, and ψ_i denote the eigenvectors of $U^T U$.

4. Developing a Python code (*.py) that implements the PCA method for high-dimensional data and analyzes a set of face images. There are 400 images of 40 faces at a resolution of $64X64$ pixels. Each face has 10 different expressions or variants. Mathematically, each image is a 4096-dimensional vector, and the data matrix is $U \in R^{NXM}$, where N = 4096 and M = 400. The first 10 columns of U are 10 variations of the first face, corresponding to id = 0, and so on. Plotting the first 40 images in 4 rows. Approximating the images considering only the first principal component and discarding all of the remaining principal components. Plotting the first-principal approximation of images projecting over the first principal components.

1.4 Hypothesis

It isn't required to find the principal direction and principal components to visualize the variation especially if it can be drawn in 2D or 3D graphs. For the first two data matrices that this projects works, we can plot the 2D and 3D graphs to capture all the variations in the data. The principal components should be the same in both the covariance and SVD approach.

2 Methods

There are two approaches in reaching the principal component analysis, Singular value decompositions(SVD) and covariance matrix.

- Covariance matrix

  First we are supposed to find the Mean by using the mean equation.

  $$\bar{U} = \frac{1}{M}\sum_{j=1}^{M} U_{ij}.$$

  Then the variatiion is found by $V = U - \bar{U}$. Following that, we are able to find the covariance matrix.

  $$Covariation = \frac{1}{M}VV^{\top}.$$

  Furthermore, the PCA is found by performing eigendecomposition of the covariance matrix.

  $$A = Covariaiton$$

  $$A = \phi\Lambda\phi^{\top}.$$

  $\phi = [\phi_1\phi_2\phi_3.....\phi_n]$ where each column of ϕ is a principal direction. Finally PCA is found by

  $$PCA = A\phi.$$

- SVD

  Diagonilizing the the data matrix as below

  $$U = \phi\Sigma\psi^{\top}.$$

  $$UU^{\top} = \phi\Sigma\psi^{\top}\psi\Sigma^{\top}\phi^{\top}.$$

  $$\frac{UU^{\top}}{M} = \phi\frac{\Sigma^2}{M}\phi^{\top}.$$

  The left singular vector of U are principal directions and right singular vector of u are principal components

- Application of PCA for image analysis

  Read the data set of 400 images and plot the graph for 40 images. Use singular value decomposition and find the principal component 1(PCA1). Approximate

the images by taking the dot product of PCA1 over the variation. Finally plot the approximated 40 images by taking first-principal approximation images over the first principal component.

3 Results

1. The principal direction and components for the data set in figure 1 computed using covariance matrix are respectively.

$$PD = \begin{bmatrix} 0.72862924 & -0.68490834 \\ 0.68490834 & 0.72862924 \end{bmatrix}$$

$$PC = \begin{bmatrix} 38.27452118 & -0.51403178 \\ 35.97788433 & 0.54684483 \end{bmatrix}$$

The principal direction and components computed using SVD are respectively.

$$PD = \begin{bmatrix} -0.72862924 & -0.68490834 \\ -0.68490834 & 0.72862924 \end{bmatrix}$$

$$PC = \begin{bmatrix} -38.27452118 & -0.51403178 \\ -35.97788433 & 0.54684483 \end{bmatrix}$$

2. The principal direction and components for the data set in figure 2 computed using SVD are respectively.

$$PD = \begin{bmatrix} -5.77350269e-01 & 8.16496581e-01 & 3.59395730e-16 \\ -5.77350269e-01 & -4.08248290e-01 & -7.07106781e-01 \\ -5.77350269e-01 & -4.08248290e-01 & 7.07106781e-01 \end{bmatrix}$$

$$PC = \begin{bmatrix} [0.31622777 & 0.10540926 & -0.10540926 & -0.31622777 & -0.52704628 \\ -0.31622777 & -0.10540926 & 0.10540926 & 0.31622777 & 0.52704628] \\ [-0.93814237 & -0.02513053 & 0.02513053 & -0.07460841 & -0.22434735 \\ -0.07460841 & 0.02513053 & -0.02513053 & 0.07460841 & 0.22434735] \\ [0.05945468 & -0.99395051 & -0.00229225 & -0.0295284 & -0.05429541 \\ -0.0295284 & -0.00229225 & 0.00229225 & 0.0295284 & 0.05429541] \end{bmatrix}$$

3. Derived expression is as follows.

$$U = \phi \Sigma \psi^T.$$

$$\frac{U^T U}{M} = \psi \frac{\Sigma}{M} \psi^T$$

$$\frac{U^T U}{M} = \psi \lambda \psi^T$$

$$\frac{U^T U}{M}\psi\lambda^{-1} = \psi$$

But we are aware that $U\psi = \phi\Sigma$

$$\frac{U^T}{M}\phi\Sigma\lambda^{-1} = \psi$$

also $\lambda = \frac{\Sigma^2}{M}$ which implies $\Sigma = \sqrt{\lambda M}$

$$\frac{U^T}{M}\phi\sqrt{\lambda M}\lambda^{-1} = \psi$$

$$\frac{U^T\phi}{\sqrt{\lambda M}} = \psi$$

$$\psi = \frac{U^T\phi}{\sqrt{\lambda M}}$$

4. PCA for image analysis
   Plot of the first 40 images in 4 rows and approximated images considering only the first principal component discarding all of the remaining principal components.

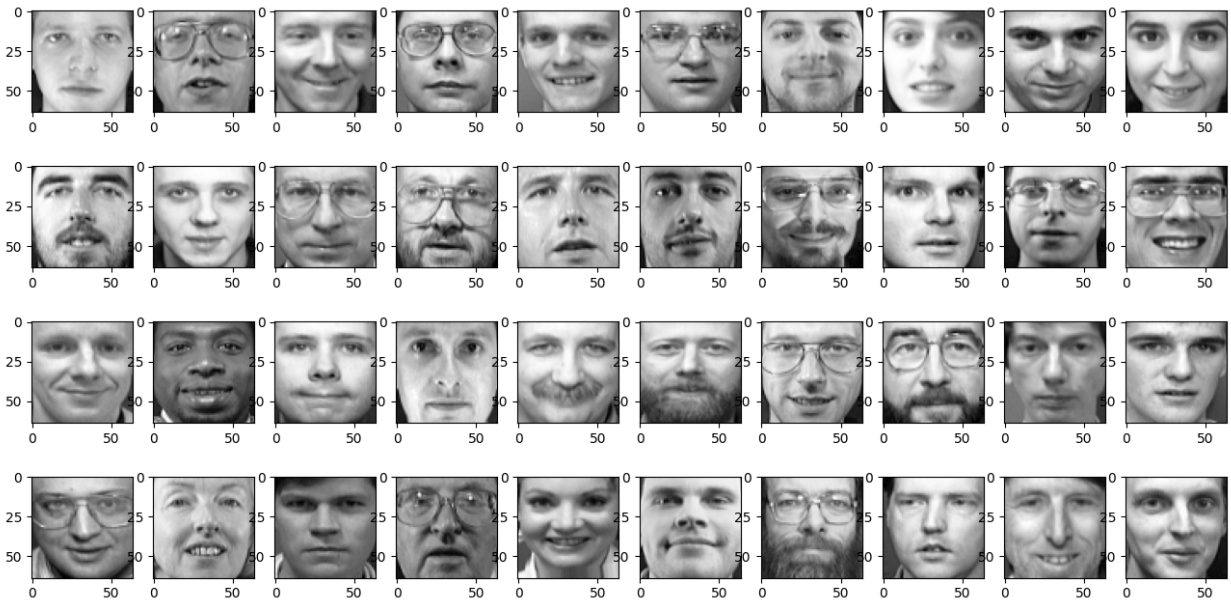

Figure 3: 40 images

Plot of the first-principal approximation of images projecting over the first principal components.

Figure 4: Approximate images

4 Analysis

I am plotting two graphs between height & weight and PCA1 & PCA2. I can realize that the principal components have really captured the essential information.

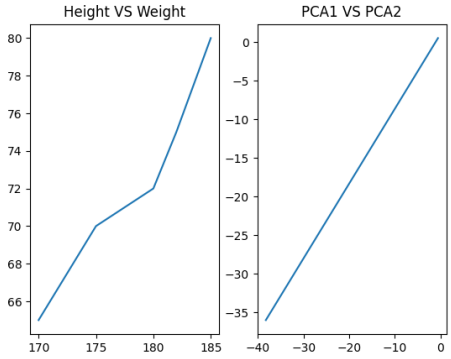

Figure 5: Graph between the two data and the principal components

Even if i didn't divide by M, while finding the Standard covariance matrix, it doesn't affect the results. It is really difficult to find the correlation with the given data. But formulating the Principal components helps us to visualize the correlation between the given data. If it is highly correlated the scattered plot will be skewed else scattered.

The pricipal direction produced by both the approaches should be similar but in my case, it is different, that's a major issue i have to face.

Large datasets are becoming more common across a variety of areas. Data scientist reduces the dimensionality of such data sets in order to maintain the most closely related data while interpreting them. Eigenvectors and eigenvalues are resolved by Principal component analysis. A significant portion of the variation is explained by PCA, which resembles a low-dimensional representation of the observations. If we have too many predictors in relation to the number of observations, we can use PCA to reduce the number of variables, eliminate multicollinearity, or both, in order to reduce the number of plots required for visual analysis while maintaining the majority of the information included.

In machine learning, feature reduction is a crucial pre-processing step. PCA is therefore a crucial pre-processing step and a great tool for data compression and noise removal. By identifying a new collection of variables that are less numerous than the initial set of variables, it lowers the dimensionality of a dataset.

5 Appendix 1

```
import matplotlib.pyplot as plt
import numpy as np


###################
# Author: Ashfaq (Applied Mathematics, Memorial University)
#
# implements the PCA method for a given data matrix.
#reading the data
U=np.array([[170,175,180,182,185],[65,70,72,75,80]])
height=U[0]
weight=U[1]



#calculating the PCA using SVD
Ubar=np.mean(U,axis=1,keepdims=1)
V=U-Ubar
Phi, Sig, PsiT=np.linalg.svd(V,full_matrices=0)

#calculating the PCA using Covariance matrix
C=np.dot(V,V.T)/V.shape[1]
eigVal, eigVec=np.linalg.eig(C)
PCA_CM=np.dot(C,eigVec)
PCA_SVD=np.dot(C,Phi)

#printing the results
print("principal direction computed using SVD: ",Phi)
print("principal direction computed based on covariance matrix: ",eigVec)
print("Principal component computed based on SVD: ",PCA_SVD )
print("Principal component computed based on covariance matrix: ",PCA_CM )
```

```
#plotting the graph between height and wieght.
plt.subplot(1,2,1)
plt.plot(height,weight)
plt.title("Height VS Weight")


#plotting the graph between PCA1 and PCA2
plt.subplot(1,2,2)
plt.plot(PCA_CM[0],PCA_CM[1])
plt.title("PCA1 VS PCA2")

plt.show()
```

6 Appendix 2

```
import matplotlib.pyplot as plt
import numpy as np


###################
# Author: Ashfaq (Applied Mathematics, Memorial University)
#
# implements the PCA method for a given data matrix.#reading the data

U=np.array([[80,85,90,95,100,95,90,85,80,75],
            [75,80,85,90,95,90,85,80,75,70],
            [70,75,80,85,90,85,80,75,70,65]])


#calculate the PCA using SVD
#
Ubar=np.mean(U,axis=1,keepdims=1)
V=U-Ubar
Phi, Sig, PsiT=np.linalg.svd(V,full_matrices=0)

PCA=np.dot(Phi,Sig)

#printing the results
print("principal direction computed using SVD: ",Phi)

print("Principal component computed based on SVD: ",PCA )

#plotting the graphs between PCA1 and PCA2
plt.plot(PCA[0],PCA[1])
```

7 Appendix 3

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.datasets import fetch_olivetti_faces


###################
# Author: Ashfaq (Applied Mathematics, Memorial University)
#
# implements the PCA method for a high-dimensional data and analyze a set of face ima

#reading the data
faces = datasets.fetch_olivetti_faces()
U = faces.data

U=U.T


ids = np.unique(faces.target)

#plotting the first 40 images
plt.subplots(nrows=4, ncols=10, figsize=(16,8))
for id in ids:
    plt.subplot(4,10,id+1)
    plt.imshow(U[:,id*10].reshape((64,64)),cmap='gray')
plt.show()

#calculating PCA using SVD
Ubar=np.mean(U,axis=0,keepdims=1)
V=U-Ubar
Phi, Sig, PsiT=np.linalg.svd(V,full_matrices=0)

PCA1=Phi[:,0]

#approximating the images
uapprox=np.dot(PCA1,V)[:, np.newaxis]
uapprox=np.dot(uapprox,PCA1.reshape(-1,1).T)+Ubar.reshape(-1,1)

#plotting over the first principal approximation over first principal components
plt.subplots(nrows=4, ncols=10, figsize=(16,8))
for id in ids:
    plt.subplot(4,10,id+1)
    plt.imshow(uapprox.T[:,id*10].reshape((64,64)),cmap='gray')
plt.show()
```