Application Of PCA

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 component analysis (PCA) is an orthogonal transformation that seeks the directions of maximum variance in the data and is commonly used to reduce the dimensionality of the data. PCA is a statistical technique for simplifying a data-sets by reducing data-sets to lower dimensions. It is a standard technique commonly used for data reduction in statistical pattern recognition and signal processing.

1 Introduction

1.1 Background

Principal components analysis (PCA) is one of the central uses of the SVD, providing a data-driven, hierarchical coordinate system to represent high-dimensional correlated data. This coordinate system involves correlation matrices. Importantly, PCA pre-processes the data by mean subtracttion and setting the variance to unity before performing the SVD. The geometry of the resulting coordinate system is determined by principal components (PCs) that are uncorrelated (orthogonal) to each other, but have maximal correlation with the measurements. This theory was developed in 1901 by Pearson [418], and independently by Hotelling in the 1930s [256, 257]. Jolliffe [268] provides a good reference text.
Typically, a number of measurements are collected in a single experiment, and these measurements are arranged into a row vector. The measurements may be features of an observable, such as demographic features of a specific human individual. A number of experiments are conducted, and each measurement vector is arranged as a row in a large matrix X.

1.2 Context

Removal of noise is an important step in the image restoration process, but denoising of image remains a challenging problem in recent research associate with image processing. Denoising is used to remove the noise from corrupted image, while retaining the edges and other detailed features as much as possible. This noise gets introduced during acquisition, transmission reception and storage retrieval processes.

Most of the data collected from Functional magnetic resonance imaging (FMRI) scans are usually noise, simply performing PCA presents the risk that we will lose the true signal especially since the variance explained by the signal is very little relative to the

noise. A key statement is the fact that the first principal component is usually the most significant, the second more significant and so on. What if we looked at the components that are less significant and Ignore the loud mouth components explaining most of the Variance? While crude, this is the principle behind PCA for Noise reduction. Depending on what proportion of Noise you want to filter for, you organize your components by the variance they explain and then reconstruct your original data. Most true signal is usually weak and will not stand out in the face of the Noise thus by removing the most significant portion of the raw data, we are left with 'more concentrated soup' of data with the true signal being more visible.

1.3 Outline

1. Write a program to simulate the data function $U = \Phi\Sigma\Psi^T$ and the noisy data $V = U + \epsilon\eta$ with n = 600 and ϵ=1. Plot U and V adjacent to each other to show their differences.

2. Remove the noise using principal component analysis. Calculate the singular values of U and V using singular value decomposition. Compare the singular values in the table.

3. Consider thresholding criteria, set all singular values that are smaller than a tolerance τ. Then compute $\overline{U} = \Phi\overline{\Sigma}\Psi^T$ as the clean data, where $\overline{\Sigma}$ is the thresholded singular value matrix.

4. Repeat the above thresholding process using $\epsilon = 0.5$. For the two thresholding tests, use a plot to indicate the level of accuracy and examine if there are any effects of ϵ.

2 Methods

1. Simulate a data matrix U such that $U = \Phi\Sigma\Psi^T$

$$\Phi(x) = \begin{bmatrix}\cos(17x)e^{-x^2} & \sin(11x)\end{bmatrix}$$

$$\Sigma = \begin{bmatrix}2 & 0\\ 0 & \frac{1}{2}\end{bmatrix}$$

$$\Psi(x) = \begin{bmatrix}\sin(5x)e^{-x^2} & \cos(13x)\end{bmatrix}$$

   Where $-3 \leq x \leq 3$ and convert the data matrix U into a noisy data matrix $V \in \mathbb{R}^{NXN}$ such that

$$V = U + \epsilon\eta.$$

   Where $\eta \in \mathbb{R}^{NxN}$ is a random matrix with standard normal distribution and ϵ is a scalar parameter.

2. Take $n = 600$, $\epsilon = 1$ and plot U and V.

3. Calculate the singular values of U and V using singular value decomposition. Compare the singular values in the table and identify which of the PCA modes are most likely relevant to the pure data.

4. Consider the following thresholding criteria. Set all singular values that are smaller than a tolerance τ. Then compute $\overline{U} = \Phi\overline{\Sigma}\Psi^T$ as the clean data, where $\overline{\Sigma}$ is the thresholded singular value matrix. The threshold τ can be chosen as $\frac{4}{\sqrt{3}}\sqrt{n}\epsilon$, which is known to be an optimal value in the data science literature.

5. Repeat the above thresholding process using $\epsilon = 0.5$. For the two thresholding tests, use a plot to indicate the level of accuracy and if there are any effects of ϵ. A good option may be to compare a row (or column) of U (or $\Phi^T U$) with that of $\bar{U}$ (or $\Phi^T\bar{U}$). The comparison can be done visually (with a plot) or numerically using a correlation matrix.

3 Results

1. Plot of pure data U and noisy data V

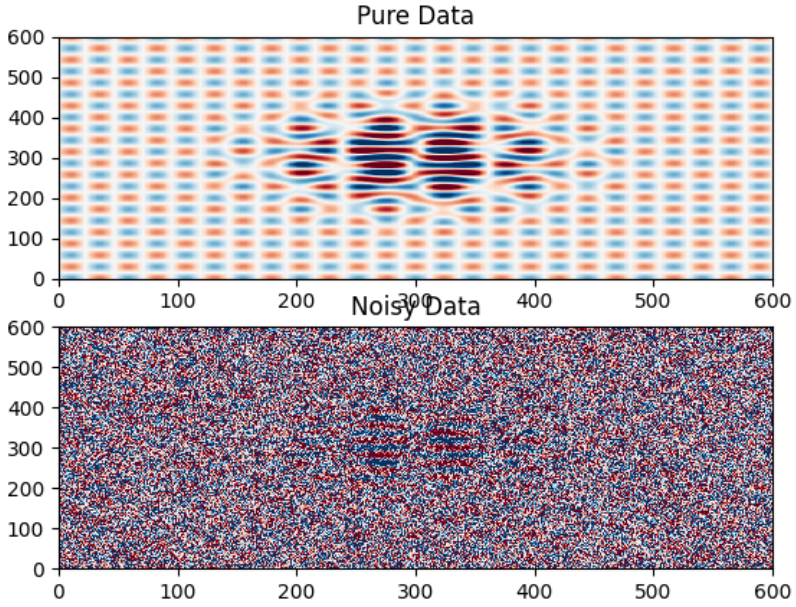

Figure 1: The two plots drawn above are adjacent to each other to indicate the differences between the pure data U and noisy data V.

2. Determining the singular values of pure data U and noisy data V using singular value decomposition.

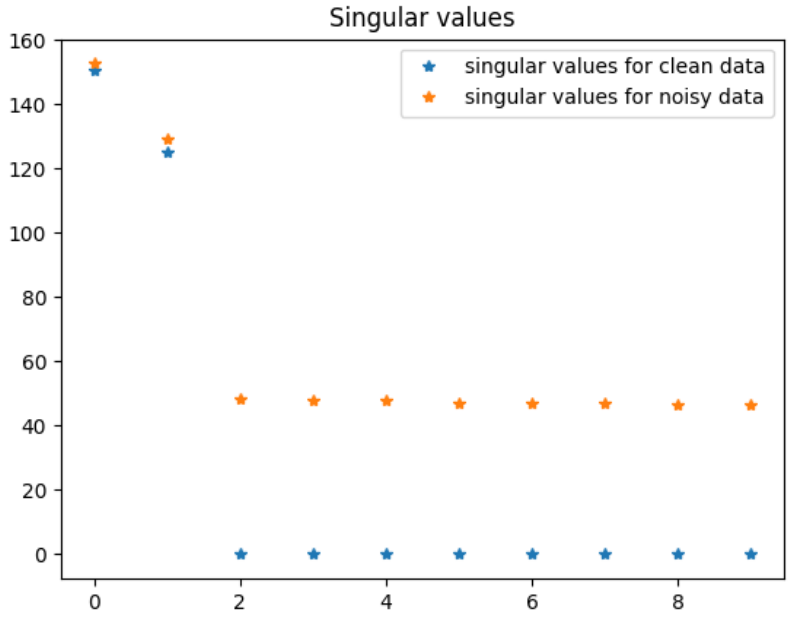

Figure 2: The above plot illustrates the 10 strongest singular values of pure data U and noisy data v to identify the PCA components which are more relevant to pure data.

Singular values of U	Singular values of V
1.50538151e+02	154.45132777
1.25122295e+02	130.26206168
1.36822069e-13	48.74400626
1.33704737e-13	48.19224507
9.77599010e-14	47.97970447
8.93097686e-14	47.73898626
8.44578185e-14	47.35166236
8.24568120e-14	47.00413398
8.04286568e-14	46.81938321
7.89057300e-14	46.62682463

3. Consider a threshold criteria τ, and determine a thresholded pure data $\bar{U}$ & thresholded noisy data $\bar{U}$.

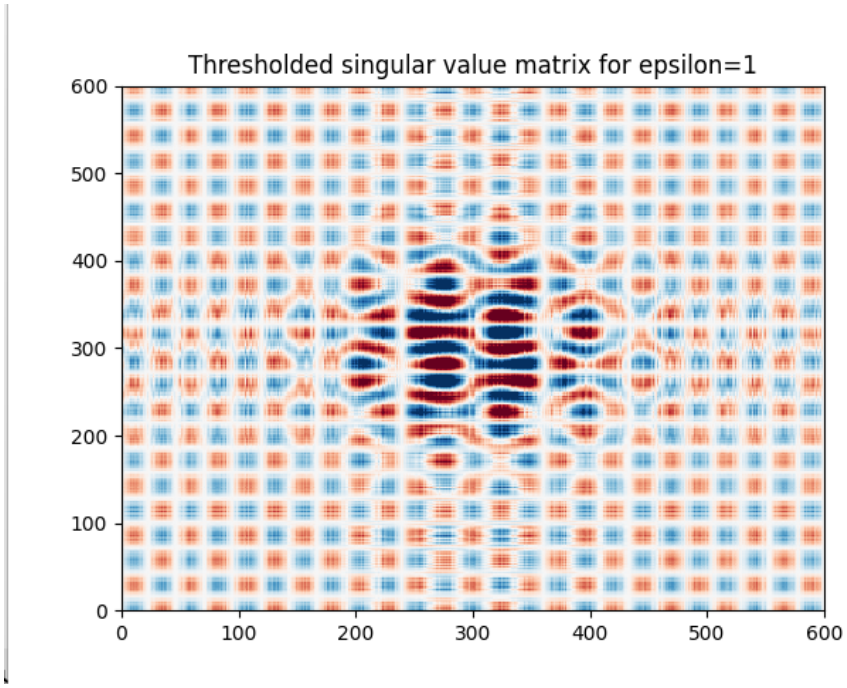

Figure 3: The above plot illustrates $\bar{U}$ by setting all singular values less than tau and considering $\epsilon = 1$.

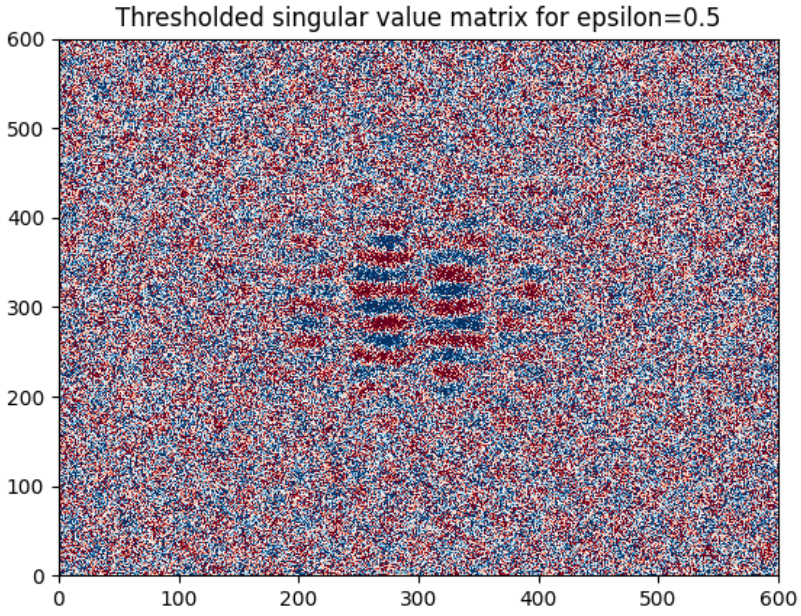

Figure 4: The above plot illustrates $\bar{U}$ by setting all singular values less than tau and considering $\epsilon = 0.5$.

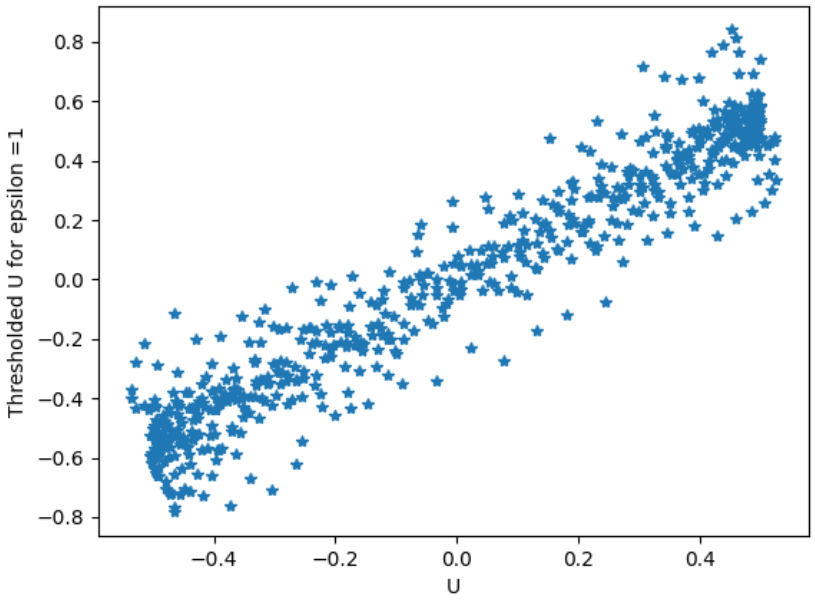

Figure 5: The plot is drawn between U and thresholded $\bar{U}$ for $\epsilon = 1$ to indicate the level of accuracy and identify the effects of $\bar{U}$ while $\epsilon = 1$

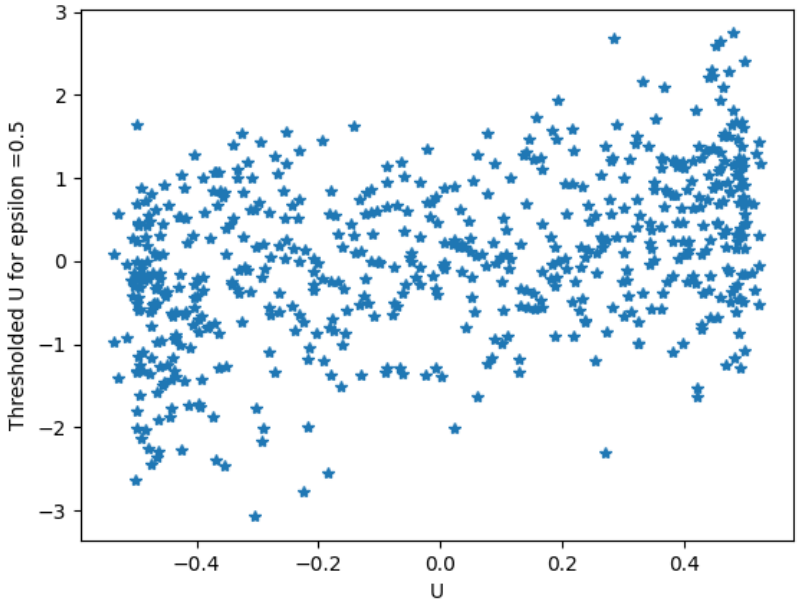

Figure 6: The plot is drawn between U and thresholded $\bar{U}$ for $\epsilon = 1$ to indicate the level of accuracy and identify the effects of $\bar{U}$ while $\epsilon = 0.5$

$$\text{Co-relation Coefficients for}_{\epsilon=1} = \begin{bmatrix} 1. & 0.39682743 \\ 0.39682743 & 1. \end{bmatrix}$$

4 Analysis

Principal component analysis methods fail if it has neither variability nor variation of variability.

- The noisy data plot looks messy than the pure data. It is very clear that analysts wouldn't be able to come to a conclusion by observing the messy data. The data measured in the real world all turns out to be messy where we extract more unnecessary data than the necessary ones.
- By considering the plot of singular values, I can certainly point out that the singular values are getting larger for noisy data than the pure data. If I consider the table of singular values, the first and second components of the pure and clean data are pretty much similar. So, even without eliminating the noise from the data, I may be able to capture something from the first and second components. Beyond the second component, I can't capture anything.
- The noise can be eliminated by considering the universal thresholding criteria. In other words, look for singular values less than the threshold and remove them.
- While the ϵ=1, the noise is reduced on a large scale but while the ϵ=0.5 most of the noise still remains the same. The correlation matrix shows that it is 39% co-related between U and $\bar{U}$ for ϵ=1. Not only the correlation matrix, the plot clearly illustrates that for ϵ=1 data is highly correlated. But for ϵ=0.5 the plot emphasizes that it is weekly correlated.
- We already know the amount of Noise we have in the function we are working. In real data, we tend not to know the exact amount of variance accounted for by the noise. To filter out the noise, we tend to rely on a lot of experimentation, fitting in a number of parameters and testing different thresholds until we achieve a level we are satisfied with. With PCA, this might entail including a various number of components. We assume our data is Gaussian. For our demo, we will randomly sample noise from a gaussian distribution. In real data too, noise can be variable in distribution but by removing the Gaussian component of it, we are sweeping the data fairly clean which makes the assumption hold.

5 Appendix 1

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


################################
#Author-Ashfaq Ahamed
#
#
#University- Memorial University Of Newfoundland
############################
n=600
```

```
x=np.linspace(-3,3,n)
Phi=np.array([np.cos(17*x)*np.exp(-x**2),np.sin(11*x)])
Sigma=np.array([[2,0],[0,1/2]])
Phsi=np.array([np.sin(5*x)*np.exp(-x**2),np.cos(13*x)])
#pure data
U=Phi.T@Sigma@Phsi

#constructing a noisy data
sigma=1
neata=np.random.randn(n,n)
V=U+sigma*neata

#plotting clean data and noisy data
plt.figure()
plt.subplot(2,1,1)
plt.pcolormesh(U,cmap='RdBu_r',vmin=-1,vmax=1)
plt.title('Pure Data')
plt.xlabel('x')
plt.ylabel('y')

plt.subplot(2,1,2)
plt.pcolormesh(V,cmap='RdBu_r',vmin=-1,vmax=1)
plt.title('Noisy Data')
plt.xlabel('x')
plt.ylabel('y')

#performing PCA for U
Phi1, Sig1, PsiT1=np.linalg.svd(U,full_matrices=0)
print(Sig1[:10])

#performing PCA for V
Phi2, Sig2, PsiT2=np.linalg.svd(V,full_matrices=0)
print(Sig2[:10])

#co-relation
plt.figure()
plt.plot(Sig1[:10],label='singular values for clean data')
plt.plot(Sig2[:10],label='singular values for noisy data')
plt.title('Singular values')
plt.legend()



#calcualting the tolerence when epsilon is 1
thau=4/np.sqrt(3)*np.sqrt(n)*sigma

i=np.max(np.where(Sig2>thau))+1
cU=np.dot(Phi2[:,:i]*Sig2[:i],PsiT2[:i])
```

```
#calcualting the tolerence when epsilon is 0.5
sigma1=0.5
thau=4/np.sqrt(3)*np.sqrt(n)*sigma1
i1=np.max(np.where(Sig2>thau))+1
cU1=np.dot(Phi2[:,:i1]*Sig2[:i1],PsiT2[:i1])

#plotting the level of accuracy
plt.figure()
plt.pcolormesh(cU,cmap='RdBu_r',vmin=-1,vmax=1)
plt.title('Thresholded singular value matrix for epsilon=1')

plt.figure()
plt.pcolormesh(cU1,cmap='RdBu_r',vmin=-1,vmax=1)
plt.title('Thresholded singular value matrix for epsilon=0.5')

cor=np.corrcoef(U[:,300],cU[:,300])
cor1=np.corrcoef(U[:,300],cU1[:,300])
print(cor)
plt.figure()
plt.plot(U[:,300],cU[:,300],'*')
plt.xlabel('U')
plt.ylabel('Thresholded U for epsilon =1')

plt.figure()
plt.plot(U[:,300],cU1[:,300],'*')
plt.xlabel('U')
plt.ylabel('Thresholded U for epsilon =0.5')
plt.show()
```