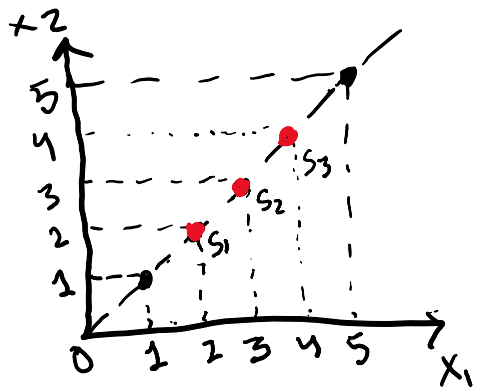
This Document will explain the real mathematical reason to do each step of PCA:

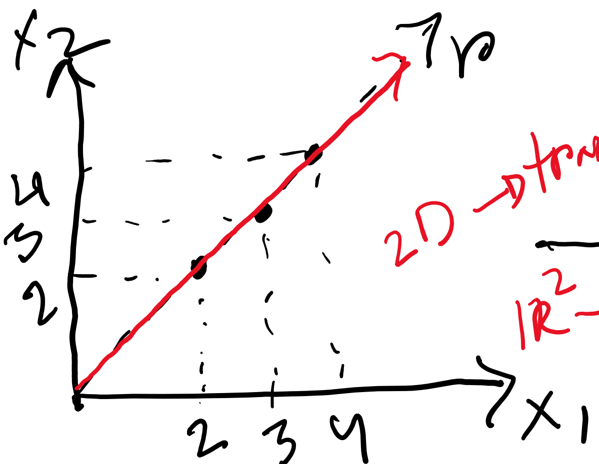
There are 3 data points in a 2D plane can be represented by the coordinate system (x1, x2).



$$X = \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 4 & 4 \end{bmatrix}$$

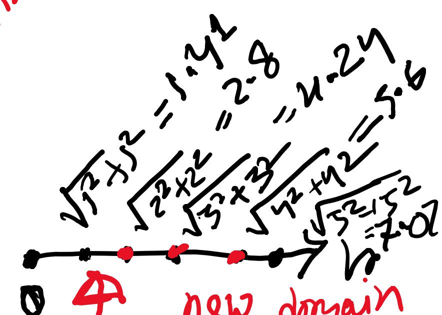
3 sample to find the transformation $\mathbb{R}^2 \to \mathbb{R}$
2 sample to test it

Here, we need two dimensions or two features x1 and x2 to present the data points. Our goal is to develop a technique where we can remove the redundancy between the features. For example, we look the data in the following direction shown in the figure below to represent the data where we will need only one dimension to represent.



2D → transformation → 1D

$\mathbb{R}^2 \to \mathbb{R}$



If transformation is correct then the transformed values will be these values

In order to find the redundancy of the data, we need to find the correlation between the features which can be calculated as the following equation:

normalize $X_{norm} = X - \bar{X} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 4 & 4 \end{bmatrix} - \begin{bmatrix} 3 & 3 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$

$$C_X = X_{norm}^T X_{norm} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} / (3-1) = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} / 2$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$C_X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

if we can diagonalize C_X we can make covariance to zero

The best way to diagonalize is to use eigen value, eigen vector.

This matrix has both diagonal and off-diagonal components. Diagonal components are presenting variance and off-diagonal covariance for each data point pairs. Covariance are presenting the correlation between the pairs. But we would like to make the correlation to zero in order to remove the redundancy between x1 and x2. The only way to do this is to diagonalize this covariance matrix. In other way, we convert Cx to eigen value and eigen vectors [Σ U] where Σ is a diagonal matrix with the variances in descending order.

After doing the eigen value, eigen vector decomposition

$$\Sigma, U = eig\left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right)$$

$$U = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 2 & 0 \end{bmatrix}$$

We have successfully diagonalized Cx. Where, we can see the variance in only first column which is able to capture all the data variation using only the new basis U[:,1] or first principal component

After testing data (1, 1) $\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$= \frac{2}{\sqrt{2}} = \sqrt{2} = 1.41$$

Same value that we wanted to obtain after transformation. Let's try $(5,5) \rightarrow \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}^T \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

which is $\frac{5}{\sqrt{2}} + \frac{5}{\sqrt{2}} = \frac{10}{\sqrt{2}} = 5\sqrt{2} = 7.07$

same value we wanted to obtain.

So we have succesfully reduced the dimension
$\mathbb{R}^2 \to \mathbb{R}$

Now in order to go back $\mathbb{R} \to \mathbb{R}^2$
We can inverse the function Here

$$Y = U^T X$$

unitary matrix

$$\Rightarrow \quad UY = UU^T X$$

$$\Rightarrow \quad UY = IX = X$$

$$\Rightarrow \quad X = UY$$

Here
$UU^T = I$
$\Rightarrow U^{-1}UU^T = IU^{-1}$
$\Rightarrow U^T = U^{-1}$

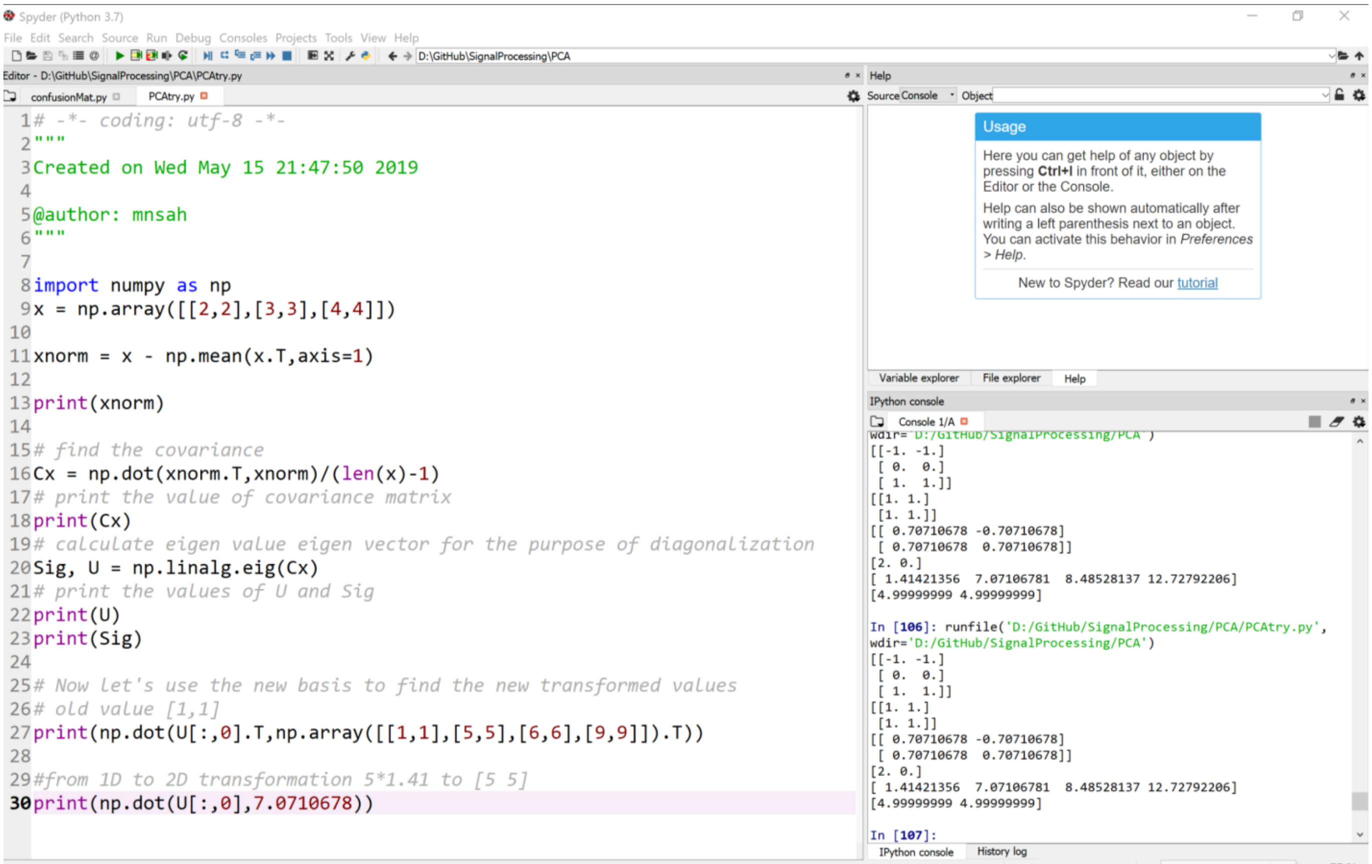
For example we want to go back
$5\sqrt{2} \to X = [x_1 \ x_2]$

$$X = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} 5\sqrt{2} = \begin{bmatrix} \frac{5\sqrt{2}}{\sqrt{2}} & \frac{5\sqrt{2}}{\sqrt{2}} \end{bmatrix}$$

$$X = [5 \quad 5]$$

$\mathbb{R} \to \mathbb{R}^2$

Here is the python code to implement the whole process:



Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\GitHub\SignalProcessing\PCA
Editor - D:\GitHub\SignalProcessing\PCA\PCAtry.py
confusionMat.py
PCAtry.py
-*- coding: utf-8 -*-
"""
Created on Wed May 15 21:47:50 2019

@author: mnsah
"""

import numpy as np
x = np.array([[2,2],[3,3],[4,4]])

xnorm = x - np.mean(x.T,axis=1)

print(xnorm)

# find the covariance
Cx = np.dot(xnorm.T,xnorm)/(len(x)-1)
# print the value of covariance matrix
print(Cx)
# calculate eigen value eigen vector for the purpose of diagonalization
Sig, U = np.linalg.eig(Cx)
# print the values of U and Sig
print(U)
print(Sig)

# Now let's use the new basis to find the new transformed values
# old value [1,1]
print(np.dot(U[:,0].T,np.array([[1,1],[5,5],[6,6],[9,9]]).T))

#from 1D to 2D transformation 5*1.41 to [5 5]
print(np.dot(U[:,0],7.0710678))
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer File explorer Help
IPython console
Console 1/A
[[-1. -1.]
[0. 0.]
[1. 1.]]
[[1. 1.]
[1. 1.]]
[[0.70710678 -0.70710678]
[0.70710678 0.70710678]]
[2. 0.]
[1.41421356 7.07106781 8.48528137 12.72792206]
[4.99999999 4.99999999]
In [106]: runfile('D:/GitHub/SignalProcessing/PCA/PCAtry.py', wdir='D:/GitHub/SignalProcessing/PCA')
[[-1. -1.]
[0. 0.]
[1. 1.]]
[[1. 1.]
[1. 1.]]
[[0.70710678 -0.70710678]
[0.70710678 0.70710678]]
[2. 0.]
[1.41421356 7.07106781 8.48528137 12.72792206]
[4.99999999 4.99999999]
In [107]:
IPython console History log