Student ID: 06996263 Student name: Fang Zhu

Q1:

1.1

()u = 0

()() =

So:

*λ*1 = 2 and *λ*2= 1

1.2

From the first question, we knew that *λ*1 = 2 and *λ*2= 1, so we can get eigenvectors of .

Suppose v1 =  and v2 = .

so

So

If a = 1 and d = -1

*v1 =* and *v2 =*

finally both the vectors are eigenvectors

1.3

According to the spectral theorem:

D is diagonal matrix:

And U is combined with *v­1* and *v2* : [*v­1* , *v2*], v are unit eigenvectors.

From the second question, we get the answer that *v1 =* and *v2 =* .

According to the phenomenon of orthogonal matrix to find the unit eigenvectors.

[ v1 , v2 ] [ v1 , v2 ]T =I

So: a = -d and a = + 0.5 or – 0.5

If a = 0.5

If a = -0.5

So a = + 0.5 or – 0.5 , d = - 0.5 or + 0.5

So U = or

1.4

According to the principal of component transform

(D = , U = [*v­1* , *v2*])

Suppose : y = uTxT ; “u” is a projection vector.

Because u is the eigenvectors of covariance matrix of the sample data.

Than:

=

1.5

yyT = UTxTxU = UTΣU = [v1 , v2]T[v1 , v2] = = D

1.6

According to the Mahalanobis distance, we can achieve:

XT = [x1 , x2] X=[x1 , x2]T

We can get the inverse of as follow:

Finally we get the inverse of XTX:

7x12 - 2x1x2 + 5x22 = 8

Using the python to get this plot as follow:

Code:

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

import math

import textwrap

import numpy as np

p3 = math.sqrt(3)

x1 = np.arange(-2,2,0.01)

x2 = np.arange(-2,2,0.01)

X,Y = np.meshgrid(x1,x2)

Z = 7\*X\*\*2+5\*Y\*\*2-2\*p3\*X\*Y-8

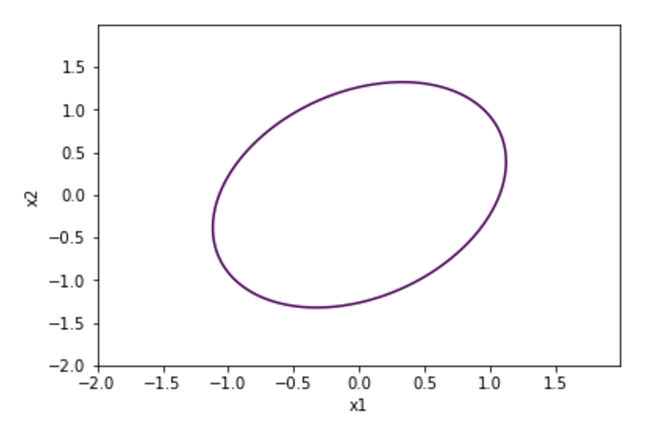
plt.contour(X,Y,Z,0)

plt.ylabel('x2')

plt.xlabel('x1')

plt.show()

picture:



Q2

Ak = (Q QT)k

= QDQTQDQT…QDQT

As we known that the special QTQ= Q-1Q = I

So:

=QDkQT

Q3.

Code:

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

import math

import textwrap

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

import scipy.spatial.distance as sc

spheres = np.loadtxt('spheres.txt')

swissroll = np.loadtxt('swissroll.txt')

ellipsoids = np.loadtxt('ellipsoids.txt')

print(spheres.shape)

print(swissroll.shape)

print(ellipsoids.shape)

#plot results

fig = plt.figure(figsize=(10,30))

ax = fig.add\_subplot(\*[3,1,1], projection='3d')

ax.scatter(spheres[:,0], spheres[:,1], spheres[:,2])

myTitle = 'spheres: ';

ax.set\_title("\n".join(textwrap.wrap(myTitle, 20)))

ax = fig.add\_subplot(\*[3,1,2], projection='3d')

ax.scatter(swissroll[:,0], swissroll[:,1], swissroll[:,2])

myTitle = 'swissroll: ';

ax.set\_title("\n".join(textwrap.wrap(myTitle, 20)))

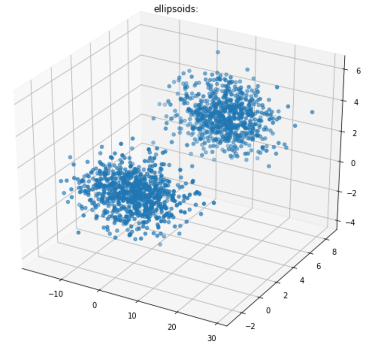
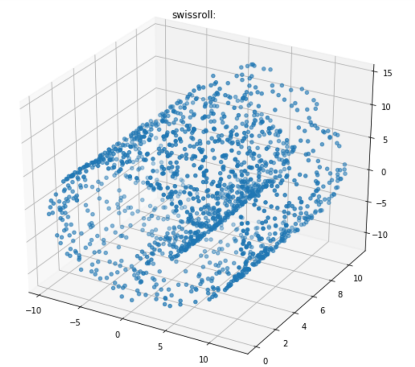
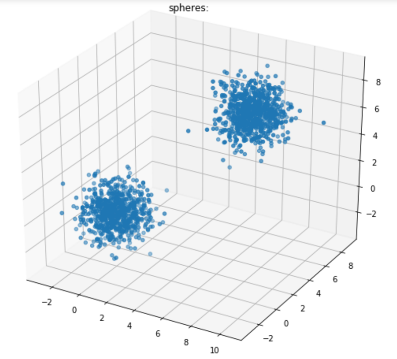
ax = fig.add\_subplot(\*[3,1,3], projection='3d')

ax.scatter(ellipsoids[:,0], ellipsoids[:,1], ellipsoids[:,2])

myTitle = 'ellipsoids: ';

ax.set\_title("\n".join(textwrap.wrap(myTitle, 20)))

plt.show();



(sphere, swissroll, elipsoids from left to right)

Code for 2-D and 1-D pca

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

import math

import textwrap

import numpy as np

spheres = np.loadtxt('spheres.txt')

swissroll = np.loadtxt('swissroll.txt')

ellipsoids = np.loadtxt('ellipsoids.txt')

x1 = spheres[:,0]

y1 = spheres[:,1]

z1 = spheres[:,2]

x2 = swissroll[:,0]

y2 = swissroll[:,1]

z2 = swissroll[:,2]

x3 = ellipsoids[:,0]

y3 = ellipsoids[:,1]

z3 = ellipsoids[:,2]

#print(x1)

Exx1 = sum(x1)/x1.size

Exy1 = sum(y1)/y1.size

Exz1 = sum(z1)/z1.size

Mx1 = np.array([(x1[m]-Exx1) for m in range(0,1500)])

My1 = np.array([(y1[m]-Exy1) for m in range(0,1500)])

Mz1 = np.array([(z1[m]-Exz1) for m in range(0,1500)])

#print(Mx1.shape)

M1 = np.matrix([Mx1,My1,Mz1])

Cov1 = M1@M1.T

#print(Cov1)

eigen\_vals\_1, eigen\_vecs\_1 = np.linalg.eig(Cov1)

#print(eigen\_vals\_1.shape)

#print(eigen\_vecs\_1)

eigen\_pairs\_1 = [(np.abs(eigen\_vals\_1[i]), np.array(eigen\_vecs\_1[:,i].T)[0]) for i in range(len(eigen\_vals\_1))]

eigen\_pairs\_1.sort(key = lambda x : x[0],reverse=True)

#print(eigen\_pairs\_1)

w1 = np.hstack((eigen\_pairs\_1[0][1][:, np.newaxis], eigen\_pairs\_1[1][1][:, np.newaxis]))

M1\_pca = M1.T@w1

w12 = np.hstack((eigen\_pairs\_1[0][1][:, np.newaxis]))

M12\_pca = M1.T@w12

fig = plt.figure(figsize = (10,20))

p11 = fig.add\_subplot(\*[2,1,1])

p11.scatter(np.array(M1\_pca[:,0].T)[0],np.array(M1\_pca[:,1].T)[0])

p11.set\_title('2 Dimension')

p12 = fig.add\_subplot(\*[2,1,2])

p12.scatter(np.array(M12\_pca),np.zeros((1500,1)))

p12.set\_title('1 Dimension')

plt.xlabel('PC 1')

plt.ylabel('PC 2')

plt.legend(loc='lower left')

plt.show()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#print(x1)

Exx2 = sum(x2)/x2.size

Exy2 = sum(y2)/y2.size

Exz2 = sum(z2)/z2.size

Mx2 = np.array([(x2[m]-Exx2) for m in range(0,1500)])

My2 = np.array([(y2[m]-Exy2) for m in range(0,1500)])

Mz2 = np.array([(z2[m]-Exz2) for m in range(0,1500)])

#print(Mx2.shape)

M2 = np.matrix([Mx2,My2,Mz2])

Cov2 = M2@M2.T

#print(Cov2)

eigen\_vals\_2, eigen\_vecs\_2 = np.linalg.eig(Cov2)

#print(eigen\_vals\_2.shape)

#print(eigen\_vecs\_2)

eigen\_pairs\_2 = [(np.abs(eigen\_vals\_2[i]), np.array(eigen\_vecs\_2[:,i].T)[0]) for i in range(len(eigen\_vals\_2))]

eigen\_pairs\_2.sort(key = lambda x : x[0],reverse=True)

#print(eigen\_pairs\_2)

w2 = np.hstack((eigen\_pairs\_2[0][1][:, np.newaxis], eigen\_pairs\_2[1][1][:, np.newaxis]))

M2\_pca = M2.T@w2

w22 = np.hstack((eigen\_pairs\_2[0][1][:, np.newaxis]))

M22\_pca = M2.T@w22

fig2 = plt.figure(figsize = (10,20))

p21 = fig2.add\_subplot(\*[2,1,1])

p21.scatter(np.array(M2\_pca[:,0].T)[0],np.array(M2\_pca[:,1].T)[0])

p21.set\_title('2 Dimension')

p22 = fig2.add\_subplot(\*[2,1,2])

p22.scatter(np.array(M22\_pca),np.zeros((1500,1)))

p22.set\_title('1 Dimension')

plt.xlabel('PC 1')

plt.ylabel('PC 2')

plt.legend(loc='lower left')

plt.show()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#print(x1)

Exx3 = sum(x3)/x3.size

Exy3 = sum(y3)/y3.size

Exz3 = sum(z3)/z3.size

Mx3 = np.array([(x3[m]-Exx3) for m in range(0,1500)])

My3 = np.array([(y3[m]-Exy3) for m in range(0,1500)])

Mz3 = np.array([(z3[m]-Exz3) for m in range(0,1500)])

#print(Mx3.shape)

M3 = np.matrix([Mx3,My3,Mz3])

Cov3 = M3@M3.T

#print(Cov3)

eigen\_vals\_3, eigen\_vecs\_3 = np.linalg.eig(Cov3)

#print(eigen\_vals\_3.shape)

#print(eigen\_vecs\_3)

eigen\_pairs\_3 = [(np.abs(eigen\_vals\_3[i]), np.array(eigen\_vecs\_3[:,i].T)[0]) for i in range(len(eigen\_vals\_3))]

eigen\_pairs\_3.sort(key = lambda x : x[0],reverse=True)

#print(eigen\_pairs\_3)

w3 = np.hstack((eigen\_pairs\_3[0][1][:, np.newaxis], eigen\_pairs\_3[1][1][:, np.newaxis]))

M3\_pca = M3.T@w3

w32 = np.hstack((eigen\_pairs\_3[0][1][:, np.newaxis]))

M32\_pca = M3.T@w32

fig3 = plt.figure(figsize = (10,20))

p31 = fig3.add\_subplot(\*[2,1,1])

p31.scatter(np.array(M3\_pca[:,0].T)[0],np.array(M3\_pca[:,1].T)[0])

p31.set\_title('2 Dimension')

p32 = fig3.add\_subplot(\*[2,1,2])

p32.scatter(np.array(M32\_pca),np.zeros((1500,1)))

p32.set\_title('1 Dimension')

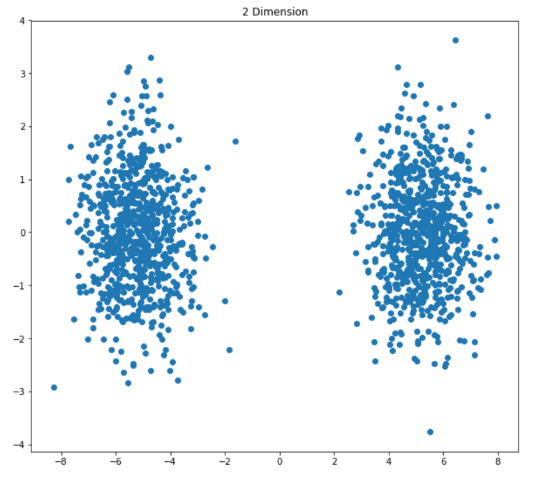
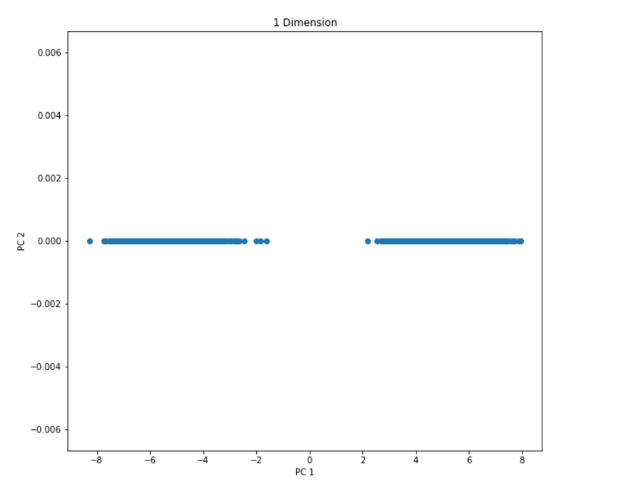
plt.xlabel('PC 1')

plt.ylabel('PC 2')

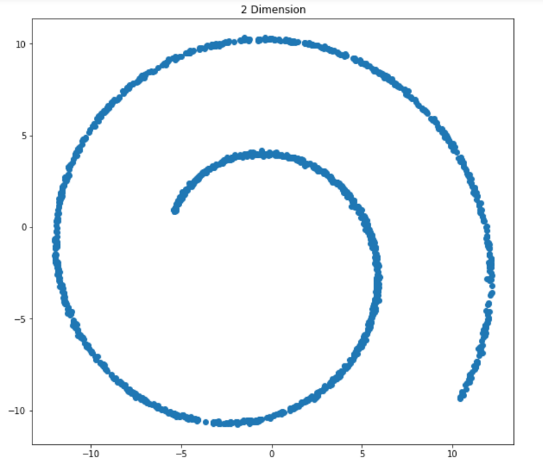
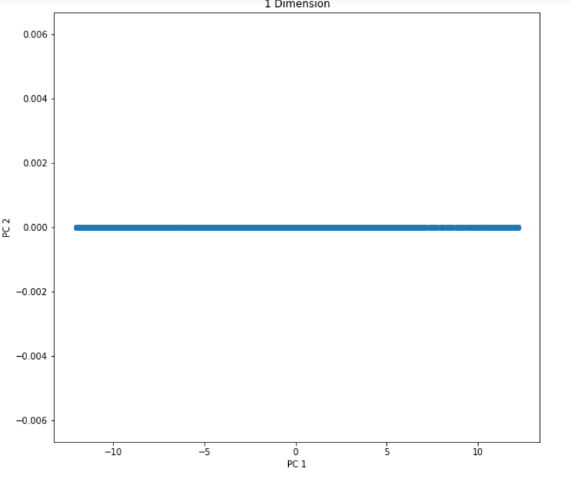
plt.legend(loc='lower left')

plt.show()

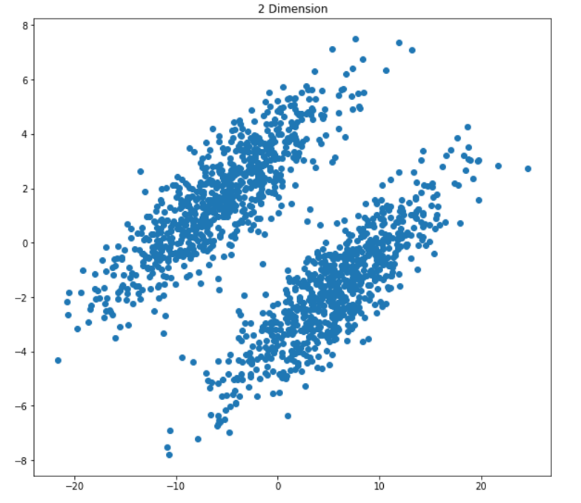
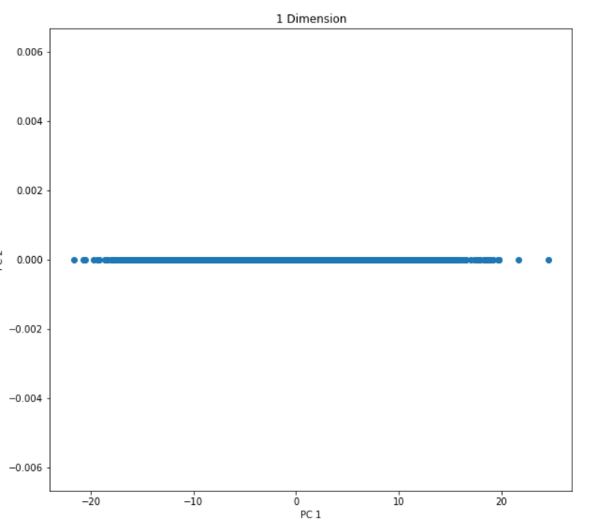
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



Sphere



Swissroll



Ellipsoids

1. Matrix

Sphere:

[[ 15045.16477116 13452.5115079 13631.98710877]

[ 13452.5115079 14876.92584157 13592.65574337]

[ 13631.98710877 13592.65574337 15383.70237092]]

Swissroll:

[[ 64889.0265442 230.10763148 6678.83066677]

[ 230.10763148 15859.39242309 231.42171507]

[ 6678.83066677 231.42171507 70685.01799753]]

Ellipsoids:

[[ 86741.32255707 22025.42123755 11042.18121689]

[ 22025.42123755 14784.6022647 6785.12588853]

[ 11042.18121689 6785.12588853 5032.15322094]]

1. Eigenvalues and eigenvectors

Sphere:

Values:

[ 42222.06446919 1584.5399621 1499.18855236]

Vectors:

[[-0.57604721 -0.62074794 0.53182855]

[-0.57310984 -0.15721136 -0.80425723]

[-0.58285051 0.76808632 0.26519558]]

Swissroll:

Values:

[ 75069.2129182 60506.65397977 15857.57006686]

Vectors:

[[ 0.54862373 0.83605889 0.00418753]

[ 0.00539983 0.00146519 -0.99998435]

[ 0.83605194 -0.54863776 0.00371074]]

Ellipsoids:

Values:

[ 94684.71190217 10321.58611625 1551.78002429]

Vectors:

[[ 0.95175893 0.30681239 0.00459253]

[ 0.27407997 -0.8433001 -0.46230412]

[ 0.13796775 -0.4412608 0.88670954]]

3.

Ratio(sphere 2 dimension) = (λ1 + λ2)/λi) = (42222.06446919+1584.5399621)/(42222.06446919+1584.5399621+1499.18855236) = 0.97>0.9

Ratio(sphere 1 dimension) = (λ1)/λi) = (42222.06446919)/(42222.06446919+1584.5399621+1499.18855236) = 0.93>0.9

Ratio(Swissroll 2 dimension) = (λ1 + λ2)/λi) = (75069.2129182+60506.65397977)/( 75069.2129182+60506.65397977+15857.57006686) = 0.89<0.9

Ratio(Swissroll 1 dimension) = (λ1)/λi) = (75069.2129182)/( 75069.2129182+60506.65397977+15857.57006686) = 0.50<0.9

Ratio(Ellipsoids 2 dimension) = (λ1 + λ2)/λi) = (94684.71190217+10321.58611625)/( 94684.71190217+10321.58611625+1551.78002429) = 0.98>0.9

Ratio(Ellipsoids 1 dimension) = (λ1)/λi) = (94684.71190217)/( 94684.71190217+10321.58611625+1551.78002429) = 0.89<0.9

If we define the ratio bigger than 0.9 is good, we can saw that both 1 and 2 dimension for sphere, 2 dimension for ellipsoids are good PCA, whereas the rest of them are not good.