Principal Component Analysis

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random as rd
```

1. Data

- the data are given by the file data-pca.txt
- the data consist of a set of points $\{(x_i, y_i)\}_{i=1}^{n}$ where $z_i = (x_i, y_i)$ denotes a 2-dimensional point in the cartesian coordinate

load the data from the files

```
path = '/content/drive/My Drive/ML_Assignment/data/data-pca.txt'
data = np.loadtxt(path, delimiter=',')
x = data[:,0]
y = data[:,1]
```

Plot the original data points

```
fig_1 = plt.figure(figsize = (6,6))
plt.scatter(x, y, c='r', marker = '+')
plt.title('original data points')
plt.show()
fig_1.savefig('original data points.png')
```

2. Normalization

- the data is normalized to have the mean = 0 and the standard deviation = 1
- $x = \frac{x - \mu_x}{\sigma_x}$ and $y = \frac{y - \mu_y}{\sigma_y}$

> - μ_x denotes the mean of x
> - σ_x denotes the standard deviation of x
> - μ_y denotes the mean of y
> - σ_y denotes the standard deviation of y

define a function to normalize the input data points x and y

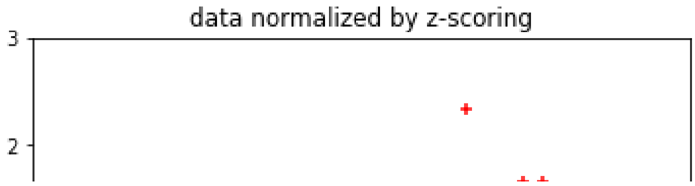
```
def normalize_data(x, y):

    xn = (x - x.mean(axis=0)) / x.std(axis=0) # normalize x. the mean of xn is zero and the standard devia
    yn = (y - y.mean(axis=0)) / y.std(axis=0) # normalize y. the mean of yn is zero and the standard devia

    return xn, yn
```

Plot the normalized data points

- $z = \frac{z - \mu}{\sigma}$
- μ denotes the average and σ denotes the standard deviation

```
xn, yn = normalize_data(x, y)
```

```
fig_2 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+')
plt.title('data normalized by z-scoring')
plt.axis([-3, 3, -3, 3])
plt.show()
fig_1.savefig('normalized data points.png')
```



3. Covariance Matrix

- compute the co-variance matrix
- $\Sigma = \frac{1}{n} \sum_{i=1}^{n} z_i z_i^T = \frac{1}{n} Z^T Z$

> - n denotes the number of data

> - $Z = \begin{bmatrix} z_1^T \\ \vdots \\ z_n^T \end{bmatrix}$

define a function to compute the co-variance matrix of the data

```
def compute_covariance(z):
    # compute the covariance matrix #
    covar = np.cov(z.T)
    return covar
```

```
Z = np.c_[xn,yn]
covariance = compute_covariance(Z)  # return 2x2 metrix
```

4. Principal Components

- compute the eigen-values and the eigen-vectors of the co-variance matrix

define a function to compute the principal directions from the co-variance matrix

```
def compute_principal_direction(covariance):

    e_value, e_vector = np.linalg.eig(covariance) # compute the principal directions from the co-variance

    return e_value, e_vector
```

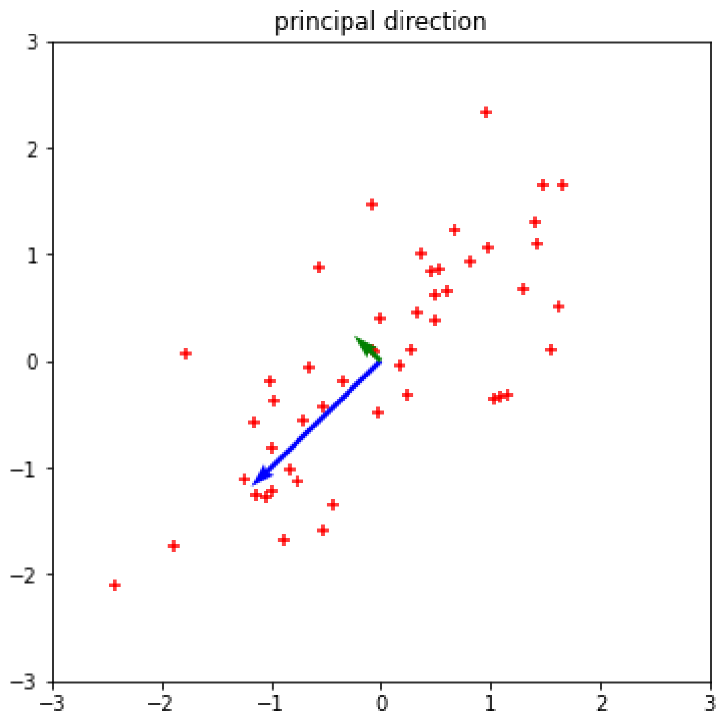
```
e_value, e_vector = compute_principal_direction(covariance)
```

```
# 내림차순 정렬
idx = np.flip(e_value.argsort())
e_value = e_value[idx]
e_vector = e_vector[:, idx]
```

Plot the principal axes

- plot the normalized data points
- plot the first principal vector
- plot the second principal vector

```
fig_3 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+')
plt.quiver([0], [0], e_vector[0, 0], e_vector[1, 0], color=['b'],angles='xy', scale_units='xy', scale=0.6)
plt.quiver([0], [0], e_vector[0, 1], e_vector[1, 1], color=['g'],angles='xy', scale_units='xy')
plt.title('principal direction')
plt.axis([-3, 3, -3, 3])
plt.show()
fig_3.savefig('principal direction.png')
```



Make LInear example

```
def VectorToLinear(vector):
  a = vector[1]/vector[0]
  test_x = np.arange(-2.8, 2.8, 0.1)
  test_y = a * test_x
  return test_x, test_y
```

define a function to compute the projection of the data point onto the principal axis

```
def compute_projection(point, axis):
    # compute the projection of point on the axis #
    pca = np.dot(point, axis)
    trans_e_vector = axis[:,None].T
    projection = np.dot(pca[:,None], trans_e_vector)
    return projection
```

Plot Output principal Axis

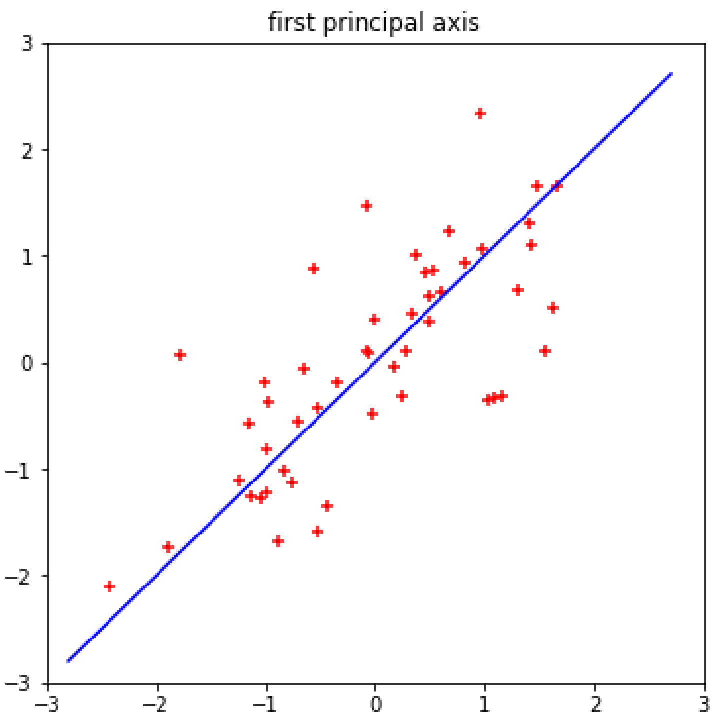
First principal axis

Plot the first principal axis

- plot the normalized data points
- plot the first principal axis

```
first_x , first_y = VectorToLinear(e_vector[:,0].T)
```

```
fig_4 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+')
plt.title('first principal axis')
plt.plot(first_x, first_y, c='b')
plt.axis([-3, 3, -3, 3])
plt.show()
fig_4.savefig('first principal axis.png')
```



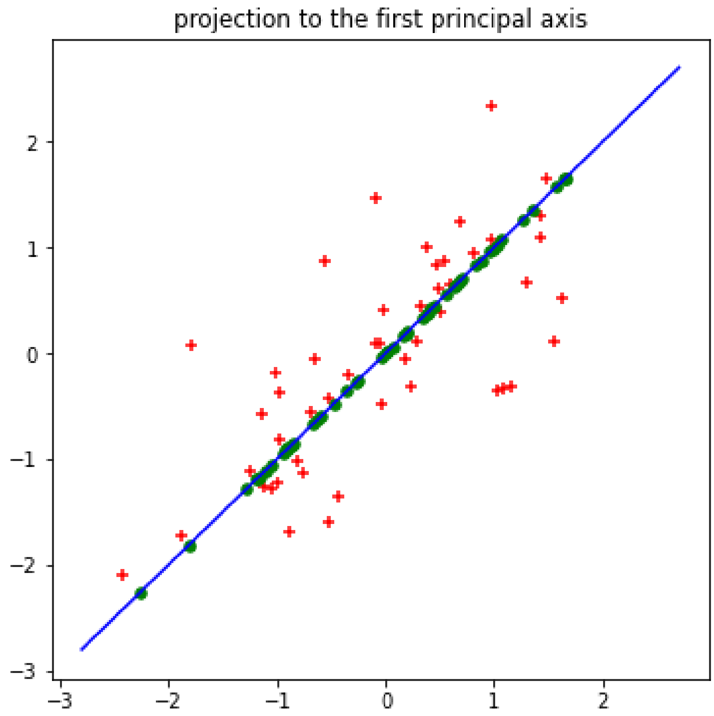

```
first_pca = compute_projection(Z, e_vector[:,0])
```

Plot the project of the normalized data points onto the first principal axis

- plot the normalized data points
- plot the first principal axis
- plot the projected points from the normalized data points onto the first principal axis

```
fig_5 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+',s=30)
```

```
zero = np.zeros(50)
plt.plot(first_x, first_y, c='b')
plt.scatter(first_pca[:,0], first_pca[:,1], c = 'g', s=30)
plt.title('projection to the first principal axis')
plt.show()
fig_5.savefig('projection to the first principal.png')
```

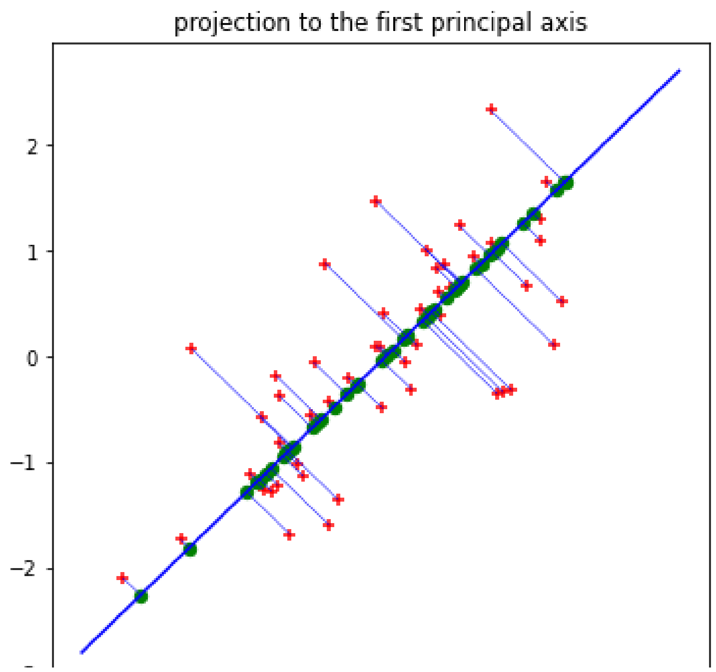


Plot the lines between the normalized data points and their projection points on the first principal axis

- plot the normalized data points
- plot the first principal axis
- plot the projected points from the normalized data points onto the first principal axis
- plot the lines that connect between the normalized data points and their projection points on the first principal axis

```
# 유클리드 좌표계에서 점사이의 거리
def compute_distance(x1, y1, x2, y2):
    distance = np.array([[x1, y1],[x2, y2]])
    return distance
```

```
fig_6 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+',s=30)
zero = np.zeros(50)
plt.plot(first_x, first_y, c='b')
plt.scatter(first_pca[:,0], first_pca[:,1], c = 'g', s=40)
for i in range (50):
    dist = np.array([[xn[i], yn[i]],[first_pca[i,0], first_pca[i,1]]])
    plt.plot(dist[:,0], dist[:,1], c = 'b', linewidth = 0.5)
plt.title('projection to the first principal axis')
plt.show()
fig_6.savefig('distance to the first projection axis.png')
```



Second principal axis

Plot the second principal axis

- plot the normalized data points
- plot the second principal axis

```
second_x , second_y = VectorToLinear(e_vector[:,1].T)
```

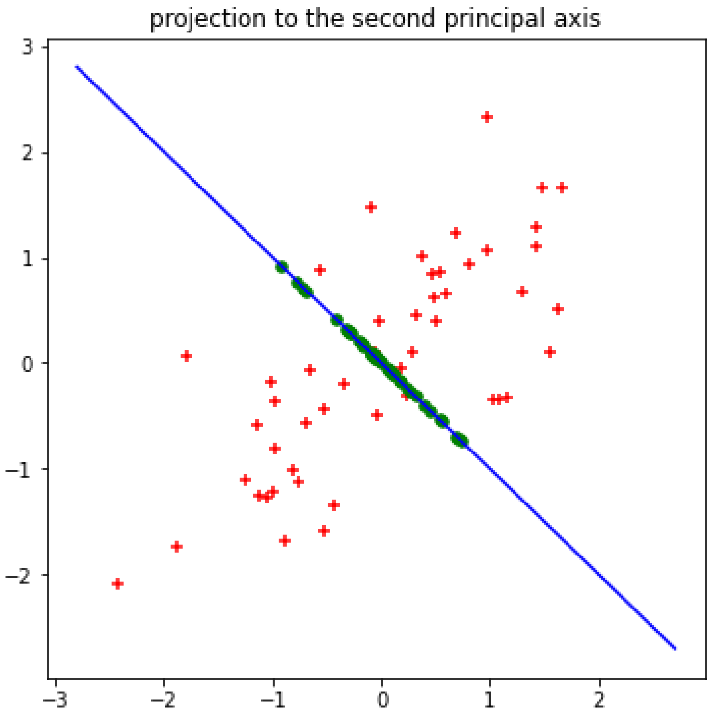
```
fig_7 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+')
plt.title('second principal axis')
plt.plot(second_x , second_y, c='b')
plt.axis([-3, 3, -3, 3])
plt.show()
fig_7.savefig('second principal axis.png')
```

Plot the project of the normalized data points onto the second principal axis

- plot the normalized data points
- plot the second principal axis
- plot the projected points from the normalized data points onto the second principal axis

```
second_pca = compute_projection(Z, e_vector[:,1])
```

```
fig_8 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+',s=30)
zero = np.zeros(50)
plt.plot(second_x , second_y, c='b')
plt.scatter(second_pca[:,0], second_pca[:,1], c = 'g', s=30)
plt.title('projection to the second principal axis')
plt.show()
fig_8.savefig('projection to the second principal.png')
```

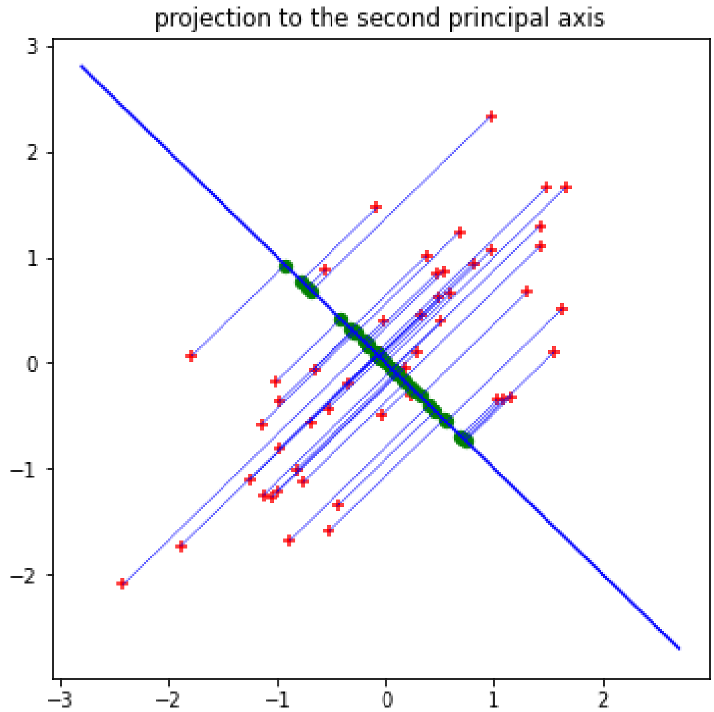


Plot the lines between the normalized data points and their projection points on the second principal axis

- plot the normalized data points
- plot the second principal axis
- plot the projected points from the normalized data points onto the second principal axis
- plot the lines that connect between the normalized data points and their projection points on the second principal axis

```
fig_9 = plt.figure(figsize = (6,6))
plt.scatter(xn, yn, c='r', marker = '+',s=30)
zero = np.zeros(50)
plt.plot(second_x, second_y, c='b')
plt.scatter(second_pca[:,0], second_pca[:,1], c = 'g', s=40)
for i in range (50):
```

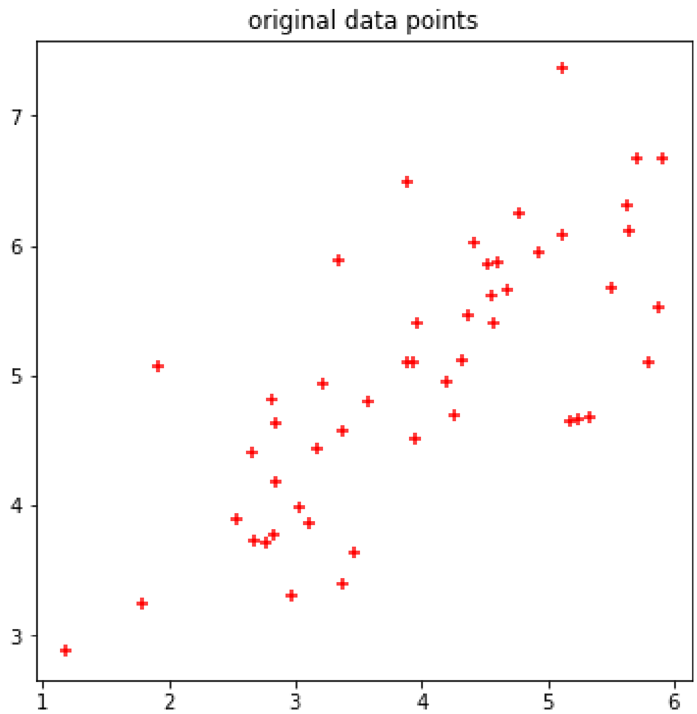
```
    dist = np.array([[xn[i], yn[i]],[second_pca[i,0], second_pca[i,1]]])
    plt.plot(dist[:,0], dist[:,1], c = 'b', linewidth = 0.5)
plt.title('projection to the second principal axis')
plt.show()
fig_9.savefig('distance to the second projection axis.png')
```



Output

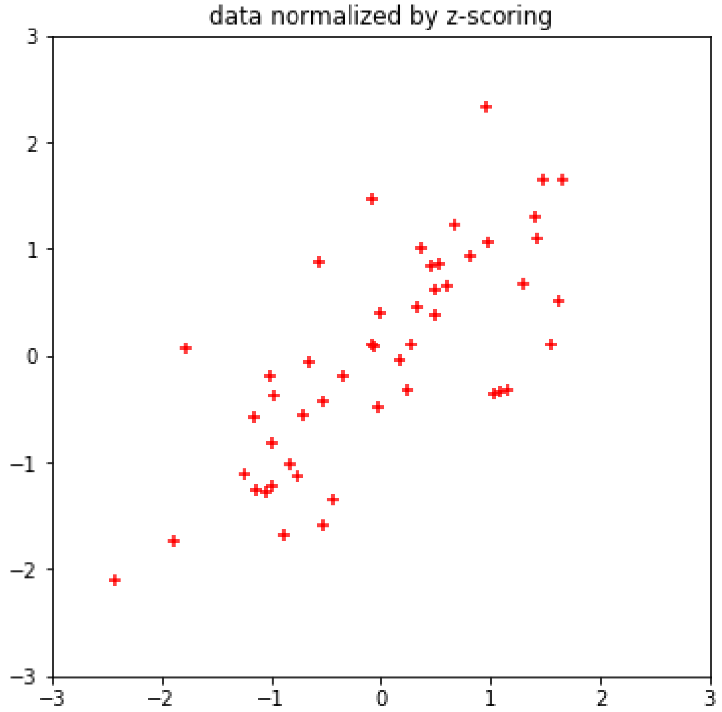
1. Plot the original data points [1pt]

```
fig_1
```



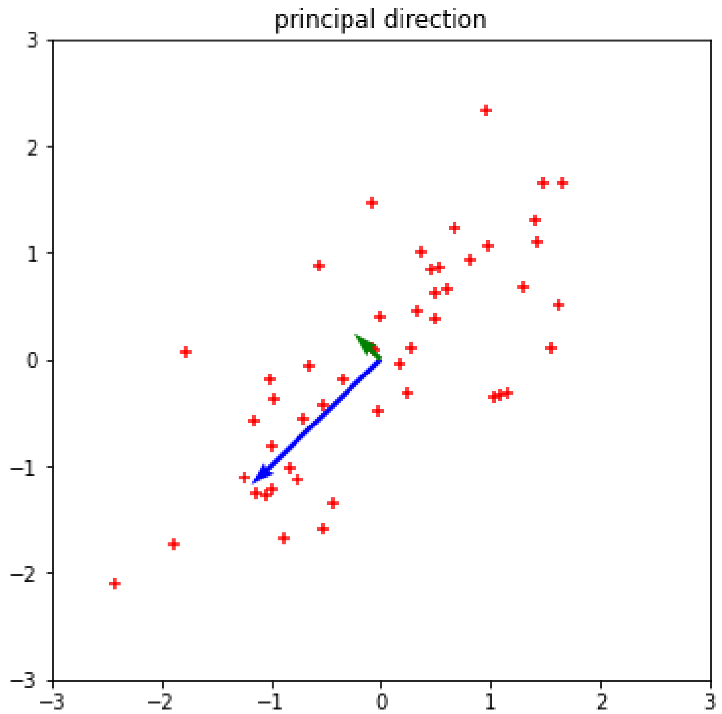
2. Plot the normalized data points [1pt]

```
1    fig_2
```



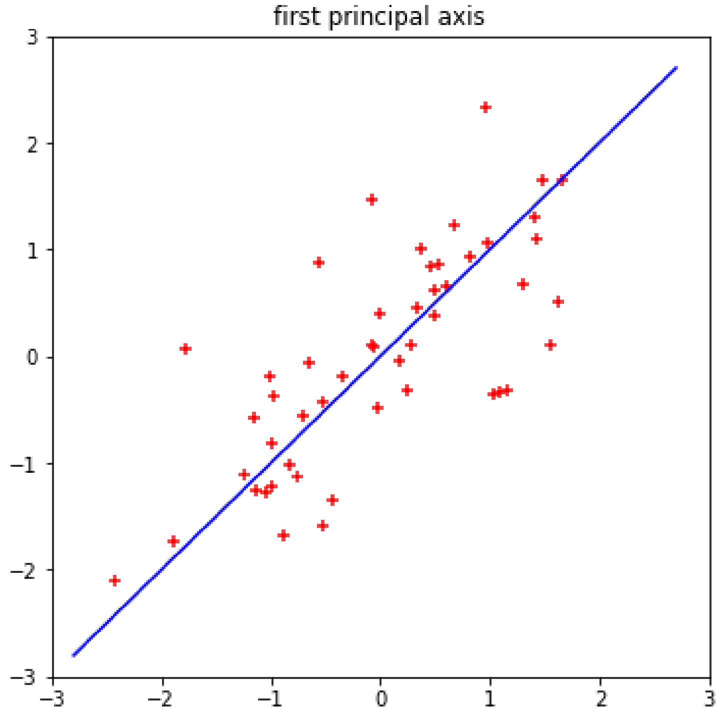
3. Plot the principal axes [2pt]

```
1    fig_3
```



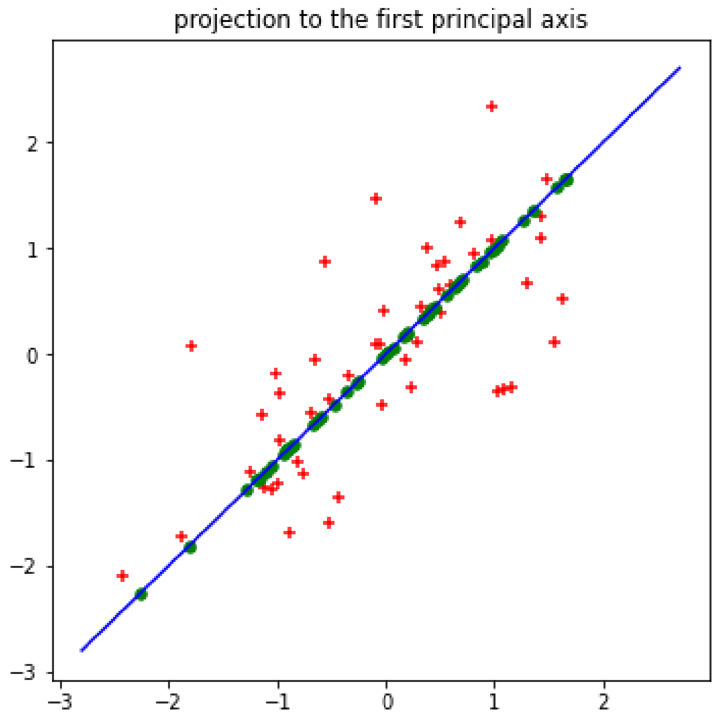
4. Plot the first principal axis [3pt]

```
fig_4
```



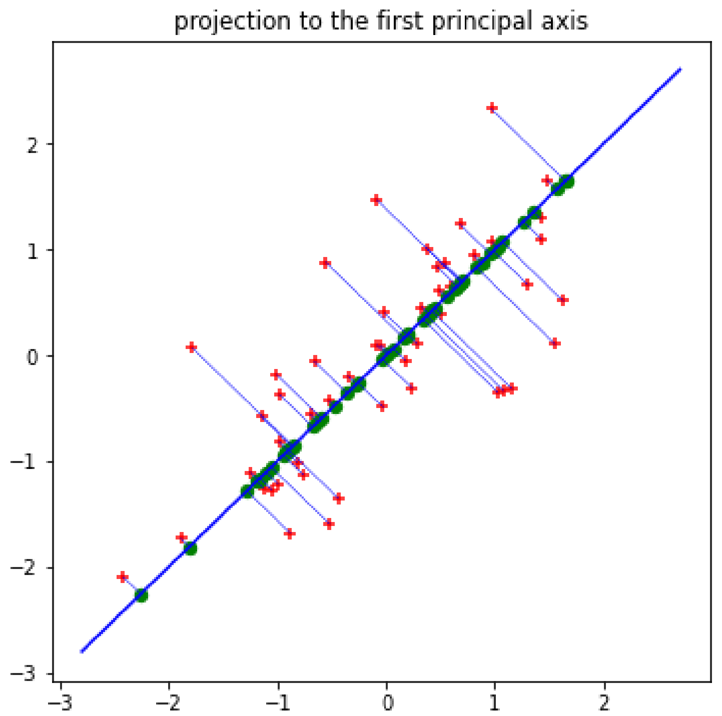
5. Plot the project of the normalized data points onto the first principal axis [4pt]

```
fig_5
```



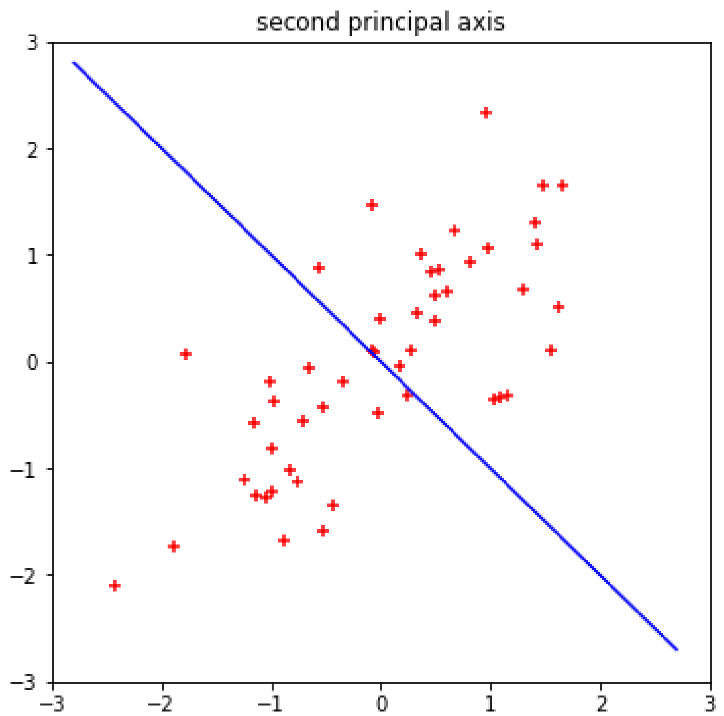
6. Plot the lines between the normalized data points and their projection points on the first principal axis [3pt]

```
fig_6
```



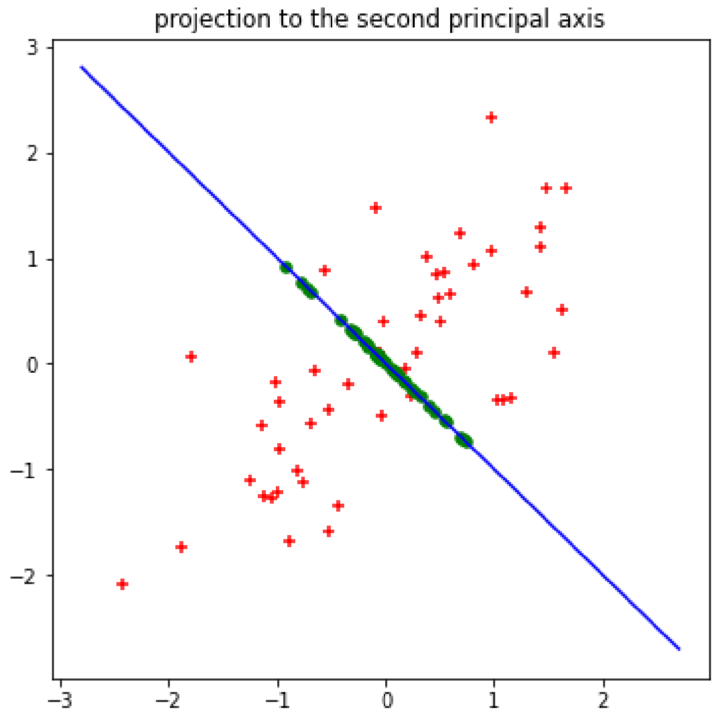
7. Plot the second principal axis [3pt]

```
fig_7
```



8. Plot the project of the normalized data points onto the second principal axis [4pt]

```
fig_8
```



9. Plot the lines between the normalized data points and their projection points on the second principal axis [3pt]

```
1 fig_9
```