HW8

108034058

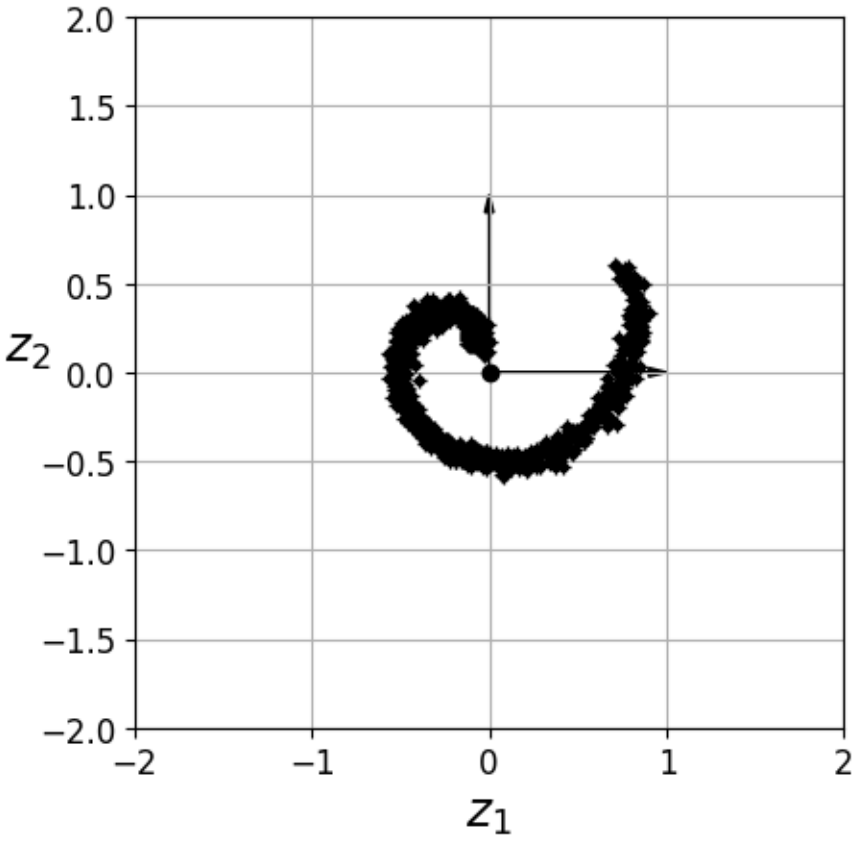
鍾岷翰

1.

(a)

```
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
# %matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

np.random.seed(4)
m = 1000
noise = 0.1

angles = np.random.rand(m) * random.uniform(0,2*np.pi)
X = np.empty((m, 3))
X[:, 0] = angles/6.28*np.cos(angles) + noise *
np.random.randn(m)  * random.gauss(0,1)
X[:, 1] = angles/6.28*np.sin(angles) + noise *
np.random.randn(m)  * random.gauss(0,1)
X[:, 2] =  noise * np.random.randn(m)* random.gauss(0,1)
```

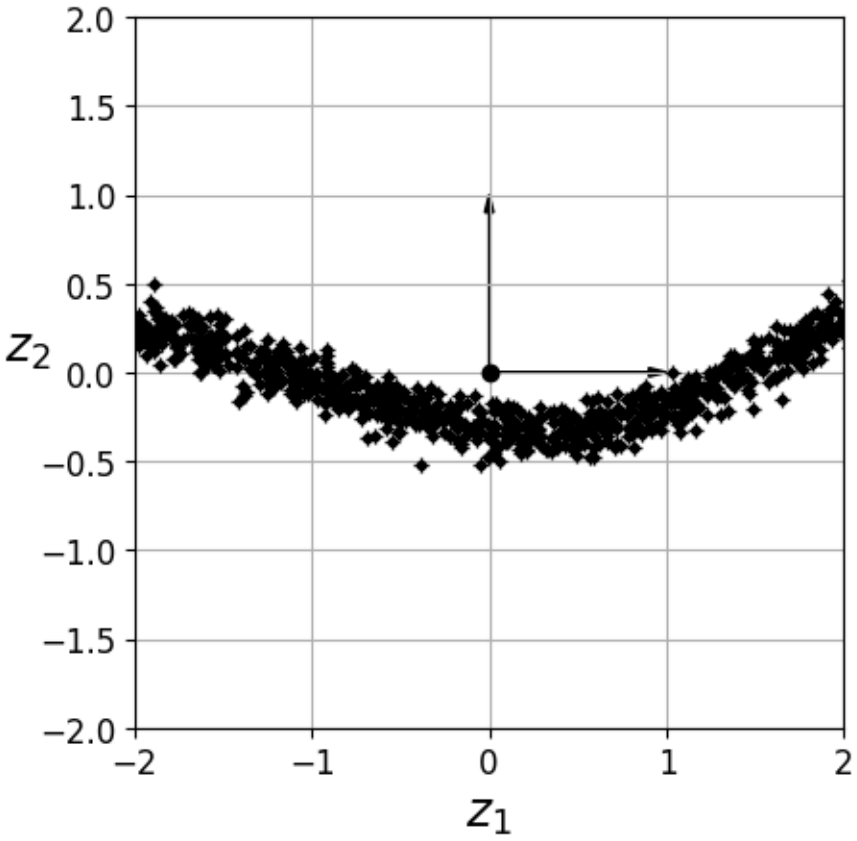
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X2D = pca.fit_transform(X)


fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal')

ax.plot(X2D[:, 0], X2D[:, 1], "k+")
ax.plot(X2D[:, 0], X2D[:, 1], "k.")
ax.plot([0], [0], "ko")
ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True,
head_length=0.1, fc='k', ec='k')
ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True,
head_length=0.1, fc='k', ec='k')
ax.set_xlabel("$z_1$", fontsize=18)
ax.set_ylabel("$z_2$", fontsize=18, rotation=0)
ax.axis([-2, 2, -2, 2])
ax.grid(True)
plt.show()
```



(b)

```
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
# %matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

np.random.seed(4)
m = 1000
noise = 0.1

angles = np.random.rand(m) * random.uniform(0,2*np.pi)
X = np.empty((m, 3))
X[:, 0] = angles/6.28*np.cos(angles) + noise *
np.random.randn(m)  * random.gauss(0,1)
X[:, 1] = angles/6.28*np.sin(angles) + noise *
np.random.randn(m)  * random.gauss(0,1)
X[:, 2] =  angles*1 + noise * np.random.randn(m)*
random.gauss(0,1)


from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X2D = pca.fit_transform(X)


fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal')

ax.plot(X2D[:, 0], X2D[:, 1], "k+")
ax.plot(X2D[:, 0], X2D[:, 1], "k.")
ax.plot([0], [0], "ko")
ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True,
head_length=0.1, fc='k', ec='k')
ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True,
head_length=0.1, fc='k', ec='k')
ax.set_xlabel("$z_1$", fontsize=18)
ax.set_ylabel("$z_2$", fontsize=18, rotation=0)
ax.axis([-2, 2, -2, 2])
ax.grid(True)
plt.show()
```



2.

```
#HW8_2
# This example is John Gutag's book
# the code is used to create clusters
# first we generate two cluster with mean and sigma
# second we apply k means cluster
# for simplicity we only use two clusters but the idea can be
generalized
# the original code is written in class and object-oriented
# here we use more procedure type programming

import random
import pandas as pd

x1_record=[]
x2_record=[]
```

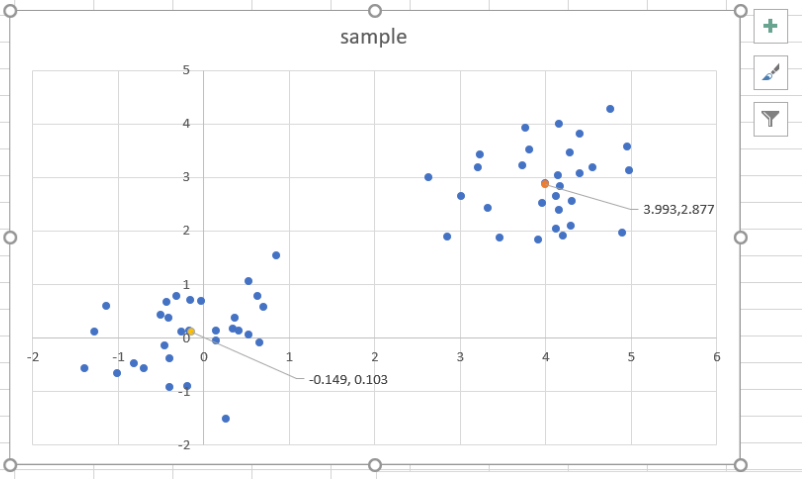
```
x1=0.0
x2=0.0
y1=0.0
y2=0.0
number_sample=30
#generate two clusters decorated with gaussian noise
for i in range(number_sample):
    x1=random.gauss(0,0.6) #create random number gauss (mean,
sigma)
    x2=random.gauss(0,0.6)
    x1_record.append(float(x1))
    x2_record.append(float(x2))
for i in range(number_sample):
    y1=4.0+random.gauss(0,0.6)
    y2=3.0+random.gauss(0,0.6)
    x1_record.append(float(y1))
    x2_record.append(float(y2))

number = len(x1_record)
print(number)
#choose two cluster
cluster1= random.choice(range(0,number))
cluster2= random.choice(range(0,number))

centroid1x= x1_record[cluster1]
centroid1y= x2_record[cluster1]
centroid2x= x1_record[cluster2]
centroid2y= x2_record[cluster2]
print('initial choice',round(centroid1x,3), round(centroid1y,3),
round(centroid2x,3), round(centroid2y,3))

index=[]
for j in range(number):
    index.append('0') # create a zero index tthis index record
which cluster
                      #the data point is associated with

#run over all the sample and compute and compare the distance
```

```
for j in range(number):
    distance_to_cluster1= (centroid1x-
x1_record[j])**2+(centroid1y-x2_record[j])**2
    distance_to_cluster2= (centroid2x-
x1_record[j])**2+(centroid2y-x2_record[j])**2
    if distance_to_cluster1>distance_to_cluster2:
        index[j]=2
    else:
        index[j]=1

centroid_1_x=0.0  # index rule index for cluster
centroid_1_y=0.0
centroid_2_x=0.0
centroid_2_y=0.0
for iteration in range(10):
    sum_1_x=0.0
    sum_1_y=0.0
    sum_2_x=0.0
    sum_2_y=0.0
    count_1=0
    count_2=0
    for j in range(number):
        if index[j]==1:
            sum_1_x=sum_1_x+x1_record[j]
            sum_1_y=sum_1_y+x2_record[j]
            count_1=count_1+1
        elif index[j]==2:
            sum_2_x=sum_2_x+x1_record[j]
            sum_2_y=sum_2_y+x2_record[j]
            count_2=count_2+1
        else:
            print('error index') #for trouble shooting
    centroid_1_x=sum_1_x/count_1
    centroid_1_y=sum_1_y/count_1
    centroid_2_x=sum_2_x/count_2
    centroid_2_y=sum_2_y/count_2
```

```
    print('step', iteration, '1 x',round(centroid_1_x,3), '1
y',round(centroid_1_y,3),'2 x',round(centroid_2_x,3),'2
y',round(centroid_2_y,3))

    for j in range(number):
        distance_to_cluster1= (centroid_1_x-
x1_record[j])**2+(centroid_1_y-x2_record[j])**2
        distance_to_cluster2= (centroid_2_x-
x1_record[j])**2+(centroid_2_y-x2_record[j])**2
        if distance_to_cluster1>distance_to_cluster2:
            index[j]=2
        else:
            index[j]=1
df = pd.DataFrame(columns=['sample', 'x1', 'x2'])
for j in range(number):
    df = df.append({"sample": j+1 , "x1":round(x1_record[j],3),
"x2":round(x2_record[j],3)},ignore_index=True)
    print( 'sample #', j,
round(x1_record[j],3),round(x2_record[j],3))

df.to_csv('/content/drive/MyDrive/HW8_2.csv', index=False)
```

(a)

sample	x1	x2
1	-0.503	0.434
2	-0.188	-0.91
3	-0.399	-0.92
4	0.342	0.165
5	-0.156	0.696
6	0.525	1.057
7	-1.275	0.121
8	-0.395	-0.394
9	-1.014	-0.66
10	-0.818	-0.488
11	0.151	-0.051
12	-0.46	-0.153
13	0.706	0.583
14	-0.031	0.695
15	0.364	0.365
16	-0.259	0.106
17	0.262	-1.514
18	-0.312	0.782
19	-0.165	0.137
20	0.854	1.546



(b)

$$\text{midpoint} = (1.992, 1.49)$$

$$m = -1.493$$

$$y = -1.493(x - 1.992) + 1.49$$

3.

$L(x,y,\lambda) = f(x,y) - \lambda g(x,y)$

$f(x,y) = x^2 + 2y \qquad g(x,y) = 3x + 2y + 1$

$L(x,y,\lambda) = x^2 + 2y - \lambda(3x + 2y + 1)$

$\frac{\partial L}{\partial x} = 2x - 3\lambda = 0 \qquad x = \frac{3}{2}\lambda$

$\frac{\partial L}{\partial y} = 2 - 2\lambda = 0 \qquad \lambda = 1 \qquad x = \frac{3}{2}$

$\frac{9}{2} + 2y + 1 = 0 \qquad y = -\frac{11}{4}$

$\frac{\partial L}{\partial \lambda} = -(3x + 2y + 1) = 0$

critical point is $(\frac{3}{2}, -\frac{11}{4})$ $\lambda = 1$