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')

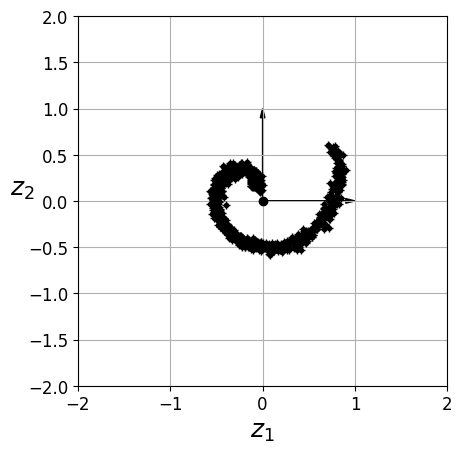
ax.set\_xlabel("$z\_1$", fontsize=18)

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')

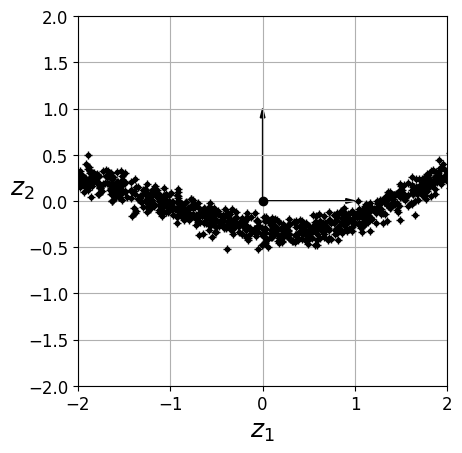
ax.set\_xlabel("$z\_1$", fontsize=18)

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= 1.0+random.gauss(0,0.6)

    y2= 1.0+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)

(b)

3.

