**附錄**

**程式說明：**

本程式利用Python編寫，Vs code編寫器，格式為Jupiter Notebook，未顯示各段輸出。

import numpy as np

import random

def generate(n):

    data=[]

    y=random.choice([-1, 1])

    if y==1:

        mean = (2, 3)

        cov = [[0.6, 0], [0, 0.6]]

        data = np.random.multivariate\_normal(mean, cov, 1, 'raise')

        data = np.c\_[data,np.ones(1)]

    if y==-1:

        mean = (0, 4)

        cov = [[0.4, 0], [0, 0.4]]

        data = np.random.multivariate\_normal(mean, cov, 1, 'raise')

        data = np.c\_[data,np.ones(1)\*-1]

    for i in range(n-1):

        y=random.choice([-1, 1])

        if y==1:

            mean = (2, 3)

            cov = [[0.6, 0], [0, 0.6]]

            data = np.append(data,np.c\_[np.random.multivariate\_normal(mean, cov, 1, 'raise'),np.ones(1)],0)

        if y==-1:

            mean = (0, 4)

            cov = [[0.4, 0], [0, 0.4]]

            data = np.append(data,np.c\_[np.random.multivariate\_normal(mean, cov, 1, 'raise'),np.ones(1)\*-1],0)

    X = data[:, :-1]

    y = data[:, -1]

    return X, y

n=200

X, y = generate(n)

#Problem 13 与 Problem 14

m=100

Ein = np.array([])

#Eout = np.array([])

for i in range(m):

    X, y = generate(n)#運行Problem 13時設為200，運行Problem14時設為5000

    X = np.c\_[np.ones(n), X]

    w = inv(X.T.dot(X)).dot(X.T).dot(y)

    ein = np.mean(np.sign(X.dot(w) \* y) < 0 )

    Ein = np.append(Ein, ein)

'''

Eout = np.array([])

for i in range(m):

    n=5000

    X, y = generate(n)#運行Problem 13時設為200，運行Problem14時設為5000

    X = np.c\_[np.ones(n), X]

    w = inv(X.T.dot(X)).dot(X.T).dot(y)

    eout = np.mean(np.sign(X.dot(w) \* y) < 0 )

    Eout = np.append(Eout, eout)

'''

print(np.average(w))#13

print("\n")

#print(np.average(Ein)-np.average(Eout))#14

#這是因為Ein和Eout兩者之差的平均值等於兩者的各自平均值相減

#Problem 15

def preprocess(X):

    """

    添加偏置項

    """

    n = X.shape[0]

    return np.c\_[np.ones(n), X]

#定義函式

def sigmoid(s):

    return 1 / (np.exp(-s) + 1)

def gradient(X, w, y):

    temp1 = - X.dot(w) \* y

    temp2 = sigmoid(temp1)

    temp3 = - X \* y

    grad = np.mean(temp3 \* temp2, axis=0).reshape(-1, 1)

    return grad

for i in range(100):

    X, y = generate(200)

    X\_train=X

    y\_train=y.reshape(-1, 1)

    X\_train=preprocess(X)

    X1, y1 = generate(5000)

    X\_test=X1

    y\_test=y1.reshape(-1, 1)

    X\_test=preprocess(X1)

    #資料的組數和維度

    n, m = X\_train.shape

    n1, m1 = X\_test.shape

    #print(n,n1)

    #print(m,m1)

    w = np.zeros((m, 1))

    k = 0.1

    for i in range(500):

        grad = gradient(X\_train, w, y\_train)

        w -= k \* grad

#计算标签

    y\_test\_pred = X\_test.dot(w)

    y\_test\_pred[y\_test\_pred > 0] = 1

    y\_test\_pred[y\_test\_pred <= 0] = -1

Eout = np.mean(y\_test\_pred != y\_test)

print(Eout)

print(w)

w = np.zeros((m, 1))

k = 0.1

for i in range(500):

    grad = gradient(X\_train, w, y\_train)

    w -= k \* grad

#计算标签

y\_test\_pred = X\_test.dot(w)

y\_test\_pred[y\_test\_pred > 0] = 1

y\_test\_pred[y\_test\_pred <= 0] = -1

#计算Eout

Eout = np.mean(y\_test\_pred != y\_test)

#求出误差

print(Eout)

print(w)

#Problem 16

def generate2(n):

    data=[]

    y=random.choice([-1, 1])

    if y==1:

        mean = (2, 3)

        cov = [[0.6, 0], [0, 0.6]]

        data = np.random.multivariate\_normal(mean, cov, 1, 'raise')

        data = np.c\_[data,np.ones(1)]

    if y==-1:

        mean = (0, 4)

        cov = [[0.4, 0], [0, 0.4]]

        data = np.random.multivariate\_normal(mean, cov, 1, 'raise')

        data = np.c\_[data,np.ones(1)\*-1]

    for i in range(n-1):

        y=random.choice([-1, 1])

        if y==1:

            mean = (2, 3)

            cov = [[0.6, 0], [0, 0.6]]

            data = np.append(data,np.c\_[np.random.multivariate\_normal(mean, cov, 1, 'raise'),np.ones(1)],0)

        if y==-1:

            mean = (0, 4)

            cov = [[0.4, 0], [0, 0.4]]

            data = np.append(data,np.c\_[np.random.multivariate\_normal(mean, cov, 1, 'raise'),np.ones(1)\*-1],0)

    data1=[]

    y=random.choice([1])

    mean = (6, 0)

    cov = [[0.3, 0], [0, 0.1]]

    data1 = np.random.multivariate\_normal(mean, cov, 1, 'raise')

    data1 = np.c\_[data1,np.ones(1)]

    for i in range(19):

        mean = (6, 0)

        cov = [[0.3, 0], [0, 0.1]]

        data1 = np.append(data1,np.c\_[np.random.multivariate\_normal(mean, cov, 1, 'raise'),np.ones(1)],0)

    data=np.append(data,data1,0)

    X = data[:, :-1]

    y = data[:, -1]

    return X, y

    #return data

print(generate2(200))

m=100

Ein = np.array([])

#Eout = np.array([])

for i in range(m):

    X, y = generate2(200)#運行Problem 13時設為200，運行Problem14時設為5000

    #n=220

    X = np.c\_[np.ones(220), X]

    w = inv(X.T.dot(X)).dot(X.T).dot(y)

    ein = np.mean(np.sign(X.dot(w) \* y) < 0 )

    Ein = np.append(Ein, ein)

print(np.average(Ein))#Problem 16的解1

for i in range(100):

    X, y = generate2(200)

    X\_train=X

    y\_train=y.reshape(-1, 1)

    X\_train=preprocess(X)

    X1, y1 = generate(5000)

    X\_test=X1

    y\_test=y1.reshape(-1, 1)

    X\_test=preprocess(X1)

    #資料的組數和維度

    n, m = X\_train.shape

    n1, m1 = X\_test.shape

    #print(n,n1)

    #print(m,m1)

    w = np.zeros((m, 1))

    k = 0.1

    for i in range(500):

        grad = gradient(X\_train, w, y\_train)

        w -= k \* grad

#计算标签

    y\_test\_pred = X\_test.dot(w)

    y\_test\_pred[y\_test\_pred > 0] = 1

    y\_test\_pred[y\_test\_pred <= 0] = -1

Eout = np.mean(y\_test\_pred != y\_test)

print(Eout)

print(w)