1(B)

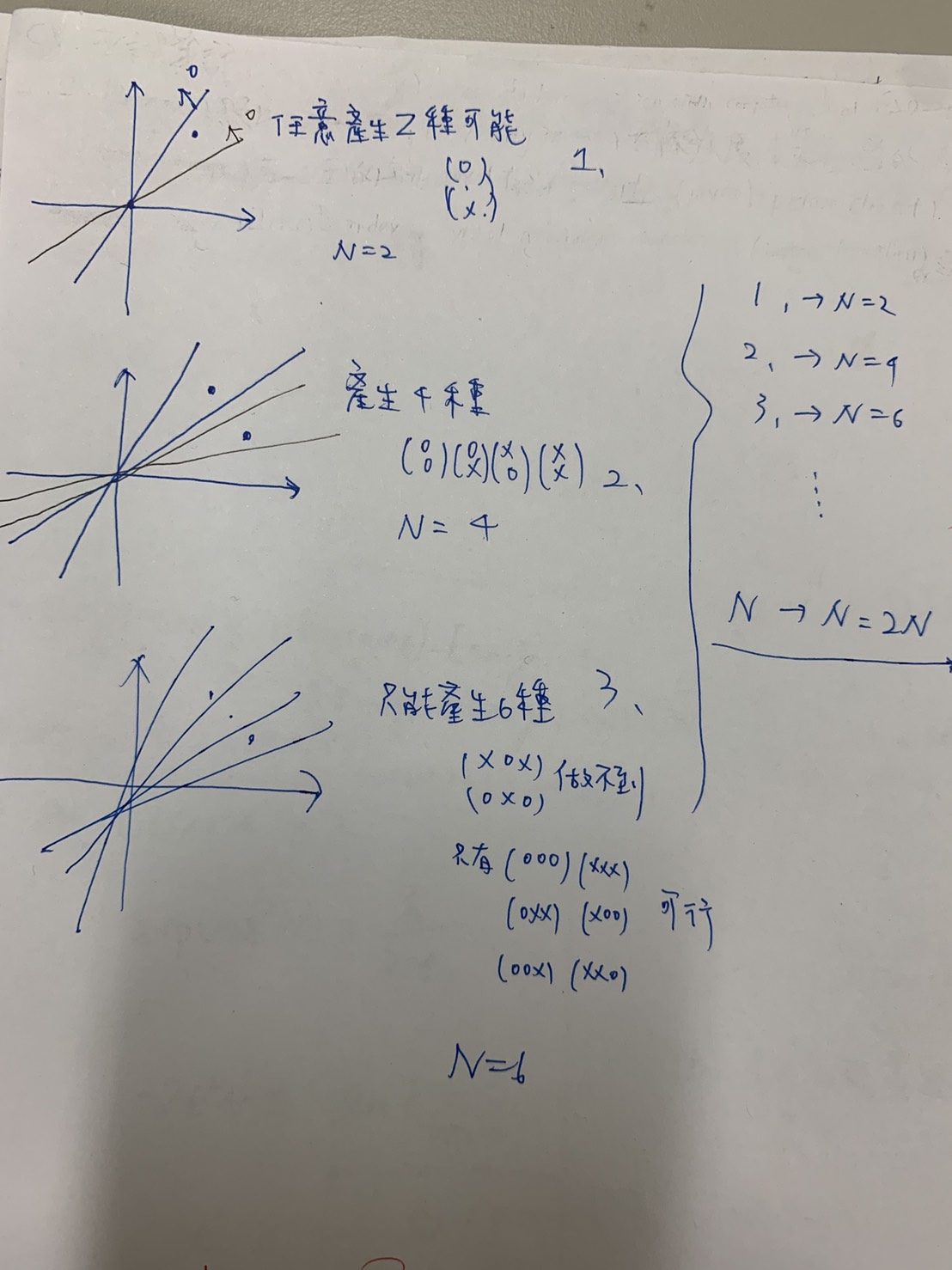
(A)共線(3-t,3,3+t)兩個點才能shatter

(B)空間中不共平面的4點 可以shatter

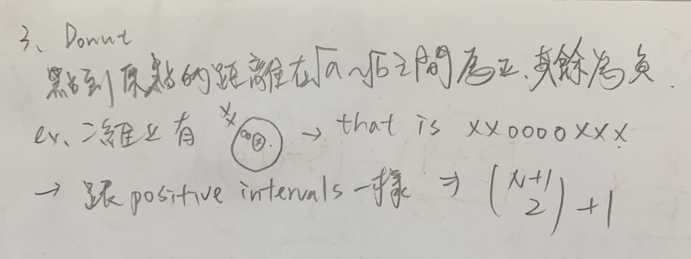
(C)共平面(2x-4y+2z=0) 又(111,222)(432,234)共線且不平行=2D 4個點，no shatter

(D) 空間中不共平面的5點 但5>d+1=4 no shatter

2.(C)



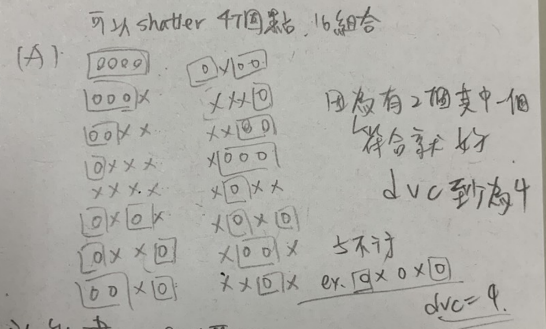
3.(A)



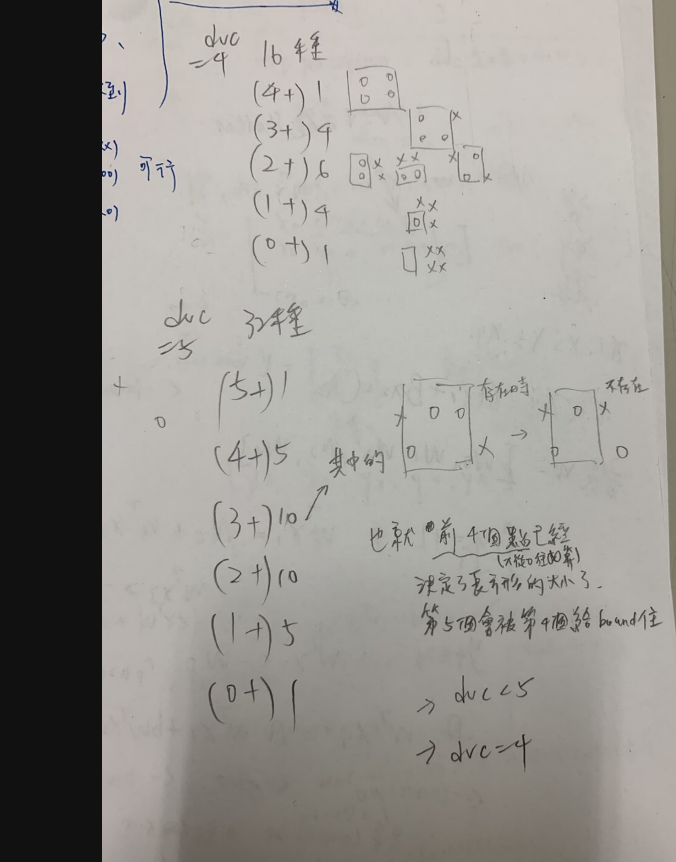
4.(D)

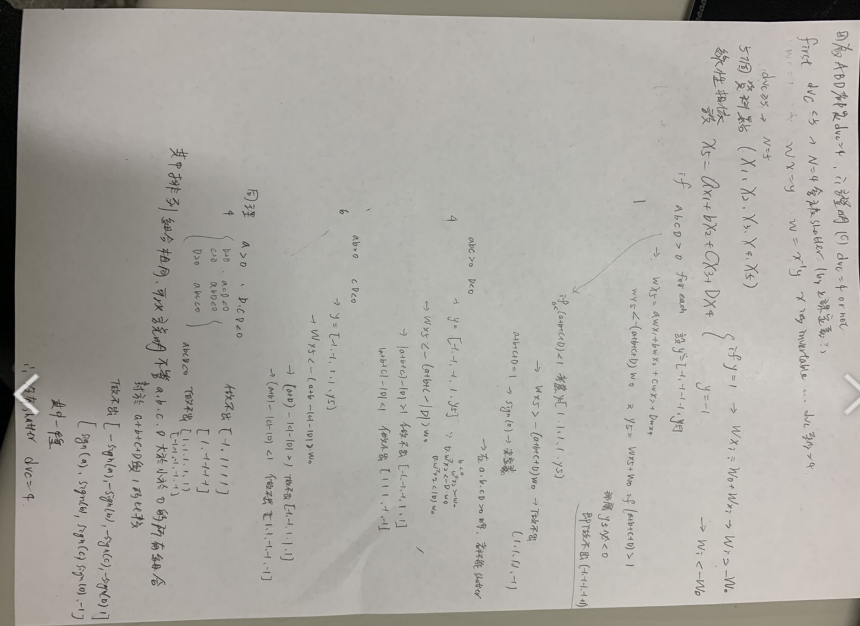
Positive interval🡪dvc = 2

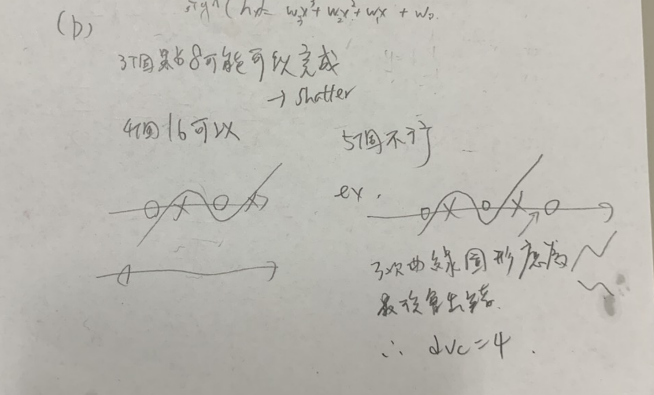
5. (E)

(A)

(B)

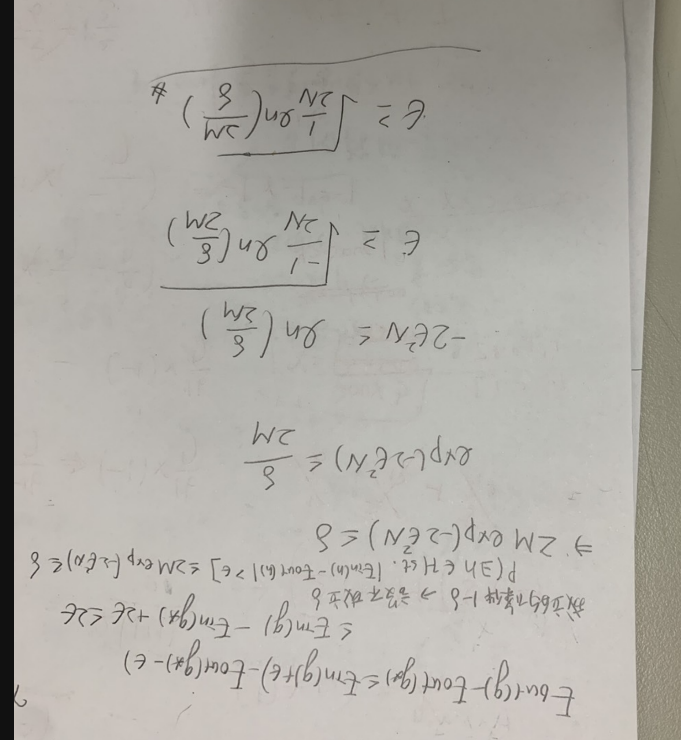


(C)

(D)

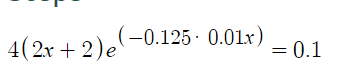
6. (D)

有限個Hypothesis set ? 1126最多做出2^10=1024 個 所以N=10

7.(C)

8.(B)

帶數字進去x=11000



9.(B)

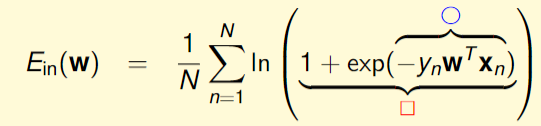
w-u =v 帶入原式得到🡪b(u)\*v +0.5\*v^2\*A(u)

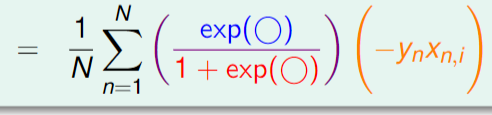
對v微分🡪b(u)+A(u)v = 0

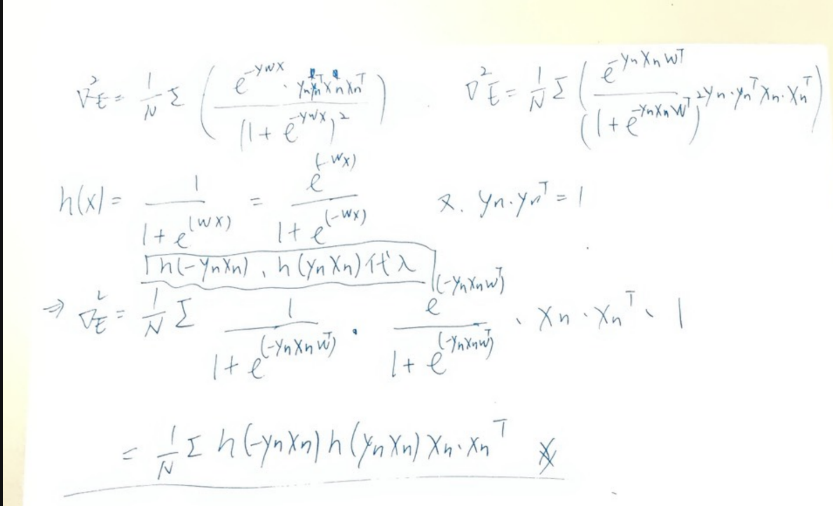
V = -A(u)^-1\*b(u)

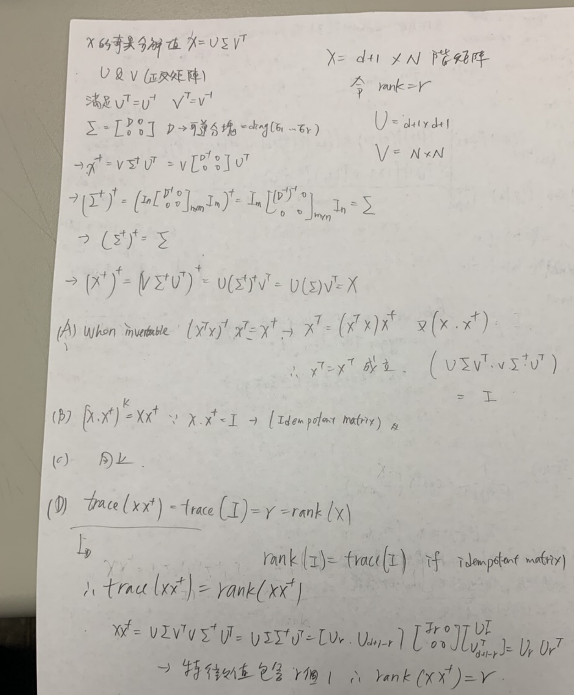
10.(D)

對E做w的2次微分

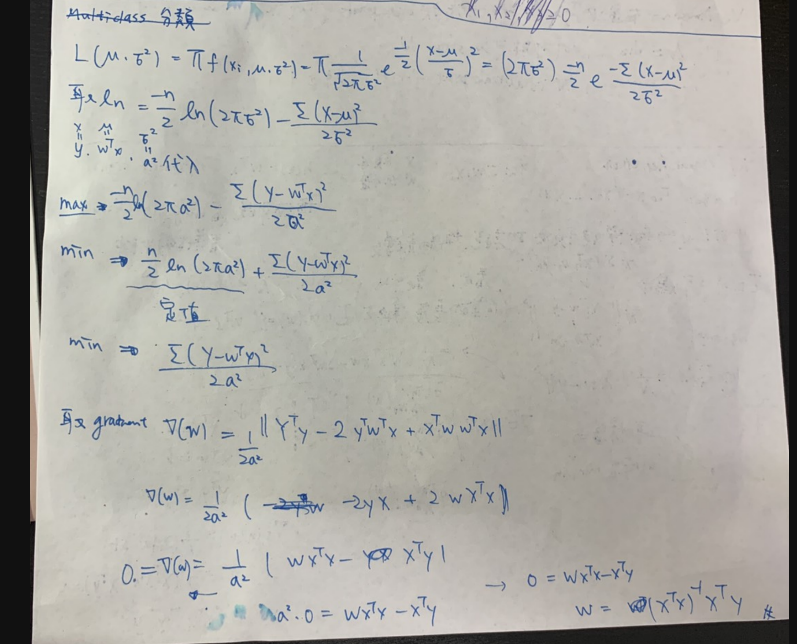
E logistic =

E 一次微分

 E 二次微分 =

11.(E)

12(A)



Q13----Q16

import numpy as np

import random

import math

import matplotlib.pyplot as plt

def flipcointogetdata(n):

# y = 0 -----> y = -1

data\_x =[]

data\_y =[]

for i in range (n):

#set random seed

random.seed()

y = random.randint(0,1)

if y == 1 :

x1 = random.gauss(2, np.sqrt(0.6))

x2 = random.gauss(3, np.sqrt(0.6))

data\_x.append([1,x1,x2])

data\_y.append(1)

elif y ==0:

x1 = random.gauss(0, np.sqrt(0.4))

x2 = random.gauss(4, np.sqrt(0.4))

data\_x.append([1,x1,x2])

data\_y.append(-1)

return data\_x ,data\_y

def flipcointogetdataadd(n,data):

data\_x =data[0]

data\_y =data[1]

for i in range (n):

#set random seed

random.seed()

x1 = random.gauss(6, np.sqrt(0.3))

x2 = random.gauss(0, np.sqrt(0.1))

data\_x.append([1,x1,x2])

data\_y.append(1)

return data\_x ,data\_y

def linear\_regression(data):

x = np.array(list (data[0]))

x\_t =np.transpose(x)

y = np.array(list (data[1]))

t = x\_t.dot(x)

t\_inv = np.linalg.inv(t)

w\_lin = (t\_inv.dot(x\_t)).dot(y)

# y\_ = w\_lin[0]+w\_lin[1]\*data[0]+w\_lin\*data[1]

return w\_lin

def e\_in(w\_lin , data , n):

x = np.array(list (data[0]))

x\_t =np.transpose(x)

x\_tx = x\_t.dot(x)

y = np.array(list (data[1]))

y\_t =np.transpose(y)

y\_ty = y\_t.dot(y)

x\_ty = x\_t.dot(y)

x\_txw = x\_tx.dot(w\_lin)

w\_t =np.transpose(w\_lin)

# e\_in = 1/n(wt\*xt\*xw-2\*w^t\*x^t\*y+y^t\*y)

e\_in = (w\_t.dot(x\_txw)-2\*(w\_t.dot(x\_ty))+y\_ty)/n

#print (e\_in)

return e\_in

def sigmoid(s):

return 1/(1 + math.exp(-s))

def logistic\_regression(data , eta,itr ):

x = np.array(list (data[0]))

y = np.array(list (data[1]))

n = y.size

w\_t =np.zeros(x.shape[1])

for i in range(itr):

for i in range(n):

xn =x[i]

yn =y[i]

e\_grad = -sigmoid(-yn\*np.ndarray.dot(w\_t, xn))\*yn\*xn

w\_t += eta\*(-e\_grad)

return (w\_t)

def test\_log(w,testdata,n):

x = np.array(list (testdata[0]))

y = np.array(list (testdata[1]))

E\_out\_bin = 0

error = 0

for i in range(n):

if (((sigmoid(-(x.dot(w)[i]))-0.5)\*y[i]))> 0 :

E\_out\_bin += 1

elif (sigmoid(-(x.dot(w)[i]))-0.5)\*y[i]< 0 :

error += 1

return (E\_out\_bin/n)

def linear01error(wlin,data,n):

x = np.array(list (data[0]))

y = np.array(list (data[1]))

right = 0

error = 0

for i in range(n):

if x.dot(wlin)[i]\*y[i] >0:

right+=1

elif x.dot(wlin)[i]\*y[i]<0:

error+=1

return error/n

# 13-14

sum = 0

sum2 = 0

sum3 = 0

sum4 = 0

eout10 = 0

ein10 = 0

ans = 0

n =100

test = 5000

train =200

ans1 = 0

itr = 500

#test

#15

for i in range(n):

random.seed(n)

traindata =flipcointogetdata(200)

testdata = flipcointogetdata(5000)

traindata = flipcointogetdataadd(20, traindata)

w\_lin = linear\_regression(traindata)

eout10 = linear01error(w\_lin, testdata, 5000)

w\_log = logistic\_regression(traindata, 0.1 ,itr )

eout10log = test\_log(w\_log, testdata, test)

sum3 += eout10

sum4 += eout10log

sum3 = sum3/n

sum4 = sum4/n

print("Q15")

print("eout10linear(D):",sum3)

print('err log', sum4)

for i in range(n):

random.seed(n)

traindata =flipcointogetdata(train)

testdata = flipcointogetdata(test)

w\_lin = linear\_regression(traindata)

eout10 = linear01error(w\_lin, testdata, test)

ein10 = linear01error(w\_lin, traindata, train)

a = e\_in(w\_lin , traindata , train)

w\_log = logistic\_regression(traindata, 0.1 ,itr )

eout10log = test\_log(w\_log, testdata, test)

sum2 += a

sum3 += eout10

sum4 += eout10log

ans += abs(eout10-ein10)

ans = ans/n

sum2 = sum2/n

sum3 = sum3/n

sum4 = sum4/n

#14

print("Q14")

print("error rate" , ans )

#13

print("Q13")

print("sqr" , sum2)

#16

print("Q16")

print("eout10linear(D):",sum3)

print("logerr", sum4)