1(B)

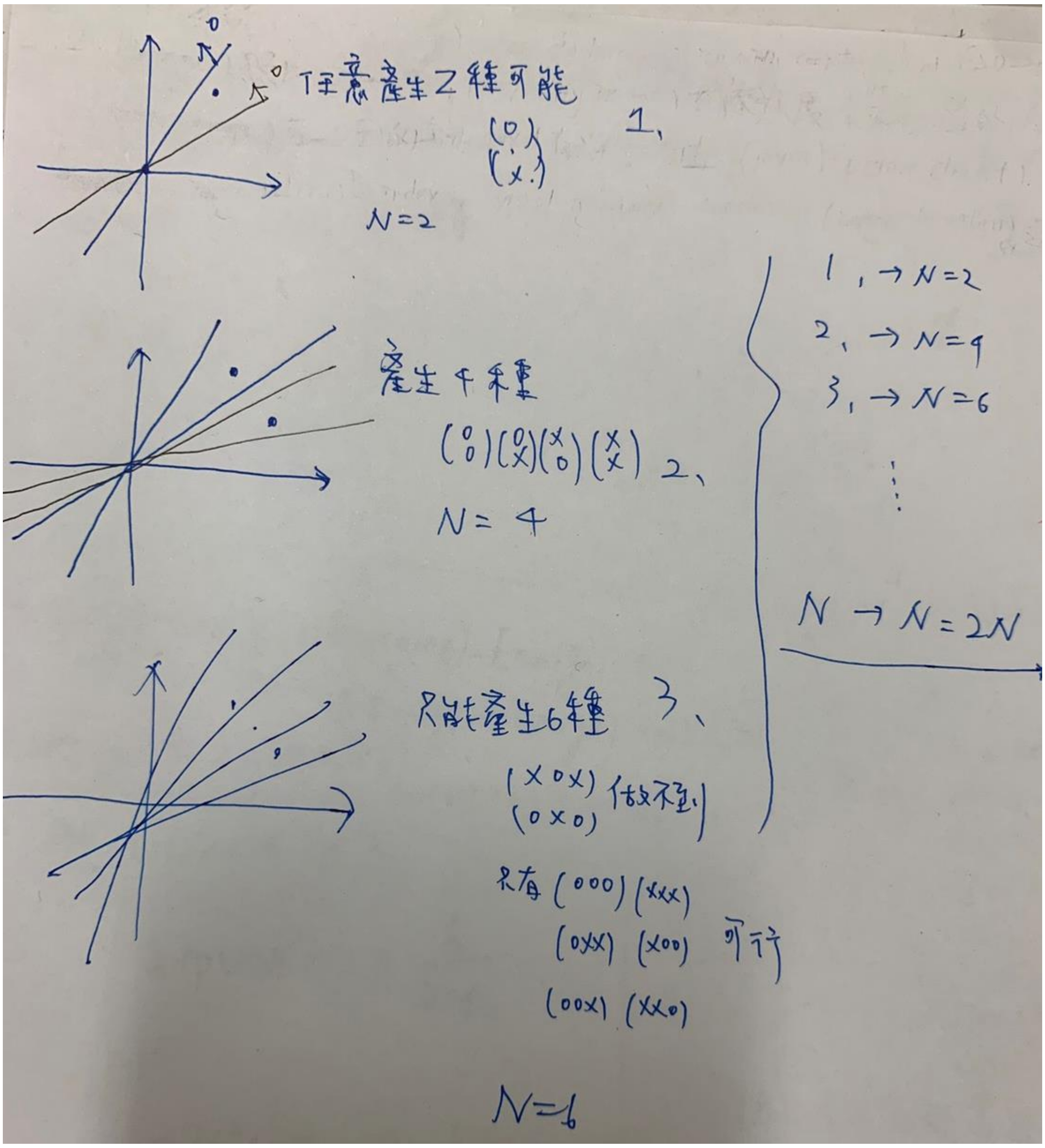
(A)共線(3-t,3,3+t)兩個點才能 shatter

(B)空間中不共平面的 4 點 可以 shatter

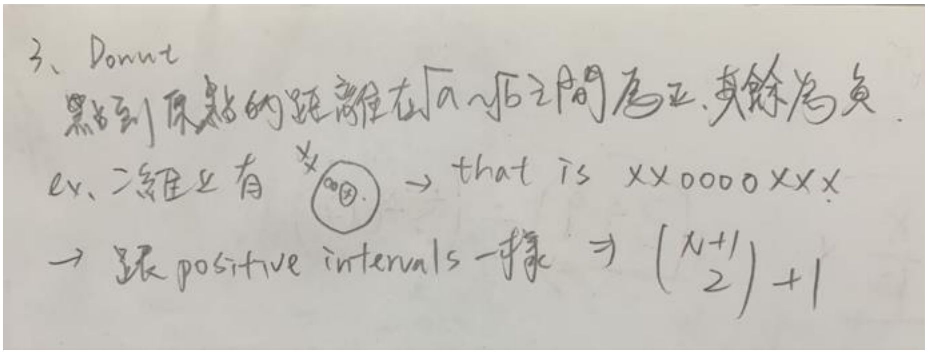
(C)共平面(2x-4y+2z=0) 又(111,222)(432,234)共線且不平行=2D 4 個點，no shatter

(D) 空間中不共平面的 5 點 但 5>d+1=4 no shatter

2.(C)



3.(A)



3、Donut

點到原點的距離在$\sqrt{a}$ ~ $\sqrt{b}$之間為正，其餘為負。

ex、二維上有 → that is xx0000xxx

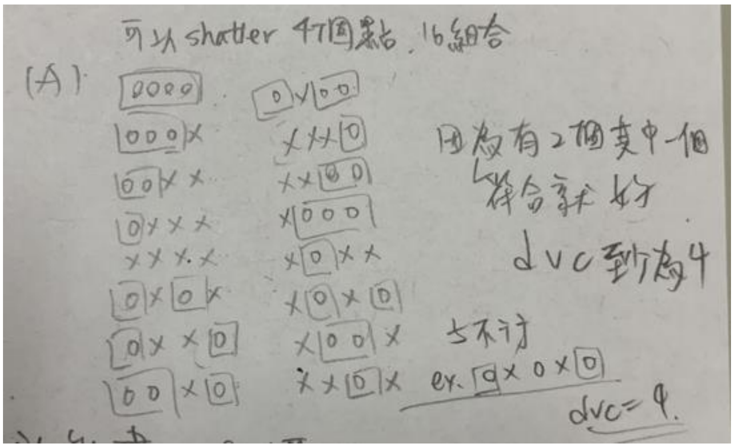
→ 跟 positive intervals 一樣 ⇒ $\binom{N+1}{2}+1$

4.(D)

Positive interval→dvc = 2

5. (E)

(A)



可以 shatter 4個點，16組合

(A)

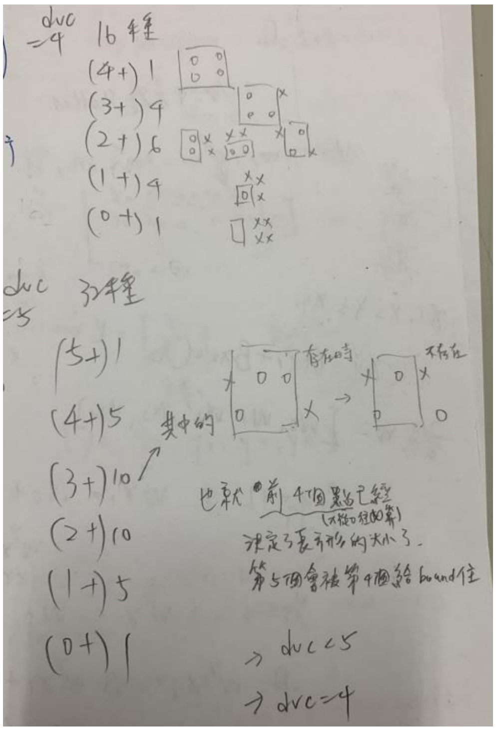
因為有2個其中一個符合就好

dvc 到為4

5不行 ex.

dvc = 4.

(B)



dvc=4 16種

(4+) 1

(3+) 4

(2+) 6

(1+) 4

(0+) 1

dvc=5 32種

(5+) 1

(4+) 5 其中的

(3+) 10

(2+) 10

(1+) 5

(0+) 1

有包含 → 不包含

也就是前4個點已經(不論如何組合)決定了長方形的大小了。

第5個會被第4個給bound住

→ dvc<5

→ dvc=4

因為ABD都是dvc=4，∴證明(C) dvc=4 or not

first dvc<5 → N=4會被shatter (by 上課定義?)

$wx=y$ $w=x^{-1}y$ x is invertable … dvc至少=4

dvc≥5 → N=5

5個資料點 $(x_1, x_2, x_3, x_4, x_5)$

線性相依 設 $x_5 = ax_1 + bx_2 + cx_3 + Dx_4$ $\begin{cases} \text{if } y=1 \rightarrow wx_i = w_0 + wx_i \rightarrow wx_i > -w_0 \\ y=-1 \rightarrow wx_i < -w_0 \end{cases}$

if abcD>0 for each 設 y=[-1, -1, -1, -1, y5]

1

→ $wx_5 = awx_1 + bwx_2 + cwx_3 + Dwx_4$

$wx_5 < -(a+b+c+D)w_0$ ∵ $y_5 = wx_5 + w_0$ if $(a+b+c+D)>1$

那麼 y5必<0

即做不出 (-1,-1,-1,-1,+1)

if 0<(a+b+c+D)<1 考慮y[1, 1, 1, 1, y5]

→ $wx_5 > -(a+b+c+D)w_0$ → 做不出 (1,1,1,1,-1)

a+b+c+D=1 → sign(0) → 未定義

→在a,b,c,D>0時，都不能shatter

abc>0 D<0

4

→ y=[-1,-1,-1,1,y5] → D<0 $w^Tx_4 > -w_0$ $D\,w^Tx_4 < -D\,w_0$ $D\,w^Tx_4 < |D|\,w_0$

→ $wx_5 < -(a+b+c-|D|)w_0$

→ |a+b+c|-|D|>1 做不出 [-1,-1,-1,1,1]

|a+b+c|-|D|<1 做不出 [1,1,1,-1,-1]

ab>0 cD<0

6

→ y=[-1,-1,1,1,y5]

→ $wx_5 < -(a+b-|c|-|D|)w_0$

→ (a+b)-|c|-|D|>1 做不出 [-1,-1,1,1,1]

→ (a+b)-|c|-|D|<1 做不出 [1,1,-1,-1,-1]

同理 a>0, b,c,D<0

4 $\begin{cases} b>0 & acD<0 \\ c>0 & abD<0 \\ D>0 & abc<0 \end{cases}$ 做不出 [-1,1,1,1,1]

[1,-1,-1,-1,-1]

abcD<0 做不出 [1,1,1,1,1]

[-1,-1,-1,-1,-1]

其中排列組合相同，可以說明不管a,b,c,D大於小於0的所有組合

對於a+b+c+D與1的比較

做不出 [-sign(a), -sign(b), -sign(c), -sign(D), 1]

[sign(a), sign(b), sign(c), sign(D), -1]

其中一種

∴無法shatter dvc=4

(C)

(D)

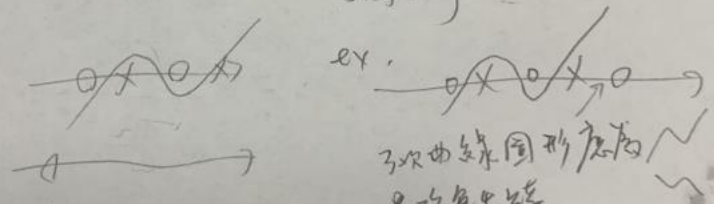
(D) sign(h(x) = $w_3x^3 + w_2x^2 + w_1x + w_0$

3個點 8可能可以完成

→ shatter

4個 16可以

5個不行



3次曲線圖形應為

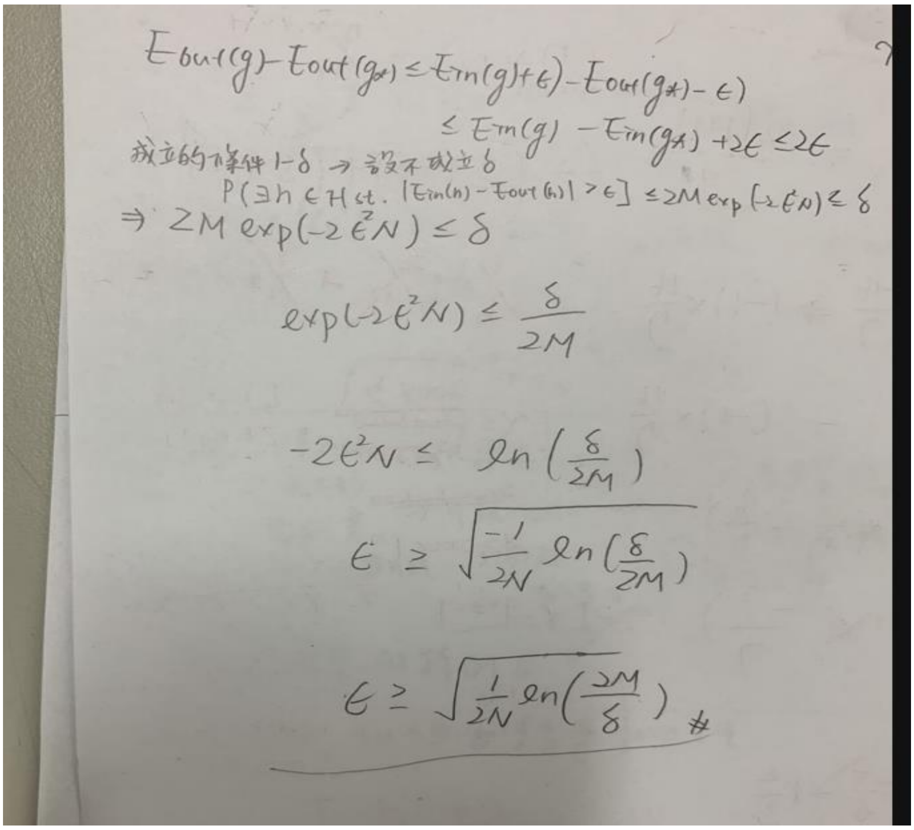
最後會出錯

∴ dvc=4.

6. (D)

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

7.(C)



$$E_{out}(g) - E_{out}(g_*) \le E_{in}(g) + \epsilon) - E_{out}(g_*) - \epsilon)$$
$$\le E_{in}(g) - E_{in}(g_*) + 2\epsilon \le 2\epsilon$$

成立的條件 $1-\delta$ → 設不成立 δ

$$P(\exists h \in H \text{ s.t. } |E_{in}(h) - E_{out}(h)| > \epsilon] \le 2M\exp(-2\epsilon^2 N) \le \delta$$
$$\Rightarrow 2M\exp(-2\epsilon^2 N) \le \delta$$
$$\exp(-2\epsilon^2 N) \le \frac{\delta}{2M}$$
$$-2\epsilon^2 N \le \ln\left(\frac{\delta}{2M}\right)$$
$$\epsilon \ge \sqrt{\frac{-1}{2N}\ln\left(\frac{\delta}{2M}\right)}$$
$$\epsilon \ge \sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)} \ \#$$

8.(B)

帶數字進去 x=11000

$$4(2x+2)e^{(-0.125\cdot 0.01x)} = 0.1 \qquad x = 10946.29979\ldots, x = -0.98751\ldots$$

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 = $$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\underbrace{1 + \exp(\overbrace{-y_n \mathbf{w}^T \mathbf{x}_n}^{\bigcirc})}_{\square}\right)$$

E 一次微分 $$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\left(-y_n x_{n,i}\right)$$

E 二次微分 =

$$\nabla^2 E = \frac{1}{N}\Sigma\left(\frac{e^{-ywx} \cdot y_n y_n^T x_n x_n^T}{(1+e^{-ywx})^2}\right) \qquad \nabla^2 E = \frac{1}{N}\Sigma\left(\frac{e^{-y_n x_n w^T}}{(1+e^{-y_n x_n w^T})^2} y_n \cdot y_n^T x_n \cdot x_n^T\right)$$

$$h(x) = \frac{1}{1+e^{(wx)}} = \frac{e^{(-wx)}}{1+e^{(-wx)}} \qquad 又.\ y_n \cdot y_n^T = 1$$

$$\boxed{h(-y_n x_n),\ h(y_n x_n) \text{代入}}$$

$$\Rightarrow \nabla^2 E = \frac{1}{N}\Sigma \frac{1}{1+e^{(-y_n x_n w^T)}} \cdot \frac{e^{(-y_n x_n w^T)}}{1+e^{(-y_n x_n w^T)}} \cdot x_n \cdot x_n^T \cdot 1$$

$$= \frac{1}{N}\Sigma\, h(-y_n x_n)\, h(y_n x_n)\, x_n \cdot x_n^T \quad ※$$

11.(E)

X的奇異分解值 $X = U\Sigma V^T$

U & V (正交矩陣)

滿足 $U^T = U^{-1}$ $V^T = V^{-1}$

$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$ $D \rightarrow$ 可逆分塊 $= diag(\sigma_1 \ldots \sigma_r)$

$X = d+1 \times N$ 階矩陣

令 $rank = r$

$U = d+1 \times d+1$

$V = N \times N$

$\rightarrow X^+ = V\Sigma^+ U^T = V \begin{bmatrix} D^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T$

$\rightarrow (\Sigma^+)^+ = \left(I_n \begin{bmatrix} D^{-1} & 0 \\ 0 & 0 \end{bmatrix}_{n \times m} I_n\right)^+ = I_m \begin{bmatrix} (D^{-1})^{-1} & 0 \\ 0 & 0 \end{bmatrix}_{m \times n} I_n = \Sigma$

$\rightarrow (\Sigma^+)^+ = \Sigma$

$\rightarrow (X^+)^+ = (V\Sigma^+ U^T)^+ = U(\Sigma^+)^+ V^T = U(\Sigma)V^T = X$

(A) when invertable $(X^T X)^{-1} X^T = X^+ \rightarrow X^T = (X^T X) X^+$ 又 $(X \cdot X^+)$

$\therefore X^T = X^T$ 成立. $(U\Sigma V^T \cdot V\Sigma^+ U^T)$

$= I$

(B) $(X \cdot X^+)^K = XX^+$ $\because X \cdot X^+ = I \rightarrow$ (Idempotent matrix) 成

(C) 同上.

(D) $\underline{trace(XX^+)} = trace(I) = r = rank(X)$

I_D

$rank(I) = trace(I)$ if Idempotent matrix)

$\therefore trace(XX^+) = rank(XX^+)$

$XX^+ = U\Sigma V^T V\Sigma^+ U^T = U\Sigma\Sigma^+ U^T = [U_r \; U_{d+1-r}] \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_r^T \\ U_{d+1-r}^T \end{bmatrix} = U_r U_r^T$

$\rightarrow$ 特徵值包含 r 個 1 $\therefore rank(XX^+) = r$

12(A)

$$L(\mu, \sigma^2) = \prod f(x_i, \mu, \sigma^2) = \prod \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} = (2\pi\sigma^2)^{\frac{-n}{2}} e^{\frac{-\Sigma (x-\mu)^2}{2\sigma^2}}$$

取 ln $= \frac{-n}{2} \ln(2\pi\sigma^2) - \frac{\Sigma (x-\mu)^2}{2\sigma^2}$

$x = y$, $\mu = w^T x$, $\sigma^2 = a^2$ 代入

$$\text{max} \Rightarrow \frac{-n}{2} \ln(2\pi a^2) - \frac{\Sigma (y - w^T x)^2}{2a^2}$$

$$\text{min} \Rightarrow \underbrace{\frac{n}{2} \ln(2\pi a^2)}_{\text{定值}} + \frac{\Sigma (y - w^T x)^2}{2a^2}$$

$$\text{min} \Rightarrow \frac{\Sigma (y - w^T x)^2}{2a^2}$$

取 gradient $\nabla(w) = \frac{1}{2a^2} \| Y^T y - 2 y^T w^T x + x^T w w^T x \|$

$$\nabla(w) = \frac{1}{2a^2} (\cancel{-2y^T w} \; -2yX + 2wX^T x)$$

$$0 = \nabla(w) = \frac{1}{a^2} (w x^T x - x^T y)$$

$$a^2 \cdot 0 = w x^T x - x^T y \rightarrow 0 = w x^T x - x^T y$$

$$w = (x^T x)^{-1} x^T y \quad \#$$

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)
```