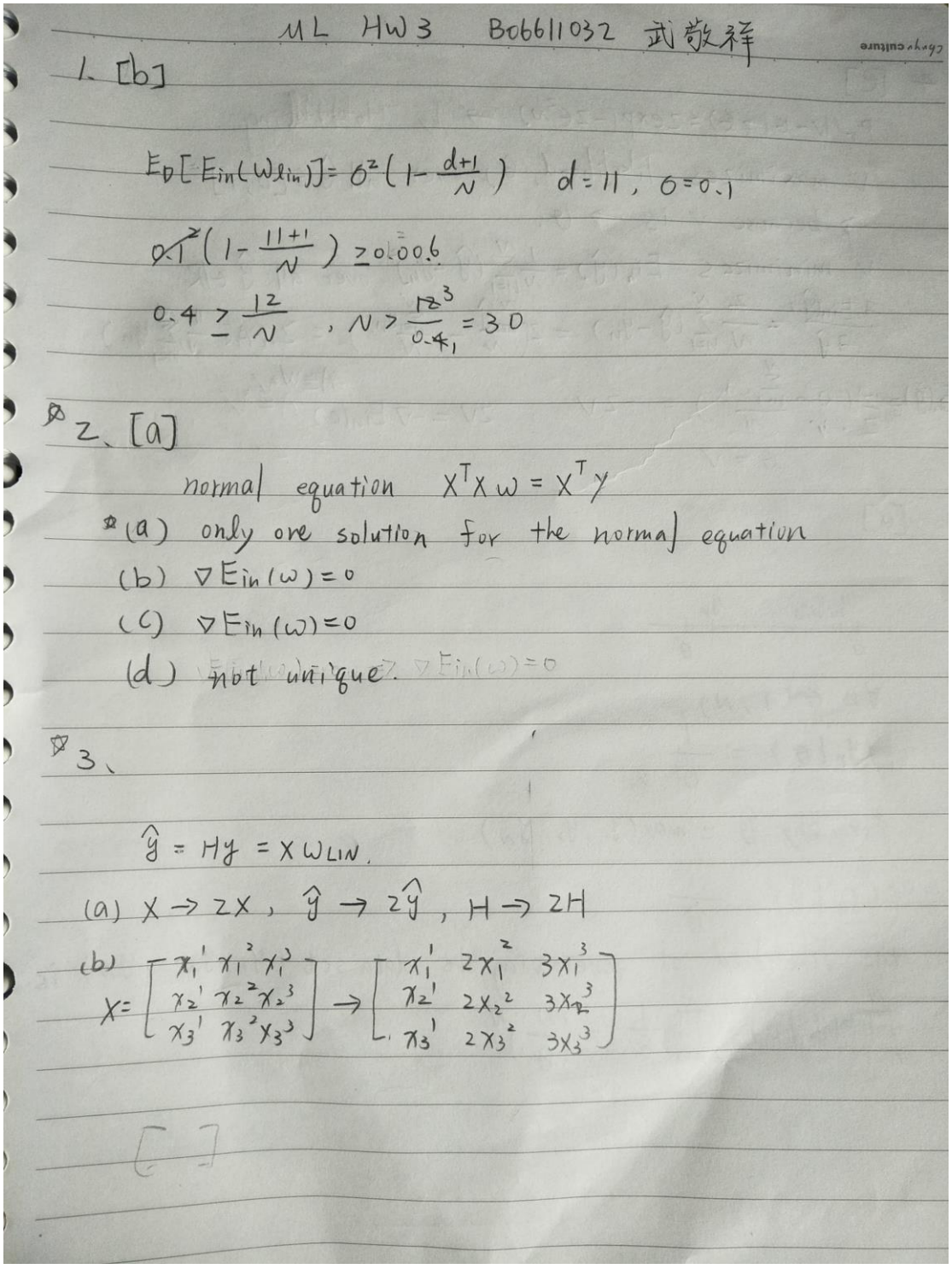
ML HW3

1,2,3:



ML HW3 B06611032 武敬祥

1. [b]

$E_D[E_{in}(w_{lin})] = \sigma^2(1-\frac{d+1}{N})$ $d=11$, $\sigma=0.1$

$0.1^2(1-\frac{11+1}{N}) \geq 0.006$

$0.4 \geq \frac{12}{N}$, $N > \frac{12^3}{0.4_1} = 30$

2. [a]

normal equation $X^TXw = X^Ty$

(a) only one solution for the normal equation

(b) $\nabla E_{in}(w) = 0$

(c) $\nabla E_{in}(w) = 0$

(d) not unique. $\nabla E_{in}(w) = 0$

3.

$\hat{y} = Hy = Xw_{LIN}$.

(a) $X \to 2X$, $\hat{y} \to 2\hat{y}$, $H \to 2H$

(b) $X = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \\ x_3^1 & x_3^2 & x_3^3 \end{bmatrix} \to \begin{bmatrix} x_1^1 & 2x_1^2 & 3x_1^3 \\ x_2^1 & 2x_2^2 & 3x_2^3 \\ x_3^1 & 2x_3^2 & 3x_3^3 \end{bmatrix}$

4,5:

4. [e]

$P_r(|\nu - \theta| > \epsilon) \le 2\exp(-2\epsilon^2 N)$ → by Hoeffding

ν maximizes likelihood(θ) over all $\hat{\theta} \in [0,1]$

→ because ν is → θ.

ν minimizes $E_{in}(\hat{y}) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y} - y_n)^2$ over all $\hat{y} \in \mathbb{R}$

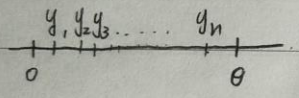
$$\frac{\partial E_{in}(\hat{y})}{\partial \hat{y}} = \frac{2}{N}\sum_{n=1}^{N}(\hat{y} - y_n) = 2\left(\frac{\sum_{n=1}^{N}\hat{y}}{N} - \frac{\sum_{n=1}^{N} y_n}{N}\right) = 2\left(A - \frac{1}{N}\sum_{n=1}^{N} y_n\right)$$

$A = \nu$

$$\nabla E_{in}(\theta) = 2\left(\theta - \frac{\sum_{n=1}^{N} y_n}{N}\right) = -2\nu \quad , \quad 2\nu = -\nabla E_{in}(\theta)$$

$\theta = \nu$

5. [a]



$\forall n \in (1, N)$,

$$P(y_n \mid \theta) = \frac{1}{\theta}$$

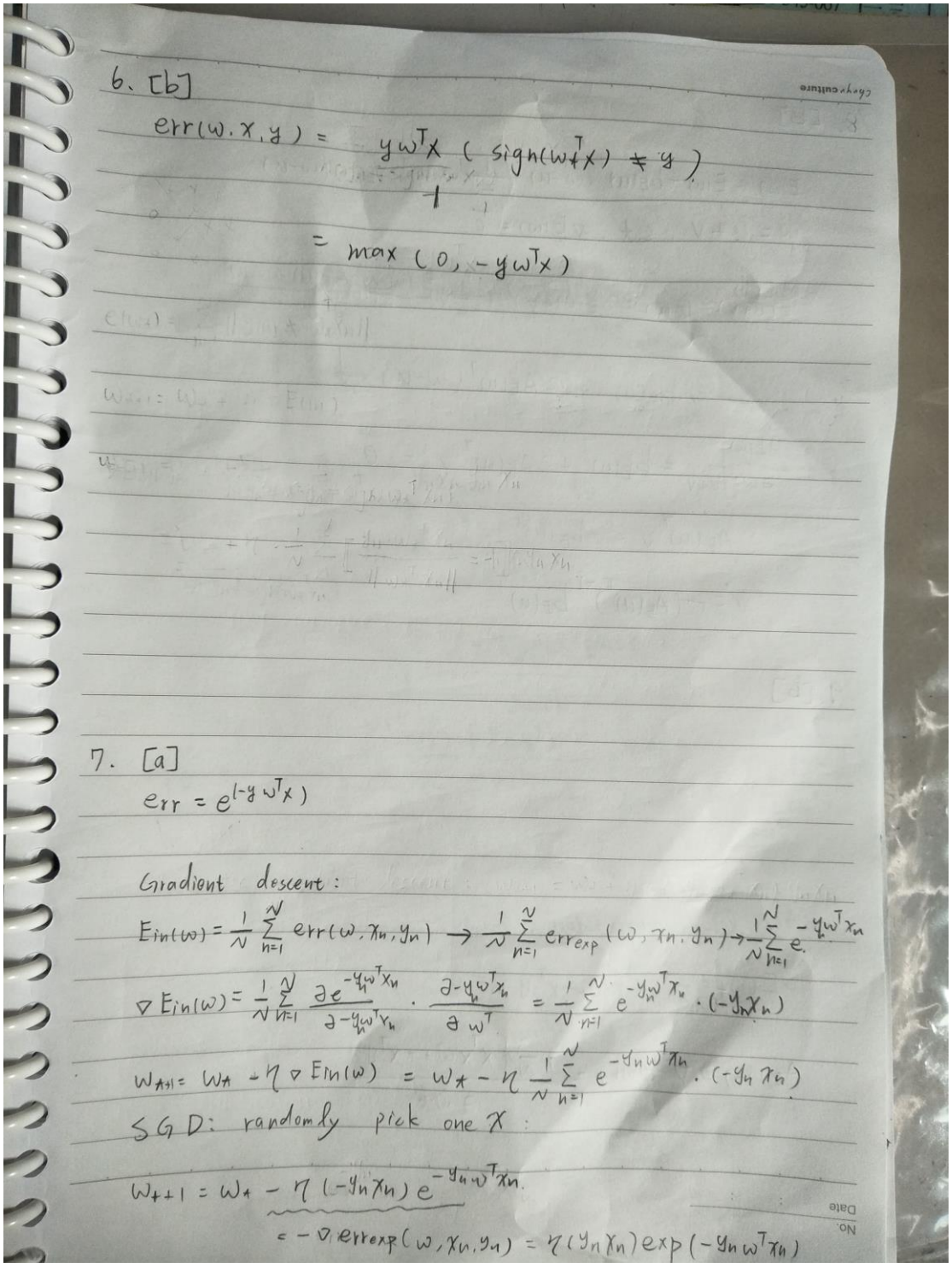
For any $\hat{\theta} = \max(y_1, y_2 \ldots y_N)$

$$P(y_n \mid \hat{\theta}) = \frac{1}{\hat{\theta}}$$

the likelihood of generating the data set $(y_1, y_2 \ldots y_N)$ is

$$\prod_{n=1}^{N} P(y_n \mid \hat{\theta}) = \prod_{n=1}^{N} \frac{1}{\hat{\theta}} = \left(\frac{1}{\hat{\theta}}\right)^N$$

6,7:



6. [b]

$$err(w, x, y) = \underbrace{y w^T x}_{-1} \; (sign(w_t^T x) \neq y)$$

$$= \max(0, -y w^T x)$$

7. [a]

$$err = e^{(-y w^T x)}$$

Gradient descent:

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} err(w, x_n, y_n) \rightarrow \frac{1}{N} \sum_{n=1}^{N} err_{exp}(w, x_n, y_n) \rightarrow \frac{1}{N} \sum_{n=1}^{N} e^{-y_n w^T x_n}$$

$$\nabla E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial e^{-y_n w^T x_n}}{\partial -y_n w^T x_n} \cdot \frac{\partial -y_n w^T x_n}{\partial w^T} = \frac{1}{N} \sum_{n=1}^{N} e^{-y_n w^T x_n} \cdot (-y_n x_n)$$

$$w_{t+1} = w_t - \eta \nabla E_{in}(w) = w_t - \eta \frac{1}{N} \sum_{n=1}^{N} e^{-y_n w^T x_n} \cdot (-y_n x_n)$$

SGD: randomly pick one x:

$$w_{t+1} = w_t - \eta (-y_n x_n) e^{-y_n w^T x_n}$$

$$= -\nabla err_{exp}(w, x_n, y_n) = \eta (y_n x_n) \exp(-y_n w^T x_n)$$

8,9:

8. [b]

$$E(w) \approx E(u) + b_E(u)^T (w-u) + \frac{1}{2!}(w-u)^T A_E(u)(w-u)$$

$$w = u + v \quad s.t. \quad \nabla E(w) = \vec{0}$$

$$\frac{\partial E(w)}{\partial w} = 0 + \frac{\partial b_E(u)^T (w-u)}{\partial w} + \frac{1}{2}\frac{\partial (w-u)^T A_E(u)(w-u)}{\partial w}$$

$$= b_E(u) + 2A_E(u)^T (w-u) \cdot \frac{1}{2}$$

$$\left.\frac{\partial E(w)}{\partial w}\right|_{u+v} = b_E(u) + A_E(u)^T v = \vec{0}$$

($b_E(u)$: $1\times d$; $A_E(u)^T$: $d\times d$; v: $d\times 1$)

$$A_E(u)^T v = -b_E(u)^T$$

$$v = -(A_E(u)^T)^{-1} b_E(u)$$

($(A_E(u)^T)^{-1}$: $d\times d$; $b_E(u)$: $1\times d$)

9. [b]

for linear regression:

$$\nabla E(w_*) = \frac{2}{N}(X^T X w_* - X^T y) = 0$$

$$A_E(w_*) = \nabla^2 E(w_*) = \frac{\partial \frac{2}{N}(X^T X w_* - X^T y)}{\partial w_*}$$

$$= \frac{2}{N} X^T X$$

10:

10. [b]

$$h_y(x) = \frac{e^{(w_y^T x)}}{\sum_{i=1}^{K} e^{(w_i^T x)}} \qquad w = \begin{bmatrix} w_1 & w_2 & \cdots & w_K \\ & & & \end{bmatrix}$$

$$err(w, x, y) = -\ln h_y(x) = -\sum_{k=1}^{K} [\![y=k]\!] \ln h_k(x)$$

$$h_y(x) \propto \prod_{n=1}^{N} h_{y_n}(x_n) = \prod_{n=1}^{N} \frac{e^{w_y^T x_n}}{\sum_{i=1}^{K} e^{w_i^T x_n}} \propto \frac{e^{w_y^T x}}{\sum_{i=1}^{K} e^{w_i^T x}}$$

$$D = \{(x_1, y_1), (x_2, y_2) \cdots (x_N, y_N)\}$$

$$\frac{\partial err(w,x,y)}{\partial w_{ik}} = \frac{-\sum_{k=1}^{K} \partial [\![y=k]\!] \ln h_k(x)}{\partial w_{ik}}$$

$$= \frac{-\sum_{k=1}^{K} \partial [\![y=k]\!] \left(w_k^T x - \ln \sum_{i=1}^{K} e^{w_i^T x} \right)}{\partial w_{ik}}$$

$$= \frac{-\partial [\![y=k]\!] w_k^T x}{\partial w_{ik}} + \frac{\partial \ln \sum_{k=1}^{K} e^{w_k^T x}}{\partial w_{ik}} [\![y=k]\!]$$

$$= -[\![y=k]\!] x_i + \frac{\partial \ln \sum_{k=1}^{K} e^{w_k^T x}}{\partial \sum_{k=1}^{K} e^{w_k^T x}} \cdot \frac{\partial \sum_{k=1}^{K} e^{w_k^T x}}{\partial w_{ik}} [\![y=k]\!]$$

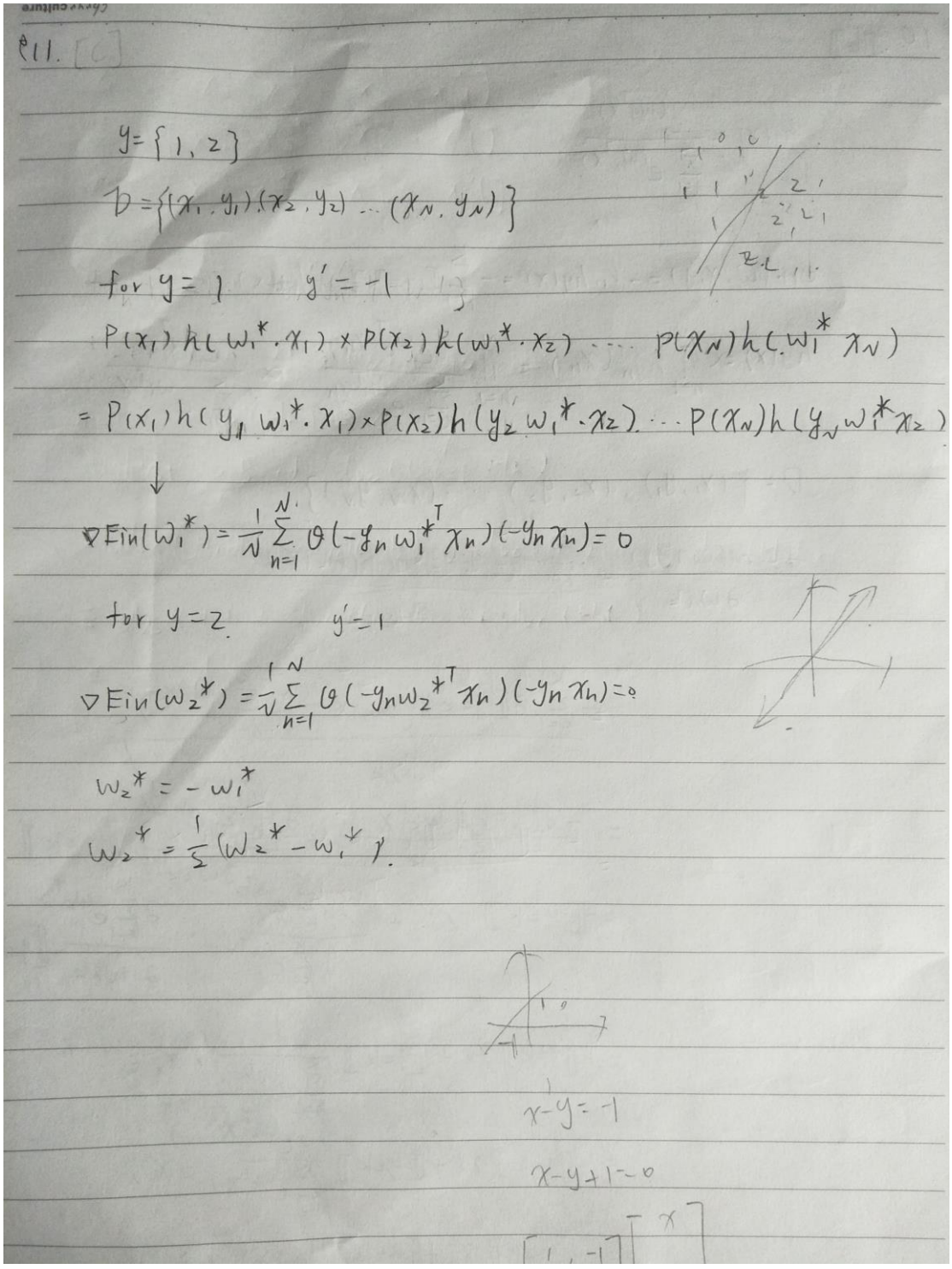
$$= -[\![y=k]\!] x_i + [\![y=k]\!] \frac{1}{\sum_{k=1}^{K} e^{w_k^T x}} \cdot x_i e^{w_k^T x_i}$$

$$= -[\![y=k]\!] x_i + [\![y=k]\!] \frac{e^{w_k^T x}}{\sum_{k=1}^{K} e^{w_k^T x}} \cdot x_i$$

$$= -[\![y=k]\!] x_i + [\![y=k]\!] h_k(x) x_i$$

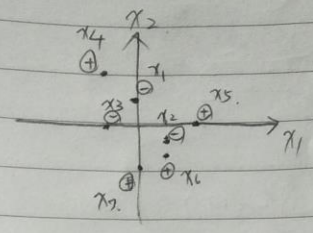
$$= (h_k(x) - [\![y=k]\!]) x_i$$

11:


11. [c]
y={1, 2}
D={(x1, y1)(x2, y2) ... (xN, yN)}
for y=1 y'=-1
P(x1) h(w1*·x1) × P(x2) h(w1*·x2) ... P(xN) h(w1* xN)
= P(x1) h(y1 w1*·x1) × P(x2) h(y2 w1*·x2) ... P(xN) h(yN w1* x2)
∇Ein(w1*) = 1/N Σ_{n=1}^{N} θ(-yn w1*T xn)(-yn xn) = 0
for y=2 y'=1
∇Ein(w2*) = 1/N Σ_{n=1}^{N} θ(-yn w2*T xn)(-yn xn) = 0
w2* = -w1*
w2* = 1/2 (w2* - w1*).
x-y=-1
x-y+1=0

12:

12. [e]



× [a] $-9-x_1+2x_1^2-2x_1x_2+3x_2^2$ [e] $-7+2x_1^2-2x_1x_2+3x_2^2$

$x_1(0,1) = -9-0+0-0+3 < 0$

$x_2(1,-0.5) = -9-1+2+1+0.75 < 0$

$x_3(-1,0) = -9+1+2-0+0 < 0$

$x_4(-1,2) = -9+1+2+4+12 > 0$

$x_5(2,0) = -9-2+8-0+0 < 0$ ×

× [b] $-5-x_1+2x_2+3x_1^2-7x_1x_2+2x_2^2$

$x_1(0,1) < 0$

$x_2(1,-0.5) = -5-1-1+3-3.5+0.5 < 0$

$x_3(-1,0) = -5+1+3 < 0$

$x_4(-1,2) = -5+1+4+3+14+8 > 0$

$x_5(2,0) = -5-2+12 > 0$

$x_6(1,-1.5) = -5-1-3+3+10.5+4.5 > 0$

$x_7(0,-2) = -5-4 < 0$.

× [c] $9-x_1+4x_2+2x_1^2-2x_1x_2+3x_2^2$

$x_1(0,1) = 9+4+ > 0$.

$x_4(-1,2) = 9+1+8+2+ > 0$

× [d] $2-x_1-4x_2-2x_1^2+7x_1x_2-4x_2^2$

$x_1(0,1) = 2-4-4 < 0$.

$x_2(1,-0.5) = 2-1+2-2-3.5-1 < 0$

$x_3(-1,0) = 2+1-2 > 0$.

13:

13.

$x_1 = [x_1^1 \; x_2^1 \cdots x_k^1 \cdots x_d^1]$

$x_2 = [x_1^2 \; x_2^2 \cdots x_k^2 \cdots x_d^2]$

$x_3 = [x_1^3 \; x_2^3 \cdots x_k^3 \cdots x_d^3]$

$\phi = (1, x_k)$

$dvc(H_k)$ in n-D pla is $n+1$

$n=1$, $dvc(H_k) = 2$

tightest UB of $dvc(\bigcup_{k=1}^{d} H_k)$

$= dvc(H_1) \cup dvc(H_2) \cup dvc(H_3) \cup \cdots \cup dvc(H_d)$

$\leq dvc(H_1) + dvc(H_2) + \cdots + dvc(H_d)$

$\leq 2 \times d$

14:

```
import numpy as np

X0 = np.ones([1000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
train = np.append(X0 , train ,axis=1)
X = np.hsplit(train,[11])[0]  ## saparate the array into two part on th
e base of the nth column
Y = np.hsplit(train,[11])[1]
Wlin = np.dot(np.linalg.pinv(X) , Y)

E = np.dot(X , Wlin) - Y
print(E)
Ein = 0
for i in range(len(E)):
    Ein += (E[i][0]) ** 2/len(E)

print(Ein)
```

15:

```
import numpy as np
from random import randint


X0 = np.ones([1000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
train = np.append(X0 , train ,axis=1)
X = np.hsplit(train,[11])[0]  ## saparate the array into two part on th
e base of the nth column
Y = np.hsplit(train,[11])[1]
Wlin = np.dot(np.linalg.pinv(X) , Y)

E = np.dot(X , Wlin) - Y
Einlin = 0
for i in range(len(E)):
    Einlin += (E[i][0]) ** 2/len(E)

n = 0.001
count = 0
times_avg = 0
while count < 1000:
    print(count)
    times = 0
    ratio = 100
    Wt = np.zeros([11,1])
    while ratio > 1.01:
        rand = randint(0,999)
        E = np.dot(X , Wt) - Y
        Einwt = 0
        for i in range(len(E)):
            Einwt += (E[i][0]) ** 2/len(E)
        x = X[rand].reshape((1,11))    # what the hell is it????

        Ein_grad = 2*(np.dot(x,Wt) - Y[rand][0]) * np.transpose(x)
        Wt = Wt- n*Ein_grad
        ratio = Einwt / Einlin
        times += 1
```

```
    times_avg += times/1000
    count += 1

print(times_avg)
```

16:

```
import numpy as np
from random import randint


X0 = np.ones([1000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
train = np.append(X0 , train ,axis=1)
X = np.hsplit(train,[11])[0]  ## saparate the array into two part on th
e base of the nth column
Y = np.hsplit(train,[11])[1]

def sigmoid(h):
    ans = 1/(1+np.exp(-h))
    return ans

n = 0.001
count = 0
Ein_avg = 0
while count < 1000:
    print(count)
    times = 0
    Wt = np.zeros([11,1])
    while times < 500:
        rand = randint(0,999)
        x = X[rand].reshape((1,11))    # what the hell is it????
        Ein_grad = sigmoid(-
Y[rand][0] * np.dot(x,Wt)) * Y[rand][0] * np.transpose(x)
        Wt = Wt + n*Ein_grad
```

```
        times += 1

    Einwt = 0
    for i in range(1000):
        x = X[i].reshape((1,11))
        CE_error = np.log(1+np.exp(-Y[i][0] * np.dot(x,Wt)[0][0]))
        Einwt += CE_error/1000

    Ein_avg += Einwt/1000
    count += 1

print(Ein_avg)
```

17:

```
import numpy as np
from random import randint


X0 = np.ones([1000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
train = np.append(X0 , train ,axis=1)
X = np.hsplit(train,[11])[0]  ## saparate the array into two part on th
e base of the nth column
Y = np.hsplit(train,[11])[1]
Wlin = np.dot(np.linalg.pinv(X) , Y)

def sigmoid(h):
    ans = 1/(1+np.exp(-h))
    return ans

n = 0.001
count = 0
Ein_avg = 0
while count < 1000:
    print(count)
    times = 0
    Wt = Wlin
    while times < 500:
        rand = randint(0,999)
        x = X[rand].reshape((1,11))    # what the hell is it????
```

```
        Ein_grad = sigmoid(-
Y[rand][0] * np.dot(x,Wt)) * Y[rand][0] * np.transpose(x)
        Wt = Wt + n*Ein_grad
        times += 1

    Einwt = 0
    for i in range(1000):
        x = X[i].reshape((1,11))
        CE_error = np.log(1+np.exp(-Y[i][0] * np.dot(x,Wt)[0][0]))
        Einwt += CE_error/1000

    Ein_avg += Einwt/1000
    count += 1

print(Ein_avg)
```

18:

```
import numpy as np


X0_train = np.ones([1000,1])
X0_test =np.ones([3000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
test = np.genfromtxt('test.txt', delimiter='\t')
train = np.append(X0_train , train ,axis=1)
test = np.append(X0_test , test ,axis=1)

X_train = np.hsplit(train,[11])[0]  ## saparate the array into two part
 on the base of the nth column
Y_train = np.hsplit(train,[11])[1]
X_test = np.hsplit(test,[11])[0]  ## saparate the array into two part o
n the base of the nth column
Y_test = np.hsplit(test,[11])[1]
Wlin = np.dot(np.linalg.pinv(X_train) , Y_train)

def sign(s):
    if s >= 0:
        return 1
    else:
        return -1
```

```
E_train = np.dot(X_train , Wlin) * Y_train  #
E_test = np.dot(X_test , Wlin) * Y_test
Ein = 0
Eout = 0
for i in range(1000):
    if sign(E_train[i]) == -1:
        Ein += 1/1000
    if sign(E_test[i]) == -1:
        Eout += 1/1000

print(Ein)
print(Eout)
print(abs(Ein-Eout))
```

19:

```
import numpy as np


X0_train = np.ones([1000,1])
X0_test =np.ones([3000,1])
train = np.genfromtxt('train.txt', delimiter='\t')
test = np.genfromtxt('test.txt', delimiter='\t')
train = np.append(X0_train , train ,axis=1)
test = np.append(X0_test , test ,axis=1)

X_train = np.hsplit(train,[11])[0]  ## saparate the array into two part
 on the base of the nth column
Y_train = np.hsplit(train,[11])[1]
X_test = np.hsplit(test,[11])[0]  ## saparate the array into two part o
n the base of the nth column
Y_test = np.hsplit(test,[11])[1]
Wlin = np.dot(np.linalg.pinv(X_train) , Y_train)

def sign(s):
    if s >= 0:
```

```
        return 1
    else:
        return -1

E_train = np.dot(X_train , Wlin) * Y_train  #
E_test = np.dot(X_test , Wlin) * Y_test
Ein = 0
Eout = 0
for i in range(1000):
    if sign(E_train[i]) == -1:
        Ein += 1/1000
    if sign(E_test[i]) == -1:
        Eout += 1/1000

print(Ein)
print(Eout)
print(abs(Ein-Eout))
```

20:

```
import numpy as np


X0_train = np.ones([1000,1])
X0_test =np.ones([3000,1])

train = np.genfromtxt('train.txt', delimiter='\t')
X_train = np.hsplit(train,[10])[0]  ## saparate the array into two part
 on the base of the nth column
Y_train = np.hsplit(train,[10])[1]
test = np.genfromtxt('test.txt', delimiter='\t')
X_test = np.hsplit(test,[10])[0]  ## saparate the array into two part o
n the base of the nth column
Y_test = np.hsplit(test,[10])[1]

Q = 10
Z_train = X0_train
```

```
Z_test = X0_test
for q in range(1,Q+1):
    Z_train2 = np.power(X_train , q)
    Z_test2 = np.power(X_test , q)
    Z_train = np.append(Z_train , Z_train2 ,axis = 1)
    Z_test = np.append(Z_test , Z_test2 ,axis = 1)

g = np.dot(np.linalg.pinv(Z_train) , Y_train)

def sign(s):
    if s >= 0:
        return 1
    else:
        return -1

E_train = np.dot(Z_train , g) * Y_train  #
E_test = np.dot(Z_test , g) * Y_test
Ein = 0
Eout = 0
for i in range(1000):
    if sign(E_train[i]) == -1:
        Ein += 1/1000
for i in range(3000):
    if sign(E_test[i]) == -1:
        Eout += 1/3000

print(Ein)
print(Eout)

print(abs(Ein-Eout))
```