1 $G(x) = \text{sign}\left(\sum_{t=1}^{11} g_t(x)\right)$ N 筆資料

(c)

$$
\begin{array}{c|cccc}
 & 1 & 2 & \cdots & N \\
e_1 & e_{1,1} & e_{1,2} & \cdots & e_{1,N} \\
e_2 & e_{2,1} & e_{2,2} & & \vdots \\
\vdots & \vdots & & \ddots & \vdots \\
e_{11} & e_{11,1} & e_{11,2} & \cdots & e_{11,N} \\
\hline
E_{out}(G) & g_1 & g_2 & & g_N
\end{array}
$$

對每個資料 至少要有 6個 $g_t(x)$ 有error $G(x)$ 才有error

$$e_t = \frac{1}{N}(e_{t,1} + e_{t,2} + \cdots e_{t,N}) \quad t = 1 \sim 11$$

$$E_{out}(G) = \frac{1}{N}(g_1 + g_2 + \cdots g_N)$$

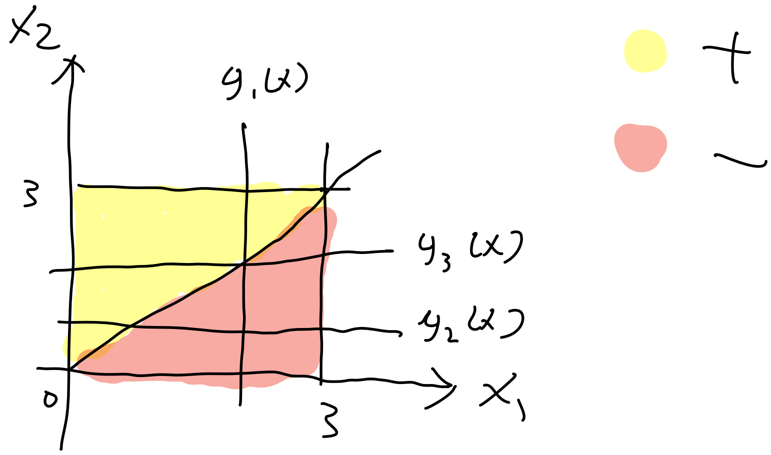
$$g_i \le \frac{1}{6}(e_{1,i} + e_{2,i} + \cdots e_{11,i})$$

$$\Downarrow$$

$$E_{out}(G) \le \frac{1}{6}(e_1 + e_2 + \cdots e_{11})$$

2

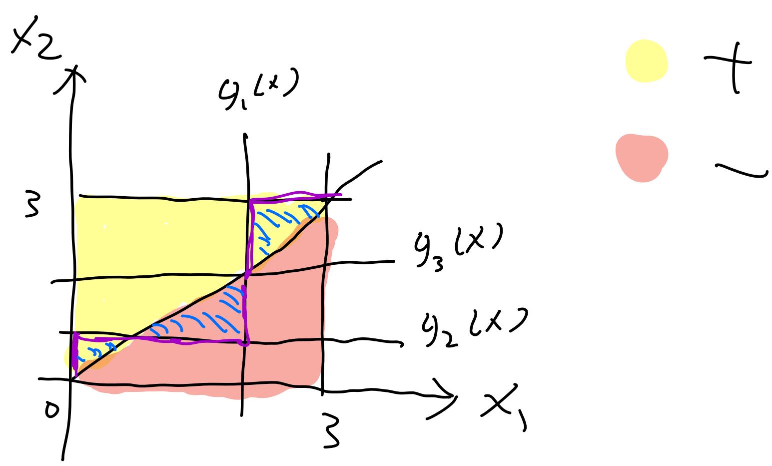
(d)



$G(x)$ 是 g_1、g_2、g_3 線性組合取 sign

所以可以想成以 $g_1 \sim g_3$ 的邊界畫出

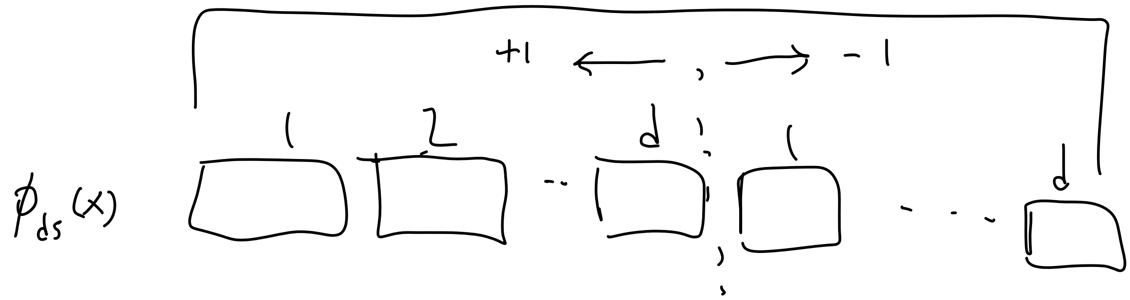
跟 $x_2 - x_1$ 最近的線



$$\frac{1}{18} \times 3 = \frac{3}{18}$$

3

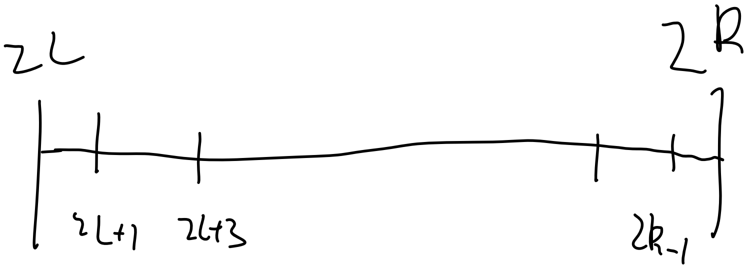
(a) $2 \times d \times (R-L)$



$(+1)^2 \operatorname{sign}(x_1-\theta_1)\operatorname{sign}(x_1'-\theta_1)$ 左右一樣

$(-1)^2 \operatorname{sign}(x_1-\theta_1)\operatorname{sign}(x_1'-\theta_1)$

$$2\times\left(\sum_{i=1}^{d}\left(R-L-|x_i'-x_i|\right)\right) = 2d(R-L)-2|x_i'-x_i|$$



for each x_i x_i':

$-1: \frac{|x_i'-x_i|}{2}$ $\quad +1: R-L-\frac{|x_i'-x_i|}{2}$

4

$$u_n^{(1)} = \frac{1}{N} \qquad n = 1 \sim N$$

(c)

incorrect : $u_n^{(2)} \leftarrow u_n^{(1)} \cdot \lozenge$ 1%

correct : $u_n^{(2)} \leftarrow u_n^{(1)} / \lozenge$ 99%

$$\lozenge = \sqrt{\frac{1-\epsilon}{\epsilon}} \qquad \epsilon = \frac{\sum_{n=1}^{N} u_n^{(1)} [y_n \neq g(x_n)]}{\sum_{n=1}^{N} u_n^{(1)}}$$

$$g_1(x) = +1$$

$$\Rightarrow \epsilon = \frac{0.01N \frac{1}{N}}{N \cdot \frac{1}{N}} = 0.01$$

$$\Rightarrow \lozenge = \sqrt{\frac{0.99}{0.01}}$$

$\Rightarrow$ incorrect : $u_n^{(2)} = \frac{1}{N}\sqrt{\frac{0.99}{0.01}}$

correct : $u_n^{(2)} = \frac{1}{N} \frac{1}{\sqrt{\frac{0.99}{0.01}}}$

$$\frac{\sum_{n: y_n > 0} u_n^{(2)}}{\sum_{n: y_n < 0} u_n^{(2)}} = \frac{\frac{1}{N} \frac{1}{\sqrt{\frac{0.99}{0.01}}} 0.99}{\frac{1}{N}\sqrt{\frac{0.99}{0.01}} \cdot 0.01} = 1$$

5

(b)

$$G_t(x) = \text{sign}\left(\sum_{\tau=1}^{t} \alpha_\tau g_\tau(x)\right)$$

$$G_{t+1}(x) = \text{sign}\left(\sum_{\tau=1}^{t} \alpha_\tau g_\tau(x) + \underline{\alpha_{T+1} g_{T+1}(x)}\right)$$

$$0 < \epsilon_t < \frac{1}{2} \Rightarrow 1 < \lozenge \Rightarrow 0 < \ln(\lozenge) = \alpha_T$$

✗ • $E_{in}(G_t) \rightarrow E_{in}(G_{t+1})$ $(\alpha_T = \ln(\lozenge)$ 舍$)$

因為 $\alpha_{T+1} g_{T+1}(x)$ 有可能有 error 即使它是好分類器

所以不保證 non-increase

✗ • $E_{out}(G_t) \rightarrow E_{out}(G_{t+1})$

跟 E_{in} 的道理一樣

但 upper bound 會越來越小

✓ • $\sum_{n=1}^{N} u_n^{(t)} \rightarrow \sum_{n=1}^{N} u_n^{(t+1)}$

$$\lozenge_t = \sqrt{\frac{\sum_{n=1}^{N} u_n^{(t)} [y_n = g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}} \begin{matrix} \leftarrow \text{correct} \\ \leftarrow \text{incorrect} \end{matrix}$$

$$\sum_{n=1}^{N} u_n^{(t+1)} = \text{correct} \times \sqrt{\frac{\text{incorrect}}{\text{correct}}} + \text{incorrect} \times \sqrt{\frac{\text{correct}}{\text{incorrect}}} = 2\sqrt{\text{correct} \times \text{incorrect}}$$

$$\sum_{n=1}^{N} u_n^{(t)} = \text{correct} + \text{incorrect}$$

$$\boxed{\frac{A+B}{2} \geq \sqrt{AB} \Rightarrow A+B \geq 2\sqrt{AB}}$$

算幾不等式

✗ • $u_n^{(t)} \rightarrow u_n^{(t+1)}$ incorrect

$$u_n^{(t+1)} = u_n^{(t)} \cdot \lozenge > u_n^{(t)}$$

✓ • $u_n^{(t)} \rightarrow u_n^{(t+1)}$ correct

$$u_n^{(t+1)} = u_n^{(t)} / \lozenge < u_n^{(t)}$$

6 承 problem 5 ③

let correct $= A$

incorrect $= B$

(b)
$$\frac{\sum_{n=1}^{N} u_n^{(t+1)}}{\sum_{n=1}^{N} u_n^{(t)}} = \frac{2\sqrt{AB}}{A+B}$$

$$\epsilon_t = \frac{B}{A+B} \qquad 1-\epsilon_t = \frac{A}{A+B}$$

$$= 2\sqrt{\frac{B}{A+B} \cdot \frac{A}{A+B}}$$

$$= 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

7

(b)

$$E_{in}(G_T) = \frac{1}{N}\sum_{n=1}^{N}\left[\operatorname{sign}\left(\sum_{t=1}^{T}\alpha_t g_t(x_n)\right) \neq y_n\right]$$

$$\leq \frac{1}{N}\sum_{n=1}^{N}\exp\left(-y_n\sum_{t=1}^{T}\alpha_t g_t(x_n)\right) = \sum_{n=1}^{N} u_n^{(T+1)}$$

prob 6 $= \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq \left(2\sqrt{\epsilon(1-\epsilon)}\right)^T$

$$\leq \exp\left(-2T\left(\tfrac{1}{2}-\epsilon\right)^2\right)$$

if $\exp\left(-2T\left(\frac{1}{2}-\epsilon\right)^2\right) \leq \frac{1}{N}$ $\quad E_{in} = 0$

$$\exp\left(-2T\left(\tfrac{1}{2}-\epsilon\right)^2\right) = \frac{1}{N}$$

$$\Rightarrow \quad -2T\left(\tfrac{1}{2}-\epsilon\right)^2 = -\ln N$$

$$\Rightarrow \quad T = \frac{\ln N}{2\left(\frac{1}{2}-\epsilon\right)^2}$$

8

(d)

$$1 - \text{no duplicated} \geq 50\%$$

$$50\% \geq \text{no duplicated}$$

$$\text{no duplicated} = \frac{P^{1126}_{N'}}{1126^{N'}}$$

$N' = 39$ no duplicated $= 0.5139$

$N' = 40$ no duplicated $= 0.4961$

9

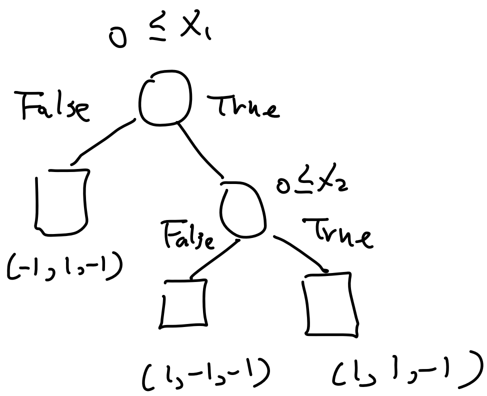
$$\frac{(N-1)^{2N}}{N^{2N}} = \left(\frac{N-1}{N}\right)^{2N}$$

(d)

$$\lim_{n\to\infty} \left(\frac{N-1}{N}\right)^{2N} \simeq 0.13535$$

10

(b)



11 (c) 0.3740000 3

12 (e) 0.5910000 4

(d)

13 (e) 0.0245 | 0.052

14 (b) 0.4550 ... 035

15 (a) 0.2360...018

16 (b) 0.1920 .. 014

```
import numpy as np
import math

def main():
    train_x = []
    train_y = []
    test_x = []
    test_y = []
    train_size = 0
    test_size = 0
    feature = 0
    T = 460
    u_set = [[]]
    g_set = []
    at = []

    with open('hw6_train.dat.txt', 'r') as f:
        ind = 0
        while True:
            line = f.readline()
            if line == '\n' or len(line) == 0:
                break
            x = []
            feature = len(line.split())-1
            for i in line.split()[:-1]:
                x.append(float(i))
            x.append(ind)
            ind+=1
            train_x.append(x)
            train_y.append(float(line.split()[-1]))
            train_size+=1

    with open('hw6_test.dat.txt', 'r') as f:
        while True:
            line = f.readline()
            if line == '\n' or len(line) == 0:
                break
            x = [float(i) for i in line.split()[:-1]]
            test_x.append(x)
            test_y.append(float(line.split()[-1]))
            test_size+=1

    for i in range(train_size):
        u_set[0].append(1/train_size)
    u_sum = 1

    min_ein_g = 1
    for i in range(T):
        # obtain gt
        min = 1
        min_ind = [0, 0, 0]
        for j in range(feature):
            sort_train = sorted(train_x, key = lambda s:s[j])
            for k in range(2):
                min_f = 0
                min_ind_f = 0
                sum_f = 0
                base = 0
                for l in range(train_size-1):
                    if (2*k-1) * train_y[sort_train[l][feature]] == 1.0:
                        sum_f+=u_set[i][sort_train[l][feature]]
                    else:
                        sum_f-=u_set[i][sort_train[l][feature]]
                        base+=u_set[i][sort_train[l][feature]]
                    if sum_f < min_f:
                        min_f = sum_f
```

```python
                min_ind_f = l+1
            min_f = min_f+base
            if min_ind_f == 0:
                min_ind_f = sort_train[0][j] - 1
            else:
                min_ind_f = (sort_train[min_ind_f][j] + sort_train[min_ind_f-1][j]) / 2

            if min_f<min:
                min = min_f
                min_ind = [2*k-1, j, min_ind_f, min_f/u_sum]

        g_set.append(min_ind)

        #compute parm
        parm = math.sqrt((1-min_ind[3])/min_ind[3])

        # update ut to ut+1
        u_sum = 0
        u_set.append([])
        for j in range(train_size):
            if min_ind[0] * np.sign(train_x[j][min_ind[1]] - min_ind[2]) == train_y[j]:
                u_set[i+1].append(u_set[i][j]/parm)
                u_sum+=u_set[i][j]/parm
            else:
                u_set[i+1].append(u_set[i][j]*parm)
                u_sum+=u_set[i][j]*parm

        # compute at
        at.append(math.log(parm))

        # compute ein_g
        ein_g = 0
        for j in range(train_size):
            sum = 0
            for k in range(i+1):
                sum+= at[k] * g_set[k][0] * np.sign(train_x[j][g_set[k][1]] - g_set[k][2])
            if np.sign(sum) != train_y[j]:
                ein_g+=1/train_size
        #print(ein_g)

        if ein_g < min_ein_g:
            min_ein_g = ein_g
    print(min_ein_g)

    eout_g = 0
    for i in range(test_size):
            sum = 0
            for k in range(T):
                sum+= at[k] * g_set[k][0] * np.sign(test_x[i][g_set[k][1]] - g_set[k][2])
            if np.sign(sum) != test_y[i]:
                eout_g+=1/test_size
    print(eout_g)

if __name__ == '__main__':
    main()
```