Machine Learning HW6 吳偉樂 R09946023

1) **(b) 36**

1. (b) 36

$\delta_j^{(\ell)}$ for $\ell \in \{1,2\}$, $j \in \{1,2,\dots,d^{(\ell)}\}$

$\delta_j^{(\ell)} = \sum_{k=1}^{d^{(\ell+1)}} \left(\delta_k^{(\ell+1)} w_{jk}^{(\ell+1)}\right)\left(\tanh'(s_j^{(\ell)})\right)$

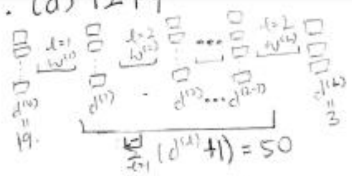
$\delta_j^{(1)} \overset{d^{(2)}=6}{=} \sum_{k=1}^{6} w_{jk}^{(2)} \delta_k^{(2)} \tanh'(s_j^{(1)}) \Longrightarrow j = \{1,2,\dots,d^{(1)}\} = \{1,2,\dots,5\} \Rightarrow$ Each j has 6 operations $5 \times 6 = 30$

$\delta_j^{(2)} \overset{d^{(3)}=1}{=} w_{j1}^{(3)} \delta_1^{(3)} \tanh'(s_j^{(1)}) \Longrightarrow j = \{1,2,\dots,d^{(2)}\} = \{1,2,\dots,6\} \Rightarrow$ Each j has 1 operation $\Rightarrow 6 \times 1 = 6$

Total : 36 operations

2) **(d) 1219**

2. (d) 1219



$\sum_{\ell=1}^{L-1}(d^{(\ell)}+1) = 50$

$, \sum_{\ell=1}^{L-1}(d^{(\ell)}+1) = 50 \Rightarrow d^{(1)} + \dots + d^{(L-1)} = 50-(L-1) = 51-L$

$\#W_{Total} = \#W^{(1)} + \#W^{(2)} + \dots + \#W^{(L-1)} + \#W^{(L)} = 20d^{(1)} + (d^{(1)}+1)d^{(2)} + \dots + (d^{(L-1)}+1)\cdot \underset{=d^{(L)}}{3}$

when $L=2$, then $d^{(1)} = 49 \Longrightarrow \#W_{Total} = 20d^{(1)} + (d^{(1)}+1)\cdot 3 = 20(49) + (50\times 3) = 1130$

Note : $L_{max} = 26$ and $L_{min} = 2$,

where $L_{max} = 26$ implies all $d^{(\ell)} = 1$ in $\sum_{\ell=1}^{L-1}(d^{\ell}+1)$

I wrote the code for the case $3 \le L \le 26$, where L in an integer

```
def part(n, k):
    def _part(n, k, pre):
        if n <= 0:
            return []
        if k == 1:
            if n <= pre:
                return [[n]]
            return []
        ret = []
        for i in range(min(pre, n), 0, -1):
            ret += [[i] + sub for sub in _part(n-i, k-1, i)]
        return ret
    return _part(n, k, n)
```

```
W_in_layer = []

for L in range(3,27):

    P = part(51-L,L-1)
    IND=[]
    for j in range(len(P)):
        S=0
        lis = P[j]
        N = len(lis)
        for i in range(N):
            if i==0:
                S+= 20*lis[i]+(lis[i]+1)*lis[i+1]
            elif i==N-1:
                S+= (lis[i]+1)*3
            else:
                S+= (lis[i]+1)*lis[i+1]
        IND.append(S)
    W_in_layer.append([f'Layer {L}',max(IND)])
print('Max number in each layer:\n',W_in_layer)
```

```
Max number in each layer:
 [['Layer 3', 1219], ['Layer 4', 1123], ['Layer 5', 1058], ['Layer 6', 995], ['Layer 7', 934], ['Layer 8', 875], ['Layer 9', 81
8], ['Layer 10', 763], ['Layer 11', 710], ['Layer 12', 659], ['Layer 13', 610], ['Layer 14', 563], ['Layer 15', 518], ['Layer 1
6', 475], ['Layer 17', 434], ['Layer 18', 394], ['Layer 19', 354], ['Layer 20', 314], ['Layer 21', 274], ['Layer 22', 234], ['L
ayer 23', 194], ['Layer 24', 154], ['Layer 25', 114], ['Layer 26', 74]]
```

3) **(d)**

3. (d) $q_k - v_k$

$V = [\,[\![y=1]\!], \dots, [\![y=K]\!]\,] = [v_1, v_2, \dots, v_K]$

$x^{(L)} = \left[\frac{e^{s_1^{(L)}}}{\sum_{k=1}^{K} e^{s_k^{(L)}}}, \frac{e^{s_2^{(L)}}}{\sum_{k=1}^{K} e^{s_k^{(L)}}}, \dots, \frac{e^{s_K^{(L)}}}{\sum_{k=1}^{K} e^{s_k^{(L)}}}\right] \equiv q = [q_1, q_2, \dots, q_K]$

$err(x, y) = -\sum_{k=1}^{K} v_k \ln q_k$

when $y=k$, $\delta_k^{(L)} = \frac{\partial err}{\partial s_k^{(L)}} = \frac{\partial}{\partial s_k^{(L)}}[-v_k \ln q_k] = \frac{\partial}{\partial s_k^{(L)}}\left[-[\![y=k]\!] \cdot \ln \frac{e^{s_k^{(L)}}}{\sum_{k=1}^{K} e^{s_k^{(L)}}}\right]$

$= -\frac{\cancel{\sum_{k=1}^{K} e^{s_k^{(L)}}}}{\cancel{e^{s_k^{(L)}}}} \cdot \frac{\cancel{e^{s_k^{(L)}}} \sum_{k=1}^{K} e^{s_k^{(L)}} - e^{s_k^{(L)}} \cancel{e^{s_k^{(L)}}}}{(\sum_{k=1}^{K} e^{s_k^{(L)}})^{\cancel{2}}} = -1 + q_k = \underline{-v_k + q_k}$

4) **(a)**

4. (a) 0

4-5-1 NN

$w_{ij}^{(\ell)}$: $1 \le \ell \le 2$, $0 \le i \le d^{(\ell-1)}$, $1 \le j \le d^{(\ell)}$

$w_{ij}^{(1)}$: $\ell = 1$, $0 \le i \le 4$, $1 \le j \le 5$

Initial: $(w_{01}^{(1)})_0 = 0$

update rule: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - x_i^{(0)} \delta_j^{(1)}$

$\Rightarrow w_{01}^{(1)} \leftarrow (w_{01}^{(1)})_0 - \delta_1^{(1)}$

$(w_{01}^{(1)})_1 = -\delta_1^{(1)}$

$(w_{01}^{(1)})_2 = (w_{01}^{(1)})_1 - \delta_1^{(1)} = -2\delta_1^{(1)}$

$(w_{01}^{(1)})_3 = (w_{01}^{(1)})_2 - \delta_1^{(1)} = -3\delta_1^{(1)}$

$\delta_1^{(1)} = \sum_{k=1}^{d^{(2)}} (\delta_k^{(2)})(w_{1k}^{(2)}) \tanh'(s_1^{(1)})$

$= \sum_{k=1}^{d^{(2)}} (\delta_k^{(2)})(w_{1k}^{(2)})(x_1^{(1)})'$

$= \sum_{k=1}^{d^{(2)}} (\delta_k^{(2)})(w_{1k}^{(2)})(1)'$

$= 0$

$\Rightarrow \underline{(w_{01}^{(1)})_3 = 0}$

5) **(e)**

5. (e) half of the average rating of the m-th movie

$\tilde{d} = 1 \Rightarrow V = [V_1 \ V_2 \cdots V_N]_{1\times N}$, where $V_i = 2 \ \forall 1 \le i \le N, i \in \mathbb{N}$ (initial)

$W_m : \tilde{d}\times 1$ dimension $\Rightarrow W_m$ is 1×1 dimension (a scalar)

Step 2.1 : Fix V_n, and minimize E_{in}

$$\Rightarrow \min_{W,V} E_{in}(\{W_m\},\{V_n\}) \propto \sum_{m=1}^{M}\Big(\sum_{(x_n, r_{nm})\in D_m}(r_{nm} - W_m^T V_n)^2\Big)$$

$$\frac{\partial}{\partial W_m}\Big[\sum_{m=1}^{M}\Big(\sum_{(x_n,r_{nm})\in D_m}(r_{nm}-W_m^T V_n)^2\Big)\Big] = \frac{\partial}{\partial W_m}\Big[\sum_n (r_{nm} - W_m^T V_n)^2\Big]$$

$$= -2\sum_n (r_{nm} - W_m^T V_n)V_n = 0 \Rightarrow \sum_{n=1}^{N}(r_{nm} - W_m^T V_n)V_n = 0$$

← assume we have total N users

$$\Rightarrow \text{Since each } V_n = 2 \Rightarrow \sum_{n=1}^{N}(r_{nm} - 2W_m^T) = 0$$

$$\Rightarrow \sum_{n=1}^{N} r_{nm} - 2N W_m^T = 0 \Rightarrow \underline{W_m = \frac{1}{2}\cdot\frac{1}{N}\sum_{n=1}^{N} r_{nm}}$$

6) **(b)**

6. $a_m \leftarrow (1-\eta)a_m + \eta(r_{nm} - W_m^T V_n - b_n)$ (b)

$\text{err}(\text{user } n, \text{movie } m, \text{rating } r_{nm}) = (r_{nm} - W_m^T V_n - a_m - b_n)^2$

$\nabla a_m = -2(r_{nm} - W_m^T V_n - a_m - b_n)$

$\Rightarrow$ per example gradient $\propto -2(\text{residual})$

$\Rightarrow$ SGD update for a_m with learning rate $\frac{\eta}{2}$:

$$a_m \longleftarrow a_m + \frac{\eta}{\cancel{2}}\cancel{2}(r_{nm} - W_m^T V_n - a_m - b_n) = \underline{(1-\eta)a_m + \eta(r_{nm} - W_m^T V_n - b_n)}$$

7) **(d)**

7. (d) [0.16, 0.08, 0.24]

$E_{out}(G) = \frac{1}{N}\sum_{i=1}^{N}[\![\text{sign}(g_1(x_i)+g_2(x_i)+g_3(x_i)) \ne y_i]\!]$, $E_{out}(g_j) = \frac{1}{N}\sum_{i=1}^{N}[\![g_j(x_i)\ne y_i]\!]$ for $j=1,2,3$

$E_{out}(G) = \frac{20}{100} \Rightarrow$ treat as we have total $N=100$ test data, and 20 wrong classification

	y_1 y_2 y_3 ……	y_{100}
TRUE LABLE	O O O O O O O O O O O O O O O O O O O O O O O O ……	O
Result of $g_1(x_i)$	X X X X X X X X X X X X X X X X O O O O O O O O O O ……	O
Result of $g_2(x_i)$	O O O O O O O O O O O O X X X X X X X X O O O O O O ……	O
Result of $g_3(x_i)$	X X X X X X X X X X X X X X X X X X X X X X X X O O ……	O
Result of G	X X X X X X X X X X X X X X X X X X X X O O O O O O ……	O

$\Rightarrow$ Like this, we have $E_{out}(g_1) = 0.16$, $E_{out}(g_2) = 0.08$, $E_{out}(g_3) = 0.24$
and the result $E_{out}(G) = 0.20$

Another view point , we can treat

Eout as 0 <= E_out(G) <= (E_out(g1)+E_out(g2)+E_out(g3))/3

So only (d) satisfies this condition.

8) **(c) 0.32**

$$E_{out}(G) = \binom{5}{3}0.4^3 0.6^2 + \binom{5}{4}0.4^4 0.6 + \binom{5}{5}0.4^5 = 0.31744 \approx 0.32$$

9) **(b)**

9. (b) 60.7%

Bootstrapping to sample 0.5 N examples out of N

$\Rightarrow$ P(An example is not sample) $= \frac{N-1}{N}$

$\Rightarrow$ P(Sample 0.5 N example out of N with replacement) $= \frac{N-1}{N} \cdot \frac{N-1}{N} \cdots \frac{N-1}{N} = \left(\frac{N-1}{N}\right)^{0.5N} = \left(1-\frac{1}{N}\right)^{0.5N}$

when N large, $\lim_{N\to\infty} P(\text{〃}) = \lim_{N\to\infty} \left(1-\frac{1}{N}\right)^{N\cdot 0.5} = e^{-0.5} = 0.6065 \simeq \underline{60.7\%}$

10) **(e) none of the other choices**

10. (e) none of the other choices

$g_{s,i,\theta}(x) = s\ \mathrm{sign}(x_i-\theta)$

$K_{ds}(x,x') = (\phi_{ds}(x))^T(\phi_{ds}(x'))$ ×2 because terms of s=1 equal to terms of s=-1

$= 2[\mathrm{sign}(x_1-(2L+1))\,\mathrm{sign}(x_1'-(2L+1)) + \dots + \mathrm{sign}(x_d-(2R-1))\,\mathrm{sign}(x_d'-(2R-1))]$

$= \sum_{j=1}^{d} \sum_{i=0}^{R-L-1} 2\,\mathrm{sign}(x_j-(2L+1+2i))\,\mathrm{sign}(x_j'-(2L+1+2i))$

Observe: $\mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta) = \begin{cases} -1, & \min\{x_i,x_i'\} < \theta < \max\{x_i,x_i'\} \\ 1, & \text{otherwise} \end{cases}$

① # of -1 depends on how many θ_i between x_j, x_j'

eg: θ_1; 2L 2L+1 2L+2 ... ⇒ $\mathrm{sign}(x_j-\theta_1)\,\mathrm{sign}(x_j'-\theta_1) = -1$

θ_1, θ_2; 2L 2L+1 2L+2 2L+3 2L+4 ⇒ $\mathrm{sign}(x_j-\theta_1)\,\mathrm{sign}(x_j'-\theta_1) = -1$, $\mathrm{sign}(x_j-\theta_2)\,\mathrm{sign}(x_j'-\theta_2) = -1$

⇒ Total # of -1 between x_j, x_j' is $\frac{|x_j-x_j'|}{2}$

② # of 1 is [maximum # of 1 minus # of -1 occurs] ⇒ $R-L-\frac{|x_j-x_j'|}{2}$

By ①,②, then for each j, we have

$$2\sum_{i=0}^{R-L-1} \mathrm{sign}(x_j-(2L+1+2i))\,\mathrm{sign}(x_j'-(2L+1+2i)) = 2\Big[\overbrace{R-L-\frac{|x_j-x_j'|}{2}}^{②} - \overbrace{\frac{|x_j-x_j'|}{2}}^{①}\Big]$$

$$= 2[R-L-|x_j-x_j'|]$$

∀j, we have $2\sum_{j=1}^{d}\sum_{i=0}^{R-L-1} \mathrm{sign}(x_j-(2L+1+2i))\,\mathrm{sign}(x_j'-(2L+1+2i))$

$$= \sum_{j=1}^{d} 2[R-L-|x_j-x_j'|] = \underline{2d(R-L) - 2\|x-x'\|_1}$$

where $\|x-x'\|_1 = |x_1-x_1'| + |x_2-x_2'| + \dots + |x_d-x_d'|$

11) **(a)**

11. (a) 19

$u^{(1)} = [u_1^{(1)}, u_2^{(1)}, \dots, u_N^{(1)}] = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]$, $u_i^{(1)} = \begin{cases} u_+^{(1)} \text{ for positive example} \\ u_-^{(1)} \text{ for negative example} \end{cases} = \frac{1}{N}$

$g_1 = -1$

wrong classification for 5% positive examples

$$\epsilon_1 = \frac{\sum_{n=1}^{N} u_n^{(1)} [\![y_n \ne g_1(x_n)]\!]}{\underbrace{\sum_{n=1}^{N} u_n^{(1)}}_{=1}} = \sum_{n=1}^{N} u_n^{(1)} [\![y_n \ne -1]\!] = \frac{1}{N}(1+1+\dots+1) = \frac{0.5N}{N} = 0.5$$

$\blacklozenge_1 = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{95}{5}} = \sqrt{19}$

incorrect examples: y_i is 1 but g_1 is -1 ($[\![1 = y_n \ne g_1(x_n) = -1]\!]$) : $u_+^{(2)} = u_+^{(1)} \cdot \blacklozenge_1$

correct examples: y_i is -1, g_1 is also -1 ($[\![-1 = y_n = g_1(x_n) = -1]\!]$) : $u_-^{(2)} = u_-^{(1)} / \blacklozenge_1$

$$\frac{u_+^{(2)}}{u_-^{(2)}} = \frac{u_+^{(1)} \cdot \blacklozenge_1}{u_-^{(1)} / \blacklozenge_1} = (\blacklozenge_1)^2 = (\sqrt{19})^2 = \underline{19}$$

12) **(d)**

12. (d) $E_{in}(G_T) \le e^{-2T(\frac{1}{2}-\varepsilon)^2}$

$$\frac{U_{t+1}}{U_t} = \frac{\sum_{n=1}^{N} u_n^{(t+1)}}{\sum_{n=1}^{N} (u_n^{(t)})} = \frac{\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq g_t(x_n)]\!] + \sum_{n=1}^{N} u_n^{(t+1)} [\![y_n = g_t(x_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!] + \sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]} = \frac{\overbrace{\blacklozenge_t \sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!]}^{\text{incorrect example}} + \overbrace{\frac{1}{\blacklozenge_t} \sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]}^{\text{correct example}}}{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!] + \sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]}$$

$$= \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \cdot \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!] + \sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]} + \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!] + \sum_{n=1}^{N} u_n^{(t)} [\![y_n = g_t(x_n)]\!]} \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}$$

$$= \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\, \varepsilon_t + (1-\varepsilon_t)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} = \sqrt{\varepsilon_t(1-\varepsilon_t)} + \sqrt{\varepsilon_t(1-\varepsilon_t)} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

$$\le 2\sqrt{\varepsilon(1-\varepsilon)} \le e^{-2(\frac{1}{2}-\varepsilon)^2}$$

$$\Rightarrow U_{t+1} \le e^{-2(\frac{1}{2}+\varepsilon)^2} U_t$$

$$U_2 \le e^{-2(\frac{1}{2}-\varepsilon)^2} U_1$$

$$U_3 \le e^{-2(\frac{1}{2}-\varepsilon)^2} U_2 \le e^{-2(2)(\frac{1}{2}-\varepsilon)^2} U_1 \qquad \frac{1}{N} + \dots + \frac{1}{N} = 1$$

$$\vdots \qquad\qquad\qquad\qquad\qquad \parallel$$

$$U_{T+1} \le e^{-2T(\frac{1}{2}-\varepsilon)^2} U_1 = e^{-2T(\frac{1}{2}-\varepsilon)^2} \sum_{n=1}^{N} u_n^{(1)} = e^{-2T(\frac{1}{2}-\varepsilon)^2} \Rightarrow \underline{E_{in}(G_T) \le U_{T+1} \le e^{-2T(\frac{1}{2}-\varepsilon)^2}}$$

13) **(d)**

13. (d) the closeness, $1 - |\mu_+ - \mu_-|$

Normalized classification error $= 2\min(\mu_+, \mu_-)$

closeness : $I = 1 - |\mu_+ - \mu_-|$ (impurity function) $\Rightarrow 0 \le I \le 1$

$\Rightarrow \max I = 1$ (at $\mu_+ = \mu_- = 0.5$)

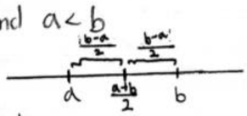
Normalized impurity function $= \frac{I}{\max I} = I = 1 - |\mu_+ - \mu_-|$

Claim : $2\min(\mu_+, \mu_-) = 1 - |\mu_+ - \mu_-|$ (Note : $\mu_+ + \mu_- = 1$)

<u>proof</u>

① Let $a, b \in \mathbb{R}$, and $a < b$

then we have :



$\Rightarrow a = \frac{a+b}{2} - \frac{b-a}{2}$

② when $a, b \in \mathbb{R}$, $b < a$, similarly we have $b = \frac{a+b}{2} - \frac{a-b}{2}$

By ①,②, we have $\forall x, y \in \mathbb{R}$,

$$\frac{x+y}{2} - \frac{|x-y|}{2} = \min\{x, y\}$$

$$\Rightarrow |x+y| - |x-y| = 2\min\{x, y\}$$

$$\Rightarrow |\mu_+ + \mu_-| - |\mu_+ - \mu_-| = 2\min\{\mu_+, \mu_-\}$$

$$\Rightarrow 1 - |\mu_+ - \mu_-| = 2\min\{\mu_+, \mu_-\}$$

14) **(c) 0.18**

Data extraction :

```python
import numpy as np
import random
import math
```

```python
#Train Data
dat_file = r'C://Users//USER//Desktop//hw6_train.dat.txt'
with open(dat_file, 'r') as f:
    text = f.read()

data = text.split()  #split string into a list
data = [float(i) for i in data]
D = []
for i in range(1000):
    K = list(data[i*11:(i+1)*11])
    D.append(K)

D = np.array(D)
X = D[:,:-1]
Y = D[:,-1]
```

```python
#Test Data
test_file = r'C://Users//USER//Desktop//hw6_test.dat.txt'
with open(test_file, 'r') as f:
    text = f.read()

test = text.split()  #split string into a list
test = [float(i) for i in test]

Test_data = []
for i in range(1000):
    K = list(test[i*11:(i+1)*11])
    Test_data.append(K)

Test_data = np.array(Test_data)
X_test = Test_data[:,:-1]
Y_test = Test_data[:,-1]
```

Written function :

```python
def gini_index(y):
    if len(y)==0:
        return 1
    k1 = (abs(sum(y[y==1]))/len(y))**2
    k2 = (abs(sum(y[y==-1]))/len(y))**2
    G = 1-k1-k2
    return G

def possible_theta(x):
    x = sorted(x)
    THETA = [x[0]-1] + [(x[i]+x[i+1])/2 for i in range(len(x)-1)] + [x[-1]+1]
    return THETA

def best_para(x,y,C):
    b_min = 999
    for theta in C:
        if y[x < theta].size == 0:
            continue
        b1 = y[x < theta ]
        b2 = y[x > theta ]
        b = (len(b1)/N)*gini_index(b1) + (len(b2)/N)*gini_index(b2)
        if b < b_min:
            b_min = b
            best_theta = theta
    return b_min,best_theta
```

```python
def decision_stump(X,Y):
    global N
    N = len(Y)
    BEST = 999
    for i in range(10):
        I = [0,0,0,0,0,0,0,0,0,0]
        I[i]=1
        x1 = np.array(X).dot(np.array(I))
        feature = sorted([[x1[k]]+[Y[k]] for k in range(len(Y))],key = lambda x: x[0])
        feature_x = np.array(feature).dot([1,0])
        feature_y = np.array(feature).dot([0,1])
        P = best_para(feature_x,feature_y,possible_theta(feature_x))
        if P[0]<BEST:
            BEST = P[0]
            index = i
            theta = P[1]
    return(BEST,index,theta)

def isStop(x, y):
    n1 = np.sum(y != y[0])
    n2 = np.sum(x != x[0, :])
    return n1 == 0 or n2 == 0

class Dtree:
    def __init__(self, theta, d, value=None):
        self.theta = theta
        self.d = d
        self.value = value
        self.left = None
        self.right = None

def predict(tree, x):
    if tree.value != None:
        return tree.value
    if x[tree.d] < tree.theta:
        return predict(tree.left, x)
    else:
        return predict(tree.right, x)
```

```python
def error(tree, X, y):
    ypredict = [predict(tree, x) for x in X]
    return np.mean(ypredict != y)

def y_predict(tree,X):
    ypredict = [predict(tree, x) for x in X]
    return ypredict

NUM = 0
def learntree(X, y):
    global NUM
    NUM += 1
    if isStop(X,y):
        return Dtree(None, None, y[0])
    else:
        score, d, theta = decision_stump(X, y)
        tree = Dtree(theta, d)
        i1 = X[:, d] < theta
        X1 = X[i1]
        y1 = y[i1]
        i2 = X[:, d] >= theta
        X2 = X[i2]
        y2 = y[i2]
        leftTree = learntree(X1, y1)
        rightTree = learntree(X2, y2)
        tree.left = leftTree
        tree.right = rightTree
        return tree
```

```python
# Q14
L = learntree(X, Y)
print(error(L, X_test, Y_test))
```

```
0.166
```

Note : Dtree and learntree function is from reference https://github.com/Doraemonzzz/ML-Foundation-and-ML-Techniques/blob/master/hw7/%E5%8F%B0%E5%A4%A7%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E4%BD%9C%E4%B8%9A%E4%B8%83.pdf

15) **(d) 0.23**

```
# Q15
max_iter = 2000
i = 0
E_Record = []
while i != max_iter:
    samp_D = D[np.random.randint(X.shape[0], size=500), :]
    samp_X = samp_D[:,:-1]
    samp_Y = samp_D[:,-1]
    L = learntree(samp_X, samp_Y)
    E_Record.append(error(L, X_test, Y_test))
    i+=1
print(f'The mean of sum of Eout = {np.mean(E_Record)}')
```

```
The mean of sum of Eout = 0.23597500000000002
```

16) **(a) 0.01**

17) **(d) 0.16**

18) **(b) 0.07**

(Wrote in same code) :

```
# Q16, Q17, Q18
max_iter = 2000
i = 0
Y_Record = [0 for i in range(1000)]    # for Q16
test_Record = [0 for i in range(1000)] # for Q17
oob_Record = [0 for i in range(1000)]  # for Q18


while i != max_iter:
    samp_D = D[np.random.randint(X.shape[0], size=500), :]
    samp_X = samp_D[:,:-1]
    samp_Y = samp_D[:,-1]
    L = learntree(samp_X, samp_Y)
    y_pred = y_predict(L,X)            # for Q16
    y_test_pred = y_predict(L,X_test)  # for Q17

    for j in range(1000):
        Y_Record[j] += y_pred[j]               # for Q17
        test_Record[j] += y_test_pred[j]       # for Q16

    # for Q 18 :
    new_samp_D = [list(row) for row in samp_D]
    IND = []
    not_used = []
    for j in range(len(D)):
        if list(D[j]) not in new_samp_D:
            not_used.append(D[j])
            IND.append(j)
    not_used = np.array(not_used)
    #not_used = np.array([D[i] for i in range(len(D)) if list(D[i]) not in new_samp_D])
    not_used_X = not_used[:,:-1]
    not_used_Y = not_used[:,-1]
    not_use_pred = y_predict(L,not_used_X)

    for k in range(len(not_use_pred)):
        oob_Record[IND[k]] += not_use_pred[k]
    i+=1
```

```
for i in range(len(Y_Record)):
    if Y_Record[i]>=0:
        Y_Record[i]=1
    else:
        Y_Record[i]=-1
print(f'E_in(G) = {np.mean(Y_Record != Y)}')    # Q16

for i in range(len(test_Record)):
    if test_Record[i]>=0:
        test_Record[i]=1
    else:
        test_Record[i]=-1
print(f'E_out(G) = {np.mean(test_Record != Y_test)}')    # Q17

for i in range(len(oob_Record)):
    if oob_Record[i]>=0:
        oob_Record[i]=1
    else:
        oob_Record[i]=-1
print(f'E_oob(G) = {np.mean(oob_Record != Y)}')      # Q18
```

```
E_in(G) = 0.013
E_out(G) = 0.155
E_oob(G) = 0.07
```

19) **(a) Support Vector Machine**

SVM 是實務中很常見的模型，每個人都會使用，包括我。但是以往我只是照著 code 輸入模型，更改參數，並不了解裡面的理論。上過其他老師的課，以及自己讀一些相關文章，裡面過於數學及複雜且抽象的理論知識非常難讓人理解。直到上了林老師的課，老師用比較淺顯易懂的方式，以及豐富的投影片（老師的投影片真的做的非常好，顏色不一的字體比較能讓大家吸收且注意）為大家講解了 SVM 的背後的數學理論。從 Hard Margin 到對偶問題，再到 Soft Margin，我都聽得津津有味。聽了林老師的課，我發現自己不是不會數學，每一條公式都看得懂，但是卻缺乏了理解公式背後的意義，我也相信“公式背後的意義”是大多數同學都難以頓悟的，很高興能從老師的課裡面學會並理解這些知識。作業 5 完全都是 SVM 相關的題目，在解題的過程也能學習到蠻多 SVM 的問題，當我解題時，不是盲目的解，而是會思考，這道題目背後，是要告訴我們什麼呢。謝謝老師為我們帶來那麼棒的課！ Respect!

20) **(b) matrix factorization**

個人覺得此單元缺少了一些實際操作的過程（例如寫相關程式）。我自己本身在吸收知識的時候，可能覺得自己了解了，但當真正實做的時候，會發現自己還有一些模糊的地方（尤其在手刻 Decision tree 時 XD）。這個單元我重複看了兩次，還是隱約覺得自己似乎哪裡還搞不清楚，但又無從考證～
所以若有機會，希望可以在實務上運用此章節所學，相信會讓同學們更了解！