Homework 5

1. (d)

A

1. We know that $\xi_i \geq 1$ for misclassified points, and $0 < \xi_i < 1$ for correct points in the margin, and $\xi_i = 0$ for correct points outside the margin.

So $\sum_{n=1}^{N} \xi_n^* \geq \sum_{i \in A} \xi_i^* \geq \sum_{i \in A} 1$, by similar idea we can find

$\xi_i \geq 1$, $(\xi_i)^{\frac{1}{2}} \geq 1$, $\lfloor \xi_i \rfloor \geq 1$, $\log_2(1+\xi_i) \geq 1$ for $i \in A$

2. (c)

2. By KKT conditions we have:

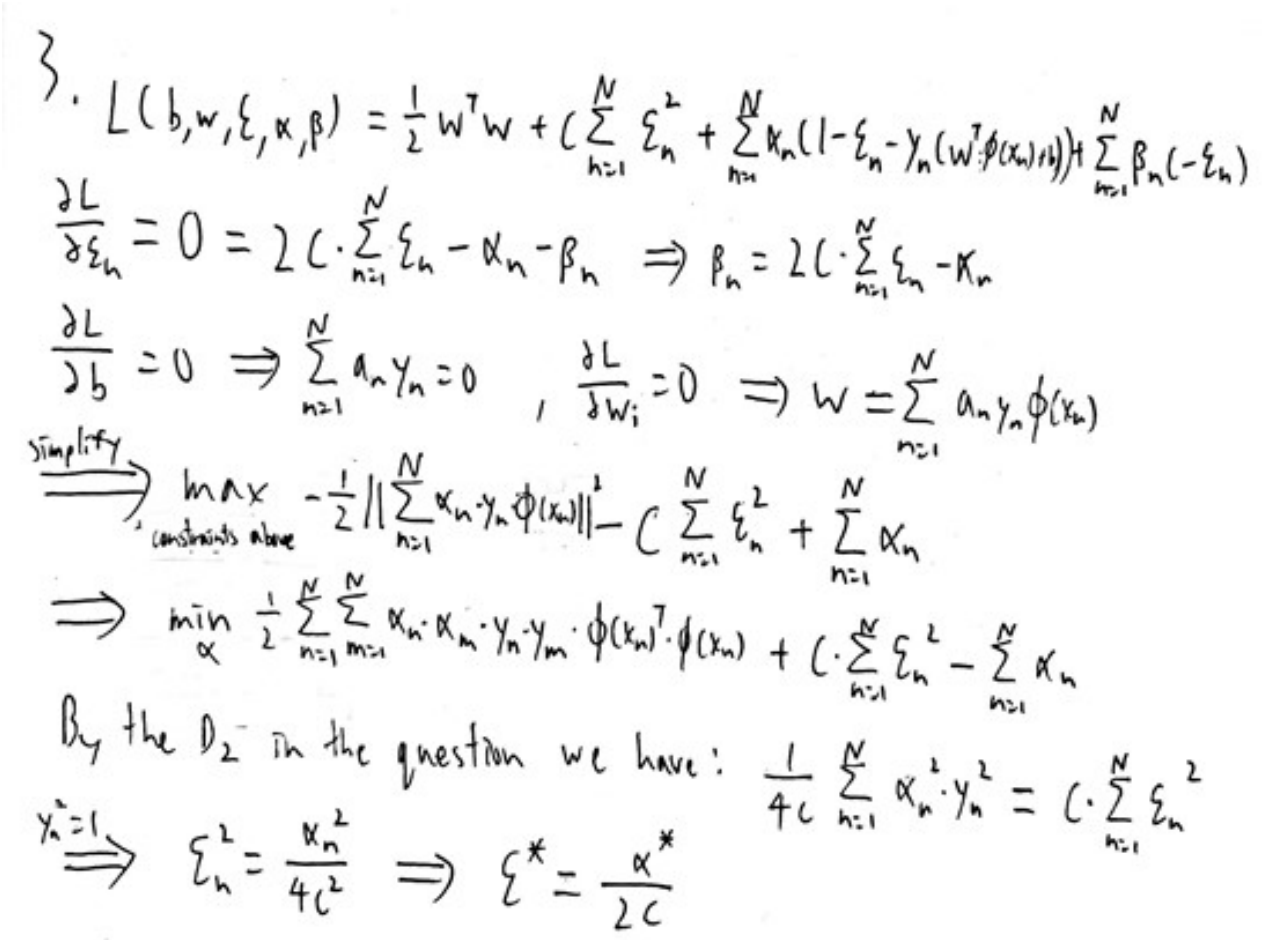
$$y_n(w^T z_n + b) \geq 1,\ \alpha_n \geq 0,\ w = \sum \alpha_n y_n z_n,\ \alpha_n(1 - y_n(w^T z_n + b)) = 0$$

Since we know $\alpha_n^* = C > 0 \Rightarrow b^* = y_n - w^T z_n$

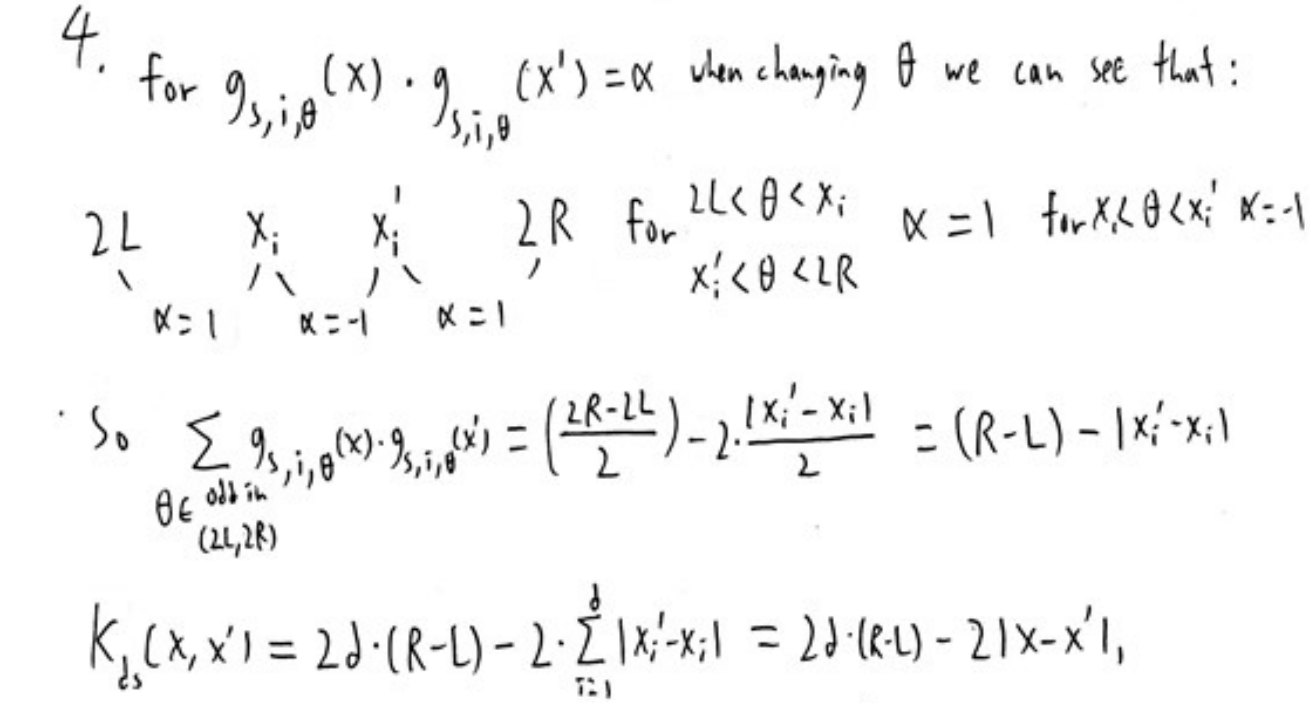
$$\Rightarrow b^* = y_n - \sum_{m=1}^{N} \alpha_m^* y_m z_m^T z_n$$

$$\xrightarrow{\text{smallest } b^*} b^*_{\min} = \min_{n:\, y_n < 0} \left(-1 - \sum_{m=1}^{N} y_m \cdot \alpha_m^* \cdot K(x_n, x_m)\right)$$

3. (a)


3. $L(b,w,\xi,\alpha,\beta) = \frac{1}{2}w^Tw + C\sum_{n=1}^{N}\xi_n^2 + \sum_{n=1}^{N}\alpha_n(1-\xi_n-y_n(w^T\phi(x_n)+b)) + \sum_{n=1}^{N}\beta_n(-\xi_n)$
$\frac{\partial L}{\partial \xi_n} = 0 = 2C\cdot\sum_{n=1}^{N}\xi_n - \alpha_n - \beta_n \Rightarrow \beta_n = 2C\cdot\sum_{n=1}^{N}\xi_n - \alpha_n$
$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N}a_n y_n = 0$, $\frac{\partial L}{\partial w_i} = 0 \Rightarrow w = \sum_{n=1}^{N}a_n y_n \phi(x_n)$
simplify
$\max_{\text{constraints above}} -\frac{1}{2}\|\sum_{n=1}^{N}\alpha_n y_n \phi(x_n)\|^2 - C\sum_{n=1}^{N}\xi_n^2 + \sum_{n=1}^{N}\alpha_n$
$\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\cdot\alpha_m\cdot y_n\cdot y_m\cdot\phi(x_n)^T\cdot\phi(x_m) + C\cdot\sum_{n=1}^{N}\xi_n^2 - \sum_{n=1}^{N}\alpha_n$
By the D_2 in the question we have: $\frac{1}{4C}\sum_{n=1}^{N}\alpha_n^2\cdot y_n^2 = C\cdot\sum_{n=1}^{N}\xi_n^2$
$y_n^2=1 \Rightarrow \xi_n^2 = \frac{\alpha_n^2}{4C^2} \Rightarrow \xi^* = \frac{\alpha^*}{2C}$

4. (e)


4. for $g_{s,i,\theta}(x)\cdot g_{s,i,\theta}(x') = \alpha$ when changing θ we can see that:
2L x_i x_i' 2R for $2L<\theta<x_i$ $\alpha=1$ for $x_i<\theta<x_i'$ $\alpha=-1$
$x_i'<\theta<2R$
$\alpha=1$ $\alpha=-1$ $\alpha=1$
So $\sum_{\theta\in \text{odd in } (2L,2R)} g_{s,i,\theta}(x)\cdot g_{s,i,\theta}(x') = \left(\frac{2R-2L}{2}\right) - 2\cdot\frac{|x_i'-x_i|}{2} = (R-L) - |x_i'-x_i|$
$K_{ds}(x,x') = 2d\cdot(R-L) - 2\cdot\sum_{i=1}^{d}|x_i'-x_i| = 2d\cdot(R-L) - 2|x-x'|_1$

5. (b)

5. We know that $E_{out}(g_t) = \frac{\sum_{i=1}^{N} [\![y_i \neq g_t(x_i)]\!]}{N} = e_t$

Since for a wrong $g_t(x_i)$ to dominate there must be $\geq M+1$ classifies that is wrong too.

$$\max E_{out}(G) = \frac{\sum_{t=1}^{2M+1} \sum_{i=1}^{N} [\![y_i \neq g_t(x_i)]\!]}{(M+1) \cdot N} = \frac{1}{M+1} \cdot \sum_{t=1}^{2M+1} e_t$$

Illustration:

$g_1(x)$ W R W ..... (N)

$g_2(x)$ R R W .....

$\vdots$

$g_{2M+1}(x)$ W W R .....

The max $E_{out}(G)$ happens when there is exactly $M+1$ W in the wrong column, that is

$$\text{Max wrong columns} = \frac{\text{total W}}{M+1}$$

↓

for a column to be wrong there must be $\geq M+1$ wrong g

6. (b)

6. Similar idea as the birthday paradox:

$$1 - p(n) = 1 \cdot (1 - \tfrac{1}{1129}) \cdot (1 - \tfrac{2}{1129}) \cdots (1 - \tfrac{n-1}{1129}) \doteq e^{\frac{-n \cdot (n-1)}{2 \cdot 1129}} \leq 0.25$$

$$n \doteq 56$$

7. (c)

7. $$\min_{w} E_{in}^{u}(w) = \frac{1}{N} \sum_{n=1}^{N} u_n (y_n - w^T x_n)^2 = \frac{1}{N} \sum_{n=1}^{N} (\tilde{y}_n - w^T \tilde{x}_n)^2$$

$$\Rightarrow \tilde{x}_n = \sqrt{u_n} \cdot x_n, \quad \tilde{y}_n = \sqrt{u_n} \cdot y_n$$

8. (e)

8. we want to choose the one with smallest Gini index

$Gini(S) = 1 - \sum_{j=1}^{n} p_j^2$ $\quad Gini_A(S) = \frac{|S_1|}{S} Gini(S_1) + \frac{|S_2|}{S} Gini(S_2)$

(a) first part : $1 - (1^2 + 0) = 0$

second part : $1 - (0.5^2 + 0.5^2) = 0.5$

total : $0.5 \cdot 0 + 0.5 \cdot 0.5 = 0.25$

(b) first part : $1 - (0.8^2 + 0.2^2) = 0.32$

second part : $1 - (0.75^2 + 0.25^2) = 0.375$

total : $0.7 \cdot 0.32 + 0.3 \cdot 0.375 = 0.3365$

(c) first part : 0.42 , second part : 0 , total : 0.378

(d) first part : 0.32 , second part : 0.18 , total : 0.292

(e) first part : 0.18 , second part : 0.18 , total : 0.18

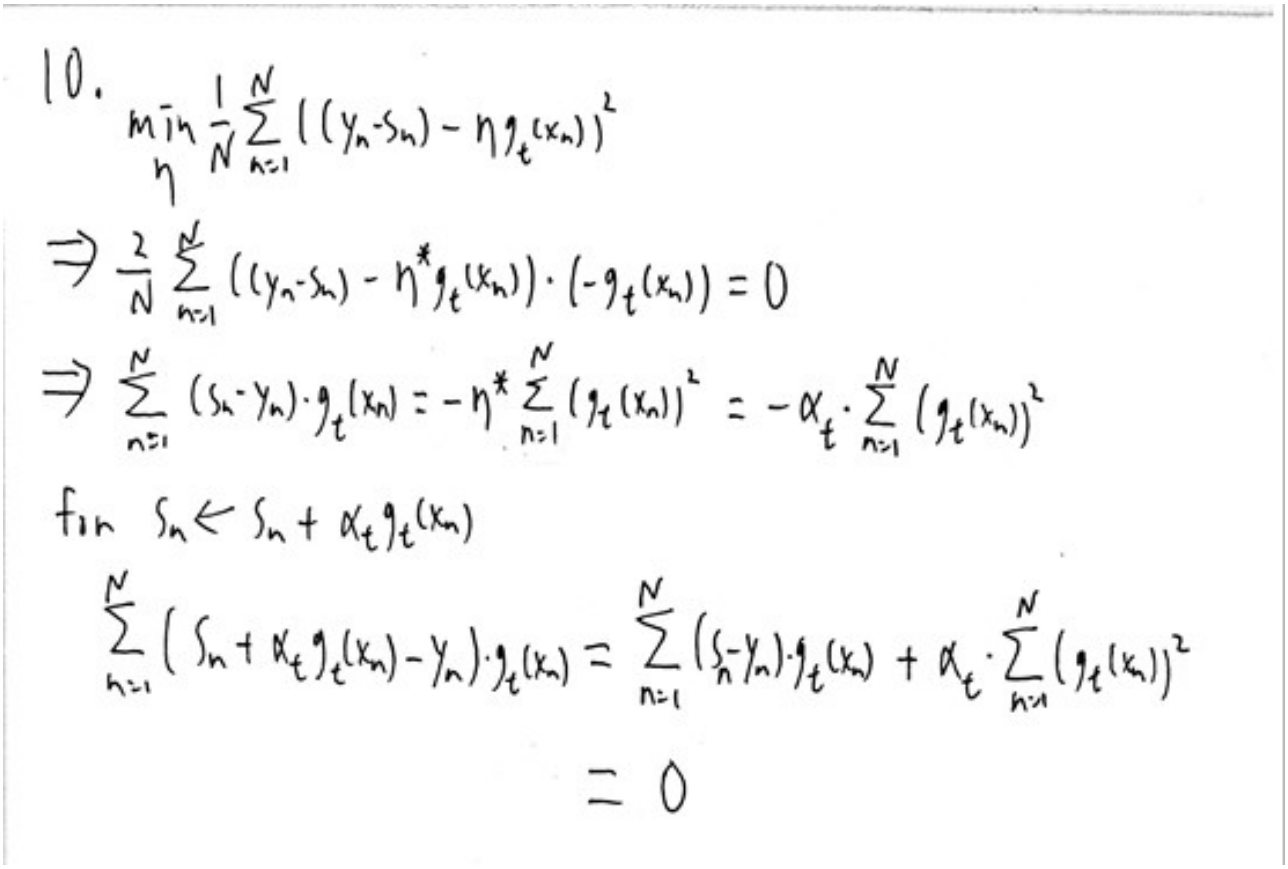
9. (d)

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

$= \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \cdot V_t \cdot \epsilon_t + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \cdot V_t \cdot (1-\epsilon_t) = 2\sqrt{\epsilon_t(1-\epsilon_t)} \cdot V_t$

$\xrightarrow{V_1 = 1} V_{T+1} = 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t \cdot (1-\epsilon_t)}$

10. (b)



11. (c) 6.309673609961579
12. (d) [0.011333333333333306, 0.006761904761904747, 0.009619047619047638, 0.014857142857142902, 0.01123809523809527]
13. (b) [588, 368, 499, 642, 503]
14. (d) [0.04519999999999991, 0.04519999999999991, 0.01419999999999999, 0.0040000000000000036, 0.00539999999999996]
15. (c) [0.04519999999999991, 0.04519999999999991, 0.040200000000000014, 0.04519999999999991, 0.04519999999999991]
16. (a) [320, 0, 180, 0, 0]
17. (a) 0.09846547314578005
18. (c) 0.571611253196931
19. (a) 0.0
20. (a) 0.002793296089385475

code:

Q11

```
import numpy as np
from libsvm.svmutil import *
from numpy import linalg as LA

y, X = svm_read_problem('train.txt')
for i in range(len(y)):
    if y[i] != 1:
        y[i] = -1
    else:
        y[i] = 1
prob  = svm_problem(y, X)
param = svm_parameter(f'-t 0 -c 1 -q')
model = svm_train(prob, param)
sv_coef = model.get_sv_coef()
sv = model.get_SV()
w = [0 for _ in range(16)]
for j,x in enumerate(sv):
    for i in range(16):
        w[i] += sv_coef[j][0] * x[i+1]
print(w)
print(LA.norm(w))
```

Q12~13

```
import numpy as np
from libsvm.svmutil import *

E_in_list = []
SV_nr_list = []
for k in [2,3,4,5,6]:
    y, X = svm_read_problem('train.txt')
    for i in range(len(y)):
        if y[i] != k:
            y[i] = -1
        else:
            y[i] = 1
    prob  = svm_problem(y, X)
    param = svm_parameter('-t 1 -g 1 -r 1 -d 2 -c 1 -q')
    model = svm_train(prob, param)
    p_label, p_acc, p_val = svm_predict(y, X, model, '-q')
    E_in_list.append(1 - p_acc[0]/100)
    SV_nr_list.append(model.get_nr_sv())
print(E_in_list)
print(SV_nr_list)
```

Q14

```
import numpy as np
from libsvm.svmutil import *

y, X = svm_read_problem('train.txt')
y_t, X_t = svm_read_problem('test.txt')
for i in range(len(y)):
    if y[i] != 7:
        y[i] = -1
    else:
        y[i] = 1
for i in range(len(y_t)):
    if y_t[i] != 7:
        y_t[i] = -1
    else:
        y_t[i] = 1

E_out_list = []
for k in [0.01, 0.1, 1, 10, 100]:
    prob  = svm_problem(y, X)
    param = svm_parameter(f'-t 2 -g 1 -c {k} -q')
    model = svm_train(prob, param)
    p_label, p_acc, p_val = svm_predict(y_t, X_t, model, '-q')
    E_out_list.append(1 - p_acc[0]/100)
print(E_out_list)
```

Q15

```
import numpy as np
from libsvm.svmutil import *

y, X = svm_read_problem('train.txt')
y_t, X_t = svm_read_problem('test.txt')
for i in range(len(y)):
    if y[i] != 7:
        y[i] = -1
    else:
        y[i] = 1
for i in range(len(y_t)):
    if y_t[i] != 7:
        y_t[i] = -1
    else:
        y_t[i] = 1

E_out_list = []
for k in [0.1, 1, 10, 100, 1000]:
    prob  = svm_problem(y, X)
    param = svm_parameter(f'-t 2 -g {k} -c 0.1 -q')
    model = svm_train(prob, param)
```

```
    p_label, p_acc, p_val = svm_predict(y_t, X_t, model, '-q')
    E_out_list.append(1 - p_acc[0]/100)
print(E_out_list)
```

Q16

```
import numpy as np
from libsvm.svmutil import *
import random
from tqdm import tqdm

y, X = svm_read_problem('train.txt')
for i in range(len(y)):
    if y[i] != 7:
        y[i] = -1
    else:
        y[i] = 1

cnt_list = [0, 0, 0, 0, 0]
for z in tqdm(range(500)):
    tmp = list(zip(y, X))
    random.shuffle(tmp)
    y_tmp, X_tmp = zip(*tmp)
    y_tmp, X_tmp = list(y_tmp), list(X_tmp)
    y_valid, X_valid = y_tmp[:200], X_tmp[:200]
    y_train, X_train = y_tmp[200:], X_tmp[200:]
    E_val_list = []
    for k in [0.1, 1, 10, 100, 1000]:
        prob  = svm_problem(y_train, X_train)
        param = svm_parameter(f'-t 2 -g {k} -c 0.1 -q')
        model_ptr = libsvm.svm_train(prob, param)
        model = toPyModel(model_ptr)
        p_label, p_acc, p_val = svm_predict(y_valid, X_valid, model, '-q')
        E_val_list.append(1 - p_acc[0]/100)
    min = 1
    id = 0
    for k in range(5):
        if(E_val_list[k] < min):
            min = E_val_list[k]
            id = k
    cnt_list[id] += 1
    print(z)
    print(min)
    print(cnt_list)
print(f'done {cnt_list}')
```

Q17~20

```
import numpy as np
import random
import math
from tqdm import tqdm

def decision_stump(t_arr, u_arr):
    s_id_theta = []
    s, id, theta = -1, -1, -1
    INF = 10000
    n = len(t_arr)
    m = 16
    g = 1
    for k in range(m):
        z = t_arr[:, k+1].argsort()
        t_arr = t_arr[z]
        u_arr = u_arr[z]
        for i in [1, -1]:
            tmp_g = 0
            for j in range(n):
                if t_arr[j][0] != i:
                    tmp_g += u_arr[j]
            if tmp_g < g:
                g, s, id, theta = tmp_g, i, k+1, -INF
            for j in range(n-1):
                if t_arr[j][0] == i:
                    tmp_g += u_arr[j]
                else:
                    tmp_g -= u_arr[j]
                if tmp_g < g and t_arr[j][k+1] != t_arr[j+1][k+1]:
                    g, s, id = tmp_g, i, k+1
                    theta = (t_arr[j][k+1]+t_arr[j+1][k+1])/2
    s_id_theta.append(s)
    s_id_theta.append(id)
    s_id_theta.append(theta)
    return s_id_theta

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

train_list, test_list = [], []

with open('train.txt') as file:
    for line in file:
        line = line.strip().split()
        if line[0] == '11':
            tmp = [1]
            for i in range(16):
                line2 = line[i+1].split(':')
                tmp.append(float(line2[1]))
            train_list.append(tmp)
```

```
        elif line[0] == '26':
            tmp = [-1]
            for i in range(16):
                line2 = line[i+1].split(':')
                tmp.append(float(line2[1]))
            train_list.append(tmp)

with open('test.txt') as file:
    for line in file:
        line = line.strip().split()
        if line[0] == '11':
            tmp = [1]
            for i in range(16):
                line2 = line[i+1].split(':')
                tmp.append(float(line2[1]))
            test_list.append(tmp)
        elif line[0] == '26':
            tmp = [-1]
            for i in range(16):
                line2 = line[i+1].split(':')
                tmp.append(float(line2[1]))
            test_list.append(tmp)

t_arr = np.array(train_list)
tt_arr = np.array(train_list)
test_arr = np.array(test_list)
n = len(t_arr)
nn = len(test_list)
u_list = [1/n] * n
u_arr = np.array(u_list)
E_in_list = []
Ein_G_list = [0]*n
Eout_G_list = [0]*nn
for t in tqdm(range(1000)):
    s_id_theta = decision_stump(t_arr, u_arr)
    s, id, theta = s_id_theta[0], s_id_theta[1], s_id_theta[2]
    z = t_arr[:, id].argsort()
    t_arr = t_arr[z]
    u_arr = u_arr[z]
    wrong_data = 0
    epsilon = 0
    for i in range(n):
        if s * sign(t_arr[i][id] - theta) != t_arr[i][0]:
            wrong_data += 1
            epsilon += u_arr[i]
    epsilon = epsilon / sum(u_arr)
    scal_factor = math.sqrt((1-epsilon)/epsilon)
    for i in range(n):
        if s * sign(t_arr[i][id] - theta) == t_arr[i][0]:
            u_arr[i] = u_arr[i] / scal_factor
        else:
            u_arr[i] = u_arr[i] * scal_factor
    E_in_list.append(wrong_data/n)
    a_t = np.log(scal_factor)
```

```
        for i in range(n):
            Ein_G_list[i] += a_t * (s * sign(tt_arr[i][id] - theta))
        for i in range(nn):
            Eout_G_list[i] += a_t * (s * sign(test_arr[i][id] - theta))

    for i in range(n):
        Ein_G_list[i] = sign(Ein_G_list[i])
    Ein_G = 0
    for i in range(n):
        if Ein_G_list[i] != tt_arr[i][0]:
            Ein_G += 1
    Ein_G = Ein_G / n

    for i in range(nn):
        Eout_G_list[i] = sign(Eout_G_list[i])
    Eout_G = 0
    for i in range(nn):
        if Eout_G_list[i] != test_arr[i][0]:
            Eout_G += 1
    Eout_G = Eout_G / nn

    print(min(E_in_list))
    print(max(E_in_list))
    print(Ein_G)
    print(Eout_G)
```