ML HW6

1. 設總共有 N 筆 example，每筆 classifier 分類錯誤的 example 共有 $N \times e_t$，$1 \le t \le 11$

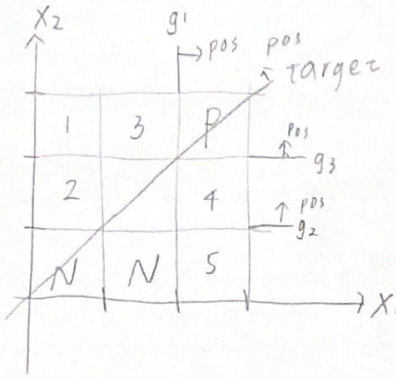
對於 $G(x) = \text{sign}\left(\sum_{t=1}^{11} g_t(x)\right)$，每筆 example 至少要被 6 筆 classifier 分錯才會是錯的

∴ worst-case 為每筆錯誤的 example 都剛好被 6 筆 classifier 分錯

⇒ 最多 $\frac{(N\times e_1 + N\times e_2 + \cdots + N\times e_{11})}{6}$ 筆 example 被分錯 ∴ upper bound of $E_{out}(G) = \frac{N \times \sum_{t=1}^{11} e_t}{6 \cdot N}$

$= \frac{1}{6}\sum_{t=1}^{11} e_t$ #

2.



P: g_1, g_2, g_3 predict positive.

N: g_1, g_2, g_3 predict negative

For region 1, 3, 5: when $\alpha_2 + \alpha_3 > \alpha_1$ ⇒ predict correctly.

For region 2: $\alpha_2 - \alpha_3 > \alpha_1$

For region 4: $\alpha_3 - \alpha_2 > \alpha_1$

> when $\alpha_2 = \alpha_3$, can predict correctly.

For the regions on the diagonal: only a half can predict correctly

∴ $E_{out}(G) = \frac{3}{9} \times \frac{1}{2} = \frac{3}{18}$ #

3. $K_{ds}(X, X') = (\Phi_{ds}(X))^T(\Phi_{ds}(X')) = g_{+1,1,2L+1}(X)\cdot g_{+1,1,2L+1}(X') + g_{+1,1,2L+3}(X)\cdot g_{+1,1,2L+3}(X') + \cdots +$

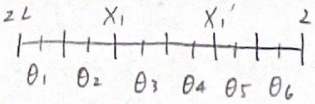
$g_{-1,d,2R-1}(X)\cdot g_{-1,d,2R-1}(X')$

又 $g_{s,i,\theta}(X) = s\cdot\text{sign}(X_i - \theta)$, ⇒ $K_{ds}(X,X') = S^2\cdot\text{sign}(X_1-\theta)\cdot\text{sign}(X_1'-\theta) + \cdots + S^2\cdot\text{sign}(X_d-\theta)\cdot\text{sign}(X_d'-\theta)$

For $S \in \{-1, +1\}$, θ is an odd integer btwn $\{2L, 2R\}$

X, X': even int

θ: odd int btwn $[2L, 2R]$



When $\theta < X_1$ or $\theta > X_1'$ ⇒ $\text{sign}(X_1-\theta)\cdot\text{sign}(X_1'-\theta) = +1$

when $X_1 < \theta < X_1'$ ⇒ $\text{sign}(X_1-\theta)\cdot\text{sign}(X_1'-\theta) = -1$

⇒ $K_{ds}(X,X') = \sum_{\forall s}\sum_{\forall \theta} \text{sign}(X_1-\theta)\cdot\text{sign}(X_1'-\theta) + \cdots + \text{sign}(X_d-\theta)\cdot\text{sign}(X_d'-\theta)$

$= 2\times\left[\left(\frac{2R-2L}{2} - \frac{|X_i'-X_i|}{2}\right)\times 1 + \frac{|X_i'-X_i|}{2}\times(-1)\right]$, $i \in \{1, 2, \ldots, d\}$

$= \sum_{\forall d} 2\left[(R-L) - |X_i'-X_i|\right] = 2d(R-L) - 2\|X-X'\|_1$ #

4. Find $\frac{\sum_{n:y_n>0} u_n^{(2)}}{\sum_{n:y_n<0} u_n^{(2)}}$, For positive examples: $u^{t+1} = u^t / \diamond_t$

For negative examples: $u^{t+1} = u^t \cdot \diamond_t$. $\diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

$\therefore \frac{\sum_{n:y_n>0} u_n^{(2)}}{\sum_{n:y_n<0} u_n^{(2)}} = \frac{0.99 \times N \times 1}{0.01 \times N \times \diamond_t^2} = \frac{99 \cdot \epsilon_t}{1-\epsilon_t} = \frac{99 \cdot 0.01}{0.99} = 1$ #

5. ② : We are not sure for the result of E_{out}.

③ $u_n^{(t+1)} = u_n^{(t)} / \diamond_t$ for correct

④

⑤ $u_n^{(t+1)} = u_n^{(t)} \cdot \diamond_t$ for incorrect example. $\diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, $0 \le \epsilon_t \le \frac{1}{2} \Rightarrow 1 \le \diamond_t \le \infty$

$\because 1 \le \diamond_t \le \infty \quad \therefore$ correct example: non-increasing

incorrect example: non-decreasing.

For ③ : According to Q6, $\frac{U_{t+1}}{U_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$, $\because 0 \le \epsilon_t \le \frac{1}{2}$,

$0 \le \frac{U_{t+1}}{U_t} \le 1 \Rightarrow$ non-increasing

① $E_{in}(G_t) \le U_t$ & U_t is non-increasing.

But we cannot sure that $E_{in}(G_t)$ is also non-increasing.

We can see that $E_{in}(G_t)$ is not non-increasing according to Q13.

(Coding part, print $E_{in}(G_t)$ for $1 \le t \le 460$).

Choose ③, ④

6. $U_t = \sum_{n=1}^{N} u_n^{(t)}$, According to Lecture 13, p.33. $\begin{cases} U_t = \sum_{n=1}^{N} \frac{1}{N} \cdot \exp\left(-y_n \sum_{i=1}^{t-1} \alpha_i g_i(x_n)\right) \\ U_{t+1} = \sum_{n=1}^{N} \frac{1}{N} \exp\left(-y_n \sum_{i=1}^{t} \alpha_i g_i(x_n)\right) \end{cases}$

$U_{t+1} = \sum_{n=1}^{N} \frac{1}{N} \exp\left(-y_n \sum_{i=1}^{t-1} \alpha_i g_i(x_n) - y_n \alpha_t g_t(x_n)\right) = \sum_{n=1}^{N} \frac{1}{N} \exp\left(-y_n \sum_{i=1}^{t-1} \alpha_i g_i(x_n)\right) \cdot \exp\left(-y_n \alpha_t g_t(x_n)\right)$

$= \sum_{n=1}^{N} \frac{1}{N} \exp\left(-y_n \sum_{i=1}^{t-1} \alpha_i g_i(x_n)\right) \cdot \left[\underbrace{\epsilon_t \exp(\alpha_t)}_{\text{incorrect.}} + \underbrace{(1-\epsilon_t) \cdot \exp(-\alpha_t)}_{\text{correct}}\right]$, $\because \alpha_t = \ln\left(\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\right)$

$= \sum_{n=1}^{N} \frac{1}{N} \exp\left(-y_n \sum_{i=1}^{t-1} \alpha_i g_i(x_n)\right) \cdot \left[\sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{(1-\epsilon_t)\cdot\epsilon_t}\right] = U_t \cdot 2\sqrt{\epsilon_t(1-\epsilon_t)}$

$\therefore \frac{U_{t+1}}{U_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ #

ML HW6

7. $E_{in}(G_T)$ = "錯誤數量 / N"，而錯誤數量為 -1 0 到 N 的整數

∴ $E_{in}(G_T)=0$ equal to $E_{in}(G_T) < \frac{1}{N}$

$E_{in}(G_T) \le U_{T+1} = \frac{U_{T+1}}{U_T} \cdot \frac{U_T}{U_{T-1}} \cdots \frac{U_2}{U_1} \cdot U_1$ $(U^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots \frac{1}{N}] \Rightarrow U_1 = 1)$

$= \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right]^T \cdot 1$ (Based on Q_6)

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

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

$= \exp(-2T(\frac{1}{2}-\epsilon)^2)$

$E_{in}(G_T) < \frac{1}{N} \le \exp(-2T(\frac{1}{2}-\epsilon)^2)$

$\Rightarrow -\ln N \le -2T(\frac{1}{2}-\epsilon)^2$

$\Rightarrow \frac{\ln N}{2(\frac{1}{2}-\epsilon)^2} \ge T$, upper bound = $\frac{\ln N}{2(\frac{1}{2}-\epsilon)^2}$ #

8 $1 - P(\text{get all different example}) > 0.5$

$\Rightarrow 1 - \underbrace{\frac{1126}{1126} \times \frac{1125}{1126} \times \frac{1124}{1126} \times \cdots}_{N' \text{項}} > 0.5$

$\Rightarrow \frac{\overbrace{1126 \times 1125 \times \cdots}^{N' \text{項}}}{(1126)^{N'}} < 0.5$ (using python to run the answer)

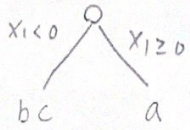
$\Rightarrow 1126 - 1087 + 1 = 40$ #

9. an example is not sampled. probability $= (1 - \frac{1}{N})$

2N examples: $(1 - \frac{1}{N})^{2N} = \left[(1 - \frac{1}{N})^{N}\right]^2$ (when N is large, $(1 - \frac{1}{N})^{N} \fallingdotseq \frac{1}{e}$)

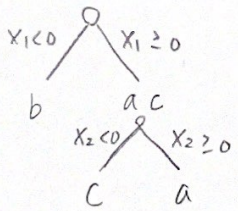
$= (\frac{1}{e})^2 \fallingdotseq 0.1353$ #

10. For (A): a (1,1,-1)
b (-1,1,-1)
C (-1,-1,-1)



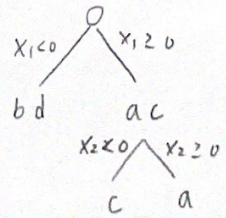
According to the question, "If $X_1 < 0$, the node connects to a leaf with some constant output"
So, we cannot distinguish b, c when their labels are different.
∴ cannot shattered.

(B) a (1,1,-1)
b (-1,1,-1)
c (1,-1,-1)



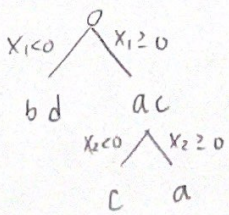
a, b, c can be shattered.

(C) a (1,1,-1)
b (-1,1,-1)
C (1,-1,-1)
d (-1,-1,-1)



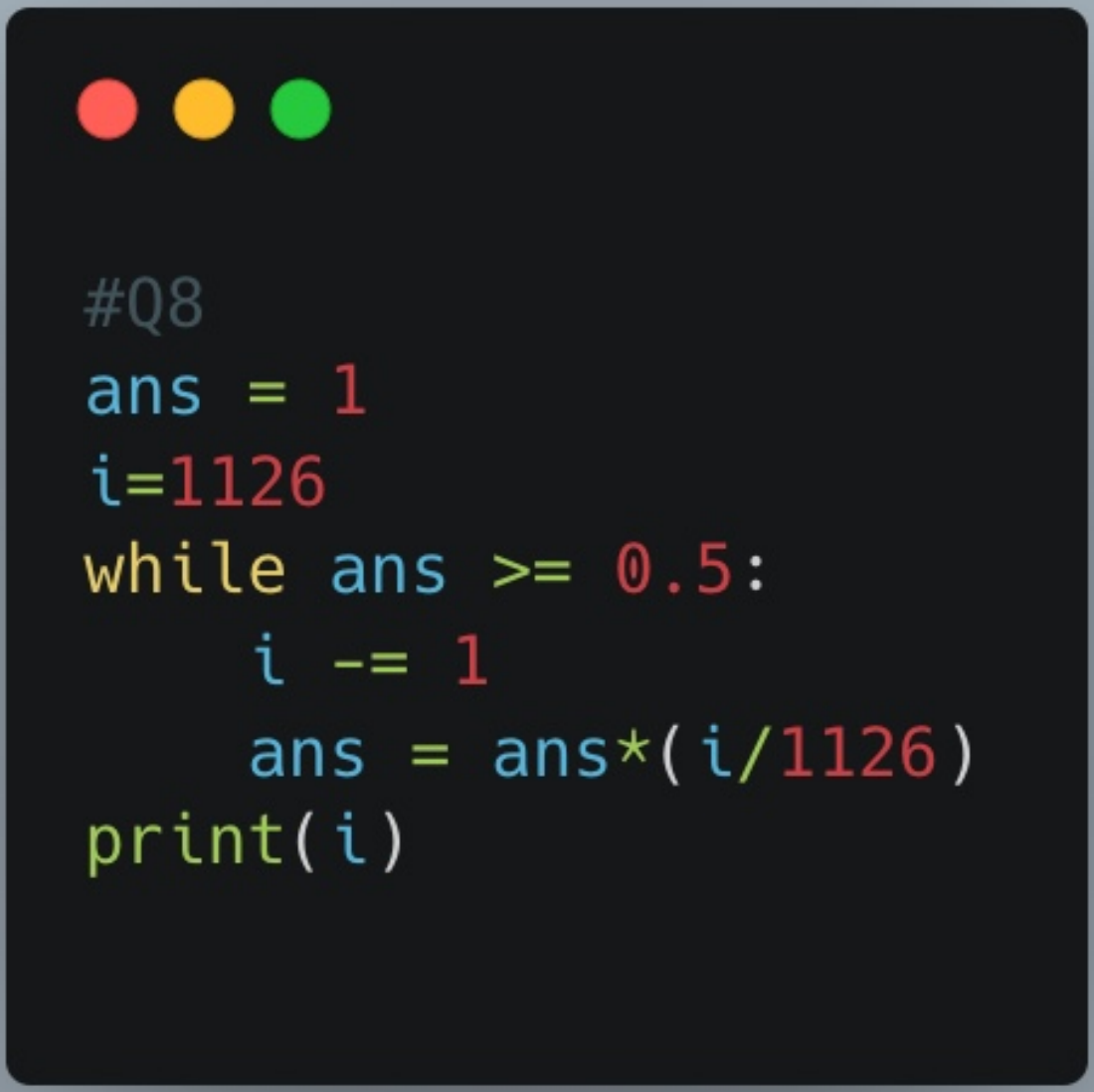
We cannot distinguish b, d.
(The reason is the same as Option(A))
∴ cannot shattered.

(D) a (1,1,-1)
b (-1,1,-1)
C (1,-1,-1)
d (-1,-1,1)



We cannot distinguish b, d
(The reason is the same as option (A))
∴ cannot shattered.

Choose B #


#Q8
ans = 1
i=1126
while ans >= 0.5:
i -= 1
ans = ans*(i/1126)
print(i)

```python
#Q11-Q16
x_train = []
y_train = []
x_test = []
y_test = []

f = open('hw6_train.dat.txt')
for line in f.readlines():
    x_train.append([float(i) for i in line.split(' ')[:10]])
    y_train.append(float(line.split(' ')[-1][:3]))
f.close

f = open('hw6_test.dat.txt')
for line in f.readlines():
    x_test.append([float(i) for i in line.split(' ')[:10]])
    y_test.append(float(line.split(' ')[-1][:3]))
f.close

import numpy as np
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_test)
y_test = np.array(y_test)
```

```python
#Q11
N = x_train.shape[0]
dim = x_train.shape[1]
u = np.array([(1.0 / N)] * N)
Ein = 1

for i in range(dim):
    Ein_i = 1
    threshold_i = 0
    feature_i = x_train[:,i]
    data_i = np.transpose(np.array([feature_i, y_train]))
    sorted_data_i = data_i[np.argsort(data_i[:, 0])]

    thresholds = np.array([float("-inf")] + [(sorted_data_i[j,0] + sorted_data_i[j+1,0]) / 2 for j in
range(N-1)])

    for threshold in thresholds:
        y_pos = np.where(sorted_data_i[:,0] > threshold, 1, -1)
        y_neg = np.where(sorted_data_i[:,0] < threshold, 1, -1)
        Epos = sum((y_pos != sorted_data_i[:,1]) * u)
        Eneg = sum((y_neg != sorted_data_i[:,1]) * u)
        if Epos > Eneg and Ein_i > Eneg:
            Ein_i = Eneg
        elif Epos < Eneg and Ein_i > Epos:
            Ein_i = Epos

    if Ein > Ein_i:
        Ein = Ein_i

print(Ein)
```

```python
#Q12 13 14 15 16
import math
def update_u(u,s,thres,feature_idx,y):
    if s == -1:
        y_pred = np.where(x_train[:,feature_idx] < thres, 1, -1)
    else:
        y_pred = np.where(x_train[:,feature_idx] > thres, 1, -1)
    incorrect = np.where(y_pred != y, 1, 0)

    e_t = sum(incorrect * u) / sum(u)
    diamond_t = math.sqrt((1-e_t)/e_t)

    new_u = np.where(y_pred == y, 1/diamond_t, diamond_t) * u

    alpha = np.log(diamond_t)

    return new_u,alpha
```

```python
#Q12
N = x_train.shape[0]
dim = x_train.shape[1]
u = np.array([(1.0 / N)] * N)
s = 1
thres = 0
max_Ein = 0
feature_idx = 0

for t in range(500):
    Ein = 1

    for i in range(dim):
        Ein_i = 1
        threshold_i = 0
        data_i = np.transpose(np.array([x_train[:,i], y_train]))
        sorted_data_i = data_i[np.argsort(data_i[:, 0])]
        sorted_u_i = u[np.argsort(data_i[:, 0])]

        thresholds = np.array([float("-inf")] + [(sorted_data_i[j,0] + sorted_data_i[j+1,0]) / 2 for j
in range(N-1)])

        for threshold in thresholds:
            y_pos = np.where(sorted_data_i[:,0] > threshold, 1, -1)
            y_neg = np.where(sorted_data_i[:,0] < threshold, 1, -1)
            Epos = sum((y_pos != sorted_data_i[:,1]) * sorted_u_i)
            Eneg = sum((y_neg != sorted_data_i[:,1]) * sorted_u_i)
            if Epos > Eneg:
                if Ein_i > Eneg:
                    Ein_i = Eneg
                    s_i = -1
                    threshold_i = threshold
            else:
                if Ein_i > Epos:
                    Ein_i = Epos
                    s_i = 1
                    threshold_i = threshold

        if Ein > Ein_i:
            Ein = Ein_i
            s = s_i
            thres = threshold_i
            feature_idx = i

    u, alpha = update_u(u,s,thres,feature_idx, y_train)

    if s == 1:
        pred_y = np.where(x_train[:,feature_idx] > thres, 1, -1)
    else:
        pred_y = np.where(x_train[:,feature_idx] < thres, 1, -1)

    Ein_01 = np.sum(pred_y != y_train)/N
    if Ein_01 > max_Ein:
        max_Ein = Ein_01

print(max_Ein)
```

```python
#Q13
def Ein_Gt(classifiers):
    T = len(classifiers)
    x_train_label = np.array([0.0] * x_train.shape[0])
    for classifier in classifiers:
        if classifier[1] == 1:
            pred_y = np.where(x_train[:,classifier[3]] > classifier[2], 1, -1)
        else:
            pred_y = np.where(x_train[:,classifier[3]] < classifier[2], 1, -1)

        x_train_label += classifier[0]*pred_y
    x_train_label = np.where(x_train_label > 0, 1, -1)
    Ein_G = sum(x_train_label != y_train)/ x_train.shape[0]
    return Ein_G

times = [60, 160, 260, 360, 460]
classifiers = []
N = x_train.shape[0]
dim = x_train.shape[1]
u = np.array([(1.0 / N)] * N)
s = 1
thres = 0
max_Ein = 0
feature_idx = 0

for t in range(460):
    Ein = 1

    for i in range(dim):
        Ein_i = 1
        threshold_i = 0
        data_i = np.transpose(np.array([x_train[:,i], y_train]))
        sorted_data_i = data_i[np.argsort(data_i[:, 0])]
        sorted_u_i = u[np.argsort(data_i[:, 0])]

        thresholds = np.array([float("-inf")] + [(sorted_data_i[j,0] + sorted_data_i[j+1,0]) / 2 for j
in range(N-1)])

        for threshold in thresholds:
            y_pos = np.where(sorted_data_i[:,0] > threshold, 1, -1)
            y_neg = np.where(sorted_data_i[:,0] < threshold, 1, -1)
            Epos = sum((y_pos != sorted_data_i[:,1]) * sorted_u_i)
            Eneg = sum((y_neg != sorted_data_i[:,1]) * sorted_u_i)
            if Epos > Eneg:
                if Ein_i > Eneg:
                    Ein_i = Eneg
                    s_i = -1
                    threshold_i = threshold
            else:
                if Ein_i > Epos:
                    Ein_i = Epos
                    s_i = 1
                    threshold_i = threshold

        if Ein > Ein_i:
            Ein = Ein_i
            s = s_i
            thres = threshold_i
            feature_idx = i

    u, alpha = update_u(u,s,thres,feature_idx, y_train)
    classifiers.append([alpha, s, thres, feature_idx])


    Ein_G = Ein_Gt(classifiers)
    if Ein_G <= 0.05:
        print(t)
        break
```

```python
#Q14
N = x_train.shape[0]
dim = x_train.shape[1]
u = np.array([(1.0 / N)] * N)
Ein = 1
s = 1
thres = 0
feature_idx = 0

for i in range(dim):
    Ein_i = 1
    s_i = 1
    threshold_i = 0
    feature_i = x_train[:,i]
    data_i = np.transpose(np.array([feature_i, y_train]))
    sorted_data_i = data_i[np.argsort(data_i[:, 0])]
    sorted_u_i = u[np.argsort(data_i[:, 0])]

    thresholds = np.array([float("-inf")] + [(sorted_data_i[j,0] + sorted_data_i[j+1,0]) / 2 for j in
range(N-1)])

    for threshold in thresholds:
        y_pos = np.where(sorted_data_i[:,0] > threshold, 1, -1)
        y_neg = np.where(sorted_data_i[:,0] < threshold, 1, -1)
        Epos = sum((y_pos != sorted_data_i[:,1]) * u)
        Eneg = sum((y_neg != sorted_data_i[:,1]) * u)
        if Epos > Eneg:
            if Ein_i > Eneg:
                Ein_i = Eneg
                s_i = -1
                threshold_i = threshold
        else:
            if Ein_i > Epos:
                Ein_i = Epos
                s_i = 1
                threshold_i = threshold

    if Ein > Ein_i:
        Ein = Ein_i
        s = s_i
        thres = threshold_i
        feature_idx = i

feature_i = x_test[:,feature_idx]
data_i = np.transpose(np.array([feature_i, y_test]))

if s == -1:
    y_pred = np.where(data_i[:,0] < thres, 1, -1)
else:
    y_pred = np.where(data_i[:,0] > thres, 1, -1)
Eout = sum(y_pred != data_i[:,1]) / x_test.shape[0]

print(Eout)
```

```python
#Q15
def Eout_G_uniform(classifiers):
    x_test_label = np.array([0.0] * x_test.shape[0])
    for classifier in classifiers:
        if classifier[1] == 1:
            pred_y = np.where(x_test[:,classifier[3]] > classifier[2], 1, -1)
        else:
            pred_y = np.where(x_test[:,classifier[3]] < classifier[2], 1, -1)

        x_test_label += pred_y
    x_test_label = np.where(x_test_label > 0, 1, -1)
    Eout_G = sum(x_test_label != y_test)/ x_test.shape[0]
    return Eout_G

#Q16
def Eout_G(classifiers):
    x_test_label = np.array([0.0] * x_test.shape[0])
    for classifier in classifiers:
        if classifier[1] == 1:
            pred_y = np.where(x_test[:,classifier[3]] > classifier[2], 1, -1)
        else:
            pred_y = np.where(x_test[:,classifier[3]] < classifier[2], 1, -1)

        x_test_label += classifier[0]*pred_y
    x_test_label = np.where(x_test_label > 0, 1, -1)
    Eout_G = sum(x_test_label != y_test)/ x_test.shape[0]
    return Eout_G
```

```python
#Q15 16
classifiers = []
N = x_train.shape[0]
dim = x_train.shape[1]
u = np.array([(1.0 / N)] * N)
s = 1
thres = 0
max_Ein = 0
feature_idx = 0

for t in range(500):
    Ein = 1
    for i in range(dim):
        Ein_i = 1
        threshold_i = 0
        data_i = np.transpose(np.array([x_train[:,i], y_train]))
        sorted_data_i = data_i[np.argsort(data_i[:, 0])]
        sorted_u_i = u[np.argsort(data_i[:, 0])]

        thresholds = np.array([float("-inf")] + [(sorted_data_i[j,0] + sorted_data_i[j+1,0]) / 2 for j
in range(N-1)])

        for threshold in thresholds:
            y_pos = np.where(sorted_data_i[:,0] > threshold, 1, -1)
            y_neg = np.where(sorted_data_i[:,0] < threshold, 1, -1)
            Epos = sum((y_pos != sorted_data_i[:,1]) * sorted_u_i)
            Eneg = sum((y_neg != sorted_data_i[:,1]) * sorted_u_i)
            if Epos > Eneg:
                if Ein_i > Eneg:
                    Ein_i = Eneg
                    s_i = -1
                    threshold_i = threshold
            else:
                if Ein_i > Epos:
                    Ein_i = Epos
                    s_i = 1
                    threshold_i = threshold

        if Ein > Ein_i:
            Ein = Ein_i
            s = s_i
            thres = threshold_i
            feature_idx = i

    u, alpha = update_u(u,s,thres,feature_idx, y_train)
    classifiers.append([alpha, s, thres, feature_idx])

Eout_G_u = Eout_G_uniform(classifiers)
Eout_G = Eout_G(classifiers)
print('Q15: ', Eout_G_u)
print('Q16: ', Eout_G)
```