ML_HW2 R09946023 吳偉樂

1） **(c)** **{ (1,1,3), (7,8,9), (15,16,17), (21,23,25) }**

We can shatter X if

$$X = \begin{bmatrix} - X_1^T - \\ - X_2^T - \\ - X_3^T - \\ - X_4^T - \end{bmatrix} \text{ is invertible such that } Sign(Xw) = y$$

$$X = \begin{bmatrix} - X_1^T - \\ - X_2^T - \\ - X_3^T - \\ - X_4^T - \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 7 & 8 & 9 \\ 1 & 15 & 16 & 17 \\ 1 & 21 & 23 & 25 \end{bmatrix} \;\text{->}\; \det\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 7 & 8 & 9 \\ 1 & 15 & 16 & 17 \\ 1 & 21 & 23 & 25 \end{bmatrix} = 0$$

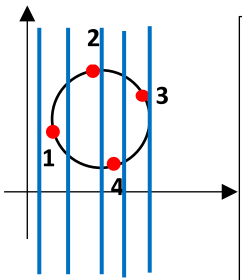
Note : (a),(e) : Linear dependent ; (b),(d) : det [*] = 0

2） **(d) 4N-2**

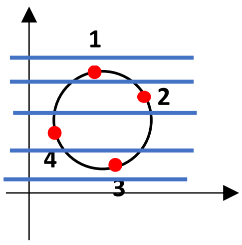
Consider 4 points, arrange it on a circle (no collinear),

Separate it with only horizontal OR vertical line that are parallel to x-axis or y-axis.

All possible points (1,2,3,4) are: (Since it is symmetric, so I just draw half)



Pick each line:
(o,o,o,o)
(o,x,x,x)
(o,o,x,x)
(o,o,x,o)
(o,o,o,o)



Pick each line
(o,o,o,o)
(x,o,x,x)
(x,o,o,x)
(x,x,x,o)
(o,o,o,o)

$2(\binom{N+1}{1}$-1)= 2N (-1 for repeated point) $2(\binom{N+1}{1}$-1) = 2N (-1 for repeated point)

Total : 4N

But we have (o,o,o,o) and (o,o,o,o) in vertical and horizontal case, so total **4N-2**

Similarly for n points

3) **(d) 3**

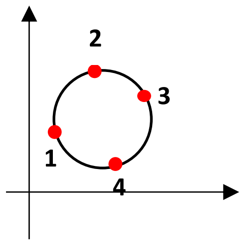
$w_0 > 0$ implies that the line DOES NOT pass through the origin.

We use the graph in (2), we can't find a line such that (o,x,o,x)

We still have infinite choices of lines even we do not restrict w_0

So the VC dimension is 3 (same as 2D-perceptrons).

Actually $d_{vc} = d + 1 = 3$



4) **(b)** $\binom{N+1}{2}+1$

In 3-D dimensional space, we use a ring as our hypothesis sets.

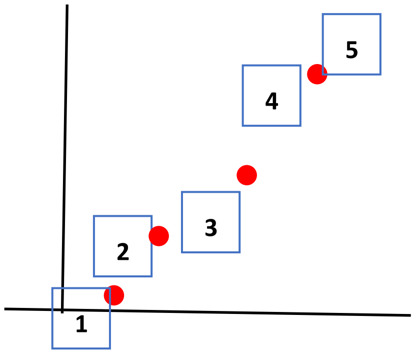
Actually it is a hollow sphere which origin at (0,0,0), we project it into 2D space (x and y axis), and fix Z-axis in same high for simplicity.

Consider N = 4 . We have Interval 1,2,3,4,5

We can choose any 2 intervals ,which are

a= smaller coordinate , b = larger coordinate,

then we have $\binom{N+1}{2}$ combination , and we still have one more combination which a and b in same interval, means that all the points are -1



So the answer should be $\binom{N+1}{2}+1$

5) **(b) 2**

Use the growth function in (4) ,

When N = 2 , $\binom{N+1}{2}+1 = 4$

When N = 3 , $\binom{N+1}{2}+1 = 7 < 2^3$

So VC dimension = 2 (break point = 3 ,and vc dimension = min(break point) -1)

6) (d)

6. (d) $2\sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$

$$E_{out}(g) - E_{out}(g_*) = E_{out}(g) - E_{in}(g) + \overbrace{E_{in}(g) - E_{in}(g_*)}^{\le 0 \text{ since } g=\arg\min_h E_{in}(h)} + E_{in}(g_*) - E_{out}(g_*)$$

$$\le E_{out}(g) - E_{in}(g) + E_{in}(g_*) - E_{out}(g_*)$$

$$\le \varepsilon + \varepsilon = 2\varepsilon$$

∵ ① the VC bound holds for any $g \in H$

② $|E_{out}(g) - E_{in}(g)| \le \varepsilon \Rightarrow \varepsilon$ is upper bound

Since $P_D[\,|E_{in}(g) - E_{out}(g)| > \varepsilon\,] \le 4m_H(2N)e^{-\frac{1}{8}\varepsilon^2 N}$ for any $g = A(D) \in H$

Set $\delta = 4m_H(2N)e^{-\frac{1}{8}\varepsilon^2 N} \Rightarrow -\frac{1}{8}\varepsilon^2 N = \ln\frac{\delta}{4m_H(2N)}$

$\Rightarrow \varepsilon^2 = \frac{8}{N}\ln\frac{4m_H(2N)}{\delta} \Rightarrow \varepsilon = \sqrt{\frac{8}{N}\ln\frac{4m_H(2N)}{\delta}}$

∴ $E_{out}(g) - E_{out}(g_*) \le 2\varepsilon = 2\sqrt{\frac{8}{N}\ln\frac{4m_H(2N)}{\delta}}$

7) (d) $\lfloor \log_2 M \rfloor$

We know that $H = \{\, hypothesis\ h: X \to \{o,x\}^N \,\}$ and $H = \{h_1, \dots, h_M\}$

$h(x_1, x_2, \dots, x_N) = (h(x_1), h(x_2), \dots, h(x_N)) \in \{o,x\}^N = 2^N$

不管有多少個 h_i (即把 N input 分成 o 或 x 不同或相同組合的線(2d)、超平面(3d)等等)，我的最終組合都是 $\{o,x\}^N$

If $2^N > M$, means that we cannot shatter X=$(x_1, x_2, \dots, x_N)$

(Because we have at most M distinct combination from hypothesis line , but we have 2^Ncombination at here, so there exists some combination that we can’t shatter)

If $2^N \le M$, means the hypothesis lines is more than or equal to the combinations of N inputs. $d_{vc}(H) = \{\, largest\ N\ for\ which\ m_H(N) = 2^N \,\} = N$

So, $2^N \le M \implies N\log 2 \le \log M \implies N \le \frac{\log 2}{\log M} = \log_2 M$

Since d_{vc}=N is an integer , so the nearest answer is $\lfloor \log_2 M \rfloor$

8) (d)

8. (d) k+1

$h: \underbrace{[-1,+1]^k}_{2^k \text{ combinations}} \to \underbrace{[-1,1]}_{2C_1}$

($h: x \to y$)

$\Rightarrow 2^k \times 2 = 2^{k+1}$

eg. $h(x)$ y

combination

① — — — — ... — → 0 or 1 ⇒ ① × 2

② — — — — ... — → 0 or 1 ② × 2

⋮

(2^k) — — — — ... — → 0 or 1 (2^k) × 2

9) (c) 3

Since $d_{vc}(H) = d$, it means there exists d input such that $m_H(d) = 2^d$.
$m_H(d)$ is growth function that take max of all possible $(x_1, x_2, \dots, x_N)$
So it means that there are some set of d distinct inputs is shattered by H.
d+1 is break point, means that any set of d+1 inputs is not shattered by H ,which contains some set of d+1 inputs is not shattered by H

10) (c) sine family

12.
$\{h_\alpha : h_\alpha(x) = \text{sign}(\sin(\alpha \cdot x))\}$ for $x, \alpha \in \mathbb{R}$
Assume we have n inputs $\{x_1, \dots, x_n\}$
then we have the output $(y_1, \dots, y_n) \in \{-1, +1\}^n$
Let $\{x_1 = \frac{1}{2}, x_2 = \frac{1}{2^2}, \dots, x_n = \frac{1}{2^n}\}$, then

- if $y=1$, means that $\sin \alpha x > 0 \Leftrightarrow 0 < \alpha x < \pi, 2\pi < \alpha x < 4\pi, \dots$
  since $x \neq 0$, we have $0 < x < \frac{\pi}{\alpha}, \frac{2\pi}{\alpha} < x < \frac{4\pi}{\alpha}, \dots$ $(\alpha \neq 0)$
- if $y=-1$, means that $\sin \alpha x < 0 \Leftrightarrow \pi < \alpha x < 2\pi, 3\pi < \alpha x < 4\pi, \dots$
  since $x \neq 0$, we have $\frac{\pi}{\alpha} < x < \frac{2\pi}{\alpha}, \frac{3\pi}{\alpha} < x < \frac{4\pi}{\alpha}, \dots$ $(\alpha \neq 0)$

One point: if $y=1$, $x = \frac{1}{2}$, then we take $\alpha = \pi$
if $y=-1$, $x = \frac{1}{2}$, then we take $\alpha = 3\pi$

two points: $\frac{1}{2}, \frac{1}{2^2}$

$\frac{1}{2}$	$\frac{1}{2^2}$	
1	-1	: take $\alpha = 5\pi$
-1	1	: take $\alpha = 3\pi$
1	1	: take $\alpha = \pi$ ← $\alpha = \pi$ if all outputs of x_i are 1
-1	-1	: take $\alpha = 7\pi$

observation: $k\pi$, where k is odd
$\alpha = \pi(1 + ?)$

Actually, $\alpha = \pi \times$ some constant that dependone to the result y_i, and input x_i
$\Rightarrow$ Set $\alpha = \pi\left(1 + \sum_{i=1}^{n} 2^{i-1}(1 - y_i)\right)$, then we get desired result !!

11) (d)

11. (d) $E_{out}(h, 0) = \dfrac{E_{out}(h, \tau) - \tau}{1 - 2\tau}$

c : correct , w : wrong

$$E_{out}(h, \tau) = \frac{c\tau + w(1-\tau)}{c+w} = \frac{w + \tau(c-w)}{c+w} = \frac{w}{c+w} + \frac{c-w}{c+w}\tau = E_{out}(h,0) + \frac{c+w-w-w}{c+w}\tau$$

$\frac{c}{c+w} = E_{out}(h,0)$

$$= E_{out}(h,0) + \tau - 2E_{out}(h,0)\tau$$

$\Rightarrow E_{out}(h,\tau) - \tau = E_{out}(h,0)[1 - 2\tau]$

$\Rightarrow E_{out}(h,0) = \dfrac{E_{out}(h,\tau) - \tau}{1 - 2\tau}$ ※ .

12) **(b) 0.6**

$E_{out}(f(x)) = E\, err(f, y)$

$$P(y|x) = \begin{cases} 0.7 & y = f(x) \\ 0.1, & y = f(x) mod\ 3 + 1 \\ 0.2 & y = (f(x) + 1) mod\ 3\ + 1 \end{cases}$$

So,

$$f(x) = \begin{cases} 1 & err = 0.1 + 4(0.2) = 0.9 \\ 2, & err = 0.1 + 0.2 = 0.3 \\ 3 & err = 4(0.1) + 0.2 = 0.6 \end{cases} ,\ \rightarrow$$

	f(x)	
1	2	3
1	2	3
2	3	1
3	1	2

$err = P(y|x) * (f(x) - y)^2$

$E_{out}(f(x)) = (0.9 + 0.3 + 0.6)/3 = 0.6$

13) **(b)**

13) (b) 0.14

$\Delta(f, f_*) = E_{x\sim P(x)}(f(x) - f_*(x))^2$, $f_*(x) = \sum_{y=1}^{3} y \cdot P(y|x)$

when $f(x)=1$, $f_*(x) = 1(0.7) + 2(0.1) + 3(0.2) = 1.5$.

when $f(x)=2$, $f_*(x) = 2(0.7) + 3(0.1) + 1(0.2) = 1.9$

when $f(x)=3$, $f_*(x) = 3(0.7) + 0.1 + 2(0.2) = 2.6$

$\Rightarrow \Delta(f, f_*) = \frac{1}{3}[(1-1.5)^2 + (2-1.9)^2 + (3-2.6)^2] = 0.14$.

14) **(d) 12000**

VC-bound = δ $\Longrightarrow$ $4m_H(2N)e^{-\frac{1}{8}\varepsilon^2 N} \le \delta \Longrightarrow 4(4N)e^{-\frac{1}{8}\varepsilon^2 N} \le \delta$

$16Ne^{-\frac{1}{8}0.1^2 N} \le 0.1$

For (a)~(c) , $16Ne^{-\frac{1}{8}0.1^2 N} \nleq 0.1$ .For (d) , $16(12000)e^{-\frac{1}{8}0.1^2(12000)}$= 0.05873 < 0.1

For (e) , $16(14000)e^{-\frac{1}{8}0.1^2(14000)}$= 0.00563 < 0.1

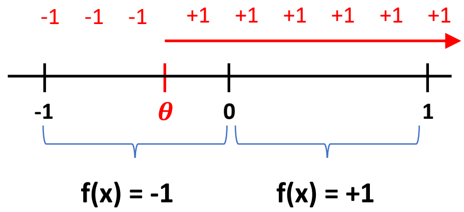
Pick smallest N , that is , N = 12000

15) (b) $\frac{1}{2}|\ \theta\ |$

$h_{+1,\theta}$ means positive ray

Since $f(x) = sign(x)$, $h_{+1,\theta}(x) = sign(x - \theta)$

We can use positive ray graph to show it easily :



$h_{+1,\theta}(x) = f(x)\ when\ x \leq \theta\ and\ x \geq 0$

$h_{+1,\theta}(x) \neq f(x)\ when\ \theta \leq\ x \leq 0$

So $E_{out}(h_{+1,\theta}, 0) = \frac{1}{2}|\ \theta\ |$, since the whole interval length is 2

16) (d) 0.3

17) (b) 0.02

18) (e) 0.4

19) (c) 0.05

20) (a) 0.00

In [2]:
```python
import numpy as np
import random
```

In [317]:
```
# 16  (d) 0.3

def h(theta,s,x):  # hypothesis function
    h = x-theta
    h=1 if h>0 else -1
    return s*h

DATA_SET = list(np.random.uniform(-1,1,100000))
E_Collect = []
iteration = 0

F = []
for i in DATA_SET :   # F(x) = sign(x)
    F.append(1) if i>0 else F.append(-1)



while iteration != 10000:
    X = sorted(list(np.random.uniform(-1,1,2)))

    f = []
    for i in X:   # f(x) = sign(x)
        f.append(1) if i>0 else f.append(-1)

    Theta = [-1]
    for i in range(len(X)-1):
        Theta.append((X[i]+X[i+1])/2)
    Theta.append(1)     # ALL THETA ELEMENTS


    Ein, S_in, Theta_in = [], [], []
    S = [1,-1]

    for i in Theta:
        if i == -1 or i == 1:
            Esum = 0
            for j in range(len(X)):
                if h(i,1,X[j]) != f[j]:
                    Esum +=1

            Ein.append(Esum/len(X))
            S_in.append(1)
            Theta_in.append(i)

        else:
            for k in S:
                Esum = 0
                for j in range(len(X)):
                    if h(i,k,X[j]) != f[j]:
                        Esum += 1

                Ein.append(Esum/len(X))
                S_in.append(k)
                Theta_in.append(i)

    FINAL = []
```

```
    b = 99999

    M = min(Ein)
    for i in range(len(Ein)):
        if Ein[i] == M :
            FINAL.append([Ein[i],S_in[i],Theta_in[i]])
    if len(FINAL) >1 :
        for i in range(len(FINAL)):
            if FINAL[i][1] + FINAL[i][2] < b :
                b = FINAL[i][1] + FINAL[i][2]
                Einbest = FINAL[i]
    else:
        Einbest = FINAL[0]

    Eout_Sum = 0
    for i in range(len(DATA_SET)):
        if h(Einbest[2], Einbest[1], DATA_SET[i]) != F[i]:
            Eout_Sum += 1
    Eout_Sum = Eout_Sum/len(DATA_SET)

    E_Collect.append(Eout_Sum - Einbest[0])


    iteration += 1

np.mean(E_Collect)

# almost 17 mins used......
```

Out[317]: 0.291485631

In [318]:
```
# 17  (b) 0.02

def h(theta,s,x):  # hypothesis function
    h = x-theta
    h=1 if h>0 else -1
    return s*h

DATA_SET = list(np.random.uniform(-1,1,100000))
E_Collect = []
iteration = 0

F = []
for i in DATA_SET :   # F(x) = sign(x)
    F.append(1) if i>0 else F.append(-1)



while iteration != 10000:
    X = sorted(list(np.random.uniform(-1,1,20)))

    f = []
    for i in X:   # f(x) = sign(x)
        f.append(1) if i>0 else f.append(-1)

    Theta = [-1]
    for i in range(len(X)-1):
        Theta.append((X[i]+X[i+1])/2)
    Theta.append(1)     # ALL THETA ELEMENTS


    Ein, S_in, Theta_in = [], [], []
    S = [1,-1]

    for i in Theta:
        if i == -1 or i == 1:
            Esum = 0
            for j in range(len(X)):
                if h(i,1,X[j]) != f[j]:
                    Esum +=1

            Ein.append(Esum/len(X))
            S_in.append(1)
            Theta_in.append(i)

        else:
            for k in S:
                Esum = 0
                for j in range(len(X)):
                    if h(i,k,X[j]) != f[j]:
                        Esum += 1

                Ein.append(Esum/len(X))
                S_in.append(k)
                Theta_in.append(i)

    FINAL = []
```

```
    b = 99999

    M = min(Ein)
    for i in range(len(Ein)):
        if Ein[i] == M :
            FINAL.append([Ein[i],S_in[i],Theta_in[i]])
    if len(FINAL) >1 :
        for i in range(len(FINAL)):
            if FINAL[i][1] + FINAL[i][2] < b :
                b = FINAL[i][1] + FINAL[i][2]
                Einbest = FINAL[i]
    else:
        Einbest = FINAL[0]

    Eout_Sum = 0
    for i in range(len(DATA_SET)):
        if h(Einbest[2], Einbest[1], DATA_SET[i]) != F[i]:
            Eout_Sum += 1
    Eout_Sum = Eout_Sum/len(DATA_SET)

    E_Collect.append(Eout_Sum - Einbest[0])


    iteration += 1

print('The answer of Question 17 is approximately ',np.mean(E_Collect))
# Almost 20 mins used....
```

```
The answer of Question 17 is approximately  0.024210523
```

In [331]:
```
# 18  (e) 0.4

def h(theta,s,x):  # hypothesis function
    h = x-theta
    h=1 if h>0 else -1
    return s*h

DATA_SET = list(np.random.uniform(-1,1,100000))
E_Collect = []
iteration = 0

F = []
for i in DATA_SET :   # F(x) = sign(x)
    F.append(1) if i>0 else F.append(-1)

P = ['H','H','H','H','H','H','H','H','H','T']
for i in range(len(F)):
    TAU = random.choice(P)
    if TAU == 'T':  #  TAU = 0.1
        F[i] = -1 * F[i]

while iteration != 10000:
    X = sorted(list(np.random.uniform(-1,1,2)))

    f = []
    for i in X:   # f(x) = sign(x)
        f.append(1) if i>0 else f.append(-1)

    for i in range(len(f)):
        TAU = random.choice(P)
        if TAU == 'T':  #  TAU = 0.1
            f[i] = -1 * f[i]


    Theta = [-1]
    for i in range(len(X)-1):
        Theta.append((X[i]+X[i+1])/2)
    Theta.append(1)    # ALL THETA ELEMENTS


    Ein, S_in, Theta_in = [], [], []
    S = [1,-1]

    for i in Theta:
        if i == -1 or i == 1:
            Esum = 0
            for j in range(len(X)):
                if h(i,1,X[j]) != f[j]:
                    Esum +=1

            Ein.append(Esum/len(X))
            S_in.append(1)
            Theta_in.append(i)

        else:
            for k in S:
```

```
                Esum = 0
                for j in range(len(X)):
                    if h(i,k,X[j]) != f[j]:
                        Esum += 1

                Ein.append(Esum/len(X))
                S_in.append(k)
                Theta_in.append(i)

    FINAL = []
    b = 99999

    M = min(Ein)
    for i in range(len(Ein)):
        if Ein[i] == M :
            FINAL.append([Ein[i],S_in[i],Theta_in[i]])
    if len(FINAL) >1 :
        for i in range(len(FINAL)):
            if FINAL[i][1] + FINAL[i][2] < b :
                b = FINAL[i][1] + FINAL[i][2]
                Einbest = FINAL[i]
    else:
        Einbest = FINAL[0]

    Eout_Sum = 0
    for i in range(len(DATA_SET)):
        if h(Einbest[2], Einbest[1], DATA_SET[i]) != F[i]:
            Eout_Sum += 1
    Eout_Sum = Eout_Sum/len(DATA_SET)

    E_Collect.append(Eout_Sum - Einbest[0])


    iteration += 1

print('The answer of Question 18 is approximately ',np.mean(E_Collect))
```

```
The answer of Question 18 is approximately  0.36793075899999994
```

In [332]:

```
# 19  (c) 0.05

def h(theta,s,x):  # hypothesis function
    h = x-theta
    h=1 if h>0 else -1
    return s*h

DATA_SET = list(np.random.uniform(-1,1,100000))
E_Collect = []
iteration = 0

F = []
for i in DATA_SET :   # F(x) = sign(x)
    F.append(1) if i>0 else F.append(-1)

P = ['H','H','H','H','H','H','H','H','H','T']
for i in range(len(F)):
    TAU = random.choice(P)
    if TAU == 'T':  #  TAU = 0.1
        F[i] = -1 * F[i]

while iteration != 10000:
    X = sorted(list(np.random.uniform(-1,1,20)))

    f = []
    for i in X:   # f(x) = sign(x)
        f.append(1) if i>0 else f.append(-1)

    for i in range(len(f)):
        TAU = random.choice(P)
        if TAU == 'T':  #  TAU = 0.1
            f[i] = -1 * f[i]


    Theta = [-1]
    for i in range(len(X)-1):
        Theta.append((X[i]+X[i+1])/2)
    Theta.append(1)     # ALL THETA ELEMENTS


    Ein, S_in, Theta_in = [], [], []
    S = [1,-1]

    for i in Theta:
        if i == -1 or i == 1:
            Esum = 0
            for j in range(len(X)):
                if h(i,1,X[j]) != f[j]:
                    Esum +=1

            Ein.append(Esum/len(X))
            S_in.append(1)
            Theta_in.append(i)

        else:
            for k in S:
```

```
                Esum = 0
                for j in range(len(X)):
                    if h(i,k,X[j]) != f[j]:
                        Esum += 1

                Ein.append(Esum/len(X))
                S_in.append(k)
                Theta_in.append(i)

    FINAL = []
    b = 99999

    M = min(Ein)
    for i in range(len(Ein)):
        if Ein[i] == M :
            FINAL.append([Ein[i],S_in[i],Theta_in[i]])
    if len(FINAL) >1 :
        for i in range(len(FINAL)):
            if FINAL[i][1] + FINAL[i][2] < b :
                b = FINAL[i][1] + FINAL[i][2]
                Einbest = FINAL[i]
    else:
        Einbest = FINAL[0]

    Eout_Sum = 0
    for i in range(len(DATA_SET)):
        if h(Einbest[2], Einbest[1], DATA_SET[i]) != F[i]:
            Eout_Sum += 1
    Eout_Sum = Eout_Sum/len(DATA_SET)

    E_Collect.append(Eout_Sum - Einbest[0])


    iteration += 1

print('The answer of Question 19 is approximately ',np.mean(E_Collect))
```

```
The answer of Question 19 is approximately  0.051816922999999994
```

In [333]:

```
# 20 (a) 0.00

def h(theta,s,x):  # hypothesis function
    h = x-theta
    h=1 if h>0 else -1
    return s*h

DATA_SET = list(np.random.uniform(-1,1,100000))
E_Collect = []
iteration = 0

F = []
for i in DATA_SET :   # F(x) = sign(x)
    F.append(1) if i>0 else F.append(-1)

P = ['H','H','H','H','H','H','H','H','H','T']
for i in range(len(F)):
    TAU = random.choice(P)
    if TAU == 'T':  #  TAU = 0.1
        F[i] = -1 * F[i]

while iteration != 10000:
    X = sorted(list(np.random.uniform(-1,1,200)))

    f = []
    for i in X:   # f(x) = sign(x)
        f.append(1) if i>0 else f.append(-1)

    for i in range(len(f)):
        TAU = random.choice(P)
        if TAU == 'T':  #  TAU = 0.1
            f[i] = -1 * f[i]


    Theta = [-1]
    for i in range(len(X)-1):
        Theta.append((X[i]+X[i+1])/2)
    Theta.append(1)     # ALL THETA ELEMENTS


    Ein, S_in, Theta_in = [], [], []
    S = [1,-1]

    for i in Theta:
        if i == -1 or i == 1:
            Esum = 0
            for j in range(len(X)):
                if h(i,1,X[j]) != f[j]:
                    Esum +=1

            Ein.append(Esum/len(X))
            S_in.append(1)
            Theta_in.append(i)

        else:
            for k in S:
```

```
                Esum = 0
                for j in range(len(X)):
                    if h(i,k,X[j]) != f[j]:
                        Esum += 1

                Ein.append(Esum/len(X))
                S_in.append(k)
                Theta_in.append(i)

    FINAL = []
    b = 99999

    M = min(Ein)
    for i in range(len(Ein)):
        if Ein[i] == M :
            FINAL.append([Ein[i],S_in[i],Theta_in[i]])
    if len(FINAL) >1 :
        for i in range(len(FINAL)):
            if FINAL[i][1] + FINAL[i][2] < b :
                b = FINAL[i][1] + FINAL[i][2]
                Einbest = FINAL[i]
    else:
        Einbest = FINAL[0]

    Eout_Sum = 0
    for i in range(len(DATA_SET)):
        if h(Einbest[2], Einbest[1], DATA_SET[i]) != F[i]:
            Eout_Sum += 1
    Eout_Sum = Eout_Sum/len(DATA_SET)

    E_Collect.append(Eout_Sum - Einbest[0])


    iteration += 1

print('The answer of Question 20 is approximately ',np.mean(E_Collect))
```

```
The answer of Question 20 is approximately  0.005134032999999998
```

In [ ]: