Answer: 1~5 cdcbb
6~10 dddcc
11~15 dbbdb
16~20 dbeca

ML foundation HW 2

Electrical Engineering
B06504016 林家宏

Solution:

1 Let's write $\{(1,1,1,3),(1,7,8,9),(1,15,16,17),(1,21,23,25)\}$ to the matrix form.

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 3 \\ 1 & 7 & 8 & 9 \\ 1 & 15 & 16 & 17 \\ 1 & 21 & 23 & 25 \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}}_{Y}$$

The 4 points can be shattered if $\Rightarrow$ column vectors of X span R^4.

The reduced row echelon form of X is $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, whose rank is 4. Hence, any $Y \in R^4$ can be spanned by column vectors of X.

$\Rightarrow$ The 4 points can be shattered by the 3D perceptron hypothesis set. $\Rightarrow$ Choose (c)

2

Horizontal line (Excluding all 1 or all −1 case): $(N-1)\times 2$
Vertical line (Excluding all 1 or all −1 case): $(N-1)\times 2$
All y are 1 : 1
All y are −1 : 1

$2(N-1)+2(N-1)+1+1 = 4N-2$ $\Rightarrow$ Choose (d)

3

$y = \text{sign}(w_0 + w_1x_1 + w_2x_2)$, if $w_0 > 0$, it means the point $O(0,0)$ should be always classified to 1. Hence, it just like adding 1 point $O(0,0)$ to run perceptron and the point should be always assigned 1 as consequence.

In class, we are taught that 4 points cannot shattered by 2D perceptron hypothesis set. In this case, 3 normal points + 1 origin point (0,0) cannot be shattered, too. Therefore, the VC dimension $= 3-1 = 2$ $\Rightarrow$ Choose (c)

4 $h(x) = \begin{cases} +1 & \text{if } \sqrt{a} \le |x| \le \sqrt{b} \\ -1 & \text{otherwise} \end{cases}$

N points mean that we have n+1 location to chose two from it. It can determine the start and end of the ring. But there is still one case excluded, which is the all -1 case. Therefore, the growth function is $C_2^{n+1} + 1$. => Chose (b).

5 The growth function $m_h(n) = \frac{n(n+1)}{2} + 1$

$n=2 \Rightarrow m_h(2) = \frac{2\times3}{2} + 1 = 4 = 2^2$

$n=3 \Rightarrow m_h(3) = \frac{3\times4}{2} + 1 = 7 \le 2^3$

3 is the break point, so 2 is VC dimension.

6 From the slide, we obtain

$$E_{in}(g) - \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)} \le E_{out}(g) \le E_{in}(g) + \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$$

$$E_{in}(g^*) - \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)} \le E_{out}(g^*) \le E_{in}(g^*) + \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$$

$$\Rightarrow E_{out}(g) \le E_{in}(g) + \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)} \le E_{in}(g^*) + \sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)} \le E_{out}(g^*) + 2\sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$$

$$\Rightarrow E_{out}(g) \le E_{out}(g^*) + 2\sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$$

$$\Rightarrow E_{out}(g) - E_{out}(g^*) \le 2\sqrt{\frac{8}{N}\ln\left(\frac{4m_H(2N)}{\delta}\right)}$$ => choose (d)

$E_{in}(g) \le E_{in}(g^*)$ since g is the hypothesis minimizing Ein.

7 N points have 2^N combination. If the hypothesis set can shatter N points, $M \ge 2^N$.

$N = \lfloor \log_2 M \rfloor \Rightarrow d_{vc}(H) = \lfloor \log_2 M \rfloor$ because H at most shattered $N = \lfloor \log_2 M \rfloor$ points.

8 $\{-1,1\}^k$: The number of 1 range from 0 to k, k+1 possible combination in total. If the number of data exceeds k+1, the outcomes will always be the same since they have same number of 1 and -1 so that k+2 data cannot be shattered. Therefore, VC dimension is k+1.

9 ① Some set of d distinct inputs is shattered by H

=> It means that $d_{vc}(H) \geq d$ (Definition)

⑥ Some set of d+1 distinct inputs is not shattered by H.

=> If $d_{vc}(H) = d$, any set of d+1 distinct is not shattered by H. It's definition. Hence, some set of d+1 distinct inputs is not shattered by H.

⑧ Any set of d+1 distinct inputs is not shattered by H.

=> It means that $d_{vc}(H) < d+1$ (Definition)

3 conditions in total. => Choose (C).

10 Let's look at (C)

Let $\alpha = \pi\left(1+\sum_{i=1}^{m} 2^i \frac{1-y_i}{2}\right)$ over a set of points $(x_1, x_2 \cdots x_m)$ with arbitrary labels $(y_1, y_2, \cdots y_m) \in \{-1,1\}^n$

$x_j = 2^{-j}$

$$\alpha x_j = \pi\left(2^{-j} + \sum_{i=1}^{m} 2^{i-j} \frac{1-y_i}{2}\right) = \pi\left(2^{-j} + \sum_{i=1}^{j} 2^{i-j} \frac{1-y_i}{2} + \boxed{\sum_{i=j+1}^{m} 2^{i-j} \frac{1-y_i}{2}}\right)$$

drop since it only contributes multiples of 2π

$x_j = 2^{-j}$

$$\Rightarrow \alpha x_j = \pi\left(2^{-j} + \sum_{i=1}^{m} 2^{i-j} \frac{1-y_i}{2} + \frac{1-y_j}{2}\right) \leq \pi\left(\sum_{i=0}^{j-1} 2^{i-j} + \frac{1-y_j}{2}\right)$$

$$< \pi\left(1+\frac{1-y_j}{2}\right)$$

$\alpha x_j > \pi \frac{1-y_j}{2}$ $\Rightarrow \pi \frac{1-y_j}{2} < \alpha x_j < \pi\left(1+\frac{1-y_j}{2}\right)$

for $y_j = -1 \Rightarrow \pi < \alpha x_j < 2\pi$, $y_j = +1 \Rightarrow 0 < \alpha x_j < \pi$ => Choose (C)

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

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

12 $f(\vec{x}) = 1$

$(1-1)^2 \times 0.7 + (1-2)^2 \times 0.1 + (1-3)^2 \times 0.2 = 0.9$

$f(\vec{x}) = 2$

$(2-2)^2 \times 0.7 + (2-3)^2 \times 0.1 + (2-1)^2 \times 0.2 = 0.3$

$f(\vec{x}) = 3$

$(3-3)^2 \times 0.7 + (3-1)^2 \times 0.1 + (3-2)^2 \times 0.2 = 0.6$ $\quad E_{out}(f) = \frac{1}{3} \times 0.9 + \frac{1}{3} \times 0.3 + \frac{1}{3} \times 0.6 = 0.6$

13 $f(\vec{x}) = 1$

$f_*(\vec{x}) = 0.7 + 2 \times 0.1 + 3 \times 0.2 = 1.5$

$f(\vec{x}) = 2$

$f_*(\vec{x}) = 0.2 + 2 \times 0.7 + 3 \times 0.1 = 1.9$

$f(\vec{x}) = 3$ $\quad \Delta(f, f_*) = \frac{1}{3}(1-1.5)^2 + \frac{1}{3}(2-1.9)^2 + \frac{1}{3}(3-2.6)^2$

$f_*(\vec{x}) = 0.1 + 2 \times 0.2 + 3 \times 0.7 = 2.6$ $\quad = 0.14$

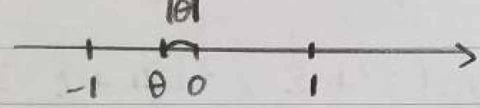
14 $P[BAD] \leq 4 m_H(2N) \exp(-\frac{1}{8}\varepsilon^2 N)$ $\quad m_H(2N) = 4N$ $\quad \varepsilon = 0.1$

$N = 11543$ $\quad P[BAD] = 0.10002 > 0.1 = \delta$

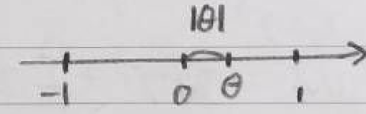
$N = 11544$ $\quad P[BAD] = 0.0999 < 0.1 = \delta$

Hence, 12000 is the smallest N which can let $P[BAD] < 0.1$

15



or



In both cases, $E_{out}(h_{s,\theta}, 0) = \frac{|\theta|}{2}$ since $\text{sign}(x-\theta) \neq \text{sign}\, x$ between 0 and θ.

16

```python
import math
import random
import numpy as np
def h(s, theta, x):
    if s * (x - theta) > 0:
        return 1
    else:
        return -1
def Ein(size, s, theta, data, data2):
    err = 0
    for i in range(size):
        if h(s, theta, data[i][0]) != data[i][1]:
            err = err +1
    return err/size
data_size = 20
tou = 0
Eoutin = []
for j in range(10000):
    data = []
    data1 = []
    data2 = [-1]
    for i in range(data_size):
        x = random.uniform(-1, 1)
        data.append((x, h(1, 0, x)))
        data1.append(x)
    data.sort(key = lambda s: s[0])
    data1.sort()
    for i in range(data_size-1):
        data2.append((data1[i]+data1[i+1])/2)
    min_Ein = 1
    cur_s = 1
    cur_theta = -1
    for i in data2:
        s = -1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
```

```
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
        s = 1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
    if cur_s == 1:
        E = abs(cur_theta)/2
    else:
        E = (2-abs(cur_theta))/2
    Eoutin.append(E - min_Ein)
average = 0
for i in Eoutin:
    average = average + i
average = average/10000
print('Eout-Ein = {}'.format(average))
```

17

```
import math
import random
import numpy as np
def h(s, theta, x):
    if s * (x - theta) > 0:
        return 1
    else:
        return -1
def Ein(size, s, theta, data, data2):
    err = 0
    for i in range(size):
        if h(s, theta, data[i][0]) != data[i][1]:
            err = err +1
    return err/size
data_size = 20
tou = 0
Eoutin = []
for j in range(10000):
    data = []
    data1 = []
```

```
    data2 = [-1]
    for i in range(data_size):
        x = random.uniform(-1, 1)
        data.append((x, h(1, 0, x)))
        data1.append(x)
    data.sort(key = lambda s: s[0])
    data1.sort()
    for i in range(data_size-1):
        data2.append((data1[i]+data1[i+1])/2)
    min_Ein = 1
    cur_s = 1
    cur_theta = -1
    for i in data2:
        s = -1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
        s = 1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
    if cur_s == 1:
        E = abs(cur_theta)/2
    else:
        E = (2-abs(cur_theta))/2
    Eoutin.append(E - min_Ein)
average = 0
for i in Eoutin:
    average = average + i
average = average/10000
print('Eout-Ein = {}'.format(average))
```

18

```
import math
import random
import numpy as np
def h(s, theta, x):
```

```
    if s * (x - theta) > 0:
        return 1
    else:
        return -1
def Ein(size, s, theta, data, data2):
    err = 0
    for i in range(size):
        if h(s, theta, data[i][0]) != data[i][1]:
            err = err +1
    return err/size
data_size = 2
tou = 0.1
Eoutin = []
for j in range(10000):
    data = []
    data2 = [-1]
    for i in range(data_size):
        x = random.uniform(-1, 1)
        y = random.uniform(0, 1)
        if y < tou:
            data.append((x, -h(1, 0, x)))
        else:
            data.append((x, h(1, 0, x)))
    data.sort(key = lambda s: s[0])
    for i in range(data_size-1):
        data2.append((data[i][0]+data[i+1][0])/2)
    min_Ein = 1
    cur_s = 1
    cur_theta = -1
    for i in data2:
        s = -1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
        s = 1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
```

```
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
    if cur_s == 1:
        E = abs(cur_theta)/2
    else:
        E = (2-abs(cur_theta))/2
    Eoutin.append(E*(1-2*tou) + tou - min_Ein)
average = 0
for i in Eoutin:
    average = average + i
average = average/10000
print('Eout-Ein = {}'.format(average))
```

19

```
import math
import random
import numpy as np
def h(s, theta, x):
    if s * (x - theta) > 0:
        return 1
    else:
        return -1
def Ein(size, s, theta, data, data2):
    err = 0
    for i in range(size):
        if h(s, theta, data[i][0]) != data[i][1]:
            err = err +1
    return err/size
data_size = 20
tou = 0.1
Eoutin = []
for j in range(10000):
    data = []
    data1 = []
    data2 = [-1]
    for i in range(data_size):
        x = random.uniform(-1, 1)
        y = random.random()
        if y < tou:
```

```
            data.append((x, -h(1, 0, x)))
        else:
            data.append((x, h(1, 0, x)))
        data1.append(x)
    data.sort(key = lambda s: s[0])
    data1.sort()
    for i in range(data_size-1):
        data2.append((data1[i]+data1[i+1])/2)
    min_Ein = 1
    cur_s = 1
    cur_theta = -1
    for i in data2:
        s = -1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
        s = 1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
    if cur_s == 1:
        E = abs(cur_theta)/2
    else:
        E = (2-abs(cur_theta))/2
    Eoutin.append(E*(1-2*tou) + tou - min_Ein)
average = 0
for i in Eoutin:
    average = average + i
average = average/10000
print('Eout-Ein = {}'.format(average))
```

20

```
import math
import random
import numpy as np
def h(s, theta, x):
    if s * (x - theta) > 0:
```

```
        return 1
    else:
        return -1
def Ein(size, s, theta, data, data2):
    err = 0
    for i in range(size):
        if h(s, theta, data[i][0]) != data[i][1]:
            err = err +1
    return err/size
data_size = 200
tou = 0.1
Eoutin = []
for j in range(10000):
    data = []
    data2 = [-1]
    for i in range(data_size):
        x = random.uniform(-1, 1)
        y = random.random()
        if y < tou:
            data.append((x, -h(1, 0, x)))
        else:
            data.append((x, h(1, 0, x)))
    data.sort(key = lambda s: s[0])
    for i in range(data_size-1):
        data2.append((data[i][0]+data[i+1][0])/2)
    min_Ein = 1
    cur_s = 1
    cur_theta = -1
    for i in data2:
        s = -1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
            min_Ein = Ein(data_size, s, i, data, data2)
        s = 1
        if Ein(data_size, s, i, data, data2)<min_Ein:
            cur_s = s
            cur_theta = i
```

```python
            min_Ein = Ein(data_size, s, i, data, data2)
    if cur_s == 1:
        E = abs(cur_theta)/2
    else:
        E = (2-abs(cur_theta))/2
    Eoutin.append(E*(1-2*tou) + tou - min_Ein)
average = 0
for i in Eoutin:
    average = average + i
average = average/10000
print('Eout-Ein = {}'.format(average))
```