HTML Homework 2

資工二 B10705005 陳思如

1. D

we can make a circle with radius = 1, and the center be the lucky point.

then we can map all the N data onto the circle.

we then make our perceptrons be the line that cross the center, which is the diameter.

For a perceptron, we can have the data above the line to be correct or wrong, which have 2 outcomes.

Also, a perceptron can have $[0, N]$ data on its one side, which will have $N+1$ outcomes.

However, there will have a overlap situation, that is when above the line is correct with 0 data is the same as when above the line is wrong with N data. so we need to deduct the 2 outcomes that we counted twice.

Then, we can know that for a perceptron cross the lucky point can have $2 \cdot (N+1) - 2$ results for the perceptron. Then the growth function is $2N$.

2. A

since the perceptron will generate 2 result, either above it is wrong or above it is right,

then we have 1126 perceptrons, then we can know for N datas

$\Rightarrow 2^N \leq 1126 \Rightarrow N \leq log_2 1126.$

3. E

(a) dvc = 4;

example: **o x o x o**

this example with 5 dots cannot be shattered by two positive intervals

(b) dvc = 4

because $h(x) = sign(\sum_{i=0}^{3} w_i x^i)$ has four parameters, which are w_0, w_1, w_2, w_3

(c) dvc = ∞

since we can make the function of $sin(\omega t)$ with different all λ, then we must can find a proper hypothesis for all the data

(d) dvc = 4

because we have 4 parameters, the lower-right corner of the traingle in $\mathbb{R}^2$ and the length of two sides

(e) dvc = 3

because we have 3 parameters, the lower-left corener of the square in $\mathbb{R}^2$ and the width

4. B

this question is similar to the M intervals question, but it restricted that we have to make the datas in the interval return $+1$, else return -1.

if we have a $2 \cdot 2 = 4$ parameter, then the example: **o x o x o** will fail because its need another interval.

So, we can get the conclusion that the dvc = $2M$

5. B

$d_{vc}(H) \leq d$ have two conditions:

(1) $d_{vc}(H) = d$,then from the definition, we know that any set of $d+1$ inputs can't be shattered, and some set of d inputs can be shattered.

(2) $d_{vc}(H) < d$, then we know that any set of $d+1$ inputs can't be shattered, and also any set of d inputs cannot be shattered because we have a smaller d_{vc}

we then get the intersection of the two conditions, then we can get that 2 statement:

"any set of $d+1$ inputs can't be shattered" and

"some set of $d+1$ inputs can't be shattered" (∵ if any set can't shattered, then some sets also can't).

That is there are 2 necessary conditions of $d_{vc}(H) \leq d$.

6. B

the optimal ω for $E_{in}(\omega)$ is the one that make $\nabla E_{in}(\omega) = 0$

so we then derivate the $E_{in}(\omega)$

$E_{in}(\omega) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n^2 = \frac{1}{N} \sum_{n=1}^{N} (\omega x_n - y_n)^2 = \frac{1}{N} \sum_{n=1}^{N} \omega^2 x_n^2 - 2\omega x_n y_n + y_n^2$

$\nabla E_{in}(\omega) = \frac{1}{N} \sum_{n=1}^{N} 2\omega x_n^2 - 2x_n y_n$

so we want to have $\frac{1}{N} \sum_{n=1}^{N} 2\omega x_n^2 - 2x_n y_n = 0$,

then if $\omega = \frac{1}{N} \sum_{n=1}^{N} \frac{2x_n y_n}{2x_n^2} = \frac{1}{N} \sum_{n=1}^{N} \frac{x_n y_n}{x_n^2}$

7. C

if we have the generating samples function $s(x)$,

we get the maximum likelihood minimizing the $\sum_{n=1}^{N} -ln(s(yw^T x))$,

which we could get our maximum likelihood when $\nabla \sum_{n=1}^{N} -ln(s(yw^T x)) = 0$

(a)

$$\begin{aligned}\nabla \sum_{n=1}^{N} -ln(P(X)) &= \sum_{n=1}^{N} \nabla - ln(\frac{e^{-\lambda}\lambda^x}{x!}) = \sum_{n=1}^{N} \nabla(-ln(e^{-\lambda}) - ln(\lambda^x) + ln(x!)) \\ &= \sum_{n=1}^{N} \nabla(\lambda - xln(\lambda) + ln(x!)) = \sum_{n=1}^{N}(1 - \frac{x}{\lambda}) = -N + \sum_{n=1}^{N} \frac{x}{\lambda}\end{aligned}$$

$\Rightarrow N = \sum_{n=1}^{N} \frac{x}{\lambda} \Rightarrow \lambda = \bar{x}$

so that we can know that the maximum likelihood of $\mu = \bar{x}$

(b)

$$\begin{aligned}\sum_{n=1}^{N} \nabla - ln(P(X)) &= \sum_{n=1}^{N} \nabla - ln(\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2}) \\ &= \sum_{n=1}^{N} \nabla(-\ln(e^{-\frac{1}{2}(x-\mu)^2}) + ln(\sqrt{2\pi})) = \sum_{n=1}^{N}(\nabla(\frac{1}{2}(x-\mu)^2 + ln(\sqrt{2\pi})))' \\ &= \sum_{n=1}^{N}(\frac{1}{2} \cdot 2(x-\mu)(-1)) = \sum_{n=1}^{N}(-x+\mu)\end{aligned}$$

$\Rightarrow \sum_{n=1}^{N} x = N\mu \ \Rightarrow \mu = \bar{x}$

so that we can know that the maximum likelihood of $\mu = \bar{x}$

(c)

$$\begin{aligned}\sum_{n=1}^{N} \nabla(-ln(P(X)) &= \sum_{n=1}^{N} \nabla(-ln(\frac{1}{2}e^{-|x-\mu|})) \\ &= \sum_{n=1}^{N} \nabla(-ln(\frac{1}{2}) - (-|x-\mu|)) = \sum_{n=1}^{N} \nabla(|x-\mu|)\end{aligned}$$

if $x \geq \mu$,

$\Rightarrow \sum_{n=1}^{N} \nabla(|x-\mu|) = \sum_{n=1}^{N} \nabla(x-\mu) = -1$

if $x < \mu$

$\Rightarrow \sum_{n=1}^{N} \nabla(|x-\mu|) = \sum_{n=1}^{N} \nabla(-x+\mu) = 1$

then we can see that after the derivation the μ disappear, so that we can know that the maximum likelihood of $\mu \neq \bar{x}$

(d)

$$\begin{aligned}\sum_{n=1}^{N} \nabla(-ln(P(X)) &= \sum_{n=1}^{N} \nabla(-ln((1-\theta)^{x-1} \cdot \theta)) \\ &= \sum_{n=1}^{N} \nabla(-(x-1) \cdot ln(1-\theta) - ln(\theta)) = \sum_{n=1}^{N}((\frac{x-1}{1-\theta}) - \frac{1}{\theta})\end{aligned}$$

$\Rightarrow \frac{\sum_{n=1}^{N} x - N}{1-\theta} = \frac{N}{\theta} \Rightarrow \theta = \frac{1}{\bar{x}}$

8. A

$$
\begin{aligned}
\nabla E_{in}(w) &= \nabla(-\frac{1}{n}\sum_{n=1}^{N} ln(y \cdot h(x) \cdot w)) = -\frac{1}{n}\sum_{n=1}^{N} \nabla ln(\frac{y_n + y_n w^T x_n + |y_n w^T x_n|}{2 + 2|w^T x_n|}) \\
&= -\frac{1}{n}\sum_{n=1}^{N} \frac{2 + 2|w^T x_n|}{y_n + y_n w^T x_n + |y_n w^T x_n|} \cdot \frac{(y_n x_n + \frac{|y_n w^T x_n|}{w^T})(2 + 2|w^T x_n|) - (y_n + y_n w^T x_n + |y_n w^T x_n|)(2(2 + 2|w^T x_n|)(\frac{2|w^T x_n|}{w^T}))}{(2 + 2|w^T x_n|)^2} \\
&= -\frac{1}{n}\sum_{n=1}^{N} \frac{y_n x_n}{(y_n + y_n w^T x_n + |y_n w^T x_n|) \cdot (2 + 2|w^T x_n|)}
\end{aligned}
$$

9. B

From the slide, we can know that $\nabla E_{in}(w) = \frac{2}{N}(X^T X W - X^T Y)$

then we then get $\nabla^2 E_{in}(w) = \nabla(\nabla E_{in}(w)) = \nabla(\frac{2}{N}(X^T X W - X^T Y)) = \frac{2}{N}(X^T X)$

10. A

when $w_0 = 0$,

$$\nabla E_{in}(w_0) = \frac{2}{N}(0 - X^T Y) \neq 0$$

then we need to adjust the w,

$$
\begin{aligned}
w_1 &= w_0 + u = w_0 + (-(\nabla^2 E_{in}(w_0))(\nabla E_{in}(w_0)) \\
&= w_0 + (\frac{2}{N} X^T X)^{-1} \cdot \frac{2}{N}(X^T X w_0 - X^T Y) = w_0 + (\frac{N}{2}(X^T X)^{-1})(\frac{2}{N}(X^T X w_0 - X^T Y)) \\
&= w_0 + w_0 - X^{-1} Y = X^{-1} Y
\end{aligned}
$$

then we calculate the $\nabla E_{in}(w_1)$,

$$\nabla E_{in}(w_1) = \frac{2}{N}(X^T X X^{-1} Y - X^T Y) = \frac{2}{N}(X^T Y - X^T Y) = 0$$

so, we can see that with only 1 iteration, we get the $\nabla E_{in}(w) = 0$

11. D

we calculate the BAD probability by the formula $4 \cdot (2N)^2 \cdot e^{-\frac{1}{8}\epsilon N}$, with $\epsilon = 0.05$

(a) $N = 100$, BAD = 155077.31751

(b) $N = 1000$, BAD = 11705850.06314

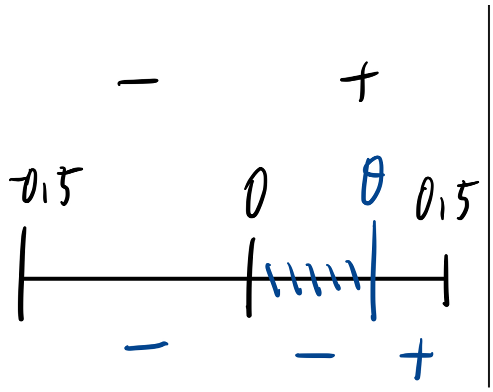
(c) $N = 10000$, BAD = 70299093.79745

(d) $N = 100000$, BAD = 0.00428, this is small than $\delta = 0.1$

(e) $N = 1000000$, BAD = 3.06969E-123

12. D

the data that are detected wrong by the hypothesis make the E_{out}



By the example I drew above, we can know that the range $[0, \theta]$ will have the wrong result.

However, we have the noise τ, so the data from $[-0.5, 0]$ and $[\theta, 0.5]$ will have the probability of τ to be wrong, and the range from $[0, \theta]$ may have the probability of τ to flip back to correct group.

Then we can know our $E_{out} = (min(|\theta|, 0.5))(1 - \tau) + (1 - min(|\theta|, 0.5))(\tau) = min(|\theta|, 0.5)(1 - 2\tau) + \tau$

Q13 - Q17

source code:

```
####Q13-Q17
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mean
import math

#create data set #keep the order x1 <= x2
time = 10000
dsize = 2 #Q13 Q15 size = 2 #Q14 Q16 size = 128
tau = 0 #Q13 Q14 tau = 0 #Q15 Q16 tau = 0.2

data = []
res = []
test = []
y = []

for i in range(time):
    data.append([])
    for j in range(dsize):
        data[i].append(random.uniform(-0.5, 0.5,))
    data[i].sort()
    res.append([])
    for j in range(dsize):
        res[i].append(-1)
        if(data[i][j] > 0):
            res[i][j] = 1
        flip = random.choices([1, -1], weights = (1-tau, tau), k = 1)
        res[i][j] *= flip[0]
    test.append(random.uniform(-0.5, 0.5))
    y.append(-1)
    if(test[i] > 0):
        y[i] = 1
    flip = random.choices([1, -1], weights = (1-tau, tau), k = 1)
    y[i] *= flip[0]


Eout_Ein = []
cal = []
#run the experiment 10000 times
for i in range(time):
    min_Ein = float(2)
    s_result = 0
    theta_result = 0

    theta = []
    theta.append(-math.inf)
    for k in range(1,dsize):
      if(data[i][k] != data[i][k-1]):
        theta.append(float((float(data[i][k-1]+data[i][k]))/2))

    s = []
    s.append(-1)
    s.append(1)

    #get the g
    for j in range(2):
        for k in range(len(theta)):
            sum_Ein = float(0)
            for l in range(dsize):
                sign = -1
                if(data[i][l] - theta[k] > 0):
                    sign = 1

                if(s[j]*sign*res[i][l] < 0):
                    sum_Ein += 1

            sum_Ein = float(float(sum_Ein)/float(dsize))
            if(sum_Ein < min_Ein):
```

```
                min_Ein = sum_Ein
                s_result = s[j]
                theta_result = theta[k]
            elif(sum_Ein == min_Ein):
                if((s[j]*theta[k]) < (s_result*theta_result)):
                    min_Ein = sum_Ein
                    s_result = s[j]
                    theta_result = theta[k]

    #do the Eout

    sum_Eout = 0
    for j in range(time):
        s = -1
        if(test[i]-theta_result > 0):
            s = 1
        if(s_result*s*y[i] < 0):
            sum_Eout += 1
    sum_Eout = float(sum_Eout)
    sum_Eout /= float(time)

    cal_Eout = min(abs(theta_result), 0.5)*(1-2*tau) +tau
    cal.append(cal_Eout - min_Ein)
    Eout_Ein.append(sum_Eout - min_Ein)
print(mean(Eout_Ein))
print(mean(cal))
```

13. B

set the `dsize = 2` `tau = 0`

I get the mean of $E_{out} - E_{in}$ from generating random test data = 0.2887

I get the mean of $E_{out} - E_{in}$ from Q12 calculation = 0.28871735370393947

14. B

set the `dsize = 128` `tau = 0`

I get the mean of $E_{out} - E_{in}$ from generating random test data = 0.0041

I get the mean of $E_{out} - E_{in}$ from Q12 calculation = 0.0038578676208335756

15. C

set the `dsize = 2` `tau = 0.2`

I get the mean of $E_{out} - E_{in}$ from generating random test data = 0.4253

I get the mean of $E_{out} - E_{in}$ from Q12 calculation = 0.3901077554492545

16. B

set the `dsize = 128` `tau = 0.2`

I get the mean of $E_{out} - E_{in}$ from generating random test data = 0.0153671875

I get the mean of $E_{out} - E_{in}$ from Q12 calculation = 0.014160031860089956

17. C

```python
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mean
import math

file = open('hw2_train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

data_n = np.array(data)
np.sort(data_n, axis = 0)

dimension = len(data[0]) - 1
size = len(data)
y = len(data[0]) - 1

min_Ein = float(2)
for i in range(dimension):

    theta = np.zeros(size)
    theta[0] = -math.inf
    for k in range(1,size):
        theta[k] = float((float(data_n[k-1][i]+data_n[k][i]))/2)

    s = np.zeros(2)
    s[0] = -1
    s[1] = 1

    for j in range(2):
        for k in range(size):
            sum_Ein = float(0)
            for l in range(size):
                sign = -1
                if(data_n[l][i] - theta[k] > 0):
                    sign = 1

                if(s[j]*sign*data_n[l][y] < 0):
                    sum_Ein += 1
            sum_Ein = float(sum_Ein)
            sum_Ein /= size

            if(sum_Ein < min_Ein):
                min_Ein = sum_Ein

print(min_Ein)
```

I get the min_Ein = 0.026041666666666668

18. E

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mean
import math

file = open('hw2_train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

data_n = np.array(data)
np.sort(data_n, axis = 0)

dimension = len(data[0]) - 1
size = len(data)
y = len(data[0]) - 1

min_Ein = float(2)
s_result = 0
theta_result = 0
i_result = 0
for i in range(dimension):

    theta = np.zeros(size)
    theta[0] = -math.inf
    for k in range(1,size):
        theta[k] = float((float(data_n[k-1][i]+data_n[k][i]))/2)

    s = np.zeros(2)
    s[0] = -1
    s[1] = 1

    for j in range(2):
        for k in range(size):
            sum_Ein = float(0)
            for l in range(size):
                sign = -1
                if(data_n[l][i] - theta[k] > 0):
                    sign = 1

                if(s[j]*sign*data_n[l][y] < 0):
                    sum_Ein += 1
            sum_Ein = float(sum_Ein)
            sum_Ein /= size

            if(sum_Ein < min_Ein):
                min_Ein = sum_Ein
                s_result = s[j]
                theta_result = theta[k]
                i_result = i

#print(min_Ein)

file = open('hw2_test.txt', 'r')
test = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    test.append(numbers)

test_n = np.array(test)

dimension = len(test_n[0]) - 1
size = len(test_n)
y = len(test_n[0]) - 1

Eout = float(0)
for i in range(size):
```

```
        s = -1
        if(test_n[i][i_result] - theta_result > 0):
            s = 1
        if(s_result*s*test_n[i][y] < 0):
            Eout += 1

Eout = float(Eout)
Eout /= float(size)
print(Eout)
```

I get the Eout = 0.078125

19. D

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mean
import math

file = open('hw2_train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

data_n = np.array(data)
np.sort(data_n, axis = 0)

dimension = len(data[0]) - 1
size = len(data)
y = len(data[0]) - 1

mini_Ein = float(2)
max_Ein = float(0)

for i in range(dimension):
    min_Ein = float(2)
    theta = np.zeros(size)
    theta[0] = -math.inf
    for k in range(1,size):
        theta[k] = float((float(data_n[k-1][i]+data_n[k][i]))/2)

    s = np.zeros(2)
    s[0] = -1
    s[1] = 1

    for j in range(2):
        for k in range(size):
            sum_Ein = float(0)
            for l in range(size):
                sign = -1
                if(data_n[l][i] - theta[k] > 0):
                    sign = 1

                if(s[j]*sign*data_n[l][y] < 0):
                    sum_Ein += 1
            sum_Ein = float(sum_Ein)
            sum_Ein /= size

            if(sum_Ein < min_Ein):
                min_Ein = sum_Ein
    if(min_Ein > max_Ein):
        max_Ein = min_Ein

    if(min_Ein < mini_Ein):
        mini_Ein = min_Ein

#print(max_Ein)
#print(mini_Ein)
print(max_Ein - mini_Ein)
```

I get the best of best Ein = 0.328125

I get the worst of best Ein = 0.026041666666666668

then I get ΔEin = 0.3020833333333333

20. B

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mean
import math

file = open('hw2_train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

data_n = np.array(data)
np.sort(data_n, axis = 0)

dimension = len(data[0]) - 1
size = len(data)
y = len(data[0]) - 1

mini_Ein = float(2)
max_Ein = float(0)
s_mini = 0
theta_mini = 0
i_mini = 0
s_max = 0
theta_max = 0
i_max = 0
for i in range(dimension):
    min_Ein = float(2)
    s_temp = 0
    theta_temp = 0
    i_temp = 0
    theta = np.zeros(size)
    theta[0] = -math.inf
    for k in range(1,size):
        theta[k] = float((float(data_n[k-1][i]+data_n[k][i]))/2)

    s = np.zeros(2)
    s[0] = -1
    s[1] = 1

    for j in range(2):
        for k in range(size):
            sum_Ein = float(0)
            for l in range(size):
                sign = -1
                if(data_n[l][i] - theta[k] > 0):
                    sign = 1

                if(s[j]*sign*data_n[l][y] < 0):
                    sum_Ein += 1
            sum_Ein = float(sum_Ein)
            sum_Ein /= size

            if(sum_Ein < min_Ein):
                min_Ein = sum_Ein
                s_temp = s[j]
                theta_temp = theta[k]
                i_temp = i
    if(min_Ein > max_Ein):
        max_Ein = min_Ein
        s_max = s_temp
        theta_max = theta_temp
        i_max = i_temp
    if(min_Ein < mini_Ein):
        mini_Ein = min_Ein
        s_mini = s_temp
        theta_mini = theta_temp
        i_mini = i_temp
#print(max_Ein)
```

```python
#print(mini_Ein)
file = open('hw2_test.txt', 'r')
test = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    test.append(numbers)

test_n = np.array(test)

dimension = len(test_n[0]) - 1
size = len(test_n)
y = len(test_n[0]) - 1

mini_Eout = float(0)
for i in range(size):
        s = -1
        if(test_n[i][i_mini] - theta_mini > 0):
            s = 1
        if(s_mini*s*test_n[i][y] < 0):
            mini_Eout += 1

mini_Eout = float(mini_Eout)
mini_Eout /= float(size)
#print(mini_Eout)

max_Eout = float(0)
for i in range(size):
        s = -1
        if(test_n[i][i_max] - theta_max > 0):
            s = 1
        if(s_max*s*test_n[i][y] < 0):
            max_Eout += 1

max_Eout = float(max_Eout)
max_Eout /= float(size)
#print(max_Eout)

print(max_Eout - mini_Eout)
```

I get the best of best Eout = 0.078125

I get the worst of best Eout = 0.421875

then I get ΔEout = 0.34375