HTML HW2   r09922a21 郭蕙綺

1、(B) $X = \begin{bmatrix} 1, 1, 1, 1 \\ 1, 2, 3, 4 \\ 1, 4, 3, 2 \\ 1, 4, 2, 3 \end{bmatrix}$ for any $y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$, find w such that $\text{sign}(Xw) = y$

$\Rightarrow Xw = y \iff w = X^{-1}y$

$$\det(X) = \begin{vmatrix} 2 & 3 & 4 \\ 4 & 3 & 2 \\ 4 & 2 & 3 \end{vmatrix} - \begin{vmatrix} 1 & 3 & 4 \\ 1 & 3 & 2 \\ 1 & 2 & 3 \end{vmatrix} + \begin{vmatrix} 1 & 2 & 4 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \end{vmatrix} - \begin{vmatrix} 1 & 2 & 3 \\ 1 & 4 & 3 \\ 1 & 4 & 2 \end{vmatrix}$$

$$= [18+24+32-48-8-36] - [9+6+8-12-4-9] + [12+4+16-16-8-6]$$

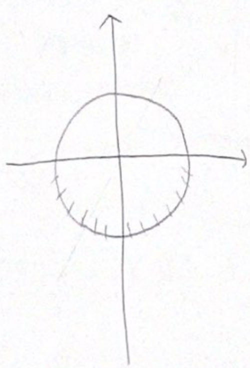
$$- [8+6+12-12-12-4]$$

$$= -18-(-2)+2-(-2) = -12 \neq 0 \Rightarrow X \text{ is invertible}$$

and dvc of 3D perceptron $= 4 \geq (d+1) = 4$

$\therefore$ X can be shattered

2.



由於是origin-passing，所以最後的line最多只能通過一、三象限 or 二、四象限.

因此，在計算max hypothesis 時、只需考慮 N 個點均落在同一個半圓的狀況（從一or三象限出發的線，落在二、四象限的點的label已經被決定好，反之亦然）

考慮 line 從一、二象限出發，能選擇的終點為 N−1 的interval

考慮正負对稱，共有 $2(N-1)$ 的 hypothesis

又考慮全部為正及全部為負的狀況共2种

最終 $m_H(N) = 2(N-1)+2 = 2N$ #

3. $h(x) = \begin{cases} +1 & \text{if } a \le \sum_{i=1}^{d} x_i^2 \le b \\ -1 & \text{otherwise} \end{cases}$ ⇒ can be viewed as  ▨ positive otherwise, negative

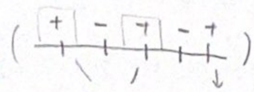
So, the growth function is like positive interval, choose 2 points between N+1 intervals

growth function $= C^{N+1}_{2} + 1$ → consider if all the points are negative

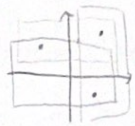
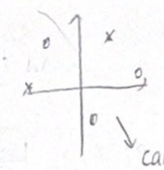
4. Since minimun break point of positive intervals is 3.

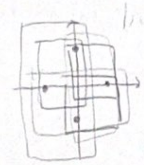
VC dimension = 2 #

5. (A) If we have 2 positive intervals, when we have 5 points,

we cannot handle for 2^5 situations ()

which means $m_H < 2^5 = 2^N$, $d_{VC} = 5 - 1 = 4$

(B) For 3 inputs :  can be shattered. For 5 input.  can't find 2^5 rectangles. cannot be classified.

For 4 inputs :  also can be shattered

So, max break point = 5, $d_{VC} = 4$ #

(D) let $X = \begin{bmatrix} 1 & x_1 & x_1 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$ similar to 3D perceptron.

$d_{VC} = 4$

(C) For 2D perceptrons, $h(x) = \text{sign}(\sum_{i=0}^{2} w_i x_i) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$

A line draw on plane based on "$w_0 + w_1 x_1 + w_2 x_2 = 0$" which means that

"$w_0 + w_1 x_1 + w_2 x_2 = 0$" equal to "$-w_0 - w_1 x_1 - w_2 x_2 = 0$".

For example, "$2x_1 + 3x_2 - 10 = 0$" can be rewrite as "$-2x_1 - 3x_2 + 10 = 0$"

HW2

6. dvc(H) is the largest N for which $m_H(N) = 2^N$

$2^{10} < 1126 < 2^{11} \Rightarrow m_H(10) = 2^{10}, \; m_H(11) = 1126$

$\therefore$ the largest N satisfy $m_H(N) = 2^N$ is 10. #

7. $E_{out}(g) - E_{out}(g_*) = E_{out}(g) - E_{in}(g) + \underline{E_{in}(g) - E_{in}(g_*)} + E_{in}(g_*) - E_{out}(g_*)$

$\leq \max_{h\in H} |E_{out}(h) - E_{in}(h)| + 0 \; (\because E_{in}(g) < E_{in}(g_*)) + \max_{h\in H} |E_{in}(h) - E_{out}(h)|$

$\leq 2 \max_{h \in H} |E_{out}(h) - E_{in}(h)|$

$\Rightarrow$ Find $|E_{out}(h) - E_{in}(h)|$ upper bound

$P[\exists h \in H \text{ s.t. } |E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2M \exp(-2\epsilon^2 N)$

$\Rightarrow P[\exists h \in H \text{ s.t. } |E_{in}(h) - E_{out}(h)| \leq \epsilon] \geq 1 - 2M \exp(-2\epsilon^2 N)$

and multiple-bin Hoeffding bound with probability more than $1-\delta$

$1 - 2M\exp(-2\epsilon^2 N) \leq 1 - \delta \;\Rightarrow\; \delta \leq 2M\exp(-2\epsilon^2 N) \Rightarrow \ln \frac{\delta}{2M} \leq -2\epsilon^2 N$

$\Rightarrow \frac{1}{2N} \ln \frac{2M}{\delta} \geq \epsilon^2 \;\Rightarrow\; \epsilon \leq \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

$\therefore E_{out}(g) - E_{out}(g_*) \leq 2\sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

8. $P[\exists h \in H \text{ s.t. } |E_{in}(h) - E_{out}(h)| > \epsilon] \le 4 m_H(2N) \exp(-\frac{1}{8}\epsilon^2 N)$

$4 m_H(2N) \exp(-\frac{1}{8} \cdot 0.01 \cdot N) \le 0.1$, According to lecture 04 P.25, $m_H(N) = N+1$

$4 \cdot (2N+1) \cdot \exp(-\frac{1}{8} \cdot 0.01 \cdot N) \le 0.1$

$-\frac{1}{8} \cdot 0.01 \cdot N \le \ln \frac{0.1}{4 \cdot (2N+1)} \Rightarrow N \ge 800 \cdot \ln \frac{40(2N+1)}{\cancel{0.1}}$

$10000 < 800 \cdot \ln 40 \cdot 20001 \fallingdotseq 10874$

$11000 > 800 \cdot \ln 40 \cdot 22001 \fallingdotseq 10950$ #

9. To minimizes the right-hand-side of the Taylor's expansion,

we want $\nabla [E(u) + b_E(u)^T (w-u) + \frac{1}{2}(w-u)^T A_E(u)(w-u)] = 0$

$\Rightarrow b_E(u) + \frac{1}{\cancel{2}} \cdot \cancel{2} \cdot 1 \cdot A_E(u)(w-u) = 0$

$\Rightarrow A_E(u)(w-u) = -b_E(u)$

$\Rightarrow w - u = -[A_E(u)]^{-1} b_E(u)$ #

10. $E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n w^T x_n))$,

$A_E(w_t) = \frac{\partial^2 E}{\partial^2 w} = \frac{\partial}{\partial w}\left[\frac{\partial}{\partial w}\left(\frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n w^T x_n))\right)\right] = \frac{\partial}{\partial w}\left(\frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)\cdot(-y_n x_n)}{1+\exp(-y_n w^T x_n)}\right)$

$= \frac{\partial}{\partial w}\left[\frac{1}{N}\sum_{n=1}^{N} \frac{(-y_n x_n)}{\exp(y_n w^T x_n)+1}\right] = \frac{1}{N}\sum_{n=1}^{N} (-y_n x_n) \frac{-(y_n x_n)\cdot \exp(y_n w^T x_n)}{(\exp(y_n w^T x_n)+1)(\exp(y_n w^T x_n)+1)}$

$= \frac{1}{N}\sum_{n=1}^{N} \underbrace{\frac{1}{\exp(y_n w^T x_n)+1}}_{= h_t(y_n x_n)} \cdot \underbrace{\frac{\exp(y_n w^T x_n)}{\exp(y_n w^T x_n)+1}}_{= \frac{1}{1+\exp(-y_n w^T x_n)} = h_t(-y_n x_n)} \cdot x_n x_n^T = \frac{1}{N}\sum_{n=1}^{N} h_t(y_n x_n) h_t(-y_n x_n) x_n x_n^T$ #

HW2

11. (a) $W_{LIN} = (X^TX)^{-1}X^Ty$, $W_{LIN} = X^+y$.

As a result, $(X^TX)^{-1}X^T = X^+$ when X^TX is invertible.

(b) According to Moore-Penrose pseudo-inverse's definition, "$XX^+X = X$"

For $(XX^+)^k = (XX^+)\cdot(XX^+)^{k-1} = (XX^+)^{k-1}$ ($\because XX^+X = X$)

$\Rightarrow (XX^+)^{k-1} = (XX^+)\cdot(XX^+)^{k-2} = (XX^+)^{k-2}$ ......

$\vdots$

$\Rightarrow (XX^+)^2 = (XX^+)\cdot(XX^+) = (XX^+)$

$\therefore$ for any $k \in \mathbb{Z}^+$, $(XX^+)^k = XX^+$

(c) The definition is $XX^+X = X$, only when X is invertible, XX^+ can be I_N.

This option lacks the condition "X is invertible", so it's wrong.

(d) According to Moore-Penrose pseudo inverse, X^+X, XX^+, $(I - X^+X)$, are idempotent matrix. And the trace of an idempotent matrix (the sum of the elements on its main diagonal) equals the rank of the matrix.

12. $\because$ p.d.f. $\therefore$ $P(y|X) = \prod_{i=1}^{n} (2\pi\sigma^2)^{-\frac{1}{2}} \cdot e^{-\frac{1}{2\sigma^2}(y_i - w^Tx_i)^2}$

$= (2\pi\sigma^2)^{-\frac{n}{2}} \cdot e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - w^Tx_i)^2}$

$= (2\pi\sigma^2)^{-\frac{n}{2}} \cdot e^{-\frac{1}{2\sigma^2}(y - w^Tx)^T(y - w^Tx)}$

$w^* = \operatorname{argmax}\left[(2\pi\sigma^2)^{-\frac{n}{2}} \cdot e^{-\frac{1}{2\sigma^2}(y-w^Tx)^T(y-w^Tx)}\right]$

$= \operatorname{argmax} \ln\left[(2\pi\sigma^2)^{-\frac{n}{2}} \cdot e^{-\frac{1}{2\sigma^2}(y-w^Tx)^T(y-w^Tx)}\right]$

$= -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y - w^Tx)^T(y - w^Tx)$

Want $\nabla\left[-\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y-w^Tx)^T(y-w^Tx)\right] = 0$

$\frac{\partial w^*}{\partial w} = 0 - \frac{1}{2\sigma^2}\frac{\partial}{\partial w}(y^2 - 2w^Txy + (w^Tx)^2)$

$= 0 - \frac{1}{2\sigma^2}\left[0 - 2X^Ty + 2wX^TX\right]$

$= 0$

$\Rightarrow 2X^Ty = 2wX^TX$

$\Rightarrow w^* = (X^TX)^{-1}X^Ty$ #

```python
import numpy as np
import random
mean1 = [2,3]
cov1 = [[0.6,0],[0, 0.6]]
mean2 = [0,4]
cov2 = [[0.4,0],[0, 0.4]]

# Q13
total_Ein = 0
for t in range(100):
    np.random.seed(t*11+2)
    random.seed(t*11+2)
    train_y = []
    train_x = []
    for i in range(200):
        train_y.append(random.choice([1,-1]))
        if train_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            train_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            train_x.append([1,x1,x2])
    test_y = []
    test_x = []
    for i in range(5000):
        test_y.append(random.choice([1,-1]))
        if test_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            test_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            test_x.append([1,x1,x2])

    X = np.array(train_x)
    y = np.array(train_y)
    W = np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose()), y)
    Ein = 0
    for i in range(200):
        Ein += (np.dot(W.transpose(), X[i]) - y[i])**2
    total_Ein += Ein/200
print(total_Ein/100)
```

```python
# Q14
total_E = 0
for t in range(100):
    np.random.seed(t*11+2)
    random.seed(t*11+2)
    train_y = []
    train_x = []
    for i in range(200):
        train_y.append(random.choice([1,-1]))
        if train_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            train_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            train_x.append([1,x1,x2])
    test_y = []
    test_x = []
    for i in range(5000):
        test_y.append(random.choice([1,-1]))
        if test_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            test_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            test_x.append([1,x1,x2])

    X = np.array(train_x)
    y = np.array(train_y)
    X_test = np.array(test_x)
    y_test = np.array(test_y)

    W = np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose()), y)

    Ein = 0
    for i in range(200):
        if (np.dot(W.transpose(), X[i]) * y[i] < 0):
            Ein += 1
    Ein = Ein/200

    Eout = 0
    for i in range(5000):
        if (np.dot(W.transpose(), X_test[i]) * y_test[i] < 0):
            Eout += 1
    Eout = Eout/5000

    total_E += (np.absolute(Ein-Eout))
print(total_E/100)
```

```python
import math
def cross_entropy(s):
    return (1/(1+math.exp(-s)))

# Q15
E_A = 0
E_B = 0
for t in range(100):
    np.random.seed(t*11+2)
    random.seed(t*11+2)
    train_y = []
    train_x = []
    for i in range(200):
        train_y.append(random.choice([1,-1]))
        if train_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            train_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            train_x.append([1,x1,x2])
    test_y = []
    test_x = []
    for i in range(5000):
        test_y.append(random.choice([1,-1]))
        if test_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            test_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            test_x.append([1,x1,x2])

    #linear regression
    X = np.array(train_x)
    y = np.array(train_y)
    X_test = np.array(test_x)
    y_test = np.array(test_y)

    W = np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose()), y)

    Eout = 0
    for i in range(5000):
        if (np.dot(W.transpose(), X_test[i]) * y_test[i] < 0):
            Eout += 1
    Eout = Eout/5000
    E_A += Eout

    #logistic regression
    w = np.array([0, 0, 0])
    v = np.array([0.0, 0.0, 0.0])
    lr = 0.1
    for iteration in range(500):
        for idx_ in range(200):
            v += cross_entropy(-y[idx_]*np.dot(w, X[idx_]))*y[idx_]*X[idx_]
        v = v/200
        w = w + lr*v

    Eout = 0
    for i in range(5000):
        if (np.dot(w.transpose(), X_test[i]) * y_test[i] < 0):
            Eout += 1
    Eout = Eout/5000
    E_B += Eout

print(E_A/100, E_B/100)
```

```python
# Q16
E_A = 0
E_B = 0
for t in range(100):
    np.random.seed(t*11+2)
    random.seed(t*11+2)
    train_y = []
    train_x = []
    for i in range(200):
        train_y.append(random.choice([1,-1]))
        if train_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            train_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            train_x.append([1,x1,x2])

    for i in range(20):
        train_y.append(1)
        mean_o = [6,0]
        cov_o = [[0.3, 0],[0, 0.1]]
        x1, x2 = np.random.multivariate_normal(mean_o, cov_o)
        train_x.append([1,x1,x2])

    test_y = []
    test_x = []
    for i in range(5000):
        test_y.append(random.choice([1,-1]))
        if test_y[i] == 1:
            x1, x2 = np.random.multivariate_normal(mean1, cov1)
            test_x.append([1,x1,x2])
        else:
            x1, x2 = np.random.multivariate_normal(mean2, cov2)
            test_x.append([1,x1,x2])

    #linear regression
    X = np.array(train_x)
    y = np.array(train_y)
    X_test = np.array(test_x)
    y_test = np.array(test_y)

    W = np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose()), y)

    Eout = 0
    for i in range(5000):
        if (np.dot(W.transpose(), X_test[i]) * y_test[i] < 0):
            Eout += 1
    Eout = Eout/5000
    E_A += Eout

    #logistic regression
    w = np.array([0, 0, 0])
    v = np.array([0.0, 0.0, 0.0])
    lr = 0.1
    for iteration in range(500):
        for idx_ in range(220):
            v += cross_entropy(-y[idx_]*np.dot(w, X[idx_]))*y[idx_]*X[idx_]
        v = v/220
        w = w + lr*v

    Eout = 0
    for i in range(5000):
        if (np.dot(w.transpose(), X_test[i]) * y_test[i] < 0):
            Eout += 1
    Eout = Eout/5000
    E_B += Eout

print(E_A/100, E_B/100)
```