Machine Learning Homework 2

1

將 $\sigma = 0.1$, $d = 19$ 代入 $\mathbb{E}_D$ 公式中

$$\mathbb{E}_D[E_{in}(w_{lin})] = \sigma^2(1 - \frac{d+1}{N}) \geq 0.005$$
$$0.01 * (1 - \frac{20}{N}) \geq 0.005$$
$$0.5N \geq 20 \ \ , \ \ N \geq 40$$

答案為 [d].

2

計算 $[0, 1]$ 區間的 squared error

$$\begin{aligned}\int_0^1 (x^2 - (w_0 + w_1 x))^2 dx &= \int_0^1 x^4 - 2x^2(w_0 + w_1 x) + (w_0 + w_1 x)^2 dx \\ &= \int_0^1 x^4 - 2w_1 x^3 - 2w_0 x^2 + w_1^2 x^2 + 2w_0 w_1 x + w_0^2 dx \\ &= \frac{1}{5}x^5 - \frac{1}{2}w_1 x^4 - \frac{2}{3}w_0 x^3 + \frac{1}{3}w_1^2 x^3 + w_0 w_1 x^2 + w_0^2 \Big|_0^1 \\ &= \frac{1}{5} - \frac{1}{2}w_1 - \frac{2}{3}w_0 + \frac{1}{3}w_1^2 + w_0 w_1 + w_0^2\end{aligned}$$

這邊用 $E(w_0, w_1)$ 代指上面的式子，為求最小值，對w_0, w_1偏微分

$$\frac{\partial E}{\partial w_0} = -\frac{2}{3} + w_1 + 2w_0 = 0$$
$$\frac{\partial E}{\partial w_1} = -\frac{1}{2} + \frac{2}{3}w_1 + w_0 = 0$$

透過聯立方程式得

$$w_0 = -\frac{1}{6} \ \ , \ \ w_1 = 1$$

答案為 [c].

3

由兩筆training data $(x_1, x_1^2), (x_2, x_2^2)$ 計算 w_1, w_2

$$x_1^2 = w_0 + w_1 x_1$$
$$x_2^2 = w_0 + w_1 x_2$$
$$w_1(x_1 - x_2) = x_1^2 - x_2^2 = (x_1 + x_2)(x_1 - x_2)$$

由於題目保證 $x_2 \neq x_2$，固可由聯立解得

$$w_1 = x_1 + x_2$$
$$w_2 = -x_1 x_2$$

代入第二題的 $E(w_0, w_1)$ 即可得到 $E_{out}(g)$

$E_{out} = \frac{1}{5} - \frac{1}{2}(x_1 + x_2) + \frac{2}{3}x_1x_2 + \frac{1}{3}(x_1 + x_2)^2 - (x_1 + x_2)x_1x_2 + (x_1x_2)^2$

透過對 x_1, x_2 在 $[0, 1]$ 區間雙重積分計算期望值

$$\begin{aligned}
&\mathbb{E}_D[E_{out}] \\
&= \int_0^1 \int_0^1 \frac{E_{out}}{(1-0)(1-0)}\, dx_1\, dx_2 \\
&= \int_0^1 \int_0^1 \frac{1}{5} - \frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{1}{3}x_1^2 + \frac{4}{3}x_1x_2 + \frac{1}{3}x_2^2 - x_1^2x_2 - x_1x_2^2 + x_1^2x_2^2\, dx_1\, dx_2 \\
&= \int_0^1 \frac{1}{5}x_1 - \frac{1}{4}x_1^2 - \frac{1}{2}x_1x_2 + \frac{1}{9}x_1^3 + \frac{2}{3}x_1^2x_2 \\
&\qquad\qquad + \frac{1}{3}x_1x_2^2 - \frac{1}{3}x_1^3x_2 - \frac{1}{2}x_1^2x_2^2 + \frac{1}{3}x_1^3x_2^2 \Big|_0^1\, dx_2 \\
&= \int_0^1 \frac{1}{5} - \frac{1}{4} - \frac{1}{2}x_2 + \frac{1}{9} + \frac{2}{3}x_2 + \frac{1}{3}x_2^2 - \frac{1}{3}x_2 - \frac{1}{2}x_2^2 + \frac{1}{3}x_2^2\, dx_2 \\
&= \frac{1}{5}x_2 - \frac{1}{4}x_2 - \frac{1}{4}x_2^2 + \frac{1}{9}x_2 + \frac{1}{3}x_2^2 + \frac{1}{9}x_2^3 - \frac{1}{6}x_2^2 - \frac{1}{6}x_2^3 + \frac{1}{9}x_2^3 \Big|_0^1 \\
&= \frac{1}{5} - \frac{1}{4} - \frac{1}{4} + \frac{1}{9} + \frac{1}{3} + \frac{1}{9} - \frac{1}{6} - \frac{1}{6} + \frac{1}{9} = \frac{1}{30}
\end{aligned}$$

由於 $E_{in} = [x_1^2 - (x_1 + x_2)x_1 + x_1x_2)]^2 + [x_2^2 - (x_1 + x_2)x_2 + x_1x_2)]^2 = 0$

$$\mathbb{E}_D[|E_{in} - E_{out}|] = \frac{1}{30}$$

答案為 [e].

4

由題目可知 $y_n = -1$ 會對應到 $y'_n = 0$；$y_n = +1$ 會對應到 $y'_n = 1$

Case $y_n = -1$:

$$-\ln\theta(y_n w^T x_n) = -\ln\theta(-w^T x_n) = -(1-y'_n)\ln\theta(-w^T x_n)$$

其中 ln 前的 $(1-y'_n)$ 保證 $y'_n = 1$ 時輸出0

Case $y_n = +1$:

$$-\ln\theta(y_n w^T x_n) = -\ln\theta(w^T x_n) = -y'_n\ln\theta(w^T x_n)$$

其中 ln 前的 y'_n 保證 $y'_n = 0$ 時輸出0

兩式相加得解為

$$\begin{aligned} E_{in}(w) &= \frac{1}{N}\sum_{n=1}^{N} -\ln\theta(y_n w^T x_n) \\ &= \frac{1}{N}\sum_{n=1}^{N} -y'_n\ln\theta(w^T x_n) - (1-y'_n)\ln\theta(-w^T x_n) \end{aligned}$$

可看出答案不在厂[a] $\sim$ [d] 中。

答案為 [e].

5

第一點，由第三張投影片公式可推得

$$\begin{aligned} \delta &\geq 2M\exp(-2\epsilon^2 N), \quad \epsilon = \nu - \mu \\ \frac{\delta}{2M} &\geq \exp(-2(\nu-\mu)^2 N) \\ \ln\frac{\delta}{2M} &\geq -2(\nu-\mu)^2 N \\ \frac{1}{2N}\ln\frac{2M}{\delta} &\geq (\nu-\mu)^2 \\ \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}} &\geq \mu - \nu \\ \mu &\leq \nu + \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}} \end{aligned}$$

由於 $M \geq 1$，我們無法保證 $\mu \leq \nu + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$，此敘述為 False

第二點，用微分求$\ln likelihood(\hat{\mu})$最大值

$$\begin{aligned}
likelihood(\hat{\mu}) &= \prod_{n=1}^{N} y_n\hat{\mu} + (1-y_n)(1-\hat{\mu}) \\
\ln likelihood(\hat{\mu}) &= \sum_{n=1}^{N} \ln(y_n\hat{\mu} + (1-y_n)(1-\hat{\mu})) \\
\frac{\partial \ln likelihood(\hat{\mu})}{\partial \hat{\mu}} &= \sum_{n=1}^{N} \frac{y_n - (1-y_n)}{y_n\hat{\mu} + (1-y_n)(1-\hat{\mu})} \\
&= \sum_{n=1}^{N} \frac{2y_n - 1}{y_n\hat{\mu} + (1-y_n)(1-\hat{\mu})} = 0 \\
\frac{\sum_{n=1}^{N} y_n}{\hat{\mu}} &= \frac{N - \sum_{n=1}^{N} y_n}{1-\hat{\mu}} \\
\sum_{n=1}^{N} y_n - \hat{\mu}\sum_{n=1}^{N} y_n &= N\hat{\mu} - \hat{\mu}\sum_{n=1}^{N} y_n \\
\hat{\mu} &= \frac{1}{N}\sum_{n=1}^{N} y_n = \nu
\end{aligned}$$

此敘述為 True

第三點，用微分求 $E^{sqr}(\hat{y})$ 最小值

$$\begin{aligned}
E^{sqr}(\hat{y}) &= \frac{1}{N}\sum_{n=1}^{N} (\hat{y} - y_n)^2 \\
\frac{E^{sqr}(\hat{y})}{\partial \hat{y}} &= \frac{1}{N}\sum_{n=1}^{N} 2(\hat{y} - y_n) \\
&= 2(\hat{y} - \frac{1}{N}\sum_{n=1}^{N} y_n) = 0 \\
\hat{y} &= \frac{1}{N}\sum_{n=1}^{N} y_n = \nu
\end{aligned}$$

此敘述為 True

第四點，用微分求 $E^{ce}(\hat{y})$ 最小值

$$
\begin{aligned}
E^{ce}(\hat{y}) &= -\frac{1}{N}\sum_{n=1}^{N} y_n \ln \hat{y} + (1-y_n)\ln(1-\hat{y}) \\
\frac{\partial E^{ce}(\hat{y})}{\partial \hat{y}} &= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n}{\hat{y}} - \frac{1-y_n}{1-\hat{y}} \\
&= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n(1-\hat{y}) + (y_n-1)\hat{y}}{\hat{y}(1-\hat{y})} \\
&= \frac{1}{N\hat{y}(1-\hat{y})}\sum_{n=1}^{N} \hat{y} - y_n \\
&= \frac{1}{\hat{y}(1-\hat{y})}(\hat{y} - \frac{1}{N}\sum_{n=1}^{N} y_n) = 0 \\
\hat{y} &= \frac{1}{N}\sum_{n=1}^{N} y_n = \nu
\end{aligned}
$$

此敘述為 True，共三個敘述正確，答案為 [d].

6

當分類錯誤時 $yw^Tx \leq 0$ 反之 $yw^Tx > 0$。故為 $\max(0, -yw^Tx)$，而預測錯誤越大 err 越大，故不用乘-1。推得

$$err(w, x, y) = max(0, -yw^Tx)$$

答案為 [a].

7

題目要求取 $k_{th}column$ 的 gradient，故由 err 對 w_k 微分得

$$
\begin{aligned}
\frac{\partial err(W, x_n, y_n)}{\partial w_k} &= \frac{\partial err(W, x_n, y_n)}{h_{y_n}(x_n)} \frac{\partial h_{y_n}(x_n)}{\partial w_k} \\
&= -\frac{1}{h_{y_n}(x_n)} \frac{exp(2w_k^Tx_n)x_n - (\sum_{i=1}^{K}\exp(w_k^Tx_n))exp(w_k^Tx_n)x_n}{(\sum_{i=1}^{K}\exp(w_k^Tx_n))^2} \\
&= -\frac{exp(w_k^Tx_n)x_n - (\sum_{i=1}^{K}\exp(w_k^Tx_n))x_n}{\sum_{i=1}^{K}\exp(w_k^Tx_n)} \\
&= -(h_{y_n}(x_n) - 1)x_n = (1 - h_{y_n}(x_n))x_n
\end{aligned}
$$

答案為 [a].

8

對四筆 data 做 quadratic transform

$$\Phi_2(x_1) = (1, 0, 1, 0, 0, 1)$$
$$\Phi_2(x_2) = (1, 0, -1, 0, 0, 1)$$
$$\Phi_2(x_3) = (1, -1, 0, 1, 0, 0)$$
$$\Phi_2(x_4) = (1, 1, 0, 1, 0, 0)$$

取四個資料對$w = (0, 0, 0, 0, 0, -1)$ 做 $sign$ 的結果

$$sign(wx_1) = sign(-1) = -1 = y_1$$
$$sign(wx_2) = sign(-1) = -1 = y_2$$
$$sign(wx_3) = sign(\ 0\ ) = +1 = y_3$$
$$sign(wx_4) = sign(\ 0\ ) = +1 = y_4$$

可見 [e] 的 w 能正確分類這四筆測資，若代入其他選項則無法。
答案為 [e].

9

經過

$$\begin{aligned} w_{lin} &= (X^T X)^{-1} X^T y \\ \tilde{w} &= ((\Gamma X)^T (\Gamma X))^{-1} (\Gamma X)^T y \\ &= (X^T \Gamma^T \Gamma X)^{-1} X^T \Gamma^T y \end{aligned}$$

我們可合理推測在維度不變下，兩者所計算出能最小化 E_{in} 的 y' 相同，也就是$w_{lin}^T X = \tilde{w}^T (\Gamma X)$，故可推得

$$w_{lin}^T X = (\tilde{w}^T \Gamma) X$$
$$w_{lin}^T = (\tilde{w}^T \Gamma)$$
$$w_{lin} = \Gamma^T \tilde{w}$$

答案為 [b].

10

依題目所寫，

$$\Phi(x_1) = (1, 0, 0, ..., 0)$$
$$\Phi(x_2) = (0, 1, 0, ..., 0)$$
$$\vdots$$
$$\Phi(x_N) = (0, 0, 0, ..., 1)$$

為了使$w^T X = y$

$$X = \begin{bmatrix} | & | & & | \\ \Phi(x_1) & \Phi(x_2) & \cdots & \Phi(x_N) \\ | & | & & | \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} = I_N$$
$$y = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}$$
$$\tilde{w} = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix} = y$$

此時的 $E_{in}^{sqr}(\tilde{w}) = \sum_{n=1}^{N}(y_i - \tilde{w}^T x_i)^2 = (y_i - y_i)^2 = 0$
答案為 [c].

11

可觀察出 $\sum_{k=1}^{K} E_{in}^{0/1}(w_{[k]}^*) \geq E_{in}^{0/1}(g)$，因為用one-versus-all做分類後可能剛好存在為一一個錯誤分類被送到 g。
可觀察出 $\forall k, 1 \leq k \leq K, E_{in}^{sqr}(w_{[k]}^*) \geq E_{in}^{0/1}(w_{[k]}^*)$，因為分類錯誤時 $E_{in}^{sqr}(w_{[k]}^*) \geq 1$。 結合以上兩點得

$$E_{in}^{0/1}(g) \leq \sum_{k=1}^{K} E_{in}^{0/1}(w_{[k]}^*) \leq \sum_{k=1}^{K} E_{in}^{sqr}(w_{[k]}^*) = \sum_{k=1}^{K} e_k$$

答案為 [b].

12, 13, 14, 15, 16

Code:

```
import numpy as np
import random
import time
import math
import tqdm

#==============================Basic Function==============================

def sign(arr):
    for i in range(arr.shape[0]):
        if arr[i] < 0:
            arr[i] = -1
        else:
            arr[i] = 1

    return arr

def E_01(x, y, w):
    y_hat = sign(np.dot(x, w))
    return np.sum(y != y_hat) / y.shape[0]

#==============================Linear Regression==========================

def LinearRegression(x, y):
    x_pi = np.linalg.pinv(x)
    w_lin = np.dot(x_pi, y)
    return w_lin

#==============================Transform==================================

def homogeneousOrderTransform(data, Q):
    newData = np.zeros((data.shape[0], data.shape[1] * Q))
    for q in range(Q):
        for i in range(data.shape[0]):
            for j in range(data.shape[1]):
                newData[i][q * data.shape[1] + j] = data[i][j] ** (q + 1)

    newData = np.hstack((np.ones((newData.shape[0], 1)), newData))

    return newData

def fullOrderTransform(data):
    newData = np.zeros((data.shape[0], 10 + 45 + 10))
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            newData[i][j] = data[i][j]

    pos = data.shape[1]
    for k in range(data.shape[1]):
        for l in range(k, data.shape[1]):
            for i in range(data.shape[0]):
                newData[i][pos] = data[i][k] * data[i][l]
            pos += 1

    newData = np.hstack((np.ones((newData.shape[0], 1)), newData))

    return newData
```

```
def lowerDimensionTransform(data, Q):
    newData = np.zeros((data.shape[0], Q))
    for i in range(data.shape[0]):
        for j in range(Q):
            newData[i][j] = data[i][j]

    newData = np.hstack((np.ones((newData.shape[0], 1)), newData))
    return newData

def randomDimensionTransform(data, sel):
    newData = np.zeros((data.shape[0], len(sel)))
    for i in range(data.shape[0]):
        for j in range(len(sel)):
            newData[i][j] = data[i][sel[j]]

    newData = np.hstack((np.ones((newData.shape[0], 1)), newData))
    return newData

#===========================================================================

def main():
    trainData = np.loadtxt('hw3_train.dat')
    testData = np.loadtxt('hw3_test.dat')
    x_train, y_train = np.hsplit(trainData, [-1])
    x_test, y_test = np.hsplit(testData, [-1])s
    p12, p13, p14, p15, p16 = True, True, True, True, True

    if p12:
        x_train_HOT = homogeneousOrderTransform(x_train, 2)
        x_test_HOT = homogeneousOrderTransform(x_test, 2)
        w = LinearRegression(x_train_HOT, y_train)
        print("Problem 12:", abs(E_01(x_train_HOT, y_train, w)
                                 - E_01(x_test_HOT, y_test, w)))

    if p13:
        x_train_HOT = homogeneousOrderTransform(x_train, 8)
        x_test_HOT = homogeneousOrderTransform(x_test, 8)
        w = LinearRegression(x_train_HOT, y_train)
        print("Problem 13:", abs(E_01(x_train_HOT, y_train, w)
                                 - E_01(x_test_HOT, y_test, w)))

    if p14:
        x_train_FOT = fullOrderTransform(x_train)
        x_test_FOT = fullOrderTransform(x_test)
        w = LinearRegression(x_train_FOT, y_train)
        print("Problem 14:", abs(E_01(x_train_FOT, y_train, w)
                                 - E_01(x_test_FOT, y_test, w)))

    if p15:
        minErr, minDim = 1000, 0
        for i in range(x_train.shape[1]):
            x_train_LDT = lowerDimensionTransform(x_train, i+1)
            x_test_LDT = lowerDimensionTransform(x_test, i+1)
            w = LinearRegression(x_train_LDT, y_train)
            err = abs(E_01(x_train_LDT, y_train, w)
                      - E_01(x_test_LDT, y_test, w))
            if err < minErr:
                minErr = err
                minDim = i + 1

        print("Problem 15:", minDim)
```

```python
    if p16:
        t = 200
        errSum = 0
        for i in tqdm.trange(t):
            random.seed(time.time())
            sel = random.sample(range(x_train.shape[1]), 5)
            x_train_RDT = randomDimensionTransform(x_train, sel)
            x_test_RDT = randomDimensionTransform(x_test, sel)
            w = LinearRegression(x_train_RDT, y_train)
            errSum += abs(E_01(x_train_RDT, y_train, w)
                        - E_01(x_test_RDT, y_test, w))

        print("Problem_16:", errSum / t)

if __name__ == '__main__':
    main()
```

答案依序為 [b], [d], [a], [c], [d].