Linear Regression

1. ANS [d]

Consider a noisy target $y = \mathbf{w}_f^T \mathbf{x} + \epsilon$, where $\mathbf{x} \in \mathbb{R}^{d+1}$ (including the added coordinate $x_0 = 1$), $y \in \mathbb{R}$, $\mathbf{w}_f \in \mathbb{R}^{d+1}$ is an unknown vector, and ϵ is an i.i.d. noise term with zero mean and σ^2 variance. Assume that we run linear regression on a training data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ generated i.i.d. from some $P(\mathbf{x})$ and the noise process above, and obtain the weight vector $\mathbf{w}_{\text{lin}}$. As briefly discussed in Lecture 5, it can be shown that the expected in-sample error $E_{\text{in}}(\mathbf{w}_{\text{lin}})$ with respect to $\mathcal{D}$ is given by:

$$\mathbb{E}_{\mathcal{D}}\left[E_{\text{in}}(\mathbf{w}_{\text{lin}})\right] = \sigma^2\left(1 - \frac{d+1}{N}\right).$$

For $\sigma = 0.1$ and $d = 19$, what is the smallest number of examples N such that $\mathbb{E}_{\mathcal{D}}\left[E_{\text{in}}(\mathbf{w}_{\text{lin}})\right]$ is no less than 0.005? Choose the correct answer; explain your answer.

[a] 25
[b] 30
[c] 35
[d] 40
[e] 45

$$\mathbb{E}_D[E_{in}(w_{lin})] = \sigma^2\left(1 - \frac{d+1}{N}\right)$$

$$\sigma = 0.1 \cdot d = 19$$

$$\Rightarrow \quad \sigma^2\left(1 - \frac{d+1}{N}\right) \leq 0.005$$

$$0.01\left(1 - \frac{20}{N}\right) \leq 0.005$$

$$0.01 - \frac{0.2}{N} \leq 0.005$$

$$-0.2 \leq -0.005N$$

$$N \geq \frac{200}{5} = 40$$

※得證

smallest number of $N \geq 40$

選 d ※

2. ANS [c]

Consider the target function $f(x) = x^2$. Sample x uniformly from $[0, 1]$, and use all linear hypotheses $h(x) = w_0 + w_1 \cdot x$ to approximate the target function with respect to the squared error. What are the weights (w_0^*, w_1^*) of the optimal hypothesis? Choose the correct answer; explain your answer.

[a] $(0, 1)$
[b] $(\frac{1}{2}, \frac{1}{2})$
[c] $(-\frac{1}{6}, 1)$
[d] $(-\frac{1}{4}, \frac{1}{4})$
[e] $(\frac{1}{3}, 0)$

(*Hint: The optimal hypothesis* g^* *must reach the minimum* $E_{\text{out}}(g^*)$.)

$$f(x) = x^2$$

$$h(x) = w_0 + w_1 x$$

$$E_{out}(w) = \mathop{\mathcal{E}}_{(x,y)\sim P} \left(h(x) - f(x) \right)^2$$

$$= \int_0^1 (w_0 + w_1 x - x^2)^2 \, dx$$

$$\int (w_0 + w_1 x - x^2)^2 \, dx$$

$$= \int (x^4 - w_1 x^3 - w_0 x^2 - w_1 x^3 + w_1^2 x^2 + w_0 w_1 x - w_0 x^2 + w_0 w_1 x + w_0^2) \, dx$$

$$= \int (x^4 - 2w_1 x^3 + (w_1^2 - 2w_0) x^2 + 2w_0 w_1 x + w_0^2) \, dx$$

$$= \frac{1}{5} x^5 - \frac{2w_1}{4} x^4 + \frac{(w_1^2 - 2w_0)}{3} x^3 + \frac{2w_0 w_1}{2} x^2 + w_0^2 x$$

令為面積函數 $F(x)$

$$\int_0^1 (w_0 + w_1 x - x^2)^2 dx = F(1) - F(0) = F(x)\Big|_0^1$$

$$= \frac{1}{5} - \frac{w_1}{2} + \frac{(w_1^2 + 2w_0)}{3} + w_0 w_1 + w_0^2$$

$$= \frac{1}{3}w_1^2 - \frac{1}{2}w_1 - \frac{2}{3}w_0 + w_0 w_1 + w_0^2 + \frac{1}{5}$$

對 w_0 微分

$$\Rightarrow -\frac{2}{3} + w_1 + 2w_0 = 0$$

對 w_1 微分

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

$$\Rightarrow \begin{cases} -\frac{2}{3} + w_1 + 2w_0 = 0 \\ \frac{2}{3}w_1 - \frac{1}{2} + w_0 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} -\frac{2}{3} + w_1 + 2w_0 = 0 \\ -1 + \frac{4}{3}w_1 + 2w_0 = 0 \end{cases}$$

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

得最佳 $(w_0^*, w_1^*) = (-\frac{1}{6}, 1)$

選 C

3. ANS [e]

Following the previous problem, assume that we sample two examples x_1 and x_2 uniformly from $[0, 1]$ to form the training set $\mathcal{D} = \{(x_1, f(x_1)), (x_2, f(x_2)\}$, and use linear regression to get g for approximating the target function with respect to the squared error. You can neglect the degenerate cases where x_1 and x_2 are the same. What is $\mathbb{E}_{\mathcal{D}}(|E_{\text{in}}(g) - E_{\text{out}}(g)|)$? Choose the correct answer; explain your answer.

[a] $\frac{1}{60}$
[b] $\frac{4}{15}$
[c] $\frac{3}{20}$
[d] $\frac{1}{25}$
[e] $\frac{1}{30}$

求 $E_D(|E_{in}(g) - E_{out}(g)|)$

由於D只有2個，故 $E_{in}(g) = 0$

剩下求 $E_D(E_{out}(g))$

由第二題得出

$$g(x) = \int_0^1 (w_0 + w_1 x - x^2)^2 dx$$

$$= \frac{1}{3}w_1^2 - \frac{1}{2}w_1 - \frac{2}{3}w_0 + w_0 w_1 + w_0^2 + \frac{1}{5}$$

用 x_1, x_2 代換 w_0, w_1

由於 $D = \{(x_1, f(x_1)), (x_2, f(x_2))\}$

且第二題說 $f(x) = x^2$, $h(x) = w_0 + w_1 \cdot x$

$$\Rightarrow \begin{cases} w_0 + w_1 x_1 = x_1^2 \\ w_0 + w_1 x_2 = x_2^2 \end{cases}$$

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

$$w_1 = \frac{x_1^2 - x_2^2}{x_1 - x_2} = x_1 + x_2$$

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

$$= -x_1 x_2$$

代入 $g(x) = \frac{1}{3}w_1^2 - \frac{1}{2}w_1 - \frac{2}{3}w_0 + w_0w_1 + w_0^2 + \frac{1}{5}$

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

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

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

$$= \frac{1}{3}x_1^2 + \frac{1}{3}x_2^2 - \frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{4}{3}x_1x_2 - x_1^2x_2 - x_1x_2^2 + x_1^2x_2^2 + \frac{1}{5}$$

$E_D(E_{out}(g))$

$$= \int_0^1\int_0^1 \left[\frac{1}{3}x_1^2 + \frac{1}{3}x_2^2 - \frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{4}{3}x_1x_2 - x_1^2x_2 - x_1x_2^2 + x_1^2x_2^2 + \frac{1}{5}\right]dx_1dx_2$$

$$= \int_0^1 \left[\frac{1}{9}x_1^3 + \frac{1}{3}x_1x_2^2 - \frac{1}{4}x_1^2 - \frac{1}{2}x_1x_2 + \frac{2}{3}x_1^2x_2 - \frac{1}{3}x_1^3x_2 - \frac{1}{2}x_1^2x_2^2 + \frac{1}{3}x_1^3x_2^2 + \frac{1}{5}x_1\Big|_0^1\right]dx_2$$

$$= \int_0^1 \left[\frac{1}{9} + \frac{1}{3}x_2^2 - \frac{1}{4} - \frac{1}{2}x_2 + \frac{2}{3}x_2 - \frac{1}{3}x_2 - \frac{1}{2}x_2^2 + \frac{1}{3}x_2^2 + \frac{1}{5}\right]dx_2$$

$$= \left[\frac{1}{9}x_2 + \frac{1}{9}x_2^3 - \frac{1}{4}x_2 - \frac{1}{4}x_2^2 + \frac{1}{3}x_2^2 - \frac{1}{6}x_2^2 - \frac{1}{6}x_2^3 + \frac{1}{9}x_2^3 + \frac{1}{5}x_2\right]\Big|_0^1$$

$$= \frac{1}{9} + \frac{1}{9} - \frac{1}{4} - \frac{1}{4} + \frac{1}{3} - \frac{1}{6} - \frac{1}{6} + \frac{1}{9} + \frac{1}{5}$$

$$= \frac{3}{9} - \frac{2}{4} + \frac{1}{3} - \frac{2}{6} + \frac{1}{5} = \frac{1}{3} - \frac{1}{2} + \cancel{\frac{1}{3}} - \cancel{\frac{1}{3}} + \frac{1}{5} = \frac{10-15+6}{30} = \frac{1}{30}$$

$\Rightarrow E_D(E_{out}(g)) = \frac{1}{30}$ ※得說選 e

Cross-Entropy Error

4. ANS [b]

In class, we introduced our version of the cross-entropy error function

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} -\ln\theta(y_n \mathbf{w}^T \mathbf{x}_n).$$

based on the definition of $y_n \in \{-1, +1\}$. If we transform y_n to $y_n' \in \{0, 1\}$ by $y_n' = \frac{y_n+1}{2}$, which of the following error function is equivalent to E_{in} above? Choose the correct answer; explain your answer.

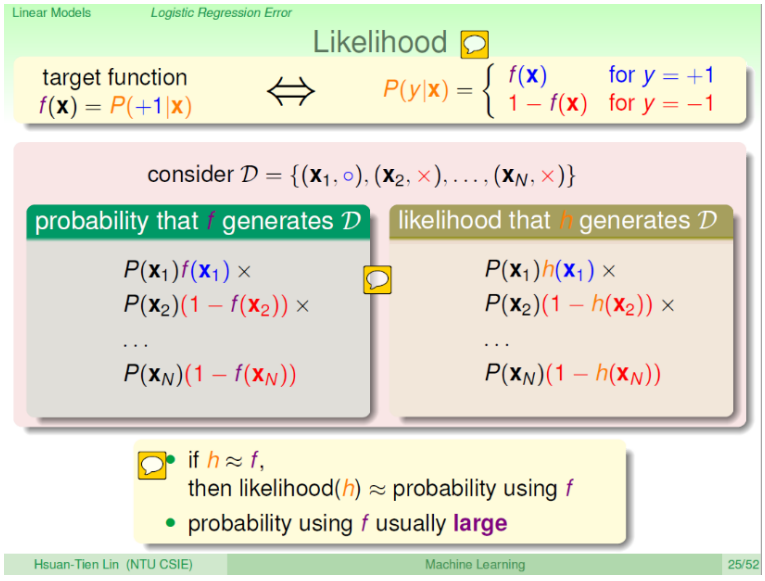
[a] $\frac{1}{N}\sum_{n=1}^{N}\left(y_n' \ln\theta(\mathbf{w}^T\mathbf{x}_n) + (1-y_n')\ln(\theta(-\mathbf{w}^T\mathbf{x}_n))\right)$

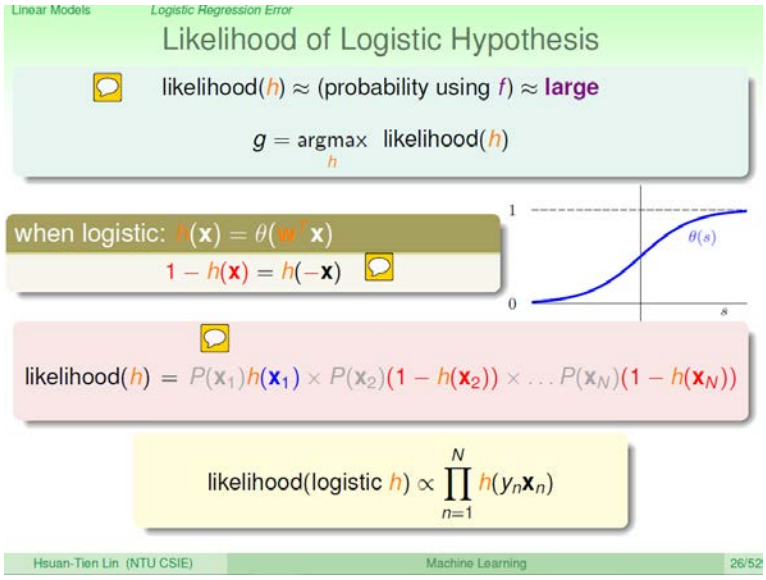
[b] $\frac{1}{N}\sum_{n=1}^{N}\left(y_n' \ln\theta(-\mathbf{w}^T\mathbf{x}_n) + (1-y_n')\ln(\theta(\mathbf{w}^T\mathbf{x}_n))\right)$

[c] $\frac{1}{N}\sum_{n=1}^{N}\left(y_n' \ln\theta(\mathbf{w}^T\mathbf{x}_n) - (1-y_n')\ln(\theta(-\mathbf{w}^T\mathbf{x}_n))\right)$

[d] $\frac{1}{N}\sum_{n=1}^{N}\left(y_n' \ln\theta(-\mathbf{w}^T\mathbf{x}_n) - (1-y_n')\ln(\theta(\mathbf{w}^T\mathbf{x}_n))\right)$

[e] none of the other choices





根據投影片與題意，

可假設 $f(x) = P(y_n = 1 \mid x)$

$= P(y_n' = 1 \mid x)$ 為 target function

且此題只有 2 種情況：$y_n' = 1$ or 0

$\Rightarrow P(y_n' \mid x) = \begin{cases} f(x) & \text{for } y_n' = 1 \\ 1 - f(x) & \text{for } y_n' = 0 \end{cases}$

$= f(x)^{y_n'} \left(1 - f(x)\right)^{1-y_n'}$

if $h \approx f$, then likelihood$(h) \approx$ probability using f

且 probability using f 為 large

所以 likelihood(h) 也大

$\Rightarrow$ likelihood$(h) \approx$ (probability using f) $\approx$ large

$g = \underset{h}{\text{argmax}}$ likelihood(h)

假設 likelihood$(h) = L(h)$

且取 log 時 $h(x) = \theta(w^T x)$，$1 - \theta(w^T x) = \theta(-w^T x)$

$$L(\log h) = P(x_1) h(x_1) \times P(x_2) h(x_2) \times \cdots \times P(x_N)(1 - h(x_N))$$

$$\Rightarrow \max_h L(\log h) \propto \prod_{n=1}^{N} h(x)^{y_n'} (1 - h(x))^{1-y_n'}$$

$$\max_w L(w) \propto \prod_{n=1}^{N} \theta(w^T x)^{y_n'} (1 - \theta(w^T x))^{1-y_n'}$$

$$\propto \prod_{n=1}^{N} \theta(w^T x)^{y_n'} \theta(-w^T x)^{1-y_n'}$$

由於 $y_n' = 0$ or 1

$$\propto \prod_{n=1}^{N} \left(\theta(z y_n' - 1) w^T x\right)$$

⇨ 求 $\max_w L(w)$ 相當於求 $\max_w \prod_{n=1}^{N} (\theta(zy_n'-1)w^Tx)$

且相當於求 $\max_w \prod_{n=1}^{N} \ln(\theta(zy_n'-1)w^Tx)$ 取ln大小關係不變

$$\max_w \prod_{n=1}^{N} \ln(\theta(zy_n'-1)w^Tx) = \min_w \frac{1}{N}\sum_{n=1}^{N} -\ln(\theta(zy_n'-1)w^Tx)$$

由於 $\theta(-s) = 1-\theta(s)$

$\min -\ln\theta(-s) \Leftrightarrow \min -\ln(1-\theta(s))$

$\Leftrightarrow \max \ln(1-\theta(s))$

$\Leftrightarrow \min \ln(\theta(s)-1)$

$\Leftrightarrow \min \ln\theta(s)$

⇨ $\min_w \frac{1}{N}\sum_{n=1}^{N} -\ln(\theta(zy_n'-1)w^Tx) \Leftrightarrow \min_w \frac{1}{N}\sum_{n=1}^{N} \ln(\theta(1-zy_n')w^Tx)$

$$\min_w \frac{1}{N}\sum_{n=1}^{N} \ln(\theta(1-zy_n')w^Tx) = \min_w \frac{1}{N}\sum_{n=1}^{N} \ln(\theta(1-y_n'-y_n')w^Tx)$$

$$= \min_w \frac{1}{N}\sum_{n=1}^{N} \ln\left[\theta\left[(1-y_n')w^Tx - y_n'w^Tx\right]\right]$$

假設為 $E_{in\_new}(w)$

$$E_{in\_new}(w) = \frac{1}{N}\sum_{n=1}^{1} \ln\left[\theta\left[(1-y_n')w^Tx - y_n'w^Tx\right]\right]$$

$y_n' = 0$, $E_{in\_new}(w) = \frac{1}{N}\sum_{n=1}^{1} \ln\theta(w^Tx)$

$y_n' = 1$, $E_{in\_new}(w) = \frac{1}{N}\sum_{n=1}^{1} \ln\theta(-w^Tx)$

⇨ $E_{in}'(w) = \frac{1}{N}\sum_{n=1}^{N} (y_n'\ln\theta(-w^Tx) + (1-y_n')\ln\theta(w^Tx))$

得證，選b ✳

5. ANS [e]

Consider a coin with an unknown head probability μ. Independently flip this coin N times to get $y_1, y_2, \ldots, y_N$, where $y_n = 1$ if the n-th flipping results in head, and 0 otherwise. Define $\nu = \frac{1}{N}\sum_{n=1}^{N} y_n$. How many of the following statements about ν are true? Choose the correct answer; explain your answer by briefly illustrating why those statements are true.

(Note: μ is similar to the role of the "target function" and $\hat{\mu}$ is similar to the role of the "hypothesis" in our machine learning framework.)

[a] 0
[b] 1
[c] 2
[d] 3
[e] 4

全部都對，選 e。

- With probability more than $1-\delta$,

$$\mu \leq \nu + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

for all $N \in \mathbb{N}$ and $0 < \delta < 1$.

- in big sample (N large), ν is probably close to μ (within ϵ)

$$\mathbb{P}\left[|\nu-\mu| > \epsilon\right] \leq 2\exp\left(-2\epsilon^2 N\right)$$

- called Hoeffding's Inequality, for marbles, coin, polling, ...

由 Hoeffding's Inequality

$\Rightarrow \underbrace{\mathbb{P}[|\nu-\mu|>\epsilon]}_{\text{BAD}} \leq \underbrace{2\exp(-2\epsilon^2 N)}_{\delta}$

With probability $\geq 1-\delta$ (GOOD: $|\nu-\mu| \leq \epsilon$)

令 $\delta = 2\exp(-2\epsilon^2 N)$

$$\frac{\delta}{2} = \exp(-2\epsilon^2 N)$$

取 ln

$$\ln\left(\frac{\delta}{2}\right) = -2\epsilon^2 N$$

$$\epsilon^2 = -\frac{1}{2N}\ln\left(\frac{\delta}{2}\right)$$

$$= \frac{1}{2N}\ln\left(\frac{2}{\delta}\right)$$

$$\epsilon = \sqrt{\frac{1}{2N}\ln\left(\frac{2}{\delta}\right)}$$

$\Rightarrow |\nu-\mu| \leq \epsilon = \sqrt{\frac{1}{2N}\ln\left(\frac{2}{\delta}\right)}$

$$\nu - \sqrt{\frac{1}{2N}\ln\left(\frac{2}{\delta}\right)} \leq \mu \leq \nu + \sqrt{\frac{1}{2N}\ln\left(\frac{2}{\delta}\right)}$$

得證

- ν maximizes likelihood($\hat{\mu}$) over all $\hat{\mu} \in [0, 1]$.

Coin Flip MLE 為伯努利分佈，$y_n = 0$ or 1

likelihood$(\hat{\mu}) = L_{y_n}(\hat{\mu}) = P(y_n | \hat{\mu}) = \hat{\mu}^{y_n}(1-\hat{\mu})^{1-y_n}$

由於拋硬幣為 i.i.d.，所以 likelihood = 所有可能性連乘

likelihood$(\hat{\mu}) = L_{y_n}(\hat{\mu}) = \prod_{y_n \in y} \hat{\mu}^{y_n}(1-\hat{\mu})^{1-y_n}$

EX: y = h, t, t, t, h, h (head, tail)
1 0 0 0 1 1

代入 $L_{y_n}(\hat{\mu}) = \hat{\mu} \cdot (1-\hat{\mu}) \cdot (1-\hat{\mu}) \cdot (1-\hat{\mu}) \cdot \hat{\mu} \cdot \hat{\mu}$

$\Rightarrow L_{y_n}(\hat{\mu}) = \hat{\mu}^h \cdot (1-\hat{\mu})^{n-h}$ (n 為丟的總數，h 為 head 次數)

想找 maximizes of $L_{y_n}(\hat{\mu})$

取 log 再微分 ($\because \hat{\mu}$ that maximize $L_{y_n}(\hat{\mu})$ 等價於 $\hat{\mu}$ that maximize $\log\text{-}L_{y_n}(\hat{\mu})$)

$\log\text{-}L_{y_n}(\hat{\mu}) = h \cdot \log(\hat{\mu}) + (n-h) \cdot \log(1-\hat{\mu})$

對 $\hat{\mu}$ 偏微分

$\Rightarrow \log\text{-}L_{y_n}(\hat{\mu})' = \frac{h}{\hat{\mu}} - \frac{n-h}{1-\hat{\mu}} = 0$

$\Rightarrow \hat{\mu} = \frac{h}{n}$，而題目中 $\nu = \frac{1}{N}\sum_{n=1}^{N} y_n$ 也就是 $\frac{\text{head 次數}}{\text{總次數}}$

$\Rightarrow \nu = \frac{h}{n} = \hat{\mu}$

故 ν 可以 maximizes likelihood$(\hat{\mu})$ #

- ν minimizes the squared error

$$E^{\mathrm{sqr}}(\hat{y}) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y} - y_n)^2$$

over all $\hat{y} \in \mathbb{R}$.

要 minimizes the squared error

⇨ 微分

$$E^{sqr}(\hat{y}) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}-y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}^2 - 2\hat{y}y_n + y_n^2)$$

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

$$\Rightarrow \frac{1}{N}\sum_{n=1}^{N}(\hat{y}-y_n) = 0$$

將 v 代入

$$\nabla E^{sqr}(v) = \frac{1}{N}\sum_{n=1}^{N}(v - y_n) = 0$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{N}\sum_{n=1}^{N}y_n - y_n\right) = 0$$

$$\Rightarrow \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{N}\sum_{n=1}^{N}y_n\right) - \frac{1}{N}\sum_{n=1}^{N}y_n = 0$$

$$\frac{1}{N}\sum_{n=1}^{N}y_n - \frac{1}{N}\sum_{n=1}^{N}y_n = 0$$

成立，得證

v 可以最小化 squared error

- When $0 < \nu < 1$, it minimizes the cross-entropy error (which is similar to the cross-entropy error for logistic regression)

$$E^{\text{ce}}(\hat{y}) = -\frac{1}{N}\sum_{n=1}^{N}\Big(y_n \ln \hat{y} + (1-y_n)\ln(1-\hat{y})\Big)$$

over all $\hat{y} \in (0,1)$.

minimizes the cross-entropy error

① 微分　　$\hat{y} \in (0,1)$

$$E^{ce}(\hat{y}) = -\frac{1}{N}\sum_{n=1}^{N}\left(y_n \ln \hat{y} + (1-y_n)\ln(1-\hat{y})\right)$$

$$\frac{\partial E^{ce}(\hat{y})}{\partial \hat{y}} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \cdot \frac{\partial \ln(\square)}{\partial \square}\cdot\frac{\partial \hat{y}}{\hat{y}} + (1-y_n)\cdot\frac{\partial \ln(\blacksquare)}{\partial \blacksquare}\cdot\frac{\partial(1-\hat{y})}{\hat{y}}\right]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \cdot \frac{1}{\hat{y}}\cdot 1 + (1-y_n)\cdot\frac{1}{1-\hat{y}}\cdot(-1)\right]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[\frac{y_n}{\hat{y}} - \frac{1-y_n}{1-\hat{y}}\right]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n}{\hat{y}} + \frac{1}{N}\sum_{n=1}^{N}\frac{1-y_n}{1-\hat{y}}$$

$$-\frac{1}{N}\sum_{n=1}^{N}\frac{y_n}{\hat{y}} + \frac{1}{N}\sum_{n=1}^{N}\frac{1-y_n}{1-\hat{y}} = 0$$

[$\hat{y}$ 為一常數]

$$-\frac{1}{N}\cdot\frac{1}{\hat{y}}\sum_{n=1}^{N}y_n + \frac{1}{N}\cdot\frac{1}{1-\hat{y}}\sum_{n=1}^{N}(1-y_n) = 0$$

$$-\frac{1}{\hat{y}}\sum_{n=1}^{N}y_n + \frac{1}{1-\hat{y}}\sum_{n=1}^{N}(1-y_n) = 0$$

$$-(1-\hat{y})\sum_{n=1}^{N}y_n + \hat{y}\sum_{n=1}^{N}(1-y_n) = 0$$

$$-\sum_{n=1}^{N}y_n + \hat{y}\sum_{n=1}^{N}y_n + \hat{y}\sum_{n=1}^{N}1 - \hat{y}\sum_{n=1}^{N}y_n = 0$$

$$-\sum_{n=1}^{N}y_n + \hat{y}N = 0$$

$$\hat{y} = \frac{1}{N}\sum_{n=1}^{N}y_n = \nu$$

成立，得證

ν 可最小化 cross-entropy error ※

Stochastic Gradient Descent

6. ANS [a]

In the perceptron learning algorithm, we find one example $(\mathbf{x}_{n(t)}, y_{n(t)})$ that the current weight vector $\mathbf{w}_t$ mis-classifies, and then update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}.$$

The algorithm can be viewed as optimizing some $E_{\text{in}}(\mathbf{w})$ that is composed of one of the following point-wise error functions with stochastic gradient descent (neglecting any non-differentiable points of the error function). What is the error function? Choose the correct answer; explain your answer.

[a] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(0, -y\mathbf{w}^T\mathbf{x})$

[b] $\text{err}(\mathbf{w}, \mathbf{x}, y) = -\max(0, -y\mathbf{w}^T\mathbf{x})$

[c] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(y\mathbf{w}^T\mathbf{x}, -y\mathbf{w}^T\mathbf{x})$

[d] $\text{err}(\mathbf{w}, \mathbf{x}, y) = -\max(y\mathbf{w}^T\mathbf{x}, -y\mathbf{w}^T\mathbf{x})$

[e] none of the other choices

$W_{t+1} \leftarrow W_t + y_{n(t)} X_{n(t)}$

$-\nabla err(w, x, y)$

[a] $err(w, x, y)$ 有2種情況

① $err(w, x, y) = 0$

$\nabla err(w, x, y) = 0$

$\Rightarrow W_{t+1} \leftarrow W_t + 0$ 代表沒有錯不需更新的情形

② $err(w, x, y) = -yw^Tx$

$\nabla err(w, x, y) = \frac{\partial -yw^Tx}{\partial w} = -yx$

$\Rightarrow W_{t+1} \leftarrow W_t - (-yx)$

$\Rightarrow W_{t+1} \leftarrow W_t + yx$ 與題幹的update function 相同，得證

Multinomial Logistic Regression

7. ANS [a]

In Lecture 6, we solve multiclass classification by OVA or OVO decompositions. One alternative to deal with multiclass classification is to extend the original logistic regression model to Multinomial Logistic Regression (MLR). For a K-class classification problem, we will denote the output space $\mathcal{Y} = \{1, 2, \cdots, K\}$. The hypotheses considered by MLR can be indexed by a matrix

$$\mathrm{W} = \begin{bmatrix} | & | & \cdots & | & \cdots & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_k & \cdots & \mathbf{w}_K \\ | & | & \cdots & | & \cdots & | \end{bmatrix}_{(d+1)\times K},$$

that contains weight vectors $(\mathbf{w}_1, \cdots, \mathbf{w}_K)$, each of length $d+1$. The matrix represents a hypothesis

$$h_y(\mathbf{x}) = \frac{\exp(\mathbf{w}_y^T \mathbf{x})}{\sum_{i=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})}$$

that can be used to approximate the target distribution $P(y|\mathbf{x})$ for any $(\mathbf{x}, y)$. MLR then seeks for the maximum likelihood solution over all such hypotheses. For a given data set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ generated i.i.d. from some $P(\mathbf{x})$ and target distribution $P(y|\mathbf{x})$, the likelihood of $h_y(\mathbf{x})$ is proportional to $\prod_{n=1}^{N} h_{y_n}(\mathbf{x}_n)$. That is, minimizing the negative log likelihood is equivalent to minimizing an $E_{\text{in}}(\mathrm{W})$ that is composed of the following error function

$$\mathrm{err}(\mathrm{W}, \mathbf{x}, y) = -\ln h_y(\mathbf{x}) = -\sum_{k=1}^{K} [\![y = k]\!] \ln h_k(\mathbf{x}).$$

When minimizing $E_{\text{in}}(\mathrm{W})$ with SGD, we update the $\mathrm{W}^{(t)}$ at the t-th iteration to $\mathrm{W}^{(t+1)}$ by

$$\mathrm{W}^{(t+1)} \leftarrow \mathrm{W}^{(t)} + \eta \cdot \mathrm{V},$$

where V is a $(d+1) \times K$ matrix whose k-th column is an update direction for the k-th weight vector. Assume that an example $(\mathbf{x}_n, y_n)$ is used for the SGD update above. What is the y_n-th column of V? Choose the correct answer; explain your answer.

[a] $(1 - h_{y_n}(\mathbf{x}_n))\mathbf{x}_n$

[b] $(h_{y_n}(\mathbf{x}_n) - 1)\mathbf{x}_n$

[c] $(-h_{y_n}(\mathbf{x}_n))\mathbf{x}_n$

[d] $(h_{y_n}(\mathbf{x}_n))\mathbf{x}_n$

[e] none of the other choices

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \cdot \underbrace{V}_{-\nabla err}$$

minimizing $E_{in}(w)$

$$h_y(x) = \frac{\exp(w_y^T x)}{\sum_{i=1}^{k} \exp(w_i^T x)}$$

$$err(w, x, y) = -\ln h_y(x) = -\sum_{k=1}^{K} [\![y = k]\!] \ln h_k(x)$$

$$W^{(t+1)} = \begin{bmatrix} | & | & | & | \\ w_1^{(t+1)} & w_2^{(t+1)} & \cdots & w_K^{(t+1)} \\ | & | & | & | \end{bmatrix}$$

$$W^{(t)} = \begin{bmatrix} | & | & | \\ w_1^{(t)} & \cdots & w_K^{(t)} \\ | & | & | \end{bmatrix}$$

$$V = \begin{bmatrix} | & | & | \\ v_1 & \cdots & v_K \\ | & | & | \end{bmatrix}$$

共有 K 個 classifier 要去跑 $w^{(t+1)} \leftarrow w^{(t)} + \eta \cdot V$

focus 在第 k 個 classifier 要跑的 update rule 上

$$\Rightarrow w_k^{(t+1)} \leftarrow w_k^{(t)} + \eta \cdot \underbrace{v_k}$$

$$v_k = -\nabla err(w, x, k) \quad (y \text{ 用 } k \text{ 代})$$

$$-\nabla err(w, x, k) = -\frac{\partial err(w, x, k)}{\partial w_k} = -\frac{\partial -\ln h_k(x)}{\partial w_k}$$

$$\left[h_k(x) = \frac{\exp(w_k^T x)}{\sum_{i=1}^{K} \exp(w_i^T x)} \right]$$

$$= -\nabla \left(-\ln \left(\frac{\exp(w_k^T x)}{\sum_{i=1}^{K} \exp(w_i^T x)} \right) \right)$$

$$= \nabla \left(\ln \exp(w_k^T x) - \ln \sum_{i=1}^{K} \exp(w_i^T x) \right)$$

$$= \frac{\partial \ln(\square)}{\partial \square} \cdot \frac{\partial \exp(\circ)}{\partial \circ} \cdot \frac{\partial w_k^T x}{\partial w_k}$$

$$- \frac{\partial \ln(\blacksquare)}{\partial \blacksquare} \cdot \frac{\partial \sum_{i=1}^{K} \exp(w_i^T x)}{\partial w_k^T x} \cdot \frac{\partial w_k^T x}{\partial w_k}$$

$$= \frac{1}{\square} \cdot \exp(0) \cdot X$$

$$- \frac{1}{\square} \cdot \frac{\partial\left(\exp(w_1^T x) + \cdots + \exp(w_k^T x) + \cdots + \exp(w_K^T x)\right)}{\partial w_k^T x} \cdot X$$

$$= \frac{\exp(w_k^T x)}{\exp(w_k^T x)} \cdot X - \frac{1}{\square} \cdot \exp(w_k^T x) \cdot X$$

$$= X - \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x)} \cdot X = X - h_k(x) \cdot X$$

$$= (1 - h_k(x)) X$$

$k = y_n$ ⇨ $v = (1 - h_{y_n}(x_n)) x_n$ #

得證. 選 a.

Nonlinear Transformation

8. ANS [e]

Given the following training data set:

$$\mathbf{x}_1 = (0,1), y_1 = -1 \quad \mathbf{x}_2 = (0,-1), y_2 = -1 \quad \mathbf{x}_3 = (-1,0), y_3 = +1 \quad \mathbf{x}_4 = (1,0), y_4 = +1$$

Use the quadratic transform $\mathbf{\Phi}_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ and take sign(0) = 1. Which of the following weight vector $\tilde{\mathbf{w}}$ represents a linear classifier in the $\mathcal{Z}$-space that can separate all the transformed examples perfectly? Choose the correct answer; explain your answer.

[a] $(0,-1,0,0,0,0)$
[b] $(0,0,-1,0,0,0)$
[c] $(0,0,0,-1,0,0)$
[d] $(0,0,0,0,-1,0)$
[e] $(0,0,0,0,0,-1)$

$\Phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

題目為對應的 $\tilde{w}$

[a] $(0,-1,0,0,0,0)$

$\Rightarrow h(x) = -x_1$

$x_1 = (0,1), y_1 = -1$

$h(x_1) = \text{sign}(0) = 1 \neq y_1$ 錯

[b] $(0,0,-1,0,0,0)$

$\Rightarrow h(x) = -x_2$

$x_1 = (0,1), y_1 = -1 \;:\; h(x_1) = -1 = y_1$

$x_2 = (0,-1), y_2 = -1 \;:\; h(x_2) = 1 \neq y_2 = -1$ 錯

[c] $(0,0,0,-1,0,0)$

$\Rightarrow h(x) = -x_1^2$

$x_1 = (0,1), y_1 = -1 \;:\; h(x_1) = \text{sign}(0) = 1 \neq y_1$ 錯

[d] $(0,0,0,0,-1,0)$

$\Rightarrow h(x) = -x_1x_2$

$x_1 = (0,1), y_1 = -1 : h(x_1) = \text{sign}(0) = 1 \neq y_1$ 錯

[e] $(0,0,0,0,0,-1)$

$\exists\ h(x) = -x_2^2$

$x_1 = (0,1)$, $y_1 = -1$: $h(x_1) = -1 = y_1$

$x_2 = (0,-1)$, $y_2 = -1$: $h(x_2) = -1 = y_2$

$x_3 = (-1,0)$, $y_3 = 1$: $h(x_3) = \text{sign}(0) = 1 = y_3$

$x_4 = (1,0)$, $y_4 = 1$: $h(x_4) = \text{sign}(0) = 1 = y_4$

都符合，選e ※

9. ANS [b]

Consider a feature transform $\Phi(\mathrm{x}) = \Gamma\mathrm{x}$ where Γ is a $(d+1)$ by $(d+1)$ invertible matrix. For a training data set $\{(\mathrm{x}_n, y_n)\}_{n=1}^{N}$, run linear regression on the original data set, and get $\mathrm{w}_{\mathrm{lin}}$. Then, run linear regression on the Φ-transformed data, and get $\tilde{\mathrm{w}}$. For simplicity, assume that the matrix X (with every row being x_n^T) satisfies that $\mathrm{X}^T\mathrm{X}$ is invertible. What is the relationship between $\mathrm{w}_{\mathrm{lin}}$ and $\tilde{\mathrm{w}}$? Choose the correct answer; explain your answer.

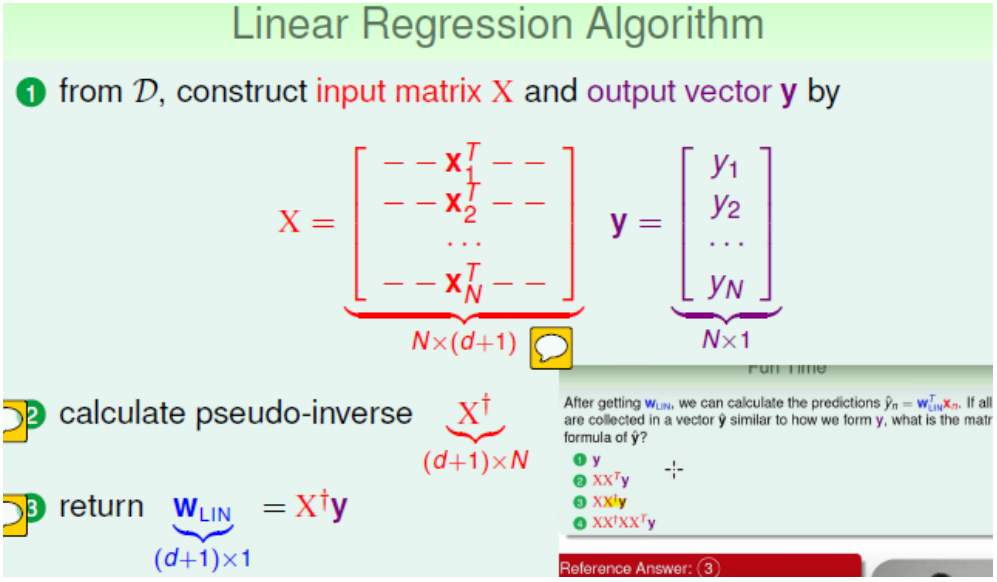
[a] $\mathrm{w}_{\mathrm{lin}} = \Gamma\tilde{\mathrm{w}}$

[b] $\mathrm{w}_{\mathrm{lin}} = \Gamma^T\tilde{\mathrm{w}}$

[c] $\mathrm{w}_{\mathrm{lin}} = (\Gamma^{-1})^T\tilde{\mathrm{w}}$

[d] $\mathrm{w}_{\mathrm{lin}} = \Gamma^{-1}\tilde{\mathrm{w}}$

[e] none of the other choices



根據投影片可知 X

$$\Phi(x) = \Gamma x \quad \left(X = \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_N^T- \end{bmatrix} \right)$$

$$\tilde{X} = \begin{bmatrix} -(\Gamma x_1)^T- \\ \vdots \\ -(\Gamma x_N)^T- \end{bmatrix} = \begin{bmatrix} -x_1^T\Gamma^T- \\ \vdots \\ -x_N^T\Gamma^T- \end{bmatrix}$$

$$= \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_N^T- \end{bmatrix}\Gamma^T = X\Gamma^T$$

$$\tilde{w} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

$$= \left((X\,T^T)^T (X\,T^T)\right)^{-1} (X\,T^T)^T y$$

$$= \left(T X^T \;\; X T^T\right)^{-1} (T X^T)\, y$$

$$= (T^T)^{-1} X^{-1} (X^T)^{-1} \underbrace{T^{-1} T}_{I_N} X^T y$$

$$= (T^T)^{-1} X^{-1} (X^T)^{-1} X^T y$$

$$= (T^T)^{-1} \underbrace{(X^T X)^{-1} X^T y}_{w_{LIN}}$$

$$= (T^T)^{-1} w_{LIN}$$

$$\tilde{w} = (T^T)^{-1} w_{LIN}$$

得證 $w_{LIN} = T^T \tilde{w}$ ~~#~~ 選 b

10. ANS [c]

After "visualizing" the data and noticing that all $\mathbf{x}_1$,, $\mathbf{x}_2$, …, $\mathbf{x}_n$ are distinct, Dr. Trans magically decides the following transform

$$\mathbf{\Phi}(\mathbf{x}) = (\llbracket \mathbf{x} = \mathbf{x}_1 \rrbracket, \llbracket \mathbf{x} = \mathbf{x}_2 \rrbracket, \ldots, \llbracket \mathbf{x} = \mathbf{x}_N \rrbracket).$$

That is, $\mathbf{\Phi}(\mathbf{x})$ is a N-dimentional vector whose n-th component is 1 if and only if $\mathbf{x} = \mathbf{x}_n$. If we run linear regression after applying this transform, what is the optimal $\tilde{\mathbf{w}}$? Choose the correct answer; explain your answer.

[a] $\mathbf{1}$, the vector of all 1s.

[b] $\mathbf{0}$, the vector of all 0s.

[c] $\mathbf{y}$

[d] $-\mathbf{y}$

[e] none of the other choices

(Note: Be sure to also check what $E_{\text{in}}(\tilde{\mathbf{w}})$ is!)

$\Phi(X) = ([\![X = X_1]\!], [\![X = X_2]\!], \cdots, [\![X = X_N]\!])$

N 筆資料經過 $\Phi(X)$

$\Phi(X_1) = (\overbrace{1, 0, 0, \cdots, 0}^{N})$

$\Phi(X_2) = (0, 1, 0, \cdots, 0)$

$\vdots$

$\Phi(X_N) = (0, 0, \cdots, 0, 1)$

∃ transform 完 $X = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \vdots & 1 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} = I_N$

Linear Regression 的 optimal w 為 w_{LIN}

$\tilde{w}_{LIN} = X^{\dagger} y$ pseudo-inverse

$\tilde{w}_{LIN} = (X^T X)^{-1} X^T y$

$(X^T X)^{-1} X^T = (I_N^T I_N)^{-1} I_N = I_N$

$\tilde{w}_{LIN} = (X^T X)^{-1} X^T y = I_N \cdot y = y$ ※ 得證

選 C。

Linear Regression

11. ANS [c]

Assume that we coulple linear regression with one-versus-all decomposition for multi-class classification, and get K weight vectors $\mathbf{w}_{[k]}^*$. Assume that the squared error $E_{\text{in}}^{\text{sqr}}(\mathbf{w}_{[k]}^*)$ for the k-th binary classification problem is e_k. What is the tightest upper bound of $E_{\text{in}}^{0/1}(g)$, where g is the multi-class classifier formed by the one-versus-all decomposition? Choose the correct answer; explain your answer.

[a] $2\sum_{k=1}^{K} e_k$

[b] $\sum_{k=1}^{K} e_k$

[c] $\frac{1}{2}\sum_{k=1}^{K} e_k$

[d] $\frac{1}{K}\sum_{k=1}^{K} e_k$

[e] $\frac{1}{2K}\sum_{k=1}^{K} e_k$

For one binary classifier

$\Rightarrow E_{in}^{sqr}(w_{[k]}^*) = e_k$

$$e_k = \left[w_{[k]}^{*T} X - (2[\![y=k]\!] - 1) \right]^2$$

$\Rightarrow E_{in}^{0/1}(w_{[k]}^*) = \left[\text{sign}(w_{[k]}^{*T} X) = [\![y=k]\!] \right]$

假設 $k=4$: △, ○, □, ☆

有4個 binary classifier

△ : △ or not △

○ : ○ or not ○

□ : □ or not □

☆ : ☆ or not ☆

假設 Label : △

猜出來

△ : !△ 錯

○ : ○ 錯

□ : !□

☆ : !☆

至少 2個 binary classifier 發生錯誤 就會使結果出錯

要找 $E_{in}^{0/1}(g)$ 的 tightest upper bound

M：至少多少個 classifier 判斷錯誤才會出現 mistake

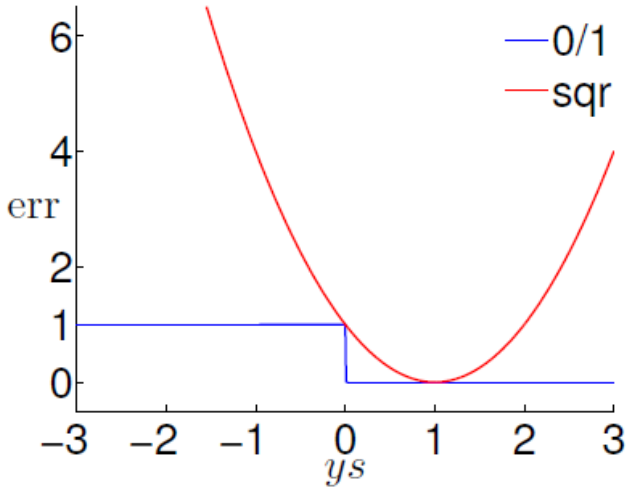
M = 2

$$E_{in}^{0/1}(g) = \frac{1}{2}\sum_{k=1}^{K} E_{in}^{0/1}(w^*_{[k]}) \le \frac{1}{2}\sum_{k=1}^{K} E_{in}^{sqr}(w^*_{[k]}) \le \frac{1}{2}\sum_{k=1}^{K} e_k$$

假設 g 有 10 個犯錯

代表在 K 個 $w^*_{[k]}$ 中至少有 2×10 個 binary classifier 犯錯

$E_{in}^{0/1}$ 會一直在 E_{in}^{sqr} 下面 (如下圖)



選 C。

Experiments with Linear and Nonlinear Models

Next, we will play with transform + linear regression for binary classification. Please use the following set for training:

`https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat`

and the following set for testing (estimating E_{out}):

`https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat`

Each line of the data set contains one $(\mathbf{x}_n, y_n)$ with $\mathbf{x}_n \in \mathbb{R}^{10}$. The first 10 numbers of the line contains the components of $\mathbf{x}_n$ orderly, the last number is y_n, which belongs to $\{-1, +1\} \subseteq \mathbb{R}$. That is, we can use those y_n for either binary classification or regression.

12. ANS [b]

(*) Consider the following *homogeneous* order-Q polynomial transform

$$\mathbf{\Phi}(\mathbf{x}) = (1, x_1, x_2, ..., x_{10}, x_1^2, x_2^2, ..., x_{10}^2, ..., x_1^Q, x_2^Q, ..., x_{10}^Q).$$

Transform the training and testing data according to $\mathbf{\Phi}(\mathbf{x})$ with $Q = 2$, and implement the linear regression algorithm on the transformed data. What is $\left|E_{\text{in}}^{0/1}(g) - E_{\text{out}}^{0/1}(g)\right|$, where g is the hypothesis returned by the transform + linear regression procedure? Choose the closest answer; provide your code.

[a] 0.28
[b] 0.32
[c] 0.36
[d] 0.40
[e] 0.44

```
21
Ein:  0.181
Eout:  0.5073333333333333
0.3263333333333333
```

Colab :

https://colab.research.google.com/drive/18Z_WwMk1o4neFzO4tIG21bEd18GpWKu9?usp=sharing

```
import urllib.request
import numpy as np
import random
import math
```

```
def  getData(url):
    #get  data
    contents  =  urllib.request.urlopen(url).read()
    text  =  str(contents,'utf-8')
    #  data  preprocessing
    sptext  =  text.split('\n')

    X  =  []
    Y  =  []

    for  i  in  range(len(sptext)-1):
        data  =  sptext[i].split('\t')
        #  print(data)
        label  =  data[-1]
        #  print(type(data))
        data.pop()
        #  print(len(data))
        X  +=  [data]
        Y  +=  [label]

    X  =  np.asarray(X,  dtype  =  float)
    Y  =  np.asarray(Y,  dtype  =  float)
    #  print(X)
    #  print(Y)
    return  X,  Y
```

```
def  transform(X,  Q):
    trans_X  =  []
    for  x  in  X:
        x_Q  =  []
        for  q  in  range(2,  Q+1):
            x_q  =  np.zeros(len(x))
            x_q  =  np.power(x,  q)
            x_Q  =  np.concatenate([x_Q,  x_q])
        #  print(x_Q)
        x  =  np.insert(x,  0,  1)
        x_Q  =  np.asarray(x_Q)
        x  =  np.concatenate([x,  x_Q])
        trans_X.append(x)
    #  print(trans_X[0])
    return  trans_X
```

```
if __name__ == '__main__':
    train_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat"
    test_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat"
    train_X ,train_Y = getData(train_data)
    test_X, test_Y = getData(test_data)
    Q = 2
    trans_train_X = transform(train_X, Q)
    # print(type(trans_train_X[0]))
    trans_test_X = transform(test_X, Q)
    # print(len(trans_test_X[0]))

    # Calculate pseudo-inverse X_cross
    trans_train_X_cross = np.linalg.pinv(trans_train_X)
    # print(trainX_cross, type(trainX_cross))

    # Calculate WLIN = X_cross · Y
    WLIN = np.matmul(trans_train_X_cross, train_Y)
    print(len(WLIN))

    # Calculate Ein
    EinN = 0
    for i in range(len(trans_train_X)):
        h_xn = np.matmul(WLIN.T, trans_train_X[i])
        if np.sign(h_xn) != train_Y[i]:
            EinN += 1
    # print("EinN: ", EinN)
    Ein = EinN/len(trans_train_X)
    print("Ein: ", Ein)

    # Calculate Eout
    EoutN = 0
    for i in range(len(trans_test_X)):
        h_xn = np.matmul(WLIN.T, trans_test_X[i])
        if np.sign(h_xn) != test_Y[i]:
            EoutN += 1
    # print("EoutN: ", EoutN)
    Eout = EoutN/len(trans_test_X)
    print("Eout: ", Eout)

    result = abs(Ein - Eout)
    print(result)
```

結果：0.3263333333333333，選 b。

13. ANS [d]

(*) Repeat the previous problem, but with $Q = 8$ instead. What is $\left|E_{\text{in}}^{0/1}(g) - E_{\text{out}}^{0/1}(g)\right|$, where g is the hypothesis returned by the transform + linear regression procedure? Choose the closest answer; provide your code.

[a] 0.30
[b] 0.35
[c] 0.40
[d] 0.45
[e] 0.50

```
81
Ein:  0.052
Eout:  0.5096666666666667
0.4576666666666667
```

Colab :

https://colab.research.google.com/drive/15zNujr4Yuvh9ChRQ0bCMdUIbBm0j69I_?usp=sharing

```
import urllib.request
import numpy as np
import random
import math
```

```
def  getData(url):
    #get  data
    contents  =  urllib.request.urlopen(url).read()
    text  =  str(contents,'utf-8')
    #  data  preprocessing
    sptext  =  text.split('\n')

    X  =  []
    Y  =  []

    for  i  in  range(len(sptext)-1):
        data  =  sptext[i].split('\t')
        #  print(data)
        label  =  data[-1]
        #  print(type(data))
        data.pop()
        #  print(len(data))
        X  +=  [data]
        Y  +=  [label]

    X  =  np.asarray(X,  dtype  =  float)
    Y  =  np.asarray(Y,  dtype  =  float)
    #  print(X)
    #  print(Y)
    return  X,  Y
```

```
def  transform(X,  Q):
    trans_X  =  []
    for  x  in  X:
        x_Q  =  []
        for  q  in  range(2,  Q+1):
            x_q  =  np.zeros(len(x))
            x_q  =  np.power(x,  q)
            x_Q  =  np.concatenate([x_Q,  x_q])
        #  print(x_Q)
        x  =  np.insert(x,  0,  1)
        x_Q  =  np.asarray(x_Q)
        x  =  np.concatenate([x,  x_Q])
        trans_X.append(x)
    #  print(trans_X[0])
    return  trans_X
```

```python
if __name__ == '__main__':
    train_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat"
    test_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat"
    train_X ,train_Y = getData(train_data)
    test_X, test_Y = getData(test_data)
    Q = 8
    trans_train_X = transform(train_X, Q)
    # print(type(trans_train_X[0]))
    trans_test_X = transform(test_X, Q)
    # print(len(trans_test_X[0]))

    # Calculate pseudo-inverse X_cross
    trans_train_X_cross = np.linalg.pinv(trans_train_X)
    # print(trainX_cross, type(trainX_cross))

    # Calculate WLIN = X_cross · Y
    WLIN = np.matmul(trans_train_X_cross, train_Y)
    print(len(WLIN))

    # Calculate Ein
    EinN = 0
    for i in range(len(trans_train_X)):
        h_xn = np.matmul(WLIN.T, trans_train_X[i])
        if np.sign(h_xn) != train_Y[i]:
            EinN += 1
    # print("EinN: ", EinN)
    Ein = EinN/len(trans_train_X)
    print("Ein: ", Ein)

    # Calculate Eout
    EoutN = 0
    for i in range(len(trans_test_X)):
        h_xn = np.matmul(WLIN.T, trans_test_X[i])
        if np.sign(h_xn) != test_Y[i]:
            EoutN += 1
    # print("EoutN: ", EoutN)
    Eout = EoutN/len(trans_test_X)
    print("Eout: ", Eout)

    result = abs(Ein - Eout)
    print(result)
```

結果： 0.4576666666666667 ，選 d。

14. ANS [a]

(*) Repeat the previous problem, but with Φ_2 (the full order-2 polynomial transform introduced in the lecture, which is of $1+10+45+10$ dimensions) instead. What is $\left|E_{\text{in}}^{0/1}(g) - E_{\text{out}}^{0/1}(g)\right|$, where g is the hypothesis returned by the transform + linear regression procedure? Choose the closest answer; provide your code.

[a] 0.33
[b] 0.41
[c] 0.49
[d] 0.57
[e] 0.65

```
66
Ein:  0.168
Eout:  0.5066666666666667
0.33866666666666667
```

Colab :

https://colab.research.google.com/drive/1FV1xQNaU9aZW-1zMVmUhwhK7fiNlkKEz?usp=sharing

```
import urllib.request
import numpy as np
import random
import math
```

```
def getData(url):
    #get data
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # data preprocessing
    sptext = text.split('\n')

    X = []
    Y = []

    for i in range(len(sptext)-1):
        data = sptext[i].split('\t')
        # print(data)
        label = data[-1]
        # print(type(data))
        data.pop()
        # print(len(data))
        X += [data]
        Y += [label]

    X = np.asarray(X, dtype = float)
    Y = np.asarray(Y, dtype = float)
    # print(X)
    # print(Y)
    return X, Y
```

```
def  transform_all(X,  Q):
    trans_X  =  []
    for  x  in  X:
        x_Q  =  []
        for  i  in  range(len(x)):
            for  j  in  range(i+1):
                x_Q.append(x[i]*x[j])
        #  print(len(x_Q))
        x  =  np.insert(x,  0,  1)
        x_Q  =  np.asarray(x_Q)
        x  =  np.concatenate([x,  x_Q])
        trans_X.append(x)
    #  print(trans_X[0])

    return  trans_X
```

主要就是差在要產生交叉的項，利用雙層 for 迴圈去兩兩相乘。

```python
if __name__ == '__main__':
    train_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat"
    test_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat"
    train_X ,train_Y = getData(train_data)
    test_X, test_Y = getData(test_data)
    Q = 8
    trans_train_X = transform_all(train_X, Q)
    # print(type(trans_train_X[0]))
    trans_test_X = transform_all(test_X, Q)
    # print(len(trans_test_X[0]))

    # Calculate pseudo-inverse X_cross
    trans_train_X_cross = np.linalg.pinv(trans_train_X)
    # print(trainX_cross, type(trainX_cross))

    # Calculate WLIN = X_cross · Y
    WLIN = np.matmul(trans_train_X_cross, train_Y)
    print(len(WLIN))

    # Calculate Ein
    EinN = 0
    for i in range(len(trans_train_X)):
        h_xn = np.matmul(WLIN.T, trans_train_X[i])
        if np.sign(h_xn) != train_Y[i]:
            EinN += 1
    # print("EinN: ", EinN)
    Ein = EinN/len(trans_train_X)
    print("Ein: ", Ein)

    # Calculate Eout
    EoutN = 0
    for i in range(len(trans_test_X)):
        h_xn = np.matmul(WLIN.T, trans_test_X[i])
        if np.sign(h_xn) != test_Y[i]:
            EoutN += 1
    # print("EoutN: ", EoutN)
    Eout = EoutN/len(trans_test_X)
    print("Eout: ", Eout)

    result = abs(Ein - Eout)
    print(result)
```

結果：0.33866666666666667，選 a。

15. ANS [c]

(*) Instead of transforming to a higher dimensional space, we can also transform to a lower dimensional space. Consider the following 10 transforms:

$$\begin{aligned} \mathbf{\Phi}^{(1)}(\mathbf{x}) &= (x_0, x_1) \\ \mathbf{\Phi}^{(2)}(\mathbf{x}) &= (x_0, x_1, x_2) \\ &\cdots \\ \mathbf{\Phi}^{(10)}(\mathbf{x}) &= (x_0, x_1, x_2, \ldots, x_{10}) \end{aligned}$$

Run $\mathbf{\Phi}^{(i)}$ + linear regression to get a hypothesis g_i. Which i leads to a $\mathbf{\Phi}^{(i)}$ that reaches the minimum $\left|E_{\text{in}}^{0/1}(g_i) - E_{\text{out}}^{0/1}(g_i)\right|$? Choose the closest answer; provide your code.

[a] 1
[b] 2
[c] 3
[d] 5
[e] 8

```
3
0.1323333333333333
```

Colab :

https://colab.research.google.com/drive/1dswq4NWMBq0R6YL0orOvoMyh30TwAoZn?usp=sharing

```
import urllib.request
import numpy as np
import random
import math
```

```
def  getData(url):
    #get  data
    contents  =  urllib.request.urlopen(url).read()
    text  =  str(contents,'utf-8')
    #  data  preprocessing
    sptext  =  text.split('\n')

    X  =  []
    Y  =  []

    for  i  in  range(len(sptext)-1):
        data  =  sptext[i].split('\t')
        #  print(data)
        label  =  data[-1]
        #  print(type(data))
        data.pop()
        #  print(len(data))
        X  +=  [data]
        Y  +=  [label]

    X  =  np.asarray(X,  dtype  =  float)
    Y  =  np.asarray(Y,  dtype  =  float)
    #  print(X)
    #  print(Y)
    return  X,  Y
```

```
def  transform_low(X,  i):
    trans_X  =  []
    for  x  in  X:
        x  =  x[:i+1]
        x  =  np.insert(x,  0,  1)
        trans_X.append(x)
    #  print(trans_X[0])
    return  trans_X
```

每次只抓出前 i+1 項(i 從 0 開始)，再去補 1。

```python
if __name__ == '__main__':
    train_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat"
    test_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat"
    train_X ,train_Y = getData(train_data)
    test_X, test_Y = getData(test_data)
    I = 10
    E = []
    for i in range(I):
        trans_train_X = transform_low(train_X, i)
        # print(type(trans_train_X[0]))
        trans_test_X = transform_low(test_X, i)
        # print(len(trans_test_X[0]))

        # Calculate pseudo-inverse X_cross
        trans_train_X_cross = np.linalg.pinv(trans_train_X)
        # print(trainX_cross, type(trainX_cross))

        # Calculate WLIN = X_cross · Y
        WLIN = np.matmul(trans_train_X_cross, train_Y)
        print(len(WLIN))

        # Calculate Ein
        EinN = 0
        for i in range(len(trans_train_X)):
            h_xn = np.matmul(WLIN.T, trans_train_X[i])
            if np.sign(h_xn) != train_Y[i]:
                EinN += 1
        # print("EinN: ", EinN)
        Ein = EinN/len(trans_train_X)
        print("Ein: ", Ein)

        # Calculate Eout
        EoutN = 0
        for i in range(len(trans_test_X)):
            h_xn = np.matmul(WLIN.T, trans_test_X[i])
            if np.sign(h_xn) != test_Y[i]:
                EoutN += 1
        # print("EoutN: ", EoutN)
        Eout = EoutN/len(trans_test_X)
        print("Eout: ", Eout)

        result = abs(Ein - Eout)
        print(result)
        E.append(result)

    print(E)
    print(E.index(min(E))+1)
    print(min(E))
```

結果：
```
3
0.1323333333333333
```
，選 c。

16. ANS [d]

(*) Consider a transform that randomly chooses 5 out of 10 dimensions. That is, $\mathbf{\Phi}(\mathbf{x}) = (x_0, x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, x_{i_5})$, where i_1 to i_5 are distinct random integers uniformly and independently generated within $\{1, 2, \ldots, 10\}$. Run $\mathbf{\Phi}$ + linear regression to get a hypothesis g. What is the average $\left|E_{\text{in}}^{0/1}(g_i) - E_{\text{out}}^{0/1}(g_i)\right|$ over 200 experiments, each generating $\mathbf{\Phi}$ with a different random seed? Choose the closest answer; provide your code.

[a] 0.06
[b] 0.11
[c] 0.16
[d] 0.21
[e] 0.26

200
0.19996333333333333

Colab :

https://colab.research.google.com/drive/1VSDBvl55SRdAiN53wn4tOIBPqlOItgjN?usp=sharing

```
import urllib.request
import numpy as np
import random
import math
```

```
def  getData(url):
    #get  data
    contents  =  urllib.request.urlopen(url).read()
    text  =  str(contents,'utf-8')
    #  data  preprocessing
    sptext  =  text.split('\n')

    X  =  []
    Y  =  []

    for  i  in  range(len(sptext)-1):
        data  =  sptext[i].split('\t')
        #  print(data)
        label  =  data[-1]
        #  print(type(data))
        data.pop()
        #  print(len(data))
        X  +=  [data]
        Y  +=  [label]

    X  =  np.asarray(X,  dtype  =  float)
    Y  =  np.asarray(Y,  dtype  =  float)
    #  print(X)
    #  print(Y)
    return  X,  Y
```

```
def  transform_ran(X,  x_i):
    trans_X  =  []
    for  x  in  X:
        xl  =  []
        for  i  in  x_i:
            xl.append(x[i-1])
        #  print(xl)
        x  =  np.asarray(xl)
        x  =  np.insert(x,  0,  1)
        trans_X.append(x)
    #  print(trans_X[0])
    return  trans_X
```

在主程式產生好每次 iteration 要隨機抓取的五個 index，再進入 transform_ran function，依照 random index list：x_i 來抓出每一筆 x。

```
if __name__ == '__main__':
    train_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_train.dat"
    test_data = "https://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw3/hw3_test.dat"
    train_X ,train_Y = getData(train_data)
    test_X, test_Y = getData(test_data)
    iteration = 200
    E = []
    x_index = [1,2,3,4,5,6,7,8,9,10]
    for iter in range(iteration):
        random.seed(iter)
        x_i = random.sample(x_index, 5)
        print(x_i)
        trans_train_X = transform_ran(train_X, x_i)
        # print(type(trans_train_X[0]))
        trans_test_X = transform_ran(test_X, x_i)
        # print(len(trans_test_X[0]))

        # Calculate pseudo-inverse X_cross
        trans_train_X_cross = np.linalg.pinv(trans_train_X)
        # print(trainX_cross, type(trainX_cross))

        # Calculate WLIN = X_cross · Y
        WLIN = np.matmul(trans_train_X_cross, train_Y)
        # print(len(WLIN))
```

```
        # Calculate Ein
        EinN = 0
        for i in range(len(trans_train_X)):
            h_xn = np.matmul(WLIN.T, trans_train_X[i])
            if np.sign(h_xn) != train_Y[i]:
                EinN += 1
        # print("EinN: ", EinN)
        Ein = EinN/len(trans_train_X)
        print("Ein: ", Ein)

        # Calculate Eout
        EoutN = 0
        for i in range(len(trans_test_X)):
            h_xn = np.matmul(WLIN.T, trans_test_X[i])
            if np.sign(h_xn) != test_Y[i]:
                EoutN += 1
        # print("EoutN: ", EoutN)
        Eout = EoutN/len(trans_test_X)
        print("Eout: ", Eout)

        result = abs(Ein - Eout)
        print("Ein - Eout: ",result)
        E.append(result)

    print(len(E))
    print(np.mean(E))
```

```
---
0.19996333333333333
```

結果： ，選 d。