Q1

a noisy target : $y = w_f^T x + \epsilon$

i.i.d. ϵ : $\mu = 0$, $\sigma^2 = \sigma^2$

$P(X) \rightarrow \mathcal{X}$

$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

Expected $E_{in}(w_{lin})$ w.r.t. $D = \sigma^2\left(1 - \frac{d+1}{N}\right)$

$\sigma = 0.1$, $d = 19$

$0.1^2\left(1 - \frac{19+1}{N}\right) \geq 0.005$

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

$2 - \frac{40}{N} \geq 1$

$\frac{40}{N} \leq 1$

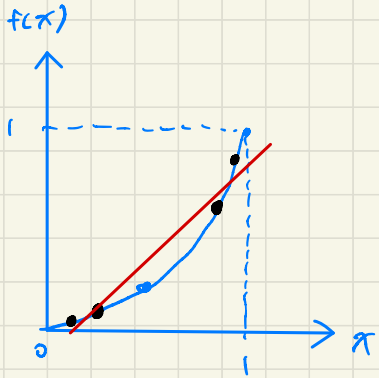
$N \geq 40$ #

Q2

Target function : $f(x) = x^2$

$X \sim$ uniform $[0, 1]$

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



squared error

optimal hypothesis $g^* \rightarrow \min E_{out}(g^*)$

$$E_{out}(w^*) = \underset{(x,y)\sim P}{\mathcal{E}} (w^{*T} x - y)^2$$

$$= \int_0^1 \left(\begin{bmatrix} w_0 & w_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} - x^2 \right)^2 dx$$

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

$$= \int_0^1 \left[w_0^2 + w_1^2 x^2 + x^4 + 2(w_0 w_1 x - w_1 x^3 - w_0 x^2) \right] dx$$

$$= \int_0^1 \left[x^4 - 2w_1 x^3 + (w_1^2 - 2w_0) x^2 + 2w_0 w_1 x + w_0^2 \right] dx$$

$$= \frac{1}{5} x^5 - \frac{w_1}{2} x^4 + \frac{w_1^2 - 2w_0}{3} x^3 + w_0 w_1 x^2 + w_0^2 x \Big|_0^1$$

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

$\mathbb{E}[r(x)]$

$= \int_{-\infty}^{\infty} r(x) p(x) dx$

Here $p(x) = 1$.

$$\frac{\partial E_{out}}{\partial w_0} = -\frac{2}{3} + w_1 + 2w_0$$

$$\frac{\partial E_{out}}{\partial w_1} = -\frac{1}{2} + \frac{2}{3}w_1 + w_0$$

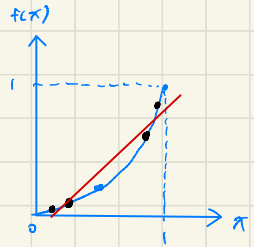
$$\begin{cases} w_1 + 2w_0 = \frac{2}{3} \\ \frac{2}{3}w_1 + w_0 = \frac{1}{2} \end{cases} \Rightarrow \begin{cases} 3w_1 + 6w_0 = 2 \\ 4w_1 + 6w_0 = 3 \end{cases}$$

$$w_1 = 1$$

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

#

Q3



$X \sim$ uniform $[0, 1]$

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

squared error, neglect degenerate case

$\Rightarrow \mathbb{E}_D (| E_{in}(g) - E_{out}(g) |)$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}$$

$$X^\dagger = (X^T X)^{-1} X^T$$

$$= \left(\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & x_1 + x_2 \\ x_1 + x_2 & x_1^2 + x_2^2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$$

$$= \frac{1}{2x_1^2 + 2x_2^2 - x_1^2 - 2x_1 x_2 - x_2^2} \begin{bmatrix} x_1^2 + x_2^2 & -x_1 - x_2 \\ -x_1 - x_2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$$

$$= \frac{1}{x_1^2 + x_2^2 - 2x_1 x_2} \begin{bmatrix} x_1^2 + x_2^2 - x_1^2 - x_1 x_2 & x_1^2 + x_2^2 - x_1 x_2 - x_2^2 \\ -x_1 - x_2 + 2x_1 & -x_1 - x_2 + 2x_2 \end{bmatrix}$$

$$= \frac{1}{(x_1 - x_2)^2} \begin{bmatrix} \frac{x_2^2 - x_1 x_2}{x_2(x_2 - x_1)} & \frac{x_1^2 - x_1 x_2}{x_1(x_1 - x_2)} \\ x_1 - x_2 & x_2 - x_1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{x_2}{x_2 - x_1} & \frac{x_1}{x_1 - x_2} \\ \frac{1}{x_1 - x_2} & \frac{1}{x_2 - x_1} \end{bmatrix}$$

$$w_{LIN} = X^\dagger y$$

$$= \begin{bmatrix} \frac{x_2}{x_2 - x_1} & \frac{x_1}{x_1 - x_2} \\ \frac{1}{x_1 - x_2} & \frac{1}{x_2 - x_1} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} \frac{x_2 x_1^2}{x_2 - x_1} + \frac{x_1 x_2^2}{x_1 - x_2} \\ \frac{x_1^2}{x_1 - x_2} + \frac{x_2^2}{x_2 - x_1} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{-x_1^2 x_2 + x_1 x_2^2}{x_1 - x_2} \\ \frac{x_1^2 - x_2^2}{x_1 - x_2} \end{bmatrix} = \begin{bmatrix} \frac{x_1 x_2(-x_1 + x_2)}{x_1 - x_2} \\ \frac{(x_1 + x_2)(x_1 - x_2)}{x_1 - x_2} \end{bmatrix} = \begin{bmatrix} -x_1 x_2 \\ x_1 + x_2 \end{bmatrix}$$

$$\mathbb{E}_D(|E_{in}(g) - E_{out}(g)|)$$

$$= \mathbb{E}_D(|0 - E_{out}(g)|)$$

$$= \mathbb{E}_D(E_{out}(g))$$

Based on Q2's result, $E_{out}(w) = \frac{1}{5} - \frac{w_1}{2} + \frac{w_1^2 - 2w_0}{3} + w_0 w_1 + w_0^2$

$\mathbb{E}_D(E_{out}(g))$ →

$\mathbb{E}[r(x_1, x_2)] = \iint_{R^2} r(x_1, x_2) p(x_1, x_2) dx_1 dx_2$

Here $p(x_1, x_2) = 1$.

$$= \int_0^1 \int_0^1 \left(w_0^2 - \frac{2}{3} w_0 + w_0 w_1 + \frac{1}{3} w_1^2 - \frac{1}{2} w_1 + \frac{1}{5} \right) dx_1 dx_2$$

which $\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -x_1 x_2 \\ x_1 + x_2 \end{bmatrix}$

$$= \int_0^1 \int_0^1 \left(x_1^2 x_2^2 + \frac{2}{3} x_1 x_2 - x_1^2 x_2 - x_1 x_2^2 + \frac{1}{3} x_1^2 + \frac{2}{3} x_1 x_2 + \frac{1}{3} x_2^2 - \frac{1}{2} x_1 - \frac{1}{2} x_2 + \frac{1}{5} \right) dx_1 dx_2$$

$$= \int_0^1 \int_0^1 \left(x_1^2 x_2^2 - x_1^2 x_2 - x_1 x_2^2 + \frac{4}{3} x_1 x_2 + \frac{1}{3} x_1^2 + \frac{1}{3} x_2^2 - \frac{1}{2} x_1 - \frac{1}{2} x_2 + \frac{1}{5} \right) dx_1 dx_2$$

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

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

$$= \int_0^1 \left(\frac{1}{6}x_2^2 - \frac{1}{6}x_2 + \frac{11}{180} \right) dx_2$$

$$= \frac{1}{18}x_2^3 - \frac{1}{12}x_2^2 + \frac{11}{180}x_2 \Big|_0^1$$

$$= \frac{1}{18} - \frac{1}{12} + \frac{11}{180}$$

$$= \frac{10 - 15 + 11}{180}$$

$$= \frac{6}{180}$$

$$= \frac{1}{30}$$

#

Q4

$-\ln\theta(y_n w^T x_n)$

$\Leftrightarrow \ln\theta(-y_n w^T x_n)$

→ Minimizing $\sum -\ln\theta(y_n w^T x_n)$ is equivalent to minizing $\sum \theta(-y_n w^T x_n)$

For $y_n = 1 \rightarrow \ln\theta(-w^T x_n)$, $(y_n' = 1)$

[a] → ~~$\ln\theta(w^T x_n)$~~

[b] → $\ln\theta(-w^T x_n)$

[c] → ~~$\ln\theta(w^T x_n)$~~

[d] → $\ln\theta(-w^T x_n)$

⇒ For $y_n' = 1$, [b], [d] are the same as the original error term.

For $y_n = -1 \rightarrow \ln\theta(w^T x_n)$, $(y_n' = 0)$

[b] → $\ln\theta(w^T x_n)$ ✓

[d] → $-\ln\theta(w^T x_n)$

$\Leftrightarrow$ ~~$\ln\theta(-w^T x_n)$~~

⇒ For $y_n' = 0$, just [b] is left to be the same as the original error term.

#

Q5

Coin. $\begin{cases} f(1) = \mu \\ f(0) = 1-\mu \end{cases}$

$y_1, y_2, \dots, y_N$

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

② Likelihood function:

$$f_n(y|\theta) = \prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{1-y_i}$$

$$L(\theta) = \log f_n(y|\theta) = \sum_{i=1}^{n}[y_i \log\theta + (1-y_i)\log(1-\theta)]$$

$$= \left(\sum_{i=1}^{n} y_i\right)\log\theta + \left(n - \sum_{i=1}^{n} y_i\right)\log(1-\theta)$$

$$\frac{\partial L(\theta)}{\partial\theta} = \frac{\sum y_i}{\theta} + \frac{\sum y_i - n}{1-\theta}$$

$$\frac{\sum y_i}{\theta} + \frac{\sum y_i - n}{1-\theta} = 0$$

$$\frac{\sum y_i}{\theta} = \frac{n - \sum y_i}{1-\theta}$$

$$(1-\theta)\sum y_i = \theta(n - \sum y_i)$$

$$n\theta = \sum y_i, \quad \theta = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (\text{For } \sum_{i=1}^{n} y_i \notin \{0, n\})$$

$\sum y_i = 0, \ \frac{-n}{1-\theta} = 0$ (X)

$\sum y_i = n, \ \frac{n}{\theta} = 0$ (X)

If $\sum_{i=1}^{n} y_i = 0$, $L(\theta)$ achieves its maximum at $\theta = 0$.

If $\sum_{i=1}^{n} y_i = n$, $L(\theta)$ achieves its maximum at $\theta = 1$.

$$\Rightarrow \hat{\theta} = \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i = \nu$$

#

③ $$E^{sqr}(\hat{y}) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y} - y_n)^2$$

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

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

$$= \frac{1}{N}2N\hat{y} - \frac{1}{N}\sum_{n=1}^{N}2y_n$$

$$\frac{1}{N}2N\hat{y} - \frac{1}{N}\sum_{n=1}^{N}2y_n = 0$$

$$2N\hat{y} = \sum_{n=1}^{N}2y_n$$

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

#

④ $E^{ce}(\hat{y}) = -\frac{1}{N}\sum_{n=1}^{N}\left(y_n \ln \hat{y} + (1-y_n)\ln(1-\hat{y})\right)$ $\leq Z$

Minimize $E^{ce}(\hat{y}) \iff$ Maximize Z

∵ Based on ②'s result, $0 < V < 1$ can maximize $L(\theta)$. And $L(\theta)$ is in the same form as Z.

∴ $0 < V < 1$ can maximize Z and then minimize $E^{ce}(\hat{y})$. #

① Hoeffding's Inequality

$$P(|V-M| > \epsilon) \leq 2\exp(-2\epsilon^2 N)$$

$$\delta$$

$$P(|V-M| \leq \epsilon) \geq 1-\delta$$

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

$$-2\epsilon^2 N = \ln\frac{\delta}{2}$$

$$\epsilon^2 = -\frac{1}{2N}\ln\frac{\delta}{2} = \frac{1}{2N}\ln\frac{2}{\delta}$$

$$\epsilon = \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

With probability $\geq 1-\delta$,

$$|V-M| \leq \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

$$|M-V| \leq \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

$$-\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}} \leq M-V \leq \sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

$$V-\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}} \leq M \leq V+\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$$

For $M \leq V+\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}$, its probability is obviously more than $1-\delta$. #

Q6

PLA

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

SGD

$$W_{t+1} \leftarrow W_t + \eta \cdot \underbrace{\theta(-y_n W_t^T x_n)(y_n x_n)}_{-\nabla err(w_t, x_n, y_n)}$$

$$\nabla err(w_t, x_n, y_n) = \theta(-y_n w_t^T x_n)(-y_n x_n)$$

$$W_{t+1} = W_t - \eta \nabla err$$

[a] $err(w, x, y) = \max(0, -yw^T x)$

if classified falsely

$$-yw^T x > 0$$

$$err(w, x, y) = -yw^T x$$

$$\nabla err = \frac{\partial err}{\partial w} = -yx$$

$$\Rightarrow W_{t+1} = W_t - \eta \nabla err$$

$$= \underline{W_t + \eta yx}$$

$\Rightarrow$ if $\eta = 1$, same as PLA

if classified correctly

$$-yw^T x < 0$$

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

$$\nabla err = \frac{\partial err}{\partial w} = 0$$

$$\Rightarrow w_{t+1} = w_t - \eta \nabla err$$

$$= w_t$$

$\Rightarrow$ When meet correct samples, w_t will not have correction.

Same as PLA. #

Q7

MLR

$$W = \begin{bmatrix} | & | & & | & & | \\ w_1 & w_2 & \cdots & w_K & \cdots & w_K \\ | & | & & | & & | \end{bmatrix}_{(d+1)\times K}$$

$$h_y(x) = \frac{\exp(w_y^T x)}{\sum_{i=1}^{K} \exp(w_i^T x)} \longrightarrow \text{approximate } P(y|x)$$

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

$$err(W, x_n, y_n) = -\ln h_{y_n}(x_n)$$

$$= -\ln \left[\frac{\exp(w_{y_n}^T x_n)}{\sum_{i=1}^{K} \exp(w_i^T x_n)} \right]$$

$$= \ln \frac{\boxed{\sum_{i=1}^{K} \exp(w_i^T x_n)}}{\exp(w_{y_n}^T x_n)} \quad \boxed{\sum_{i=1}^{K} \exp(w_i^T x_n)} = \Delta, \quad \square$$

$$\frac{\partial err}{\partial w_{y_n}} = \frac{1}{\square} \times \frac{\partial \square}{\partial w_{y_n}}$$

$$= \frac{1}{\square} \times \frac{\exp(w_{y_n}^T x_n) \times \frac{\partial \Delta}{\partial w_{y_n}} - \Delta \times \frac{\partial \exp(w_{y_n}^T x_n)}{\partial w_{y_n}}}{[\exp(w_{y_n}^T x_n)]^2}$$

$$= \frac{1}{\square} \times \frac{\exp(w_{y_n}^T x_n) \times \exp(w_{y_n}^T x_n) \times x_n - \triangle \times \exp(w_{y_n}^T x_n) \times x_n}{\left[\exp(w_{y_n}^T x_n)\right]^2}$$

$$= \frac{1}{\square} \times \frac{\diamond \times x_n - \triangle \times x_n}{\diamond}$$

$$= \frac{\diamond}{\triangle} \times \frac{x_n(\diamond - \triangle)}{\diamond}$$

$$= x_n\left(\frac{\diamond}{\triangle} - 1\right)$$

$$= x_n\left(h_{y_n}(x_n) - 1\right)$$

$$W_{t+1} = W_t - \eta \nabla err$$

$$= W_t - \eta \frac{\partial err}{\partial w_{y_n}}$$

$$= W_t + \eta \underbrace{\left[\left(1 - h_{y_n}(x_n)\right) x_n\right]}_{V}$$

#

Q8

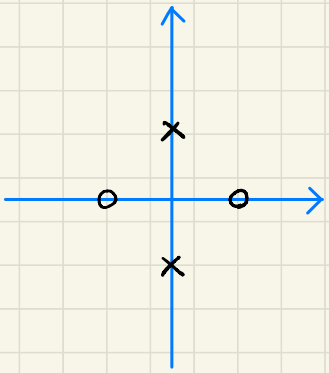
D:

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

$x_2 = (0, -1)$, $y_2 = -1$ (x)

$x_3 = (-1, 0)$, $y_3 = +1$ (o)

$x_4 = (1, 0)$, $y_4 = +1$ (o)



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

$\Phi_2(x) = -x_2^2$

$\Phi_2(x_1) = -1 \xrightarrow{\text{sign}(-1)} (-1)$

$\Phi_2(x_2) = -1 \xrightarrow{\text{sign}(-1)} (-1)$

$\Phi_2(x_3) = 0 \xrightarrow{\text{sign}(0)} (+1)$

$\Phi_2(x_4) = 0 \xrightarrow{\text{sign}(0)} (+1)$

$\Rightarrow$ Can separate all the transformed examples perfectly. #

Q9

$\Phi(x) = Px$, $P \to (d+1) \times (d+1)$, invertible

$D: \{(x_n, y_n)\}_{n=1}^{N}$

$$\begin{bmatrix} -x_1^T- \\ -x_2^T- \\ \vdots \\ -x_n^T- \end{bmatrix} \to W_{lin}$$

$$\begin{bmatrix} -(Px_1)^T- \\ \vdots \\ -(Px_n)^T- \end{bmatrix} \to \tilde{W}$$

$$X = \begin{bmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{bmatrix}, \quad X^TX \to \text{invertible}$$

$$W_{lin} = (X^TX)^{-1}X^Ty$$

$$Z = \left(P\begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}\right)^T$$

$$= (PX^T)^T$$

$$= XP^T$$

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

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

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

$$= (P^T)^{-1} (X^T X)^{-1} I_{d+1} X^T y$$

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

$$= (P^T)^{-1} w_{lin}$$

$$\Rightarrow w_{lin} = P^T \tilde{w}$$

#

Q10

$$\Phi(x) = \left([\![x = x_1]\!], [\![x = x_2]\!], \cdots, [\![x = x_n]\!] \right)$$

$$\Phi(x_1) = (1, 0, \cdots, 0)_{1\times N}$$

$$\Phi(x_2) = (0, 1, \cdots, 0)_{1\times N}$$

$$\Phi(x_n) = (0, 0, \cdots, 1)_{1\times N}$$

$$\begin{bmatrix} \Phi(x_1) \\ \Phi(x_2) \\ \vdots \\ \Phi(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{I_N} = Z$$

$$Z^{\dagger} = (Z^T Z)^{-1} Z^T$$

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

$$= I_N$$

$$\tilde{w} = Z^{\dagger} y$$

$$= I_N y = y$$

#

Q11

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

$$E_{in}^{0/1}(s, y) \le E_{in}^{sqr}(s, y)$$

$$E_{in}^{0/1}(w_k^*) \le E_{in}^{sqr}(w_k^*) = e_k$$

$$g(x) = \operatorname{argmax}_{k \in y} (w_k^T x)$$

$$E_{in}^{0/1}(w_1^*) \le e_1$$

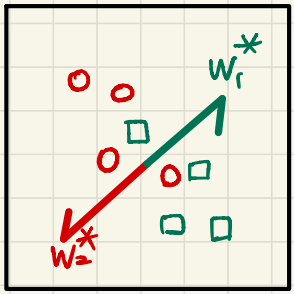
$$E_{in}^{0/1}(w_2^*) \le e_2$$

For example

$$E_{in}^{0/1}(g)$$

$$= \frac{E_{in}^{0/1}(w_1^*) + E_{in}^{0/1}(w_2^*)}{2} > 0$$

$$\le \frac{e_1 + e_2}{2}$$



In general

我們知道今天如果有1個 sample 錯了，那勢必會至少有2個 W^* 在自己的 one versus all，也就是 o/1 的判斷上出現錯誤。

因此，

$$\cancel{N} \cdot E_{in}^{o/1}(g)$$

$$\leq N \cdot \sum_{k=1}^{K} E_{in}^{o/1}(W_k^*)/2$$

$$\leq N \cdot \sum_{k=1}^{K} E_{in}^{sqr}(W_k^*)/2$$

From lecture, o/1 error ≤ sqr error

$$= \cancel{N} \cdot \sum_{k=1}^{K} e_k/2$$

$$\Rightarrow E_{in}^{o/1}(g) \leq \sum_{k=1}^{K} e_k/2$$

#

Q12

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 21 20:18:20 2021

@author: md703
"""

import numpy as np


def polyQ(x, degree):
    # add x0 = 1 first
    z = np.insert(x, 0, 1, axis=1)
    # do transform for order-degree
    for q in range(2, degree+1):
        z = np.concatenate((z, x**q), axis=1)
    return z


# parameters
Q = 2
trainDataPath = "hw3_train.dat"
testDataPath = "hw3_test.dat"

# read data
trainData = np.genfromtxt(trainDataPath)
testData = np.genfromtxt(testDataPath)

# homogeneous order-Q polynomial transform
trainData = np.concatenate((polyQ(trainData[:, :-1], degree=Q), trainData[:, -1][:, None]), axis=1)
testData = np.concatenate((polyQ(testData[:, :-1], degree=Q), testData[:, -1][:, None]), axis=1)

# do linear regression
dagger = np.linalg.pinv(trainData[:, :-1])
wlin = np.matmul(dagger, trainData[:, -1])

# calculate Ein w.r.t wlin
trainPred = np.matmul(trainData[:, :-1], wlin)
trainPred[trainPred>=0] = 1
trainPred[trainPred<0] = -1
Ein = np.mean(trainPred != trainData[:, -1])

# calculate Eout w.r.t wlin
testPred = np.matmul(testData[:, :-1], wlin)
testPred[testPred>=0] = 1
testPred[testPred<0] = -1
Eout = np.mean(testPred != testData[:, -1])

# save final result
result = abs(Ein-Eout)
```

Q13

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 21 20:56:23 2021

@author: md703
"""

import numpy as np


def polyQ(x, degree):
    # add x0 = 1 first
    z = np.insert(x, 0, 1, axis=1)
    # do transform for order-degree
    for q in range(2, degree+1):
        z = np.concatenate((z, x**q), axis=1)
    return z


# parameters
Q = 8
trainDataPath = "hw3_train.dat"
testDataPath = "hw3_test.dat"

# read data
trainData = np.genfromtxt(trainDataPath)
testData = np.genfromtxt(testDataPath)

# homogeneous order-Q polynomial transform
trainData = np.concatenate((polyQ(trainData[:, :-1], degree=Q), trainData[:, -1][:, None]), axis=1)
testData = np.concatenate((polyQ(testData[:, :-1], degree=Q), testData[:, -1][:, None]), axis=1)

# do linear regression
dagger = np.linalg.pinv(trainData[:, :-1])
wlin = np.matmul(dagger, trainData[:, -1])

# calculate Ein w.r.t wlin
trainPred = np.matmul(trainData[:, :-1], wlin)
trainPred[trainPred>=0] = 1
trainPred[trainPred<0] = -1
Ein = np.mean(trainPred != trainData[:, -1])

# calculate Eout w.r.t wlin
testPred = np.matmul(testData[:, :-1], wlin)
testPred[testPred>=0] = 1
testPred[testPred<0] = -1
Eout = np.mean(testPred != testData[:, -1])

# save final result
result = abs(Ein-Eout)
```

Q14

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 21 21:00:35 2021

@author: md703
"""

import numpy as np
import itertools


def getFullOrder2(x):
    dim = x.shape[1]
    # add x0 = 1
    z = np.insert(x, 0, 1, axis=1)
    # add x1x2, x2x3, ...
    combSet = np.array(list(itertools.combinations(np.arange(dim), 2)))
    for comb in combSet:
        multiplyTerm = x[:, comb[0]] * x[:, comb[1]]
        z = np.concatenate((z, multiplyTerm[:, None]), axis=1)
    # add x1^2, x2^2, ...
    z = np.concatenate((z, x**2), axis=1)
    return z


# parameters
trainDataPath = "hw3_train.dat"
testDataPath = "hw3_test.dat"

# read data
trainData = np.genfromtxt(trainDataPath)
testData = np.genfromtxt(testDataPath)

# full order-2 polynomial transform
trainData = np.concatenate((getFullOrder2(trainData[:, :-1]), trainData[:, -1][:, None]), axis=1)
testData = np.concatenate((getFullOrder2(testData[:, :-1]), testData[:, -1][:, None]), axis=1)

# do linear regression
dagger = np.linalg.pinv(trainData[:, :-1])
wlin = np.matmul(dagger, trainData[:, -1])

# calculate Ein w.r.t wlin
trainPred = np.matmul(trainData[:, :-1], wlin)
trainPred[trainPred>=0] = 1
trainPred[trainPred<0] = -1
Ein = np.mean(trainPred != trainData[:, -1])

# calculate Eout w.r.t wlin
testPred = np.matmul(testData[:, :-1], wlin)
testPred[testPred>=0] = 1
testPred[testPred<0] = -1
Eout = np.mean(testPred != testData[:, -1])

# save final result
result = abs(Ein-Eout)
```

Q15

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 21 21:31:14 2021

@author: md703
"""

import numpy as np


def getTransform(x, leftDim):
    # transform to a lower dimension
    x = x[:, :leftDim]
    # add x0 = 1
    z = np.insert(x, 0, 1, axis=1)
    return z


# parameters
trainDataPath = "hw3_train.dat"
testDataPath = "hw3_test.dat"

# read data
rawTrainData = np.genfromtxt(trainDataPath)
rawTestData = np.genfromtxt(testDataPath)

# main - evaluate each transform function
resultSet = []
for i in range(1, 11):
    # transform both training and testing dataset to a lower dimensional space
    trainData = np.concatenate((getTransform(rawTrainData[:, :-1], leftDim=i), rawTrainData[:, -1][:,
None]), axis=1)
    testData = np.concatenate((getTransform(rawTestData[:, :-1], leftDim=i), rawTestData[:, -1][:,
None]), axis=1)

    # do linear regression
    dagger = np.linalg.pinv(trainData[:, :-1])
    wlin = np.matmul(dagger, trainData[:, -1])

    # calculate Ein w.r.t wlin
    trainPred = np.matmul(trainData[:, :-1], wlin)
    trainPred[trainPred>=0] = 1
    trainPred[trainPred<0] = -1
    Ein = np.mean(trainPred != trainData[:, -1])

    # calculate Eout w.r.t wlin
    testPred = np.matmul(testData[:, :-1], wlin)
    testPred[testPred>=0] = 1
    testPred[testPred<0] = -1
    Eout = np.mean(testPred != testData[:, -1])

    # save result
    resultSet.append(abs(Ein-Eout))

# save final best idx
bestIdx = np.argmin(resultSet)
```

Q16

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 22 00:32:33 2021

@author: md703
"""

import numpy as np


def getTransform(x, idxSet):
    # do transform
    x = x[:, idxSet]
    # add x0 = 1
    z = np.insert(x, 0, 1, axis=1)
    return z


# parameters
trainDataPath = "hw3_train.dat"
testDataPath = "hw3_test.dat"

# read data
rawTrainData = np.genfromtxt(trainDataPath)
rawTestData = np.genfromtxt(testDataPath)

# main
resultSet = []
for _ in range(200):
    # do transform that randomly chooses 5 distinct dimensions.
    chosenIdx = np.random.choice(rawTrainData.shape[1]-1, 5, replace=False)
    trainData = np.concatenate((getTransform(rawTrainData[:, :-1], chosenIdx), rawTrainData[:, -1][:,
None]), axis=1)
    testData = np.concatenate((getTransform(rawTestData[:, :-1], chosenIdx), rawTestData[:, -1][:,
None]), axis=1)

    # do linear regression
    dagger = np.linalg.pinv(trainData[:, :-1])
    wlin = np.matmul(dagger, trainData[:, -1])

    # calculate Ein w.r.t wlin
    trainPred = np.matmul(trainData[:, :-1], wlin)
    trainPred[trainPred>=0] = 1
    trainPred[trainPred<0] = -1
    Ein = np.mean(trainPred != trainData[:, -1])

    # calculate Eout w.r.t wlin
    testPred = np.matmul(testData[:, :-1], wlin)
    testPred[testPred>=0] = 1
    testPred[testPred<0] = -1
    Eout = np.mean(testPred != testData[:, -1])

    # save result
    resultSet.append(abs(Ein-Eout))

# get the final mean
resultSetMean = np.mean(resultSet)
```