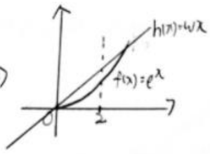
ML HW4 R09946023 吳偉樂

1) (c)

1. (C) $\left|e^x-\left(\frac{3+3e^2}{8}\right)x\right|$

$f(x)=e^x$, $h(x)=wx$, $x\in[0,2]$ ⇒



In fact, there are infinite lines that satisfy $h(x)=wx$, which pass through origin

⇒ Area between line & curve:

$$\int_0^2 (f(x)-h(x))^2dx=\int_0^2(e^x-wx)^2dx$$
$$=\int_0^2[e^{2x}-2e^xwx+w^2x^2]dx$$
$$=\left[\frac12 e^{2x}+\frac13w^2x^3\right]_0^2-2w\int_0^2e^xx\,dx$$
$$=\frac12e^4+\frac83w^2-\frac12-2w\left[xe^x-e^x\right]_0^2$$
$$=\frac12e^4+\frac83w^2-\frac12-4we^2+2we^2-2w$$
$$=\frac12e^4+\frac83w^2-\frac12-2we^2-2w\triangleq A(w)$$

best hypothesis ⇔ minimum A

$$\therefore \frac{dA}{dw}=0=\frac{16}{3}w-2e^2-2$$
$$\Rightarrow \frac83w=e^2+1 \Rightarrow w=\frac38(e^2+1)$$

∴ deterministic noise

$$=|f(x)-h(x)|$$
$$=|e^x-wx|$$
$$=\left|e^x-\frac38(e^2+1)x\right|$$
$$=\left|e^x-\frac18[3e^2+3]x\right|$$

2) (b) 1

2. (b) 1

D → train, → test, iid $\sim P$

best hypothesis s.t. E_{in} is min

$$E_D[E_{in}(A(D))]=E\left[\frac1N\sum_{i=1}^N[\![h^*(x_i)\neq y_i]\!]\right]\overset{iid}{=}E\left[[\![h^*(x_i)\neq y_i]\!]\right]\text{ for any }1\le i\le N$$
$$E_D[E_{out}(A(D))]=E\left[\frac1M\sum_{j=1}^M[\![h^*(\tilde x_j)\neq\tilde y_j]\!]\right]\overset{iid}{=}E\left([\![h^*(\tilde x_j)\neq\tilde y_j]\!]\right)\text{ for any }1\le j\le M$$

So, $E_D[E_{in}(A(D))]$ & $E_D[E_{out}(A(D))]$ regardless of N, M, depend on its hypothesis!!

Since the data is iid, so if we find a best hypothesis g^* such that $E_{out}(g^*)=\frac1M\sum_{i=1}^M[\![g^*(\tilde x_i)\neq\tilde y_i]\!]$ is minimum, then $E_D[E_{in}(h^*)]=E_D[E_{out}(g^*)]$

$$\Rightarrow E_D[E_{in}(h^*)]=E_D[E_{out}(g^*)]\le E_D[E_{out}(h^*)]$$
$$\Rightarrow E_D[E_{in}(A(D))]\le E_D[E_{out}(A(D))]$$

※.

3) (d) $2X^TX + N\sigma^2 I_{d+1}$

3. (d) $2X^TX + N\sigma^2 I_{d+1}$

$x_i = (x_{i0}, x_{i1}, x_{i2}, \cdots x_{id})^T \in \mathbb{R}^{d+1}$, $\epsilon \sim N(0, \sigma^2 I_{d+1})$ $\in \mathbb{R}^{d+1}$, $X_h = [x_1 \; x_1 \cdots x_N \; \hat{x}_1 \cdots \hat{x}_N]^T$

$$X_h^T X_h = \underbrace{\begin{bmatrix} | & | & & | & | & & | \\ x_1 & x_2 & \cdots & x_N & \hat{x}_1 & \cdots & \hat{x}_N \\ | & | & & | & | & & | \end{bmatrix}}_{(d+1)\times 2N} \underbrace{\begin{bmatrix} - x_1 - \\ \vdots \\ - x_N - \\ - \hat{x}_1 - \\ \vdots \\ - \hat{x}_N - \end{bmatrix}}_{2N\times(d+1)} = \begin{bmatrix} | & & | & | & & | \\ x_1 & \cdots & x_N & x_1+\epsilon & \cdots & x_N+\epsilon \\ | & & | & | & & | \end{bmatrix} \begin{bmatrix} - x_1 - \\ \vdots \\ - x_N - \\ - x_1+\epsilon - \\ \vdots \\ - x_N+\epsilon - \end{bmatrix}$$

$$= \Big(\underbrace{[X^T \; 0]}_{(d+1)\times 2N} + [0 \; X^T] + [0 \; \mathcal{E}^T] \Big) \Big(\underbrace{\begin{bmatrix} X \\ 0 \end{bmatrix}}_{2N\times(d+1)} + \begin{bmatrix} 0 \\ X \end{bmatrix} + \begin{bmatrix} 0 \\ \mathcal{E} \end{bmatrix} \Big)$$

$\epsilon = \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_{d+1} \end{pmatrix} \in \mathbb{R}^{d+1}$

$$= X^TX + X^TX + X^T\mathcal{E} + \mathcal{E}^TX + \mathcal{E}^T\mathcal{E}$$

$\mathcal{E}^T = \underbrace{[\epsilon \; \epsilon \cdots \epsilon]}_{(d+1)\times N}$

$$E(X_h^TX_h) = E[2X^TX] + E[X^T\mathcal{E}] + E[\mathcal{E}^TX] + E[\mathcal{E}^T\mathcal{E}]$$

$$= 2X^TX + X^T \cancel{E(\mathcal{E})} + X \cancel{E[\mathcal{E}^T]} + E(N\epsilon^2)$$

$$= 2X^TX + N E(\epsilon^2)$$

$Var(\epsilon) = \sigma^2 I_{d+1} = E(\epsilon^2) - [\cancel{E(\epsilon)}]^2$

$$= \underline{2X^TX + N\sigma^2 I_{d+1}}$$

4) (e) $2X^Ty$

4. (e) $2X^Ty$

$$X_h^Ty = ([X^T \; 0] + [0 \; X^T] + [0 \; \mathcal{E}^T]) \begin{bmatrix} y \\ y \end{bmatrix}$$

$$= X^Ty + X^Ty + \mathcal{E}^Ty$$

$$E(X_h^Ty) = 2X^Ty + y\cancel{E(\mathcal{E}^T)} = \underline{2X^Ty}$$

5） **(d)** $\frac{\gamma_i}{\gamma_i+\lambda}$

5. (d) $\frac{\gamma_i}{\gamma_i+\lambda}$

$$\frac{d}{dw}\left[\frac{1}{N}\|Zw-y\|^2+\frac{\lambda}{N}w^Tw\right] = \frac{d}{dw}\left[\frac{1}{N}(w^TZ^TZw-2w^TZ^Ty+y^Ty)+\frac{\lambda}{N}w^Tw\right]$$

$$= \frac{2}{N}[Z^TZw-Z^Ty]+\frac{\lambda}{N}\cdot 2w = 0 \Rightarrow (Z^TZ+\lambda)w = Z^Ty \Rightarrow w=(Z^TZ+\lambda)^{-1}Z^Ty$$

$$w=(Z^TZ+\lambda)^{-1}Z^Ty \overset{Z=XQ}{=} (Q^TX^TXQ+\lambda)^{-1}Q^TX^Ty = (Q^TQTQ^TQ+\lambda)^{-1}Q^TX^Ty = (T+\lambda)^{-1}Q^TX^Ty$$

when $\lambda=0$, $v = w_{lin} = T^{-1}Q^TX^Ty = \begin{bmatrix}\frac{1}{\gamma_0}&&\\&\frac{1}{\gamma_i}&\\&&\frac{1}{\gamma_d}\end{bmatrix}Q^TX^Ty$

$$\Rightarrow v = \begin{pmatrix}v_0\\v_1\\\vdots\\v_d\end{pmatrix} \Rightarrow v_i = \frac{1}{\gamma_i}\underbrace{[q_{i0}\ q_{i1}\cdots q_{id}]\,X^Ty}_{1\times1 \Rightarrow \text{scalar}}$$

$v_i = \frac{1}{\gamma_i}\cdot s$, where $s=[q_{i0}\cdots q_{id}]X^Ty$ is a scalar

Note: $T=\begin{pmatrix}\gamma_0&&\\&\ddots&\\&&\gamma_d\end{pmatrix}$, $T^{-1}=\begin{pmatrix}\frac{1}{\gamma_0}&&\\&\ddots&\\&&\frac{1}{\gamma_d}\end{pmatrix}$

Q^T : $(d+1)\times(d+1)$

X^T : $(d+1)\times N$

y : $(N\times1)$

T^{-1} : $(d+1)\times(d+1)$

$Q^T=\begin{bmatrix}q_{00}&\cdots&q_{0d}\\q_{i0}&&q_{id}\\\vdots&&\vdots\\q_{d0}&&q_{dd}\end{bmatrix}$

$X^T=\begin{bmatrix}1&1&\cdots&1\\x_1&x_2&\cdots&x_n\end{bmatrix}$

when $\lambda>0 \Rightarrow w=u=(T+\lambda)^{-1}Q^TX^Ty$

$$u=\begin{bmatrix}\gamma_0+\lambda&&\\&\gamma_i+\lambda&\\&&\gamma_d+\lambda\end{bmatrix}^{-1}Q^TX^Ty=\begin{bmatrix}\frac{1}{\gamma_0+\lambda}&&\\&\frac{1}{\gamma_i+\lambda}&\\&&\frac{1}{\gamma_d+\lambda}\end{bmatrix}Q^TX^Ty$$

$$u_i=\frac{1}{\gamma_i+\lambda}[q_{i0}\cdots q_{id}]X^Ty=\frac{1}{\gamma_i+\lambda}s$$

$$\frac{u_i}{v_i}=\frac{1}{\gamma_i+\lambda}\cdot\frac{1}{\frac{1}{\gamma_i}}=\frac{\gamma_i}{\gamma_i+\lambda}$$

6） **(a)**

6. (a) $C=\left(\frac{\sum_{n=1}^N x_ny_n}{\sum_{n=1}^N x_n^2+\lambda}\right)^2$

$$\frac{d}{dw}\left[\frac{1}{N}\sum_{n=1}^N(wx_n-y_n)^2+\frac{\lambda}{N}w^2\right]=\cancel{\frac{2}{N}}\sum_{n=1}^N(wx_n-y_n)x_n+\cancel{\frac{2}{N}}\lambda w=0$$

$$w\left[\sum_{n=1}^N x_n^2+\lambda\right]=\sum_{n=1}^N x_ny_n \Rightarrow w^*=\frac{\sum_{n=1}^N x_ny_n}{\sum_{n=1}^N x_n^2+\lambda}$$

$C=(w^*)^2$

7) **(d) $(y-0.5)^2$**

7. (d) $(y-0.5)^2$

$$\frac{d}{dy}\left[\frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2+\frac{2K}{N}\Omega(y)\right]=\frac{2}{N}\sum_{n=1}^{N}(y-y_n)+\frac{2K}{N}\Omega'(y)=0$$

$\Rightarrow Ny-\sum_{n=1}^{N}y_n+K\Omega'(y)=0 \Rightarrow \underbrace{Ny+K\Omega'(y)=\sum_{n=1}^{N}y_n}$ — ①

↳ solve this, we can get $y=$ ___ as optimal sol

We know that $y=\frac{\sum_{n=1}^{N}y_n+K}{N+2K}$ is optimal sol.

$\therefore \sum_{n=1}^{N}y_n = Ny+2Ky-K$ — ②

① = ② : $Ny+2Ky-K = Ny+K\Omega'(y) \Rightarrow \Omega'(y)=2y-1 \Rightarrow \Omega(y)=y^2-y+c$

$\therefore \Omega(y)=y^2-y+c$, where $c\in\mathbb{R}$, then we satisfies the eqn above !!

$\Omega(y)=y^2-y+c=(y-0.5)^2-\frac{1}{4}+c$

If we want indicate $\Omega(y)$ into sum of square, then we can take $c=\frac{1}{4}$

$\Rightarrow \underline{\Omega(y)=(y-0.5)^2}$ ※

8) **(b) $W^T\Gamma^2W$**

8. (b) $w^T\Gamma^2 w$

$$\min_{\tilde{w}\in\mathbb{R}^{d+1}}\frac{1}{N}\sum_{n=1}^{N}(\tilde{w}^T\Phi(x_n)-y_n)^2+\frac{\lambda}{N}(\tilde{w}^T\tilde{w})=\min_{\tilde{w}\in\mathbb{R}^{d+1}}\frac{1}{N}\sum_{n=1}^{N}(\underbrace{\tilde{w}^T\Gamma^{-1}}x_n-y_n)^2+\frac{\lambda}{N}(\tilde{w}^T\tilde{w})$$

Let $w^T=\tilde{w}^T\Gamma^{-1}\in\mathbb{R}^{d+1}$, and the above is equivalent to $\min_{w\in\mathbb{R}^{d+1}}\frac{1}{N}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\frac{\lambda}{N}\Omega(w)$

$\therefore$ find $\Omega(w)$ s.t. when we substi. $w=\Gamma^{-1}\tilde{w}$ into it,

$\Omega(\Gamma^{-1}\tilde{w})$ become $\tilde{w}^T\tilde{w}$

$w=\Gamma^{-1}\tilde{w} \Rightarrow \Gamma w=\tilde{w} \Rightarrow \tilde{w}^T\tilde{w}=w^T\Gamma\Gamma w=\underline{w^T\Gamma^2w=\Omega(w)}$

$w^T=\tilde{w}^T\Gamma^{-1} \Rightarrow w^T\Gamma=\tilde{w}^T$

9) (b) $\tilde{X} = \sqrt{\lambda B}$, $\tilde{y} = 0$

9. (b) $\tilde{X} = \sqrt{\lambda} \cdot \sqrt{B}$, $\tilde{y} = 0$

① $\frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \frac{\lambda}{N}\sum_{i=0}^{d}\beta_i w_i^2 = \frac{1}{N}\|Xw-y\|^2 + \frac{\lambda}{N}\sum_{i=0}^{d}\beta_i w_i^2 = \frac{1}{N}[w^T X^T X w - 2w^T X^T y + y^T y] + \frac{\lambda}{N}\sum_{i=0}^{d}\beta_i w_i^2$

$\frac{d}{dw}\left[\frac{1}{N}(w^T X^T X w - 2w^T X^T y + y^T y + \frac{\lambda}{N} B w^T w\right]$

$= \frac{2}{N}(X^T X w - X^T y) + \frac{2}{N}\lambda B w = 0 \Rightarrow \underline{w = (X^T X + \lambda B)^{-1} X^T y}$ —①

② $\frac{d}{dw} \frac{1}{N+K}\left[\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \sum_{n=1}^{K}(w^T \tilde{x}_n - \tilde{y}_n)^2\right]$

$= \frac{d}{dw}\frac{1}{N+K}\left[w^T X^T X w - 2w^T X^T y + y^T y + w^T \tilde{X}^T \tilde{X} w - 2w^T \tilde{X}^T \tilde{y} + \tilde{y}^T \tilde{y}\right]$

$= \frac{2}{N+K}\left[X^T X w - X^T y + \tilde{X}^T \tilde{X} w - \tilde{X}^T \tilde{y}\right] = 0 \Rightarrow \underline{w = (X^T X + \tilde{X}^T \tilde{X})^{-1}(X^T y + \tilde{X}^T \tilde{y})}$ —②

① = ② : $\underline{\tilde{y} = 0}$, $\tilde{X}^T \tilde{X} = \lambda B = \sqrt{\lambda}\sqrt{B}\sqrt{\lambda}\sqrt{B} \Rightarrow \tilde{X}^T = \underline{\underline{\tilde{X} = \sqrt{\lambda}\sqrt{B}}}$

Note : • B is diagonal matrix, $\therefore B = B^T$

• $K = d+1$, so $\dim(\tilde{X}) = \dim(X)$, $\sum_{l=0}^{d} = \sum_{k=1}^{d+1}$

10) (e) 1

10. (e) 1

$E_{loocv}(A_{majority}^{constant}) = \frac{1}{2N}[\underbrace{e_1 + e_2 + \ldots + e_N}_{N \text{ positive}} + \underbrace{\tilde{e}_1 + \tilde{e}_2 + \ldots + \tilde{e}_N}_{N \text{ negative}}]$

e_i : i-th positive example for test, N-1 positive , N negative to train ⇒ majority class : Negative

$\therefore e_i = [\![\underbrace{h_i(x_i)}_{\text{predict negative}} \neq \underbrace{y_i}_{\text{positive}}]\!] = 1$

Similarly , $\tilde{e}_i = [\![h_i(\tilde{x}_i) \neq \tilde{y}_i]\!] = 1$

$\Rightarrow \underline{\underline{E_{loocv}(A_{majority}^{const.}) = \frac{1}{2N}[2N] = 1}}$

11) (c) $\frac{2}{N}$

11. (c) $\frac{2}{N}$

$E_{LOOCV}\left(\begin{smallmatrix}\text{Decision}\\ \text{stump}\end{smallmatrix}\right) = \frac{1}{N}(e_1 + e_2 + \dots + e_N)$

≥ 2 positive, ≥ 2 negative, ∵ tightest, ∴ we choose 极端 case

choose : ① ② ③ ④ ⑤ … ⓝ

o o x x x x x x x x

then $e_1 = 0$ ∵ ② ③ ④ ; o x x …

$e_2 = 0$ or 1 ∵ ① ③ ④ ; o x x ; ②可能被分错

$e_3 = 0$ or 1 ∵ ① ② ④ ⑤ … ; o o x x ; ③可能被分错

$e_4 = 0$ ∵ ① ② ③ ⑤ ; o o x x

$\Rightarrow e_i = 0$ for $i \geq 4$

$\Rightarrow \underbrace{0}_{\frac{1}{N}\Sigma(0+\dots+0)} \leq E_{LOOCV}\left(\begin{smallmatrix}\text{Decision}\\ \text{stump}\end{smallmatrix}\right) \leq \underbrace{\frac{2}{N}}_{\frac{1}{N}\Sigma(0+1+1+0+\dots)}$ = tightest bound!

12) (e) $\sqrt{81+36\sqrt{6}}$

12. (e) $\sqrt{81+36\sqrt{6}}$

$(x_1, y_1) = (3, 0)$, $(x_2, y_2) = (\rho, 2)$, $(x_3, y_3) = (-3, 0)$, $\rho \geq 0$

constant : $h_1(x) = 1$, $h_2(x) = 0$, $h_3(x) = 1$

$h_i(x)$: the hypothesis that without point (x_i, y_i)

$\therefore E_{LOOCV}(\text{const}) = \frac{1}{3}[(1-0)^2 + (0-2)^2 + (1-0)^2] = 2$

Linear : $\frac{h_1(x)-2}{2-0} = \frac{x-\rho}{\rho+3} \Rightarrow (\rho+3)h_1(x) - \cancel{2\rho} - 6 = 2x - \cancel{2\rho} \Rightarrow h_1(x) = \frac{2}{\rho+3}x + \frac{6}{\rho+3}$

$\frac{h_2(x)-0}{0-0} = \frac{x-3}{3+3} \Rightarrow h_2(x) = 0$

$\frac{h_3(x)-2}{2-0} = \frac{x-\rho}{\rho-3} \Rightarrow (\rho-3)h_3(x) - \cancel{2\rho} + 6 = 2x - \cancel{2\rho} \Rightarrow h_3(x) = \frac{2}{\rho-3}x - \frac{6}{\rho-3}$

$E_{LOOCV}(\text{Linear}) = \frac{1}{3}\left[(h_1(x_1)-y_1)^2 + (h_2(x_2)-y_2)^2 + (h_3(x_3)-y_3)^2\right]$

$= \frac{1}{3}\left[\left(\frac{12}{\rho+3}\right)^2 + 4 + \left(\frac{-12}{\rho-3}\right)^2\right] = E_{LOOCV}(\text{const.}) = 2$

$\Rightarrow 72(\rho-3)^2 + 72(\rho+3)^2 = (\rho-3)^2(\rho+3)^2 = (\rho^2-6\rho+9)(\rho^2+6\rho+9)$

$\Rightarrow 72(\rho^2 - \cancel{6\rho} + 9) + 72(\rho^2 + \cancel{6\rho} + 9) = \rho^4 + \cancel{6\rho^3} + 9\rho^2 - \cancel{6\rho^3} - 36\rho^2 - \cancel{54\rho} + 9\rho^2 + \cancel{54\rho} + 81$

$\rho^4 - 162\rho^2 = 1215 \Rightarrow (\rho^2 - 81)^2 = 7776 \Rightarrow (\rho^2 - 81) = \pm 36\sqrt{6}$ (negative reject since ρ^2 always positive)

$\rho^2 = 81 + 36\sqrt{6} \Rightarrow \rho = \sqrt{81+36\sqrt{6}}$

13) **(d) 1/K**

13. (d) $\frac{1}{K}$

$(x_i, y_i) \overset{iid}{\sim} P$

$$\underset{D_{val}\sim P}{Var}\left[E_{val}(h)\right] = \underset{D_{val}\sim P}{Var}\left[\frac{1}{K}\sum_{i=1}^{K} err(h(x_k), y_k)\right] = \frac{1}{K^2}\underset{D_{val}\sim P}{Var}\left[\sum_{i=1}^{K} err(h(x_k), y_k)\right]$$

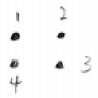
$$= \frac{1}{K^2}\sum_{i=1}^{K}\underset{D_{val}\sim P}{Var}\left(err(h(x_k), y_k)\right) \underset{(x_i, y_i)\ iid}{=} \frac{1}{K^2}\cdot K\underset{(x,y)\sim P}{Var}\left[err(h(x), y)\right]$$

$$= \frac{1}{K}\underset{(x,y)\sim P}{Var}\left[err(h(x), y)\right]$$

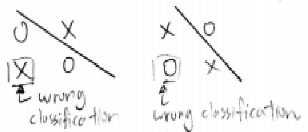
14) **(c) 2/64**

14. (c) $\frac{2}{64}$

4 vertices of rectangle in $\mathbb{R}^2$:



For ⑦-th and ⑭-th data :



output

	1	2	3	4		1	2	3	4
①	O	O	O	O	⑨	X	X	X	X
②	O	O	O	X	⑩	X	X	X	O
③	O	O	X	O	⑪	X	X	O	X
④	O	X	O	O	⑫	X	O	X	X
⑤	O	O	X	X	⑬	X	X	O	O
⑥	O	X	X	O	⑭	X	O	O	X
⑦	O	X	O	X	⑮	X	O	X	O
⑧	X	O	O	O	⑯	O	X	X	X

cannot find a line to get these combination

$\therefore \min E(w) = \frac{1}{4}\sum_{i=1}^{4}[\![h(x_i) \neq y_i]\!] = \frac{1}{4}$ for ⑦-th and ⑩-th

For the data except ⑦-th & ⑩-th, we can find a line to seperate it perfectly.

$\therefore \min E(w) = 0$ for the data except ⑦, ⑩-th

$$\Rightarrow \mathbb{E}_{y_1, y_2, y_3, y_4}\left(\min_{w\in\mathbb{R}^{d+1}} E_{in}(w)\right) = \frac{1}{16}\left[\min E_{in}(w)\big|_{①} + \dots + \min E_{in}(w)\big|_{⑯}\right] = \frac{1}{16}\left[\frac{1}{4}+\frac{1}{4}\right] = \frac{2}{64}$$

15) **(a)**

15. (a) $P = \frac{1-\varepsilon_-}{\varepsilon_+ - \varepsilon_- + 1}$

$$E_{out}(g) = \frac{1}{N}\sum_{i=1}^{N}[\![g(x_i) \neq y_i]\!] = \frac{1}{N}\sum_{\substack{g(x_i)=1\\ y_i=-1}}[\![g(x_i) \neq y_i]\!] + \frac{1}{N}\sum_{\substack{g(x_i)=-1\\ y_i=1}}[\![g(x_i) \neq y_i]\!]$$

$$= P(g(x_i)=1, y_i=-1) + P(g(x_i)=-1, y_i=1)$$

$$= P(y_i=-1)P(g(x_i)=1 \mid y_i=-1) + P(y_i=1)P(g(x_i)=-1 \mid y_i=1)$$

$$= (1-p)\varepsilon_- + p\,\varepsilon_+$$

$E_{out}(g_c) = 1-P$

$$E_{out}(g) = E_{out}(g_c) \Leftrightarrow (1-p)\varepsilon_- + p\,\varepsilon_+ = 1-P$$

$$\Rightarrow p(\varepsilon_+ - \varepsilon_- + 1) = 1-\varepsilon_- \Rightarrow P = \frac{1-\varepsilon_-}{\varepsilon_+ - \varepsilon_- + 1}$$

16) (b) -2

Relation between C & $\frac{1}{\lambda}$

Lecture: $\min_{w} \frac{\lambda}{N} w^T w + \frac{1}{N} \underbrace{\sum_{n=1}^{N} \ln \ldots}_{\square}$

$\Rightarrow \frac{d}{dw} : \frac{2\lambda}{N} w + \frac{1}{N} \frac{d\square}{dw} = 0$

$\Rightarrow w + \frac{1}{2\lambda} \frac{d\square}{dw} = 0$

Liblinear: $\min_{w} \frac{1}{2} w^T w + C \underbrace{\sum_{n=1}^{N} \ln \ldots}_{\square}$

$\Rightarrow \frac{d}{dw} : w + C \frac{d\square}{dw} = 0$

$\Rightarrow$ take $C = \frac{1}{2\lambda}$, then

function in liblinear = function in Lecture

```python
import scipy
from liblinearutil import *
import numpy as np
import math
```

```python
# Convert the .dat to List
def dat_to_list(dat_file):
    with open(dat_file, 'r') as f:
        text = f.read()

    data = text.split()  #split string into a list
    Data = []
    for i in range(int(len(data)/7)):
        K = list(data[i*7:(i+1)*7])
        Data.append(K)

    for i in range(len(Data)):
        for j in range(len(Data[0])):
            Data[i][j] = float(Data[i][j])
    return Data

# Second-order polynomial transformation
def poly_2(X):
    F = []
    k = 0
    for i in range(len(X)):
        for j in range(len(X)-k):
            F.append(X[i]*X[j+k])
        k+=1
    F = [1]+[i for i in X] + F
    return F

# Expectation Error (Ein Eval Eout ... )
def E(X,Y,W):

    def sign(J):
        if J>= 1/2 : J = 1
        else: J = -1
        return J

    Sum = 0
    for i in range(len(X)):
        h = 1/(1+math.exp(-np.array(W).dot(X[i])))
        if sign(h)!=Y[i]:
            Sum += 1
    Sum = Sum/len(X)
    return Sum
```

```python
Train = dat_to_list(dat_file=r'C://Users//USER//Desktop//hw4_train.dat.txt')
Test = dat_to_list(dat_file=r'C://Users//USER//Desktop//hw4_test.dat.txt')

Train_X = [poly_2(Train[i][0:6]) for i in range(len(Train))]
Train_Y = [Train[i][-1] for i in range(len(Train))]

Test_X = [poly_2(Test[i][0:6]) for i in range(len(Test))]
Test_Y = [Test[i][-1] for i in range(len(Test))]
```

```python
# 16  (b) -2

prob = problem(Train_Y,Train_X)

# lamba = 0.0001 => c = 5000  #10000
m1 = train(prob , parameter('-s 0 -c 5000 -e 0.000001 '))
p_label, p_acc, p_val = predict(Test_Y, Test_X, m1, '-b 1')
[W1, b1] = m1.get_decfun()
print(f'Eout = {E(Test_X,Test_Y,W1)} when lamba = 0.0001\n')

# lamba = 0.01 => c = 50  # 100
m2 = train(prob , parameter('-s 0 -c 50 -e 0.000001 '))
p_label, p_acc, p_val = predict(Test_Y, Test_X, m2, '-b 1')
[W2, b2] = m2.get_decfun()
print(f'Eout = {E(Test_X,Test_Y,W2)} when lamba = 0.01\n')

# lamba = 1 => c = 0.5   # 1
m3 = train(prob , parameter('-s 0 -c 0.5 -e 0.000001 '))
p_label, p_acc, p_val = predict(Test_Y, Test_X, m3, '-b 1')
[W3, b3] = m3.get_decfun()
print(f'Eout = {E(Test_X,Test_Y,W3)} when lamba = 1\n')

# lamba = 100 => c =0.005     # 0.01
m4 = train(prob , parameter('-s 0 -c 0.005 -e 0.000001 '))
p_label, p_acc, p_val = predict(Test_Y, Test_X, m4, '-b 1')
[W4, b4] = m4.get_decfun()
print(f'Eout = {E(Test_X,Test_Y,W4)} when lamba = 100\n')

# lamba = 10000 => c = 0.00005     # 0.0001
m5 = train(prob , parameter('-s 0 -c 0.00005 -e 0.000001 '))
p_label, p_acc, p_val = predict(Test_Y, Test_X, m5, '-b 1')
[W5, b5] = m5.get_decfun()
print(f'Eout = {E(Test_X,Test_Y,W5)} when lamba = 10000\n')

print('Best lamda = 0.01')
print('log(Best Largest lamda)= ', math.log(0.01,10)) # math.log(lamda,base)
```

```
Accuracy = 86.6667% (260/300) (classification)
Eout = 0.13333333333333333 when lamba = 0.0001

Accuracy = 87% (261/300) (classification)
Eout = 0.13 when lamba = 0.01

Accuracy = 80.6667% (242/300) (classification)
Eout = 0.19333333333333333 when lamba = 1

Accuracy = 74.3333% (223/300) (classification)
Eout = 0.25666666666666665 when lamba = 100

Accuracy = 51.6667% (155/300) (classification)
Eout = 0.48333333333333334 when lamba = 10000

Best lamda = 0.01
log(Best Largest lamda)=  -1.9999999999999996
```

17) (a) -4

```
# 17  (a) -4

prob = problem(Train_Y, Train_X)

# lamba = 0.0001 => c = 5000
m1 = train(prob , parameter('-s 0 -c 5000 -e 0.000001 '))
p_label, p_acc, p_val = predict(Train_Y, Train_X, m1, '-b 1')
[W1, b1] = m1.get_decfun()
print(f'Ein = {E(Train_X,Train_Y,W1)} when lamba = 0.0001\n')

# lamba = 0.01 => c = 50
m2 = train(prob , parameter('-s 0 -c 50 -e 0.000001 '))
p_label, p_acc, p_val = predict(Train_Y, Train_X, m2, '-b 1')
[W2, b2] = m2.get_decfun()
print(f'Ein = {E(Train_X,Train_Y,W2)} when lamba = 0.01\n')

# lamba = 1 => c = 0.5
m3 = train(prob , parameter('-s 0 -c 0.5 -e 0.000001 '))
p_label, p_acc, p_val = predict(Train_Y, Train_X, m3, '-b 1')
[W3, b3] = m3.get_decfun()
print(f'Ein = {E(Train_X,Train_Y,W3)} when lamba = 1\n')

# lamba = 100 => c = 0.005
m4 = train(prob , parameter('-s 0 -c 0.005 -e 0.000001 '))
p_label, p_acc, p_val = predict(Train_Y, Train_X, m4, '-b 1')
[W4, b4] = m4.get_decfun()
print(f'Ein = {E(Train_X,Train_Y,W4)} when lamba = 100\n')

# lamba = 10000 => c = 0.00005
m5 = train(prob , parameter('-s 0 -c 0.00005 -e 0.000001 '))
p_label, p_acc, p_val = predict(Train_Y, Train_X, m5, '-b 1')
[W5, b5] = m5.get_decfun()
print(f'Ein = {E(Train_X,Train_Y,W5)} when lamba = 10000\n')

print('Best largest lamda = 0.01 ')
print('log(Best largest lamda)= ', math.log(0.0001,10))  # math.log(lamda,base)
```

```
Accuracy = 91% (182/200) (classification)
Ein = 0.09 when lamba = 0.0001

Accuracy = 90% (180/200) (classification)
Ein = 0.1 when lamba = 0.01

Accuracy = 87% (174/200) (classification)
Ein = 0.13 when lamba = 1

Accuracy = 80.5% (161/200) (classification)
Ein = 0.195 when lamba = 100

Accuracy = 46.5% (93/200) (classification)
Ein = 0.535 when lamba = 10000

Best largest lamda = 0.01
log(Best largest lamda)=  -3.999999999999999
```

18) (e) 0.14

```
# 18  (e) 0.14
D_train_X = Train_X[0:120]
D_train_Y = Train_Y[0:120]
D_val_X = Train_X[120:]
D_val_Y = Train_Y[120:]
```

```
prob = problem(D_train_Y, D_train_X)
```

```
# lamba = 0.0001 => c = 5000
m1 = train(prob , parameter('-s 0 -c 5000 -e 0.000001 '))
print('Training Result :')
p_label, p_acc, p_val = predict(D_train_Y, D_train_X, m1, '-b 1')
print('Validation Result :')
p_label, p_acc, p_val = predict(D_val_Y, D_val_X, m1, '-b 1')
[W1, b1] = m1.get_decfun()
print(f'Eval_m1 = {E(D_val_X, D_val_Y,W1)} when lamba = 0.0001\n')

# lamba = 0.01 => c = 50
m2 = train(prob , parameter('-s 0 -c 50 -e 0.000001 '))
print('Training Result :')
p_label, p_acc, p_val = predict(D_train_Y, D_train_X, m2, '-b 1')
print('Validation Result :')
p_label, p_acc, p_val = predict(D_val_Y, D_val_X, m2, '-b 1')
[W2, b2] = m2.get_decfun()
print(f'Eval_m2 = {E(D_val_X, D_val_Y,W2)} when lamba = 0.01\n')

# lamba = 1 => c = 0.5
m3 = train(prob , parameter('-s 0 -c 0.5 -e 0.000001 '))
print('Training Result :')
p_label, p_acc, p_val = predict(D_train_Y, D_train_X, m3, '-b 1')
print('Validation Result :')
p_label, p_acc, p_val = predict(D_val_Y, D_val_X, m3, '-b 1')
[W3, b3] = m3.get_decfun()
print(f'Eval_m3 = {E(D_val_X, D_val_Y,W3)} when lamba = 1\n')

# lamba = 100 => c = 0.005
m4 = train(prob , parameter('-s 0 -c 0.005 -e 0.000001 '))
print('Training Result :')
p_label, p_acc, p_val = predict(D_train_Y, D_train_X, m4, '-b 1')
print('Validation Result :')
p_label, p_acc, p_val = predict(D_val_Y, D_val_X, m4, '-b 1')
[W4, b4] = m4.get_decfun()
print(f'Eval_m4 = {E(D_val_X, D_val_Y,W4)} when lamba = 100\n')

# lamba = 10000 => c = 0.00005
m5 = train(prob , parameter('-s 0 -c 0.00005 -e 0.000001 '))
print('Training Result :')
p_label, p_acc, p_val = predict(D_train_Y, D_train_X, m5, '-b 1')
print('Validation Result :')
p_label, p_acc, p_val = predict(D_val_Y, D_val_X, m5, '-b 1')
[W5, b5] = m5.get_decfun()
print(f'Eval_m5 = {E(D_val_X, D_val_Y,W5)} when lamba = 10000\n')

print(f'Best lamba = 0.01')
```

```
Training Result :
Accuracy = 90.8333% (109/120) (classification)
Validation Result :
Accuracy = 80% (64/80) (classification)
Eval_m1 = 0.2 when lamba = 0.0001

Training Result :
Accuracy = 90% (108/120) (classification)
Validation Result :
Accuracy = 86.25% (69/80) (classification)
Eval_m2 = 0.1375 when lamba = 0.01

Training Result :
Accuracy = 85% (102/120) (classification)
Validation Result :
Accuracy = 76.25% (61/80) (classification)
Eval_m3 = 0.2375 when lamba = 1

Training Result :
Accuracy = 80% (96/120) (classification)
Validation Result :
Accuracy = 73.75% (59/80) (classification)
Eval_m4 = 0.2625 when lamba = 100

Training Result :
Accuracy = 49.1667% (59/120) (classification)
Validation Result :
Accuracy = 42.5% (34/80) (classification)
Eval_m5 = 0.575 when lamba = 10000

Best lamba = 0.01
```

```python
# Apply w_(best lamda) to E_out    ( lamba = 0.01 => c = 100 )
p_label, p_acc, p_val = predict(Test_Y, Test_X, m2, '-b 1')
E_out = E(Test_X,Test_Y,W2)
print(f'E_out = {E_out}')
```

```
Accuracy = 85.6667% (257/300) (classification)
E_out = 0.14333333333333334
```

19) (d) 0.13

```
# 19  (d) 0.13
# best lamba = 0.01 in the previous problem
prob = problem(Train_Y, Train_X)
m = train(prob , parameter('-s 0 -c 50 -e 0.000001 '))
[W, b] = m.get_decfun()
print('Training Result :')
p_label, p_acc, p_val = predict(Train_Y, Train_X, m, '-b 1')
print('Test Result :')
p_label, p_acc, p_val = predict(Test_Y, Test_X, m, '-b 1')
E_out = E(Test_X,Test_Y,W)
print(f'E_out = {E_out}')
```

```
Training Result :
Accuracy = 90% (180/200) (classification)
Test Result :
Accuracy = 87% (261/300) (classification)
E_out = 0.13
```

20) (c) 0.12

```
# 20 (c) 0.12

# 5-fold Val set
D_5folds_X = [Train_X[40*i:40*(i+1)] for i in range(5)]
D_5folds_Y = [Train_Y[40*i:40*(i+1)] for i in range(5)]

# Training set
Train_5folds_X = [Train_X[40:200],Train_X[0:40]+Train_X[80:200],Train_X[0:80]+Train_X[120:200],
                  Train_X[0:120]+Train_X[160:200],Train_X[0:160]]
Train_5folds_Y = [Train_Y[40:200],Train_Y[0:40]+Train_Y[80:200],Train_Y[0:80]+Train_Y[120:200],
                  Train_Y[0:120]+Train_Y[160:200],Train_Y[0:160]]
```

```python
# Lamba = 0.0001 => c = 5000
E_cv = 0
for i in range(5):
    prob = problem(Train_5folds_Y[i] , Train_5folds_X[i])
    m = train(prob , parameter('-s 0 -c 5000 -e 0.000001 '))
    [W, b] = m.get_decfun()
    E_cv += E(D_5folds_X[i],D_5folds_Y[i],W)
E_cv = E_cv/5
print(f'E_cv = {E_cv} for lamda = 0.0001')

# Lamba = 0.01 => c = 50
E_cv = 0
for i in range(5):
    prob = problem(Train_5folds_Y[i] , Train_5folds_X[i])
    m = train(prob , parameter('-s 0 -c 50 -e 0.000001 '))
    [W, b] = m.get_decfun()
    E_cv += E(D_5folds_X[i],D_5folds_Y[i],W)
E_cv = E_cv/5
print(f'E_cv = {E_cv} for lamda = 0.01')

# Lamba = 1 => c = 0.5
E_cv = 0
for i in range(5):
    prob = problem(Train_5folds_Y[i] , Train_5folds_X[i])
    m = train(prob , parameter('-s 0 -c 0.5 -e 0.000001 '))
    [W, b] = m.get_decfun()
    E_cv += E(D_5folds_X[i],D_5folds_Y[i],W)
E_cv = E_cv/5
print(f'E_cv = {E_cv} for lamda = 1')

# Lamba = 100 => c = 0.005
E_cv = 0
for i in range(5):
    prob = problem(Train_5folds_Y[i] , Train_5folds_X[i])
    m = train(prob , parameter('-s 0 -c 0.005 -e 0.000001 '))
    [W, b] = m.get_decfun()
    E_cv += E(D_5folds_X[i],D_5folds_Y[i],W)
E_cv = E_cv/5
print(f'E_cv = {E_cv} for lamda = 100')

# Lamba = 10000 => c = 0.00005
E_cv = 0
for i in range(5):
    prob = problem(Train_5folds_Y[i] , Train_5folds_X[i])
    m = train(prob , parameter('-s 0 -c 0.00005 -e 0.000001 '))
    [W, b] = m.get_decfun()
    E_cv += E(D_5folds_X[i],D_5folds_Y[i],W)
E_cv = E_cv/5
print(f'E_cv = {E_cv} for lamda = 10000')
print(f'\nargmin E_val = 0.125 for lamda = 0.01')
```

```
E_cv = 0.145 for lamda = 0.0001
E_cv = 0.12 for lamda = 0.01
E_cv = 0.15500000000000003 for lamda = 1
E_cv = 0.18 for lamda = 100
E_cv = 0.5199999999999999 for lamda = 10000

argmin E_val = 0.125 for lamda = 0.01
```