1. c

以 SGD 算出 hypothesis 再求 noise

```
In [1]: import numpy as np

In [2]: steps = 1000
        repeat = 500
        sample = 100
        lr = 0.01

        def target(x):
            return np.exp(x)

        w_total = np.zeros(1)

        x = np.linspace(0, 2, sample)

        for j in range(repeat):
            w = np.zeros(1)
            for i in range(steps):
                k = np.random.randint(sample)
                w = w + lr*(target(x[k]) - w*x[k])*x[k]
            w_total += w

        print ((w_total / repeat)[0])

        3.1540241762703327

In [3]: print ('wb =', (3+np.exp(1)**2)/8)
        print ('wc =', (3+3*np.exp(1)**2)/8)
        print ('wd =', (np.exp(1)**2)/8)
        print ('we =', (3*np.exp(1)**2)/8)

        wb = 1.298632012366331
        wc = 3.1458960370989937
        wd = 0.9236320123663312
        we = 2.7708960370989937
```

2. c

第一個是一般情況，第三個不可能發生，第二個假設有一個線性可分的分布，有可能找到最佳解使 Ein=Eout=0，但如果產生的 data set 有不好的情況的話，即使是最簡單的 case 也有可能 train 出 Eout 很大的情況，所以以期望值來看 Eout 一定會大於 Ein，2 個選項 always false 故選 c

3. d

3.

$$E(X_h^T X_h) = E\left(\left[\underbrace{x_1 \cdots x_N}_{X^T} \ \underbrace{\tilde{x}_1 \cdots \tilde{x}_N}_{X^T+\varepsilon}\right]\left[x_1 \cdots x_N \ \tilde{x}_1 \cdots \tilde{x}_N\right]^T\right)$$

$$= E(X^TX + X^TX + \underbrace{X^T\varepsilon}_{0} + \underbrace{\varepsilon X}_{0} + \varepsilon^2)$$

$$= 2X^TX + N\sigma^2 I_{d+1} \qquad E\left(\left[\varepsilon_1 \cdots \varepsilon_N\right]\left[\varepsilon_1 \cdots \varepsilon_N\right]\right)$$

$$E(\mathrm{diag}\left[\varepsilon_1^2 \ \varepsilon_2^2 \ \varepsilon_3^2 \cdots \varepsilon_N^2\right])$$

$$\Rightarrow N\sigma^2 I_{d+1}$$

4. e

4. $E(X_h^T y_h) = E\left(\underbrace{\begin{bmatrix} | & & | \\ x_1 & \cdots & x_N \\ | & & | \end{bmatrix}}_{X^T} \underbrace{\begin{bmatrix} | & & | \\ \tilde{x}_1 & \cdots & \tilde{x}_N \\ | & & | \end{bmatrix}}_{X^T+\varepsilon} \begin{bmatrix} y \\ y \end{bmatrix}\right)$

$= E(X^T y + X^T y + \varepsilon y)$

$= 2X^T y$

5. d

5. $w_{reg} = (Z^T Z + \lambda I)^{-1} Z^T y$

$= (Q^T X^T X Q + \lambda I)^{-1} Q X^T y$

$X^T X = Q P Q^T$

$X^T X Q = Q P$

$Q^T X^T X Q = P = diag\{\gamma_0, \cdots, \gamma_d\}$

$Q^T X^T X Q + \lambda I = diag\{\gamma_0 + \lambda, \gamma_1 + \lambda, \cdots, \gamma_d + \lambda\}$

$(Q^T X^T X Q + \lambda I)^{-1} = diag\{(\gamma_0+\lambda)^{-1}, (\gamma_1+\lambda)^{-1}, \cdots, (\gamma_d+\lambda)^{-1}\}$

$\Rightarrow w_{reg} = diag\{(\gamma_0+\lambda)^{-1}, (\gamma_1+\lambda)^{-1}, \cdots, (\gamma_d+\lambda)^{-1}\} Q X^T y$

when $\lambda = 0$

$\Rightarrow w_{lin} = diag\{\gamma_0^{-1}, \gamma_1^{-1}, \cdots, \gamma_d^{-1}\} Q X^T y$

$\frac{u_i}{v_i} = \frac{w_{reg}}{w_{lin}} = \frac{(\gamma_i+\lambda)^{-1}}{\gamma_i^{-1}} = \frac{\gamma_i}{\gamma_i + \lambda}$

6. a

6. $\|w^*\|^2 = w_{reg}^T w_{reg}$

$$= \left[(Z^T Z + \lambda I)^{-1} Z^T y\right]^T (Z^T Z + \lambda I)^{-1} Z^T y$$

$$= y^T Z \left[(Z^T Z + \lambda I)^{-1}\right]^T (Z^T Z + \lambda I)^{-1} Z^T y$$

$$(Z^T Z + \lambda I) = P \, diag\{k_1 + \lambda, k_2 + \lambda, \cdots k_N + \lambda\} P^T$$

$$(Z^T Z + \lambda I)^{-1} = P^T diag\{(k_1 + \lambda)^{-1}, (k_2 + \lambda)^{-1}, \cdots (k_N + \lambda)^{-1}\} P$$

$$= y^T Z P^T [\quad] P P^T [\quad] P Z^T y$$

$$= y^T Z \underline{P^T} [\quad]^2 \underline{P} Z^T y = y^T Z (Z^T Z + \lambda I)^{-2} Z^T y.$$

$$Z^T Z \sim x^2$$

$$Z^T y \sim xy$$

$$(a) \quad C = \left(\frac{\sum x_n y_n}{\sum x_n^2 + \lambda}\right)^2$$

7. d

7. $\frac{1}{N} \sum 2(y - y_n) = 0$

$$2y - \frac{2}{N} \sum y = 0$$

$$\Rightarrow y = \frac{1}{N} \sum y$$

$$2y - \frac{2}{N} \sum y + \frac{4kc}{N}(y + c) = 0$$

$$y + \frac{2k}{N} y + \frac{2kc}{N} = \frac{1}{N} \sum y$$

$$y = \frac{\frac{1}{N} \sum y - \frac{2kc}{N}}{1 + \frac{2k}{N}}$$

$$\Rightarrow c = -0.5$$

8. b

$$8.\ (\tilde{w}^T \phi(x_n) - y_n + \frac{\lambda}{N}(\tilde{w}^T \tilde{w}) = (\tilde{w}^T P^{-1} x_n - y_n)^2 + \frac{\lambda}{N}(\tilde{w}^T \tilde{w})$$

$$\therefore \text{Let } \tilde{w}^* = \arg\min \frac{1}{N}\sum_{n=1}^{N}(\tilde{w}^T P^{-1} x_n - y_n)^2 + \frac{\lambda}{N}(\tilde{w}^T\tilde{w})$$

$$(P^{-1}\tilde{w})^T = \tilde{w}^T (P^{-1})^T = \tilde{w}^T P^{-1} \quad (P = P^T)$$

$$\therefore \text{Let } w^* = P^{-1}\tilde{w},\ \text{then } w^{*T} P^2 w^* = \tilde{w}^{*T} P^{-1} P^2 P^{-1} \tilde{w}^* = \tilde{w}^{*T}\tilde{w}^*$$

$$\therefore \text{if } \tilde{w}^* = \arg\min \frac{1}{N}\sum_{n=1}(\tilde{w}^T P^{-1} x_n - y_n)^2 + \frac{\lambda}{N}(\tilde{w}^T\tilde{w})$$

$$\text{then } w^* = P^{-1}\tilde{w}^* = \arg\min \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \frac{\lambda}{N}(w^T P^2 w)$$

$$\Rightarrow \Omega(w) = w^T P^2 w$$

9. b

$$9.\quad X = [x_1, \cdots, x_N]^T,\ y = [y_1, \cdots, y_N]^T$$

$$X' = \begin{bmatrix} X \\ \tilde{X} \end{bmatrix} \quad y' = \begin{bmatrix} y \\ \tilde{y} \end{bmatrix}$$

$$\Rightarrow \min \frac{1}{N+k} \| X'w - y \|^2$$

$$w = [(X')^T X]^{-1} (X')^T y$$

$$= \left(\begin{bmatrix} X \\ \tilde{X} \end{bmatrix}^T \begin{bmatrix} X \\ \tilde{X} \end{bmatrix} \right)^{-1} \begin{bmatrix} X \\ \tilde{X} \end{bmatrix}^T \begin{bmatrix} y \\ \tilde{y} \end{bmatrix}$$

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

$$\text{compare with } w_{reg} = (X^T X + \lambda B I)^{-1} X^T y$$

$$\tilde{X} = \sqrt{\lambda}\sqrt{B}\, I,\ y = 0$$

10.e

不管抽掉哪一個，選最多數的 Algorithm 都會產生把抽出的 sample 分到另一個的 hypothesis，使 error=1，每種情況都相同，平均後還是 1，故選 e

$+\ +\ -\ -$

leave one $\Rightarrow$ hypothesis

$\Rightarrow$ 分錯1個

$$\frac{2N}{2N}$$

11.c

11. First, let N samples be $x_1 < x_2 < \cdots < x_N$ ($\therefore x_1 < x_2 < 0,\ 0 < x_{N-1} < x_N$)

if x_i is left for validation and $x_j < 0 < x_{j+1}$ for some j

then $\begin{cases} i = j \\ i = j+1 \end{cases} \Rightarrow$

$\theta = \dfrac{x_{j-1} + x_{j+1}}{2} \Rightarrow$ may make wrong prediction for x_j, since $x_j < 0,\ x_{j+1} > 0 \Rightarrow \theta \overset{?}{>} 0$

$\theta = \dfrac{x_j + x_{j+2}}{2} \Rightarrow$ may make wrong prediction same reason above

for other case won't make mistake

$\therefore E_{loocv} \le \dfrac{2}{N}$

12. e

12. set $h_0(x) = w_0$

先看兩點 (x_1, y_1) (x_2, y_2)

$$E_{in} = (w_0 - y_1)^2 + (w_0 - y_2)^2 = 2w_0^2 - 2(y_1 + y_2)w_0 + (y_1^2 + y_2^2)$$

when $w_0 = \frac{y_1 + y_2}{2}$, E_{in} is min

if $(3,0)$, $(\rho, 2)$ $\Rightarrow w_0 = 1$ error $= (1-0)^2 = 1$

if $(\rho, 2)$, $(-3, 0)$ $\Rightarrow w_0 = 1$ error $= (1-0)^2 = 1$

if $(3,0)$, $(-3,0)$ $\Rightarrow w_0 = 0$ error $= (2-0)^2 = 4$

平均 $= \frac{1}{3}(1+1+4) = 2$

set $h_1(x) = w_0 + w_1 x$
(x_1, y_1) (x_2, y_2) for E_{in} min $\begin{cases} w_1 x_1 + w_0 = y_1 \\ w_1 x_2 + w_0 = y_2 \end{cases}$

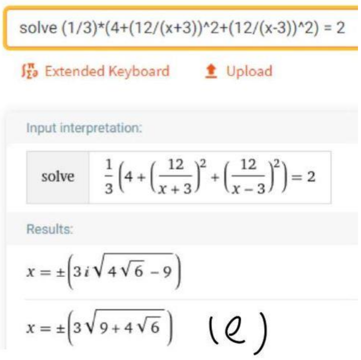
$$\Rightarrow w_1 = \frac{y_1 - y_2}{x_1 - x_2}, \quad w_0 = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$$

if $(3,0)$, $(\rho, 2)$ $\Rightarrow w_1 = \frac{-2}{3-\rho}$ $w_0 = \frac{6-0}{3-\rho}$

$$\text{error} = \left(\frac{6}{3-\rho} + \frac{6}{3-\rho}\right)^2$$

if $(\rho, 2)$, $(-3, 0)$ $\Rightarrow w_1 = \frac{2}{\rho+3}$ $w_0 = \frac{0+6}{\rho+3}$

$$\text{error} = \left(\frac{6}{\rho+3} + \frac{6}{\rho+3}\right)^2$$


solve (1/3)*(4+(12/(x+3))^2+(12/(x-3))^2) = 2

Extended Keyboard Upload

Input interpretation:

solve $\frac{1}{3}\left(4 + \left(\frac{12}{x+3}\right)^2 + \left(\frac{12}{x-3}\right)^2\right) = 2$

Results:

$x = \pm\left(3i\sqrt{4\sqrt{6} - 9}\right)$

$x = \pm\left(3\sqrt{9 + 4\sqrt{6}}\right)$

if $(3,0)$, $(-3,0)$ $\Rightarrow w_1 = 0$ $w_0 = 0$

error $= (-2)^2 = 4$

$$平均 = \frac{1}{3}\left[4 + \left(\frac{12}{\rho+3}\right)^2 + \left(\frac{12}{\rho-3}\right)^2\right]$$

solve $\frac{1}{3}\left[4 + \left(\frac{12}{\rho+3}\right)^2 + \left(\frac{12}{\rho-3}\right)^2\right] = 2$

(e)

13. d

13.

sample mean

$$Var(\bar{X}) = Var\left(\frac{x_1+\cdots+x_n}{n}\right)$$

$$= Var\left(\frac{1}{n}x_1+\frac{1}{n}x_2+\cdots+\frac{1}{n}x_n\right)$$

$$= \frac{1}{n^2}Var(x_1)+\frac{1}{n^2}Var(x_2)+\cdots+\frac{1}{n^2}Var(x_n)$$

$$= \frac{1}{n^2}[\underbrace{\sigma^2+\sigma^2+\cdots+\sigma^2}_{n}] = \frac{1}{n^2}\{n\cdot\sigma^2\}$$

$$= \frac{1}{n}\sigma^2$$

[equivalent to our $\underset{D_{val}\sim P^K}{Var}[E_{val}(h)]$]

14. e

14.

O O X X	O X O X	X X O O	X O X O
O X X X	X O X X	X X X O	X X O X
X O O O	O X O O	O O O X	O O X O
O O O O	X X X X		

$\min_{w} E_{in}(w) = 0$

∴ separable

(14 graphs)

O X X O	O X X O

$\min E_{in}(w) = 1$

(2 graphs)

$$\therefore E_{y_1,y_2,y_3,y_4}(\min E_{in}) = \frac{0+2\cdot 1}{16} = \frac{8}{64}$$

15.a

15. If $p(y=+1)=p$

then $E_{out}(g)=P(g(x)\neq y)$

$$=P(y=1)\cdot P(y(x)=-1|y=1)+P(y=-1)\cdot P(g(x)=1|y=-1)$$

$$=p\cdot\varepsilon_{+}+(1+p)\varepsilon_{-}$$

$\therefore$ If $E_{out}(g)=E_{out}(g_c)$, then $p\varepsilon_{+}+(1-p)\varepsilon_{-}=1-p$

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

Data processing for p16 to p20

```
In [1]: import numpy as np
        from numpy import savetxt
        from sklearn.preprocessing import PolynomialFeatures
        import csv
```

```
In [2]: poly = PolynomialFeatures(2, interaction_only=False, include_bias=True, order='C')

        data_train = np.genfromtxt("hw4_train.dat")
        X_train = data_train[:, :-1]
        y_train = data_train[:, -1].reshape(-1, 1)
        X_train = poly.fit_transform(X_train)
        data_test = np.genfromtxt("hw4_test.dat")
        X_test = data_test[:, :-1]
        y_test = data_test[:, -1].reshape(-1, 1)
        X_test = poly.fit_transform(X_test)
```

```
In [3]: with open('train_set.txt','a') as f:
            writer = csv.writer(f)
            for j in range(len(X_train)):
                f.write(str(y_train[j].item()) + " ")
                for i in range(len(X_train[0])):
                    f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
                f.write("\n")
```

```
In [4]: with open('test_set.txt','a') as f:
            writer = csv.writer(f)
            for j in range(len(X_test)):
                f.write(str(y_test[j].item()) + " ")
                for i in range(len(X_test[0])):
                    f.write(str(i+1) + ":" + str(X_test[j][i].item()) + " ")
                f.write("\n")
```

```
In [5]: with open('120_train_set.txt','a') as f:
            writer = csv.writer(f)
            for j in range(120):
                f.write(str(y_train[j].item()) + " ")
                for i in range(len(X_train[0])):
                    f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
                f.write("\n")
        with open('80_val_set.txt','a') as f:
            writer = csv.writer(f)
            for j in range(80):
                f.write(str(y_train[j+120].item()) + " ")
                for i in range(len(X_train[0])):
                    f.write(str(i+1) + ":" + str(X_train[j+120][i].item()) + " ")
                f.write("\n")
```

In [11]:

```python
with open('fold1_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
        f.write("\n")
with open('fold2_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j+40].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+40][i].item()) + " ")
        f.write("\n")
with open('fold3_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j+80].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+80][i].item()) + " ")
        f.write("\n")
with open('fold4_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j+120].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+120][i].item()) + " ")
        f.write("\n")
with open('fold5_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j+160].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+160][i].item()) + " ")
        f.write("\n")
```

In [10]:

```python
with open('lv_fold1_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(160):
        f.write(str(y_train[j+40].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+40][i].item()) + " ")
        f.write("\n")

with open('lv_fold2_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(40):
        f.write(str(y_train[j].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
        f.write("\n")
    for j in range(120):
        f.write(str(y_train[j+80].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+80][i].item()) + " ")
        f.write("\n")

with open('lv_fold3_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(80):
        f.write(str(y_train[j].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
        f.write("\n")
    for j in range(80):
        f.write(str(y_train[j+120].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+120][i].item()) + " ")
        f.write("\n")

with open('lv_fold4_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(120):
        f.write(str(y_train[j].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
        f.write("\n")
    for j in range(40):
        f.write(str(y_train[j+160].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j+160][i].item()) + " ")
        f.write("\n")
```

```python
with open('lv_fold5_train_set.txt','a') as f:
    writer = csv.writer(f)
    for j in range(160):
        f.write(str(y_train[j].item()) + " ")
        for i in range(len(X_train[0])):
            f.write(str(i+1) + ":" + str(X_train[j][i].item()) + " ")
        f.write("\n")
```

16. b

$$50 = \frac{1}{2 \cdot \lambda^*}$$

$$\lambda^* = 10^{-2}$$

```
(base) [r08245012@cluster liblinear]$ ./train -s 0 -c 50 -e 0.000001 train_set.txt
init f 6.931e+03 |g| 3.955e+03
iter  1 f 3.842e+03 |g| 9.887e+02 CG   4 step_size 1.00e+00
iter  2 f 2.901e+03 |g| 4.447e+02 CG   6 step_size 1.00e+00
iter  3 f 2.693e+03 |g| 1.326e+02 CG   7 step_size 1.00e+00
iter  4 f 2.667e+03 |g| 7.797e+01 CG   8 step_size 1.00e+00
iter  5 f 2.666e+03 |g| 7.387e+00 CG   2 step_size 1.00e+00
iter  6 f 2.666e+03 |g| 6.305e+00 CG   2 step_size 1.00e+00
iter  7 f 2.662e+03 |g| 5.281e-01 CG   9 step_size 1.00e+00
iter  8 f 2.662e+03 |g| 5.527e-01 CG   5 step_size 1.00e+00
iter  9 f 2.662e+03 |g| 5.571e-01 CG   2 step_size 1.00e+00
iter 10 f 2.662e+03 |g| 1.649e+00 CG   8 step_size 1.00e+00
iter 11 f 2.662e+03 |g| 9.726e-02 CG   3 step_size 1.00e+00
iter 12 f 2.662e+03 |g| 2.778e-02 CG   9 step_size 1.00e+00
iter 13 f 2.662e+03 |g| 1.687e-02 CG   3 step_size 1.00e+00
iter 14 f 2.662e+03 |g| 2.997e-02 CG   9 step_size 1.00e+00
iter 15 f 2.662e+03 |g| 5.162e-03 CG   5 step_size 1.00e+00
iter 16 f 2.662e+03 |g| 1.192e-04 CG  11 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict test_set.txt train_set.txt.model prediction
Accuracy = 87% (261/300)
```

17. e

$$5 \times 10^{-5} = \frac{1}{2 \cdot \lambda^*}$$

$$\lambda^* = 10^{4}$$

```
(base) [r08245012@cluster liblinear]$ ./train -s 0 -c 0.00005 -e 0.000001 train_set.txt
init f 6.931e-03 |g| 3.955e-03
iter  1 f 6.924e-03 |g| 4.088e-08 CG   2 step_size 1.00e+00
iter  2 f 6.924e-03 |g| 5.435e-11 CG   2 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict test_set.txt train_set.txt.model prediction
Accuracy = 51.6667% (155/300)
```

18. e

c=50

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 120_train_set.txt
init f 4.159e+03 |g| 2.864e+03
iter  1 f 2.281e+03 |g| 7.205e+02 CG   3 step_size 1.00e+00
iter  2 f 1.728e+03 |g| 2.224e+02 CG   7 step_size 1.00e+00
iter  3 f 1.632e+03 |g| 5.710e+01 CG   7 step_size 1.00e+00
iter  4 f 1.622e+03 |g| 3.800e+01 CG   8 step_size 1.00e+00
iter  5 f 1.622e+03 |g| 2.226e+00 CG   2 step_size 1.00e+00
iter  6 f 1.622e+03 |g| 1.728e-01 CG   9 step_size 1.00e+00
iter  7 f 1.622e+03 |g| 5.976e-01 CG   8 step_size 1.00e+00
iter  8 f 1.622e+03 |g| 8.347e-02 CG   3 step_size 1.00e+00
iter  9 f 1.622e+03 |g| 1.574e-02 CG   9 step_size 1.00e+00
iter 10 f 1.622e+03 |g| 7.303e-03 CG   3 step_size 1.00e+00
iter 11 f 1.622e+03 |g| 6.187e-03 CG   3 step_size 1.00e+00
iter 12 f 1.622e+03 |g| 1.909e-05 CG  11 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict 80_val_set.txt 120_train_set.txt.model prediction
Accuracy = 86.25% (69/80)
```

最接近的為 e 選項

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 120_train_set.txt
init f 4.159e+03 |g| 2.864e+03
iter  1 f 2.281e+03 |g| 7.205e+02 CG   3 step_size 1.00e+00
iter  2 f 1.728e+03 |g| 2.224e+02 CG   7 step_size 1.00e+00
iter  3 f 1.632e+03 |g| 5.710e+01 CG   7 step_size 1.00e+00
iter  4 f 1.622e+03 |g| 3.800e+01 CG   8 step_size 1.00e+00
iter  5 f 1.622e+03 |g| 2.226e+00 CG   2 step_size 1.00e+00
iter  6 f 1.622e+03 |g| 1.728e-01 CG   9 step_size 1.00e+00
iter  7 f 1.622e+03 |g| 5.976e-01 CG   8 step_size 1.00e+00
iter  8 f 1.622e+03 |g| 8.347e-02 CG   3 step_size 1.00e+00
iter  9 f 1.622e+03 |g| 1.574e-02 CG   9 step_size 1.00e+00
iter 10 f 1.622e+03 |g| 7.303e-03 CG   3 step_size 1.00e+00
iter 11 f 1.622e+03 |g| 6.187e-03 CG   3 step_size 1.00e+00
iter 12 f 1.622e+03 |g| 1.909e-05 CG  11 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict test_set.txt 120_train_set.txt.model prediction
Accuracy = 85.6667% (257/300)
```

19.d

0.13

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 train_set.txt
init f 6.931e+03 |g| 3.955e+03
iter  1 f 3.842e+03 |g| 9.887e+02 CG   4 step_size 1.00e+00
iter  2 f 2.901e+03 |g| 4.447e+02 CG   6 step_size 1.00e+00
iter  3 f 2.693e+03 |g| 1.326e+02 CG   7 step_size 1.00e+00
iter  4 f 2.667e+03 |g| 7.797e+01 CG   8 step_size 1.00e+00
iter  5 f 2.666e+03 |g| 7.387e+00 CG   2 step_size 1.00e+00
iter  6 f 2.666e+03 |g| 6.305e+00 CG   2 step_size 1.00e+00
iter  7 f 2.662e+03 |g| 5.281e-01 CG   9 step_size 1.00e+00
iter  8 f 2.662e+03 |g| 5.527e-01 CG   5 step_size 1.00e+00
iter  9 f 2.662e+03 |g| 5.571e-01 CG   2 step_size 1.00e+00
iter 10 f 2.662e+03 |g| 1.649e+00 CG   8 step_size 1.00e+00
iter 11 f 2.662e+03 |g| 9.726e-02 CG   3 step_size 1.00e+00
iter 12 f 2.662e+03 |g| 2.778e-02 CG   9 step_size 1.00e+00
iter 13 f 2.662e+03 |g| 1.687e-02 CG   3 step_size 1.00e+00
iter 14 f 2.662e+03 |g| 2.997e-02 CG   9 step_size 1.00e+00
iter 15 f 2.662e+03 |g| 5.162e-03 CG   5 step_size 1.00e+00
iter 16 f 2.662e+03 |g| 1.192e-04 CG  11 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict test_set.txt train_set.txt.model prediction
Accuracy = 87% (261/300)
```

20.

c=50

(0.15 + 0.2 + 0.05 + 0.15 + 0.05) / 5 = 0.12

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 lv_fold1_train_set.txt
init f 5.545e+03 |g| 3.384e+03
iter  1 f 3.100e+03 |g| 8.031e+02 CG   3 step_size 1.00e+00
iter  2 f 2.315e+03 |g| 4.649e+02 CG   6 step_size 1.00e+00
iter  3 f 2.144e+03 |g| 1.404e+02 CG   6 step_size 1.00e+00
iter  4 f 2.118e+03 |g| 1.560e+01 CG   7 step_size 1.00e+00
iter  5 f 2.115e+03 |g| 4.904e-01 CG   9 step_size 1.00e+00
iter  6 f 2.115e+03 |g| 6.078e-02 CG   9 step_size 1.00e+00
iter  7 f 2.115e+03 |g| 1.243e-02 CG   9 step_size 1.00e+00
iter  8 f 2.115e+03 |g| 3.836e-03 CG   9 step_size 1.00e+00
iter  9 f 2.115e+03 |g| 8.844e-04 CG   9 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict fold1_train_set.txt lv_fold1_train_set.txt.model prediction
Accuracy = 85% (34/40)
```

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 lv_fold2_train_set.txt
init f 5.545e+03 |g| 2.645e+03
iter  1 f 3.035e+03 |g| 7.273e+02 CG   4 step_size 1.00e+00
iter  2 f 2.139e+03 |g| 2.814e+02 CG   6 step_size 1.00e+00
iter  3 f 1.924e+03 |g| 1.078e+02 CG   6 step_size 1.00e+00
iter  4 f 1.888e+03 |g| 4.157e+01 CG   8 step_size 1.00e+00
iter  5 f 1.884e+03 |g| 3.806e+00 CG   8 step_size 1.00e+00
iter  6 f 1.883e+03 |g| 6.837e-01 CG   5 step_size 1.00e+00
iter  7 f 1.883e+03 |g| 1.090e+00 CG   3 step_size 1.00e+00
iter  8 f 1.883e+03 |g| 8.929e-02 CG   9 step_size 1.00e+00
iter  9 f 1.883e+03 |g| 3.134e-02 CG   6 step_size 1.00e+00
iter 10 f 1.883e+03 |g| 3.763e-02 CG   9 step_size 1.00e+00
iter 11 f 1.883e+03 |g| 5.451e-03 CG   3 step_size 1.00e+00
iter 12 f 1.883e+03 |g| 6.300e-05 CG  12 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict fold2_train_set.txt lv_fold2_train_set.txt.model prediction
Accuracy = 80% (32/40)
```

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 lv_fold3_train_set.txt
init f 5.545e+03 |g| 3.053e+03
iter  1 f 3.130e+03 |g| 7.362e+02 CG   4 step_size 1.00e+00
iter  2 f 2.463e+03 |g| 3.532e+02 CG   6 step_size 1.00e+00
iter  3 f 2.340e+03 |g| 8.159e+01 CG   6 step_size 1.00e+00
iter  4 f 2.325e+03 |g| 2.722e+01 CG   8 step_size 1.00e+00
iter  5 f 2.323e+03 |g| 1.842e+01 CG   8 step_size 1.00e+00
iter  6 f 2.323e+03 |g| 3.631e+00 CG   3 step_size 1.00e+00
iter  7 f 2.323e+03 |g| 4.113e+00 CG   2 step_size 1.00e+00
iter  8 f 2.323e+03 |g| 1.867e+00 CG   8 step_size 1.00e+00
iter  9 f 2.323e+03 |g| 1.914e-01 CG   4 step_size 1.00e+00
iter 10 f 2.323e+03 |g| 2.748e-01 CG   2 step_size 1.00e+00
iter 11 f 2.323e+03 |g| 6.637e-02 CG   7 step_size 1.00e+00
iter 12 f 2.323e+03 |g| 1.284e-02 CG   9 step_size 1.00e+00
iter 13 f 2.323e+03 |g| 1.498e-02 CG   3 step_size 1.00e+00
iter 14 f 2.323e+03 |g| 1.437e-02 CG   9 step_size 1.00e+00
iter 15 f 2.323e+03 |g| 7.705e-04 CG   7 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict fold3_train_set.txt lv_fold3_train_set.txt.model prediction
Accuracy = 95% (38/40)
```

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 lv_fold4_train_set.txt
init f 5.545e+03 |g| 3.519e+03
iter  1 f 3.076e+03 |g| 8.538e+02 CG   3 step_size 1.00e+00
iter  2 f 2.233e+03 |g| 2.818e+02 CG   6 step_size 1.00e+00
iter  3 f 2.053e+03 |g| 1.183e+02 CG   6 step_size 1.00e+00
iter  4 f 2.025e+03 |g| 2.565e+01 CG   8 step_size 1.00e+00
iter  5 f 2.023e+03 |g| 9.052e+00 CG   8 step_size 1.00e+00
iter  6 f 2.023e+03 |g| 7.030e-01 CG   5 step_size 1.00e+00
iter  7 f 2.023e+03 |g| 1.601e+00 CG   2 step_size 1.00e+00
iter  8 f 2.022e+03 |g| 4.516e+00 CG   8 step_size 1.00e+00
iter  9 f 2.022e+03 |g| 8.310e-02 CG   3 step_size 1.00e+00
iter 10 f 2.022e+03 |g| 9.272e-03 CG   9 step_size 1.00e+00
iter 11 f 2.022e+03 |g| 6.167e-03 CG   3 step_size 1.00e+00
iter 12 f 2.022e+03 |g| 1.751e-02 CG   9 step_size 1.00e+00
iter 13 f 2.022e+03 |g| 1.289e-03 CG   8 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict fold4_train_set.txt lv_fold4_train_set.txt.model prediction
Accuracy = 85% (34/40)
```

```
(base) [r08245012@cluster liblinear]$ ./train -c 50 -s 0 -e 0.000001 lv_fold5_train_set.txt
init f 5.545e+03 |g| 3.288e+03
iter  1 f 3.191e+03 |g| 8.386e+02 CG   3 step_size 1.00e+00
iter  2 f 2.439e+03 |g| 3.373e+02 CG   7 step_size 1.00e+00
iter  3 f 2.287e+03 |g| 9.570e+01 CG   7 step_size 1.00e+00
iter  4 f 2.270e+03 |g| 8.030e+01 CG   8 step_size 1.00e+00
iter  5 f 2.269e+03 |g| 4.215e+00 CG   2 step_size 1.00e+00
iter  6 f 2.268e+03 |g| 1.201e+00 CG   9 step_size 1.00e+00
iter  7 f 2.268e+03 |g| 3.627e-01 CG   4 step_size 1.00e+00
iter  8 f 2.268e+03 |g| 7.097e-01 CG   2 step_size 1.00e+00
iter  9 f 2.268e+03 |g| 3.552e-01 CG   3 step_size 1.00e+00
iter 10 f 2.268e+03 |g| 3.674e-01 CG   9 step_size 1.00e+00
iter 11 f 2.268e+03 |g| 8.845e-02 CG   3 step_size 1.00e+00
iter 12 f 2.268e+03 |g| 1.134e-01 CG   3 step_size 1.00e+00
iter 13 f 2.268e+03 |g| 2.120e-02 CG   9 step_size 1.00e+00
iter 14 f 2.268e+03 |g| 2.094e-02 CG   3 step_size 1.00e+00
iter 15 f 2.268e+03 |g| 5.245e-04 CG  10 step_size 1.00e+00
(base) [r08245012@cluster liblinear]$ ./predict fold5_train_set.txt lv_fold5_train_set.txt.model prediction
Accuracy = 95% (38/40)
```