1

(c)

$$E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} w^T w$$

$$\Downarrow$$

$$\nabla E_{aug}(w) = \nabla E_{in}(w) + \frac{2\lambda}{N} w(t)$$

$$w(t+1) = w(t) - \eta\left(\nabla E_{in}(w) + \frac{2\lambda}{N} w(t)\right)$$

$$= \left(1 - \eta \cdot \frac{2\lambda}{N}\right) w(t) - \eta \nabla E_{in}(w)$$

$$\Downarrow$$

$$\zeta = 1 - \eta \cdot \frac{2\lambda}{N}$$

2

(b)

(1) $\min_{w \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} (w - y_n)^2 + \frac{\lambda}{N} w^2$

$$||$$

(2) $\min_{w \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} (w - y_n)^2$ subject to $w^2 \le C$, $C = (w^*)^2$

w^* is the optimal solution of (1)

$\lambda > 0 \Rightarrow$ (1) is covex function

$\Rightarrow$ optimal is at $\nabla(1) = 0$

w* is the solution

$\Downarrow$

$$\nabla(1) = \frac{1}{N} \sum_{n=1}^{N} (2w - 2y_n) + \frac{2\lambda}{N} w = 0$$

$$\Downarrow$$

$$\sum_{n=1}^{N} (w^* - y_n) + \lambda w^* = 0$$

$$\Rightarrow \quad N w^* + \lambda w^* = \sum_{n=1}^{N} y_n$$

$$w^* = \frac{\sum_{n=1}^{N} y_n}{N + \lambda}$$

$$(w^*)^2 = \left(\frac{\sum_{n=1}^{N} y_n}{N + \lambda} \right)^2$$

3 $\quad \Phi(x) = Vx = \begin{bmatrix} v_1 & & & 0 \\ & v_2 & & \\ & & \ddots & \\ 0 & & & v_n \end{bmatrix} x \qquad v_i > 0$

(d)

let $w' = [v_1 w_1 \ v_2 w_2 \cdots v_n w_n]$

$$\min_{\tilde{w}} \frac{1}{N} \sum_{n=1}^{N} (\tilde{w}^T V x_n - y_n)^2 + \frac{\lambda}{N} (\tilde{w}^T \tilde{w})$$

$$\Rightarrow \min_{w'} \frac{1}{N} \sum_{n=1}^{N} (w'^T x_n - y_n)^2 + \frac{\lambda}{N} (w'^T (V^{-1})^2 w')$$

$$[w_1 \cdots w_n] \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = [v_1 w_1 \cdots v_n w_n] \begin{bmatrix} \frac{1}{v_1^2} & & \\ & \ddots & \\ & & \frac{1}{v_n^2} \end{bmatrix} \begin{bmatrix} v_1 w_1 \\ \vdots \\ v_n w_n \end{bmatrix}$$

We can see w' as new w

so $\Gamma = (V^{-1})^2$

4

(a)

$$E\left[\frac{1}{N}\sum_{n=1}^{N}\left(w^T(x_n+\epsilon)-y_n\right)^2\right]$$

$(w^T x_n - y_n)^2$

$$=E\left[\frac{1}{N}\sum_{n=1}^{N}\left((w^T x_n)^2+(w^T\epsilon)^2+y_n^2+2w^T x_n w^T\epsilon - 2w^T x_n y_n - 2w^T\epsilon y_n\right)\right]$$

$$=E\left[\frac{1}{N}\sum_{n=1}^{N}(w^T x_n-y_n)^2\right]+E\left[\frac{1}{N}\sum_{n=1}^{N}\left((w^T\epsilon)^2+2w^T x_n w^T\epsilon-2w^T\epsilon y_n\right)\right]$$

$$=\frac{1}{N}\sum_{n=1}^{N}(w^T x_n-y_n)^2+E\left[\frac{1}{N}\sum_{n=1}^{N}(w^T\epsilon)^2\right]+E\left[\frac{1}{N}\sum_{n=1}^{N}(2w^T x_n w^T-2y_n w^T)\epsilon\right]$$

$Var(x)=E[x^2]-E[x]^2$

$\Rightarrow E[x^2]=Var(x)+E[x]^2$

$$\sigma^2 w^T w=\sigma^2|w|_2^2 \qquad\qquad 0$$

$$\sigma^2|w|_2^2=\frac{N\sigma^2}{N}|w|_2^2$$

5

(d)

(1) $\dfrac{\left(\sum_{n=1}^{N} y_n\right)+a}{N+ak}$ is optimal solution of

(2) $\min_{y \in R} \dfrac{1}{N} \sum_{n=1}^{N} (y-y_n)^2 + \dfrac{ak}{N}\Omega(y)$

(2) 的微分在 $y=(1)$ 等於 0

$$\nabla(2) = \frac{1}{N}\sum_{n=1}^{N} 2(y-y_n) + \frac{ak}{N}\Omega(y)'$$

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

$$\frac{\left(\sum_{n=1}^{N} y_n\right)+a}{N+ak} - \frac{1}{N}\sum_{n=1}^{N} y_n + \frac{ak}{2N}\Omega((1))' = 0$$

$$\Omega((1))' = \frac{2}{ak}\sum_{n=1}^{N} y_n - \frac{2N\left(\sum_{n=1}^{N} y_n\right)+2Na}{ak(N+ak)}$$

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

把 (1) 轉成 y

$$\Omega(y)' = 2y - \frac{2ka}{k(N+ak)} - \frac{2N}{k(N+ak)} = 2y - \frac{2ka+2N}{k(N+ak)}$$

$\frac{2}{k}$

$\Downarrow$ 積分

$$y^2 - \frac{2}{k}y + C$$

$\frac{1}{k^2}$

常數不影響最佳化 因可為任意值，設定成可以配方法

$$\Rightarrow \left(y - \frac{1}{k}\right)^2$$

6 (d) minimizer of $\tilde{E}_{aug}(w) \Rightarrow \nabla \tilde{E}_{aug}(w) = 0$

$$\nabla \tilde{E}_{aug}(w) = \nabla \tilde{E}_{in}(w) + \frac{2\lambda}{N} w = 0$$

$$\underbrace{\nabla E_{in}(w^*)}_{0} + 0 + \underbrace{\nabla \frac{1}{2}(w-w^*)^T H (w-w^*)}_{IH(w-w^*) + IH^T(w-w^*)}$$

$$= (H + H^T)(w - w^*) \qquad \boxed{H^T = H}$$

$$= 2H(w - w^*)$$

$$\Rightarrow H(w - w^*) + \frac{\lambda}{N} w = 0$$

$$\Rightarrow Hw + \frac{\lambda}{N} w = Hw^*$$

$$\Rightarrow \left(H + \frac{\lambda}{N} I\right) w = Hw^*$$

$$\Rightarrow w = \left(H + \frac{\lambda}{N} I\right)^{-1} H w^*$$

7 N個 x N個 o

if 選到 o out

(a) train data 變成 xxxxx oooo

classifier 輸出 o

$o = o \Rightarrow E_{loocv}(A_{minority}) = 0$

if 選到 x out

train data 變成 xxxx ooooo

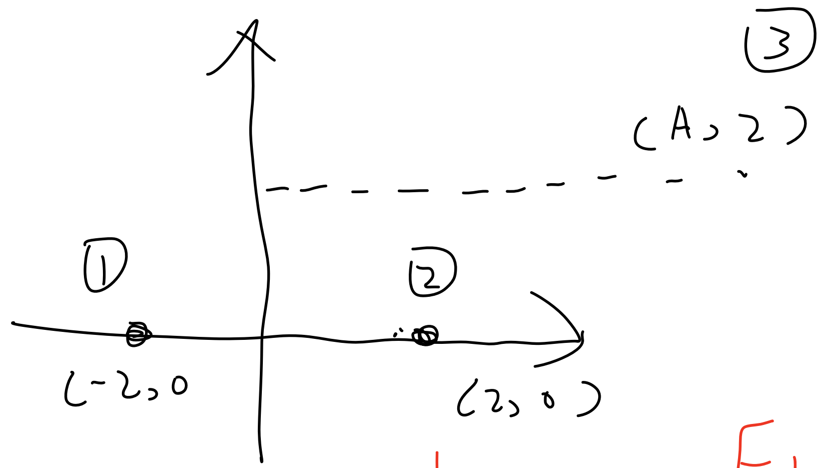
classifier 輸出 x

$x = x \Rightarrow E_{loocv}(A_{minority}) = 0$

8

$A \geq 0$

(1)



	$h(x)$	E_{loocv}
① ②	$y=0$	4
① ③	$y=1$	1
② ③	$y=1$	1

等於

① ②	$y=0$	4
① ③	$y=\frac{4}{2+A}+\frac{2}{2+A}x$	$\left(\frac{8}{2+A}\right)^2$

② ③ $y = \frac{-4}{-2+A} + \frac{2}{-2+A} \times \left(\frac{-8}{-2+A}\right)^2$

$$4+1+1 = 4 + \left(\frac{8}{2+A}\right)^2 + \left(\frac{-8}{-2+A}\right)^2$$

$$2 = \frac{64}{4+4A+A^2} + \frac{64}{4-4A+A^2}$$

$$(4+4A+A^2)(4-4A+A^2) = 32(2A^2+8)$$

$$A^4 - 8A^2 + 16 = 64A^2 + 256$$

$$A^4 - 72A^2 - 240 = 0$$

$$\Rightarrow A \simeq 8.6713 \simeq 8.5$$

9

(b) $\bar{y} = \frac{1}{N-K} \sum_{n=1}^{N-K} y_n$

$$E\left(\frac{1}{K} \sum_{n=N-K+1}^{N} (y_n - \bar{y})^2\right) =$$

$$\underbrace{E\left(\frac{1}{K} \sum_{n=N-K+1}^{N} y_n^2\right)}_{\sigma^2} + E\left(\frac{1}{K} \sum_{n=N-K+1}^{N} 2 y_n \bar{y}\right) + \underbrace{E\left(\frac{1}{K} \sum_{n=N-K+1}^{N} \bar{y}^2\right)}$$

因為 y_n 是 i.i.d

$$\frac{2}{K} \sum_{n=N-K+1}^{N} E(y_n) \underbrace{E(\bar{y})}_{0} = 0$$

$$E(\bar{y}^2) = E\left(\frac{\left(\sum_{n=1}^{N-k} y_n\right)^2}{(N-k)^2}\right)$$

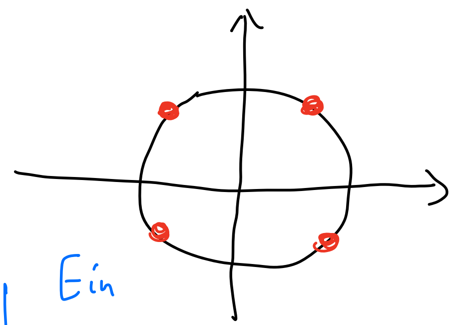
$$= \frac{(N-k)\sigma^2}{(N-k)^2}$$

$$= \frac{\sigma^2}{N-k}$$

$$= \sigma^2 + \frac{\sigma^2}{N-k}$$

10

(a)



dichotomy				Ein
O	O	O	O	0
O	O	O	X	0
O	O	X	O	0
O	X	O	O	0
X	O	O	O	0
O	O	X	X	0
O	X	O	X	$\frac{1}{4}$
X	O	O	X	0
O	X	X	O	0
X	O	X	O	$\frac{1}{4}$
X	X	X	O	0
X	X	O	X	0
X	O	X	X	0
O	X	X	X	0
X	X	X	X	0

$$\frac{\frac{1}{4}+\frac{1}{4}}{16} = \frac{1}{32}$$

11)

(b)

$$E_{out}(g) = P(g(x) = -1 \mid y = +1)\, P(y = +1) + P(g(x) = +1 \mid y = -1)\, P(y = -1)$$

$$E_{out}(g_-) = p = (\epsilon_+)p + (\epsilon_-)(1-p)$$

$$= (\epsilon_+)p + (\epsilon_-) - (\epsilon_-)p$$

$$\Rightarrow (1 - \epsilon_+ + \epsilon_-)p = \epsilon_-$$

$$\Rightarrow \quad p = \frac{\epsilon_-}{\epsilon_- - \epsilon_+ + 1}$$

12 (d)

13 (b)

14 (e)

15 (e)

16 (c)

$c = \frac{1}{\lambda}$

```python
import numpy as np
from liblinear.liblinearutil import *

def transform_fullorder(x_data, output):
    for i in range(len(x_data)):
        output.append([])
        data_len = len(x_data[i])
        output[i].append(1.0)

        for j in range(data_len):
            output[i].append(x_data[i][j])

        for j in range(data_len):
            for k in range(data_len)[j:]:
                output[i].append(x_data[i][j]*x_data[i][k])

        for j in range(data_len):
            for k in range(data_len)[j:]:
                for l in range(data_len)[k:]:
                    output[i].append(x_data[i][j]*x_data[i][k]*x_data[i][l])

def main():
    train_x = []
    train_y = []
    test_x = []
    test_y = []
    train_size = 0
    test_size = 0

    with open('hw4_train.dat.txt', 'r') as f:
        while True:
            line = f.readline()
            if line == '\n' or len(line) == 0:
                break
            x = [float(i) for i in line.split()[:-1]]
            train_x.append(x)
            train_y.append(float(line.split()[-1]))
            train_size+=1

    with open('hw4_test.dat.txt', 'r') as f:
        while True:
            line = f.readline()
            if line == '\n' or len(line) == 0:
                break
            x = [float(i) for i in line.split()[:-1]]
            test_x.append(x)
            test_y.append(float(line.split()[-1]))
            test_size+=1

    train_X = []
    train_Y = train_y.copy()
    test_X = []
    test_Y = test_y.copy()

    transform_fullorder(train_x, train_X)
    cv_err =0.0
    for i in range(5):
        prob = problem(train_Y[:i*40]+train_Y[(i+1)*40:], train_X[:i*40]+train_X[(i+1)*40:])
        param = parameter('-s 0 -c 0.01 -e 0.000001 -q')
        m = train(prob, param)
        p_label, p_acc, p_val = predict(train_Y[i*40:(i+1)*40], train_X[i*40:(i+1)*40], m, '')
        cv_err+=p_acc[0]
    print(1-cv_err/500.0)

    transform_fullorder(test_x, test_X)
    p_label, p_acc, p_val = predict(test_Y, test_X, m, '-q')
```

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