Machine Learning Homework 4

資工碩一 r09922055 陳柏妤

1. **(c)**

square error between $[0,\ 2] = \int_0^2 (wx - e^x)^2 dx = \int_0^2 (w^2x^2 - 2wxe^x + e^{2x})dx$

$= \frac{1}{3}w^2x^3 - 2wxe^x + 2we^x + \frac{1}{2}e^{2x}\ |_0^2 = \frac{8}{3}w^2 - 4we^2 + 2we^2 + \frac{1}{2}e^4 - 2w - \frac{1}{2}$

想要：square error 對 w 微分 $= 0$

$\Rightarrow \frac{16}{3}w - 4e^2 + 2e^2 - 2 = 0 \Rightarrow w = (2e^2 + 2)\frac{3}{16} = \frac{3 + 3e^2}{8}$

magnitude of deterministic noise $= |\, e^x - \frac{3 + 3e^2}{8}x \,|$ （$f(x)$ 和 $h(x)$ 的差）

2. **(b)**

設 $h* = arg \min_{h \in H} \ E_{out}(h)$

・證 $\mathbb{E}_D[E_{out}(h*)] \le \mathbb{E}_D[E_{out}(A(D))]$：

因為 $h*$ 是 E_{out} 最小的 hypothesis，所以 $E_{out}(h*) \le E_{out}(A(D))$

$\Rightarrow \mathbb{E}_D[E_{out}(h*)] \le \mathbb{E}_D[E_{out}(A(D))]$

・證 $\mathbb{E}_D[E_{in}(A(D))] \le \mathbb{E}_D[E_{out}(h*)]$：

因為 $\mathbb{E}_D[E_{in}(h*)] = \mathbb{E}_D[E_{out}(h*)]$

(固定 $h*$ ，in-sample 和 out-sample 是來自一樣的 data distribution，因此期望值會相同)

且 $A(D)$ 是 E_{in} 最小的 hypothesis $\Rightarrow \mathbb{E}_D[E_{in}(A(D))] \le \mathbb{E}_D[E_{in}(h*)]$

所以 $\mathbb{E}_D[E_{in}(A(D))] \le \mathbb{E}_D[E_{in}(h*)] = \mathbb{E}_D[E_{out}(h*)]$

綜上所述，$\mathbb{E}_D[E_{in}(A(D))] \le \mathbb{E}_D[E_{in}(h*)] = \mathbb{E}_D[E_{out}(h*)] \le \mathbb{E}_D[E_{out}(A(D))]$

$\Rightarrow \mathbb{E}_D[E_{in}(A(D))] \le \mathbb{E}_D[E_{out}(A(D))]$ （第三點 always wrong）

3. **(d)**

$$X_h = \underbrace{\begin{bmatrix} | & | & \dots & | & | & | & \dots & | \\ x_1 & x_2 & \dots & x_N & x_1 & x_2 & \dots & x_N \\ | & | & \dots & | & | & | & \dots & | \end{bmatrix}}_{\tilde{X}} + \underbrace{\begin{bmatrix} | & | & \dots & | & | & | & \dots & | \\ 0 & 0 & \dots & 0 & \epsilon & \epsilon & \dots & \epsilon \\ | & | & \dots & | & | & | & \dots & | \end{bmatrix}}_{K}$$

$X_h = \tilde{X} + K$

$\mathbb{E}[X_h^T X_h] = \mathbb{E}[(\tilde{X} + K)^T(\tilde{X} + K)]$

$= \mathbb{E}[\tilde{X}^T\tilde{X}] + \mathbb{E}[\tilde{X}^T K] + \mathbb{E}[K^T\tilde{X}] + \mathbb{E}[K^T K]$

（因為 $\mathbb{E}[K] = \mathbf{0}$，所以 $\mathbb{E}(\tilde{X}^T K)$ 和 $\mathbb{E}(K^T\tilde{X})$ 等於 $\mathbf{0}$）

$= 2X^TX + \mathbb{E}[N\epsilon^2] = 2X^TX + N(\sigma^2 \cdot I_{d+1} - (\mathbb{E}[\epsilon])^2)$ （$\mathbb{E}[\epsilon]$ 等於 $\mathbf{0}$）

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

4. **(e)**

承上題，$X_h = \underbrace{\begin{bmatrix} | & | & \cdots & | & | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_N & x_1 & x_2 & \cdots & x_N \\ | & | & \cdots & | & | & | & \cdots & | \end{bmatrix}}_{\tilde{X}} + \underbrace{\begin{bmatrix} | & | & \cdots & | & | & | & \cdots & | \\ 0 & 0 & \cdots & 0 & \epsilon & \epsilon & \cdots & \epsilon \\ | & | & \cdots & | & | & | & \cdots & | \end{bmatrix}}_{K}$

$X_h = \tilde{X} + K$

$\mathbb{E}[X_h^T y_h] = \mathbb{E}[(\tilde{X} + K)^T(y_h)]$

$= \mathbb{E}[\tilde{X}^T y_h] + \mathbb{E}[K^T y_h]$（因為 $\mathbb{E}[K] = \mathbf{0}$，所以 $\mathbb{E}[K^T y_h]$ 等於 $\mathbf{0}$）

$= 2X^T y$

5. **(d)**

求 $\frac{1}{N}(Zw - y)^T(Zw - y) + \frac{\lambda}{N} w^T w$ 最小值：

$$\frac{\partial \frac{1}{N}(Zw - y)^T(Zw - y) + \frac{\lambda}{N} w^T w}{\partial(w)} = \frac{2}{N}(Z^T Z w - Z^T y) + \frac{2\lambda}{N} w = 0$$

$$v = (Z^T Z)^{-1} Z^T y = ((XQ)^T (XQ))^{-1} (XQ)^T y = (Q^T X^T (XQ))^{-1} (XQ)^T y$$
$$= (Q^T (Q \Gamma Q^T) Q)^{-1} (XQ)^T y = (\Gamma)^{-1} (XQ)^T y$$
$$u = (Z^T Z + \lambda I)^{-1} Z^T y = ((XQ)^T (XQ) + \lambda I)^{-1} (XQ)^T y = (Q^T X^T (XQ) + \lambda I)^{-1} (XQ)^T y$$
$$= (Q^T (Q \Gamma Q^T) Q + \lambda I)^{-1} (XQ)^T y = (\Gamma + \lambda I)^{-1} (XQ)^T y$$

$\Gamma v = (\Gamma + \lambda I) u$

$\Rightarrow \gamma_i v_i = (\gamma_i + \lambda) u_i$

$\Rightarrow \frac{u_i}{v_i} = \frac{\gamma_i}{\gamma_i + \lambda}$

6. **(a)**

求 $\frac{1}{N} \sum_{n=1}^{N} (w \cdot x_n - y_n)^2 + \frac{\lambda}{N} w^2$ 的最小值：

$$\frac{\partial \frac{1}{N} \sum_{n=1}^{N} (w \cdot x_n - y_n)^2 + \frac{\lambda}{N} w^2}{\partial w} = \frac{\partial \frac{1}{N} \sum_{n=1}^{N} (w^2 x_n^2 - 2 w x_n y_n + y_n^2) + \frac{\lambda}{N} w^2}{\partial w}$$

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

$$\Rightarrow w = \frac{\sum_{n=1}^{N} x_n y_n}{(\sum_{n=1}^{N} x_n^2) + \lambda}$$

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

7. **(d)**

求 $\frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2+\frac{2K}{N}\Omega(y)$ 的最小值：

$$\frac{\partial \frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2+\frac{2K}{N}\Omega(y)}{\partial y}=\frac{\partial \frac{1}{N}\sum_{n=1}^{N}(y^2-2yy_n+y_n^2)+\frac{2K}{N}\Omega(y)}{\partial y}$$

$$=\frac{2}{N}\sum_{n=1}^{N}(y-y_n)+\frac{2K}{N}\Omega'(y)=0$$

by 選項可知 $\Omega(y)$ 為二次式，$\Omega'(y)$ 可表示為 $2y+c$，因此：

$$\Rightarrow \frac{2}{N}\sum_{n=1}^{N}(y-y_n)+\frac{2K}{N}\Omega'(y)=\frac{2}{N}\sum_{n=1}^{N}(y-y_n)+\frac{2K}{N}(2y+c)=0$$

$$\Rightarrow y=\frac{(\sum_{n=1}^{N}y_n)-Kc}{N+2K}=\frac{(\sum_{n=1}^{N}y_n)+K}{N+2K}$$ （by 上式結果與題目）

$$\Rightarrow c=-1,\ \Omega'(y)=2y-1$$

$$\Rightarrow \Omega(y)=y^2-y+(0.5)^2=(y-0.5)^2$$

8. **(b)**

$\min\limits_{\tilde{w}\in\mathbb{R}_{d+1}}\frac{1}{N}\sum_{n=1}^{N}(\tilde{w}^T\Gamma^{-1}x_n-y_n)^2+\frac{\lambda}{N}(\tilde{w}^T\tilde{w})$ is equivalent to

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

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

$$\Rightarrow \Omega(w)=\tilde{w}^T\tilde{w}=w^T\Gamma(w^T\Gamma)^T=w^T\Gamma\Gamma^Tw=w^T\Gamma^2w$$

9. **(b)**

・求 $\dfrac{1}{N}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\dfrac{\lambda}{N}\sum_{i=0}^{d}\beta_i w_i^2$ 最小值：

$$\frac{\partial \frac{1}{N}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\frac{\lambda}{N}\sum_{i=0}^{d}\beta_i w_i^2}{\partial w}=\frac{\partial \frac{1}{N}(w^TX^TXw-2w^TX^Ty+y^Ty)+\frac{\lambda}{N}w^T\beta w}{\partial w}$$

$$=\frac{2}{N}(X^TXw-X^Ty)+\frac{2\lambda}{N}w^T\beta=0$$

$$\Rightarrow w=(X^TX+\lambda\beta)^{-1}X^Ty$$

・求 $\dfrac{1}{N+K}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\sum_{k=1}^{K}(w^T\tilde{x}_k-\tilde{y}_k)^2$ 最小值：

$$\frac{\partial \frac{1}{N+K}\left(\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\sum_{k=1}^{K}(w^T\tilde{x}_k-\tilde{y}_k)^2\right)}{\partial w}$$

$$=\frac{\partial \frac{1}{N+K}(w^TX^TXw-2w^TX^Ty+y^Ty+w^T\tilde{X}^T\tilde{X}w-2w^T\tilde{X}^T\tilde{y}+\tilde{y}^T\tilde{y})}{\partial w}$$

$$=\frac{2}{N+K}(X^TXw-X^Ty+\tilde{X}^T\tilde{X}w-\tilde{X}^T\tilde{y})=0$$

$$\Rightarrow w=(X^TX+\tilde{X}^T\tilde{X})^{-1}(X^Ty+\tilde{X}^T\tilde{y})$$

by 以上兩點，$(X^TX+\lambda\beta)^{-1}X^Ty=(X^TX+\tilde{X}^T\tilde{X})^{-1}(X^Ty+\tilde{X}^T\tilde{y})\Rightarrow\begin{cases}\lambda\beta=\tilde{X}^T\tilde{X}\\ \tilde{X}^T\tilde{y}=\mathbf{0}\end{cases}$

$\Rightarrow \tilde{X}=\sqrt{\lambda}\cdot\sqrt{\beta}$ (size=(d+1)(d+1)) , $\tilde{y}=\mathbf{0}$

10. **(e)**

・如果選出來的 x_n 是正的：

training data 會變成「N-1 筆 positive data」＋「N 筆 negative data」

$\Rightarrow$ classifier 總是預測為 negative $\Rightarrow E_{val}^{(n)}(\bar{g}_n)=e_n=1$

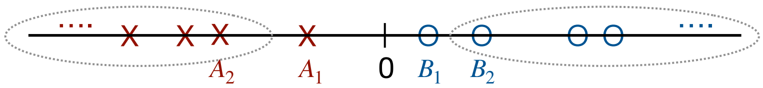
・如果選出來的 x_n 是負的：

training data 會變成「N 筆 positive data」＋「N-1 筆 negative data」

$\Rightarrow$ classifier 總是預測為 positive $\Rightarrow E_{val}^{(n)}(\bar{g}_n)=e_n=1$

$$E_{loocv}(A_{majority})=\frac{1}{N}\sum_{n=1}^{N}e_n=1$$

11. **(c)**



如上圖，考慮離原點最近的四個點，座標依序為 A_2、A_1、B_1、B_2

・若 A_2 或其他座標更小的點（左邊灰虛線處）被 leave out，當 $\theta = \dfrac{A_1 + B_1}{2}$ 時 E_{in} 最小

$\Rightarrow E_{val}^{(n)}(\bar{g}_n) = e_n = 0 \ \ (n \leq A_2)$

・若 A_1 被 leave out，當 $\theta = \dfrac{A_2 + B_1}{2}$ 時 E_{in} 最小

$\Rightarrow$ 若 $A_1 > \theta$，$E_{val}^{(A_1)}(\bar{g_{A_1}}) = e_{A_1} = 1$

・若 B_1 被 leave out，當 $\theta = \dfrac{A_1 + B_2}{2}$ 時 E_{in} 最小

$\Rightarrow$ 若 $B_1 < \theta$，$E_{val}^{(B_1)}(\bar{g_{B_1}}) = e_{B_1} = 1$

・若 B_2 或其他座標更大的點（右邊灰虛線處）被 leave out，當 $\theta = \dfrac{A_1 + B_1}{2}$ 時 E_{in} 最小

$\Rightarrow E_{val}^{(n)}(\bar{g}_n) = e_n = 0 \ \ (n \geq B_2)$

$$E_{loocv} = \frac{1}{N}\sum_{n=1}^{N} e_n \leq \frac{2}{N}$$

12. **(e)**

・constant

leave out $(3, 0) : h(x) = (2+0)/2 = 1 \Rightarrow E_{sqr} = (0-1)^2 = 1$

leave out $(\rho, 2) : h(x) = (0+0)/2 = 0 \Rightarrow E_{sqr} = (2-0)^2 = 4$

leave out $(-3, 0) : h(x) = (0+2)/2 = 1 \Rightarrow E_{sqr} = (0-1)^2 = 1$

・linear

leave out $(3, 0) : h(x) = \dfrac{6}{\rho+3} + \dfrac{2}{\rho+3}x \Rightarrow E_{sqr} = (\dfrac{12}{\rho+3} - 0)^2 = \dfrac{144}{(\rho+3)^2}$

leave out $(\rho, 2) : h(x) = 0 + 0x \Rightarrow E_{sqr} = (0-2)^2 = 4$

leave out $(-3, 0) : h(x) = \dfrac{-6}{\rho-3} + \dfrac{2}{\rho-3}x \Rightarrow E_{sqr} = (\dfrac{-12}{\rho-3} - 0)^2 = \dfrac{144}{(\rho-3)^2}$

want $\dfrac{1+4+1}{3} = \dfrac{\frac{144}{(\rho+3)^2} + 4 + \frac{144}{(\rho-3)^2}}{3}$

$$\Rightarrow \frac{2\rho^2 + 18}{(\rho^2+6\rho+9)(\rho^2-6\rho+9)} = \frac{1}{72} \Rightarrow \rho^4 - 162\rho^2 - 1215 = 0$$

$$\Rightarrow \rho^2 = \frac{162 + 72\sqrt{6}}{2} \Rightarrow \rho = \sqrt{81 + 36\sqrt{6}}$$

13. **(d)**

已知 $E_{val}(h) = \dfrac{1}{K}\sum_{n=1}^{K} err(h(x_n), y_n)$

$Varience[E_{val}(h)] = Varience[\dfrac{1}{K}\sum_{n=1}^{K} err(h(x_n), y_n)]$

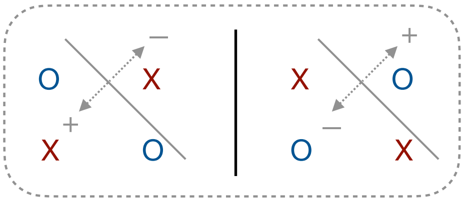
{by $Var[aX] = E[(ax)^2] - (E[ax])^2 = E[a^2x^2] - (aE[x])^2 = a^2E[x^2] - a^2(E[x])^2$
$= a^2(E[x^2] - (E[x])^2) = a^2(Var[X])$}

$$= \frac{Varience[\sum_{n=1}^{K} err(h(x_n), y_n)]}{K^2} = \frac{K(Varience[err(h(x), y)])}{K^2}$$

$$= \frac{Varience[err(h(x), y)]}{K}$$

14. **(c)**

共有 $2^4 = 16$ 種 y 的排列方式，其中只有兩種情況的 $E_{in}(w) \neq 0$（如下圖）：



此時 $\min E_{in}(w) = \dfrac{1}{4}$

$$\mathbb{E}_{y_1,y_2,y_3,y_4}(\min_{w\in\mathbb{R}^{2+1}} E_{in}(w)) = \frac{2\cdot\frac{1}{4} + 14\cdot 0}{16} = \frac{2}{64}$$

15. **(a)**

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

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

$\Rightarrow p = \dfrac{1-\epsilon_-}{\epsilon_+ - \epsilon_- + 1}$

16. **(b)** -2

17. **(a)** -4

18. **(e)** 0.143

19. **(d)** 0.13

20. **(c)** 0.12

In [1]:
```
import numpy as np
from liblinearutil import *
```

In [2]:
```
def readfile(filename):
    X = []
    Y = []
    for lines in open(filename).readlines():
        temp = lines.strip().split()
        x = [1]
        for i in range(6):
            x.append(float(temp[i]))
        for i in range(6):
            for j in range(6):
                if i<=j:
                    x.append(float(temp[i])*float(temp[j]))
        X.append(x)
        Y.append(float(temp[-1]))
#     X = np.asarray(X)
#     Y = np.asarray(Y)
    return X, Y
```

In [3]:
```
trainfile = "../../hw4_train.dat.txt"
testfile = "../../hw4_test.dat.txt"
X, Y = readfile(trainfile)
Xt, Yt = readfile(testfile)
```

In [4]:
```
##16, 17
l = [10**(-4), 10**(-2), 10**0, 10**2, 10**4]
max_out_acc = 0
l_out = -1
max_in_acc = 0
l_in = -1
for i in range(len(l)):
    C = 1/(2*l[i])
    param = '-s 0 -c ' + str(C) + ' -e 0.000001'
    m = train(Y, X, param)
    p_out_labels, p_out_acc, p_out_vals = predict(Yt, Xt, m)
    if p_out_acc[0] >= max_out_acc:
        max_out_acc = p_out_acc[0]
        l_out = i
    p_in_labels, p_in_acc, p_in_vals = predict(Y, X, m)
    if p_in_acc[0] >= max_in_acc:
        max_in_acc = p_in_acc[0]
        l_in = i
print("16.: ", np.log10(l[l_out]))
print("17.: ", np.log10(l[l_in]))
```

```
Accuracy = 86.6667% (260/300) (classification)
Accuracy = 91% (182/200) (classification)
Accuracy = 87% (261/300) (classification)
Accuracy = 90% (180/200) (classification)
Accuracy = 80.6667% (242/300) (classification)
Accuracy = 87% (174/200) (classification)
Accuracy = 74.3333% (223/300) (classification)
Accuracy = 80.5% (161/200) (classification)
Accuracy = 51.6667% (155/300) (classification)
Accuracy = 46.5% (93/200) (classification)
16.:  -2.0
17.:  -4.0
```

In [5]:
```
##18
l = [10**(-4), 10**(-2), 10**0, 10**2, 10**4]
X_train = X[:120]
Y_train = Y[:120]
X_val = X[120:]
Y_val = Y[120:]
max_val_acc = 0
l_val = -1
for i in range(len(l)):
    C = 1/(2*l[i])
    param = '-s 0 -c ' + str(C) + ' -e 0.000001'
    m = train(Y_train, X_train, param)
    p_val_labels, p_val_acc, p_val_vals = predict(Y_val, X_val, m)
    if p_val_acc[0] >= max_val_acc:
        max_val_acc = p_val_acc[0]
        l_val = i
        best_m = m
p_labels, p_acc, p_vals = predict(Yt, Xt, best_m)
print("18.: ", (100-p_acc[0])*0.01)
```

```
Accuracy = 80% (64/80) (classification)
Accuracy = 86.25% (69/80) (classification)
Accuracy = 76.25% (61/80) (classification)
Accuracy = 73.75% (59/80) (classification)
Accuracy = 42.5% (34/80) (classification)
Accuracy = 85.6667% (257/300) (classification)
18.:  0.14333333333333328
```

In [6]:
```
##19
##retrain by lambda*
best_C = 1/(2*l[l_val])
param = '-s 0 -c ' + str(best_C) + ' -e 0.000001'
m = train(Y, X, param)
p_labels, p_acc, p_vals = predict(Yt, Xt, m)
print("19.: ", (100-p_acc[0])*0.01)
```

```
Accuracy = 87% (261/300) (classification)
19.:  0.13
```

In [7]:

```
##20
l = [10**(-4), 10**(-2), 10**0, 10**2, 10**4]
min_Ecv = 10
for i in range(len(l)):
    Ecv = 0
    C = 1/(2*l[i])
    param = '-s 0 -c ' + str(C) + ' -e 0.000001'
    for j in range(5):
        X_train = X[:40*j] + X[40*(j+1):]
        Y_train = Y[:40*j] + Y[40*(j+1):]
        X_val = X[40*j:40*(j+1)]
        Y_val = Y[40*j:40*(j+1)]
#         print(np.shape(X_train), ":", np.shape(X_val))
        m = train(Y_train, X_train, param)
        p_val_labels, p_val_acc, p_val_vals = predict(Y_val, X_val,
m)
        Ecv += (100-p_val_acc[0])*0.01
    Ecv /= 5
    if Ecv <= min_Ecv:
        min_Ecv = Ecv
print("20.: ", min_Ecv)
```

```
Accuracy = 87.5% (35/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 82.5% (33/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 92.5% (37/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 75% (30/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 42.5% (17/40) (classification)
Accuracy = 65% (26/40) (classification)
Accuracy = 47.5% (19/40) (classification)
Accuracy = 40% (16/40) (classification)
Accuracy = 45% (18/40) (classification)
20.:  0.12
```

In [ ]: