Machine Learning Homework 4

1

由題目推得

$$
\begin{aligned}
&E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} w^T w \\
&w(t+1) \leftarrow w(t) - \eta \nabla E_{aug}(w(t)) \\
&w(t+1) \leftarrow w(t) - \eta (\nabla E_{in}(w(t)) + \frac{2\lambda}{N} w(t)) \\
&w(t+1) \leftarrow (1 - \frac{2\eta\lambda}{N}) w(t) - \eta \nabla E_{in}(w(t))
\end{aligned}
$$

答案為 [c].

2

由 lagrange multiplier 的性質得到 $\nabla \frac{1}{N} \sum_{n=1}^{N} (w - y_n)^2 + \frac{\lambda}{N} w^2$ 的 w 滿足 $w^2 = C$，所以可推得

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

答案為 [b].

3

對上下兩式微分取得 $\hat{w}, w$

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

$$\begin{aligned}
&\nabla(\frac{1}{N}\sum_{n=1}^{N}(\hat{w}^T\Phi(x_n)-y_n)^2+\frac{\lambda}{N}(\hat{w}^T\hat{w}))\\
&=\frac{2}{N}\sum_{n=1}^{N}(\Phi(x_n))^2\hat{w}-\Phi(x_n)y_n+\frac{2\lambda}{N}\hat{w}\\
&=\frac{2}{N}\sum_{n=1}^{N}(\Phi(x_n)^2+\frac{\lambda}{N})\hat{w}-\Phi(x_n)y_n=0
\end{aligned}$$

$$\hat{w}=\frac{\sum_{n=1}^{N}Vx_ny_n}{\sum_{n=1}^{N}(Vx_n)^2+\frac{\lambda}{N}}$$

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

$$\begin{aligned}
&\nabla(\frac{1}{N}\sum_{n=1}^{N}(w^Tx_n-y_n)^2+\frac{\lambda}{N}(w^TUw))\\
&=\frac{2}{N}\sum_{n=1}^{N}(x_n^2w-x_ny_n)+\frac{2\lambda}{N}Uw\\
&=\frac{2}{N}\sum_{n=1}^{N}(x_n^2+\frac{\lambda U}{N})w-x_ny_n=0
\end{aligned}$$

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

由此可推斷出 $U=(V^{-1})^2$，使得 $\hat{w}=w$ 達到最小值。

答案為 [d].

4

將上式期望值之結果拆開得到

$$
\begin{aligned}
&\mathbb{E}[\frac{1}{N}\sum_{n=1}^{N}(w^T\Phi(x_n)-y_n)^2] \\
&= \frac{1}{N}\mathbb{E}[\sum_{n=1}^{N}(w^T\Phi(x_n))^2 - 2w^T\Phi(x_n)y_n + y_n^2] \\
&= \frac{1}{N}(\mathbb{E}[\sum_{n=1}^{N}(w^T(x_n+\epsilon))^2] - 2\mathbb{E}[\sum_{n=1}^{N}(w^T(x_n+\epsilon))] + \mathbb{E}[\sum_{n=1}^{N}y_n^2]) \\
&= \frac{1}{N}(\sum_{n=1}^{N}((w^Tx)^2 + \sigma^2\|w\|^2) - 2\sum_{n=1}^{N}w^Tx_n + \sum_{n=1}^{N}y_n^2) \\
&= \sigma^2\|w\|^2 + \frac{1}{N}\sum_{n=1}^{N}(x^Tx_n - y_n)^2
\end{aligned}
$$

可推得 $\lambda = N\sigma^2$。 答案為 [a].

5

對下式微分取得 y

$$
\nabla\frac{1}{N}\sum_{n=1}^{N}(y-y_n)^2 + \frac{\alpha k}{N}\Omega(y) = 2y - \frac{2}{N}\sum_{n=1}^{N}y_n + \frac{\alpha k}{N}\frac{\partial\Omega(y)}{\partial y} = 0
$$

$$
\rightarrow Ny - \sum_{n=1}^{N}y_n + \frac{\alpha k}{2}\frac{\partial\Omega(y)}{\partial y} = 0
$$

設 $\frac{\partial\Omega(y)}{\partial y} = ay + b$，推得

$$
\begin{aligned}
(N + \frac{\alpha k}{2}a)y &= (\sum_{n=1}^{N}y_n - \frac{\alpha k}{2}b) \\
y &= \frac{\sum_{n=1}^{N}y_n - \frac{\alpha k}{2}b}{N + \frac{\alpha k}{2}a}
\end{aligned}
$$

由 $y = \frac{\sum_{n=1}^{N}y_n + \alpha}{N + \alpha k}$ 推得

$$
\begin{aligned}
a = 2, b &= -\frac{2}{k} \\
\frac{\partial\Omega(y)}{\partial y} &= 2y - \frac{2}{k} \\
\Omega(y) &= (y - \frac{1}{k})^2
\end{aligned}
$$

答案為 [d].

6

由題目推得

$$\begin{aligned}
\nabla \hat{E}_{in} &= \nabla E_{in}(w^*) + 0 + H(w - w^*) \\
\nabla \hat{E}_{aug} &= \nabla \hat{E}_{in} + \frac{2\lambda}{N} I w \\
&= 0 + H(w - w^*) + \frac{2\lambda}{N} I w = 0 \\
w &= (H + \frac{2\lambda}{N} I)^{-1} H w^*
\end{aligned}$$

答案為 [b]

7

無論 validation data 的那一筆資料為 1 或 -1，皆會使得該種類剩 $N-1$ 筆 training data，少於另一種類的 N 筆，故必定被 $A_{minority}$ 正確預測，所以推得 $E_{loocv}(A_{minority}) = 0$。
答案為 [a].

8

列舉出移除不同筆 data 後最小化 square error 的 hypothesis: constant model for $(2,0),(\rho,2)$: $h(x)=1$

constant model for $(\rho,2),(-2,0)$: $h(x)=1$

constant model for $(2,0),(-2,0)$: $h(x)=0$

linear model for $(2,0),(\rho,2)$: $h(x)=\frac{2}{\rho-2}x-\frac{4}{\rho-2}$

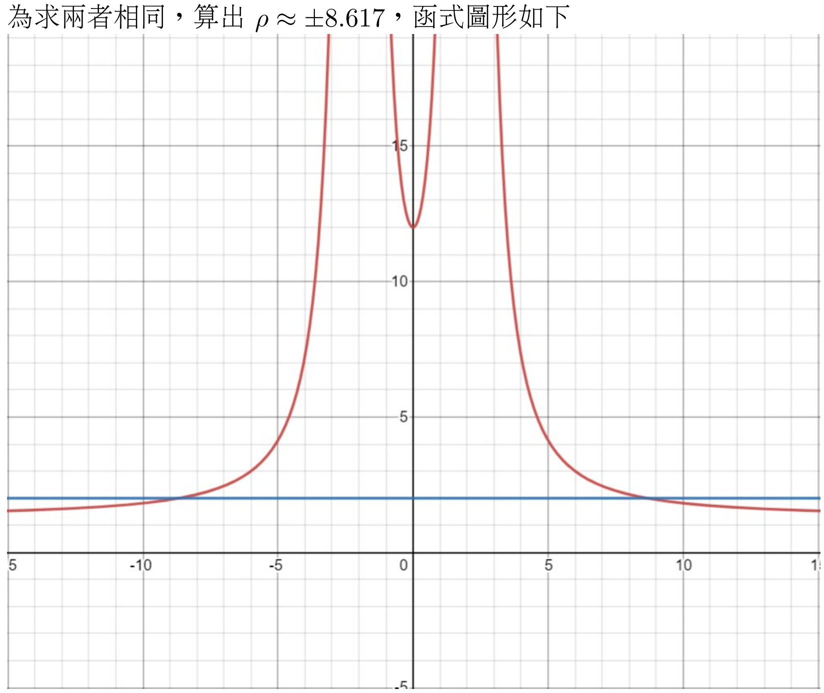
linear model for $(\rho,2),(-2,0)$: $h(x)=\frac{2}{\rho+2}x+\frac{4}{\rho+2}$

linear model for $(2,0),(-2,0)$: $h(x)=0$

計算各自的 E_{loocv}:

$$constant : E_{loocv} = \frac{1}{3}(1+1+4) = 2$$
$$linear : E_{loocv} = \frac{1}{3}((\frac{-8}{\rho-2})^2 + (\frac{8}{\rho+2})^2 + 4)$$

為求兩者相同，算出 $\rho \approx \pm 8.617$，函式圖形如下



答案為 [c].

9

由題目推得

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

$$\begin{aligned}
\mathbb{E}(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\bar{y})^2) &= \frac{1}{K}\mathbb{E}(\sum_{n=N-K+1}^{N}(y_n^2-2y_n\bar{y}+\bar{y}^2)) \\
&= \frac{1}{K}(\mathbb{E}(\sum_{n=N-K+1}^{N}y_n^2)-2\mathbb{E}(\sum_{n=N-K+1}^{N}y_n\bar{y})+K\mathbb{E}(\bar{y}^2)) \\
&= \frac{1}{K}(K\sigma^2-2\mathbb{E}(\sum_{n=N-K+1}^{N}y_n)\mathbb{E}(\bar{y})+K(\frac{1}{N-K})^2\mathbb{E}(\sum_{n=1}^{N-K}y_n^2)) \\
&= \frac{1}{K}(K\sigma^2-2*0*0+K*\frac{1}{N-K}\sigma^2) = (1+\frac{1}{N-K})\sigma^2
\end{aligned}$$

答案為 [b].

10

指定該 unit circle 上任意一點為 x_1，並順時針依序為 x_1, x_2, x_3, x_4，而2D perceptron 無法正確的 case 只有兩種，也就是 $x_1 = 1, x_2 = -1, x_3 = 1, x_4 = -1$ 及 $x_1 = -1, x_2 = 1, x_3 = -1, x_4 = 1$，這兩個 case 的 $\min_{w\in\mathbb{R}^2+1} E_{in}(w) = \frac{1}{4}$，故可算出

$$\mathbb{E}_{y_1,y_2,y_3,y_4}(\min_{w\in\mathbb{R}^2+1} E_{in}(w)) = \frac{1}{16}(\frac{1}{4}+\frac{1}{4}) = \frac{1}{32}$$

答案為 [a].

11

由題目推得

$$\begin{aligned}
E_{out}(g) &= p\epsilon_+(1-p)\epsilon_- \\
E_{out}(g_-) &= p \\
p &= p\epsilon_+ + (1-p)\epsilon_- \\
(1-\epsilon_+ + \epsilon_-)p &= \epsilon_- \\
p &= \frac{\epsilon_-}{\epsilon_- - \epsilon_+ + 1}
\end{aligned}$$

答案為 [b].

12, 13, 14, 15, 16

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

#========================================basic function=======================================

def E_01(y, predict):
    return np.sum(y != predict) / y.shape[0]

def calLambda(lambdaValue):
    return - round(math.log(float(lambdaValue) * 2, 10), 0)

#========================================Transform============================================

def ThirdOrderPolynomialTransform(data):
    data = np.hstack((np.ones((data.shape[0], 1)), data))
    #print(transformedData.shape) # 1 + 6= 7

    transformedData = np.ones((data.shape[0], 1))

    for i in range(data.shape[1]):
        for j in range(i, data.shape[1]):
            for k in range(max(1, j), data.shape[1]):
                newData = np.zeros((data.shape[0], 1))

                for l in range(data.shape[0]):
                    newData[l][0] = data[l][i] * data[l][j] * data[l][k]

                transformedData = np.hstack((transformedData, newData))

    #print(transformedData.shape[0]) # 1 + (6) + (6 + 15) + (6 + 15 * 2 + 20) = 84

    return transformedData

#========================================V folds==============================================

def VfoldsSplit(x_folds, y_folds, val):
    x_spl_val = x_folds[val]
    x_spl_train = [x_folds[i] for i in range(len(x_folds)) if i != val]
    x_spl_train = np.reshape(x_spl_train, (-1, x_folds[0].shape[1]))
    y_spl_val = y_folds[val]
    y_spl_train = [y_folds[i] for i in range(len(y_folds)) if i != val]
    y_spl_train = np.reshape(y_spl_train, -1)

    return x_spl_train, x_spl_val, y_spl_train, y_spl_val

#========================================liblinear============================================

def FindParam(x_train, y_train, x_val, y_val, x_test, y_test, params):
    prob = problem(y_train, x_train)
    minEval, minPar, Eout, lambdaValue = 1, 0, 1, ''
    for par in params:
        param = parameter(par)
        model = train(prob, param)
        p_lab, p_acc, p_val = predict(y_train, x_train, model, '-q')
        #print('Ein', E_01(y_train, p_lab))

        p_lab, p_acc, p_val = predict(y_val, x_val, model, '-q')
        #print('Eval', E_01(y_val, p_lab))

        if E_01(y_val, p_lab) <= minEval:
            minEval = E_01(y_val, p_lab)
            minPar = par
            lambdaValue = par[8:-15]

        p_lab, p_acc, p_val = predict(y_test, x_test, model, '-q')
        #print('Eout', E_01(y_test, p_lab))

    param = parameter(minPar)
    model = train(prob, param)
    p_lab, p_acc, p_val = predict(y_test, x_test, model, '-q')
    Eout = E_01(y_test, p_lab)

    return minEval, lambdaValue, Eout

#============================================================================================
```

```python
def main():
    trainData = np.loadtxt('hw4_train.dat')
    testData = np.loadtxt('hw4_test.dat')
    x_train, y_train = np.hsplit(trainData, [-1])
    x_test, y_test = np.hsplit(testData, [-1])

    x_train = ThirdOrderPolynomialTransform(x_train)
    y_train = np.reshape(y_train, -1)
    x_test = ThirdOrderPolynomialTransform(x_test)
    y_test = np.reshape(y_test, -1)

    p12, p13, p14, p15, p16 = True, True, True, True, True
    pStr = ['-s 0 -c 5000 -e 0.000001 -q',
            '-s 0 -c 50 -e 0.000001 -q',
            '-s 0 -c 0.5 -e 0.000001 -q',
            '-s 0 -c 0.005 -e 0.000001 -q',
            '-s 0 -c 0.00005 -e 0.000001 -q']

    if p12:
        minEout, lambdaValue, Eout = FindParam(x_test, y_test, x_test, y_test, x_test, y_test, pStr)
        print('Problem 12', 'lambda =', calLambda(lambdaValue), 'min Eout =', minEout)

    if p13:
        minEin, lambdaValue, Ein = FindParam(x_train, y_train, x_train, y_train, x_train, y_train,
pStr)
        print('Problem 13', 'lambda =', calLambda(lambdaValue), 'min Ein =', minEin)

    if p14 or p15:
        x_spl_train, x_spl_val = x_train[:120], x_train[120:]
        y_spl_train, y_spl_val = y_train[:120], y_train[120:]
        minEval, lambdaValue, Eout = FindParam(x_spl_train, y_spl_train, x_spl_val, y_spl_val, x_test,
y_test, pStr)

        if p14:
            print('Problem 14', 'lambda =', calLambda(lambdaValue), 'Eout =', Eout)

        if p15:
            prob = problem(y_train, x_train)
            param = parameter('-s 0 -c ' + lambdaValue + ' -e 0.000001 -q')
            model = train(prob, param)
            p_lab, p_acc, p_val = predict(y_test, x_test, model, '-q')
            Eout = E_01(y_test, p_lab)
            print('Problem 15', 'lambda =', calLambda(lambdaValue), 'Eout =', Eout)

    if p16:
        x_folds = [x_train[:40], x_train[40:80], x_train[80:120], x_train[120:160], x_train[160:]]
        y_folds = [y_train[:40], y_train[40:80], y_train[80:120], y_train[120:160], y_train[160:]]
        minErr, lambdaValue = 1, ''

        for par in pStr:
            E_val = 0
            for val in range(len(x_folds)):
                x_spl_train, x_spl_val, y_spl_train, y_spl_val = VfoldsSplit(x_folds, y_folds, val)
                prob = problem(y_spl_train, x_spl_train)
                param = parameter(par)
                model = train(prob, param)
                p_lab, p_acc, p_val = predict(y_spl_val, x_spl_val, model, '-q')
                E_val += E_01(y_spl_val, p_lab) / len(x_folds)

            if E_val <= minErr:
                minErr = E_val
                lambdaValue = par[8:-15]

        print('Problem 16', 'lambda =', calLambda(lambdaValue), 'Ecv =', minErr)


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

答案依序為[b], [b], [c], [d], [b].