HTML2023 HW4 R12631055 林東甫

1. total CPU time : aN^3 for size N binary.

C_2^k binary classifiers, each trained $\frac{2N}{K}$ size data

$\therefore a \cdot C_2^k \cdot \left(\frac{2N}{k}\right)^3$ #

2. $\phi_0(x) = 1$, $\phi_1(x) = (\phi_0(x), x_1, x_2 \cdots x_d)$

$\vdots$

$\phi_Q(x) = (\phi_{Q-1}(x), x_1^Q, x_1^{Q-1}, x_2, \cdots x_d^Q$

$\mathcal{H}_{\phi_0} \subset \mathcal{H}_{\phi_1} \subset \cdots \subset \mathcal{H}_{\phi_Q}$ H_1 $\cdots$ H_Q

$\therefore E_{in}(g_0) \geq E_{in}(g_1) \geq \cdots \geq E_{in}(g_Q)$, let $g = \tilde{w}\phi_Q(x)$

$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (y_n - w^T \phi_Q(x_n))^2$, $\det(V) = \prod_{1 \le n < m \le N} (x_m - x_n)$

$\therefore$ $\Phi_Q(x) = [1, x \ x^2 \cdots x^{Q-1}]$, let $Q = N$

$\phi_N(x)$ can be expressed as a row of V.

Since $\det(V) \neq 0$, $\phi_Q^T \Phi_Q$ is invertible.

Therefore $\exists\, Q$ s.t. $E_{in}(g) = 0$ #

3

3. Given $\Phi(x) = (x, x^2, \cdots, x^4) \Rightarrow z = (z_1, z_2, \cdots, z_N)$

where $z_n = x^n$

$g(x) = \hat{w}^T \Phi(x)$, $E_{in}(g) = \frac{1}{N} \sum_{n=1}^{N} E[(y_n - g(x_n))^2]$

since $y = x + \epsilon$, and Gaussian with $\mu = 0$, $\sigma^2 = 1$

$E_{in}(g) = \frac{1}{N} \sum_{n=1}^{N} E[(x_n - \hat{w}^T \Phi(x_n))^2 + 1]$

$\therefore E_{out}(g) = E[(y - g(x))^2]$ since $y = x + \epsilon$

$E_{out}(g) = E[(x + \epsilon - \hat{w}^T \Phi(x))^2]$

$\therefore E_{out}(g) = E[(x - \hat{w}^T \Phi(x))^2 + 1]$

4.

$$E(X_h^T X_h) = E\left(\begin{bmatrix} | & | & & | & | & & | \\ x_1 & x_2 & \cdots & x_N & \tilde{x}_1 & \cdots & \tilde{x}_N \\ | & | & & | & | & & | \end{bmatrix} \begin{bmatrix} - x_1 - \\ \vdots \\ - x_N - \\ - \tilde{x}_1 - \\ \vdots \\ - \tilde{x}_N - \end{bmatrix} \right)$$

$$= E\left(\begin{bmatrix} | & & | & | & & | \\ x_1 & \cdots & x_N & x_1+\epsilon & \cdots & x_N+\epsilon \\ | & & | & | & & | \end{bmatrix} \begin{bmatrix} - x_1 - \\ \vdots \\ - x_N - \\ - x_1+\epsilon - \\ \vdots \\ - x_N+\epsilon - \end{bmatrix} \right)$$

$$= E\left(\begin{bmatrix} | & & | \\ x_1 & \cdots & x_N+\epsilon \\ | & & | \end{bmatrix} \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{N,1} \\ x_{1,1}+\epsilon \\ \vdots \\ x_{N,1}+\epsilon \end{bmatrix} + \cdots + \begin{bmatrix} | & & | \\ x_1 & \cdots & x_N+\epsilon \\ | & & | \end{bmatrix} \begin{bmatrix} x_{1,d} \\ \vdots \\ x_{N,d} \\ x_{1,d}+\epsilon \\ \vdots \\ x_{N,d}+\epsilon \end{bmatrix} \right)$$

$$= E(X^T X + X^T X) + N \cdot Var\left(\begin{bmatrix} \epsilon & \cdots & \epsilon \\ \vdots & \ddots & \\ \epsilon & & \epsilon \end{bmatrix} \right)$$

$$= X^T X + X^T X + N \frac{4\delta^2}{12} = \underline{2X^T X + N \cdot \frac{\delta^2}{3}} \ \#$$

5

5. $E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} w^T w$, $\lambda > 0$ with $\eta > 0$

$\cancel{w_{t+1}}$ $w_{t+1} \leftarrow w_t - \eta \nabla E_{aug}(w_t)$

$\therefore$ $w_{t+1} \leftarrow w_t - \eta \nabla \left(E_{in}(w_t) + \frac{\lambda}{N} w_t^T w_t \right)$

$\Leftrightarrow$

$w_{t+1} \leftarrow \alpha(w_t - \beta \nabla E_{in}(w_t))$

$\cancel{w_{t+1} \leftarrow}$ $\nabla \left(E_{in}(w_t) + \frac{\lambda}{N} w_t^T w_t \right)$

$$= \frac{\partial E_{in}(w_t)}{\partial w_t} + \frac{\partial \frac{\lambda}{N} w_t^T w_t}{\partial w_t} = \nabla E_{in}(w_t) + \frac{2\lambda}{N} w_t$$

$\therefore$ $w_{t+1} \leftarrow w_t - \eta \left(E_{in}(w_t) + \frac{2\lambda}{N} w_t \right)$

$\therefore$ $w_{t+1} \leftarrow \left(1 - \frac{2\eta\lambda}{N}\right) w_t - \eta E_{in}(E_{in})$

$\alpha = 1 - \frac{2\eta\lambda}{N}$ $\qquad$ $\beta = \frac{\eta}{1 - \frac{2\eta\lambda}{N}}$

6.

$$\min_{w \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} (w x_n - y_n)^2 + \frac{\lambda}{N} w^2$$, Look for gradient by w i.e. $\frac{d}{dw}$

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

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

$$\therefore \lambda = \frac{\sum_{n=1}^{N} x_n y_n}{w^*} - \sum_{n=1}^{N} x_n^2$$ since $C = (w^*)^2$

$\cancel{\alpha = \sum_{n=1}^{N} w^* x_n^2}$, $\cancel{\beta = \sum_{n=1}^{N} x_n y_n}$

$$\alpha = \sum_{n=1}^{N} x_n y_n \qquad \beta = -\sum_{n=1}^{N} x_n^2$$

\#

7

1. since $\min_{\tilde{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\tilde{w}^T\Phi(x_n)-y_n)^2+\frac{\lambda}{N}\|\tilde{w}\|_1$

$$\Longleftrightarrow$$

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

Known. $\Phi(x)=Vx$, $h(x)=\tilde{h}(\Phi(x))$,

$\therefore \tilde{w}^T\Phi(x)=W^Tx$, $\tilde{W}^T=W^TX(VX)^{-1}=W^TV^{-1}$

$\therefore \underline{\Omega(w)=\|\tilde{w}\|_1}$, $\underline{\tilde{W}=V^{-1}W}$ since V is diagonal matrix $V=V^T$ $V^{-1}=V^{-T}$

$$\Updownarrow$$

$$V\tilde{W}=W$$

8

8. $E_{loocv}(A_{minority}) = \frac{1}{N}\sum_{n=1}^{N} e_n = \frac{1}{N}\sum_{n=1}^{N} err(g_n^-(x_n) y_n)$

If it predict	positive	negative data	$g_n^-(x_n)$
positive	$N-1$	N	$A_{min} \rightarrow$ positive
$g()$ negative	N	$N-1$	$A_{min} \rightarrow$ negative

$\Rightarrow$ it always wins!!

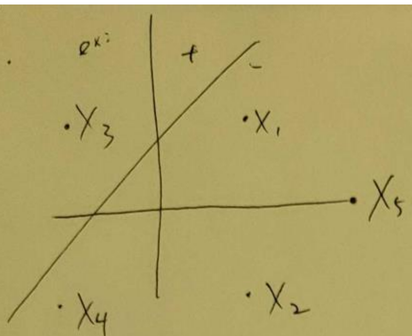
$err(g_n^-(x_n) y_n)$

$$\begin{cases} err(+,+) = 0 \\ err(-,-) = 0 \end{cases} \Rightarrow E_{loocv}(A_{min}) = 0$$

#

9

9.



X_1, X_2, X_3, X_4, X_5 are five point $\in \mathbb{R}^2$ cannot be shattered in these kind of situatio

	X_1	X_2	X_3	X_4	X_5
label	-	-	+	-	+
predict	$\underline{+}$	-	+	-	$-$

brutal:

$2 \cdot C_5^0 C_5^5$

$\cdot C_5^1 C_5^4$ $\quad \frac{1}{5}\ \frac{1}{5}$

$C_5^2 C_5^3$ $\quad 10\ 10$

-	+	+	+	+	o
+	-	+	+	+	o
+	+	-	+	+	o
+	+	+	-	+	o
+	+	+	+	-	o

<=>

ex:

X_1	X_2	X_3	X_4	X_5	
+	-	+	+	+	X
-	+	+	-	-	X
+	-	⋮	+	-	X
-	+	+	-	+	X
+	+	+	+	+	O
-	-	-	-	-	O
+	-	-	-	-	O
-	+	-	-	-	O
-	-	+	-	-	O
-	-	-	+	-	O
-	-	-	-	+	O

$C_5^2 <=> C_5^3$

+	+	-	-	-	X
+	-	+	-	-	O
+	-	-	+	-	X
+	-	-	-	+	O
-	+	+	-	-	X
-	+	-	+	-	O
-	+	-	-	+	O
-	-	+	+	-	O
-	-	+	-	+	X
-	-	-	+	+	X

there's total 10 combination

$E_{in} = \frac{1}{5}$

$\left(\frac{1}{5} \times 10\right) \cdot \frac{1}{32} = \frac{1}{16}$ #

10

```python
Q = 3
x_train, y_train = data("hw4_train.dat", Q)
#x_test, y_test = data("hw4_test.dat", Q)

log_lambdas = np.array([2, 0, -2, -4, -6], dtype=float)
ans = np.zeros((3, 5))
for i in range(len(log_lambdas)):
    prob = problem(y_train, x_train)

    params = parameter(f"-s 0 -c {1/10**log_lambdas[i]/2} -e 0.000001 -q")
    m = train(prob, params)
    p_label, p_acc, p_val = predict(y_train, x_train, m, "-q")
    ans[0][i] = p_acc[0]

print(100-ans)

argans = np.argmax(ans, axis=1)
print(log_lambdas[argans])
```
✓ 0.1s

```
[[ 19.5  12.5   9.    8.    4. ]
 [100.  100.  100.  100.  100. ]
 [100.  100.  100.  100.  100. ]]
[-6.  2.  2.]
```

```python
import itertools
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from liblinear.liblinearutil import *
import matplotlib.pyplot as plt
```
✓ 0.0s

```python
def transform(X, Q):
    X_output = np.ones_like(X[:, 0])[np.newaxis, :]
    for L in range(1, Q+1):
        for subset in itertools.combinations_with_replacement(range(len(X[0])), L):
            tmp = np.ones_like(X[:, 0])[np.newaxis, :]
            for idx in subset:
                tmp = tmp*X[:, idx]
            X_output = np.vstack((X_output, tmp))
    return X_output.T
```
✓ 0.0s

```python
def data(path, Q):
    data = np.genfromtxt(path)
    y, X_data = data[:, -1], transform(data[:, :-1], Q)
    return X_data, y
```
✓ 0.0s

Ans:2

11

```python
def valid_split(X, y, val_size):
    idx = np.random.permutation(len(X))
    x_train = X[idx[val_size:]]
    x_val = X[idx[:val_size]]
    y_train = y[idx[val_size:]]
    y_val = y[idx[:val_size]]
    return x_train, y_train, x_val, y_val
```

✓ 0.0s

```python
def error_val(x_train, y_train, x_val, y_val):
    err = np.full(5, np.inf)
    for i in range(len(log_lambdas)):
        prob = problem(y_train, x_train)
        params = parameter(f"-s 0 -c {1/10**log_lambdas[i]/2} -e 0.000001 -q")
        m = train(prob, params)
        p_label, p_acc, p_val = predict(y_val, x_val, m, "-q")
        err[i] = 100 - p_acc[0]
        if err[i] == np.min(err):
            best_m = m

    return err, best_m
```

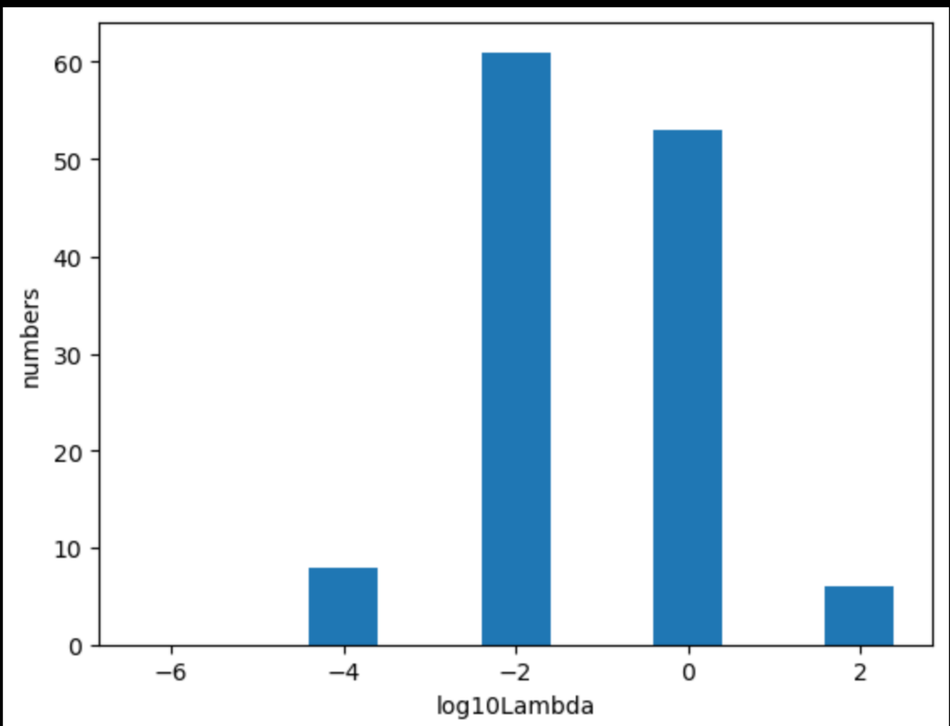
✓ 0.0s

```python
print(ans11)
plt.bar( log_lambdas, ans11)
plt.xlabel('log10Lambda')
plt.ylabel('numbers')
```

✓ 10.0s

```
[ 6. 53. 61.  8.  0.]

Text(0, 0.5, 'numbers')
```



-2

12

```
def error_cv(x, y, V_fold, i):
    err = 0
    for v in range(V_fold):
        x_train = np.concatenate(
            (x[:len(x) // V_fold*v], x[len(x) // V_fold*(v+1):]))
        y_train = np.concatenate(
            (y[:len(y) // V_fold*v], y[len(y) // V_fold*(v+1):]))
        x_val = x[len(x) // V_fold * v: len(x) // V_fold*(v+1)]
        y_val = y[len(y) // V_fold * v: len(y) // V_fold*(v+1)]
        prob = problem(y_train, x_train)
        params = parameter(f"-s 0 -c {1/10**log_lambdas[i]/2} -e 0.000001 -q")
        m = train(prob, params)
        p_label, p_acc, p_val = predict(y_val, x_val, m, "-q")
        err += 100-p_acc[0]
    return err/V_fold
✓ 0.0s
```

12

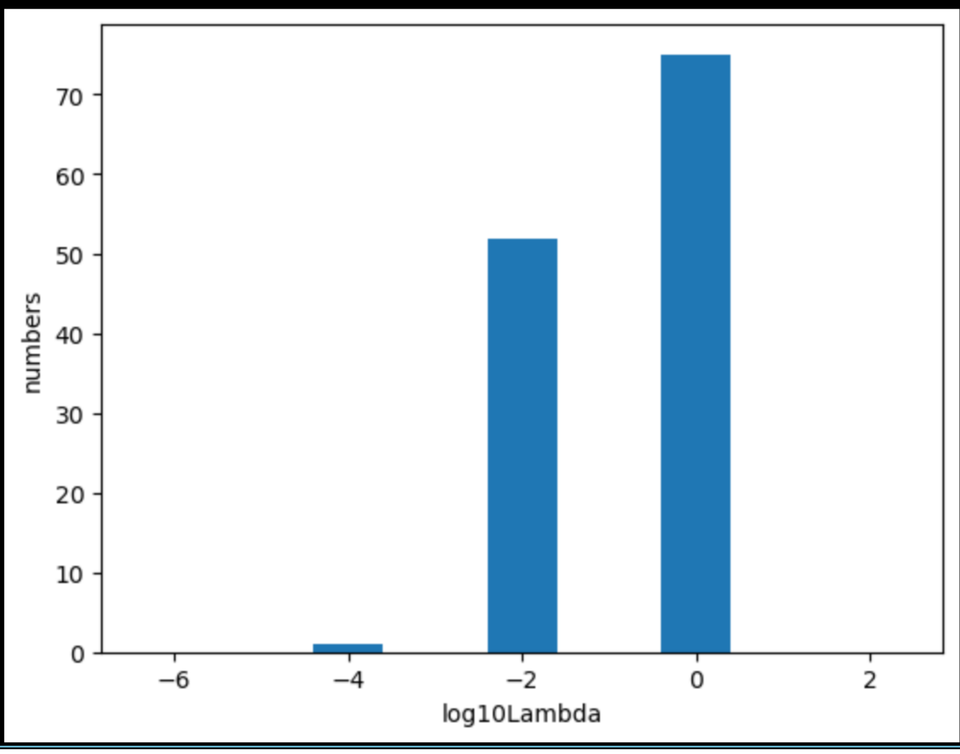
```
np.random.seed(1126)
Q = 3
x_train, y_train = data("hw4_train.dat", Q)
#x_test, y_test = data("hw4_test.dat", Q)
N, train_size, val_size = 128, 120, 80
V_fold = 5
log_lambdas = [2, 0, -2, -4, -6]

# Q12
err = np.zeros(5)
```

```
print(ans12)
plt.bar( log_lambdas, ans12)
plt.xlabel('log10Lambda')
plt.ylabel('numbers')
✓ 1m 10.6s
```

```
[ 0. 75. 52.  1.  0.]

Text(0, 0.5, 'numbers')
```



0 , Q10 Q11 Q12 分別是 2 -2 0