5. Let $\vec{w} = [w_0 \cdots w_d]^T$

$$\boxed{\sum_{k=1}^{K} (\vec{w}^T \tilde{x}_k - \tilde{y}_k)^2}$$

$$= \sum_{k=1}^{d+1} \left([w_0 \ w_1 \cdots w_d] \begin{bmatrix} \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} {\scriptstyle k-1} \\ \sqrt{\lambda \alpha_{k-1}} \\ 0 \\ \vdots \\ 0 \end{bmatrix} - 0 \right)^2$$

$$= \sum_{k=1}^{d+1} \left(\sqrt{\lambda \alpha_{k-1}}\, w_{k-1} \right)^2$$

$$= \sum_{k=1}^{d+1} \lambda \alpha_{k-1} w_{k-1}^2 = \lambda \alpha_0 w_0^2 + \lambda \alpha_1 w_1^2 + \cdots + \lambda \alpha_d w_d^2 = \underline{\lambda \sum_{i=0}^{d} \alpha_i w_i^2}$$

P_V is to solve:

$$\min_{\vec{w} \in \mathbb{R}^{d+1}} \frac{1}{N+K} \left(\sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2 + \boxed{\sum_{k=1}^{K} (\vec{w}^T \tilde{x}_k - \tilde{y}_k)^2} \right)$$

$$= \min_{\vec{w} \in \mathbb{R}^{d+1}} \frac{1}{N+K} \left(\sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2 + \lambda \sum_{i=0}^{d} \alpha_i w_i^2 \right) \quad \because K = d+1$$

$$= \min_{\vec{w} \in \mathbb{R}^{d+1}} \frac{1}{N+d+1} \left(\sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2 + \lambda \sum_{i=0}^{d} \alpha_i w_i^2 \right)$$

P_R is to solve:

$$\min_{\vec{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2 + \frac{\lambda}{N} \sum_{i=0}^{d} \alpha_i w_i^2$$

$$= \min_{\vec{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \left(\sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2 + \lambda \sum_{i=0}^{d} \alpha_i w_i^2 \right)$$

equivalent $\because \frac{1}{N}, \frac{1}{N+d+1} > 0$

$\square$

6. L2 regularization, twice differentiable, convex $E_{in}(\vec{w})$

$$\min_{\vec{w} \in \mathbb{R}^{d+1}} E_{aug}(\vec{w})$$

$$E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \frac{\lambda}{N} \|\vec{w}\|_2^2 \qquad \nabla E_{in}(\vec{w}^*) = 0$$

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

$H \in \mathbb{R}^{(d+1)\times(d+1)}$: Hessian, positive semi-definite ($\because$ convex)

approx $E_{aug}(\vec{w})$: $\tilde{E}_{aug}(\vec{w}) = \tilde{E}_{in}(\vec{w}) + \frac{\lambda}{N}\|\vec{w}\|^2$

$$E_{aug}(\vec{w}) = E_{in}(\vec{w}^*) + \cancelto{0}{(\vec{w}-\vec{w}^*)^T \nabla E_{in}(\vec{w}^*)} + \frac{1}{2}(\vec{w}-\vec{w}^*)^T H (\vec{w}-\vec{w}^*) + \frac{\lambda}{N}\vec{w}^T\vec{w}$$

$$\frac{\partial}{\partial \vec{w}} E_{aug}(\vec{w}) = \frac{\partial}{\partial \vec{w}} E_{in}(\vec{w}^*) + \frac{1}{2}\frac{\partial}{\partial \vec{w}}(\vec{w}-\vec{w}^*)^T H (\vec{w}-\vec{w}^*) + \frac{\partial}{\partial \vec{w}}\left(\frac{\lambda}{N}\vec{w}^T\vec{w}\right)$$

$$= 0 + \frac{1}{2}(H^T+H)(\vec{w}-\vec{w}^*) + \frac{\lambda}{N}\cdot 2\vec{w}$$

$$= H\cdot(\vec{w}-\vec{w}^*) + \frac{2\lambda}{N}\vec{w}$$

$\frac{d}{d\vec{w}} \vec{w}^T A \vec{w} = (A^T + A)\vec{w}$

$\vec{w}^T\vec{w} = \vec{w}^T I \vec{w}$ $\therefore \frac{\partial}{\partial \vec{w}} = (I^T + I)\vec{w} = 2\vec{w}$

$\because E_{in}(\vec{w})$ twice differentiable

$\therefore$ the second partial derivatives $\rightarrow$ continuous

$\rightarrow$ Hessian: symmetric

$\Rightarrow \frac{1}{2}(H^T+H)(\vec{w}-\vec{w}^*) = \frac{1}{2}\cdot 2H\cdot(\vec{w}-\vec{w}^*) = H(\vec{w}-\vec{w}^*)$

Set $\frac{\partial}{\partial \vec{w}} E_{aug}(\vec{w}) = 0$

$$H\cdot(\vec{w}-\vec{w}^*) + \frac{2\lambda}{N}\vec{w} = 0$$

$$\rightarrow H\vec{w} - H\vec{w}^* + \frac{2\lambda}{N}\vec{w} = 0$$

$$\rightarrow \left(H + \frac{2\lambda}{N} I\right)\vec{w} = H\vec{w}^*$$

$$\rightarrow \vec{w} = \left(H + \frac{2\lambda}{N} I\right)^{-1} H\vec{w}^* \quad \square$$

$[h_1, h_2, \ldots, h_{d+1}]$ $\|$ let

since H : positive semi-definite, $\forall h_i \in \lambda(H)$, $h_i \geq 0$

also, $\frac{2\lambda}{N} > 0$

Therefore, $\lambda\left(H + \frac{2\lambda}{N} I\right) = [\underbrace{h_1 + \frac{2\lambda}{N}}_{>0}, \ldots, \underbrace{h_{d+1} + \frac{2\lambda}{N}}_{>0}]$

$\Rightarrow \left(H + \frac{2\lambda}{N} I\right)$: invertible

Lec14 p.10

7. N labels: $y_1, \dots, y_N \overset{\text{i.i.d.}}{\sim} \mu = 0, \sigma^2$

$\swarrow$ $N-K$ training

$\searrow$ K validation

expected validation error: $E\left(\frac{1}{K}\underbrace{\sum_{n=N-K+1}^{N}(y_n-0)^2}_{E[(X-E[X])^2]=\sigma^2}\right) = \sigma^2 \Rightarrow \frac{1}{K}E\left[\sum_{n=N-K+1}^{N} y_n^2\right] = \sigma^2 \Rightarrow E\left[\sum_{n=N-K+1}^{N} y_n^2\right] = K\sigma^2$

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

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

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

linearity of expectation

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

linearity of expectation

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

$$= \frac{1}{K}\left(K\sigma^2 - \cancelto{0}{2\sum_{n=N-K+1}^{N} E[y_n]E[\bar{y}]} + E\left[\sum_{n=N-K+1}^{N}\bar{y}^2\right]\right)$$

$\because E[y_n] = 0$

(labels generated from distribution with mean = 0)

$\boxed{\text{if } X, Y: \text{independent then } E[XY] = E[X]E[Y]}$

$\because$ values of y_n range from $N-K+1$ to N

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

that range from 1 to $N-K$

$\therefore$ $y_n, \bar{y}$: independent

Thus, $E[y_n\bar{y}] = E[y_n]E[\bar{y}]$

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

$E[\bar{y}] = E\left[\frac{1}{N-K}\sum_{n=1}^{N-K} y_n\right] = \frac{1}{N-K}\sum_{n=1}^{N-K} E[y_n] = 0$ $\quad \because E[y_n] = 0$

$$E\left[\sum_{n=N-K+1}^{N}\bar{y}^2\right] \overset{E[\bar{y}]=0}{=} E\left[\sum_{n=N-K+1}^{N}(\bar{y}-E[\bar{y}])^2\right]$$

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

$$= K \cdot Var(\bar{y})$$

def of variance, expected value: constant $\Rightarrow \sum_{n=N-K+1}^{N} c = Kc$

$$= K \cdot Var\left(\frac{1}{N-K}\sum_{n=1}^{N-K} y_n\right)$$

$$= K \cdot \left(\frac{1}{N-K}\right)^2 Var\left(\sum_{n=1}^{N-K} y_n\right)$$

Bienaymé's identity

$$= K \cdot \left(\frac{1}{N-K}\right)^2 \left[\sum_{n=1}^{N-K} Var(y_n) + \sum_{i}\sum_{j \neq i} Cov(y_i, y_j)\right]$$

$i, j \in \{1, \dots, N-K\}$

$$= K \cdot \left(\frac{1}{N-K}\right)^2 \sum_{n=1}^{N-K} Var(y_n)$$ $\because$ $Cov(y_i, y_j) = 0 \quad \forall i,j \in \{1, \dots, N-K\}$
since $y_1, \dots, y_N$ drawn independently

$$= K \cdot \left(\frac{1}{N-K}\right)^2 \sum_{n=1}^{N-K} \sigma^2$$ given by the problem

$$= K \cdot \left(\frac{1}{N-K}\right)^2 (N-K)\sigma^2$$

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

$$= \frac{1}{K}\left(K\sigma^2 + \frac{K\sigma^2}{N-K}\right)$$

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

$$= \sigma^2\left(1 + \frac{1}{N-K}\right) \quad \square$$

7. N labels, $y_1, \dots, y_N \overset{i.i.d.}{\sim} \mu < \infty$

w_0: estimate mean

$E_{in}[w_0] = \frac{1}{N}\sum_{n=1}^{N}(w_0 - y_n)^2$ $\quad w_0^* = \frac{1}{N}\sum_{n=1}^{N} y_n$ $\quad E_{in}[w_0^*] = \frac{1}{N}\sum_{n=1}^{N}(w_0^* - y_n)^2$

$= \frac{1}{N}\sum_{n=1}^{N}[(w_0^*)^2 - 2w_0^* y_n + y_n^2]$ —— $*$

Claim:

for $N \geq 2$

$E_{loocv}(A_{avg}) = \left(\frac{N}{N-1}\right)^2 E_{in}(w_0^*)$

Prove: for $N \geq 2$, each time we can partition the N examples into $\begin{cases} N-1 : \text{training} \\ 1 : \text{validation} \end{cases}$

Let D_n be the new training set, which is:

$D_n = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, \cancel{(\vec{x}_n, y_n)}, \dots, (\vec{x}_N, y_N)\}$

Also, let g_n^- be the hypothesis learned from D_n

$\to$ Let error on validation set $\{(\vec{x}_n, y_n)\}$: e_n

where $e_n = E_{val}[g_n^-] = e(g_n^-(x_n), y_n)$

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

$= \frac{1}{N}\sum_{n=1}^{N}\left[\left(\boxed{\frac{1}{N-1}\sum_{\substack{i=1 \\ i\neq n}}^{N} y_i}\right) - y_n\right]^2$

$w_0^* = \frac{1}{N}\sum_{i=1}^{N} y_i$

$\to N w_0^* = \sum_{i=1}^{N} y_i$

$\to N w_0^* - y_n = \sum_{i=1}^{N} y_i - y_n = \sum_{\substack{i=1 \\ i\neq N}} y_i$

$\to \boxed{\frac{N w_0^* - y_n}{N-1} = \frac{1}{N-1}\sum_{\substack{i=1 \\ i\neq n}}^{N} y_i}$

$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{N w_0^* - y_n}{N-1} - y_n\right)^2$

$= \frac{1}{N}\sum_{n=1}^{N}\left[\frac{N w_0^* - y_n - y_n(N-1)}{N-1}\right]^2$

$= \frac{1}{N(N-1)^2}\sum_{n=1}^{N}[N w_0^* - y_n - y_n(N-1)]^2$

$= \frac{N^2}{N(N-1)^2}\underline{\sum_{n=1}^{N}[(w_0^*)^2 - 2w_0^* y_n + \overset{\shortparallel\, y_n^2}{1}]}$ $*$

$= \frac{N^2}{N(N-1)^2} \underline{N \cdot E_{in}[w_0^*]}$

$= \frac{N^2}{(N-1)^2} E_{in}[w_0^*]$ □

$[N w_0^* - y_n - y_n(N-1)]^2$

$= [(N w_0^* - y_n) - y_n(N-1)]^2$

$(2N w_0^* - 2y_n) y_n (N-1)$
$= (2N w_0^* y_n - 2y_n^2)(N-1)$
$= 2N(N-1) w_0^* y_n - 2(N-1)$

$= (N w_0^* - y_n)^2 - \underline{2(N w_0^* - y_n) y_n (N-1)} + y_n^2 (N-1)^2$

$[(N-1)+1]^2 = N^2$

$= N^2 (w_0^*)^2 - 2N w_0^* y_n + \underline{1} - 2N(N-1) w_0^* y_n + \underline{2(N-1)} + \underline{(N-1)^2}$

$= N^2 (w_0^*)^2 - 2N w_0^* y_n [1 + (N-1)] + N^2$

$= N^2 (w_0^*)^2 - 2N^2 w_0^* y_n + N^2$

$= N^2 [(w_0^*)^2 - 2w_0^* y_n + 1]$

9. binary classifier g: $P(g(\vec{x}) = -1 \mid y = +1) = \varepsilon_+$

$P(g(\vec{x}) = +1 \mid y = -1) = \varepsilon_-$

distribution: $P(y=+1) = P(y=-1) = \frac{1}{2}$

$\rightarrow E_{out}(g) = \frac{1}{2}\varepsilon_+ + \frac{1}{2}\varepsilon_-$

distribution: $P(y=-1) = p$

constant classifier $g_c \rightarrow E_{out}(g_c) = p$
(always predict +1)

$E_{out}(g)$ over distribution $P(y=-1)=p$:
$\rightarrow P(y=+1) = 1-p$

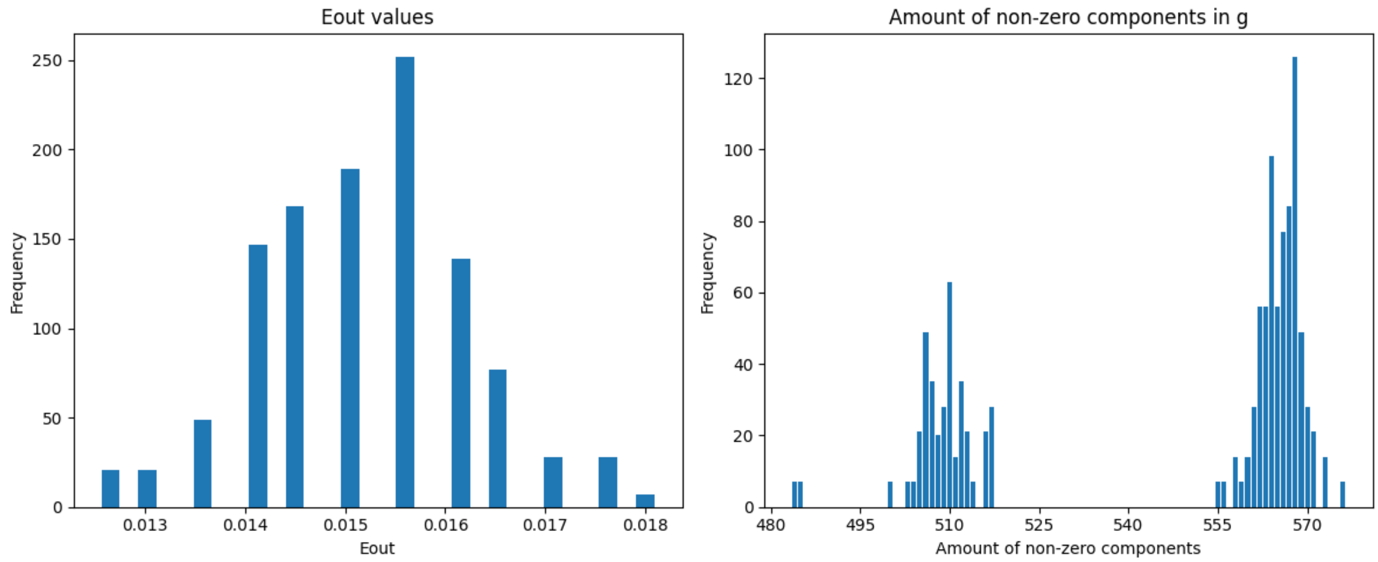
$$E_{out}(g) = P(g(\vec{x}) = -1 \mid y=+1)\, P(y=+1) + P(g(\vec{x}) = +1 \mid y=-1)\, P(y=-1)$$
$$= (1-p)\varepsilon_+ + p\varepsilon_-$$

$E_{out}(g) = E_{out}(g_c)$:

$$(1-p)\varepsilon_+ + p\varepsilon_- = p$$
$$\rightarrow \varepsilon_+ - p\varepsilon_+ + p\varepsilon_- = p$$
$$\rightarrow \varepsilon_+ = p + p\varepsilon_+ - p\varepsilon_-$$
$$\rightarrow \varepsilon_+ = p(1+\varepsilon_+ - \varepsilon_-)$$
$$\rightarrow p = \frac{\varepsilon_+}{1+\varepsilon_+ - \varepsilon_-} \quad \square$$

HTML homework 5: q10 report

The resulting plot is as follows:



Some additional information that I printed in order to do the comparison between this question and question 11:

```
Statistics for Eout:
Mean: 0.0152
Median: 0.0151
Standard Deviation: 0.0010
Min: 0.0126
Max: 0.0181

Statistics for Non-zero Components:
Mean: 546.5
Median: 563.0
Standard Deviation: 27.2
Min: 484
Max: 576
```

Code[1]:

Preprocessing the dataset:

```python
from liblinear.liblinearutil import *
from itertools import combinations_with_replacement
import numpy as np
import scipy.sparse
import matplotlib.pyplot as plt
from tqdm import tqdm
from joblib import Parallel, delayed

```

[28] ✓ 0.4s

Dataset

- `mnist.scale` for training
- `mnist.scale.t` for testing

```python
train_data = 'mnist.scale'
test_data = 'mnist.scale.t'
```

[2]

Empty markdown cell, double-click or press enter to edit.

Read and save data

The training data is saved in the `X_train` list, with each element being a dictionary of features. example format: {153: 0.0117647, 154: 0.0705882, 155: 0.0705882, ...}

The corresponding label list is saved in `y_train`.

Similarily, the testing data is saved in `X_test` and the corresponding label list is saved in `y_test`.

```python
def read_linear_format(file_path):
    X, y = [], []
    with open(file_path, 'r') as f:
        for line in f:
            parts = line.strip().split()
            if parts[0] == '2' or parts[0] == '6':
                y.append(int(parts[0]))
                features = {}
                for item in parts[1:]:
                    index, value = item.split(":")
                    features[int(index)] = float(value)
                X.append(features)
    return X, y

X_train, y_train = read_linear_format(train_data)
X_test, y_test = read_linear_format(test_data)

```

Error function:

$0/1$ error

```python
def ZeroOneError(predictions, y):
    return sum(p_i != y_i for p_i, y_i in zip(predictions, y)) / len(predictions)
```

[6]

[1] Most of the screenshots here are from my .ipynb file, since I wrote the explanations here, however, I ran the code in a .py file (which has equivalent contents), and part of the screenshot is from this file.

Explanation of deriving the relationship between C, λ:

Regularized logistic regression

Original Formulation in README.md

As the explanation in the liblinear README file, for L1-regularized logistic regression `(-s 6)`, we solve

$$\min_{\vec{w}} \sum |w_j| + C \sum \log(1 + \exp(-y_i \vec{w}^T \vec{x}_i))$$

Components of this equation

In this equation, the L1 regularizer is defined as:

$$\Omega(\vec{w}) = \sum |w_j| = ||\vec{w}||_1$$

> info in Lec14 slide 18

- Note: using L1 regularization encourages sparsity, which means fewer parameters.

And the in-sample error for logistic regression is:

$$E_{in}(\vec{w}) = \frac{1}{N} \sum \log(1 + \exp(-y_i \vec{w}^T \vec{x}_i))$$

> in textbook p.91

Derivation of C

Since we knew that:

$$E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \frac{\lambda}{N} \Omega(\vec{w})$$

We can derive that:

$$E_{aug}(\vec{w}) = E_{in}(\vec{w}) + \frac{\lambda}{N} \sum |w_j|$$

For minimization:

$$\begin{aligned}
\arg\min_{\vec{w}} E_{aug}(\vec{w}) &= \arg\min_{\vec{w}} (E_{in}(\vec{w}) + \frac{\lambda}{N} \sum |w_j|) \\
&= \arg\min_{\vec{w}} \frac{N}{\lambda} (\frac{1}{N} \sum \log(1 + \exp(-y_i \vec{w}^T \vec{x}_i)) + \frac{\lambda}{N} \sum |w_j|) \\
&= \arg\min_{\vec{w}} \frac{1}{\lambda} \sum \log(1 + \exp(-y_i \vec{w}^T \vec{x}_i)) + \sum |w_j|
\end{aligned}$$

Compare to the original equation marked with ($\star$), we can see that:

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

Some additional information:

Select the best λ^*

Selecting optimal λ^* using the following equation:

$$\lambda^* = \arg\min_{\log_{10}(\lambda) \in \{-2,-1,0,1,2,3\}} E_{in}(\vec{w}_\lambda)$$

get amount of non-zeros

To get the amount of non-zero components in the model, we can use the function `.get_decfun()`:

```
[W, b] = model_.get_decfun()
```

Main code:

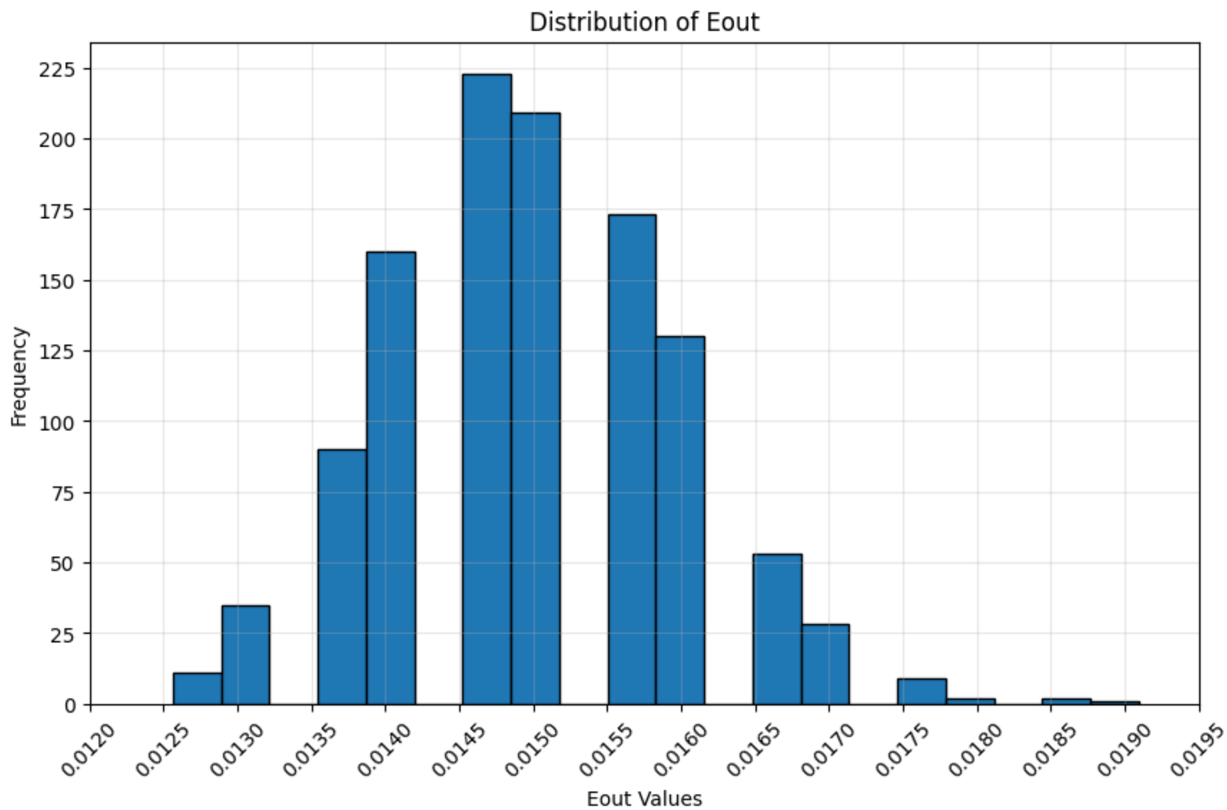
```python
def run_single_experiment(experiment):
    np.random.seed(experiment)
    min_Ein = np.inf
    opt_log10_lambda = 0

    for log10_lambda in (-2, -1, 0, 1, 2, 3):
        c = 1 / (10 ** log10_lambda)
        prob = problem(y_train, X_train)
        param = parameter('-s 6 -c ' + str(c))
        model = train(prob, param)

        train_label, _, _ = predict(y_train, X_train, model)
        Ein = ZeroOneError(train_label, y_train)
        if Ein == min_Ein:
            opt_log10_lambda = max(opt_log10_lambda, log10_lambda)
            if opt_log10_lambda == log10_lambda:
                opt_model = model
        elif Ein < min_Ein:
            min_Ein = Ein
            opt_log10_lambda = log10_lambda
            opt_model = model

    test_label, _, _ = predict(y_test, X_test, opt_model)
    Eout = ZeroOneError(test_label, y_test)

    W = np.array(opt_model.get_decfun()[0])
    non_zero_count = np.count_nonzero(W)

    return Eout, non_zero_count

experiment_amount = 10
results = Parallel(n_jobs=-1)(                                              # use all cores by setting n_jobs to -1
    delayed(run_single_experiment)(i) for i in tqdm(range(experiment_amount))
)

Eouts, non_zero_count_list = zip(*results)
Eouts = list(Eouts)
non_zero_count_list = list(non_zero_count_list)
```

Plotting:

```python
# aim: plotting
# subaim: First subplot (Eout values)
ax1.hist(Eouts, bins=30)
ax1.set_title('Eout values')
ax1.set_xlabel('Eout')
ax1.set_ylabel('Frequency')

# subaim: Second subplot (non-zero components)
min_val = int(min(non_zero_count_list))
max_val = int(max(non_zero_count_list))
integer_bins = np.arange(min_val, max_val + 2) - 0.5  # +2 to include max_val, -0.5 for bin edges

ax2.hist(non_zero_count_list,
         bins=integer_bins,
         align='mid',
         rwidth=0.8)

ax2.xaxis.set_major_locator(plt.MaxNLocator(integer=True))
ax2.set_title('Amount of non-zero components in g')
ax2.set_xlabel('Amount of non-zero components')
ax2.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```

HTML homework 5: q11 report

The resulting picture is as follows:



In order to compare with the previous question, some statistics are:

```
Statistics for Eout:
Mean: 0.0150
Median: 0.0151
Standard Deviation: 0.0010
Min: 0.0126
Max: 0.0191
```

```
Statistics for Eout:
Mean: 0.0152
Median: 0.0151
Standard Deviation: 0.0010
Min: 0.0126
Max: 0.0181

Statistics for Non-zero Components:
Mean: 546.5
Median: 563.0
Standard Deviation: 27.2
Min: 484
Max: 576
```
[1]

[1] Left: statistics of q11 result; Right: statistics of q10 result

In this problem, the difference from the previous is that we further divide the training set into a subtrain set (with 8000 exampls), and a validation set (3876 examples).
From the statistics, we can see that most of them are quite similar, I think part of the reason is that, even though 3876 examples are taken out from the original training set, so one may expect that as we train on a smaller subtrain set, the out of sample error should be bigger than in q10 while we have the whole training set used, <u>we got an expected value slightly smaller because we retrain the whole training set after the optimal λ is chosed</u>. We did not use g^- to predict the test set, but g instead.

Apart from this reason, I think this result may also due to the precision, if more digits are taken into consideration, maybe we can observe the difference.

Another cause may be the limited number of possible λ, this constraints the variability of our model, even though there are infinitely many possible models, most of them are alike, while only slightly different by the λ value chosen.

Code:

```
Eouts = []

for experiment in tqdm(range(1126)):

    np.random.seed(experiment)

    total_size = len(X_train)
    indices = np.random.permutation(total_size)

    subtrain_indices = indices[:8000]
    validation_indices = indices[8000:]

    X_subtrain = [X_train[i] for i in subtrain_indices]
    X_validation = [X_train[i] for i in validation_indices]
    y_subtrain = [y_train[i] for i in subtrain_indices]
    y_validation = [y_train[i] for i in validation_indices]

    subtrain_prob = problem(y_subtrain, X_subtrain)

    min_validation_err = np.inf
    opt_log10_lambda = 0
    for log10_lambda in (-2, -1, 0, 1, 2, 3):
        subtrain_pred_res = []
        c = 1 / (10 ** log10_lambda)
        param = parameter('-s 6 -c ' + str(c))
        model_by_subtrain = train(subtrain_prob, param)

        validation_label, _, _ = predict(y_validation, X_validation, model_by_subtrain)
        validation_err = ZeroOneError(validation_label, y_validation)
        if validation_err == min_validation_err:
            opt_log10_lambda = max(opt_log10_lambda, log10_lambda)      # break tie by choosing the larger lambda
        elif validation_err < min_validation_err:
            min_validation_err = validation_err
            opt_log10_lambda = log10_lambda

    # rerun the model with the best lambda on the whole training set
    whole_train_prob = problem(y_train, X_train)
    param_with_opt_lambda = parameter('-s 6 -c ' + str(1 / (10 ** opt_log10_lambda)))
    whole_train_model = train(whole_train_prob, param_with_opt_lambda)

    test_label, _, _ = predict(y_test, X_test, whole_train_model)
    Eout = ZeroOneError(test_label, y_test)

    Eouts.append(Eout)
```

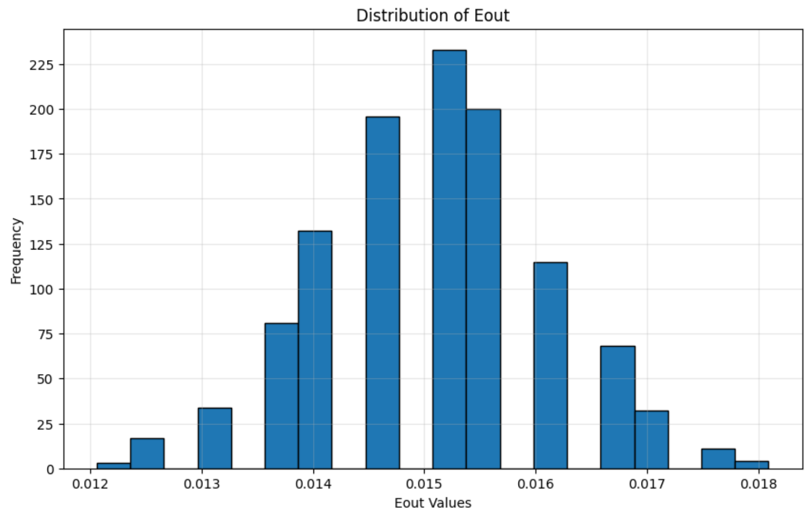
✓ 61m 42.5s

[2]

[2] Other part of codes that are similar to problem 10 (like reading in the data, split the data to get proper form of X and y, the ZeroOneError function…) are omitted in this report, can check for problem 10 for details.

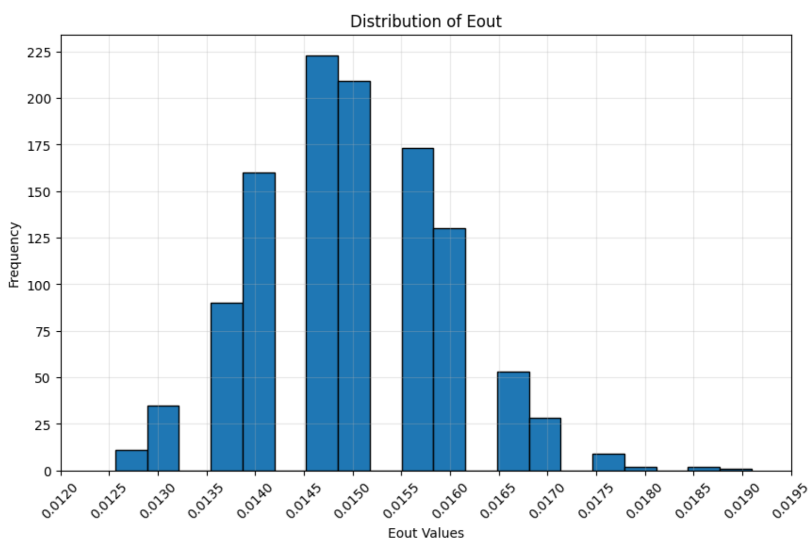
HTML homework 5: q12 report

The resulting picture is as follows:



(q12)

To find the difference, the plot below is the result of the previous problem:



(q11)

From the two plots we can see that the subtrain / validation split is more dispersed, and the variability of the 3-fold cross validation is smaller.

The reason why the subtrain / validation split is more dispersed may due to the fact that the validation set size is not that big compared to the 3-fold version. In the previous question, the validation set size is 3876, and in this question, using 3-fold splits the training set into 3 sets, with each about 3959 examples, which is slightly higher. Also, E_{CV} is calculated by the mean error over using each of the 3 sets as the validation set, therefore these 2 reasons introduce stability in the 3-fold version.

On the other hand, the subtrain / validation split relies only on one split, if the validation set is not that representative, our selection of λ may be biased, this would magnify the effect of the unstableness in the previous question.

Code:

The markdown screenshot below describes the modification of this problem, other parts like reading in the data and the error function…, are the same as problem 10:

3 folds cross validation

> textbook p.150

In this problem, we're asked to conduct 3-folds cross validation on the training data, which means we do the following steps:

1. partition the training data into 3 disjoint subsets (folds): `D_1, D_2, D_3`, each of size $\approx \frac{N}{3}$
2. for each fold `D_i`, we train the model using the remaining 2 folds, and evaluate the model on `D_i`

> Each set serves as a validation set to compute the validation error for the hypothesis g learned on a training set which is the complement of the validation set $D \setminus D_i$

$\rightarrow$ we select the optimal λ^* by this validation error E_{CV}

3. rerun the model with the best λ^* on the whole training set
4. evaluate the model on the testing data

```python
def run_single_experiment(experiment):
    np.random.seed(experiment)

    total_size = len(X_train)
    indices = np.random.permutation(total_size)

    fold_size = total_size // 3

    fold1_indices = indices[:fold_size]
    fold2_indices = indices[fold_size:2*fold_size]
    fold3_indices = indices[2*fold_size:]

    X_fold1 = [X_train[i] for i in fold1_indices]
    X_fold2 = [X_train[i] for i in fold2_indices]
    X_fold3 = [X_train[i] for i in fold3_indices]

    y_fold1 = [y_train[i] for i in fold1_indices]
    y_fold2 = [y_train[i] for i in fold2_indices]
    y_fold3 = [y_train[i] for i in fold3_indices]

    min_Ecv = np.inf
    opt_log10_lambda = 0
    for log10_lambda in (-2, -1, 0, 1, 2, 3):
        each_fold_as_valid_err = []
        c = 1 / (10 ** log10_lambda)

        for i in range(1,4):
            if i == 1:
                X_subtrain = X_fold2 + X_fold3
                y_subtrain = y_fold2 + y_fold3
                X_validation = X_fold1
                y_validation = y_fold1
            elif i == 2:
                X_subtrain = X_fold1 + X_fold3
                y_subtrain = y_fold1 + y_fold3
                X_validation = X_fold2
                y_validation = y_fold2
            else:
                X_subtrain = X_fold1 + X_fold2
                y_subtrain = y_fold1 + y_fold2
                X_validation = X_fold3
                y_validation = y_fold3

            subtrain_prob = problem(y_subtrain, X_subtrain)
            param = parameter('-s 6 -c ' + str(c))
            model_by_subtrain = train(subtrain_prob, param)

            validation_label, _, _ = predict(y_validation, X_validation, model_by_subtrain)
            validation_err = ZeroOneError(validation_label, y_validation)
            each_fold_as_valid_err.append(validation_err)

        Ecv_each_lambda = np.mean(each_fold_as_valid_err)
```

```python

        if Ecv_each_lambda == min_Ecv:
            opt_log10_lambda = max(opt_log10_lambda, log10_lambda)  # break tie by choosing larger lambda
        elif Ecv_each_lambda < min_Ecv:
            min_Ecv = Ecv_each_lambda
            opt_log10_lambda = log10_lambda

    whole_train_prob = problem(y_train, X_train)
    param_with_opt_lambda = parameter('-s 6 -c ' + str(1 / (10 ** opt_log10_lambda)))
    whole_train_model = train(whole_train_prob, param_with_opt_lambda)

    test_label, _, _ = predict(y_test, X_test, whole_train_model)
    return ZeroOneError(test_label, y_test)

n_jobs = -1
Eouts = Parallel(n_jobs=n_jobs)(
    delayed(run_single_experiment)(experiment)
    for experiment in tqdm(range(1126))
)
```

✓ 166m 7.7s

13. elastic net solves:

$$\min_{\vec{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\|\vec{y}-X\vec{w}\|_2^2 + \frac{\lambda_1}{N}\underbrace{\|\vec{w}\|_1}_{\text{not smooth}} + \frac{\lambda_2}{N}\|\vec{w}\|_2^2$$

$\therefore$ not smooth

$\therefore$ use coordinate descent (instead of gradient descent):

$$w_i^{(t+1)} = \underset{w_i\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\Big(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j} - w_i x_{n,i}\Big)^2 + \frac{\lambda_1}{N}\Big(\underbrace{\sum_{j\neq i}|w_j^{(t)}|}_{\text{constant}} + |w_i|\Big) + \frac{\lambda_2}{N}\Big(\underbrace{\sum_{j\neq i}(w_j^{(t)})^2}_{\text{constant}} + w_i^2\Big)$$

$\rightarrow$ closed form sol: $\quad w_i^{(t+1)} \leftarrow \alpha\cdot\max(\beta, 0)$, $\quad \alpha \in \{-1, 1\}$ (sign flip), $\beta \in \mathbb{R}$

if $\beta \le 0$, $w_i^{(t+1)} = 0$ $\rightarrow$ sparsity

$$\Big(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j} - w_i x_{n,i}\Big)^2$$
$$= \Big[\Big(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j}\Big) - w_i x_{n,i}\Big]^2$$
$$= \Big(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j}\Big)^2 - 2\Big(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j}\Big)w_i x_{n,i} + (w_i x_{n,i})^2$$
$$= y_n^2 - 2y_n\sum_{j\neq i} w_j^{(t)}x_{n,j} + \Big(\sum_{j\neq i} w_j^{(t)}x_{n,j}\Big)^2 - 2y_n w_i x_{n,i} + 2\sum_{j\neq i} w_j^{(t)}x_{n,j}w_i x_{n,i} + (w_i x_{n,i})^2$$

expand the update equation:

$$w_i^{(t+1)} = \underset{w_i\in\mathbb{R}}{\operatorname{argmin}}\Big\{\frac{1}{N}\sum_{n=1}^{N}\Big[y_n^2 - 2y_n\sum_{j\neq i} w_j^{(t)}x_{n,j} + \Big(\sum_{j\neq i} w_j^{(t)}x_{n,j}\Big)^2 - 2y_n w_i x_{n,i} + 2\sum_{j\neq i} w_j^{(t)}x_{n,j}w_i x_{n,i} + (w_i x_{n,i})^2\Big]$$
$$+ \frac{\lambda_1}{N}\sum_{j\neq i}|w_j^{(t)}| + \frac{\lambda_1}{N}|w_i| + \frac{\lambda_2}{N}\sum_{j\neq i}(w_j^{(t)})^2 + \frac{\lambda_2}{N}w_i^2\Big\}$$

$$= \underset{w_i\in\mathbb{R}}{\operatorname{argmin}}\Big[\cancel{1} - \cancel{\frac{2}{N}\sum_{n=1}^{N}y_n\sum_{j\neq i}w_j^{(t)}x_{n,j}} + \cancel{\frac{1}{N}\sum_{n=1}^{N}\Big(\sum_{j\neq i}w_j^{(t)}x_{n,j}\Big)^2} - \frac{2w_i}{N}\sum_{n=1}^{N}x_{n,i} + \frac{2w_i}{N}\sum_{n=1}^{N}\sum_{j\neq i}w_j^{(t)}x_{n,j}x_{n,i} +$$
$$\frac{w_i^2}{N}\sum_{n=1}^{N}(x_{n,i})^2 + \underbrace{\cancel{\frac{\lambda_1}{N}\sum_{j\neq i}|w_j^{(t)}|}}_{\text{constant}} + \frac{\lambda_1}{N}|w_i| + \cancel{\frac{\lambda_2}{N}\sum_{j\neq i}(w_j^{(t)})^2} + \frac{\lambda_2}{N}w_i^2\Big]$$

$$= \underset{w_i\in\mathbb{R}}{\operatorname{argmin}} \frac{2w_i}{N}\sum_{n=1}^{N}\Big(\sum_{j\neq i}w_j^{(t)}x_{n,j}x_{n,i} - x_{n,i}\Big) + \Big(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\Big)w_i^2 + \frac{\lambda_1}{N}|w_i|$$

Case1:

If $w_i > 0$, then we get:

$$\underset{w_i\in\mathbb{R}}{\operatorname{argmin}} \frac{2w_i}{N}\sum_{n=1}^{N}\Big(\sum_{j\neq i}w_j^{(t)}x_{n,j}x_{n,i} - x_{n,i}\Big) + \Big(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\Big)w_i^2 + \frac{\lambda_1}{N}w_i$$

$$\Rightarrow \frac{\partial}{\partial w_i}\Big(\frac{2w_i}{N}\sum_{n=1}^{N}\Big(\sum_{j\neq i}w_j^{(t)}x_{n,j}x_{n,i} - x_{n,i}\Big) + \Big(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\Big)w_i^2 + \frac{\lambda_1}{N}w_i\Big) = 0$$

$$\rightarrow \frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] + 2w_i\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right) + \frac{\lambda_1}{N} = 0$$

$$\rightarrow 2w_i\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right) = -\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] - \frac{\lambda_1}{N}$$

$$\rightarrow \boxed{w_i = \frac{-\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] - \frac{\lambda_1}{N}}{2\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)}}$$

case 2:

else if $w_i < 0$, then we get:

$$\underset{w_i \in \mathbb{R}}{\arg\min} \; \frac{2w_i}{N}\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right) + \left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)w_i^2 - \frac{\lambda_1}{N}w_i$$

$$\rightarrow \frac{\partial}{\partial w_i}\left(\frac{2w_i}{N}\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right) + \left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)w_i^2 - \frac{\lambda_1}{N}w_i\right) = 0$$

$$\rightarrow \frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] + 2w_i\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right) - \frac{\lambda_1}{N} = 0$$

$$\rightarrow \boxed{w_i = \frac{-\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] + \frac{\lambda_1}{N}}{2\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)}}$$

$\therefore$ For case 1:

$$-\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] - \frac{\lambda_1}{N} > 0 \quad (\because \text{this case is } w_i > 0)$$

$$\rightarrow -\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] > \frac{\lambda_1}{N} > 0$$

$$\rightarrow \sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right) < -\frac{\lambda_1}{2} < 0$$

△

For case 2:

$$-\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] + \frac{\lambda_1}{N} < 0$$ (if this case is $w_i < 0$)

$$\to -\frac{2}{N}\left[\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right] > -\frac{\lambda_1}{N} < 0$$

$$\to \underline{\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right) > \frac{\lambda_1}{2} > 0}$$

$\triangle$:

The two results shows that, if

$$\left|\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right| > \frac{\lambda_1}{2},$$

then we should update w_i as

$$\operatorname{sign}\left(\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right) \frac{\left|-\frac{2}{N}\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right| - \frac{\lambda_1}{N}}{2\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)}$$

Therefore, we have $\alpha = \operatorname{sign}\left(\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right)$

$$\beta = \frac{\left|-\frac{2}{N}\sum_{n=1}^{N}\left(\sum_{j\neq i} w_j^{(t)} x_{n,j} x_{n,i} - x_{n,i}\right)\right| - \frac{\lambda_1}{N}}{2\left(\frac{\sum_{n=1}^{N}(x_{n,i})^2}{N} + \frac{\lambda_2}{N}\right)}$$

$\square$