ML HW5

R13922146 葉富銘

Problem 5

We can find that $\tilde{\mathbf{x}}_k$ has only one non-zero entry at the (k - 1)-th position(index starts from 0), which is $\mathbf{x}_{k,k-1} = \sqrt{\lambda\alpha_{k-1}}$, hence $\mathbf{w}^T\tilde{\mathbf{x}}_k = \mathbf{w}_{k-1} \cdot \sqrt{\lambda\alpha_{k-1}}$, and given $\tilde{y}_k = 0$, K = d + 1, we can derive :

$$\sum_{k=1}^{K}(\mathbf{w}^T\tilde{\mathbf{x}}_k - \tilde{y}_k)^2 = \sum_{k=1}^{K}(\mathbf{w}^T\tilde{\mathbf{x}}_k)^2 = \sum_{k=1}^{d+1}\mathbf{w}_{k-1}^2\lambda\alpha_{k-1} = \lambda\sum_{k=0}^{d}\alpha_k\mathbf{w}_k^2$$

Therefore, P_V can be rewritten as :

$$\min_{\mathbf{w}\in\mathbb{R}^{b+1}} \frac{1}{N+K}\left(\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 + \lambda\sum_{k=0}^{d}\alpha_k\mathbf{w}_k^2\right)$$

Comparing with P_R, we can easily find that P_V is just P_R scaled by a factor of $\frac{N}{N+K}$, and hence the optimal $\mathbf{w}^*$ by solving P_R and P_V are exactly the same.

Problem 6

$\tilde{E}_{in}(\mathbf{w}) = E_{in}(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T\ \mathrm{H}\ (\mathbf{w} - \mathbf{w}^*)$

$$\begin{aligned}\tilde{E}_{aug}(\mathbf{w}) &= \tilde{E}_{in}(\mathbf{w}) + \frac{\lambda}{N}\|\mathbf{w}\|^2 \\ &= E_{in}(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T\mathrm{H}(\mathbf{w} - \mathbf{w}^*) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}\end{aligned}$$

To find the minimizer, we take the gradient of $\tilde{E}_{aug}(\mathbf{w})$ with respect to $\mathbf{w}$ and set it to zero, and since H is the Hessian matrix of $E_{in}(\mathbb{w})$ evaluated at $\mathbf{w}^*$, its value is fixed with respect to $\mathbf{w}$ and hence can be treated as constant.

$\nabla_w\tilde{E}_{aug}(\mathbf{w}) = 0 + \mathrm{H}(\mathbf{w} - \mathbf{w}^*) + \frac{2\lambda}{N}\mathbf{w} = 0$

Given the regularization constraint that λ needs to be non-negative, we can split into two cases :

1. If $\lambda = 0$, then $\mathbf{w}$ is simply $\mathbf{w}^*$
2. If $\lambda > 0$, then $\mathrm{H}(\mathbf{w} - \mathbf{w}^*) = -\frac{2\lambda}{N}\mathbf{w}$, $(\mathrm{H} + \frac{2\lambda}{N}\mathrm{I})\mathbf{w} = \mathrm{H}\mathbf{w}^*$

   Since $\lambda > 0$ and H is positive semi-definite, hence $(\mathrm{H} + \frac{2\lambda}{N}\mathrm{I})\mathbf{w}$ is positive definite and thus has inverse matrix, so the minimizer $\mathbf{w}$ can be expressed as

   $\mathbf{w} = \left(\mathrm{H} + \frac{2\lambda}{N}\mathrm{I}\right)^{-1}\mathrm{H}\mathbf{w}^*$

Problem 7

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

Let's split it into three cases :

1. Since each y_n is i.i.d with variance σ^2 and mean 0, $\mathbb{E}(y_n)^2=\sigma^2$, and hence

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

2. y_n is in the validation set and $\overline{y}$ is the mean of training set, hence they are independent, so

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

   Since y_n is i.i.d with variance σ^2 and mean 0, $\mathbb{E}(y_n)=0$, and hence this term is 0.

3. $\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}\overline{y}^2\right)=\mathbb{E}(\overline{y}^2)$

   Since $\overline{y}$ is the mean of (N - K) i.i.d variables with variance σ^2, $\mathrm{Var}(\overline{y})=\frac{\sigma^2}{N-K}$, hence

   $\mathbb{E}(\overline{y}^2)=\mathrm{Var}(\overline{y})=\frac{\sigma^2}{N-K}$

Combine the above result, we can derive $\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\overline{y})^2\right)=\sigma^2+\frac{\sigma^2}{N-K}$.

Problem 8

Define $w_0^{(-n)}=\frac{1}{N-1}\sum_{j=1,j\neq n}^{N}y_j$, Given $w_0^*=\frac{1}{N}\sum_{n=1}^{N}y_n$, we can express $w_0^{(-n)}$ in terms of w_0^* :
$w_0^{(-n)}=\frac{Nw_0^*-y_n}{N-1}$

$$\begin{aligned}
E_{loocv}(A_{avg}) &= \frac{1}{N}\sum_{n=1}^{N}\left(w_0^{(-n)}-y_n\right)^2\\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{Nw_0^*-y_n}{N-1}-y_n\right)^2\\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{N(w_0^*-y_n)}{N-1}\right)^2\\
&= \left(\frac{N}{N-1}\right)^2\frac{1}{N}\sum_{n=1}^{N}(w_0^*-y_n)^2\\
&= \left(\frac{N}{N-1}\right)^2E_{in}(w_0^*)
\end{aligned}$$

proved.

Problem 9

Given the new distribution, we can derive $E_{out}(g)$:

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

So to find p so that g will be as good as the constant classifier g_c, we need to find p that satisfy the following question :

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

After computing :

$(1-p)\epsilon_+ + p\epsilon_- = \mathrm{p}$

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

$p = \frac{\epsilon_+}{\epsilon_+ - \epsilon_- + 1}$.

Problem 10

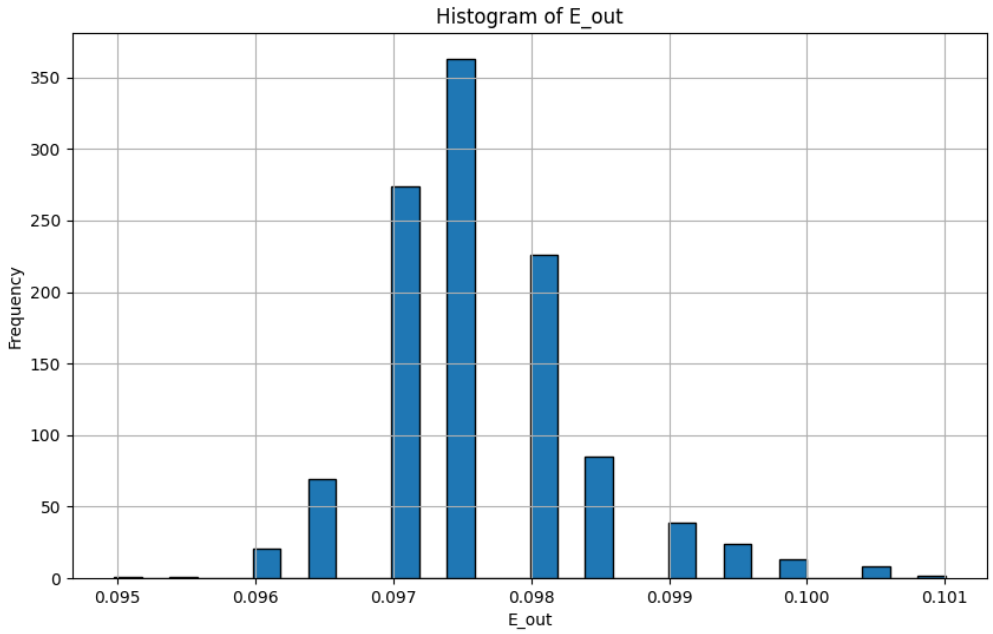

Figure 1:

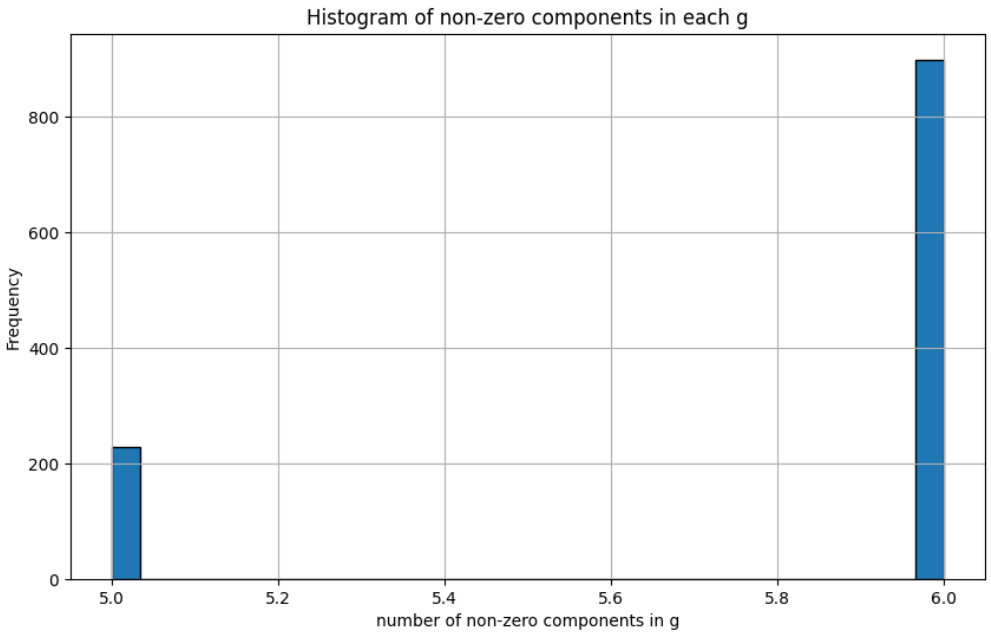

Figure 2:

```python
def _experiment(y_train_filtered, x_train_filtered, y_test_filtered, x_test_filtered, lambda_values):

    for candidate in lambda_values:

        best_lambda = lambda_values[0]
        best_acc = 0
        best_model = None

        # calculate C
        C = 1 / candidate
        model = train(y_train_filtered, x_train_filtered, f'-s 6 -c {C}')

        _, p_acc, _ = predict(y_train_filtered, x_train_filtered, model)

        # select best lambda with breaking tie by selecting larger lambda
        if best_acc <= p_acc[0]:
            best_acc = p_acc[0]
            best_lambda = candidate
            best_model = model

    print(f'Find the best lambda : {best_lambda}')

    # calculate E_out
    _, p_acc, _ = predict(y_test_filtered, x_test_filtered, model)
    E_out = float(1) - (p_acc[0] / float(100))

    # extract w
    w_vector = model.w[:best_model.nr_feature]

    # number of non-zero components
    non_zero_count = sum(1 for ele in w_vector if ele != 0)

    print(f'\nE_out : {E_out}')
    print(f'\nnon-zero count : {non_zero_count}\n')

    return E_out, non_zero_count
```

Figure 3:

```python
def main():
    start_time = time.time()
    # load dataset
    y_train, x_train = load_LIBVSM('./mnist.scale')
    y_test, x_test = load_LIBVSM('./mnist.scale.t')

    # pad x_0
    x_train = csr_matrix(np.hstack([np.ones((x_train.shape[0], 1)), x_train]))
    x_test = csr_matrix(np.hstack([np.ones((x_test.shape[0], 1)), x_test]))

    # extract class 2 & 6
    train_filter = np.where(np.isin(y_train, [2, 6]))[0]
    test_filter = np.where(np.isin(y_test, [2, 6]))[0]

    x_train_filtered = x_train[train_filter]
    y_train_filtered = y_train[train_filter]
    x_test_filtered = x_test[test_filter]
    y_test_filtered = y_test[test_filter]

    # convert to binary label (2 -> 1, 6 -> -1)
    y_train_filtered = np.where(y_train_filtered == 2, 1, -1)
    y_test_filtered = np.where(y_test_filtered == 2, 1, -1)

    log_lambda = [-2, -1, 0, 1, 2, 3]
    lambda_values = [10**x for x in log_lambda]

    iterations = 1126

    E_out_list, non_zero_count_list = experiment(y_train_filtered, x_train_filtered, y_test_filtered, x_test_filtered, lambda_values, iterations)

    plot_E_out(E_out_list, iterations)
    plot_non_zero(non_zero_count_list, iterations)

    end_time = time.time()
    execution_time = end_time - start_time

    print(f"Execution time: {execution_time} seconds")
```

Figure 4:

Problem 11

E_{out} in problem 10 are very small and they seem to appear on a few certain values, while E_{out} in problem 11 are larger, also they seem to appear on more values and are more continuous. I think it may be because in problem 11, it randomly splits the training examples to sub-training set and validation set, which add the randomness in each experiment, and hence g is more diverse and is reflected on E_{out}.

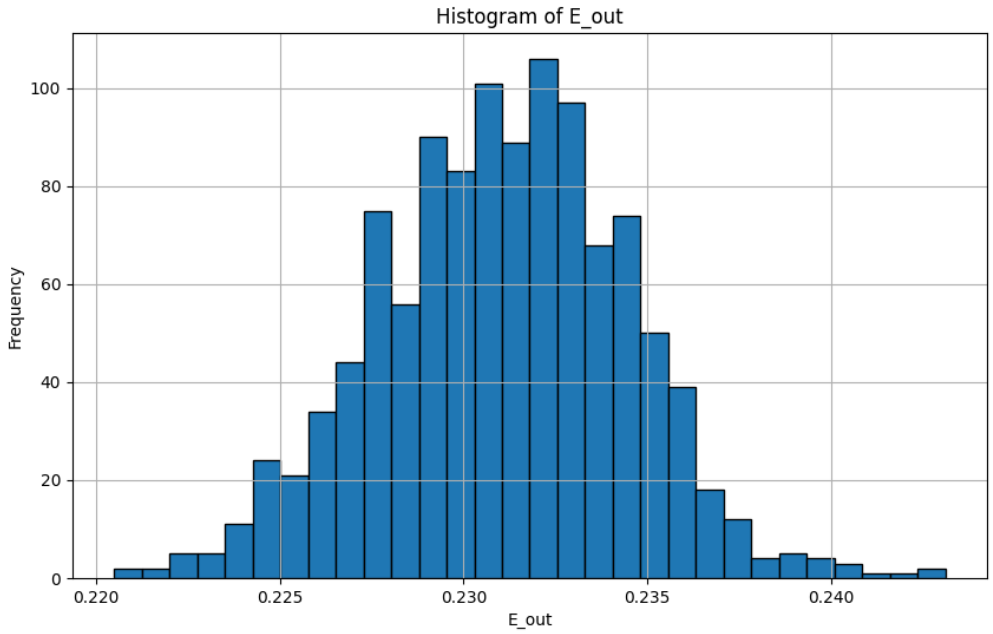

Figure 5:

```python
def _experiment(y_train, x_train, y_test, x_test, lambda_values):

    # split into training & validation set
    x_sub_train, x_val, y_sub_train, y_val = train_test_split(x_train, y_train, train_size=8000, random_state=42)

    # find best lambda
    for candidate in lambda_values:

        best_lambda = lambda_values[0]
        best_acc = 0

        # calculate C
        C = 1 / candidate
        model = train(y_sub_train, x_sub_train, f'-s 6 -c {C}')

        _, p_acc, _ = predict(y_val, x_val, model)

        # select best lambda with breaking tie by selecting larger lambda
        if best_acc <= p_acc[0]:
            best_acc = p_acc[0]
            best_lambda = candidate

    print(f'Find the best lambda : {best_lambda}')

    # re-run on the whole training set with the best lambda
    model = train(y_train, x_train, f'-s 6 -c {1 / best_lambda}')

    _, p_acc, _ = predict(y_test, x_test, model)

    E_out = float(1) - (p_acc[0] / float(100))

    print(f'E_out : {E_out}')

    return E_out
```

Figure 6:

```python
def main():
    start_time = time.time()

    # load dataset
    y_train, x_train = load_LIBVSM('./mnist.scale')
    y_test, x_test = load_LIBVSM('./mnist.scale.t')

    # pad x_0
    x_train = csr_matrix(np.hstack([np.ones((x_train.shape[0], 1)), x_train]))
    x_test = csr_matrix(np.hstack([np.ones((x_test.shape[0], 1)), x_test]))

    log_lambda = [-2, -1, 0, 1, 2, 3]
    lambda_values = [10**x for x in log_lambda]

    iterations = 1126

    E_out_list = experiment(y_train, x_train, y_test, x_test, lambda_values, iterations)

    plot_E_out(E_out_list, iterations)

    end_time = time.time()

    execution_time = end_time - start_time

    print(f"Execution time: {execution_time} seconds")
```

Figure 7:

Problem 12

E_{out} in problem 11 and 12 seem to have a little difference. I think that it may be because they both use the same strategy during training, that is, they both split the original training dataset into training and validation, and after selecting the largest λ, they both re-run the model on the whole training set with with λ^* to get g, although their ratio between training and validation are different.

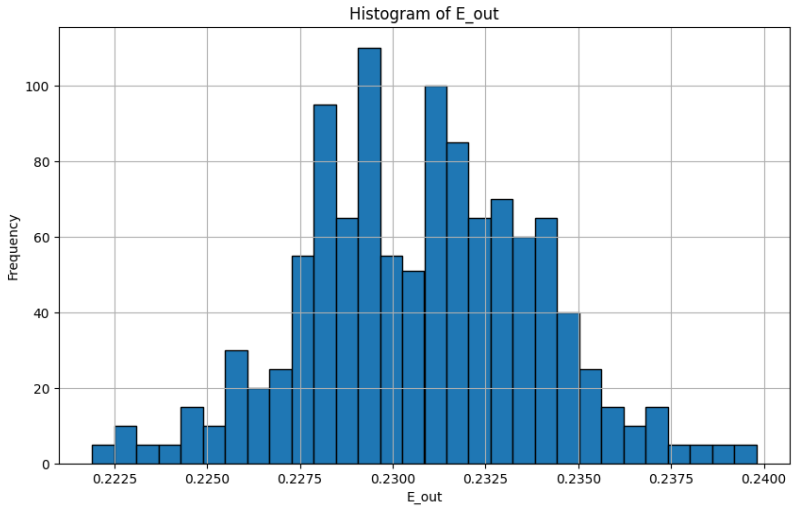

Figure 8:

```python
def _experiment(y_train, x_train, y_test, x_test, lambda_values):

    # find best lambda
    for candidate in lambda_values:

        best_lambda = lambda_values[0]
        best_acc = 0

        # calculate C
        C = 1 / candidate
        p_acc = train(y_train, x_train, f'-s 6 -c {C} -v 3')

        # select best lambda with breaking tie by selecting larger lambda
        if best_acc <= p_acc:
            best_acc = p_acc
            best_lambda = candidate

    # print(f'Find the best lambda : {best_lambda}')

    # re-run on the whole training set with the best lambda
    model = train(y_train, x_train, f'-s 6 -c {1 / best_lambda}')

    _, p_acc, _ = predict(y_test, x_test, model)

    E_out = float(1) - (p_acc[0] / float(100))

    # print(f'E_out : {E_out}')

    return E_out
```

Figure 9:

13.

$$w_i^{(t+1)} \leftarrow \underset{w_i \in R}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\underbrace{\left(y_n - \sum_{j\neq i} w_j^{(t)} x_{n,j} - w_i x_{n,i}\right)^2}_{①} + \underbrace{\frac{\lambda_1}{N}\left(\sum_{j\neq i}|w_j^{(t)}| + |w_i|\right)}_{②} + \underbrace{\frac{\lambda_2}{N}\left(\sum_{j\neq i}(w_j^{(t)})^2 + w_i^2\right)}_{③}$$

To find the optimal w_i that minimize the objective function, we can first isolate and consider only the terms involving w_i, so let's first split the objective function into three terms:

①

Square loss term:

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

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

So the only term depending on w_i is :

$$\frac{1}{N}\sum_{n=1}^{N}\left(-2\left(y_n - \sum_{j\neq i} w_j^{(t)} x_{n,j}\right) w_i x_{n,i} + x_i^2 + w_i^2 x_{n,i}^2\right)$$

let $A = \frac{1}{N}\sum_{n=1}^{N} x_{n,i}^2$, $B = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \sum_{j\neq i} w_j^{(t)} x_{n,j}\right) x_{n,i}$

which are both constant with respect to w_i,

the above term can be rewritten as

$$Aw_i^2 - 2Bw_i$$

② L1 regularization term :

the W_i - dependent part is $\frac{\lambda_1}{N}|W_i|$

③

L_2 regularization term :

the W_i - dependent part is $\frac{\lambda_2}{N}W_i^2$

Combine the above, the coordinate descent update for W_i becames :

$$W_i^{(t+1)} \leftarrow \underset{W_i \in R}{\operatorname{argmin}} \left(AW_i^2 - 2BW_i + \frac{\lambda_1}{N}|W_i| + \frac{\lambda_2}{N}W_i^2\right)$$

$$= \underset{W_i \in R}{\operatorname{argmin}} \left((A+\frac{\lambda_2}{N})W_i^2 - 2BW_i + \frac{\lambda_1}{N}|W_i|\right)$$

$$= \underset{W_i \in R}{\operatorname{argmin}} \left((A+\frac{\lambda_2}{N})\left(W_i^2 - \frac{2}{A+\frac{\lambda_2}{N}}BW_i\right) + \frac{\lambda_1}{N}|W_i|\right)$$

$$= \underset{W_i \in R}{\operatorname{argmin}} \left((A+\frac{\lambda_2}{N})\left(\left(W_i - \frac{B}{A+\frac{\lambda_2}{N}}\right)^2 - \left(\frac{B}{A+\frac{\lambda_2}{N}}\right)^2\right) + \frac{\lambda_1}{N}|W_i|\right)$$

$$= \underset{w_i \in R}{\operatorname{argmin}} \left(\left(A+\frac{\lambda_2}{N}\right)\left(w_i - \underbrace{\frac{B}{A+\frac{\lambda_2}{N}}}_{\gamma}\right)^2 - \underbrace{\left(A+\frac{\lambda_2}{N}\right)\left(\frac{B}{A+\frac{\lambda_2}{N}}\right)^2}_{\text{constant, ignore it}} + \frac{\lambda_1}{N}|w_i| \right)$$

$$\Rightarrow w_i^{(t+1)} \leftarrow \underset{w_i \in R}{\operatorname{argmin}} \underbrace{\left(\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)^2 + \frac{\lambda_1}{N}|w_i| \right)}_{f(w_i)}$$

Since $\frac{\lambda_1}{N}|w_i|$ is non-differentiable at point 0,

we split it into three cases : $w_i > 0, w_i < 0, w_i = 0$

and check for the optimal solution in each case

① $w_i > 0$: $f(w_i) = \left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)^2 + \frac{\lambda_1}{N} w_i$

$$\frac{\partial f(w_i)}{\partial w_i} = 2\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma) + \frac{\lambda_1}{N}$$

$$\Rightarrow 2\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma) + \frac{\lambda_1}{N} = 0, \quad w_i = \gamma - \frac{\lambda_1}{2N\left(A+\frac{\lambda_2}{N}\right)}$$

② $w_i < 0$: $f(w_i) = \left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)^2 - \frac{\lambda_1}{N} w_i$

$$\Rightarrow 2\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)-\frac{\lambda_1}{N}=0, \quad w_i=\gamma+\frac{\lambda_1}{2N\left(A+\frac{\lambda_2}{N}\right)}$$

③

$w_i = 0$:

The subdifferential of $\frac{\lambda}{N}|w_i|$ is :

$$\partial\left(\frac{\lambda_1}{N}|w_i|\right)=\begin{cases}\frac{\lambda_1}{N} & , w_i>0\\ -\frac{\lambda_1}{N} & , w_i<0\\ \left[-\frac{\lambda_1}{N},\frac{\lambda_1}{N}\right] & , w_i=0\end{cases}$$

To minimize $f(w_i)$, it must satisfy this condition : $0 \in \frac{\partial}{\partial w_i}(\text{Quadratic term}) + \partial(\text{Regularization term})$

so if $w_i = 0$ is optimal :

$$-\frac{\lambda_1}{N} \le \frac{\partial}{\partial w_i}(\text{Quadratic term}) \le \frac{\lambda_1}{N}$$

$$\Rightarrow -\frac{\lambda_1}{N} \le -2\left(A+\frac{\lambda_2}{N}\right)\gamma \le \frac{\lambda_1}{N}$$

$\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)^2$,

$2\left(A+\frac{\lambda_2}{N}\right)(w_i-\gamma)$

$$\Rightarrow \frac{\lambda_1}{2N(A+\frac{\lambda_2}{N})} \geq \gamma \geq \frac{-\lambda_1}{2N(A+\frac{\lambda_2}{N})} \Rightarrow |\gamma| \leq \frac{\lambda_1}{2N(A+\frac{\lambda_2}{N})}$$

Combine the result above, we can get

$$W_i^{(t+1)} = \text{sign}(\gamma) \cdot \max\left(|\gamma| - \frac{\lambda_1}{2N(A+\frac{\lambda_2}{N})}, 0\right)$$

$$= \text{sign}(\gamma) \cdot \max\left(\frac{|B| - \frac{\lambda_1}{2N}}{A+\frac{\lambda_2}{N}}, 0\right)$$

Therefore, $\alpha = \text{sign}\left(\frac{B}{A+\frac{\lambda_2}{N}}\right)$, $\beta = \frac{|B| - \frac{\lambda_1}{2N}}{A+\frac{\lambda_2}{N}}$

where $A = \frac{1}{N}\sum_{n=1}^{N} x_{n,i}^2$, $B = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \sum_{j \neq i} W_j^{(t)} x_{n,j}\right) x_{n,i}$

#

p.s Source : chatgpt