Homework # 5

姓名	學號
操之晴	R13922A04

5.

$$(P_R)\ \min \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \frac{\lambda}{N}\sum_{i=0}^{d}\alpha_i w_i^2$$

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

$$= \frac{1}{N+K}\left(\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \sum_{k=1}^{K}(\lambda\alpha_{k-1} w_{k-1}^2)^2\right)$$

$$= \frac{1}{N+K}\left(\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \lambda\sum_{i=0}^{d}\alpha_i w_i^2\right)$$

$$= \frac{N}{N+K}(P_R)$$

Thus, solving $(P_v\ \ )$ is the same as the optimal solution obtained by solving (P_R).

6.

$\tilde{E}_{aug}(w) = \tilde{E}_{in}(w) + \frac{\lambda}{N}\left||w|\right|^2$

From the Taylor expansion provided around w^* , we have:

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

Substitute this approximation into $\tilde{E}_{aug}(w)$

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

The minimizer of $\tilde{E}_{aug}(w)$ is found by taking the derivative with respect to w and setting it to zero:

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

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

So, we get: $$w = \left(H + \tfrac{2\lambda}{N} I\right)^{-1} Hw^*$$

7.

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

we can decompose the expression $(y_n-\tilde{y})^2$ as follows:

$$(y_n-\tilde{y})^2=(y_n-0)^2-2(y_n-0)(\tilde{y}-0)+(\tilde{y}-0)^2$$

Taking the expectation over the validation set, we get:$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\tilde{y})^2\right)=$

$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)^2\right)-2\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)(\tilde{y}-0)\right)+\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(\tilde{y}-0)^2\right)$

$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)^2\right)=\sigma^2$ as the labels y_n are independent and identically distributed with mean = 0 and variance σ^2

$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)(\tilde{y}-0)\right)=0$ as $\tilde{y}\ and\ y_n$ are independent, and both have mean 0.

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(\tilde{y}-0)^2\right)=\frac{1}{K}*K*\frac{\sigma^2}{N-K}=\frac{\sigma^2}{N-K}$$

. Therefore, the answer is $\sigma^2+\frac{\sigma^2}{N-K}$

8.

The given optimal solution $w_0^* = \frac{1}{N}\sum_{n=1}^{N} y_n$ minimizes

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

In LOOCV, we compute the prediction error for each sample when it is left out of the t raining set. For the n-th sample, let w_{-n} denote the mean calculated without y_n

$$w_{-n} = \frac{1}{N-1}\sum_{i \neq n} y_i$$

The LOOCV error for the n-th sample is $(w_{-n} - y_n)^2$

The total LOOCV error is the average over all N samples $E_{LOOCV} = \frac{1}{N}\sum_{n=1}^{N}(w_{-n} - y_n)^2$

Using the relationship between w_{-n} and w_0^*, we observe:

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

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

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

The LOOCV error is the average of the squared errors over all n:

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

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

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

9. $E_{out} = \frac{1}{N}\sum[g(x) \neq y] = \frac{1}{N}\sum[g(x) = -1, y = 1] + \frac{1}{N}\sum[g(x) = 1, y = -1]$

$$= \frac{1}{N}\sum \mathbb{P}(g(x) = -1|y = 1)\mathbb{P}(y = 1) + \frac{1}{N}\mathbb{P}(g(x) = 1|y = -1)\mathbb{P}(y = -1)$$

$$= \frac{1}{N}N\left(\frac{1}{2}\epsilon_+\right) + \frac{1}{N}N\left(\frac{1}{2}\epsilon_-\right) = \frac{1}{2}\epsilon_+ + \frac{1}{2}\epsilon_-$$

$\mathbb{P}(y = -1) = p,\ \mathbb{P}(y = 1) = 1 - p$

$E_{out}(g_c) = \mathbb{P}(g(x) = -1|y = 1)\mathbb{P}(g(x) = 1|y = -1)$

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

So, $p = \frac{\epsilon_+}{1+\epsilon_+-\epsilon_-}$

10.

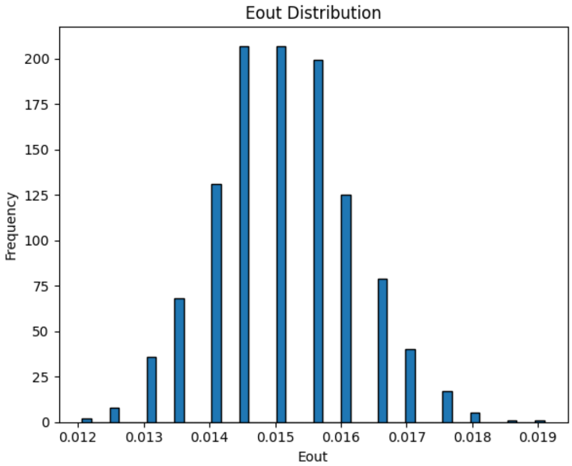


Figure 10a. histogram of Eout(g)

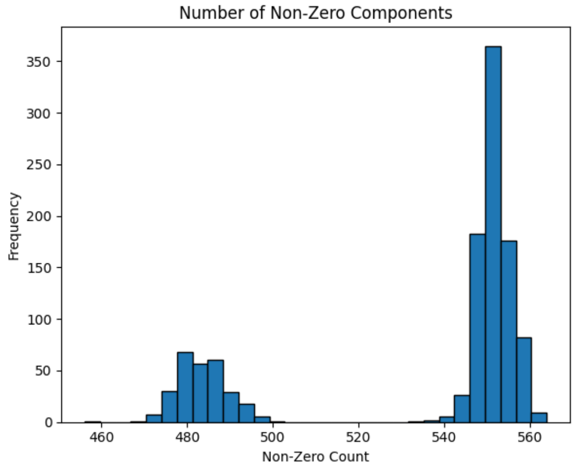


Figure 10b. histogram of non-zero components

```python
from liblinear.liblinearutil import *
import numpy as np
from sklearn.datasets import load_svmlight_file
import matplotlib.pyplot as plt
from tqdm import tqdm

# parameters
EXPERIMENTS = 1126
lambdas = [-2, -1, 0, 1, 2, 3]

# loading data
train_x, train_y = load_svmlight_file('mnist.scale')
test_x, test_y= load_svmlight_file('mnist.scale.t')

train_x = train_x.astype(np.float64)
test_x = test_x.astype(np.float64)

# filter class 2 & 6
def filter_data(x, y, class1=2, class2=6):
    mask = (y == class1) | (y == class2)
    x_filtered = x[mask]
    y_filtered = np.where(y[mask] == class1, 1, -1)  # class 2 : 1，class 6 : -1
    return x_filtered, y_filtered

train_x, train_y = filter_data(train_x, train_y)
test_x, test_y = filter_data(test_x, test_y)

# finding best lambda
Ein_histogram = []
Eout_histogram = []
non_zero_histogram = []

for i in tqdm(range(EXPERIMENTS), desc="Experiments"):
    best_lambda = None
    best_Ein = float('inf')

    for log_lambda in lambdas:
        current_lambda = 10 ** log_lambda
        c = 1 / (2 * current_lambda)

        model = train(train_y, train_x, f'-s 6 -c {c} -B 1 -q')

        _, Ein_acc, _ = predict(train_y, train_x, model, '-q')
        ein_error = 1 - (Ein_acc[0] / 100)  # 0/1 error
```

Figure 10c. screenshots of my code

Eout Distribution

The test error E_{out} is mainly concentrated between [0.014, 0.017], and the peak is around 0.015, showing very stable model performance. It looks like a normal distribution, indicating that different random seeds have minimal impact.

Number of Non-Zero Components

The number of non-zero weights shows two clusters, one around 540 and another around 480. This could be because the best λ values chosen in different experiments mainly fall into two categories:
Larger λ (0.1) stronger regularization, shrinking more weights, leading to fewer non-zero components
Smaller λ(0.01) relaxes regularization, allowing more weights to remain non-zero.

11.

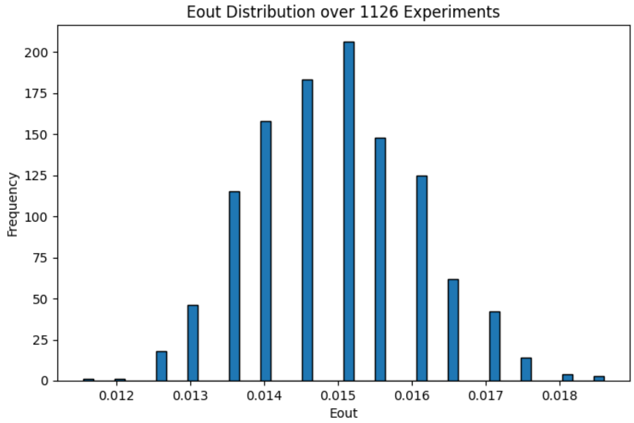


Figure 11a. histogram of Eout(g)

HW5 > src > q11.py > ...

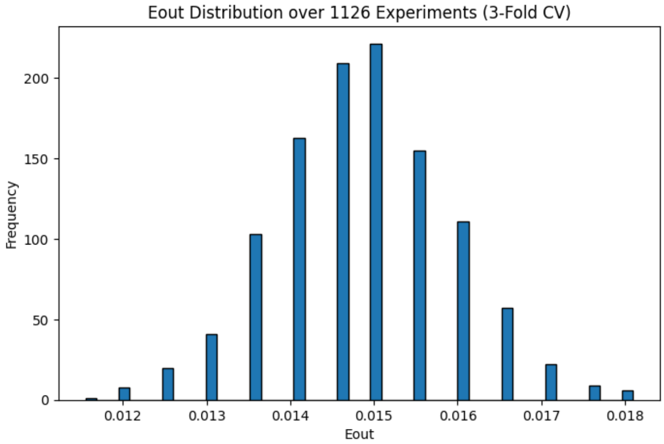
```python
from liblinear.liblinearutil import *
import numpy as np
from sklearn.datasets import load_svmlight_file
import matplotlib.pyplot as plt
from tqdm import tqdm
from sklearn.model_selection import train_test_split

# parameters
EXPERIMENTS = 1126  # 重複實驗次數
lambdas = [-2, -1, 0, 1, 2, 3]
SUB_TRAIN_SIZE = 8000  # 子訓練集大小

# loading data
train_x, train_y = load_svmlight_file('mnist.scale')
test_x, test_y = load_svmlight_file('mnist.scale.t')

train_x = train_x.astype(np.float64)
test_x = test_x.astype(np.float64)

# filter class 2 & 6
def filter_data(x, y, class1=2, class2=6):
    mask = (y == class1) | (y == class2)
    x_filtered = x[mask]
    y_filtered = np.where(y[mask] == class1, 1, -1)  # class 2 : 1，class 6 : -1
    return x_filtered, y_filtered

train_x, train_y = filter_data(train_x, train_y)
test_x, test_y = filter_data(test_x, test_y)

Eout_histogram = []

for i in tqdm(range(EXPERIMENTS), desc="Experiments"):
    # spliting sub train set
    sub_train_x, val_x, sub_train_y, val_y = train_test_split(
        train_x, train_y, train_size=SUB_TRAIN_SIZE)

    # finding best lambda on subset
    best_lambda = None
    best_Eval = float('inf')

    for log_lambda in lambdas:
        current_lambda = 10 ** log_lambda
        c = 1 / (2 * current_lambda)
```

Figure 11b. screenshots of my code

The E_{out} range is roughly between [0.012, 0.018], which is slightly wider than in Problem 10. The distribution is still bell-shaped, with a peak around 0.015. However, the tails are slightly longer. This could because problem 10 is more stable since the entire training set was used to find the best $\lambda*$. Problem 11 shows more instability due to data splitting.

In conclusion, the model in Problem 10 is more stable, with concentrated test errors, making it suitable for evaluating the best model performance. Problem 11 is closer to real world, as data splitting simulates actual training/validation processes, but it introduces some instability.

12.



HW5 > src > q12.py > ...

```python
from liblinear.liblinearutil import *
import numpy as np
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
from tqdm import tqdm

# parameters
EXPERIMENTS = 1126
lambdas = [-2, -1, 0, 1, 2, 3]
K_FOLDS = 3

# loading data
train_x, train_y = load_svmlight_file('mnist.scale')
test_x, test_y = load_svmlight_file('mnist.scale.t')

train_x = train_x.astype(np.float64)
test_x = test_x.astype(np.float64)

# filter class 2 & 6
def filter_data(x, y, class1=2, class2=6):
    mask = (y == class1) | (y == class2)
    x_filtered = x[mask]
    y_filtered = np.where(y[mask] == class1, 1, -1)  # class 2 : 1, class 6 : -1
    return x_filtered, y_filtered

train_x, train_y = filter_data(train_x, train_y)
test_x, test_y = filter_data(test_x, test_y)

Eout_histogram = []

for i in tqdm(range(EXPERIMENTS), desc="Experiments"):
    # 3-fold cross validation
    kf = KFold(n_splits=K_FOLDS, shuffle=True)
    best_lambda = None
    best_Ecv = float('inf')

    for log_lambda in lambdas:
        current_lambda = 10 ** log_lambda
        c = 1 / (2 * current_lambda)
        fold_errors = []

        # split current fold (train & valid set)
        for train_idx, val_idx in kf.split(train_x):
            sub_train_x, val_x = train_x[train_idx], train_x[val_idx]
            sub_train_y, val_y = train_y[train_idx], train_y[val_idx]
```

Figure 12a. histogram of Eout(g)   Figure 12b. screenshots of my code

Problem 12 is very similar to problem 11. Both have their error distribution centers around 0.015 with similar shapes. However, there's still some slightly difference. In Problem 12, E_{out} is mainly concentrated between [0.014, 0.016], making it more focused than in Problem 11. Extreme values also appear less frequently compared to Problem 11, and the tails are shorter. It indicated that problem 12 are more stable, with higher error concentration and almost no extreme values.

This might because problem 11 uses a single random split, and the distribution of the sub-training set and validation set might be uneven. This causes more variation in the selected λ^*, leading to greater fluctuations in the results. Problem 12 uses 3-fold cross-validation, which reduces the impact of uneven distributions by validating in turns. This makes the selected λ^* more representative and the errors more concentrated.

For problems 10 to 12, the full code has been made available on GitHub.
Feel free to check it out if required or interested!