

5.

no collaborators

$$\begin{aligned}
& (P_R) \min_{\underline{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \frac{\lambda}{N} \sum_{i=0}^d \alpha_i w_i^2 \\
& = (P_R) \min_{\underline{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \frac{\lambda}{N} ((\underline{\alpha} \underline{w})^T (\underline{\alpha} \underline{w})), \quad \underline{\alpha} = \begin{bmatrix} \sqrt{\alpha_0} & 0 & \dots & 0 \\ 0 & \sqrt{\alpha_1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\alpha_d} \end{bmatrix} \\
& \Rightarrow \text{solving } \nabla E_{\text{in}}(\underline{w}_{\text{REG}}) + \frac{2\lambda}{N} \underline{\alpha} \underline{w}_{\text{REG}} = \underline{0} \\
& \Rightarrow \frac{2}{N} (\underline{X}^T \underline{X} \underline{w}_{\text{REG}} - \underline{X}^T \underline{y}) + \frac{2\lambda}{N} \underline{\alpha} \underline{w}_{\text{REG}} = \underline{0}, \quad \underline{X} = \underbrace{\begin{bmatrix} -\underline{x}_1^T \\ -\underline{x}_2^T \\ \vdots \\ -\underline{x}_N^T \end{bmatrix}}_{N \times (d+1)}, \quad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \\
& \Rightarrow (\underline{X}^T \underline{X} + \lambda \underline{\alpha}) \underline{w}_{\text{REG}} = \underline{X}^T \underline{y} \\
& \Rightarrow \underline{w}_{\text{REG}} = (\underline{X}^T \underline{X} + \lambda \underline{\alpha})^{-1} \underline{X}^T \underline{y}
\end{aligned}$$

$$\begin{aligned}
\underline{\tilde{x}}_k &= [0, \dots, 0, \sqrt{\lambda \alpha_{k-1}}, 0, \dots, 0]^T \in \mathbb{R}^{d+1} \\
&= \sqrt{\lambda \alpha_{k-1}} \underline{e}_{k-1}
\end{aligned}$$

$$\begin{aligned}
& \frac{1}{N+K} \left(\sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \sum_{k=1}^K (\underline{w}^T \underline{\tilde{x}}_k - \tilde{y}_k)^2 \right) \\
&= \frac{1}{N+K} \left(\sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \sum_{k=1}^K (\underline{w}^T \sqrt{\lambda \alpha_{k-1}} \underline{e}_{k-1})^2 \right) \\
&= \frac{1}{N+K} \left(\sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \sum_{k=1}^K (w_{k-1} \sqrt{\lambda \alpha_{k-1}})^2 \right) \\
&= \frac{1}{N+d+1} \left(\sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \lambda \sum_{k=1}^{d+1} \alpha_{k-1} w_{k-1}^2 \right) \\
&= \frac{1}{N} \left(\sum_{n=1}^N (\underline{w}^T \underline{x}_n - y_n)^2 + \lambda \sum_{i=0}^d \alpha_i w_i^2 \right) \times \frac{N}{N+d+1}
\end{aligned}$$

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

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

which is same as (P_R)

\therefore optimal \underline{w}^* obtained by (P_V) is the same as the optimal solution

(= $\underline{w}_{\text{REG}} = (\underline{X}^T \underline{X} + \lambda \underline{\alpha})^{-1} \underline{X}^T \underline{y}$) obtained by solving (P_R)

6.

no collaborators

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

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

Let $\underline{w}_{\text{aug}}$ is the minimizer of $\tilde{E}_{\text{aug}}(\underline{w})$

$$\nabla \tilde{E}_{\text{aug}}(\underline{w}_{\text{aug}}) = \underline{0}$$

$$\Rightarrow \underline{H} (\underline{w}_{\text{aug}} - \underline{w}^*) + \frac{2\lambda}{N} \underline{w}_{\text{aug}} = \underline{0}$$

$$\Rightarrow \left(\underline{H} + \frac{2\lambda}{N} \underline{I} \right) \underline{w}_{\text{aug}} = \underline{H} \underline{w}^*$$

$$\Rightarrow \underline{w}_{\text{aug}} = \left(\underline{H} + \frac{2\lambda}{N} \underline{I} \right)^{-1} \underline{H} \underline{w}^* \quad \left[\because \underline{H} + \frac{2\lambda}{N} \underline{I} \text{ is positive definite} \right]$$

7.

collaborator:
B11901073 林禹融

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

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

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

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

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

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

$$\begin{aligned} \sum_{n=1}^N y_n &= 0 \\ \Rightarrow \sum_{n=1}^{N-K} y_n + \sum_{n=N-K+1}^N y_n &= 0 \\ \Rightarrow \sum_{n=N-K+1}^N y_n &= - \sum_{n=1}^{N-K} y_n \\ &= -(N-K) \bar{y} \end{aligned}$$

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

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

$$= \sigma^2 + \frac{2N-K}{K} \times \mathbb{E}(\bar{y}^2)$$

$$= \sigma^2 + \frac{2N-K}{K} \text{Var}(\bar{y})$$

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

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

$$\begin{aligned} \text{Var}(\bar{y}) &= \text{Var}\left(\frac{1}{N-K} \sum_{n=1}^{N-K} y_n\right) \\ &= \left(\frac{1}{N-K}\right)^2 \text{Var}\left(\sum_{n=1}^{N-K} y_n\right) \\ &= \left(\frac{1}{N-K}\right)^2 \sum_{n=1}^{N-K} \text{Var}(y_n) \\ &= \left(\frac{1}{N-K}\right)^2 (N-K) \sigma^2 \\ &= \frac{1}{N-K} \sigma^2 \end{aligned}$$

8. P 17 of lecture 15

no collaborators

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

$$\Rightarrow N^2 E_{in}(\omega^*) = \frac{1}{N} \sum_{n=1}^N \left(\sum_{n=1}^N y_n - N y_n \right)^2$$

$$E_{locv}(\bar{A}_{avg}) = \frac{1}{N} \sum_{n=1}^N e_n$$
$$= \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{N-1} \sum_{i \neq n} y_i - y_n \right)^2$$
$$= \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{N-1} \left(\sum_{i=1}^N y_i - y_n \right) - y_n \right)^2$$
$$= \left(\frac{1}{N-1} \right)^2 \left[\frac{1}{N} \sum_{n=1}^N \left(\sum_{i=1}^N y_i - y_n - (N-1)y_n \right)^2 \right]$$
$$= \left(\frac{1}{N-1} \right)^2 \left[\frac{1}{N} \sum_{n=1}^N \left(\sum_{i=1}^N y_i - N y_n \right)^2 \right]$$
$$= \left(\frac{N}{N-1} \right)^2 E_{in}(\omega^*) \text{ for } N \geq 2$$

Q.E.D.

9.

no collaborators

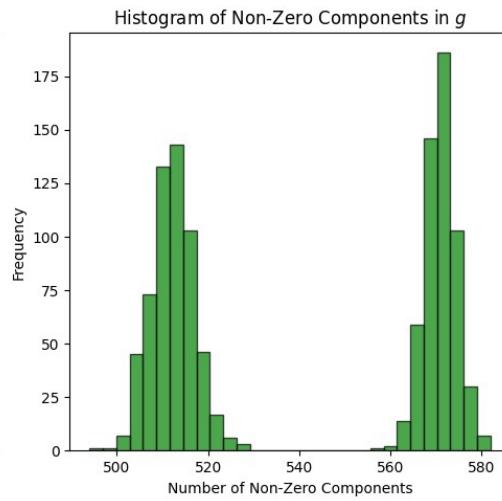
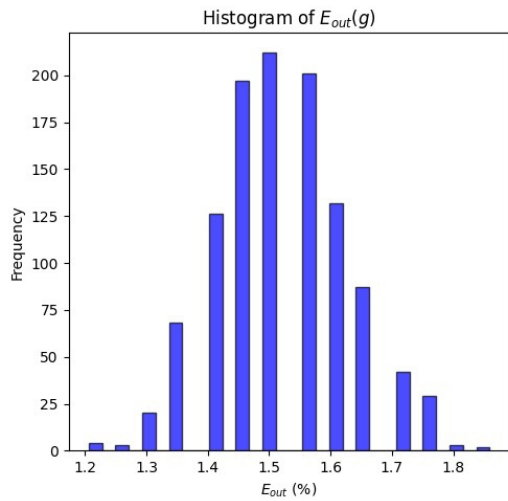
$$E_{out}(g) = E_{out}(g_c)$$

$$\Rightarrow pE_- + (1-p)E_+ = p$$

$$\Rightarrow p(E_- - E_+ - 1) = -E_+$$

$$\Rightarrow p = \frac{E_+}{1 + E_+ - E_-}$$

10.



collaborators:

B11201009 黃勤元

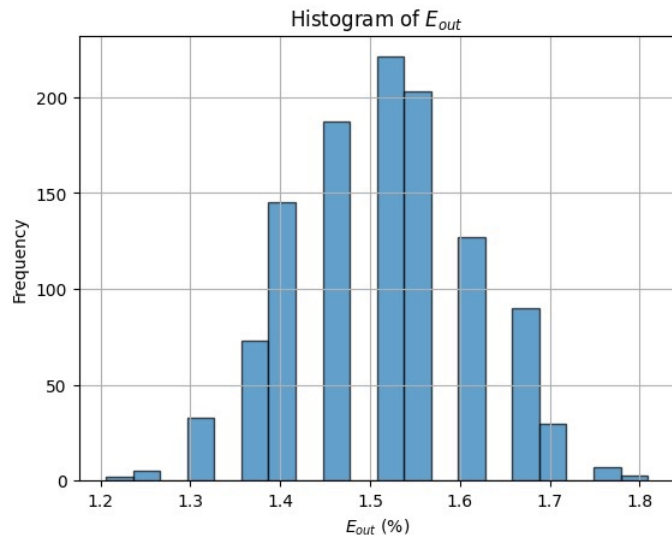
B11901073 林禹融

```

hw5-10.py > ...
1 import os
2 import requests
3 import bz2
4 import numpy as np
5 import scipy.sparse
6 from liblinear.liblinearutil import *
7 import matplotlib.pyplot as plt
8
9 # Step 1: Download and decompress the data
10 > def download_and_extract(url, dest_path):
11
12
13
14
15
16
17
18
19
20 > def decompress_bz2(file_path, output_path):
21
22
23
24
25
26
27
28
29
30 train_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.bz2"
31 test_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.t.bz2"
32 train_file_compressed = "mnist.scale.bz2"
33 test_file_compressed = "mnist.scale.t.bz2"
34 train_file = "mnist.scale"
35 test_file = "mnist.scale.t"
36
37 download_and_extract(train_url, train_file_compressed)
38 download_and_extract(test_url, test_file_compressed)
39 decompress_bz2(train_file_compressed, train_file)
40 decompress_bz2(test_file_compressed, test_file)
41
42 # Step 2: Load and preprocess the data
43 y_train, X_train = svm_read_problem(train_file, return_scipy=True)
44 y_test, X_test = svm_read_problem(test_file, return_scipy=True)
45
46 # Filter for classes 2 and 6, and relabel them for binary classification
47 train_mask = np.isin(y_train, [2, 6])
48 test_mask = np.isin(y_test, [2, 6])
49 y_train, X_train = y_train[train_mask], X_train[train_mask]
50 y_test, X_test = y_test[test_mask], X_test[test_mask]
51 y_train = np.where(y_train == 2, 1, -1)
52 y_test = np.where(y_test == 2, 1, -1)
53
54 > def align_features(train, test):
55
56
57
58
59
60
61
62
63
64
65 # Align feature dimensions between training and test sets
66 X_train, X_test = align_features(X_train, X_test)
67
68
69 # Scale the data using csr_scale
70 scale_param = csr_find_scale_param(X_train, lower=0)
71 X_train = csr_scale(X_train, scale_param)
72 X_test = csr_scale(X_test, scale_param)
73 | Ctrl+L to chat, Ctrl+K to generate
74 # Step 3: Define helper functions
75 def lambda_to_C(lmbda):
76     return 1 / lmbda
77
78 lambdas = [0.01, 0.1, 1, 10, 100, 1000]
79 N = len(y_train)
80 best_lambda, min_error = None, float('inf')
81 errors_in = []
82
83 # Step 4: Find the best lambda and repeat experiments with 1126 seeds
84 random_seeds = range(1126)
85 errors_out, non_zero_counts = [], []
86
87 for seed in random_seeds:
88     for lmbda in lambdas:
89         C = lambda_to_C(lmbda)
90         model = train(y_train, X_train, f's 6 -c {C} -B 1 -q')
91         p_acc, _ = predict(y_train, X_train, model)
92         E_in = 100 - p_acc[0]
93         errors_in.append(E_in)
94         if E_in < min_error or (E_in == min_error and lmbda > best_lambda):
95             best_lambda, min_error = lmbda, E_in
96
97 C = lambda_to_C(best_lambda)
98 np.random.seed(seed)
99 model = train(y_train, X_train, f's 6 -c {C} -B 1 -q')
100 p_acc, _ = predict(y_test, X_test, model)
101 E_out = 100 - p_acc[0] # accuracy to error
102 errors_out.append(E_out)
103
104 # Access weights and bias terms
105 weights, bias = model.get_decfun()
106 non_zero_counts.append(np.sum(np.abs(weights) > 1e-6))
107
108 # Step 6: Plot histograms
109 plt.figure(figsize=(10, 5))
110 plt.subplot(1, 2, 1)
111 plt.hist(errors_out, bins=30, color="blue", edgecolor="black", alpha=0.7)
112 plt.title("Histogram of  $E_{out}(g)$ ")
113 plt.xlabel(" $E_{out}$  (%)")
114 plt.ylabel("Frequency")
115
116 plt.subplot(1, 2, 2)
117 plt.hist(non_zero_counts, bins=30, color="green", edgecolor="black", alpha=0.7)
118 plt.title("Histogram of Non-Zero Components in  $g$ ")
119 plt.xlabel("Number of Non-Zero Components")
120 plt.ylabel("Frequency")
121
122 plt.tight_layout()

```

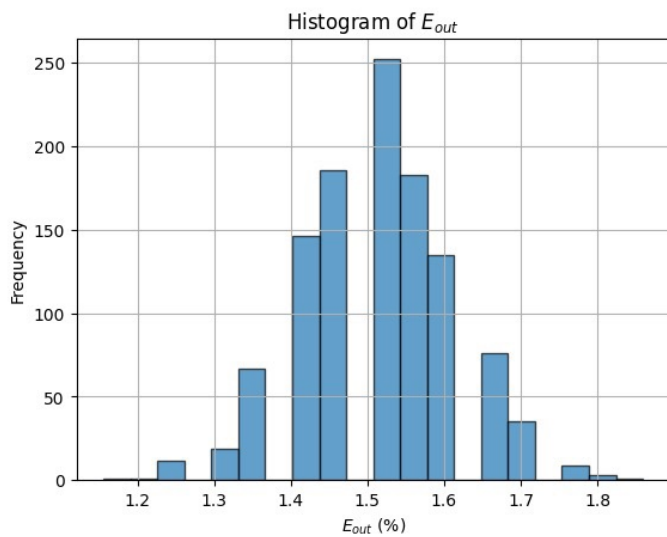
11. collaborator: B11901073 林禹融



Both histograms are center around $E_{out} \approx 1.5\%$, suggesting that performance is similar between both methods. While the distribution in Problem 11 is slightly wider than that in Problem 10, reflecting that cross-validation approach introduces additional variability because of the random splits into sub-training and validation sets.

```
hw5-11.py > ...
1 import os
2 import requests
3 import bz2
4 import numpy as np
5 from liblinear.liblinearutil import *
6 from sklearn.model_selection import train_test_split
7 import matplotlib.pyplot as plt
8 from scipy.sparse import csr_matrix, hstack
9
10 # Constants
11 LAMBDA_VALUES = [0.01, 0.1, 1, 10, 100, 1000]
12 N_EXPERIMENTS = 1126
13 TRAIN_SIZE = 8000
14
15 # Step 1: Download and decompress the data
16 > def download_and_extract(url, dest_path):-
17
18 > def decompress_bz2(file_path, output_path):-
19
20 train_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.bz2"
21 test_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.t.bz2"
22 train_file_compressed = "mnist.scale.bz2"
23 test_file_compressed = "mnist.scale.t.bz2"
24 train_file = "mnist.scale"
25 test_file = "mnist.scale.t"
26
27 download_and_extract(train_url, train_file_compressed)
28 download_and_extract(test_url, test_file_compressed)
29 decompress_bz2(train_file_compressed, train_file)
30 decompress_bz2(test_file_compressed, test_file)
31
32 # Step 2: Load and preprocess the data
33 y_train, X_train = svm_read_problem(train_file, return_scipy=True)
34 y_test, X_test = svm_read_problem(test_file, return_scipy=True)
35
36 # Filter for classes 2 and 6, and relabel them for binary classification
37 train_mask = np.isin(y_train, [2, 6])
38 test_mask = np.isin(y_test, [2, 6])
39 y_train, X_train = y_train[train_mask], X_train[train_mask]
40 y_test, X_test = y_test[test_mask], X_test[test_mask]
41 y_train = np.where(y_train == 2, 1, -1)
42 y_test = np.where(y_test == 2, 1, -1)
43
44 > def align_features(train, test):-
45
46 # Align feature dimensions between training and test sets
47 X_train, X_test = align_features(X_train, X_test)
48
49 # Scale the data using csr_scale
50 scale_param = csr_find_scale_param(X_train, lower=0)
51 X_train = csr_scale(X_train, scale_param)
52 X_test = csr_scale(X_test, scale_param)
53
54 # Function to calculate error
55 def calculate_error(y_true, y_pred):
56     return np.mean(y_true != y_pred) * 100
57
58 # Prepare for histogram
59 E_out_values = []
60
61 # Main loop for 1126 experiments
62 for experiment in range(N_EXPERIMENTS):
63     np.random.seed(experiment)
64     # Step 1: Split data into sub-training and validation sets
65     X_subtrain, X_val, y_subtrain, y_val = train_test_split(
66         X_train, y_train, train_size=TRAIN_SIZE, random_state=experiment
67     )
68     best_lambda, best_E_val = None, float('inf')
69
70     # Step 2: Evaluate each lambda
71     for lambda_value in LAMBDA_VALUES:
72         C = 1 / lambda_value
73         model = train(y_subtrain, X_subtrain, f'-s 6 -c {C} -B 1 -q')
74         p_val, _, _ = predict(y_val, X_val, model, '-q')
75         E_val = calculate_error(y_val, p_val)
76         if E_val < best_E_val or (E_val == best_E_val and lambda_value > best_lambda):
77             best_E_val = E_val
78             best_lambda = lambda_value
79
80     # Step 3: Re-train with the best lambda on the full training set
81     C = 1 / best_lambda
82     final_model = train(y_train, X_train, f'-s 6 -c {C} -B 1 -q')
83
84     # Step 4: Evaluate E_out on the test set
85     p_test, _, _ = predict(y_test, X_test, final_model, '-q')
86     E_out = calculate_error(y_test, p_test)
87     E_out_values.append(E_out)
88
89 # Step 5: Plot histogram of E_out
90 plt.hist(E_out_values, bins=20, edgecolor='black', alpha=0.7)
91 plt.title('Histogram of  $E_{out}$  (%)')
92 plt.xlabel('E_out (%)')
93 plt.ylabel('Frequency')
94 plt.grid(True)
95 plt.show()
```


12. collaborator: B11901073 林禹融



Both histograms are center around $E_{out} \approx 1.5\%$, suggesting that performance is similar between both methods. Additionally, histogram in Problem 12 is more concentrated around the mean. The 3-fold cross validation may be generally a better approach to minimize variance in model selection process

```
hws-12.py > ...
1 import os
2 import requests
3 import bz2
4 import numpy as np
5 from liblinear.liblinearutil import *
6 from sklearn.model_selection import KFold
7 import matplotlib.pyplot as plt
8 from scipy.sparse import csr_matrix, hstack
9
10 # Constants
11 LAMBDA_VALUES = [0.01, 0.1, 1, 10, 100, 1000]
12 N_EXPERIMENTS = 1126
13 K_FOLDS = 3 # 3-fold cross-validation
14
15 # Step 1: Download and decompress the data
16 > def download_and_extract(url, dest_path):~
25
26 > def decompress_bz2(file_path, output_path):~
35
36 train_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.bz2"
37 test_url = "https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.t.bz2"
38 train_file_compressed = "mnist.scale.bz2"
39 test_file_compressed = "mnist.scale.t.bz2"
40 train_file = "mnist.scale"
41 test_file = "mnist.scale.t"
42
43 download_and_extract(train_url, train_file_compressed)
44 download_and_extract(test_url, test_file_compressed)
45 decompress_bz2(train_file_compressed, train_file)
46 decompress_bz2(test_file_compressed, test_file)
47
48 # Step 2: Load and preprocess the data
49 y_train, X_train = svm_read_problem(train_file, return_scipy=True)
50 y_test, X_test = svm_read_problem(test_file, return_scipy=True)
51
52 # Filter for classes 2 and 6, and relabel them for binary classification
53 train_mask = np.isin(y_train, [2, 6])
54 test_mask = np.isin(y_test, [2, 6])
55 y_train, X_train = y_train[train_mask], X_train[train_mask]
56 y_test, X_test = y_test[test_mask], X_test[test_mask]
57 y_train = np.where(y_train == 2, 1, -1)
58 y_test = np.where(y_test == 2, 1, -1)
59
60 > def align_features(train, test):~
71
72 # Align feature dimensions between training and test sets
73 X_train, X_test = align_features(X_train, X_test)
74
75 # Scale the data using csr_scale
76 scale_param = csr_find_scale_param(X_train, lower=0)
77 X_train = csr_scale(X_train, scale_param)
78 X_test = csr_scale(X_test, scale_param)
79
80 # Function to calculate error
81 > def calculate_error(y_true, y_pred):~
83
84 # Function for 3-fold cross-validation
85 def cross_validate(X, y, lambda_value):
86     kf = KFold(n_splits=K_FOLDS, shuffle=True, random_state=None)
87     total_error = 0
88
89     for train_index, val_index in kf.split(X):
90         X_train_fold, X_val_fold = X[train_index], X[val_index]
91         y_train_fold, y_val_fold = y[train_index], y[val_index]
92
93         C = 1 / lambda_value
94         model = train(y_train_fold, X_train_fold, f'-s 6 -c {C} -B 1 -q')
95
96         p_val, _, _ = predict(y_val_fold, X_val_fold, model, '-q')
97         total_error += calculate_error(y_val_fold, p_val)
98
99     # Return average cross-validation error
100     return total_error / K_FOLDS
101
102 # Prepare for histogram
103 E_out_values = []
104
105 # Main loop for 1126 experiments
106 for experiment in range(N_EXPERIMENTS):
107     np.random.seed(experiment)
108
109     best_lambda, best_E_CV = None, float('inf')
110
111     # Step 1: Perform 3-fold cross-validation for each lambda
112     for lambda_value in LAMBDA_VALUES:
113         E_CV = cross_validate(X_train, y_train, lambda_value)
114
115         # Update best lambda
116         if E_CV < best_E_CV or (E_CV == best_E_CV and lambda_value > best_lambda):
117             best_E_CV = E_CV
118             best_lambda = lambda_value
119
120     # Step 2: Re-train with the best lambda on the full training set
121     C = 1 / best_lambda
122     final_model = train(y_train, X_train, f'-s 6 -c {C} -B 1 -q')
123
```