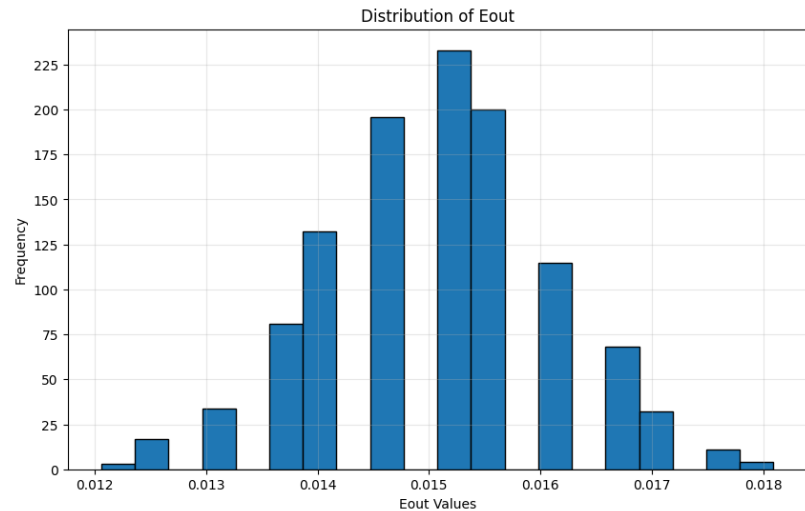


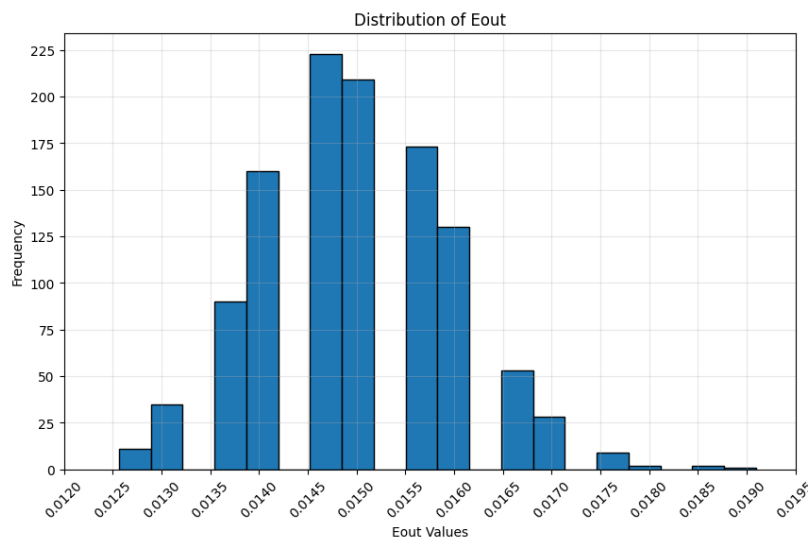
HTML homework 5: q12 report

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 $D \setminus D_i$

→ 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

```

1  def run_single_experiment(experiment):
2      np.random.seed(experiment)
3
4      total_size = len(X_train)
5      indices = np.random.permutation(total_size)
6
7      fold_size = total_size // 3
8
9      fold1_indices = indices[:fold_size]
10     fold2_indices = indices[fold_size:2*fold_size]
11     fold3_indices = indices[2*fold_size:]
12
13     X_fold1 = [X_train[i] for i in fold1_indices]
14     X_fold2 = [X_train[i] for i in fold2_indices]
15     X_fold3 = [X_train[i] for i in fold3_indices]
16
17     y_fold1 = [y_train[i] for i in fold1_indices]
18     y_fold2 = [y_train[i] for i in fold2_indices]
19     y_fold3 = [y_train[i] for i in fold3_indices]
20
21     min_Ecv = np.inf
22     opt_log10_lambda = 0
23     for log10_lambda in (-2, -1, 0, 1, 2, 3):
24         each_fold_as_valid_err = []
25         c = 1 / (10 ** log10_lambda)
26
27         for i in range(1,4):
28             if i == 1:
29                 X_subtrain = X_fold2 + X_fold3
30                 y_subtrain = y_fold2 + y_fold3
31                 X_validation = X_fold1
32                 y_validation = y_fold1
33             elif i == 2:
34                 X_subtrain = X_fold1 + X_fold3
35                 y_subtrain = y_fold1 + y_fold3
36                 X_validation = X_fold2
37                 y_validation = y_fold2
38             else:
39                 X_subtrain = X_fold1 + X_fold2
40                 y_subtrain = y_fold1 + y_fold2
41                 X_validation = X_fold3
42                 y_validation = y_fold3
43
44             subtrain_prob = problem(y_subtrain, X_subtrain)
45             param = parameter('-s 6 -c ' + str(c))
46             model_by_subtrain = train(subtrain_prob, param)
47
48             validation_label, _, _ = predict(y_validation, X_validation, model_by_subtrain)
49             validation_err = ZeroOneError(validation_label, y_validation)
50             each_fold_as_valid_err.append(validation_err)
51
52     Ecv_each_lambda = np.mean(each_fold_as_valid_err)

```

```

53
54     if Ecv_each_lambda == min_Ecv:
55         opt_log10_lambda = max(opt_log10_lambda, log10_lambda) # break tie by choosing larger lambda
56     elif Ecv_each_lambda < min_Ecv:
57         min_Ecv = Ecv_each_lambda
58         opt_log10_lambda = log10_lambda
59
60     whole_train_prob = problem(y_train, X_train)
61     param_with_opt_lambda = parameter('-s 6 -c ' + str(1 / (10 ** opt_log10_lambda)))
62     whole_train_model = train(whole_train_prob, param_with_opt_lambda)
63
64     test_label, _, _ = predict(y_test, X_test, whole_train_model)
65     return ZeroOneError(test_label, y_test)
66
67 n_jobs = -1
68 Eouts = Parallel(n_jobs=n_jobs)(
69     delayed(run_single_experiment)(experiment)
70     for experiment in tqdm(range(1126))
71 )

```

✓ 166m 7.7s