COSI123A HW4

HyunJae Pi

October 26, 2020

Task: There are 10 data sets (5 in folder a and 5 in fold b) in the attached zip file. Each data set has a training subset (for example, Train_1a.csv) and a test subset (for example, Test_1a.csv). Build a regression model using the training subset and apply the model to the test subset. Each row in a data file is a sample. The training data files contain the desired outputs. The headers of the data files indicate the feature columns and the output columns. The test files do not contain the outputs. Save all your prediction results in a file named "hw4_result.csv" using the provided template file "hw4_result.csv". The order of the predictions in the result file should be the same to the order of their samples in the test file.

Requirements: In addition to the result file "hw4_result.csv", please submit your codes and a written report explaining your preprocessing method if any and your regression model. You can use any existing package/function, and should clearly specify your chosen settings and explain why your choices.

Grading: The score will be calculated in the following way. Report takes 30% and prediction performance takes another 70%. For prediction part, first, the root-mean-square error (RMSE) is calculated for each submission. The RMSEs from all students will be ranked. The best prediction (i.e., the lowest RMSE) will get 70. The scores of the rest predictions will be linearly scaled between 66.5 and 35 as: 66.5 – (rank-1)*(66.5 – 35)/(number of submissions-1). You will receive 0 if you produce random predictions or do not submit your prediction results.

Deadline: 10/26 11:59pm (late penalty 15 pts)
Cutoff date: 10/28 11:59pm

Tips: There may exist outliers or redundant variables in the data, you can use any kind of preprocessing method, prediction model and parameter settings you want, but linear regression model is recommended. You may want to use the same method/setting for datasets in folder a, and the same method/setting for datasets in folder b.

1. Base model performance

First, a linear regression model was run on the TEST dataset without removing outliers and regularization. Here, both column and row names are folder names in HW4data, therefore filenames can be identified using this information. For instance, 'a' and '3' gives the filename 'Train_1a.csv.'

	a	**b**
1	.28	.24
2	.44	.37
3	.22	.33
4	.41	.34
5	.28	.41

2. Removing outliers and redundant features

a). Outlier detection

These are techniques that I tried to detect outliers.

- Standardization (z-score): I checked data and they were already standardized.

- boxplot and 1.5IQR(inter-quantile range): The idea is to detect data points outside of 1.5 IQR. It didn't reduce RMSE much.

- PCA: PCA is one outlier detection technique that was covered in the class. However, it didn't reduce RMSE much. **- SVM (python package):** I used the one in python sklearn package. The idea of the algorithm is that it fits a curve and center of data with a few mistakes. The data points outside of this boundary can be considered as outliers. My pilot simulation showed that it reduced RMSE.

- Isolation Forest (python package): I also used the one in python sklearn package. The intuition of the algorithm is that anormal data points are easier to be separated than normal data points. Recursively, partitions are generated on the data points by randomly selecting an attribute. Then a split value is chosen between min and max. Anormalies require fewer random partitionings compared to normal data, therefore they can be more easily isolated. I applied isolation forest to the training data and it also reduced RMSE.

b). Regularization

In order to remove redundant features, both L2 (Ridge) and L1 (Lasso) regularizations were tried. Both methods reduced RMSE.

3. Optimal hyper-parameters

Among these options, different combinations of SVM and Isolation Forest with L1 and L2 regularization were tested. Hyperparameters were scanned in 2D space to find the optimal values that give a minimal RMSE.

a) Isolation Forest & L1

The following table shows contamination factors for Iosolation Forest and regularization constants (alpha) for L1 in each data. For instance, column a & row 3 indicates that contamination factor=0.42 and alpha=0.01 give the minimal RMSE=0.17 for 'Test_3a.csv.' All RMSEs are smaller than the values calculated from the base model.

	a (contamination, alpha, RMSE)	b (contamination, alpha, RMSE)
1	(0.25, 0.02, 0.26)	(0.48, 0.01, 0.13)
2	(0.03, 0.03, 0.41)	(0.47, 0.02, 0.31)
3	(0.47, 0.01, 0.17)	(0.00, 0.01, 0.32)
4	(0.44, 0.01, 0.38)	(0.05, 0.03, 0.32)
5	(0.37, 0.01, 0.19)	(0.28, 0.01, 0.39)

b) Isolation Forest & L2

This is the result for Isolation Forest and L2 regularization. RMSEs are smaller than that of the base model. Compared to the result of a) Isolation Forest & L1 regularization, RMSEs are slightly worse.

	a (contamination, alpha, RMSE)	b (contamination, alpha, RMSE)
1	(0.10, 0.49, 0.26)	(0.49, 0.49, 0.14)
2	(0.23, 0.49, 0.42)	(0.41, 0.49, 0.32)
3	(0.45, 0.49, 0.18)	(0.00, 0.49, 0.33)
4	(0.03, 0.49, 0.39)	(0.00, 0.49, 0.34)
5	(0.47, 0.49, 0.20)	(0.43, 0.49, 0.38)

c) SVM & L1

This is the result for SVM outlier detection and L1 regularization. RMSEs are smaller than that of the base model. RMSEs are comparable but they are slightly better or worse for each datasets).

	a (contamination, alpha, RMSE)	b (contamination, alpha, RMSE)
1	(0.26, 0.02, 0.27)	(0.43, 0.02, 0.19)
2	(0.19, 0.03, 0.40)	(0.35, 0.00, 0.31)
3	(0.39, 0.01, 0.17)	(0.14, 0.01, 0.32)
4	(0.10, 0.05, 0.38)	(0.16, 0.03, 0.34)
5	(0.39, 0.08, 0.22)	(0.14, 0.04, 0.38)

d) SVM & L2

This is the result for SVM outlier detection and L2 regularization. RMSEs are smaller than that of the base model. Again, RMSEs are comparable.

	a (contamination, alpha, RMSE)	b (contamination, alpha, RMSE)
1	(0.09, 0.49, 0.27)	(0.48, 0.49, 0.20)
2	(0.19, 0.49, 0.40)	(0.35, 0.49, 0.31)
3	(0.39, 0.49, 0.18)	(0.13, 0.49, 0.33)
4	(0.08, 0.49, 0.39)	(0.16, 0.49, 0.35)
5	(0.47, 0.49, 0.22)	(0.46, 0.49, 0.39)

Taken together, the combination of Isolation Forest & L1 regularization(Lasso) gives the best result except Training_2a.csv.

4. Prediction on Test datasets.

Using L1 regularization and Isolation Forest and the optimal hyper-parameters (values in table in a)), test datasets were tested. The predicted values were saved and uploaded in Latte.

5. Codes

Code 1 : tests multiple combination of regularization and outlier detection models and find the optimal hyper-parameters

```
"""
Created on Fri Oct 23 15:25:04 2020

hw4 code 1 --
This program tests multiple combination of regularization and outlier detection models and fi

@author: HyunJae Pi, hyunpi@brandeis.edu
"""

# import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM

import warnings
warnings.filterwarnings('ignore')

# calculate RMSE using K-fold cross validation
def rmse_cross_validation(model, x_train, y_train, folds):
    mse = cross_val_score(model, x_train, y_train, scoring='neg_mean_squared_error', cv=folds
    return np.sqrt(np.abs(mse))

# remove indices of outliers using Isolation Forest
def indices_inliers_by_isolation_forest(data, contamination_factor):
    iso = IsolationForest(contamination=contamination_factor)
    indices_outliers = iso.fit_predict(data)
    # select all rows that are not outliers
    mask = indices_outliers != -1
    return mask

# remove indices of outliers using SVM
def indices_inliers_by_svm(data, nu):
    ocs = OneClassSVM(nu=nu)
    indices_outliers = ocs.fit_predict(data)
    mask = indices_outliers != -1 # select all rows that are not outliers
    return mask

# get filenames in the HW4data folder
def filenames():
    folder1 = ['a','b']
    folder2 = ['1','2','3','4','5']
    str0 = './HW4data/'
```

```
    filenames=[]
    for f1 in folder1:
        if f1 == 'a':
            str_tmp =""
        else:
            str_tmp ="X"

        for f2 in folder2:
                filenames.append(str0 + f1 + '/' + f2 + '/' + 'Train'+str_tmp+'_'+f2+f1+'.csv

    return filenames


# base model: linear regression without outlier detection and regularization
def base_model(filename, folds):
    data = pd.read_csv(filename).to_numpy()
    X = data[:, 0:-1]
    y = data[:, -1]
    return np.average(rmse_cross_validation(LinearRegression(), X, y, folds))

# find the combination of optimal hyper-parameters
def find_hyperparams_for_min_rmse(filename, regularization_method, regularization_factor_range
    data = pd.read_csv(filename).to_numpy()
    X = data[:, 0:-1]
    y = data[:, -1]

    # intialization
    ave_rmse = np.empty((len(outlier_detection_factor_range),len(regularization_factor_range)
    #const_outlier_detection = np.empty((len(outlier_detection_factor_range),len(regularizati
    #const_regularization = np.empty((len(outlier_detection_factor_range),len(regularization_

    # outlier detection factor
    i=0
    for odf in outlier_detection_factor_range:
        if outlier_detection_method == 'IF': # isolation forest
            mask = indices_inliers_by_isolation_forest(X, odf)
        elif outlier_detection_method == 'SVM': # SVM
            mask = indices_inliers_by_svm(X, odf)
        else:
            warning("only two options: IF or SVM")
        X_, y_ = X[mask, :], y[mask]

        # regularization factor
        j=0
        for rf in regularization_factor_range:
            if regularization_method == 'L1':
                rmse = rmse_cross_validation(Lasso(alpha = rf, max_iter=1000, tol=0.0001), X_
            elif regularization_method == 'L2':
                rmse = rmse_cross_validation(Ridge(alpha = rf, max_iter=1000, tol=0.0001), X_
            else:
                warning("only two options: L1 or L2")
            ave_rmse[i][j] =np.average(rmse)
            j=j+1
        i=i+1

    ij = np.where(ave_rmse == np.min(ave_rmse))
```

```
    return (outlier_detection_factor_range[ij[0]][0], regularization_factor_range[ij[1]][0],


# k-fold cross validation
folds = KFold(n_splits = 5, shuffle = True, random_state = 1001)

# 1. testing a base model
rmse_base=[]
for f in filenames():
    rmse_base.append(base_model(f, folds))
print("\nBase model\n")
print(rmse_base)


# 2. L1 and Isolation Forest
contamination = np.arange(0, 0.5, 0.01) # contamination range = [0, 0.5]
alpha = np.arange(0, 0.5, 0.01) # regularization factor
RMSEs=[]
for f in filenames():
    rmse = find_hyperparams_for_min_rmse(f, 'L1', alpha, 'IF', contamination, folds)
    RMSEs.append(rmse)
print("\nL1 & Isolation Forest\n")
print(RMSEs)


# 3. L2 and Isolation Forest ---------- check alpha_range
contamination = np.arange(0, 0.5, 0.01) # contamination range = [0, 0.5]
alpha = np.arange(0, 0.5, 0.01) # regularization factor
RMSEs=[]
for f in filenames():
    rmse = find_hyperparams_for_min_rmse(f, 'L2', alpha, 'IF', contamination, folds)
    RMSEs.append(rmse)
print("\nL2 & Isolation Forest\n")
print(RMSEs)


# 4. L1 and SVM
nu = np.arange(0.01, .5, .01) # SVM factor
alpha = np.arange(0, 0.5, 0.01) # regularization factor
RMSEs=[]
for f in filenames():
    rmse = find_hyperparams_for_min_rmse(f, 'L1', alpha, 'SVM', nu, folds)
    RMSEs.append(rmse)
print("\nL1 & SVM\n")
print(RMSEs)


# 5. L2 and SVM
nu = np.arange(0.01, .5, .01) # SVM factor
alpha = np.arange(0, 0.5, 0.01) # regularization factor
RMSEs=[]
for f in filenames():
    rmse = find_hyperparams_for_min_rmse(f, 'L2', alpha, 'SVM', nu, folds)
    RMSEs.append(rmse)
print("\nL2 & SVM\n")
print(RMSEs)
```

Code 2 : predict outputs from the test datasets

```
"""
Created on Mon Oct 26 17:44:19 2020

hw4 code 2 --
Using L1 regularization and  Isolation Forest,
this program predicts output values from test datasets.

@author: hyunjaepi, hyunpi@brandeis.edu
"""

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import IsolationForest
from sklearn.metrics import mean_squared_error
from numpy import savetxt

import warnings
warnings.filterwarnings('ignore')

# identify inlier indices using Isolation Forest (outlier detection)
def indices_inliers_by_isolation_forest(data, contamination_factor):
    iso = IsolationForest(contamination=contamination_factor)
    indices_outliers = iso.fit_predict(data)
    mask = indices_outliers != -1 # select all rows that are not outliers
    return mask

# read train and test data
def read_train_and_test_files():
    folder1 = ['a','b']
    folder2 = ['1','2','3','4','5']
    str0 = './HW4data/'

    filename_train = []
    filename_test = []
    for f1 in folder1:
        if f1 == 'a':
            str_tmp =""
        else:
            str_tmp ="X"

        for f2 in folder2:
                filename_train.append(str0 + f1 + '/' + f2 + '/' + 'Train'+str_tmp+'_'+f2+f1+
                filename_test.append(str0 + f1 + '/' + f2 + '/' + 'Test'+'_'+f2+f1+'.csv')

    return [filename_train, filename_test]

# predict outputs from test datasets
def prediction_on_test_data(filenames, const_regularization, const_outlier_detection):
    # file names
    filename_train = filenames[0];
    filename_test = filenames[1];
```

```
    # read training data
    data = pd.read_csv(filename_train).to_numpy()
    X = data[:, 0:-1]
    y = data[:, -1]

    # outlier detection
    mask = indices_inliers_by_isolation_forest(X, const_outlier_detection)
    x_train, y_train = X[mask, :], y[mask]

    # L1 regression w/ regularization
    lasso = Lasso(alpha = const_regularization)
    lasso.fit(x_train, y_train)

    # prediction on test dataset
    x_test = pd.read_csv(filename_test).to_numpy()
    y_pred = lasso.predict(x_test)

    return y_pred


# run & save results
const_regularization = [.02, .03, .01, .01, .01, .01, .02, .01, .03, .01]
const_outlier_detection = [.25, .03, .47, .44, .37, .48, .47, .00, .05, .28]
filenames = read_train_and_test_files()

predictions = np.empty([15, 10])
i = 0;
for i in range(10):
    predictions[:, i] = prediction_on_test_data([filenames[0][i],filenames[1][i]], const_regu
    i = i + 1

savetxt('results.csv', predictions, delimiter=',')
```