mini project2

XIN Hao
Jian Xun
Yu Jinxing
Department of Computer Science

Abstract

This is a short report for the mini-project2 of the class Math6380 in HKUST. The project we choose is a regression problem.

1 introduction

We choosed combinatorial drug 20 efficacy data and participated in the kaggle inclass contest. We are the team "3654". Our best submission gets **0.01193** Mean-Square-Error. The dataset contains 140 cancer cell line samples in response to a combination of 20 drugs in 4 dosage levels. There are the discrete dosage levels of 20 drugs, and the real valued response as viability measured by difference between normalized cells and cancer cells. The higher is the viability, the more effective is the drug combination. 120 of the cells viability information are known, our job is to predict the rest 20 cells' viability. So the sample size of the datasets is 120, and the features dimension is 20.

2 Methodology

2.1 data preprocessing

This is a quite clean dataset, comparing to cleave dataset in mini-project1. We can easily form a feature matrix using pandas.

2.2 feature engeneering

The feature dimension of this dataset is not quite high. And they all have actual meanings, presenting dosage levels of 20 different drugs. So we didn't manually create other features using their combinations.

2.3 model selection

In this part, We did 5-fold cross validation on the training dataset for model selection. Our models includes LASSO, Ridge Regression, Support Vector Regression, Elastic Net and Gradient Boosting. We think the key point of this project is how to tune the parameters.

3 Results and Discussion

We first compare different regression method based on the Mean Square Error through 5-fold cross validation. The training errors are also included to check the overfitting. We then picked several models and trained them on the whole training dataset and submitted them on Kaggle. From the results in Table1, We found is that the difference between these regression models' validation error

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

model	validation error	training error
lasso($\alpha = 0.005$)	0.013677	0.008775
Elastic Net($\alpha = 0.01, ratio_{l_1} = 0.5$)	0.013680	0.008796
ridge($\alpha = 20$)	0.014146	0.009051
bayes ridge($\alpha_1 = 10, \alpha_2 = 0.010$)	0.014197	0.008665
svr	0.019456	0.006158
lars	0.021611	0.020714
kernel ridge($\alpha = 0.01$)	0.021955	0.014500
lasso Lars($\alpha = 0.1$)	0.022331	0.021887
boost	0.023385	0.000494

Table 1: Cross validation results measured by mean square error for each method

is not quite big. And based on the result we get in the kaggle testing, all team get training error lower than 0.014. We think this problem is a simple task.

4 Remark on Contributions

The project is finished under the discussion and close collaboration of our group members. Hao Xin wrote the code skeleton and wrote the majority of the report draft. Jinxing Yu tuned many regression methods and their parameters by cross validation and tried to get a better test result. Xun Jian proposed some feature selection method and implemented it, which got the best cross validation result.

Appendix

```
import pandas as pd
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import KFold
import numpy as np
from sklearn.feature_selection import *
# from sklearn.ensemble import ExtraTreesClassifier
# from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import mean_squared_error
from sklearn.kernel_ridge import KernelRidge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn import linear_model
from sklearn import svm, tree
# from sklearn.neural_network import MLPRegressor
predict_index = None


def test(X_train, X_test, y_train, y_test):
    '''run test test dataset'''
    names = ["lasso", "ridge", "Elastic Net",
             "lars", "lasso Lars", "bayes ridge", "kernel ridge", "boost", "svr"]
    classifiers = [linear_model.Lasso(alpha=0.005),
                   linear_model.Ridge(alpha=20),
                   linear_model.ElasticNet(alpha=0.01, l1_ratio=0.5),
                   linear_model.Lars(n_nonzero_coefs=1),
                   linear_model.LassoLars(alpha=0.1),
                   linear_model.BayesianRidge(alpha_1=10, alpha_2=0.001),
                   KernelRidge(alpha=0.01),
                   GradientBoostingRegressor(),
```

```
                    svm.SVR(),
                    # tree.DecisionTreeRegressor()
                    ]

    res = dict()
    # print '=================='
    for name, clf in zip(names, classifiers):
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        y_pred_t = clf.predict(X_train)
        # print name, 'test', mean_squared_error(y_test, y_pred)
        # print name, 'train', mean_squared_error(y_train, y_pred_t)
        # res[name] = mean_squared_error(y_test, y_pred)
        res[name + '_train'] = mean_squared_error(y_train, y_pred_t)
    return res


def read_data():
    data_frame = pd.read_csv('data/DrugEfficacy_train.csv', index_col=0)
    data_train = data_frame.loc[~np.isnan(data_frame['Viability'])]
    data_predict = data_frame.loc[np.isnan(data_frame['Viability'])]
    train_x = data_train.filter(regex='^D', axis=1)
    train_y = data_train.filter(regex='Viability', axis=1)
    predict_x = data_predict.filter(regex='^D', axis=1)
    global predict_index
    predict_index = data_predict.index.values
    return train_x.values, train_y.values.T[0], predict_x.values


def write_result(predict_y):
    result_data_frame = pd.DataFrame(
        predict_y, index=predict_index, columns=['Viability'])
    result_data_frame.index.name = 'ID'
    result_data_frame.to_csv('data/predict.csv', encoding='utf-8')


def model_select():
    result = []
    X, y, predict_x = read_data()
    for train_index, test_index in KFold(n_splits=5).split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        scaler = StandardScaler().fit(X_train)
        X_train = scaler.transform(X_train)
        X_test = scaler.transform(X_test)
        res = test(X_train, X_test, y_train, y_test)
        result.append(res)
    res_df = pd.DataFrame(result, index=range(len(result)))
    print res_df.mean().sort_values()


if __name__ == '__main__':
    # model_select()
    train_x, train_y, predict_x = read_data()
    print train_x.shape
    regressor = linear_model.Lasso(alpha=0.005)
    regressor.fit(train_x, train_y)
    predict_y = regressor.predict(predict_x)
    write_result(predict_y)
```