2020 Pattern Recognition and Machine Learning Technical Report

Ruian He, 16307110216,

I. TASK DESCRIPTION

DESIGN a regression system to predict housing prices. The data are available at kaggle[1] (also available at Sklearn). The regression algorithms should contain ridge regression, lasso regression, and decision tree for regression. The visualization can be done by "graphviz" for Python and is available at sklearn[2] .

II. DATA DESCRIPTION

Each record in the database describes a Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. [2] The attributes are defined as follows (taken from the UCI Machine Learning Repository):

1) CRIM: per capita crime rate by town
2) ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3) INDUS: proportion of non-retail business acres per town CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
4) NOX: nitric oxides concentration (parts per 10 million)
5) RM: average number of rooms per dwelling
6) AGE: proportion of owner-occupied units built prior to 1940
7) DIS: weighted distances to five Boston employment centers
8) RAD: index of accessibility to radial highways
9) TAX: full-value property-tax rate per $10,000
10) PTRATIO: pupil-teacher ratio by town
11) B: 1000(Bk-0.63)2 where Bk is the proportion of blacks by town
12) LSTAT: % lower status of the population
13) MEDV: Median value of owner-occupied homes in $1000s

We can see that the input attributes have a mixture of units.

III. DATA PREPROCESSING

We can load the database using `sklearn` module in python, and extract data as X and target as y. Then we perform zero-mean normalization on X to get rid of the effect from different units and scale.

$$x^* = \frac{x - \mu}{\sigma}$$

We also split the dataset into training set and test set and use k-fold cross-validation on the training set. Because of the linear regression need to get polynomial features, we stack the features and their powers horizontally to feed the linear model.

Listing 1. Generation of polynomial of X

```
X_train_poly = np.hstack([X_train,np.
    power(X_train,2),np.power(X_train,3),
    np.power(X_train,4),np.power(X_train
    ,5)])
X_test_poly  = np.hstack([X_test,np.power
    (X_test,2),np.power(X_test,3),np.power
    (X_test,4),np.power(X_test,5)])
```

IV. ALGORITHM INTRODUCTION

A. Algorithm principle

1) Linear Regression: In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. [1]

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D$$

where $\mathbf{x} = (x_1, ..., x_D)^T$. The key property of this model is that it is a linear function of the parameters $w_0, ..., w_D$. It is also, however, a linear function of the input variables xi, and this imposes significant limitations on the model.

2) Rigid and Lasso Regression: Ridge and Lasso regression are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients. Our final object is to minimize the following function, where n is the number of samples.

$$\min_w \frac{1}{2n} ||\mathbf{X}w - y||^2 + \alpha ||w||_2^2$$

For lasso regression, the regularization term is L1 norm of the coefficient vector. And there is also a hyperparameter alpha to control the regularization.

$$\min_w \frac{1}{2n} ||\mathbf{X}w - y||^2 + \alpha ||w||_1$$

[1] https://www.kaggle.com/vikrishnan/boston-house-prices
[2] http://sklearn.apachecn.org/cn/0.19.0/modules/tree.html#tree-algorithms

3) Decision Tree for Regression: Decision trees can also be applied to regression problems. Decision Tree can choose propriate features and thresholds to split the dataset to get maxnimum information gain which shows the effectiveness of the regression. Consequently, we travel through the decition tree to a leaf node, and we take the value of leaf node as the prediction.

B. Algorithm implementation

We can implement the above algorithms using `sklearn` library which have simple APIs to be called. All we should do is prepare the data and assign the hyperparameter.

Here we use the polynomial input for linear regression model and normal input for decision tree.

Listing 2. Call sklearn apis

```
# rigid regression
rid = linear_model.Ridge(alpha=best_alpha
    )
rid.fit(X_train_poly,y_train)
y_pred = rid.predict(X_test)

# lasso regression
las = linear_model.Lasso(alpha=best_alpha
    )
las.fit(X_train_poly,y_train)
y_pred = las.predict(X_test)

# decision tree
dt = tree.DecisionTreeRegressor(max_depth
    =best_depth)
dt = dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
```

When chossing the hyperparamaters, we use K-fold cross-validation.Each fold is then used once as a validation while the k - 1 remaining folds form the training set. Every possible value of the hyperparamater should pass all k folds and take avarage mean-squared error(mse) score to compare. Finally, we choose the one with the lowest mse score.

Take the rigid regression for example.

Listing 3. K-fold cross-validation

```
alpha_range = np.power(10.0,range
    (-10,10,1))
avg_error = []

kf = KFold(n_splits=n_splits)

for alpha in alpha_range:
  tot_error = 0
  for train_index, test_index in kf.split
      (X_train_poly):
    X_train_kf, X_test_kf = X_train_poly[
        train_index], X_train_poly[
        test_index]
    y_train_kf, y_test_kf = y_train[
        train_index], y_train[test_index]
    rid = linear_model.Ridge(alpha=alpha)
    rid.fit(X_train_kf,y_train_kf)
    y_pred_kf = rid.predict(X_test_kf)
    tot_error += mean_squared_error(
        y_pred_kf,y_test_kf)
  avg_error.append(tot_error/n_splits)

best_alpha = alpha_range[np.argmin(
    avg_error)]
```

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. K-fold Cross-validation

For the first two models, the range of alpha in Rigid Regression and Lasso Regression is from 10^{-10} to 10^{10} inceasing by multiplying 10, so we use log of the parameter to plot the figure. From the following figure, the both curves reached the lowest point at middle. Since the y axis is the mse error, the model achieve the best performance at the lowest point.

Fig. 1. K-fold of Rigid Regression

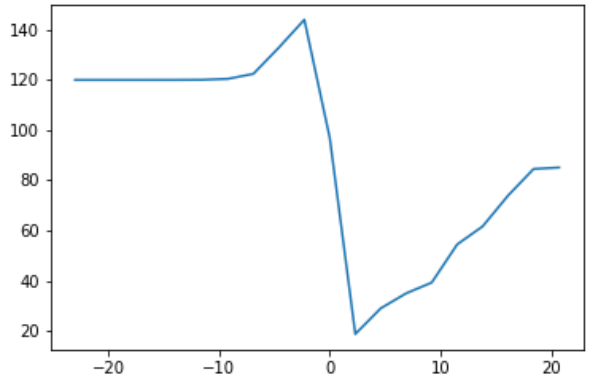
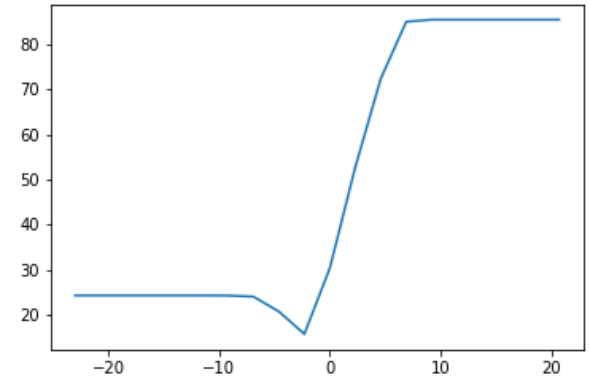


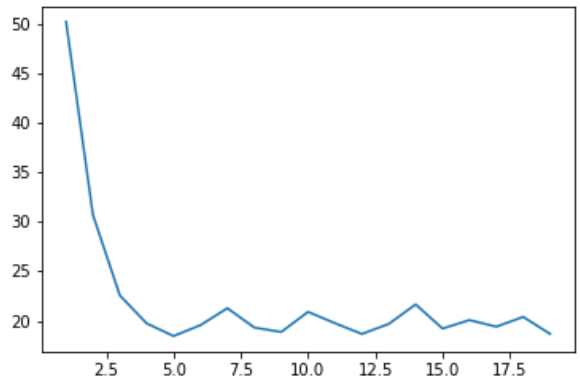
Fig. 2. K-fold of Lasso Regression



For the decision tree model, we choose the max depth as the hyperparameter to finetune because it can directly affect the complexity of the decition tree. To prevent the model from overfitting, we must carefully choose an appropriate max

depth. The curve is different from the above, it reached the botton at 5 and fluctuate after it. So we just choose 5 as the best one.

Fig. 3. K-fold of Decision Tree



B. Comparison of the three algorithms

From the table below, the three algorithm of regression seem to have the same performance on Boston House Price dataset. And we also can plot the Decision Tree with `graphviz` library. The figure is at the bottol and the color stands for the target value of the prediction.

TABLE I
MSE AND MAE COMPARISON

Model	Hyperparameter	Training MSE	Testing MSE	Testing MAE
Rigid Regression	10(alpha)	10.60	16.19	2.43
Lasso Regression	0.1(alpha)	11.81	16.29	2.30
Decision Tree	5(max_depth)	6.17	17.89	2.74

REFERENCES

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
[2] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. 1978.

Fig. 4. Visualization of Decision Tree