CS405 Machine Learning Lab

#4 Linear Regression

1. Introduction

As you have learnt linear regression in the lecture, now it is time to implement the machine learning techniques in practice. In this lab, you will use linear regression to fit a house price model. You will use some real-world data as the test set to evaluate your model. The scikit-learn package for Python provides many modules for easily developing machine learning algorithms.

2. Scikit learn package

Implementation of a linear regression by scratch is not difficult, but here we use the scikit-learn package for Python directly. This package contains many classical machine learning algorithms and is easy to use.

Datasets: scikit-learn provides a number of datasets which can be directly loaded by using a function. There are some small datasets called *toy datasets* and some large ones with thousands of samples called real world datasets. First we load a toy dataset as an example.

```
import warnings
from sklearn import datasets
boston = datasets.load_boston()
print(boston.DESCR)
```

See sklearn website for details. To do this you have to import right packages and modules. This is a small dataset containing 506 samples and 13 attributes. We need to use proper visualization methods to have an intuitive understanding. We choose the sixth attribute and draw a scattering plot to see the distribution of samples. We use *matplotlib* for data visualization.

```
# Use one feature for visualization
x = boston.data[:,5]

# Get the target vector
y = boston.target

# Scattering plot of prive vs. room number
from matplotlib import pyplot as plt
plt.scatter(x,y)
plt.show()
```

It can be seen that the samples have some exceptional distributions at the top of the plot. They may be outliers owing to some practical operation during the data input (e.g., convert any price larger than 50 into 50). However, these data are harmful to the model training, and should be removed.

```
x = x[y<50.0]
y = y[y<50.0]

plt.scatter(x,y)
plt.show()
```

Now it can be seen that the data is nearly linear, although just in one dimension. Now we use X to denote all attributes

```
X = boston.data
y = boston.target

X = X[y<50.0]
y = y[y<50.0]

X.shape
```

Split data

Now we divide the whole dataset into a training set and a test set using the the scikit-learn model_selection module.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y)
y_train.shape
```

Usually we also use a validation set. When we use the test set for evaluation, the model will not be changed after the evaluation. However, sometime we want to optimize our model by changing its parameters according to prediction results. The solution is to split a validation set from the training set for adjusting our model. When we believe that the model is good enough, then we evaluate our model on the test set. A more rigorous and costly way is cross validation. With that method, the training set is divided into several pieces in the same size and take every piece as a validation set in turn.

Linear Regression

Now we try to implement a simple linear regression model because the dataset seems linear.

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

The model has been trained just by using a few lines of codes. Now let's make a prediction for testing

```
# Make a prediction
y_0_hat = lin_reg.predict(X_test[0].reshape(1,-1))
y_0_hat
```

```
y_test[0]
```

Notice that in scikit-learn, the standard interface for machine learning is

1. instantiate a learner with super parameters or none;
2. use `fit()` method and feed the learner with training data;
3. use `predict()` for prediction.

Moreover, the data preprocessing algorithms also have the same interface, they just use `transform()` instead of `predict()`.

Below are the trained parameters.

```
lin_reg.coef_
```

```
lin_reg.intercept_
```

Use the evaluation method to see if it is a good model. The `score()` method uses R-square.

```
lin_reg.score(X_test, y_test)
```

3.Polynomial Regression

If you have understood the concept of linear regression, you can easily implement polynomial regression. Just a little bit more you should know:

1. Extend the attributes to polynomial attributes. we can achieve that easily by using scikit-learn. See PolynomialFeatures in module preprocessing.
2. When using polynomial features upon data, the values would be extremely large or small because of the power operation. That will influence the use of gradient descent which runs in background when we call `fit()`. So a normalization or standardization is necessary. See StandardScaler in preprocessing.
3. Pipeline can help us assemble several preprocessing functions and the learning process together. The `poly_reg` learner has the same interface as other learners.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures

poly_reg = Pipeline([
    ('poly', PolynomialFeatures(degree=2)),
    ('std_scaler', StandardScaler()),
    ('lin_reg', LinearRegression())
])
```

4. Regularization in scikit learn is `RidgeRegression`, which is in `linear_model`. Use it if you need

regularization in your model.

4. Lab Excercises

Please use the real world dataset, **California housing price**, for model training and evaluate the model's prediction performance. You can use simple linear regression, polynomial regression or more complicated base functions such as Gaussian function or use regularization methods. Make sure at least **20% data for testing** and choose one evaluation method you think good. **Please do not just train your model and say that is good enough, you need to analyze the bias and variance**. For that end, validation or cross validation is needed. Compare the score in the training set and the validation set. If they are both good enough, then use the model on the test set.

Your test set can only be used for final evaluation!

```
########## Write Your Code Here ##########

##########################################
```

5. Questions

1. Describe another real-world application where the regression method can be applied
2. What are the strengths of the linear/polynomial regression methods; when do they perform well?
3. What are the weaknesses of the linear/polynomial regression methods; when do they perform poorly?
4. What makes the linear regression method a good candidate for the regression problem, if you have enough knowledge about the data?

```

```