THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP4434 Big Data Analytics

Lab 4 Regularization Practice

HUANG Xiao
xiaohuang@comp.polyu.edu.hk

Types of Regularization Regression

- $||\theta||_2$: Ridge Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

- $||\theta||_1$: LASSO Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} |\theta_j|\right]$$

LASSO regression results in sparse solutions – vector with more zero coordinates. Good for high-dimensional problems – don't have to store all coordinates!

Supplement Material: Visual for Ridge Vs. LASSO Regression https://www.youtube.com/watch?v=Xm2C_gTAl8c

Boston Housing (has an ethical problem)

The Boston Housing Dataset consists of price of houses in various places in Boston. The Boston Housing Dataset has 506 cases. There are **13** Features in each case of the dataset. Alongside with price, the dataset also provide information such as Crime (CRIM), areas of non-retail business in the town (INDUS), the age of people who own the house (AGE), and there are many other attributes.

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTST	Price
0.006	18.0	2.31	0.0	0.538	6.575	65.2	4.090	1.0	296.0	15.3	396.9	4.98	24.0
0.027	0	7.07	0.0	0.469	6.421	78.9	4.967	2.0	242.0	17.8	396.9	9.14	21.6
…	…	…	…	…	…	…	…	…	…	…	…	…	…

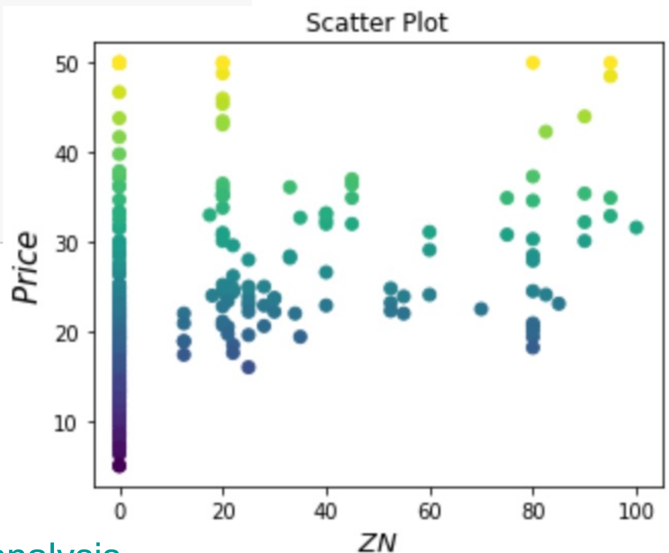
Generate Training Data

```
In [41]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.datasets import load_boston, load_diabetes
         from sklearn.model_selection import train_test_split

         np.random.seed(42)

         def load_data():
             dataset = load_boston()
             print(dataset.feature_names)
             return train_test_split(dataset.data, dataset.target, test_size=0.25, random_state=0)

         X_train, X_test, Y_train, Y_test = load_data()
         print(X_train.shape)

         plt.figure(figsize=(5,4))
         plt.scatter(X[:,1],y,c=y)
         plt.ylabel("$Price$", fontsize=15)
         plt.xlabel("$ZN$", rotation=0, fontsize=15)
         plt.title('Scatter Plot')
         plt.show()

         ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO
          'B' 'LSTAT']
         (379, 13)
```

Boston Housing Data

Split dataset

Plot figure



ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

https://www.kaggle.com/tolgahancepel/boston-housing-regression-analysis

Build Model

```
In [56]: from sklearn.linear_model import Ridge
         from sklearn.model_selection import cross_val_score

         alpha = 0
         model = Ridge(alpha=alpha, solver='auto', random_state=42)

         model.fit(X_train, Y_train)
         Y_pred = model.predict(X_test)
         cross_valid = cross_val_score(model, data, target, scoring='neg_mean_squared_error', cv = 5)
         print('Cross Validation Errors:\n', -np.mean(cross_valid))
         print('theta 0: \n', model.intercept_)
         print('theta 1-13: \n', model.coef_)
```

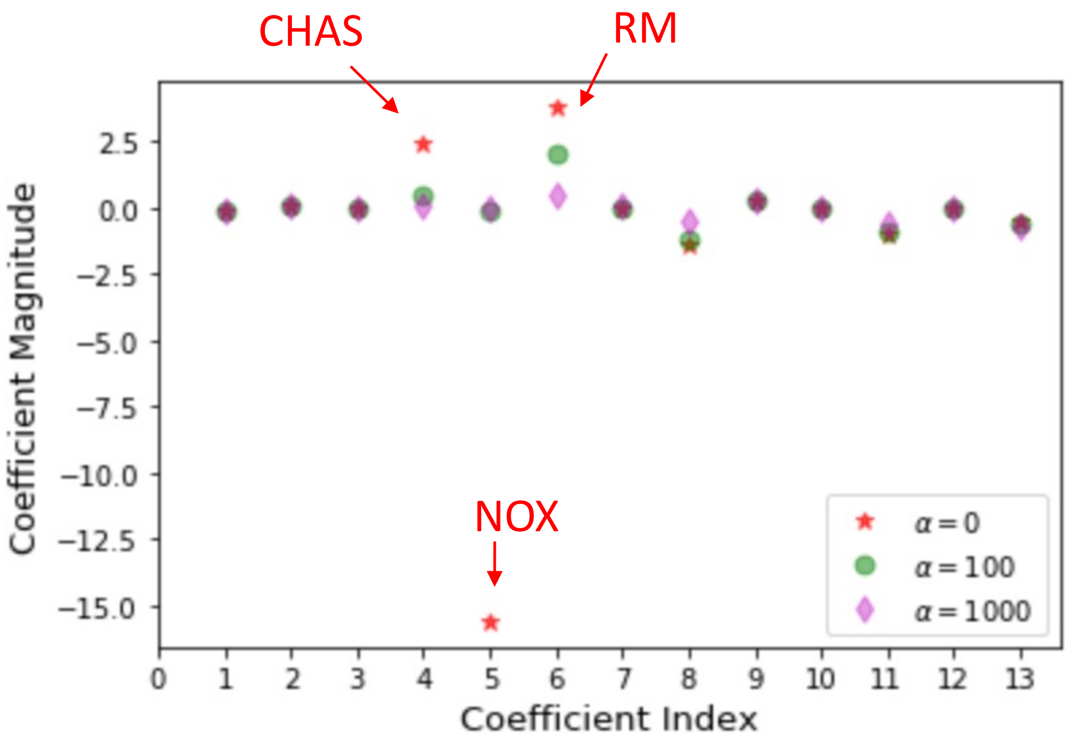
Train model

Cross validation

```
Cross Validation Errors:
 37.13180746769889
theta 0:
 36.933255457119316
theta 1-13:
 [-1.17735289e-01  4.40174969e-02 -5.76814314e-03  2.39341594e+00
 -1.55894211e+01  3.76896770e+00 -7.03517828e-03 -1.43495641e+00
  2.40081086e-01 -1.12972810e-02 -9.85546732e-01  8.44443453e-03
 -4.99116797e-01]
```

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_{13} x_{13}$$

Regularization



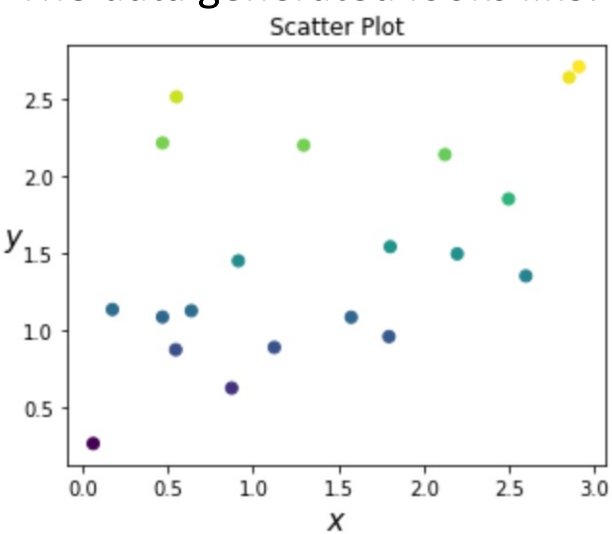
α	MSE
0	37.1318 (overfitting)
100	29.9057
1000	32.8280 (underfitting)

The magnitudes of coefficient indices 4,5,6 are considerably reduced after regularization with α = 100, resulting in lower mean square error

Generate Random Dataset

```
In [317]: import numpy as np
          from sklearn.preprocessing import PolynomialFeatures

          np.random.seed(42)
          m = 20
          x = 3 * np.random.rand(m, 1)
          y = 1 + 0.5 * x + np.random.randn(m, 1) / 1.5
          x_test = np.linspace(0, 3, 100).reshape(100, 1)

          x_poly = PolynomialFeatures(degree=10, include_bias = True)
          x_poly.fit_transform(x)
          print(x_poly.get_feature_names())

          ['1', 'x0', 'x0^2', 'x0^3', 'x0^4', 'x0^5', 'x0^6', 'x0^7', 'x0^8', 'x0^9', 'x0^10']
```

The data generated looks like:



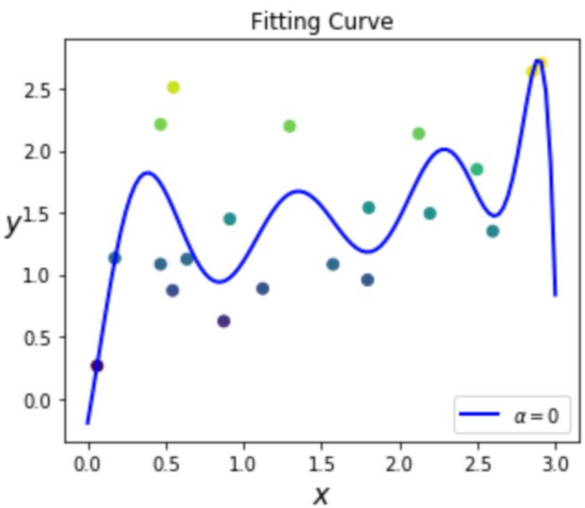
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_{10} x^{10}$$

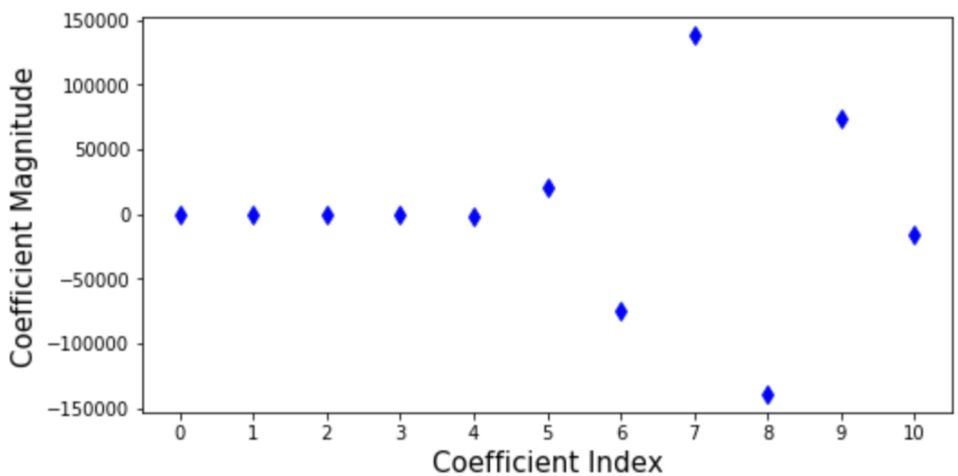
Build Model (Without Regularization)

```
In [298]: from sklearn.linear_model import Ridge
          from sklearn.pipeline import Pipeline
          from sklearn.preprocessing import PolynomialFeatures, StandardScaler
          from sklearn.model_selection import cross_val_score

          alpha = 0

          model = Ridge(alpha=alpha, solver = 'auto', random_state =42)
          model = Pipeline([
                  ("poly_features", PolynomialFeatures(degree=10, include_bias=True)),
                  ("std_scaler", StandardScaler()),
                  ("regul_reg", model),
              ])
          model.fit(x, y)
          y_pred = model.predict(x_test)
          cross_valid = cross_val_score(model, x, y, scoring='neg_mean_squared_error', cv = 5)
          print('Cross Validation Errors:', -np.mean(cross_valid))
          print('Theta:', model.named_steps["regul_reg"].coef_)

          Cross Validation Errors: 5.209334418802447
          Theta: [[ 0.00000000e+00  6.43580702e+00  2.57281708e+01 -2.75008050e+02
            -1.90585038e+03  2.11658121e+04 -7.55141493e+04  1.37974411e+05
            -1.39291200e+05  7.39888222e+04 -1.61744622e+04]]
```

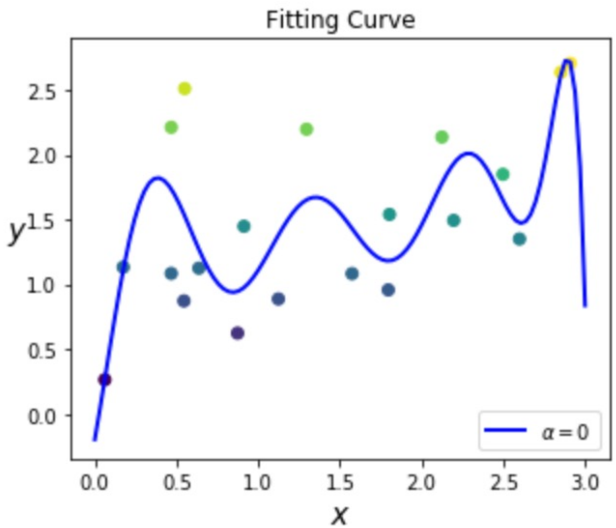
overfitting:

- Cross validation Error is large

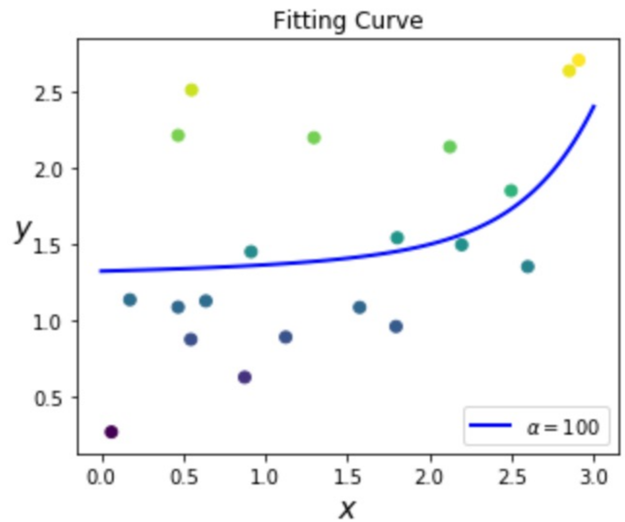




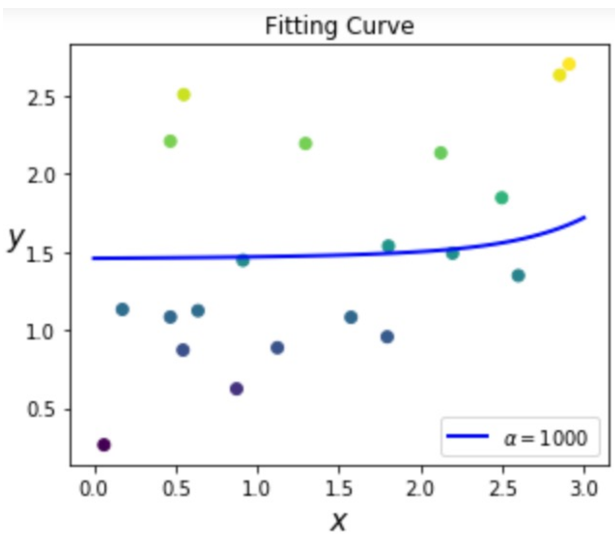
Train Model with Different Alphas



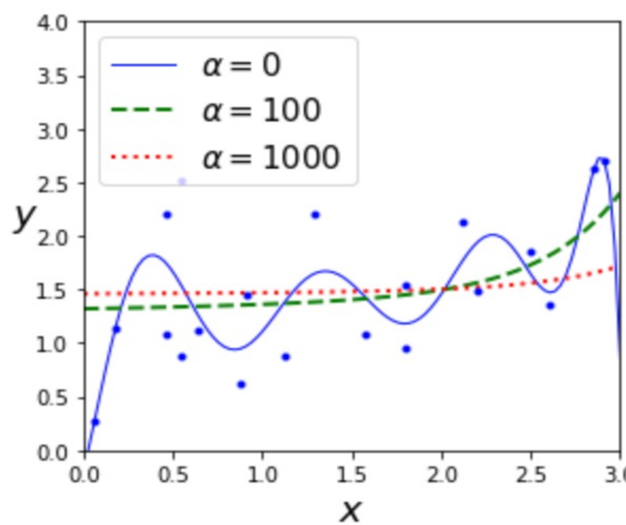
overfitting



correct fitting



underfitting



Further Practice

Further tasks:

- Implement the L1 regularization
- Plot the parameter figures, and modify alpha to check the difference

Further readings:

- https://harish-reddy.medium.com/regularization-in-python-699cfbad8622
- https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20a%20resampling,k%2Dfold%20cross%2Dvalidation
- https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b
- https://www.kaggle.com/apapiu/regularized-linear-models