CSE353 Assignment6 Report

Chengzhi Dong
ID 112890166

Due December 9, 2021

1 Introduction

For Assignment 6, I will experiment with nonlinear transform, overfitting / underfitting, regularization, and cross validation. For part 1, I will choose different Q-values to perform Q^{th} order polynomial transform on the training dataset, then estimate w_{poly} by applying linear regression on the transformed dataset and calculate the training error. I will test the w_{poly} by apply it on the transformed testing dataset and calculate the test. Also, I will plot the the training & testing sample points, and the estimated curve. By comparing the results of different Q-values, I can determine which one is overfitting and which one is underfitting. For part 2, I will apply Q^{th} order polynomial transform on the training dataset and estimate $w_{regpoly}$ by apply regularized linear regression to the transformed training dataset with $Q = 6$ and $\lambda = 0.1$, then test $w_{regpoly}$ with the testing dataset. I will calculate the squared error costs on the training and testing dataset, and plot the training & testing sample points and the estimated curve. I will repeat part 2 with $Q = 6$ and $\lambda = 100$ and compare the results. For part 3, I will apply the leave-one-out and v-fold cross validation strategies to select the best λ from the pool of $\{0.01, 0.1, 1, 10, 100, 1000, 10^6\}$.

2 Method

My algorithm implemented the polynomial transform, linear regression, regularized linear regression, and leave-one-out and v-fold cross validation strategies. The Python code imported the numpy, scipy.linalg, and matplotlib.pyplot libraries.

I defined a function called Poly_Transform to perform the Q^{th} order polynomial transform on the dataset. It is a recursive function that takes Q-value and dataset as parameters, and return the transformed dataset $Z = [z_1^T; z_2^T; z_3^T; ...; z_N^T]$ s.t. $z_n = \phi(x_n) = [1, x_n, x_n^2, x_n^3, ..., x_n^Q]^T$.

I leveraged my previous assignment code of Linear Regression, I defined a function called LinearReg_Algorithm that takes x dataset and y dateset, and returns the closed form solution $w^* = (X^TX)^{-1}X^TY$. I defined a function called Regularization for the Regularized Linear Regression which takes λ, z dataset and y dataset, and return the closed form solution $w^* = (Z^TZ + \lambda I)^{-1}Z^TY$.

I also defined the Sqr_Err function to calculate the square error: $err_{sqr} = \sum_{n=1}^{N}(w^T\phi(x_n) - y_n)^2$

For the cross validation, I defined a function called LOO which perform the Left-One-Out Cross Validation with a Q-value, λ, x dataset, and y dataset. The function each time took one sample out for validation and use the rest for training. It calculated the square error for that one sample and add the error the total error with every sample as the validation set. Then it return the average error. I defined a function called VFold which perform the V-fold Cross Validation with a Q-value, λ, V-value, x dataset and y dataset. The VFold function separate the dataset into V same size subset. Then each time it used one of the subset for validation and the rest for training. It sums up the total square err for validation using each subset. Then return the average error by divide the total error by V, the number of subsets.

Last, I defined a function calls Visualization, which plot the training and testing data points and the estimated curve w.

After I defined all the functions I needed, I first load the training and testing data to arrays called x_training, y_training, x_testing, and y_testing.

For part 1, I created a Q-value pool of 2, 3, 10, 15, 18 because I want to try different Q-values. For each Q-value, I call the Poly_Transform function to transform x_training dataset into z_training dataset, then I estimated the w_{poly} by calling the LinearReg_Algorithm with z_training and y_training. I calculated the square error for training. Then apply w_{poly} to testing dataset by transform the x_testing and calculate the square error for testing. I called the Visualization function to plot the training & testing data points, and the estimated curve $g(x) = w_{poly}^T\phi(x)$.

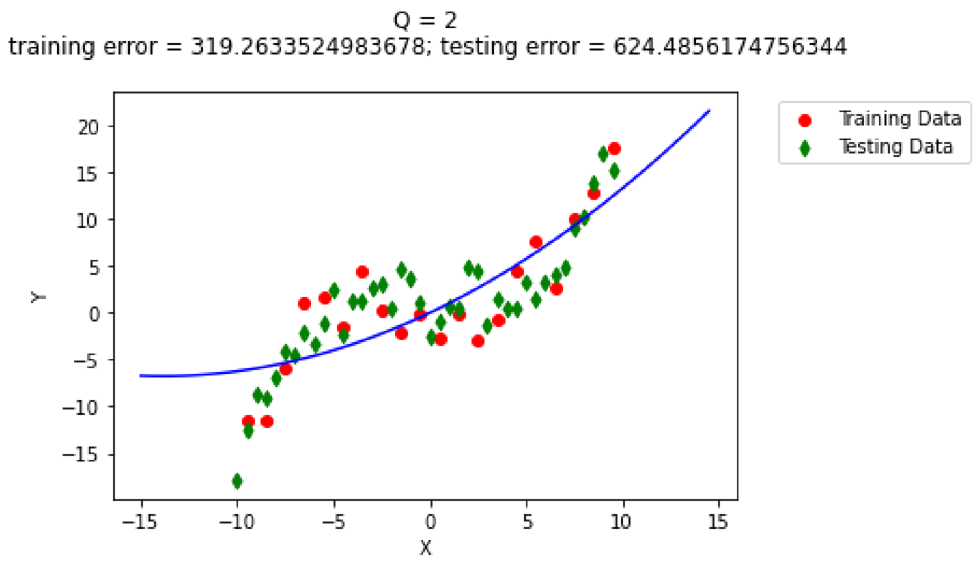
For part 2, I set $Q = 6$ and $\lambda = 0.1$. I call the Poly_Transform function with the Q-value to transform x_training into z_training; then then I estimated the $w_{regpoly}$ by calling the Regularization function with λ, z_training and y_training. Then I calculated the training error, transform the testing data set, and calculate the testing error. I plot the training & testing data points, and the estimated curve $g(x) = w_{regpoly}^T\phi(x)$. I repeated the above steps to for $Q = 6$ and $\lambda = 100$ and compared the results with different λ.

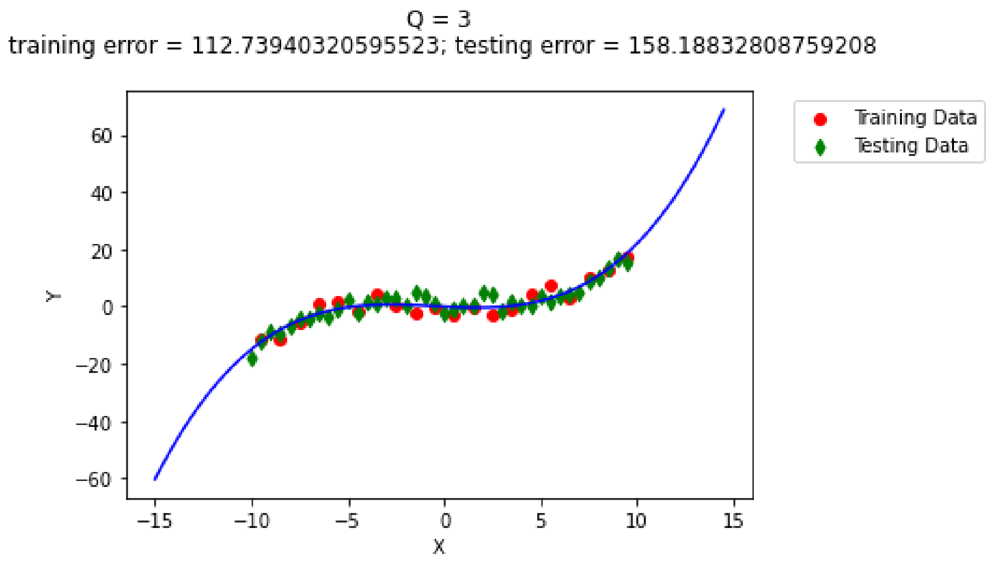
For part 3, I loop through the λ pool of 0.01, 0.1, 1, 10, 100, 1000, 1000000, and for each λ I used the Left-One-Out and V-fold Cross Validation strategies to select the best λ with the least error. I experimented the cross validation with

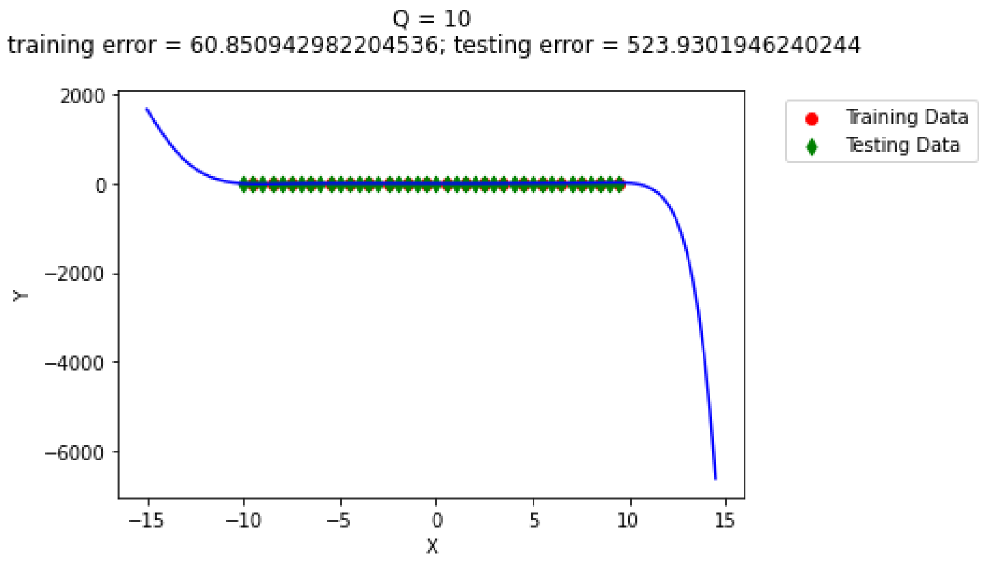
Q-value = 6 and 10.

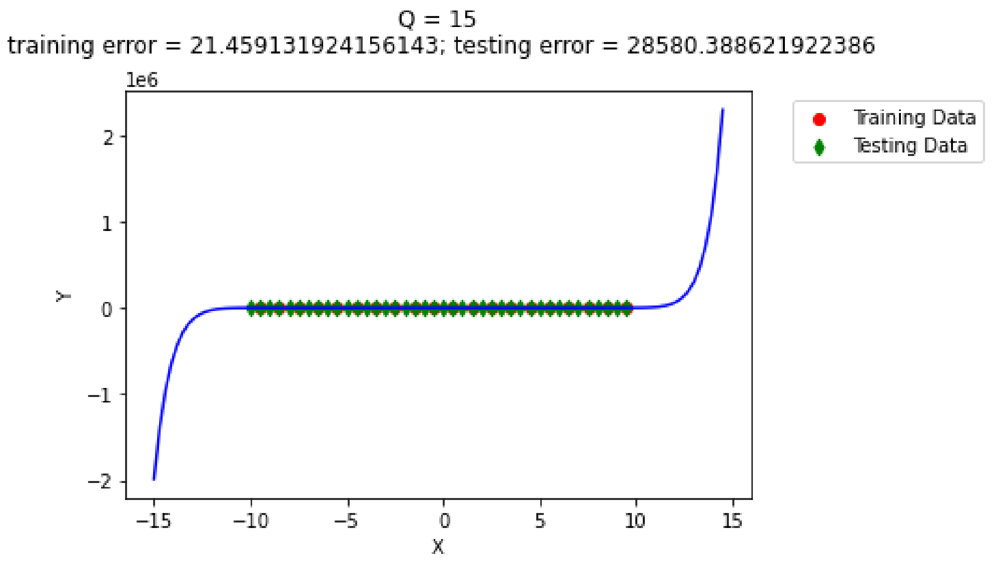
3 Experiments

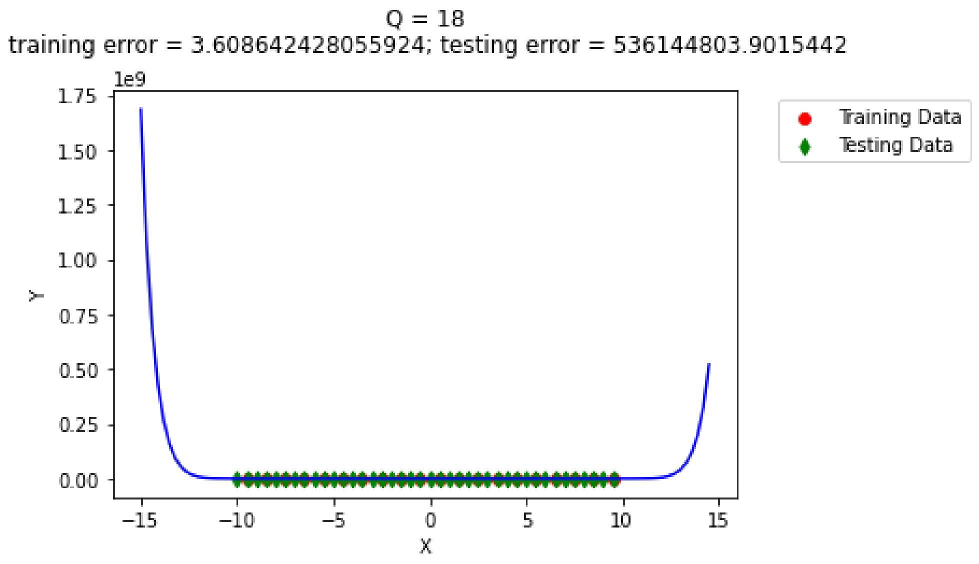
Below are the results for Part 1:





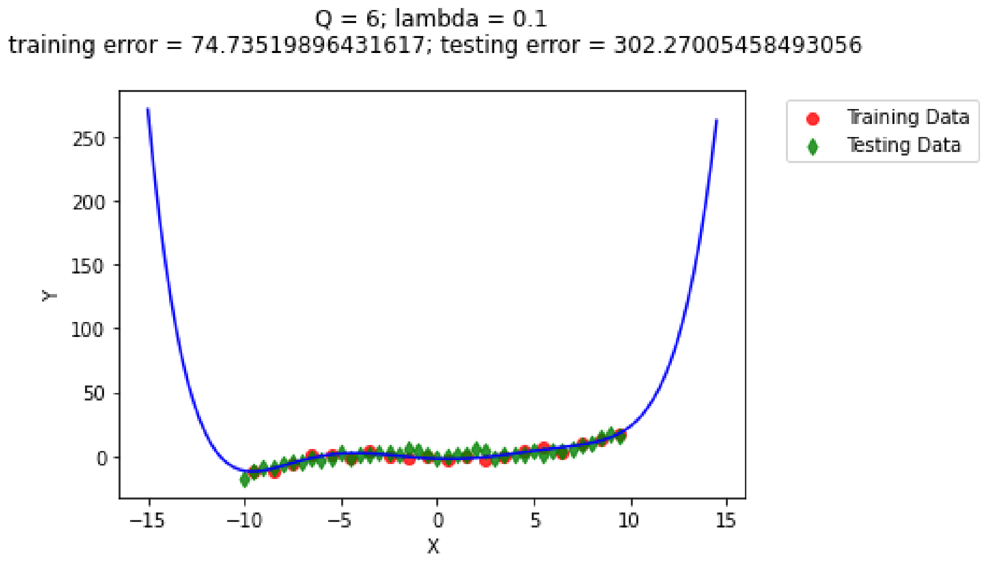

Q = 10
training error = 60.850942982204536; testing error = 523.9301946240244
2000
0
−2000
−4000
−6000
Y
−15
−10
−5
0
5
10
15
X
● Training Data
⧫ Testing Data

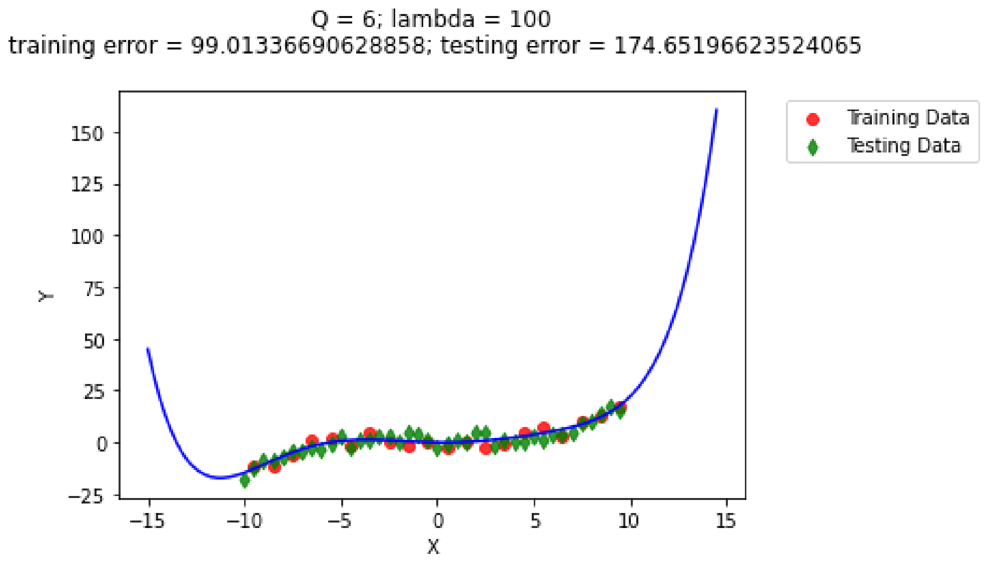

Q = 15
training error = 21.459131924156143; testing error = 28580.388621922386
1e6
2
1
0
−1
−2
Y
−15
−10
−5
0
5
10
15
X
● Training Data
⧫ Testing Data



As the results shown:
When $Q = 2$, the training error is 319.26 and the testing error is 624.49.
When $Q = 3$, the training error is 112.74 and the testing error is 158.19.
When $Q = 10$, the training error is 60.85 and the testing error is 523.93.
When $Q = 15$, the training error is 21.46 and the testing error is 28580.39.
When $Q = 18$, the training error is 3.61 and the testing error is 536144803.90.
According to the trends, as the Q-value increases, the training error decreases but the testing error increases. As the Q-value increases, the estimated curve changes from underfitting to overfitting. When $Q = 2$, it is considered as underfitting because both the training error and testing error are large. I think when $Q = 3$, the estimated curve fits the data just right when the training error and testing error are kind of balance and relatively low. On the other hands, when $Q = 10, 15, 18$, the estimated curves are considered as overfitting where the training error is low but the testing error is very large.

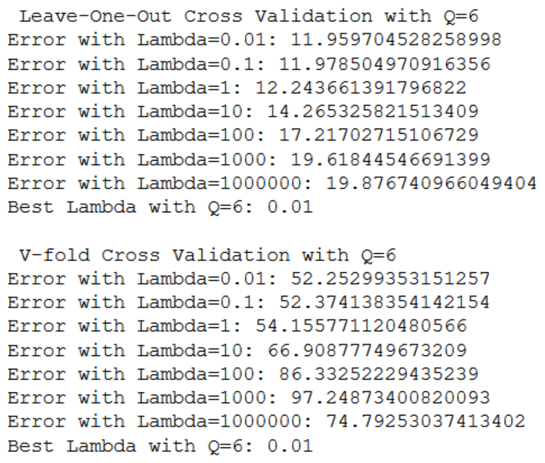
Below are the results for Part 2:





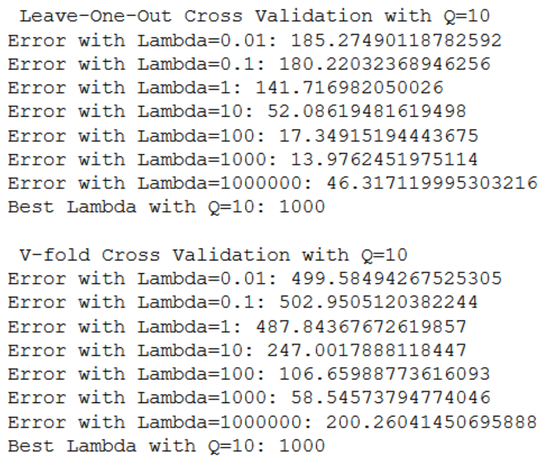
As the results shown:
When $Q = 6$ and $\lambda = 0.1$, the training error is 74.74 and the testing error is 302.27.
When $Q = 6$ and $\lambda = 100$, the training error is 99.01 and the testing error is 174.65.
After the λ changed from 0.1 to 100, the training error of the estimated curve increased by a bit (24.27) but the testing error decreased by a lot (126.63). These results show that in this case when $Q = 6$, $\lambda = 100$ is better than $\lambda = 0.1$.

Below are the results for Part 3:



```
 Leave-One-Out Cross Validation with Q=6
Error with Lambda=0.01: 11.959704528258998
Error with Lambda=0.1: 11.978504970916356
Error with Lambda=1: 12.243661391796822
Error with Lambda=10: 14.265325821513409
Error with Lambda=100: 17.21702715106729
Error with Lambda=1000: 19.61844546691399
Error with Lambda=1000000: 19.876740966049404
Best Lambda with Q=6: 0.01

 V-fold Cross Validation with Q=6
Error with Lambda=0.01: 52.25299353151257
Error with Lambda=0.1: 52.374138354142154
Error with Lambda=1: 54.155771120480566
Error with Lambda=10: 66.90877749673209
Error with Lambda=100: 86.33252229435239
Error with Lambda=1000: 97.24873400820093
Error with Lambda=1000000: 74.79253037413402
Best Lambda with Q=6: 0.01
```



```
 Leave-One-Out Cross Validation with Q=10
Error with Lambda=0.01: 185.27490118782592
Error with Lambda=0.1: 180.22032368946256
Error with Lambda=1: 141.716982050026
Error with Lambda=10: 52.08619481619498
Error with Lambda=100: 17.34915194443675
Error with Lambda=1000: 13.9762451975114
Error with Lambda=1000000: 46.317119995303216
Best Lambda with Q=10: 1000

 V-fold Cross Validation with Q=10
Error with Lambda=0.01: 499.58494267525305
Error with Lambda=0.1: 502.9505120382244
Error with Lambda=1: 487.84367672619857
Error with Lambda=10: 247.0017888118447
Error with Lambda=100: 106.65988773616093
Error with Lambda=1000: 58.54573794774046
Error with Lambda=1000000: 200.26041450695888
Best Lambda with Q=10: 1000
```

I first chose $Q = 6$ to perform the Cross Validations. Both Leave-One-Out and V-Fold Cross Validation with $Q = 6$ shows the best λ is 0.01. Based on the validation error, the error of $\lambda = 0.1$ is smaller that the error of $\lambda = 100$. However, this result did not match the reality where it as shown in Part 2, $\lambda = 100$ is better that $\lambda = 0.1$. This might be caused by the size of training data set is not large enough, and it shows some limitations of the Cross Validation. I also perform the Cross Validation with $Q = 10$. Both Leave-One-Out and V-Fold Cross Validation with $Q = 10$ shows the best λ is 1000, which I think is reasonable because based on Part 1, $Q = 10$ produce overfitting problem, We can solve it by regularization with a large λ.

4 Discussion

Originally, I was going to experiment with $Q = 19$ in part 1 because of the assignment documentation, and base on the professor's example the training error of the curve that estimated from $Q = 19$ should approximately zero with a super large testing error. However, when I try to set the Q-value = 19 or any number larger, I obtained some strange results where both the training error and testing error are super large. Later I found out that there is something wrong with numpy's linalg.inv() function, which calculates a wrong inverse matrix when the Q-value is greater than 18. Everything works fine when $Q \leq 18$; therefore, I choose to experiment with Q-values of 2, 3, 10, 15, 18 in part 1.

For regularization and cross validation, I found out that we need a larger λ-value to regularized with a larger Q^{th} order polynomial transform. However, the cross validation has its limitation and would not always give the real best λ. But it is better than nothing since in reality we can only access the training data and would not know what the testing data is.