CMSC498D Homework 1 Report

William Lin

February 11th 2025

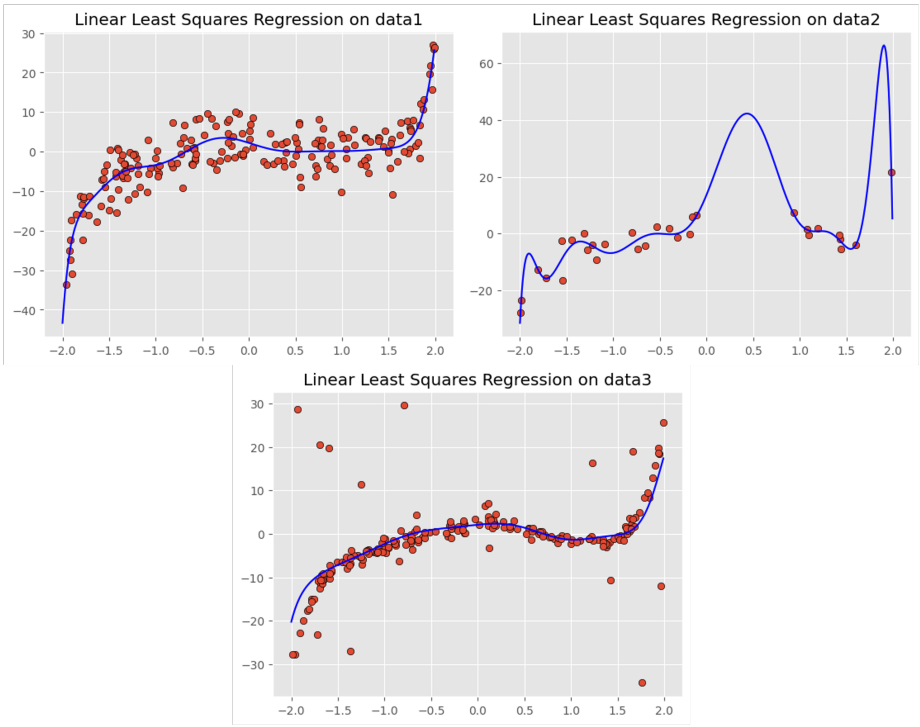
1 Introduction

This report will outline how I used linear least squares regression and ridge regression to fit a 12th order polynomial to the three provided datasets. Besides refreshing myself on common Python libraries like numpy and matplotlib which I have used before, I took a couple hours to learn the main ideas of linear least squares and ridge regression which were previewed in class. After I finished preparing, it became fairly straightforward what I had to do without much trouble.

2 Linear Least Squares Regression

At first, I started out by playing around with just dataset1. I tried to calculate the lifted data matrix using the preprovided function which I realized was just the familiar Vandermonde matrix taught in Linear Algebra class. I separated the data into its x and y parts which is done using some Python magic and then ran lift(x) in which the output was used to calculate theta. The formula for theta was given in class and is fairly easy to calculate using numpy function. I experimented around a little to get this all to work after having trouble with the shape of data and sometimes referencing the wrong x (x part of the data vs. X matrix which was the lifted matrix). Eventually, I was able to calculate $\hat{y}$ given the formula shown in class. The requirements also specifically mentioned using `x_test=np.arange(-2,2,.01)`, so I used this to show the fitted polynomial.

After struggling a little bit with understanding how to plot the data points and the fitted polynomial, I was able to get everything to work. The plots shown below are the results:

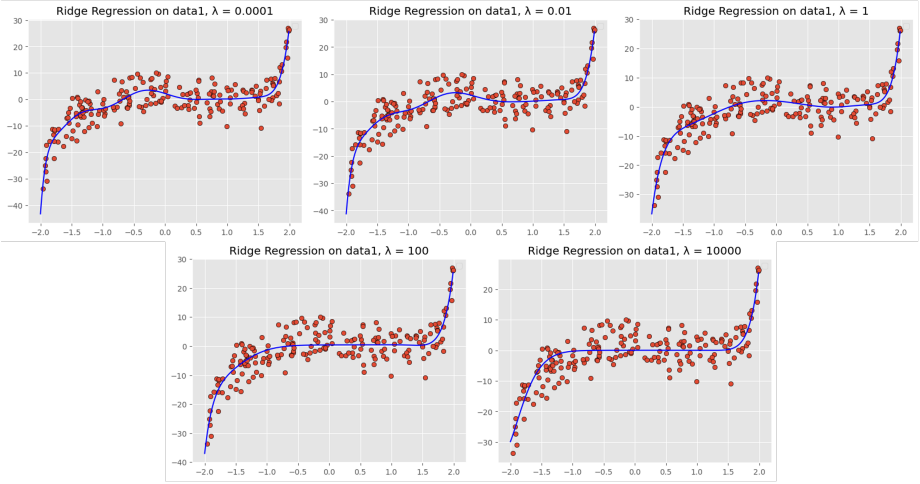


We see that the polynomial fits the data pretty well for datasets 1 and 3. However, the polynomial for dataset 2 exhibits some evidence of overfitting since linear tries to fit to the noises in the datasets especially the ones towards the end of the plot. Additionally, there seems to be a weird hump around $x = 0.5$. These irregularities maybe due to the polynomial focusing too much on the training data and so it performs weirdly (and less accurately) on test data hence we call it overfitting. Next, we will be see that we can reduce this overfitting through the use of regularization which is used in the methods of ridge regression. We will see that the polynomials that already have a good fit won't change much as a result of the regularization (in the case of datasets 1 and 3) while the regularization will work to correct polynomial fits that are overfitting (in the case of dataset 2).

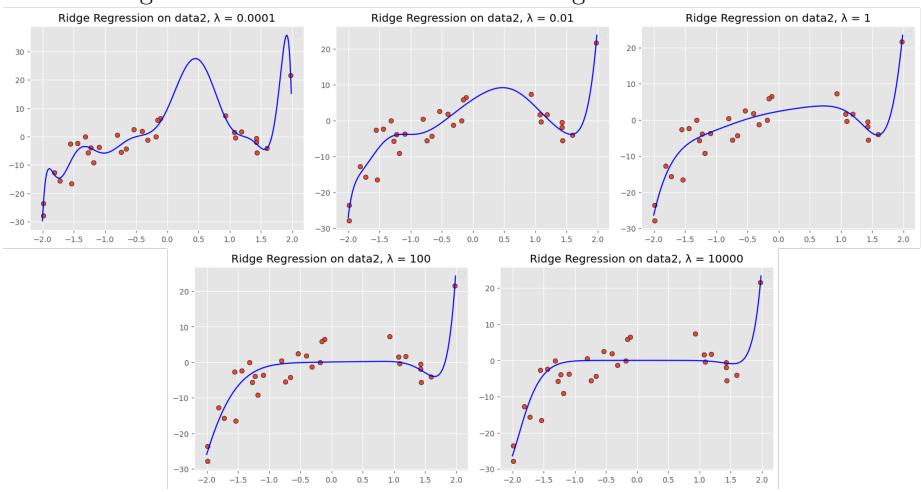
3 Ridge Regression

After struggling through and implementing the linear least squares regression, it was fairly easy to add ridge regression. Everything is the same except for an extra term when calculating theta ($(X^TX)^{-1}X^Ty$ vs. $(X^TX + \lambda I)^{-1}X^Ty$). The λ is a hyperparameter that will be varied.

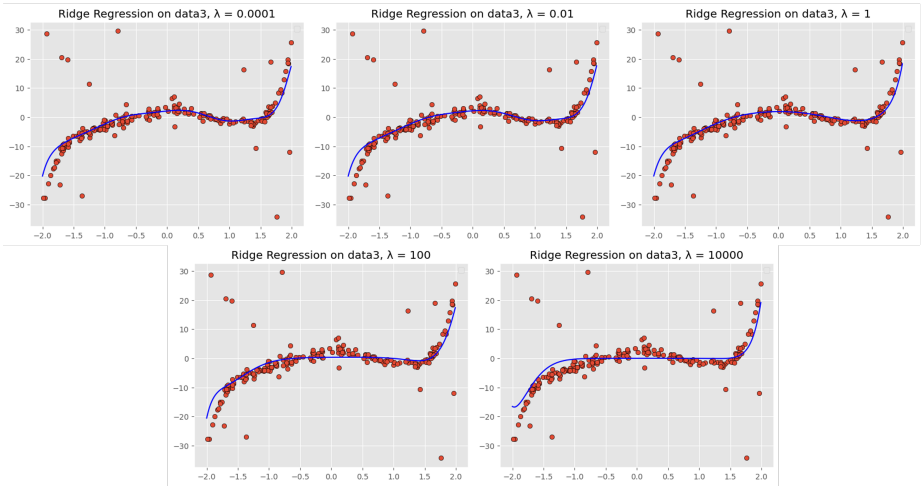
Here are some results:



As stated above, the polynomial fit for dataset 1 with the linear least squares regression already did a good job and didn't overfit the dataset. Hence, regularization didn't necessarily affect much. With higher lambda values, the model get a little too general which could lead to overfitting.



With dataset 2, we previously observed that the polynomial was overfitting and thus the regularization did a good job in correcting this. With higher lambda, the weird bumps that were due to the polynomial fitting to noise disappears.

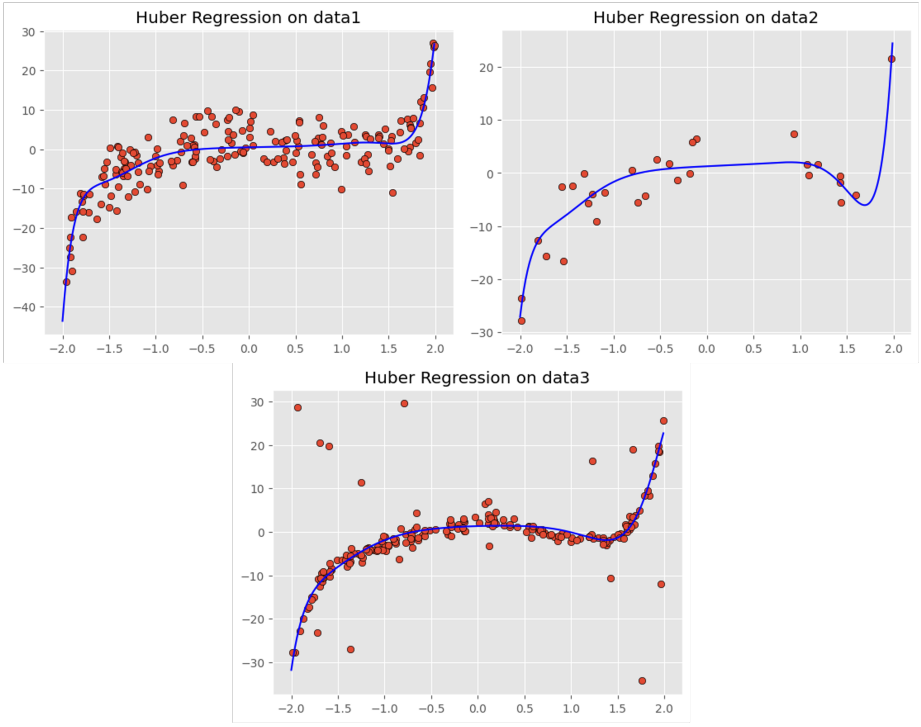


Dataset 3 was similar to dataset 1 in that the linear least squares regression already did a good job and thus regularization isn't really required.

We learn a lot from performing ridge regressions on the three datasets. The main limitation that was evident and taught in class was the introduction of bias. In all three datasets, setting a lambda value that was too high is seen to overgeneralize the polynomial fit and thus would lead to underfitting. The underfitting would lead to bad performance when predicting both training and testing values. This also plays into another problem which is the tuning of the lambda hyperparameter which doesn't always help our models. Not only can tuning the be hyperparameter expensive, but might not always be necessary as seen with datasets 1 and 3.

4 Robust Regression

Using sklearn's HuberRegressor, I was able to get the following polynomial fits:



The main motivations for using this Huber robust regression is due to the fact that linear least squares regression is prone to outliers. This is because the optimization problem that we tried to solve with linear least squares is to minimize the mean squared loss. When it comes to mean squared loss, outliers have high influence on the model as the error is squared. Huber robust regression tries to combat this issue by introducing a Huber loss which acts in a more linear fashion when it comes to loss. On top of that, a hyperparameter (epsilon) is used to further tune the weight of outliers when fitting the polynomial. Thus, the regression seen above does a very nice job at trying to fit the polynomial to the dataset without overtunning the polynomial to noise and thus reduces overfitting.