Report

Part 1 - Geometric interpretation of eigenvalues/covariance matrix

For this part, following the instructions, I found the covariance matrix by calculating the expectation of x and y, then plug those into formula: $\sigma(x, y) = \mathbb{E}[(x - \mathbb{E}(x))(y - \mathbb{E}(y))]$

After getting the covariance matrix looking like this:

$$\Sigma = \begin{bmatrix} \sigma(x, x) & \sigma(x, y) \\ \sigma(y, x) & \sigma(y, y) \end{bmatrix}$$

I used NumPy to find the eigenvalues and their corresponding eigenvectors. Eigenvectors will be indicating the directions of the data lies in, and their eigenvalue will basically indicate the span of the data in that particular direction. Since we have all 4 datasets being 2 dimensional, we will have two eigenvalues and those two directions will be spanning in the 2-d space. For n dimensional data more generally speaking, the covariance matrix will be n by n, and there will be n eigenvectors spanning in n directions. This is similar to what we will be doing in PCA. That's basically my understanding of eigenvectors and eigenvalues.

One problem I had is that I forget to center the data that I had, which makes data 4's graph looks off by a margin. After centering the data, the graph seems to be working correctly.

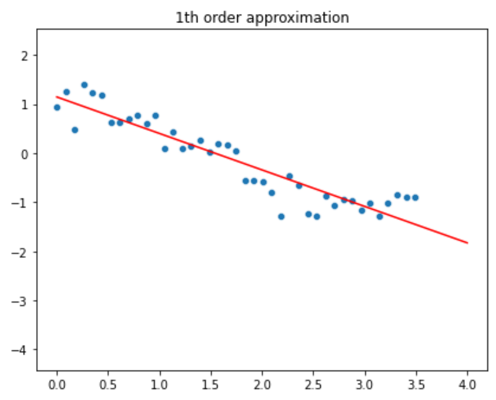
Part 2 - Least squares line fitting without regularization

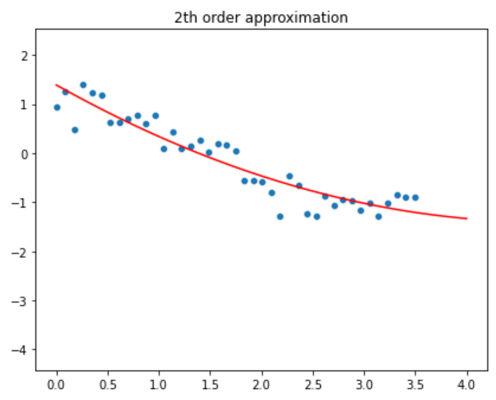
For this part, I wanted to see the effect of what order of approximation will have on my line fitting, so I made a list of order containing 1, 2, 4, 8, 12 and 16. Then I made a for loop iterating through the list. For each order, I first transformed the x from dataset to matrix with column number equals to order + 1, each column corresponding to the coefficient of an order. For example, say we have 4th order approximation, then a 5-column matrix will be generated, first is the coefficient of x^0, second is x^1, then x^2, x^3, x^4. Then feed the new X generated into this formula: $\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

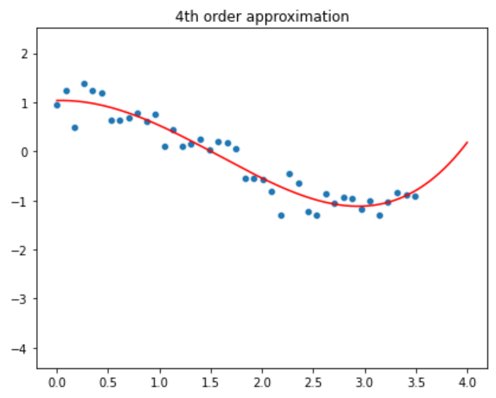
Beta will be a 1 by n vector, corresponding to the coefficient of each order again. (n = order + 1).

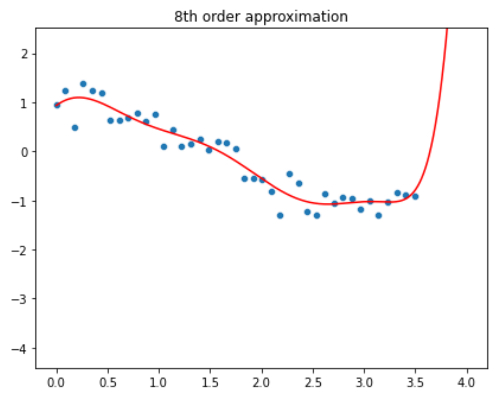
From looking at the graph I'd say most line fitting looks alright with 1st order approximation, after going up to 8th or even 16th, the approximation is really overfitted. To avoid overfitting, I calculated the loss.

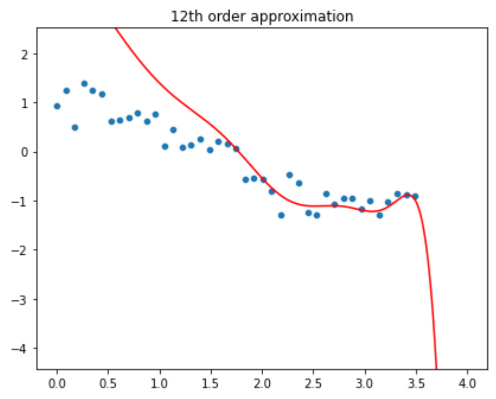
From this graph showing the least square approximation without regularization on data4.csv, it's not hard to see that starting on order 12, the approximation isn't doing a good job and on order 16, the prediction is not even close to what those points actually are.

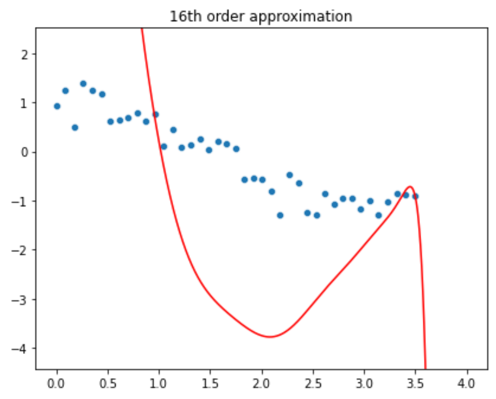












Part 3 - Least squares line fitting with regularization

Continuing from previous part, I kept basically the same layout with the addition of an array called loss and another called lam_range. X is transformed the same way as it was in part 2.

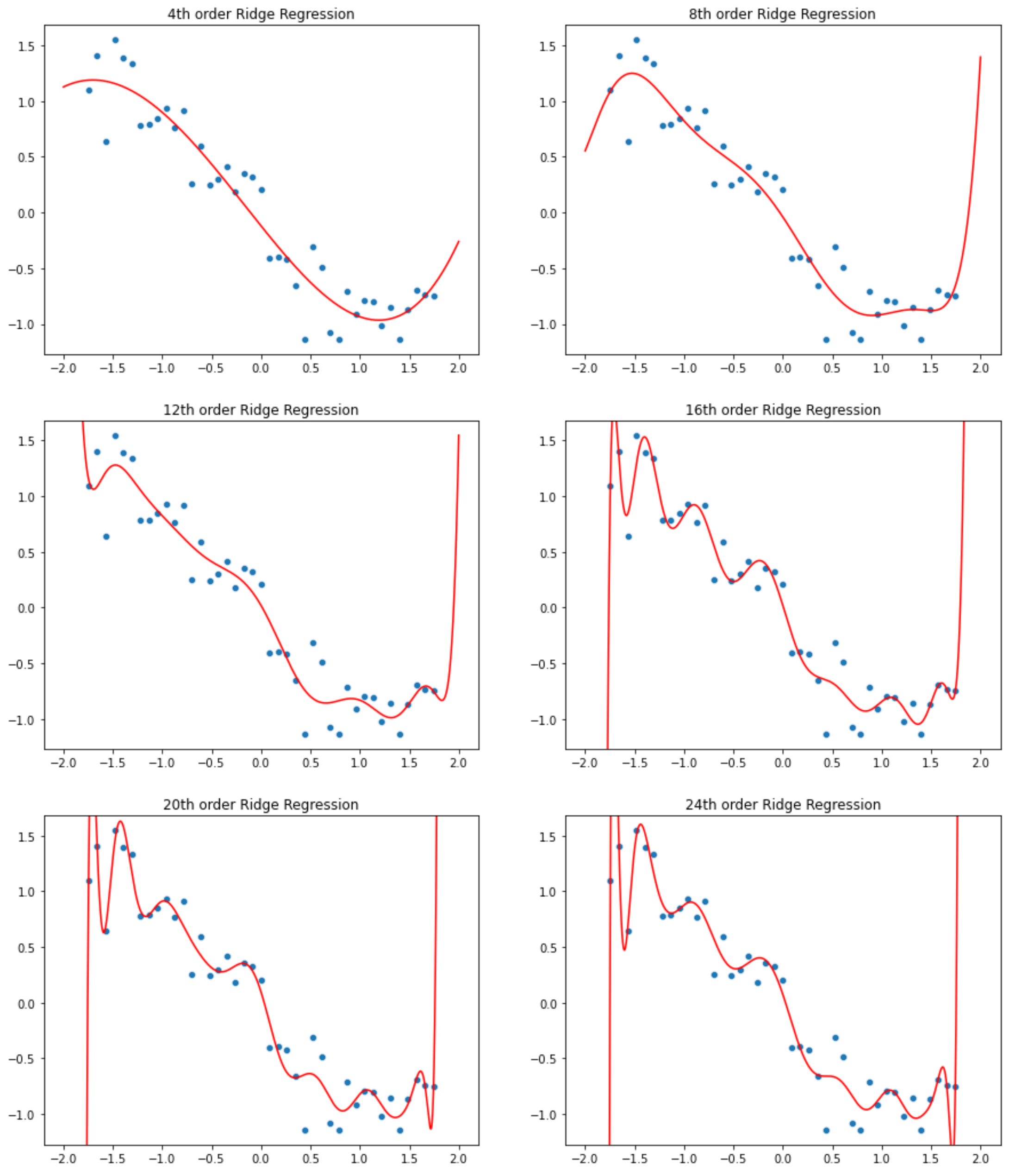
I set the lam_range to be ranging from 10^-25 to 10^5. Then I used a for loop to plug in different lambda into ridge regression's closed form solution to get corresponding theta:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

After getting the beta, I find the loss with the formula below and then save it into the loss array.

$$\min_{\theta_1,\theta_2,\ldots,\theta_m} \frac{1}{2n}\left[\left(X\theta - y\right)^T\left(X\theta - y\right) + \lambda\theta^T\theta\right]$$

The min of loss array after the for loop will be set to the most optimal lambda and then gets outputted. It’s pretty obvious that this time, with ridge regression, the lines fits much better to the points. The lambda was found to work best at around 10^-25 for models till order 16, for order 20, lambda 10^-14 works best, and for order 24, lambda 10^-6 works best. It’s worth noting that the model is quite over-fit.



I’d say that the ridge regression doesn’t make huge difference when we have low order for approximation, and the effect of lambda will be increasingly significant as the order of approximation increase. I think another the limitation of ridge regression is that the lambda value has huge influence on performance, and there’s not always a minimum to be found, it creates bias in exchange for lower variance, which is a move that won’t always be helpful.