Report

Nikshay Jain - MM21B044

September 2024

1 Data Visualisation

The given train and test dataset had 3 columns, for x1, x2 and y, respectively, which, when visualised, give a non-linear plot. I add a constant column to the dataset to account for the bias while fitting the model. this acts as the 3rd column in the X set for both train and test. We also mention m,n as the dimensions of the X matrix.

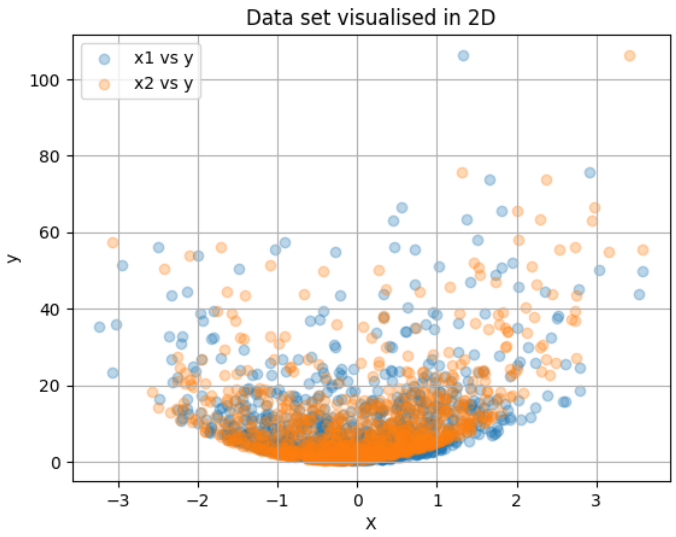

Figure 1: Training data visualization plot

2 Part (i)

The code for closed form solution of w gives us these values: [1.76570568, 3.5215898, 9.89400832], where the last term is the constant bias term.

3 Part (ii)

Coding the same by Gradient descent gives us the final value: [1.76546732, 3.52157758, 9.89354075]. It takes a lot of iterations to converge, but in the process, we observe that:

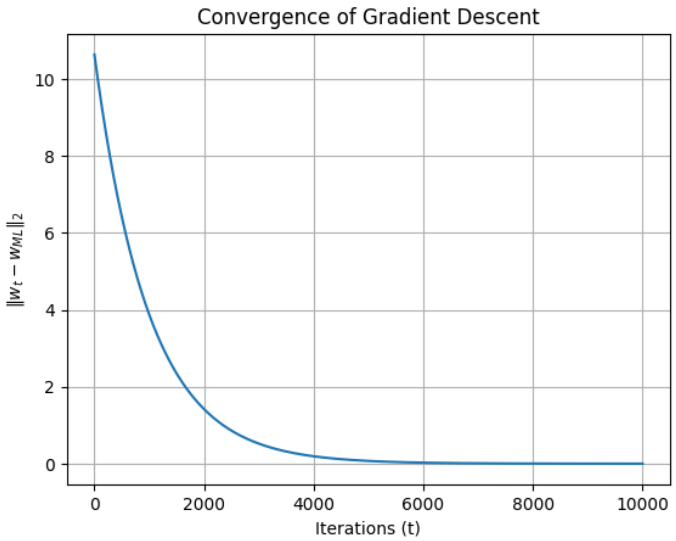

Figure 2: Standard Gradient Descent Convergence

- Initially, the norm decreases very rapidly, giving us rapid convergence. As the algorithm progresses, the decline in the norm slows down until the weights converge to the optimal value (the norm approaches zero).
- The choice of learning rate (lr) has a large impact on convergence speed:
  - If it is too large, the method may fail to converge.
  - If it is too small, convergence may be slow.

  So it is important to try out a few values of the learning rate to ensure it is optimum for the algorithm and given data.

4 Part (iii)

Coding the same by using Stochastic gradient descent gives us the final value: [1.76550418 3.52158385 9.89355694]. It takes lesser number of iterations to converge, but in the process, we observe that:

- The plot shows that the error decreases over time, indicating that the weights gradually converge to the optimal weights (w_{ml}).

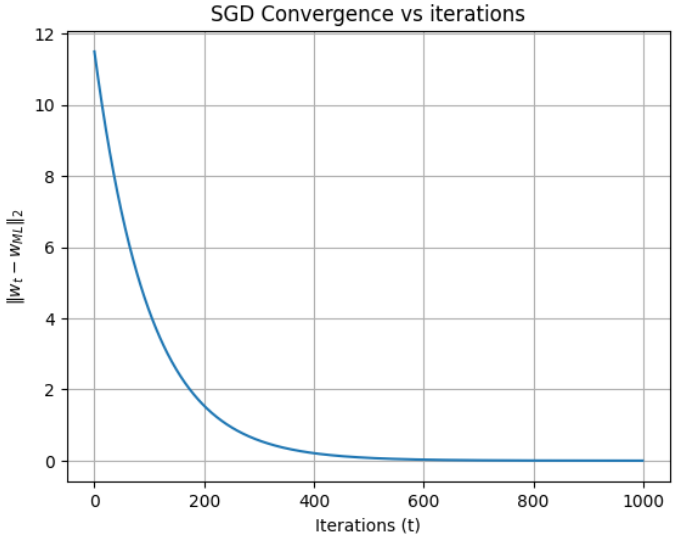

Figure 3: Stochastic Gradient Descent Convergence with batch size = 100

- The no of iterations needed here is much lower than the previous one (with the same learning rate), as this works on random sampling and averaging.
- This algorithm is better in terms of time complexity.

5 Part (iv)

Consider the cost function as

$$J = \frac{1}{2}||X^T w - y||_2^2 + \frac{1}{2}\lambda||w||_2^2.$$

Which can also be represented as

$$J = \frac{1}{2m}\sum_{i=1}^{m}(wx_i - y_i) + \frac{\lambda}{2m}\sum_{j=1}^{n} w^2.$$

Its gradient becomes

$$\nabla J = \frac{1}{m}(X^T(Xw - y) + \lambda w).$$

The plot shows us that the best regularisation parameter = 78, i.e. regularisation gives us the overall coefficient = 78/2 = 39.

On substituting lambda = 78 in the algorithm, we get weights w_r = [1.61374116, 3.28916492, 9.18127538].

We notice that:

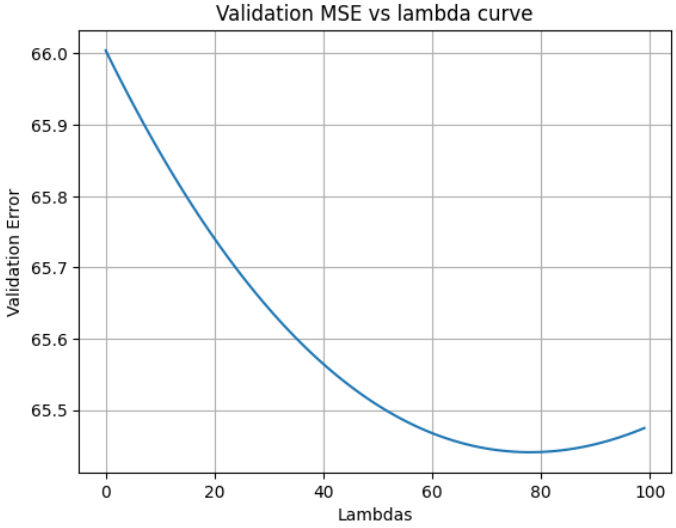

Figure 4: Cross Validation loss for ridge regression, we get a minima at 78

- Total MSE by OLS estimator = 66.00545933461238
- Total MSE by Ridge = 65.44104337942724

Ridge regression imposes a penalty for some weights, which helps us to reduce overfitting. Ridge Regression frequently might result in a minor increase in bias (due to regularization), but it can dramatically reduce variance, allowing for improved generalization on previously unseen data.

OLS method can reduce bias but may have large variance, particularly in complex models with noisy data. So, we get less MSE in ridge regression than in MLE. Hence, Ridge regression is better on test data.

6 Part (v)

Finally, we try to perform kernel regression on the dataset - computing both polynomial kernels (with degree 2) and Gaussian kernels.

The results obtained as follows:

- Total MSE by Polynomial kernel = 59.475665501673646
- Total MSE by Gaussian kernel = 0.9987910678941685

The phenomenally low MSE in the case of Gaussian kernel is due to the further optimisation of the variable σ in

$$k(x, x') = exp\left(\frac{-||x - x'||^2}{2\sigma^2}\right); \sigma > 0$$

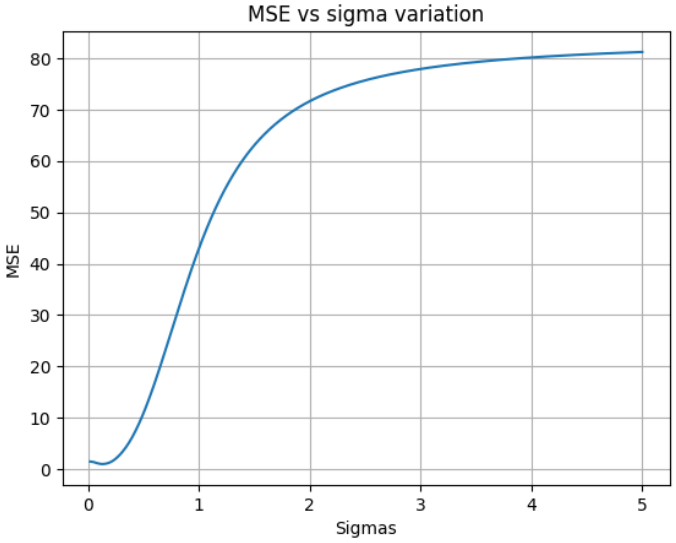

Figure 5: Optimising Gaussian Kernel over parameter sigma, we get a minima close to 0

On visualising the data, we find that it is certainly a non-linear relationship between X and y, which is governing it. Both Polynomial and Gaussian kernels help the model fit the dataset very well. But, the Gaussian kernel maps X to an infinite dimension space, not just a quadratic space like the polynomial kernel. So, it is able to capture the non-linear relations in the data extremely well. This is why we chose the Gaussian kernel.

Kernel Regression, especially the Gaussian kernel, can model the data with high accuracy as it transforms the feature vector to higher dimensional space, which can be fitted better by regression models. This transformation to a higher dimension suits the data really well, which is better than a simple OLS case.

As kernel regression helps to diminish the effect of outliers, it makes the model more robust too.

All of the above mentioned points explain us why the overall MSE of test data diminishes to a value even less than that obtained in ridge regression and therefore, the gaussian kernel is much better than standard least squares regression algorithms.