CS5691: Pattern Recognition and Machine Learning

Assignment 2: Regression

Shrivarshan K, MM20B058

Question 1:

(i).

The least squares solution to the regression problem is the vector w_{ML}, which minimizes the function $f(w) = ||X^Tw - y||^2$, where w is the variable. The optimum w (w_{ML}) is given by $w_{ML} = (XX^T)^{-1}Xy$. w_{ML} is calculated using this expression and error value is calculated.
The error value which is $f(w_{ML}) = 396.8644186272515$.

(ii).

The alternate way of obtaining w_{ML} is using the Gradient Descent approach as it is not always computationally efficient to calculate w_{ML} using the above expression.

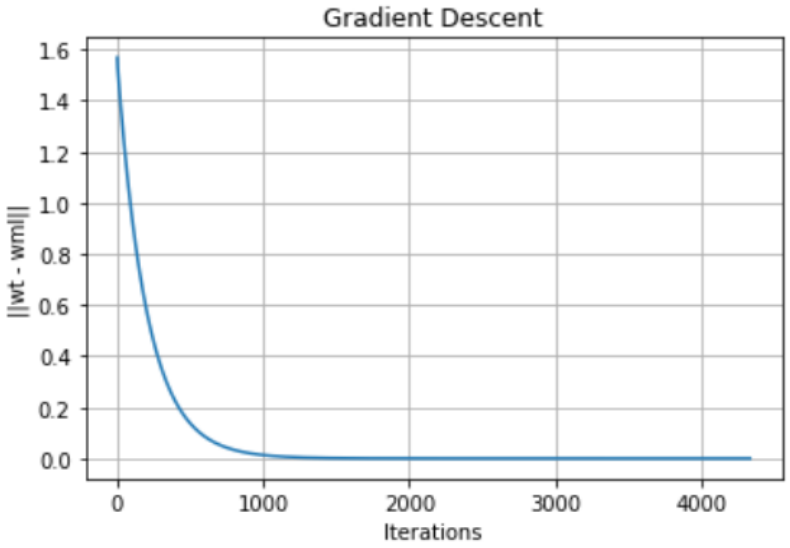
The step size or learning rate for each iteration has to be chosen from our side. The learning rate is decided to be kept constant for all iterations to reduce computational cost. After varying the values from $1*10^{-6}$ to $1*10^{-5}$, the optimal step size obtained is $4*10^{-6}$. Beyond this, the algorithm never converges as the gradient explodes. The algorithm converges in around 4336 iterations using this step size.

The gradient function is f'(w) which should be calculated for each w. The gradient is given by $f'(w) = 2*(XX^Tw - Xy)$. For the algorithm, we have initialized our w vector to be a zero vector. Now we run the algorithm with this w. The w gets updated using the following rule:

$$\mathbf{w^{t+1} = w^t - (lr)*grad\ (w^t)}$$

Where w^t is the parameter vector at the t^{th} iteration, this update rule is performed till the algorithm converges. Ideally the convergence criterion would be when the gradient becomes 0. I have taken the convergence criterion as gradient reaching $1*10^{-5}$. This is done because after this value, the w barely changes and just adds to more computational cost.

The plot of $||w^t - w_{ML}||$ versus the number of iterations obtained is shown below:



The value || w^t - w_{ML} || varies steeply for initial 500 iterations and converges at 4336^{th} iteration with the value of around $7.12*10^{-9}$. The error value which is $f(w^{4336}) = 396.86441862725167$. This means that w obtained from gradient descent has converged to w_{ML}.

(iii).

We know that the gradient of error term involves calculating the XX^T which is the covariance matrix of the data matrix X. This might be computationally costly when X is a high dimensional matrix. To overcome this Stochastic Gradient approach is used which is as follows:

First, we decide on a batch size (given 100) and a suitable learning rate. We obtained the optimal step size by trial and error again. The optimal step size is 0.0002. From the training dataset, we randomly select 100 datapoints and consider this as our new dataset. Using this new dataset, we calculate the gradient of our parameter vector (w, after being initialized to 100*1 zero vector) using the same formula as we used in the Gradient Descent algorithm. We update the w by using the same update rule:

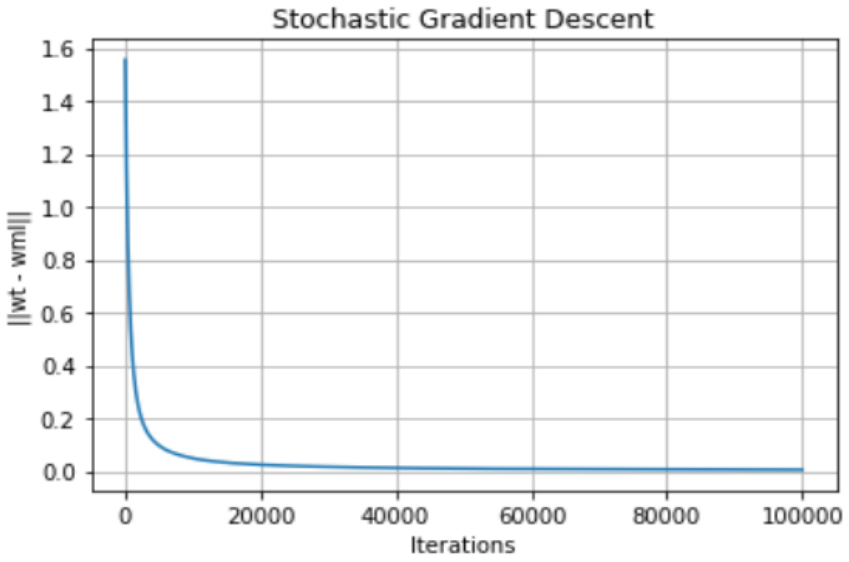
$$\mathbf{w^{i+1} = w^i - (lr)*grad(w^i)}$$

For each iteration, we randomly choose 100 points as the new dataset and update the w using the gradient of the new dataset. This is done till convergence. The convergence criterion differs from

that of the one used in Gradient Descent. Another step in Stochastic Gradient Descent is that the wt is given by the mean of all the w that were calculated till the t^{th} iteration:

$$w^t = (\text{Sum of } w^k \text{ for } (k = 1 \text{ to } t)) / t$$

The plot of $||w^t - w_{ML}||$ versus the number of iterations for Stochastic Gradient Descent obtained is shown below:



We can observe that the graph falls steeply till 10000 iterations and then starting to converge. The $||w^t - w_{ML}||$ value for Gradient Descent after the final iteration was $7.12*10^{-9}$. Meanwhile, after 100000 iterations, the $||w^t - w_{ML}||$ value for Stochastic Gradient Descent was 0.00546731.

Even though the plot is decreasing, it is not monotonically decreasing. This is because only some datapoints are being used for computation. So the value at each iteration might be greater than the previous iteration also by a small amount (this depends on the datapoints chosen).

The error value after 100000 iterations for this algorithm is 396.8886025212374, which is higher than that obtained from Gradient Descent and closed form solution.

Question 2:

(i):

We are implementing the Gradient Descent Algorithm for Ridge Regression. Ridge regression is not an unconstrained optimization problem but has an L2-norm regularisation. This constraint pushes the weights in the w vector towards 0. The objective function that we optimize in ridge regression is given by:

$$f(w) = ||X^T w - y||^2 + \text{lambda}*||w||^2$$

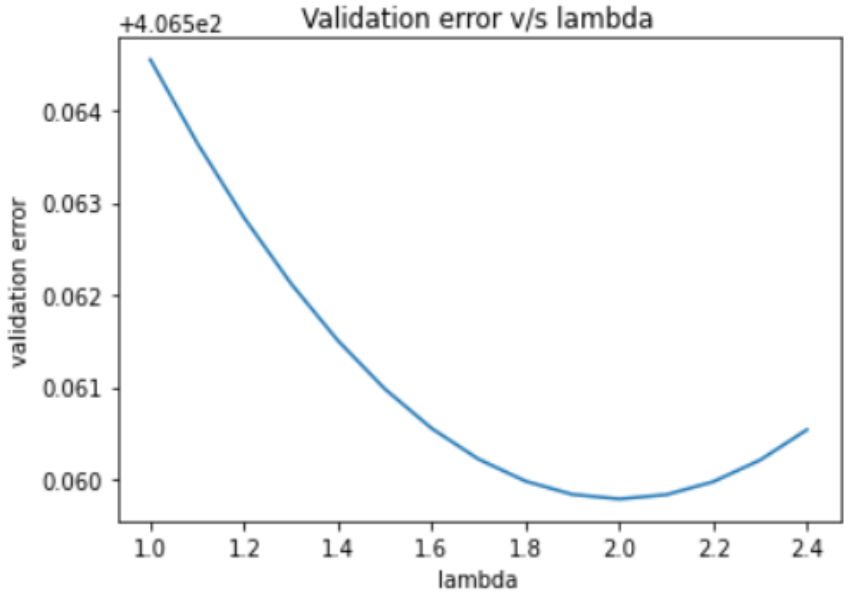
The Gradient Descent algorithm used for Ridge regression is the same except for that the gradient term will change. It is given by:

$$f'(w) = 2*(XX^T - Xy + \text{lambda}*w)$$

Here, lambda is a hyperparameter, and it influences the value of w_R obtained. For different lambdas, we will get different solutions. Hence we need to find the lambda that best suits our dataset.

(ii):

To find Lambda, we implement a K-Fold Cross Validation. Here we divide the data into K bins and use one bin for testing and the rest for training. To find the optimal w for each validation, gradient descent approach is used. A 5-fold CV was implemented then, the error on the validation set is calculated for each lambda. Validation error was plotted for different values of Lambda ranging from 1 to 2.5 in steps of 0.1. Initially to find optimal lambda, lambdas were considered in the powers of 10 from [1e-4,1e4] and then to find exact value lambdas from 1 to 2.5 were considered. The plot obtained is shown below:



From the plot optimal lambda is obtained as 2. Using this lambda w_R obtained by plugging in lambda into $f(w) = ||X^T w - y||^2 + \text{lambda}*||w||^2$ and then gradient descent approach is carried out with initial w as 100*1 zero vector.

Now we evaluate the performance of w_{ML} and w_R for lambda = 2 on our test dataset consisting of 500 datapoints.
The mean of the sum of squared errors for each is as follows:

- For $w_R = 0.3698822679778238$
- For $w_{ML} = 0.37072731043925455$

As we can see, w_R seems to perform a bit better than w_{ML}. Hence Ridge Regression performs better on test data compared to Linear Regression. This might be because, in Ridge, the weights for insignificant features are close to 0. This reduces the tendency of our model to overfit on the training data and hence perform better on new test data. The w_{ML} obtained for Linear Regression has likely overfitted to the training data and hence performs worse when new data is tested on it.