A Project Report

On

Linear Regression (Assignment-2)

BY

KESHAV BERIWAL	2017B4A71301H
AMAN BADJATE	2017B3A70559H
GARVIT SONI	2017B3A70458H

UNDER THE SUPERVISION OF

DR. N.L.BHANU MURTHY

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF

CS F320: FOUNDATIONS OF DATA SCIENCE

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

HYDERABAD CAMPUS

OCTOBER - NOVEMBER, 2020

Contents

List of Figures

1 Objectives

The main focus of this assignment is to implement various Linear Regression techniques. The following techniques are implemented in this assignment in python using numpy, pandas and matplotlib libraries.

- Normal Equation Solution
- Gradient Descent Algorithm
- Stochastic Gradient Descent Algorithm

2 Data Preprocessing

The data consists of 1338 rows and 4 columns. The 4 columns are age, bmi, children and charges. "age, bmi, children" are features (independent variables) and "charges" is the target variable (dependent variable). Before training the model, the data was pre-processed by standardizing all the variables. Standardizing a variable means subtracting it from its mean value and then dividing it by standard deviation. The comparison between the original and standardised data is shown in figure 1. This is done to ensure that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

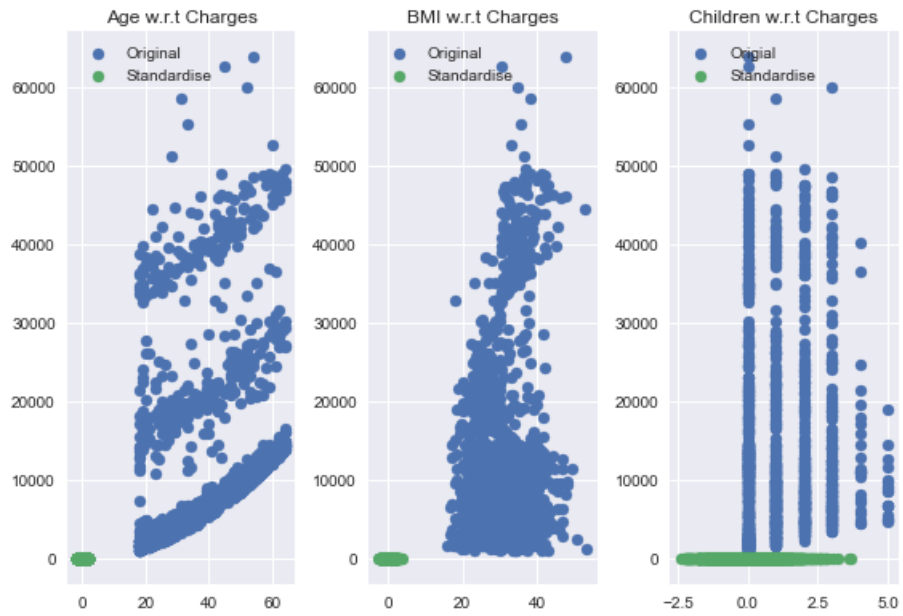

Figure 1: Comparison between Original and Standardised Data

3 Methodology

After preprocessing, the data was shuffled randomly 20 times to get 20 different models. Each model was then split into training data (70%) and testing data (30%). Training data was passed on to different algorithms to find out the weights of the features. After calculating these weights, for each algorithm we found out the predicted value of the target variable y_{pred}. The RMSE is given by the formula -

$$RMSE = \sqrt{\frac{1}{2} * \sum_{i=1}^{n} (y_{pred}^{i} - t^{i})^2}$$

The R^2 is given by the formula -

$$R^2 Score(\%) = (1 - \frac{\sum_{i=1}^{n} (y_{pred}^{i} - t^{i})^2}{\sum_{i=1}^{n} (y_{avg}^{i} - t^{i})^2}) * 100$$

Accuracy of each model is measured in terms of R^2 score and root mean squared error (RMSE). We then report the weights obtained by each method for the model which has the highest accuracy.

4 Linear Regression Optimisation Methods

4.1 Normal Equation Solution

In this method, we solve simple linear equations that are obtained by taking the partial derivative of error function with respect to each weight and setting it to 0. By taking weights as a column vector, we can represent all the linear equations in a matrix of form AW = B and W = $A^{-1}B$. On running our python code for this method on 20 models, we get the following results-

- The final weights obtained are $w_0 = -0.0189, w_1 = 0.2772, w_2 = 0.1396$ and $w_3 = 0.0575$.
- The RMSE mean, variance and minimum value for training data are 0.9420, 0.0002 and 0.9235 respectively.
- The RMSE mean, variance and minimum value for testing data are 0.9299, 0.0010 and 0.8528 respectively.
- The R^2 score mean, variance and maximum value for training data are 12.3458, 0.6205, 14.2310 respectively.

- The R^2 score mean, variance and maximum value for testing data are 10.4552, 4.4087, 13.9188 respectively.

4.2 Gradient Descent Algorithm

This is the basic Gradient descent algorithm without any regularization technique. The hypothesis of gradient descent for 3 features is given by

$$y(w) = w_o + w_1x_1 + w_2x_2 + w_3x_3.$$

The error function (loss function) that we used for the algorithm is sum of squares of errors which is given by

$$J(w) = \frac{1}{2} * \sum_{i=1}^{n} (y_{pred}^i - t^i)^2$$

where,y_{pred}^i is the predicted value of target variable and t^i is the true value for i^{th} example. The learning rate that we used is 0.00001. The initial weights are assumed to be 1. The weights for the algorithm are updated after every iteration and total number of iterations is 10^5. We also have added a breaking condition in the loop which is the absolute difference between current error and the previous error. If this value is less than 10^{-6} then we break the loop. On running our python code for this method on 20 models, we get the following results-

- The final weights obtained are $w_0 = -0.0186, w_1 = 0.2772, w_2 = 0.1396$ and $w_3 = 0.0577$.
- The RMSE mean, variance and minimum value for training data are 0.9420, 0.0002 and 0.9235 respectively.
- The RMSE mean, variance and minimum value for testing data are 0.9299, 0.0010 and 0.8528 respectively.
- The R^2 score mean, variance and maximum value for training data are 12.3458, 0.6205 and 14.2310 respectively.
- The R^2 score mean, variance and maximum value for testing data are 10.4536, 4.4220 and 13.9210 respectively.

4.3 Stochastic Gradient Descent Algorithm

In this algorithm, the same error function and hypothesis are used as that of gradient descent, the only difference here is that we update the weights by calculating the gradient using a randomly selected data point. The initial weights are assumed to be 1. The learning rate that we used is 0.00005. We run the loop for 10^5 iterations. On running our python code for this method on 20 models, we get the following results-

- The final weights obtained are $w_0 = -0.0101, w_1 = 0.2763, w_2 = 0.1446$ and $w_3 = 0.0687$
- The RMSE mean, variance and minimum value for training data are 0.9421, 0.0002 and 0.9236 respectively.
- The RMSE mean, variance and minimum value for testing data are 0.9303, 0.0009 and 0.8535 respectively.
- The R^2 score mean, variance and maximum value for training data are 12.8235, 0.6187 and 14.2146 respectively.
- The R^2 score mean, variance and maximum value for testing data are 10.3687, 4.9714 and 13.9688 respectively.

5 Result Visualisation

5.1 Loss vs Epochs

The loss vs Epochs graphs have been plotted for both Gradient Descent and Stochastic Gradient Descent as shown in Fig 2 for minimum rmse value and the conditions specified above. As we can see from the graphs, loss/error for gradient descent saturates around 400 iterations while for stochastic gradient descent it saturates around 60000. This implies that gradient descent converges faster than stochastic gradient descent.

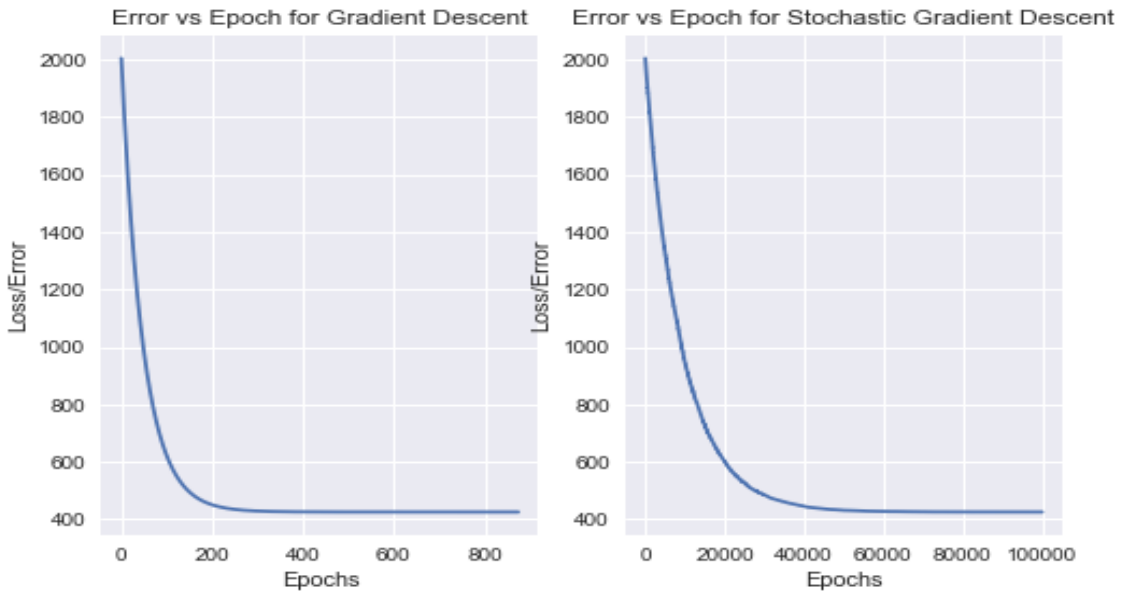

Figure 2: Loss vs Epochs

5.2 Weights vs Epochs

The Weights vs Epochs graphs have been plotted for both Gradient Descent and Stochastic Gradient Descent as shown in Fig 3. As we can see from the graphs, weights of gradient descent reach a constant value around 400 iterations while for stochastic gradient descent, the weights reach a constant value around 80000 iterations. This is because the loss function of gradient descent reaches saturation before the loss function of stochastic gradient descent algorithm.

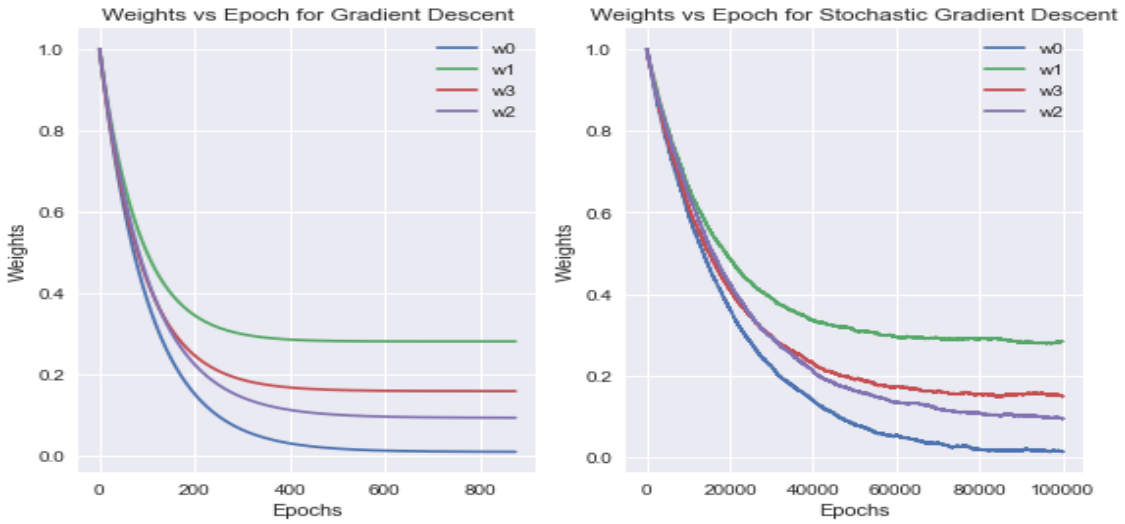

Figure 3: Weights vs Epochs

5.3 Accuracy(R^2 Score) vs Epochs

The Accuracy(R^2 Score) vs Epochs graphs have been plotted for both Gradient Descent and Stochastic Gradient Descent as shown in Fig 4. As we can see from the graph, accuracy increases initially as the iterations increases and then the accuracy saturates after sometime and does not change with number of iterations. The accuracy in the case of gradient descent saturates early than that in the case of stochastic gradient descent.

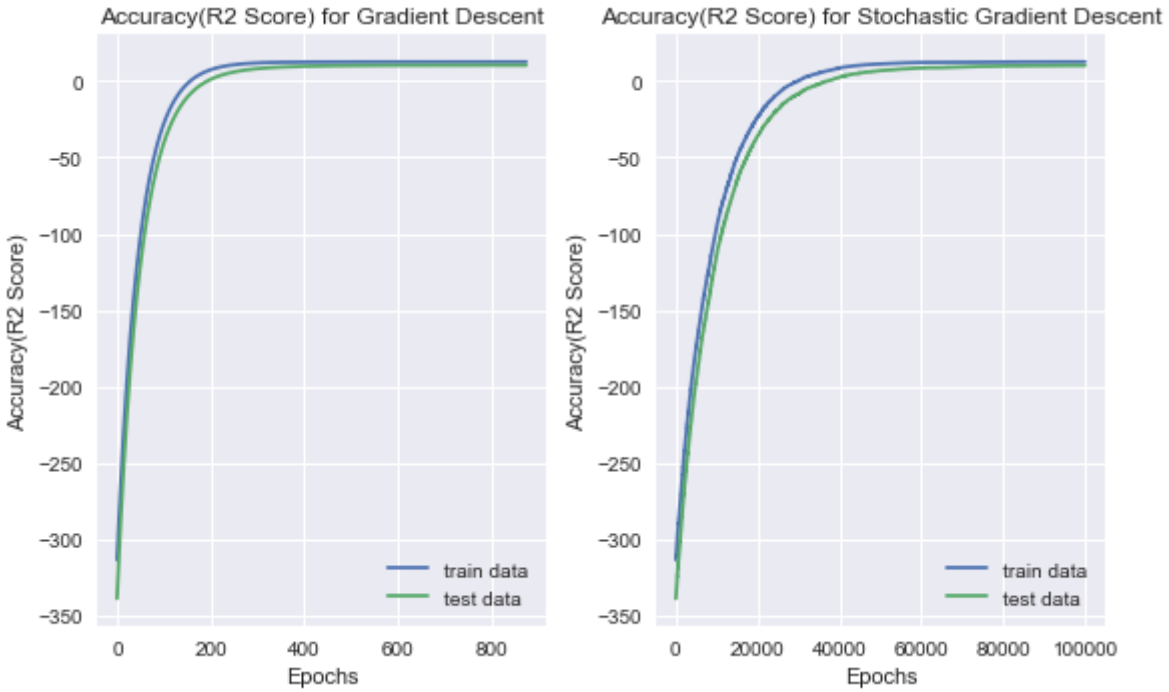

Figure 4: Accuracy(R^2 Score) vs Epochs

5.4 Loss vs Epochs for various Learning rates

In this section, we have plotted Error vs Epochs graph for Gradient Descent and Stochastic Gradient Descent for different learning rates.

5.4.1 Learning rate = 0.00001

In this graph, we can see that the error for gradient descent saturates for the given learning rate while the error for stochastic gradient descent does not saturate.

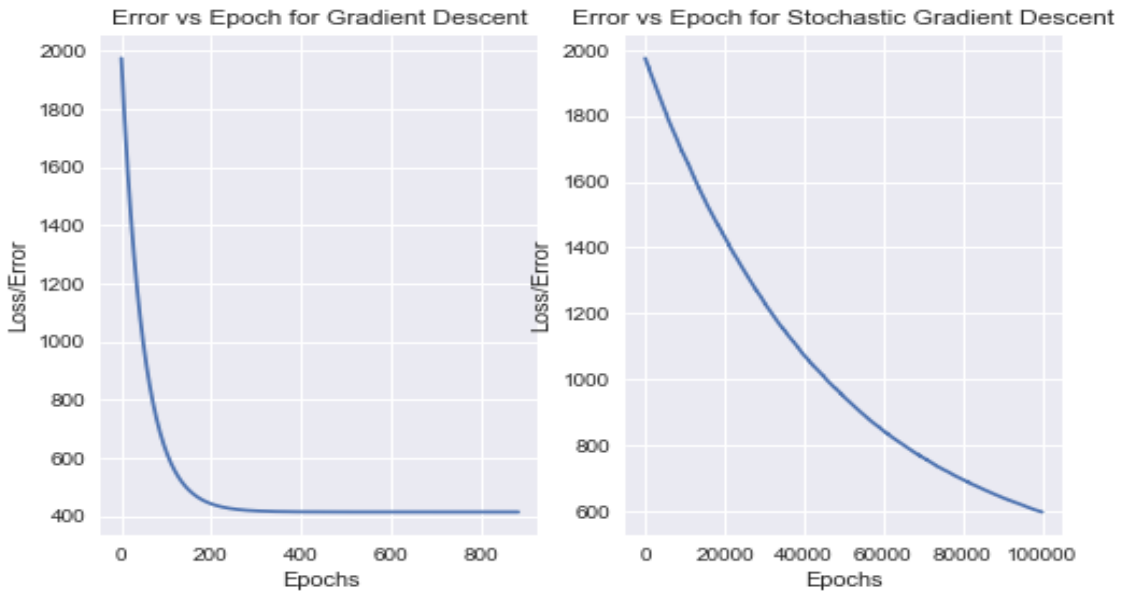

Figure 5: Loss vs Epochs(LR = 0.00001)

5.4.2 Learning rate = 0.0001

In this graph, we can see that the error for gradient descent saturates for the given learning rate around 40 iterations and the error for stochastic gradient descent also saturates around 4000 iterations. We can say that this is the optimal learning rate for both the algorithms.

Figure 6: Loss vs Epochs(LR = 0.0001)

5.4.3 Learning rate = 0.001

In this graph, we can see that the error for gradient descent saturates for the given learning rate very early because of large learning rate while the error for stochastic gradient descent does not saturate properly.

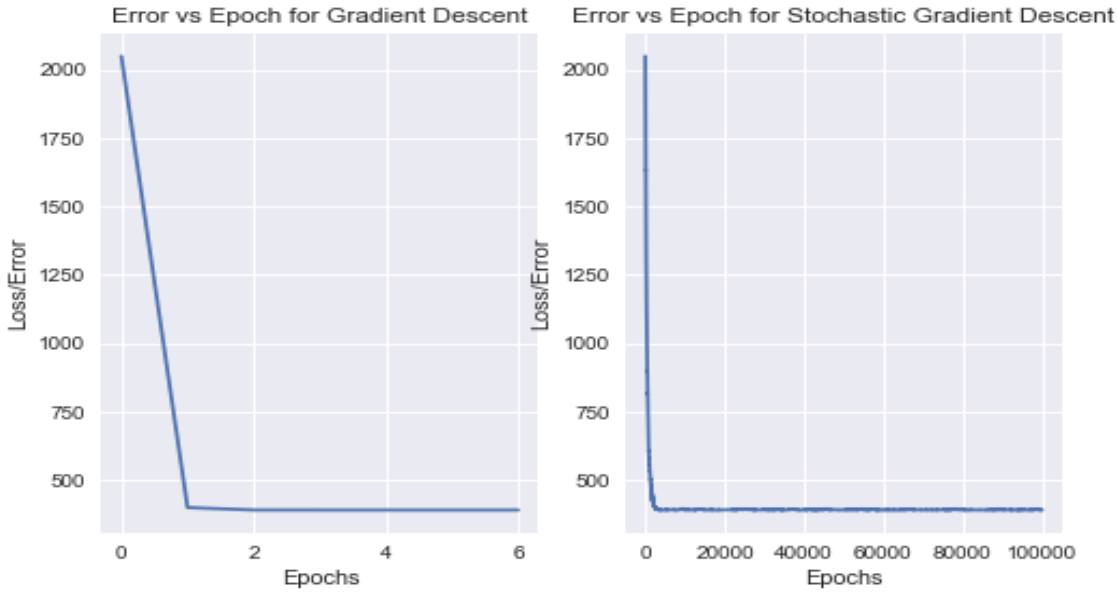

Figure 7: Loss vs Epochs(LR = 0.001)

5.5 Comparison of Different Linear Regression Algorithms

From the above results we can conclude that all of the above optimization techniques are good for approximating the linear regression model. For the given data, the model built using gradient descent has the minimum RMSE for the testing data. The minimum RMSE of the other 2 models are also very close to this range.

	RMSE			R2 Score (%)		
	Mean	Variance	Minimum	Mean	Variance	Maximum
Gradient Descent	0.9299	0.0010	0.8528	10.4536	4.4220	13.9210
Normal Equation	0.9299	0.0010	0.8528	10.4552	4.4087	13.9188
Stochastic Gradient Descent	0.9305	0.0009	0.8535	10.3687	4.9714	13.9688

Figure 8: Summary of Different Linear Regression Algorithms

6 Discussion

After doing this project we can draw the following conclusions-

- The regression model built using Gradient Descent algorithm, Stochastic gradient Descent algorithm, by solving normal equations will not built exactly the same model but will be quite similar because by solving normal equations we will get the exact global minimum whereas in the other two a value close to global minimum is calculated. Gradient Descent method is computationally complex because we calculate the slope each time while for stochastic gradient descent we choose only one value and calculate the slope. Stochastic gradient descent is better than both solving normal equations and gradient descent while working with real world data because it is computationally less complex.

- Machine learning algorithms like linear regression that use gradient descent as an optimization technique require data to be scaled.

  $$w_j = w_j - \alpha \sum_{i=1}^{n} (y_{pred}^i - t^i) x_j^i$$

  The above image is how the weights change in a simple gradient descent algorithm. For stochastic gradient descent, summation won't be there. The presence of feature value X in the formula will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature. To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

- If we keep on increasing the number of iterations while calculating weights for a given learning rate, the difference between the current error and the previous error would keep on decreasing to the point where we are very close to global minima. However, after a very large number of epochs (around 10^{10}), error will saturate and very minute difference will exist between two iterations.

- On studying the plot of gradient descent and stochastic gradient descent, we can infer that for lower learning rates(0.00001,0.0001 etc), weights in gradient descent come to a

constant level in less number of iterations while weights in stochastic gradient descent take more number of iterations. Also the error in gradient descent decreases faster than the error in stochastic gradient descent. Hence, gradient decent, though computationally more intensive, converge to global minima at a faster rate for low values of learning rate. However, for high values of LR (0.01,0.1 etc), Stochastic Gradient converges faster as compared to Gradient Descent.

- The learning rate has to be chosen in a very wise manner. The learning rate controls how quickly the model is adapted to the problem. Very small learning rate, will result in more number of iterations to reach the global minima. On the other hand very large learning rates (2 or higher) will result in divergence and we will never be able to converge to global minima. Hence the learning rate should be chosen in such a way that can cause the model to converge too quickly to a suboptimal solution.

- Whether to use bias term or not, depends on how the data is distributed. For example, if it is centred around 0 then the bias term won't matter much and would give us the the same minima. However, we generally need it.

- The weight vector associated to each input dimension after training gives information about its relevance for predicting the target attribute. The feature which affects target variable i.e. charges the most is the age and the variable which affects the target variable least is number of children.