ONLINE LINEARIZED LASSO

Implementation of Online Linearized LASSO

-presented by
B.STAT 1ST YEAR STUDENTS
AKANKSHA DAS(BS2307)
SOHAN KARMAKAR(BS2349)
UNNATI MANDAL(BS2355)
SOUMYA CHANDA(BS2350)

A project work, presented for the Course:
PROGRAMMING AND DATA STRUCTURE ,
B.STAT 1st year, Semester-1

Department Name:BACHELOR OF STATISTICS
University Name:INDIAN STATISTICAL INSTITUTE
Country:INDIA
Date:21 DECEMBER,2023

Acknowledgement

We are grateful to our respected Professor Malay Bhattacharyya, who gave us the opportunity to work on such an interesting topic. He has also been instrumental in guiding us through this project successfully. With his wisdom and knowledge, we were able to complete this report with ease under his supervision which was a very enriching experience for all of us!

Contents

Implementation of Online Linearized LASSO

Project Report

December 2023

1 Objective

While the algorithm in the research paper is self-explanatory, we aim to implement it in python. There is no python library available for Online Linearized LASSO, so we first implement LASSO regression using gradient descent algorithm for minimising the cost function without using Python standard library, then we would attempt to implement Online Linearized LASSO using the same algorithm.

2 Gradient Descent

Gradient Descent is an optimisation algorithm used for minimising the cost function of various machine learning algorithms. It is used for updating the parameters of the learning model.

for e.g $a_2 = a_1 - L\frac{dJ}{dw}$
where,
L = learning rate
J = cost function
a_2 = updated parameter
a_1 = original parameter

This is repeated numerous times to minimise the cost function.

3 LASSO

3.1 Algorithm

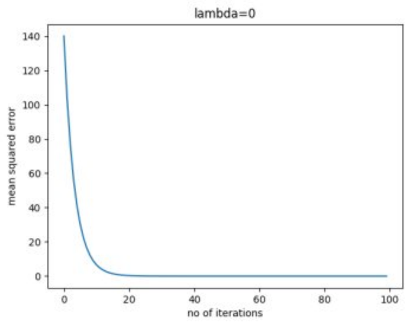
1. Define the class Lasso_Regression() and define the methods: init() : to initialise all the parameters
   fit() : to fit the data-points on a straight line using LASSO
   update_weights() : To update the parameters weight and bias in each iteration
   predict() : to predict Y with respect to X using the present weight and bias
   mean_squared_error() : to print the mean squared error

2. Differentiate the Loss function of LASSO regression with respect to weight and bias and apply gradient descent algorithm to get optimum values for weight and bias.

3. Calculate mean square error.

3.2 Exploring the effect of parameters on the mean squared error

3.2.1 Linear Data with no Error



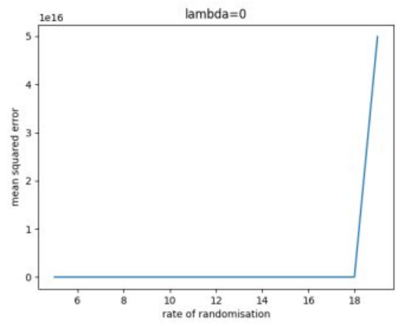
As we can see clearly see, as the **number of iterations** increase, the mean squared error decreases drastically till a certain stage then remains more or less constant. We have enclosed the code for this graph in the LASSO file. We will add many such with one of the parameter constant and other two varying to find out the optimal loss function. We have taken $\lambda = 0$ as that indicates no bias.

The **learning rate** increases the accuracy of predictions and as we increase the learning rate we also need to increase the number of iteration otherwise we will get peculiar output.

We also have introduced a new parameter r that is **degree of randomization** for example choosing random integer linear data between $1-10$ will have less randomization than choosing similar data between $1-25$ and for that we have to take more number of input data points or increase the value of lambda parameter which in turn increases the bias.

Now we had checked we have to increase the number of input data points by a large extent to counter the increase in degree of randomization so much so that it is not supported by our system due to an overwhelming amount of time complexity so let us increase the lambda value instead which even though increases the bias value by a little for a small batch size, it brings significant improvement to the Mean Squared Error without increasing time complexity.

But in practical implementation, the bias will increase and decrease the mean square error with no uniformity so its not a good approach to increase the bias too much.



To conclude, we observe that the more the no of iterations, the less the learning rate, the less the degree of randomisation is, the less in general the mean squared error is. The λ parameter brings upon bias but if it reduces the mean square error by a lot then its a reasonable trade off.

3.3 Non-Linear Sparse Data

Data generation: Take a sparce matrix as the true underlying beta star and add a small error that follows $N(0,1)$. For learning_rate=0.01,no_of_iterations=100, lambda_parameter=20 the algorithm gives an error of order 10^{-1}

4 Online Linearized LASSO

4.1 Algorithm

1. We define all functions as in LASSO. There will be an extra parameter of weight on the initial batch and each successive data point.

2. We will differentiate the loss function of Online Linearized LASSO and define the update_weights() function accordingly.

3. We will print the beta cap sequence. (the list of the optimised weights after addition of each data point.)

4. We will calculate the mean squared error for training and test set.

4.2 Implementation for non linear sparse data

We have implemented the code for the same example in the LASSO code. As the code is taking a ridiculous amount of time to run, we were not able to check the trend of mean square error with the various parameters. Hence we could not make a guess of of the optimum conditions under which the loss function is minimised. We also have not yet received any conclusive evidence that Online Linearized LASSO has an edge over the popular LASSO Regression.

.

5 Beyond Implementation

We would like to explore the advantages and limitations of Online Linearized LASSO further.