A Report

On

**Polynomial regression using**

**Gradient Descent and Stochastic Gradient Descent methods**

BY

**Aaryan Gupta 2018B1A70775H**

**Abhinava M 2018A8PS0844H**

**Shubham Singla 2019A3PS0392H**

Under the supervision of

**Dr. N.L. Bhanu Murthy**

**SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS OF**

**CS F320: Foundations of Data Science**

**Assignment-1**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**HYDERABAD CAMPUS**

**(October 2021**)

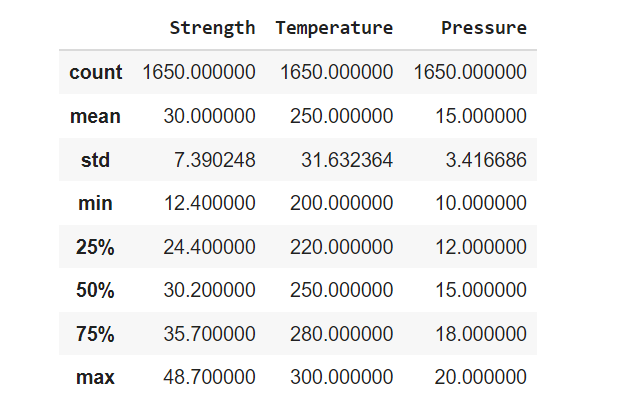
**Introduction**

In this assignment, we have implemented Polynomial regression with varying degrees from 0-9, using Gradient Descent and Stochastic Gradient Descent algorithms. The Polynomial Regression technique models the relationship between the independent variable and dependent variable to be a degree-n polynomial.

Both the algorithms i.e, Gradient descent (GD) and Stochastic Gradient Descent (SGD), minimize an error function by iteratively updating a set of parameters. In Gradient Descent algorithm, all the samples in the training data are run to do a single update for a parameter in a particular iteration, but in Stochastic Gradient Descent algorithm, a subset of training data is run to do the update for a parameter in a particular iteration.

The dataset consists of two features i.e. ‘Strength’ and ‘Temperature’ applied to a certain

piece of plastic. Using these features as independent variables (x1 and x2), we predict amount of ‘Pressure’, i.e., the dependent variable (y) that the plastic can withstand by constructing polynomial features and optimizing the weights by using GD and SGD without any regularization. Here, the dataset was given in a csv file named ‘FoDS-A1.csv’, which had 3 columns i.e., ‘Strength’, ‘Temperature’ and ‘Pressure’ with 1650 rows with data respective to each column. Below is the description of data with mean, standard deviation, minimum, maximum and values corresponding to each percentile- 25, 50 and 75, when ordered.



The dataset was first converted into an array with the help of numpy and also shuffled with the help of numpy’s built-in function of random shuffling. Next, all the data features(x and y attributes both) were normalized. This shuffled and normalised dataset was split into 70% training data and 30% testing data.

**Without regularization:**

We compared the Gradient Descent and Stochastic Gradient Descent optimization models for the given dataset, fitting polynomials of degrees 0,2.....9.

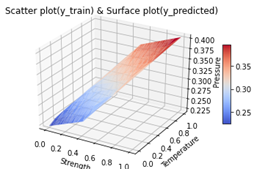
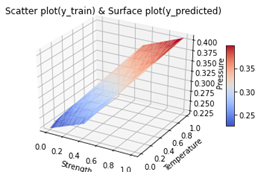
Below tabulated are training and testing errors obtained for each degree of polynomial:-

| Polynomial degree | GD train rmse | GD test rmse | SGD train rmse | SGD test rmse |
| --- | --- | --- | --- | --- |
| 0 | 8.15794377 | 5.84788297 | 8.14611566 | 5.84013965 |
| 1 | 6.11164678 | 4.53108397 | 6.11727901 | 4.53484685 |
| 2 | 4.85011973 | 3.71254385 | 4.76293142 | 3.65562305 |
| 3 | 4.27308435 | 3.32560245 | 4.25236116 | 3.31241156 |
| 4 | 4.09360967 | 3.19236746 | 4.06050215 | 3.17077225 |
| 5 | 4.10589819 | 3.18432373 | 4.20918397 | 3.25231868 |
| 6 | 4.19655948 | 3.22981896 | 4.21328741 | 3.23975788 |
| 7 | 4.30970122 | 3.29298958 | 4.39957603 | 3.35188925 |
| 8 | 4.41984973 | 3.35710889 | 4.44808243 | 3.37485024 |
| 9 | 4.51661414 | 3.41504195 | 4.53485 | 3.42502048 |

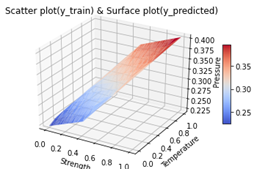
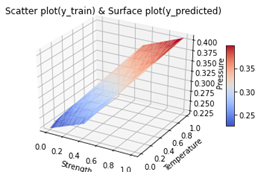
**Surface plots of polynomials- varying the degrees:**

**GD SGD**

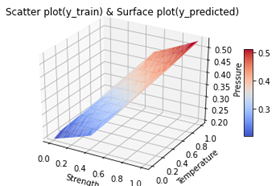
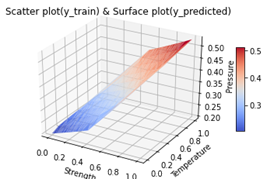
**Degree-0**



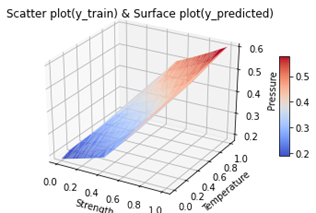
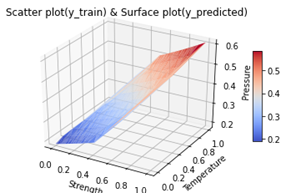
**Degree-1**



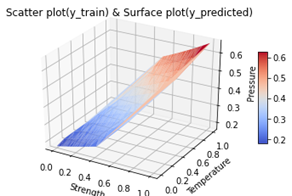
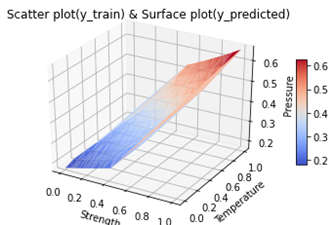
**Degree-2**



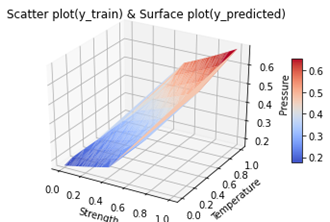
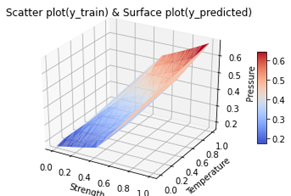
**Degree-3**



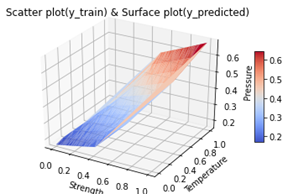
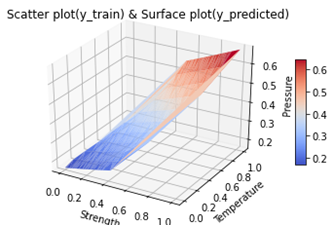
**Degree-4**



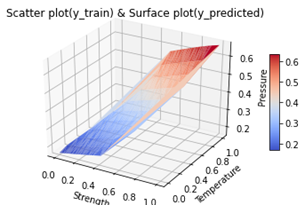
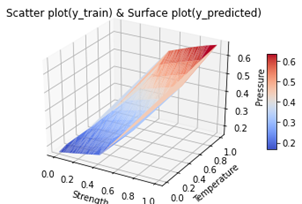
**Degree-5**



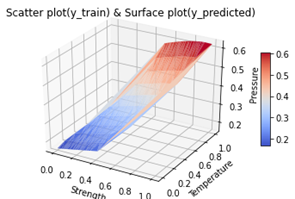
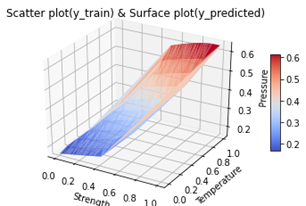
**Degree-6**



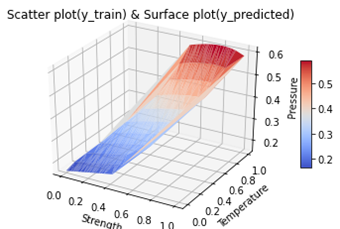
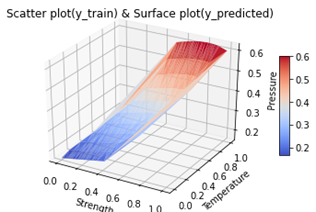
**Degree-7**



**Degree-8**



**Degree-9**



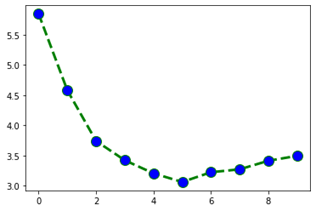
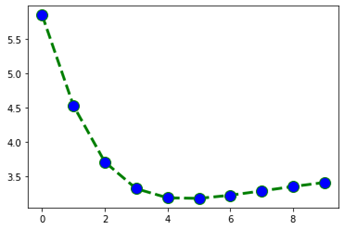
The graphs below show the (a) GD RMSE vs Degree of Polynomials and (b) SGD RMSE vs Degree of Polynomial

1. Values For GD RMSE for all 0-9 degrees-

[5.84788297 4.53108397 3.71254385 3.32560245 3.19236746 3.18432373 3.22981896 3.29298958 3.35710889 3.41504195]

1. Values For SGD RMSE for all 0-9 degrees-

[5.84551798 4.58013129 3.74070957 3.42109904 3.20452878 3.06126006 3.22305535 3.27121259 3.41294268 3.49428818]



(a) (b)

**Overfitting and regularization:**

For higher degrees, without regularization the parameters get tuned to the training data more and

more, reducing the training error; which also means they are getting tuned to that random noise

in the data. This means that the noise or random fluctuations in the training data is picked up and

learned as concepts by the model. Due to this, overfitted models give extremely low, nearly zero

training error, but abnormally high testing error.

As can be seen from the graphs above, higher degree plots try to accommodate the one or two

data points lying much higher than the others. It is getting tune to the random error in the data to

minimise training error. The minima is found at GD RMSE 3.18432373 for 5-degree polynomial and SGD RMSE 3.06126006 for 5-degree polynomial. Hence 5-degree polynomial provides us the best fit here in both the algorithms.

**Regularization**

Overfitting is a phenomenon that occurs when a machine learning or statistics model is tailored

to a particular dataset and is unable to generalise to other datasets. This usually happens in

complex models, like deep neural networks. Regularisation is a process of introducing additional

information in order to prevent overfitting. L1 and L2 regularisation owes its name to the L1 and

L2 norm of a vector w respectively.‘λ’ is a Hyper-parameter used in regularisation. If ‘λ’ was a

parameter, Gradient Descent would nicely set it to 0 and travel to the global minimum. Hence,

The control on ‘λ’ cannot be given to Gradient Descent and needs to be kept out. It will not be a

parameter but a Hyper-parameter.

A linear regression model that implements L1 norm for regularisation is called lasso regression, and one that implements (squared) L2 norm for regularisation is called ridge regression. So in our model firstly, to implement these two, the linear regression model stays the same.

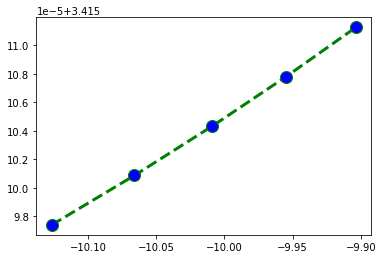
Below we have tabulated the minimum training and testing errors achieved using regularisation for GD, SGD for degree-9 polynomial with 5 different values of Lambda:

**Ridge regression: polynomial degree 9**

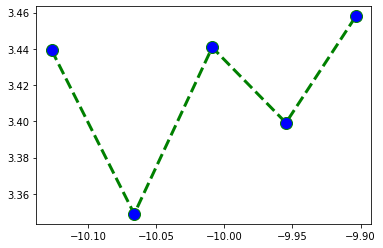
| Lambda | GD train error | GD test error | SGD train error | SGD test error |
| --- | --- | --- | --- | --- |
| 0.000004 | 4.5167 | 3.4151 | 4.5414 | 3.3966 |
| 0.00000425 | 4.5167 | 3.4151 | 4.5762 | 3.4552 |
| 0.00000450 | 4.5167 | 3.4151 | 4.7415 | 3.5638 |
| 0.00000475 | 4.5167 | 3.4151 | 4.6312 | 3.4915 |
| 0.000005 | 4.5167 | 3.4151 | 4.4447 | 3.3674 |

**Rmse vs log(lambda):**

**GD:**

****

**SGD:**

****

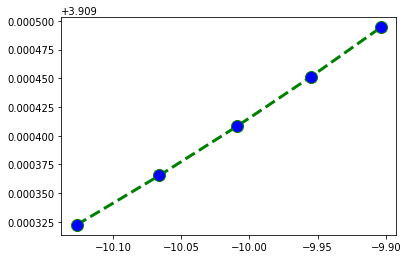
**The above graphs are for RMSE values for GD and SGD vs log lambda, the Lambda value is optimal for lambda=**0.000004 for GD, and for SGD 2nd value of lambda is most optimal

**Lasso regression: polynomial degree 9**

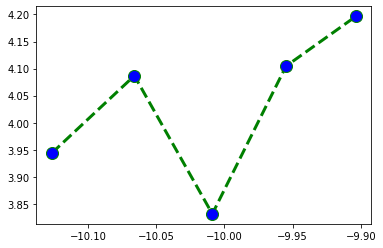
| Lambda | GD train error | GD test error | SGD train error | SGD test error |
| --- | --- | --- | --- | --- |
| 0.000004 | 0.52676 | 3.9093 | 5.3184 | 3.9436 |
| 0.00000425 | 0.52676 | 3.9093 | 5.5388 | 4.0867 |
| 0.00000450 | 0.52677 | 3.9094 | 5.1473 | 3.8315 |
| 0.00000475 | 0.52678 | 3.9094 | 5.5613 | 4.1046 |
| 0.000005 | 0.52678 | 3.9094 | 5.7033 | 4.1966 |

**Rmse vs log(lambda):**

**GD:**

****

**SGD:**

****

**The above graphs are for RMSE values for GD and SGD vs log lambda, the Lambda value is optimal for lambda=**0.000004 for GD, and for 2nd value of lambda is most optimal

SGD RMSE 3.06126006 for 5-degree polynomial with no regularization, is more optimal than all the models with regularization.