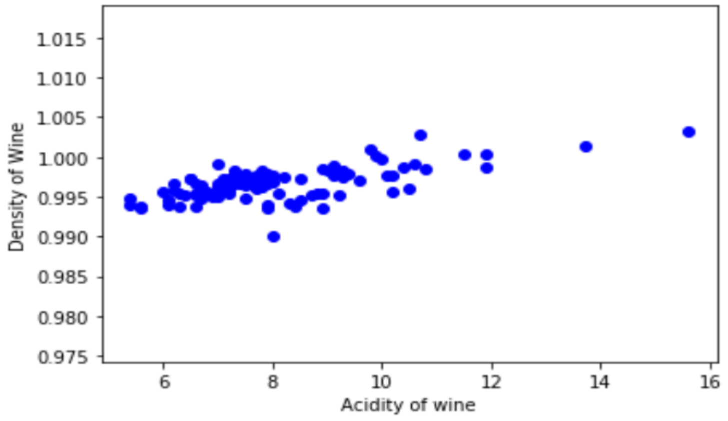
CVG Summer 2020 - Assignment 2 Machine Learning: Regression Analysis

Team Name: **Runtime_Esther**

1. Amogha TS (01FE18BAR006)
2. Saloni Shah (01FE18BCS183)

Question 1: Batch Gradient Descent

The task was to predict density of wine based on its acidity used least squares linear regression.



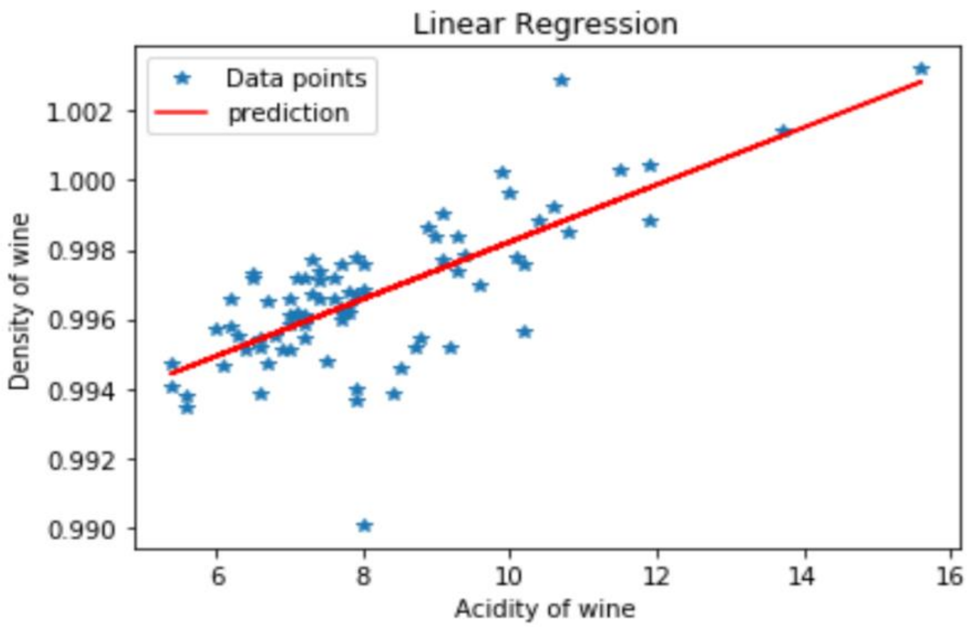
Algorithm:

```
Step 0: Initialize weights(here theta0 and theta1- Zero vector).
Step 1: Choose a value of learning rate and gamma(for convergence).
Step 2: Normalize the data to have 0 mean and variance 1.
Step 3: do    {
Step 4:               $\nabla\theta J(\theta)=(h\theta(x^i)-y^i)(x^i)$
Step 5:               $\theta^{t+1}=\theta^t$ - learning rate*$\nabla\theta J(\theta)$
Step 6: while{ ( $J(\theta+1)-J(\theta)$ ) >gamma (till convergence)
```

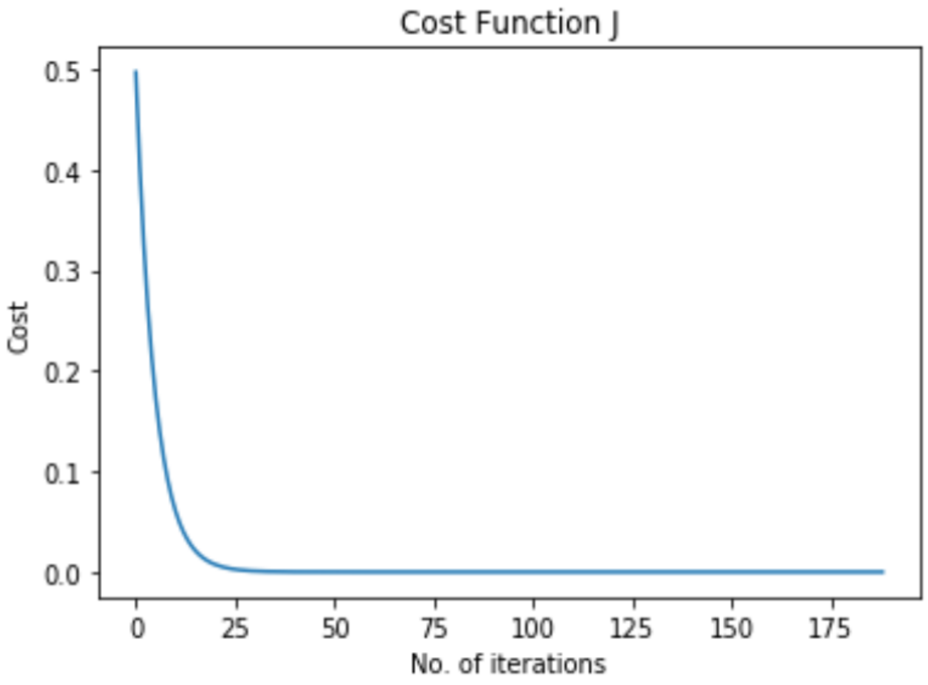
a.

- theta vector initialized to a zero vector-[0,0]
- learning rate=0.1
- Stopping Criteria: The difference in Cost function value i.e., $J(\theta+1)-J(\theta)$ becomes less than gamma(0.000000000001)
- Final Parameters: [0.99666962 0.00280919]
- Final Cost value:1.0500471943373033e-06
- No. of Iterations taken to converge-189 (may vary from approx. 160-190,because of randomness in train test split every time we run.)

c.



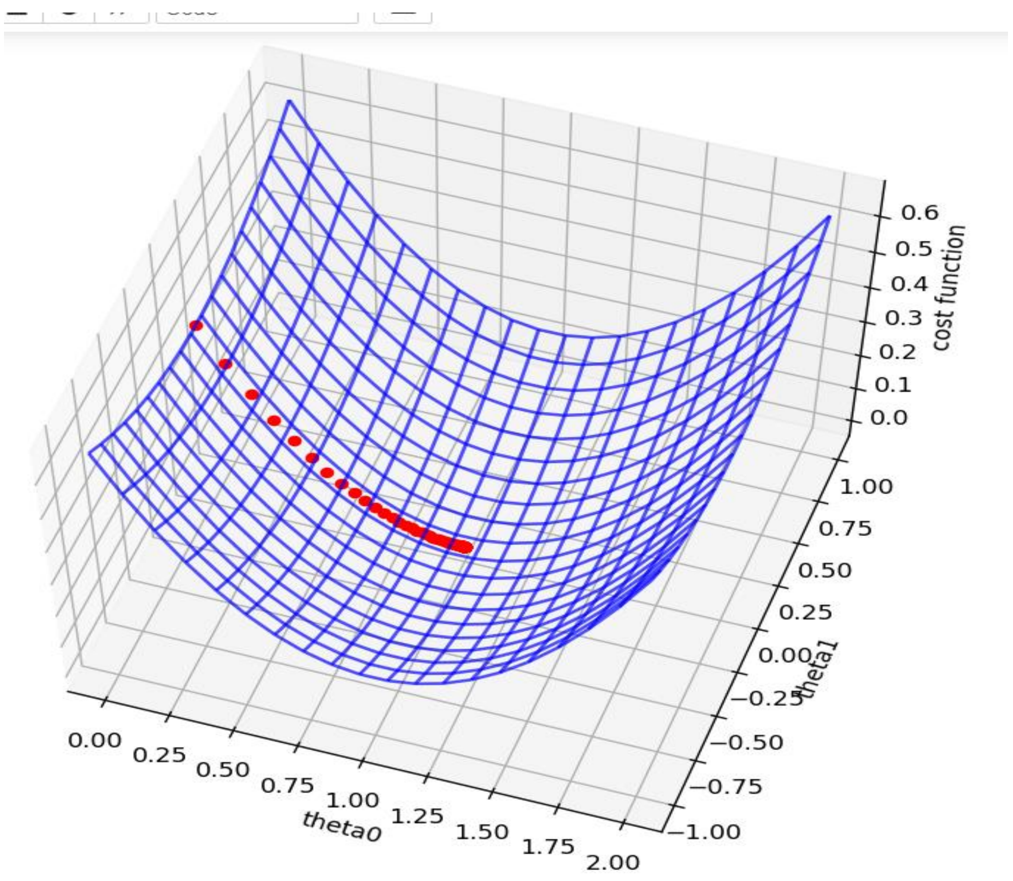
- Cost function decreases on every iteration with learning rate-0.1



d.

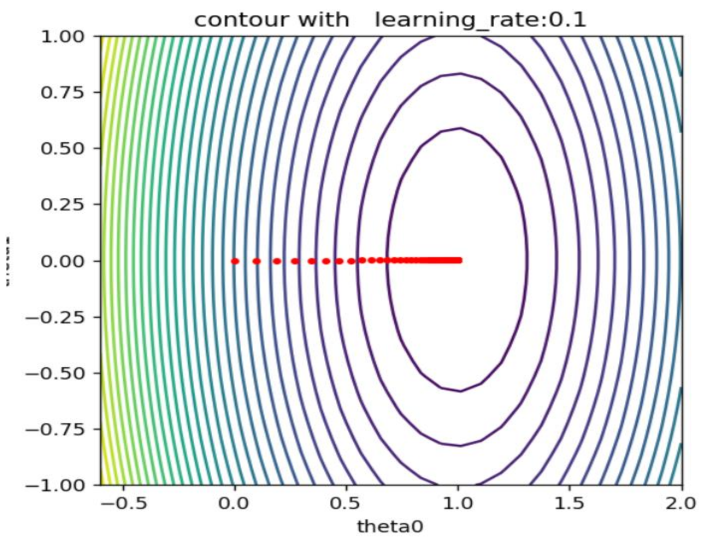
3D PLOT showing error function on Z-axis and parameters(theta0 and theta1) on x,y axis

```
iter-0,cost-0.49673886019746843
iter-20,cost-0.007343827942355857
iter-40,cost-0.00010991608518158756
iter-60,cost-2.9243412393729155e-06
iter-80,cost-1.3219162013609792e-06
iter-100,cost-1.2917634978288578e-06
iter-120,cost-1.2893253085275255e-06
iter-140,cost-1.2886754142259706e-06
iter-160,cost-1.2884766901455897e-06
iter-180,cost-1.2884154889018693e-06
```



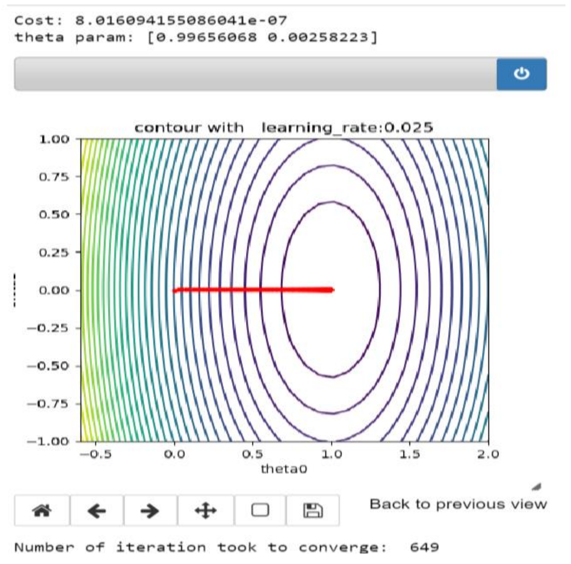
Video Link:
https://drive.google.com/file/d/1VMqbyadPToxvy1k5O1NYOEIcIRDtNo3G/view?usp=sharing

e.



Video Link:
https://drive.google.com/file/d/1NP1wLNqMGiUSC2UR5xbBR6q_8S_Yuw9I/view?usp=sharing

f. Learning rate=0.025



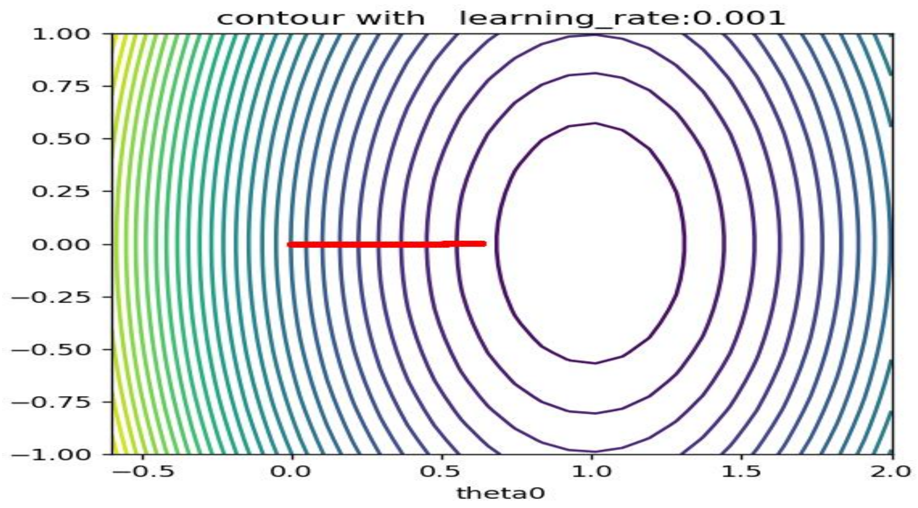
```
Cost: 8.016094155086041e-07
theta param: [0.99656068 0.00258223]
```

```
Number of iteration took to converge:  649
```

Video link:
https://drive.google.com/file/d/1mV7Vf-eDdj8hbpUPIDyl60xp03NFPmsL/view?usp=sharing

Learning rate=0.001

Number of iterations taken to converge:1000



Video Link:
https://drive.google.com/file/d/1-RWTrV-yD3CjH9AnaS0kTCx5yjr93rMb/view?usp=sharing

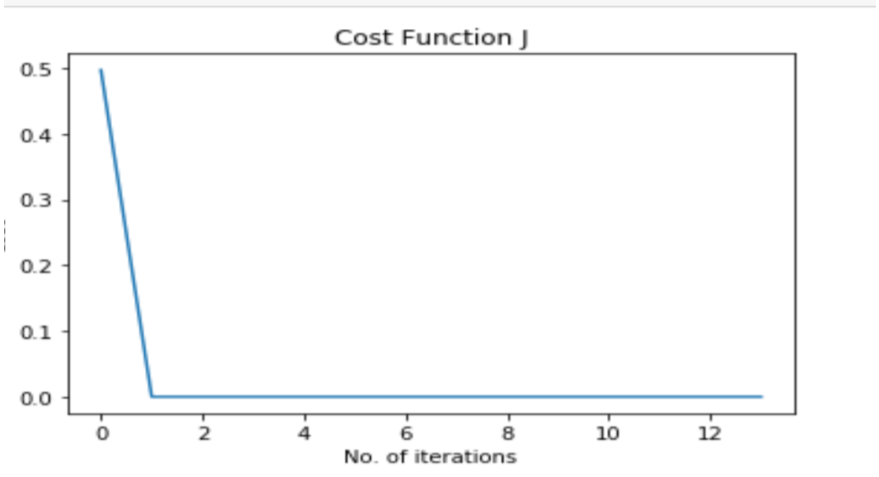
From above observations we can conclude that as the learning rate value increases, algorithm takes smaller steps, hence consumes less number of iterations to converge, and when we decrease the learning rate value algorithm takes smaller steps, hence consumes more time and more number of iterations to converge. If the learning rate value is very high. Cost function overshoots and may diverge.

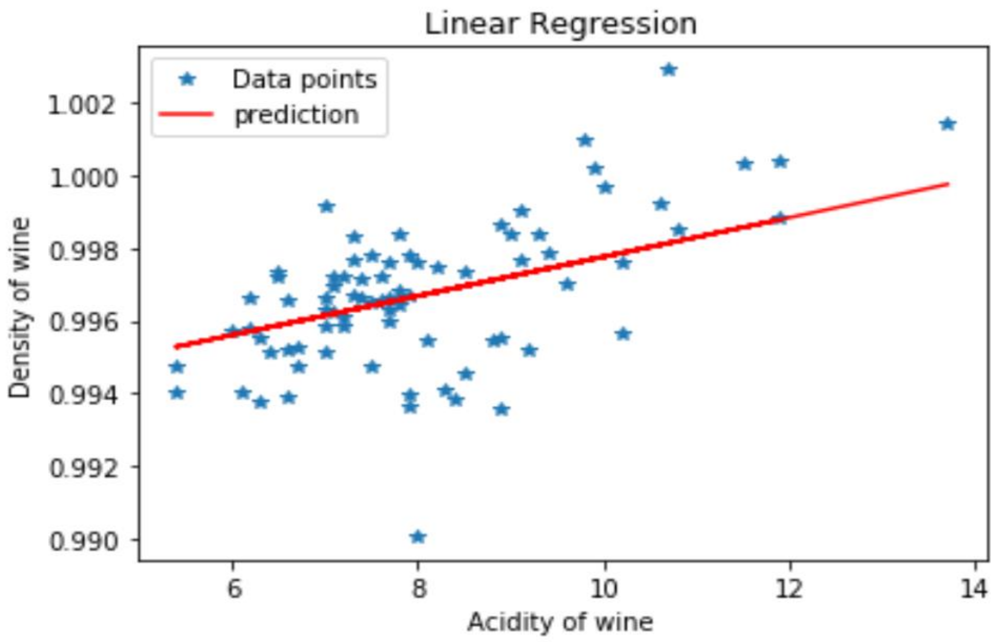
b. Batch Gradient Descent

Algorithm:

```
Step 0: Initialize weights(here theta0 and theta1- Zero vector).
Step 1: Choose a value of learning rate and gamma(for convergence).
Step 2: Normalize the data to have 0 mean and variance 1.
Step 3: do    {
Step 4:             $\nabla\theta J(\theta)=(h\theta(x^i)-y^i)(x^i)$
Step 5:             learning rate'=learning rate/√iteration number
Step 6:             $\theta^{t+1}=\theta^t$ - learning rate'*$\nabla\theta J(\theta)$
Step 7: while{ ( J(θ+1)-J(θ) ) >gamma (till convergence)
```

- theta vector initialized to a zero vector-[0,0]
- initial learning rate=1
- Final learning rate=1.2672428274337011e-05
- Stopping Criteria: The difference in Cost function value i.e., J(θ+1)-J(θ) becomes less than gamma(0.000000000001)
- Final Parameters: [0.99643696, 0.00111551]
- Final Cost value:1.2922597344368284e-06
- No. of Iterations taken to converge-15(may vary, because of randomness in train test split every time we run.)




Linear Regression
Data points
prediction
Density of wine
Acidity of wine
1.002
1.000
0.998
0.996
0.994
0.992
0.990
6
8
10
12
14

Question 2: Sampling and Stochastic Gradient Descent

Algorithm for Stochastic Gradient Descent:

```
Step 0:  t= 0
Step 1: theta[0] = some initial value
Step 2: do{
Step 3:      for (b = 1; b= m/r; b++) {
Step 4:          $\theta(t+1)$ =$\theta(t)$– eta * $\nabla\theta Jb(\theta)|\theta = \theta t$
Step 5:              $\theta(t)$ = $\theta(t+1)$
                }
          } while($\theta(t+1)$ ! $= \theta(t)$
```

(a.)

1 million data points taking values of θ = [θ0, θ1, θ2]T = [3 1 2]T, and x1 ∼ N (3, 4) and x2 ∼ N (−1, 4) independently, and noise variance in y is given by, σ2 = 2 .

The sampled data is stored in a csv file ‘sample.csv’.

Method followed to sample data-

Generate some random data.

new_x1 = random_x1 - mean / var. (where mean = 3, var = 4)

new_x2 = random_x2 - mean / var. (where mean = -1, var = 4)

new_var = random_var - mean / var. (where mean = 0, var = 2)

Y = $\theta 0 * X0 + \ \theta 1 * X1 + \ \theta 2 * X2 + \epsilon$

(b.)

η = 0.001

∀j θj = 0

Relearned θ values after implementing stochastic gradient descent.

(Convergence criteria used – cost[-1] – cost[-2] > 0.0000001)

Batch Size	Theta0 (actual)	Theta0 (learned)	Theta1 (actual)	Theta1 (learned)	Theta2 (actual)	Theta2 (learned)
1	3	2.996701	1	1.000356	2	1.997837
100	3	2.995109	1	1.000259	2	1.999652
10000	3	2.852971	1	1.016804	2	1.995277
1000000	3	2.837813	1	1.016915	2	1.995589

(c.)

Relearned parameters are almost equal to the original parameters.
Observations made:

i. If the stopping condition for the convergence had been determined by some maximum number of iterations, and the maximum iterations was set to be something lower than 10,000 the error increases for batch size 1 and 100. (Iterations made for the above mentioned relearned parameter values for batch size of 1 is 13400 and for batch size of 100 is 10000).
ii. If the stopping condition had been determined by some less difference, there was a considerable difference between the learned and original parameter values.

No. of iterations to converge:

Batch Size	No. of iterations
1	13400
100	10000
10000	4300
1000000	2850

Error comparison with provided dataset.
Note: We have calculated error on the data generated in part a. of the question, on the data provided (q2test.csv) and also on a randomly generated data of the same sample space. (This was done since there was some error with the X_1

values of q2test.csv and to have clear idea of how accurately the model performs on a dataset from exactly same sample space).

i. Batch Size = 1
   Theta0 = 2.996701
   Theta1= 1.000356
   Theta2= 1.997837

   Error in the data generated in part a. with the actual parameter values (3, 1, 2): **0.0**
   Error in the data generated in part a. with the relearned parameter values: **3.845701433195385e-05**

   Error in the dataset(q2test.csv) with actual parameter values(3, 1, 2):**0.98294692150000091**
   Error in the dataset (q2test.csv) with relearned parameter values: **0.9834850777115355**

   Error in the test data randomly generated (from the same sample space) with actual parameter values (3, 1, 2): **0.0**
   Error in the test data randomly generated (from the same sample space) with relearned parameter values: **3.314334242554118e-05**

ii. Batch Size = 100
   Theta0 = 2.995109
   Theta1= 1.000259
   Theta2= 1.999652

   Error in the data generated in part a. with the actual parameter values (3, 1, 2): **0.0**
   Error in the data generated in part a. with the relearned parameter values: **5.953329183427096e-05**

Error in the dataset (q2test.csv) with actual parameter values (3, 1, 2): **0.9829469215000091**
Error in the dataset (q2test.csv) with relearned parameter values: **0.9832002501349189**

Error in the test data randomly generated (from the same sample space) with actual parameter values (3, 1, 2): **0.0**
Error in the test data randomly generated (from the same sample space) with relearned parameter values: **6.241353319685452e-05**

iii. Batch Size = 10000
Theta0 = 2.852971
Theta1= 1.016804
Theta2= 1.995277

Error in the data generated in part a. with the actual parameter values (3, 1, 2): **0.0**
Error in the data generated in part a. with the relearned parameter values: **0.006659555397086677**

Error in the dataset (q2test.csv) with actual parameter values (3, 1, 2): **0.9829469215000091**
Error in the dataset (q2test.csv) with relearned parameter values: **1.0062016288972666**

Error in the test data randomly generated (from the same sample space) with actual parameter values (3, 1, 2): **0.0**
Error in the test data randomly generated (from the same sample space) with relearned parameter values: **0.007170300243816281**

iv. Batch Size = 1000000

Theta0 = 2.837813
Theta1= 1.016915
Theta2= 1.995589

Error in the data generated in part a. with the actual parameter values (3, 1, 2): **0.0**
Error in the data generated in part a. with the relearned parameter values: **0.008159287733110282**

Error in the dataset (q2test.csv) with actual parameter values (3, 1, 2): **0.9829469215000091**
Error in the dataset (q2test.csv) with relearned parameter values: **1.0082813009072078**

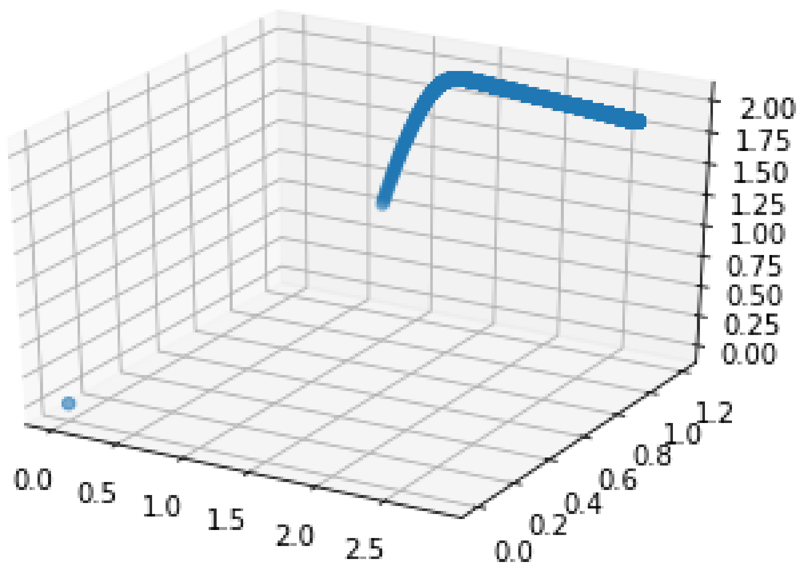
Error in the test data randomly generated (from the same sample space) with actual parameter values (3, 1, 2): **0.0**
Error in the test data randomly generated (from the same sample space) with relearned parameter values: **0.008311681517028265**

From the above results we can infer,

i. The error in the data with actual parameter values and relearned parameter values is much less.
ii. There is no over fitting, since the parameters work considerably well on test data.
iii. The error has increased with the increase batch sizes, which can be rectified by reducing the stopping condition (which in our case was set to be 0.000001, reducing this value would reduce the error in the higher batch sizes).

(d.)

Plot for parameter values for batch size = 10000



The values of parameters initially set to 0, 1; the shape of curve infers:

i. The parameter values change on a very small scale for each iteration, ensuring that the minimum values are not dropped.

ii. Less oscillation taken towards the minima of the loss function due to updating the parameters by computing the average of all the training samples rather than the value of a single sample.

Video Link:
https://drive.google.com/file/d/1voawxapJuN3vkyLLlY0XdNw1Ou_HnsUh/view?usp=sharing

Individual Contributions:

Sampling, Stochastic Gradient descent, Plots, Report-	Saloni Shah
Batch Gradient Descent,Adaptive lr,Plots,Report-	Amogha T S

[Ps:We have helped each other in debugging, planning and other tasks.