COL774:A1

Sidharth Agarwal 2019CS50661

September 10, 2022

1: Linear Regression

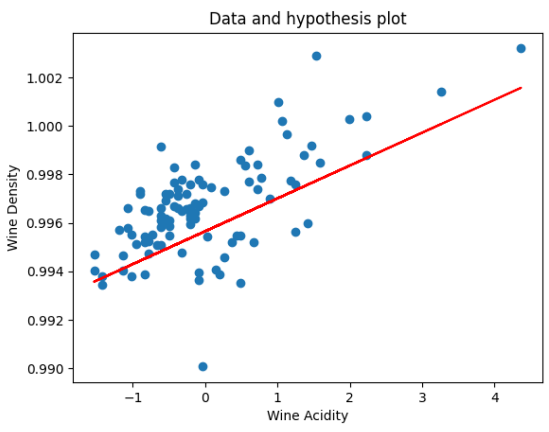
Here I first normalized the data and then applied linear regression least squared error metric and applying gradient descent on the cost function. I checked the convergence of the cost function when the difference between new cost and previous cost is less than some epsilon. I added the intercept dimension as the last feature in the feature matrix.

1.a

As described above the stopping criteria, I used the value of epsilon to be 1e-8 and the value of learning rate to be 0.01. The parameters learned by the algorithm were theta0 = 0.0013566 and theta1 = 0.995565, where theta1 is here the intercept. The final loss before convergence was 1.6702e-06.

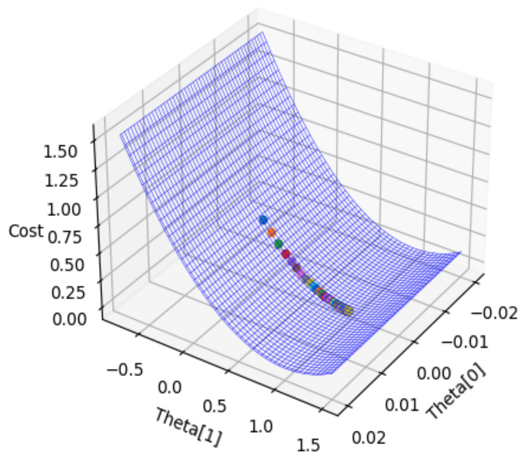
1.b

The equation of line obtained as per the parameters given above, turns out to be y = 0.00135x + 0.99556.



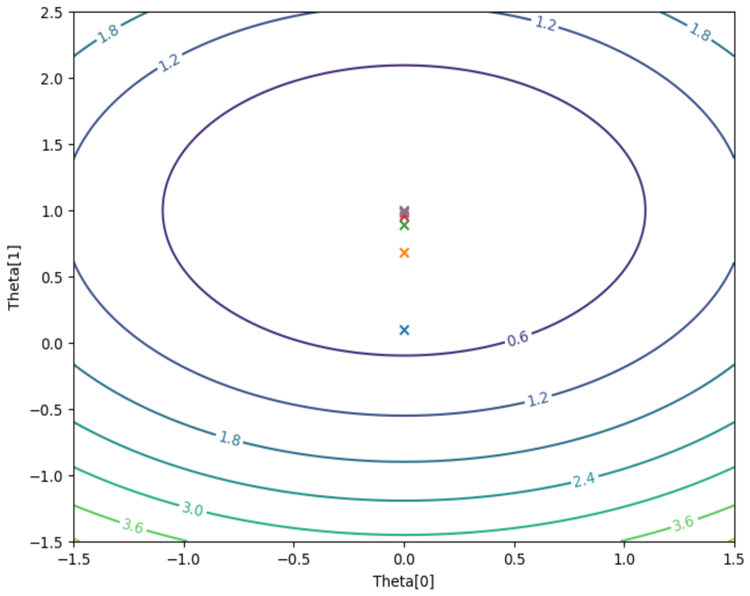
1.c

The final 3d plot obtained is as follows..

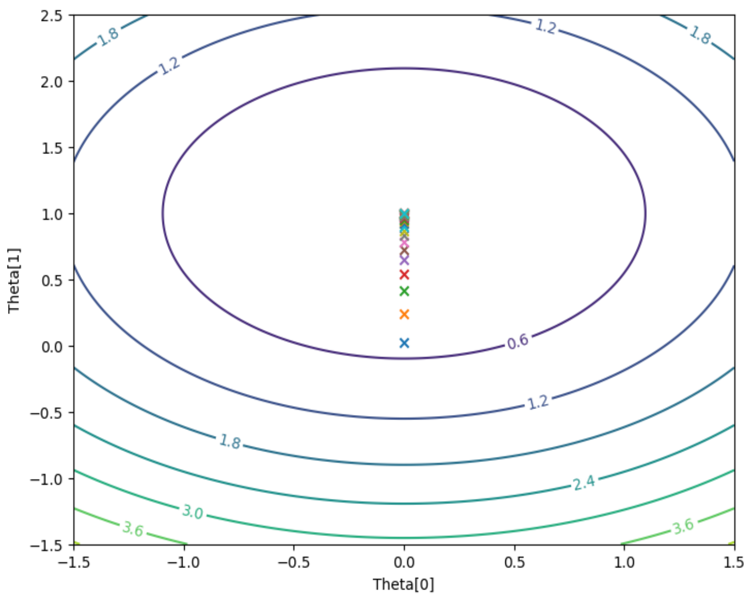


1.d and 1.e

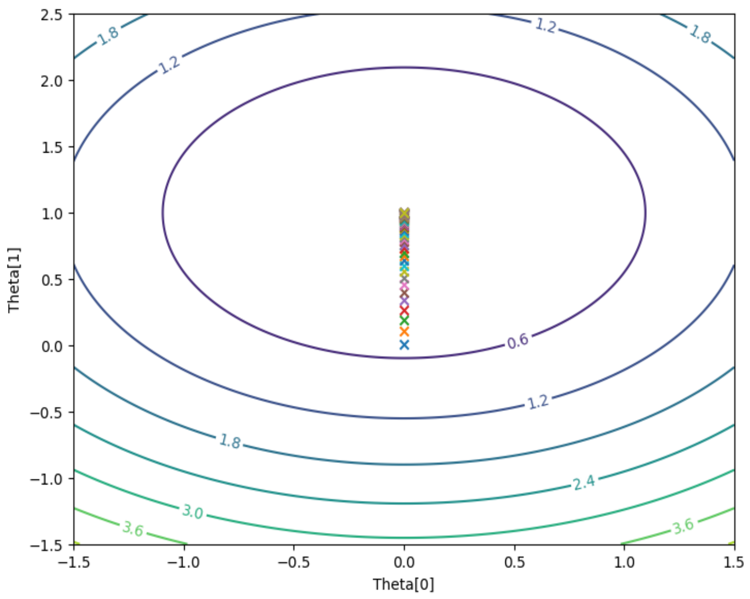
The plot for learning rate = 0.1



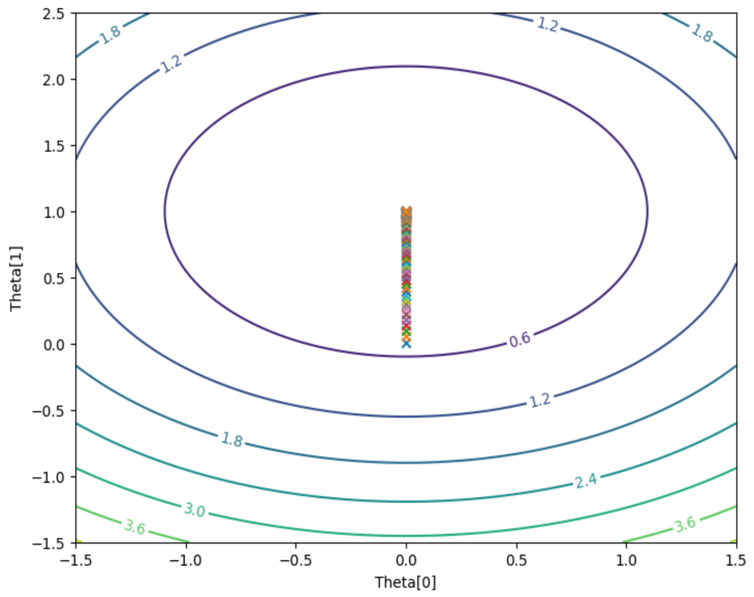
The plot for learning rate = 0.025



The plot for learning rate = 0.01



The plot for learning rate = 0.001



It is clearly observable as we increase the learning rate the convergence becomes much faster. Now since the epsilon is same for all the 4 cases, the final cost is almost same for all the cases. There maybe data for which large learning rate doesn't lead to any convergence.

2

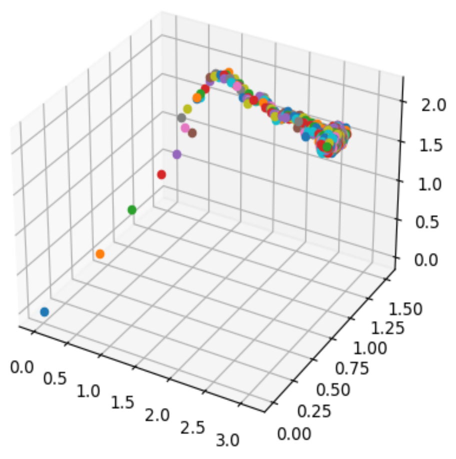
For this problem we took learning rate to be 0.001 and the value of epsilon to be also 0.001. For convergence I defined one iteration as the training on one batch. Now for convergence the difference in loss function must be less than epsilon as well as their must have been at least (totalSamples/batchSize) iterations, so as to cover the whole data at least once. Now to check the difference in loss, average loss across a totla 1000 iterations is taken into consideration.

Batch Size	Theta	Time(in s)	Iterations	Train loss	Test loss
1	(3.04544919,1.0122903,1.97678753)	69.45s	1039001	1.0675	1.4523
100	(2.97697054,1.003075,1.99865383)	0.8616s	18001	1.00246	0.983
10000	(2.9219189,1.01707437,1.99326347)	3.70717	13001	1.00058	1.00150
1000000	(2.92047172,1.01670439,1.99496389)	255.49	13001	1.0013	0.982

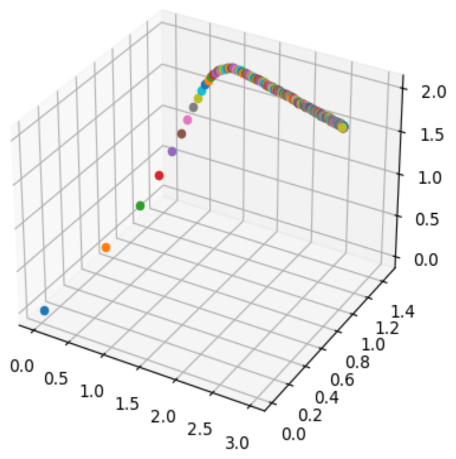
Error of new data on the original hypothesis is 0.9829. We can observer that all the three cases have very close values of theta. The number of iterations are seen to be increasing as we decrease the batch size, this is because we are taking less randomly directed and useless paths for convergence. The train loss and test loss is high in batch size of 1, this is because even a small noise can take us away from the path of convergence, other batch sizes perform equally decent in terms of convergence. Now the overall time is a combination of both computation per iteration and number of iterations taken at each. That is why don't observe a monotonic pattern in time, since computation per iteration increases with batch size, whereas total number iterations taken decrease with increasing batch size.

Now for the shape of movement as observed in the graphs below, we can see that as batch size increases, the movement is less random and more directed towards the final goal.

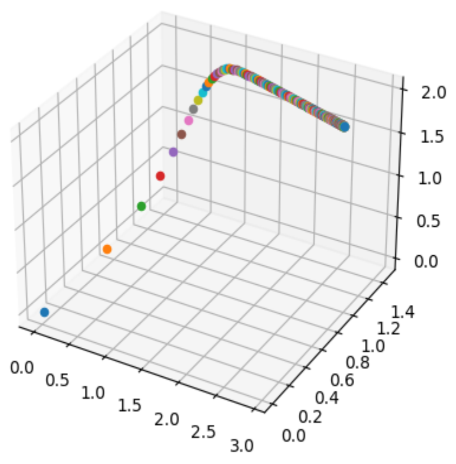
For batch size = 1



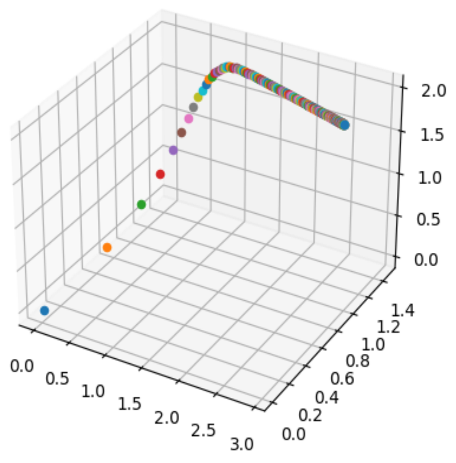
For batch size = 100



For batch size = 10000

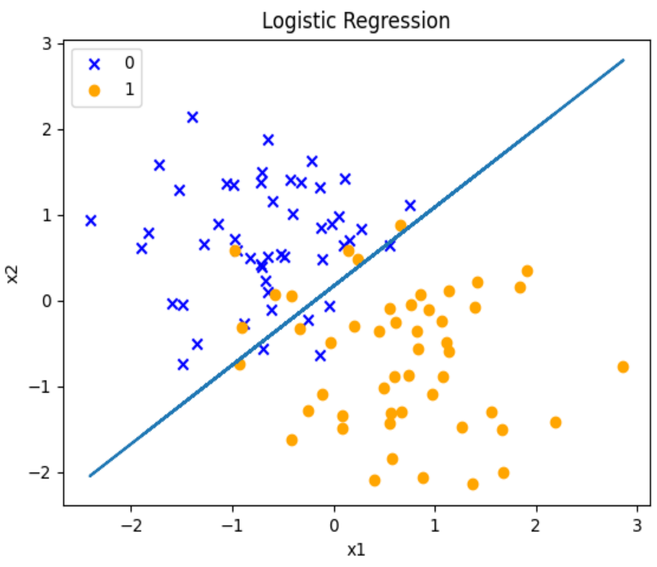


For batch size = 1M



3: Logistic regression

The final value of theta(theta1,theta2,theta0) obtained is (2.51178627,-2.72999772,0.44962448), where 0.44962 is the intercept theta. I used the condition of convergence when the absolute difference between updated theta and previous theta was is less than epsilon where I took epsilon to be 0.0001. The final equation of line comes out to be y = 0.9514x+ 0.147.



4: GDA

The equation of decision boundary can be found by equating the probabilities of having any of the class given the data are equal. Now simplifying these equations we get.

The equation of linear boundary is as follows:

$$x^T\Sigma^{-1}(\mu_1-\mu_0)+ln(\frac{phi}{1-phi})-(\mu_1^T\Sigma^{-1}\mu_1-\mu_0^T\Sigma^{-1}\mu_0)/2=0$$

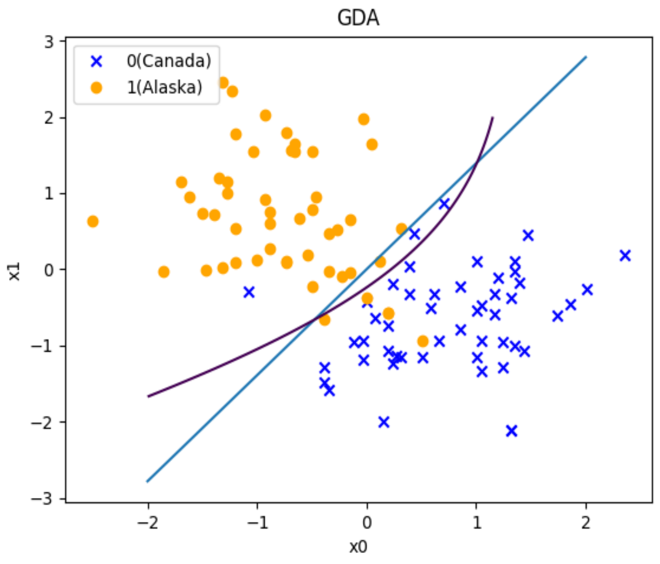
The equation of the quadratic boundary is as follows:

$$ln(\frac{phi}{1-phi})-(1/2)ln(\frac{|\Sigma_1|}{|\Sigma_0|})+(1/2)((x-\mu_0)^T\Sigma_0(x-\mu_0)-(x-\mu_1)^T\Sigma_1(x-\mu_1))=0$$

The values of various constants obtained after taking Alaska as class 1 and Canada as class 0 are as follows:

$$\phi = 0.5$$
$$\mu_0 = (0.75529433, -0.68509431)$$
$$\mu_1 = (-0.75529433, 0.68509431)$$
$$\Sigma = ((0.42953048, -0.02247228), (-0.02247228, 0.53064579))$$
$$\Sigma_0 = ((0.47747117, 0.1099206), (0.1099206, 0.41355441))$$
$$\sigma_1 = ((0.38158978, -0.15486516), (-0.15486516, 0.64773717))$$

The plot of data, linear GDA and quadratic GDA are as follows.



Here we can clearly see that the quadratic classifier seems to perform better on the training data and atleast from the data given also doesn't seem to overfit the data. So I believe that quadratic classifer seems to be a better choice overall.