COL774 – Machine Learning

Assignment – 1

Report

Submitted By:
Siddharth Gupta
Entry No. - 2021EE10627

Submitted To:
Parag Singla

Indian Institute of Technology, Delhi

Hauz Khas, New Delhi

Question – 1:

1. **Overview:**
   - **Algorithm – Linear Regression**
   - **Optimization Method – Batch Gradient Descent**

2. **Assignment Questions:**

a) There are two hyperparameters in the algorithm – learning rate and stopping criteria.

As stopping criteria, I am using the following rule:

$$\left| J\left(\theta^{(t+1)}\right) - J\left(\theta^{(t)}\right) \right| \leq \delta$$

Given the small size of the training dataset, the algorithm converges fairly fast even for $\delta = 0$.

However, to ensure speed for a possibly larger training dataset, I fix $\delta = 10^{-6}$

As for the learning rate, I tried with a few values of η .

Learning Rate	Allowed Error	Number of Iterations	Final Cost Function	Comment
0.01	0	1557	0.285445298	Most Accurate
0.01	1e-6	417	0.285495419	Too Slow
0.1	1e-6	52	0.285449914	Fairly Fast and More Accurate
1	1e-6	2	0.285445298	Too Fast and Risk of Not Converging

Keeping in mind to reduce the number of iterations as well as have good accuracy, I fix $\eta = 0.1$

With these parameters, the final values of parameters:

$$\left| J\left(\theta^{(t+1)}\right) - J\left(\theta^{(t)}\right) \right| \leq 10^{-6}$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - 0.1 \times \nabla_{\theta} J(\theta)|_{\theta^{(t)}}$$

$$\#\, iterations = 52$$

$$\Theta^{*} = \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 0.655064426 \\ -5.99354594 \times 10^{-14} \end{bmatrix}$$

$$J(\Theta^{*}) = 0.2854499147496327$$

b) I have used Pandas to draw a scatter plot of the input data and the learnt hypothesis function on the same plot.

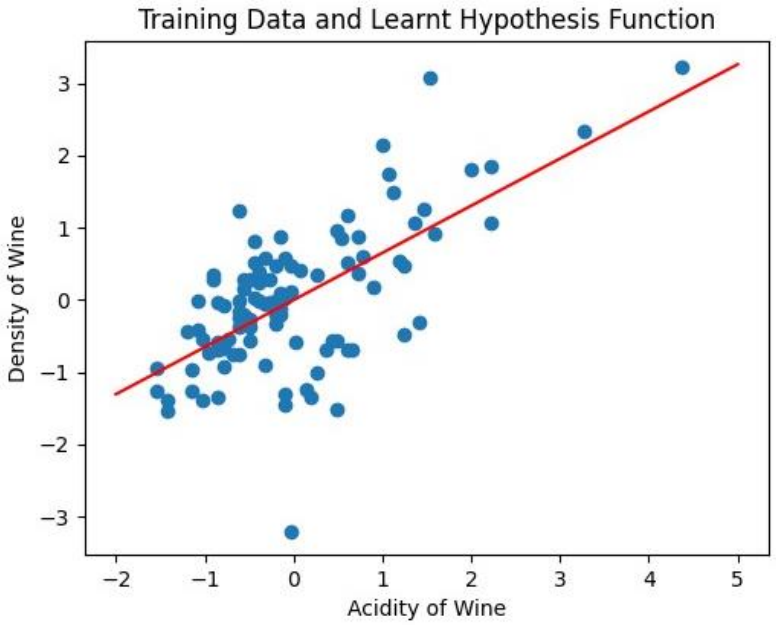


Fig: Scatter Plot of Input Data and Hypothesis Function

c) I draw a mesh plot of the cost function as a function of theta parameters and plot the values of the cost function at each iteration on the same mesh.
For better understanding, I have different the scales, elevation, and azimuth for viewing the mesh and observing the animation.

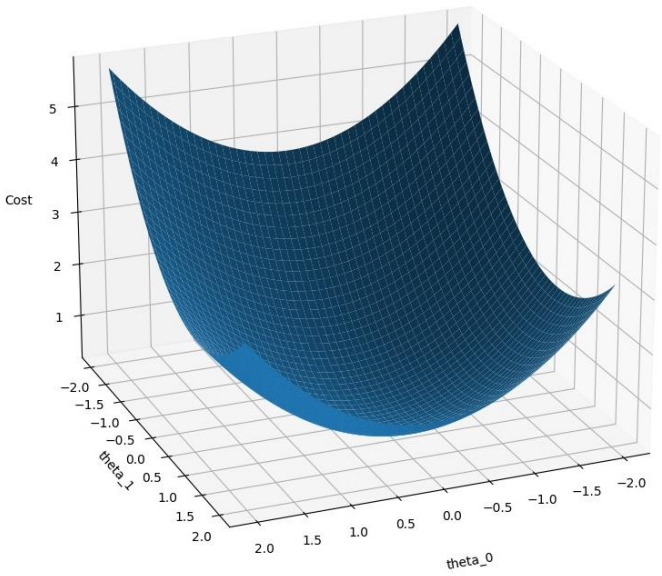


Fig: 3-D Mesh Plot of the Cost Function

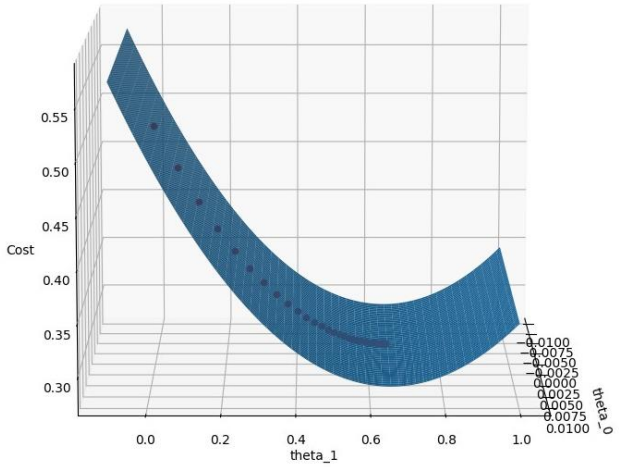


Fig: Observing the decrease in cost value with progress of algorithm

d) I draw a contour plot of the cost function as a function of theta parameters. The lines in the plot correspond to 50 values of $J(\theta)$ equally spaced between 1 and 50.

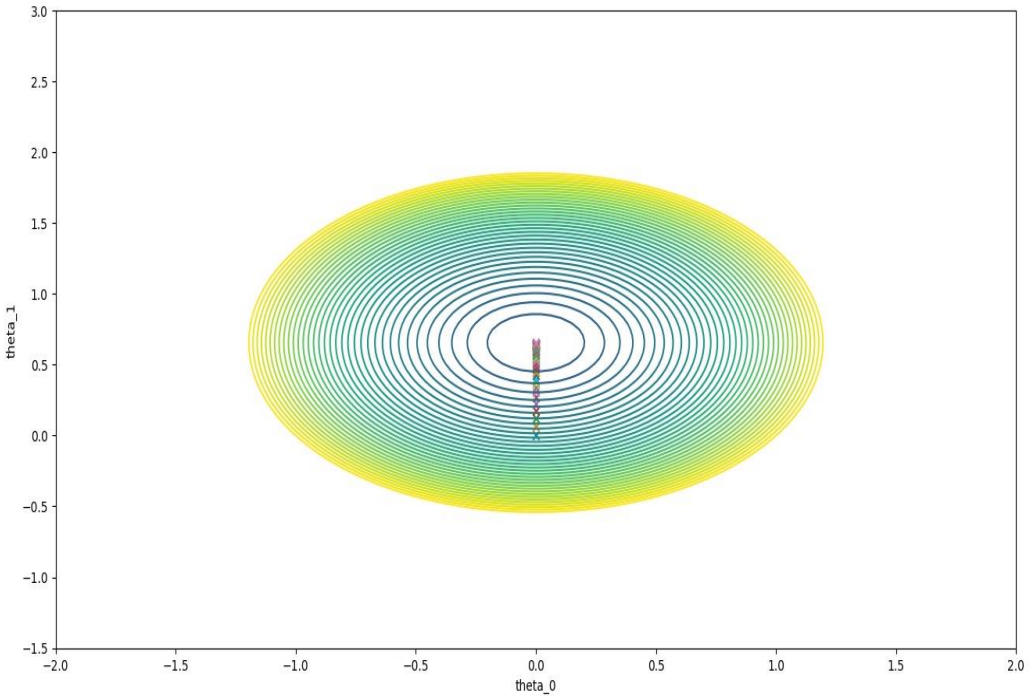


Fig: Contour plot of the cost function and movement of the theta as the algorithm progresses

e) We run the algorithm for $\eta = 0.001, 0.025, 0.1$. As we decrease the value of η, the number of iterations increases (too long for $\eta = 0.001$) Therefore, the lines marking the change in theta parameters becomes more and more continuous and the animation consequently takes more time.

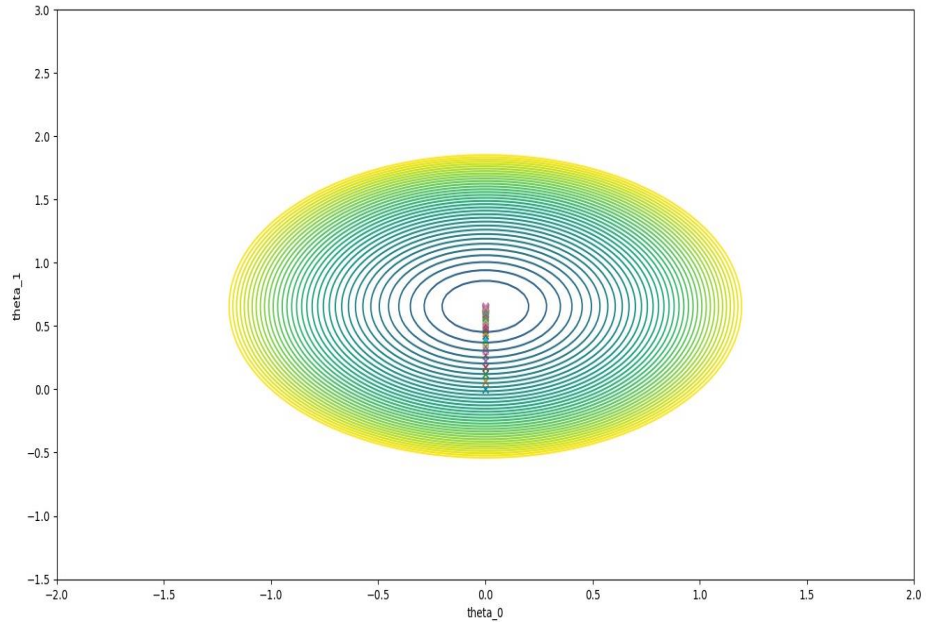
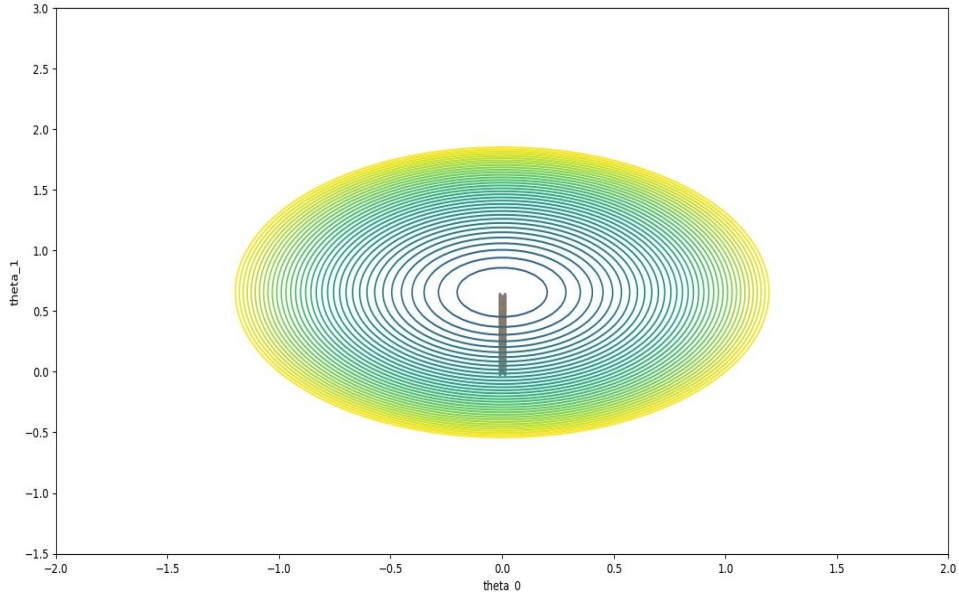
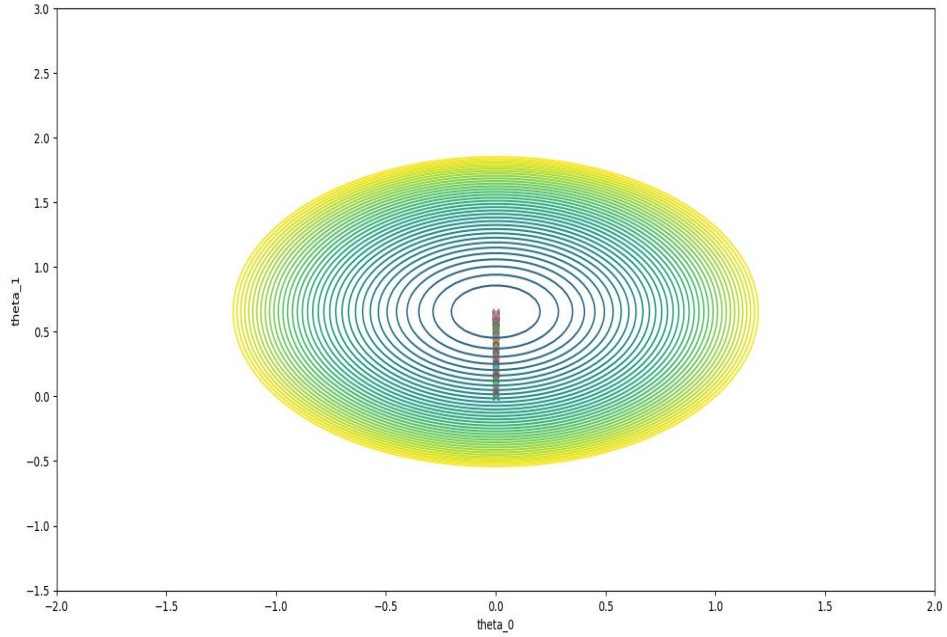


Fig: $\eta = 0.1$



$Fig: \eta = 0.001$



$Fig: \eta = 0.025$

For the given training dataset, we observe that movement of theta parameters on the contour plot is a straight line and always towards the minima. This implies that the direction of gradient is always towards the global minima.

This may not be generally true. In such cases, though the movement will always be inwards, we final plot would be zig-zag line. For a smaller learning rate, this zig-zag pattern will be less noticeable as we take very small steps at each iteration.

Question – 2:

1. **Overview:**
   - **Algorithm – Linear Regression**
   - **Optimization Method – Stochastic Gradient Descent**

2. **Assignment Questions:**

   a) The training dataset is sampled using $np.random.normal(loc,\ scale)$, where loc is the mean of the Gaussian distribution and $scale$ is the standard deviation of the distribution.

   There are several parameters in SGD. The learning rate is fixed by the assignment question as *0.001.*

   Batch size is another hyperparameter, also fixed in the assignment.

   Another hyperparameter is allowed error which decides upon the convergence of the algorithm. I have varied this for various batch sizes.

   b) The stopping criteria used in the algorithm is:

$$\left|\boldsymbol{J}(\boldsymbol{\theta}^{(t+1)}) - \boldsymbol{J}(\boldsymbol{\theta}^{(t)})\right| \leq \boldsymbol{\delta}\ \boldsymbol{or}\ \#\ \boldsymbol{iteration} > \mathbf{100000}$$

   The limit on number of iterations is especially included for the case when batch size is 1. Here, I observed that the theta parameters were converging to $[3 \quad 1 \quad 2]^T$ fairly quickly. However, due to stochasticity in the algorithm, the algorithm was not terminating due to oscillations about the minima. Since, we have randomly shuffled the training dataset, leaving some training examples during the algorithm does not affect the result much.

   The allowed error is decided keeping in mind the need to time of training. The allowed error is minimum for batch size of 1 because this algorithm is taking the longest to terminate.

Batch Size	Allowed Error	Theta Parameters	# iterations
1	10^{-3}	$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}2.99126118\\ 0.98676954\\ 1.99971672\end{bmatrix}$	100001
100	10^{-5}	$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}3.00278755\\ 1.00101226\\ 2.00018735\end{bmatrix}$	40000
10000	10^{-6}	$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}2.99218055\\ 1.00117659\\ 1.99880495\end{bmatrix}$	20400
1000000	10^{-6}	$\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}2.88962911\\ 1.0234208\\ 1.9913588\end{bmatrix}$	11784

As expected, the number of iterations is maximum for largest batch size. However, this also takes the longest time (ignoring the oscillations in the first case). This is because, though each iteration takes the longest time, the jump taken during each iteration is the largest.

c) We observe that the cost function value over the entire training set for various values of batch sizes is approximately same and almost 0.
Similarly, the cost function value over the entire test set for various values of batch sizes is approximately same and equal to the cost w.r.t the original hypothesis.
The reason for difference in the cost of the training and test set is probably because of the noise in the training data. It can also be due to slight over-fitting of the model on the training data.

Cost on Test Dataset w.r.t original hypothesis	5.501423921500001

Batch Size	**Cost on Training Dataset**	**Cost on Test Dataset**
1	0.9998972997688351	5.556978721423168
100	0.9985188377565453	5.498421933286241
10000	0.998504365163609	5.493618162214144
1000000	1.0003275560762028	5.422300324408873

d) The plots of $\theta_0 \; vs \; \theta_1 \; vs \; \theta_2$ for each batch size is:

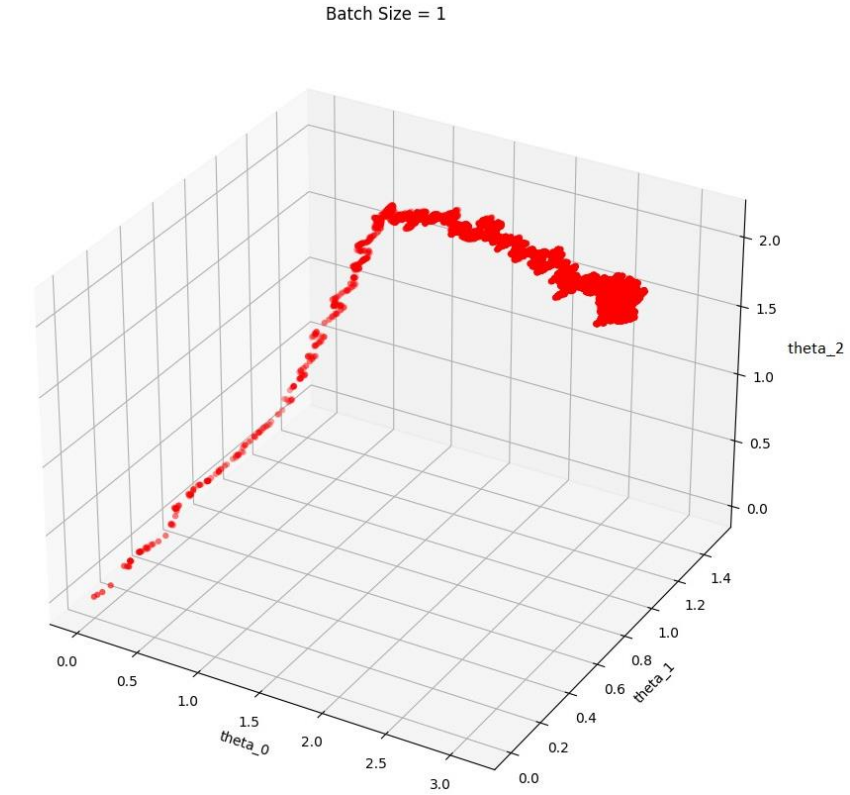


Fig: Batch Size = 1

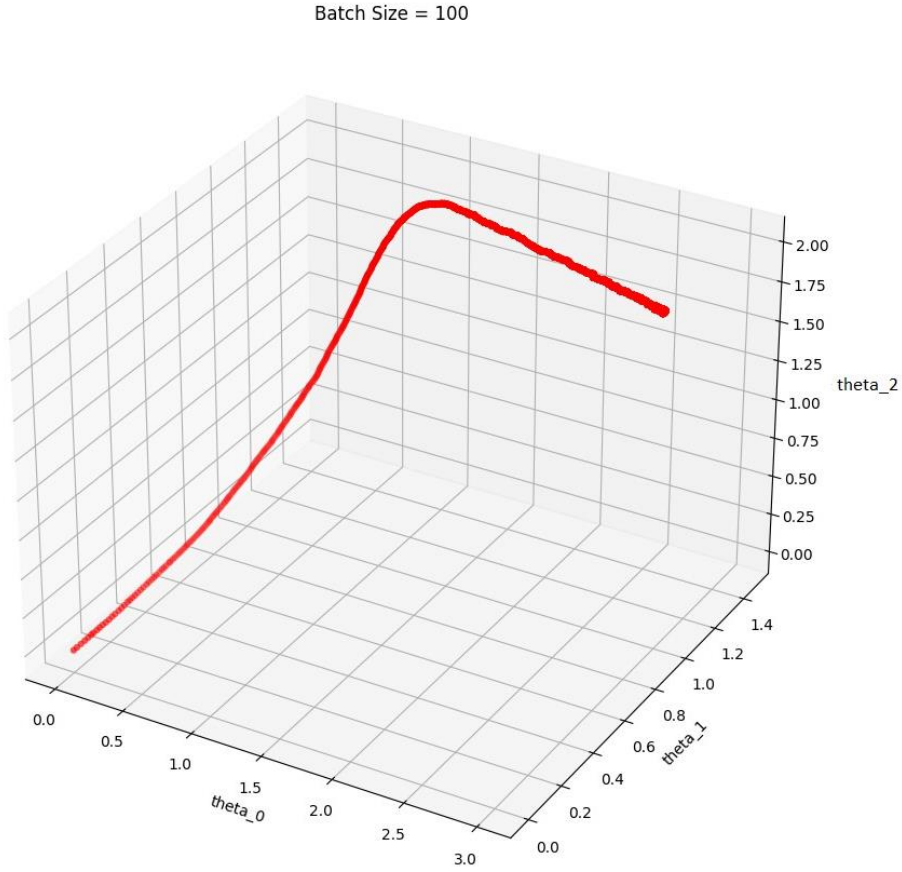


Fig: Batch Size = 100

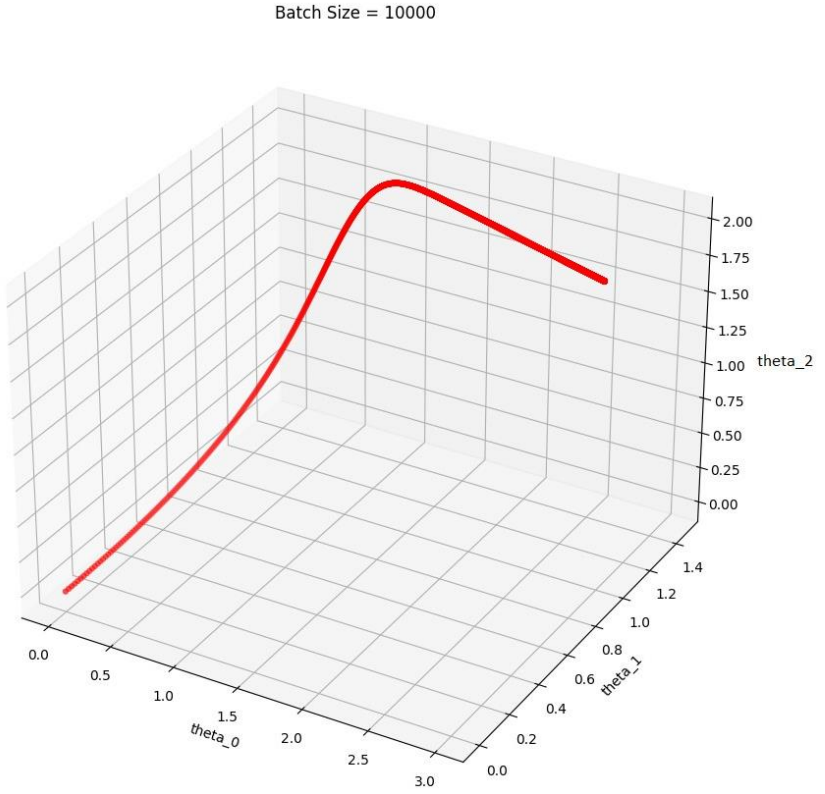


Fig: Batch Size = 10000

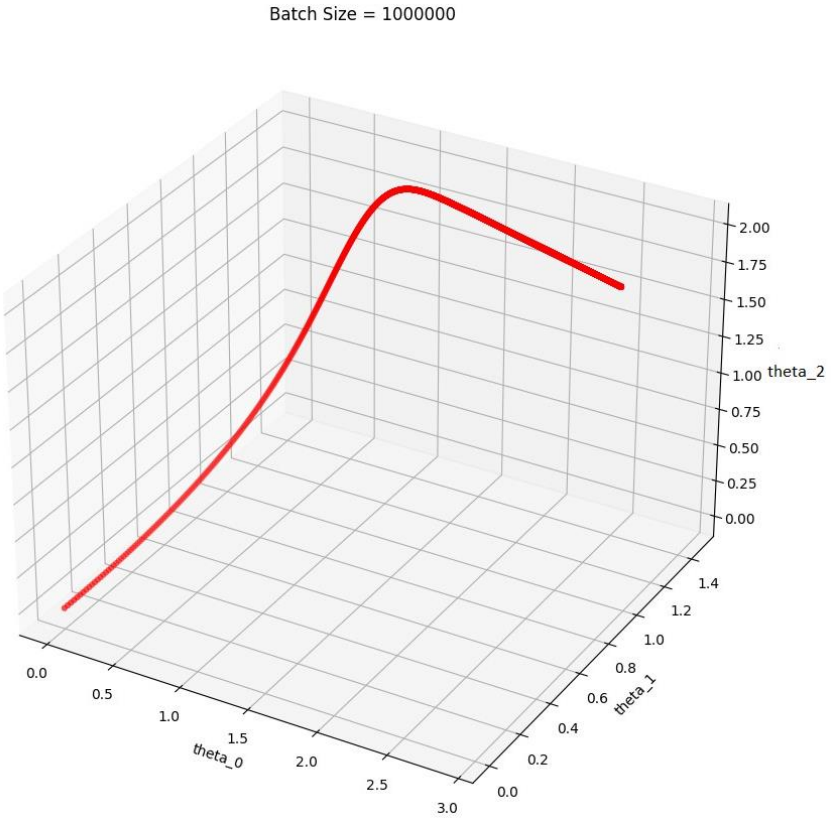


Fig: Batch Size = 1000000

We observe that for batch size of 1, there is a lot of noise in the variation of theta parameters. This is because we are computing the gradient with respect to only one sample in one iteration. This leads to stochasticity in the algorithm. As we increase the batch size, the curve becomes more and more smooth. Also, observing the scatter plots of the same data, we can see that the number of points plotted are maximum in case of batch size of 1. This also matches with expectation since the maximum number of iterations are in the case where batch size is 1.

Question – 3:

1. **Overview:**
   - **Algorithm – Logistic Regression**
   - **Optimization Method – Newton's Method**

2. **Assignment Questions:**

a) There is one hyperparameter in the algorithm – the stopping criteria. This is used to decide when to complete the algorithm.
As stopping criteria, I am using the following rule:

$$\left|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\right| \leq \boldsymbol{\delta}$$

Since Newton's Method is quite fast, I have fixed $\delta = 0$. Even with this condition, the algorithm is converging in less than 10 iterations which justifies the decision.

The update rule for Newton's Method is:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \mathcal{H}^{-1}.\nabla_\theta L(\theta)|_{\theta^{(t)}}$$

where, $L(\theta)$ is the log-likelihood given by:

$$L(\theta) = \frac{1}{m} \times \sum_{i=1}^{m} \{y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\}$$

where, $h_\theta\left(x^{(i)}\right)$ is the hypothesis function given by:

$$h_\theta\left(x^{(i)}\right) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

The $1/m$ in the log-likelihood is taken to average over all the training examples.

After training the model according to the above rules, the final values of the parameters are as follows:

$$\#\ iterations = 9$$

$$\Theta^* = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}0.40125316\\ 2.5885477\\ -2.72558849\end{bmatrix}$$

b) The following is the scatter plot of the input data along with the linear separator learnt by the model:

$\underline{Legend}$:

$Blue\ Circles := y = 0$

$Red\ Crosses := y = 1$

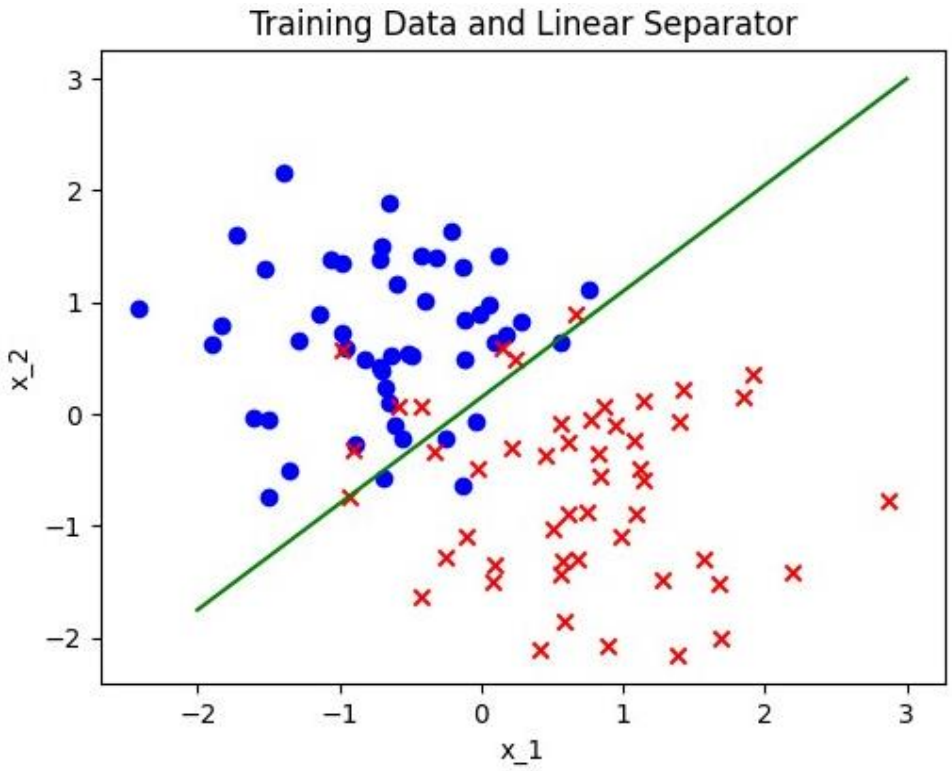


Fig: Scatter Plot of Training Data and Linear Separator Learnt

Question – 4:

1. **Overview:**
   - **Algorithm – Gaussian Discriminant Analysis**
   - **Optimization Method – Analytical Method**

2. **Assignment Questions:**

   a) Since the algorithm is analytical, there are no hyperparameters in this algorithm.

In this algorithm, I have assumed:

$$Alaska := 0$$
$$Canada := 1$$

We assume that the covariance matrix for each class is same. Therefore, the parameters are:

$$\phi\ ,\ \mu_0\ ,\ \mu_1\ ,\ \Sigma_0 = \Sigma_1 = \Sigma$$

I have calculated the parameters by iterating over the dataset and add to the respective parameters according to whether the output is 'Alaska' or 'Canada'.

The parameters come out to be:

$$\phi = 0.5$$

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

b) The scatter plot of the training data is:

$\underline{Legend}$:

$Blue\ Crosses := y = 0\ (Alaska)$

$Red\ Circles\ \ := y = 1\ (Canada)$

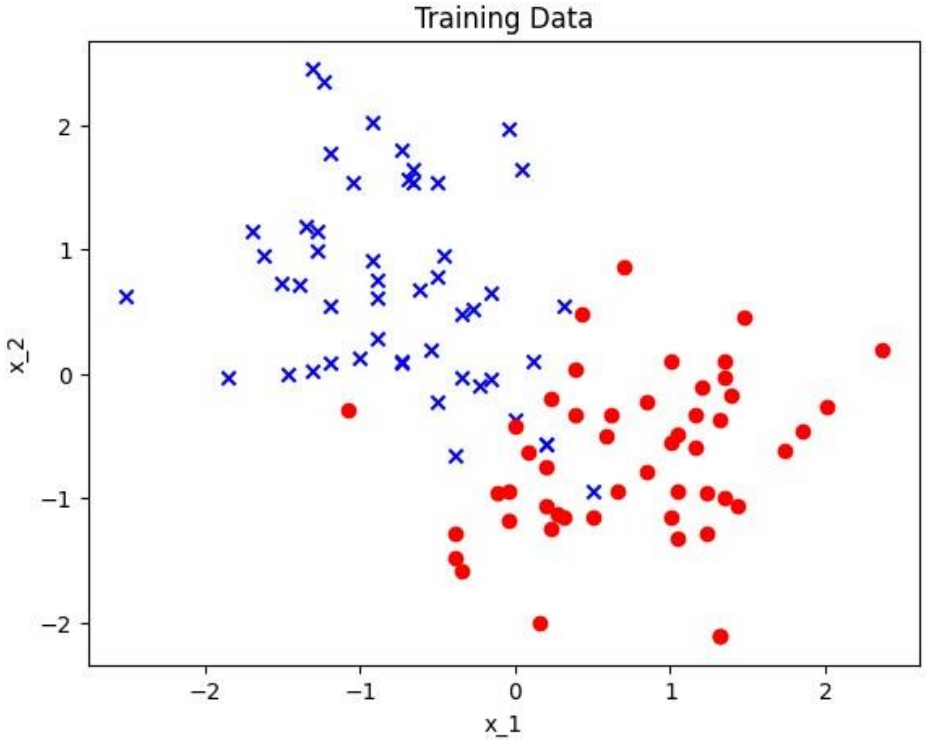


Fig: Scatter Plot of Training Data

c) In the case when the covariance matrix of both the classes are assumed to be same, the equation of the separator comes out to be:

$$-(\mu_1^T - \mu_0^T).\Sigma^{-1}.x + \frac{1}{2}.(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0) + \log\left(\frac{1-\phi}{\phi}\right) = 0$$

Substituting the matrices by variables,

$-(\mu_1^T - \mu_0^T).\Sigma^{-1} := [a \;\; b]$ and $\frac{1}{2}.(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0) + \log\left(\frac{1-\phi}{\phi}\right) := C$

Representing $'x'$ in the vector form and simplifying it to get x_2 in terms of x_1, we get:

$$x_2 = -\frac{a}{b}x_1 - \frac{C}{b}$$

After training the algorithm, we get the line as:

$$x_2 = 1.389845x_1 + 4.552736 \times 10^{-16}$$

This is approximately a line passing through origin. We can verify this with the plot obtained.

Legend:
Blue Crosses $:= y = 0$ *(Alaska)*
Red Circles $:= y = 1$ *(Canada)*

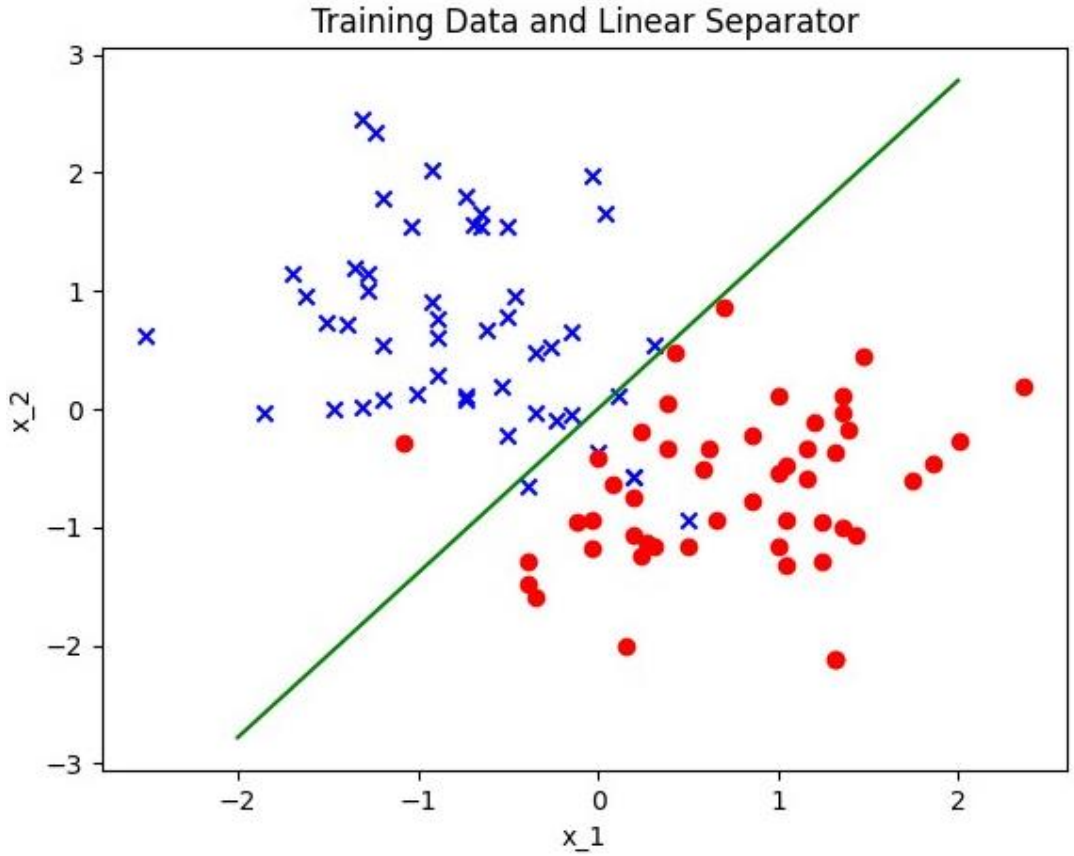


Fig: Scatter Plot of Training Data and Linear Separator

d) We now handle the general case where the covariance matrices of each class may be different. Under this condition, we again train the model. Now, the values of our parameters come out to be:

$$\phi = 0.5$$

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\Sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

e) In the general case when the covariance matrices of both the classes may be different, the equation of the separator comes out to be:

$$\frac{1}{2}.x^T.(\Sigma_1^{-1} - \Sigma_0^{-1}).x - (\mu_1^T.\Sigma_1^{-1} - \mu_0^T.\Sigma_0^{-1})x + \frac{1}{2}(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) + \log\left(\frac{1-\phi}{\phi} \times \frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}}\right) = 0$$

Substituting the matrices by variables,

$$\frac{1}{2}.(\Sigma_1^{-1} - \Sigma_0^{-1}) := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$(\mu_1^T.\Sigma_1^{-1} - \mu_0^T.\Sigma_0^{-1}) := [e \ \ f]$$

$$\frac{1}{2}(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) + \log\left(\frac{1-\phi}{\phi} \times \frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}}\right) := C$$

Representing $'x'$ in the vector form and simplifying it to get x_2 in terms of x_1, we get:

$$ax_1^2 + (b+c)x_1x_2 + dx_2^2 - ex_1 - fx_2 + C = 0$$

$$\Rightarrow dx_2^2 + \big((b+c)x_1 - f\big)x_2 + (ax_1^2 - ex_1 + C) = 0$$

After training the algorithm, we get the line as:

$$0.43296x_2^2 + (-1.28683x_1 - 2.85967)x_2 + (-0.33567x_1^2 - 3.80785x_1 + 0.58478) = 0$$

To plot the boundary, we take an array of x_1 values and find the corresponding x_2. From this, we get the following plots:

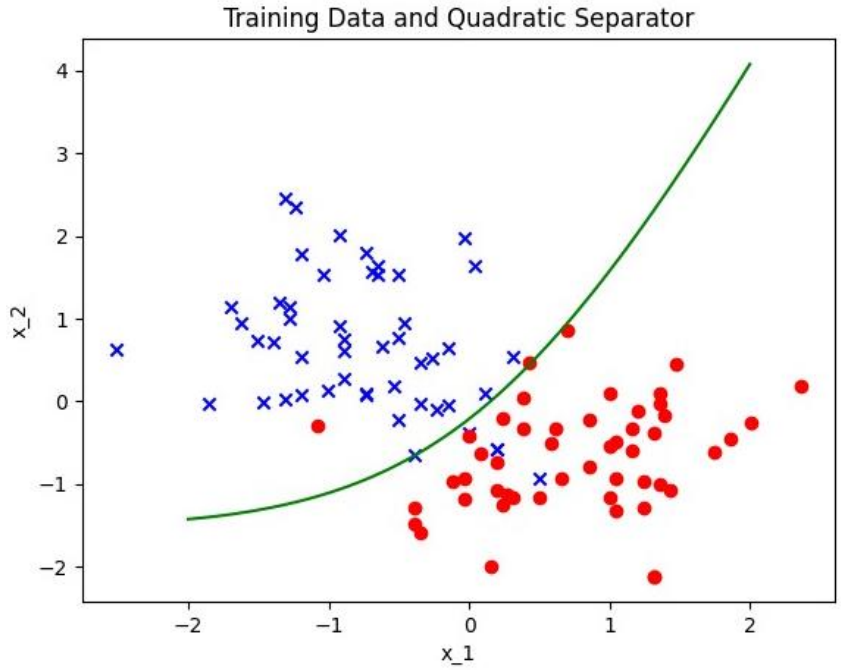


Fig: Quadratic Boundary Separating the Data

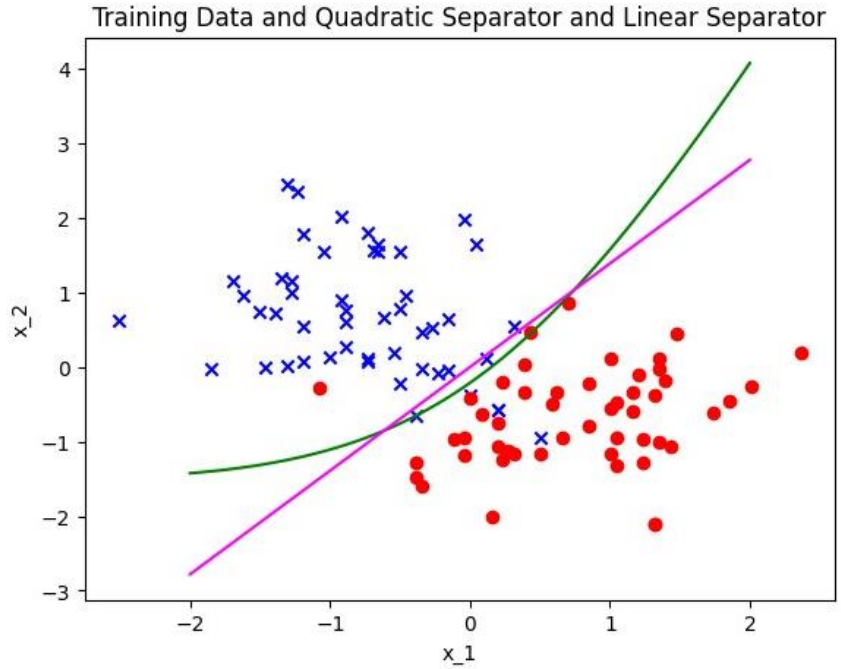


Fig: Linear and Quadratic Boundaries on the Same Plot

f) Visually, we observe that the quadratic boundary is a better separator for classifying the given data. This suggests that the modelling assumption taken by GDA algorithm, namely that $p(x|y)$ is multivariate Gaussian distributed, is approximately correct. However, the covariance matrices of the two classes should be considered unequal.

The quadratic and linear boundaries intersect at two points. This suggests that the data is such that both algorithms produce similar result for majority of the training data. The validity of the comparison between the algorithms is limited by the small size of the training dataset.