COL774 Assignment 1 Report

Japneet Singh
2019MT10696

Q1. Linear Regression

We implemented least squares linear regression to predict density of wine based on its acidity.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))^2$$

where $h\theta(x) = \theta^T x$

Linear Regression was performed 4 times, each with different set of parameters making use of the **Batch Gradient Descent Algorithm**. The set of parameters used for each were:

Note: The convergence criterion used was on the magnitude of the change in value of $J(\theta)$ over two iterations i.e., when the delta in $J(\theta)$ dips below the stopping criteria defined below, we mark the gradient descent process to have converged.

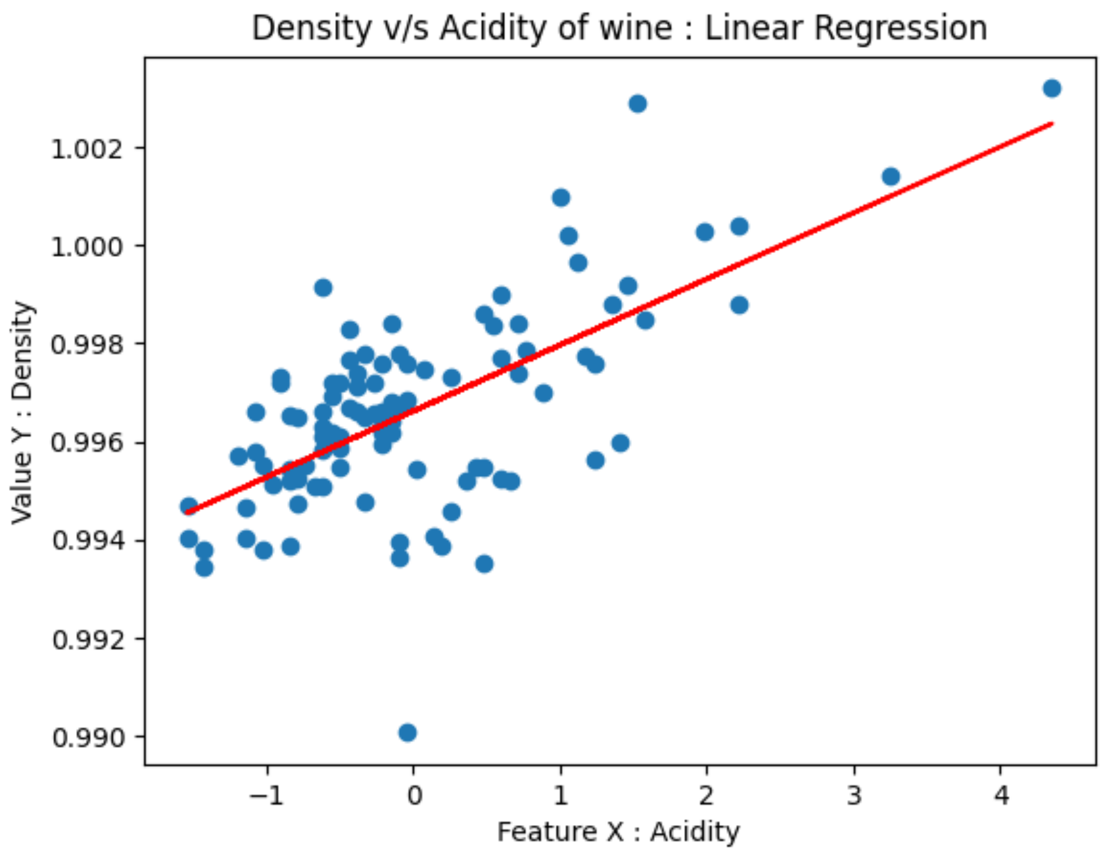
Case 1:
Learning Rate: 0.015
Stopping Criteria: 1e-13
Initial Theta: [0,0]
Final Theta: **[0.99661759,0.00134694]**
Number of iterations taken: 853
Value of J(Theta) at the end: 1.194793057122376e-06

Case 2:
Learning Rate: 0.01
Stopping Criteria: 1e-13
Initial Theta: [0,0]
Final Theta: **[0.99661701,0.00134694]**
Number of iterations taken: 1262
Value of J(Theta) at the end: 1.194794686168393e-06

Case 3:
Learning Rate: 0.025
Stopping Criteria: 1e-8
Initial Theta: [0,0]
Final Theta: **[0.99602185,0.00134608]**
Number of iterations taken: 293
Value of J(Theta) at the end: 1.3830358159370875e-06

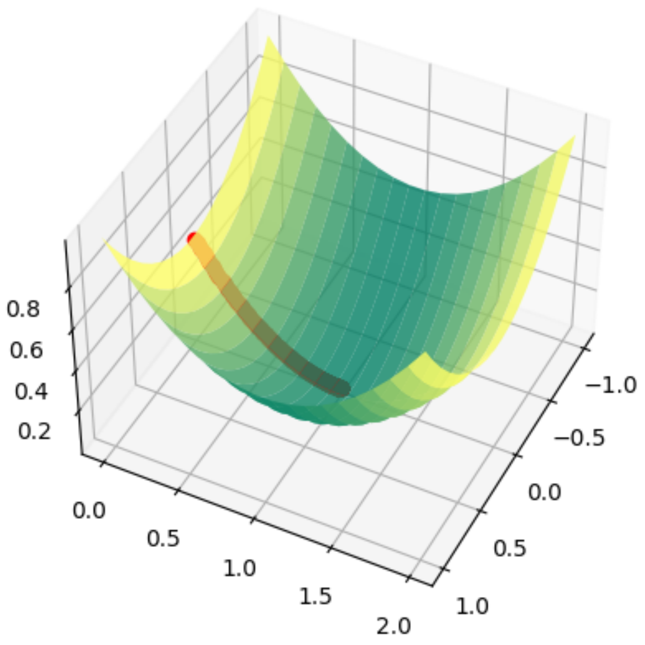
Case 4:
Learning Rate: 0.1
Stopping Criteria: 1e-6
Initial Theta: [0,0]
Final Theta: **[0.9941634,0.00134341]**
Number of iterations taken: 57
Value of J(Theta) at the end: 4.920347001607319e-06

For **Case 1, i.e. learning rate = 0.015**, the scatter plot of training data and the hypothesis function (line) learned can be visualised as:



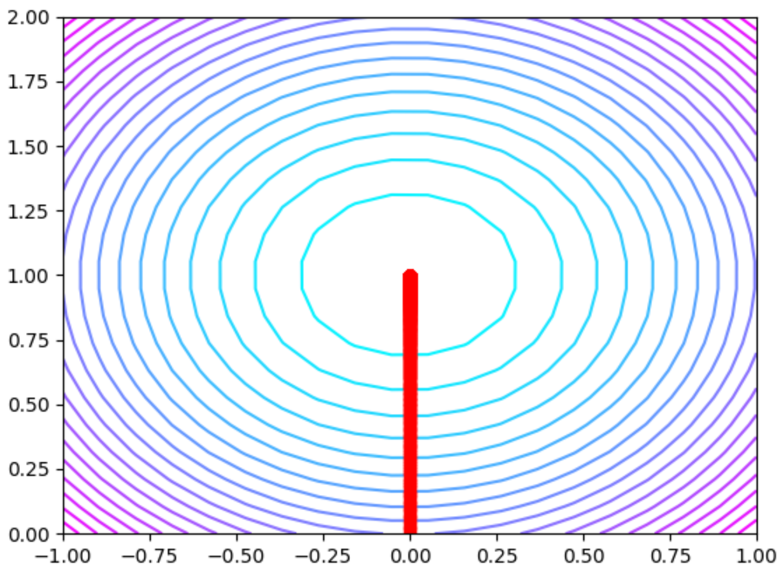
Equation of line is of the form y = mx + c,
where m = 0.9961759
c = 0.00134694

For **Case 1, , i.e. learning rate = 0.015** the movement of (J(θ)) versus the theta vector can be plotted on a 3D mesh:



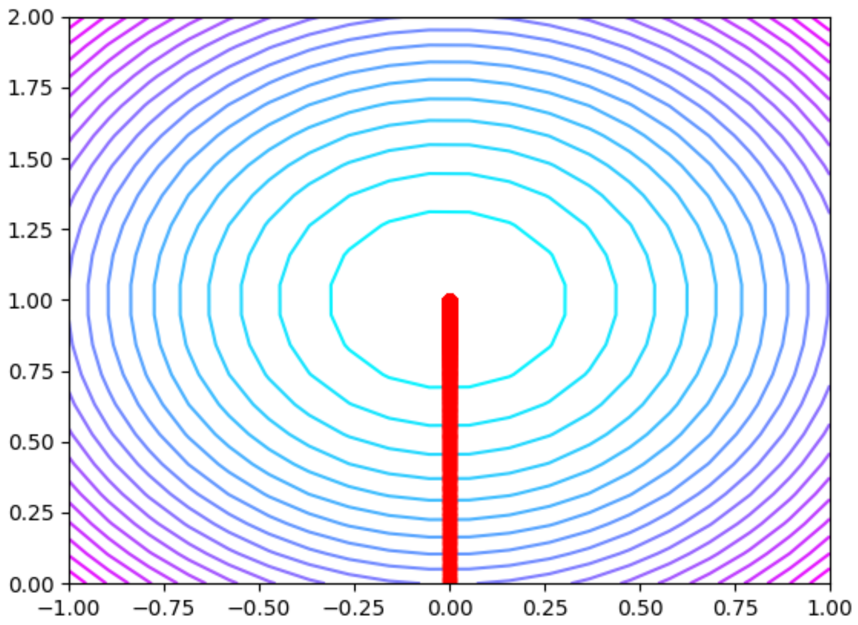
We can see that the J(theta) function's value slowly moves to the minima of the error function as iterations increase. Ultimately, the convergence happens at the minima. For **Case 1, , i.e. learning rate = 0.015** the movement of (J(θ)) versus the number of iterations can be visualized:

For **Case 1, i.e., learning rate = 0.015** contours of the error function at each iteration of the gradient descent obtained were:

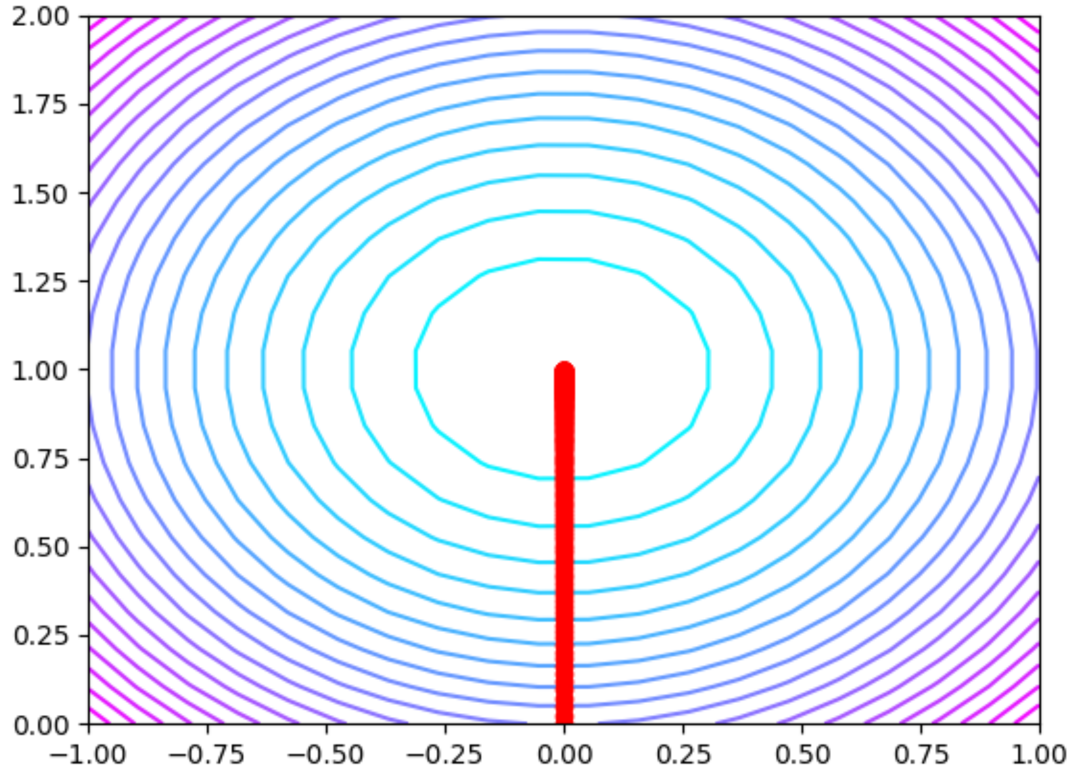


The learning rate being really small, the movement of the point at successive iterations are really really close as jumps are small. (Jumps are proportional to the learning rate)

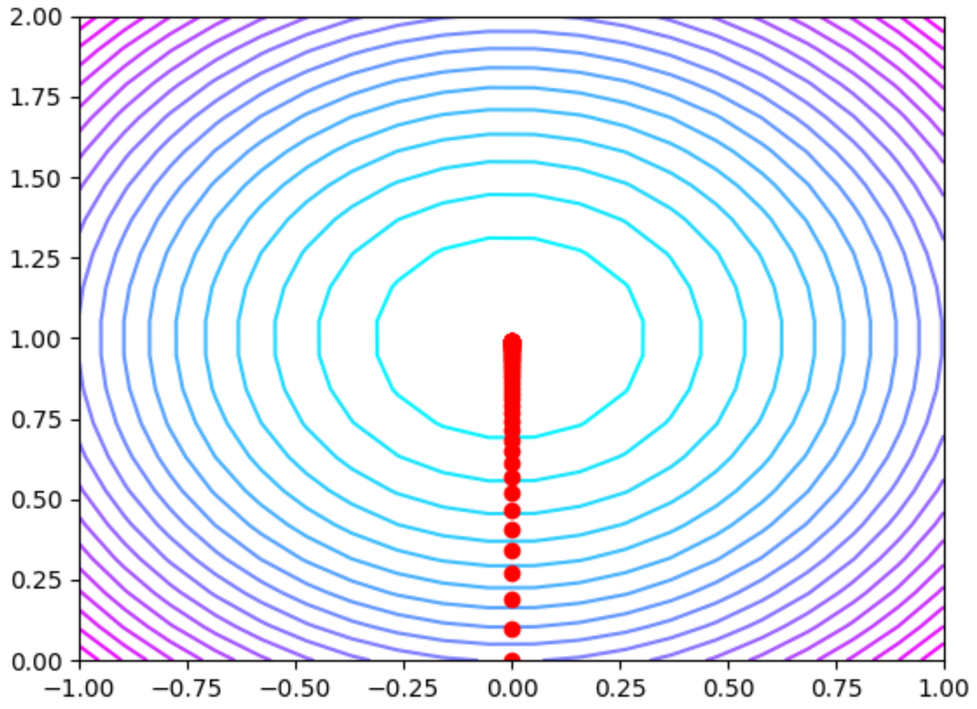
For **Case 2,** contours of the error function at each iteration of the gradient descent obtained were:



For **Case 3,** contours of the error function at each iteration of the gradient descent obtained were:



For **Case 4,** contours of the error function at each iteration of the gradient descent obtained were:



Observations and comments:

In **Cases 1 and 2**, the learning rate being really small, the movement of the point at successive iterations are really really close as jumps are small. (Jumps are proportional to the learning rate)

In **Case 3**, we can better make out individual. Points at the earlier iterations but not as properly as in Case 4.

In **Case 4**, we can see the points separated by significant distance. As the learning rate is high, initial jumps are bigger and then become smaller as points go closer to the convergence point i.e. minima.

Q2. Sampling and Stochastic Gradient Descent

a) Example data generation

1 million examples were sampled, of the form –
$y=h\theta(x)=\theta 0 +\theta 1x1 +\theta 2x2$,
Adding Gaussian noise, equation becomes
$y = \theta 0 +\theta 1x1 +\theta 2x2 +\varepsilon$
$x1 \sim N(3,4)$ and $x2 \sim N(-1, 4)$ independently, and noise $\varepsilon \sim N(0,2)$

b) Implementing SGD

Determining Convergence: We measure the average cost for a given epoch (1000 consecutive iterations for Cases 1,2 and 3 and 100 consecutive iterations for Case 4). When the change in the average error for 2 consecutive epochs falls below the stopping criteria, we can assume that the parameters are sufficiently well fitted and the stochastic gradient descent has 'converged'

Stochastic Gradient Descent was performed four times, each with different set of parameters:

Case 1:
Learning Rate: 0.001
Stopping Criteria: 1e-4
Total sample size: 1,000,000
Batch size: 1
Batch count: 1,000,000
Initial Theta: [0,0,0]
Final Theta: [2.98886831,0.97366594,1.99186265]
Number of epochs taken: 1
Number of iterations per epoch: 1000

Case 2:
Learning Rate: 0.001
Stopping Criteria: 1e-4
Total sample size: 1,000,000
Batch size: 100
Batch count: 10,000
Initial Theta: [0,0,0]
Final Theta: [3.0042368,1.00078961,1.99996945]
Number of epochs taken: 2
Number of iterations per epoch: 1000

Case 3:
Learning Rate: 0.001
Stopping Criteria: 1e-4
Total sample size: 1,000,000
Batch size: 10,000
Batch count: 100
Initial Theta: [0,0,0]
Final Theta: [2.97909372,1.00497879,1.99886228]
Number of epochs taken: 180
Number of iterations per epoch: 1000

Case 4:
Learning Rate: 0.001
Stopping Criteria: 1e-4
Total sample size: 1,000,000

Batch size: 1,000,000
Batch count: 1
Initial Theta: [0,0,0]
Final Theta: [2.89011232,1.02441374,1.99237995]
Number of epochs taken: 11900
Number of iterations per epoch: 100

c) Observations on obtained theta values
All cases converge to slightly different values of theta, although all of them are nearly equal to the theta we used to generate our y values i.e. [3,1,2]. The absolute deltas from [3,1,2] for each case are really low for Case 1 and Case 3.

In Case 1, batch size is 1 and we go through a very small part of the dataset and hence, convergence is absolutely fast - it takes only 1 epoch i.e., 1000 iterations to achieve the stopping criteria. As can be seen in the plot below, the path to convergence is highly random.

In Case 2, batch size is 100 and we go through a very small part of the dataset (larger part than Case 1) and hence, convergence is still very fast - it takes only 2 epochs i.e., 2000 iterations to achieve the stopping criteria. As can be seen in the plot below, the path to convergence is highly random but much more regular than that seen in Case 1.

In Case 3, batch size is 10000 and we go through a substantial part of the dataset (much larger part than both earlier cases) and hence, convergence takes a bit of time. 180 epochs were taken to achieve the stopping criteria i.e., 180,000 iterations. As can be seen in the plot below, the path to convergence is really regular and. The theta values are converging bit by bit.

In Case 4, batch size is 1000,000 and we go through all of the dataset every time and hence, convergence takes a long long time as compared to previous 3 cases. 11900 epochs were taken to achieve the stopping criteria i.e., 1,190,000 iterations. As can be seen in the plot below, the path to convergence is a proper regular path, and a converging trend can be seen easily as iterations happen.

Error comparison for q2test.csv

Case 1:
Error with original hypothesis: 0.9963613839273432
Error with new test samples: 1.024257556312748
Absolute difference: 0.02789617238540487
Case 2:
Error with original hypothesis: 0.9976397220770931
Error with new test samples: 0.982920729657035
Absolute difference: 0.01471899242005803
Case 3:
Error with original hypothesis: 1.0004159077248003
Error with new test samples: 0.9843260587365432
Absolute difference: 0.01608984898825705

Case 4:
Error with original hypothesis: 1.0021074735961992
Error with new test samples: 1.0186395015648515
Absolute difference: 0.016532027968652274

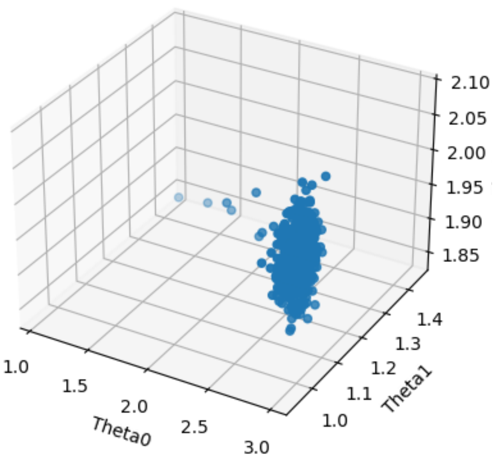
As we can see, the errors were comparable to the bigger dataset for the smaller dataset as well. Lowest error was observed for Case 3.

d) Theta plots

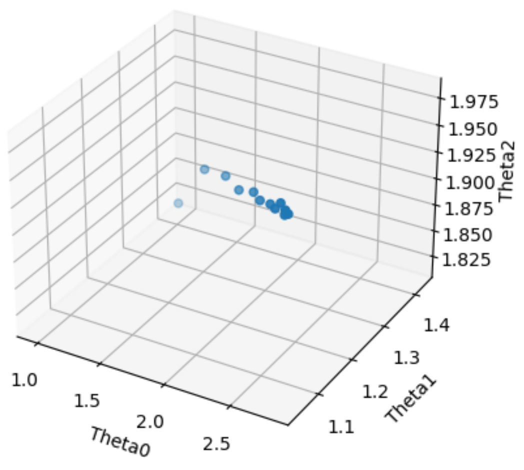
The plots for values of theta obtained as iterations go on are on expected lines of the class discussion. As batch size. Increases with respect to the total dataset's size, the randomness in the path decreases. This is seen here as the path taken for batch Size = 1is extremely random and becomes less random and more regular as batch size goes from 1 to 100,000.

Scatter Plots -

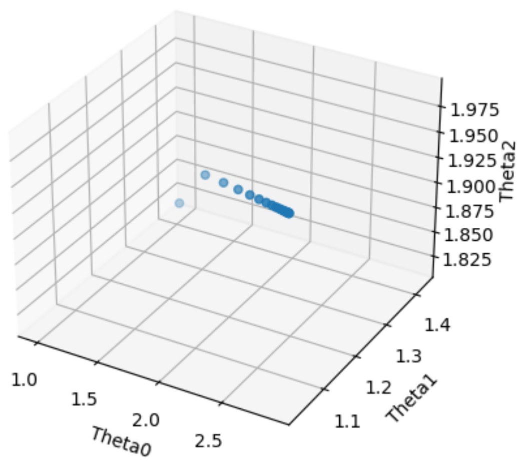
Movement of Theta, Batch Size 1



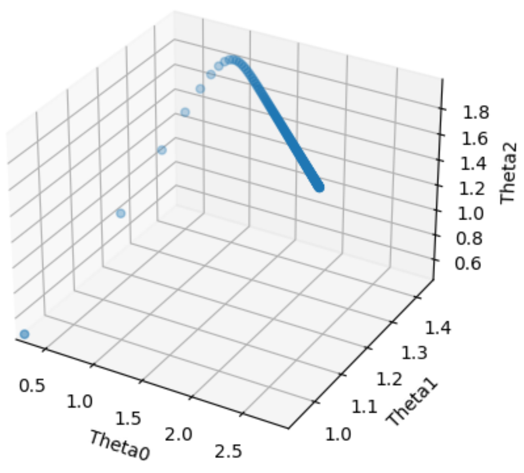
Movement of Theta, Batch Size 100



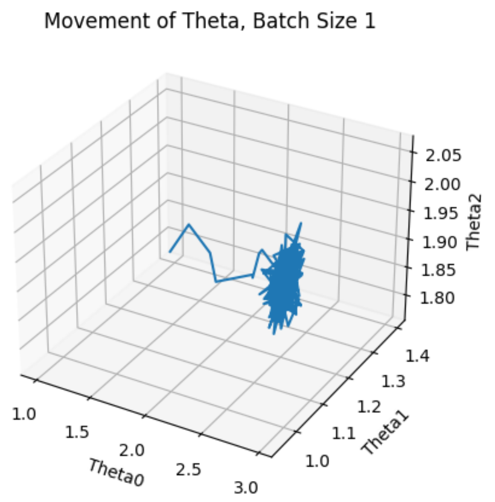
Movement of Theta, Batch Size 10000

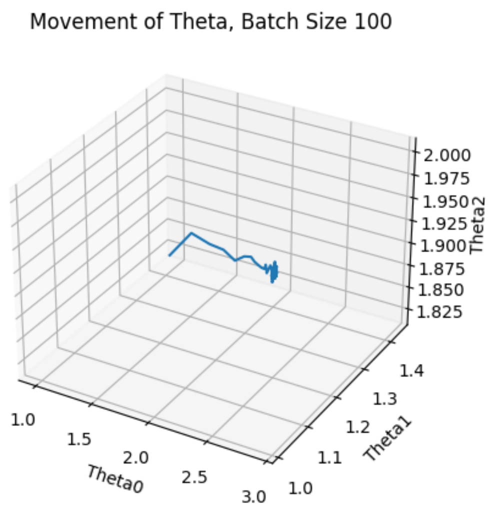


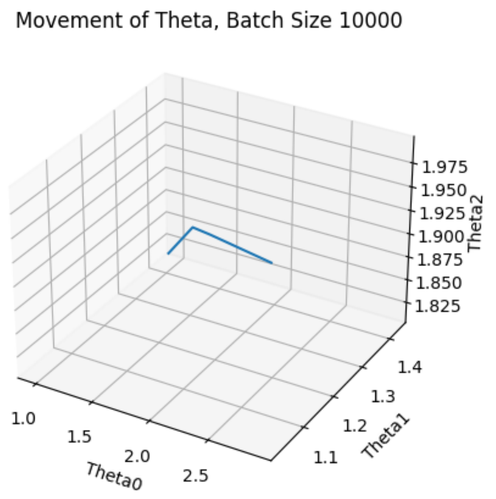
Movement of Theta, Batch Size 1000000

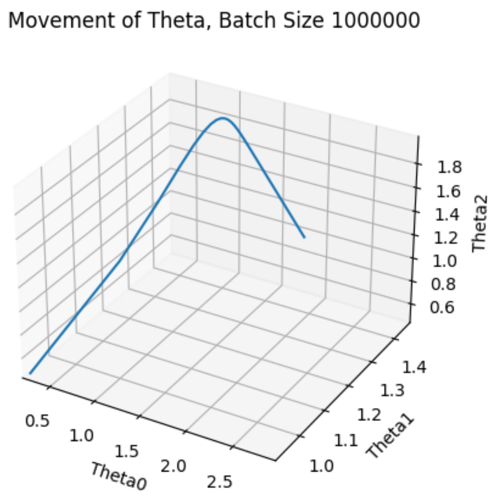


Line plots -


Movement of Theta, Batch Size 1
Theta0
Theta1
Theta2
1.0
1.5
2.0
2.5
3.0
1.0
1.1
1.2
1.3
1.4
1.80
1.85
1.90
1.95
2.00
2.05


Movement of Theta, Batch Size 100
Theta0
Theta1
Theta2
1.0
1.5
2.0
2.5
3.0
1.0
1.1
1.2
1.3
1.4
1.825
1.850
1.875
1.900
1.925
1.950
1.975
2.000


Movement of Theta, Batch Size 10000
Theta0
Theta1
Theta2
1.0
1.5
2.0
2.5
1.1
1.2
1.3
1.4
1.825
1.850
1.875
1.900
1.925
1.950
1.975

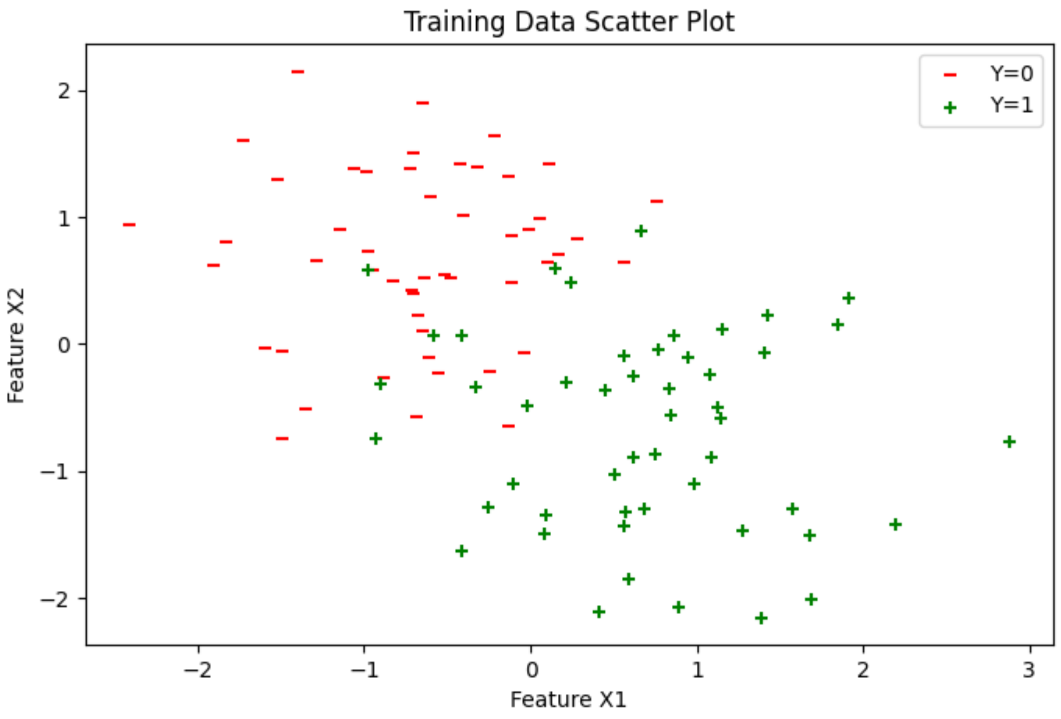

Movement of Theta, Batch Size 1000000
Theta0
Theta1
Theta2
0.5
1.0
1.5
2.0
2.5
1.0
1.1
1.2
1.3
1.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8

Q3. Logistic Regression

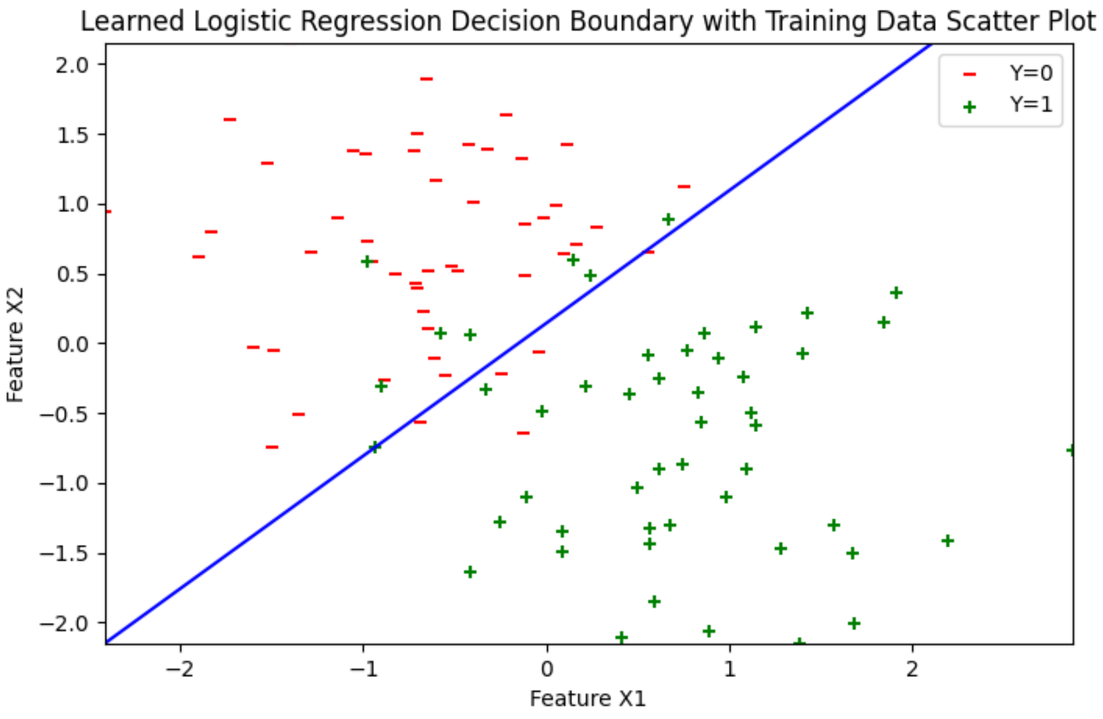
Log Likelihood : $L(\theta) = \sum_{i=1}^{m} \; y^{(i)} \log h\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h\theta(x^{(i)}))$

Parameters
Initial theta used: [0,0,0]
Final theta obtained: [0.37816933, 2.52286505, -2.65400492]
Stopping criteria for Newton's method: 1e-4
Note: Stopping criteria applied on change in value of theta over successive iterations

Hypothesis data can be visualised as:



Linear decision boundary obtained can be visualised as:


Learned Logistic Regression Decision Boundary with Training Data Scatter Plot
Y=0
Y=1
Feature X2
Feature X1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2

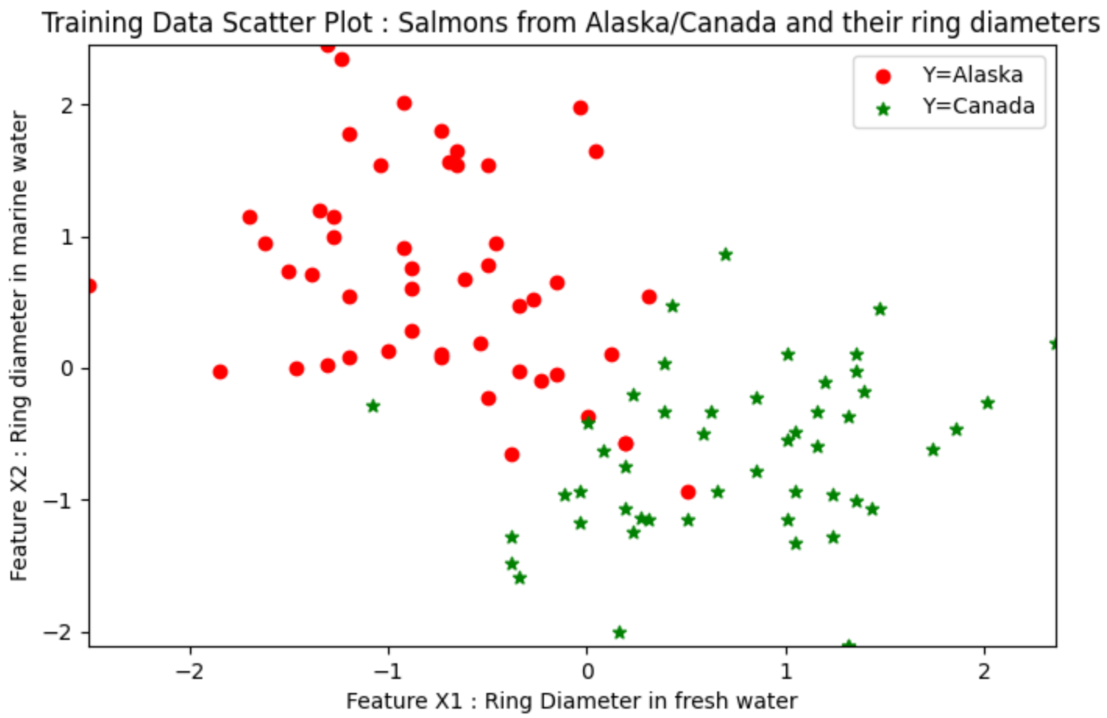
Q4. Gaussian Discriminant Analysis

Y = 0 represents the class **Alaska**
Y = 1 represents the class **Canada**

a) Assuming same covariance matrices

$$\mu_0 = [-0.75529433, 0.68509431]$$
$$\mu_1 = [\,0.75529433, -0.68509431]$$
$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

b) Scatter plot of training data with labels for Alaska or Canada

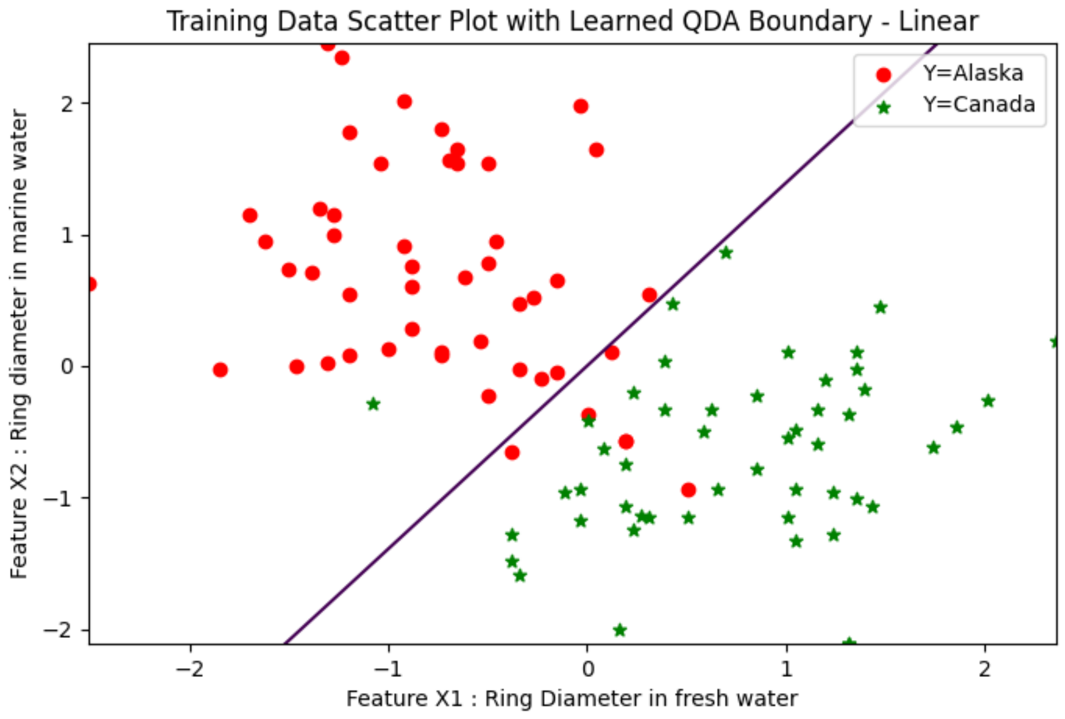


c) Linear Decision Boundary

The equation of the boundary in the linear case is derived to be as follows –

$$x^T\Sigma^{-1}(\mu_1-\mu_0) - 0.5(\mu_1^T\Sigma^{-1}\mu - \mu_0^T\Sigma^{-1}\mu_0) = \ln(Ø/1-Ø)$$

Ø is 0.5 for the training set.



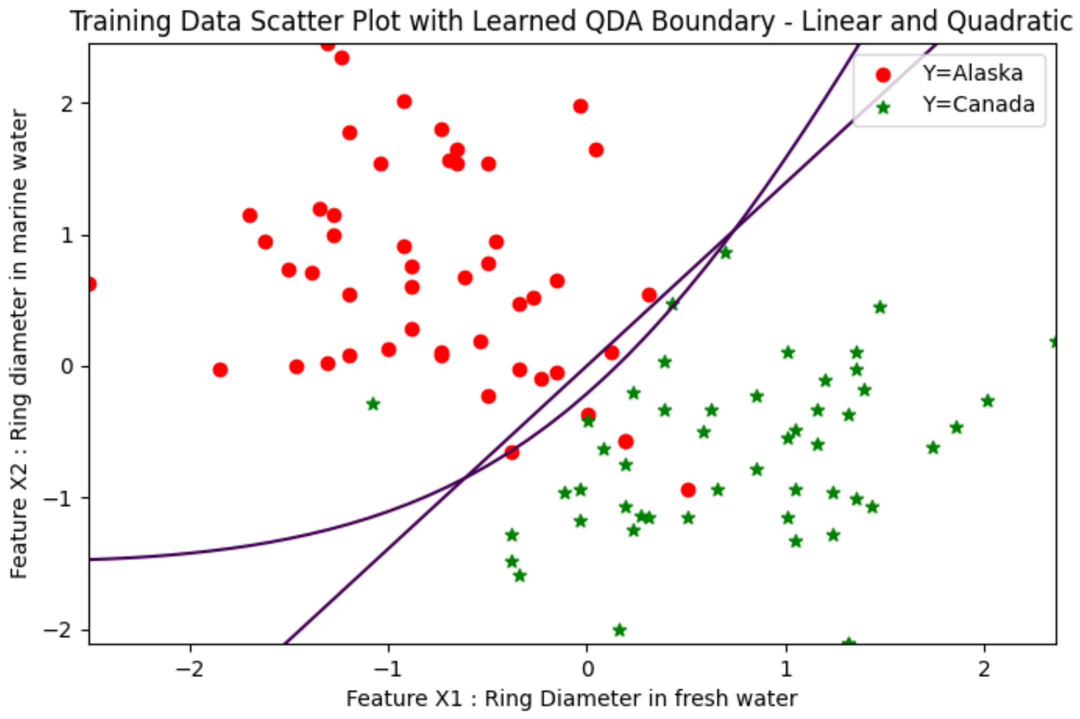
d) Different covariance matrix cases

$$\mu_0 = [-0.75529433, 0.68509431]$$
$$\mu_1 = [0.75529433, -0.68509431]$$
$$\Sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$
$$\Sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

e) Quadratic Decision Boundary

The equation for the quadratic decision boundary is derived to be as follows –

$$\mathbf{0.5\ln(|\Sigma_0|/|\Sigma_1|) + 0.5((X - \mu_1)^T\Sigma_1^{-1}(X - \mu_1) - (X - \mu_0)^T \Sigma_0^{-1} (X - \mu_0)) = \ln(\text{Ø}/1\text{-Ø})}$$



e) Observations and Comments

Both linear and quadratic boundary cases perform decently on the training data, both are highly accurate but the quadratic boundary performs slightly better. Quadratic boundary accommodates a couple of corner cases that the linear boundary cannot fit well.
Quadratic boundary has the extra knowledge of the covariance of the two classes, hence, is better at classifying points we may receive at test time.
However, quadratic may also suffer from overfitting if we have a smaller dataset, hence for small datasets linear boundary can outperform quadratic boundary as well. Therefore, which model we choose is highly dependent on the kind and quantity of training data we have.