Assignment 1

Machine Learning

Yuyutsu Saini

$9^{st} September, 2022$

COL 774 Machine Learning

1. Linear Regression
2. Stochastic Gradient Descent
3. Logistic Regression
4. Gaussian Discriminant Analysis

1 Linear Regression

In statistics, linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). We predit the outcome value as a linear function of features of given data point.

$$y = \theta^T x = \theta_0 + \theta_1 x_1 \tag{1}$$

1.1 Implementing Batch Gradient Descent

Here in this problem I minimized cost function using batch gradient descent with help of library such as numpy, performed all matrix calculation using numpy. The cost function and update rule for batch gradient descent is given below:

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

Now, the gradient update rule is as follows:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{3}$$

$$\theta_j := \theta_j - \alpha(y^{(i)} - \theta^T x^{(i)})x_j^{(i)} \tag{4}$$

The following is the parameters used and then some obtained by batch gradient descent:

$$\theta = \begin{pmatrix} 0.99661018 \\ 0.00134018 \end{pmatrix} \tag{5}$$

- → Learning rate :$\alpha = 10^{-2}$
- → Error Bound (after we need to stop iterating) = 10^{-12}
- → No. of Iterations = 1146

1.2 Linear Regression Plot

Here is the plot for linear regression with all data points of the training data plotted using matplotlib:

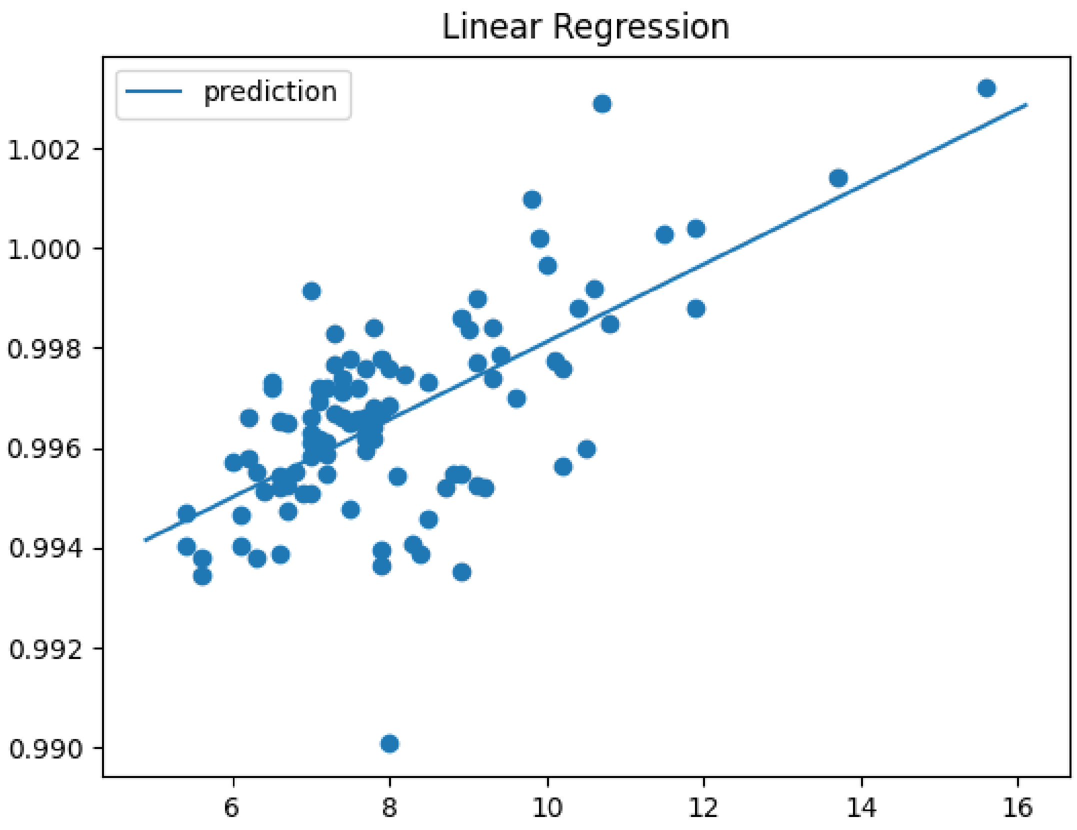

Figure 1: Linear Regression plot

1.3 Cost function Plot

Here is the plot for Cost function, I also implemented animation using matplotlib.animation and the plot is created using matplotlib with z axis as cost function and x and y axis as θ_0 and θ_1 values respectively:

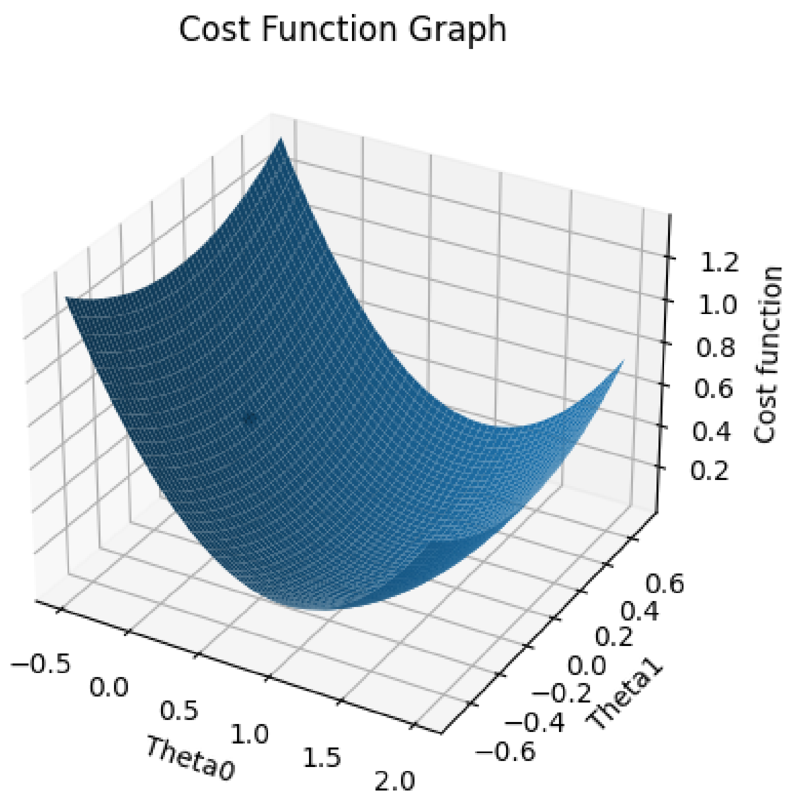

Figure 2: Cost Function plot

1.4 Contours Plot

I've plotted contours of $J(\theta)$ and the line shows the variation of the θ parameters on each iteration.

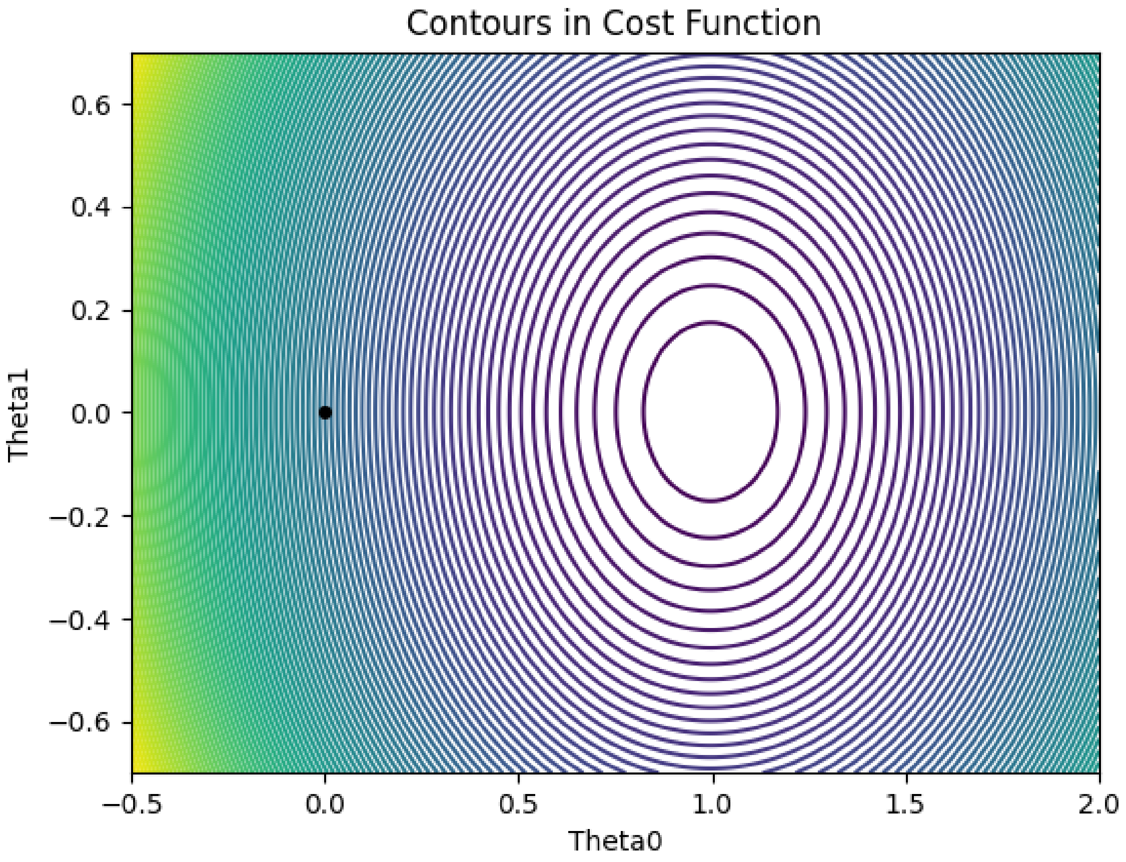

Figure 3: Contours in Cost function

1.5 Analysis w.r.t. to different Learning Rates

With the large value of learning rate, the convergence attains higher speed (number of iterations decreases) and jump towards minima increases but gains risks of skipping the minima and also the oscillation occurs with higher amplitude and hence higher learning is prone to high error and low learning is prone to very less convergence speed and hence, more time. A wise choice of learning rate is necessary for good working of a model w.r.t. to both training and accuracy.

a) $\alpha = 0.001$

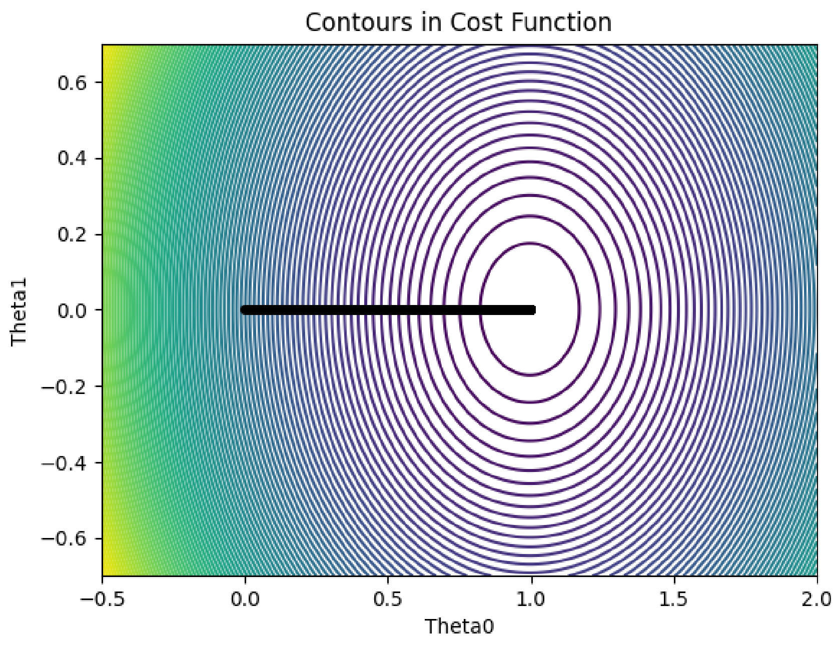

Figure 4: Movement of Theta with α = 0.001

b) $\alpha = 0.025$

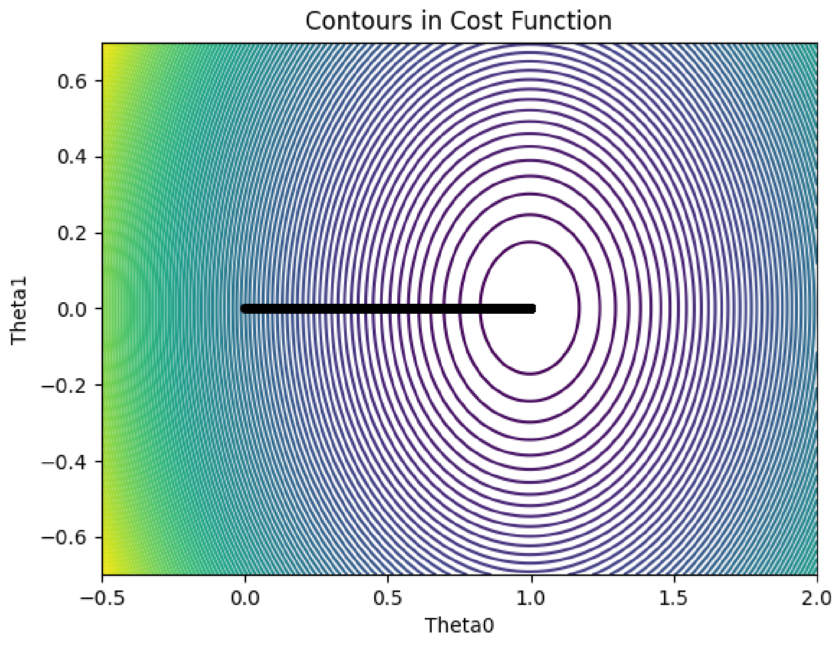

Figure 5: Movement of Theta with α = 0.025

c) $\alpha = 0.1$

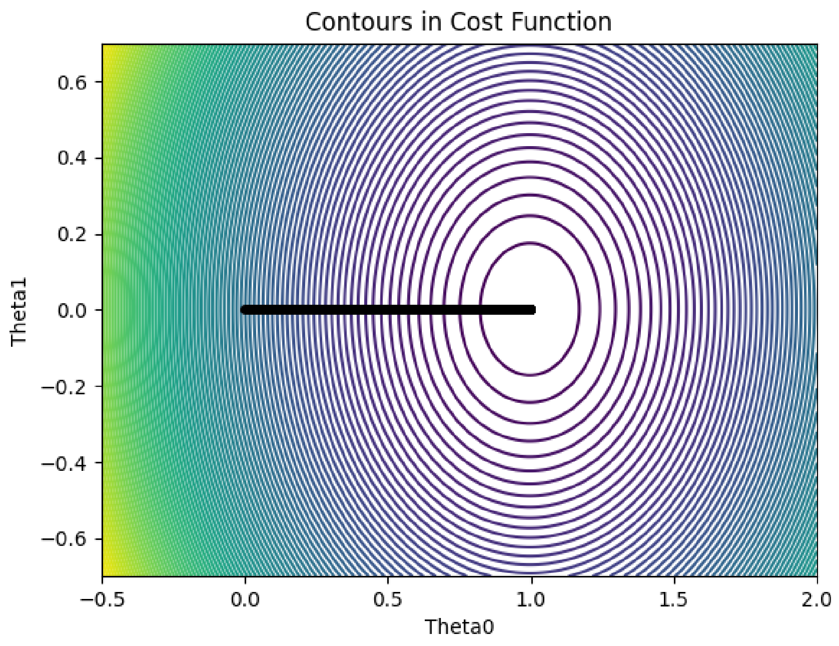

Figure 6: Movement of Theta with α = 0.1

2 Stochastic Gradient Descent

2.1 Sampling 1 million points

$$x_1 \sim \mathcal{N}(3, 4) \tag{6}$$

$$x_2 \sim \mathcal{N}(-1, 4) \tag{7}$$

$$y \sim \theta^T x + \mathcal{N}\left(0, \sqrt{2}\right) + \epsilon^{(i)} \tag{8}$$

In this problem, I have generated data for 1 million points according to above distribution using numpy random number generator.

2.2 Stochatic Gradient Descent

Here in this part, I have implemented stochastic gradient descent for a varying batch size b and used the stopping criteria explained in Andrew Ng video as suggested to us. Then, tried few variations in that criteria and chose the best one.

The value of theta obtained for different batch sizes is with learning rate 0.001 and with error bound = 10^{-2}.

Batch size (r)	θ	**Iterations**
1	$\begin{pmatrix} 2.9400261 \\ 0.9548869 \\ 1.98625056 \end{pmatrix}$	18000
100	$\begin{pmatrix} 2.68369856 \\ 1.07110958 \\ 1.97589524 \end{pmatrix}$	40000
8000	$\begin{pmatrix} 2.75907777 \\ 1.0530667 \\ 1.98210365 \end{pmatrix}$	9000
1000000	$\begin{pmatrix} 0.65677868 \\ 1.45581687 \\ 1.66016601 \end{pmatrix}$	630

Stopping criteria = 10^{-2} is very less for batch gradient to converge to a correct value of θ that why's we observe large error in the case where b = 1 million.

2.3 Analysis of convergence on different batch sizes

No, different algorithms doesn't converges to the same parameter as different set of noises are encountered in every iterations. Also, it depends on the order of data points that are encountered in the different iterations.
Actually these values are very close to each other and have very little differences. Deviation increases with the decrease in batch size as there is more variation in noises encountered with the change in set of data points in different iterations.
Algorithm works faster for smaller batch due to less number of data points in each set of data points and hence less time for each iteration. Number of iterations may increase with

decrease in batch size.
The error decreases as curve becomes smooth and wobbles less when reaches minima. This makes the curve move towards the right direction when it is very close to the answer.

2.4 Movement of θ on varying batch sizes

The movement is more oscillatory for smaller batch sizes and becomes smoother as we increase batch sizes. Yes, this makes sense as for larger batch sizes change in theta is along the gradient and thus directed towards the minima.
The plots for different batch sizes is as follows:

a) b=1

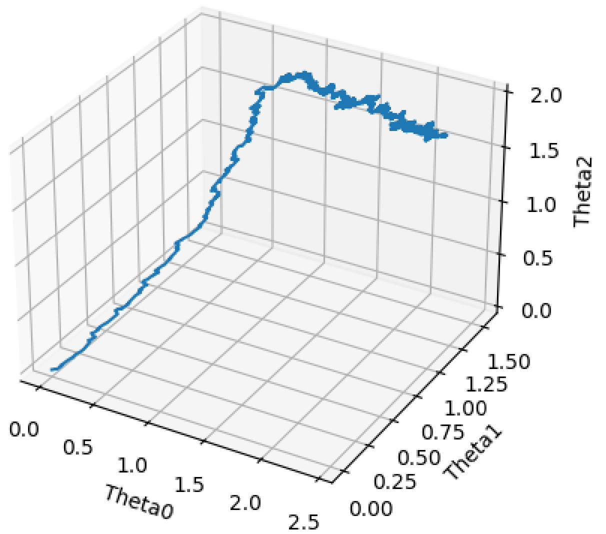

Figure 7: Movement of θ for b=1

b) b=100

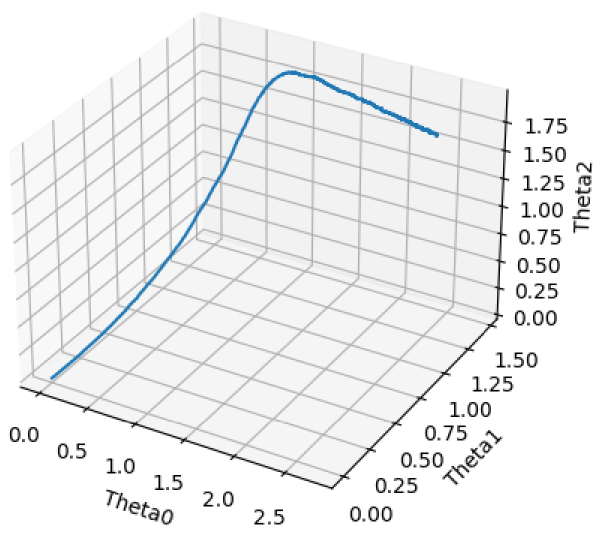

Figure 8: Movement of θ for b=100

c) b=10000

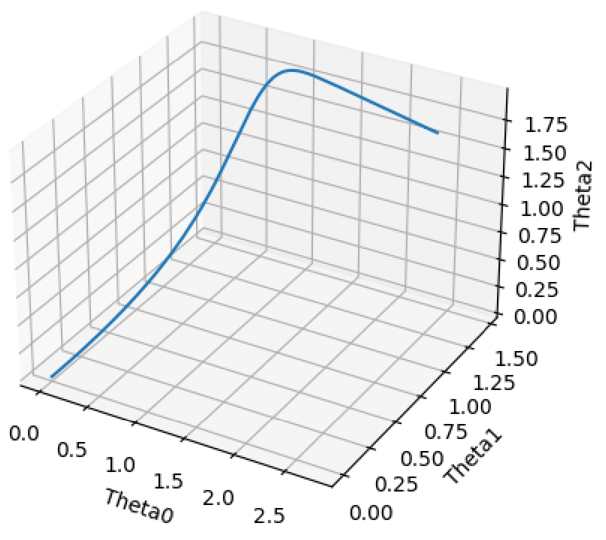

Figure 9: Movement of θ for b=10000

d) b=1000000

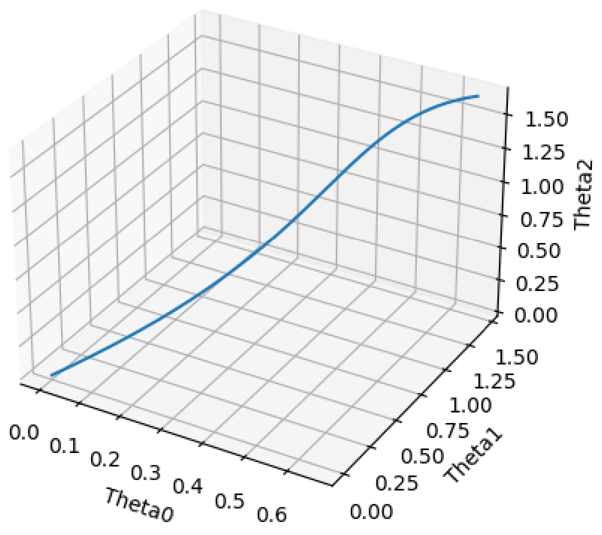

Figure 10: Movement of θ for b=1000000

3 Logistic Regression

This is a classification which to classify data into classes and uses sigmoid function in order do so. Trains on the given data to find the appropriate values of θ to maximize the log likelihood of the function.

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp^{-\theta^T x}} \tag{9}$$

$$P(y = 1|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{10}$$

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y(i)} (1 - h_\theta(x^{(i)}))^{1-y} \tag{11}$$

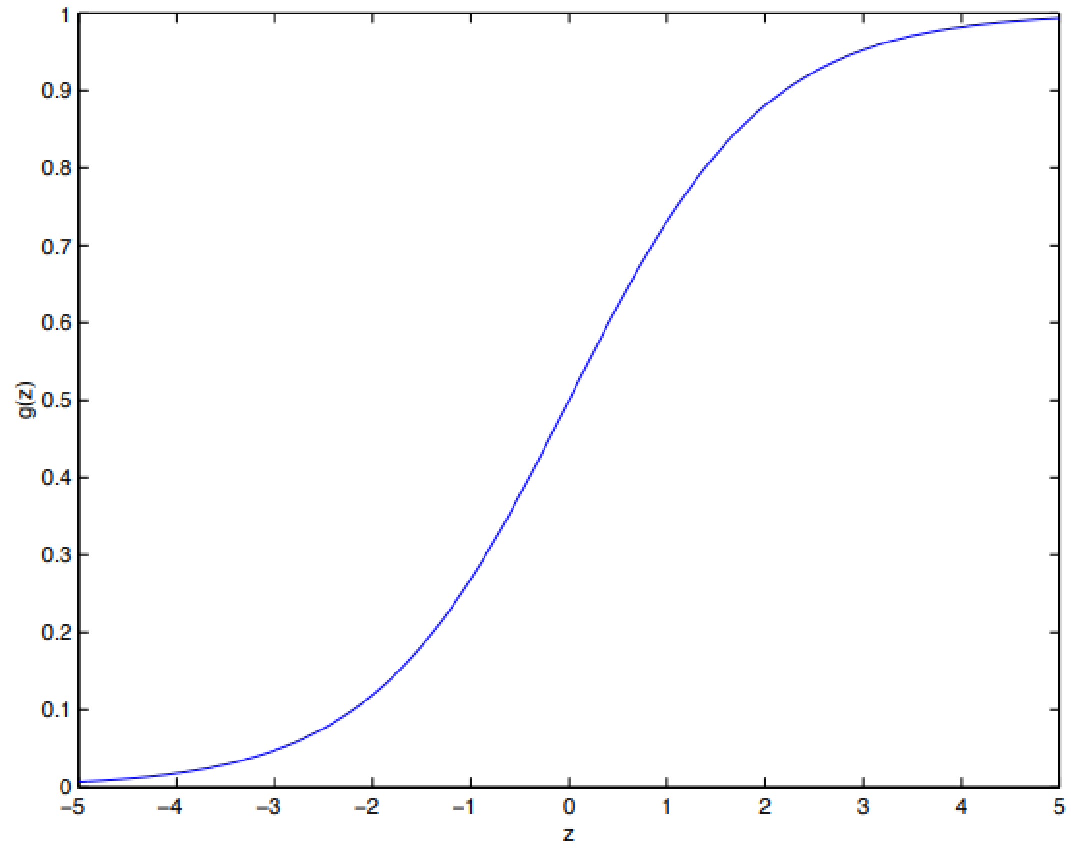

Figure 11: Sigmoid Function

3.1 Newton's Method

Newton's method finds the θ by calculating roots of derivative of cost function using newton raphson's method. The equation goes as follows:

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)} \tag{12}$$

which solves to the following:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \mathcal{H}^{-1} \nabla_{\theta} LL(\theta) \tag{13}$$

The hessian for the equation turns is as follows:

$$\mathcal{H} = \sum_{i=1}^{n} \frac{\exp\left(-\theta^T x^{(i)}\right)}{\left(1 + \exp\left(-\theta^T x^{(i)}\right)\right)^2} \left(x^{(i)} (x^{(i)})^T\right) \tag{14}$$

Gradient of log likelihood function is calculated using:

$$\nabla_\theta LL(\theta) = \sum_{i=1}^{n} \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x^{(i)}$$

The value of θ obtained is:

$$\theta = \begin{pmatrix} 0.40125316 \\ 2.5885477 \\ -2.72558849 \end{pmatrix} \tag{15}$$

Here, the error bound (stopping criteria) used is $1e - 15$.

3.2 Decision Boundary

The data points and decision boundary found using logistic regression is depicted in plot given below.

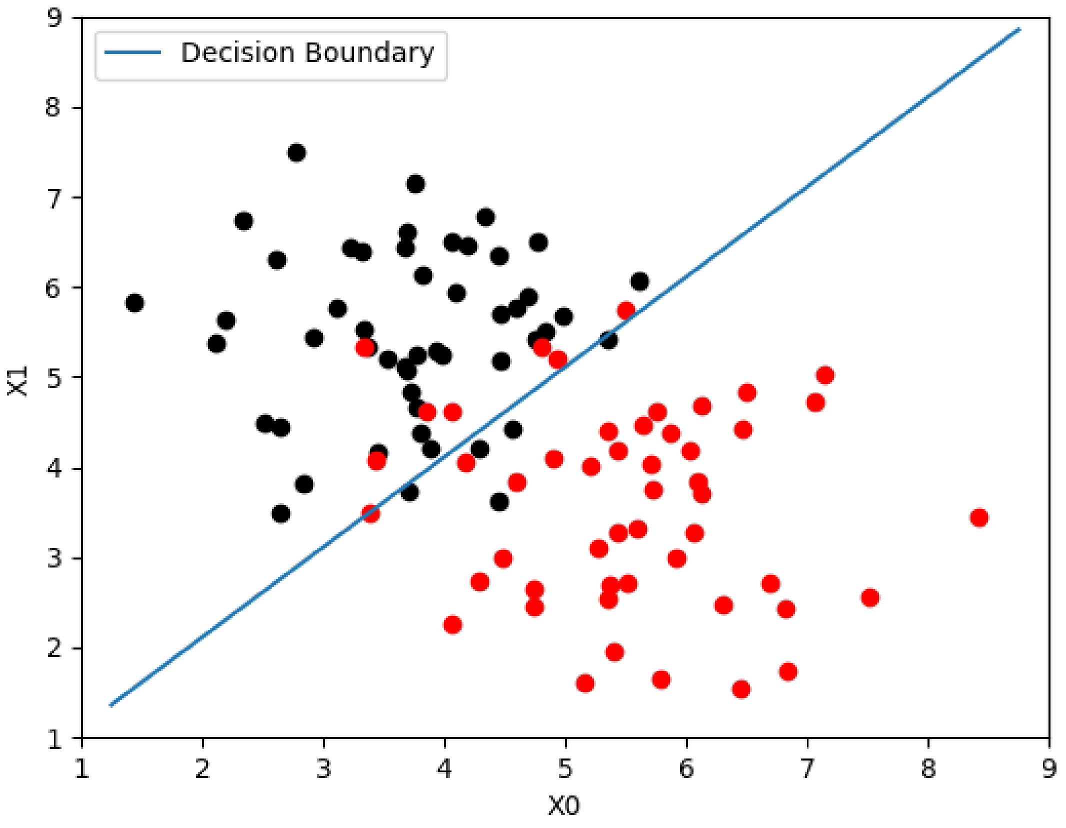

Figure 12: Logistic Regression plot

4 Gaussian Discriminant Analysis

In this model, we'll assume that p(x|y) is distributed according to a multivariate normal distribution and then we apply Bayes theorem to find out the probability of p(y—x).

$$P(y) = \phi^y(1-\phi)^{1-y} \tag{16}$$

$$P(x|y=0) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)} \tag{17}$$

$$P(x|y=1) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)} \tag{18}$$

$$P(y|x) = \frac{P(x|y=0)(1-\phi) + P(x|y=1)\phi}{P(x)} \tag{19}$$

4.1 Values of Parameter

The maximum likelihood parameters µ0, µ1 and the co-variance matrix Σ are:

$$\phi = 0.5 \tag{20}$$

$$\mu_0 = \begin{pmatrix} -0.75529433 \\ 0.68509431 \end{pmatrix} \tag{21}$$

$$\mu_1 = \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \tag{22}$$

$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix} \tag{23}$$

4.2 Training Data Plot

The plot with data points is:

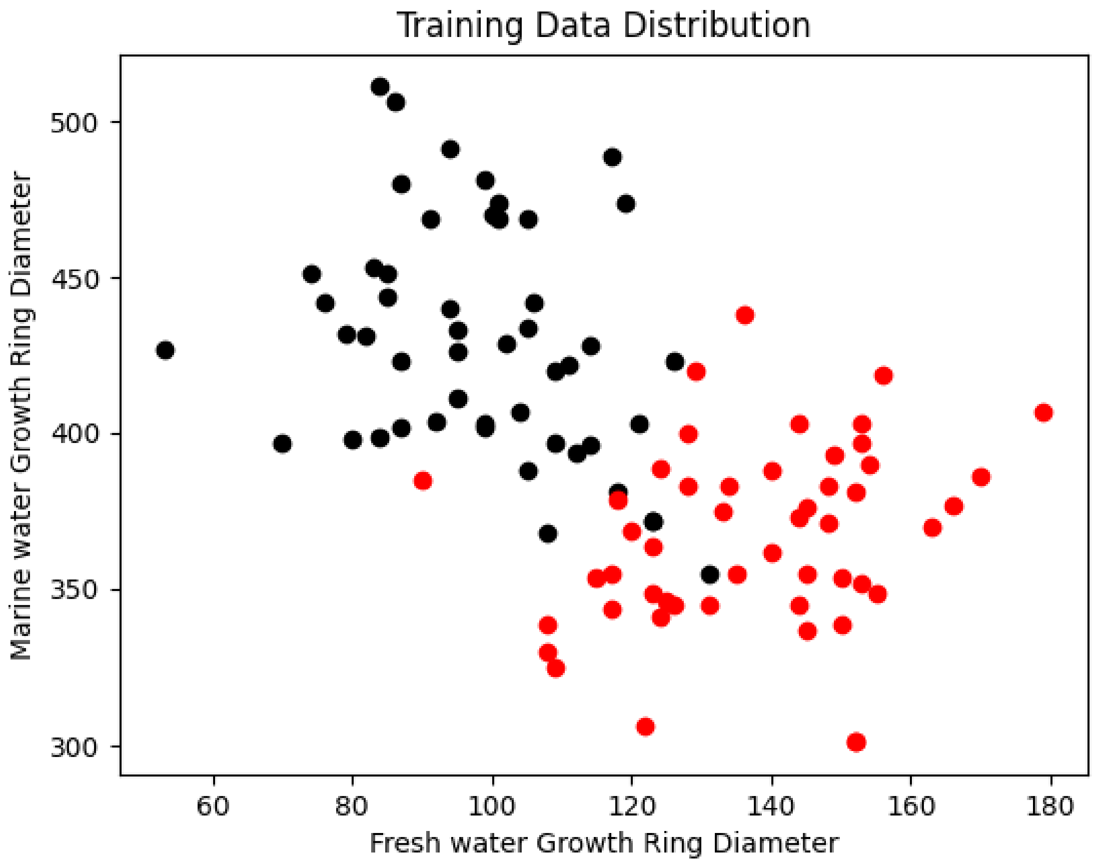

Figure 13: Training Data points

4.3 Linear Decision Boundary

Equation for linear boundary is:

$$\left(\mu_1^T - \mu_0^T\right)\Sigma^{-1}x + \log\left(\frac{\phi}{1-\phi}\right) + \frac{1}{2}\left(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1\right) = 0 \tag{24}$$

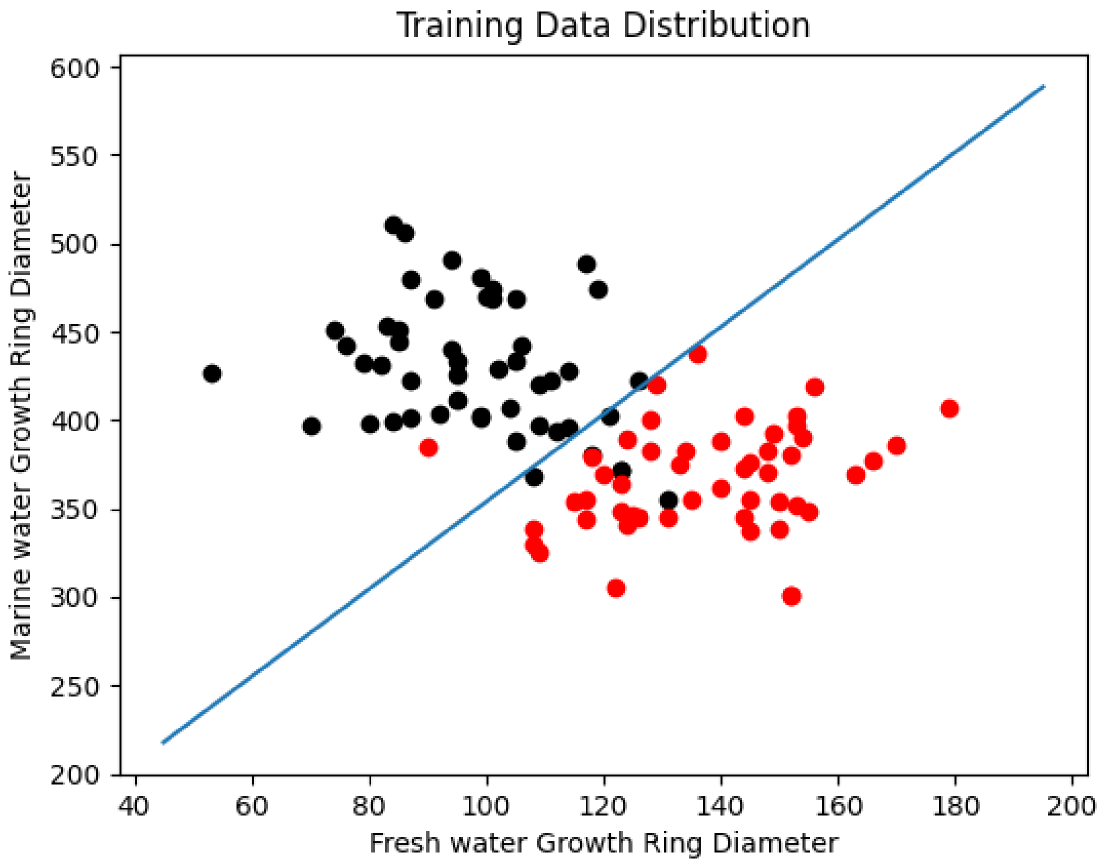

Figure 14: Linear Decision Boundary

4.4 Value of parameter with different CoVariance Matrix

The values obtained are:-

$$\phi = 0.5 \tag{25}$$

$$\mu_0 = \begin{pmatrix} -0.75529433 \\ 0.68509431 \end{pmatrix} \tag{26}$$

$$\mu_1 = \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \tag{27}$$

$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix} \tag{28}$$

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

4.5 Parabolic Decision Boundary

The equation of the quadratic boundary is:-

$$\frac{1}{2}x^T\left(\Sigma_0^{-1}-\Sigma_1^{-1}\right)x+\left(\mu_1^T\Sigma_1^{-1}-\mu_0^T\Sigma_0^{-1}\right)x+\log\left(\frac{\phi}{1-\phi}\right)+\frac{1}{2}\log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right)+\frac{1}{2}\left(\mu_0^T\Sigma_0^{-1}\mu_0-\mu_1^T\Sigma_1^{-1}\mu_1\right)=0 \tag{29}$$

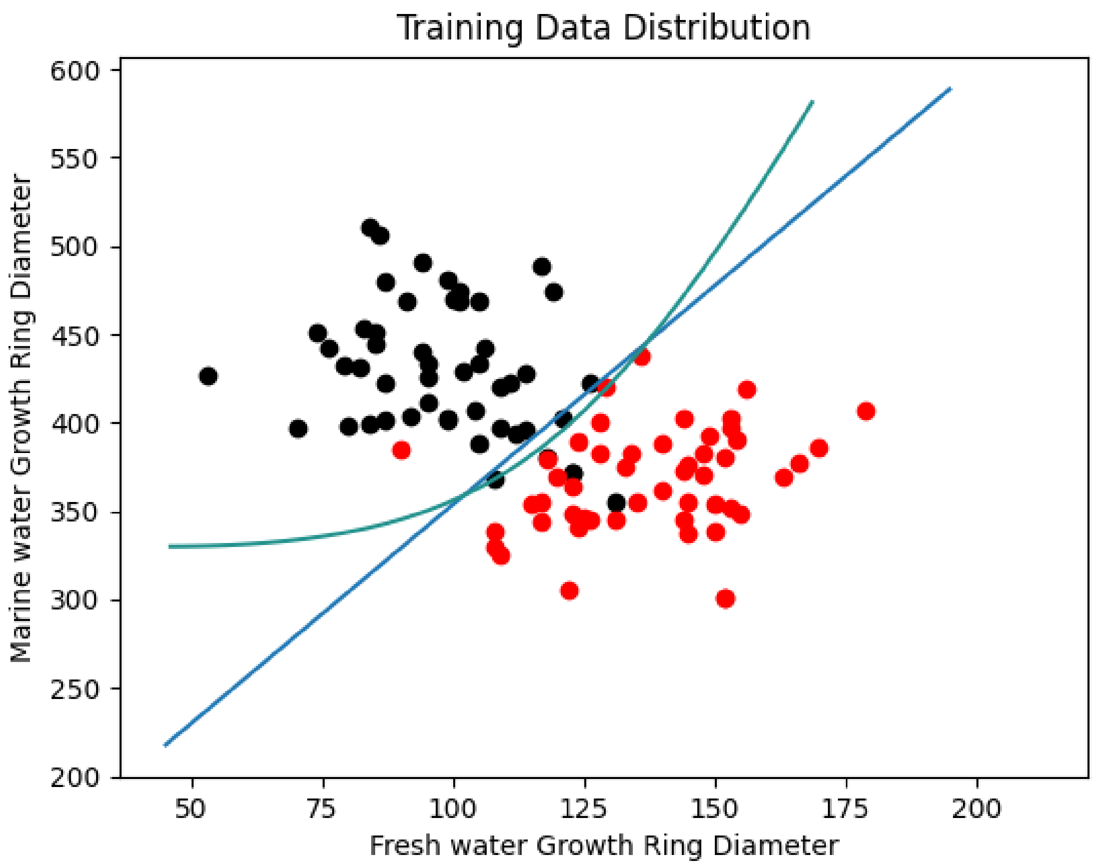

Figure 15: Parabolic Decision Boundary

4.6 Observation

Looking at the separators in the plot, it seems that both are separating the data preety well. The quadratic separator just barely accommodates two extra spots correctly. Since the quadratic separator bends in the direction of Alaska, it suggests that we are more likely to have salmon from Canada than Alaska. However, it is clear from the training set of data that this is not the case. Due to overfitting on the test data, the quadratic separator does not actually produce noticeably superior results, even on the test data.