COL 774, Assignment 1

Rishi Jain
2020CS10373

September 2022

1 Linear Regression

a) I vectorized linear regression for improved performance. This was accomplished by noting that using matrix multiplication, the vector of all predictions by the current can be determined. Then, by subtracting the initial values, we can obtain a vector of errors that can be used to determine the gradient as well as the error function's value (this is faster than the Python for loop since numpy is implemented internally in a faster language).

- Learning rate $=10^{-2}$
- stopping criteria is that the change in error function becomes less than 10^{-15}.
- $\theta = [0.99661979, 0.0013402]$

b) The graph for linear regression is
The plot is

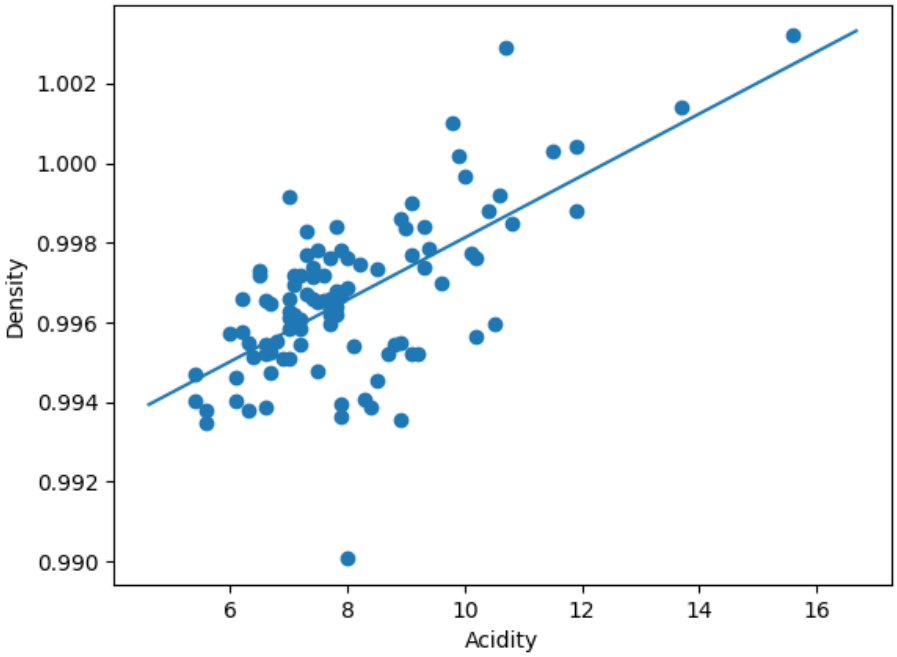

Figure 1: Data and Hypothesis plot for Q1

c) On the Z axis I've plotted $J(\theta)$ and the line shows the variation of the θ parameters on each iteration.
The plot is

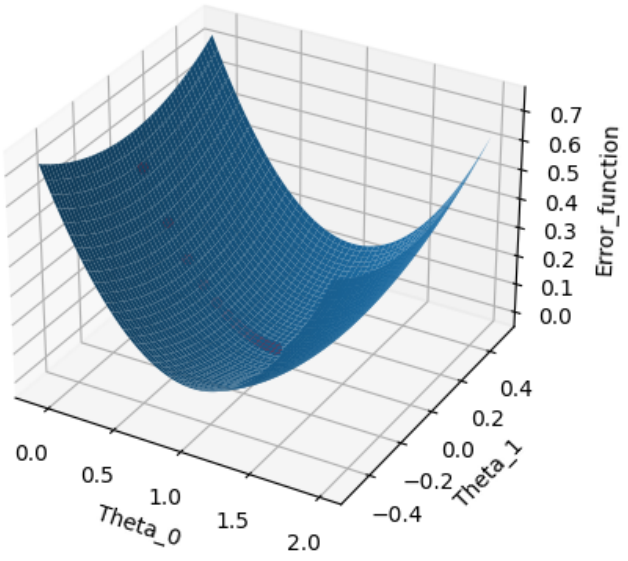

Figure 2: error function plot

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

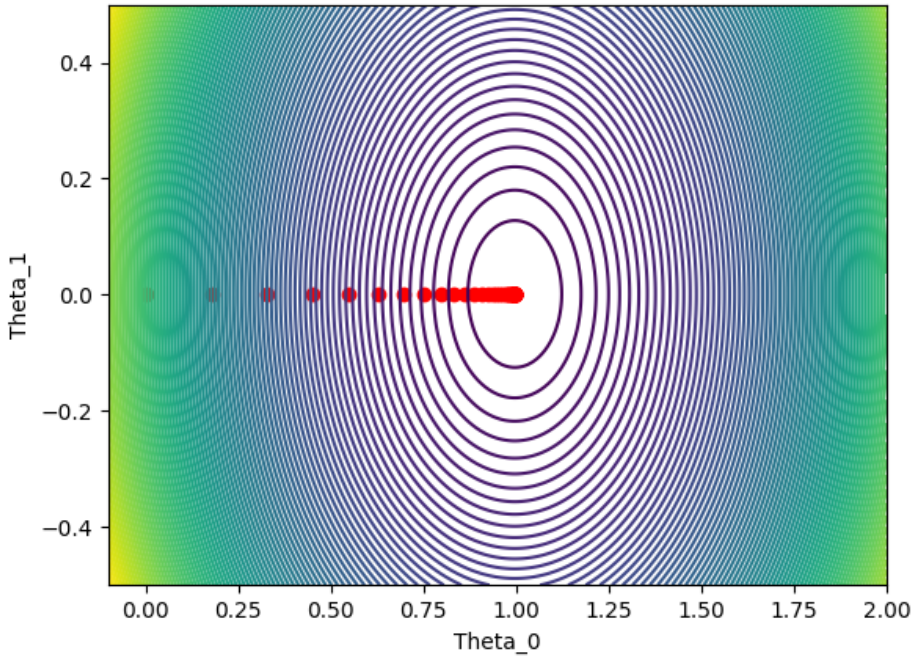

Figure 3: contour for eta = 0.001

e) As I increase the learning rate, number of iterations decreases and theta takes bigger jump towards the minima at each iterations. Number of iterations for respected θ is

1. 0.001 = 13807 (lowest speed)
2. 0.025 = 611
3. 0.1 = 155 (highest speed)

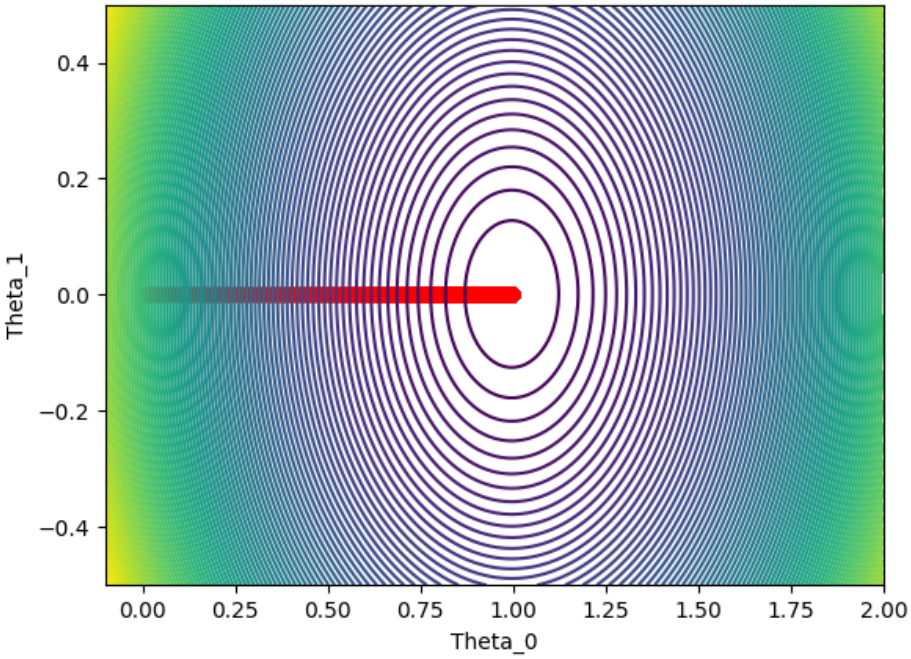

Figure 4: contour for eta = 0.001

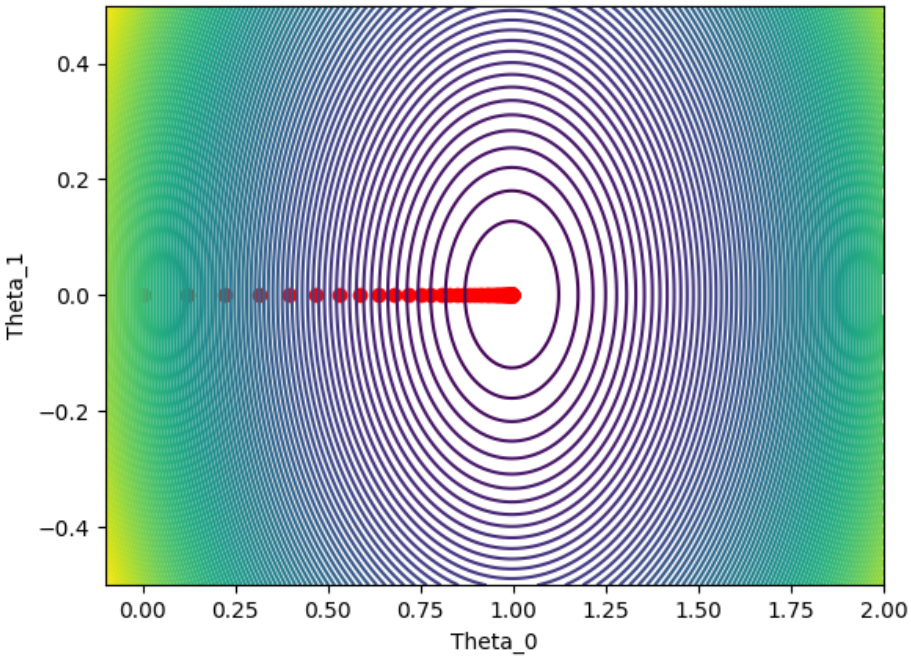

Figure 5: contour for eta = 0.025

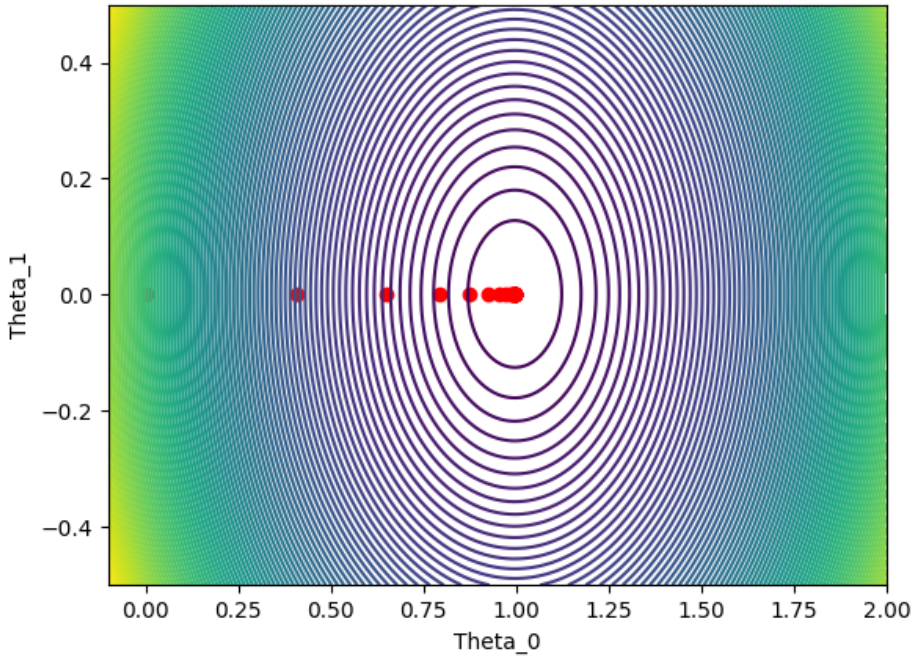

Figure 6: contour for eta = 0.1

2 Sampling and Stochastic Gradient Descent

a) I used the numpy random normal function for generating random numbers around the given mean and variance and added these numbers to $\theta^t X$ to get the Y data for stochastic gradient descent.
b) Convergence criteria was taken from Andrew NG lectures that the average error calculated in last 1000 iterations - last calculated average error ¡ ϵ, then the process stops. last calculated average is updated on every 1000 iterations.
The value of theta obtained for different batch sizes is.

1. **b=1** = [3.245855630.789307272.11404194]
2. **b=100** = [3.010569811.004434192.00679563]
3. **b=10000** = [2.999265531.001924452.00071191]
4. **b=1000000** = [2.99913711.00076322.00000214]

c) No, different algorithms doesn't converge to the same parameter values as according to gradient noise in change in thetas changes and thus convergence also changes.
These values are very similar to the original values and deviates with difference less than 0.01. With smaller batch size deviation is seen more as theta changes smoothly for larger batch sizes.
Speed for smaller batch size is smaller due to faster calculation but iterations decreases as batch size increases as the change in theta vector comes closer to the normal and thus change is more

towards the minima point.
The number of iterations for different batch sizes is.

1. **b=1** = 62000 iterations
2. **b=100** = 7000 iterations
3. **b=10000** = 3000 iterations
4. **b=1000000** = 3000 iterations

The error values for different theta values are

1. **original** θ = 0.9829469215
2. **b=1,**θ = [3.245855630.789307272.11404194] = 3.8737876864877037
3. **b=100,**θ = [3.010569811.004434192.00679563] = 0.9850033307642443
4. **b=10000,**θ = [2.999265531.001924452.00071191] = 0.98294443009428
5. **b=1,**θ = [2.99913711.00076322.00000214] = 0.9829336565439751,

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.
d) The plots for different batch sizes is as follows.

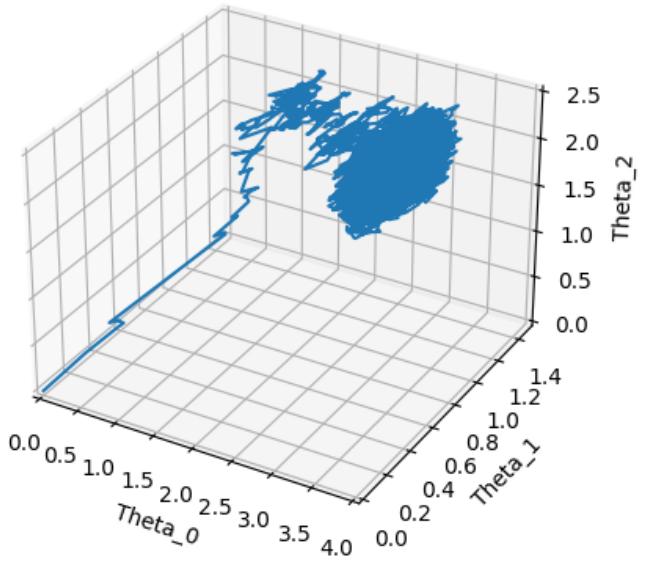

Figure 7: Variation of theta for batch size =1

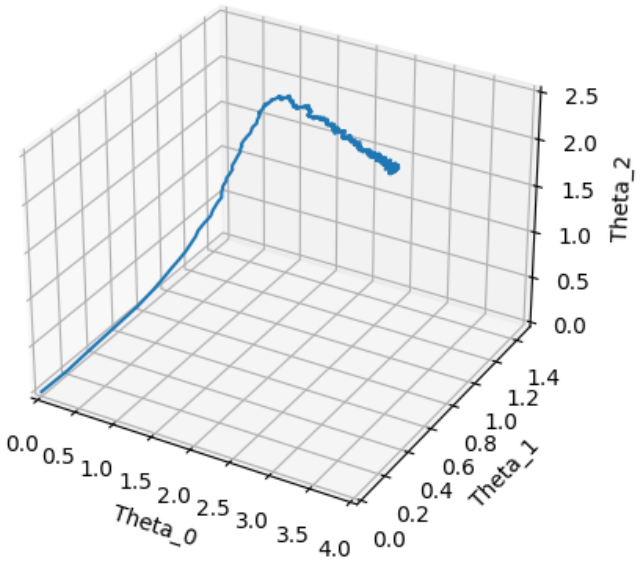

Figure 8: Variation of theta for batch size =100

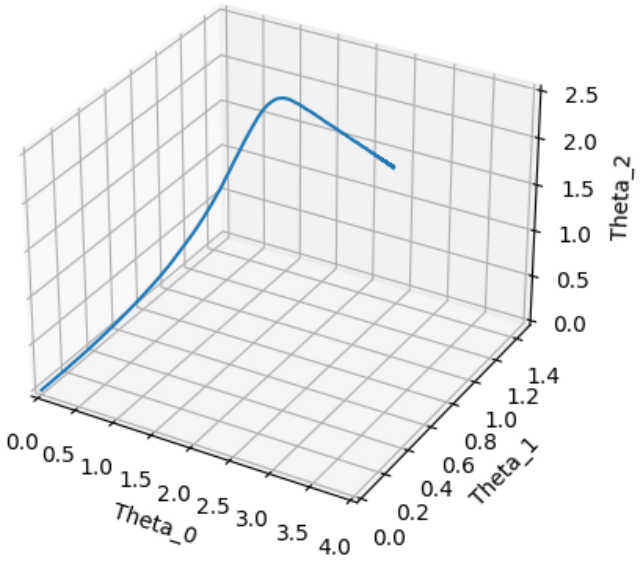

Figure 9: Variation of theta for batch size =10000

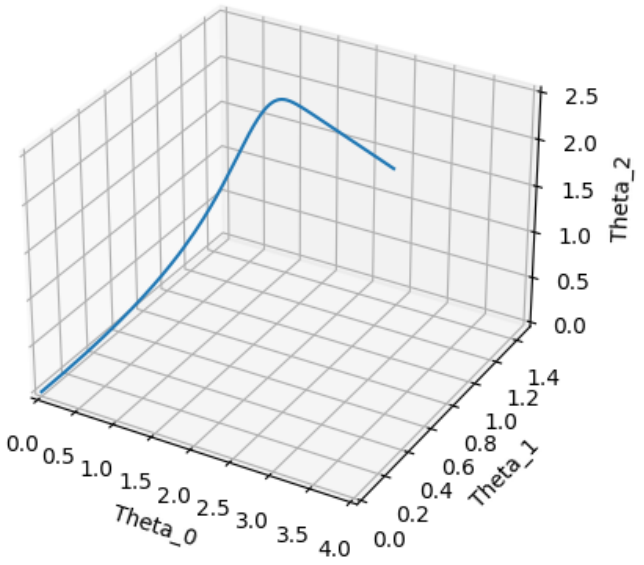

Figure 10: Variation of theta for batch size =1000000

The movement is more smooth for larger batch sizes. Yes, this makes sense as for larger batch sizes change in theta is along the gradient and thus directed towards the minima.

3 Logistic Regression

a) Newton's method is

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

$$\begin{aligned}
\mathcal{H}(LL(\theta)) &= \frac{\partial\left(\nabla_\theta(LL(\theta))\right)}{\partial\theta} \\
&= \frac{\partial}{\partial\theta}\left(X^T\left(Y - \frac{1}{1+e^{-X\theta}}\right)\right)^T \\
&= \begin{pmatrix} \frac{\partial}{\partial\theta_0}\left(X^T\left(Y - \frac{1}{1+e^{-X\theta}}\right)\right)^T \\ \frac{\partial}{\partial\theta_1}\left(X^T\left(Y - \frac{1}{1+e^{-X\theta}}\right)\right)^T \\ \vdots \\ \frac{\partial}{\partial\theta_n}\left(X^T\left(Y - \frac{1}{1+e^{-X\theta}}\right)\right)^T \end{pmatrix}
\end{aligned} \tag{1}$$

Now, computing each row separately, we get:

$$\begin{aligned}\mathcal{H}_i &= \frac{\partial}{\partial\theta_i}\left(X^T\left(Y-\frac{1}{1+e^{-X\theta}}\right)\right)^T \\ &= \frac{\partial}{\partial\theta_i}\left(\left(\frac{1}{1+e^{-X\theta}}\right)^T X\right) \\ &= \frac{\partial(X\theta)^T}{\partial\theta_i}\frac{\partial\left(\frac{1}{1+e^{-X\theta}}\right)^T}{\partial(X\theta)^T}X \\ &= X_i^T\left(\frac{e^{-(X\theta)^T}}{\left(1+e^{-(X\theta)^T}\right)^2}\right)X\end{aligned} \tag{2}$$

The hessian for the equation turns out to be after calculation.

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

Gradient is calculated directly using

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

$x^{(i)}x^{(j)}$ is calculated using outer product function in numpy.
The value of θ obtained is [0.40125316 2.5885477 -2.72558849]
c) The plot obtained is

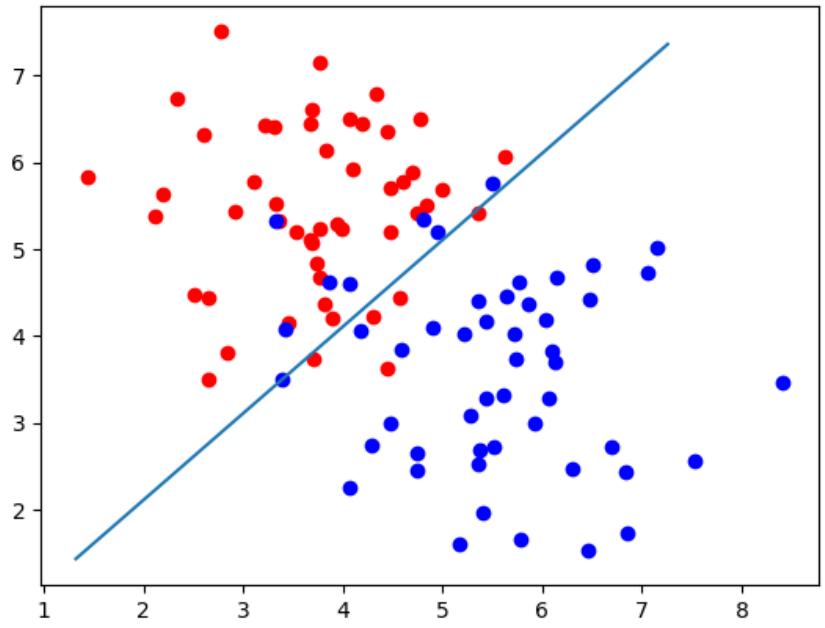

Figure 11: logistic regression plot is

4 Gaussian Discrmimant Analysis

a) The values of the means μ_0, μ_1 and the co-variance matrix Σ are computed as mentioned in the class. The values computed by the algorithm are

$$\phi = 0.5$$
$$\mu_0 = [-0.75529433, 0.68509431]$$
$$\mu_1 = [0.75529433, -0.68509431]$$
$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

b) The plot obtained is:-

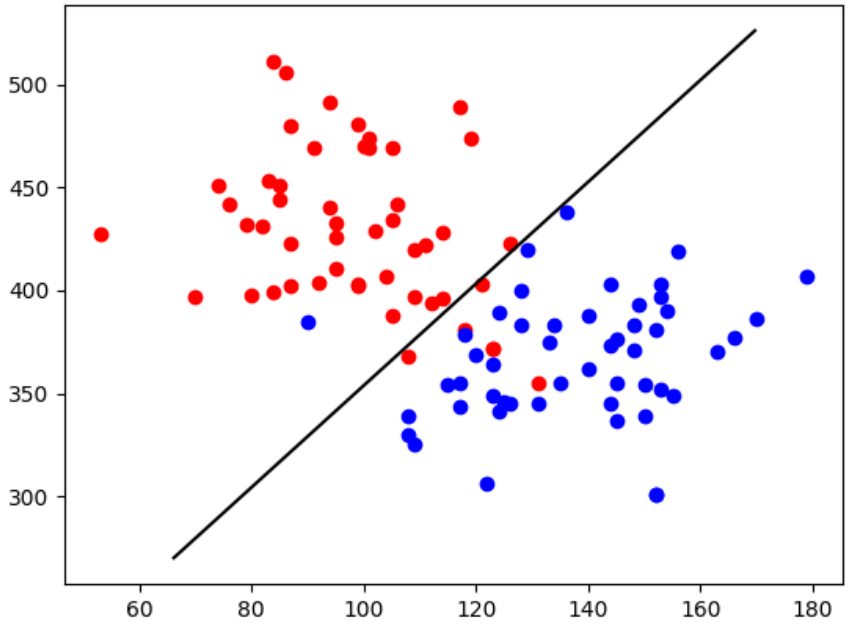

Figure 12: plot with linear separator

c) Equation for linear boundary is

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

This is converted to the form $\theta^T X$:-

d) The values obtained are:-

$$\phi = 0.5$$

$$\mu_0 = [-0.75529433, 0.68509431]$$

$$\mu_1 = [0.75529433, -0.68509431]$$

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

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

e) The equation for the quadratic boundary is:-

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

Used a single contour of the left expression in a 2D plot which corresponds to those values of x which satisfy the above equation. The expansion of the expression of the above expression was done to convert vector form into scalar form. Scalar form was used to get the f(x_1, x_2)=0 where the function is the left expression and x_1, x_2 are the parameteres.

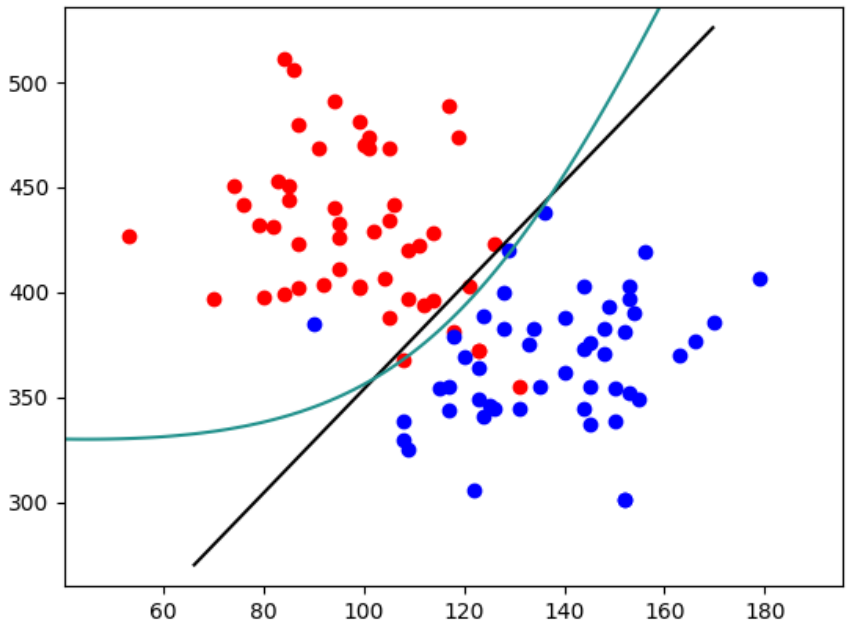

Figure 13: plot with quadratic separator

f) The plots show that both appear to be effective separators, but because of the quadratic nature of the separator, the GDA model with various covariance matrices assumes that the Alaska salmon features have a "round" boundary. Additionally, the fit to shape doesn't appear to generalise

well to additional data because, despite the addition of one additional parameter to the model, the proportion of misclassified points hasn't decreased as much as we might have expected. Thus we can say that was modeled for linear separation.