COL774
Assignment 1

Vaibhav Seth

1st September 2023

1 Linear Regression

1.a Batch Gradient Descent

On performing *batch gradient descent* with $learning - rate = 0.05$ and $stopping - threshold = 10^{-15}$, and stopping criteria the absolute difference between consecutive losses, the following parameters are learned (first is the intercept term $theta_0$):

$$\theta = \begin{pmatrix} 0.99661997 \\ 0.0013402 \end{pmatrix} \tag{1}$$

1.b Regression Plot

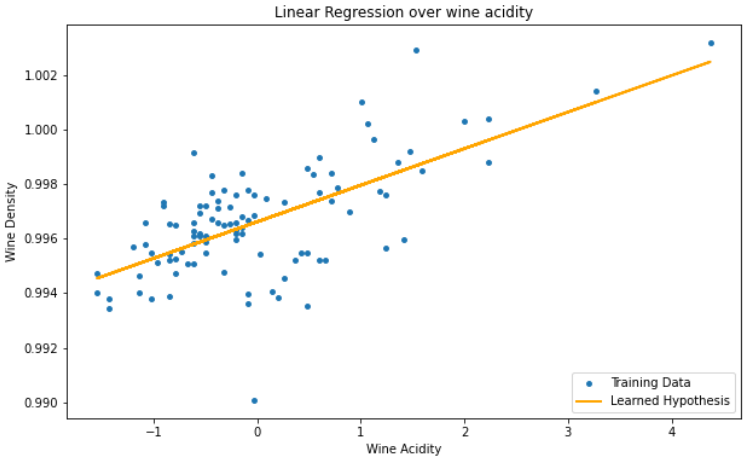


Figure 1: Training Data and Learned Hypothesis plot for Q1

1.c Loss function and change of θ

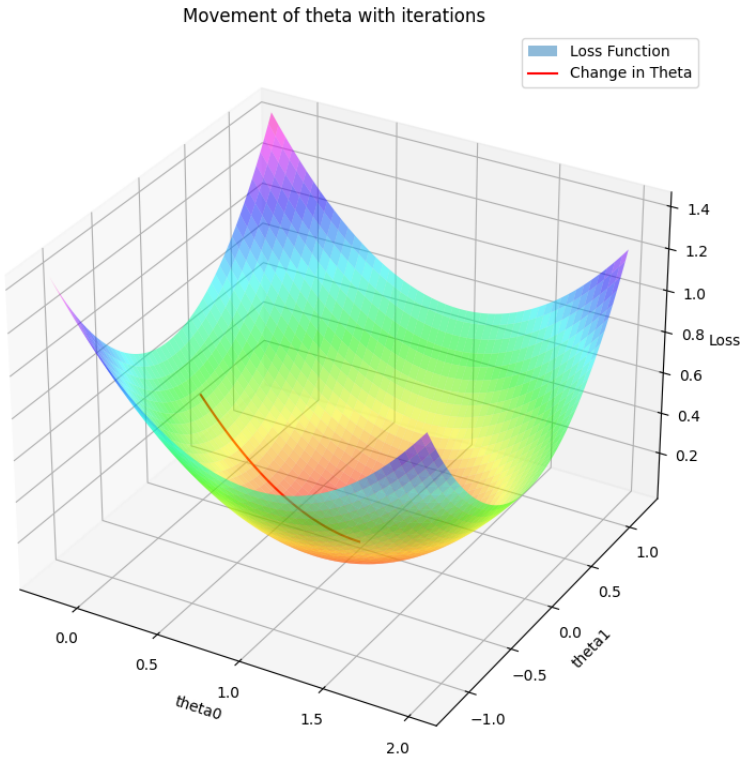


Figure 2: Mesh of Cost Function and Movement of θ

The learning takes 309 iterations and $0.011ms$.

1.d Contour plots

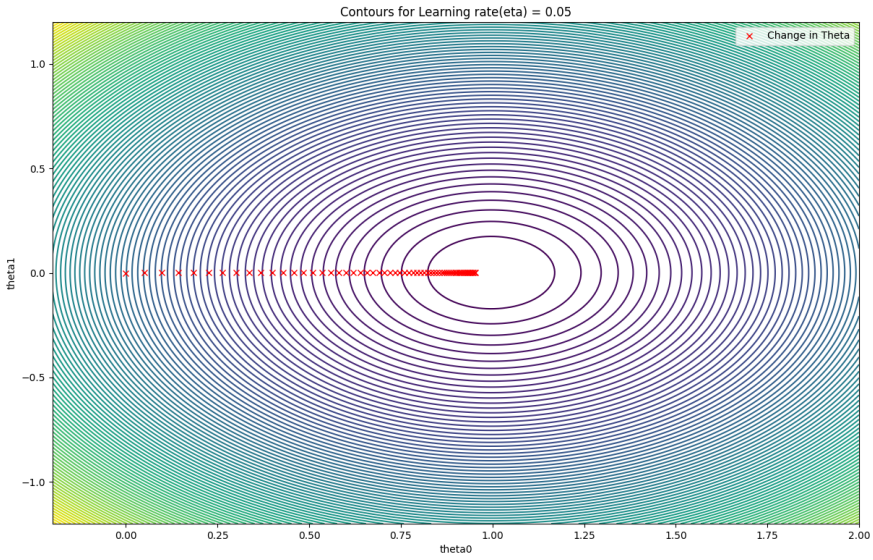


Figure 3: Movement of θ over Contours

1.e Experimenting with varying η

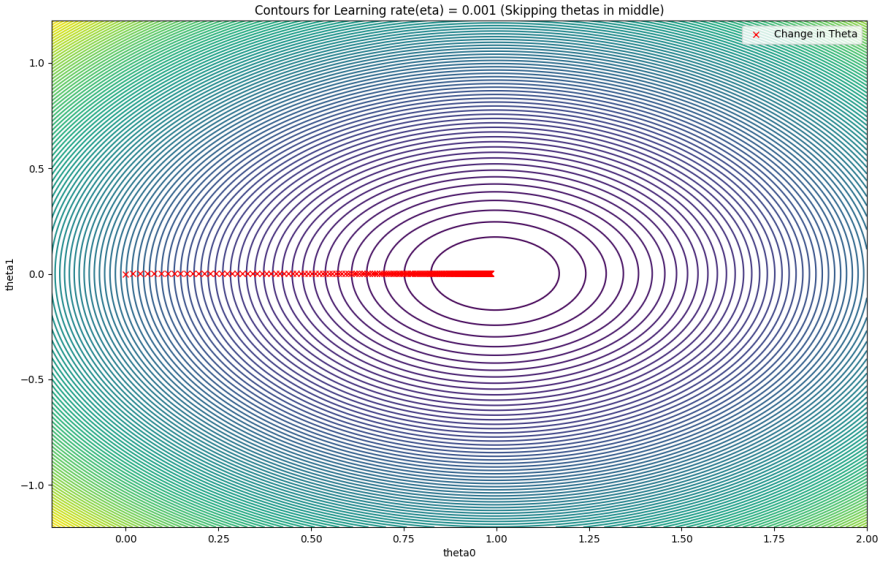


Figure 4: Movement of θ for learning rate = 0.001

For the above plot, some datapoints were left to decrease the animation time

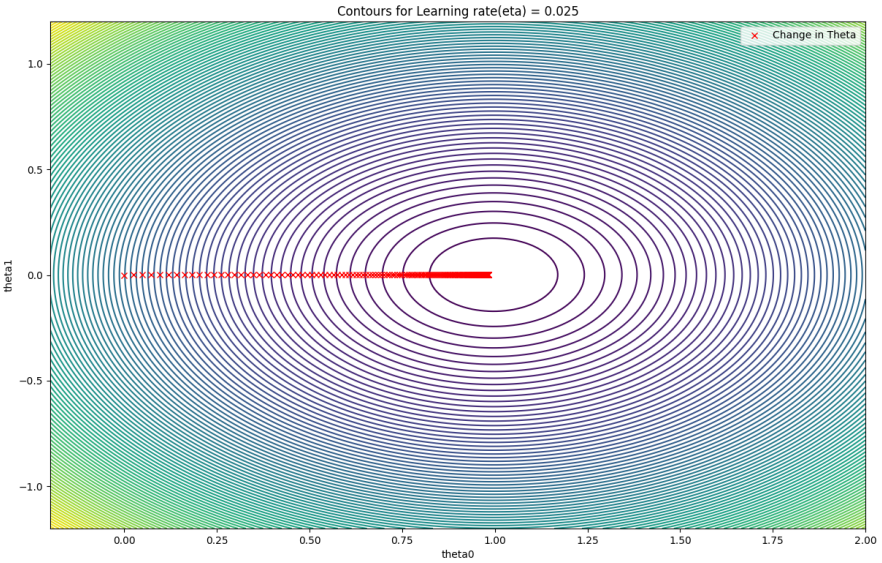


Figure 5: Movement of θ for learning rate = 0.025

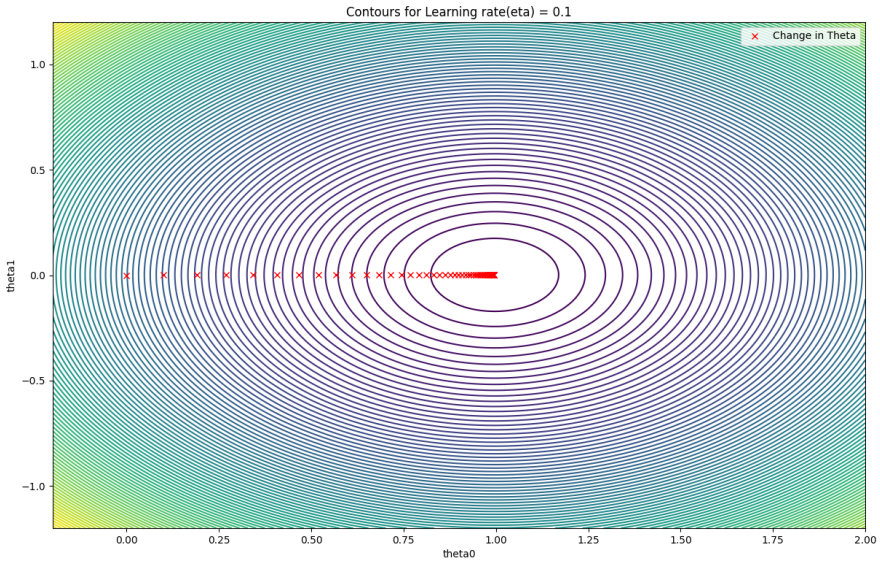


Figure 6: Movement of θ for learning rate = 0.01

It can be observed that the order of gradient step is as follows : $0.001 < 0.025 < 0.01$
This is to be expected as the size of gradient step is decided by the learning rate size. As a result, we can see that their are a fewer gradient steps for larger learning rates as compared to lower learnign rates.

2 Sampling and Stochastic Gradient Descent

2.a Sampling

The data is sampled from a multivariate normal distribution using numpy.random.multivariate_normal, using a mean and covariance matrix as follows:

$$\mu = \begin{pmatrix} 3 \\ -1 \end{pmatrix} \tag{2}$$

$$cov = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \tag{3}$$

$$\begin{aligned} x &\sim \left(\mathcal{N}(\mu, cov)\right) \\ y &\sim \theta^T x + \mathcal{N}(0, 2) \end{aligned} \tag{4}$$

1 million samples are generated using the above sampling criteria.

2.b Stochastic Gradient Descent

We use change in average loss(for an epoch) as the convergence criteria

$$|\underset{\text{epoch } t+1}{averageLoss}(\theta^{t+1}) - \underset{\text{epoch } t}{averageLoss}(\theta^t)| \leq \text{stopThresh, where stopThresh} = 10^{-7} \tag{5}$$

The following parameters are learnt for different batch sizes:

Batch size	θ	**Iterations**	**Time taken**
1	$\begin{pmatrix} 3.02319698 \\ 0.97836492 \\ 2.02194238 \end{pmatrix}$	3000000 (3 epochs)	84.4245 seconds
100	$\begin{pmatrix} 2.99889159 \\ 1.00047573 \\ 2.00158016 \end{pmatrix}$	70000 (7 epochs)	2.2594 seconds
10000	$\begin{pmatrix} 2.99307483 \\ 1.00172197 \\ 1.99925055 \end{pmatrix}$	46600 (466 epochs)	6.7231 seconds
1000000	$\begin{pmatrix} 2.99307483 \\ 1.00172197 \\ 1.99925055 \end{pmatrix}$	29490 (29490 epochs)	500.3315 seconds

2.c Test Error

The errors on the data set provided, for different batch sizes are:

Batch size	**Error**
1	1.029359
100	0.982846
10000	0.983171
1000000	0.991238
original θ	0.982947

Different Algorithm Convergence and Parameter Values:

1) Various algorithms with different batch sizes lead to the convergence of different parameter values.

2) This variation in parameter values is a direct result of stochastic gradient descent, where batches differ due to random shuffling.

3) Consequently, different θ values result in distinct errors on the test set.

4) Batch sizes of 100 and 10,000 demonstrate the best predictions and achieve rapid convergence.

Optimal Batch Size for Speed of Convergence:

1) The speed of convergence is highest when using moderately small batch sizes.

2) Smaller batch sizes are computationally efficient compared to larger ones.

3) However, excessively small batch sizes suffer from a drawback - they fail to represent the entire data-set adequately.

4) Consequently, θ doesn't converge directly along the contours but progresses obliquely, slowing down convergence.

5) An optimal choice is to use a slightly larger batch size that is still relatively small.

Impact of Larger Batch Sizes on Convergence:

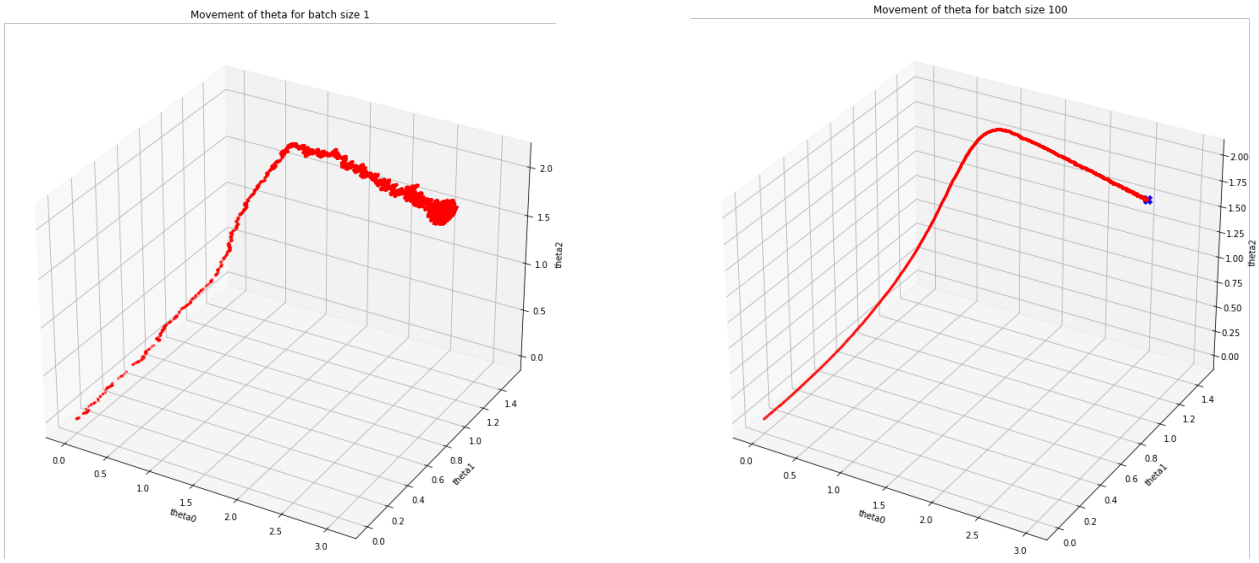
1) Larger batch sizes lead to significantly larger steps in θ toward the optimal value.

2) However, each step comes at a higher computational cost.

3) This direct step toward convergence can cause early convergence as θ's change diminishes near the optimal value.

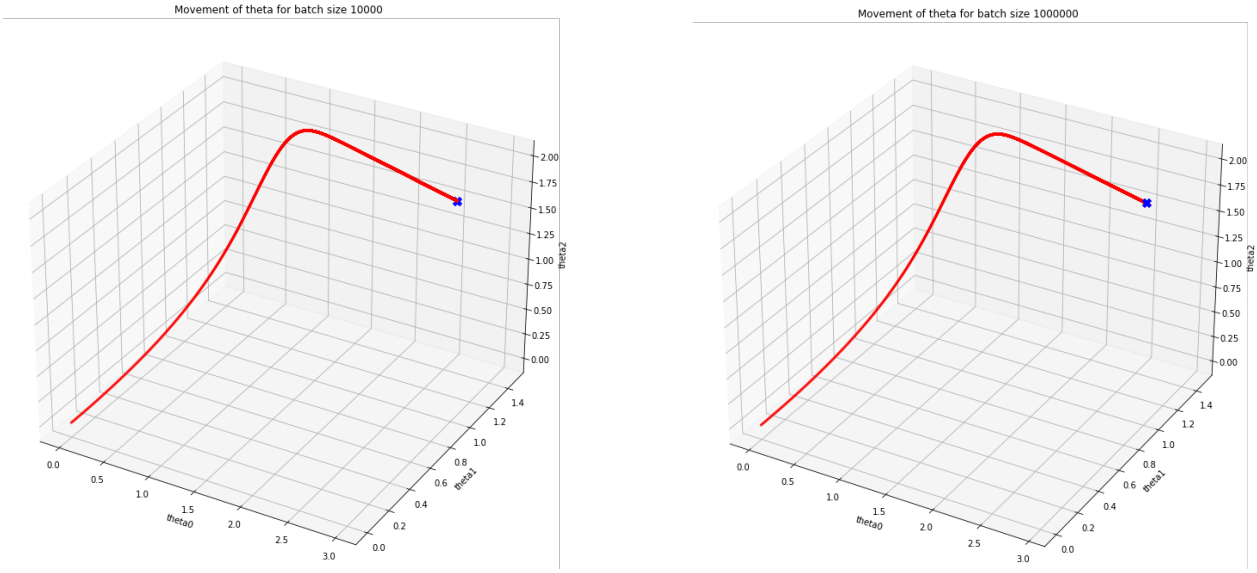
4) A smaller ε (epsilon) is required when using larger batch sizes to achieve convergence to the same value due to reduced alterations in the cost function.

2.d Movement of θ in the θ Space



(a) $r = 1$ (b) $r = 100$

Figure 7: Movement of θ for different batch sizes



(a) $r = 10000$ (b) $r = 1000000$

Figure 8: Movement of θ for different batch sizes

The following points should be noted:

1) The behavior of θ in each of the four scenarios corresponds to the explanation provided in Section 2.c.

2) When r is equal to 1, θ shows significant fluctuations near the minimum value, causing a slowdown in the convergence process.

3) In contrast, for r values of 100 and 10,000, θ moves more smoothly, leading to faster convergence.

4) In the case where θ equals 1,000,000, the convergence criteria are met well before θ gets close to the optimal value.

5) This discrepancy occurs because consecutive changes in θ result in a smaller impact on the cost function when the contours are more widely spaced as we approach the optimal point.

3 Logistic Regression

3.a Newton's Method

The *Gradient Vector* for Logistic Regression :

$$\nabla_\theta(LL(\theta)) = X^T\left(Y - \frac{1}{1+e^{-X\theta}}\right) \tag{6}$$

The complete *Hessian matrix* for logistic regression:

$$\mathcal{H} = X^T Diag\left(\frac{e^{-(X\theta)^T}}{\left(1+e^{-(X\theta)^T}\right)^2}\right) X \tag{7}$$

$$\text{where } Diag = \text{Diagonal matrix with sigmoid*(1-sigmoid) for all training example as entries} \tag{8}$$

We can now perform iterations as follows:

$$\theta^{t+1} \leftarrow \theta^t - \mathcal{H}^{-1}\nabla_\theta(LL(\theta)) \tag{9}$$

On learning the model using *Newton's method*, we get the following θ in 8 iterations with a Stopping Threshold of 10^{-20}:

$$\theta = \begin{pmatrix} 0.40125316 \\ 2.5885477 \\ -2.72558849 \end{pmatrix} \tag{10}$$

3.b Regression Plot

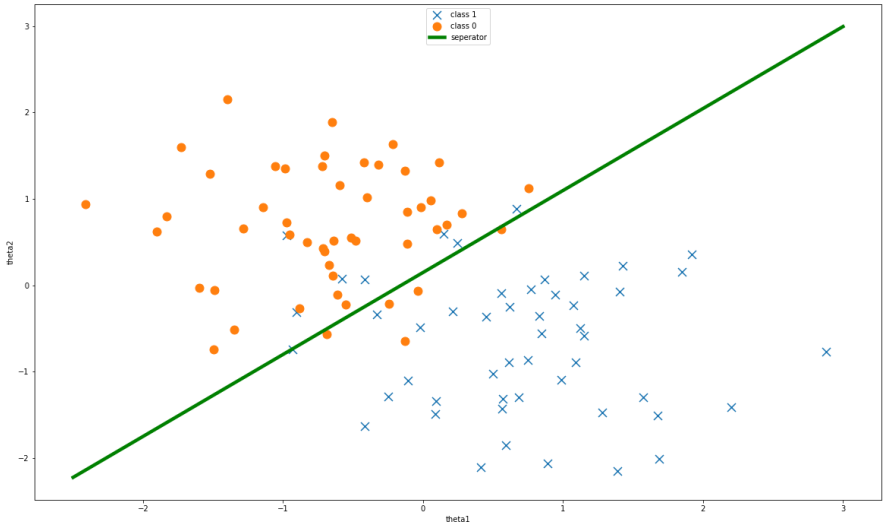


Figure 9: Plot of the data along with the separator

4 Gaussian Discriminant Analysis

4.a Linear GDA

Assigning 1 to Alaska and 0 to Canada 0, we arrive at the following parameters, (Θ), for *linear GDA*:
*Note that μ_0 and μ_1 are the same

$$\begin{aligned} \phi &= 0.5 \\ \mu_0 &= \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \\ \mu_1 &= \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \\ \Sigma &= \begin{pmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{pmatrix} \end{aligned} \tag{11}$$

4.b Training Data Plot

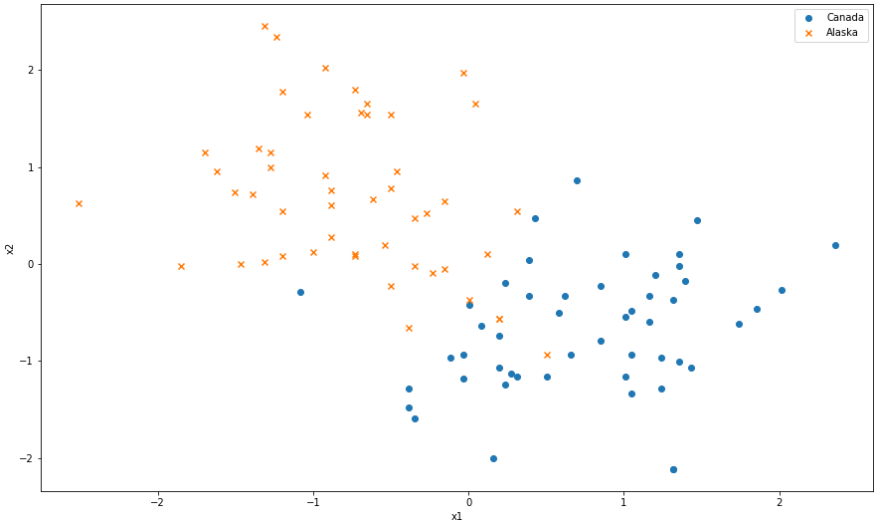


Figure 10: Plot of training Data

4.c Linear Separator Plot

The separator is given by:

$$\log\left(\frac{1-\phi}{\phi}\right) - (\mu_1 - \mu_0)^T \Sigma^{-1} x + \frac{1}{2}\left(\mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0\right) = 0 \tag{12}$$

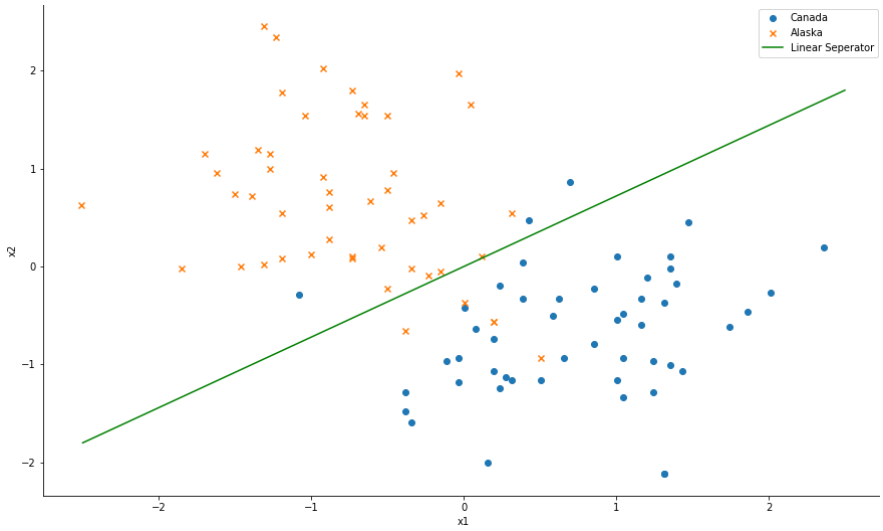


Figure 11: Plot of Linear Seperator

4.d Generic (Quadratic) Separator

The following parameters (Θ) are obtained for *generic GDA*:

$$\begin{aligned}
\phi &= 0.5 \\
\mu_0 &= \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \\
\mu_1 &= \begin{pmatrix} 0.75529433 \\ -0.68509431 \end{pmatrix} \\
\Sigma_0 &= \begin{pmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{pmatrix} \\
\Sigma_1 &= \begin{pmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{pmatrix}
\end{aligned} \tag{13}$$

4.e General Separator Plot

The equation general separator is obtained by taking P(y = 1—x) = P(y = 0—x) and taking logarithm:

$$\log\left(\frac{1-\phi}{\phi}\sqrt{\frac{|\Sigma_1|}{|\Sigma_0|}}\right) + \frac{1}{2}\left(x^T(\Sigma_1^{-1} - \Sigma_0^{-1})x - 2(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})x + (\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0)\right) = 0 \tag{14}$$

The plot of the data along with the linear and generic separator is:

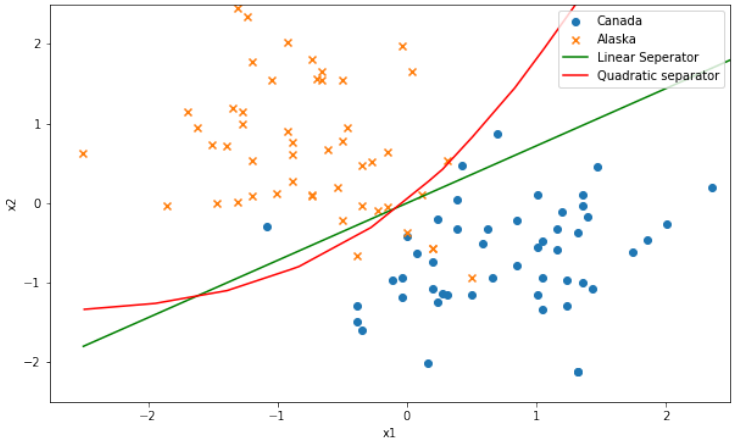


Figure 12: Plot of the data along with the separators

4.f Analysis of the Seperators

1) The quadratic separator only marginally improves the correct classification of approximately three more points when compared to the linear separator.

2) Despite the quadratic separator's appearance of favoring the classification of Canadian salmon over Alaskan salmon due to its bending toward Alaska, this is not reflected in the actual training data.

3) Upon closer examination, it becomes apparent that the quadratic separator tends to overfit the test data, meaning it doesn't provide significantly better results when applied to the test dataset.

4) This is usually observed when the number of parameters are higher for a smaller dataset. The model will try to overfit.