Linear Regression

(a) Final parameters learnt (after normalizing the data):

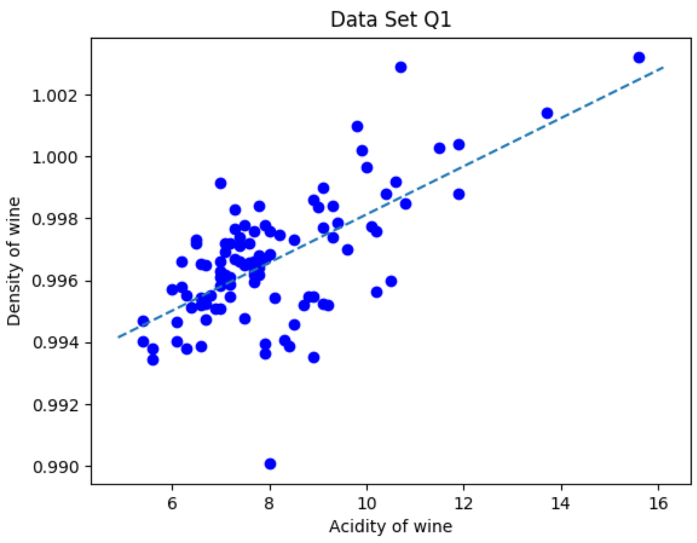
$$\theta = [0.00867704,\ 0.65805745]$$

learning_rate = 0.05

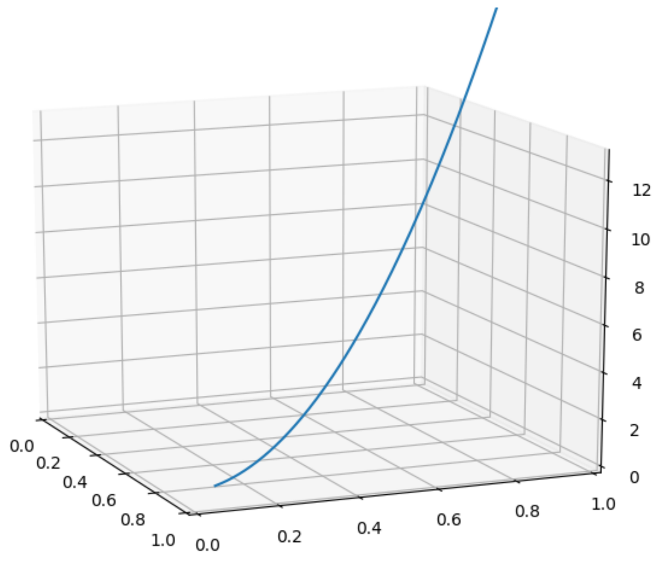
Stopping criteria: cost_difference < 1.6855447430592224e-8

After 284 iterations, the parameters converges, but slope is learnt in 102 iterations. Thus fixing the slope can make our parameters converge much faster.

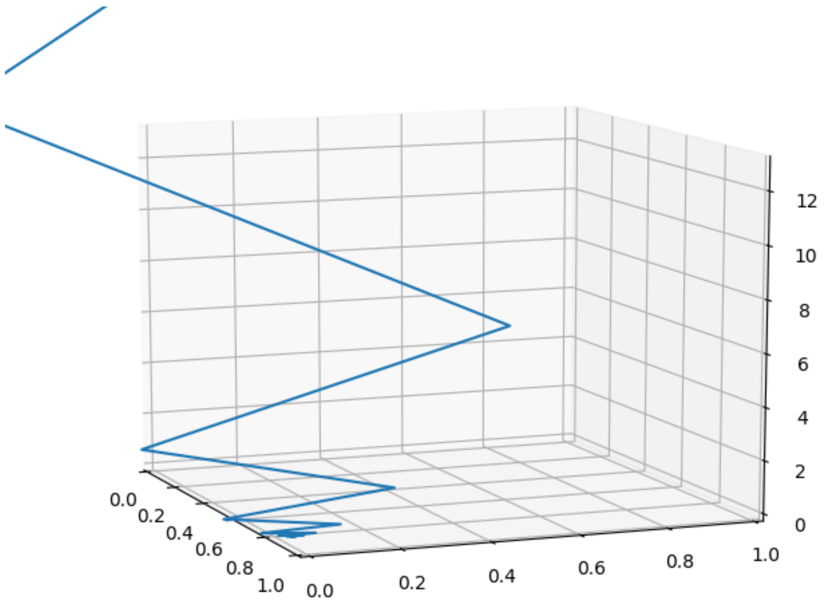
(b)



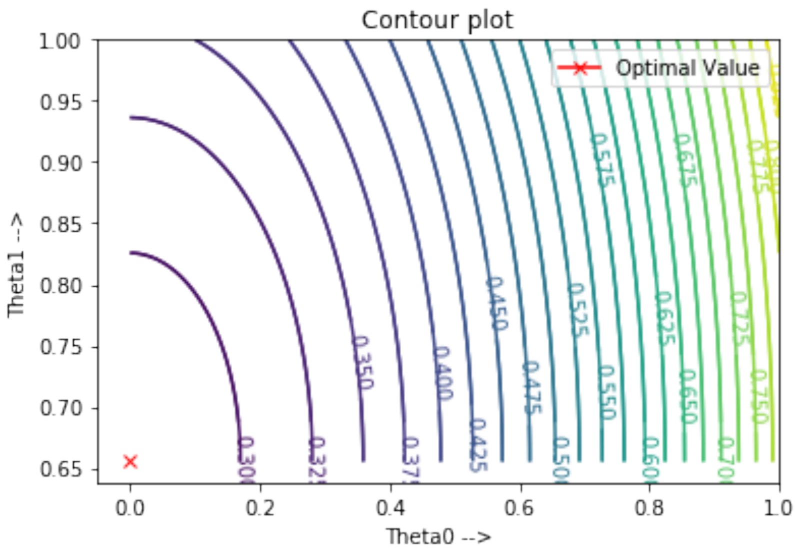
(c) This is the case of constant learning_rate throughout the algorithm

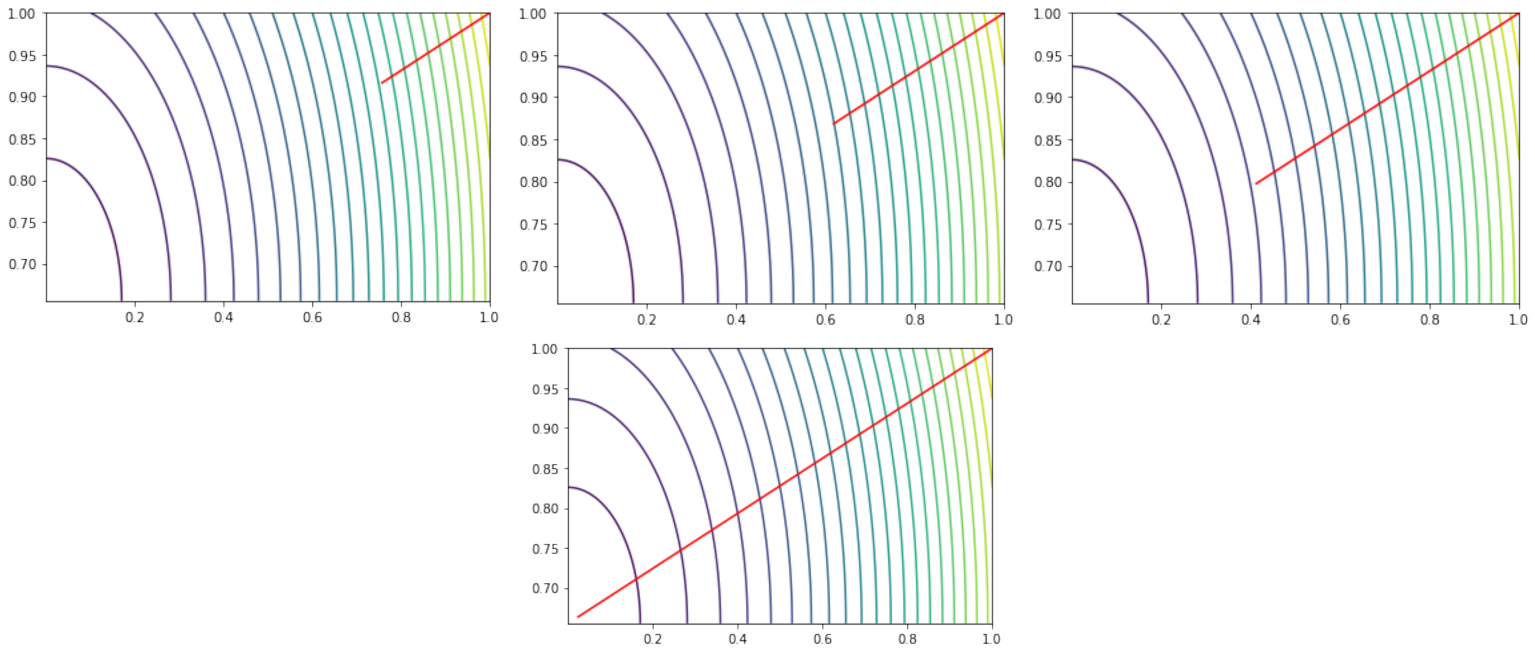


This is the case with higher learning_rate in the start of algorithm, i.e. 0.05 else 0.0004 for rest of the iterations.



(c)



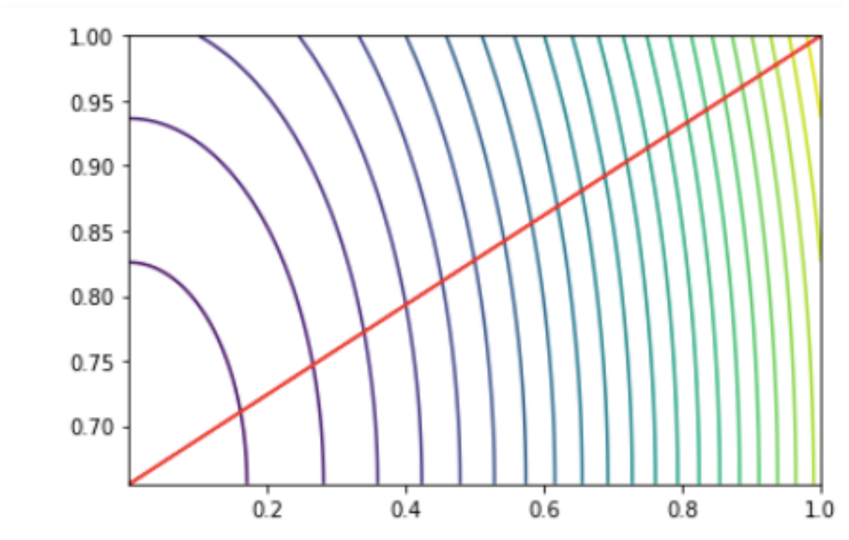


(d) Parameters learnt at each :

$\theta_{0.001} =$ [0.00075239 0.65532395] (converged in 10409 iterations)
$\theta_{0.025} =$ [0.00074298 0.65532071] (converged in 543 iterations)
$\theta_{0.1} =$ [5.18853073e-04 6.55243397e-01] (converged in 148 iterations)

The contour plot was much slower for low learning rate. Also, the contour plot was same in shape for the three learning_rates mentioned. This behaviour was expected because more lr means more oscillations.

This is the contour for $\eta = 0.1$:-



```
Iteration : 146
It converges in 148 iterations
```

Sampling and Stochastic Gradient Descent

(a) Sampling of 1 M data points was done with $\theta = [3\ 1\ 2]$.

(b) Relearning θ through SGD, with constant learning_rate = 0.001

batch_size	cost_threshold	batch_threshold
1	1E-04	< 1 epochs
100	1E-04	< 2 epochs
10,000	2.1E-05	12 epochs
10,00,000	2.5E-06	31 epochs

The table above shows different converging conditions for each batch_size.

(c) **Yes**, in practice, for different batch_size, parameters converge to different values if stopping conditions are same. If stopping conditions are as above, then parameters are converging to pretty close values.

Speed of convergence, with smaller batch_size comes out be vary high as compared to larger batch_size. For batch_size = 1, parameters converge to the true value on 2200 iterations of round robin itself, i.e. it didn't even need to go through 1 epoch.

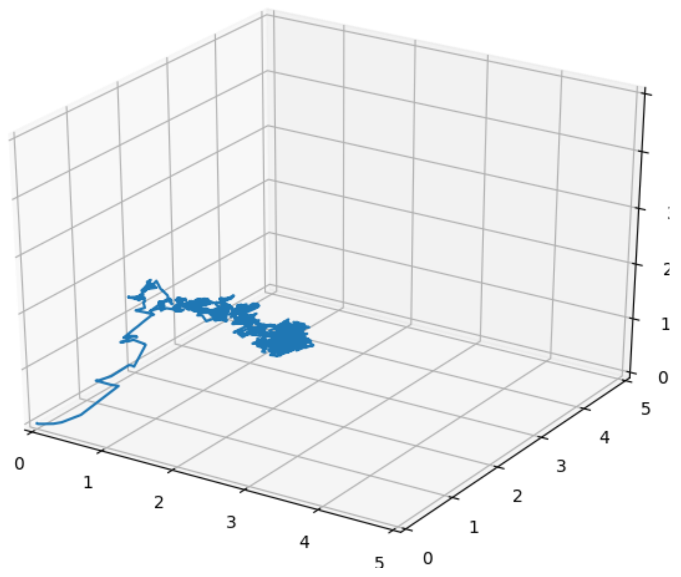
Ideally we should have a higher learning_rate for greater batch_size and lower learning_rate for smaller batch_size.

With such a slow learning_rate, higher batch_size algorithm will require high number of epochs to converge.

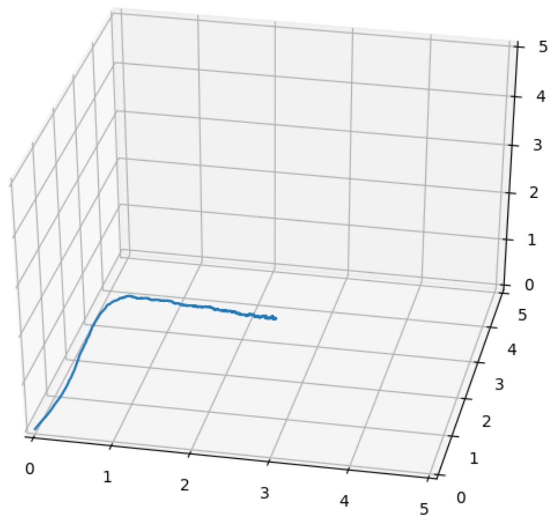
The provided learning rate(0.001) is very small even for batch_size = 1. Because of low lr, not able to see much randomness in the path of sgd even with batch_size = 1. Experimentally, it should have been 0.01 for faster convergence.

(d) Plotting 3-D parameter space, shape of the plots for different batch_size differ alot.

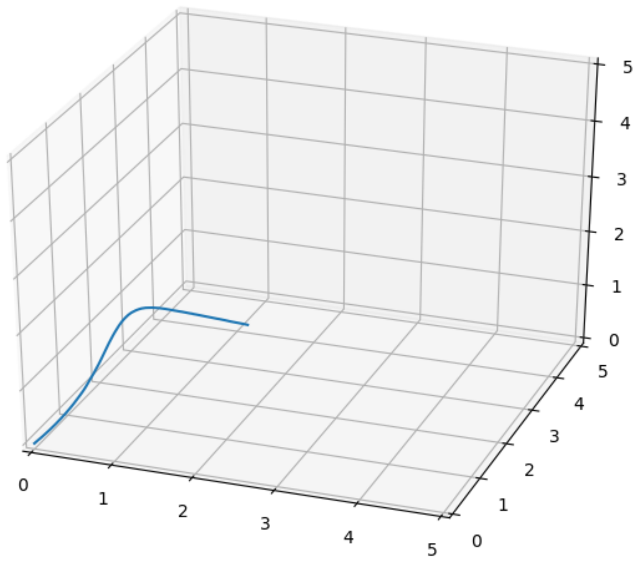
batch_size = 1



batch_size = 100



batch_size = 10,000



batch_size = 1000000

Not available since it was computationally difficult to render and calculate at the same time.

This is the example of convergence with batch_size = 10,000

```
epoch 1
[0.03759595 0.1497441  0.04054682]
epoch 2
[0.88569435 1.44818183 1.80849284]
epoch 3
[1.38722398 1.35071981 1.88137666]
epoch 4
[1.76976129 1.26736337 1.90998581]
epoch 5
[2.06154862 1.20361645 1.93131348]
epoch 6
[2.28411493 1.15498925 1.94757249]
epoch 7
[2.4538816  1.11789788 1.9599742 ]
epoch 8
[2.58337435 1.08960574 1.96943384]
epoch 9
[2.68214739 1.06802537 1.97664935]
epoch 10
[2.74328816  1.1051288 1.9563122 ]
epoch 11
[2.88974312 1.09222574 1.98843384]
epoch 12
[2.98212735 1.06806126 1.98162935]
```

Logistic Regression - Newton's Method

In this sub-part, I calculated the Hessian and first derivative of the Logarithmic Loss given as $L(\theta) = \sum y^i log h_\theta(x^i) + (1-y^i) log(1-h_\theta(x^i))$.

(a) Upon performing iterative Newton's method given as:

$\theta_{k+1} = \theta_k - H^{-1}X^T(\pi_k - y)$ where $H_k = X^T S_k X$

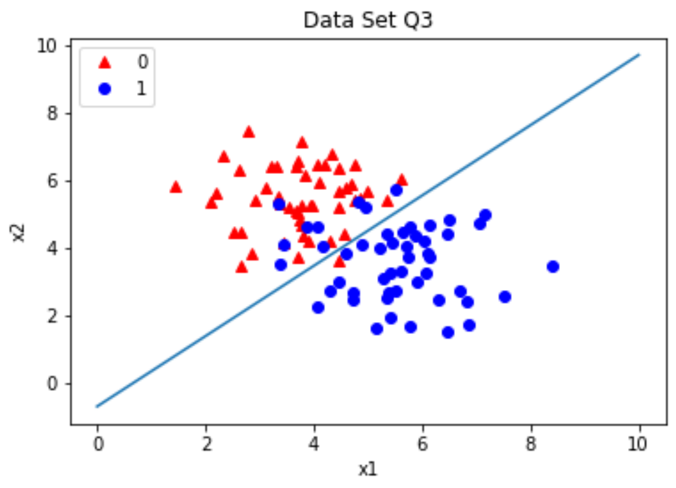
$S_k = diag(\pi_{1k}(1-\pi_{1k}), \ldots \pi_{nk}(1-\pi_{nk}))$

Convergence condition on the difference of theta's are set as convergence_condition = $1.1e-10$. Final parameters were :

$\theta = [0.76311601 \quad -0.73164417 \quad -0.51600465]$

This was reported at 9th iteration.

(b) The decision boundary mady using the following algorithm is:



$h_\theta(x)$ > 0.5 on the right side (Class 0) of the plot and < 0.5 on the left side (Class 1).

Gaussian Discriminative Analysis

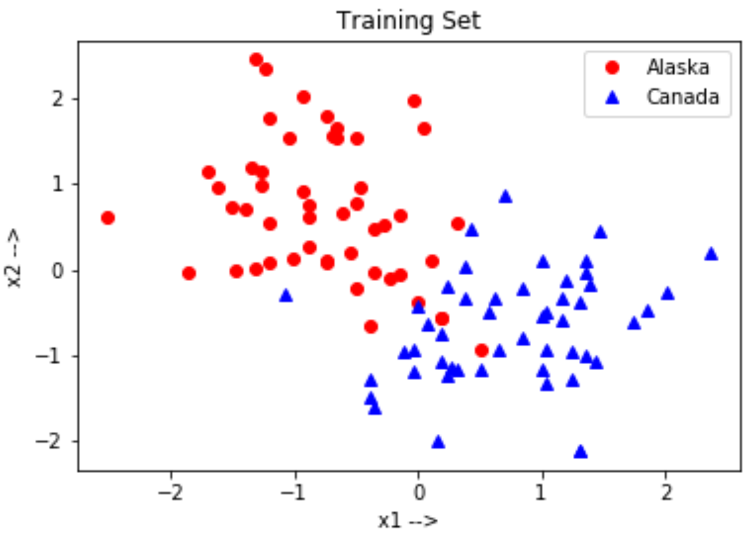
I load the data from q4x.dat and q4y.dat. Mean normalised the X data accordingly. Replaced 'Alaska' and 'Canada' to Class 0 and Class 1, for binary classification. I defined functions to calulate sigma, mu, sigma.

(a) Using the closed form equations and assuming Σ0 = Σ1 = Σ, I found out the values of means, μ0 and μ1, and the co-variance matrix Σ as shown on the right.

```
 μ0 = [-0.75529433, 0.68509431]

μ1 = [ 0.75529433, -0.68509431]

Σ = [[ 2.93378037 -0.04494456]
```



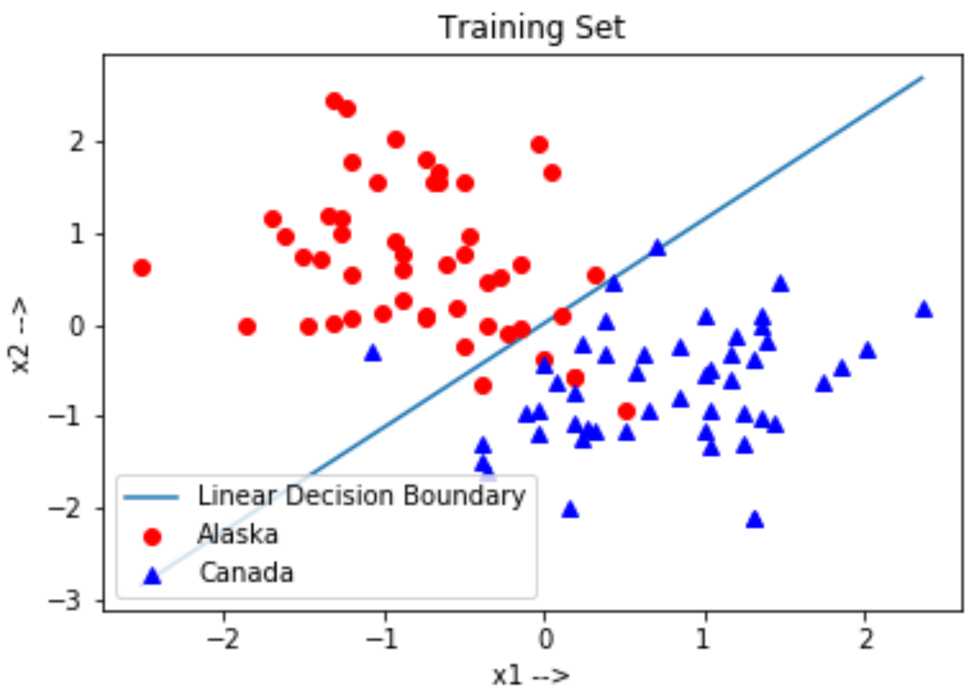
(b) Following is the plot of training data of input features as x1 and x2, showing the corresponding class (Alaska or Canada). This is the plot after normalisation.

(c) Since the covariance matrices are same for p(x|y=0) and p(x|y=1), thus a linear decision boundary is expected. The equation of boundary is given as:

$$x = [x1 \ x2]^T$$

$$ln(\frac{1-\phi}{\phi}) = x^T\Sigma^{-1}(\mu 1 - \mu 0) + \frac{(\mu 0^T\Sigma^{-1}\mu 0 - \mu 1^T\Sigma^{-1}\mu 1)}{2}$$

The decision boundary fit by GDA is shown below.



(d)Here we have considered diffrerent values of Σ₁ and Σ₂ given by the closed form equations in the assignment. After implementing GDA these were the values of parameters:

$$\Sigma_0 = [\frac{2.83789898 \quad -0.30973032}{-0.30973032 \quad 3.37019376}]$$

$$\Sigma_1 = [\frac{3.02966176 \quad 0.2198412}{0.2198412 \quad 2.90182824}]$$

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

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

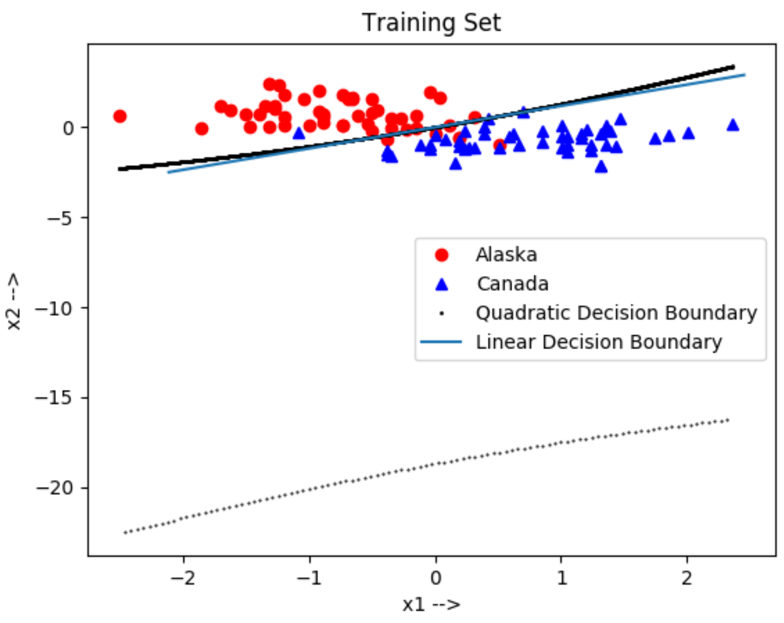
(e) The equation of decision boundary (when Σ₁ != Σ₂) is as follows:

$$2\ ln(\frac{1-\phi}{\phi}) = x^T({\Sigma_0}^{-1} - {\Sigma_1}^{-1})x + (\mu_0^T * {\Sigma_0}^{-1} - \mu_1^T * {\Sigma_1}^{-1})x + x^T({\Sigma_1}^{-1}\mu_1 - {\Sigma_0}^{-1}\mu_0)$$
$$+(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1) + log|\Sigma_0| - log|\Sigma_1|$$

Putting $x = [x \ \ y]$, we will have a quadratic equation of the form:

$$[x_1 \quad y_1]\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - [x_0 \quad y_0]\begin{bmatrix} p & q \\ r & s \end{bmatrix}\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = C$$

Thus reduced to $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$



Result (shown aproximate values because of smaller font)-

$$-0.024049x^2 + -0.11571xy + 0.0467900y^2 + -1.02867x + 0.874039y + 0.021327$$

(f) Linear decision boundary is the result of both covariance being same. The boundary passes through origin in the normalised data. In case of non-linear decision boundary we get determinant of the conic section positive, thus concluding to be a hyperbola. As a result, on plotting the quadratic boundary, this indeed comes out to be a hyperbola as we can see in the earlier figure. Both linear and non-linear decision boundaries came out to be well accurate for the given data set. However, Quadratic boundary is slighlty better for 2 or 3 training data set sample. This implies that the distribution of the data is indeed gaussian.