Machine Learning Assignment I

Vivek Singh

2019MCS2574

1 Linear Regression

1.1 Implement Batch gradient descent

- Implemented batch gradient descent by minimizing the cost function
  $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^i - h(\theta x^i))^2$

- Following are the functions used to implement Batch gradient descent

  inputData() : uses pandas to read the csv data files.

  normalize() : Normalizes the vector by its mean and standard deviation.

  align() : used to create design matrix from input feature vectors.

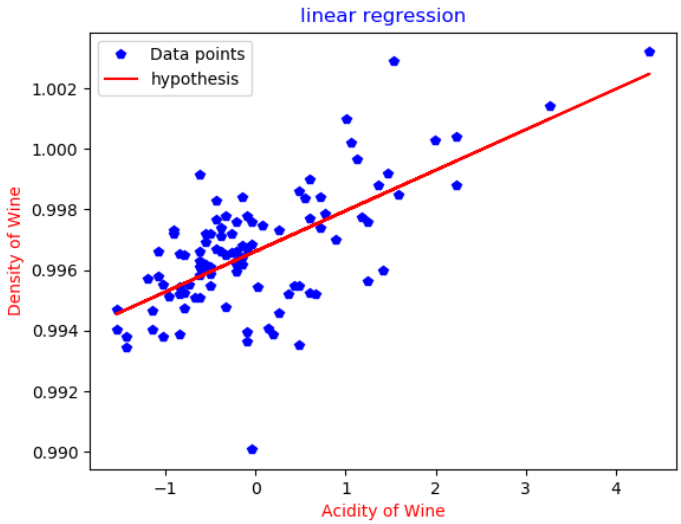
  costFuction() : calculates the loss function of the data.

  gradientFunction() : finds the gradient of the cost function.

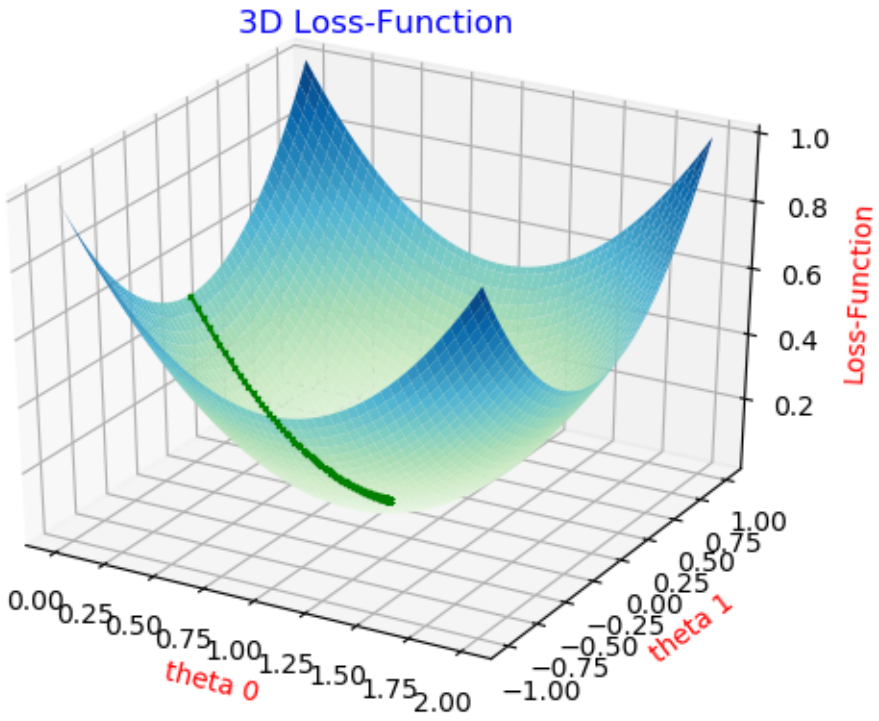
  gradientDescent() : Updates the θ values until it converges.

- **Learning rate** η : 0.03
- **Stopping Criteria is** : abs(costNew - costOld) $\leq$ 1e-15
- **Converged** θ **values is** $\begin{bmatrix}0.99661993 & 0.0013402\end{bmatrix}$

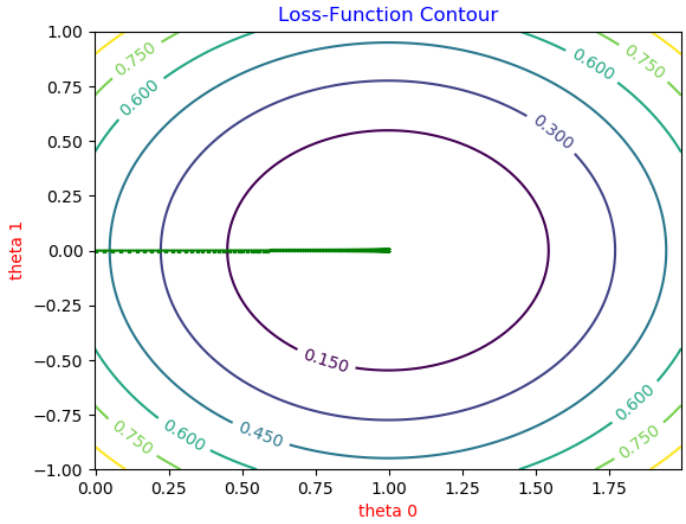
1.2 Plot the data on a two-dimensional graph and plot the hypothesis function



1.3 3D Mesh of J(θ) and animation of learned θ

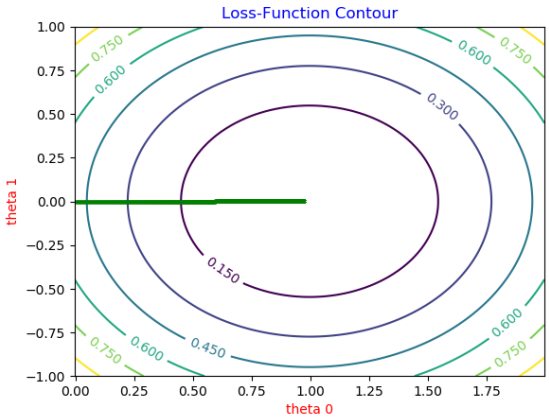


1.4 2D Contours of J(θ) and animation of learned θ

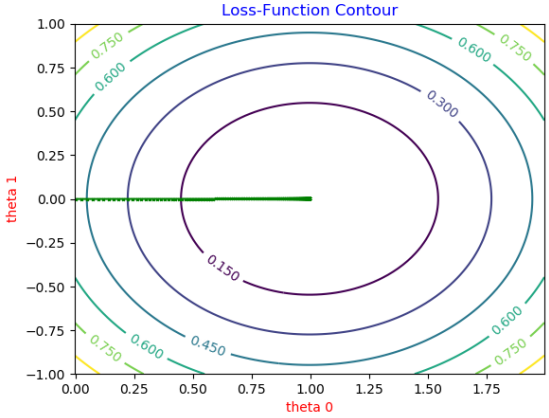


1.5 Different η and their Contours

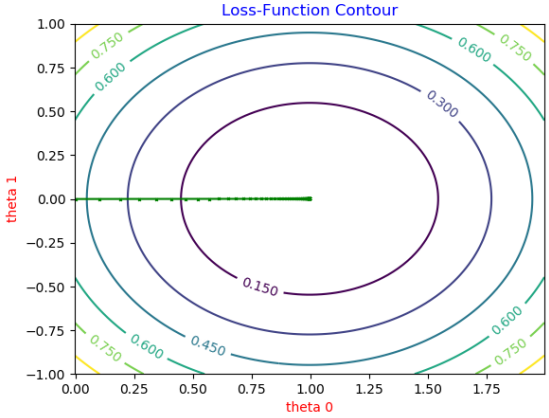
- $\eta = 0.001$ θ : $\begin{bmatrix} 0.9966191 & 0.00134019 \end{bmatrix}$



- $\eta = 0.025$ θ : $\begin{bmatrix} 0.9966199 & 0.0013402 \end{bmatrix}$



- $\eta = 0.1$ θ : $\begin{bmatrix} 0.99662001 & 0.0013402 \end{bmatrix}$



Comment: As we decrease our learning rate η , our gradient descent takes smaller steps hence more iterations to reach the convergence state.

Taking η too small: our descent may never converge.

Taking η too big : it may overshoot.

2 Sampling and Stochastic Gradient Descent

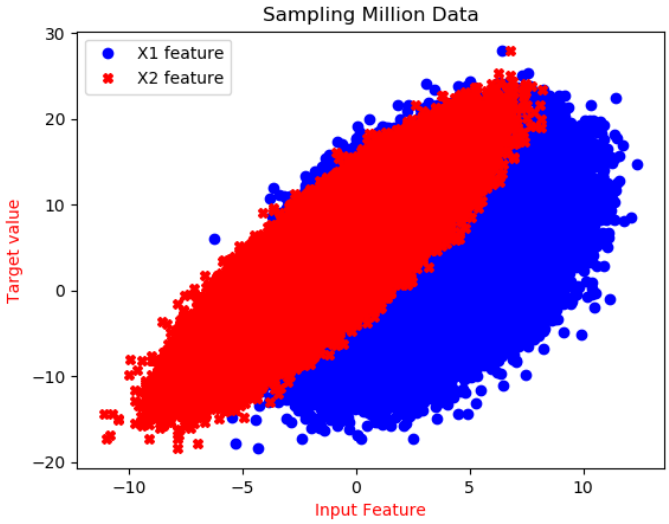
2.1 Sample 1 million data points

sampleData() function is used to generate million data points.

$$\theta = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

$$X1 \sim \mathcal{N}(3,\, 2^2)\,.$$

$$X2 \sim \mathcal{N}(-1,\, 2^2)\,.$$



2.2 Implement Stochastic gradient descent method for optimizing J(θ)

Following are the functions used to implement Batch gradient descent

sampleData() : uses numpy.random.normal to generate data points and makes design matrix of data.
costFuction() : calculates the loss function(cost function) of the data.
gradientFunction() : finds the gradient of the cost function.
stochasticGradientDescent() : Updates the θ values until it converges.

Learning rate is the measure of rate at which we move away from gradient.

Stopping criteria is the measure of the difference in costFunction to check convergence.

Average Iterations is total iterations of batches after which we compare mean cost to check for convergence.

Total iterations is count of total iterations of batches till convergence.

Total time is time taken till convergence.

Batch Sizes r:

Batch Size	Learning Rate	Stopping Criteria	Average iteration	$\theta 0$	$\theta 1$	$\theta 2$
1	0.001	0.0001	2000	2.98868	0.97034	2.00147
100	0.001	0.0001	2000	3.00290	0.99868	1.99991
10000	0.001	0.0001	2000	2.98863	1.0021	1.99885
1000000	0.001	0.001	1000	2.91580	1.019092	1.994093

2.3 Testing data

Do different algorithms in the part above (for varying values of r) converge to the same parameter values?

Ans: No, the different algorithm do not converge to same parameters, but they converge to close approximation of each other.

How much different are these from the parameters of the original hypothesis from which the data was generated?

Ans: Data was generated with parameter $\theta = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix}$ and the algorithms for varying batch sizes return very close approximation to original parameter.

Batch Size	$\theta 0$	$\theta 1$	$\theta 2$
1	2.9886824064476016	0.9703436080610405	2.001475951113718
100	3.0029076235175127	0.9986811675470472	1.9999170562222848
10000	2.9886389527007298	1.002182808959908	1.998857525697211
1000000	2.915806307932103	1.0190926622627925	1.994093937833007

Comment on the relative speed of convergence and also on number of iterations in each case.

Ans:

Batch Size	Time (sec)	Iterations
1	2.486	48000
100	59.27	1000001
10000	13.09	20000
1000000	479.75	13000

Compute the test error with respect to the prediction of the original hypothesis, and compare with the error obtained using learned hypothesis in each case.

Ans:

Test Error		
Batch Size	Learned Hypothesis	Original Hypothesis
1	1.029931409668015	0.9829469215000001
100	0.9831107299734582	0.9829469215000001
10000	0.9833721466789044	0.9829469215000001
1000000	1.004666770931491	0.9829469215000001

Comment: As we increase our batch size the algorithm shifts more toward batch gradient descent. Hence improvement (minor) can be noticed in test error.

This marginal improvement comes at cost of speed.

Smaller is the batch size faster the algorithm converges, as it take fewer time to update θ values.

2.4 Plot the movement of θ

Comment: The shape of θ updates tell us that in smaller batch size θ updates faster and as it has less training example to see before each iterations, hence movement of r=1 is more random.

As batch size increased the movement of θ becomes more smoother as it's processing more training examples in each batch hence more likeness of batch gradient descent.

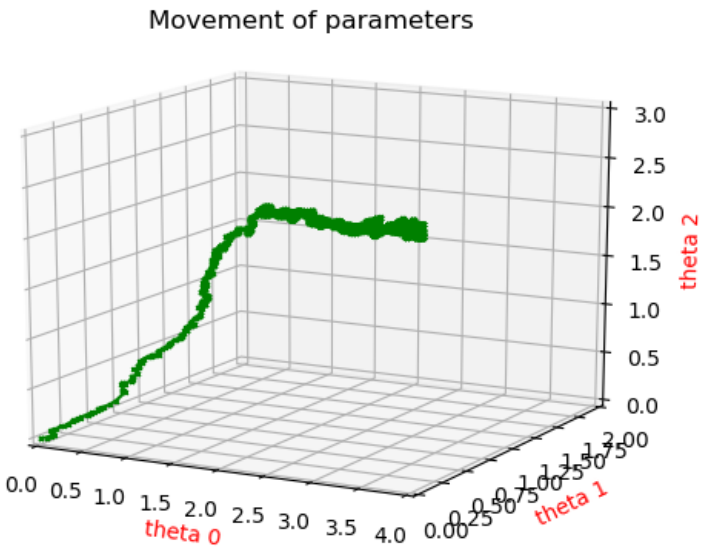


Figure 1: Batch size = 1

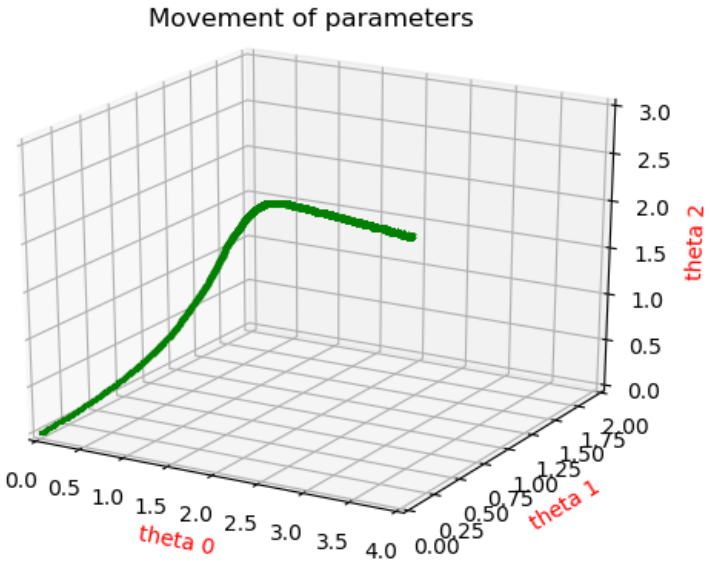


Figure 2: Batch size = 100

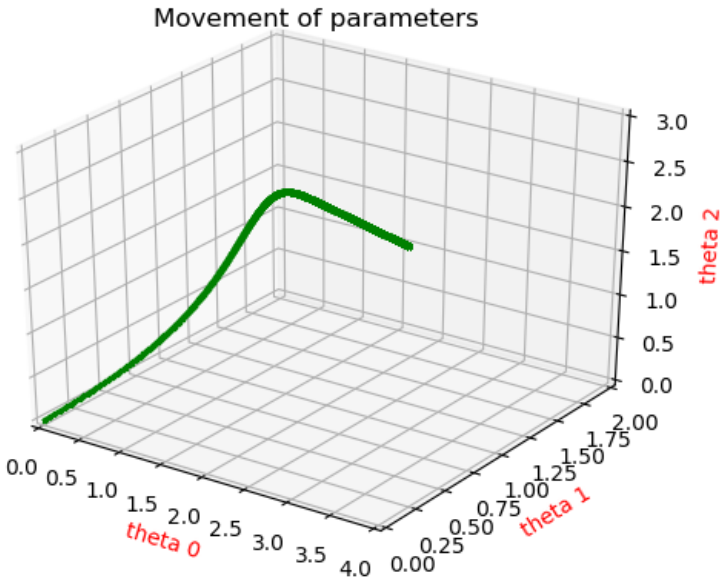


Figure 3: Batch size = 10000

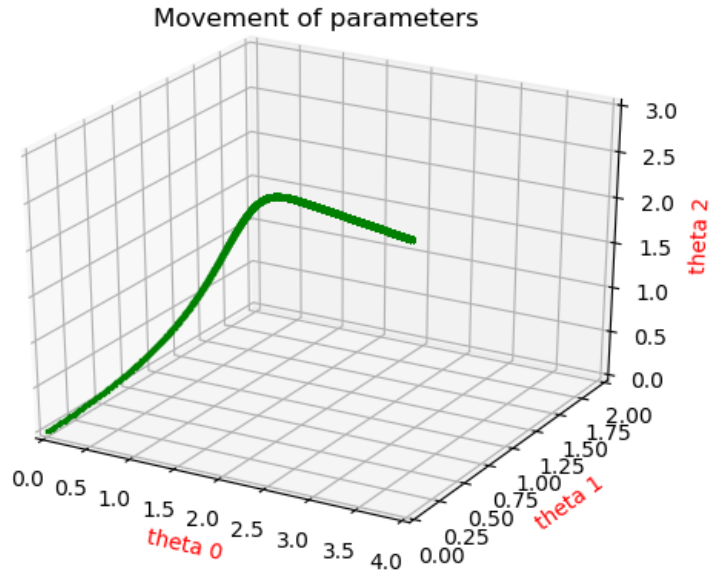


Figure 4: Batch size = 1000000

3 Logistic Regression

Following are the functions used in logistic regression.

inputData() uses pandas to read input data and return design matrix.

sigmoid() implements sigmoid function

$$g(z) = \frac{1}{1 + e^{-\theta^T x}}$$

newtonUpdate() this functions updates the value of theta

forumula used :

$$\theta := \theta - H^{-1} \nabla_\theta L(\theta)$$

$$H^{-1} = \sum_{i=1}^{m} -x_{\mathrm{j}}^{(i)} h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) x_{\mathrm{k}}^{(i)}$$

$$\nabla_\theta L(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_k^{(i)}$$

3.1 What are the coefficients θ resulting from your fit?

Ans:

$$\theta = \begin{bmatrix} 0.40125316 \\ 2.5885477 \\ -2.72558849 \end{bmatrix}$$

3.2 Plot training data

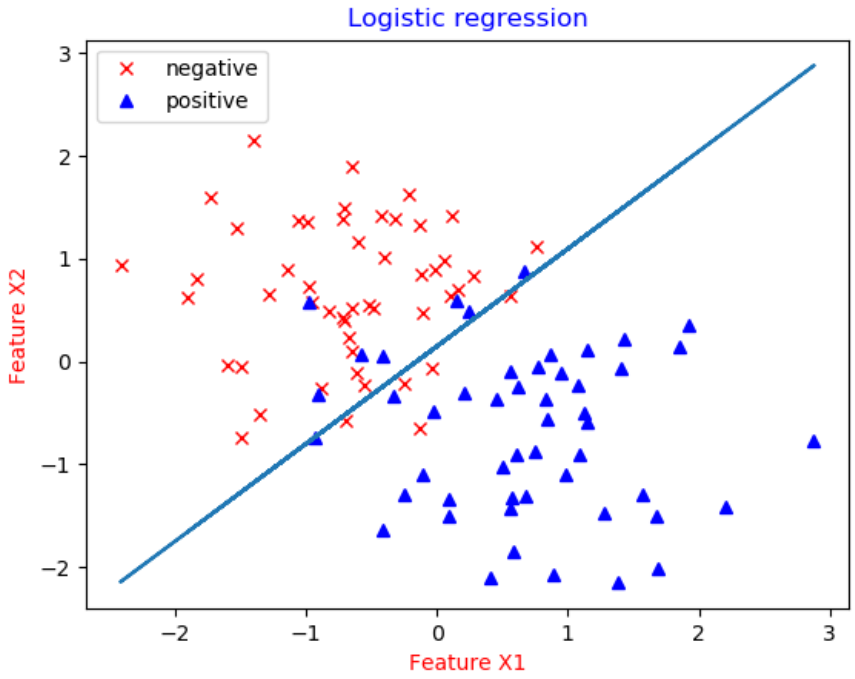


Figure 5: Decision Boundary after newton update

Comment: Newton update takes much fewer iterations than gradient descent, but it has to calculate Hessian inverse which is $O(n^3)$ hence each iterations is more costly than iteration of gradient descent.

4 Gaussian Discriminant Analysis

4.1 Implement Gaussian Discriminant Analysis $\Sigma_0 = \Sigma_1 = \Sigma$

$$\Sigma = \sum_{i=1}^{m} \frac{(x^{(i)} - \mu_y^{(i)})(x^{(i)} - \mu_y^{(i)})^T}{m}$$

$$\phi = \frac{\sum_{i=1}^{m} (1\{y^{(i)} = 1\})}{m}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} x^{(i)}}{1\{y^{(i)} = 1\}}$$

$$\mu_0 = \frac{\sum_{i=0}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{1\{y^{(i)} = 0\}}$$

Report the values of the means ,$\mu_0$and μ_1, and the co-variance matrix Σ.

Ans:

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

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

4.2 Plot the training data corresponding to the two coordinates of the input features

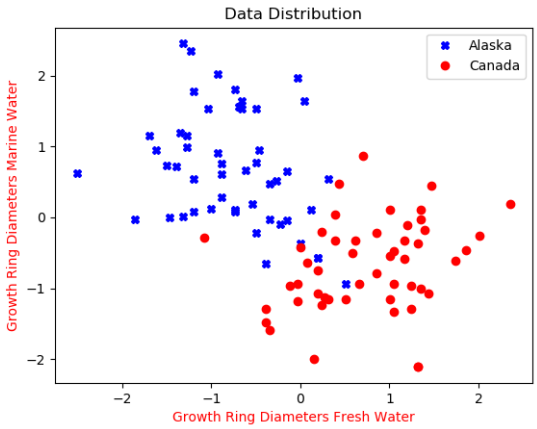


Figure 6: Alaska vs Canada

4.3 Describe the equation of the boundary separating the two regions in terms of the parameters μ_0, $\mu_1$and Σ.

Since Both co-variance matrix are same hence boundary will be linear as equation turns out to be degree one polynomial in x

$$\log(\frac{\phi}{(1-\phi)}) + \frac{1}{2}\log(\frac{|\sum_1|^{0.5}}{|\sum_0|^{0.5}} + +x^T(\sum_1^{-1}\mu_1 - \sum_0^{-1}\mu_0) + \frac{\mu_0^T\sum_0^{-1}\mu_0 - \mu_1^T\sum_1^{-1}\mu_1}{2}$$

Hence using the linear equation in x we make function that find x_2 corresponding to x_1 to plot decision boundary .

$$\theta_1 x_1 + \theta_2 x_2 + c = 0$$

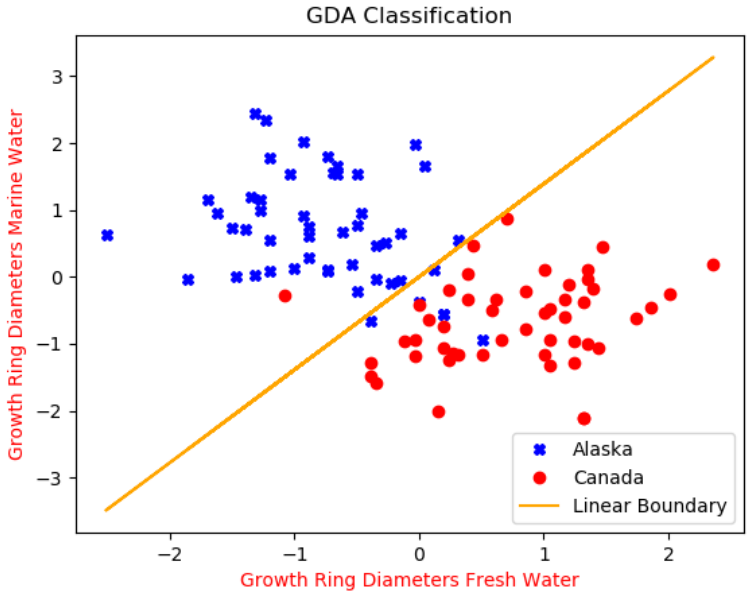


Figure 7: Alaska vs Canada

4.4 Report the values of the parameter estimates i.e. μ_0, μ_1, Σ_0, Σ_1

$$\Sigma_0 = \sum_{i=1}^{m} \frac{1\{y^{(i)} = 0\}(x^{(i)} - \mu_0)(x^{(i)} - \mu_0)^T}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\Sigma_1 = \sum_{i=1}^{m} \frac{1\{y^{(i)} = 1\}(x^{(i)} - \mu_1)(x^{(i)} - \mu_1)^T}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

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

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

4.5 Describe the equation for the quadratic boundary separating the two regions in terms of the parameters μ_0, μ_1, Σ_0, Σ_1

Since our co-variance matrix is different hence our boundary is now quadratic in shape. Which can be visualized by looking at the below equation

$$\log(\frac{\phi}{(1-\phi)}) + \frac{1}{2}\log(\frac{|\sum_1|}{|\sum_0|} + x^T \frac{(\sum_0^{-1} - \sum_1^{-1})}{2} x + x^T(\sum_1^{-1}\mu_1 - \sum_0^{-1}\mu_0) + \frac{\mu_0^T \sum_0^{-1} \mu_0 - \mu_1^T \sum_1^{-1} \mu_1}{2}$$

Above mentioned equation is quadratic in x hence can be converted to general quadratic equation.

$$\theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2 + \theta_3 x_1 x_2 + \theta_4 x_1 + \theta_5 x_2 = 0$$

Now we solve for x_2 for a given x_1 to plot the quadratic decision boundary .

$$\theta_0 = \log(\frac{\phi}{(1-\phi)}) + \frac{1}{2}\log(\frac{|\sum_1^{-1}|}{|\sum_0^{-1}|} + \frac{\mu_0^T \sum_0^{-1} \mu_0 - \mu_1^T \sum_1^{-1} \mu_1}{2}$$

let $A = \frac{(\sum_0^{-1} - \sum_0^{-1})}{2}$

$$\theta_1 = A_{00}$$

$$\theta_2 = A_{11}$$

$$\theta_3 = A_{01} + A_{10}$$

$$\begin{bmatrix}\theta_4 & \theta_5\end{bmatrix} = \sum_1^{-1} \mu_1 - \sum_0^{-1} \mu_0$$

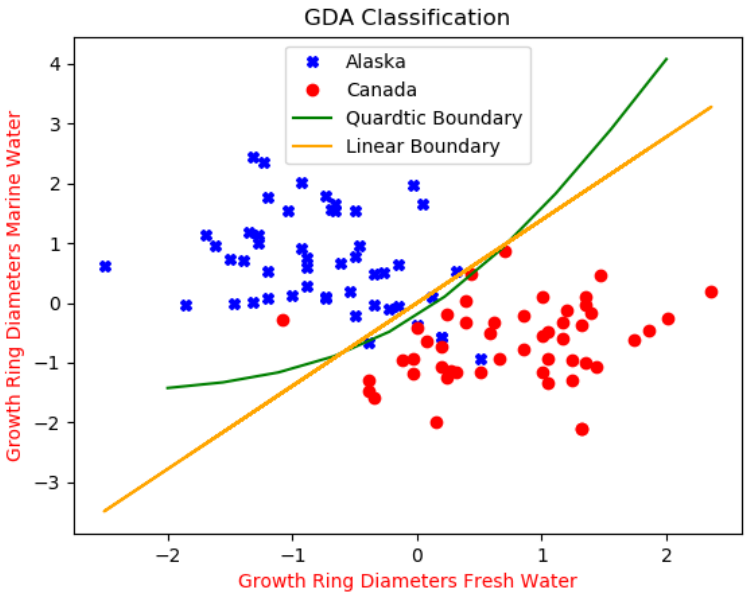


Figure 8: Alaska vs Canada

4.6 Carefully analyze the linear as well as the quadratic boundaries obtained. Comment on your observations.

Comment: In linear boundary we assumed that co variance matrix is same. Hence by our equation we got a fit as shown in above figure.

But if we drop that assumption and calculated the co-variance of both class of distribution we got that co variance is different hence our boundary came to be quadratic in shape.

Since there is less assumptions in second case it tends to give better result.

As discussed in class if our assumption of data matches the GDA outperforms logistic regression, this can be seen as linear boundary signifies the logistic regression.