COL774 – Assignment 1

- **By Nishant Kumar**
2012CS10239

Problem – 1

a)

In order to implement batch Gradient Descent to learn the hypothesis function, first the X data was imported from file and normalized as follows :

$$x_j^{(i)} = \frac{x_j^{(i)} - mean\left(X_j\right)}{std\left(X_j\right)}$$

Where $X_j represents\ the\ values\ of\ the\ jth\ feature.$

The error function is

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

$$= \frac{1}{2m}(Y - X\theta)^T(Y - X\theta)$$

Calculating the gradient, we have

$$\nabla_\theta J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T x^{(i)}\right) * x^{(i)}$$

$$= -\frac{1}{m}\left(X^T * (Y - X\theta)\right)$$

So, the batch gradient descent update becomes,

$$\theta_{new} = \theta_{old} - \eta\, \nabla_\theta J(\theta)$$

$$\Rightarrow \theta_{new} = \theta_{old} + \frac{\eta}{m}\left(X^T * (Y - X\theta)\right)$$

$$Initialization : \theta_{Init} = 0\ (given)$$

$$Stopping\ criteria : J(\theta_i) - J(\theta_{i-1}) < eps$$

Observations

- The following table shows the number of iterations taken by batch Gradient Descent Algorithm and the corresponding values of $\theta\ and\ J(\theta)$ obtained.

TABLE (1.1)

Learning Rate	Number of iterations	θ_0	θ_1	$J(\theta)$
0.001	5479	5.814832	4.596568	4.477472
0.1	76	5.837191	4.615224	4.476975
0.3	26	5.838587	4.616415	4.476972
0.4	19	5.838779	4.616581	4.476972
0.9	6	5.839129	4.616893	4.476971
1.0	4	5.839135	4.616901	4.476971
1.1	6	5.839129	4.616899	4.476971
1.3	9	5.839250	4.616961	4.476971
1.5	14	5.838779	4.616720	4.476972
1.9	74	5.836735	4.616529	4.476975
1.99	636	5.829355	4.616529	4.477020
2.0	314	0.000000	4.610245	21.524743

Also, experimenting with the different values of eps, we get

TABLE (1.2)

Eps	*Number of iterations*	θ_0	θ_1	$J(\theta)$
1.0e-6	4	5.839135	4.616901	4.476971
1.0e-4	4	5.839135	4.616901	4.476971
1.0e-2	3	5.839135	4.616896	4.476971
0.1	3	5.839135	4.616896	4.476971
1	3	5.839135	4.616896	4.476971

- From the above table, it can be safely concluded that $Learning\ Rate = 1$ and $eps = 1.0e-6$ are good for our dataset. Using a learning rate = 1, ensures fast convergence to desired values of $J(\theta)$.
  Also, the same can be concluded from the contour and mesh plots. [run the script for the same].
- Using these values, we get the following plots:

$$LearningRate = 1.0\,, eps = 1.0e-6\,, \theta_{init} = 0$$

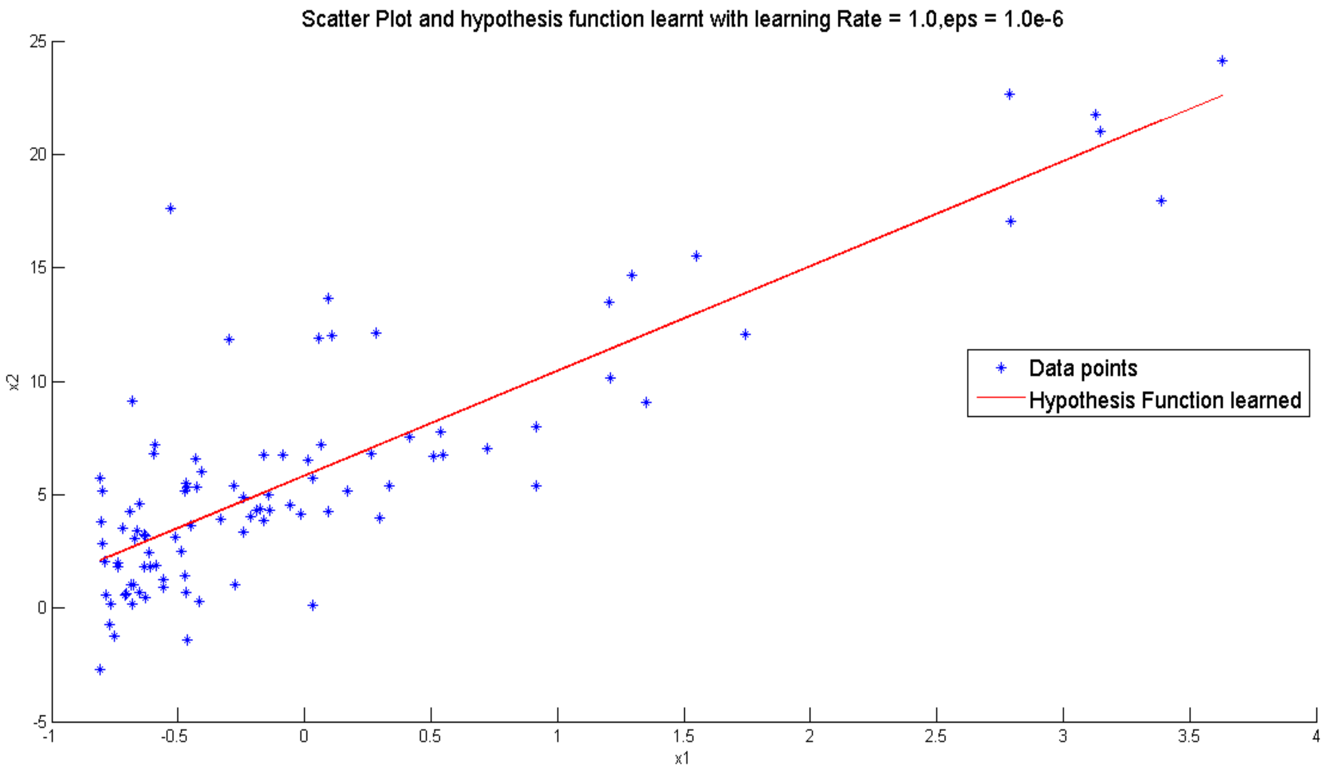
Final values obtained:

$$[\theta_0\ \theta_1] = [5.839\ 4.617]$$

$$Number\ of\ iterations = 4$$

$$J(\theta_{final}) =\ 4.477$$

b) **Figure 1.1**



c) d) Using a time gap of 0.2 seconds, the following plots were obtained.

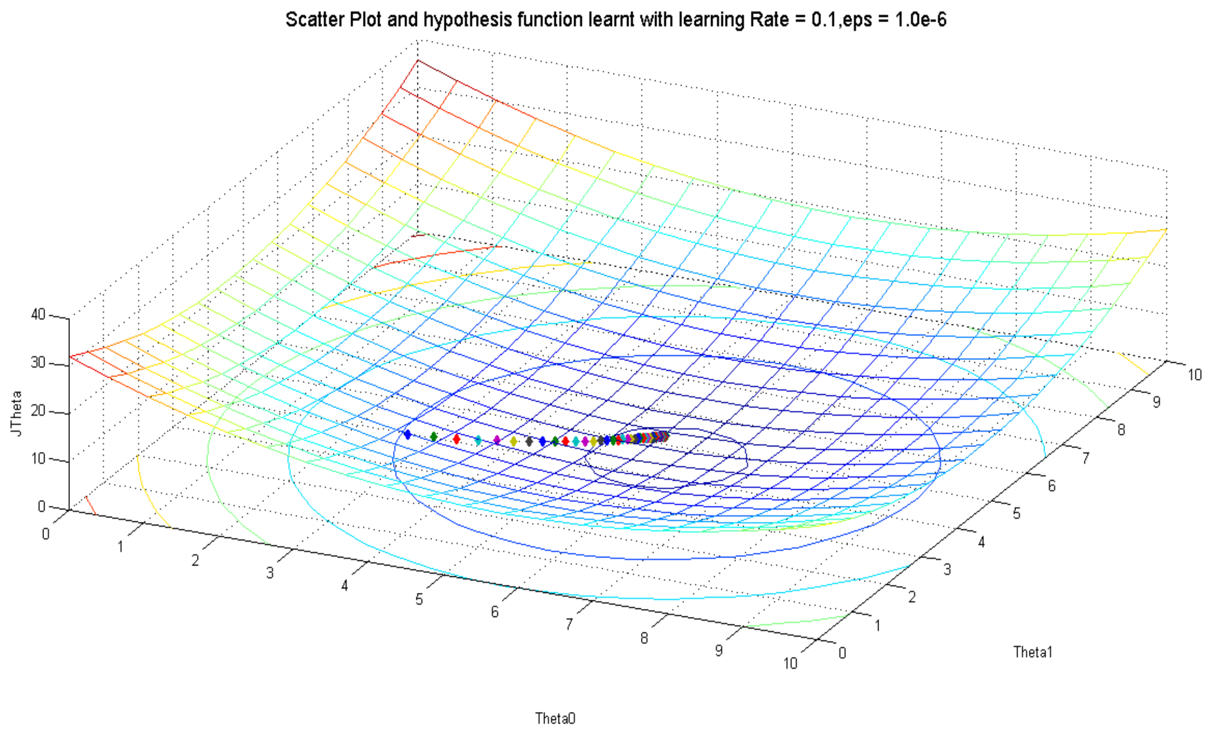
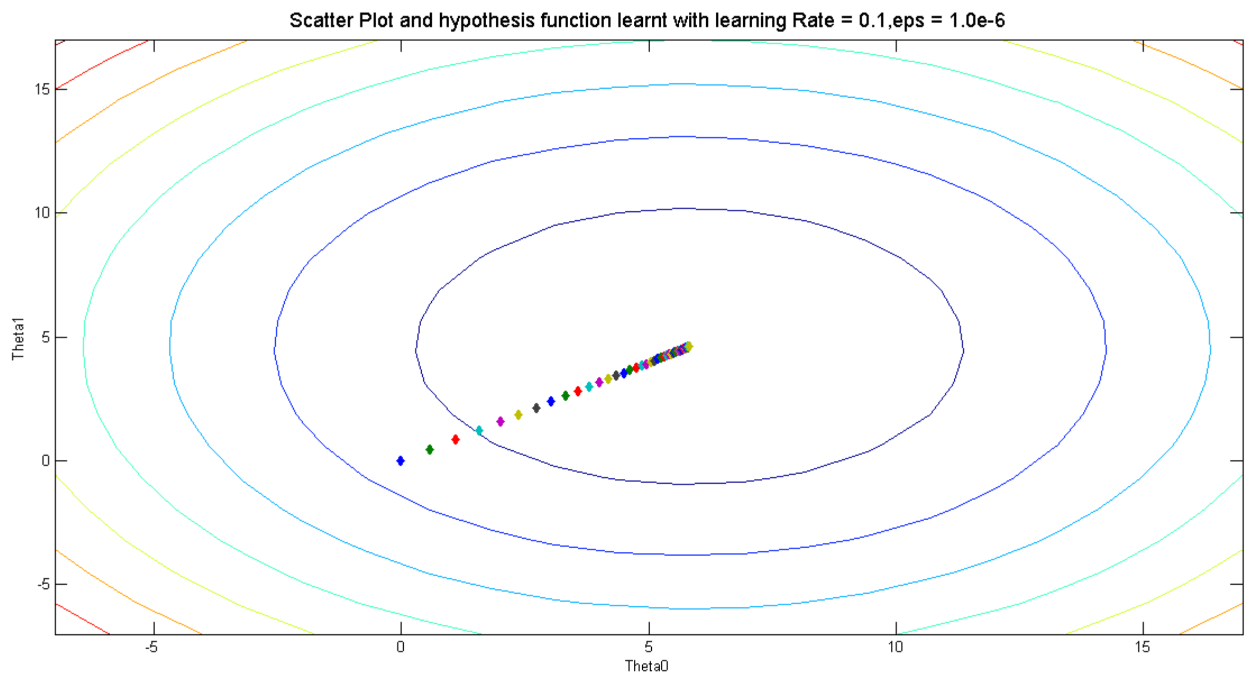


Figure 1.2

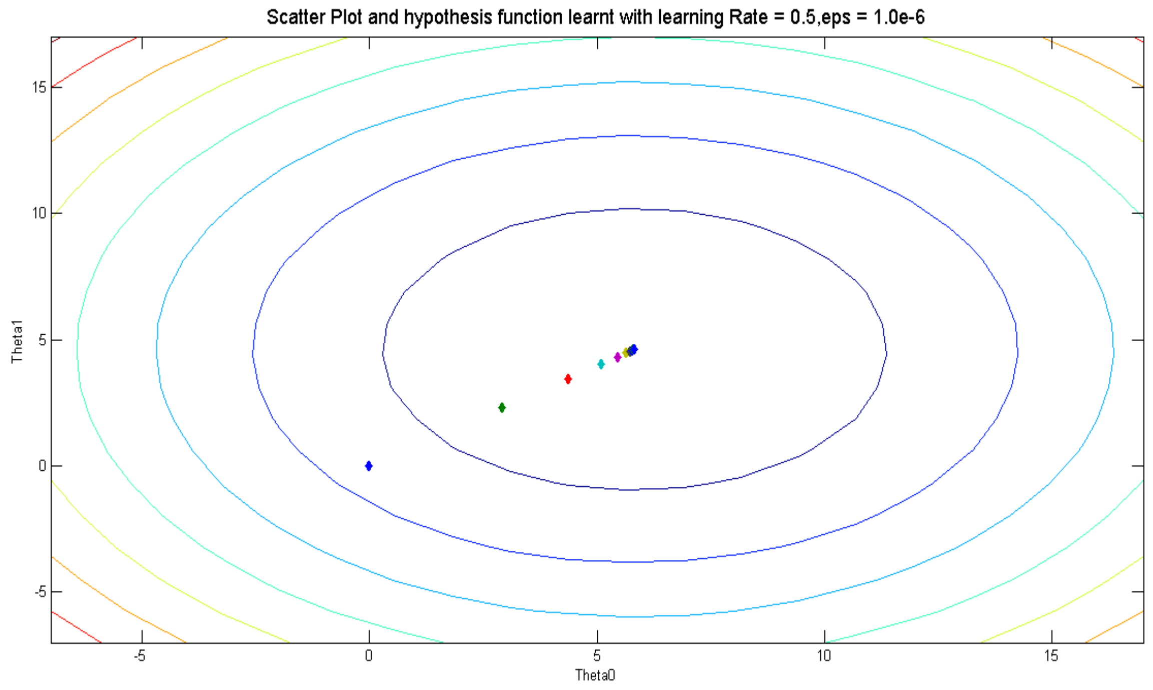
Figure 1.3

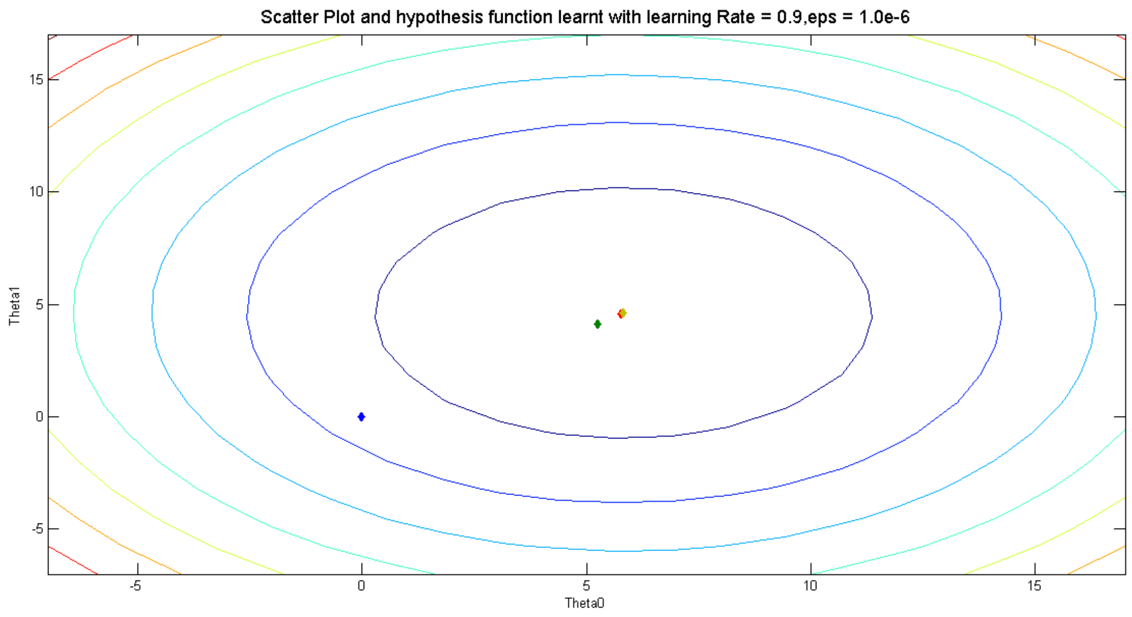


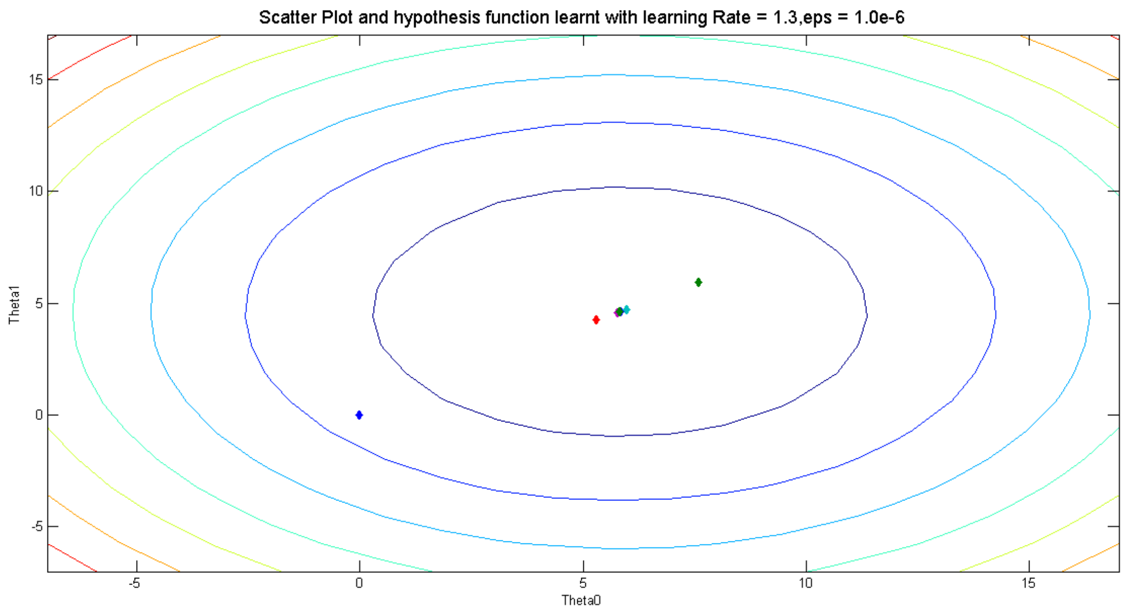
e)

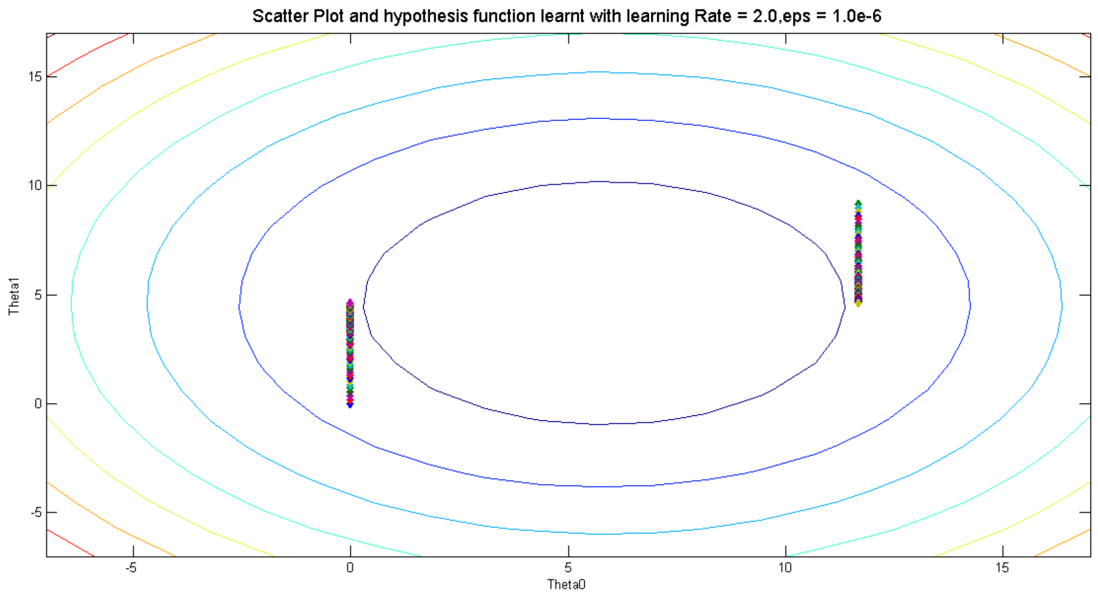
- As is clear from table (1.1), if the learning rate is very small – 0.001, the number of iterations needed to converge to optimum value becomes large.
- Below 1.0, there is no divergence and the algorithm converges to optimum.
- For learning Rate = 1.0, the convergence happens fairly quickly in 4 iterations.
- If learning Rate is then increased, the number of iterations keeps on increasing till 1.99.
- At $\text{learning Rate} = 2.0$, gradient descent is unable to converge.
- Above 2.0, there is divergence and the algorithm is unable to finish.

Some figures to illustrate the same are shown here.




Scatter Plot and hypothesis function learnt with learning Rate = 0.9,eps = 1.0e-6
Theta1
Theta0
15
10
5
0
-5
-5
0
5
10
15


Scatter Plot and hypothesis function learnt with learning Rate = 1.3,eps = 1.0e-6
Theta1
Theta0
15
10
5
0
-5
-5
0
5
10
15



Problem – 2

To implement linear regression, the X values were first imported from the file and normalized. Then a column corresponding to $x_0^{(i)} = 1$ was added to X to get *dataX*. This *x* was then used to in the following normal equations to find the hypothesis function.

Derivation of normal Equation

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

$$= \frac{1}{2m}(X\theta - Y)^T W(X\theta - Y)$$

$$\Rightarrow (2m) * J(\theta) = ((X\theta)^T - Y^T)(WX\theta - WY)$$

$$= \theta^T(X^TWX)\theta - \theta^T(X^TWY) - Y^TWX\theta + Y^TWY$$

$$= \theta^T(X^TWX)\theta - 2 * \theta^T(X^TWY) + Y^TWY$$

$\Rightarrow (2m) * \nabla_\theta J(\theta) = 2(X^T W X)\theta - 2X^T W Y = 0$

$\Rightarrow X^T W X \theta = \;\; X^T W Y$

$\Rightarrow \theta = (X^T W X)^{-1}(X^T W Y)$

a) The following normal equation was used to implement the un-weighted linear regression :

$$\theta = (X^T X)^{-1}(X^T Y)$$
$$y = \theta^T x$$

Using the above normal equations, the final value of Theta obtained was

$$\theta_0 = \; 1.031280$$
$$\theta_1 = \; 0.839396$$

b) Using the normal equation derived for weighted linear regression, plot of the weighted linear regression was obtained. For this, the weight matrix W was calculated at each point on x as follows:

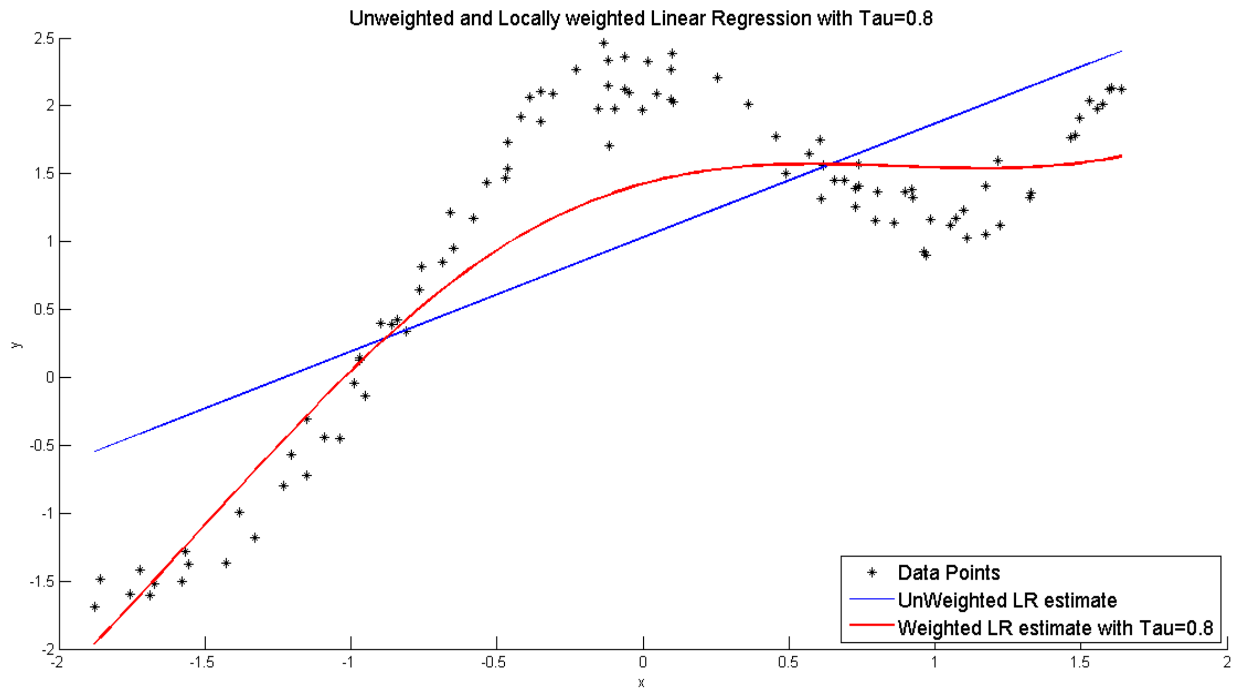
$$W_{ij} = w^{(i)} = \exp\left(-\frac{\left(x - x^{(i)}\right)^2}{2\tau^2}\right) \; if \; i = j$$

$$= 0 \; if \; i \neq j$$

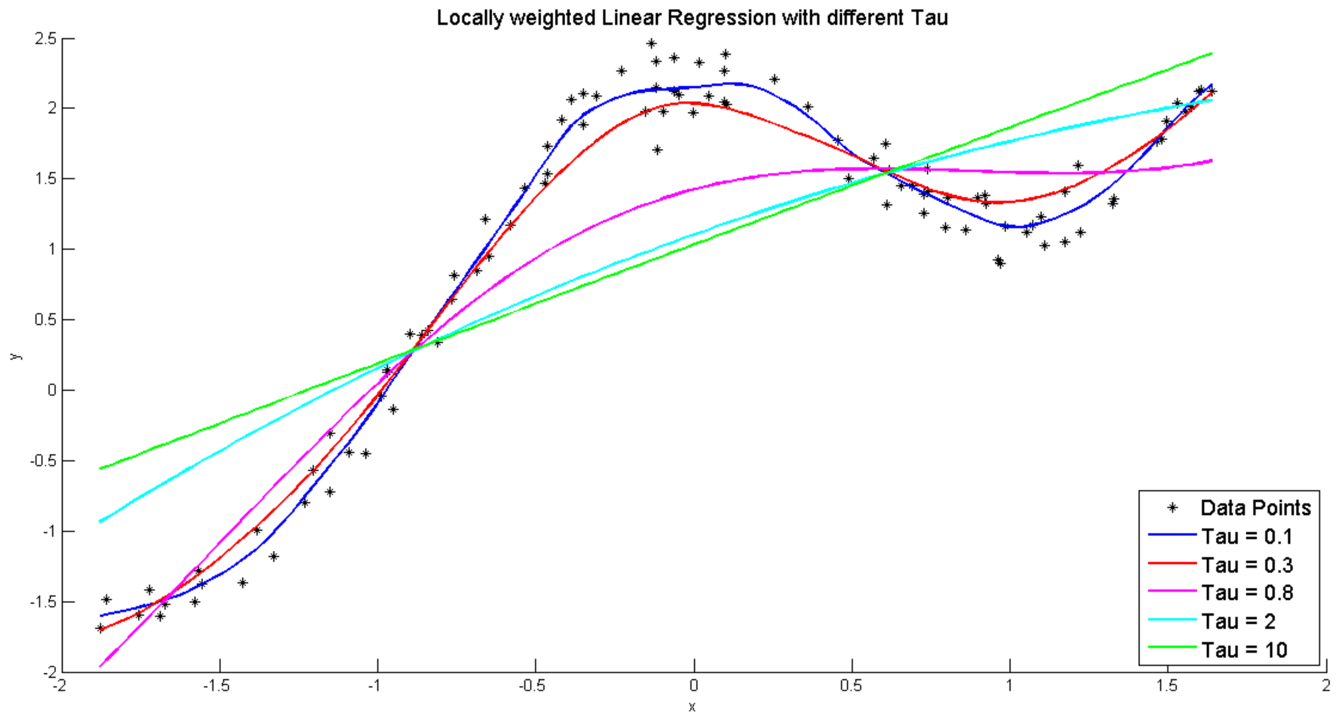
$$\theta = (X^T W X)^{-1}(X^T W Y)$$

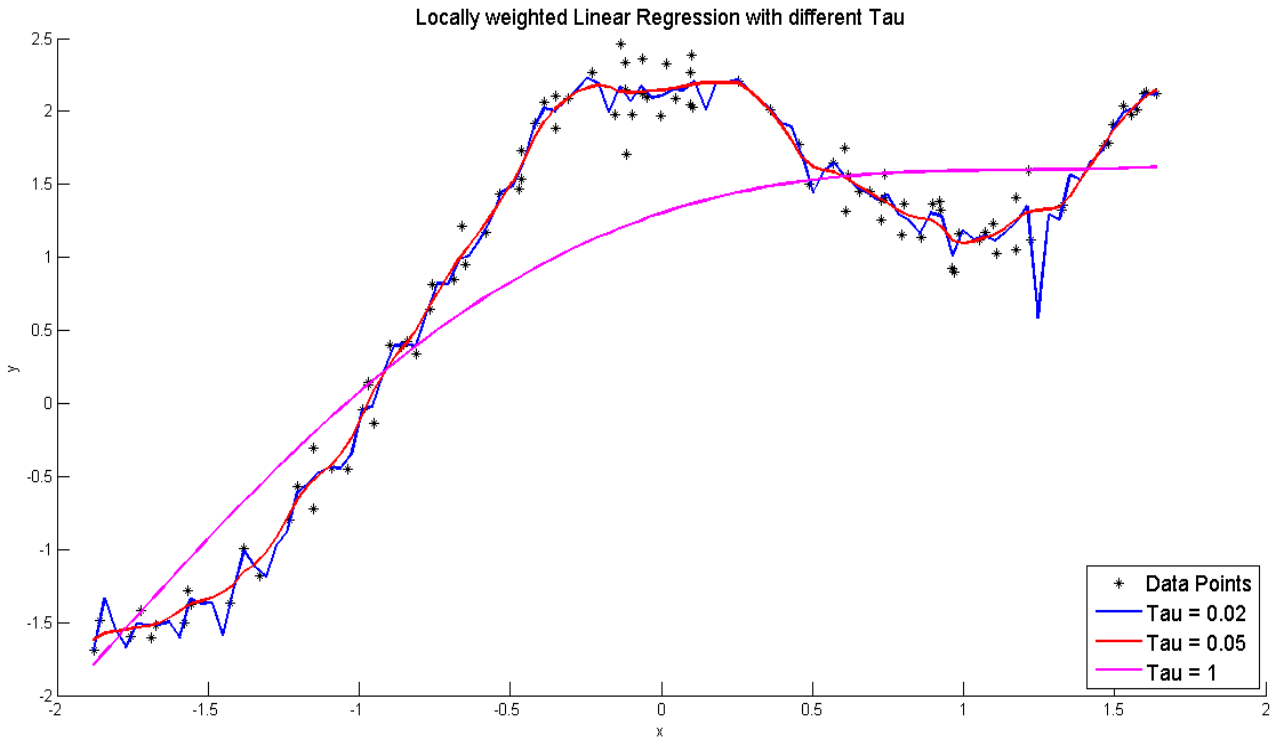
$$y = \theta^T x$$

The following plot was thus obtained:



c) The following plot was obtained on experimenting with different values of Tau.





As can clearly be seen from the above figures, the value of $\tau = 0.3$ seems to fit the curve without under or over-fitting.

As τ decreases from 0.8, initially the curve seems to better fit, but if τ becomes very small, over-fitting starts happening. For example, if $\tau = 0.02$, there is clearly over-fitting happening as can be seen in the above graph. And if τ is increased from 0.8, the curve comes closer and closer to linear. If $\tau = 10$, (as can be seen above) the curve becomes closer to linear curve and does not seem to fit the data at all.

Problem – 3

In this question, the log-likelihood function for logistic regression was optimized using newton's method.

The gradient and the hessian were calculated as follows:

METHOD

The X data obtained from the file was first normalized and then used in the rest of the parts.

Let $H_\theta(X)$ be a vector s.t.

$$H_\theta(X)_i = \frac{1}{1 + e^{-\theta^T x^{(i)}}} \ where\ X_i = x^{(i)}$$

Also, the log-likelihood function for logistic regression is:

$$L(\theta) = \sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \sum_{i=1}^{m} (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

$$= Y^T * \log\left(H_\theta(X)\right) + (1 - Y)^T * \log\left(1 - H_\theta(X)\right)$$

Applying gradient operator on both sides, we get (as derived in class):

$$\nabla_\theta L(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta\left(x^{(i)}\right)) * x^{(i)} = X^T * (Y - H_\theta(X))$$

Also, applying the gradient again, we end up with the following hessian matrix:

$$Hessian_{jk} = -\sum_{i=1}^{m} x_j^i * x_k^i * h_\theta\left(x^i\right) * \left(1 - h_\theta\left(x^i\right)\right) = -X^T S X$$

$$where\ S_{ij} = h_\theta\left(x^i\right) * \left(1 - h_\theta\left(x^i\right)\right), i = j$$

$$and\ \ S_{ij} = 0, i \neq j$$

So, to optimize the log-likelihood function the following was applied:

Newton's method :

$$\theta_{new} = \theta_{old} - Hessian^{-1} \nabla_{\theta} L(\theta)$$

$$Initialization : \theta_0 = 0$$

$$Stopping\ criteria : L(\theta_i) - L(\theta_{i-1}) < eps$$

RESULTS

The following were the final values of θ obtained on running the newton's method:

$$eps = 10^{-6}$$

$$\theta_0 = -0.047176$$

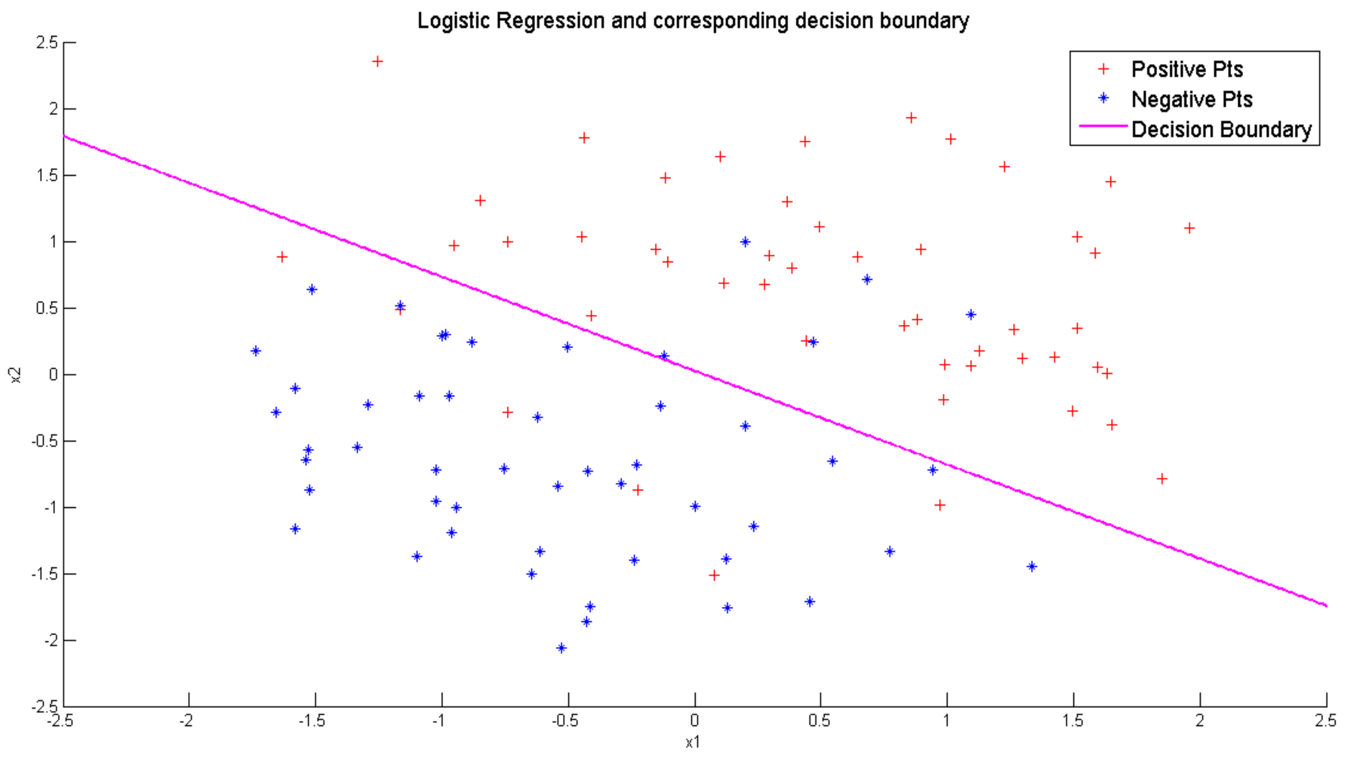
$$\theta_1 = 1.467489$$

$$\theta_2 = 2.076375$$

$$num\ of\ Iterations = 7$$

PLOT

The decision boundary was plotted using the final value of theta obtained.



Problem – 4

a) Assuming that the two classes have the same covariance matrix, we have (as derived in class) :

$$\phi = \frac{\sum_{i=1}^{m} 1\{y^i = 1\}}{\sum_{i=1}^{m} 1\{y^i = 0\} + \sum_{i=1}^{m} 1\{y^i = 1\}}$$

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

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

$$\Sigma_0 = \Sigma_1 = \Sigma = \frac{1}{m} * \sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right) * \left(x^{(i)} - \mu_{y^{(i)}}\right)^T = \phi\Sigma_1 + (1-\phi)\Sigma_0$$

Using this closed form formula, we get,

$$\phi = 0.5$$

$$\mu_0 = \begin{matrix}\mu_{0_1} \\ \mu_{0_2}\end{matrix} = \begin{matrix}0.7515 \\ -0.6817\end{matrix} \quad , \quad \mu_1 = \begin{matrix}\mu_{1_1} \\ \mu_{1_2}\end{matrix} = \begin{matrix}-0.7515 \\ 0.6817\end{matrix}$$

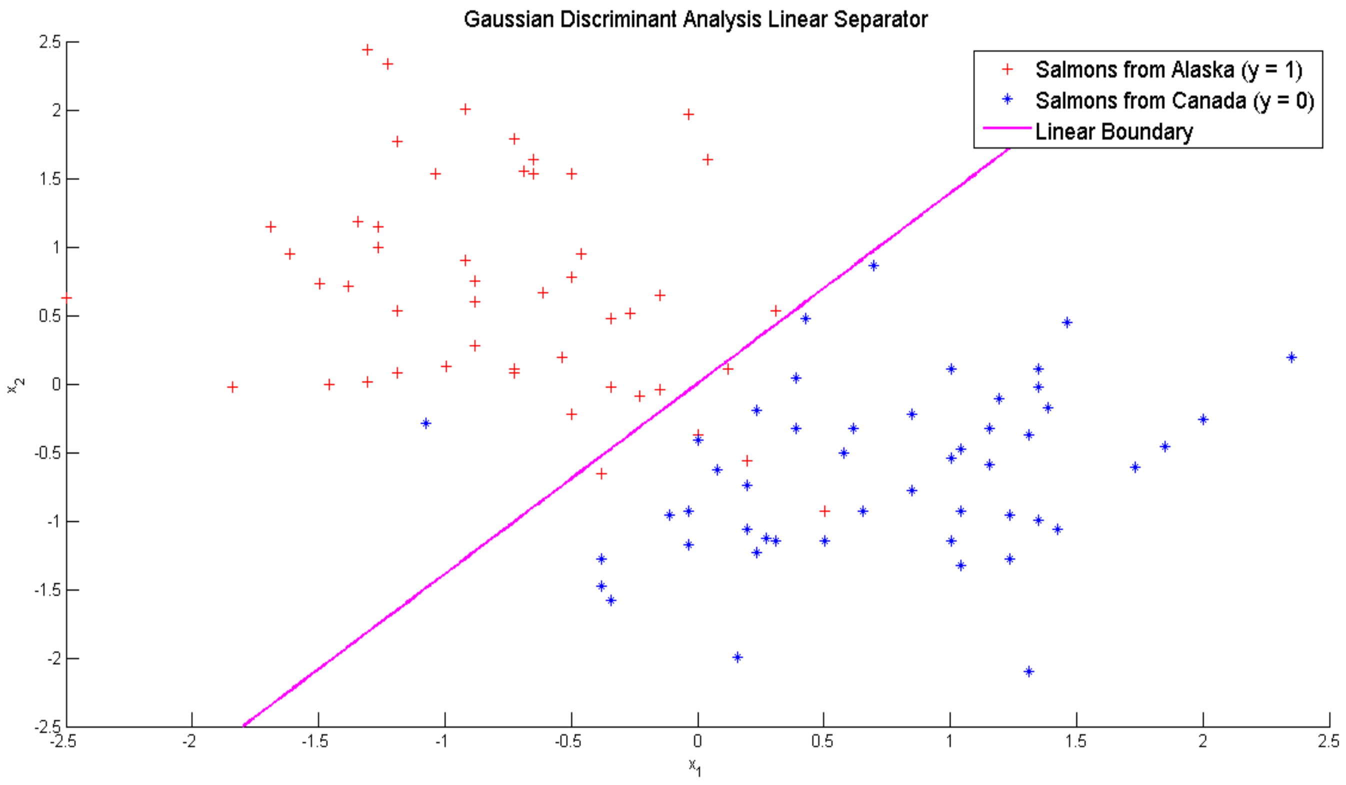
$$\Sigma = \begin{matrix}0.4252 & -0.0222 \\ -0.0222 & 0.5253\end{matrix}$$

b) c)

Since the covariance matrices of both classes are same, we will get a linear separator. To find the linear separator, we have (as derived in class)

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

Here $\phi = 0.5$, so using ezplot to plot the linear curve, we get the following plot.



d)

Assuming, $\Sigma_0 \neq \Sigma_1$, we get a quadratic separator as follows:

$$\Sigma_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}\left(x^{(i)} - \mu_{y^{(i)}}\right) * \left(x^{(i)} - \mu_{y^{(i)}}\right)^T}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\Sigma_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}\left(x^{(i)} - \mu_{y^{(i)}}\right) * \left(x^{(i)} - \mu_{y^{(i)}}\right)^T}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

Values found:

$$\phi = 0.5$$

$$\mu_0 = \begin{matrix} \mu_{0_1} \\ \mu_{0_2} \end{matrix} = \begin{matrix} 0.7515 \\ -0.6817 \end{matrix} \;, \; \mu_1 = \begin{matrix} \mu_{1_1} \\ \mu_{1_2} \end{matrix} = \begin{matrix} -0.7515 \\ 0.6817 \end{matrix}$$

$$\Sigma_0 = \begin{matrix} 0.4727 & 0.1088 \\ 0.1088 & 0.4094 \end{matrix}$$

$$\Sigma_1 = \begin{matrix} 0.3778 & -0.1533 \\ -0.1533 & 0.6413 \end{matrix}$$

Its worth noting that $\Sigma = \phi\Sigma_1 + (1-\phi)\Sigma_0$ holds.

e)

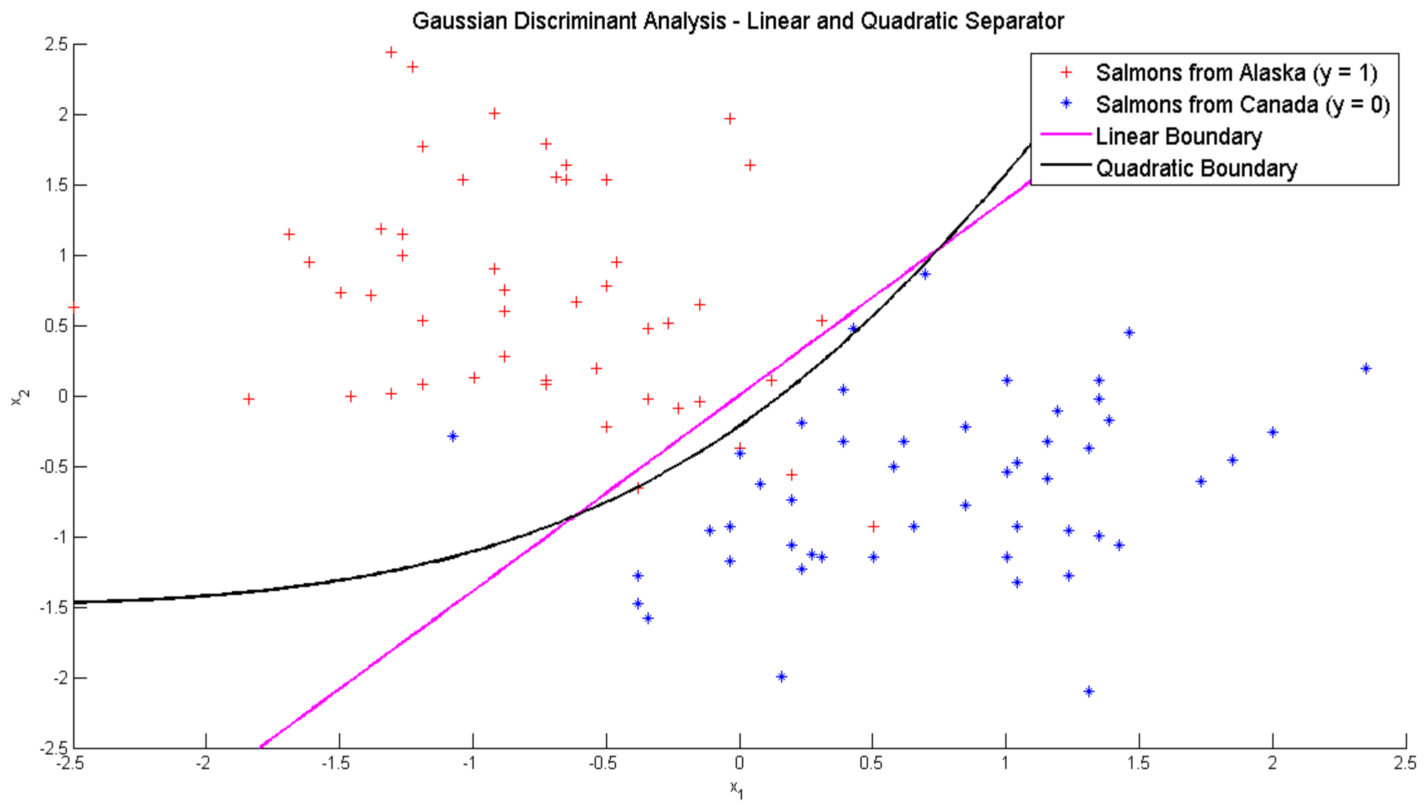
The equation of the quadratic boundary is as follows:

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

Here $\phi = 0.5$, so we get the equation of the boundary as

$$\log\frac{|\Sigma_0|}{|\Sigma_1|} = (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)$$

Plotting this using ezplot, we have,



f)

- Analyzing the above figure, the linear boundary has 6 points that get classified in the wrong class. Whereas, the quadratic boundary has 5 points that lie in the wrong class.
- The quadratic classifier allows more flexibility in the generative process by allowing the covariance matrices of the 2 classes to be different, but, the linear classifier assumes the covariance matrices to be the same.
- Carefully seeing the plot, one finds that the linear classifier wrongly classifies some of the points that lie near the boundary. But the quadratic one tries its best to classify the points near boundary into the right class.
- Thus, it can concluded that the quadratic classifier does better classification than linear one.