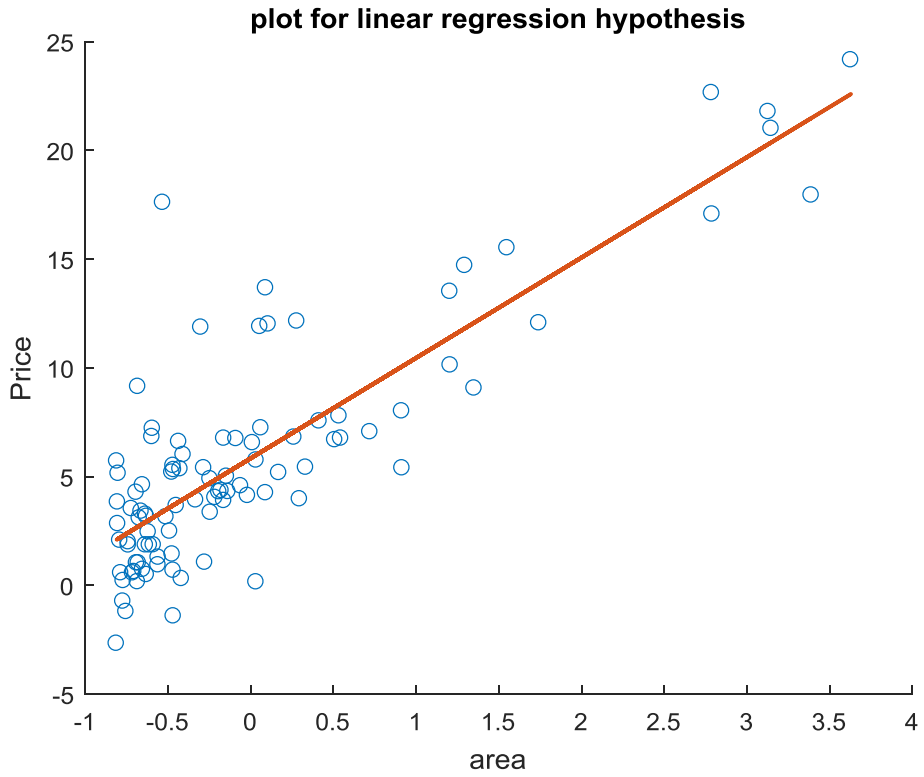
Report for Assignment1

Ques1:

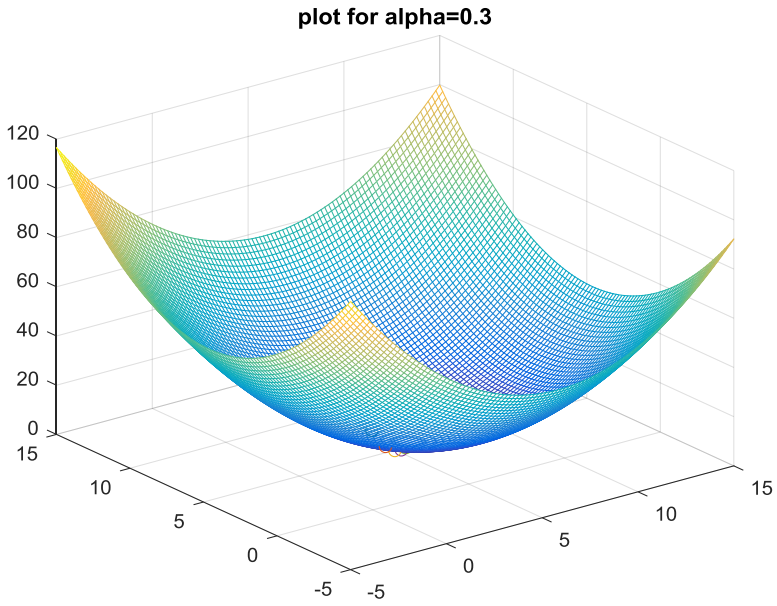
a. Implement Batch Gradient Descent (choose appropriate learning rate and stopping criteria)

- For implementing batch gradient descent, first started the algorithm with initial value of theta set to zeros. Also the data was normalized so that algorithm could converge in less number of iteration with an appropriate learning rate. Then, the loop is started which calculates new theta with an alpha rate of 0.3. This theta is calculated by subtracting gradient (learning rate times) at previous point from the previous theta value. The learning rate is set to 0.3 by experimenting again and again. With a high learning rate, the theta would converge way too fast and may possibly start diverging, while with small learning rate theta would converge very slowly. So, 0.3 was chosen to be appropriate value. Also the convergence criteria chosen was when difference between new cost and previous cost is less than 10^ (-6) as this was sufficiently a very small value to stop the loop. Final **cost** with this learning rate and stopping criteria was found to be **4.4770** and corresponding **theta** is **[5.8384; 4.6162]**

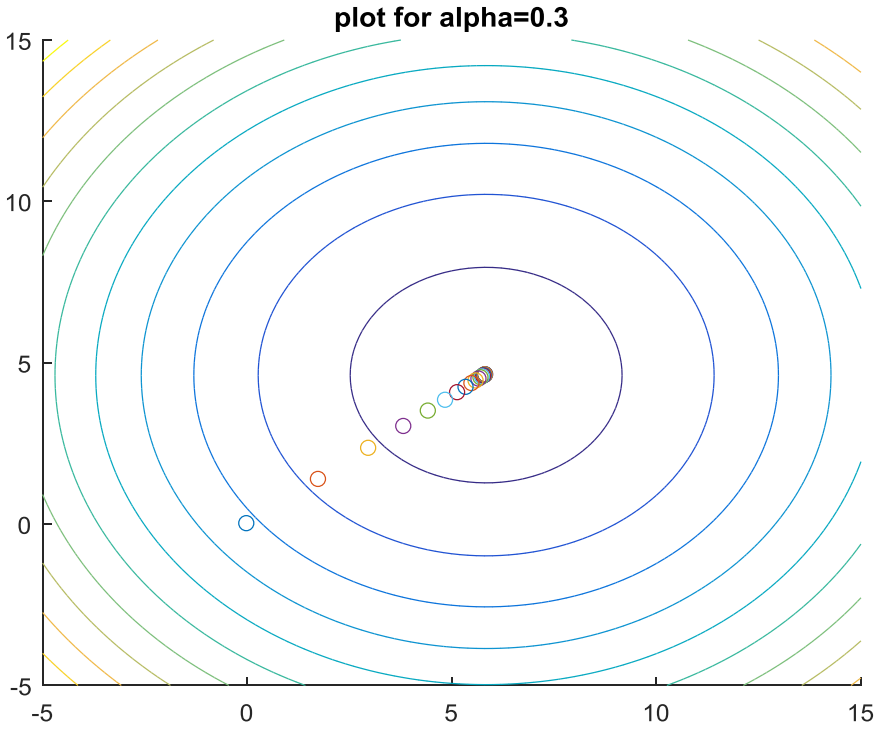
b. Plot data and hypothesis obtained



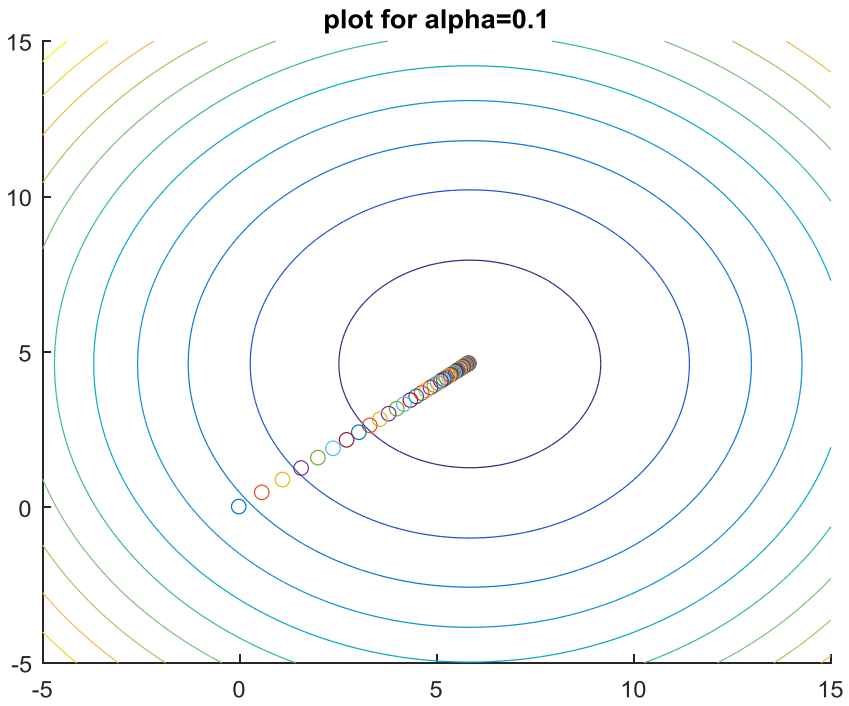
c. Draw 3d mesh



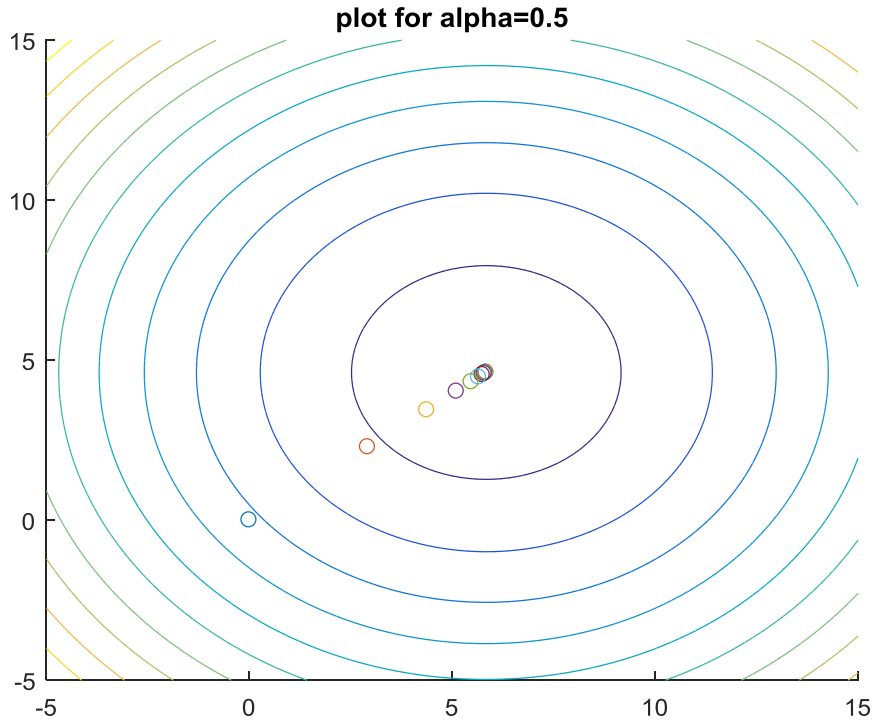
c. Draw contour for your learning rate



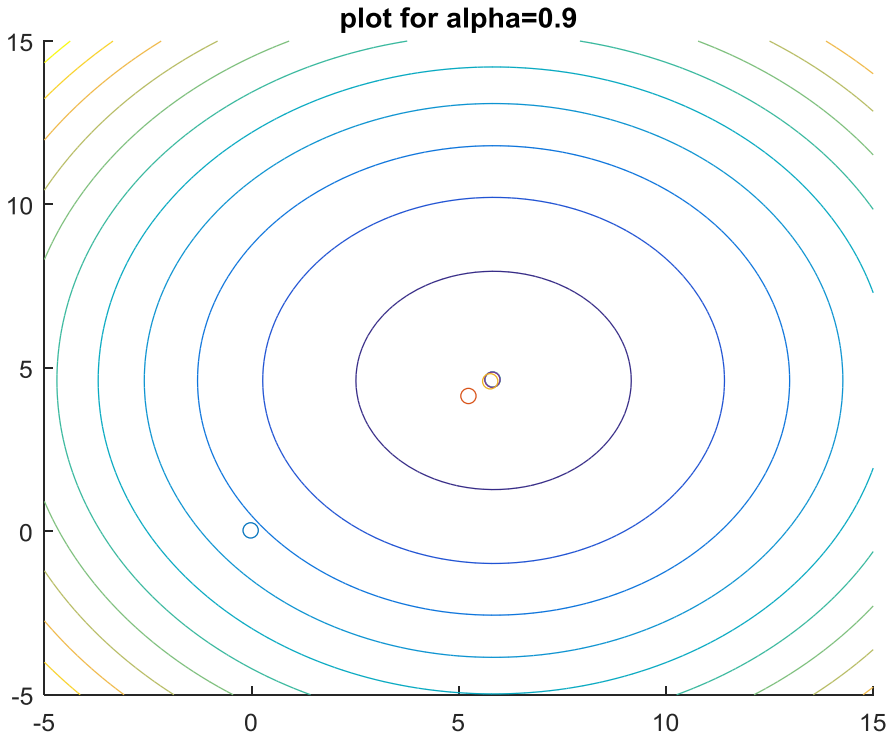
d. Contour for learning rate=0.1



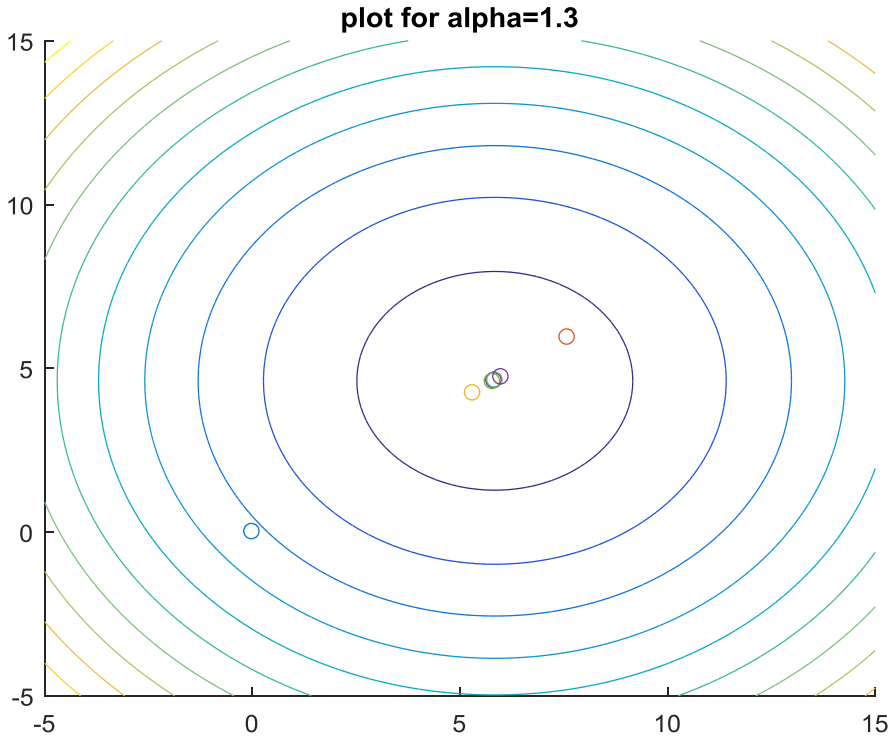
Contour for learning rate=0.5



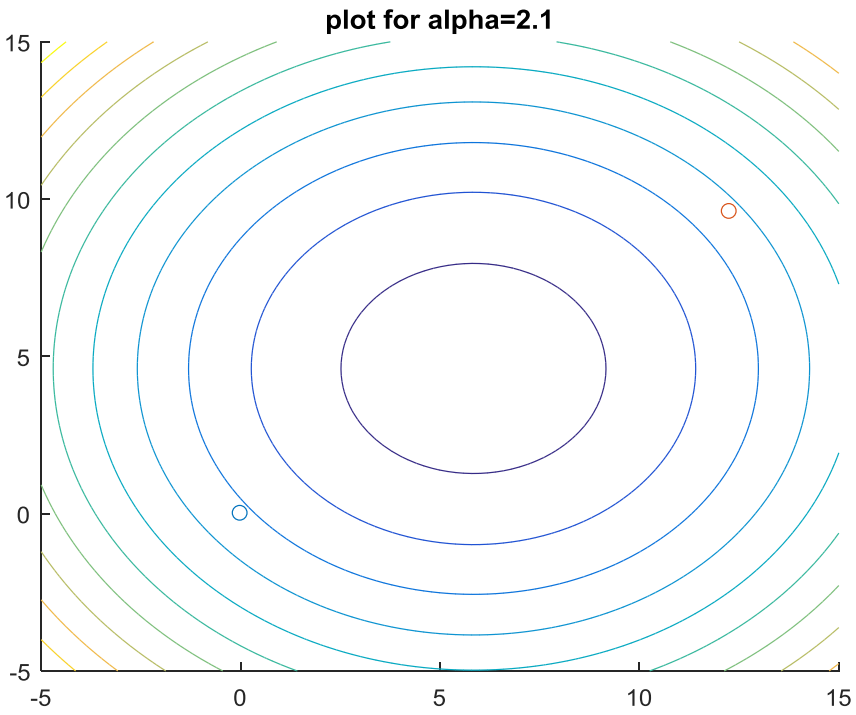
Contour for learning rate=0.9



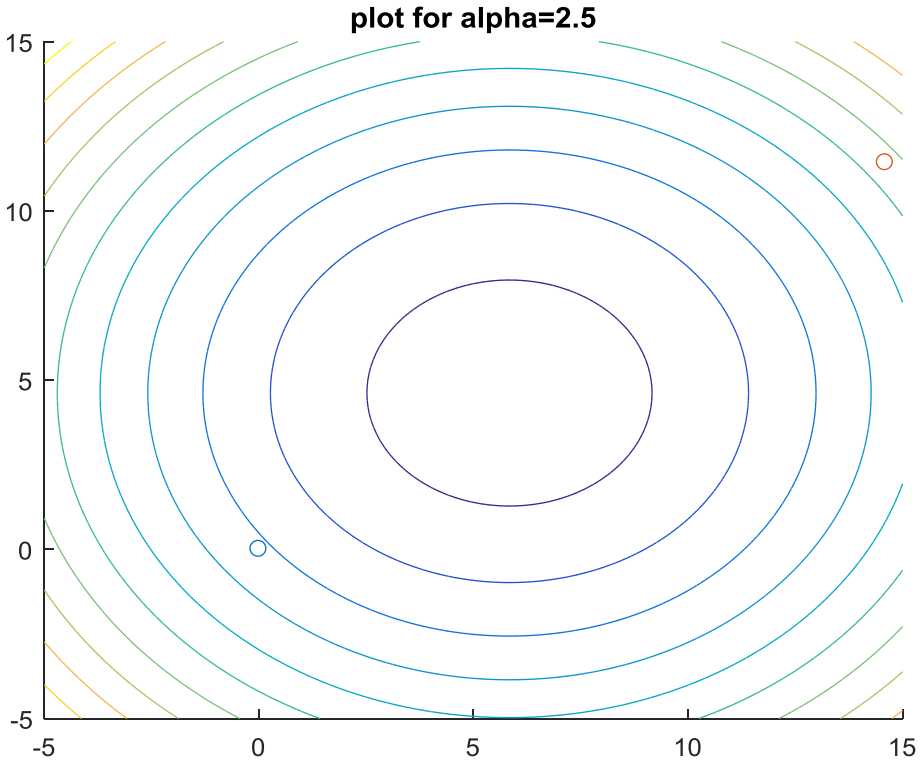
Contour for learning rate=1.3



Contour for learning rate=2.1



Contour for learning rate=2.5



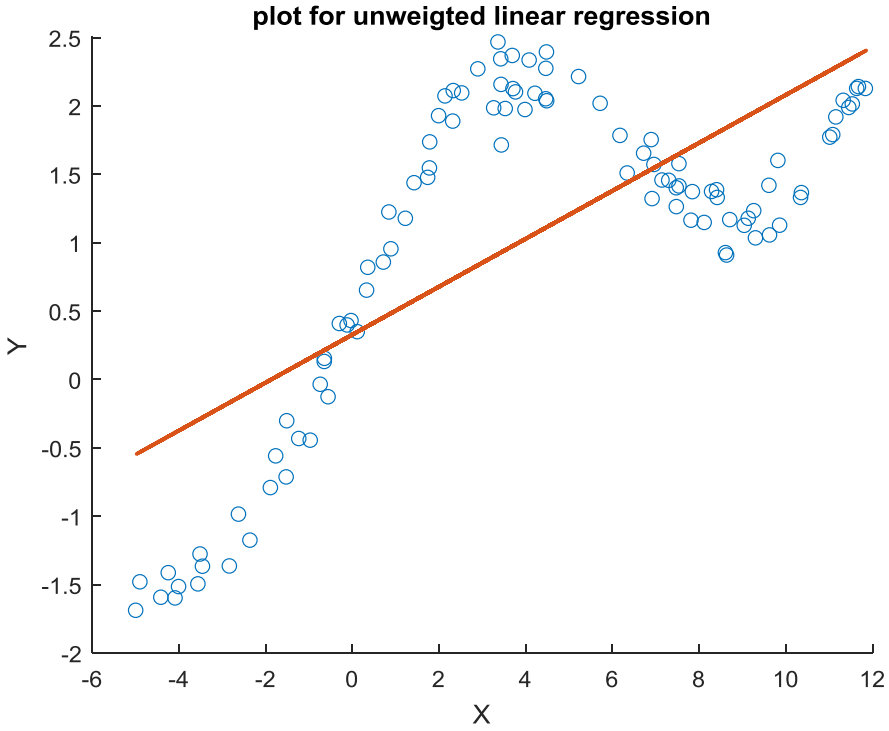
From above, we can observe that with a very low learning rate of 0.1, algorithm took too many steps to converge, i.e. with a very low learning rate algorithm converges too slow. With a learning rate of 0.5, algorithm performed quiet well as

it converged in lesser no. of steps. But with a rate of 0.9, algorithm converged very fast i.e. in just two three steps. Whereas for a high learning rate of 1.3, 2.1 and 2.5, algorithm over-shoot from minimum value i.e. instead of converging, it started to diverge. So, a **learning rate** should be neither too low nor too high as in the case of **0.5.**

Ques2:

a. Implement un-weighted linear regression using normal equation and plot the hypothesis.

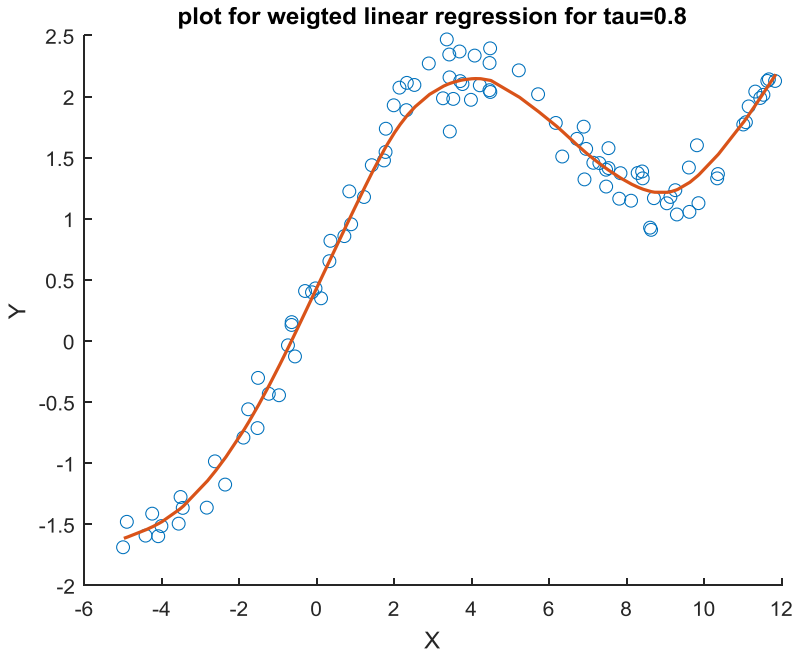
First, the theta is initialized with zeros and the optimal theta solution is obtained using normal equation. With this, **theta** calculated was **[0.3277; 0.1753]** and **cost** that was calculated was **0.3334.**



b. Implement locally weighted regression with tau=0.8 using normal equation.

Normal equation for theta was calculated by finding derivative of J theta for weighted regression and setting it to zero. By this theta was calculated as: $\mathbf{(X^T W X)^{-1} X^T W X}$

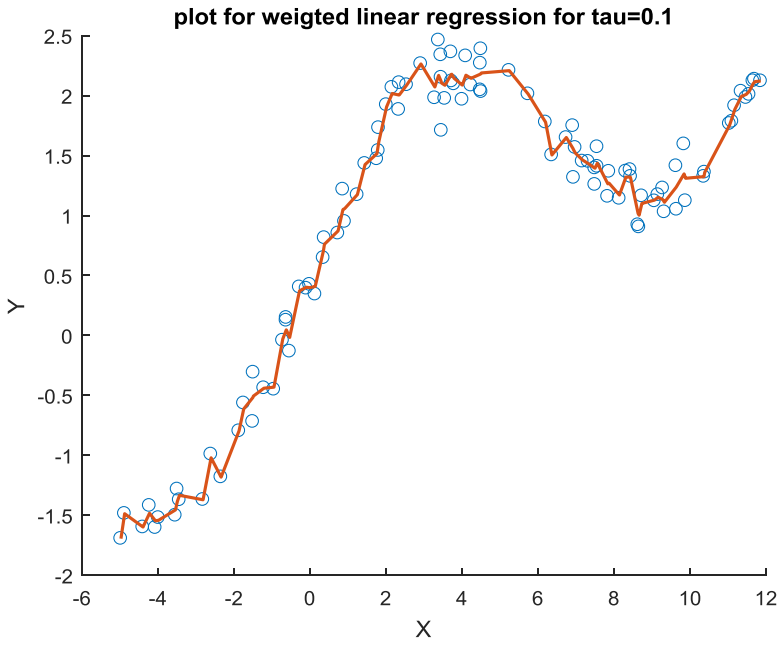
For, each training example weight matrix was calculated, corresponding theta was calculated using the above normal equation and then estimated value was evaluated and finally the graph was plotted.



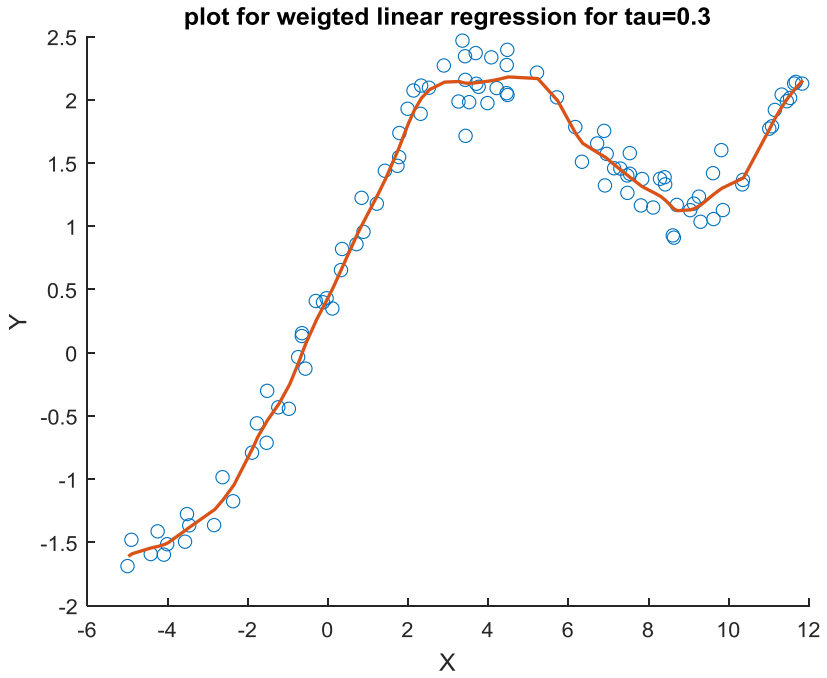
c. Implement locally weighted regression for tau=0.1, 0.3, 2 and 10

Same procedure is repeated for each value of tau and following graphs were plotted:

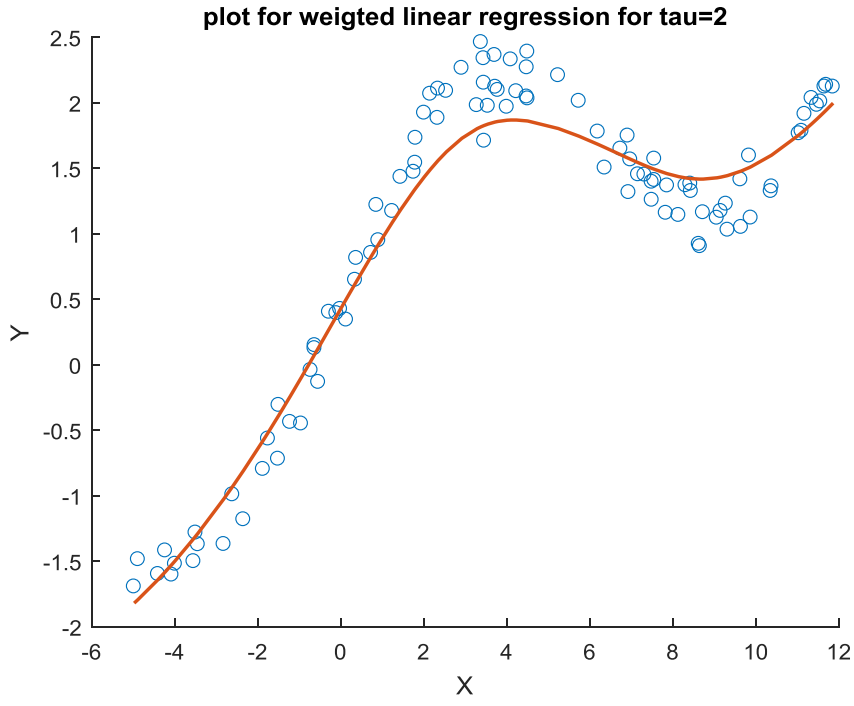
Tau=0.1



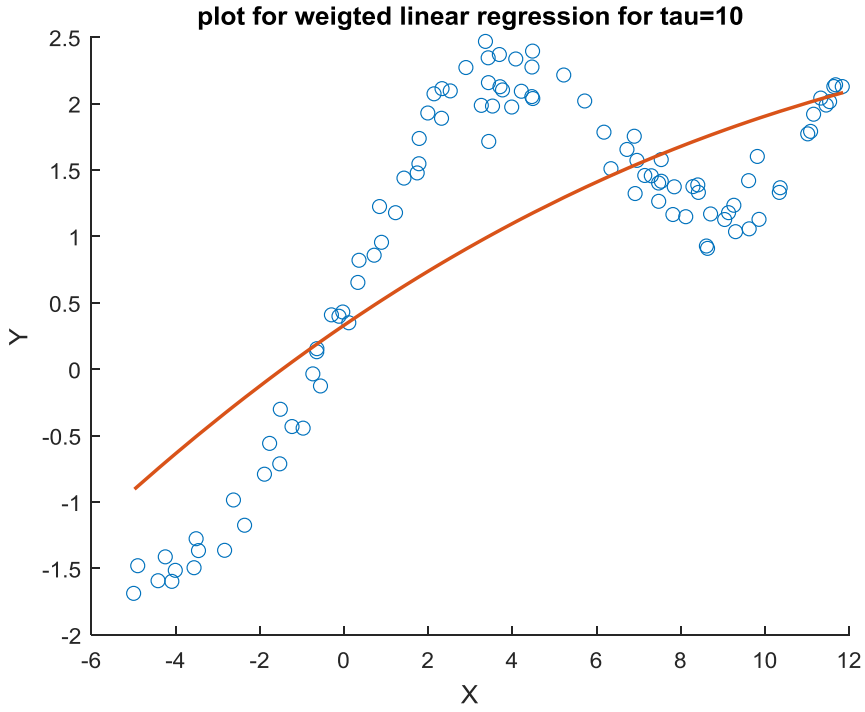
Tau=0.3


plot for weigted linear regression for tau=0.3
Y
X
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-6
-4
-2
0
2
4
6
8
10
12

Tau=2


plot for weigted linear regression for tau=2
Y
X
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-6
-4
-2
0
2
4
6
8
10
12

Tau=10



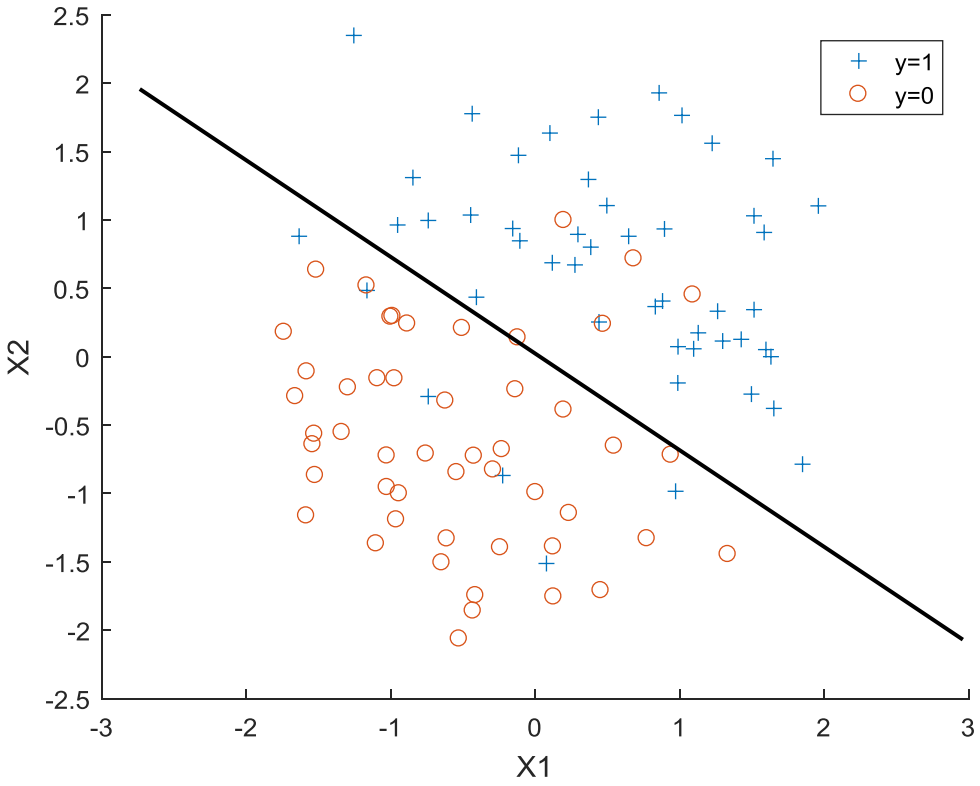
From above, we observed that, with a very low value of tau over fitting occurs and thus it may not generalize for the entire data set whereas with a very high value of tau, under-fitting occurs and thus it may not give a proper estimated value. Also, with a very high tau value, it starts giving results almost same as linear regression. So, the most **appropriate value of Tau** is one which is neither too low nor too high i.e. **0.8**.

Ques3:

a. Implement Newton's method for logistic regression.

For implementing logistic regression using Newton's method, first started the algorithm with initial value of theta set to zeros. Also the data was normalized so that algorithm could converge in less number of iteration. Then, for each iteration of loop, derivative of J theta as well as Hessian was calculated. With this, new value of theta is calculated in each iteration by subtracting the product derivative of J theta and inverse of Hessian matrix at previous value of theta from previous value of theta. **Theta** obtained was **[-0.0472; 1.4675; 2.0764].**

b. Plot training data and decision boundary



Ques4:

a. Implement Gaussian Discriminant Analysis with $\sum_1=\sum_2=\sum$ and report values of μ_1, μ_2, $\emptyset$ and $\sum$

- $\emptyset$ was calculated using the following formula:

$$\emptyset = \frac{\sum_{i=1}^{m} 1\left\{y^i = 1\right\}}{m}$$

Resulting $\emptyset$=**0.5**

- μ_1 was calculated using the following formula:

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^i = 0\}\, x^i}{\sum_{i=1}^{m} 1\{y^i = 0\}}$$

Resulting $\mu_1 = \begin{matrix} -0.7515 \\ 0.6817 \end{matrix}$

- μ_2 was calculated using the following formula:

$$\mu_2 = \frac{\sum_{i=1}^{m} 1\{y^i = 1\}\, x^i}{\sum_{i=1}^{m} 1\{y^i = 1\}}$$

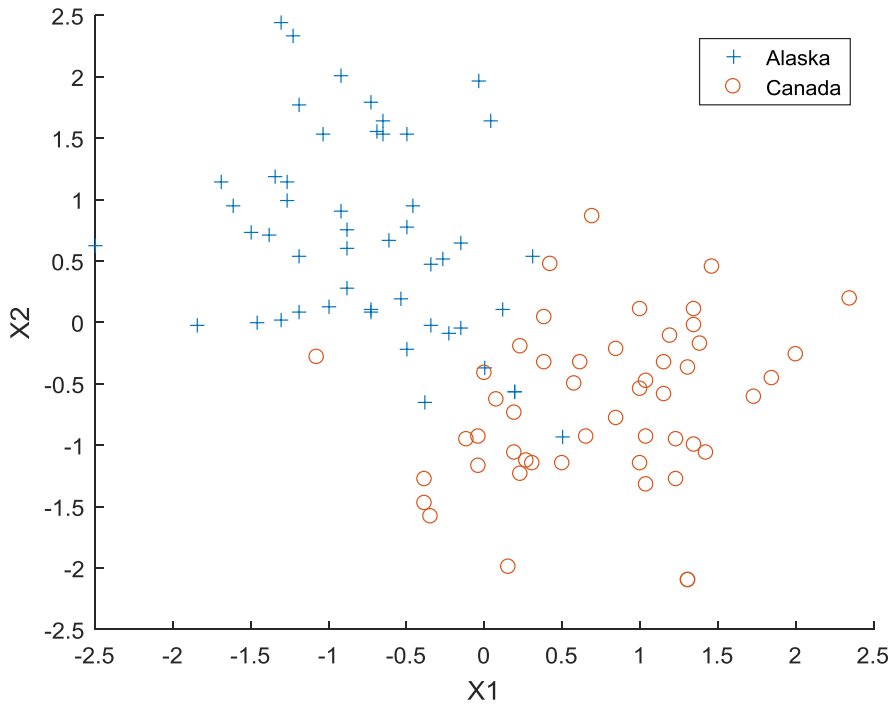
Resulting $\mu_2 = \begin{matrix} 0.7515 \\ -0.6817 \end{matrix}$

- $\sum$ calculated using the following formula:

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

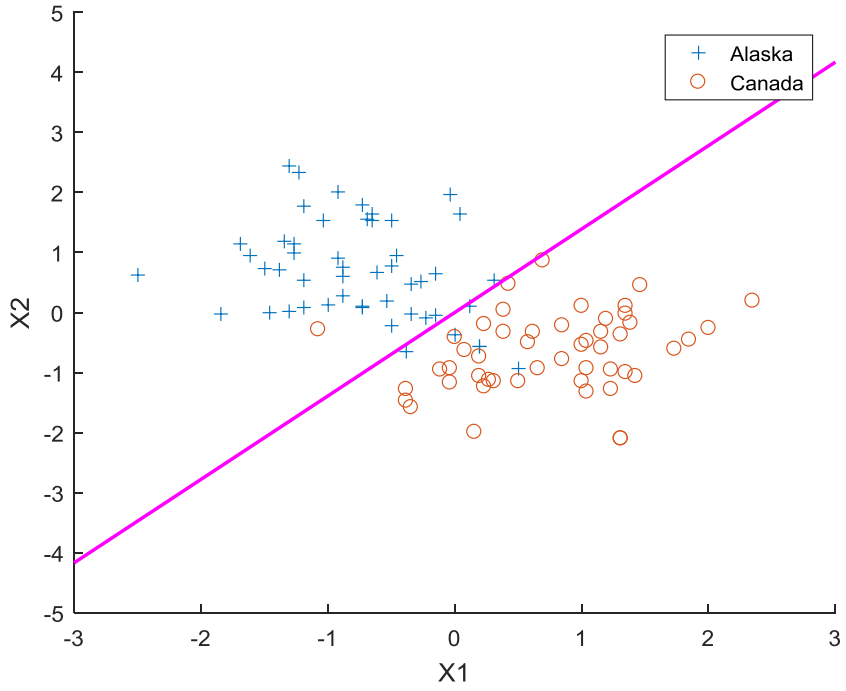
Resulting $\sum = \begin{matrix} 0.4252 & -0.0222 \\ -0.0222 & 0.5253 \end{matrix}$

b. Plot data



c. Describe the equation of linear boundary and plot decision boundary.

$$\left(-\frac{1}{2}\right)\left[-2\left(\mu_2{}^T\sum{}^{-1}x\right)+\mu_2{}^T\sum{}^{-1}\mu_2+2\left(\mu_1{}^T\sum{}^{-1}x\right)-\mu_1{}^T\sum{}^{-1}\mu_1\right]=log\left(\frac{1-\emptyset}{\emptyset}\right)$$



d. Implement Gaussian Discriminant Analysis with $\sum_1 \neq \sum_2$ and report values of μ_1, μ_2, $\emptyset$, $\sum_1$, $\sum_2$.

For this, the μ_1, μ_2, $\emptyset$ will remain same as above. Only the values of $\sum_1, \sum_2$ will be calculated as follows:

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

$$\sum 2 = \frac{\sum_{i=1}^{m}\{y^i=1\}(x^i-\mu_{y^i})(x^i-\mu_{y^i})^T}{\sum_{i=1}^{m}1\{y^i=1\}}$$

Resulting ∑1= **0.3778 -0.1533**

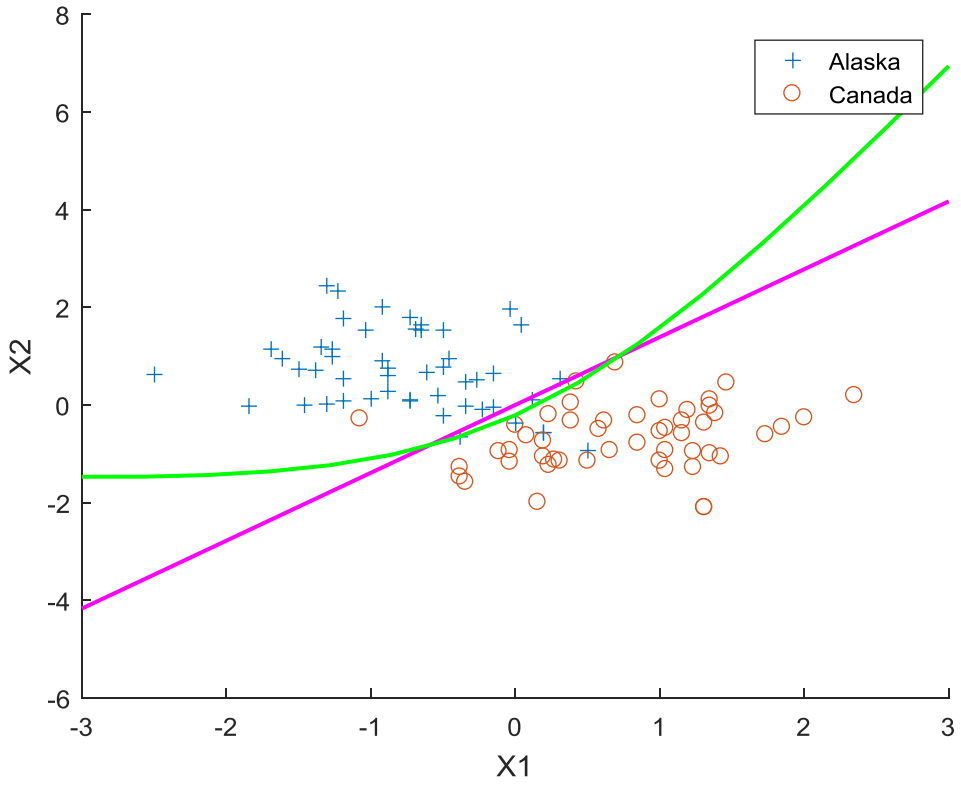
-0.1533 0.6413

Resulting ∑2= **0.4727 0.1088**

0.1088 0.4094

e. Describe the equation of quadratic boundary and plot decision boundary.

$$log\frac{|\Sigma_1|}{|\Sigma_2|}+\left(-\frac{1}{2}\right)\left[x^T\sum_2^{-1}\mu_2-2\left(\mu_2{}^T\sum_2^{-1}x\right)+\mu_2{}^T\sum_2^{-1}\mu_2-x^T\sum_1^{-1}\mu_1+2\left(\mu_1{}^T\sum_1^{-1}x\right)-\mu_1{}^T\sum_1^{-1}\mu_1\right]=log\left(\frac{1-\emptyset}{\emptyset}\right)$$



f. Analyze the two boundaries.

The linear boundary is plotted the same way as in case of logistic regression and hence a linear boundary is plotted where the probability of (y=1|x;θ) = probability of (y=0|x;θ) as the covariance matrixes of two classes are exactly same so it won't

result in the boundary bend towards either direction. But in case of quadratic boundary, co-variances matrixes for the two classes are different, which in turn results in a quadratic boundary. As we can observe that the off diagonal entries of first covariance matrix are negative, so its data will be distributed more towards –x1 and +x2 upward direction whereas the off diagonal entries of second covariance matrix are positive, so its data will be distributed more towards +x1 and +x2 upward direction. So, this type distribution results in the type of quadratic boundary shown in the figure which is convex towards the first class distribution.