Assignment 1 Report

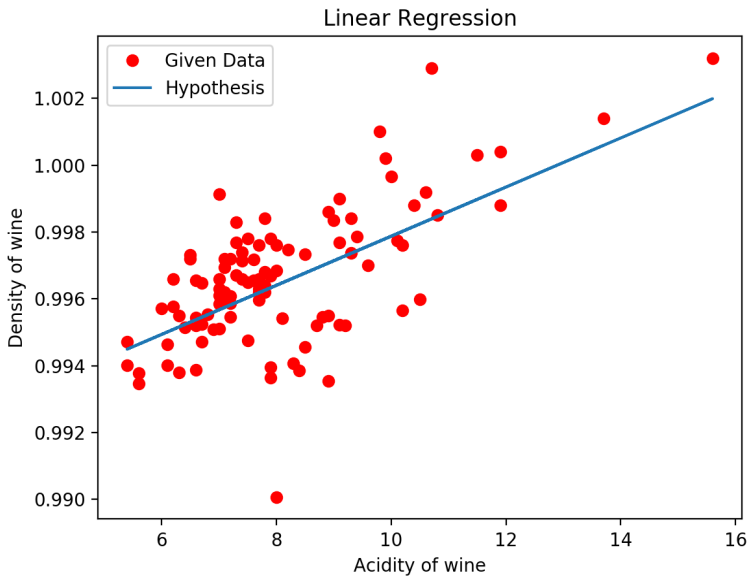
Pradyumna Meena - 2016CS10375

Q1

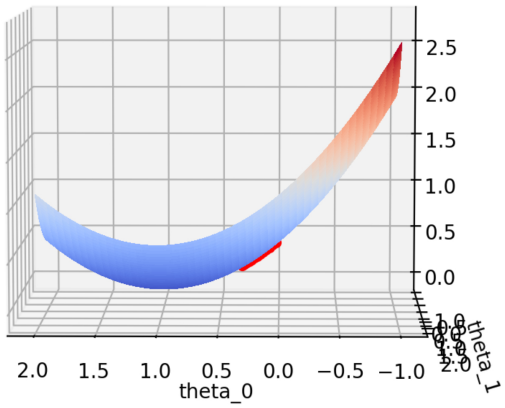
Observations for various learning rates where test-set consists the last 20 data-points from the given data-set. Hence the learning rate was chosen and used thereafter.

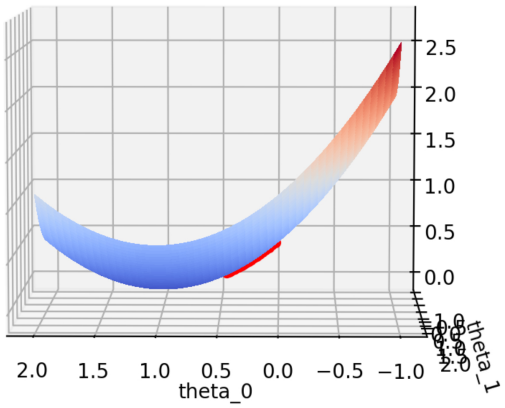
Learning Rate	Iterations	Cost (training-set) (10^{-6})	Cost (test-set) (10^{-6})
0.001	9220	1.4100187465131329	0.9759962195011227
0.003	3513	1.3305684468289872	0.8737000319310789
0.01	1206	1.3008637736256783	0.8186746547546557
0.03	448	1.2921961609153327	0.7928671381016129
0.1	148	1.2891887819930612	0.7777339925937648
0.3	52	1.288329165073939	0.7698203011594116

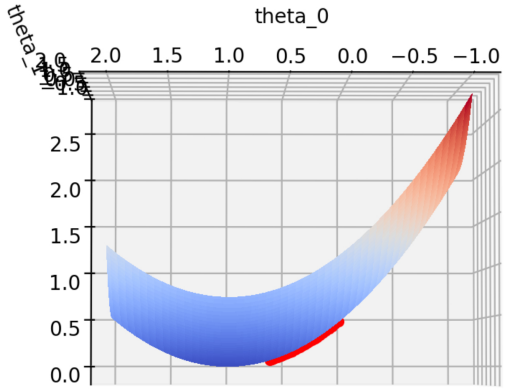
- Stopping criteria: Change in cost < 10^{-9}
- Results
  - Learning Rate = 0.03 (To avoid having large steps in other cases)
  - Parameters = [*0.99657156 , 0.00222505*]

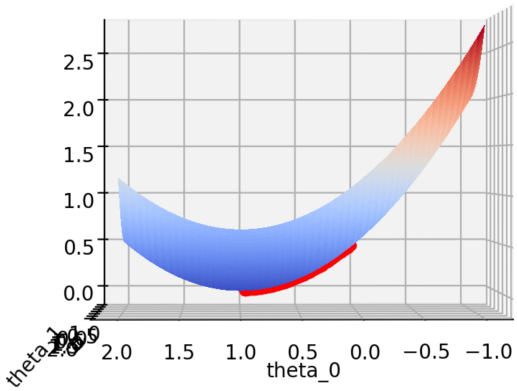


Linear Regression Plot

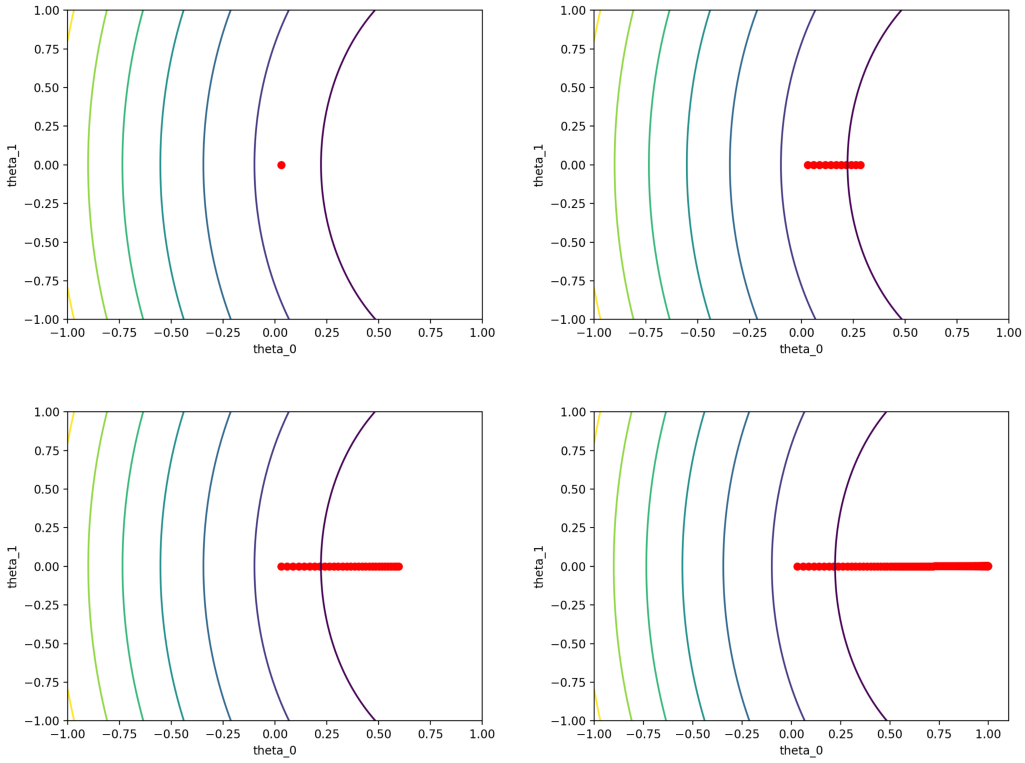




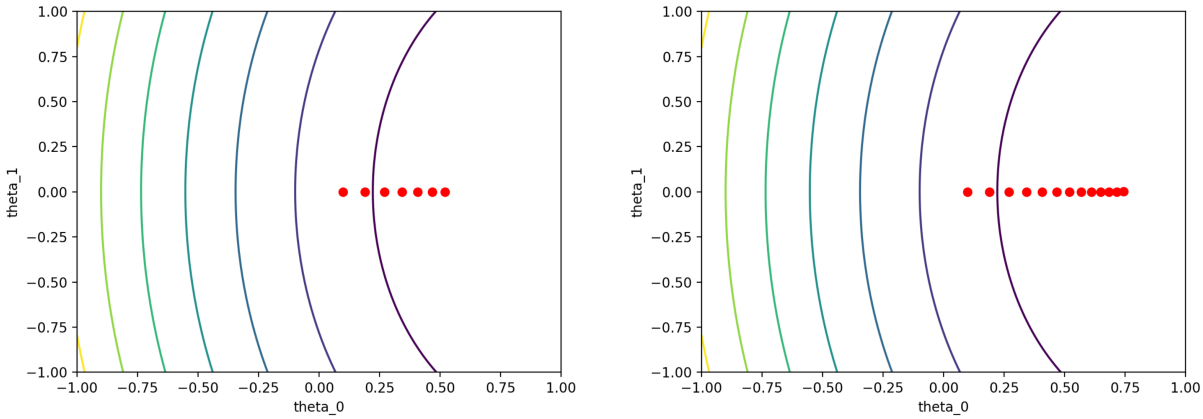




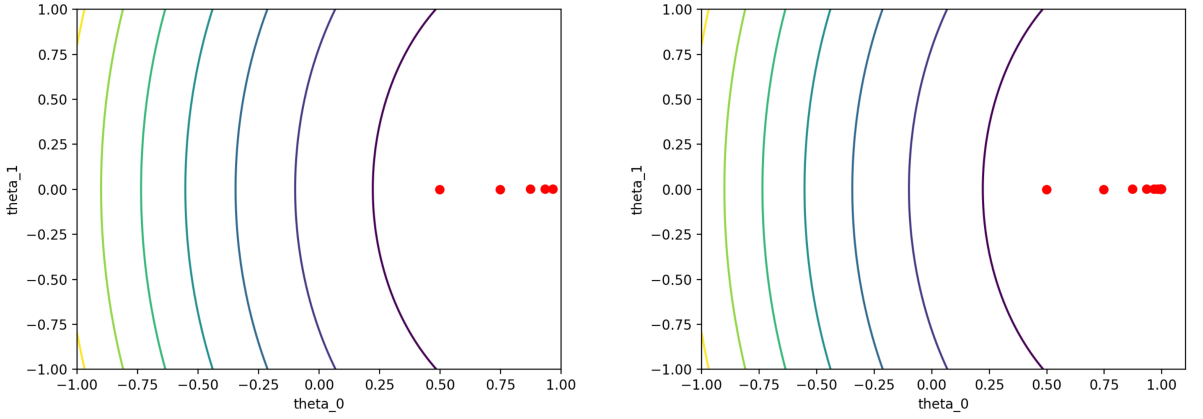
3-D mesh plot for cost function varying with parameters



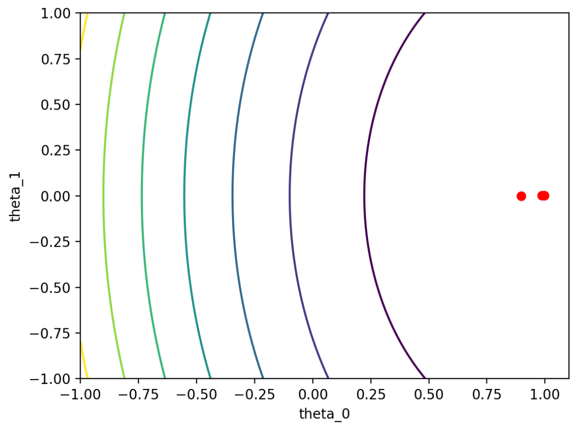
Contour plots for step size = 0.03 (chosen learning rate)



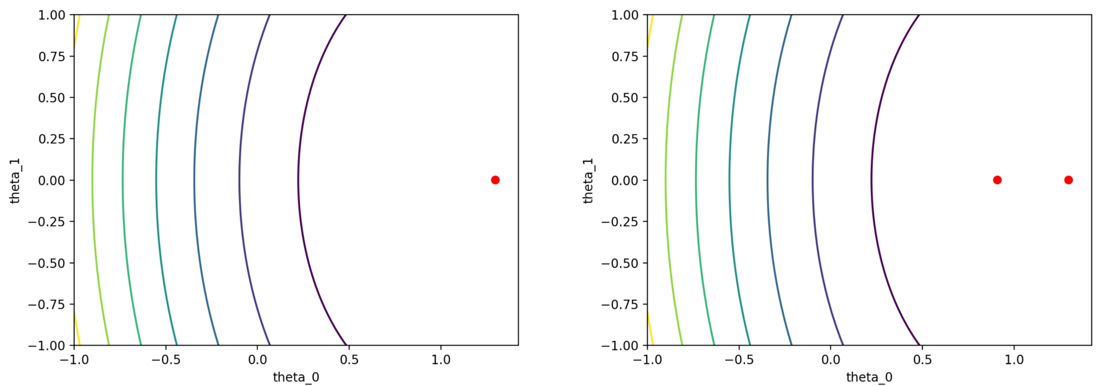
Contour plots for step size = 0.1



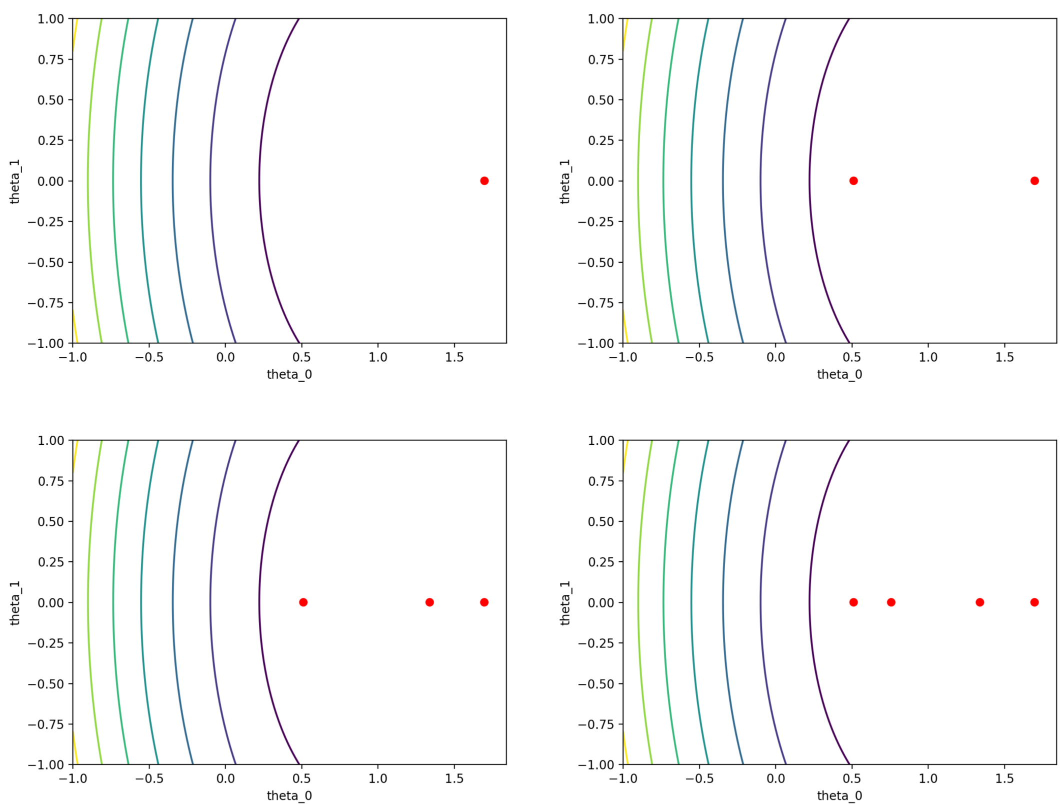
Contour plots for step size = 0.5



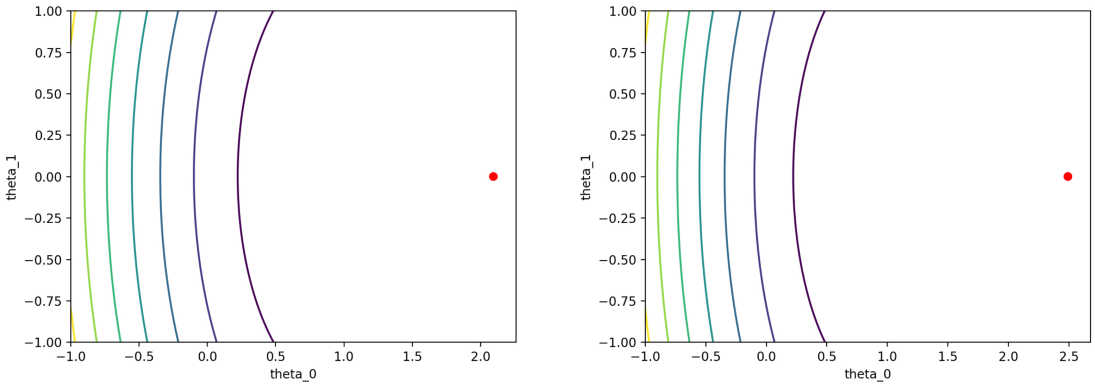
Contour plots for step size = 0.9



Contour plots for step size = 1.3



Contour plots for step size = 1.7



Contour plots for step size = 2.1 and 2.5 (Only one iteration took place in both cases)

As we increase the step size the algorithm converges in fewer number of steps. However as discussed in class if we take a quite larger step size then the gradient descent oscillates around minima as is seen in the case with 1.7 as the step size. However the next two ones are the interesting ones. They take only one iteration to converge i.e. in a single step they fulfill the criteria of convergence which was change in cost should be less than 10^{-9}. It is because after this the costs start to rise again and hence the difference between old and new become negative. This is the reason behind the convergence with these step sizes.

Q2

The equation can be written as $J(\theta) = (X\theta-Y)^T W(X\theta-Y)$

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

$= \theta^T X^T W X\theta - \theta^T X^T WY - YWX\theta + YWY$

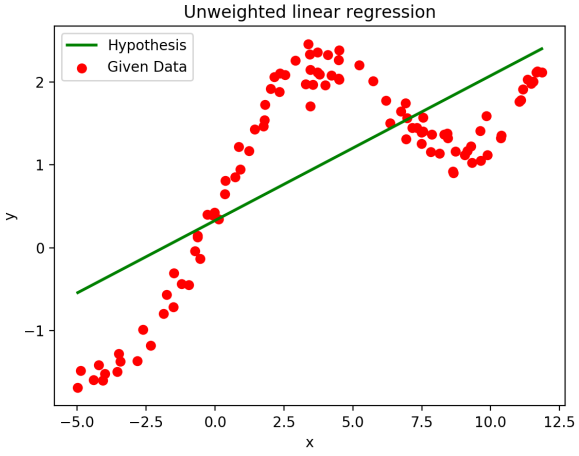
Put $\nabla_\theta J(\theta) = 0$ $\Rightarrow 2X^T WX\theta - X^T WY = 0$

$\Rightarrow 2\theta = (X^T W)^{-1}(X^T WY)$

Hence $\theta = ((X^T W)^{-1}(X^T WY))/2$

Part A:

- Convergence criteria: change in cost < 10^{-6}.

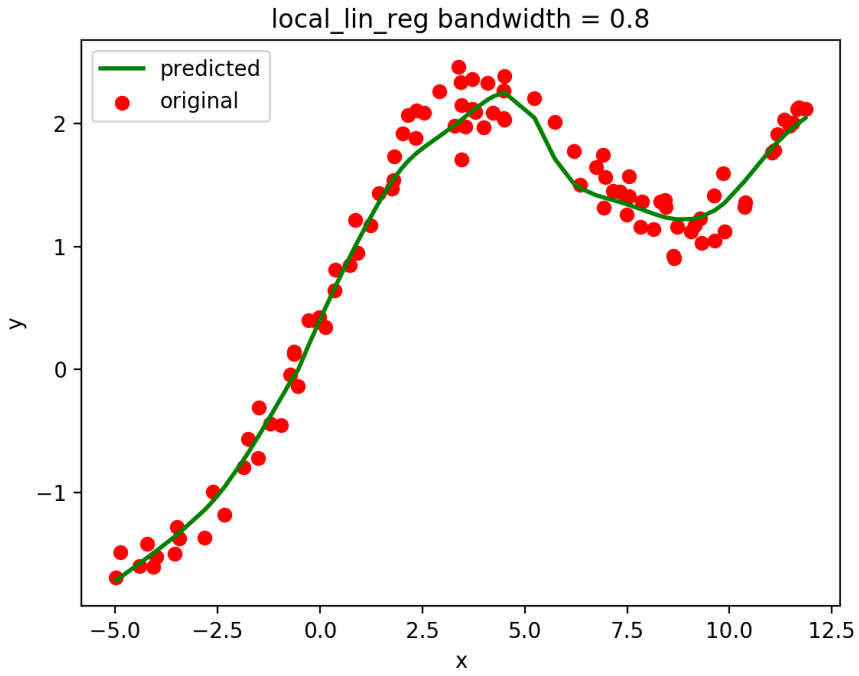


Learning Rate	Iterations	Cost (test-set) (10^{-6})	Cost (training-set) (10^{-6})
0.001	39777	300126.28362072624	351988.90767426966
0.003	17258	311764.03390232527	344707.5024437655
0.03	2563	325761.0696255268	341430.4292296926
0.3	338	330830.6240484761	341103.2128473918

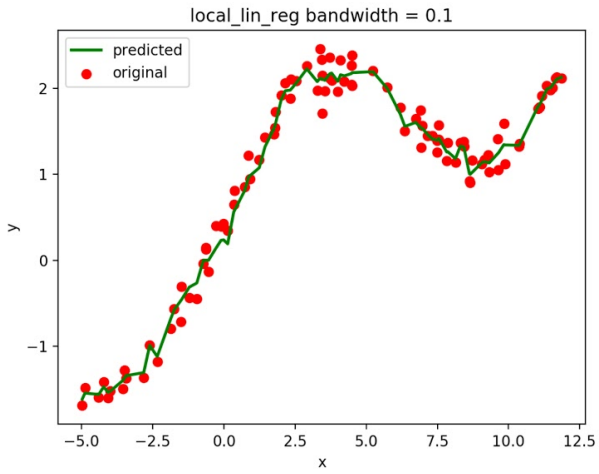
Since its a non-linear curve, using linear hypothesis is giving huge errors in prediction. Above plot is for ***α = 0.03*** and parameters obtained are [*1.03128116 , 3.84809705*].

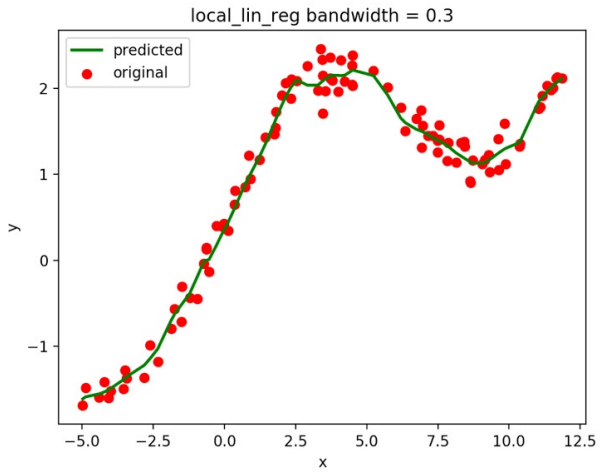
Part B:

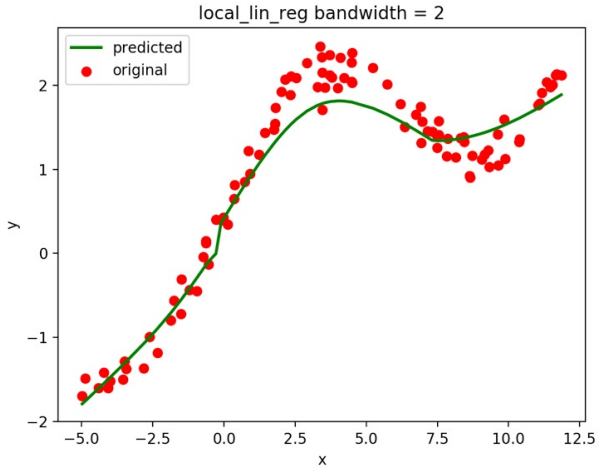
- Convergence criteria for underlying gradient descent: Change in predicted value<10^{-4}
- Learning rate for gradient descent = 0.03

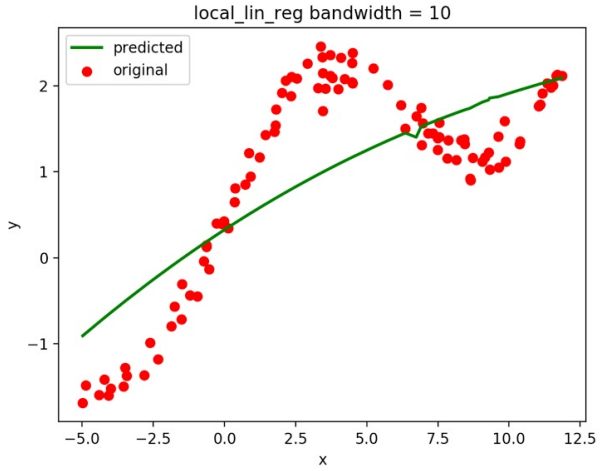


As the plot suggests choosing choosing 0.8 as the bandwidth parameter gives the best results. The bandwidth parameter essentially captures the proximity of the points which are important for prediction on the current point. Very low bandwidth means we are only focussing on the points which are closest i.e. not fitting into general behaviour but the local behaviour. Very high value of bandwidth means we are underfitting the curve since we are trying to capture behaviour of those points also which are very far away from me. Hence very low and very high value of bandwidth parameter leads to overfitting and underfitting respectively.

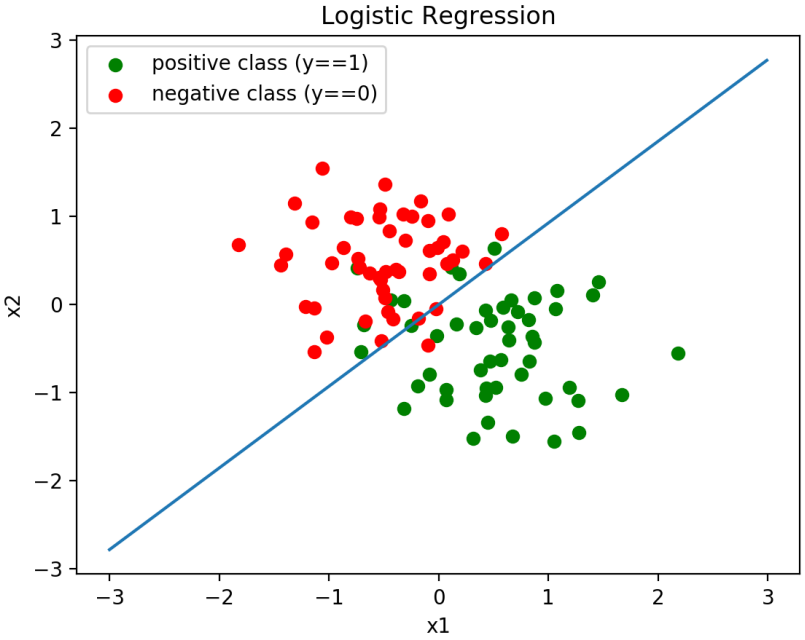








Q3



The parameters obtained are [-1.67966358e-16 , 1.18352831 , -1.27634840]

Q4

A.

μ_0 = [1 , 0.02919496 , -0.01489068] μ_1 = [1 , -0.02919496 , 0.01489068]

$$\sum = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.00149411 & -0.00045361 \\ 0 & -0.00045361 & 0.00047242 \end{bmatrix}$$

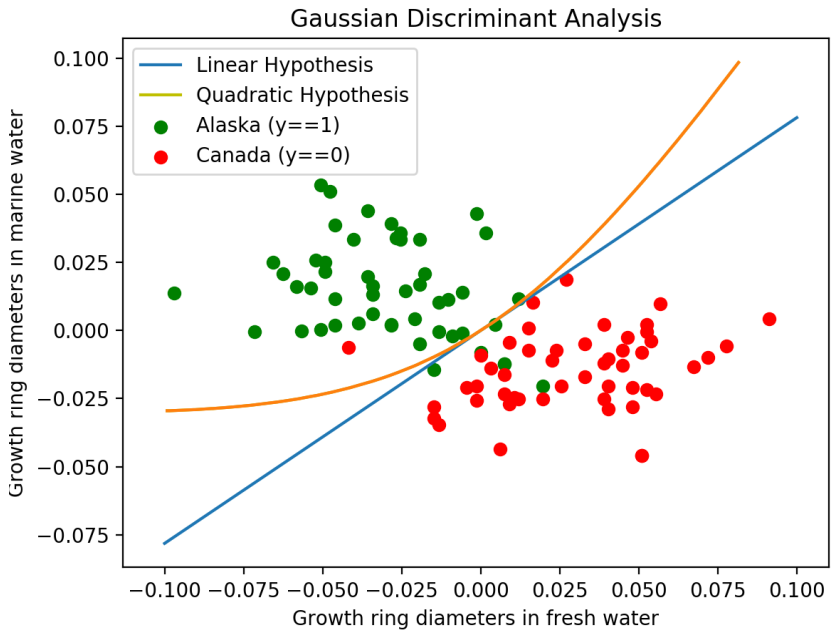
D.

μ_0 = [1 , 0.02919496 , -0.01489068]
μ_1 = [1 , -0.02919496 , 0.01489068]

$$\sum_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.000713395 & 0.000092349 \\ 0 & 0.000092349 & 0.000195371 \end{bmatrix}$$

$$\sum_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.00057014 & -0.00013011 \\ 0 & -0.00013011 & 0.000306 \end{bmatrix}$$

B.



C.

Ø = 0.5 (since equal number of data-points from each class)

μ_0 = [1 , 0.02919496 , -0.01489068]

μ_1 = [1 , -0.02919496 , 0.01489068]

$\sum^{-1}$ = [[0,0,0],[0,944.68112741,907.07305869],[0,907.07305869,2987.72257965]]

When both the covariance matrices are same the equation essentially reduces to the form

$$2(\mu_1^T - \mu_0^T)\sum^{-1}x + (\mu_0^T\sum^{-1}\mu_0 - \mu_1^T\sum^{-1}\mu_1) = log((1-Ø)/Ø)$$

E.

Ø = 0.5 (since equal number of data-points from each class)

Since both the covariance matrices are different we have the following equation

$\sum_0^{-1}-\sum_1^{-1}$ = [[0,0,0],[0,-449.3287854,-1531.6781],[0,-1531.6781,1832.97052567]]

$$x^T(\sum_0^{-1} - \sum_1^{-1})x + 2(\mu_1^T\sum_1^{-1} - \mu_0^T\sum_0^{-1})x + (\mu_0^T\sum_0^{-1}\mu_0 - \mu_1^T\sum_1^{-1}\mu_1) = log((1-Ø)/Ø)$$

F.

The boundary obtained is only with respect to the points available to us which are not much. It cannot be said in general that which will perform better. Here since the bulk of the data lies just inside the bulb of the quadratic curve, it was easy to seperate via both of the methods. However since quadratic curve requires two separate co-variance matrices it might happen that at times of deficiency of data the covariance matrices computed are not liable enough for predicting using this model. In such cases linear model will prevail.