MACHINE LEARNING (COL-774)

ASSIGNMENT-1 REPORT FILE

Submitted by:
Garima
2018MCS2017

Q1. Linear Regression (Normalised data)

(a) Learning rate chosen = 0.03

Stopping criteria = $J(\theta_{t+1}) - J(\theta_t) = 10 ** -15$

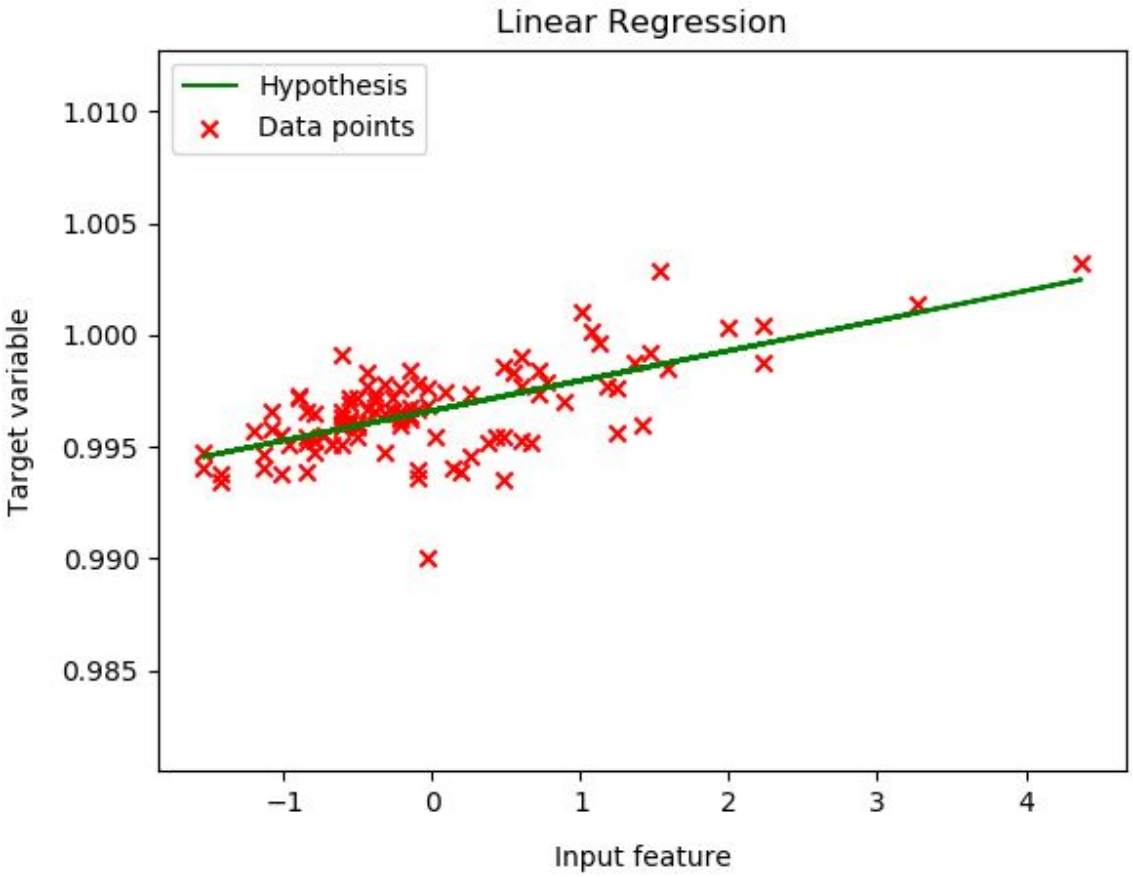
Final set of parameters obtained:

$\theta_0 = 0.99661993$

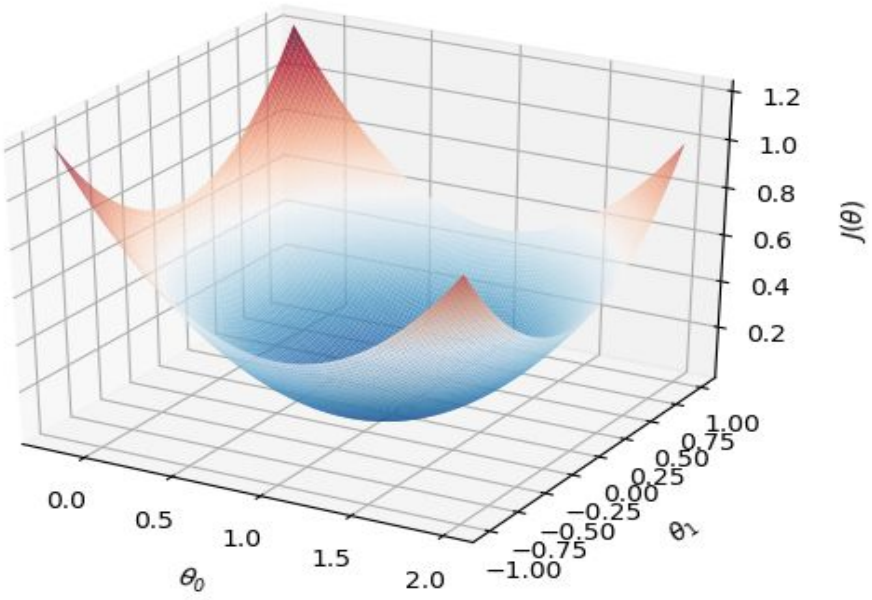
$\theta_1 = 0.0013402$

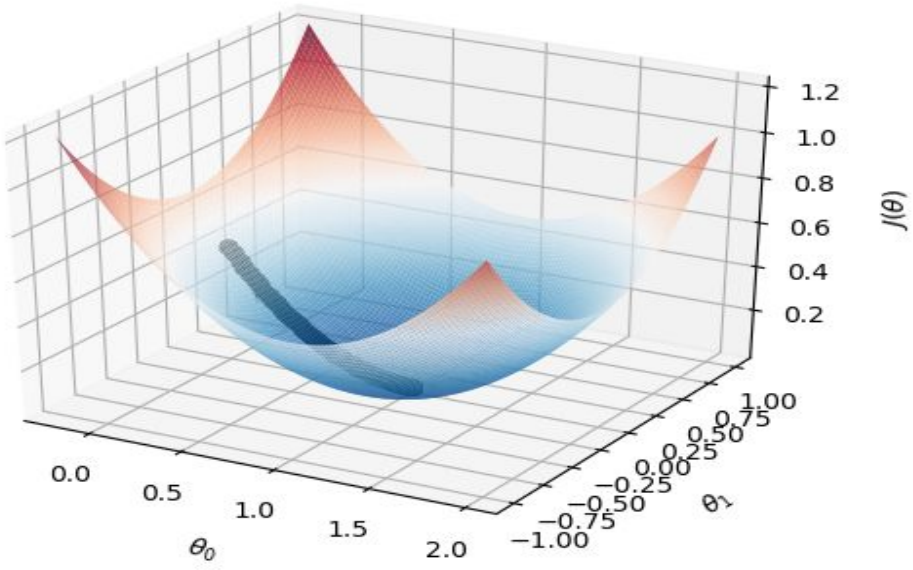
Number of iterations to converge = 511

(b)

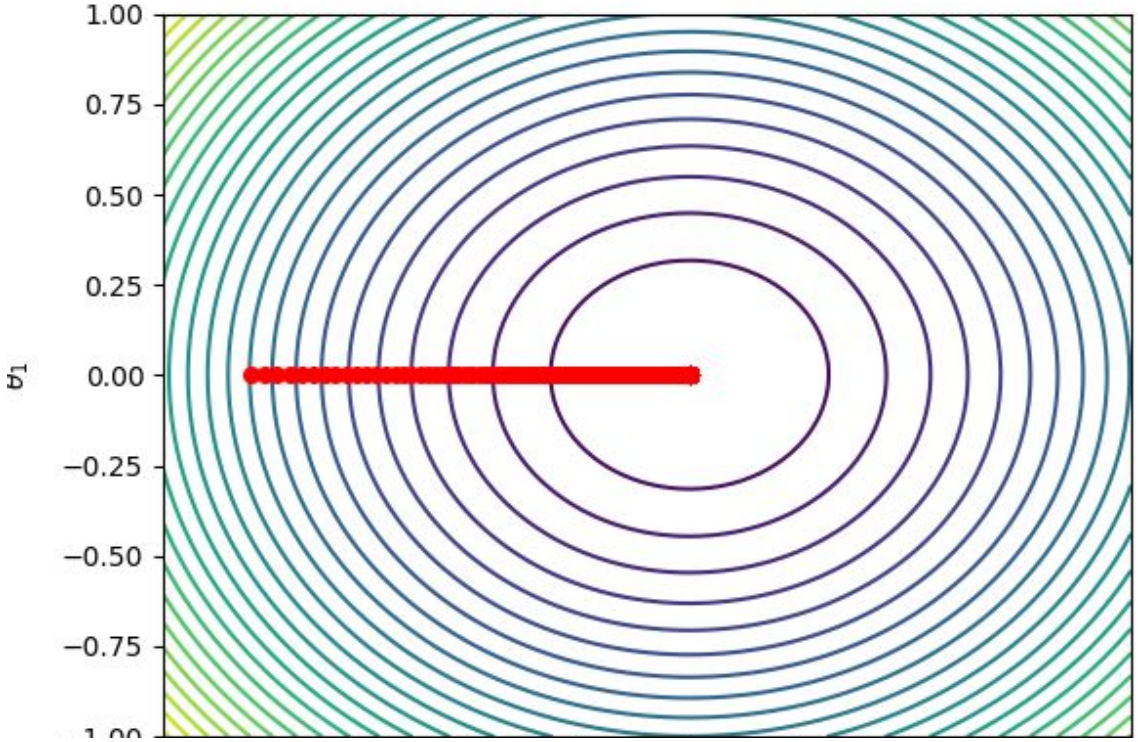


(c)

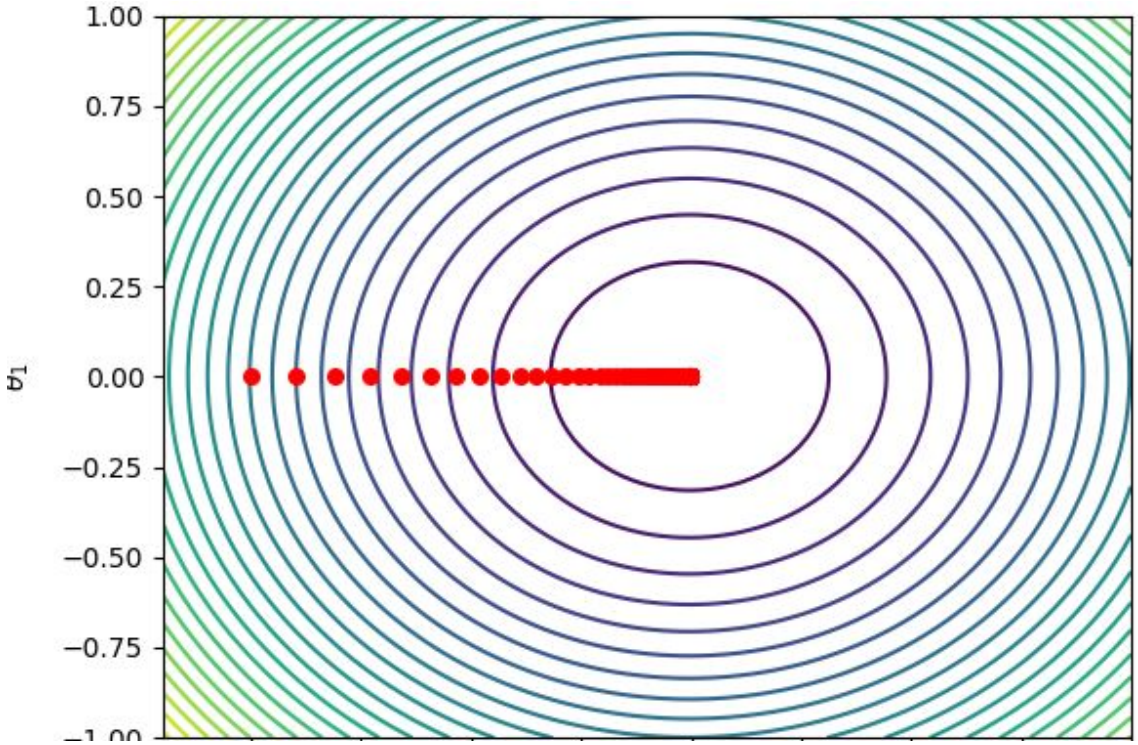

1.2
1.0
0.8
0.6
0.4
0.2
$J(\theta)$
0.0
0.5
1.0
1.5
2.0
θ_0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1


1.2
1.0
0.8
0.6
0.4
0.2
$J(\theta)$
0.0
0.5
1.0
1.5
2.0
θ_0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1

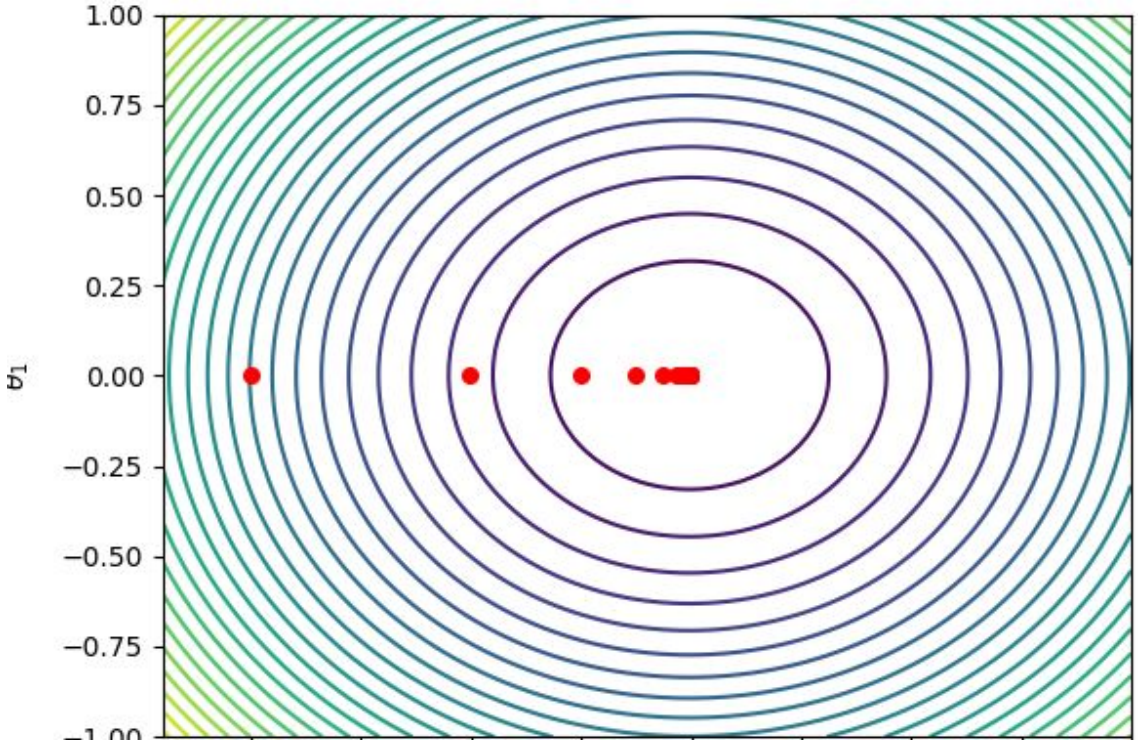
(d) **η= 0.03**


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
θ_1

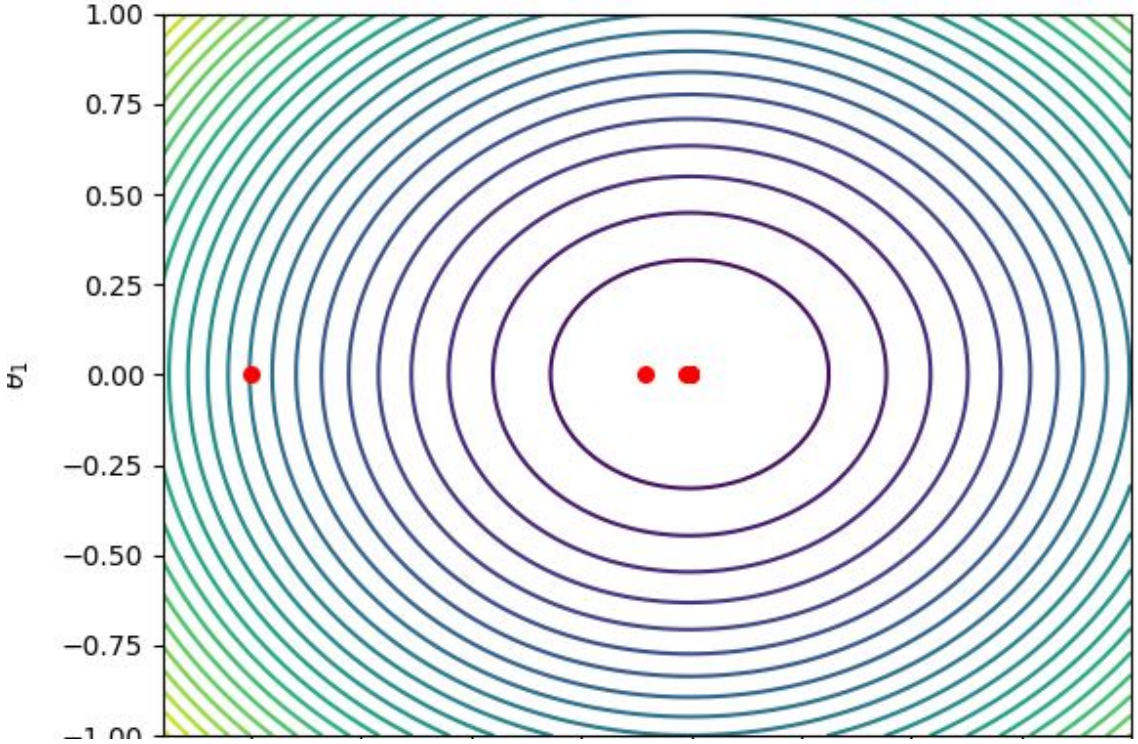
(e) **η = 0.1**


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
θ_1

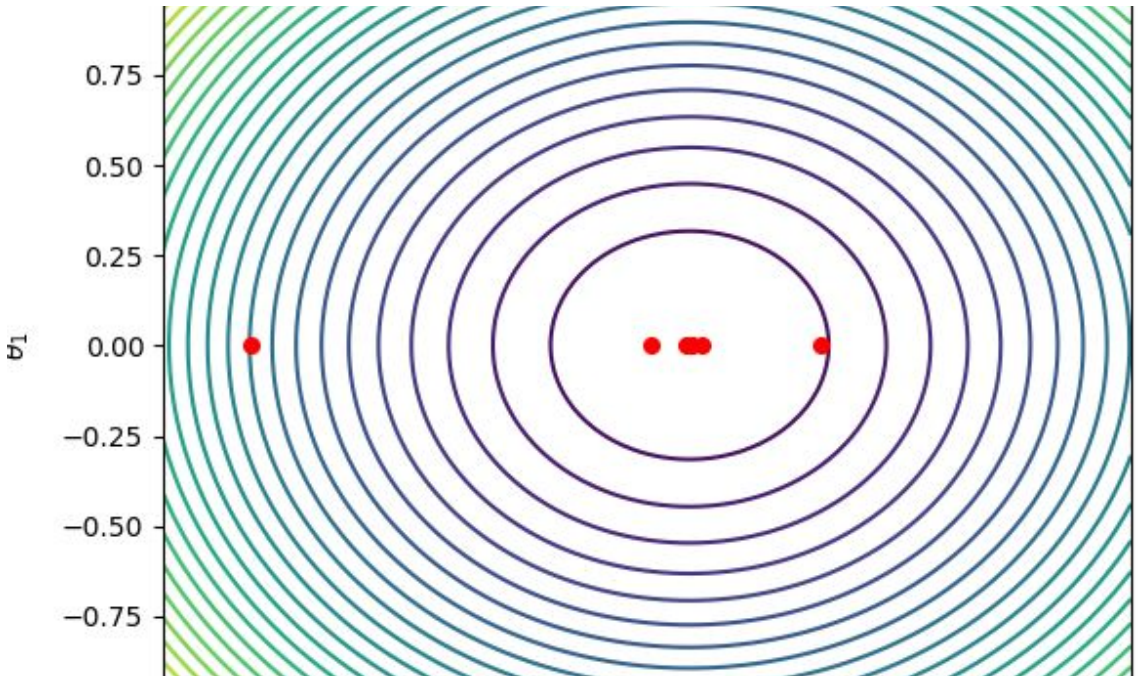
η = 0.5


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1

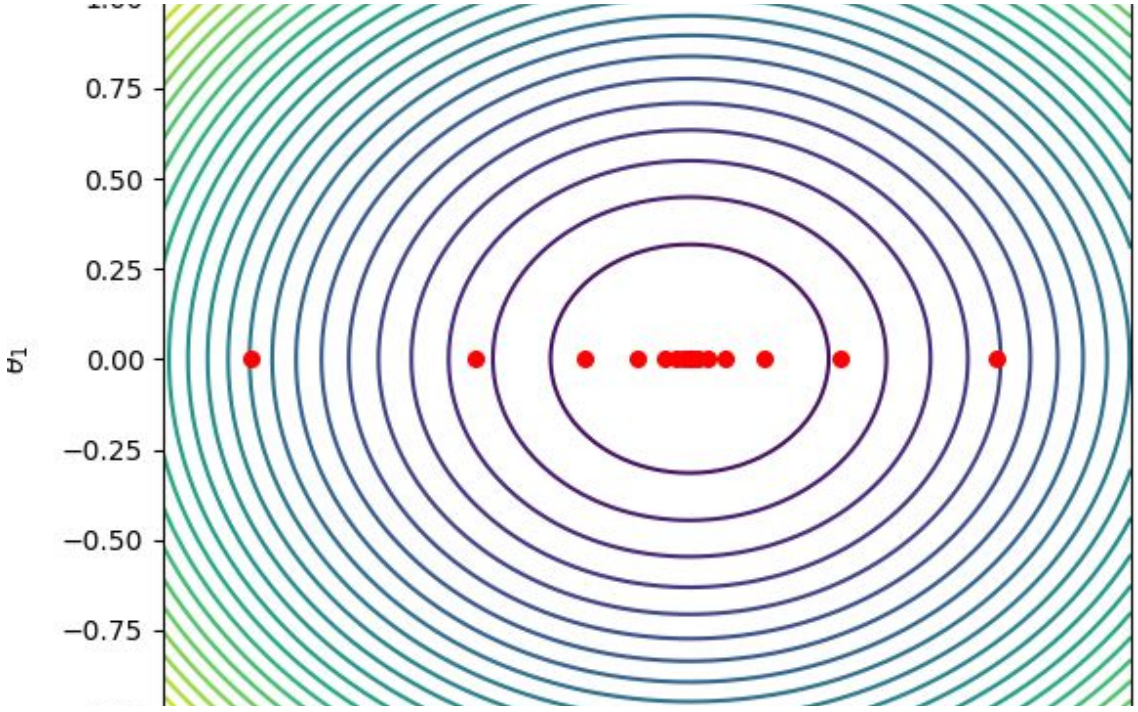
η = 0.9


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1

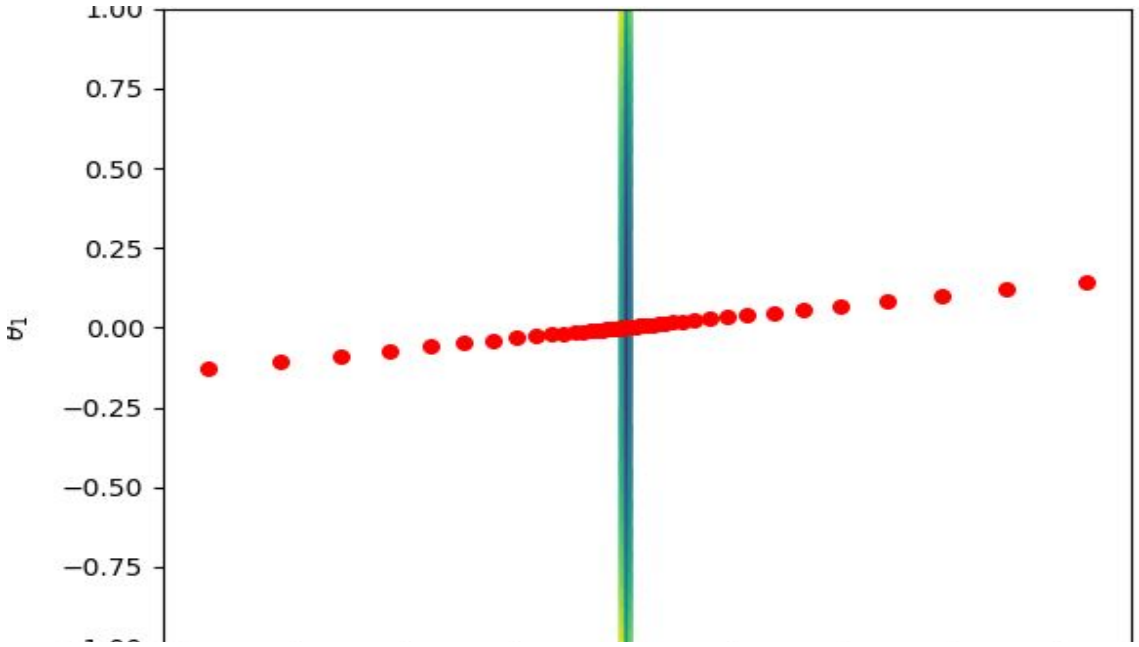
η= 1.3


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
θ_1

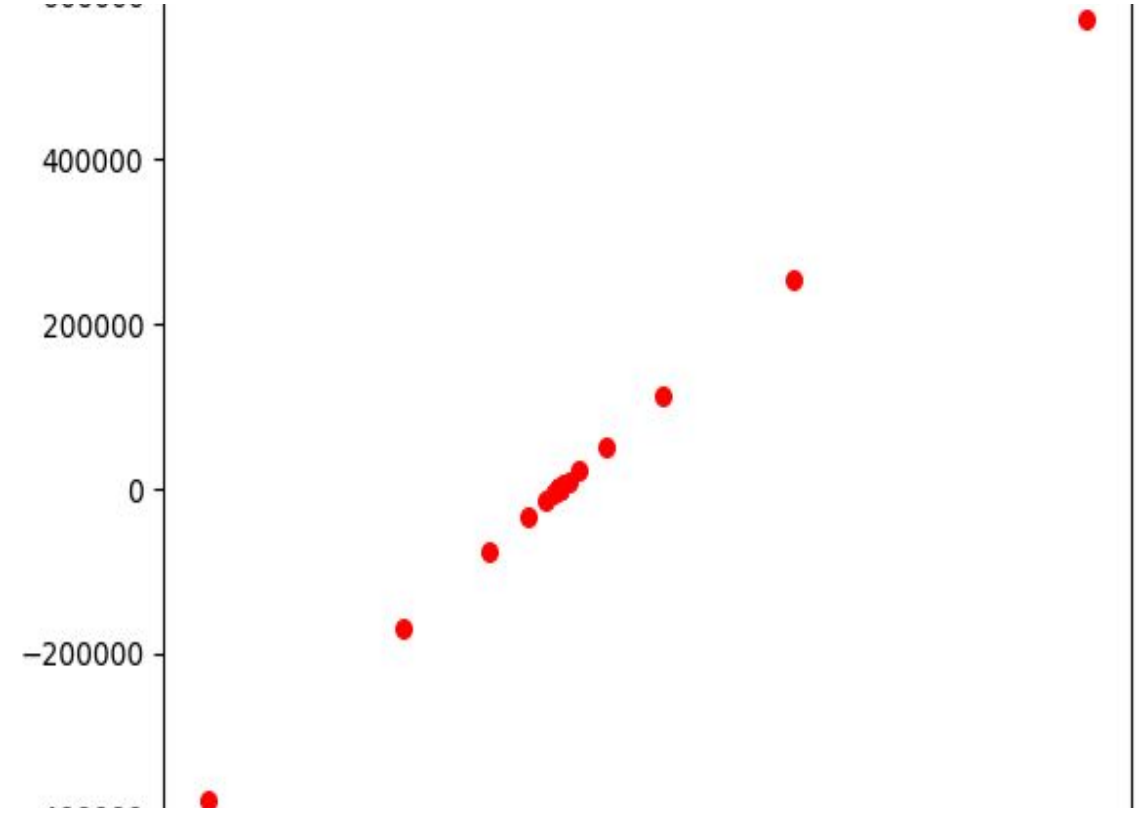
η= 1.7


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
θ_1

η= 2.1



η= 2.5

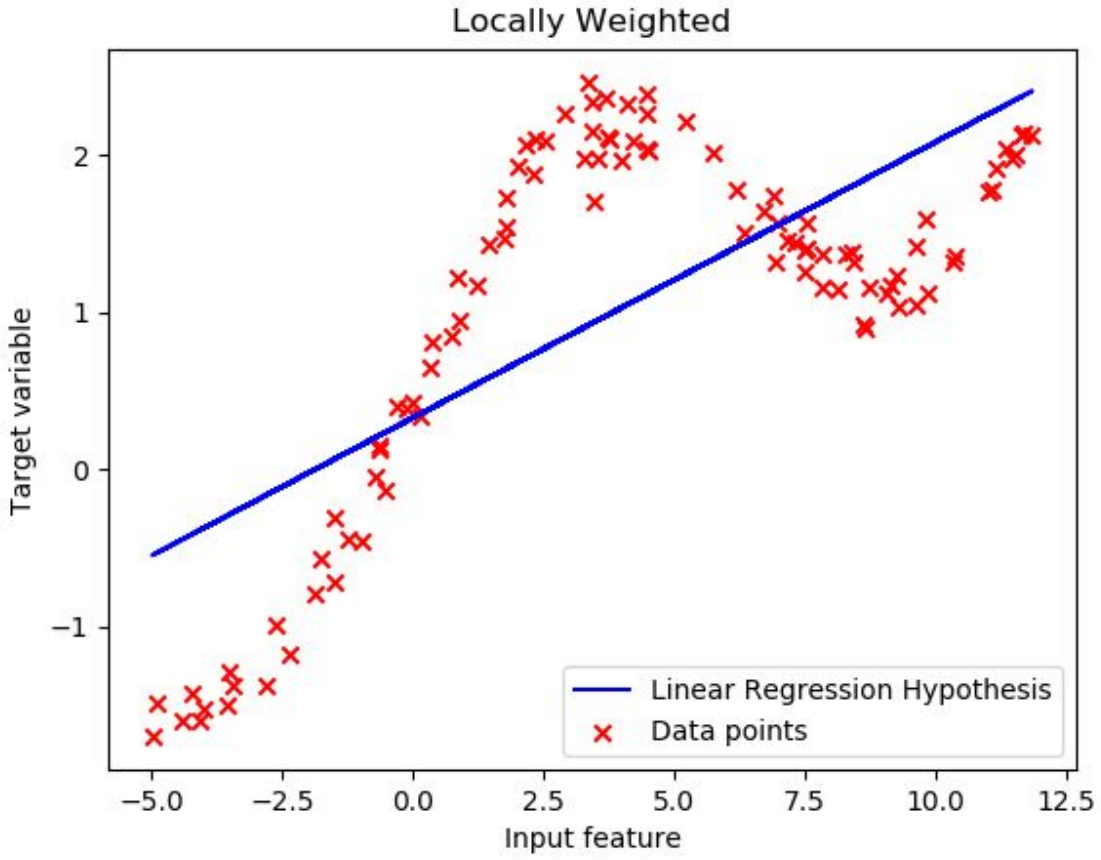


Observations:

When η is very small like 0.03 to 0.1 , then gradient descent takes very small steps in every iteration but eventually it does converge to the local minima after some large number of iterations.
As η increases to 0.9 or 1.3 the iterations taken to converge gradually decrease(since a bigger step is taken in each iteration) and thus η in this range gives us the local minima at a much faster pace than η in range 0.01 to 0.1 or so.
But as η is further increased to 2.1 or 2.5, then the algorithm takes even bigger steps to reach minima and as a result it starts oscillating and the contours get hidden. The function never converges to the minima, instead it keeps oscillating around the minima even after a large large number of iterations.

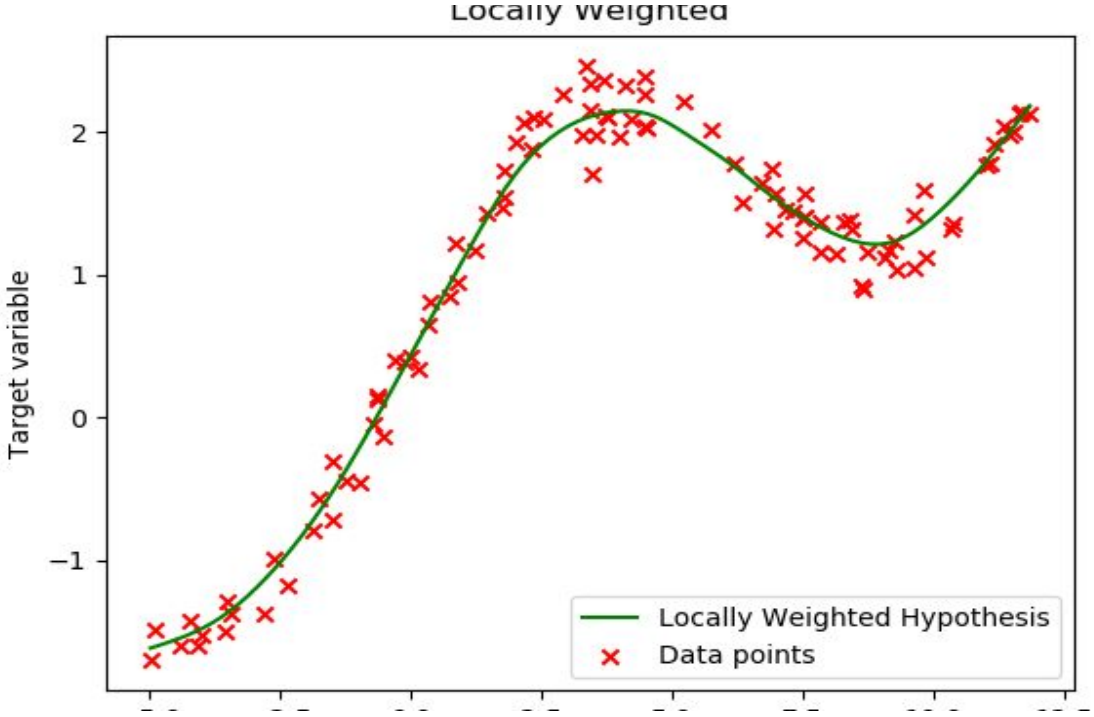
Q2. Locally Weighted Linear Regression

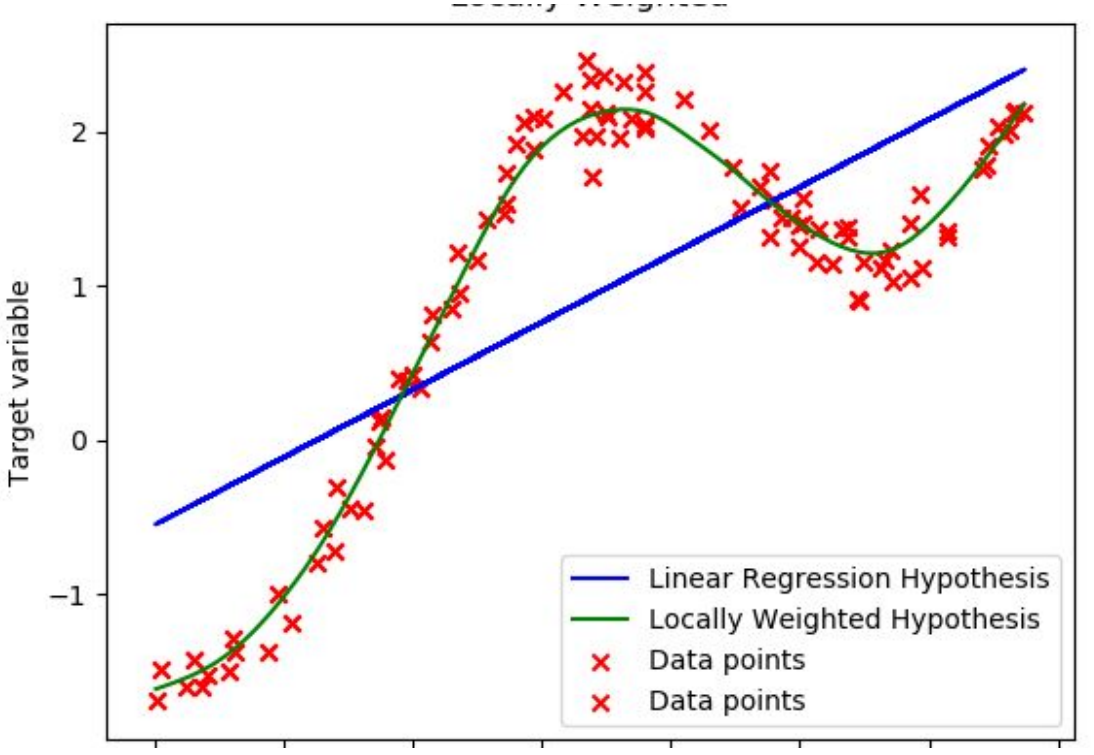
(a) Linear regression without considering weights. It is plotted using the normal equations.
(I've not done any normalisation in this question.)



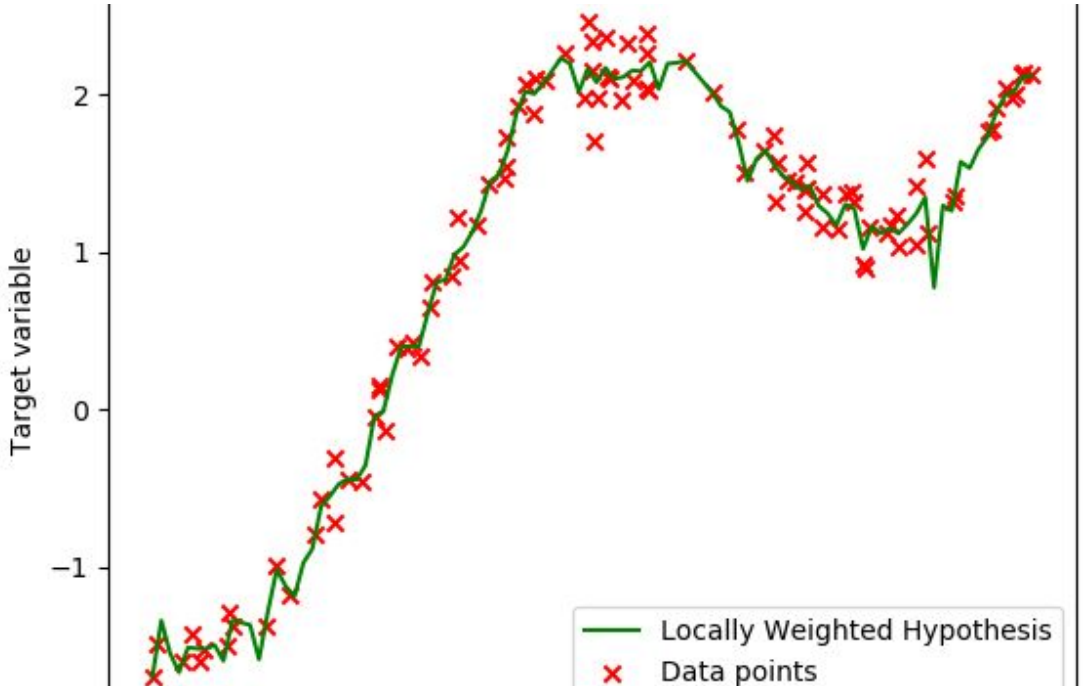
Since the training data set accuracy is very poor, as it is evident from the graph, this isn't quite the right fit to the data since our data can no longer be approximated by a linear function. Therefore, we need to do something bigger and better than linear regression.

(b) Implementing locally weighted linear regression using normal equations with τ(tau) = 0.8

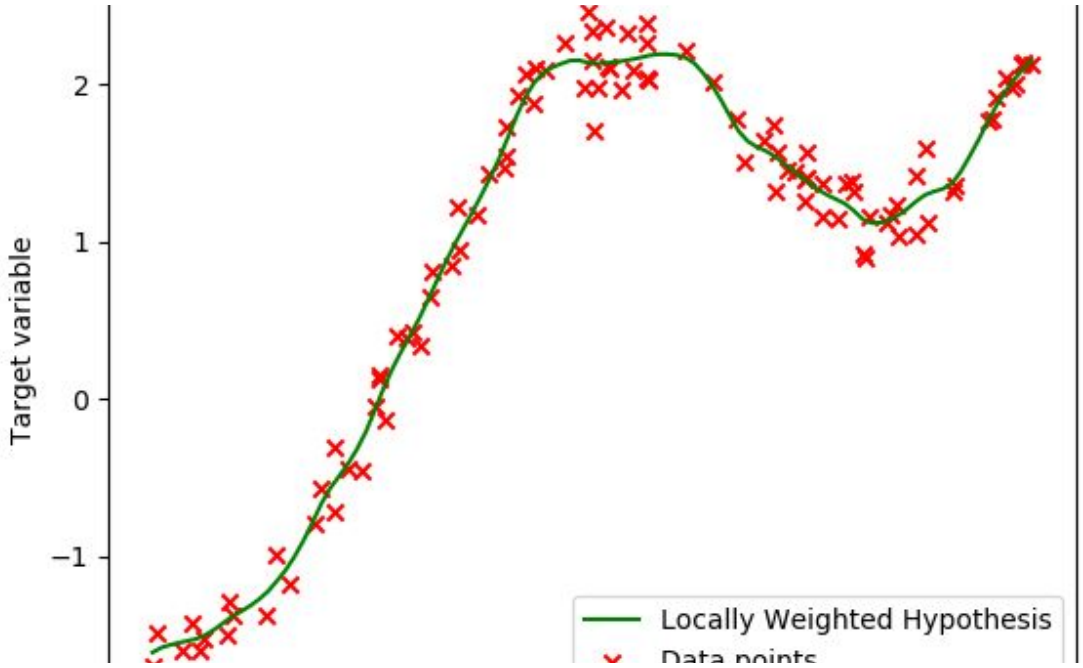




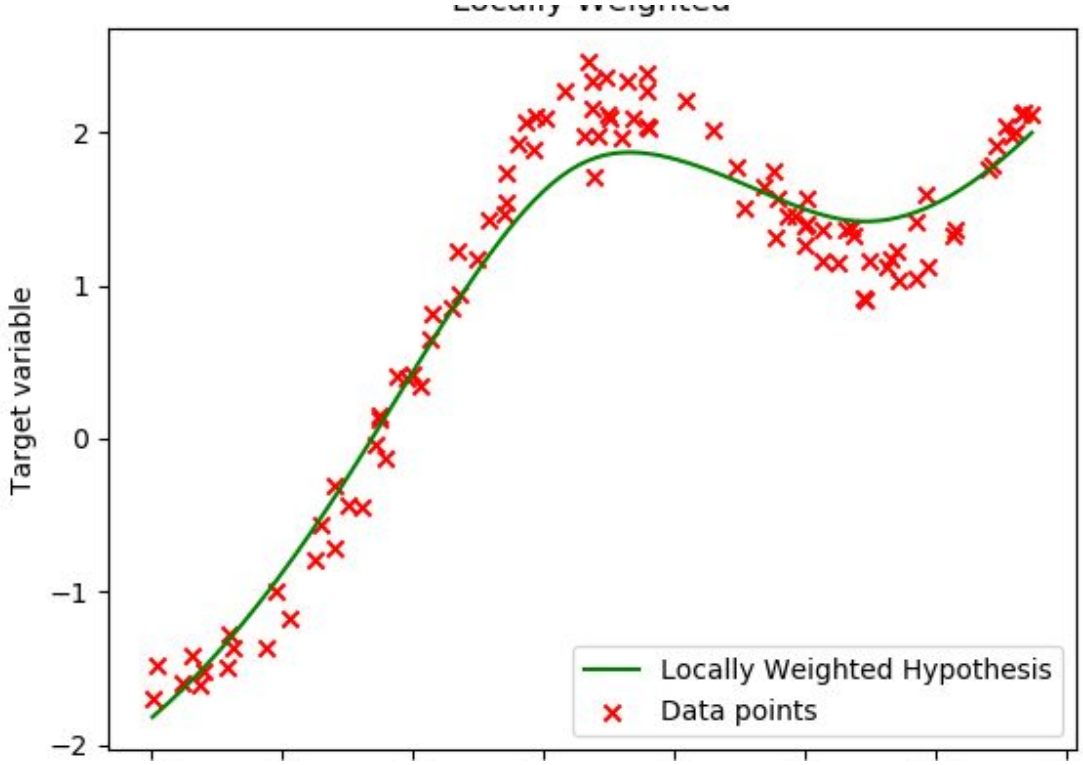
(c) τ(tau) = 0.1


Target variable
2
1
0
−1
Locally Weighted Hypothesis
Data points

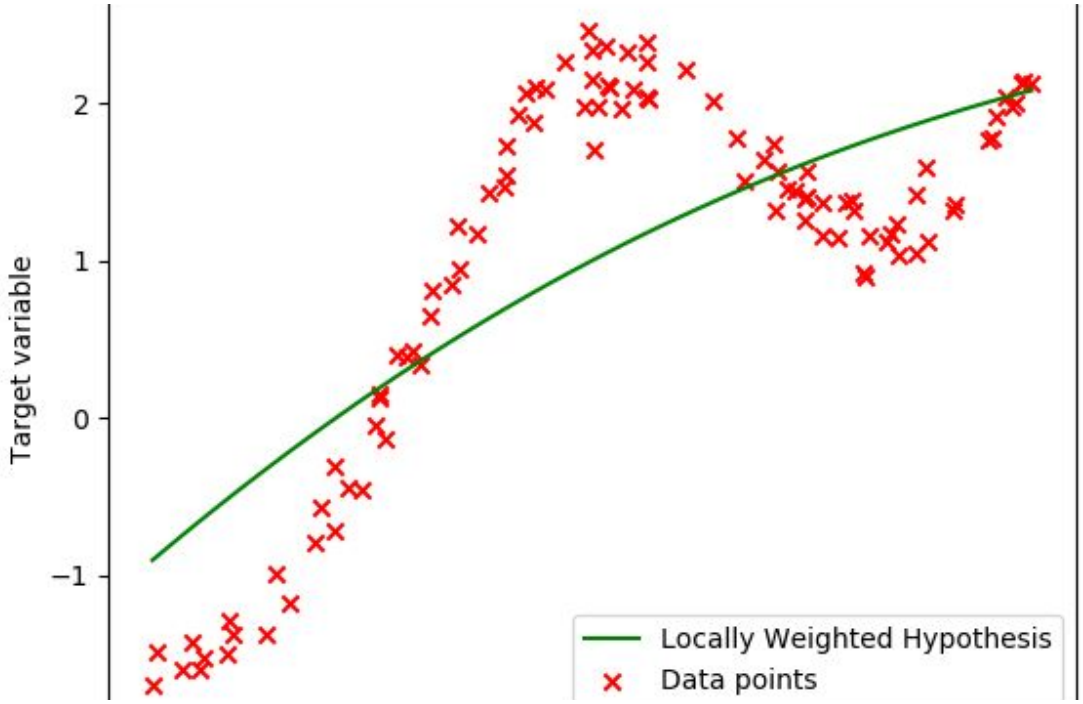
τ = 0.3


Target variable
2
1
0
−1
Locally Weighted Hypothesis
Data points

τ = 2


Target variable
2
1
0
−1
−2
Locally Weighted Hypothesis
Data points

τ = 10


Target variable
2
1
0
−1
Locally Weighted Hypothesis
Data points

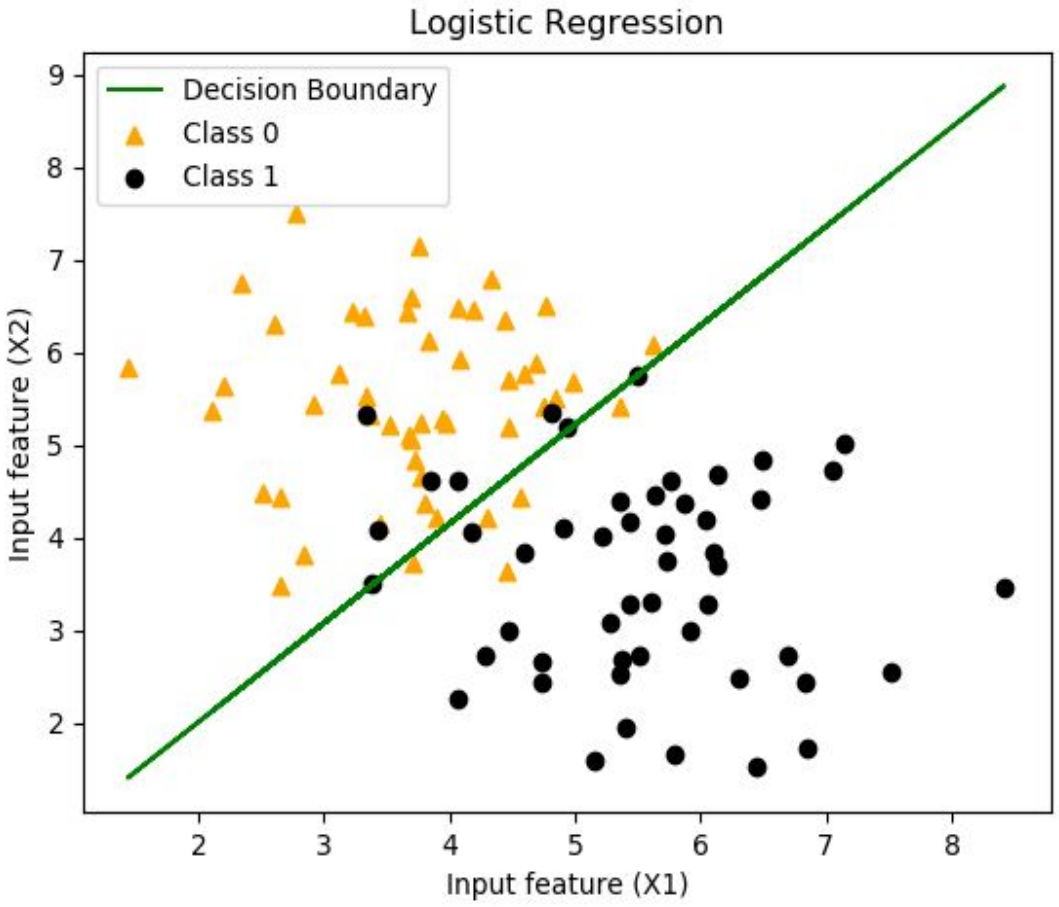
Observations:

I think the best fit is obtained when τ = 0.8 on the un-normalised data.
When τ is very small like 0.1 or 0.3, then the algorithm tries to fit every point and thus results in overfitting. As τ approaches 0, while doing locally weighted regression at point x, all its neighbour points get weight close to 0 and thus algorithm tries to fit the point x perfectly. This happens for every x and thus leads to overfitting.
When τ is too large like 2 or 10, algorithm doesn't perform well and since the training set accuracy is poor too, it results in underfitting. As τ approaches infinity, all neighbour points get weight close to 1 and then it simply reduces to linear regression without any weights.

Q3. Logistic Regression

(a) Coefficients theta resulting from Newton's method are:
θ[0,1,2] = [5.38187837e+09, -4.28967301e+10, 4.00607286e+10]
Number of iterations to converge = 5 (…..No normalisation done)

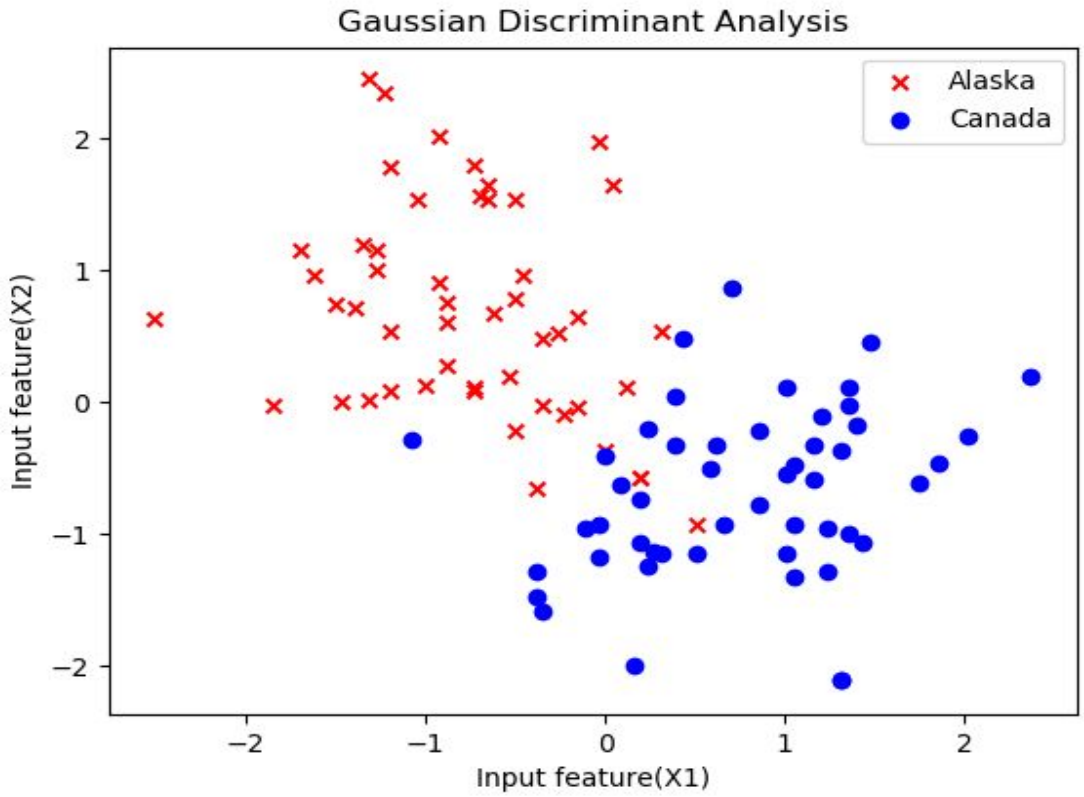
(b) Decision boundary:



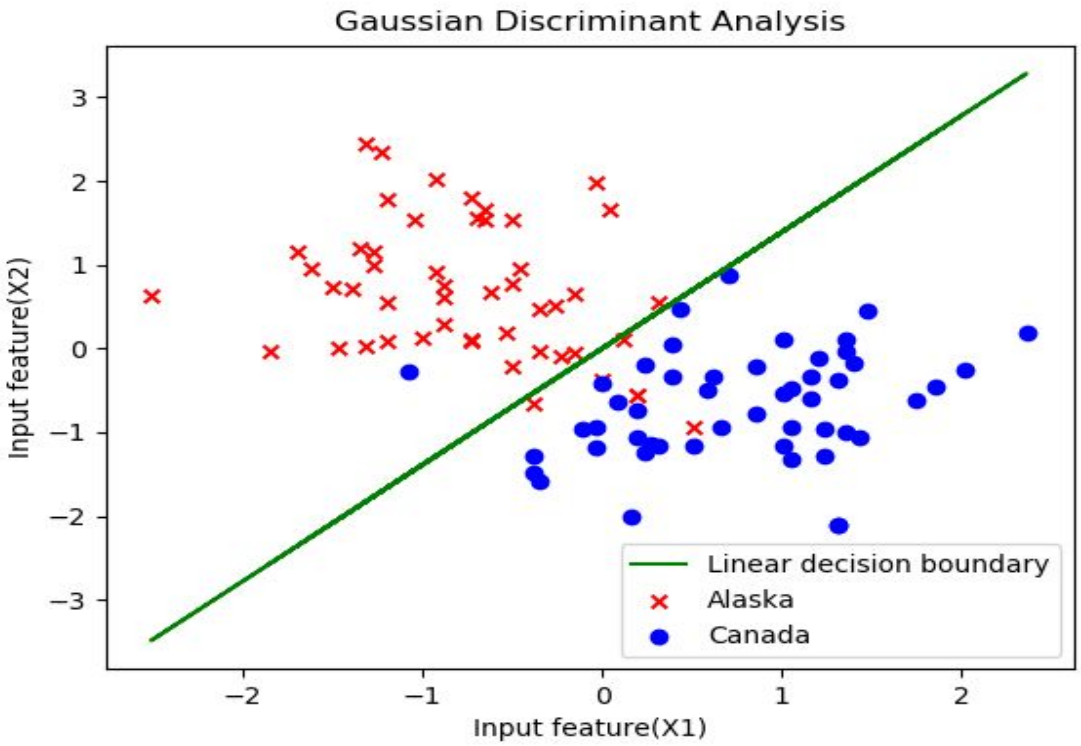
Q4. Gaussian Discriminant Analysis

(a) μ_0 = [[-0.75529433]
[0.68509431]]
μ_1 = [[0.75529433]
[-0.68509431]]
$\sum$ = [[0.42953048 -0.02247228]
[-0.02247228 0.53064579]] (...I've used normalised data)

(b) Training data plot:



(c) Linear boundary**:**



Equation of line: AX = B

Where A = 2 * ($\mu_o^T\Sigma^{-1}$ - $\mu_1^T\Sigma^{-1}$) and

B = ($\mu_o^T\Sigma_0^{-1}\mu_o$ - $\mu_1^T\Sigma_1^{-1}\mu_1$ - 2 * log((1/Φ) -1)

I used this equation of line and for every input feature X1, I calculated X2 using this equation. It resulted in a line which is as shown above.

(d) μ_o = [[-0.75529433]
[0.68509431]]

$\mu_{1=}$ = [[0.75529433]
[-0.68509431]]

$\sum_0$ = [[0.38158978 -0.15486516]
[-0.15486516 0.64773717]]

$\sum_1$ = [[0.47747117 0.1099206]
[0.1099206 0.41355441]]

(e) Equation of quadratic curve resulting from GDA when $\sum_0 \neq \sum_1$.

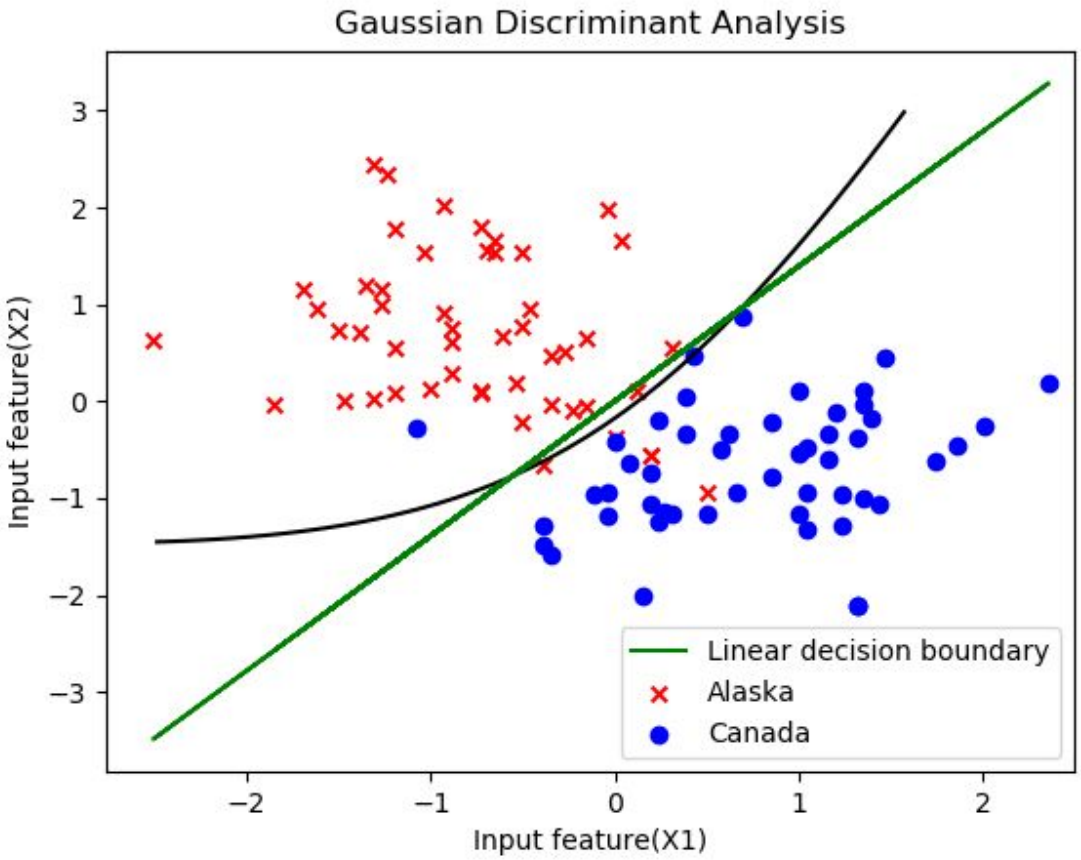
$X^T AX + BX + C = 0$

Where $A = \Sigma_0^{-1} - \Sigma_1^{-1}$

$B = -2 * (\mu_o^T \Sigma_0^{-1} - \mu_1^T \Sigma_1^{-1})$

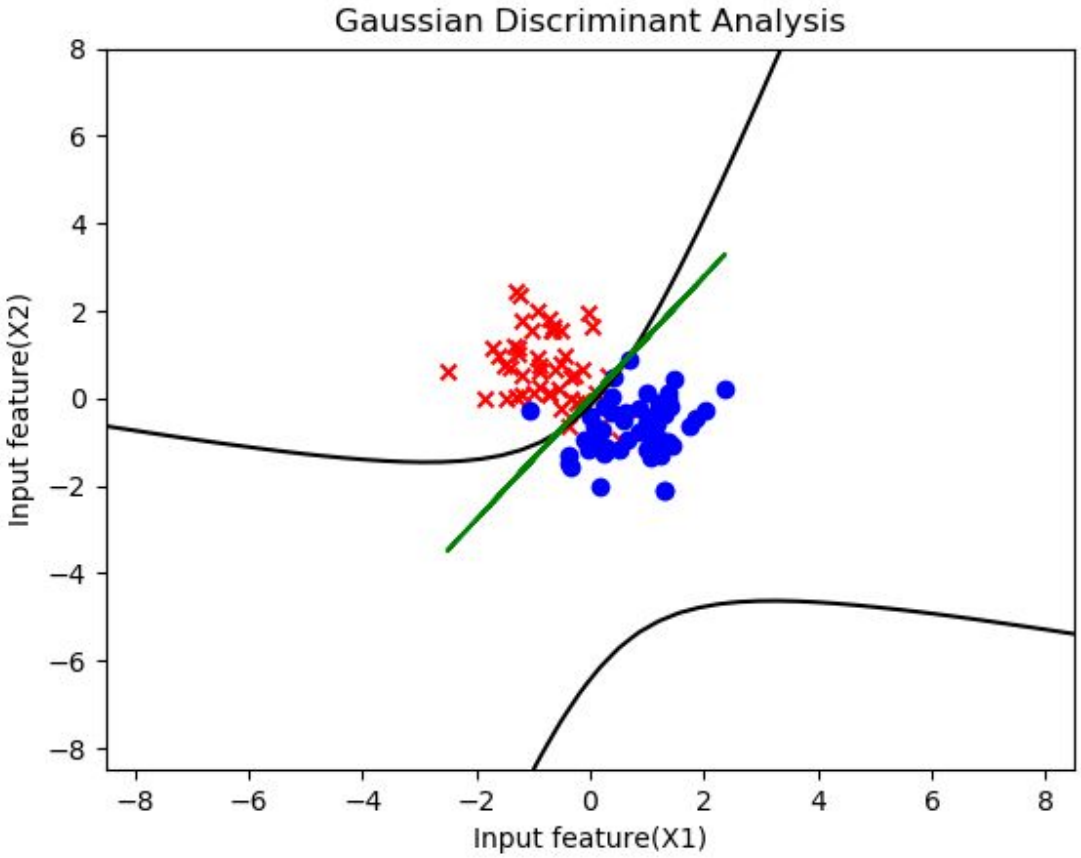
And $C = \mu_o^T \Sigma_0^{-1} \mu_o - \mu_1^T \Sigma_1^{-1} \mu_1 - 2 * \log(((1/\Phi) -1)) - (|\Sigma_0| / |\Sigma_1|))$

These are the coefficients of the quadratic equation.



(f) The quadratic boundary turns out to be a better separator because it classifies few Alaska salmons correctly unlike the linear separator.

The quadratic curve resulted from the quadratic separator is actually a hyperbola.



Hyperbolic curve.