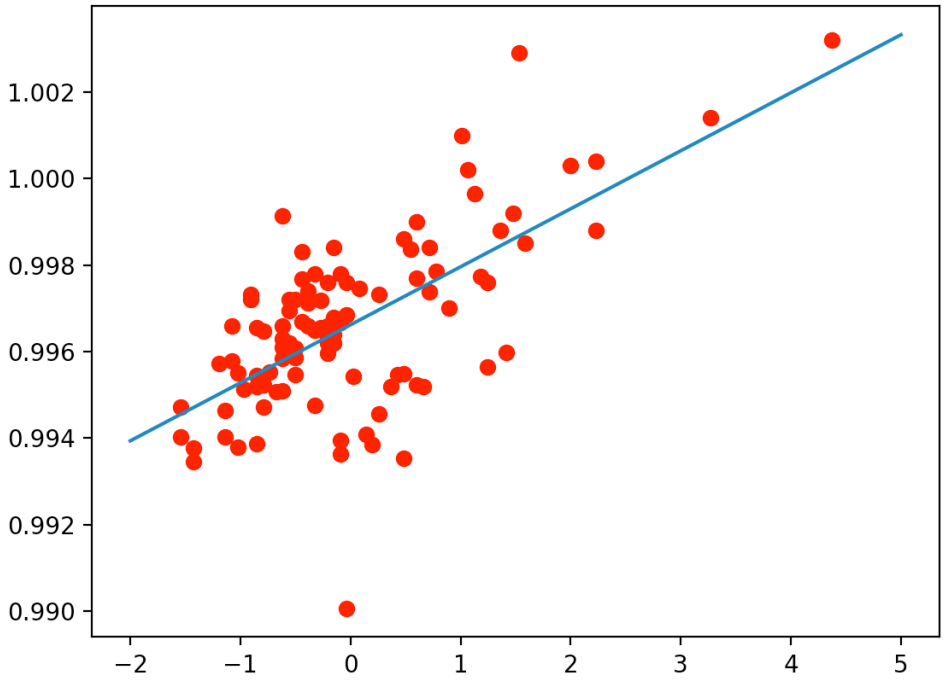
Machine Learning
Assignment 1
Sagar Goyal
2015CS10253

Q1. LINEAR REGRESSION

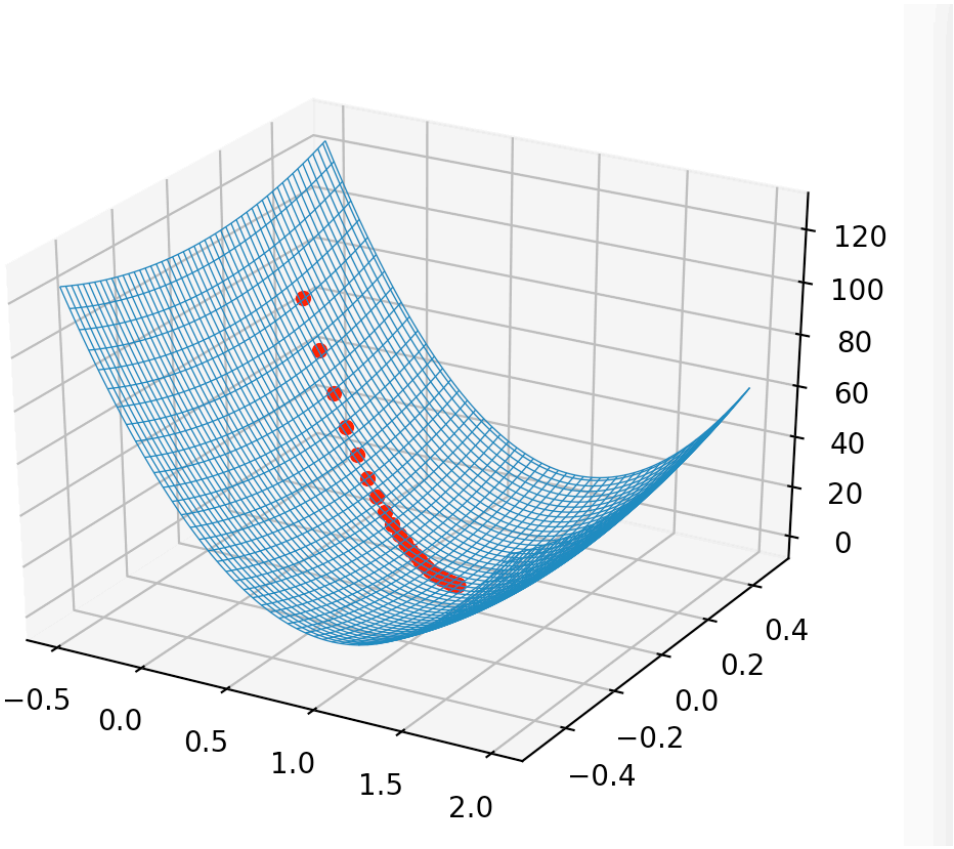
(a) The learning rate used=0.009, the stopping criteria was when the difference in the loss function becomes less than $10^{-9.}$

The final set of parameters obtained is: [`0.99662009` `0.0013402`]
$\emptyset_0$ = `0.99662009`
$\emptyset_1$ = `0.0013402`

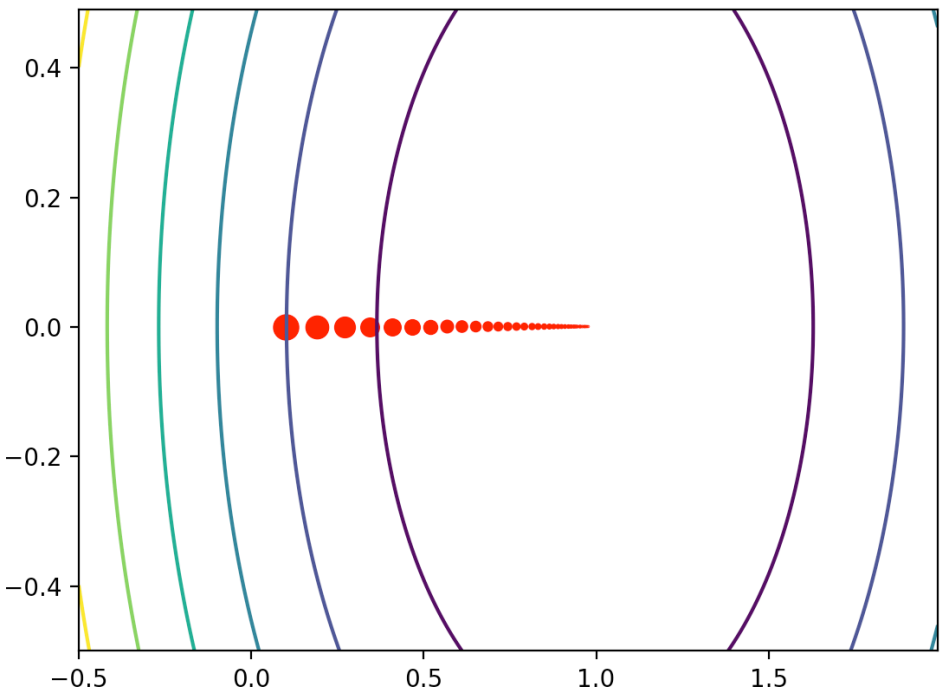
(b)



(c)

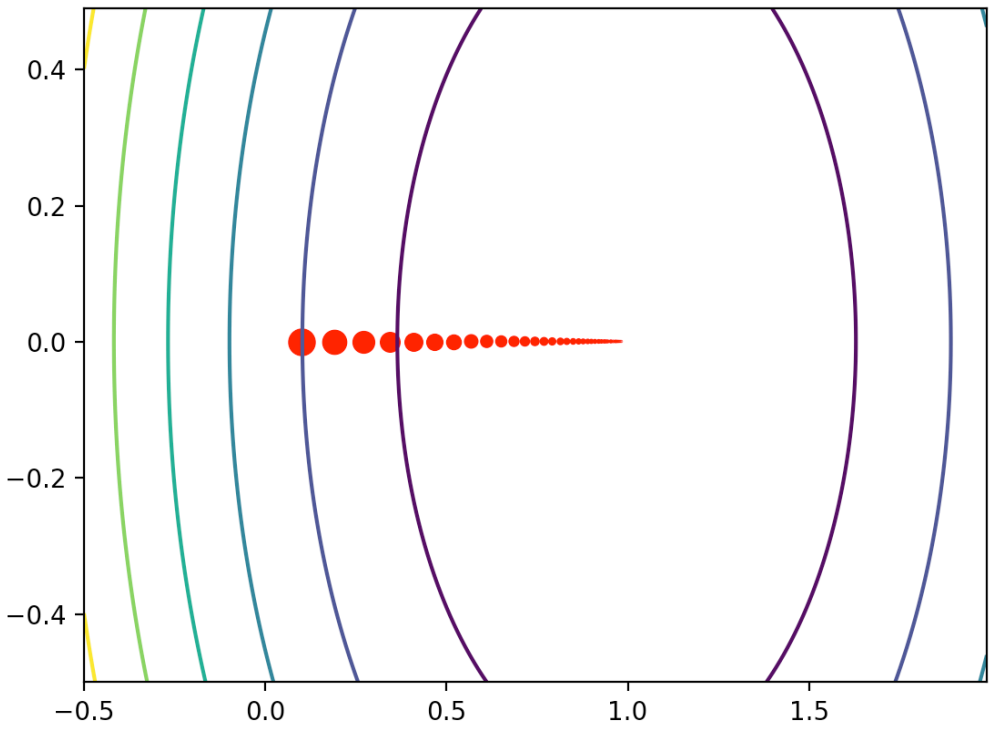

120
100
80
60
40
20
0
−0.5
0.0
0.5
1.0
1.5
2.0
0.4
0.2
0.0
−0.2
−0.4

(d)

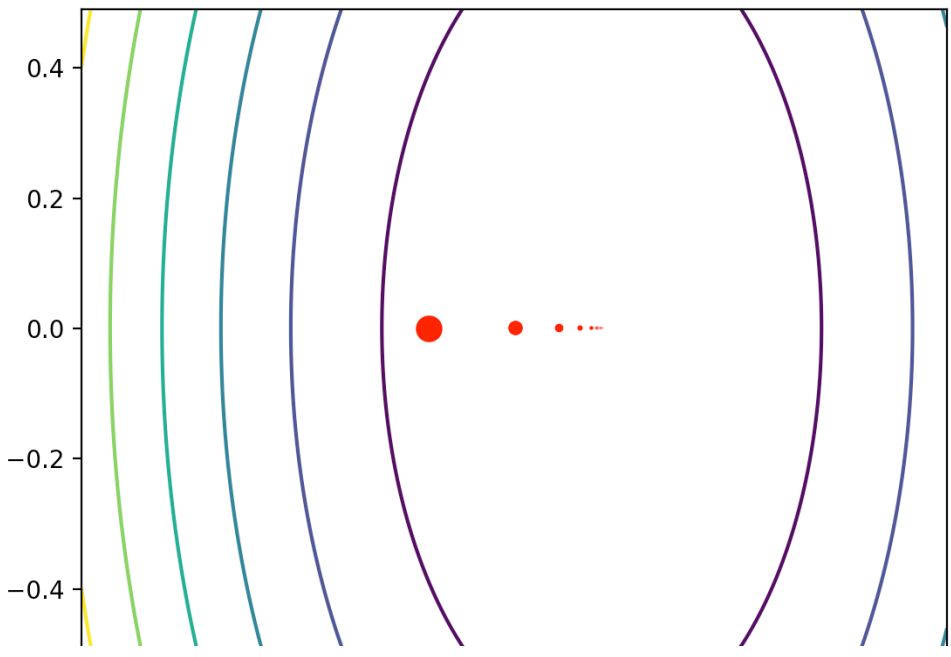

0.4
0.2
0.0
−0.2
−0.4
−0.5
0.0
0.5
1.0
1.5

(e)

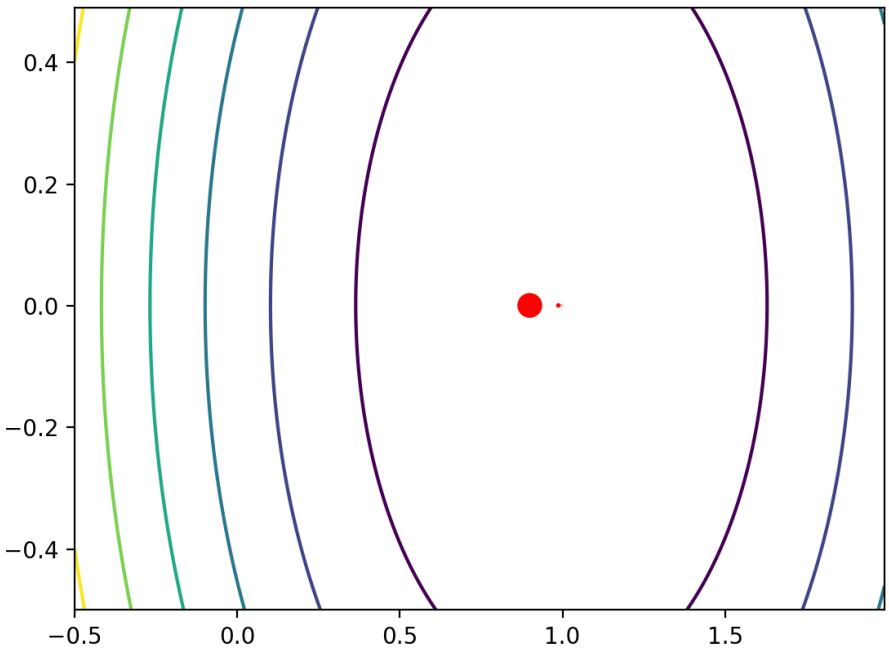
1. **0.001:**



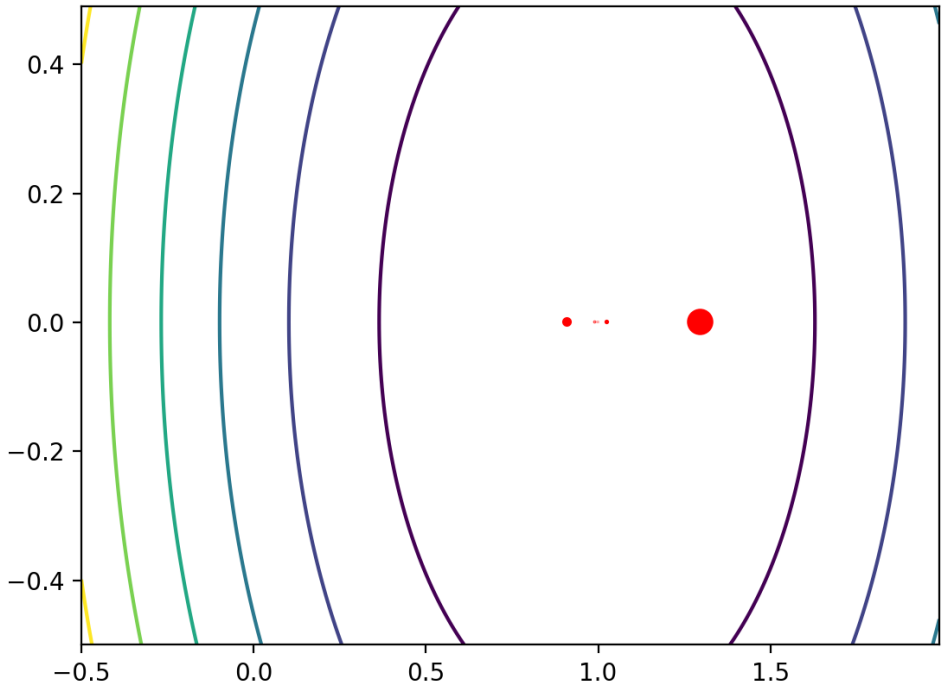
2. **0.005:**



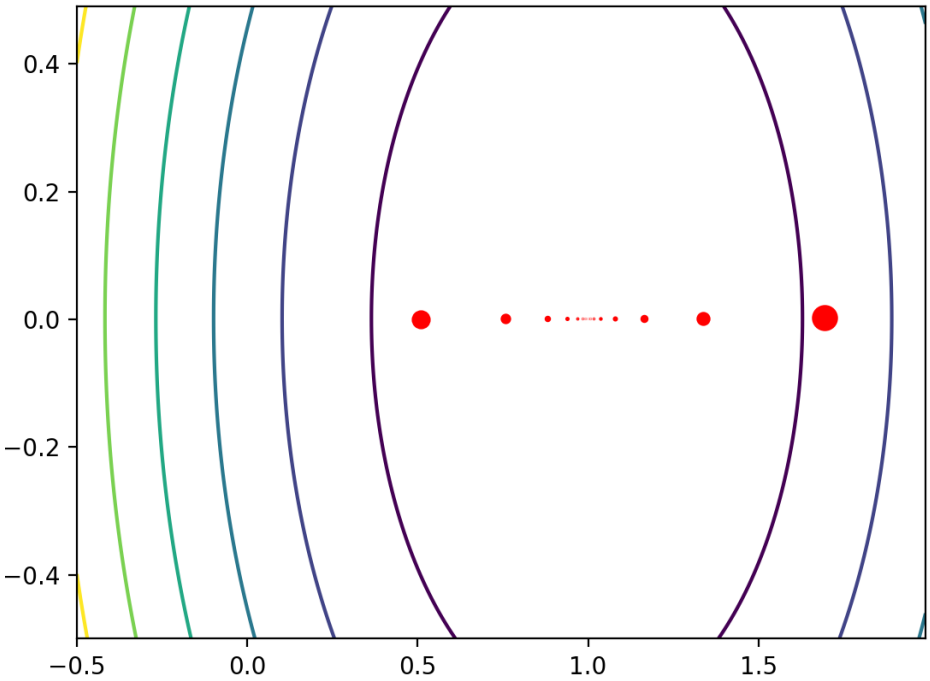
3. **0.009 :**



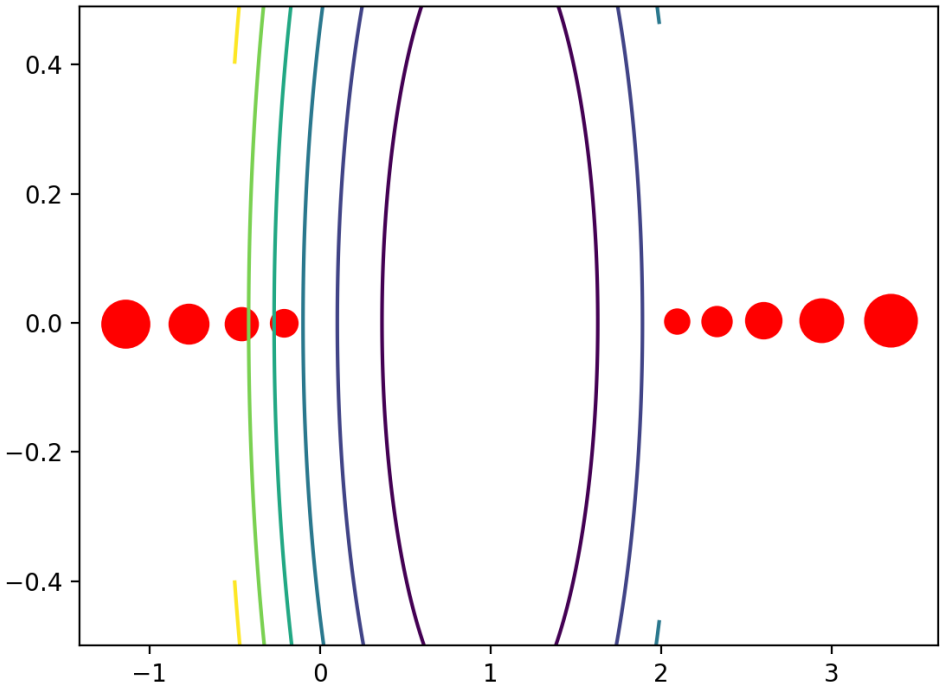
4. **0.013 :**



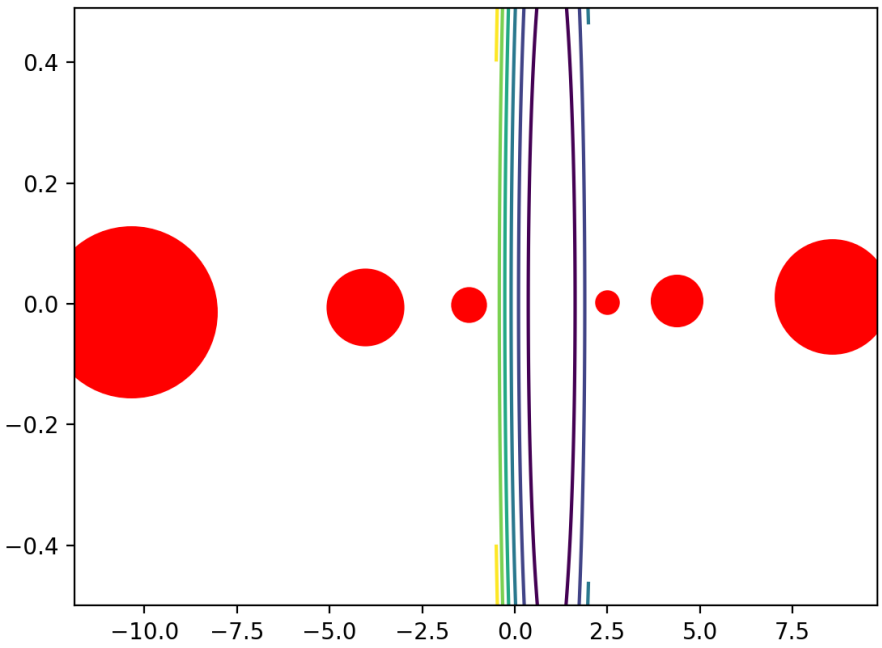
5. **0.017**



6. **0.021:**



7. **0.025 :**



For learning rate less than 0.009 the loss function converges while remaining on one side of the least point only and does not jump to the other side. However we see that the number of iterations are reduced on increasing the learning rate.

Once we increase the learning rate more (0.013), the value of Theta keeps jumping to the other side of the minima but however converges in a few steps to get a answer.
On increasing it more (0.019) the number of iterations before convergence are increased but it still converges. We can see that the point jumps through both the sides of the minima.

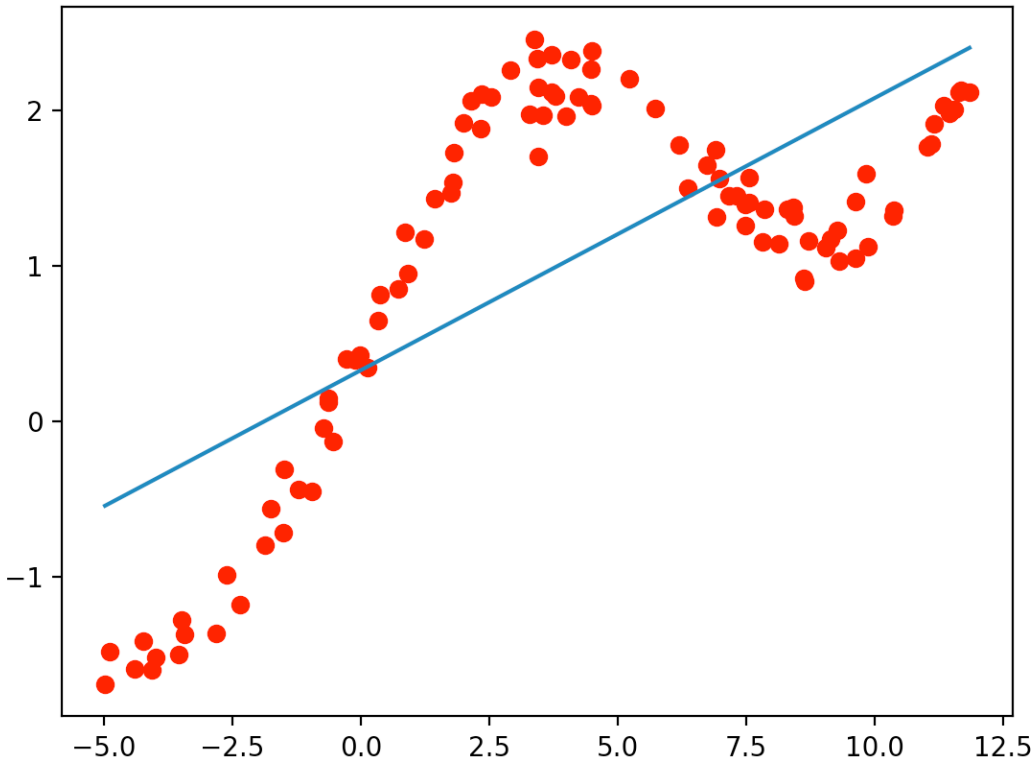
If we increase the learning rate more (0.021 and 0.025), the function becomes a diverging one and Theta goes to infinity therefore we never find Theta reaching the minima.

Thus we can say that **learning rate= 0.009** gives fastest convergence without overshooting.

Q2. WEIGHTED LINEAR REGRESSION

(a) Used the unweighted linear regression formula for Ø in the closed form and found out its value. Then the line obtained from it was plotted along with the data points. This was captured in the **desc_normaleq()** function

```
Ø=[1.04083409e-16 7.15048254e-01]
```

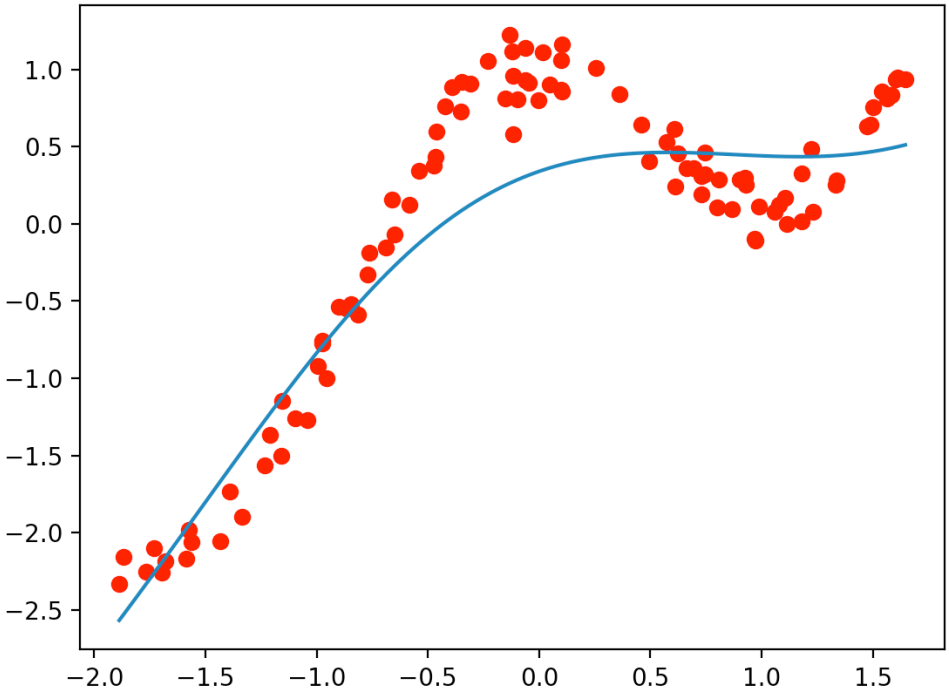


We can clearly see that a straight line is not enough to capture the data efficiently as most of the points are pretty far away from line obtained.

(b) The locally weighted linear regression was implemented by using the distribution of the weights to be Gaussian. We generated an array of uniformly distributed numbers between the maximum and the minimum value of x in the data and evaluated linear regression for each of those points. The **" create_w() "** function evaluates the W matrix which is a diagonal matrix with the (i,i)th entry to be the weight with respect to $x^{(i)}$ evaluated using the formula -

$$w^{(i)} = \exp\left(-\frac{(x - x^{(i)})^2}{2\tau^2}\right)$$

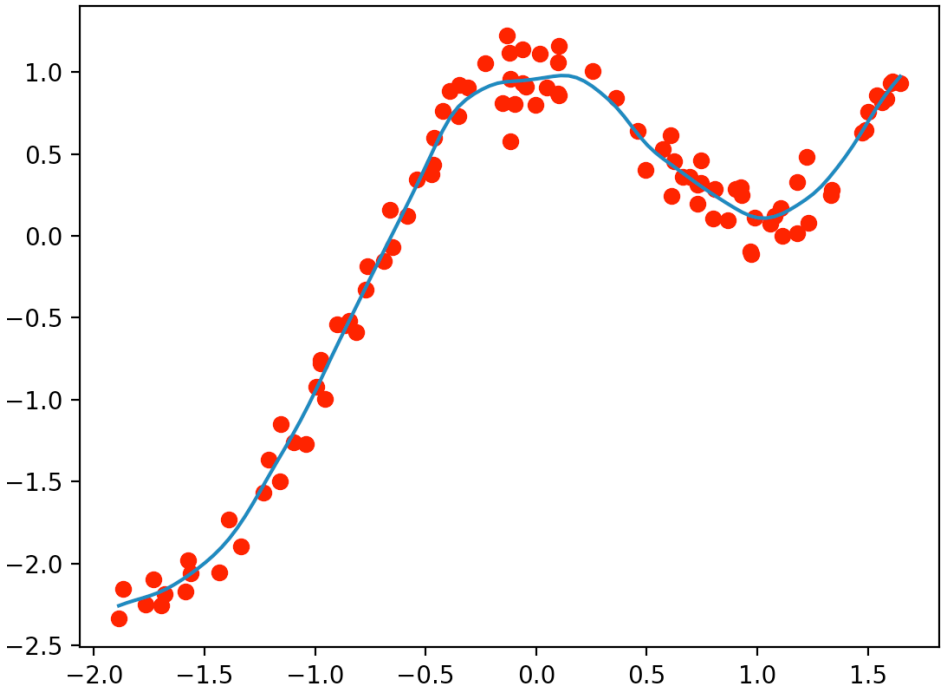
tau (bandwidth parameter) was kept as ‘0.8’ initially to obtain better results. Now weighted linear regression was performed for each point and the Ø obtained for each point was used to evaluate the value of corresponding y for each x. These values were then plotted to obtain the curve:



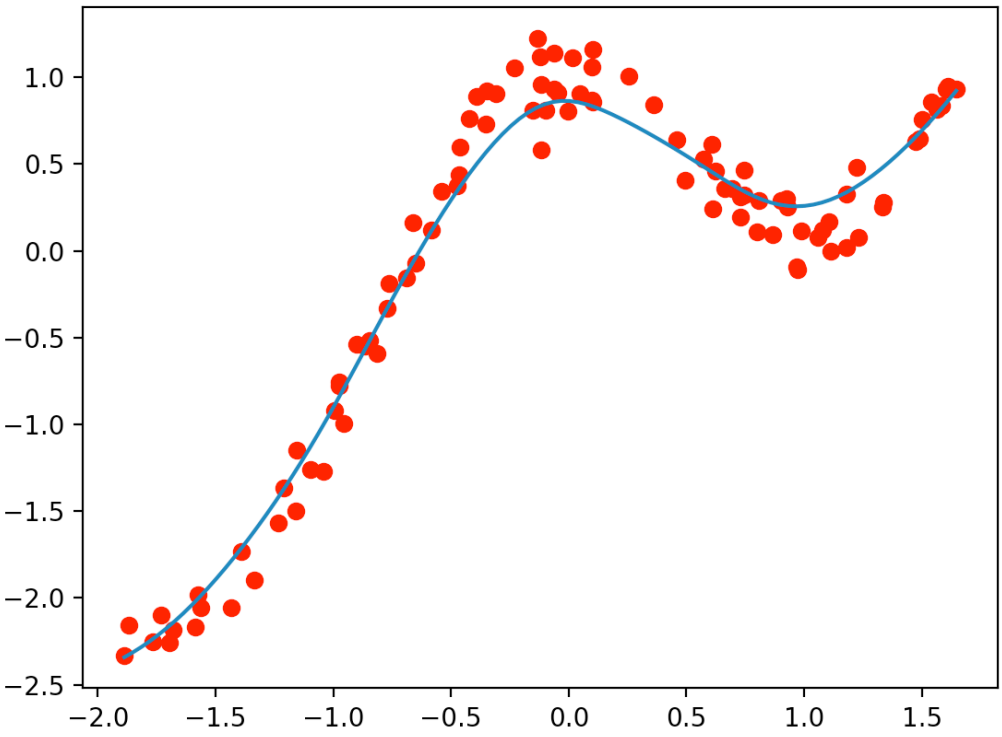
We can see that we get a better match to the training data than simple linear regression and thus weighted Linear Regression proves useful to get a match of how this data is distributed.

(c) The following curves were obtained:

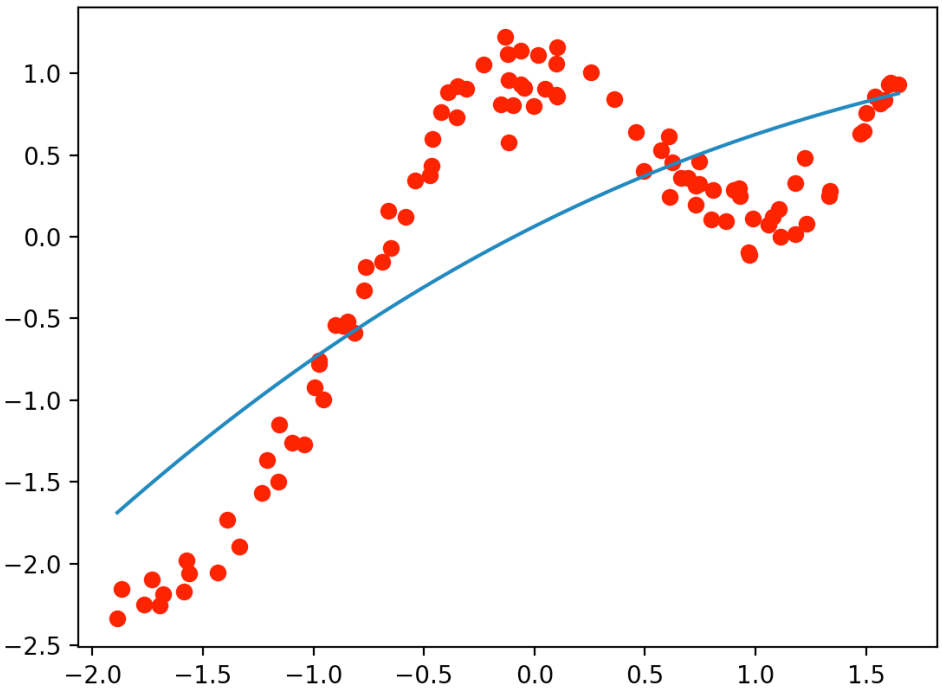
1. Tau=0.1



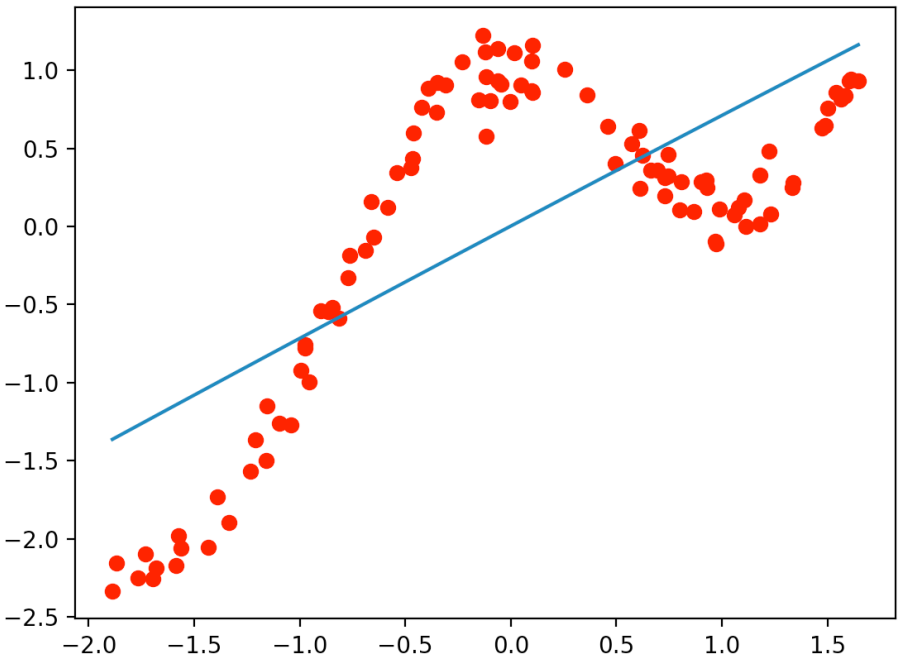
2. Tau = 0.3



3. Tau= 2



4. Tau=10



We can see that the best of the fit is obtained in the cases when Tau=0.3 or 0.1. When Tau is larger, then most of the points are not properly covered and their distances from the line is very high and we start getting straight lines are boundaries which are like the non-weighted linear regression. This can be understood as when we increase the tau very much, each of the weights become very close to each other and the features of their distribution are not observed.

Q3. LOGISTIC REGRESSION

(a) Logistic regression was implemented using the Newton's method. The "calc_log_hx()" function calculates the $h_{ø}(x)$ and it is then used to calculate the grad_J and the hessian matrix. Then the Newtons formula:

$$Ø^{j+1}=Ø^{j} - H^{-1}(grad\_J)$$

was used to calculate the final value of Ø.

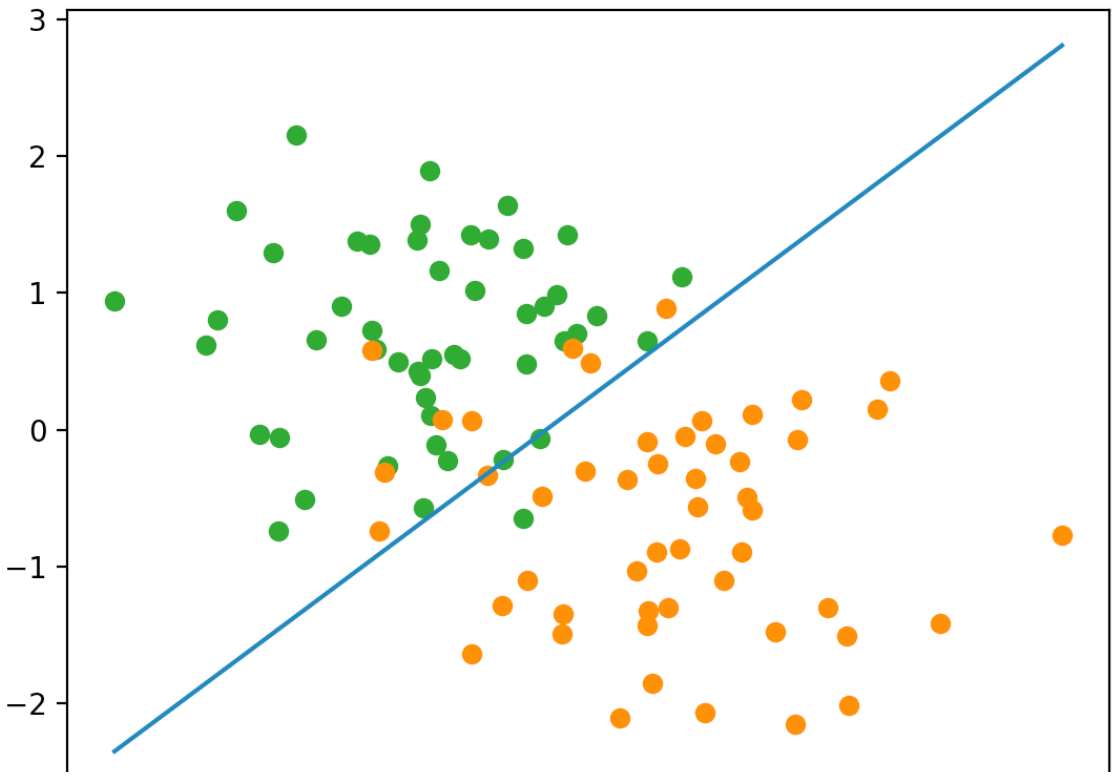
The value of Ø obtained =

$Ø_0 = -1.38205682e-18$

$Ø_1 = 8.07607267e-03$

$Ø_2 = -8.28101055e-03$

(b) The two colours denote the two classes of points. The line drawn is the separator of the data into two classes. The colour green represents: 0 and the colour yellow represents: 1

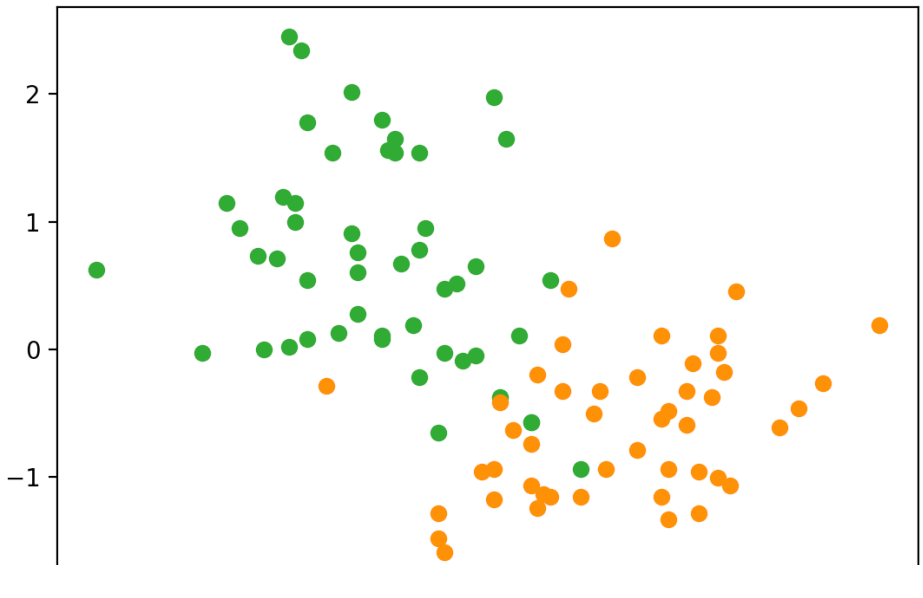


Q4. Gaussian Discriminant Analysis

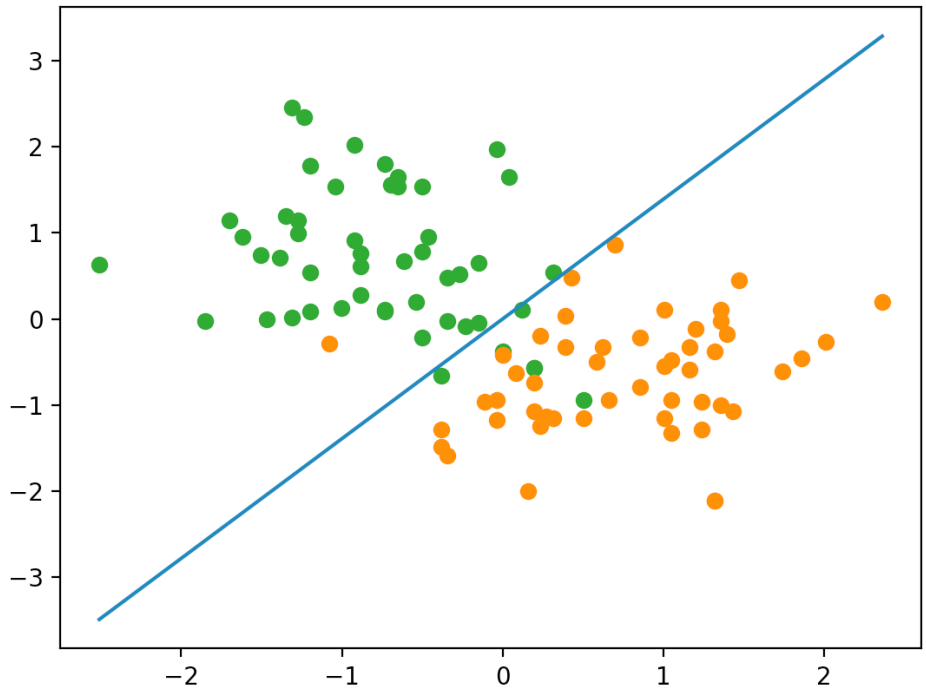
(a) The values of mean0, mean1 and covariance matrix when assumed the same covariance matrix are as follows:

```
mean0= [-0.75529433  0.68509431]

mean1= [ 0.75529433 -0.68509431]

covariance matrix =  [[ 0.42953048 -0.02247228]
                      [-0.02247228  0.53064579]]
```

(b) The X-axis is the feature X0 and the feature X1 is on the Y-axis. The green colour represents: Alaska and Yellow colour represents Canada.



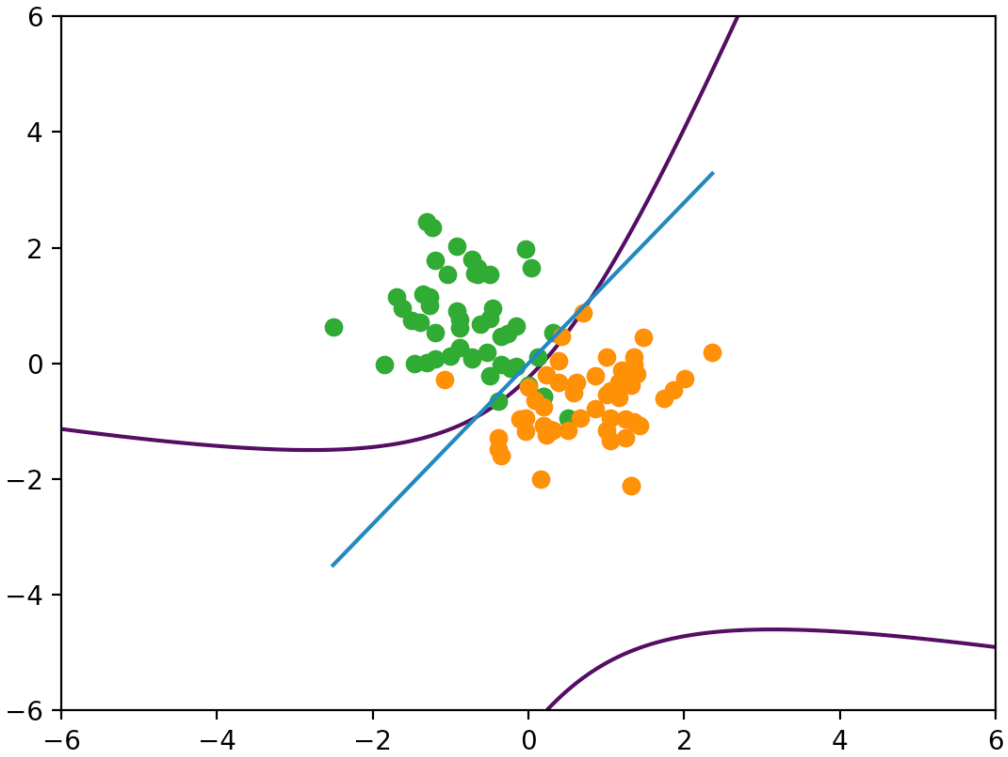
(c)



(d) When we assumed the covariances to be different, on normalized data, we get a quadratic boundary with the following characteristics. The function **"gda_theta()"** returns all the parameters in Theta and are used accordingly.

```
Mean0 = [-0.75529433  0.68509431]

Mean1 = [ 0.75529433 -0.68509431]


Covariance0 =   [[ 0.38158978 -0.15486516]
                 [-0.15486516  0.64773717]]

Covariance1 =   [[0.47747117 0.1099206 ]
                 [0.1099206  0.41355441]]
```

(e) This is the quadratic boundary that is obtained: It turns out to be a hyperbola where the data is separated by only one arm of the hyperbola.



(f) The quadratic boundary gives a better description of the data that we have as we can see that some point which are wrongly classified by the straight line (2 green dots that intersect the straight line) are classified clearly i.e. without intersection by the quadratic boundary.