COL774 Assignment 1

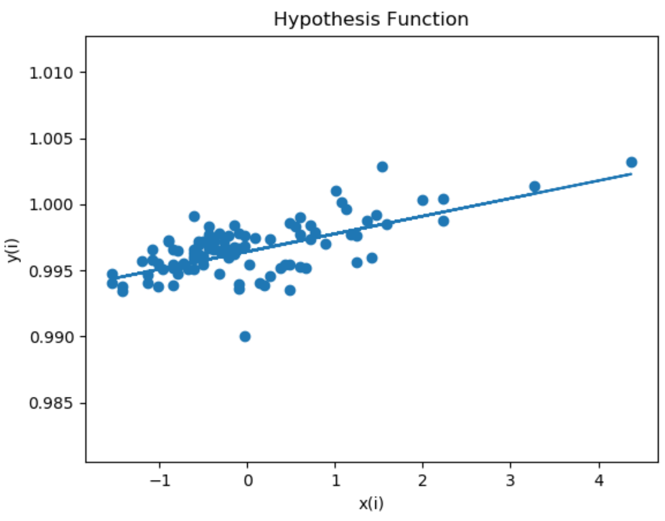
Dipen Kumar (2018CS50098)

November 13, 2020

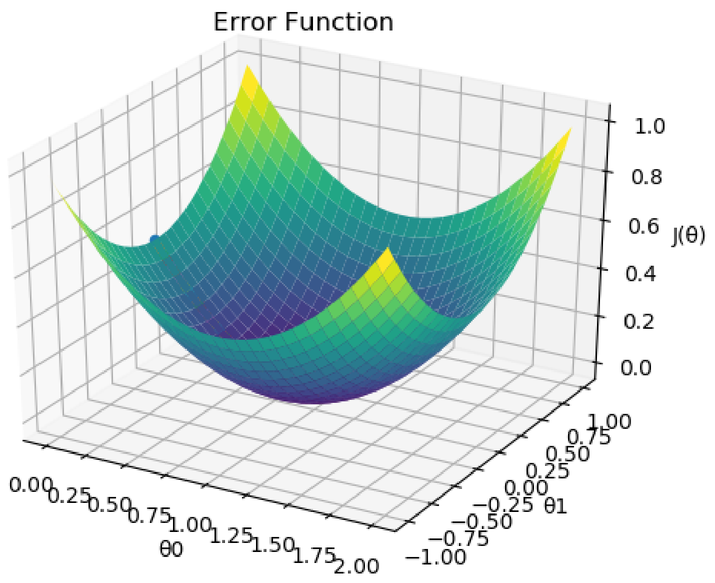
1. Linear Regression

1. My learning rate is 0.025. I stop when difference in cost function goes less than 0.000000001. The final set of parameters obtained by my algorithm is (0.99642863,0.00133994)

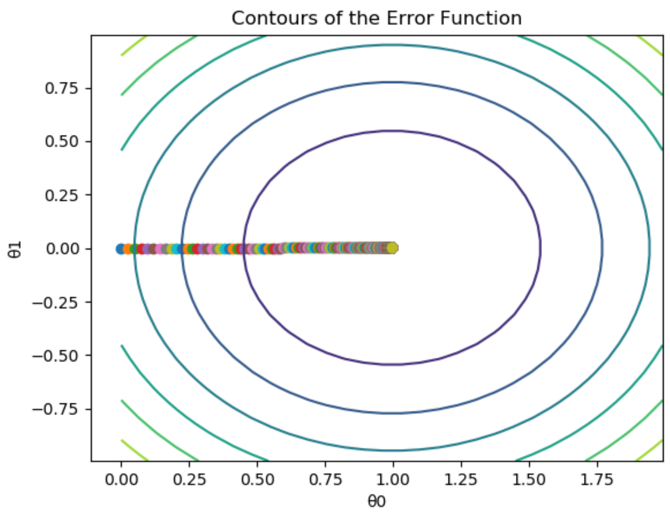
2. Data and the hypothesis function learned by my algorithm in the previous part is shown below



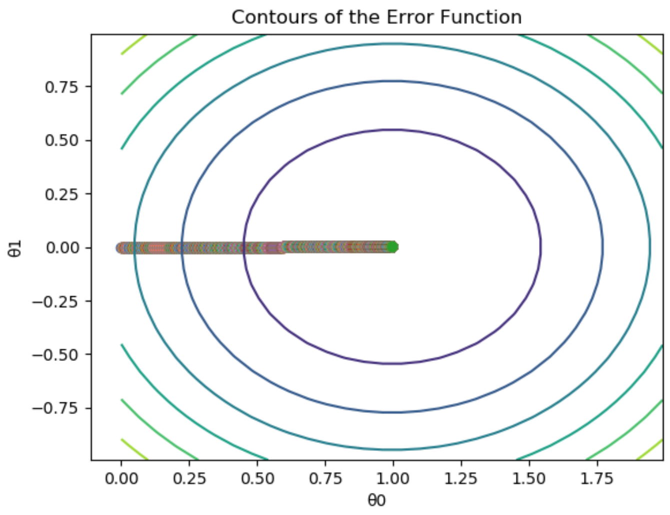
3. 3-dimensional mesh showing the error function (J()) on z-axis and the parameters in the x y plane is shown below

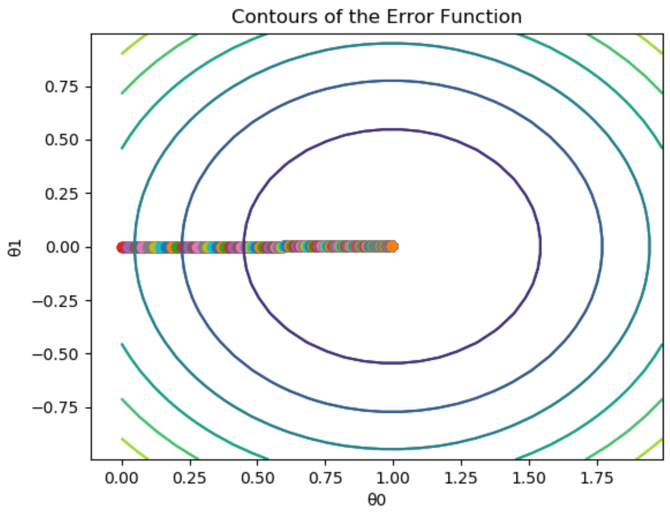


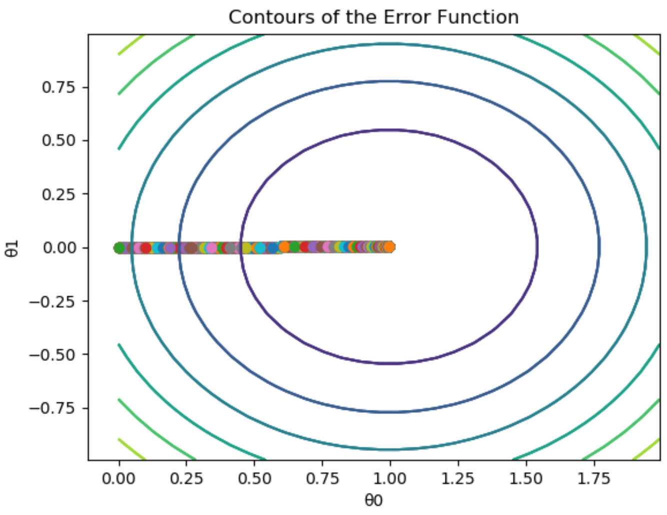
4. The contours of the error function at each iteration of the gradient descent is shown below.



5. The contours at each learning iteration for the step size values of $= 0.001, 0.025, 0.1$ is shown below. Smaller the learning rate more is the iteration to reach the optimal parameters because change is small(learning rate is small). More the iteration more is the time to run the program. Hence a good value of learning rate is necessary. If it is very large then instead of converging to the optimal value of cost function it may diverge.


Contours of the Error Function
θ1
θ0
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75


Contours of the Error Function
θ1
θ0
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75



2. Sampling and Stochastic Gradient Descent

1. 1 million data points as specified is stored in output directory as sampleX.csv and sampleY.csv

2. The learned each time separately for values of batch size r = 1, 100, 10000, 1000000 and convergence criteria in each case is mentioned below in tabular format.

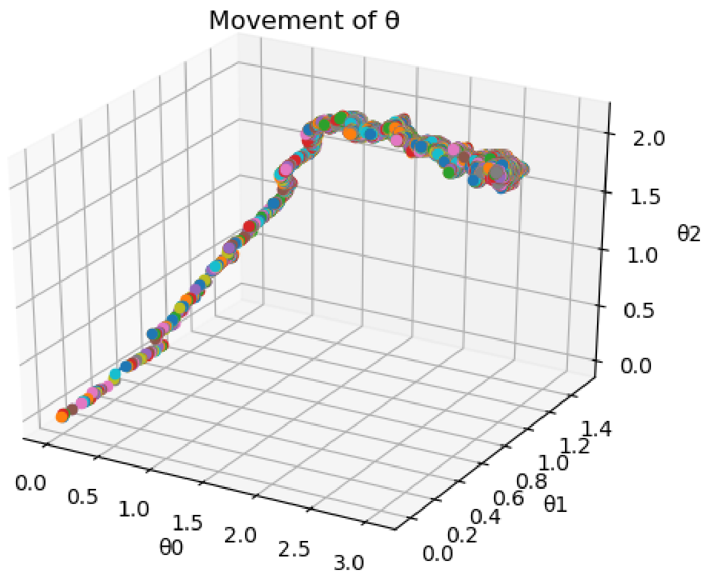
	$r = 1$	$r = 100$	$r = 10000$	$r = 1000000$
parameters	[2.9754665 1.00032475 1.93653744]	[2.90603021 1.01982179 1.99228299]	[2.99993399 1.00019731 2.00045825]	[2.98854397 1.00197054 1.99823539]
$\lvert J(\theta^{t+1}) - J(\theta^{t})\rvert$	< 15	< 1	< 0.1	< 0.000001
for k iterations	10000	10000	100000	1
learning rate	0.001	0.001	0.001	0.1

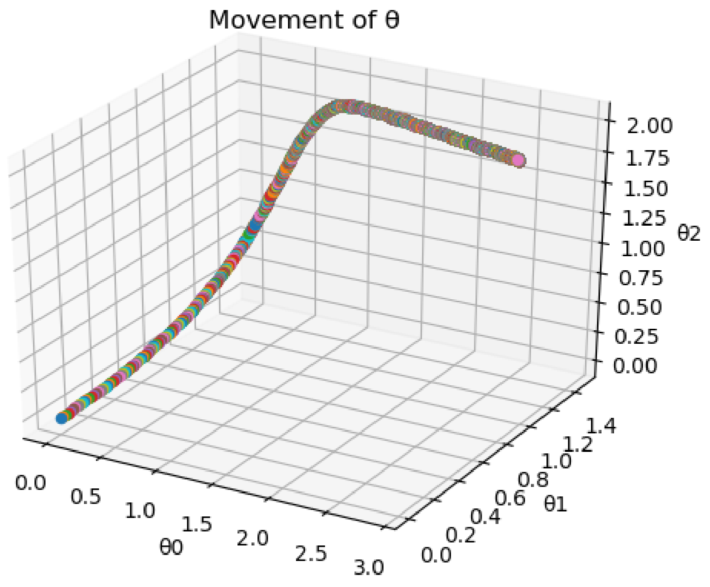
3. yes, different algorithms in the part above (for varying values of r) almost converge to the same parameter values and that is equal to the parameters of the original hypothesis from which the data was generated. Time for convergence, number of epochs, and error cost on a new test data of 10,000 samples provided in the file named q2test.csv in each case is mentioned below in tabular format.

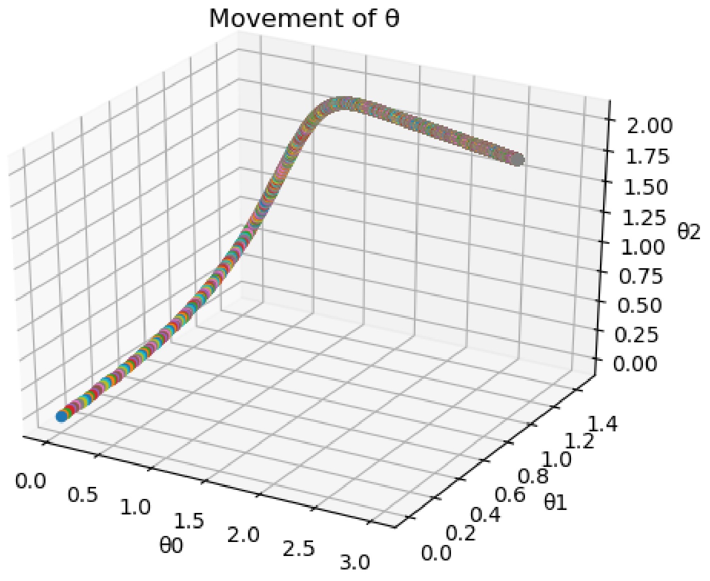
	$r = 1$	$r = 100$	$r = 10000$	$r = 1000000$
time	0.2688	0.2711	24.6375	6.4712
epochs	1	2	1006	201
cost	0.011561	0.010233	0.009833	0.009834

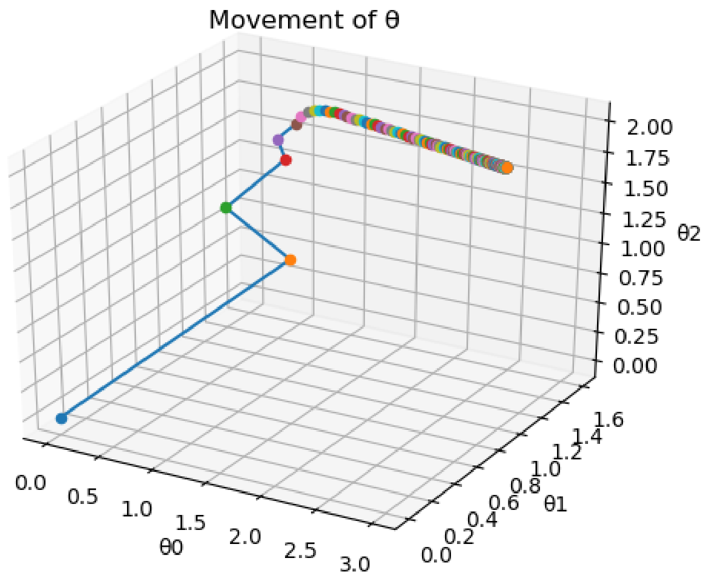
Time taken by $r = 1000000$ on learning rate 0.001 was very very high. So to reduce the time I made the learning rate 0.1 which took 6 seconds. Relative speed of convergence is $r = 1 > r = 100 > r = 10000 > r = 1000000$. The test error with respect to the prediction of the original hypothesis is 0.009829 which is close to the error obtained using learned hypothesis in each case. As batch size increases it approaches to the test error with respect to the prediction of the original hypothesis.

4. As we keep increasing the size of batch, the movement of parameters get smoother. It make intuitive sense because, by increasing batch size we are averaging over large data and hence we get less noise. Average cancels out too high and too low value to normal level.




Movement of θ
θ0
θ1
θ2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.2
0.4
0.6
0.8
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

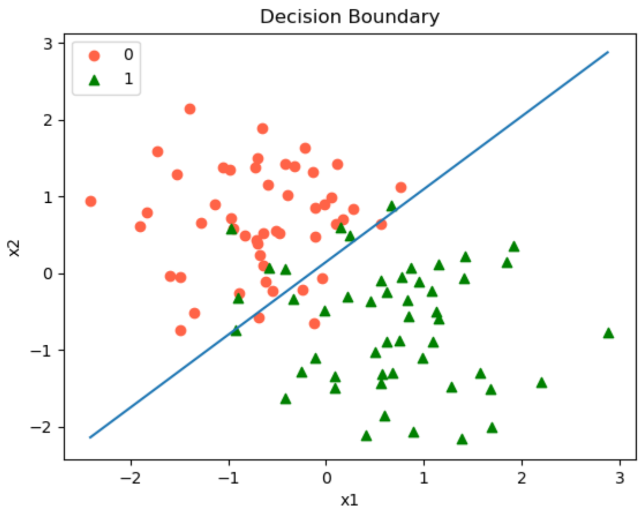

Movement of θ
θ0
θ1
θ2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.2
0.4
0.6
0.8
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



3. Logistic Regression

1. The coefficients resulting from your fit is $(0.40125316, 2.5885477, -2.72558849)$

2. Training data and decision boundary is shown below

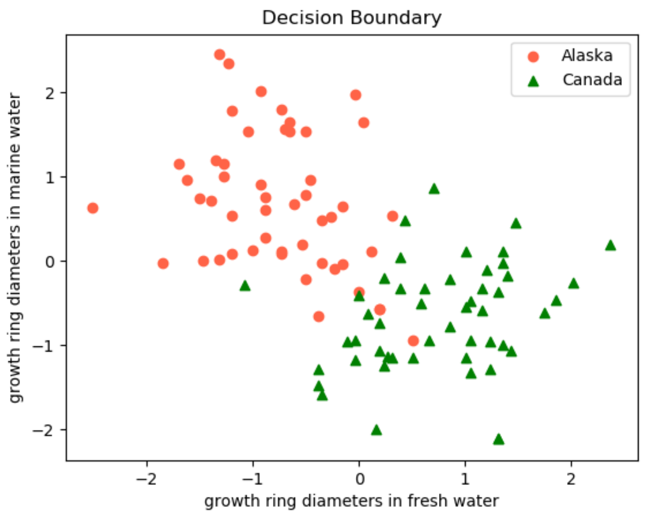


4. Gaussian Discrmimant Analysis

1. the values of the means, $\mu_0 = (-0.75529433, 0.68509431)$ and $\mu_1 = (0.75529433, -0.68509431)$, and the co-variance matrix

$$\sum = \begin{bmatrix} 21.47652391 & -1.12361397 \\ -1.12361397 & 26.53228963 \end{bmatrix}$$

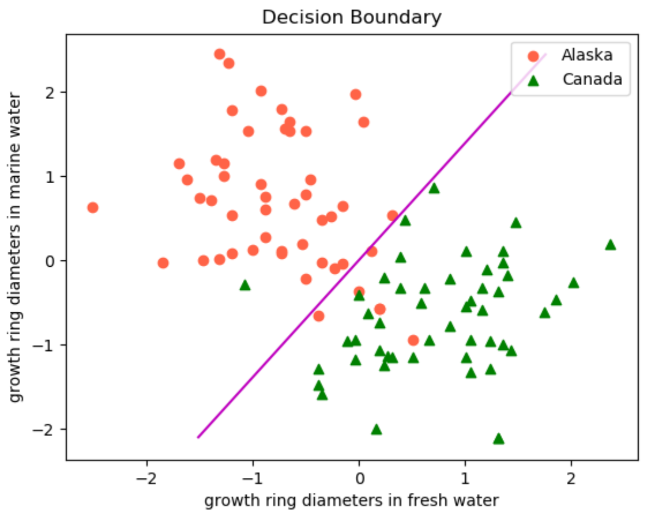
2. The training data corresponding to the two coordinates of the input features is shown below



3. Equation of the linear boundary separating the two regions in terms of the parameters μ_0, μ_1 and $\sum$ is shown below.

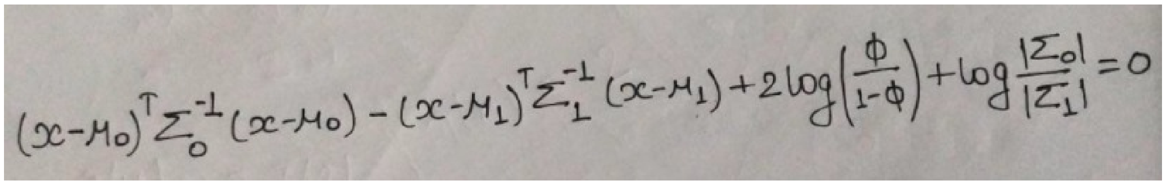
$$(x-\mu_1)\Sigma^{-1}(x-\mu_1)^T - (x-\mu_0)\Sigma^{-1}(x-\mu_0)^T - 2\log\left(\frac{\phi}{1-\phi}\right) = 0$$

Linear boundary obtained is shown below



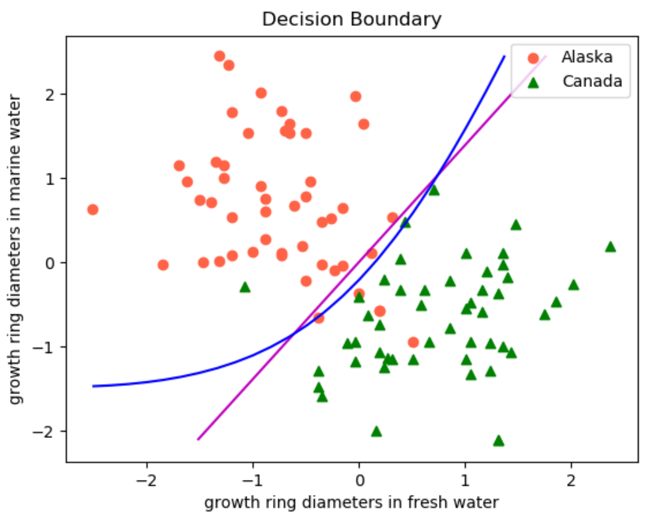
4. The values of the parameter estimates i.e. $\mu_0 = (-0.75529433, 0.68509431)$, $\mu_1 = (0.75529433, -0.68509431)$, $\sum_0 = \begin{bmatrix} 21.47652391 & -1.12361397 \\ -1.12361397 & 26.53228963 \end{bmatrix}$, and $\sum_1 = \begin{bmatrix} 21.47652391 & -1.12361397 \\ -1.12361397 & 26.53228963 \end{bmatrix}$

5. Equation of the quadratic boundary separating the two regions in terms of the parameters μ_0, μ_1 and $\sum$ is shown below.



$$(x-\mu_0)^T \Sigma_0^{-1} (x-\mu_0) - (x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1) + 2\log\left(\frac{\phi}{1-\phi}\right) + \log\frac{|\Sigma_0|}{|\Sigma_1|} = 0$$

Quadratic boundary obtained is shown below



6. When we assume our training data to be normally distributed and also if we further assume that both our classes have same spread that is co-variance matrix then in that case we have linear boundary obtained separating two classes. But if we don't assume this additional strong assumption and let our data tell us what is the co-variance matrix for each class then we end up with more general equation for the boundary. We allow our algorithm to learn from data. In case of less data given to us, we may over-fit the data in later approach while in former approach we may work good with less data.