

COL774 Assignment – 1

Anirudha Kulkarni

2019CS50421

Code running instructions:

1. All codes are arranged in respective folders
2. Each question contains a asset folder where all plots/GIFs will be stored
3. Each code is in a python file and the parameters are written in the very first line of each function which can be changed
4. Data sets need to be put in a directory in submission directory with name “data” which can be passed as an argument if not
5. python files can be run with `python q1.py`

Q1: Linear Regression

Update rule,

$$\theta_j := \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}$$

a)

Learning rate: 0.1

Stopping criteria: Stop if either of condition is met:

1. Number of iterations exceeds max iterations = 1000
2. Absolute error is less than epsilon = 10^{-9}
3. Absolute difference between last error and current error is less than epsilon = 10^{-9}

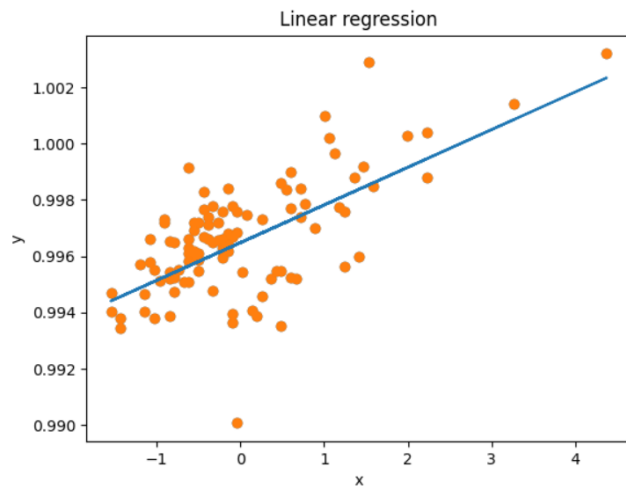
Final Parameters: $\theta_0 = 0.99648753$, $\theta_1 = 0.00134002$

Comments:

Smaller learning rates fits data more efficiently i.e. upto 10^{-10} but require larger iterations. 10^{-9} is still good considering it converges in much less iterations and cost decrease is far less.

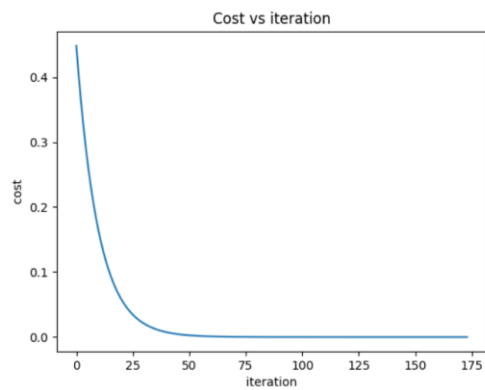
LR 0.001 took 600 iterations to converge and LR 0.1 took 174 iterations but the difference in cost obtained is less than 10^{-9} . Larger LR like 5 quickly diverged the cost

b)

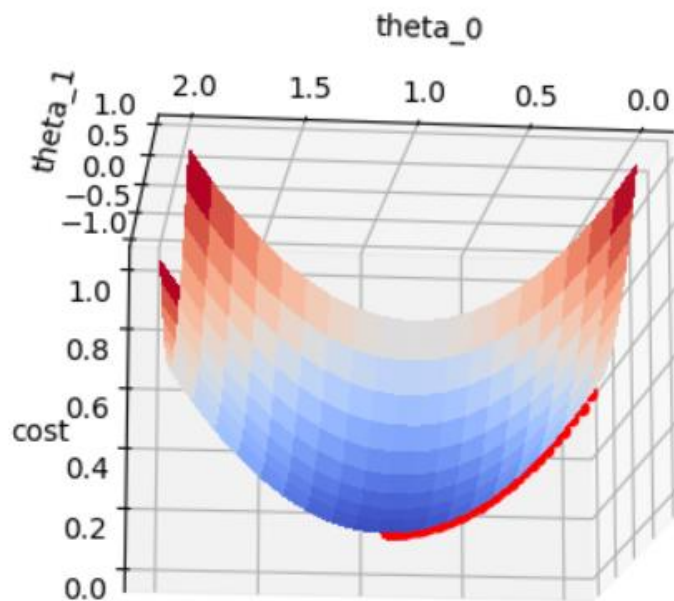


Data and hypothesis function for linear regression at LR = 0.1

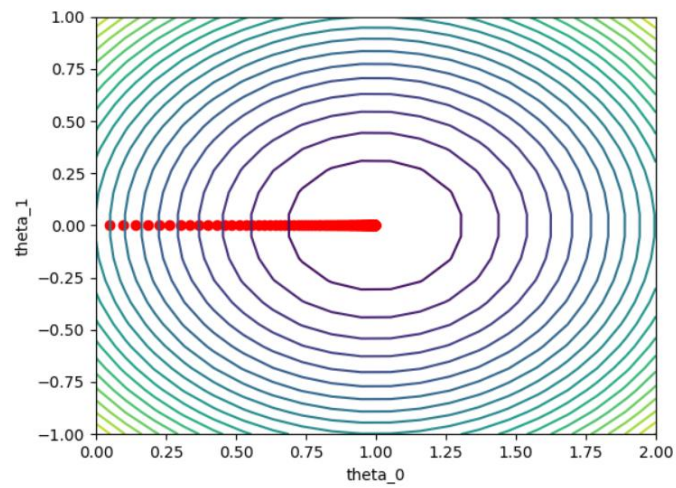
Cost function:



c) 3-dimensional mesh of error function $J(\theta)$ Contours

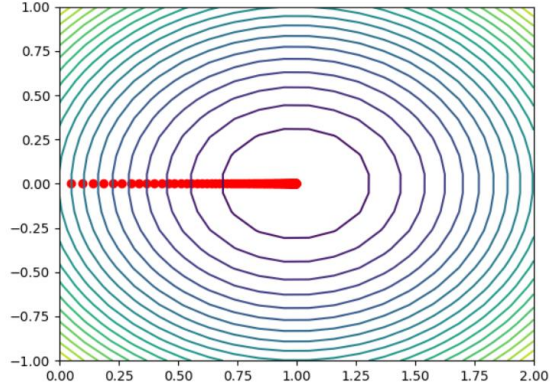
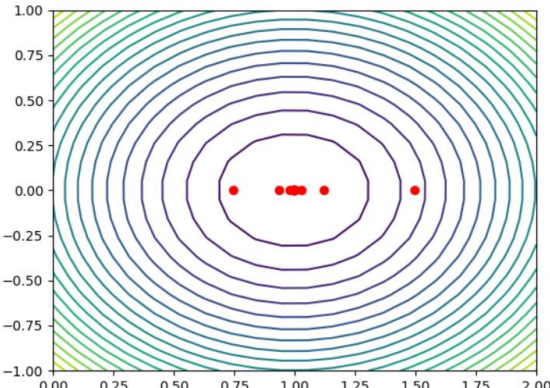
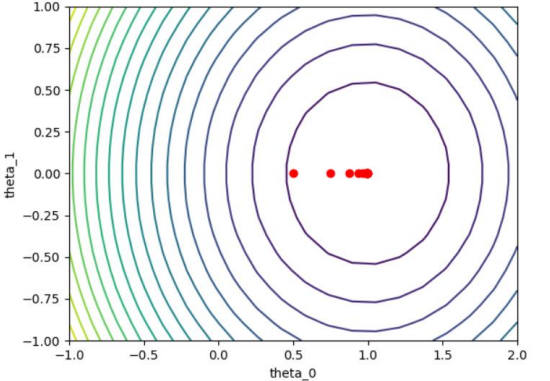


d) Contours of error function after each iteration



e)

η	Contours	Iterations	Cost	θ_0	θ_1
0.001	<p>(Alternate 100 points are plotted to reduce complexity)</p>	13114	2.19×10^{-6}	0.99520708	0.0013383
0.025		651	1.23×10^{-6}	0.99634329	0.00133982

0.1		174	1.20×10^{-6}	0.99648753	0.00134002
3		16	1.19×10^{-6}	0.9966	0.0013
1		16	1.19×10^{-6}	0.9966	0.0013

The contours indicate that as the learning rate increases cost start to converge fast but after a certain limit cost starts to diverge. At learning rates like 3 it oscillates but still converges but at higher learning rates like 10 the cost diverges to very large values in few iterations.

Q2. Stochastic Gradient Descent (SGD)

a) Given dataset

$$X^1 \sim N(3, 4)$$

$$X^2 \sim N(-1, 4)$$

$$Y = \theta^T X + \varepsilon$$

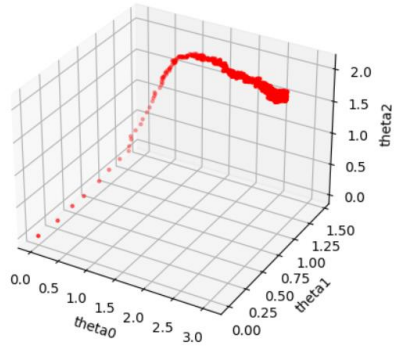
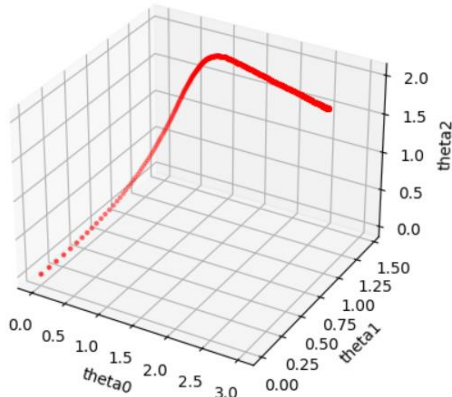
Learning rate is 0.001

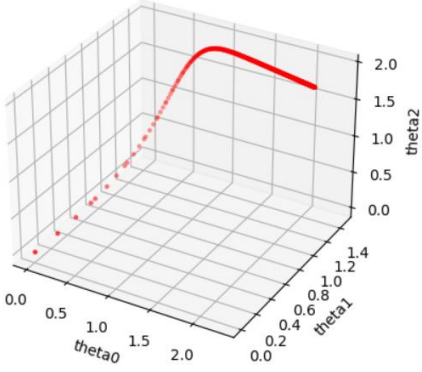
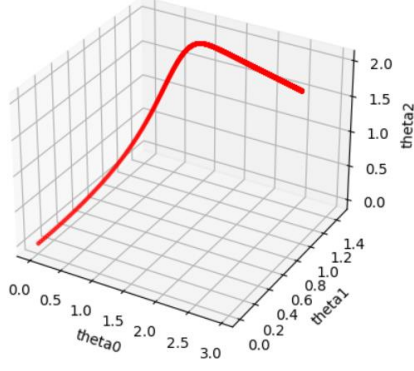
b),c),e)

Convergence Criteria:

1. Checkpoint: According to Andrew NG's video and suggestion, let checkpoint be a parameter which decides number of iterations over which the cost is being averaged before checking convergence criteria. Checkpoint is inversely proportional to batch size. As batch size decreases, noisier is the step of theta. Hence, we need to take average of larger iterations

2. Epsilon convergence: we define epsilon = 10^{-9} as the minimum difference between adjacent costs

Train loss	r	Checkpoint	Movement of Theta	Theta parameters
Loss: 1.0029862840537156	1	1000	<p>3d plot of theta</p> 	[[2.9966591 1] [1.006016] [1.97046427]]
Loss: 1.000302316973423	100	100	<p>3d plot of theta</p> 	[[3.0009300 9] [1.0019621] [1.99806702]]

Loss: 1.00460459161673 43	10000	10	3d plot of theta 	[[2.8244210 6] [1.03746836] [1.9862213]]
Loss: 1.00055999956720 72	100000 0	1	3d plot of theta 	[[2.9517945 4] [1.00982069] [1.99552167]]

d) Test Data:

r	Cost with original parameters	Cost with learned parameters	Iterations
1	0.9829469215000003	1.0331299888910825	100001
100	0.9829469215000003	0.9934818758230094	100001
10000	0.9829469215000003	1.0004189681299212	20001
1000000	0.9829469215000003	0.9896101576514509	6001

Convergence remarks:

1. Larger batch size require less number of iterations but net epochs are more
2. Converged values are almost same due to smaller learning rate. Though it should be expected that the theta converges more at higher batch sizes due to noisy steps with smaller batch sizes as seen in above figures but the learning rate is small enough to counteract the effect. The divergence can be seen in larger learning rates like 0.1 where larger batch size implies closer convergence
3. Cost with r: as r increases the cost shows decreasing trend which can be explained as above reasons

Movement of Theta: Theta moves with noise in case of smaller batch sizes as the updates are much more frequent and hence with noise. But the actual graph might look bit different as number of points in each case are different.

Q3.

a)

Log likelihood of loss function:

$$LL(\theta) = \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))$$

Where hypothesis is:

$$h_{\theta}(x) = \frac{1}{1 + e^{-X_i \cdot \theta}}$$

Newton's method uses hessian to reach the optimal solution,

$$\theta := \theta - H^{-1} \nabla_{\theta} \ell(\theta).$$

where hessian is:

$$H_{\theta}(LL(\theta)) = X^T \cdot \text{diag}(\sigma(X \cdot \theta)(1 - \sigma(X \cdot \theta))) \cdot X$$

Newton's Method:

Θ : [[0.40125316] [2.5885477] [-2.7255885]]

$Y = mX + c$ where,

c: -0.1472170739885948 m: -0.9497206596108467

accuracy: 0.88

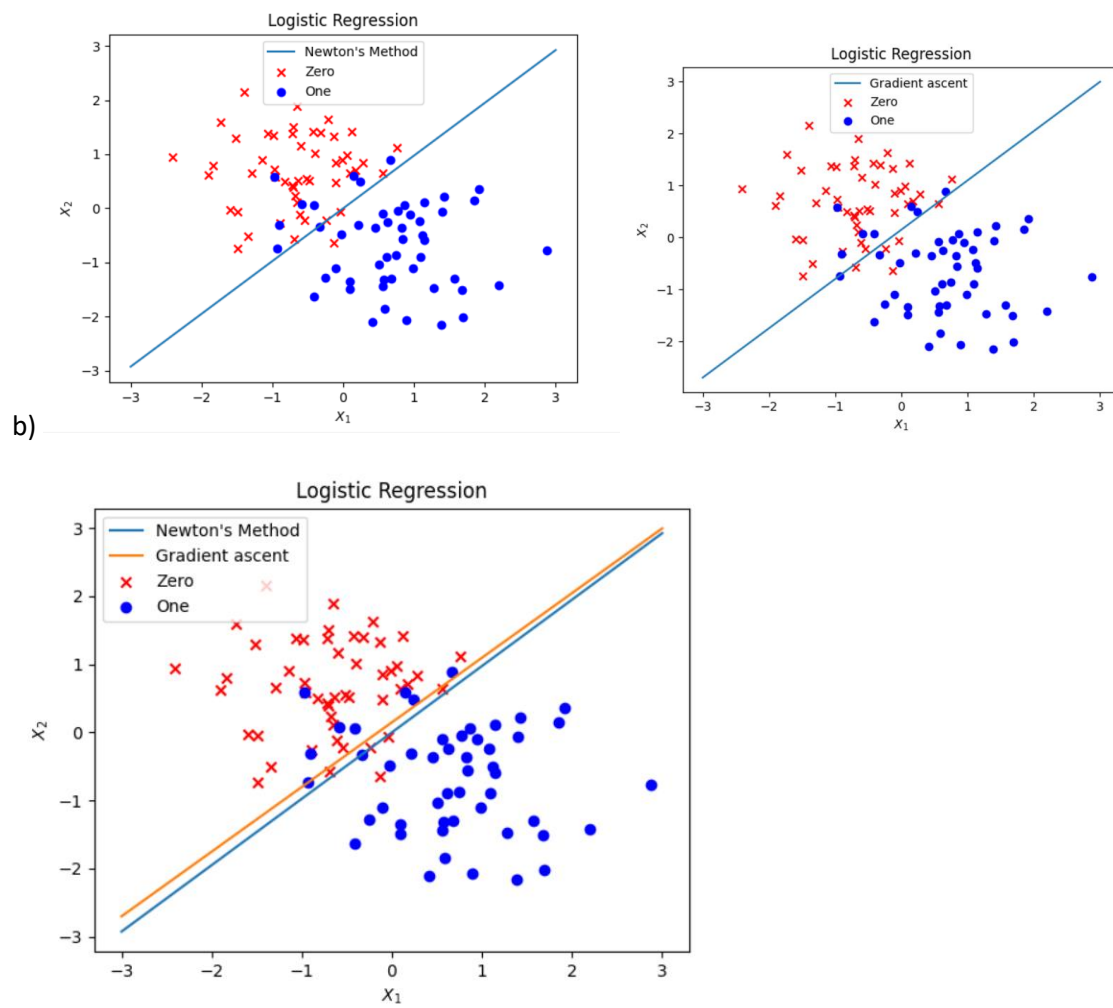
Gradient ascent Method:

Θ : $\begin{bmatrix} -4.22396529e-14 \\ 9.06314822e+01 \\ -9.29313406e+01 \end{bmatrix}$

$Y = mX + c$ where

c: 4.545253803171075e-16 m: -0.9752520691913545

accuracy: 0.9



Q4.

a)

Equations used from supplementary material:

$$\mu_0 = \frac{\sum_{i=1}^m \mathbb{1}\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^m \mathbb{1}\{y^{(i)} = 0\}}$$
$$\mu_1 = \frac{\sum_{i=1}^m \mathbb{1}\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^m \mathbb{1}\{y^{(i)} = 1\}}$$
$$\Sigma = \frac{\sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T}{m}$$

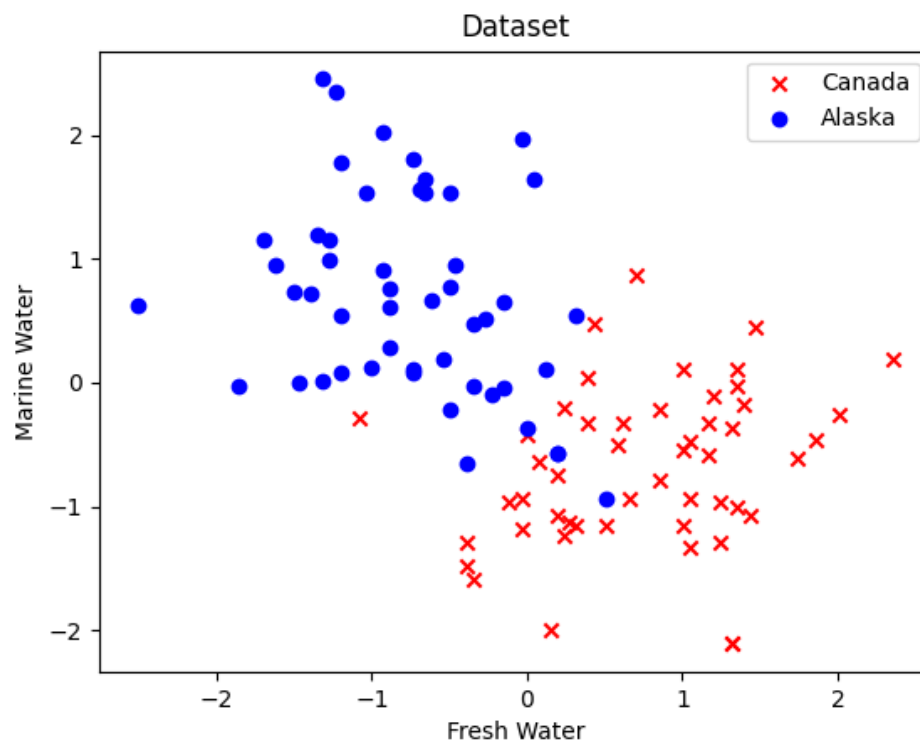
Decision boundary is evaluated by equating the log likelihood function of generating the entire data set to zero.

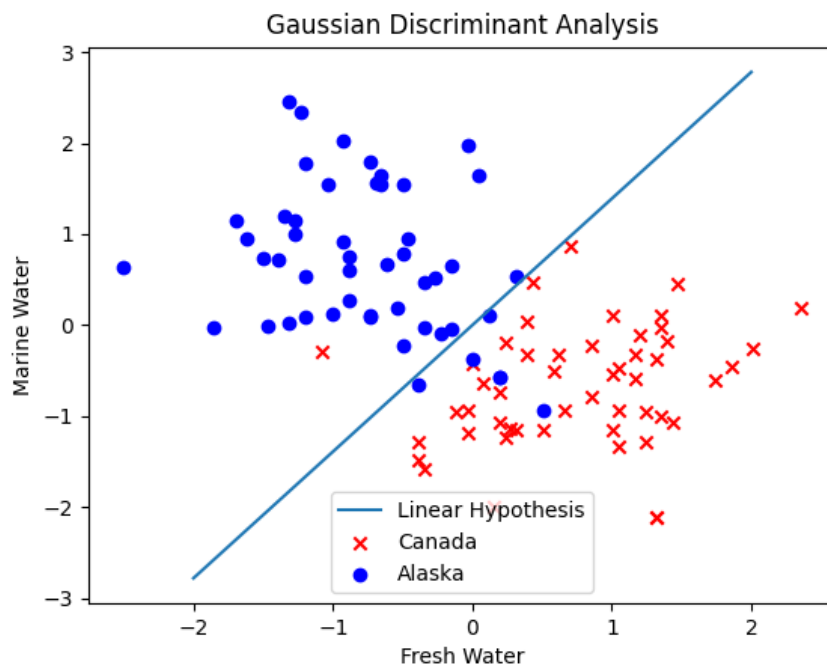
$$\mu_0 = [-0.75529433, 0.68509431]$$

$$\mu_1 = [0.75529433, -0.68509431]$$

$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

b)





c)

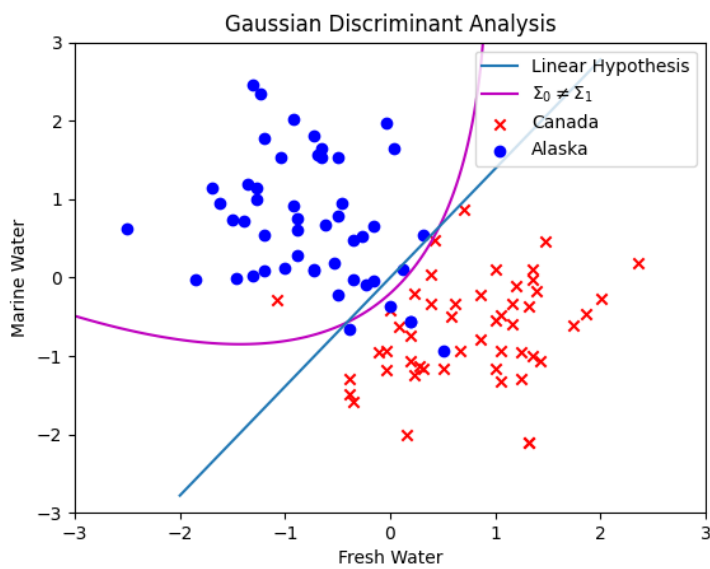
d)

$$\mu_0 = [-0.75529433, 0.68509431]$$

$$\mu_1 = [0.75529433, -0.68509431]$$

$$\Sigma_0 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$



e)

f) Quadratic in this case performs better as it classifies 2 more points correctly compared with linear Hypothesis. Taking account the covariance increases the accuracy but might overfit in case of large datasets.