Report

How to read the report:

1. Report contains 4 sections titled with question number.
2. Each section contains 3 sub-sections: (how to run, what I did, observations)

How to run files:

1. Directory consists of four sub-directories Q1, Q2, Q3 and Q4. Each sub-directory contains python file (for example, 1.py in Q1).
2. Please go into each sub-directory and run the files as python 1.py --arguments. More details of each argument are given in **How to Run** sub-sections of each section.
3. Python version : 3.8.5

I have also added requirements.txt containing all the libraries used in my virtual environment on windows.

Q1.

How to Run:

python 1.py -x 'Path to CSV linearX.csv' -y 'Path to CSV linearY.csv' -b 'Path to output png for Q1.b' -c1 'Path to png file for Q1.c' -d1 'Path to png file for Q1.d' -c2 'Path to mp4 file for Q1.c' -d2 'Path to mp4 file for Q1.d' -n 'Learning rate' --delta 'Convergence criteria'

Note: I have set the all parameters with default values, i.e.,

```
-x : '../ass1_data/data/q1/linearX.csv'
-y : '../ass1_data/data/q1/linearY.csv'
-b : '1b.png'
-c1 : '1c.png'
-d1 : '1d.png'
-c2 : '1c.mp4'
-d2 : '1d.mp4'
-n : 0.025
```

```
--delta : 1e-10
```

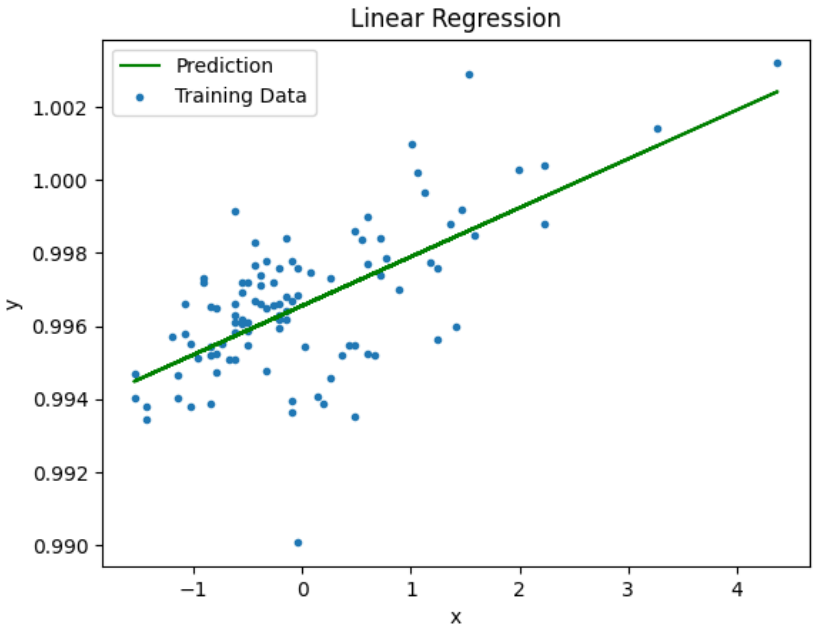
What I did:

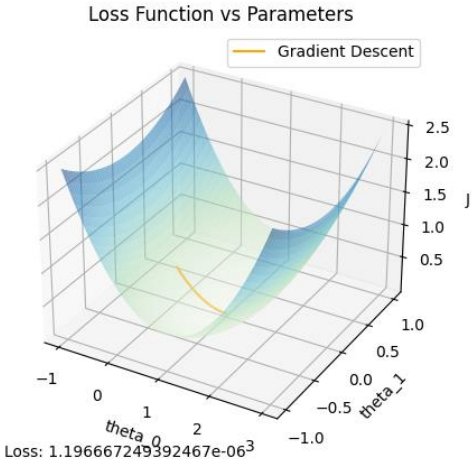
1. Normalized x : normalize(x)
2. Performed gradient descent :
   batch_gradient_descent(x,y,theta,lr,delta)
3. Calculated loss : J
4. Added plots and animation using matplotlib.

Observations:

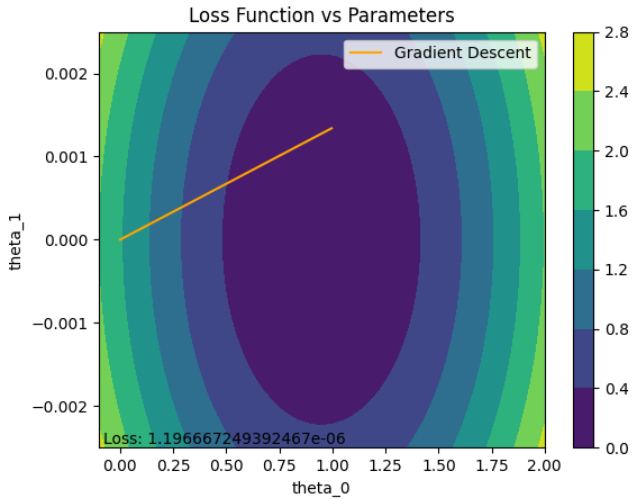
a. Learning Rate : 0.025,
   Stopping Criteria: delta = 1e-10,
   Final Parameters:
   theta:
   [[0.99655882]
   [0.00134011]], i.e., theta_0 = 0.99655882, and theta_1 = 0.00134011

b.





c.



d.

Animation included in directory as mp4 files for part c and d.

e. When I repeated the above plots for learning rate = 0.001, 0.025 and 0.1, I noticed that the training time increases with decreasing learning rate, number of iterations also increase. Moreover, as seen in gradient descent animations, the gradient descent also moves slowly for lower learning rate.

Q1 directory already contains plots and animations named as <question_number>_x.<extension> where “0.x” is the learning rate.

Q2.

How to Run:

python 2.py -t ‘Path to q2test.csv’ -l ‘path to loss curve png output’ -p ‘path to png file for showing delta plot with iterations’ -n ‘learning rate’ -b ‘batch size’ -k ‘Number of Iterations over which to take Average ‘ --delta ‘convergence criteria’

Default values were:

```
-t  :  ../ass1_data/data/q2/q2test.csv
-l  :  2_losses.png
-p  :  2.png
-n  :  0.001
-b  :  1
-k  :  5000
--delta  :  8e-5
```

What I did:

1. Did not perform normalization, instead sampled values using np.random.normal()
2.

Observations:

a. Sampling done
b. Each batch size had a different convergence criteria.
c. 1. Yes, the different algorithms (with different batch sizes) converge to same value under different carefully decided settings.
   2. The parameters learned were very similar to original hypothesis.
   3. The lower the batch size was, the faster the convergence was observed because model learned more quickly with smaller batch sizes.
   4. Number of iterations were the highest for lowest batch sizes.

Batch Size	Iterations	Epochs
1	375000	0.375

100	13600	1.36
10000	15012	150.12
1000000	11792	11792

5. Errors

Batch Size	Error
1	1.0325598499721198
100	0.9942059861480101
10000	0.9891972039534598
1000000	1.01983128638645

6. Original Hypothesis Error: 0.9829469215
7. The errors obtained were relatively similar to the Original Hypothesis Error.

Note: the parameters of each batch size are as follows:

Batch	Learning Rate	K	Delta
1	0.001	5000	8e-5
100	0.001	20	4e-5+5e-6
10000	0.001	18	1.2e-5
1000000	0.001	1	1e-6

8.

Batch	Theta_0	Theta_1	Theta_2
1	2.96305079	1.02869594	2.01550435
100	2.93427794	1.01428529	1.99817144
10000	2.9551381	1.00988383	1.99617959
1000000	2.88739119	1.02421128	1.99080953

d. When I plotted the loss curve and movement of theta in 3d plane, lower batch sizes had noisier curves and as the batch size was increased, the curves became smoother. Intuitively this makes sense, because when updating parameters, we only have a subset of data and hence more noise.

Q3.

How to Run:

python 3.py -x 'Path to logisticX.csv' -y 'path to logisticY.csv' -p 'path to png plot for Q3' --delta 'Convergence criteria'

I have the set the following values as default:

```
-x : ../ass1_data/data/q3/logisticX.csv
-y : ../ass1_data/data/q3/logisticY.csv
-p : 3.png
--delta : Convergence criteria
```

What I did:

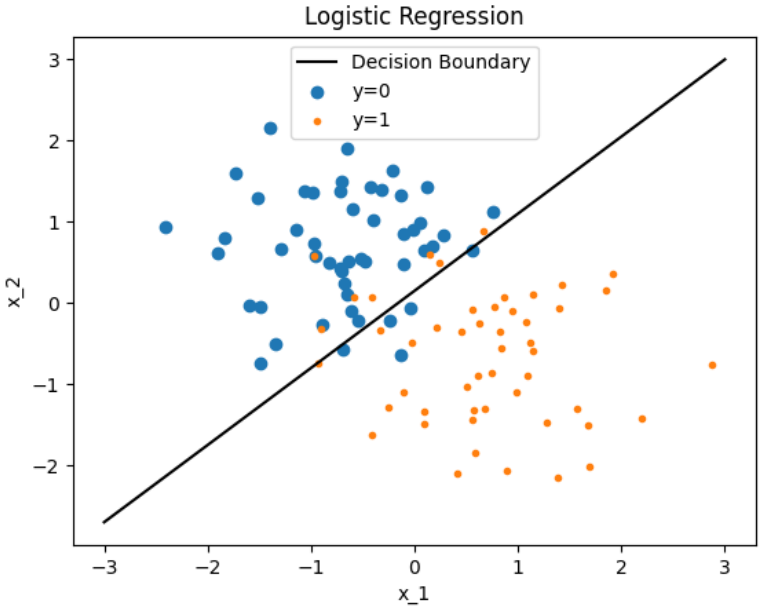
1. Normalized x : normalize(x)
2. Implemented sigmoid() function
3. Implemented hessian() function giving hessian matrix given x and theta as input.
4. Implemented newton optimization in newton() function.
5. Plotted the graphs using matplotlib.

Observations:

a. Theta came out to be:
   theta:
   [[0.40125316]
   [2.5885477]
   [-2.72558849]], i.e.
   Theta_0 = 0.40125316, theta_1 = 2.5885477, theta_2 = -2.72558849
b. The decision boundary perfectly fits the data as seen the following figure:

I also checked whether hessian matrix computed was correct by checking its symmetry and using Cholesky factorization on – (hessian matrix) to check if its negative semi-definite (because the

logistic regression loss function is concave and we generally perform gradient ascent).



Q4.

How to Run:

python 4.py -x 'Path to q4x.dat' -y 'Path to q4y.dat' -p 'path to png file for Q4 showing GDA decision boundaries'

Default values were:

```
-x  :  ../ass1_data/data/q4/q4x.dat
-y  :  ../ass1_data/data/q4/q4y.dat
-p  :  4.png
```

What I did:

1. Normalized input.
2. Calculated mu0, mu1, sigma, sigma0, sigm1 as per the equations derived in class. More details in observations.
3. Implemented linear_decision_boundary(), quadratic_decision_boundary() and quadratic_decision_boundary2().

4. Plotted graphs.

Observations:

a. mu_0 =

[[-0.75529433]

[0.68509431]]

mu_1 =

[[0.75529433]

[-0.68509431]]

Sigma =

[[0.42953048 -0.02247228]

[-0.02247228 0.53064579]]

c. The equation for decision boundary comes by assuming log(A) = 0, as discussed in class.

Equation of Linear Decision Boundary (common Sigma) is:

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

Equation of x2 in terms of x1 is:

$$x_2 = \frac{1}{\left((\mu_0^T - \mu_1^T)\Sigma^{-1}\right)_2}\left(-\left((\mu_0^T - \mu_1^T)\Sigma^{-1}\right)_1 x_1 - C + \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1)\right)$$

Where C = $log\left(\frac{1-\varphi}{\varphi}\right)$

d. mu_0 and mu_1 were same as in part a.

Sigma_0 =

[[0.38158978 -0.15486516]

[-0.15486516 0.64773717]]

Sigma_1 =

[[0.47747117 0.1099206]

[0.1099206 0.41355441]]

e. Equation of quadratic decision boundary is:

$$x^T(\Sigma_0^{-1} - \Sigma_1^{-1})x - 2\left(\mu_0^T \sum_0^{-1} - \mu_1^T \Sigma_1^{-1}\right)x + \mu_0^T \Sigma_0^{-1}\mu_0 - \mu_1^T \Sigma_1^{-1}\mu_1 = 2\log\left(\frac{|\Sigma_1|^{1/2}(1-\varphi)}{|\Sigma_0|^{1/2}\varphi}\right)$$

Equation of x2 in terms of x1: (solved by quadratic expansion and finding solutions to quadratic equation in x2)

$$x_2 = \frac{-(cx_1 + e) \pm \sqrt{(cx_1 + e)^2 - 4b(ax_1^2 + dx_1 + b)}}{2b}$$

Where:

$$a = (\Sigma_0^{-1} - \Sigma_1^{-1})_{11}$$

$$b = (\Sigma_0^{-1} - \Sigma_1^{-1})_{22}$$

$$c = (\Sigma_0^{-1} - \Sigma_1^{-1})_{12} + (\Sigma_0^{-1} - \Sigma_1^{-1})_{21}$$

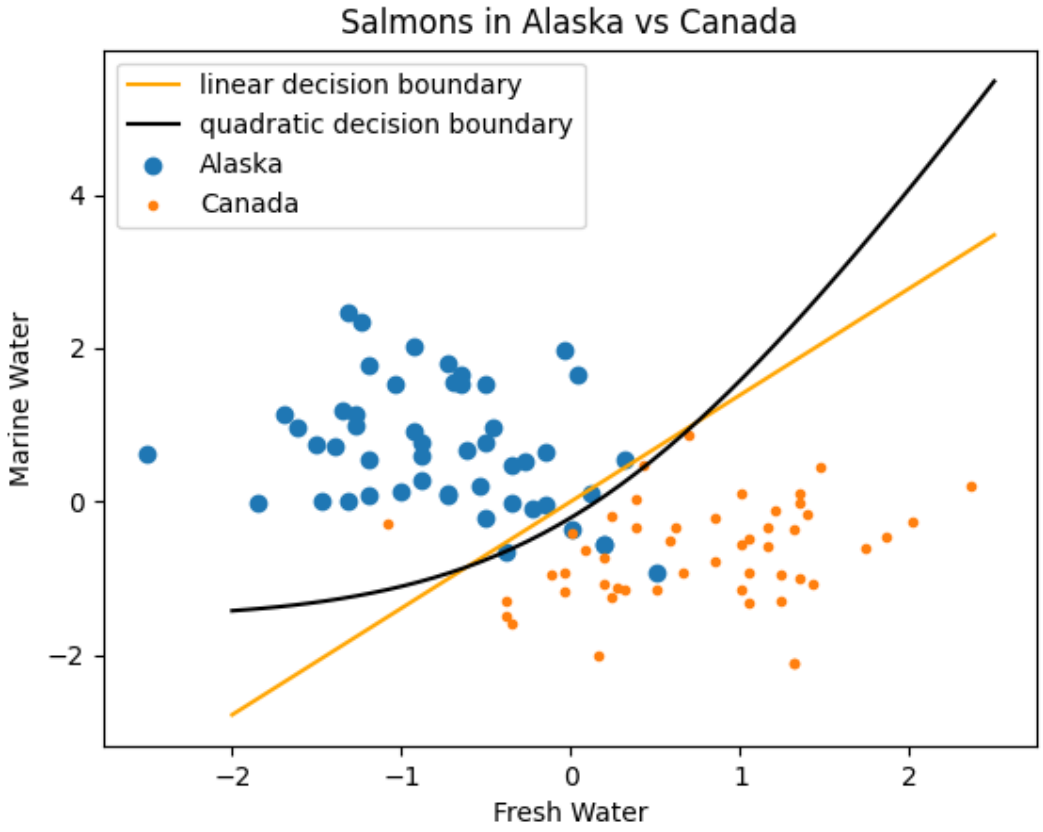
$$d = -2(\mu_0^T \Sigma_0^{-1} - \mu_1^T \Sigma_1^{-1})_1$$

$$e = -2(\mu_0^T \Sigma_0^{-1} - \mu_1^T \Sigma_1^{-1})_2$$

$$f = -2C' + \mu_0^T \Sigma_0^{-1}\mu_0 - \mu_1^T \Sigma_1^{-1}\mu_1$$

$$C' = \log\left(\frac{|\Sigma_1|^{1/2}(1-\varphi)}{|\Sigma_0|^{1/2}\varphi}\right)$$

The plot then comes out to be:



f. We see that linear decision boundary is a straight line and quadratic decision boundary is slightly bent towards the Alaska examples. Moreover, on solving quadratic equation in x2 there were two solutions, and I used that solution which came in between the points and ignored the one which completely outside the figure.

Using two values of x2 gave a hyperbola.

Moreover, I chose the value of phi = 0.5, based on frequencies of y=0 and y=1 examples in training set.