Assignment 1 Report

Shubham, 2018CS10641

September 16, 2021

"main.py" for running the program.

Q1. Linear Regression

Data normalized (x's) to have zero mean and unit variance in each dimension

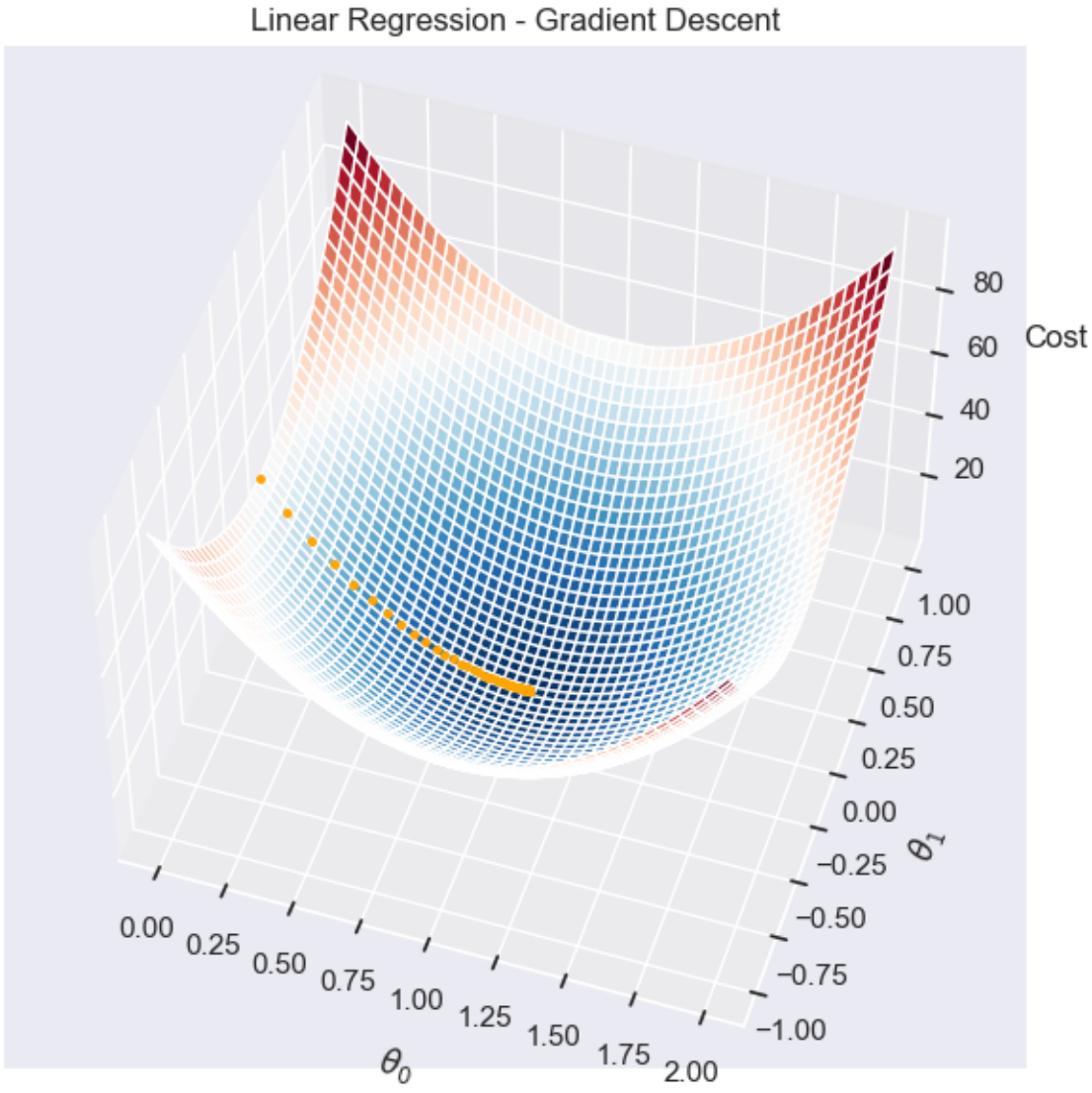
Part a

Learning Rate (η) - 0.001
Stopping criteria - Based on number of iterations and the minimum cost value.
$cost < 0.0001$ or $iterations > 125$
Final Parameters Learned - $\theta_0 = 0.99663196$ and $\theta_1 = 0.00135794$

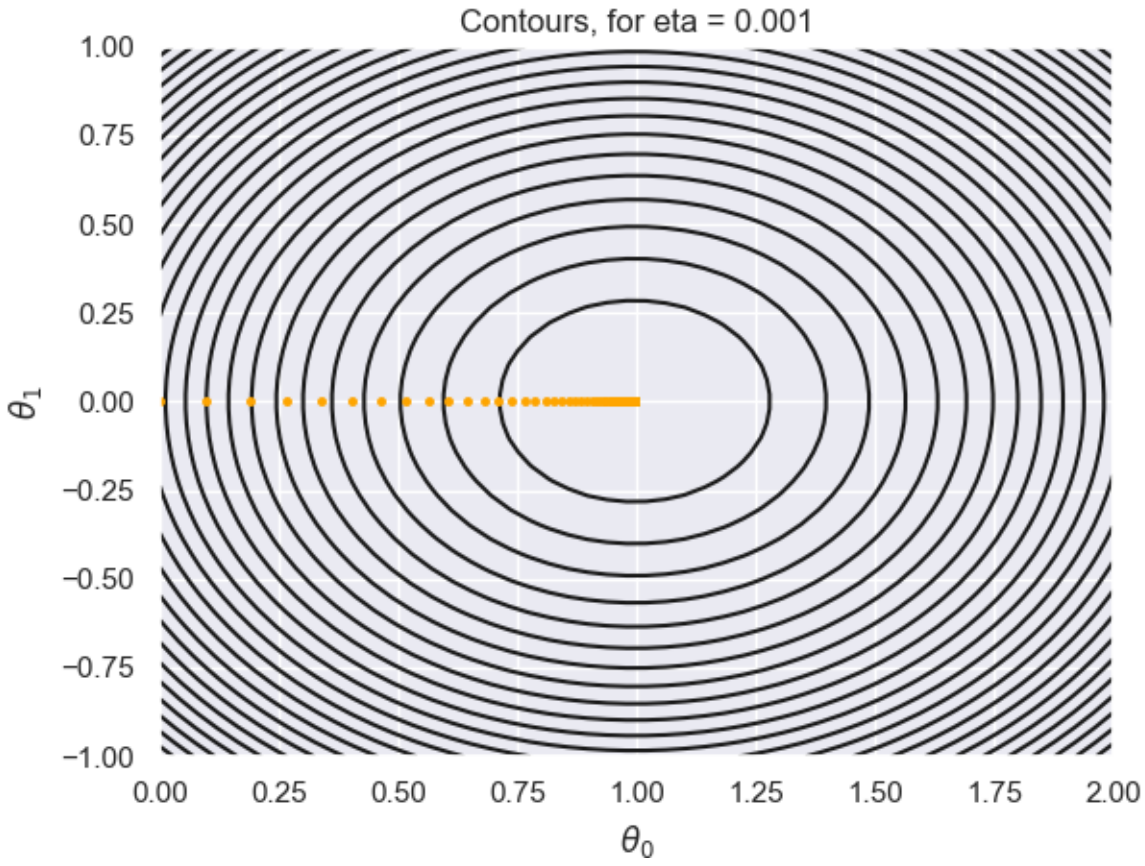
Part b


Linear Regression
Density of wine
1.002
1.000
0.998
0.996
0.994
0.992
0.990
−2
−1
0
1
2
3
4
Acidity of wine(Normalised)

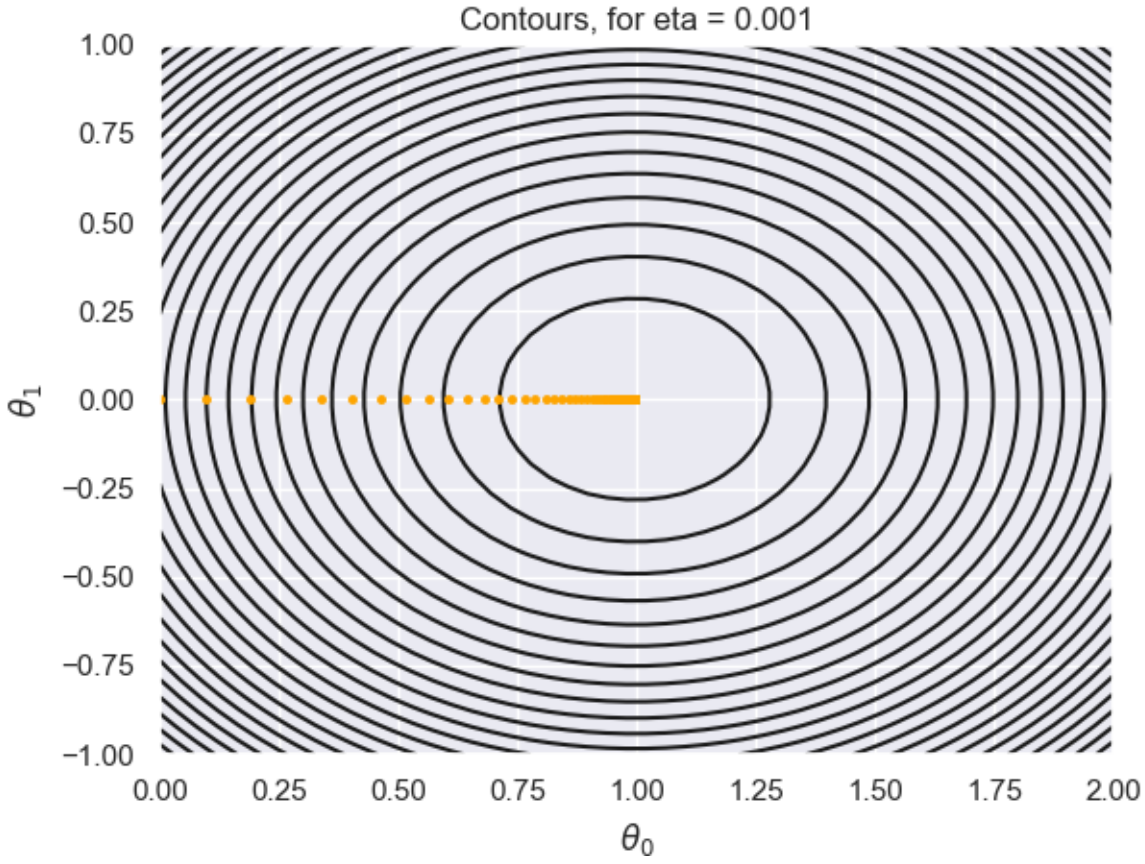
Part c

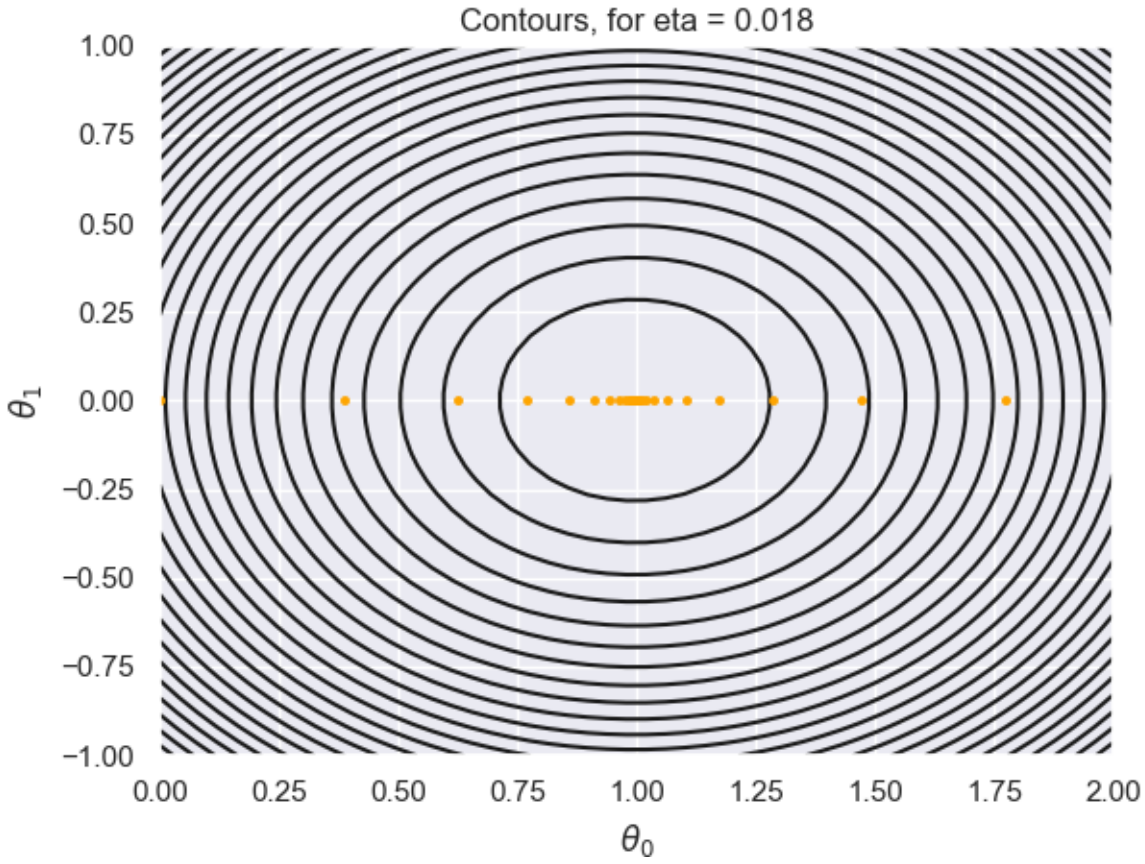

Linear Regression - Gradient Descent
Cost
80
60
40
20
θ_1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

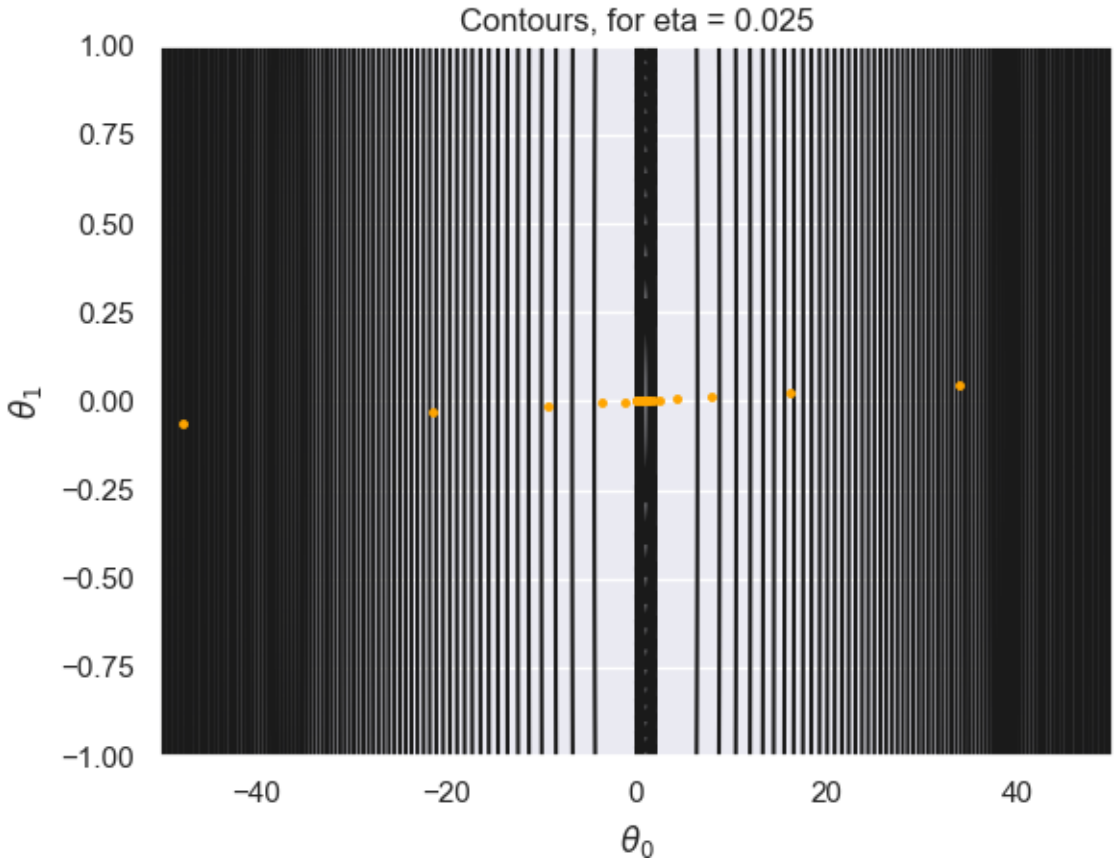
Part d


Contours, for eta = 0.001
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
θ_0

Part e


Contours, for eta = 0.001
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
θ_0


Contours, for eta = 0.018
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
θ_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
θ_0



Above are the plots of contours and for different learning rates. We can observe that as learning rate increases beyond a certain point, the cost functions as well as the learned parameters start diverging. We can observe that at $\eta = 0.001$, the cost functions converges very well. But well we increase η to 0.018, there are small oscillations but the functions still converges. When we further increase η to 0.025, the cost function diverges. As η is further increases, the overshooting increases more and more. Thus we learnt how important it is to select an appropriate learning rate.

Q2. Sampling and Stochastic Gradient Descent

Part a

Sampled 1 million data points taking values as described in problem statement.

Part b and c

Batch size = 1000000
η: 0.001
θ learned: [2.98232101 1.00396182 1.99864671]. Same as the initial parameters we used to sample the data.
Convergence Criteria: Difference between the mean costs of 2 adjacent 1 iterations over the batches.
Number of epochs = 12000
Time Taken(s): 1713.908392906188965
Test error on Original hypothesis: 0.9829469215
Test error on learned hypothesis: 0.983633467583517
Difference in errors; 0.0006865460835170367

Converges slowly but less randomly. The smooth path of θ convergence can be seen in part(d). Difference in errors is small.

Batch size = 10000
η: 0.001
θ learned: [2.99417947 1.00109212 1.99912775]. Same as the initial parameters we used to sample the data.
Convergence Criteria: Difference between the mean costs of 2 adjacent 100 iterations over the batches.
Number of epochs = 465
Time Taken(s): 17.908392906188965
Test error on Original hypothesis: 0.9829469215
Test error on learned hypothesis: 0.9831401010577495
Difference in errors; 0.00019317955774955653

Converges way faster that the 1 million case as we can see from the time taken as well as the number of epochs. Error still small as the batch is still large. Little randomness is there as we can see in part(d).

Batch size = 100
η: 0.001
θ learned: [2.99622256 1.00319295 1.99998647]. Same as the initial parameters we used to sample the data.
Convergence Criteria: Difference between the mean costs of 2 adjacent 10000 iterations over the batches.
Number of epochs = 8
Time Taken(s): 1.6076371669769287
Test error on Original hypothesis: 0.9829469215
Test error on learned hypothesis: 0.9828314700748714
Difference in errors; 0.00011545142512858764

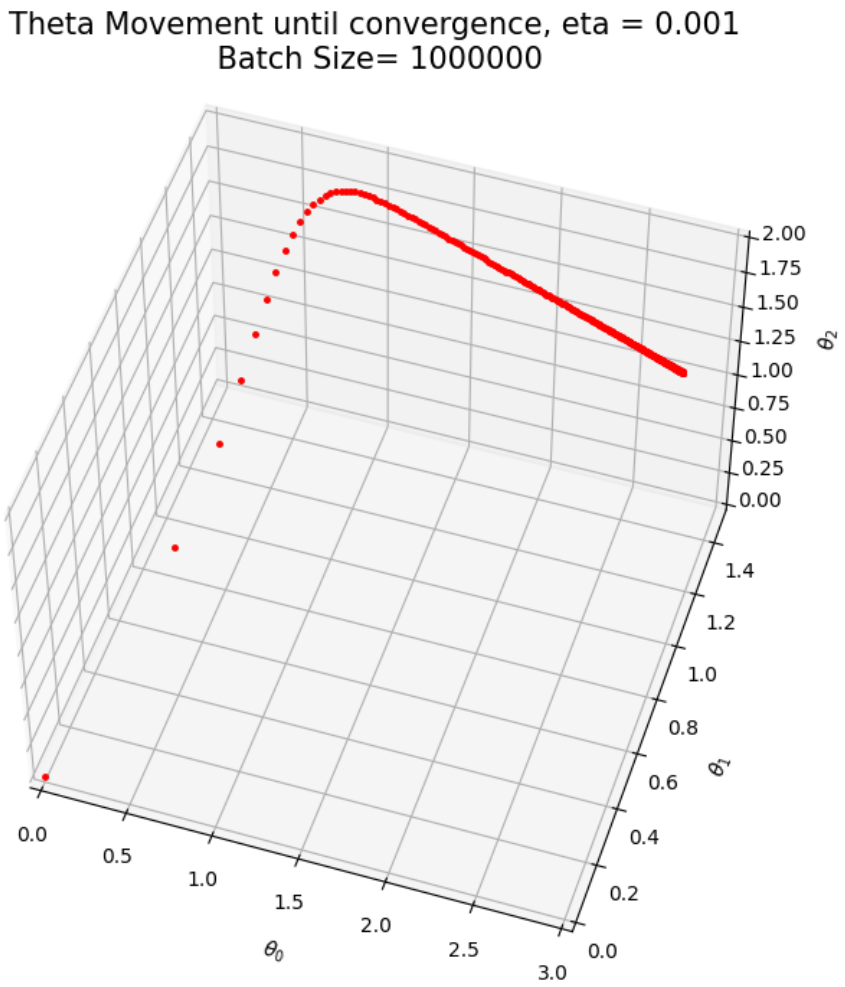
Convergence is not as smooth as the larger batch sizes, as the updates based on smaller batch sizes are more random and stochasticity is there. But the convergence is really fast. The test error is still small as the randomness is still less.

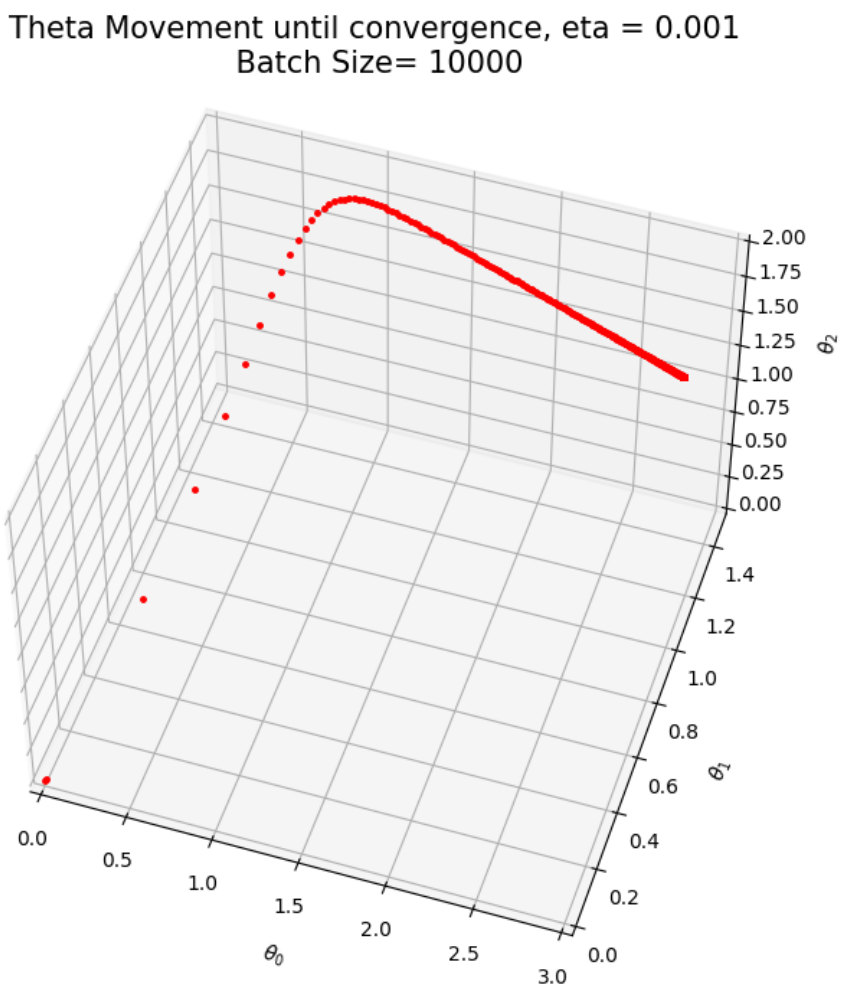
Batch size = 1
η: 0.001

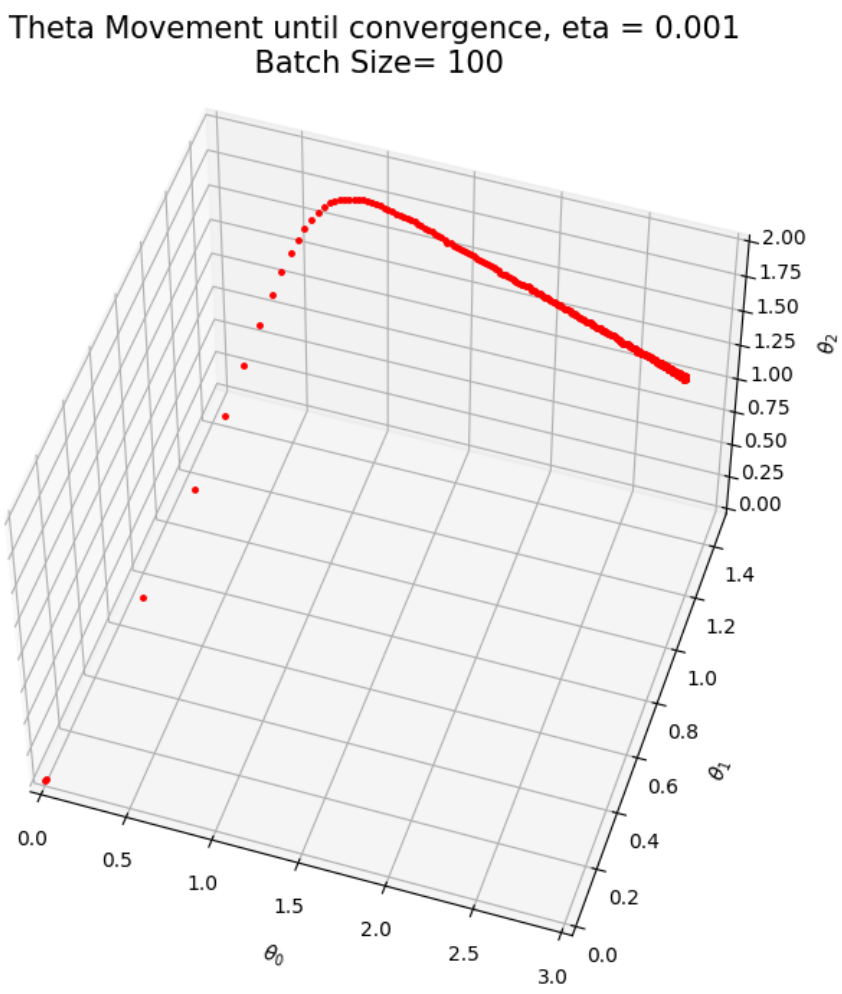
θ learned: [3.0209866 0.98763753 2.0380056]. Same as the initial parameters we used to sample the data.
Convergence Criteria: Difference between the mean costs of 2 adjacent 15000 iterations over the batch.
Number of epochs = ¡‘1
Time Taken(s): 1.2646143436431885
Test error on Original hypothesis: 0.9829469215
Test error on learned hypothesis: 1.0738925348744897
Difference in errors; 0.0909456133744897

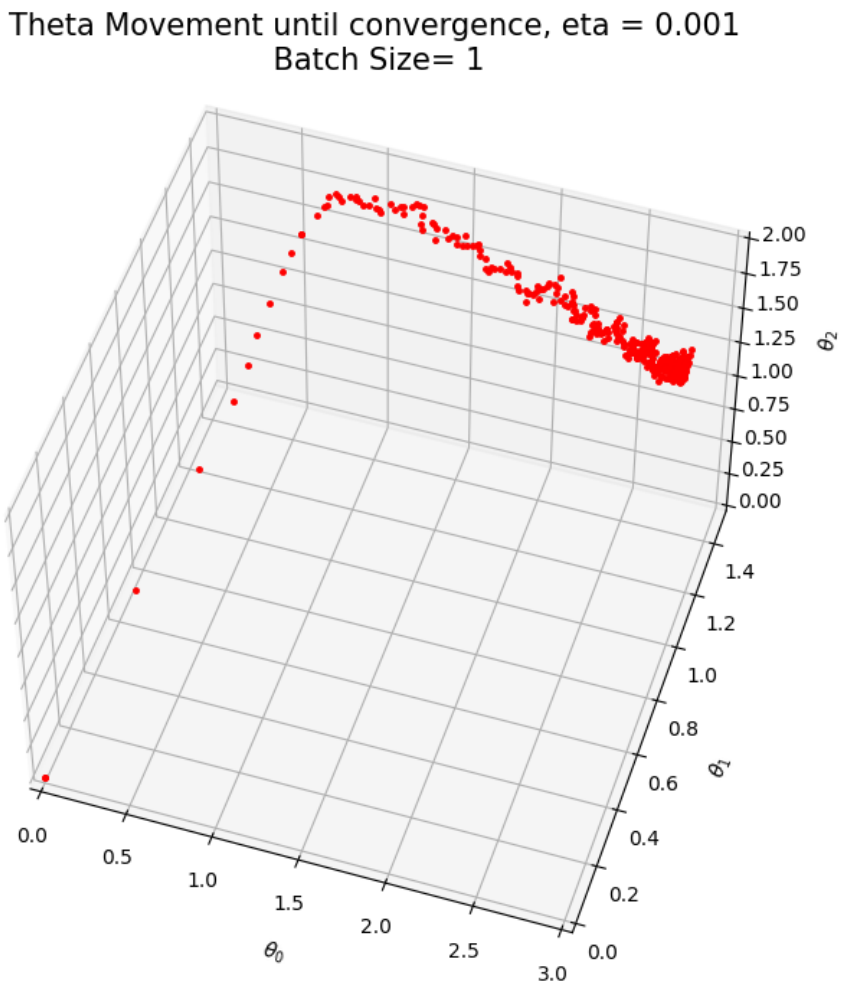
The convergence in this case is really fast, as it takes less than one epoch through the data to converge. But as we can see the figure in part 1(d), the path is highly random due to stochastic randomness. The test error is a bit higher due to the stochasticity. The model keeps hovering around the minima and doesn't actually converge.

Part c




Theta Movement until convergence, eta = 0.001
Batch Size= 10000
θ_0
θ_1
θ_2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Theta Movement until convergence, eta = 0.001
Batch Size= 100
θ_0
θ_1
θ_2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



Yes, the movement of theta towards the final parameters does make intuitive sense. As the batch size decreases, there is an increase in randomness(as expected) of theta movement due to stochasticity. Batch size 1 has the highest stochasticity, while for batch sizes 10000 and 1000000, the convergence is pretty smooth.

Logistic Regression

Part a

Final Parameters Learned - [0.00921424, -0.00898329, -0.00064394]

Part b


Logistic regression with Decision Boundary learned using Newton's method
Boundary
Label=0
Label=1
Feature x2
Feature x1
3
2
1
0
−1
−2
−2
−1
0
1
2

Gaussian Discriminant Analysis

Part a

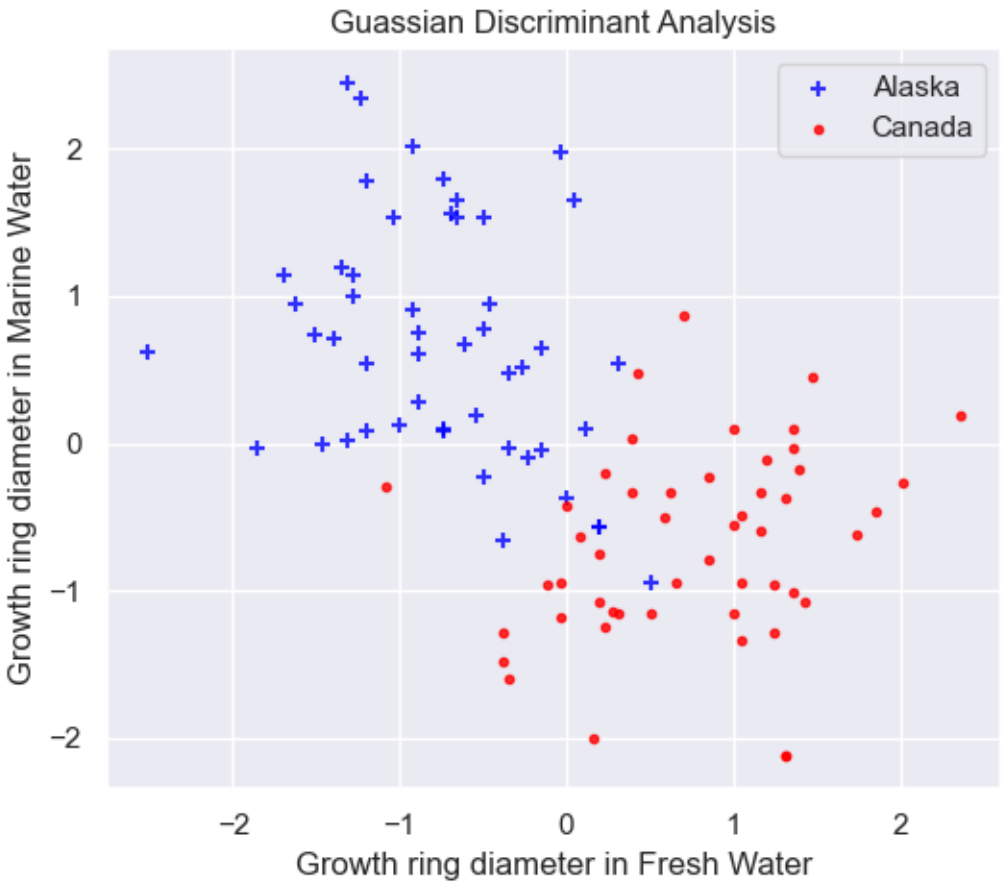
$$\phi = 0.5$$

$$\mu_{Alaska} = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

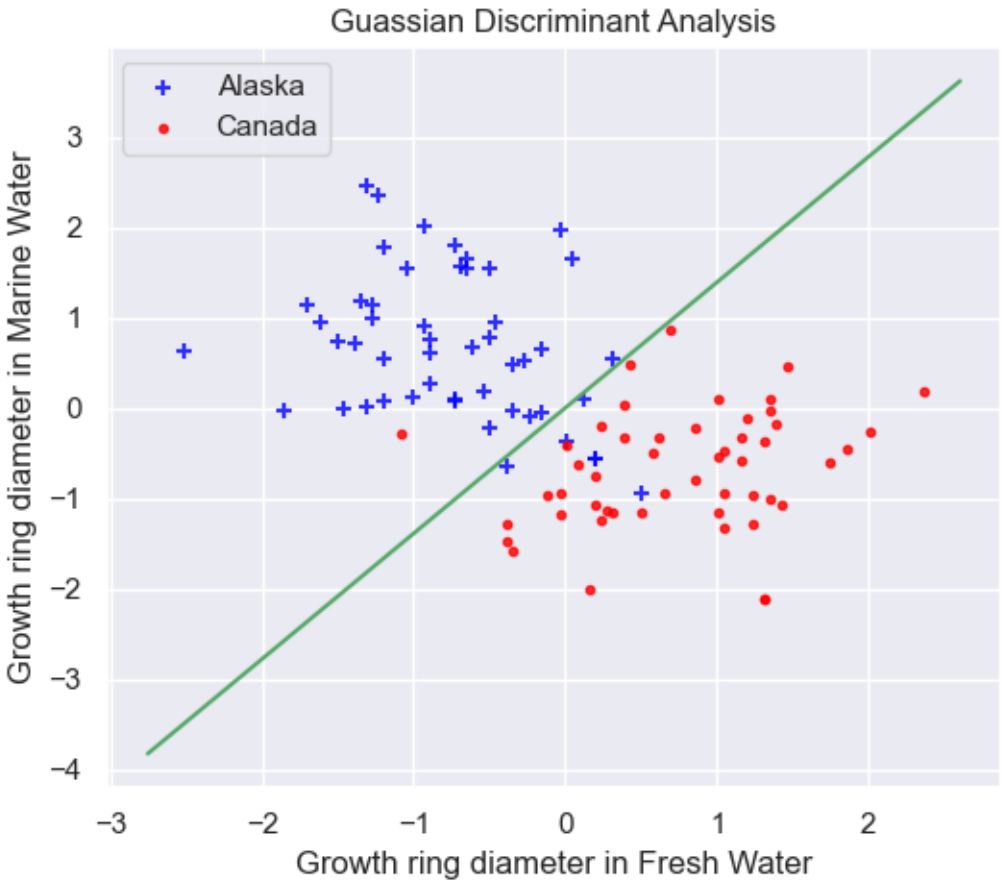
$$\mu_{Canada} = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

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

Part b



Part c



Part d

$\phi = 0.5$

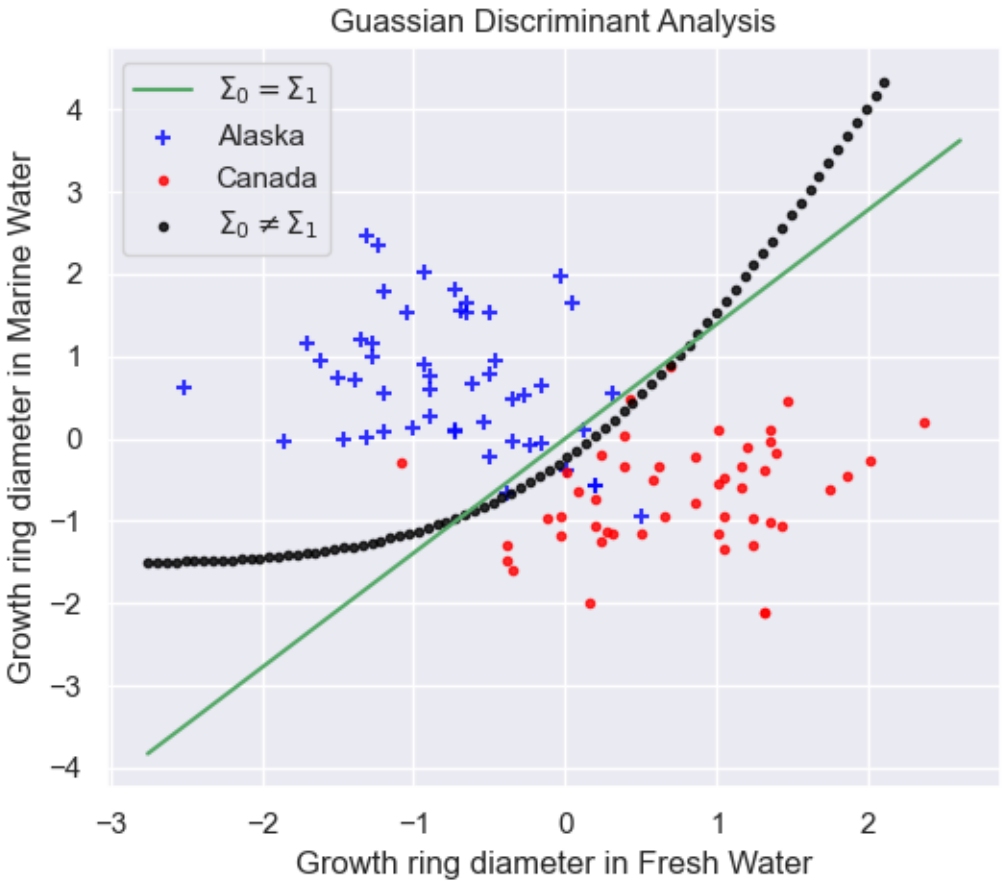
$$\mu_{Alaska} = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_{Canada} = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

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

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

Part e



Part f

From the above plot (1e), we can see that the linear decision boundary is a good and optimal **LINEAR separator, but not a good SEPARATOR overall** for our data. Overall it is not an optimal separator for the above classification problem due to the lack of covariance information between the two classes, and is equivalent to Logistic Regression. **WHEREAS**, quadratic decision boundary does a better job in separating the two classes since it uses the **covariance information** and knows how the two classes are correlated.

As the quadratic separator does well on our data, the assumption that the underlying data belongs to Gaussian distribution is valid to a great extent.