COL 703: Assignment 1

Machine Learning

Part 1: Linear Regression

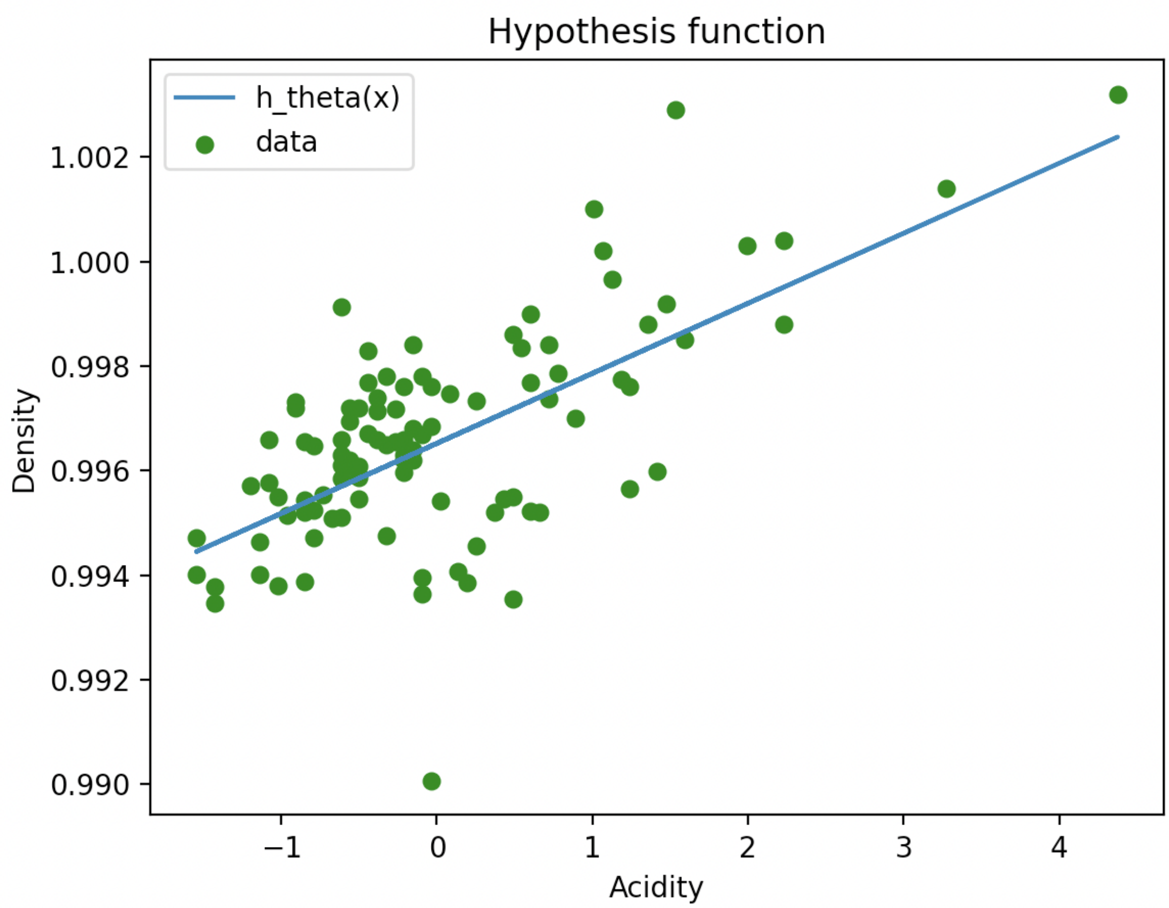
Least squares linear regression was implemented in this part. The files from which data should be obtained are taken as input from command line in the following format:

python bgd.py <input file for acidity> <input file for density> <qpart>

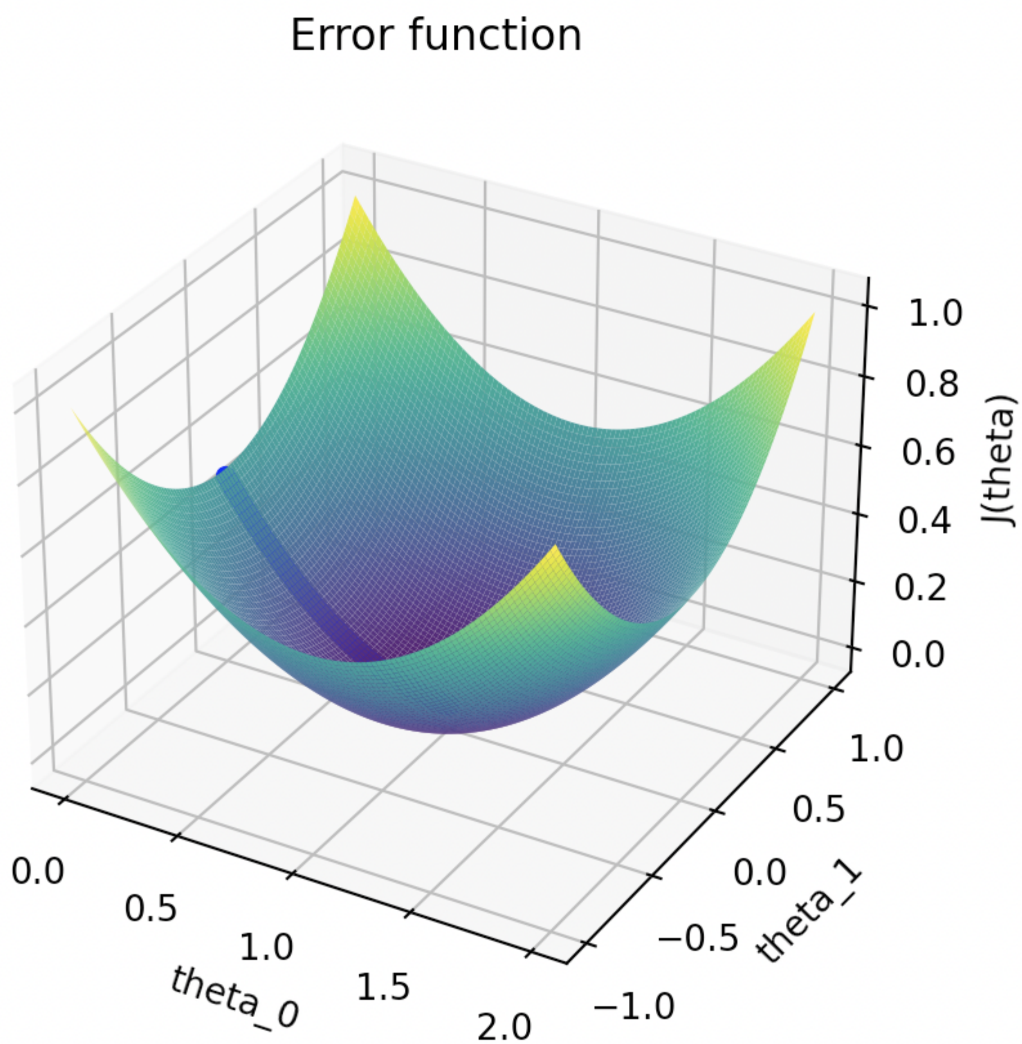
where qpart is the subpart number of the question i.e., for part b, the value of part would be 2.

1)Theta is started as a vector of all zeros, and in each iteration it is updated using a particular value of learning rate and cost function dependent on theta. The parameters used were:
Learning rate(eta): 0.01
Stopping Criteria: abs(difference in cost of two iterations) < 1e-10
Final set of parameters obtained: [0.99652102 0.00134006]

2) Hypothesis function(transpose(theta)*x) learnt by the algorithm is:

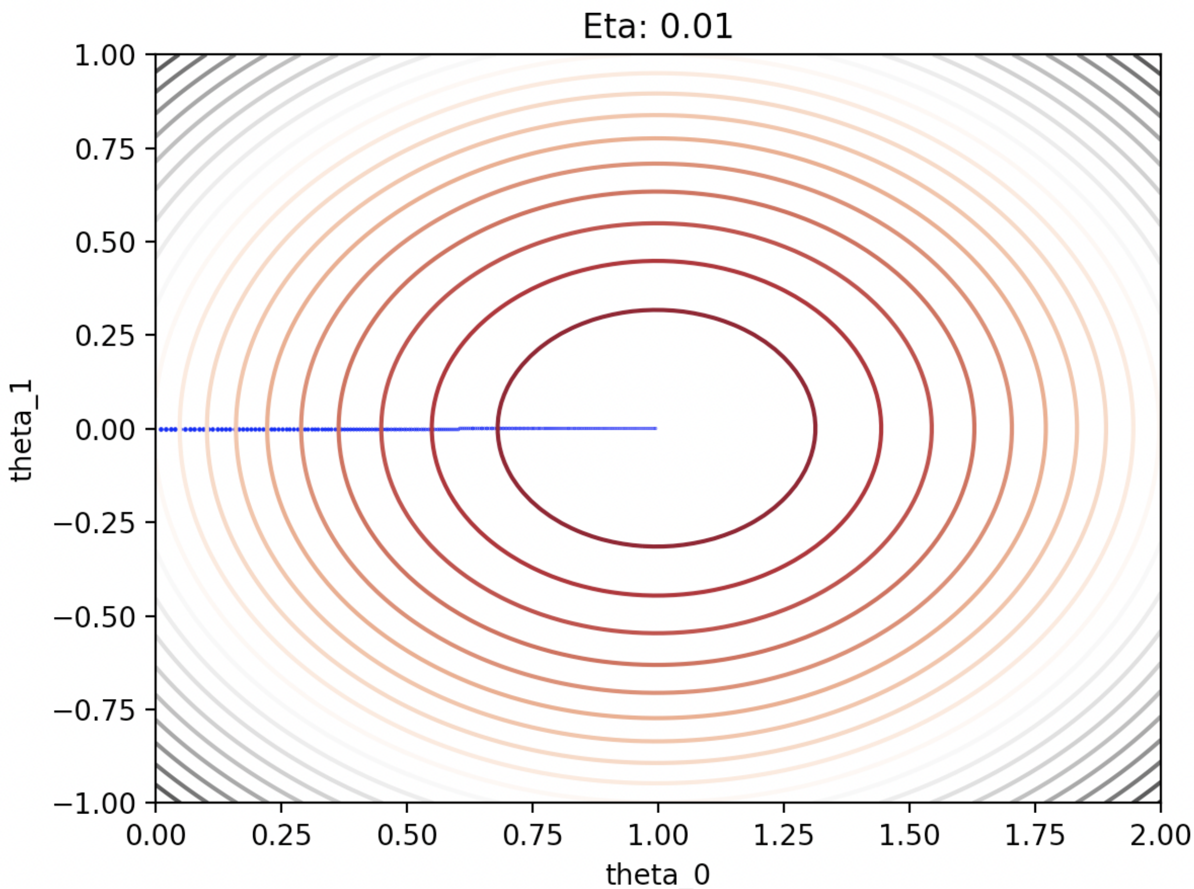


3) The mesh obtained is as follows:

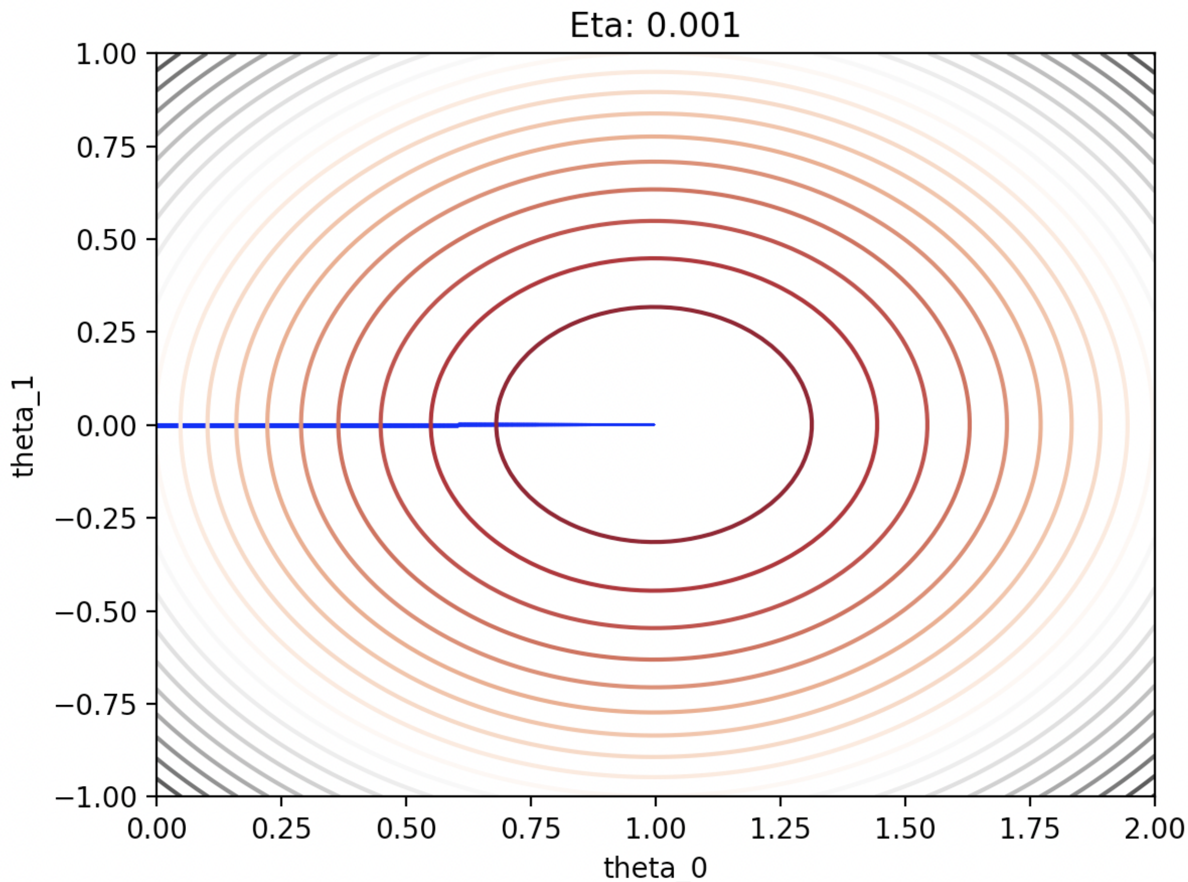


It can be seen that the cost is going down(in the blue colour) towards minima, as the regression progresses.

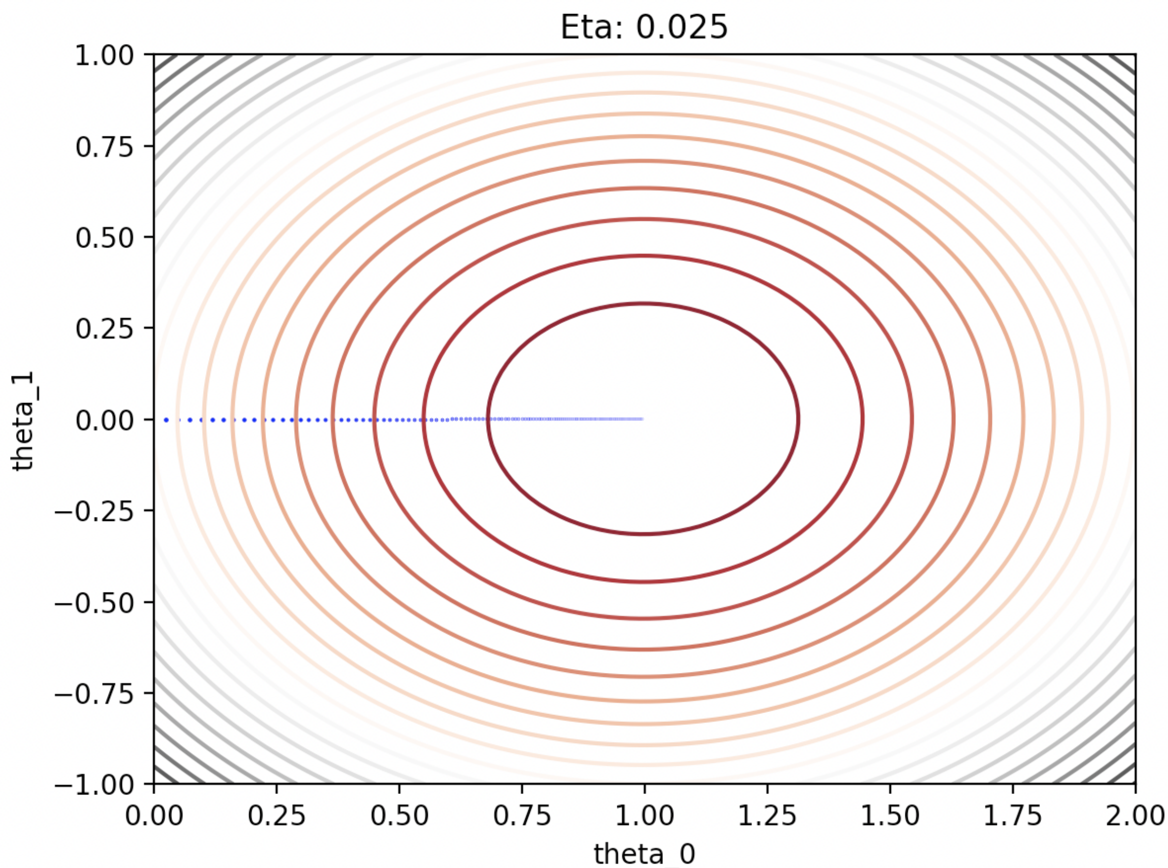
4) The contour for eta = 0.01 is as follows:

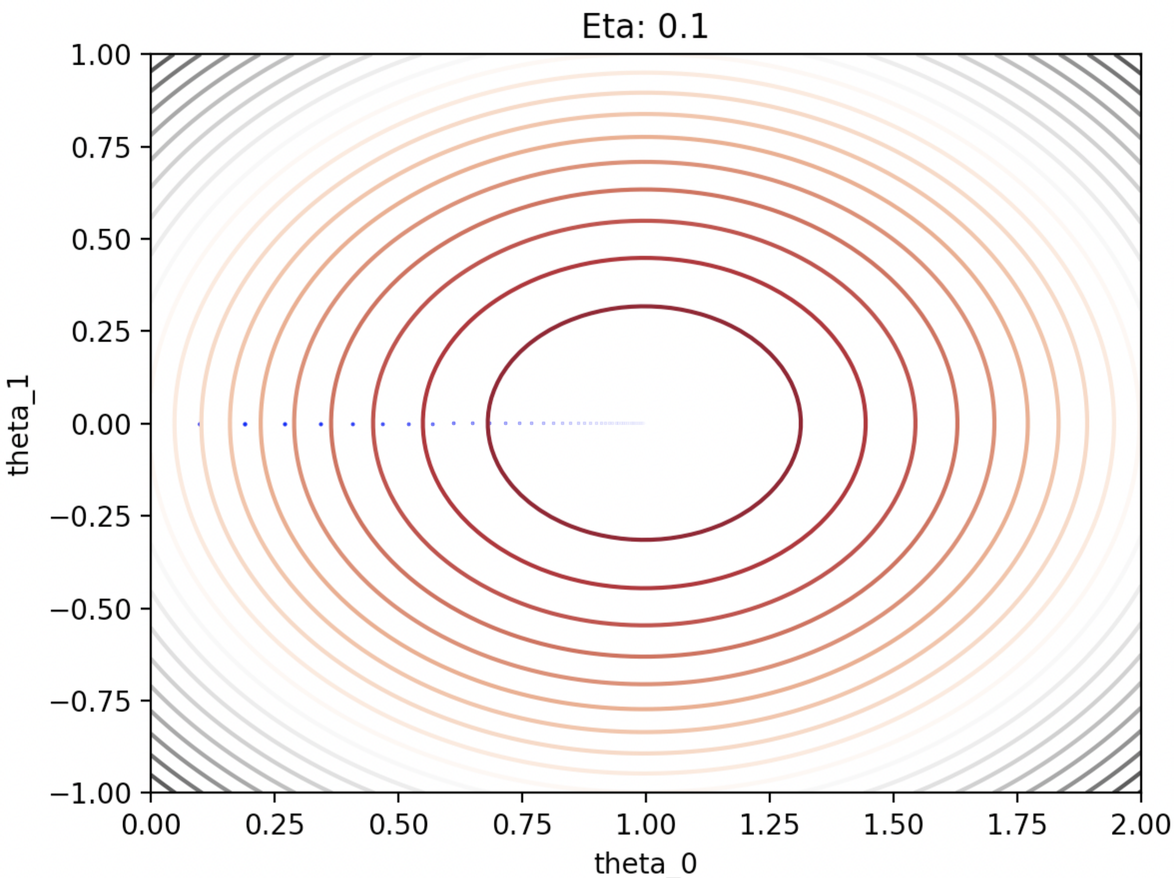


5) The required contours for different eta are attached below:



Eta: 0.001
theta_1
theta_0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Eta: 0.025
theta_1
theta_0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



Eta: 0.1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
theta_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
theta_0

Part 2: Sampling and Stochastic Gradient Descent

1) Data was sampled using np.random, and mean and standard deviation specified was obtained using np.random.normal.

   Y = transpose(theta)*x + epsilon

   This epsilon(noise) was also sampled using np.random.normal.

2) The parameters used were:
   Learning rate(eta): 0.001
   Stopping Criteria: A particular mean_size is decided for each batch size. While regression, the cost is appended to a list for mean_size number of iterations. Once mean_size entries are obtained, their average is taken and stored. Now, similarly, for the next mean_size iterations, the cost is stored and finally the two averages are compared. When the difference of this average reaches less than a threshold value(dependent on batch size), the regression is stopped and the value of learnt theta is returned.

   Criteria for varying batch sizes decided is:

Batch Size	Mean_size	Threshold
1	50000	1e-12
100	5000	1e-10
10000	500	1e-8
1000000	50	1e-6

   Following are the values of parameters learnt for each batch sizes:

Batch Size	theta_0	theta_1	theta_2
1	2.9942439	1.00513607	2.04775115
100	3.00495263	1.00495217	1.9945523
10000	2.99892581	0.99996523	2.00021722

1000000	2.99346497	1.00072432	1.99981428

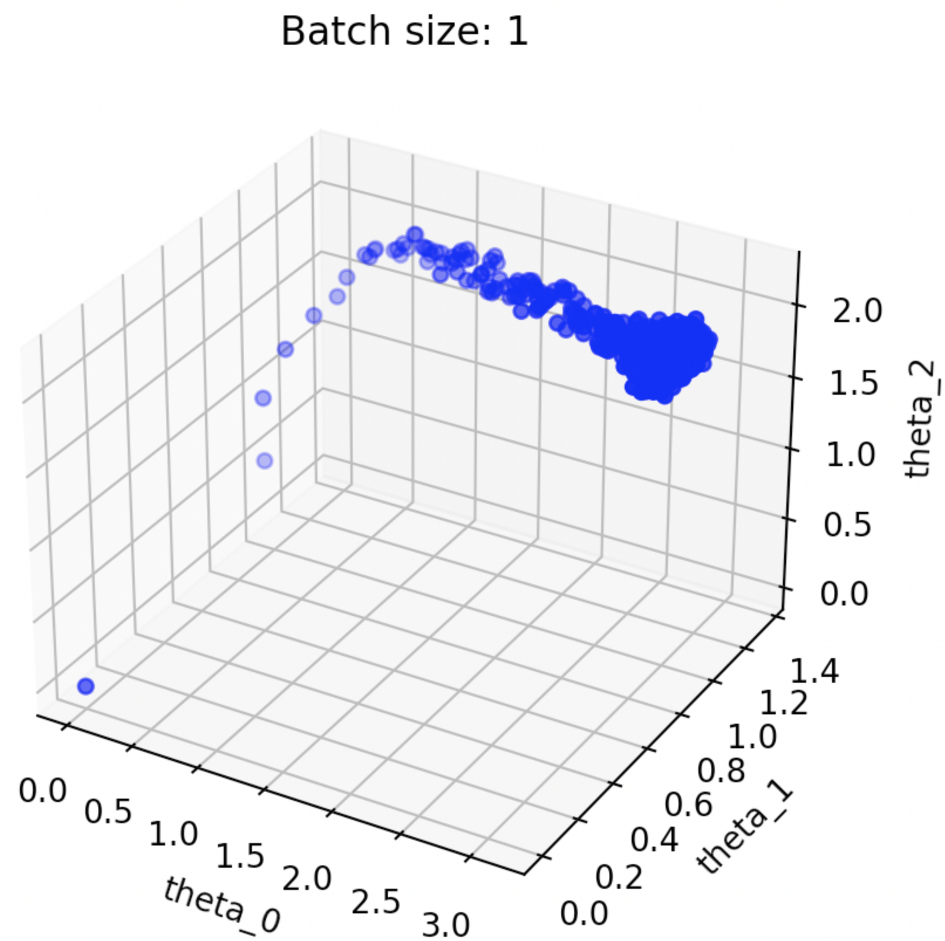
The eta, and mean_sizes are set so that we can get closest to the expected values but still there is a visible difference in the thetas for batch sizes. We can see that it is closest for the original theta, i.e., [3,1,2] for the batch sizes 100 and 10000.

3) Errors for each batch size are tabulated below:

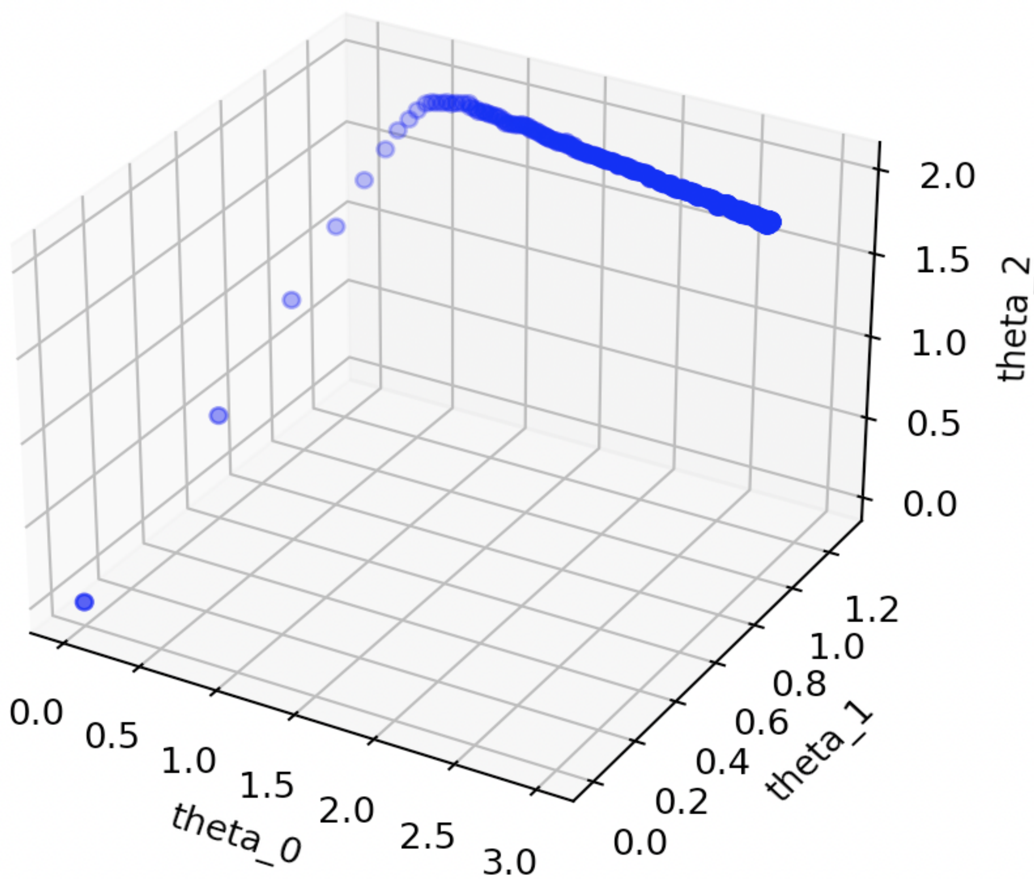
Batch Size	Errors
1	1.3997871528338197
100	0.9837688072796315
10000	0.9829430122643933
1000000	0.9829509229955081

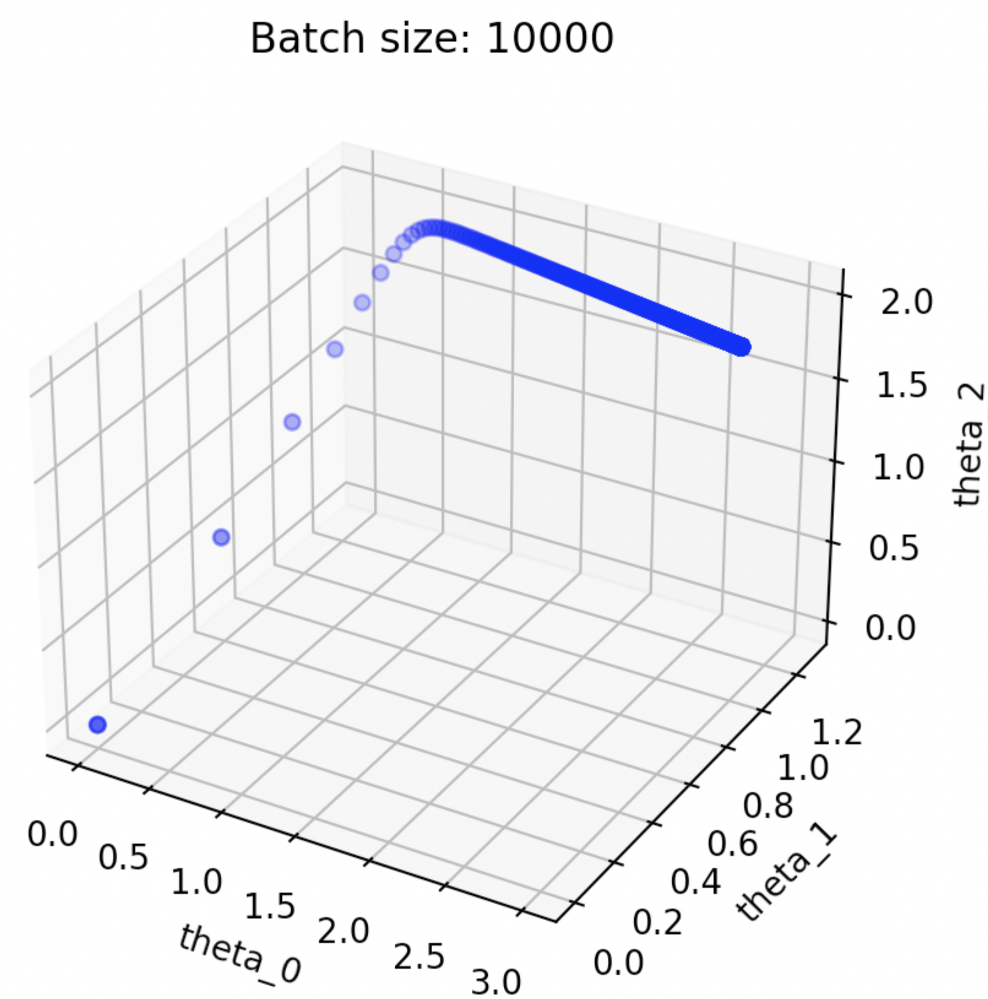
As can be seen, the error is least for batch size 100.

4)

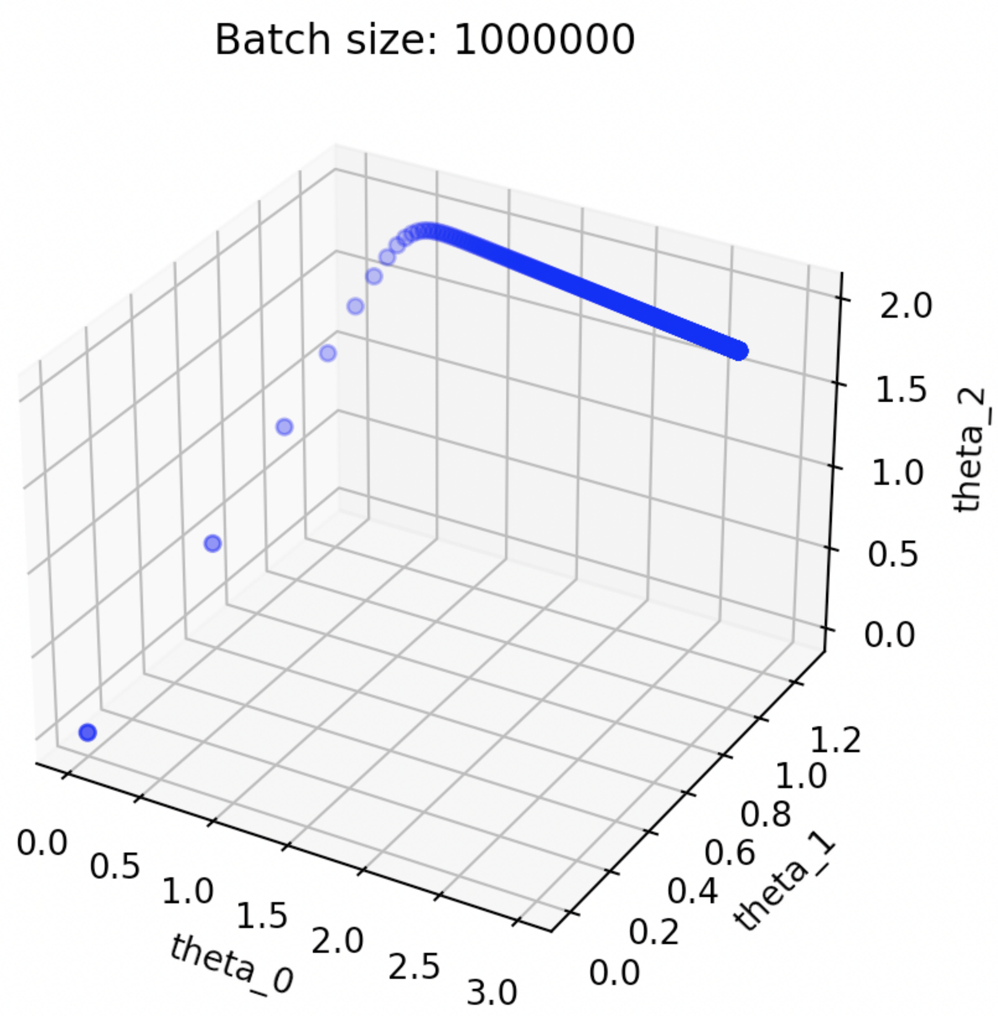

Batch size: 1
theta_0
theta_1
theta_2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.0
0.5
1.0
1.5
2.0

Batch size: 100





Batch size: 10000
theta_0
theta_1
theta_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
0.0
0.5
1.0
1.5
2.0

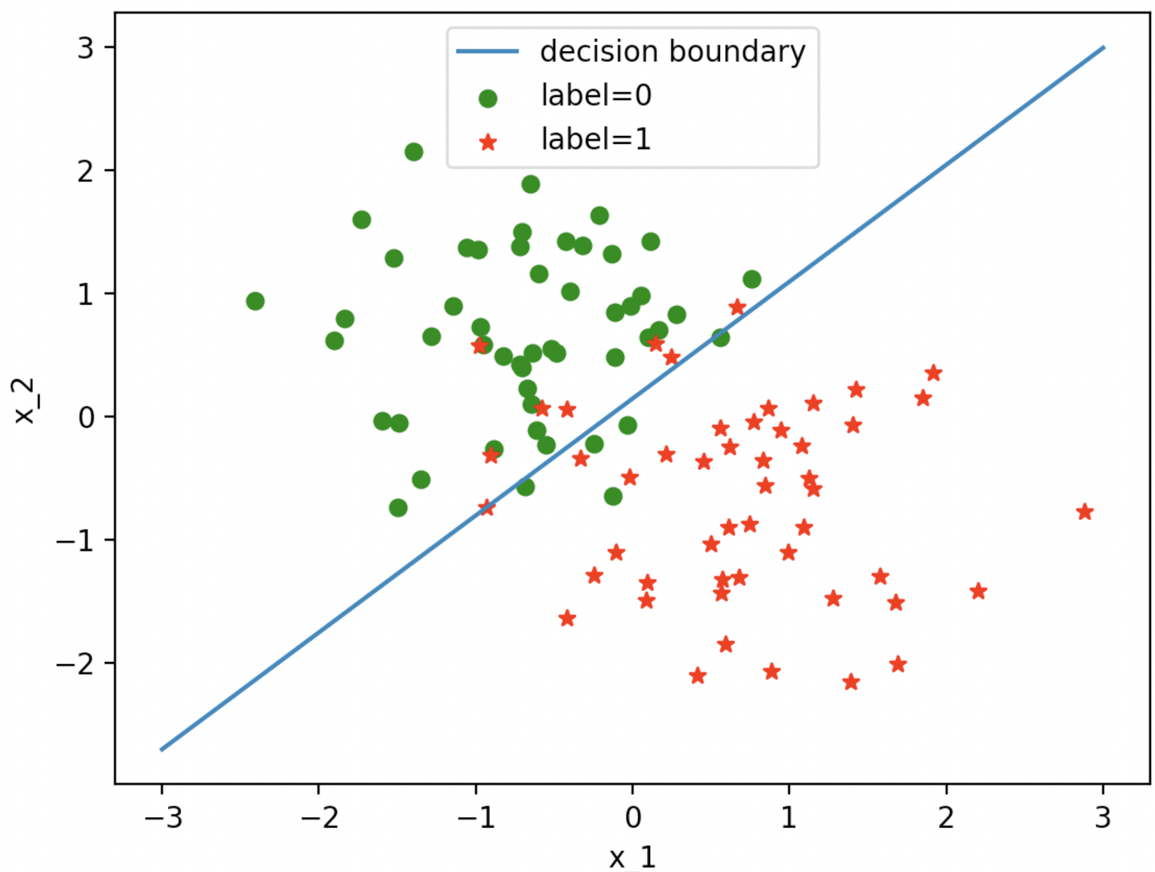


It can be observed that as the batch sizes are increased, theta is taking less random values and the curve is becoming sharper.

Part 3: Logistic Regression

1) The parameters used were:
   Stopping Criteria: abs(difference in cost of two iterations) < 3e-2
   Final set of parameters obtained: [0.39743593 2.57939935 -2.71602845]

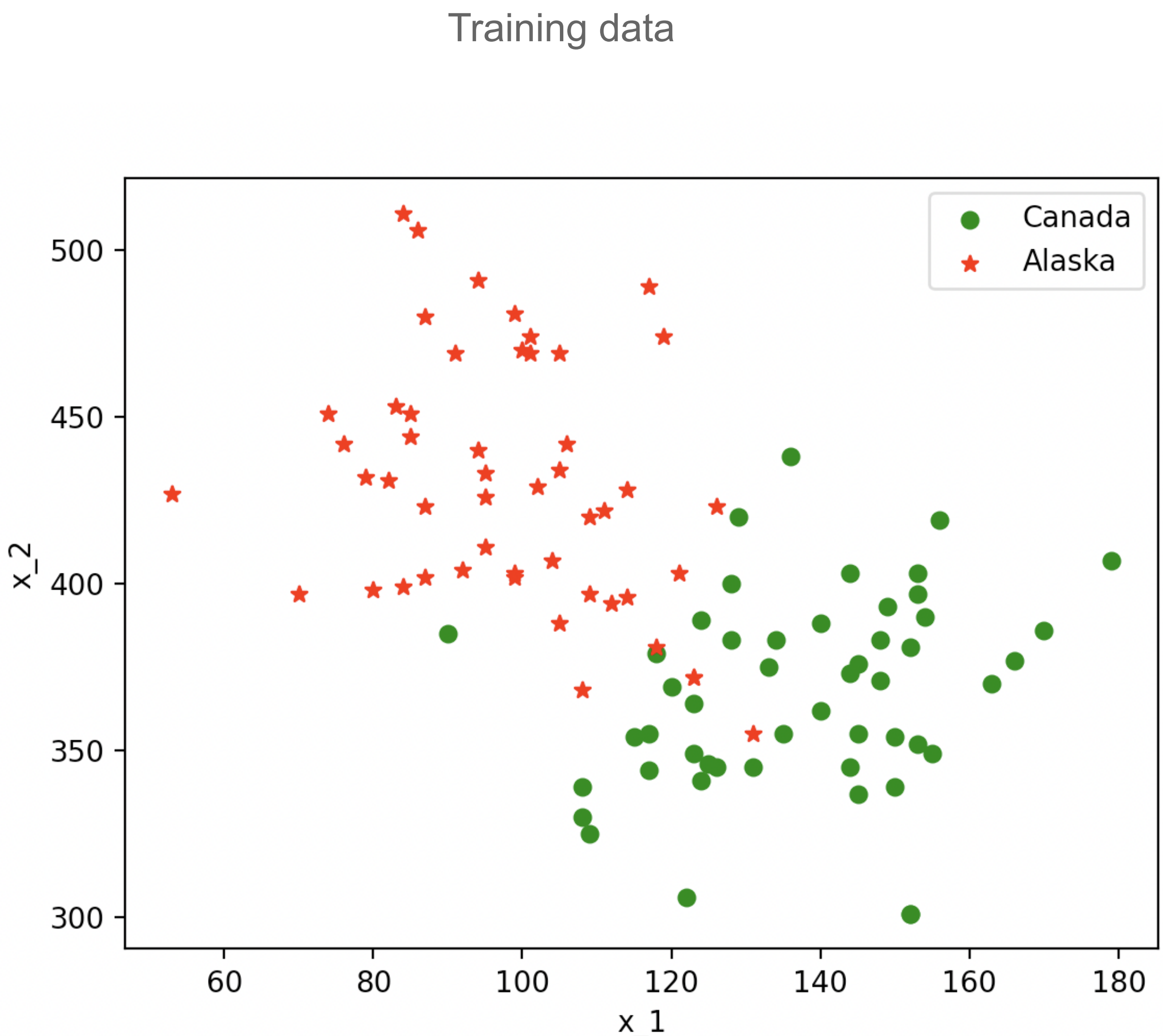
2) Equation of decision boundary : y=transpose(theta)*x=0



Part 4: Gaussian Discriminant Analysis

1) 5The results obtained are:
   phi: 0.5
   mu0: [0.75529433 -0.68509431]
   mu1: [-0.75529433 0.68509431]
   sigma: [[0.42953048 -0.02247228], [-0.02247228 0.53064579]]

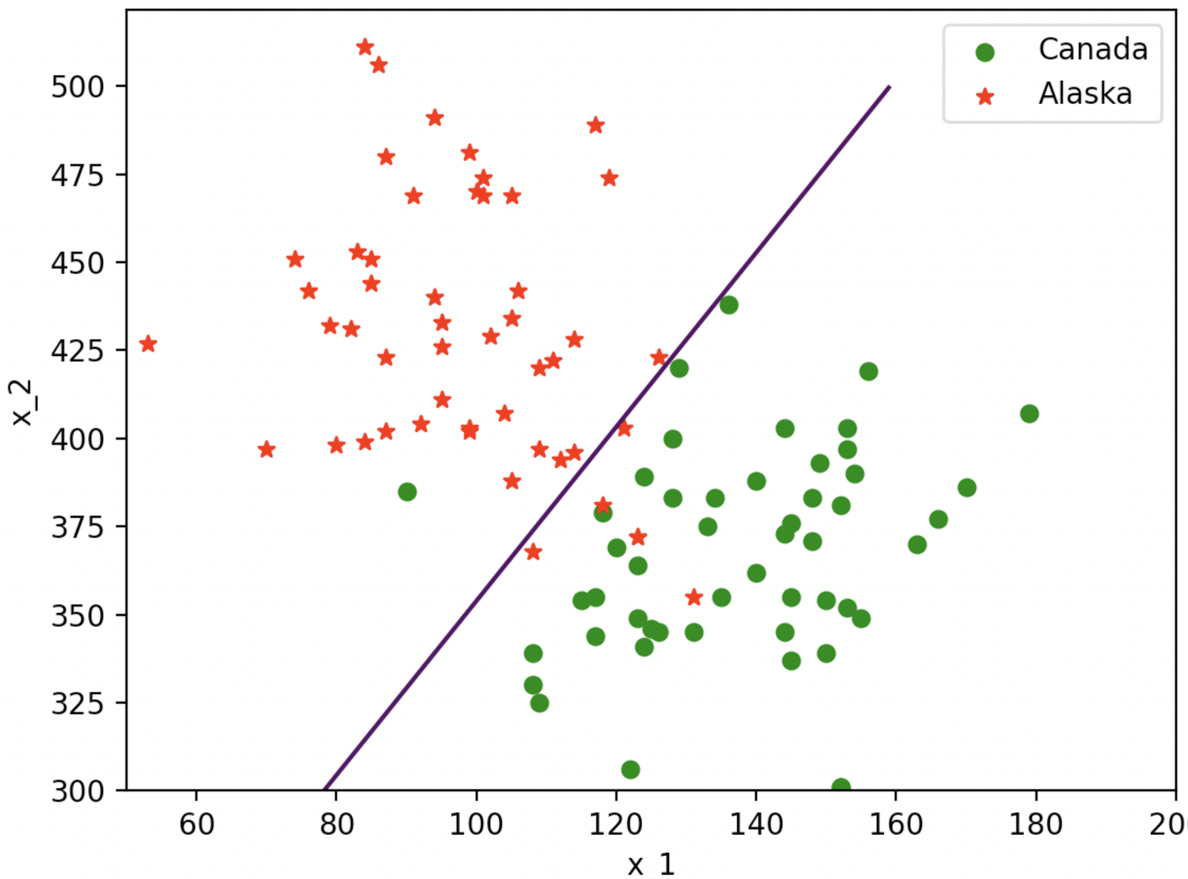
2)



3) Linear decision boundary is calculated by:

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

Linear decision boundary

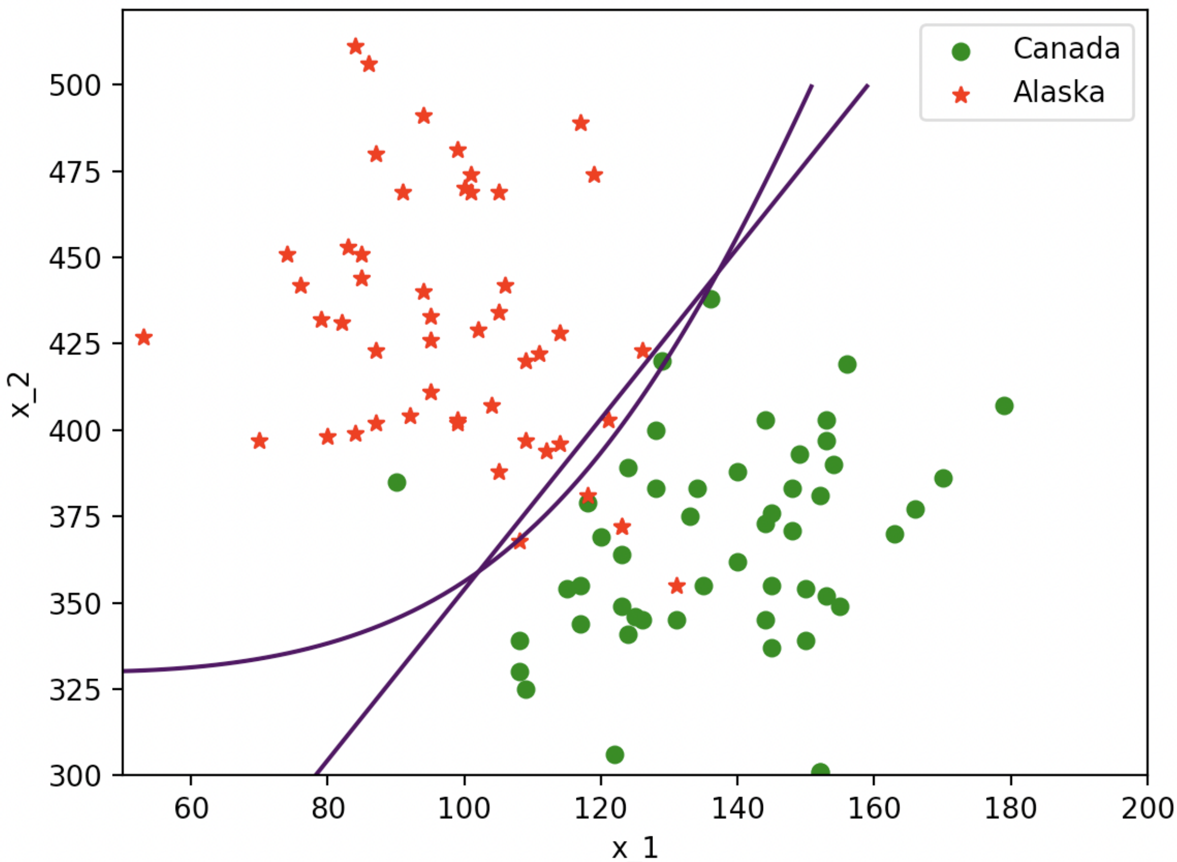


4) The parameters obtained are:
mu0: [0.75529433 -0.68509431]
mu1: [-0.75529433 0.68509431]
sigma0: [[0.47747117 0.1099206], [0.1099206 0.41355441]]
sigma1: [[0.38158978 -0.15486516], [-0.15486516 0.64773717]]

5) Quadratic decision boundary is calculated by:

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

Quadratic decision boundary



6) It can be seen that the quadratic separation is better at separating the two clusters, as in the linear one there were 5 Alaska points in Canada, but the quadratic separation has successfully segregated 2 of those in Alaska without compromising other points.