COL774: ASSIGNMENT – 1

SUBMITTED BY: SAMIDHA VERMA (2020CSY7575)

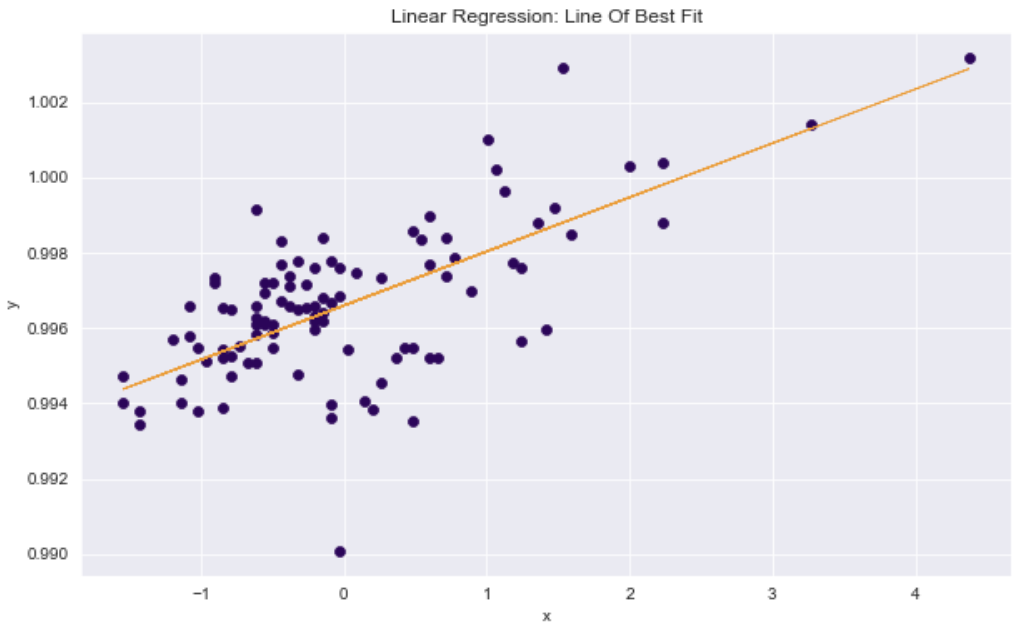
Instructions for running the codes:

- Execute statement: (i) cd <directory_name>
  (ii) python <filename>.py
- Directory names: q1, q2, q3, q4 corresponding to each question.
- Filenames are as follows:
  - Q1 : Linear_Regression.py
  - Q2 : Sampling_and_SGD.py
  - Q3 : Logistic_Regression.py
  - Q4 : Gaussian_Discriminant_Analysis.py
- The results would be printed on the terminal window and the plots would be stored in the respective directory.
- It is assumed that the data will be in subfolders named q1/q2/q3/q4, in a folder called 'Data' placed inside the top-level directory '2020csy7575_samidha_verma'.

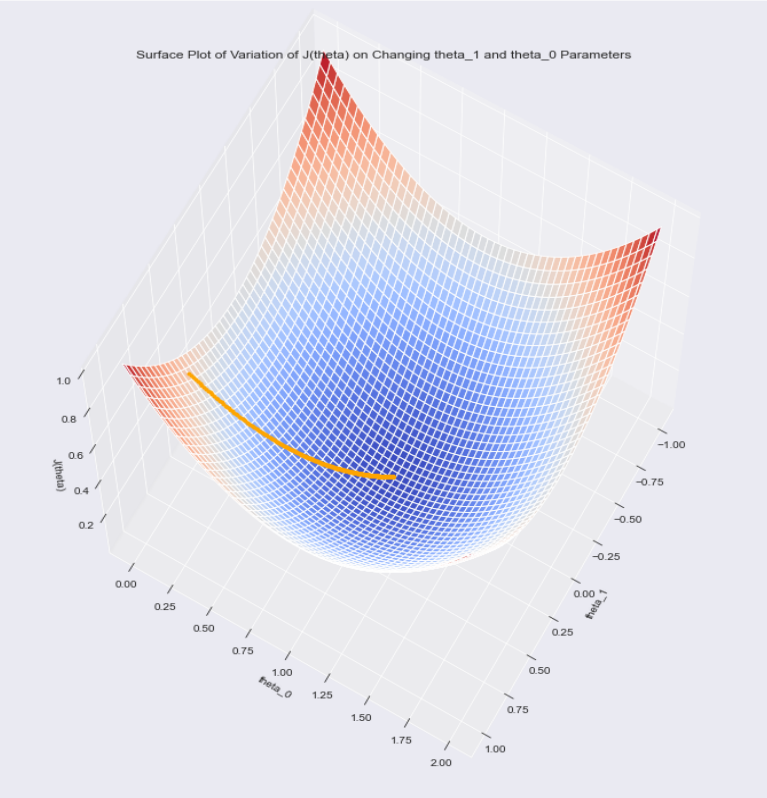
1. Linear Regression

(a) Learning rate : 0.01
Stopping criteria : abs(J(theta_t) – J(theta_(t+1))) <= delta, where delta = $10^{(-10)}$
Final parameters [theta_1, theta_0] : [0.00143684, 0.99660025]

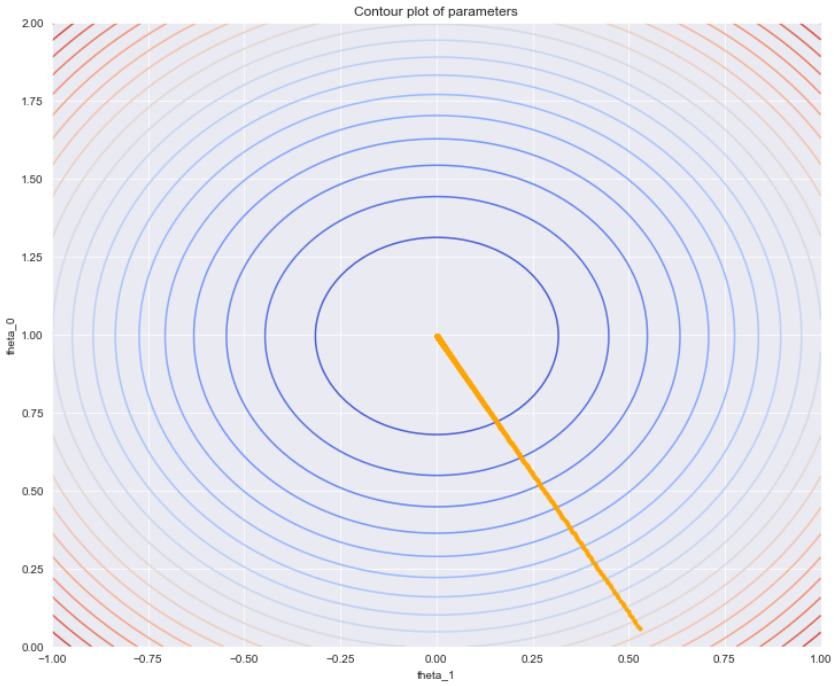
(b) Line of best fit learned by linear regression



(c) 3 dimensional mesh plot of J(theta) vs parameters

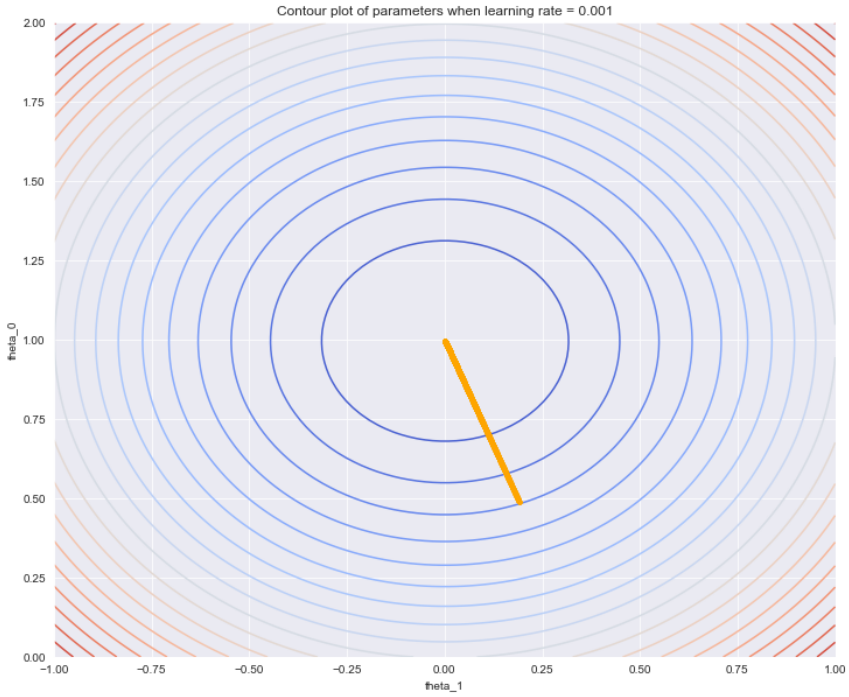


(d) Contour plot of the error function at each iteration of the gradient descent.

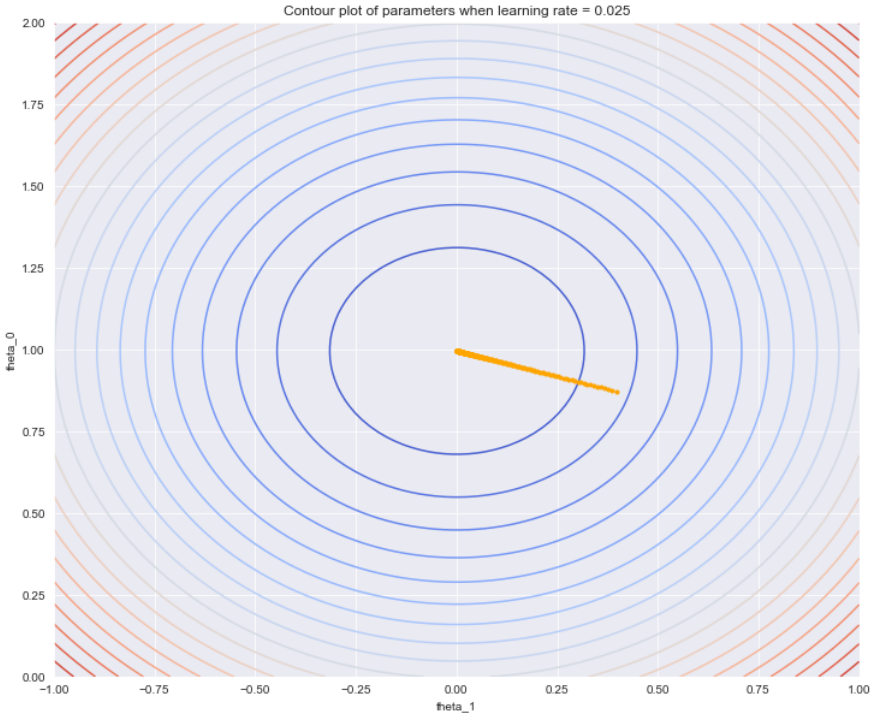


(e) Contour plot of the error function at each iteration of the gradient descent w.r.t different learning rates:

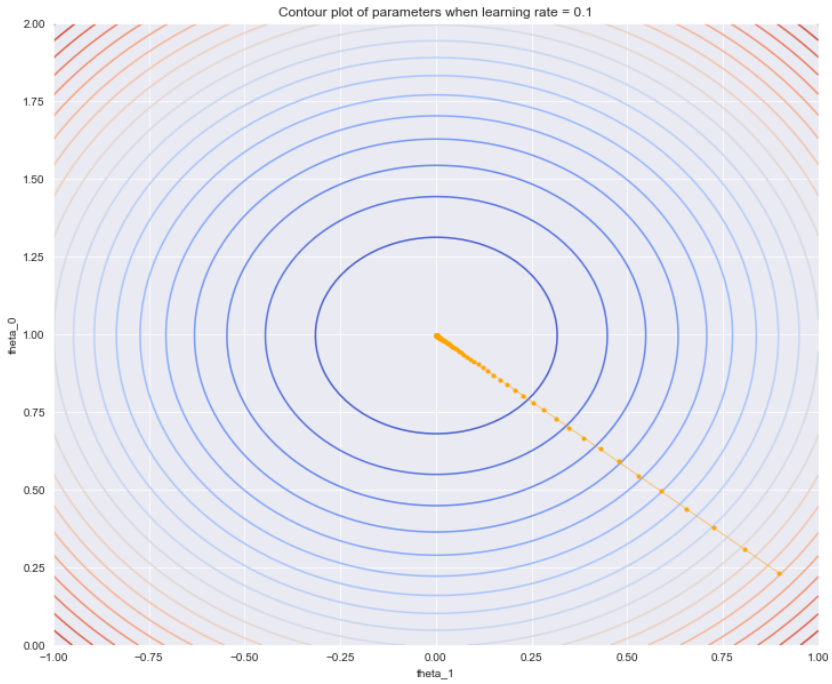
- Learning rate = 0.001



- Learning rate = 0.025



- Learning rate = 0.1



Observation: From the above plots I observed the importance of selecting the correct learning rate while performing gradient descent. It was observed that when the learning rate is sufficiently large, gradient descent takes a smaller number of epochs to reach the minima as it takes larger steps in the direction perpendicular to the contour. This can be concluded from the density of the lines (larger number of points and close together for small learning rates) in plots above. However, if we take a very small learning rate, we may not reach the minima within a certain number of maximum epochs or may take larger number of passes through the data to reach the minima. This can be very inefficient in case the dataset is very large. Also, if we take a very large learning rate then, the cost function diverges. Thus, we need to hit the sweet spot between sufficiently fast convergence by taking a sufficiently large learning rate, which is also sufficiently small in the sense that the cost function should not diverge.

2. Sampling and Stochastic Gradient Descent

(a) The points were sampled according to the instructions, in sample_data() function of the code.

(b) learning rate = 0.001 (given), Final parameters are in the form of a 3*1 vector [[theta_2],[theta_1],[theta_0]]

Batch size	Final parameters	k (#batches, the score is averaged over)	delta(stopping criteria)
1	[[1.98845615] [1.01186702] [3.02385305]]	**300**	**10**(-3)**
100	[[1.99795692] [1.00061089] [2.99760572]]	**150**	**10**(-4)**
10000	[[1.99941403] [1.00235808] [2.9886529]]	**100**	**10**(-6)**
1000000	[[1.999379] [1.0024313] [2.98828027]]	**1**	**10**(-8)**

TABLE 2.1

Convergence criteria:
abs(avg cost for k batches – avg cost for next k batches)) <= delta, where delta = 10**(-10), cost = ½ * MSE

The above table shows the values of k and delta taken in each case, to hit minima within an appropriate amount of time. To verify the values, I plotted average cost of k batches vs timesteps, and decided the values from the observations. I observed for smaller batch size, I needed a larger value of k to smoothen out the plot, and a larger delta, since the fluctuation of costs for smaller batch size was more to converge within a feasible time. The plots can be found in ‘q2’ directory.

(c)

Batch Size	L2 Norm (Original theta – Learned theta)	Time taken for convergence(in sec)	# iterations taken for convergence (not equal to epochs)
1	0.029035407638823982	0.5051999092102051	17700
100	0.0032062386204201876	0.8713061809539795	38400
10000	0.011604337919822224	1.8293178081512451	20400
1000000	0.011985362103491878	303.28630399703 98	20269

TABLE 2.2

Batch Size (Model w.r.t to batch size used)	MSE on test dataset [formula: (1/2m)*$\sum_{i=1}^{m}(y - \text{np.dot(theta.T, x)}\text{^}2)$]
1	0.99873811
100	0.98343432
10000	0.98327846
1000000	0.98330276
Original parameters	0.98294692

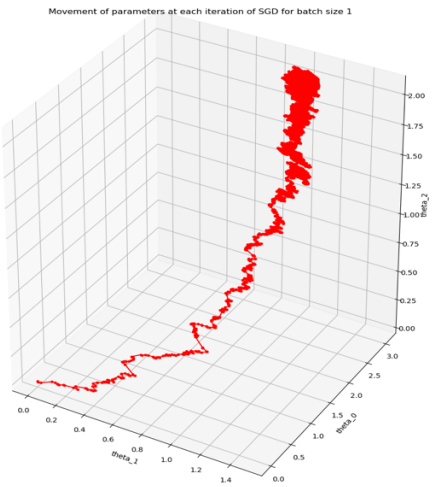
TABLE 2.3

Observation:

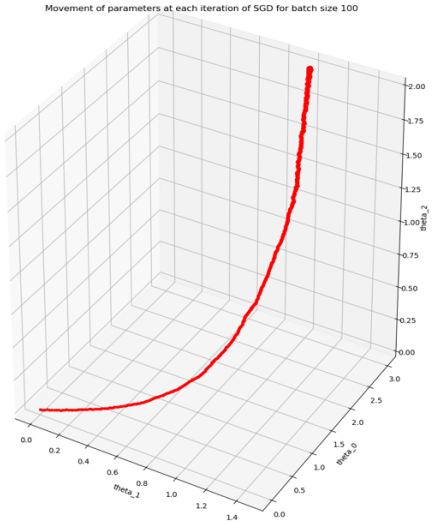
- From table 2.1 we can observe that running SGD with different batch sizes results in almost the same final parameters. Therefore, the fact that SGD gives us the minima in expectation holds true in practice.
- From 'L2 Norm' column of table 2.2 we observe that the estimated parameters by various models are very close to the actual parameters.
- From 'Time taken for convergence' column of table 2.2 we observe that SGD converges relatively fast for smaller batch size given the stopping criteria is appropriate. This is so, because for smaller batch size, SGD takes larger number of gradient descent steps. For batch size 1, SGD took 17700 iterations, i.e. it looked at the 17700 examples out of 1 million examples and converged. We can also observe that using the entire data of 1 million data points for gradient descent takes a lot of time to reach the minima. A similar statement can be said for the number of iterations.
- From the 'L2 Norm' column we can also observe that the closest SGD got to minima was in the case of 100 batch size. Therefore, we can sense a tradeoff between the batch size, convergence, and convergence rate.
- From table 2.3 we can observe that almost all models give a similar mean squared error on the test data. This further solidifies the hypothesis that SGD gives faster convergence and good performance by looking only at a few examples at a time.

(d) Movement of theta in 3D parameter space w.r.t batch size:

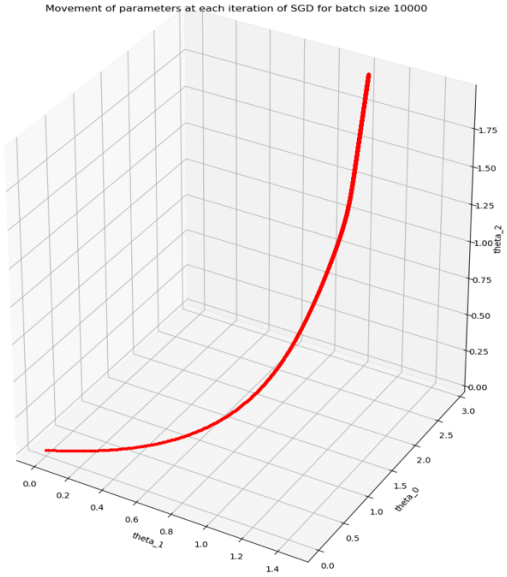
- **Batch Size: 1**



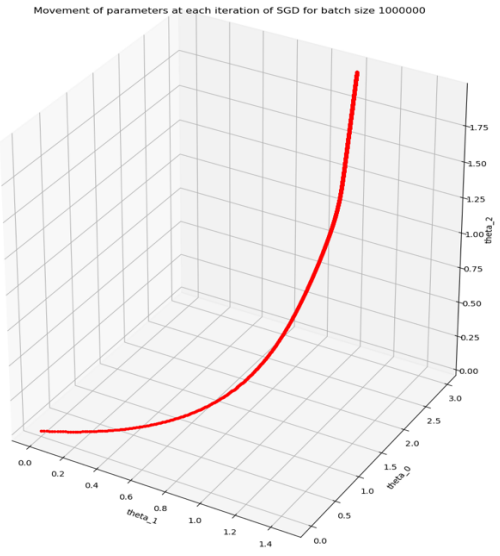
- **Batch Size: 100**



- **Batch Size: 10000**



- **Batch Size: 1000000**



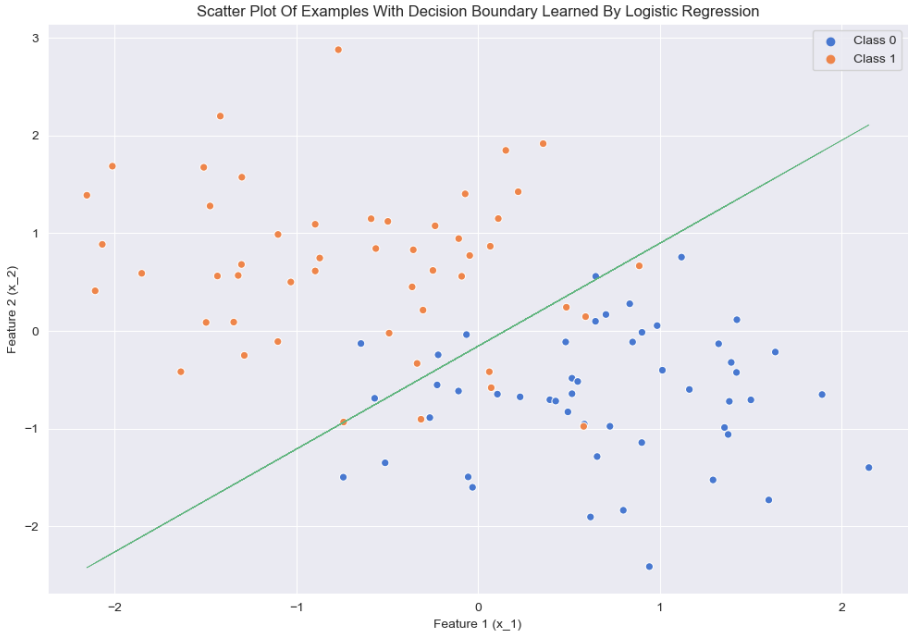
Observation: For batch size of 1 we observe a zig zag movement of parameters due to updating parameters after seeing a single example only. However, it does finally converge near the minima. As we increase the batch size, we observe the movement of parameters becomes more and more smooth and each gradient step is in the direction of the minima with increasingly less zig zag behavior.

3. Logistic Regression

In this question the cost function taken is negative log likelihood.

(a) The coefficients θ learned by the model are (theta_2, theta_1, theta_0):
[2.58854155, -2.72558182, 0.40125115]
Number of iterations taken for convergence: 38

(b) Here I have treated 0^{th} column of input features as x_2 and 1^{st} column of input features as x_1.



4. Gaussian Discriminant Analysis

Here I have treated 0^{th} column of input features as x_1 and 1^{st} column of input features as x_2.

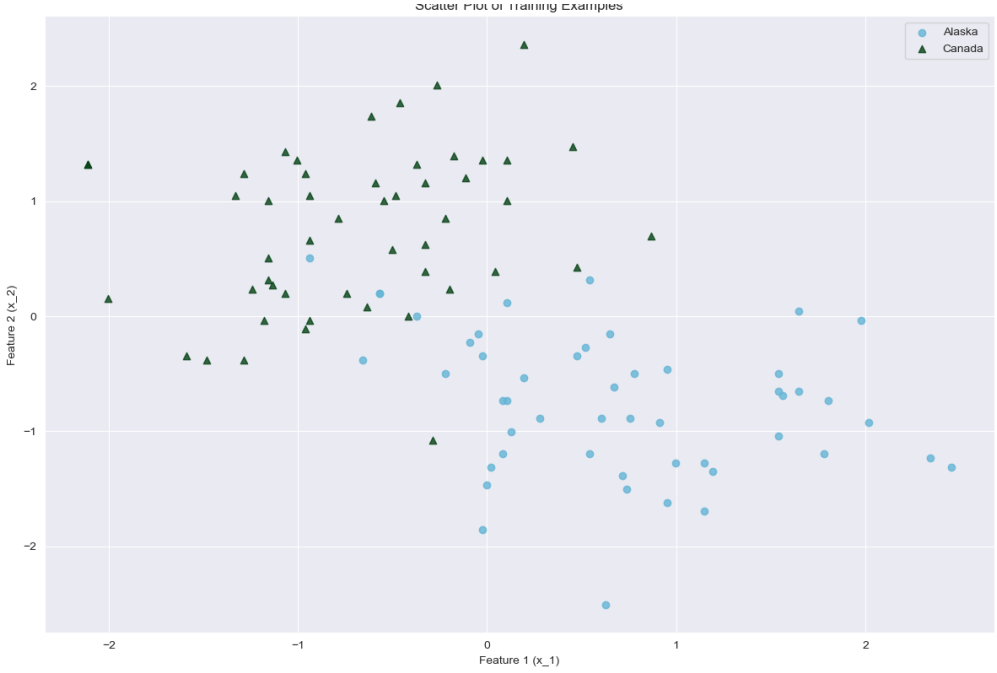
(a) phi : 0.5

mu_0 : $[-0.75529433 \quad 0.68509431]$

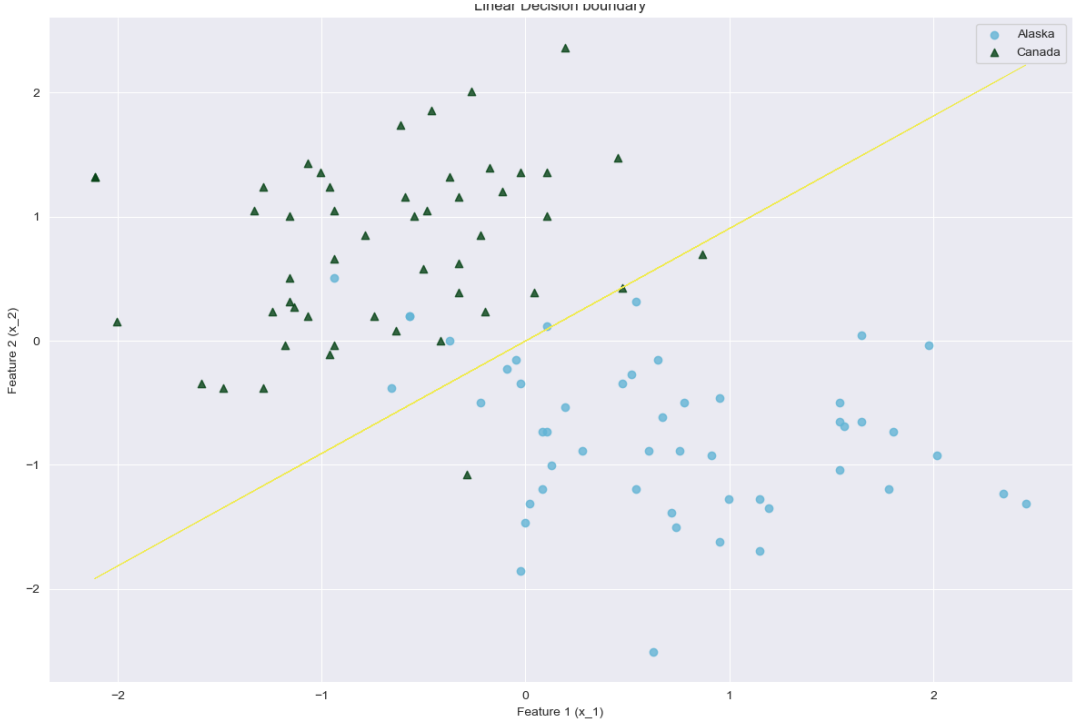
mu_1 : $[0.75529433 \quad -0.68509431]$

sigma : $\begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$

(b) Plot of training data



(c) Linear decision boundary learned by GDA



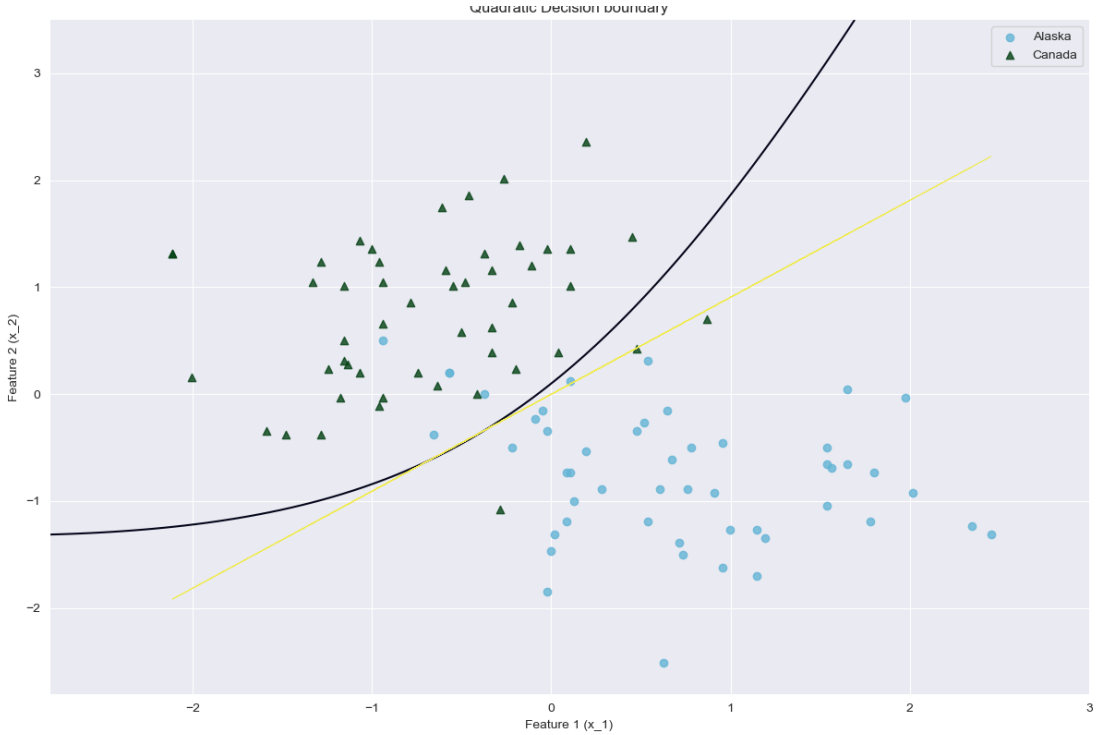
(d) mu_0 : $[-0.75529433 \quad 0.68509431$

mu_1 : $[0.75529433 \quad -0.68509431]$

sigma_0 : $\begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$

sigma_1 : $\begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$

(e) Quadratic decision boundary learned by GDA



(f) The misclassification in case of linear boundary was (assuming classification at boundary points is Alaska):
Class Alaska : 4
Class Canada : 3

The misclassification in case of linear boundary was (assuming classification at boundary points is Alaska):
Class Alaska : 4
Class Canada : 3

Although the number of misclassified points remain the same under the assumption, that the points on the boundary are being classified to class Alaska. However, we can observe, that the points representing Alaskan fishes that lay on the linear boundary, are more likely to be classified correctly in case of a quadratic boundary since the margin from the decision boundary is larger. Since misclassification rate is only 0.7% in case of both decision and quadratic boundary, therefore the assumption that the features follow a gaussian distribution seems valid.