COL774, Assignment 1 Report

Abhishek Burnwal, 2017CS50401

Question 1: Linear Regression

In the first problem, I used the batch gradient descent to minimize the error function. I have tried to use numpy arrays for calculations almost everywhere for faster calculations.

1. J(theta) = (1/2m) * (theta.T*X - Y)* (theta.T*X - Y).T where A.T = transpose of A, m = size of dataset
2. theta = theta + 1/m*rate*(Y - theta.T*X)*(X.T)
3. Stopping condition was difference in values of cost(old and new), abs(old-new)<epsilon, Where epsilon is stopping parameter

1. a) Parameters obtained are:

   theta parameters: [[0.99647964] [0.00134001]]
   rate: 0.005 stopping condition: 1e-10
   final cost: 1.2046546443061066e-06
   Number of iterations: 1770

1. b) Data and Hypothesis plot, hypothesis = theta.T*X

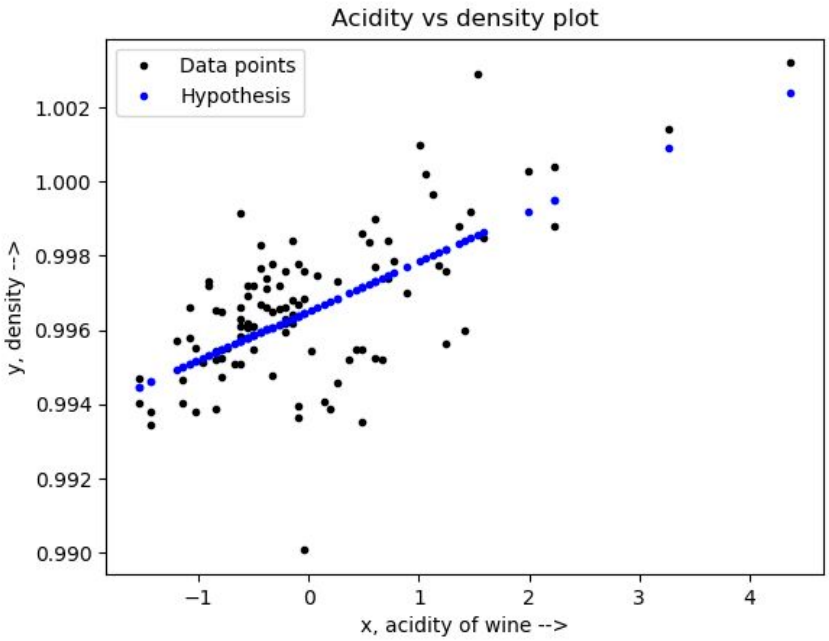


Fig1 - Plot of Data points and hypothesis learnt

1. c) 3-D mesh with gradient descent, cost function was plotted for 100*100 pairs of (theta0, theta1). Gradient Descent of the cost function has also been shown, in red.

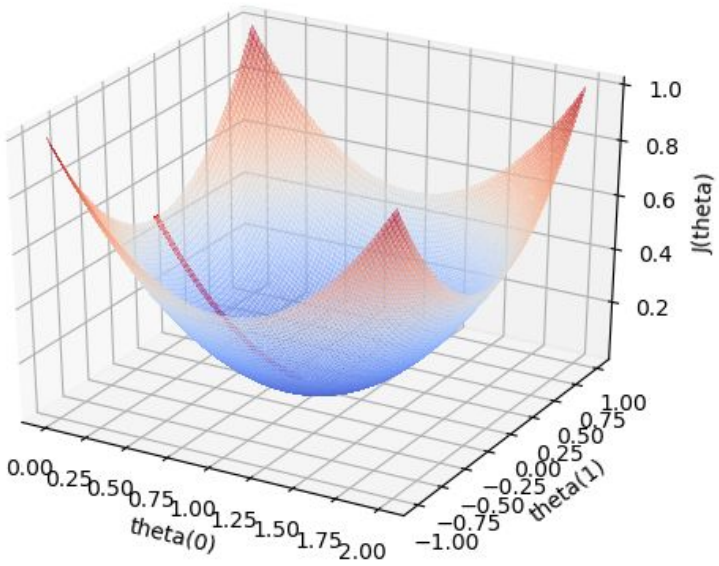


Fig 2 - Plotting 3-D mesh with Gradient Descent of cost Function(J)

1. d) Plotting contours of Cost Function, shown in red

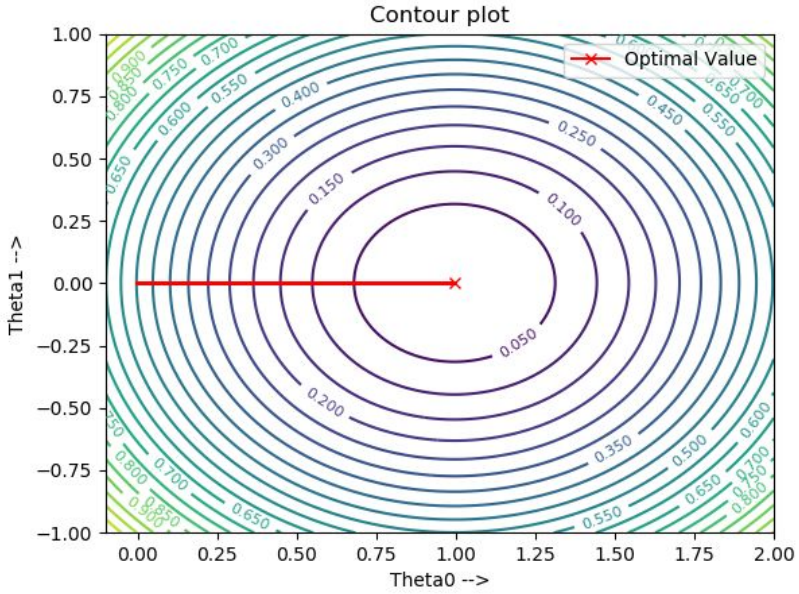


Fig 3: Contours of the Cost function with the red line as GD of J (at rate=0.0001)

1. e) Repeat above parts for rate = 0.001, 0.025, 0.1

Value of rate	Theta	Number of iterations	Final cost
0.001	[[0.9963043] [0.00133977]]	8054	1.24465338928501e-06
0.025	[[0.99655882] [0.00134011]]	384	1.1966672493924672e-06
0.1	[[0.9966172 0.00134019]]	101	1.1951401850562593e-06

For rate = 0.001:
theta parameters: [[0.9963043] [0.00133977]]
rate: 0.001
stopping condition: 1e-10
final cost: 1.24465338928501e-06
Number of iterations: 8054

For rate = 0.025:
theta parameters: [[0.99655882] [0.00134011]]
rate: 0.025
stopping condition: 1e-10
final cost: 1.1966672493924672e-06
Number of iterations: 384

For rate = 0.1:
theta parameters: [[0.99659363] [0.00134016]]
rate: 0.1
stopping condition: 1e-10
final cost: 1.1951401850562593e-06
Number of iterations: 101

The number of iterations required to reach convergence decreases drastically with increase in rate. This is because the rate essentially acts as the size of the step of gradient descent. Larger the rate, longer the step and therefore faster the convergence is reached. It can be observed in the animation of contour plots of two values.
Also we observe that beyond a certain point, the cost function doesn't decrease despite the decrease in rate from 0.1 to 0.001. On the contrary, it has increased by a very slight amount. This could be explained with the help of oscillations that occur when cost approaches the

optimum point. Thus, true optimum can only be reached if we slowly decrease rate as we approach optimum. That is when cost approaches optimum, rate approaches zero.

Question 2: Stochastic Gradient Process:

In this question, I implemented Stochastic Gradient Descent for predicting the values of the parameters of the hypothesis, with respect to varying values of r(mini-batch size) and fixed Rate = 0.001

2. a) First, we generated a sample data set the size of one million with normal distribution parameters.

2. b) Then we ran SGD on it with different stopping criteria for different values of r, without normalizing X, for different values of r. Convergence condition is when average value of gradient descent (over m//r) loops is less than the epsilon(convergence criteria).

r	Convergence criteria	Number of epochs	Final parameters
1	0.00000001	1	[[3.04604067] [1.02073313] [2.01204444]]
100	0.00000001	7	[[2.9988188] [1.0000218] [2.00109634]]
10000	0.00000001	245	[[2.99377609] [1.00044951] [1.99955971]]
1000000	1e-11	15929	[[2.9673217] [1.00752327] [1.99898988]]

2. c)
Different values of r converge to almost the same values of theta. However, the number of iterations taken to reach convergence vary a lot.
As r increases the number of iterations of epochs to achieve convergence also increases.

However, the time taken for one iteration is less for higher values of r. For eg. for r=1, the time for one iteration is very high, but it converges after one iteration.

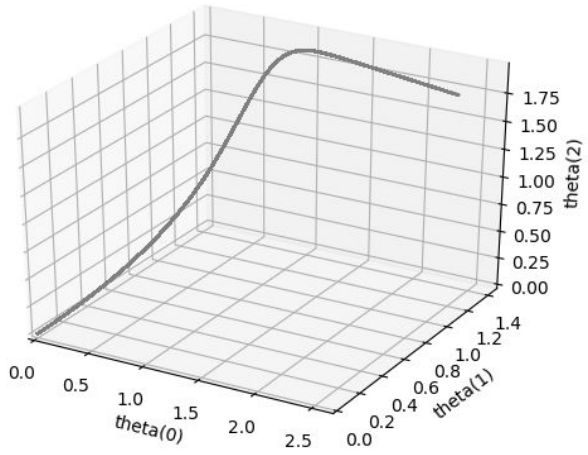
This difference is because when r is low, the number of mini-batches is very high (m/r), so for an epoch you have to do (m/r) loops which is a slow process. But with more number of loops, you also get a good precision with each epoch. Thus fewer epochs are required for less r but more time is required for each epoch.
Similarly, with high r, you have less number of mini-batches, so the number of loops is low to complete an epoch, but the accuracy is limited per epoch, so SGD runs on multiple epochs. For r=1 million, we had to change the convergence criteria because the graph would very early converge with such small convergence and not give proper results.

The costs are given below.
The costs are almost equal to that calculated with the original parameters. This tells us that SGD is a good model to work with and with batch sizes of 100, 10000, it gives the optimum results in the best time.
For any r, time taken = number of epochs * time per epoch

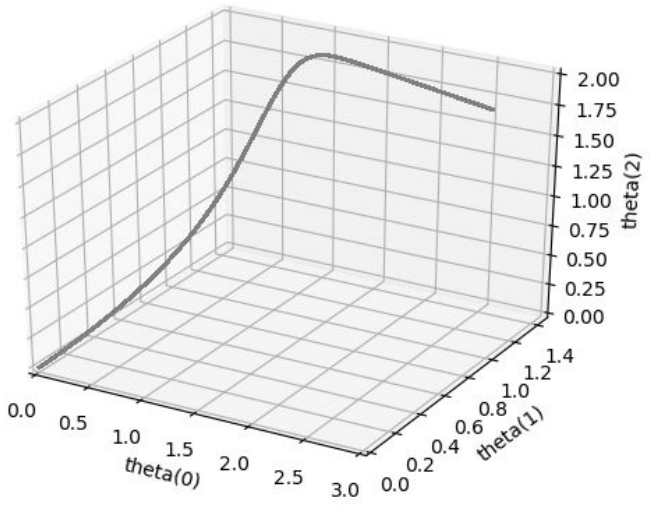
For r =1, the time taken per epoch is very high, so overall time is high.
For r = 1m the number of epochs is very high so overall time is high.
The best values of r are 100 and 10k, for which the convergence sets on early and gives good vaue of parameters.

r	Cost with orig para	Cost with learnt para
1	0.9829469214999997	1.0549312606486412
100	0.9829469214999997	0.984622494781643
10000	0.9829469214999997	0.9830544986811237
1000000	0.9829469214999997	0.985964305188744

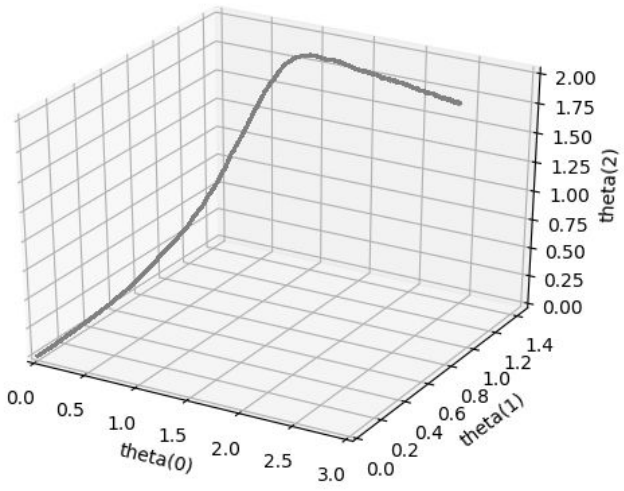
4. d) Plotting X,Y,Z = (theta0 , theta1, theta2) plot for different values of r at each minibatch iteration. The graphs have the same outlying structure. However, in r = 1, the noise is much is visible in the graph while it is smoothened out in r = 100, 10000. This is because when increasing the batch size, the noise gets averaged out.



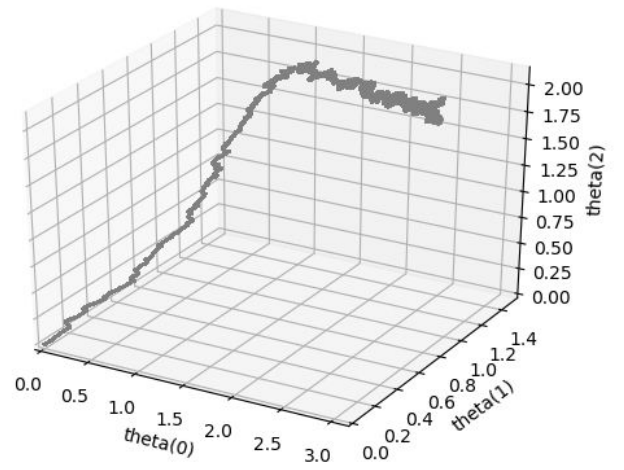
X,Y,Z = (theta0 , theta1, theta2) plot when r = 1000000



X,Y,Z = (theta0 , theta1, theta2) plot when r = 10000



X,Y,Z = (theta0 , theta1, theta2) plot when r = 100



X,Y,Z = (theta0 , theta1, theta2) plot when r = 1

Question 3: Logistic regression

Logistic regression was implemented in this question to find out the value of the parameters that could be used to make a prediction about the data. The problem is binary classification and the classes are {0,1}.
Our job was to find the parameters to describe the data using Logistic regression using the given data and log-likelihood function of Logistic regression.
We had to implement Newton's method for optimizing the likelihood function, given by:

1. theta is a matrix of size 3, as the number of parameters (n) is 2.
2. theta = theta - H^{-1} * del(L(theta))
   L(theta) is the log-likelihood, H is the hessian matrix.
3. del(theta) = [..del(L(theta))/$theta_j$..] for j = 0,1,2 and comes out to be (Y -H(theta,X))*(X.T)
   H(theta, X) = 1/(1+exp(- theta.T*X))
4. Hessian is calculatd as [... del(del(L(theta))/$theta_k$)/$theta_j$…] for all j, k = 0,1,2. This comes out to be

$$(-1)^* \sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} \exp(-\text{theta}^T x^{(i)})/(1+\exp(-\text{theta}^T x^{(i)}))^2 \quad \text{for all j, k = 0,1,2}$$

5. Converging Criteria = max j |(θ (t+1) - θ (t))| < ε => Algorithm will converge when the value of maximum component of the absolute difference becomes less than some certain epsilon.
   Here, epsilon = 10^(-10)

1. a) The parameters are: (calculated as explained above)
   theta = [[0.40125316] [2.5885477] [-2.72558849]]

   b) The plot of data and decision boundary:

   The decision boundary is calculated at the curve at which P(y=0|x)=P(y=1|x)
   Solving the equation and plugging in values P(y) = 1/(1+exp(-theta^Tx))
   We get x2 = (-theta[1]*x1-theta[0])/theta[2])

   The algorithm only takes 8 iterations for completion.

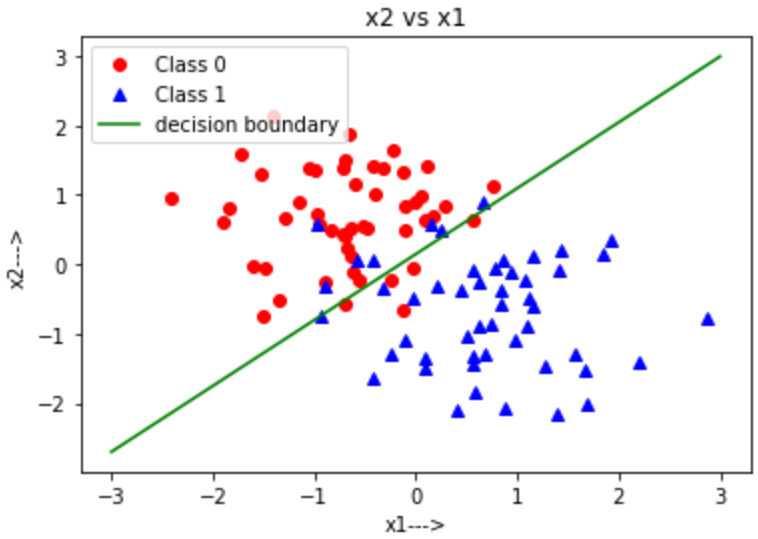


Fig 5. Plot of x2 vs x1, green shows the boundary line

Question 4: Gaussian Discriminant Analysis:

There were two parameters(x1, x2) in this classification problem which were first normalized to have 0 mean and 1 std dev. The classes {Alaska, Canada} were mapped to {0,1}

x1 = diameter in fresh water
x2 = diameter in marine water

We had to use the equations described in class to calculate values of mu1, mu0, common_sigma, sigma1, sigma0. The formula for each of these is attached below in handwritten format (Equation 1 to 5)

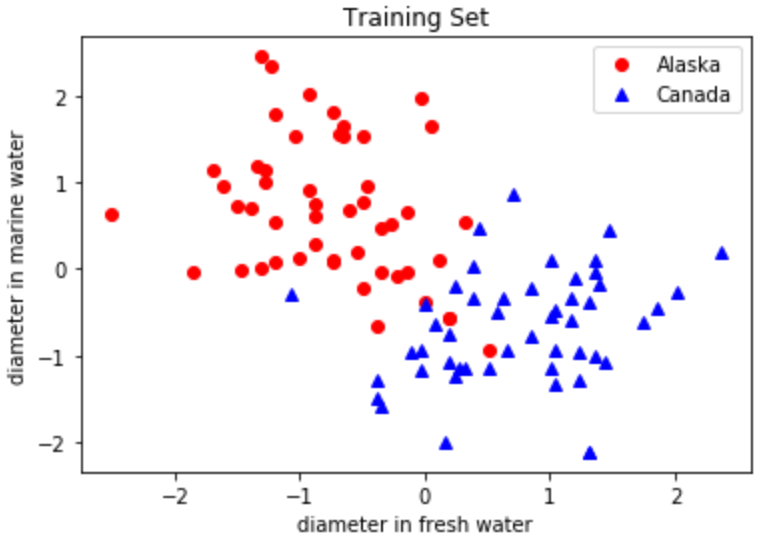
The next step was to calculate the decision boundary when the covariance matrix for both distributions(x1 and x2) were considered the same, that is sigma1 = sigma0 = common_sigma. As we saw in class, this assumption turned the GDA into logistic regression, thus generating a linear decision boundary, given by Equation 6.

However, in general, the covariance matrix is not equal for two distributions and there if we consider sigma1 != sigma2, we end up with a quadratic decision boundary given by equation 7.

4. a) The values of parameters recorded are:

mu0: [[-0.75529433] [0.68509431]]
mu1: [[0.75529433] [-0.68509431]]
common_sigma: [[0.42953048 -0.02247228]
[-0.02247228 0.53064579]]

4. b) Normalized x2 and x1 plot:



Plot of x2 vs x1

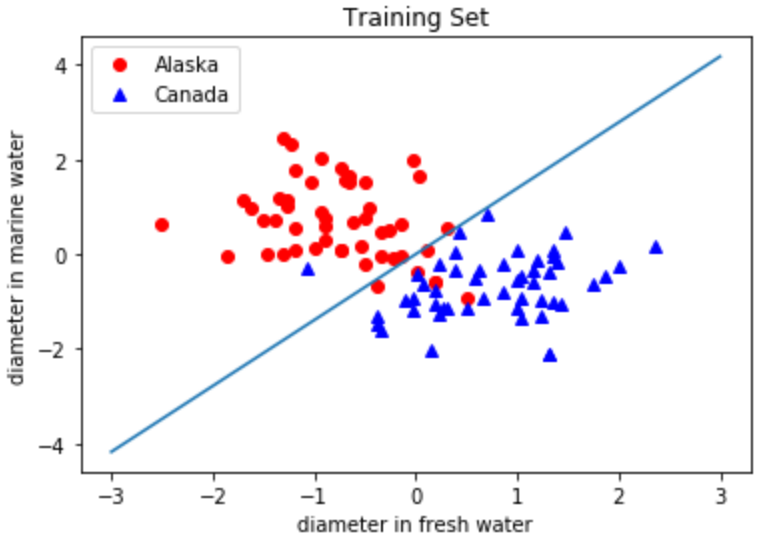
4. c) The equation for linear boundary is given by:

$$x = [x_1\ x_2]^T$$

$$(\mu_0 - \mu_1)^T \Sigma^{-1} x \ + \ \frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 \ - \ \mu_0^T \Sigma^{-1} \mu_0) \ + \ \log(\frac{\phi}{1 \ - \ \phi}) \ = \ 0$$

Phi = P(y=1|x), that is P(fish is from Canada given its parameters)

In this example x2 = (1.11e-15 + 3.39*x1)/(2.44)



x2 vs x1 and linear boundary decision line, in blue

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

Note that the means remain the same as they are independent of the covariance matrix and calculated in an altogether different way.

4.

(e) The equation for the quadratic boundary separating the two regions is:

$x = [x_1\ x_2]^T$

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

Plotting the boundary means solving the equation for y in terms of x. Let $x = [x\ y]^T$
Setting the probability of y = 1 given x to the value of y = 0 given x, we have

P(y = 1/x) = P(y = 0/x)

Plugging in the values of these probabilities by Bayes Theorem and simplifying it further we get,

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

The right side of the equation is a constant, let's call it C. Plugging in the value of phi=p1

$$C=\log|\mathbf{\Sigma_0}|-\log|\mathbf{\Sigma_1}|+2\log p_1-2\log p_0$$

Thus we have to solve the quadratic equation:

$$(\mathbf{x}-\mu_1)'\mathbf{\Sigma}^{-1}{}_1(\mathbf{x}-\mu_1)-(\mathbf{x}-\mu_0)'\mathbf{\Sigma}^{-1}{}_0(\mathbf{x}-\mu_0)=C$$

Which is basically a quadratic equation in x2 and x1

For simplification, x2 = y, x1 = x

x0=x−μ00
y0=y−μ01
x1=x−μ10
y1=y−μ11

then
[x1 y1][a c / b d] [x1 y1] − [x0 y0][p r / q s][x0 y0]=C

[ac/bd] = sigma1_inv
[pr/qs] = sigma0_inv
[ac/bd] means a c in first row and b d in second

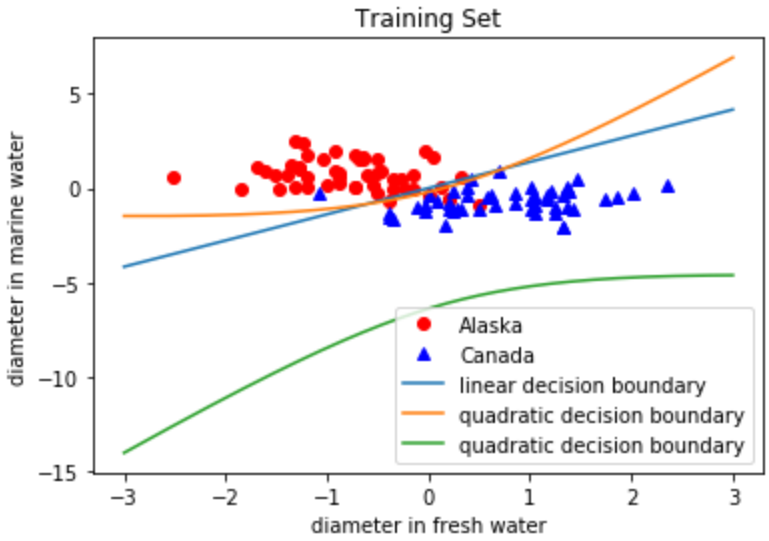
Solving the equation we get something of the form

$$y = \frac{-v \pm \sqrt{v^2 + 4uw}}{2u}$$

where

$$u = d - s$$
$$v = -2d\mu_{11} + 2s\mu_{01} + bx - b\mu_{10} + cx - c\mu_{10} - qx + q\mu_{00} - rx + r\mu_{00}$$
$$w = C - a(x - \mu_{10})^2 + p(x - \mu_{00})^2 + b\mu_{11}x + c\mu_{11}x - q\mu_{01}x - r\mu_{01}x + d\mu_{11}^2 - s\mu_{01}^2 - b\mu_{10}\mu_{11} - c\mu_{10}\mu_{11} + q\mu_{01}\mu_{00} + r\mu_{01}\mu_{00}$$

Finally, the computed curve comes out to be:



Plot of x2 vs x1, linear boundary and quadratic boundary

4. f) We observe a conic section in the form of y = f(x) in the above equation and is rightfully shown to be a hyperbola by the graph. The insides of both the parts of curve contain the points for the class “Alaska” while the space between the curves is for the points of class “Canada”.

We note that logistic regression also gives a good approximation of this classification problem as seen from the straight line.
Remember that GDA has strong assumptions of multivariate normal distribution. In this case it seems, GDA fits the data very well as seen from the above graph.