COL774 ASSIGNMENT 1

Raval Vedant Sanjay

2017CS10366

1 Linear Regression

1.1 Implementation and results

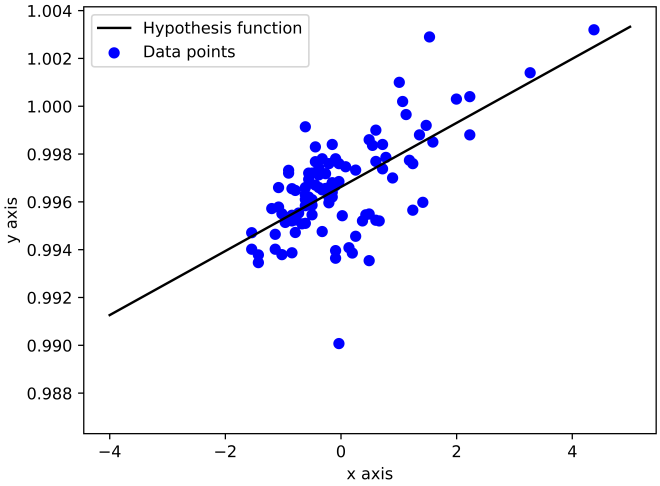
I have implemented this using gradient descent as described in class and assignment on J(θ).

Learning rate: $0.1(\eta = 0.1)$

Stopping criteria:
If difference in the norms of the consecutive values of (θ) is less 10^{-10}

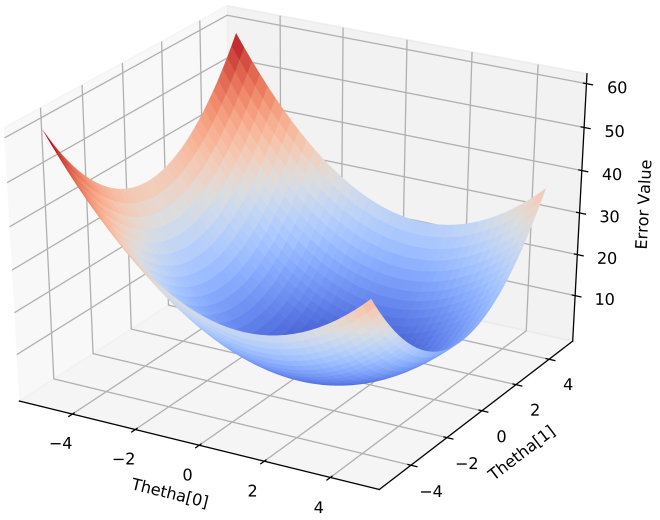
Final Set of Parameters:
$\theta = [0.9966201, 0.0013402]$

1.2 Plot of the Hypothesis function



1.3 3D mesh for error value

I made the animations for the Values of θ converging to the position, having $\rightarrow$ 0 value. I made the use of FuncAnimation for this. The recordings of the screen for these animations are attached in the zip for the different values of η. The mesh plot for the same is included below (Corresponding to $\eta = 0.1$) :



2 Sampling and Stochastic Gradient Descent

2.1 Sampling of data

I sampled the random data as given in the question. A file containing the values corresponding to a run is included (Note that since the generation is random, so you will never get the exact same values again!)

2.2 Stochastic Gradient Descent on the 1 million points

I had run the algorithm corresponding to the different values of r as specified in the question. The values of the parameters obtained by me and the stopping criteria are as given in the following sub sections (Note that the parameter values obtained for these different runs will not be obtained the exact same again because I ran them for different random data)

2.2.1 $r = 1$

This algorithm converged very quickly and it required just somewhat more iterations after all the 1 million data points have been covered. Those 1 million

iterations would be enough in this case to bring the parameters close to the optimum. Then I imposed the condition of the consecutive **batch average error** corresponding to a batch size of 3000, should be less than $5 * 10^{-5}$

Thus, this algorithm achieves convergence in some slightly more iterations after going over all the 1 million data points. And it does this in the fastest manner among all the other algorithms

The values of θ obtained corresponding to this algorithm are:

$$\theta = [3.038301642588017, 1.00928428823021, 1.9811896842795802]$$

2.2.2 $r = 100$

This algorithm also converged very quickly and it again required just somewhat more passes than the 1 million ones. Those 1 million iterations would be enough in this case to bring the parameters close to the optimum. Then I imposed the condition of the consecutive **batch average error** corresponding to a batch size of 5000, should be less than 10^{-3} (The difference is lesser than the one in the $r = 1$ case since here the differences of the error values was obtained more than what was observed for the earlier case, corresponding to the same number of iterations. Though, in the case of the number of iterations as large as 1 million, the difference wasn't that substantial since the parameter values for both the cases had already converged)

Thus, this algorithm achieves convergence in some slightly more iterations after going over all the 1 million data points. And it does this in a very fast manner (Though somewhat slower than the $r = 1$ case)

The values of θ obtained corresponding to this algorithm are:

$$\theta = [2.9920771531749293, 1.0036560602622213, 2.001015275358533]$$

2.2.3 $r = 10000$

Now, This algorithm was very slow in it's execution and it a lot of those 1 million passes (in fact of the order of 100). So, because of this, I imposed the condition of the total number of 1 million passes to exceed 180, and then I imposed the condition of the consecutive **batch average error** corresponding to a batch size of 2000 (Lower size since the algorithm is anyways very slow, so didn't want to add any further reduction in speed), should be less than 10^{-3}

Thus, this algorithm has a much slower speed of convergence than the $r = 1$ and the $r = 100$ case, as well as it requires many more iterations than them (An important reason being the factor of r divided in the parameter update term, which makes this algorithm much slower than the other two)

The values of θ obtained corresponding to this algorithm are:

$$\theta = [2.9768598287664783, 1.0047669699104917, 1.9979146661810465]$$

2.2.4 $r = 1000000$

Now, This algorithm was very very slow in it's execution and it a lot of those 1 million passes (in fact of the order of 1000 this time!). So, because of this, I imposed the condition of the total number of 1 million passes to exceed 5000, and then I imposed the condition of the consecutive **batch average error** corresponding to a batch size of 2000 should be less than 10^{-3}

Thus, this algorithm has the slowest speed of convergence as well as it requires the highest number of iterations among all (The reason being again the factor of r divided in the parameter update term, as well as the huge batch size which makes each step of the algorithm very expensive. It is the conventional Batch Gradient Descent in fact!). This algorithm took literally hours to converge and so it's technically impossible to run this algorithm during the demo

The values of θ obtained corresponding to this algorithm are:

$$\theta = [3.0019348435234035, 0.9993534593453501, 2.0002348235234019]$$

2.2.5 Observation

Thus here we see that all these different algorithms converge to the same value and this is the same as the initial hypothesis of $[3, 1, 2]$ set by us for getting the random data points.

2.3 Error values for the $Q2_{test}$ file

The values of the Errors corresponding to the different parameters are given as below:

- The error for the Assumed hypothesis is: 0.982947
- The error for the First hypothesis is: 1.009360
- The error for the Second hypothesis is: 0.983359
- The error for the Third hypothesis is: 0.984557
- The error for the Fourth hypothesis is: 0.982967

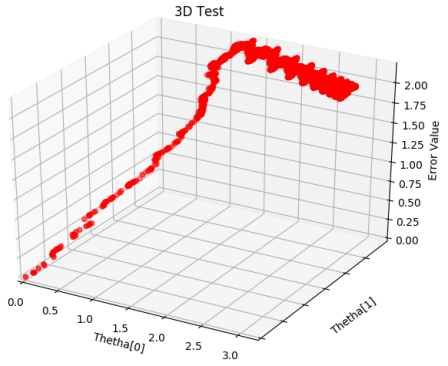
Here, from these different values we see that the initial guess for the hypothesis does not perfectly fit the data as we may have thought. This is because of the error term involved while obtaining the $\mathbf{Y}$ values in the process of the generation of the data. And now, since this error is nowhere related to θ, we can

see that the exact values of the parameters will be different from the assumed initial values. Though, the difference between these values will be negligible and so it can be a good enough choice to assume the initial guess ass the exact set of parameters!

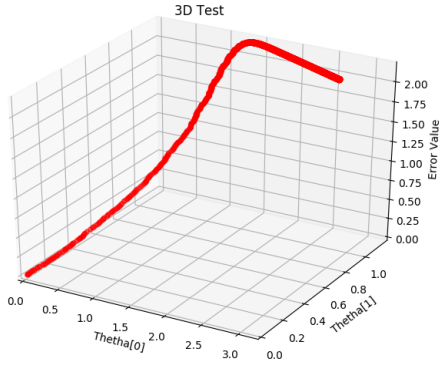
2.4 Movement of the Parameters

The following curves were obtained corresponding to the random data generated of size 1000 and the values of r were correspondingly 1,10,100, and 1000 (The value of η was taken as 0.001 here as well)

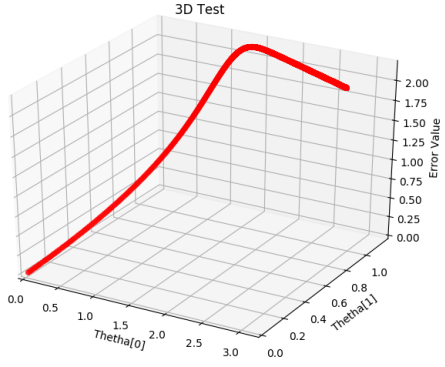
Curve for r=1:

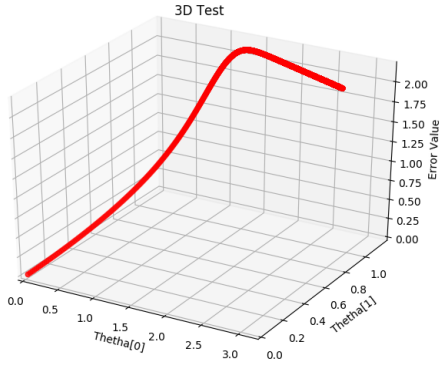


Curve for r=10:



Curve for r=100:





Curve for r=1000:

An important observation from these traces here is that as the values of r is increasing, the parameters are converging to the same location, but in a smoother manner (i.e. the local fluctuations corresponding to the typical SGD seems to be decreasing with the higher values of r, when it tends to be behaving like a typical Batch Gradient descend)

3 Logistic Regression

3.1 Implementation

I have implemented logistic regression by maximizing log likelihood function of logistic regression as given in the problem. I have done this using newton's method. I got hessian matrix using the equation

$$H[j,k] = \sum_{i=1}^{m} h_\theta(x^i)(1 - h_\theta(x^i))(x_j^i)(x_k^i)$$

where m is input size.

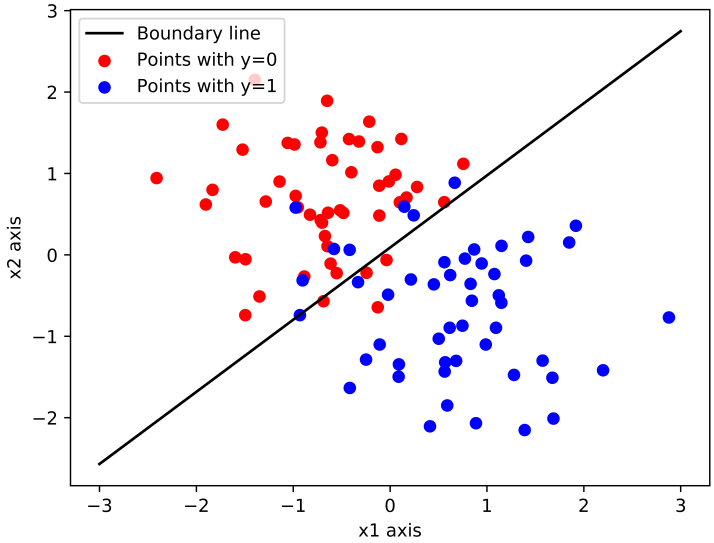
To get derivatives i have used the following equation:

$$\frac{\partial(l(\theta))}{\partial\theta_j} = \sum_{i=1}^{m}(h_\theta(x^i) - y^i)(x_j^i)$$

I applied newton's method using things mentioned above.

Results obtained:
$\theta = [0.225, 2.25678138, -2.54669366]$

3.2 Plot



Here equation of line is obtained corresponding to the $h(x) = 0.5$ will be $\theta^T x = 0$
Thus, the equation of line obtained correspondingly is: $\theta_0 + \theta_1 x_1 + \theta_1 x_2 = 0$

4 Gaussian Discriminant Analysis

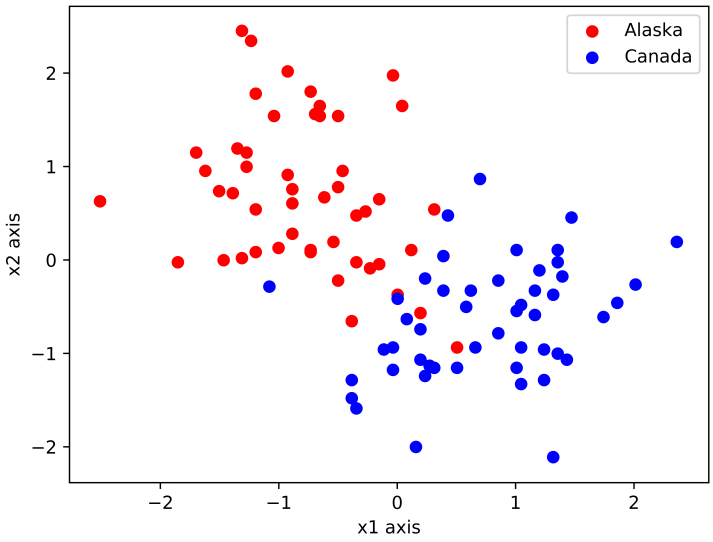
4.1 Data obtained assuming same co-variance matrix of both the classes

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

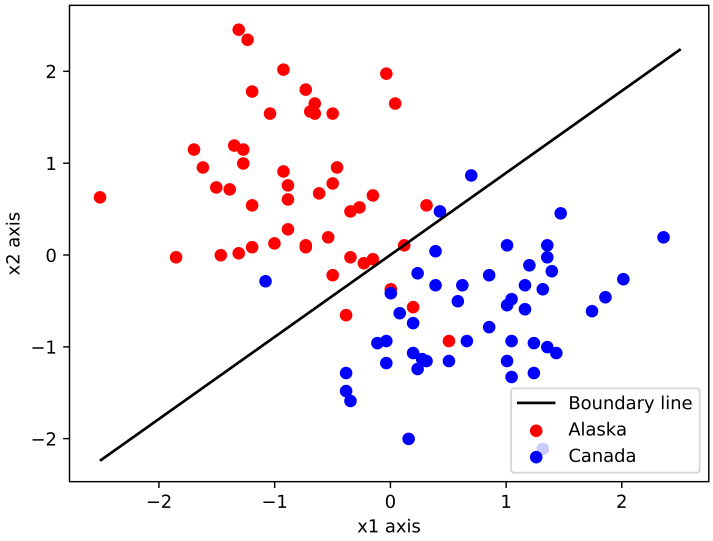
$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\sum_0 = \sum_1 = \sum = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

4.2 Plot of input data



4.3 Plot with decision boundary



4.3.1 Linear Equation Obtained

By solving the equations corresponding to $P(Y|X = 1) > P(Y|X = 0)$ and using the equations derived by us in the class for the different probability distributions and expanding the equations, along with using the properties that

$\mu_1 = -\mu_0$ and $\sum$ is a symmetric matrix, I came up with the equation of the line as:

$(\sum^{(1,1)} \mu_0^{(1)} + \sum^{(0,1)} \mu_0^{(0)})x_0 + (\sum^{(0,0)} \mu_0^{(0)} + \sum^{(0,1)} \mu_0^{(1)})x_1 = 0$

4.4 Classes with different co-variance matrix

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\sum\nolimits_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.68509431 \end{bmatrix}$$

$$\sum\nolimits_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

4.5 Quadratic Equation Obtained

Using the same strategy to obtain the Decision boundary for the general consideration of unequal co variance matrices, just as done in the case of equal co variance matrices, we get the decision boundary as a general curve of:

$$AX_0^2 + BX_0X_1 + CX_1^2 + DX_0 + EX_1 + F = 0$$

$$A = \sum\nolimits_1^{-1}(0,0) - \sum\nolimits_0^{-1}(0,0)$$

$$B = 2(\sum\nolimits_1^{-1}(0,1) - \sum\nolimits_0^{-1}(0,1))$$

$$C = \sum\nolimits_1^{-1}(1,1) - \sum\nolimits_0^{-1}(1,1)$$

$$D = temp_1^{(0)} - temp_2^{(0)}$$

$$E = temp_1^{(1)} - temp_2^{(1)}$$

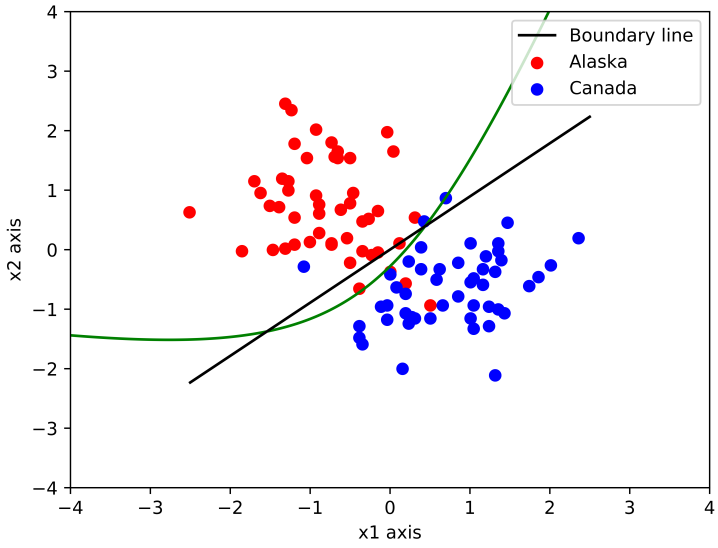
$$F = \log|\sum\nolimits_0| - \log|\sum\nolimits_1| + (\mu_1^T \sum\nolimits_1^{-1} \mu_1) - (\mu_0^T \sum\nolimits_0^{-1} \mu_0)$$

Where, the $temp_1$ and $temp_2$ are defined as follows:

$$temp_1 = \mu_0^T \sum\nolimits_0^{-1} - \mu_1^T \sum\nolimits_1^{-1}$$

$$temp_2 = \sum_0^{-1} \mu_0 - \sum_1^{-1} \mu_1$$

The combined graph with scatter points, the linear and quadratic boundary



4.6 Comments on the observations

On observing the above given graph, it can seem to appear that the quadratic boundary is dividing the scatter in a slightly more proper way than that done by the Linear boundary.

Though this difference seems very negligible, it should be noted that the differences between these boundaries can be much more significant in the data set containing more than a million training examples and being scatter in a more random way than the given data, which is seemingly symmetric in it's coordinates as well as has very less training points.

Thus, in general, the general assumption of having unequal co variance matrices will work better than the simple assumption of equal co variance matrices.

Also, from the equation of the curve obtained, we can see that it is a hyperbola!