COL774 Assignment 1

Rishit Singla

September 2023

1 Linear Regression

(a)

For learning rate $\alpha = 0.03$ and stopping criterion $|J(\theta_t) - J(\theta_{t+1})| < \epsilon = 10^{-14}$. The batch gradient descent algorithm gives the value of Theta to be

$$\theta = \begin{pmatrix} 0.99662 \\ 0.00134 \end{pmatrix}$$

The algorithm took 473 iterations to reach the result.

(b)

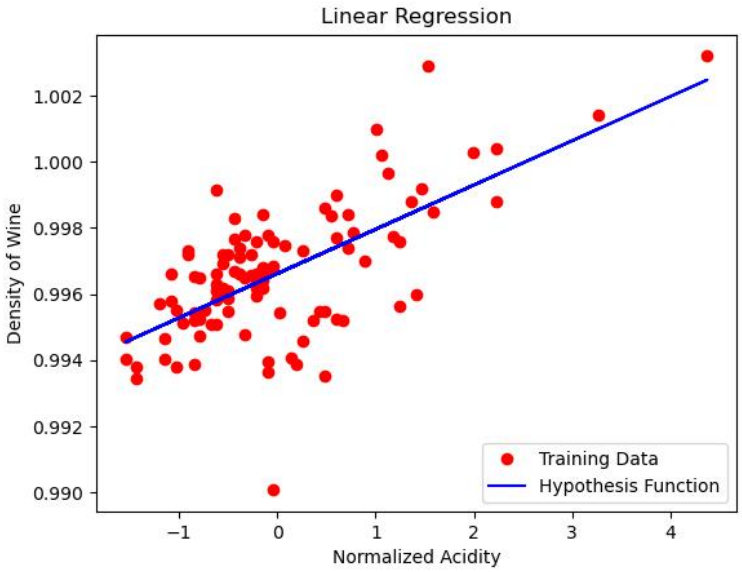

Figure 1: Data and Hypothesis functions plot

(c)

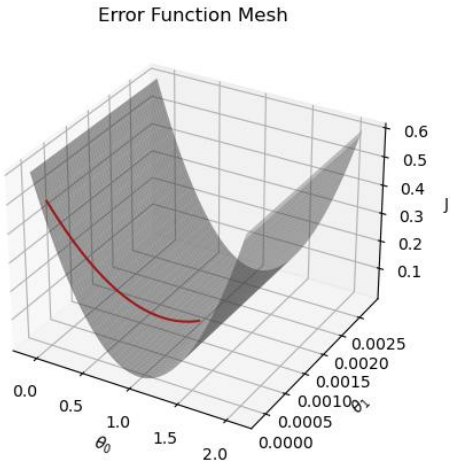

Figure 2: Mesh plot of cost function against Theta

(d)

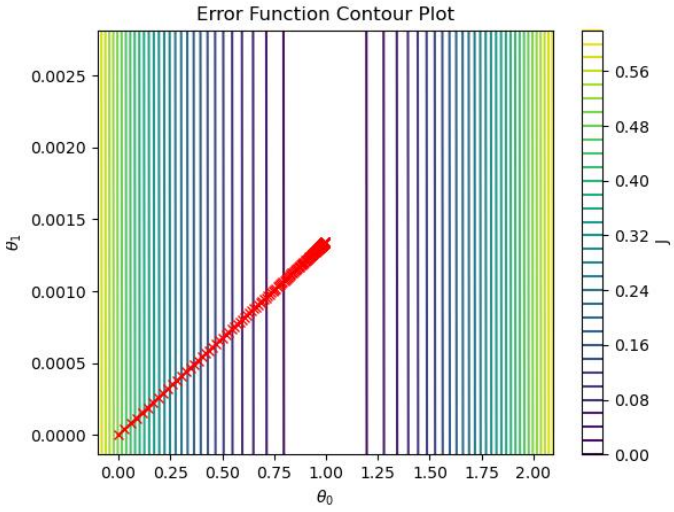

Figure 3: Contour Plot of cost function against Theta

(e)

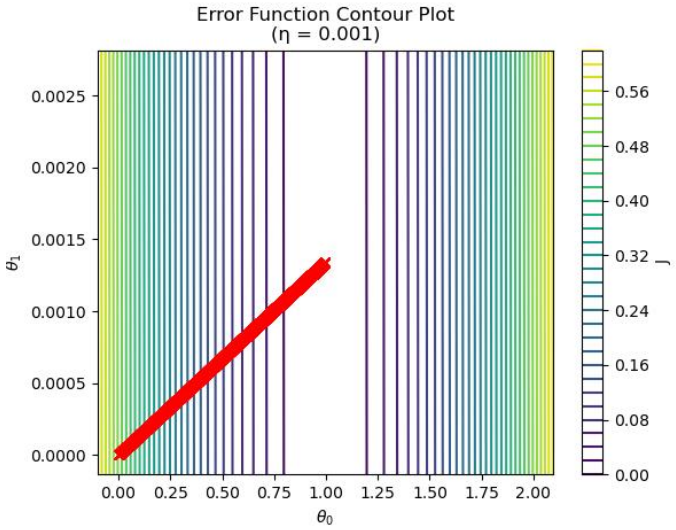

Figure 4: Contour Plot of cost function against Theta for $\eta = 0.001$

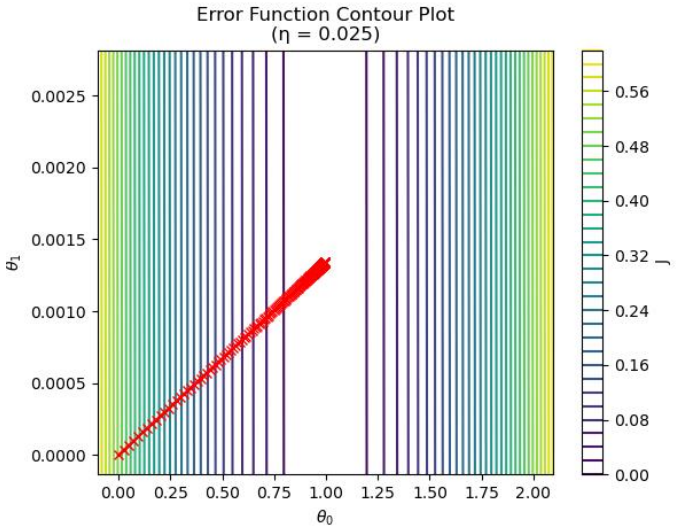

Figure 5: Contour Plot of cost function against Theta for $\eta = 0.025$

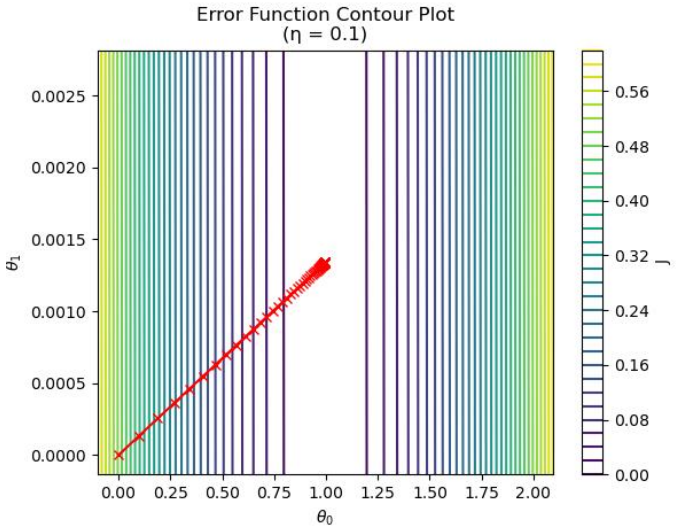

Figure 6: Contour Plot of cost function against Theta for $\eta = 0.1$

η	θ_0	θ_1	Number of iterations
0.1	0.99662	0.00134	143
0.025	0.99662	0.00134	565
0.001	0.99662	0.00134	12656

Observations: We observe that for small η the steps taken by batch gradient descent is smaller and hence results into larger number of iterations. Thus takes more time to reach the desired value of hypothesis function. The accuracy of algorithm for different learning rates is almost same. Though if we take even larger learning rate it might be possible that the algorithm misses the optimal solution. So, our learning rate should not be much high to avoid missing the optimal solution and not too low as it will be very slow.

2 Sampling and Stochastic Gradient Descent

(a)

One million data points sampled are stored in X.csv and Y.csv.

$$X = \begin{pmatrix} 1 \\ \mathcal{N}(3,4) \\ \mathcal{N}(-1,4) \end{pmatrix}$$

$$Y = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}^T X + \mathcal{N}(0,2)$$

(b)

Convergence Criterion: $|J(\theta_t) - J(\theta_{t+1})| < \epsilon$. I have avoided shuffling since it was taking significant amount of time for one million points and also even without shuffling the results are quite accurate, precise and reachable within a reasonable amount of time. So there was no loss in avoiding shuffling.

ϵ	$r = 1$	$r = 100$	$r = 10000$	$r = 1000000$
1×10^{-10}	(14.4, 1.2306)	(0.8, 0.9832)	(8.5, 0.9829)	($\geq$ 180, ?)
1×10^{-9}	(13.7, 1.1595)	(0.5, 0.9846)	(6.4, 0.9830)	($\geq$ 180, ?)
1×10^{-8}	(15.0, 1.2306)	(0.6, 0.9832)	(6.6, 0.9829)	($\geq$ 180, ?)
1×10^{-7}	(14.7, 1.2306)	(0.6, 0.9832)	(4.5, 0.9830)	($\geq$ 180, ?)
1×10^{-6}	(14.0, 1.2306)	(0.4, 0.9834)	(4.0, 0.9834)	($\geq$ 180, ?)
1×10^{-5}	(13.8, 1.2306)	(0.4, 0.9832)	(3.3, 0.9870)	(157.4, 1.3595)
1×10^{-4}	(13.9, 1.2306)	(0.3, 0.9832)	(2.3, 0.9832)	(69.6, 4.7293)

Table 1: Comparison of Results for values of r and ϵ. Each cell in table corresponds to (time in seconds, J (cost function)). The optimal value of J is 0.9829. Program is run for only 180 sec and if some case took longer then its value is not reported.

For $r \in \{1, 100, 10000\}$, the value of ϵ is taken to be 10^{-10} as it gives best possible result in reasonable amount of time. For $r = 1000000$, ϵ is taken to be 10^{-4} due to large amount of time taken by code to execute for lower values of ϵ.

r	θ	ϵ	epoch	iterations	time (sec)
1	$\begin{pmatrix} 2.977 \\ 0.977 \\ 1.977 \end{pmatrix}$	10^{-10}	2	2000000	13.4
100	$\begin{pmatrix} 3.000 \\ 0.998 \\ 1.996 \end{pmatrix}$	10^{-10}	7	70000	0.8
10000	$\begin{pmatrix} 2.999 \\ 0.999 \\ 2.000 \end{pmatrix}$	10^{-10}	371	37100	6.2
1000000	$\begin{pmatrix} 1.865 \\ 1.247 \\ 1.918 \end{pmatrix}$	10^{-4}	3299	3299	78.5

Table 2: Theta values for different batch sizes

(c)

r	$J(\theta)$
1	0.9979
100	0.9848
10000	0.9831
1000000	4.6640

Table 3: Values of $J(\theta)$ for different values of batch size when calculated on new 10000 samples. For original θ the value of J is 0.9829

The value of J for original hypothesis is 0.9829. Stochastic Gradient descent converges to different parameter values depending upon the batch size. Thus, each one has a different error. Gradient descent with batch size 10000 is the closest to original answer followed by batch size 100 and batch size 1. Stochastic Gradient Descent with batch size 1000000 is most inaccurate. Algorithm converges very fast in case of r = 100, 10000. For r = 1, it takes a little long to converge, even though the number of epochs are low. This is because each epoch has quite a large number of iterations (1000000 iterations to be precise). Due to this Theta fluctuates too much and slows down the convergence. In case of r = 1000000, the algorithm is extremely slow. The stopping criterion (ϵ) also has to be increased in order to make the algorithm converge within a few minutes. Due to large size each step of stochastic gradient descent is quite large, but computationally costly. Due to large steps, change in θ reduces as we come close to optimal value and hence change in J reduces considerably. Thus, we need a smaller stopping criterion to avoid this early convergence in case of r = 1000000 (i.e. large batch size).

(d)

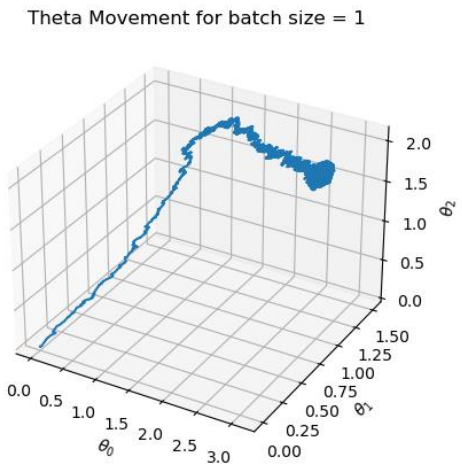

Figure 7: Theta Movement for r = 1

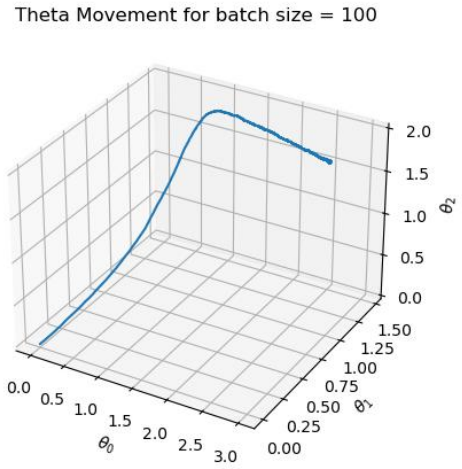

Figure 8: Theta movement for r = 100

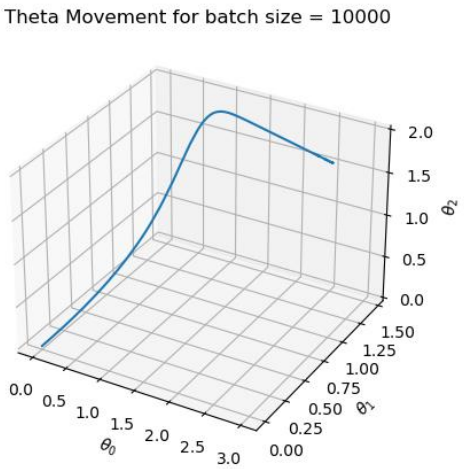

Figure 9: Theta movement for r = 10000

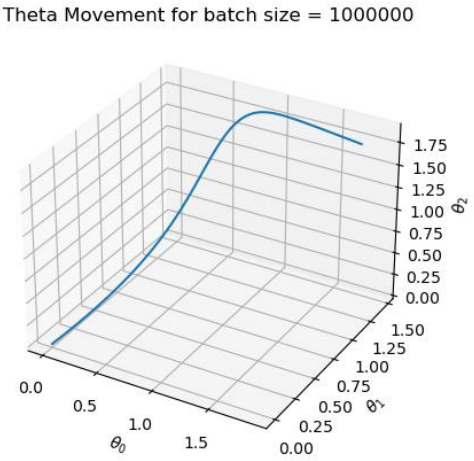

Figure 10: Theta movement for r = 1000000

As we go from r = 1 to r = 1000000, the fluctuations in path of theta reduces. r = 1 has many fluctuations which makes it slower. r = 100, 10000 has smoother path so converges quickly. For r = 1000000, path is quite smooth but it converges early due to large steps taken by the algorithm.

3 Logistic Regression

(a)

Lets take the logistic function to be sigmoidal. Therefore $h(x) = \dfrac{1}{1+e^{-\theta^T x}}$

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))$$

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

$$\text{Let } X = \begin{pmatrix} x^{(1)T} \\ x^{(2)T} \\ . \\ . \\ . \\ x^{(m)T} \end{pmatrix}, h(X) = \begin{pmatrix} h(x^{(1)}) \\ h(x^{(2)}) \\ . \\ . \\ . \\ h(x^{(m)}) \end{pmatrix}, Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ . \\ . \\ . \\ y^{(m)} \end{pmatrix}$$

$$\boxed{\therefore \nabla_\theta l(\theta) = X^T(Y - h(X))}$$

$$\begin{aligned} \frac{\partial^2 l(\theta)}{\partial \theta_j \theta_k} &= \frac{\partial}{\partial \theta_k} \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)})) x_j^{(i)} \\ &= -\frac{\partial}{\partial \theta_k} \sum_{i=1}^{m} h(x^{(i)}) x_j^{(i)} \\ &= -\sum_{i=1}^{m} h(x^{(i)})(1-h(x^{(i)})) x_j^{(i)} x_k^{(i)} \end{aligned}$$

$$\boxed{\therefore H(\theta) = X^T h' X \text{ where } h' = \operatorname{diag}\Big(h(x^{(1)})(1-h(x^{(1)})), ..., h(x^{(m)})(1-h(x^{(m)}))\Big)}$$

Using Newtons method when gradient and hessian is calculated using above equations we get the following results in 9 iterations for stopping criterion of $|l(\theta_t) - l(\theta_{t+1})| < \epsilon = 10^{-20}$

$$\theta = \begin{pmatrix} 0.40125 \\ 2.58855 \\ -2.72559 \end{pmatrix}$$

(b)

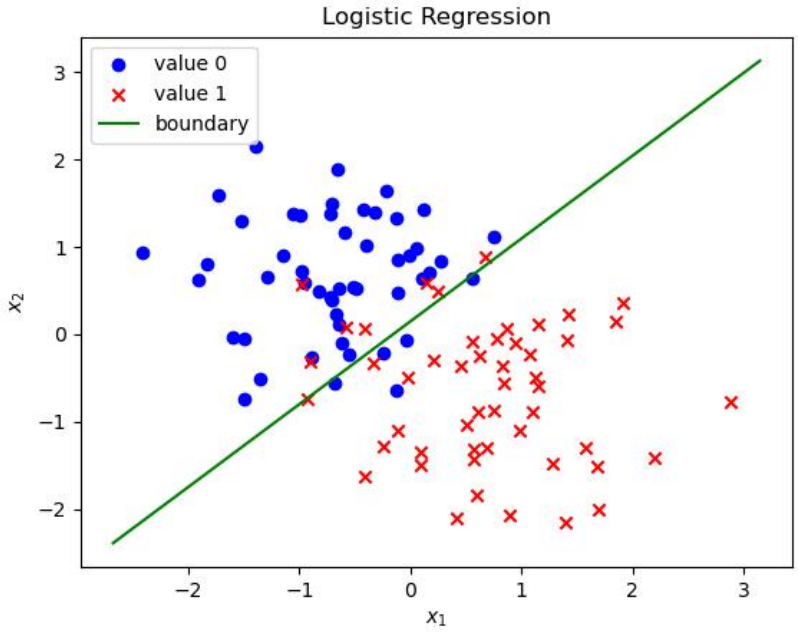

Figure 11: Decision Boundary and Training data for logistic regression

4 Gaussian Discriminant Analysis

(a)

Taking Alaska as 0 and Canada as 1 we get

$$\phi = 0.5$$

$$\mu_0 = \begin{pmatrix} -0.75529 \\ 0.68509 \end{pmatrix}$$

$$\mu_1 = \begin{pmatrix} 0.75529 \\ -0.68509 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.42953 & -0.02247 \\ -0.02247 & 0.53065 \end{pmatrix}$$

(b)

Plot is in part (e)

(c)

Equation of separator:

$$\phi\mathcal{N}(\mu_1, \Sigma) = (1-\phi)\mathcal{N}(\mu_0, \Sigma)$$

$$\begin{aligned}\frac{\phi}{1-\phi} &= \frac{\mathcal{N}(\mu_0, \Sigma)}{\mathcal{N}(\mu_1, \Sigma)} \\ &= e^{-\frac{1}{2}\Big((x-\mu_0)^T\Sigma^{-1}(x-\mu_0)-(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big)}\end{aligned}$$

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

Above equation is linear and not quadratic because on simplification it reduces to

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

Plot is in part (e)

(d)

Following values are obtained from general GDA:

$$\phi = 0.5$$

$$\mu_0 = \begin{pmatrix}-0.75529 \\ 0.68509\end{pmatrix}$$

$$\mu_1 = \begin{pmatrix}0.75529 \\ -0.68509\end{pmatrix}$$

$$\Sigma_0 = \begin{pmatrix}0.38159 & -0.15487 \\ -0.15487 & 0.64774\end{pmatrix}$$

$$\Sigma_1 = \begin{pmatrix}0.47747 & 0.10992 \\ 0.10992 & 0.41355\end{pmatrix}$$

Equation of separator in this case becomes:

$$\log(\frac{\phi}{1-\phi}\sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}}) + \frac{1}{2}\Big((x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\Big) = 0$$

(e)

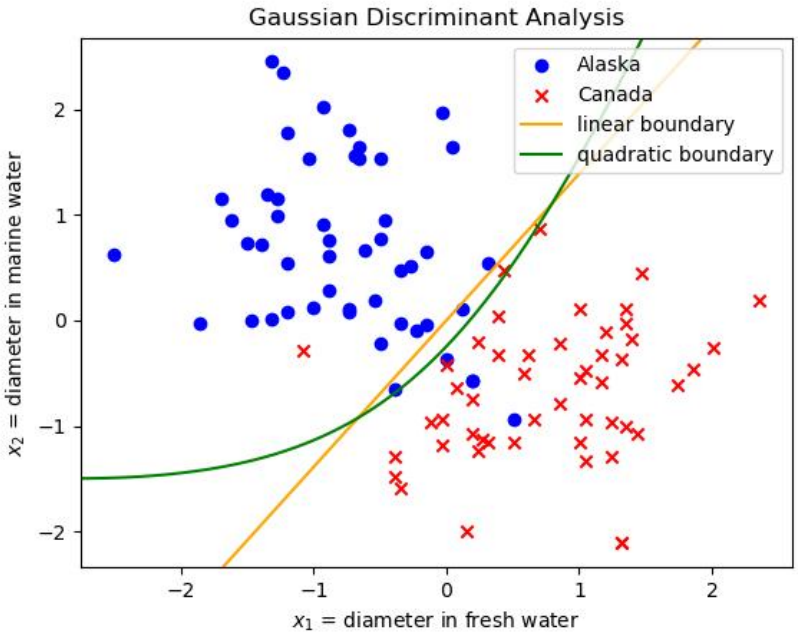

Figure 12: Plot for data with linear and quadratic separator

(f)

The quadratic separator takes two blue points in Alaska region along with one red point as compared to linear separator. Thus in total it only increase the accuracy of hypothesis by one point of training data. Also it intersects with the linear separator in two points and bends on the side of Alaska. Thus, due to bending towards Alaska, quadratic separator implies that it is more probable to find points in Canada region as compared to Alaska region. Though the training data does not show such pattern. Hence the quadratic separator is a result of overfit of data.