COL774 - A1

Prakul Virdi

September 2022

1 Question 1

1.1 (a)

Since the input data x is 1-dimensional, we will have 2 parameters for θ, i.e., $\theta = \binom{\theta_0}{\theta_1}$

$$h_\theta(x) = \theta_0 * x_0 + \theta_1 * x_1 \text{ where } x_0 = 1$$

Since we are implementing gradient descent we need to have a stopping criteria when the algorithm converges. Let θ_0^i and θ_1^i be the parameters at the i^{th} iteration. When,

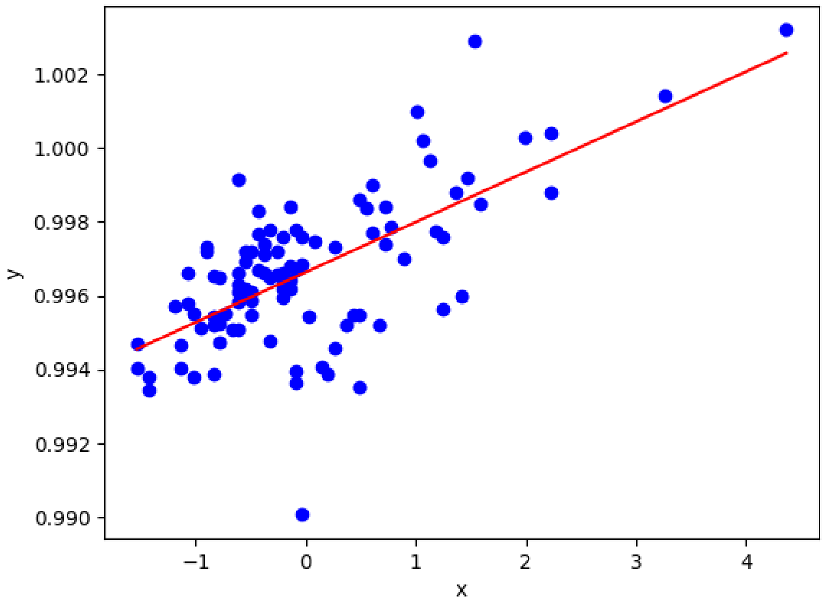
$$|\theta j_{i+1} - \theta j_i| < \epsilon \ \forall j \in \{0, 1\} \text{ where } \epsilon \text{ is some small value}$$

we stop the algorithm. The following parameters were used/returned by gradient descent.

Parameters	Values
η	0.01
ϵ	10^{-15}
θ_0	0.9967
θ_1	0.00135
$cost$	$1.18 * 10^{-06}$

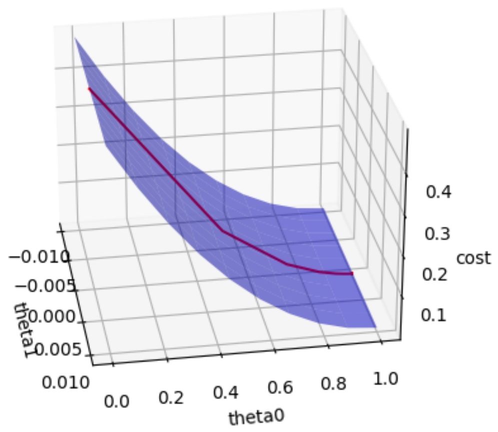
1.2 (b)

We obtained the following graph where the blue points represent the data we were given and the red line is the hypothesis that we have learnt.



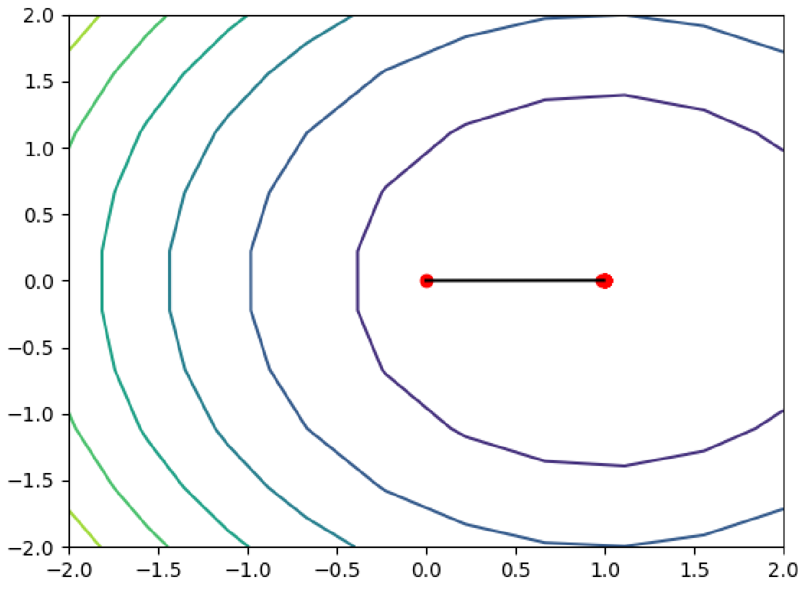
1.3 (c)

Since the cost is a convex function, we can observe that the thetas converge to the minimum value.



1.4 (d)

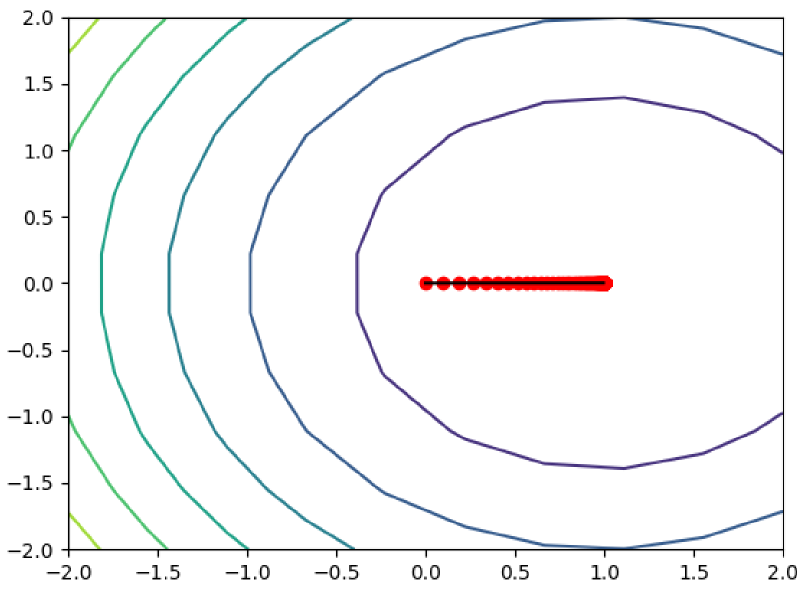
Since the learning rate, η is relatively large, it converges rather quickly.



1.5 (e)

1.5.1 Convergence

- $\eta = 0.001$; on the $x - axis$ and $y - axis$ we θ_0 and θ_1 respectively. The lines are the contours of the cost function.



We observe that decreasing η makes gradient descent slower as it takes more iterations to converge.

1.5.2 Divergence

- $\forall\ \eta \in \{0.025, 0.1\}$ gradient descent diverges. It's also expected because we have increased it too much.

2 Question 2

2.1 (a)

I used the library function of *numpy* to sample one point from a normal distribution. Computed y for this point. Repeated the sampling process 1 million times.

2.2 (b)

Since the gradients may be noisy due to smaller batch sizes, the gradients may not always decrease. Contrary to Question 1 where we compared θ^i and θ^{i+1}, here we compare θ^t_{avg} and θ^{t+1}_{avg} where,

$$\theta^t_{avg} = 1/k * \sum_{n=t*k}^{(t+1)*k-1} \theta^i$$

When,

$$||\theta^{t+1}_{avg} - \theta^t_{avg}||_2 < \epsilon$$

where $||.||_2$ is the $L2$ norm and ϵ is some value. Hence, when we put ϵ small enough, we can guarantee convergence.

The table below shows the parameters used for convergence varying the batch size.

r	*theta*	ϵ	k	*iters*
1	$\begin{bmatrix} 2.90096727 \\ 1.00398103 \\ 1.9958781 \end{bmatrix}$	1	100	11700
10	$\begin{bmatrix} 2.97415026 \\ 1.00340122 \\ 1.99621081 \end{bmatrix}$	0.1	50	7250
10000	$\begin{bmatrix} 2.98299967 \\ 1.00181044 \\ 1.99969435 \end{bmatrix}$	0.001	10	8190
1000000	$\begin{bmatrix} 2.98714405 \\ 1.00147602 \\ 1.99980244 \end{bmatrix}$	0.00001	1	8560

2.3 (c)

The *thetas* corresponding to different values of r don't exactly converge to the same value, but all are very close to each other. Since the original hypothesis was $\theta = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$ we can safely say that stochastic gradient descent converges closely to the original hypothesis. As we increase the batch size, our stochastic gradient becomes slower. The number of iterations of each algorithm is shown in the table above.

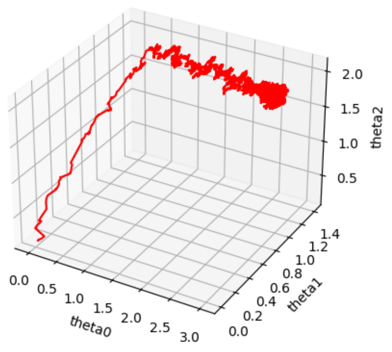
- Original hypothesis, $cost = 0.9829469215000091$

The below table shows the cost of all the hypotheses that we learnt for different batch sizes, r.

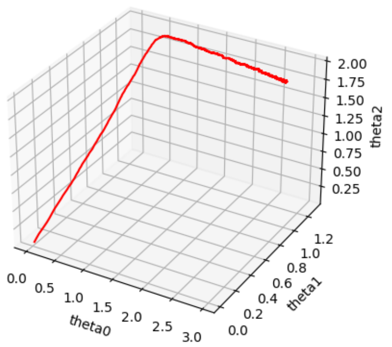
r	$cost$
1	0.9895715244767835
100	0.9848810836329218
10000	0.9832557604388013
1000000	0.9831320884069167

We observe that as we increase the batch size, the cost gets closer and closer to the cost of the original hypothesis.

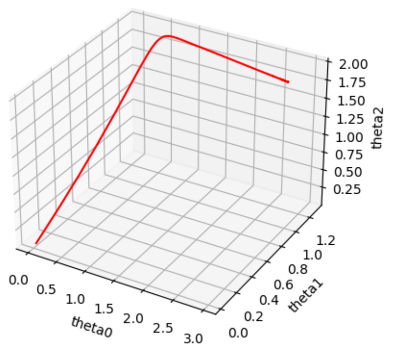
2.4 (d)



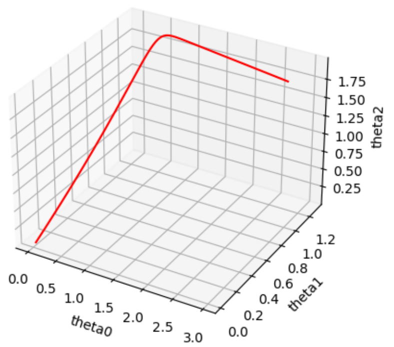
- $r = 1$



- $r = 100$



- $r = 10000$



- $r = 1000000$

It makes intuitive sense as the higher the batch size, the better approximation we have of the gradient and hence it becomes less noisy and becomes smoother.

3 Question 3

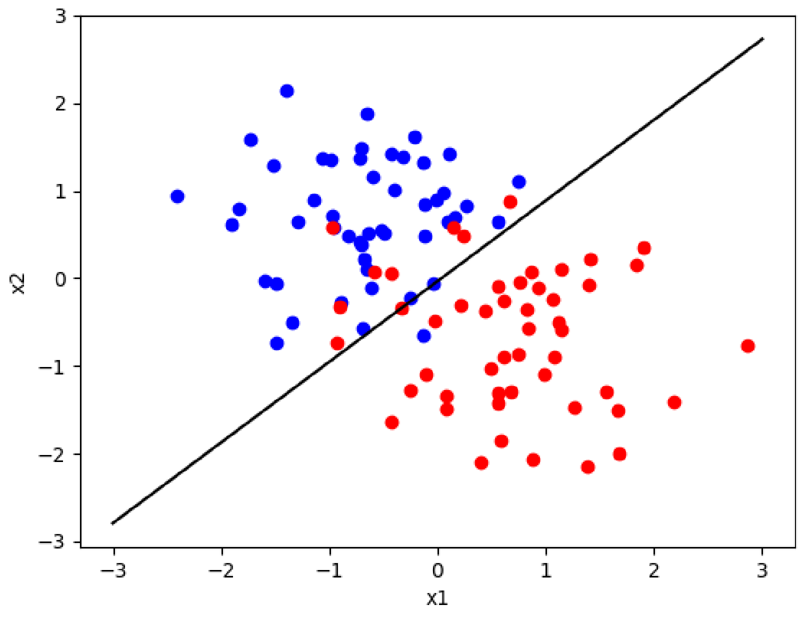
3.1 (a)

I implemented a vectorized form of the equations, and I followed the formulas suggested here to compute the Hessian, refer The coefficients observed are,

$$\theta = \begin{bmatrix} 0.0916248 \\ 0.64924624 \\ -0.65238769 \end{bmatrix}$$

3.2 (b)

In the following graphs, we have $x1$ and $x2$ on the $x-axis$ and $y-axis$ respectively. The points coloured in red represent the class $y = 1$, while blue points correspond to $y = 0$. The black coloured line is our decision boundary, i.e., the line where $h_\theta(x) = 0.5$.



4 Question 4

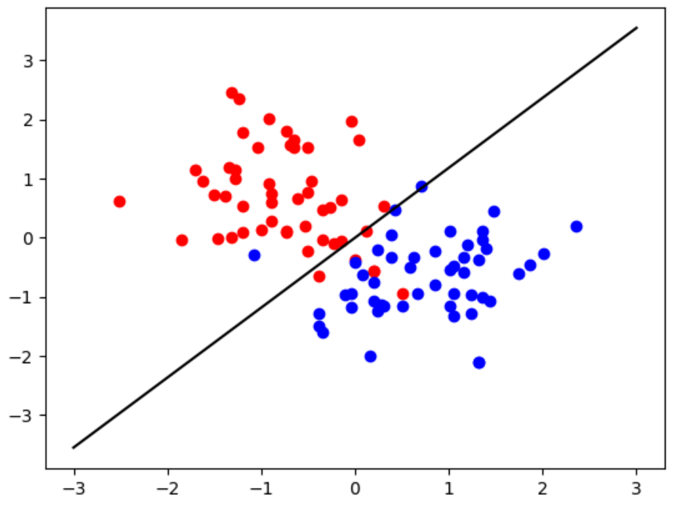
WLOG, assume $y = 1$ corresponds to the class **Alaska**

Parameters	Values
ϕ	0.5
μ_0	$\begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$
μ_1	$\begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$
$\sum$	$\begin{bmatrix} 2.93378037 & -0.04494456 \\ -0.04494456 & 3.136011 \end{bmatrix}$

Note that all the above parameters were computed using the closed-formed equations taught in class.

4.1 (b) and (c)

In the following graph, the red points and blue points denote the $y = 1$ and $y = 0$ classes respectively. The $x - axis$ and $y - axis$ correspond to x_1 and x_2 respectively.The black line shows the linear separator learnt by GDA.



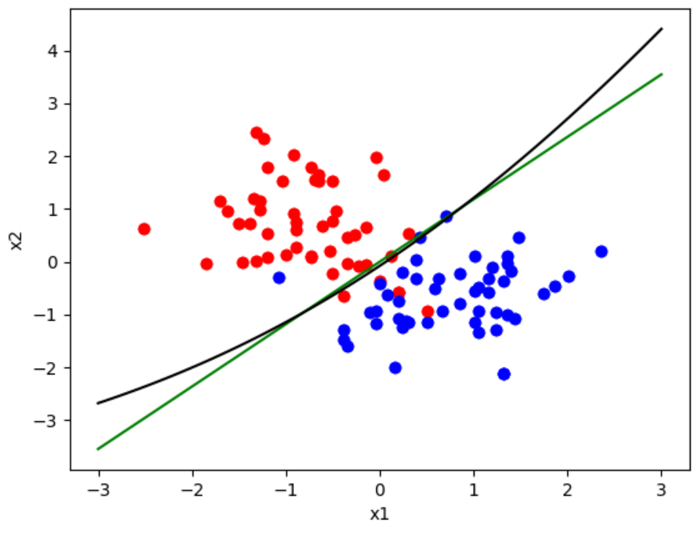
4.2 (d)

The parameters observed are,

Parameters	Values
ϕ	0.5
μ_0	$\begin{bmatrix} 0.5 \\ -0.4 \end{bmatrix}$
μ_1	$\begin{bmatrix} -0.38 \\ 0.36 \end{bmatrix}$
$\sum_0$	$\begin{bmatrix} 1.55 & 0.04 \\ 0.04 & 1.53 \end{bmatrix}$
$\sum_1$	$\begin{bmatrix} 0.274 & 0.254 \\ 0.254 & 2.234 \end{bmatrix}$

4.3 (e)

The green line represents the quadratic decision boundary learnt by GDA.



4.4 (f)

The quadratic decision boundary is a more general fit to the data than the linear boundary. We see that the linear boundary assigns more data points as $y = 1$ than the quadratic boundary. Hence, we could say that the quadratic boundary is a tighter bound to get $y = 1$.