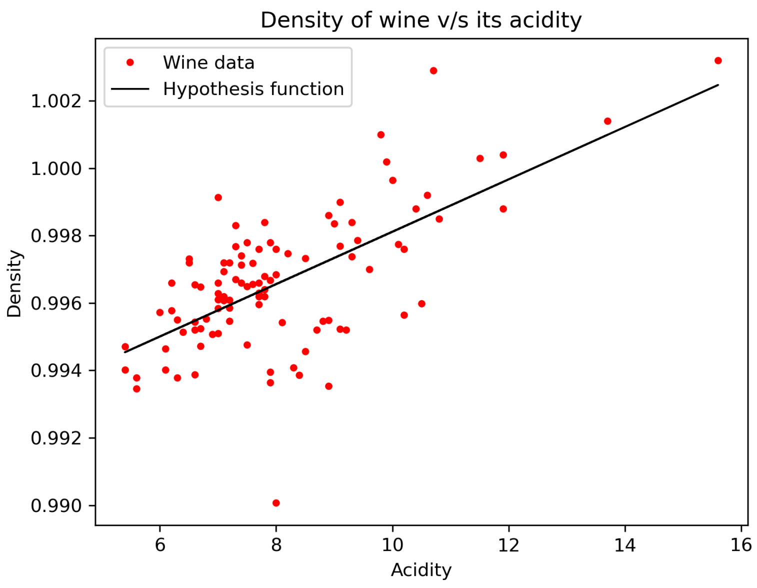
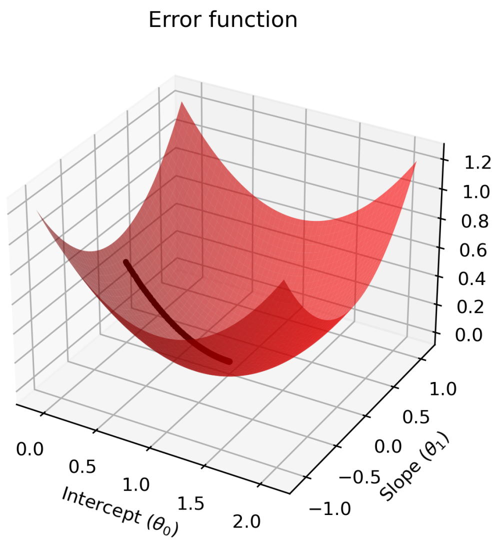
Assignment 1

1. (a) Gradient descent was implemented with learning rate $\eta = 0.01$ and stopping criteria: $\epsilon = 10^{-12}$, such that convergence occurs when absolute difference in error between 2 successive iterations is $\leq \epsilon$
   Parameters obtained: $\theta = (0.99661018, 0.00134018)$

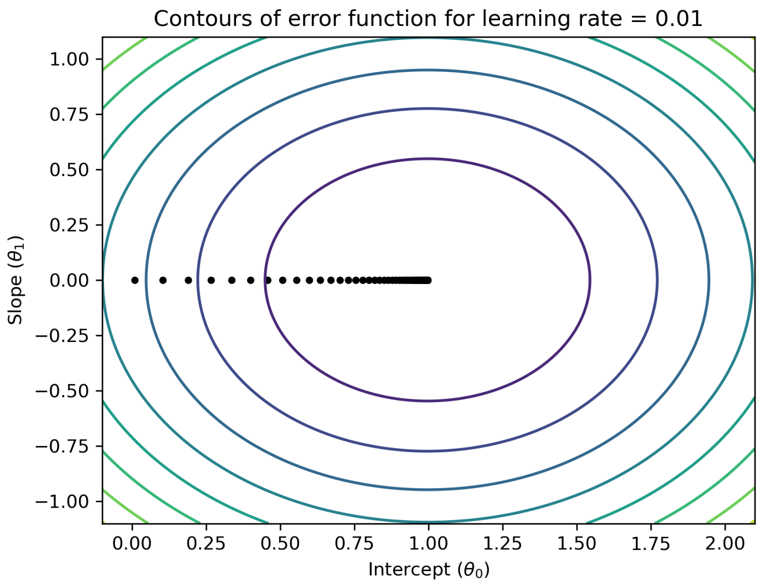
   (b) $h_\theta(x) = \theta_0 + \theta_1 \cdot x_{normalised}$



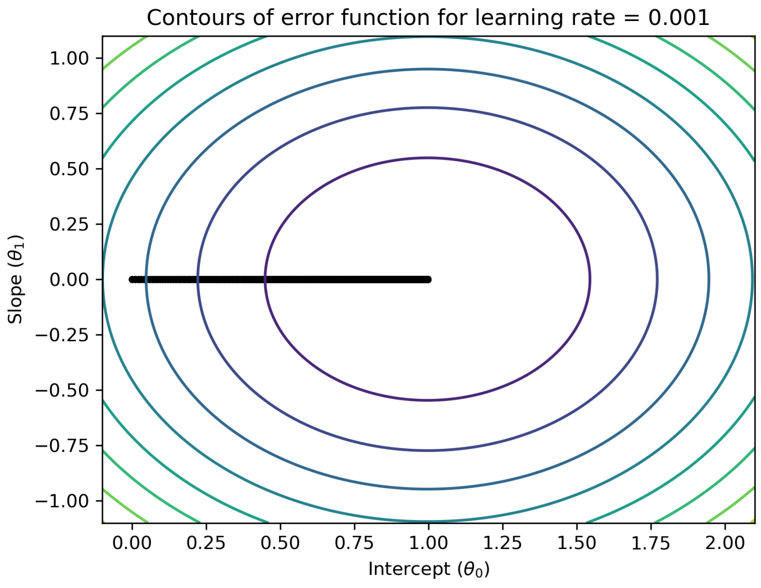
   (c) The black points below represent parameters θ at a given time instant. As seen they gradually slide down the mesh towards the minima



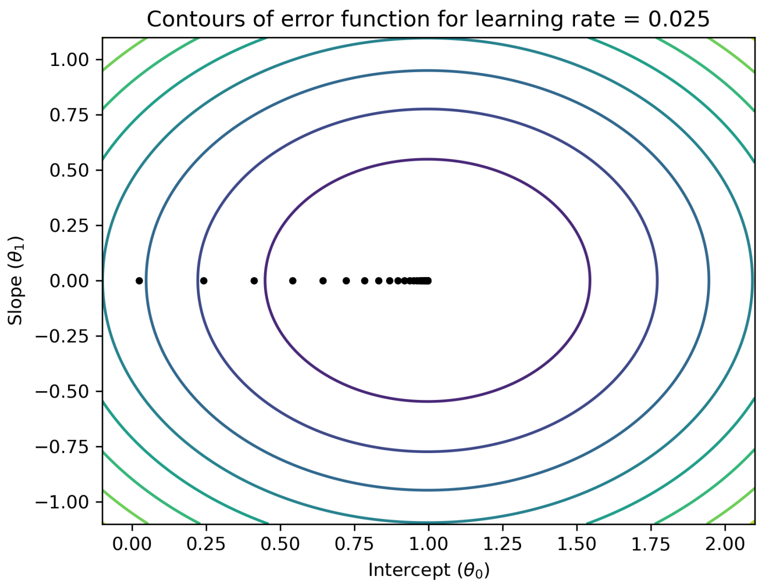
(d) Here the black points represent the value of θ after every 10 iterations. This sampling is done to compare with other learning rates in part (d)

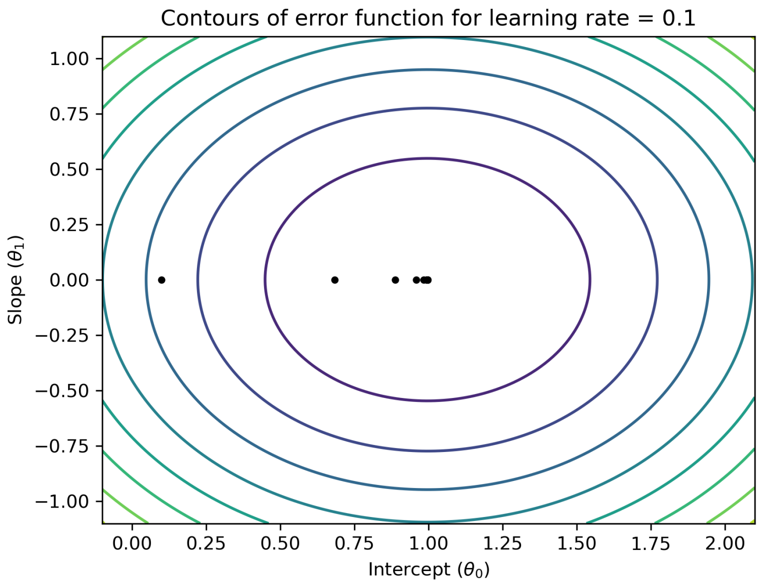


(e) For learning rate = 0.001, the values of θ are very close together after every 10 iterations, indicating that the convergence happens slowly and takes a lot of iterations.



For learning rate = 0.025 and 0.1, the θ values grow further apart, indicating that the convergence happens quicker in fewer iterations.





2. (a) x were sampled in a normal distribution using np.random.normal.
$y = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \epsilon$, where ϵ is the noise also generated using np.random.normal

(b) The convergence criteria used is
For j = 0,1,2 : $|\theta_j^{(t+1)} - \theta_j^{(t)}| \leq \delta$ for k successive iterations
The value of δ and k have been chosen appropriately for different batch sizes
As the batch size becomes smaller, δ has to be increased because θ changes more arbitrarily due to small batch size.

Table 1: Convergence criteria and θ learned for different batch sizes

Batch size	δ	k	θ_0	θ_1	θ_2
1	0.1	20000	3.03223842	0.9832025	2.01763579
100	0.005	10000	2.99781489	0.99809951	1.99941107
10000	0.001	5000	2.92272284	1.01004005	1.99655336
1000000	0.0001	1	2.83468418	1.02000365	1.99312826

(c) The ideal value of θ is (3,1,2). The learned values are very close to the ideal value. The best learning is observed in medium batch sizes as compared to too small or too large batch sizes.
The time taken also in general increases with increase in batch size, due to more time taken per iteration.

Table 2: Time taken and no of iterations till convergence for different batch sizes

Batch size	Time taken (s)	No of iterations
1	0.222	20030
100	0.158	11171
10000	0.6504	5997
1000000	73.283	4732

The error obtained on the given test set for ideal θ and actual learned value of *theta* for different batch sizes is as follows:

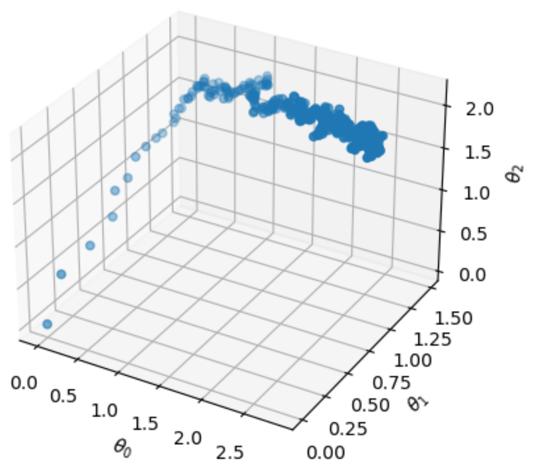
Table 3: Error J for different hypothesis

Hypothesis	Error value
Original, $\theta = (3, 1, 2)$	0.982
r = 1	1.011
r = 100	0.983
r = 10000	0.991
r = 1000000	1.015

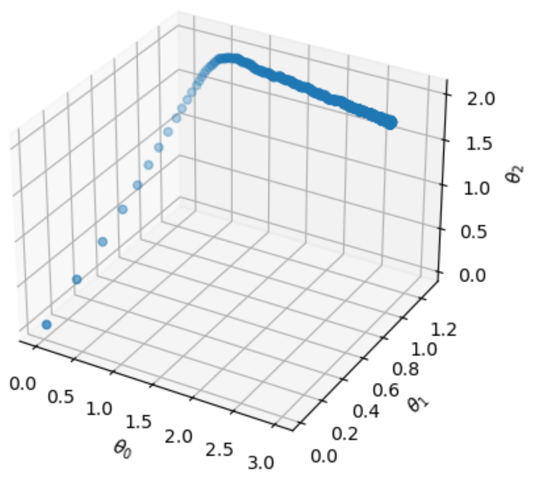
It is observed that values of J for learned hypothesis are very close to the value observed for the original hypothesis.

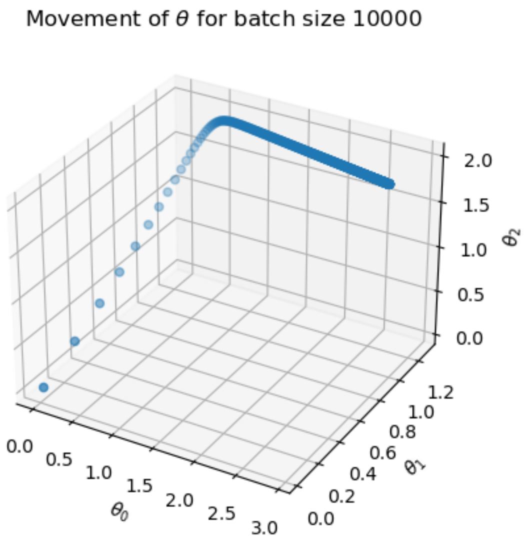
(d) The movement of θ is more "noisy" for smaller batch sizes, and smoothens out for larger batch sizes. This makes sense as in a larger batch size, more points in the dataset would be considered in one iteration.

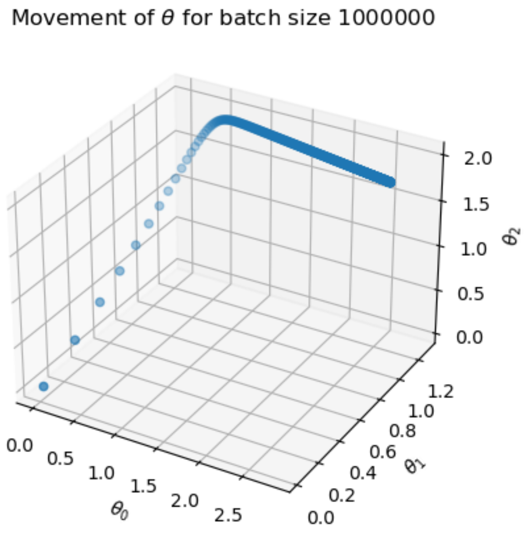
Movement of θ for batch size 1



Movement of θ for batch size 100







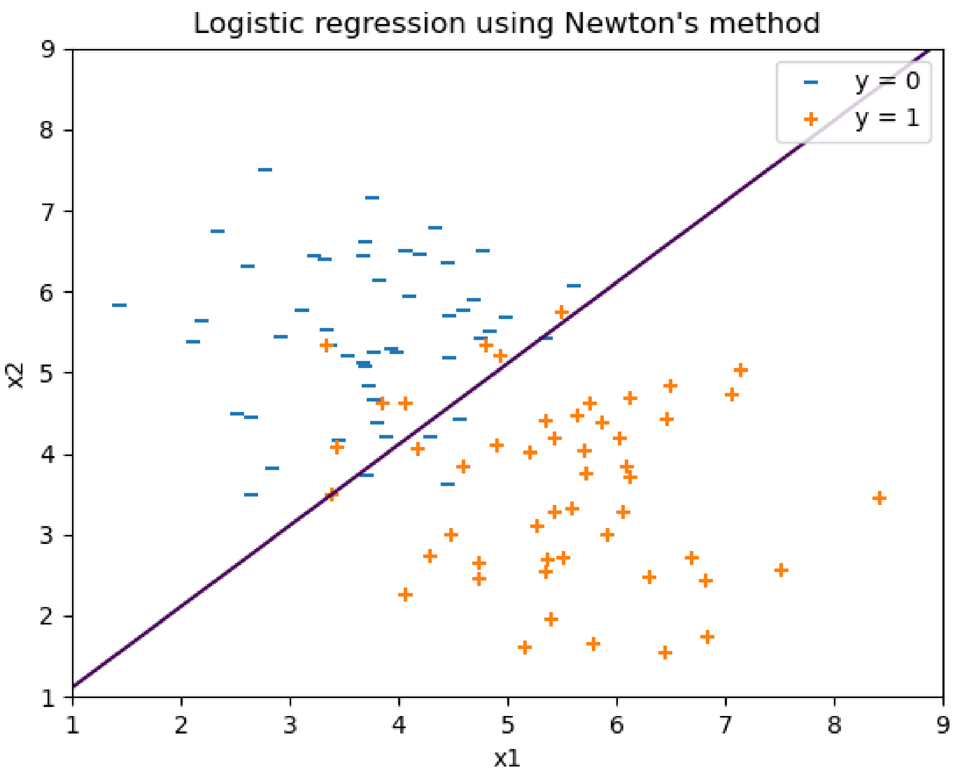
3. (a) In each iteration, the following update is used
$\theta^{(t+1)} = \theta^{(t)} - H^{-1}\ \nabla_\theta\ L(\theta)$
where $\nabla_\theta L(\theta) = -\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$
$H = -\sum_{i=1}^{m} \frac{x_j^{(i)} x_k^{(i)} e^{-\theta^T x^{(i)}}}{(1+e^{-\theta^T x^{(i)}})^2}$

The convergence criteria used is
$\forall\ j \in \{0, 1, 2\}\ \ |\theta_j^{(t+1)} - \theta_j^{(t)}| \leq 0.001$

The θ obtained from logistic regression using Newton's method is (0.40125316, 2.5885477, -2.72558849)

(b) The decision boundary is calculated as follows:
(x_1, x_2) lies on the decision boundary if
$\theta_0 + \theta_1 \cdot (x_1)_{normalised} + \theta_2 \cdot (x_2)_{normalised} = 0$
The boundary was plotted with the help of plt.contour

The plot showing all input data points and the decision boundary is as follows



4. The various parameters $(\phi, \mu_0, \mu_1, \Sigma, \Sigma_0, \Sigma_1)$ were calculated using expressions discussed in class.
The linear decision boundary, given by
$\log(\frac{\phi}{1-\phi}) = \frac{1}{2}(-2x^T\Sigma_1^{-1}\mu_1 + 2x^T\Sigma_0^{-1}\mu_0 + \mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0)$

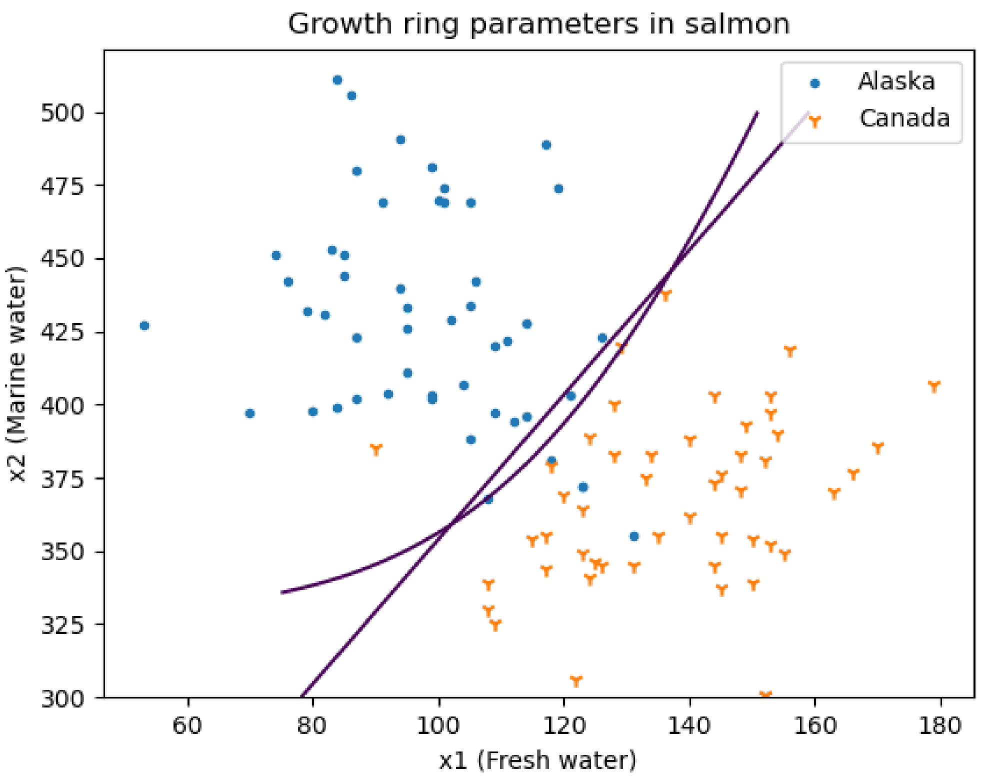
and the quadratic decision boundary, given by
$\log(\frac{\phi}{1-\phi}) + \frac{1}{2}\log(\frac{|\Sigma_0|}{|\Sigma_1|}) = \frac{1}{2}(-(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1))$

were plotted with the help of plt.contour. (x_1, x_2) lies on the decision boundary if the corresponding normalised points satisfy the above equations.

The values of parameters obtained, as well as the decision boundaries are as follows:

Considering $\Sigma_0 = \Sigma_1 = \Sigma$
$\phi = 0.5$
$\mu_0 = (-0.75529433, 0.68509431)$
$\mu_1 = (0.75529433, -0.68509431)$
$$\Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

Considering $\Sigma_0 \neq \Sigma_1$
$$\Sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$
$$\Sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$



The quadratic boundary provides a better separation of the data points than the linear boundary (since 2 additional points of Alaska are correctly classified). This is because, in the case of linear boundary we were making an extra assumption of $\Sigma_0 = \Sigma_1$, whereas for the quadratic case we are making no such assumption and hence it is more powerful.