COL774: Machine Learning
Assignment 1

Devanshi Khatsuriya, 2019CS10344

Note: Refer to README.md for instructions to run code.

1 Linear Regression

(a) Implemented Batch Gradient Descent for optimizing $J(\theta)$ with following parameters:

- Learning Rate: 0.01
- Stopping Criteria: $|J(\theta)^{(t+1)} - J(\theta)^{(t)}| < \delta$, with $\delta = 10^{-11}$, where t is iteration no.

Parameters Learned:

$$\begin{bmatrix}\theta_1 \\ \theta_0\end{bmatrix} = \begin{bmatrix}0.00134015 \\ 0.99658891\end{bmatrix} \tag{1}$$

Here the hypothesis prediction is given by:

$$\hat{y} = \theta_1 x + \theta_0$$

(b) Plot of Hypothesis Function learned on normalized data is shown in Fig. 1.

(c) 3-D mesh showing $J(\theta)$ vs. θ and gradient descent steps are shown in Fig. 2. Animation is present in Folder Q1 (File 3D-descent.mp4).

(d) Contours for $J(\theta)$ and θs achieved in gradient descent are show in Fig. 3. Animation is present in Folder Q1 (File contour-descent.mp4).

(e) Gradient descent animations on $J(\theta)$ contours for $\eta \in \{0.001, 0.025, 0.1\}$ are present in Folder Q1 (Files *l1.mp4, l2.mp4 and l3.mp4* respectively).
Observations. For higher learning rates, θ takes bigger steps to optima and both the number of iterations and the time taken for convergence are lesser.

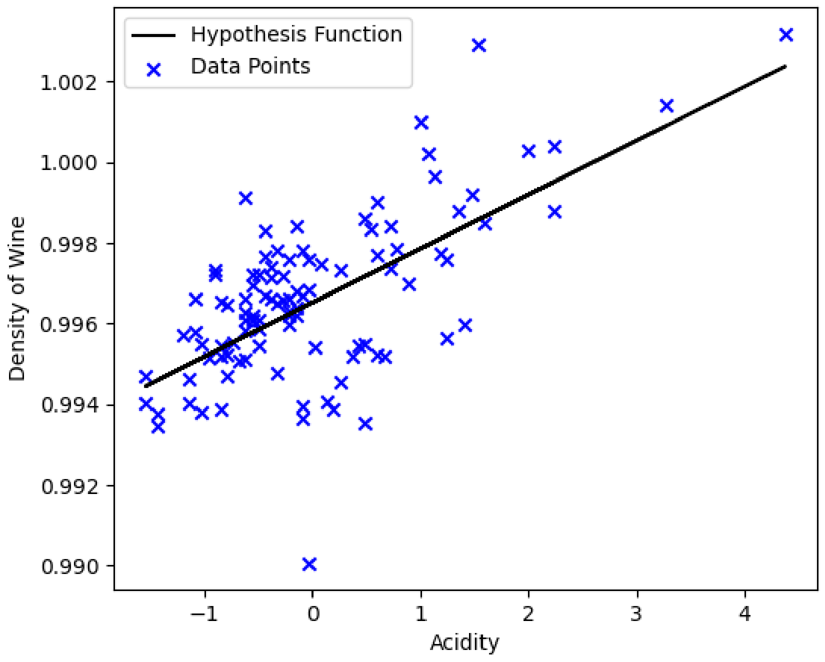

Figure 1: Data and Hypothesis Function

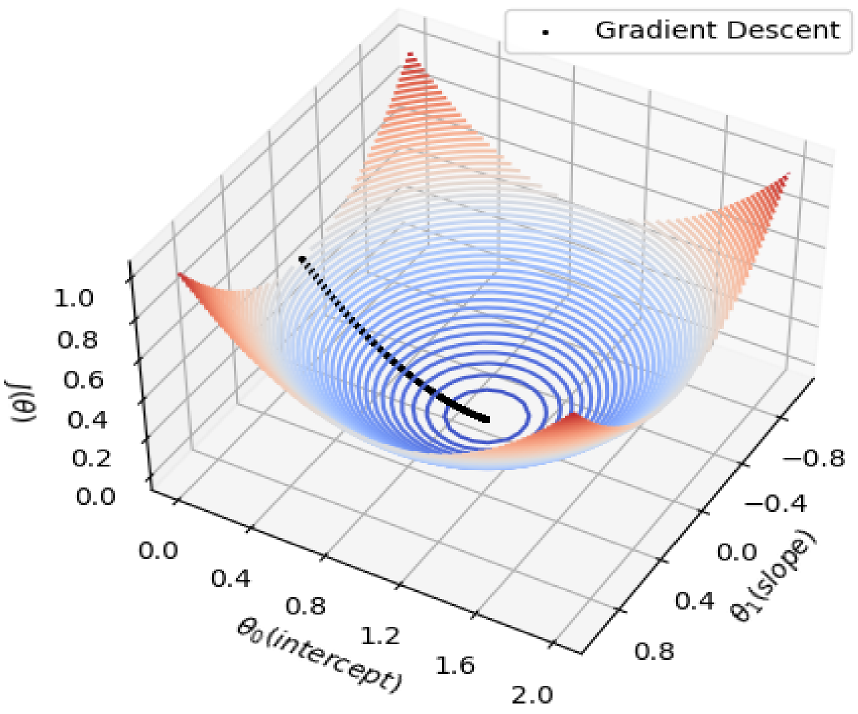

Figure 2: 1(c): $J(\theta)$ vs. θ

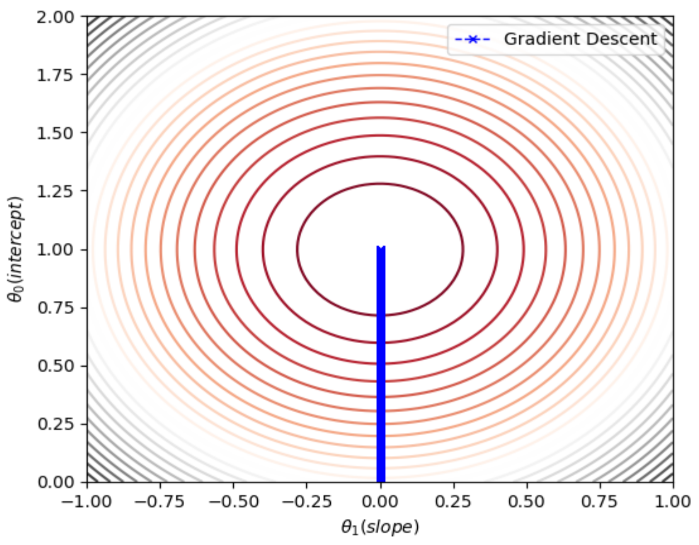

Figure 3: 1(d): $J(\theta)$ vs. θ Contours

2 Sampling and Stochastic Gradient Descent

(a) Sampled 1 million data points taking θ as given vector, $x_0^{(i)} = 1$, $x_1^{(i)}$ sampled from $\mathcal{N}(3, 4)$, $x_2^{(i)}$ sampled from $\mathcal{N}(-1, 4)$ independently, and generated $y^{(i)}$ from $\mathcal{N}(\theta^T x^{(i)}, \sigma^2 = 2)$ for each i.

(b) Implemented Stochastic Gradient Descent for optimizing $J(\theta)$ for generated data. Used batch sizes r from {1, 100, 10000, 1000000}.
Convergence Criteria. Converge if difference of average loss of last k iterations and average loss of k iterations before it has become less than δ. This condition is checked whenever iteration number becomes a multiple of k.
The values used for k and δ for different batch sizes are shown in Table 1.

Batch Size	k	δ
1	1000	10^{-3}
100	1000	10^{-3}
10000	100	10^{-4}
1000000	10	10^{-4}

Table 1: Values for k and δ for different Batch Sizes

(c) Refer to Table 2 for values.

Batch Size	Convergence Time (s)	No. of Iterations	Error on Test Set
1	0.303	6000	1.8161
100	0.735	14000	0.9917
10000	4.484	11900	1.0151
1000000	103.772	7550	1.3431

Table 2: Comparison of SGD for different Batch Sizes

- **Learned Parameters.** All parameters move towards original θ. We see the best fit obtained in case of batch size = 100. This is because of the given settings for k and δ.

$$\theta_{[1]} = \begin{bmatrix} 2.4867 \\ 1.1239 \\ 2.0049 \end{bmatrix} \quad \theta_{[100]} = \begin{bmatrix} 2.9423 \\ 1.0092 \\ 1.9924 \end{bmatrix} \quad \theta_{[10000]} = \begin{bmatrix} 2.8940 \\ 1.0232 \\ 1.9932 \end{bmatrix} \quad \theta_{[1000000]} = \begin{bmatrix} 2.6463 \\ 1.0773 \\ 1.9751 \end{bmatrix} \tag{2}$$

- **Speed of Convergence.** Batch sizes 1 and 100 converge faster. Batch size 1000000 takes considerably longer to converge, which is due to the large matrix multiplications involved.
- **No. of Iterations.** Batch size 1000000 converges in lesser iterations because it takes steps exactly towards the optima. Even though iterations are lesser, it takes more time as its each iteration is much costlier in time.
- **Error on Test Set.** The error on test set by the original hypothesis is 0.9829. Batch size 100 gives the least error which is 0.9917. This is because given the hyperparameter settings, it is the able to reach closest to the optima compared to other batch sizes. This might be because its steps are not as zig-zag as batch size 1, and its iterations are not as costly as batch size 1000000.

(d) Parameter Updation Animations for different Batch Sizes are present in Folder Q2 under *b1.mp4, b100.pm4, b10000.mp4 and b1000000.mp4.*
Inferences. The smaller the batch sizes, the more zig-zag is the motion of θ with iterations. This is because in each iteration, θ is updated in the direction to optimize the loss on a particular batch, and batches differ in their optimal θ. But θ converges to the optima for loss on the whole set because we optimize for all batches overall.

3 Logistic Regression

(a) Implemented Newton's Method for maximizing log-likelihood $LL(\theta)$.
Parameters Learned:

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 0.40125316 \\ 2.5885477 \\ -2.72558849 \end{bmatrix} \tag{3}$$

Here θ_0 is the intercept term.

(b) Decision Boundary is given by:

$$h(x) = 0.5 \implies \frac{1}{1 + e^{\theta^T x}} = \frac{1}{2} \implies \theta^T x = 0$$

Plot of Data Points and Decision Boundary in Fig. 4.

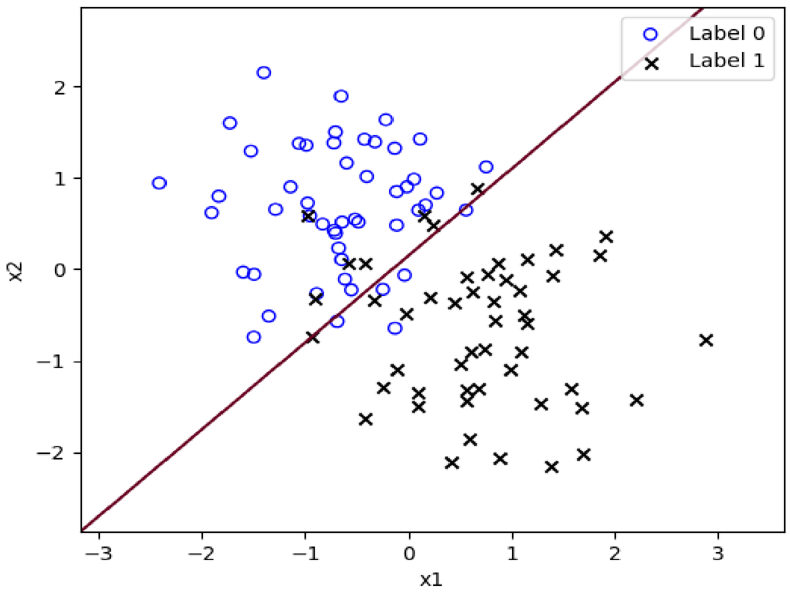

Figure 4: 3(b): Data Points and Decision Boundary

4 Gaussian Discrmimant Analysis

(a) Parameters under the assumption of same co-variance matrices ($\Sigma_0 = \Sigma_1 = \Sigma$):

Label 0 represents class Alaska, Label 1 represents class Canada

$$\phi = 0.5 \tag{4}$$

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix} \tag{5}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix} \tag{6}$$

$$\Sigma_0 = \Sigma_1 = \Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix} \tag{7}$$

(b) Plot of Data Points in Fig. 5.

(c) Equation for Linear Decision Boundary:

$$\mathbb{P}(y = 1|x; \theta) = \tfrac{1}{2}, \text{ which simplifies to}$$

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

Its plot is shown in Fig. 6.

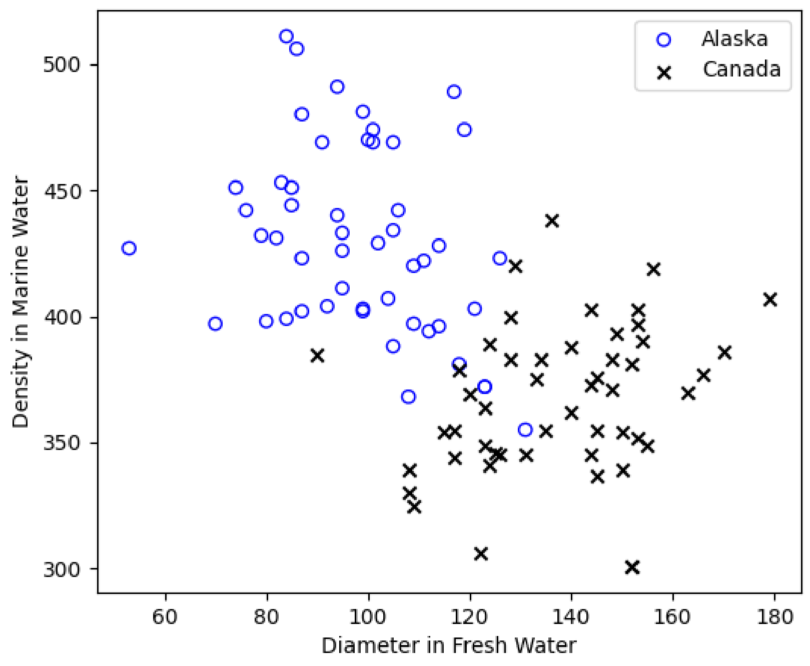

Figure 5: 4(b): Data Points for Q4

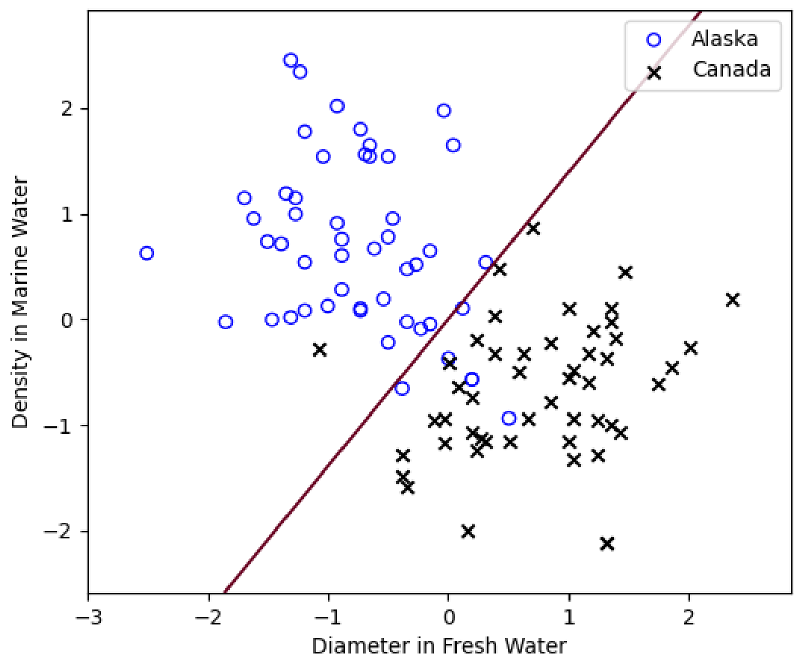

Figure 6: 4(c): Normalized Data and Linear Boundary

(d) Parameters under general setting of GDA:

$$\phi,\ \mu_0 \text{ and } \mu_1 \text{ remain the same as in 4(a)}$$

$$\phi = 0.5 \tag{8}$$

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix} \tag{9}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix} \tag{10}$$

$$\Sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix} \tag{11}$$

$$\Sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix} \tag{12}$$

(e) Equation for Quadratic Decision Boundary:

$$\mathbb{P}(y = 1|x;\theta) = \tfrac{1}{2}, \text{ which simplifies to}$$

$$\frac{1}{2}\left((x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\right) + \log\left[\left(\frac{1-\phi}{\phi}\right)\frac{|\Sigma_0|^{\frac{1}{2}}}{|\Sigma_1|^{\frac{1}{2}}}\right] = 0$$

Its plot is shown in Fig. 7.

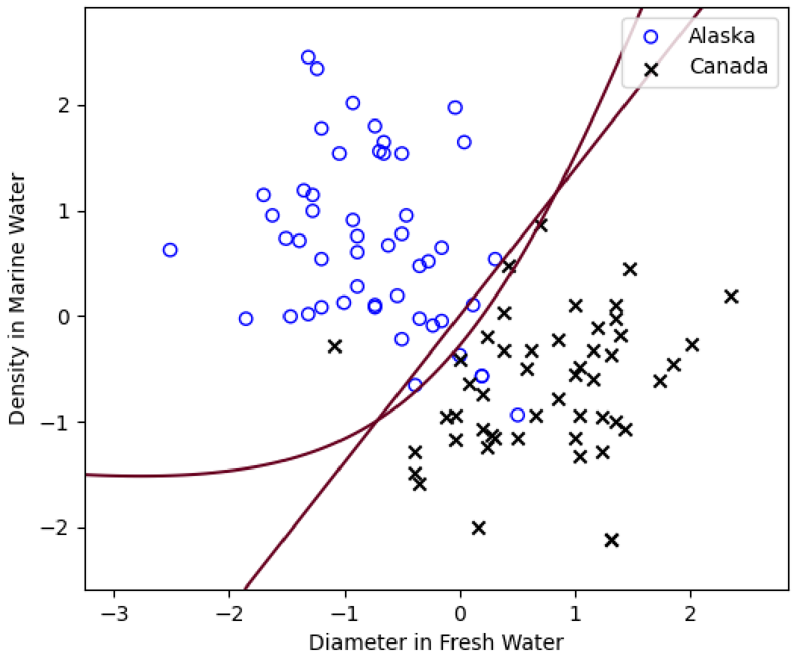

Figure 7: 4(e): Linear and Quadratic Boundaries

(f) **Observations.** The Quadratic Boundary is able to properly separate the few blue points near the boundary that were mis-classified by the Linear Boundary. Intuitively, it curves upwards to correctly classify the blue points that lie among black points.