COL774 Machine Learning Assignment 1

Abhinav Rajesh Shripad **Entry Number: 2022CS11596**

February 13, 2025

Solution 1.1(b) and 1.1(c)

Learning rate: After testing various learning rates in the range of 0.0001 to 0.1 with increments of 0.0001, we found that the optimal learning rate is 0.018. This value balances the speed and stability of the convergence, ensuring that the cost function $J(\theta)$ decreases consistently without oscillations or divergence.

Stopping Criteria: For the stopping criteria, we monitor the change in the cost function $J(\theta)$ between consecutive iterations. The optimization process stops when:

$$\mid J(\theta^{(t)}) - J(\theta^{(t-1)}) \mid < \epsilon \quad \textbf{or} \quad t > 10^5$$

where $\epsilon = 10^{-7}$ or we stop when the number of iterations exceeds 10^5

Parameters: The parameters learned are

$$\theta_0 = 6.21852703, \quad \theta_1 = 29.0640419$$

Solution 1.2 The following plot illustrates the hypothesis function in red and data for Q1.

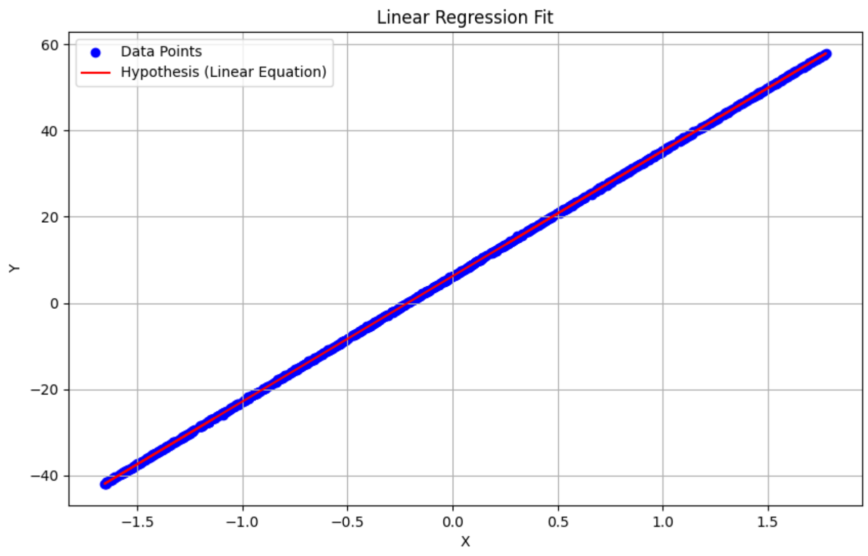


Figure 1: Hypothesis function and data for Q1

Solution 1.3(a) and 1.3(b)

The following plot illustrates the change in parameters with each iteration with loss on z axis.

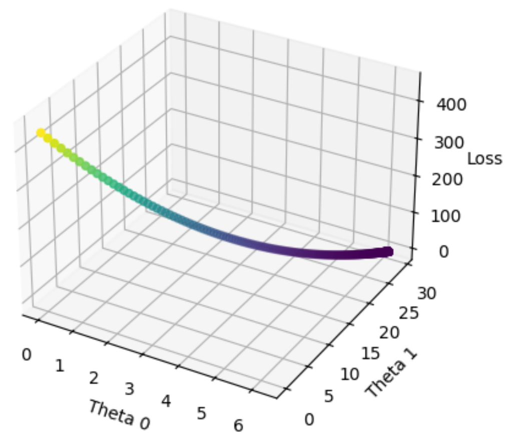


Figure 2: Parameter Space

Solution 1.4

The following plot illustrates the change in parameters with each iteration with loss marked in Heat Map.

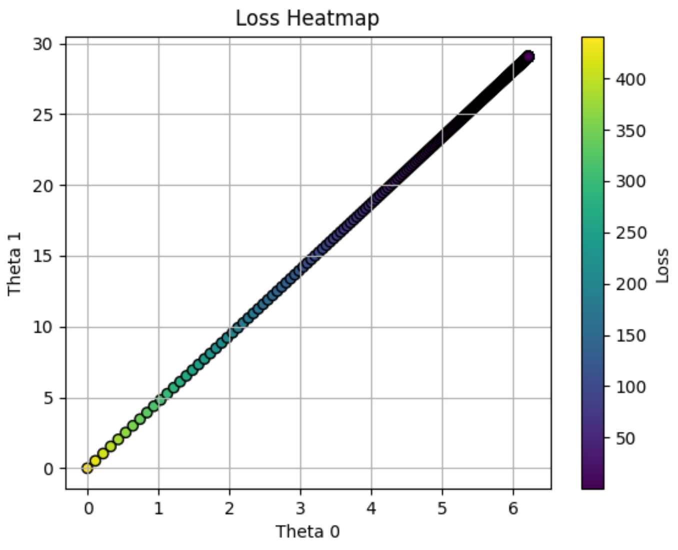
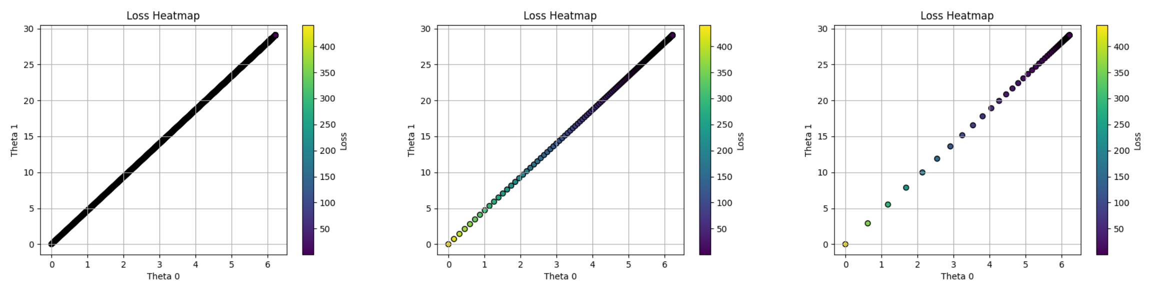


Figure 3: Heat Map

Solution 1.5

The following plot illustrates the change in parameters with each iteration with loss marked in Heat Map.



$\eta = 0.001$ $\eta = 0.025$ $\eta = 0.1$

Figure 4: Heat Map - Three Views

We observe the following behavior when using different learning rates for the contour plots at each learning iteration:

A. $\eta = 0.001$ (small learning rate): The change in the parameters is **very slow**, and the algorithm takes a **long time to converge**. The contours of the cost function show very gradual progress in the optimization process, with small steps between iterations. The model moves slowly toward the minimum, which may result in longer training times but **ensures stability**.

B. $\eta = 0.025$ (moderate learning rate): The learning rate provides a **good balance between speed and stability**. The contours show more noticeable movement toward the minimum with each iteration, resulting in a **faster convergence** compared to $\eta = 0.001$.

C. $\eta = 0.1$ (large learning rate): The parameters make large updates at each step, causing the optimization process to move quickly toward the minimum. However, the contours of the cost function show that the **model overshoots the optimal point and oscillates around it**. The points in the parameter space are more spread out, indicating that the **model may struggle to converge precisely**, and there is a risk of divergence or instability if the learning rate is too large.

Conclusion: Smaller learning rates (e.g., $\eta = 0.001$) lead to slow convergence, requiring more iterations to reach the minimum. Larger learning rates (e.g., $\eta = 0.1$) cause rapid progress but can lead to instability and overshooting of the optimal point. **The choice of learning rate is crucial to balancing convergence speed and stability in the optimization process.**

Solution 2.2(b),2.2(c) and 2.4

The parameters for different batch sizes are

Batch Size	**Iterations**	θ_0	θ_1	θ_2	**Train MSE**	**Test MSE**
1	10000	3.08058192	1.01991851	2.05185201	2.040857	2.080364
80	1220	2.99810	1.00391	2.00083	1.99909	1.99493
8000	864	2.99523	1.00025	2.00032	2.00313	2.00909
800000	8543	2.986842	1.00160	1.00160	1.99792	1.99729

Convergence Criteria: The convergence criteria for the SGD algorithm are as follows:

A. Maximum Iterations: Set to 10^4 iterations to prevent infinite loops and ensure sufficient computation time.

B. Cost Function Change Threshold: Stop when the change in the cost function is smaller than $\epsilon = 10^{-7}$, ensuring convergence when improvements become negligible.

These criteria ensure efficient and stable convergence for each batch size, balancing accuracy and computational efficiency. The threshold ensures that small updates are ignored, while the iteration limit ensures the algorithm completes in a reasonable time.

Solution 2.3(a) We observe that the parameters learned across different batch sizes are very close to each other. This suggests that the choice of batch size does not significantly impact the final learned parameters in terms of accuracy.

If we compare the Test Mean Squared Error (MSE) across different batch sizes, we find that the values are quite similar, meaning that no particular batch size clearly outperforms the others in terms of predictive performance. However, the training process differs in terms of computational efficiency.

Smaller batch sizes generally require more iterations to reach convergence since they take smaller gradient steps per update. On the other hand, larger batch sizes converge in fewer iterations but may require more time per iteration due to the increased computational cost per update.

Thus, while the final Test MSE remains nearly the same, the trade-off between batch size, number of iterations, and overall training time must be considered when choosing an optimal batch size for training.

Based on the observations above, batch size, $r = 8000$ is the fastest to converge.

Solution 2.3(b)

Parameters Learned by solving (1) is

$$\theta_0 = 2.99910402, \quad \theta_1 = 1.00008706, \quad \theta_2 = 1.99975462$$

Time Taken for this is 0.010642766952514648 seconds. For a X with dimensions $n \times m$, the time complexity to compute (1) is $O(nm^2 + m^3)$. Here $n = 0.8 \times 1e6$ and $m = 2$, Thus we can ignore m in comparison of n. Thus here the computation is very fast, but this method is not scalable when we much larger data sets. Thus gradient descent and its variations are preferred.

Problem 2.3(c) Compare the true parameters (from which we sampled), the parameters learned in closed form, and the parameters learnt by SGD. Write what you observe.

Solution Comparison of True Parameters, Closed-Form Parameters, and SGD-Learned Parameters:

A. **True Parameters** These are the actual values used to generate the training data, representing the "ground truth."

B. **Closed-Form Parameters**: Computed using the formula $\theta = (X^T X)^{-1} X^T y$, the closed-form solution is exact under ideal conditions (with no noise and well-conditioned data), and it converges quickly but can become computationally expensive with large datasets.

C. **SGD-Learned Parameters**: Stochastic Gradient Descent (SGD) iteratively adjusts θ based on the gradient of the cost function. While it can handle large datasets efficiently and is more robust to noise, it may take longer to converge and may slightly differ from the true parameters depending on the learning rate and number of iterations.

Observations:

A. Closed-Form Solution: Generally produces parameters very close to the true values under conditions of small data size and least count of system is negligible in comparison to the real parameters.

B. SGD: The learned parameters may slightly deviate from the true ones, especially with improper tuning, but can approximate the true parameters well as training progresses.

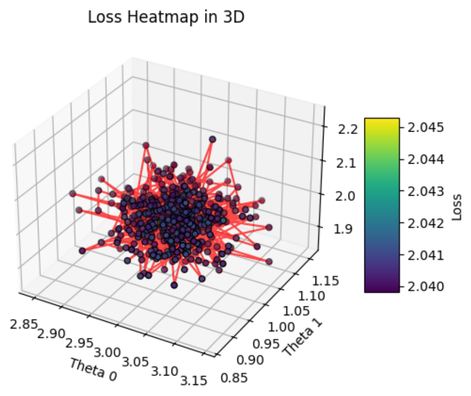
Comparison:

A. The closed-form solution is exact for small datasets, while SGD is more scalable and robust for large or noisy datasets, though it may require more iterations to converge.

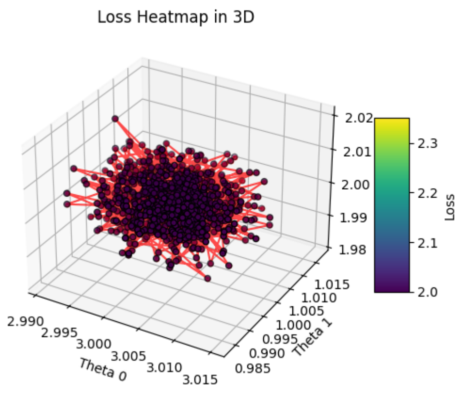
B. The closed-form solution provides accurate parameters for smaller datasets, while SGD offers a more flexible approach for large datasets, with slight deviations depending on tuning.

Solution 2.5

The following plot illustrates the change in parameters with each iteration with loss marked in Heat Map.

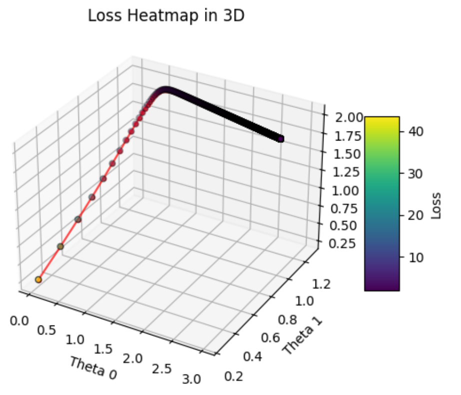


$r = 1$

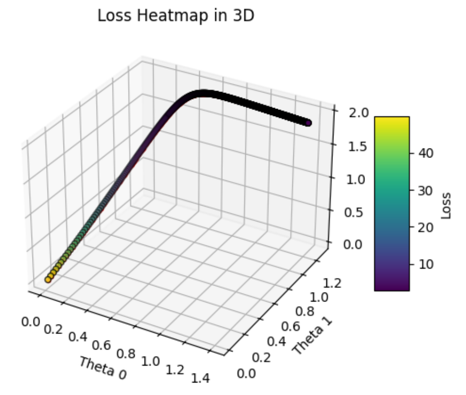


$r = 80$

Figure 5: Heat Map - Three Views



$r = 8,000$



$r = 80,000$

Figure 6: Moment of θ in 3D Parameter Space for different batch size

When plotting the movement of θ in 3D space ($\theta_0, \theta_1, \theta_2$ as axes), we observe distinct patterns based on batch size:

A. Small Batch Size (e.g., 1, 80): The movement of θ is highly irregular and noisy due to the high variance in gradient updates. The path is zig-zagging before convergence.

B. Medium Batch Size (e.g., 8000): The trajectory smoothens as more data is used in each update. The path is more direct but still retains some randomness.

C. Large Batch Size (e.g., 800000): The trajectory follows a nearly straight-line path towards convergence, resembling the closed-form solution. There is minimal noise in updates.

Intuition

A. Small batches lead to high variance in updates, making the trajectory noisy but allowing exploration of different paths.

B. Larger batches reduce variance, making updates more stable and direct, converging faster in fewer steps.

C. Extremely large batch sizes resemble the behavior of batch gradient descent, following a smooth trajectory directly towards the optimal solution.

This aligns with intuition, **smaller batches approximate the true gradient with noise, while larger batches provide more accurate updates, leading to smoother and more predictable movement.**

Solution 3.1

The parameters learned are

$$\theta_0 = 0.40125316 \quad \theta_1 = 2.58854772 \quad \theta_2 = -2.72558851$$

The normalization constants are

$$\mathbf{E}[X_1] = 4.618717 \quad \mathbf{E}[X_2] = 4.522868$$

$$\mathbf{Std}[X_1] = 1.31892756 \quad \mathbf{Std}[X_2] = 1.38716591$$

Solution 3.2

The following plot illustrates the given data and the separating line.

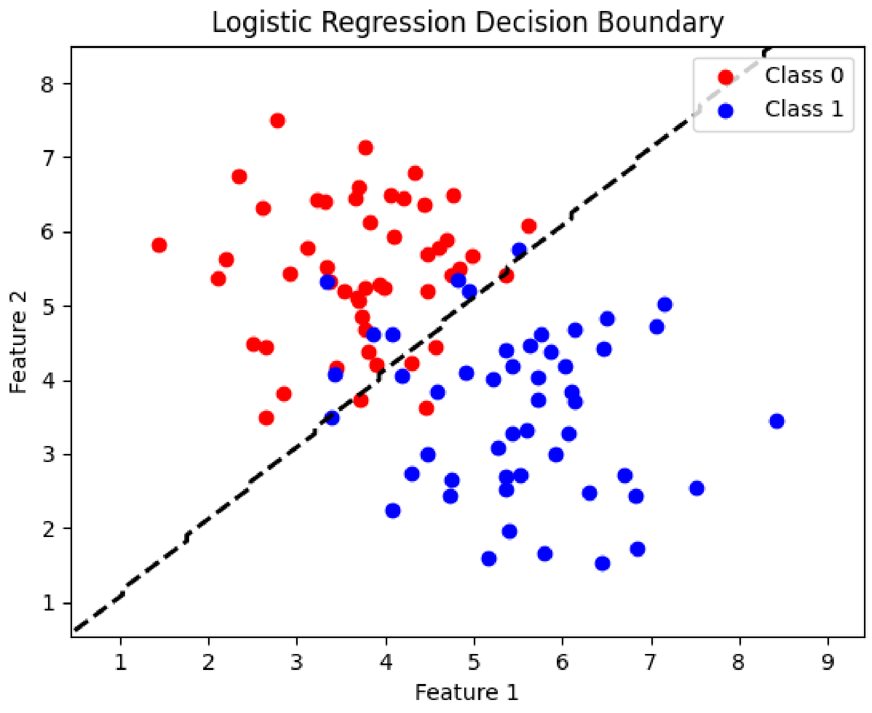


Figure 7: Separating Line and data for Q3

The equation for the separating line is $\theta_0 + \theta_1 \frac{(x_1-\mu_1)}{\sigma_1} + \theta_2 \frac{(x_2-\mu_2)}{\sigma_2} = 0$, by putting the values we get

$$y = 0.998x + 0.116$$

Solution 4.1

Mean Vector and the Covariance Matrix are given as below.

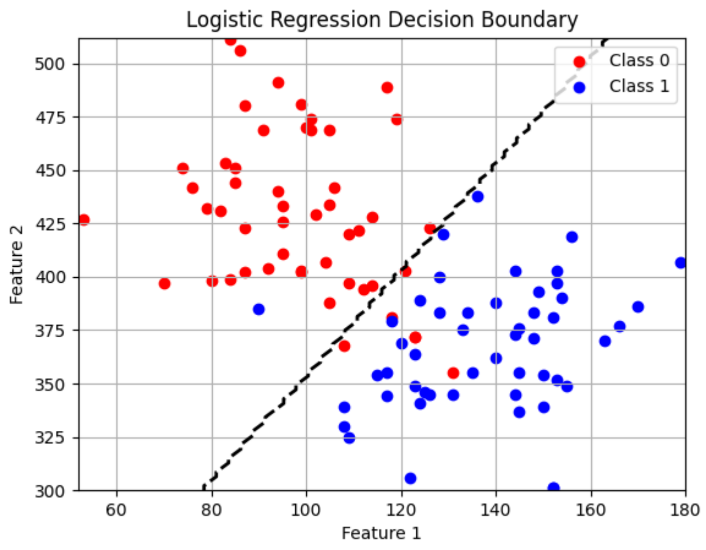
$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}, \quad \mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

The normalization constants are

$$\mathbf{E}[X_1] = 117.92 \quad \mathbf{E}[X_2] = 398.14$$

$$\mathbf{Std}[X_1] = 25.87070931 \quad \mathbf{Std}[X_2] = 46.0082644$$

Solution 4.2 and 4.3 The following plot illustrates the data for q4, with Alaska as 0 and Canada as 1



Derivation of the equation is given with **Problem 4.5**

Solution 4.4 Mean Vector and the Covariance Matrix are given as below.

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}, \quad \mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix},$$

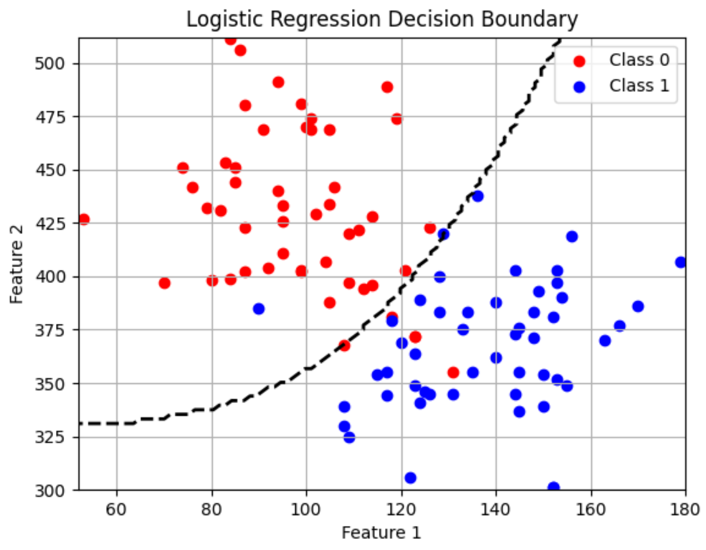
$$\sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix} \quad \sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

The normalization constants are

$$\mathbf{E}[X_1] = 117.92 \quad \mathbf{E}[X_2] = 398.14$$

$$\mathbf{Std}[X_1] = 25.87070931 \quad \mathbf{Std}[X_2] = 46.0082644$$

Solution 4.5 The following plot illustrates the data for q4, with Alaska as 0 and Canada as 1



We now derive the equation for the separating boundary for both **4.3 and 4.5** and then take the special assumption for **4.3**.

At the separating boundary we know that probability of belonging to any class is same. d is the size of the covariance matrix, Here d = 2

$$
\begin{aligned}
& p(y=0|x) = p(y=1|x) && (1)\\
\implies & \frac{p(y=0,x)}{p(x)} = \frac{p(y=1,x)}{p(x)} && (2)\\
\implies & p(x|y=0)p(y=0) = p(x|y=1)p(y=1) && (3)\\
\implies & \frac{1}{\sqrt{2\pi}|\Sigma_0|^{d/2}} \exp\Big(-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\Big)(1-\phi) = \frac{1}{\sqrt{2\pi}|\Sigma_1|^{d/2}} \exp\Big(-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\Big)(\phi) && (4)\\
\implies & \exp\Big(\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\Big) = \frac{\phi}{1-\phi}\frac{|\Sigma_0|^{d/2}}{|\Sigma_1|^{d/2}} && (5)\\
\implies & \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) = 2\log(\frac{\phi}{1-\phi}) + d\log(\frac{|\Sigma_0|}{|\Sigma_1|}) && (6)
\end{aligned}
$$

Simplifying the LHS of (6) we get it is equal to

$$x^T\Big(\Sigma_1^{-1} - \Sigma_0^{-1}\Big)x - 2\Big(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1}\Big)x + \Big(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0\Big) \tag{7}$$

Deriving 4.3 Since we assume that $\Sigma_0 = \Sigma_1 = \Sigma$, and $d = 2$ we get that the first term from (7) is 0, and thus the resulting separating equation is

$$\Big(\mu_1^T - \mu_0^T\Big)\Sigma^{-1}x = \frac{1}{2}\Big(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\Big) + \log(\frac{\phi}{1-\phi}) \tag{8}$$

Thus the above equation can be compared to $w^Tx + b = 0$, where $w = \Sigma^{-1}(\mu_1 - \mu_0)$ and $b = -\frac{1}{2}\Big(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\Big) - \log(\frac{\phi}{1-\phi})$, thus the linear separator is clearly a straight curve.

Putting in the values, and remembering that we Normalized the data, we get the equation to be

$$y = 2.471x + 107.216$$

Deriving 4.5 Since here we have no assumption, we can directly write the equation as $(d = 2)$ to give

$$x^T\Big(\Sigma_1^{-1} - \Sigma_0^{-1}\Big)x - 2\Big(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1}\Big)x + \Big(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0\Big) = 2\log(\frac{\phi}{1-\phi}) + 2\log(\frac{|\Sigma_0|}{|\Sigma_1|}) \tag{9}$$

Since the above equation can be compared with $x^TAx + 2b^Tx + c = 0$ where $A = \Sigma_1^{-1} - \Sigma_0^{-1}$, $b = -\Big(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1}\Big)$ and $c = -2\log(\frac{\phi}{1-\phi}) - 2\log\Big(\frac{|\Sigma_0|}{|\Sigma_1|}\Big)$,thus the separator is clearly a quadratic curve. Putting in the values, and remembering that we Normalized the data, we get the equation to be

$$0.671(f(x))^2 - 0.865(g(y))^2 + 2.573f(x)g(y) + 7.615f(x) - 5.719g(y) = 0.983$$

where $f(x) = \frac{(x-117.92)}{25.87070931}$ and $g(y) = \frac{(y-398.14)}{46.00826447}$

Solution 4.6

Case 1: Same Covariance for Both Classes (Linear Boundary)

A. When we assume that both classes share the same covariance structure, the decision boundary is a straight line (or a hyperplane in higher dimensions).

B. The boundary perfectly bisects the space between the two class means, adjusted for prior probabilities.

C. The orientation of the line is determined by the difference in mean vectors of the two classes.

D. This works well when the two classes are well-separated and have a similar spread.

E. If the distributions overlap significantly, a linear boundary may fail to capture the true separation, leading to misclassifications.

F. In real-world data, assuming equal covariances might be overly simplistic, making the model less adaptable to variations in spread and orientation.

Case 2: Different Covariances for Each Class (Quadratic Boundary)

A. When we allow each class to have its own covariance matrix, the decision boundary becomes curved or nonlinear.

B. The boundary adapts to the shape of the data distribution, making it more flexible.

C. If one class has a much larger variance than the other, the boundary bends toward the class with higher spread, reflecting the fact that points far from the mean are still likely to belong to that class.

D. The shape of the boundary can vary. If covariance matrices are full (with correlations between features), the boundary can take more complex shapes, potentially resembling an ellipse or hyperbola.

E. This flexibility comes at a cost of More parameters to estimate, which requires more data for accurate learning, and Higher risk of over fitting if the dataset is small.

Comparison and Practical Implications

A. Linear boundaries are computationally efficient and work well in cases where class distributions are similar in spread. They are often preferred in high-dimensional spaces where quadratic boundaries might lead to over fitting.

B. Quadratic boundaries offer better performance when class distributions have different orientations or variances, making them useful for complex classification problems. However, they require more data to estimate the additional parameters reliably.

If the data is well-structured and relatively homogeneous in variance, a linear decision boundary is often sufficient. But when dealing with more complex distributions, allowing for different covariances significantly improves classification performance.