COL 774: Assignment 1 Report

Disha (2022CS11118)

Linear Regression

Part 1. Observations

Learning Rate	Theta 0	Theta 1	Iterations	Cost
0.001	6.2180	29.0617	9154	0.004880
0.01	6.218475	29.063798	1028	0.004875
0.025	6.218600	29.064200	428	0.004875
0.05	6.2186	29.0644	219	0.004875
0.1	6.218600	29.064500	111	0.004875

Table 1: Effect of Learning Rate on Convergence in Linear Regression

Conclusions

- Smaller learning rates require more iterations to converge (0.0001 took $80,034$; 0.001 took $9,154$).
- Learning rates 0.025, 0.05, and 0.1 converge efficiently with similar final costs (~ 0.004875).
- Large learning rates (0.5, 1.0) converge very quickly, but too high a rate (e.g., 2.0) causes divergence.
- Final parameters for 0.025 to 1.0 are close to optimal.
- The optimal learning rate is between 0.05 and 0.1 for this dataset.
- Very high learning rates risk instability, and very low rates are too slow.

Final Parameters: $\theta_0 = 6.2186$, $\theta_1 = 29.0645$.
Stopping Criteria: Absolute cost difference $< 10^{-8}$
Optimal Learning Rate: 0.05 to 0.1

Part b. Data and Hypothesis

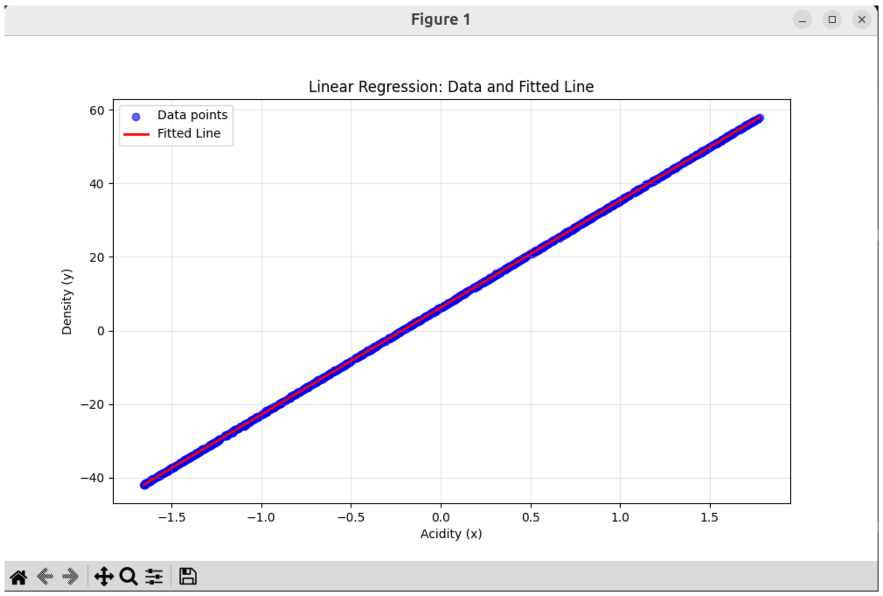


Figure 1: Wine Density vs. Acidity: Data points and hypothesis

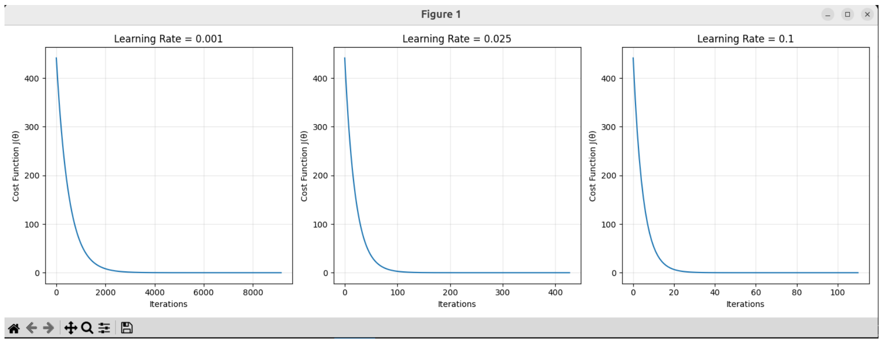


Figure 2: Loss vs Iterations

Part c. Error Surface and Gradient Descent Path

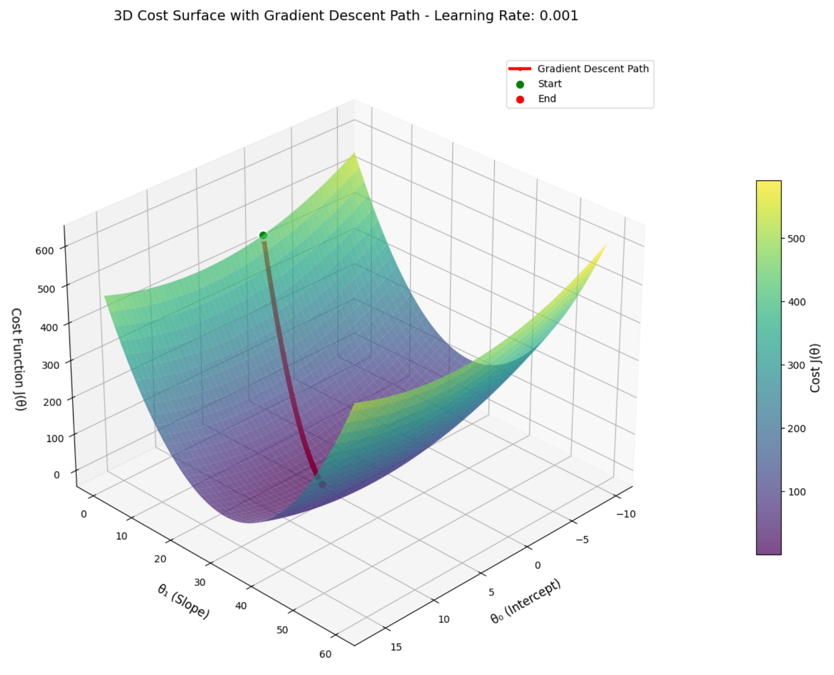


Figure 3: Surface Plot - Learning Rate: 0.0001
3D surface with error $J(\theta)$ on the z-axis. The gradient descent path is shown in red moving toward the minima.

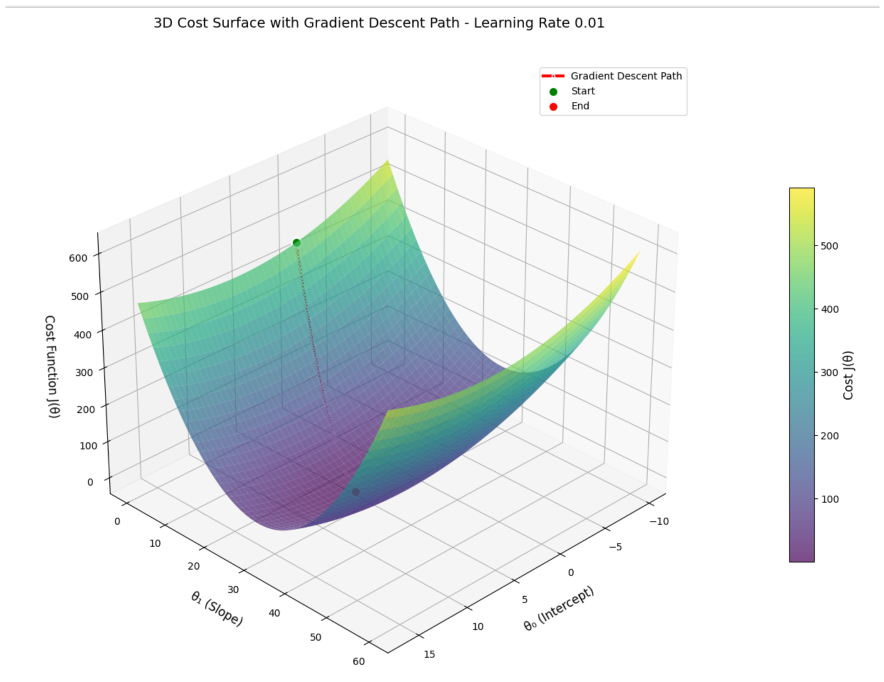


Figure 4: Surface Plot - Learning Rate: 0.01

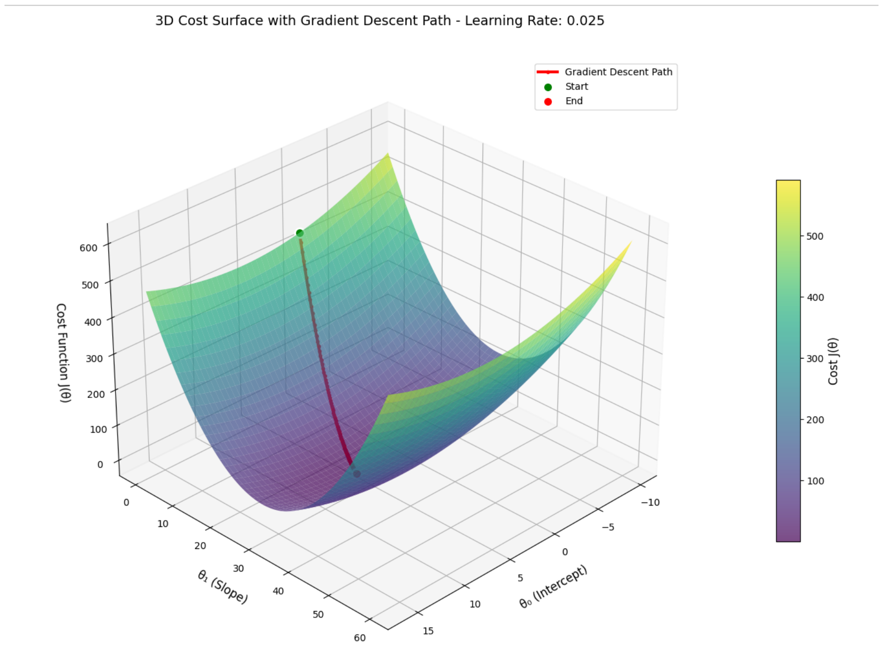


Figure 5: Surface Plot - Learning Rate: 0.025

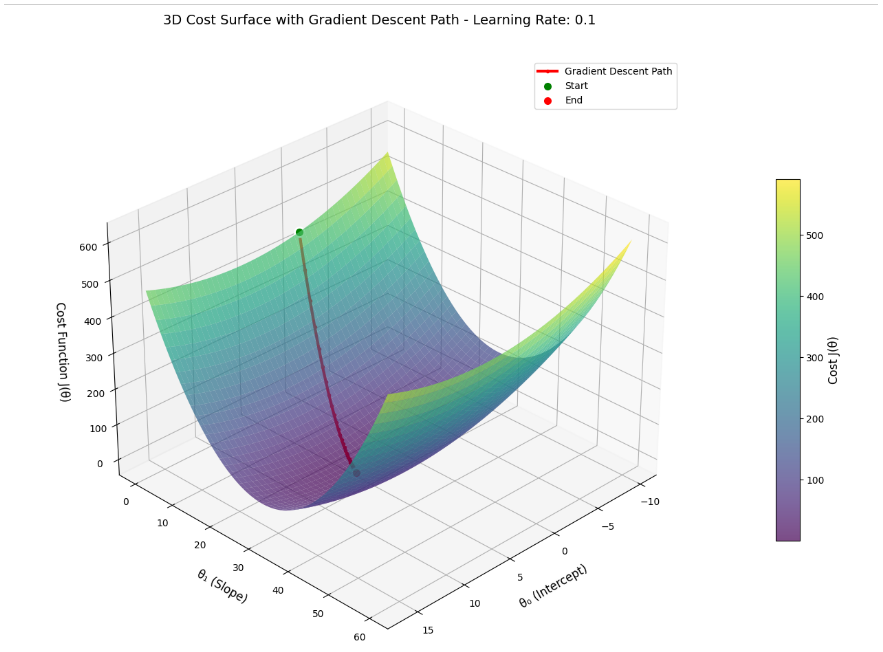


Figure 6: Surface Plot - Learning Rate: 0.1

Part d. Contour Plots

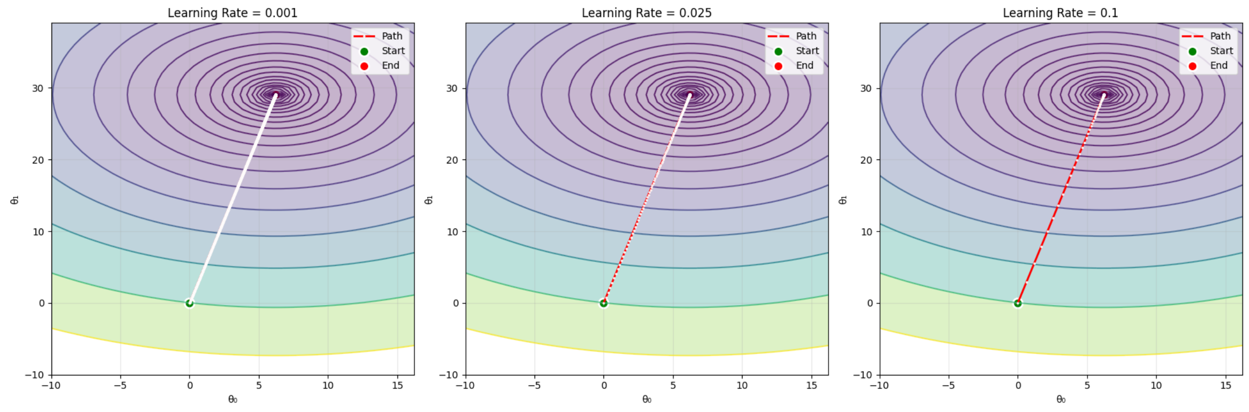


Figure 7: Contour PlotGradient steps are depicted in white trace.

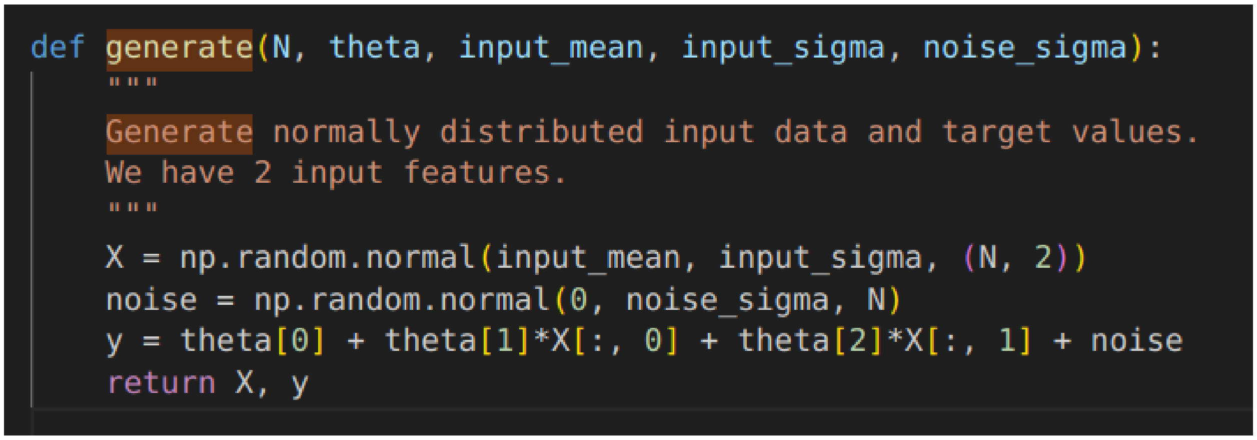
Part e. Learning Rate Comparison

Observations:

- Gradient steps are largest for $\eta = 0.1$, smallest for $\eta = 0.001$.
- Too large a learning rate can cause divergence.
- The cost function oscillates and overshoots the minimum at high learning rates.

Sampling and Stochastic Gradient Descent

Part a. Sampling



```python
def generate(N, theta, input_mean, input_sigma, noise_sigma):
    """
    Generate normally distributed input data and target values.
    We have 2 input features.
    """
    X = np.random.normal(input_mean, input_sigma, (N, 2))
    noise = np.random.normal(0, noise_sigma, N)
    y = theta[0] + theta[1]*X[:, 0] + theta[2]*X[:, 1] + noise
    return X, y
```

Figure 8: Sampling was done using normal distribution, with the values given in the assignment.

Part b. SGD batch sizes and theta

1. (a) The Stochastic gradient descent is implemented in the StochasticLinearRegressor class using the fit method.
2. (b) The learned θ for each batch size is printed during the execution.
3. (c) The convergence criteria is based on the change in the average loss being less than a tolerance of 1e-6. This is checked after each epoch. The max epochs is set to 20000 to allow sufficient iterations for larger batch sizes.

Batch Size	Learned Theta
1	$\begin{pmatrix} 3.02047516 \\ 1.03597717 \\ 1.98865994 \end{pmatrix}$
80	$\begin{pmatrix} 3.00252785 \\ 1.00527501 \\ 2.00378549 \end{pmatrix}$
8000	$\begin{pmatrix} 3.00026481 \\ 1.00099963 \\ 2.00005258 \end{pmatrix}$
800000	$\begin{pmatrix} 2.98277774 \\ 1.02251327 \\ 2.00005258 \end{pmatrix}$

Table 2: Theta Values and Stopping Criteria for Different Batch Sizes

Part c. SGD Performance

1. The algorithms for varying batch sizes converge to approximately the same parameter values, as the objective is convex and the data is the same. The relative speed of convergence in terms of epochs is faster for smaller batch sizes (fewer epochs needed), but in terms of number of updates (gradient steps), smaller batch sizes require more updates. The number of iterations (updates) decreases as batch size increases.
2. The closed-form solution is implemented in the `closed_form_solution` method and computed in the `run_experiment` function.
3. The true parameters are [3, 1, 2]. The closed-form parameters are very close to the true parameters due to the large sample size. The SGD parameters for different batch sizes are also close to the closed-form and true parameters, with minor variations due to the stochastic nature and convergence tolerance.

Batch Size	Epochs	Time (s)	Theta	Error in Theta
1	21	394.2	$\begin{pmatrix}2.97297126\\1.01006731\\2.03841411\end{pmatrix}$	0.00076922
80	24	15.61	$\begin{pmatrix}2.99812122\\1.0060964\\2.00051594\end{pmatrix}$	0.000012704
8000	474	45.80	$\begin{pmatrix}2.99290071\\1.00191675\\1.99944082\end{pmatrix}$	0.000018121
800000	29448	262.96	$\begin{pmatrix}2.94667618\\1.01192081\\1.99608486\end{pmatrix}$	0.00099907

Table 3: Training Performance for Different Batch Sizes

Observations:

- Large batch sizes increase per-epoch time, small batch sizes have more noise.
- Batch sizes 80 and 8000 yield fastest and best convergence.
- Parameters closest to true values for moderate batch sizes.
- Large batch size (full gradient descent) converges slower.

Closed Form Solution

$$\theta = (X^T X)^{-1} X^T Y$$

Closed Form Theta: (3.0038, 0.9993, 2.0001), error: 0.00387
True Theta: (3.0000, 1.0000, 2.0000)

Part d. MSE Loss

Batch Size	Train MSE	Test MSE	Error Diff
1	2.0064	2.0067	0.00076922
80	1.9992	1.9995	0.000012704
8000	1.9988	1.9992	0.000018121
800000	1.9998	1.9999	0.00099907

Table 4: Comparison of MSE values across batch sizes

Train and test MSE losses are close to the sampling variance.

Part e. Movement of Theta

In the 3D parameter space, the movement for small batch sizes (e.g., r=1) is more erratic and noisy, with zigzagging paths due to the high variance in gradient estimates from small batches. As batch size increases, the path becomes smoother and more direct toward the minimum, resembling deterministic gradient descent. This makes intuitive sense because larger batches provide a better approximation of the true gradient, reducing noise in the updates and leading to steadier progress.

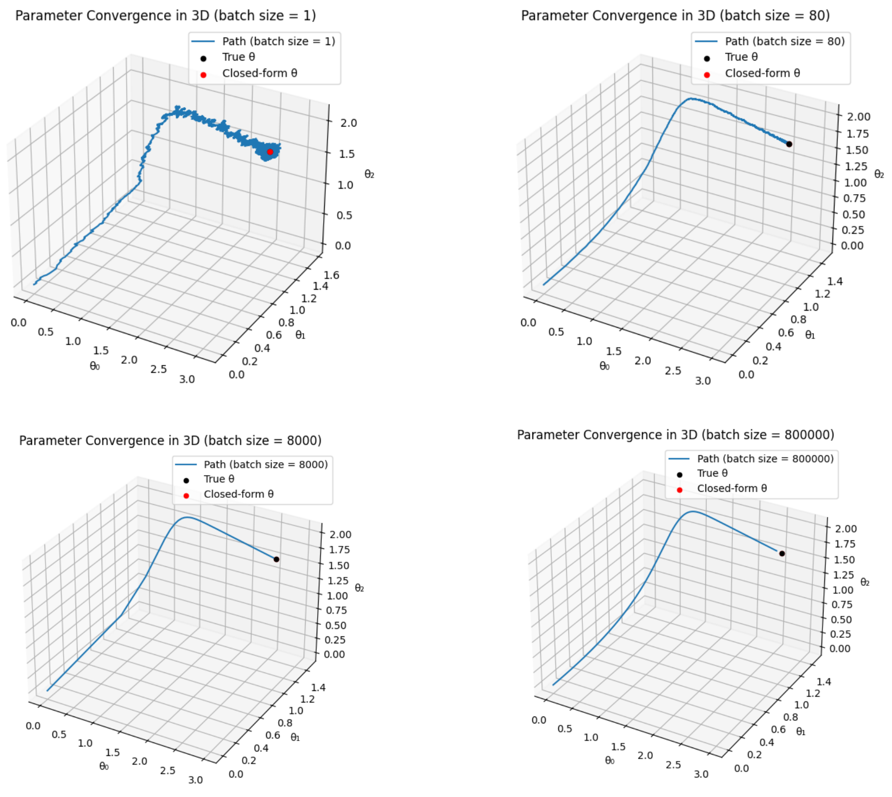


Figure 9: Movement of θ for different batch sizes. Smaller batches have higher fluctuations.

Logistic Regression

Part a. Log-likelihood and Optimization

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

$$\nabla_\theta LL(\theta) = X^T(Y - h_\theta(X))$$

Hessian:

$$H = \sum_{i=1}^{m} x^{(i)}(x^{(i)})^T h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))$$

Final Parameters (whole data):

$$\theta = \begin{pmatrix} 0.40125316 \\ 2.5885477 \\ -2.72558849 \end{pmatrix}$$

Final Parameters (train 80%):

$$\theta = \begin{pmatrix} 0.0472 \\ 2.6843 \\ -2.2154 \end{pmatrix}$$

Part b. Data and Decision Boundary

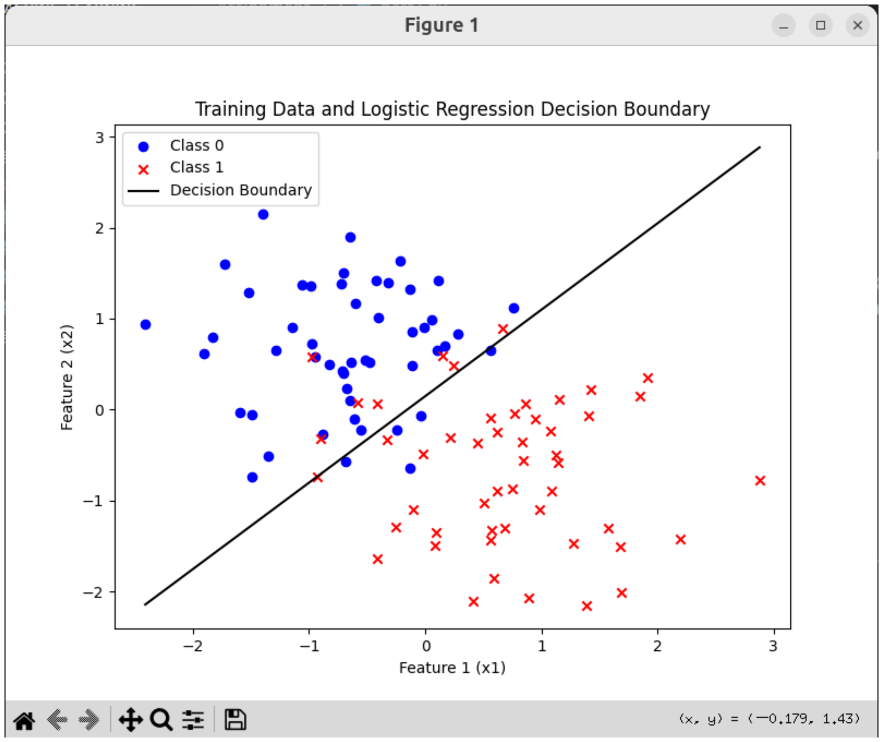


Figure 10: Training data (x1, x2) with class labels and logistic regression decision boundary

Gaussian Discriminant Analysis

Part a. Shared Covariance

Means:

$$\mu_0 = [-0.7553, \;\; 0.6851], \qquad \mu_1 = [0.7553, \;\; 0.6851]$$

Shared covariance matrix:

$$\Sigma = \begin{pmatrix} 0.4295 & -0.0225 \\ -0.0225 & 0.5306 \end{pmatrix}$$

Part b. Data Plot

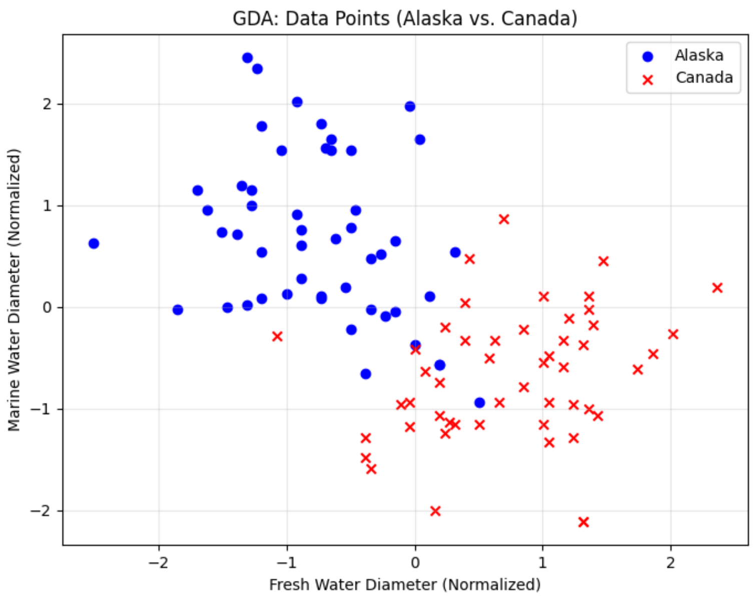


Figure 11: Normalized growth ring diameters (marine vs. fresh water), Alaska and Canada labels

Part c. Decision Boundary Equation (Linear)

Equation for boundary:

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

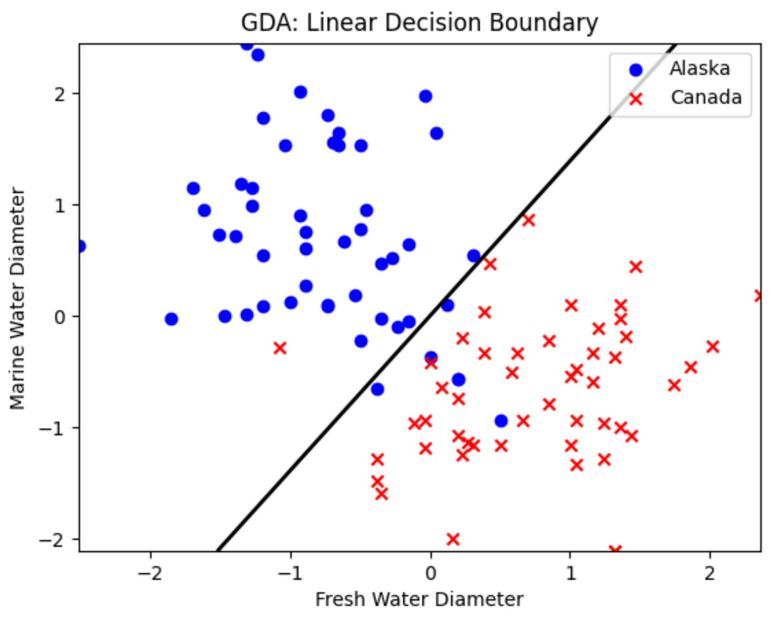


Figure 12: GDA Linear Decision Boundary

Part d. Separate Covariances

Means:

$$\mu_0 = [-0.75529433,\ \ 0.68509431], \quad \mu_1 = [0.75529433,\ -0.68509431]$$

Covariance matrices:

$$\Sigma_0 = \begin{pmatrix} 0.38937733 & -0.15802567 \\ -0.15802567 & 0.6609563 \end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix} 0.48721548 & 0.11216388 \\ 0.11216388 & 0.4219943 \end{pmatrix}$$

Part e. Quadratic Decision Boundary Equation

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

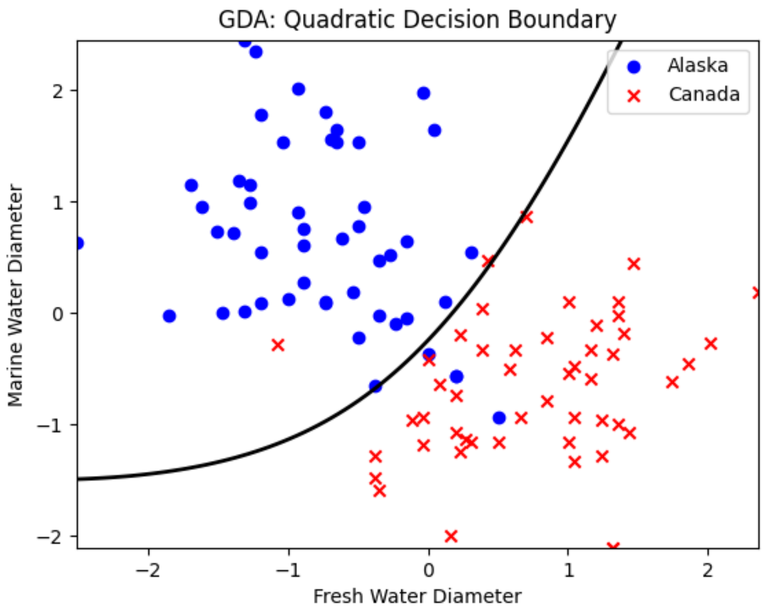


Figure 13: GDA Quadratic Decision Boundary

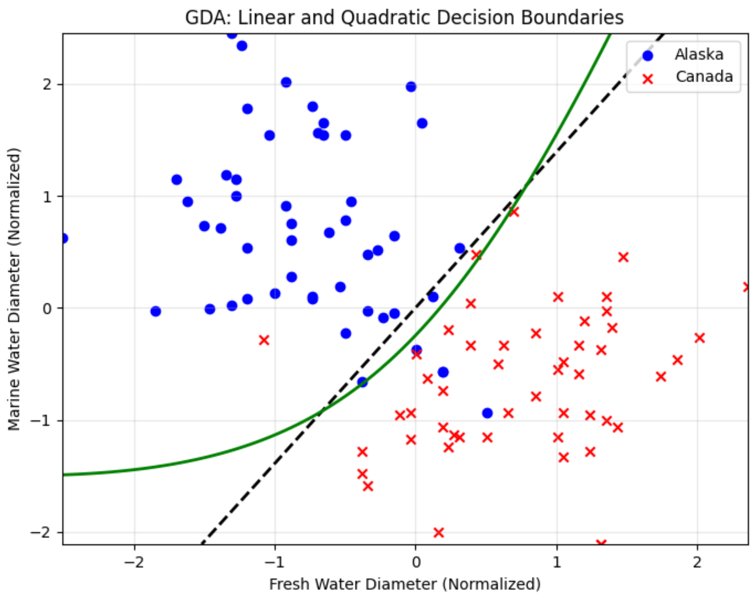


Figure 14: GDA Linear and Quadratic Decision Boundaries

Part f. Model Comparison and Observations

- Quadratic boundary classifies points more accurately than linear.
- Assumption of Gaussian distribution fits the data well.

- No overfitting observed on test set: quadratic boundary generalizes better.
- Training accuracy (linear): 0.9625, Testing: 0.8
  Training accuracy (quadratic): 0.95, Testing: 0.85
- Quadratic boundary gives improved test accuracy, showing better generalization.