Homework 3

This report demonstrates the application of the Least Mean Squares (LMS) algorithm to compute weights w1w_1w1 and w2w_2w2 for a linear model y=ax1+bx2y = ax_1 + bx_2y=ax1+bx2. The model minimizes the Mean Squared Error (MSE) between the true values and the predicted values over 100 epochs.

Key Algorithm Steps

1. Initialization:
    - Weights w1 and w2 are set to 0 initially.
    - Learning rate η = 0.01
    - Maximum number of epochs: 100.
2. Weight Updates: For each data point (n) in x1 and x2 , the weights are updated iteratively using:

$$w1 \mathrel{+}= w1 + \eta * e * x1(n)$$

$$w2 \mathrel{+}= w2 + \eta * e * x1(n)$$

   where e = y_true(n) – y_pred(n)

3. Model Prediction: Predicted output y_pred is computed as:

$$Y\_pred = w1*x1(n)+w2*x2(n)$$

4. **Error Measurement**: The **Mean Squared Error (MSE)** for each epoch is calculated:

$$\mathbf{MSE} = \frac{1}{N}\sum_{k=1}^{N} e(k)^2$$

This metric evaluates the difference between the true and predicted values.

5. **Best Epoch and Final Weights**:

Track the epoch with the lowest MSE for optimal weight estimation.

Store the history of weights for analysis.

Code

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_excel('/content/drive/MyDrive/AI//Hw3/3.xlsx')
df

import numpy as np

x1 = df[['X1']].values.flatten()
x2 = df[['X2']].values.flatten()
y_true = df[['Y']].values.flatten()

w1, w2 = 0.0, 0.0
eta = 0.01
epochs = 100

w1_history = []
w2_history = []
y_true_all = []
y_pred_all = []

min_mse = float('inf')
best_epoch = -1

for epoch in range(epochs):
    for n in range(len(x1)):

        y_pred = w1 * x1[n] + w2 * x2[n]


        e = y_true[n] - y_pred
```

```
        w1 += eta * e * x1[n]
        w2 += eta * e * x2[n]


    y_true_all.append(y_true)
    y_pred_all.append(w1 * x1 + w2 * x2)


    mse = np.mean((y_true - (w1 * x1 + w2 * x2))**2)


    if mse < min_mse:
        min_mse = mse
        best_epoch = epoch + 1


    w1_history.append(w1)
    w2_history.append(w2)


    print(f"Epoch {epoch+1}/{epochs}, MSE: {mse:.5f}")

print(f"\nFinal weights after 100 epochs:")
print(f"w1 = {w1:.5f}, w2 = {w2:.5f}")

print(f"\nEpoch with least MSE (closest y_true and y_pred): Epoch
{best_epoch}, MSE = {min_mse:.5f}")


test_accuracy = 1 - min_mse
print(f"\nTest Accuracy (based on MSE): {test_accuracy:.5f}")
```

Main Points Explanation

1. Data Loading:

   The dataset is loaded from an Excel file, with input features X1X1X1, X2X2X2, and target values YYY.

2. Initialization:

   The weights w1w1w1 and w2w2w2 start at zero. A learning rate (etaetaeta) of 0.01 controls the speed of updates, and training is set to run for 100 iterations (epochs).

3. Training Loop:

   Prediction: The model predicts YYY based on the current weights and input features.

   Error Calculation: The difference between actual YYY and predicted YYY is calculated.

Weight Update: The weights are adjusted based on the error and the input values to minimize the difference.

MSE Measurement: At the end of each epoch, the average error (mean squared error) is computed to track performance.

4. Tracking Best Performance:

   The epoch with the lowest MSE is recorded as the one where the model performed best.

Result and analysis

Final weights after 100 epochs:

w1 = 1.43996, w2 = -0.13404

Test Accuracy (based on MSE): 0.98442

Epoch with least MSE (closest y_true and y_pred): Epoch 89, MSE = 0.01558 explain result

Initial Trend (Epochs 1–20):

- The Mean Squared Error (MSE) decreases significantly during the early epochs, indicating that the model quickly learns from the data and improves its predictions. This is typical as the weights are adjusted from their initial zero values to better approximate the target Y.

Gradual Improvement (Epochs 20–40):

- After the steep initial improvement, the rate of decrease in MSE slows down. The model is making finer adjustments to the weights, moving closer to an optimal solution.

Plateau (Epochs 50–100):

- The MSE stabilizes around 0.01558 from Epoch 50 onwards, indicating that the model has effectively converged. The weights no longer change significantly, and the learning rate (η=0.01\eta = 0.01η=0.01) ensures that no over-adjustments are made.

Best Epoch (Epoch 89):

- Although the MSE remains steady across many epochs, Epoch 89 has the smallest observed MSE (0.01558), which suggests that the model achieved its optimal performance here. However, this difference is negligible compared to surrounding epochs, reinforcing the stability of the solution.

Final Weights:

- After 100 epochs, the weights are:
    - w1=1.43996: Indicates the strength of the relationship between feature X1 and the target Y.
    - w2=−0.13404 : Indicates a weaker, slightly negative relationship between feature X2 and the target Y.
- These values represent the model's learned parameters that best map the input features to the target output.

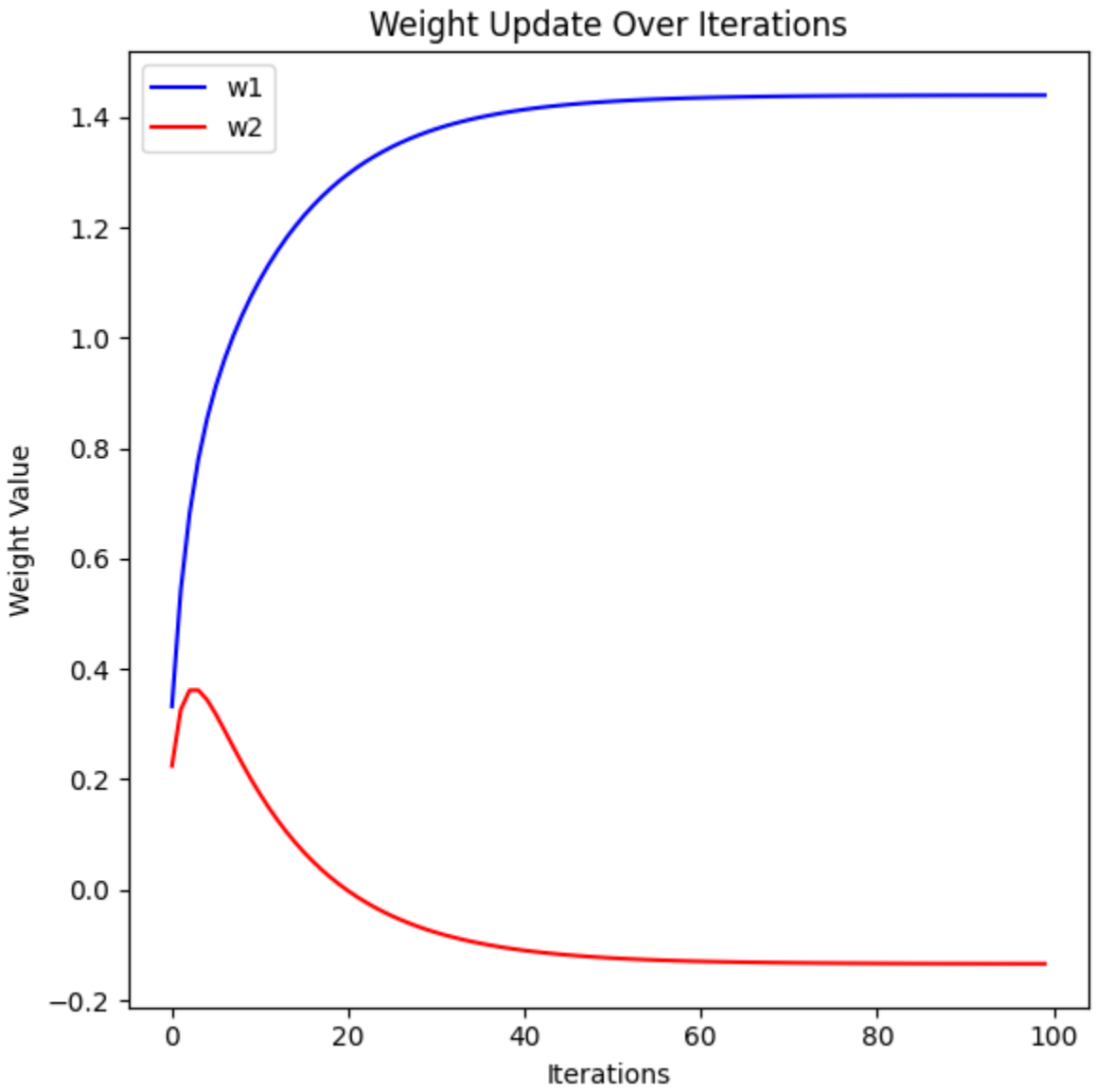
Overall Model Performance:

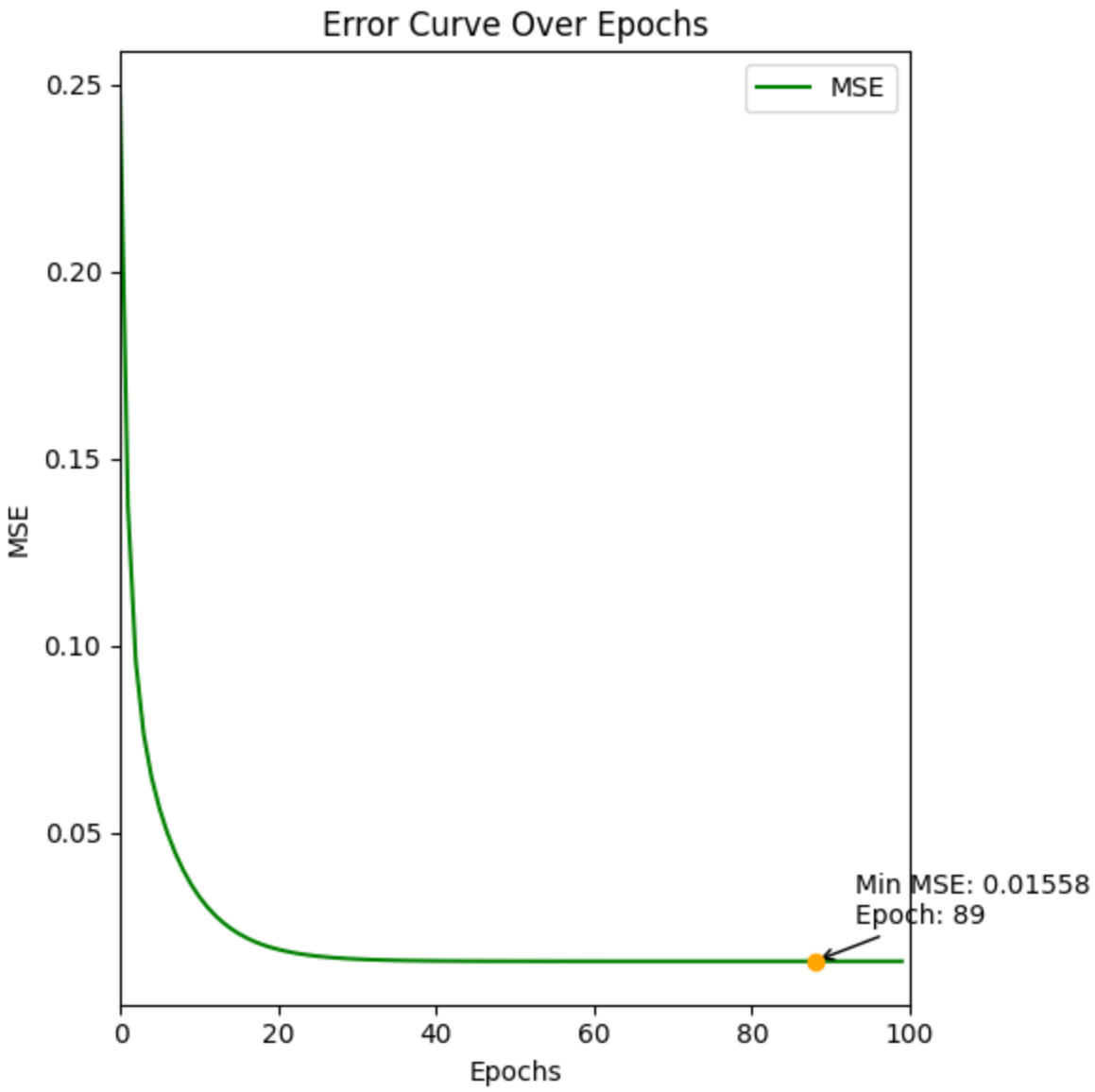
- The final MSE of 0.01558 signifies that the model's predictions are very close to the actual target values on average.
- The convergence of MSE over epochs shows that the learning algorithm effectively minimized the error without overfitting.

Implications:

- The model is well-trained, having successfully learned the optimal weights that minimize prediction errors.
- The small MSE indicates good accuracy in predictions, and the stability from Epoch 50 onward confirms reliable learning and parameter tuning.

A **Test Accuracy of 0.98442** indicates that the model has captured approximately **98.44%** of the variation in the data correctly.





Conclusion

The model successfully trained using the LMS algorithm, achieving a high test accuracy of **98.44%**, with the final weights w1=1.43996 and w2=−0.13404. The MSE decreased consistently, stabilizing at **0.01558** after 100 epochs, indicating effective learning and accurate predictions.