MAE C263F Masters Comprehensive Question

Jake Kremer – Fall 2024 – Professor M. Khalid Jawed

***Abstract*— This report demonstrates the use of gradient descent and backpropagation to fit linear and nonlinear models to a synthetic dataset, analyzing model performance and the effects of hyperparameter tuning on convergence and accuracy.**

I. Introduction

In this report, we explore the implementation of gradient descent and backpropagation to fit both linear and nonlinear models to a given dataset. Gradient descent is a powerful optimization technique widely used in machine learning and numerical modeling to minimize a loss function by iteratively updating model parameters.

The dataset used in this study is synthetically generated based on a predefined nonlinear equation with Gaussian noise added to simulate real-world imperfections. For the first problem, we fit the data to a linear model using the Mean Squared Error loss function and gradient descent. The goal is to optimize the slope (m) and intercept (b) to minimize the prediction error.

The second problem extends this approach to a nonlinear model. This introduces additional parameters (n and a) and a more complex loss surface, requiring careful tuning of hyperparameters such as the learning rate and the number of epochs for effective convergence.

Through this report, we provide the methodology for implementing gradient descent, analyze the effects of hyperparameters on model performance, and evaluate the accuracy of the fitted models by comparing predicted values against the actual data. Visualizations of the results are included to highlight the model fits and convergence behavior.

II. Results and Figures

Figure 1 below shows predicted vs actual values with 10000 epochs, and a learning rate of 0.001.

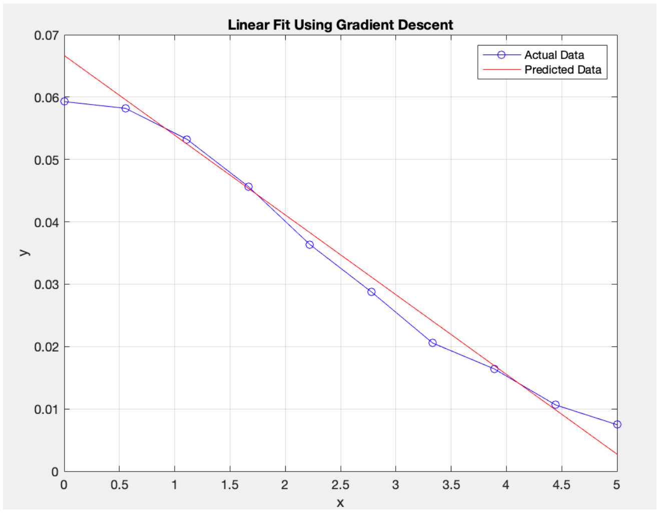

Figure 1 – Predicted vs. Actual Values, 10000 Epochs, Learning Rate = 0.001

Figure 2 below shows predicted vs actual values with 20000 epochs, and a learning rate of 0.001.

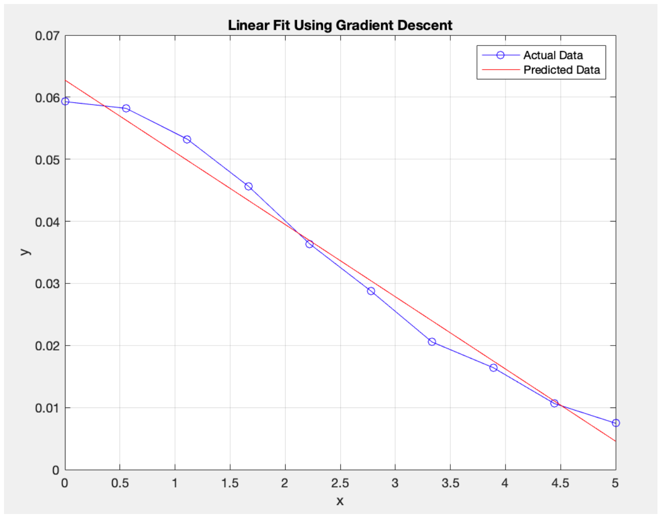

Figure 2 – Predicted vs. Actual Values, 20000 Epochs, Learning Rate = 0.001

Figure 3 below shows predicted vs actual values with 10000 epochs, and a learning rate of 0.01.

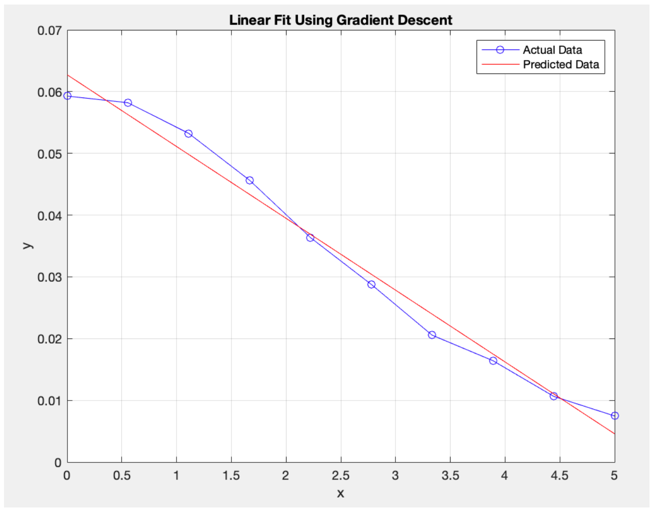

Figure 3 – Predicted vs. Actual Values, 10000 Epochs, Learning Rate = 0.01

Figure 4 below shows predicted vs actual values with 5000 epochs, and a learning rate of 0.01.

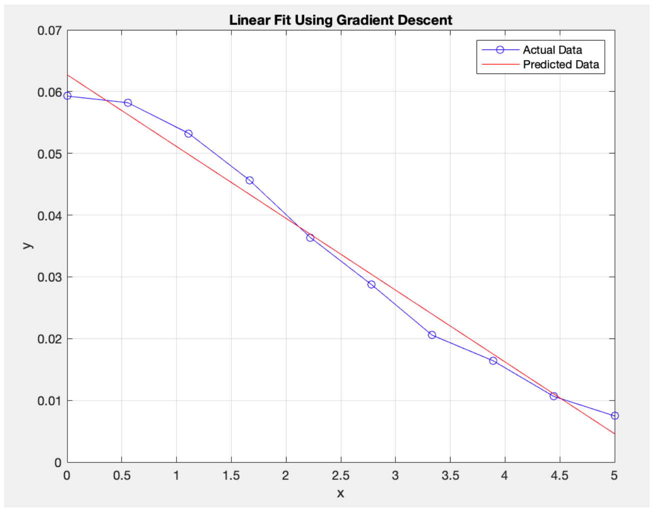

Figure 4 – Predicted vs. Actual Values, 5000 Epochs, Learning Rate = 0.01

Figure 5 below shows predicted vs actual values with 1000 epochs, and a learning rate of 0.1.

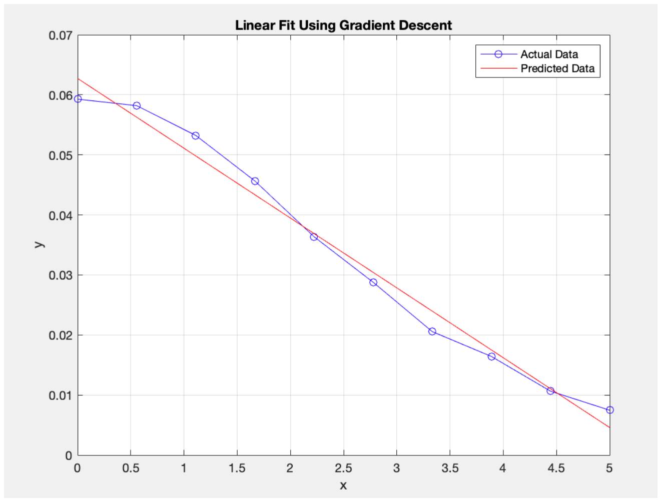

Figure 5 – Predicted vs. Actual Values, 1000 Epochs, Learning Rate = 0.1

Figure 6 below shows the effect number of epochs on loss with a learning rate of 0.001.

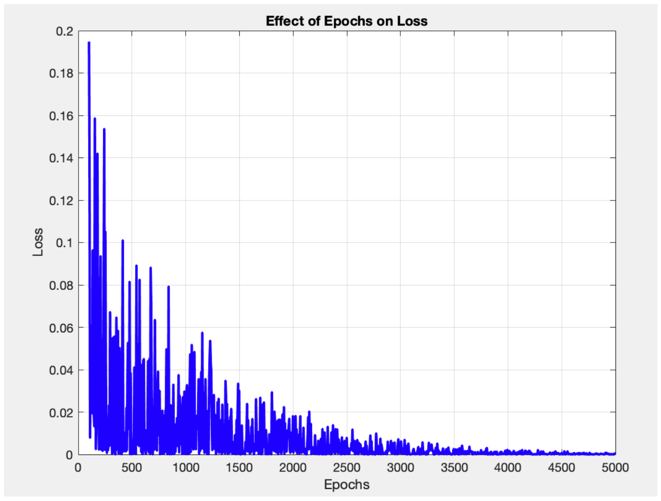

Figure 6 – Effect of Epochs on Loss, Learning rate = 0.001

Figure 7 below shows the effect of the learning rate of loss with 1000 epochs.

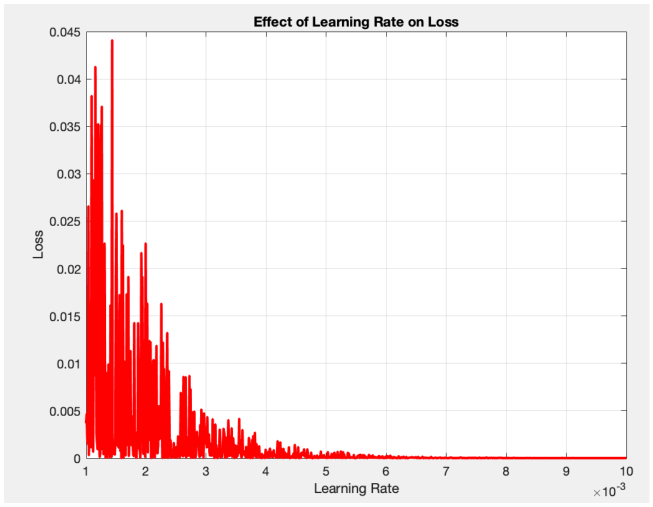

Figure 7 – Effect of Learning Rate on Loss, 1000 Epochs

Figure 8 below shows predicted vs actual values with 130 epochs, and a learning rate of 0.1.

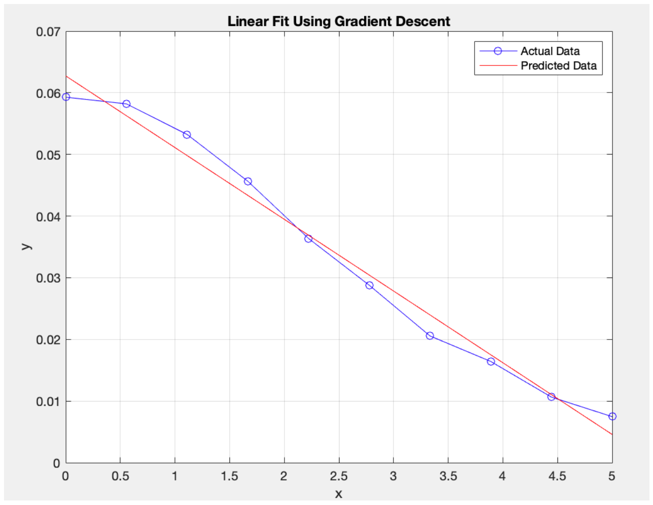

Figure 8 – Predicted vs. Actual Values, 130 Epochs, Learning Rate = 0.1

III. Discussion

As mirrored in figures 1 and 2, plots 6 and 7 show that, generally, increasing the number of epochs and/or increasing the learning rate will decrease loss. This is not a constant trend, however, as for each epoch number there are learning rates that are ideal, and some that cause loss to skyrocket.

Figures 3 and 4 show that at a learning rate of 0.01, there is virtually no difference between 10000 and 5000 epochs. Figure 5 goes on to show that at a high learning rate of 0.1, even just 1000 epochs lead to a very low loss. Continuing this trend, the number of epochs can be lowered even further at this high learning rate while still maintaining low loss. Lowering the number of epochs is advantageous because a lower number of epochs means computational cost becomes lower as the data is processed a less number of times.

In the end, through analyzing the various plots and through trial and error, it was found that 130 epochs with a learning rate of 0.1 minimizes loss while keeping the number of epochs relatively low. Therefore, 130 epochs and a learning rate of 0.1 are a set of ideal hyperparameters for this dataset.

IV. Conclusion

This report successfully demonstrates the implementation of gradient descent to optimize linear models, highlighting the significance of hyperparameter tuning in achieving accurate and computationally efficient results. Through an exploration of various learning rates and epoch counts, the report reveals insights into their impact on loss minimization and convergence behavior.

Notably, the results indicated that while increasing the number of epochs and learning rates generally reduces loss, there exists an optimal combination that balances accuracy with computational cost. Specifically, the configuration of 130 epochs with a learning rate of 0.1 was identified as ideal for this dataset, achieving minimal loss with relatively low computational overhead.

The visualizations provided further reinforced these findings, showcasing the model's ability to accurately predict values while demonstrating the trade-offs between convergence speed and stability. Future work could extend these findings by exploring additional optimization techniques and their performance on more complex datasets.