Gradient Descent and Backpropagation

Premkumar Sivakumar

***Abstract*— This report summarizes the modeling techniques to linearly and non-linearly fit data using gradient descent and backpropagation.**

I. Introduction

In data analysis and machine learning, fitting models to data is a fundamental task that enables us to understand relationships and make predictions. This assignment explores linear and non-linear data fitting techniques using gradient descent and backpropagation. Gradient descent, a widely used optimization algorithm, iteratively minimizes the error between predicted and actual values by updating model parameters. Backpropagation, extends gradient descent by efficiently computing gradients for non-linear models. Through this assignment, we will implement and compare the performance of linear and non-linear models in fitting datasets, emphasizing the role of optimization and gradient computation in achieving accurate results.

II. Linear Fit

A. Model

First, 10 random values for *x* were generated within a range and the corresponding values of *y* were generated based on the following non-linear function along with added Gausian noise

$$y_{generated} = n \exp\left(-a\left(m\, x_{generated} + b\right)^2\right) + noise$$

where the parameters for the function were set randomly as $n = 0.06$, $a = 0.25$, $m = 0.57$, $b = 0.11$. In this case of linearly fitting data, the predicted values were fit to the following linear equation using gradient descent and backpropagation

$$y = m\, x + b$$

Mean Squared Error (MSE) loss function was used:

$$Loss(m, b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right)^2$$

The gradients of the loss function with respect to m and b are:

$$\frac{\partial Loss}{\partial m} = -\frac{2}{N}\sum_{i=1}^{N} x_i(y_i - (m\, x_i + b))$$

$$\frac{\partial Loss}{\partial b} = -\frac{2}{N}\sum_{i=1}^{N} (y_i - (m\, x_i + b))$$

Using gradient descent, the parameters *m* and *b* are updated as follows:

$$m \leftarrow m - \eta\frac{\partial Loss}{\partial m},\ b \leftarrow b - \eta\frac{\partial Loss}{\partial b}$$

where η is the learning rate. The hyperparameters used were epochs = 10000 and learning rate = 0.001.

B. Results

Figure 1 shows the linear fit for the given hyperparameters. It can be seen that the predicted values almost closely match the generated values for the given model.

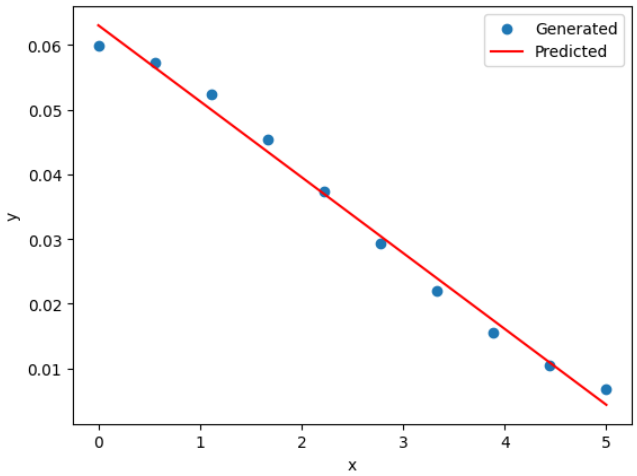

Figure 1. Linear fit for epochs = 10000, η = 0.001

To see how the model would be affected by the choice of hyperparameters, the data was fit for different values of learning rate and epochs.

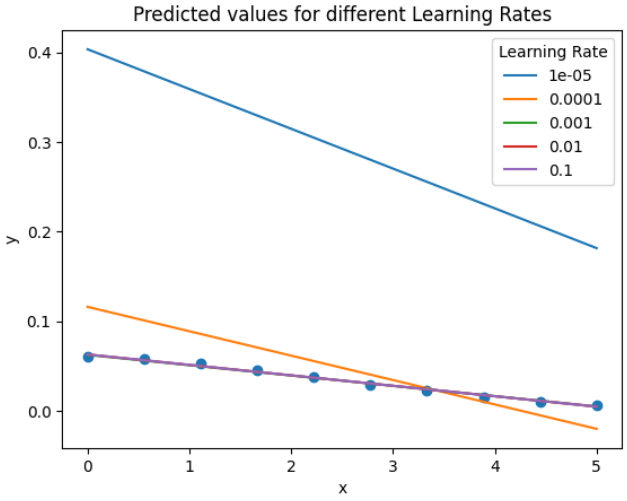

Figure 2. Sensitivity of Learning Rate

Premkumar Sivakumar is a Masters student in Mechanical Engineering department at University of California, Los Angeles

Figure 2 shows the sensitivity of learning rate. The learning rate was varied by setting the epochs to 10000. For smaller learning rates of 0.0001 and 0.00001 the model is far from the generated data. The model fits better for higher values of learning rate.

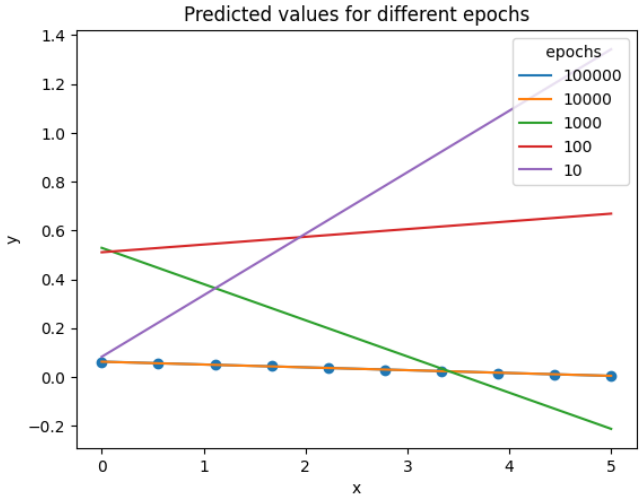

Figure 3. Sensitivity of epochs

Figure 3 shows the sensitivity of epochs. The epochs was varied by setting the learning rate to 0.001. As expected for lower values of epochs the model does not fit with the generated data, while with increasing epochs the model is trained higher number of times to fit the data as in the case for epochs = 10000 or 100000.

III. Non-Linear Fit

A. Model

Same data as the linear fit case was used to fit the data to the non-linear equation:

$$y = n\, e^{(-a\, y_{int})},$$

$$\text{where } y_{int} = (m\, x + b)^2$$

using gradient descent and backpropagation. Mean Squared Error (MSE) loss function was used:

$$Loss(n, a, m, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - n\exp(-a\,(m\,x_i + b)^2))^2$$

The gradients of the loss function with respect to *n*, *a*, *m*, and *b* are:

$$\frac{\partial Loss}{\partial n} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - n\exp(-a\,y_{int}))\exp(-a\,y_{int})$$

$$\frac{\partial Loss}{\partial a} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - n\exp(-a\,y_{int}))\,n\exp(-a\,y_{int})(-y_{int})$$

$$\frac{\partial Loss}{\partial m} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - n\exp(-a\,y_{int}))\,n\exp(-a\,y_{int})(-a)2(mx_i + b)x_i$$

$$\frac{\partial Loss}{\partial b} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - n\exp(-a\,y_{int}))\,n\exp(-a\,y_{int})(-a)2(mx_i + b)$$

Using gradient descent, the parameters *n*, *a*, *m*, and *b* were updated as follows:

$$n \leftarrow n - \eta\frac{\partial Loss}{\partial n},\quad a \leftarrow a - \eta\frac{\partial Loss}{\partial a}$$

$$m \leftarrow m - \eta\frac{\partial Loss}{\partial m},\quad b \leftarrow b - \eta\frac{\partial Loss}{\partial b}$$

Where η is the learning rate. The hyperparameters used were epochs = 10000 and learning rate = 0.001

B. Results

Figure 4 shows the non-linear curve fitted using the given hyperparameters. As evident, the given hyperparameters could be further optimized to fit the curve better.

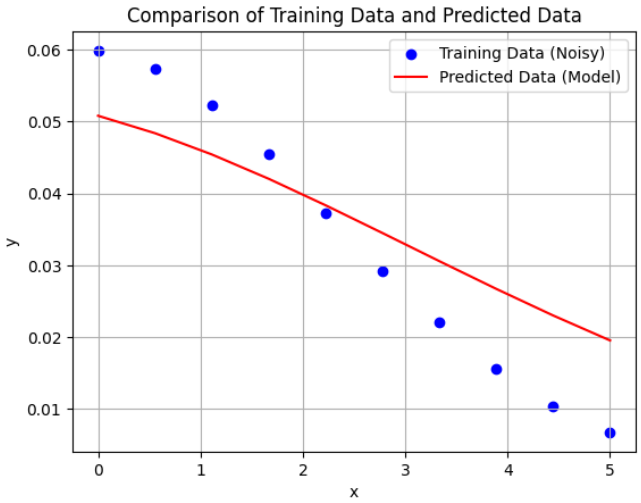

Figure 4. Non-Linear fit for epoch = 10000, η = 0.001

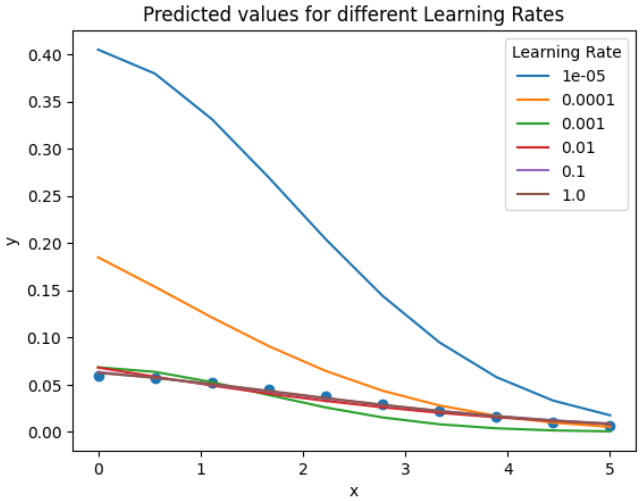

Figure 5. Sensitivity of Learning Rate

Figure 5 shows the sensitivity of learning rate. The learning rate was varied by setting the epochs to 10000. For smaller values of learning rate, the curve moves farther away from the data. Learning rate values of 1 and 0.1 give better results.

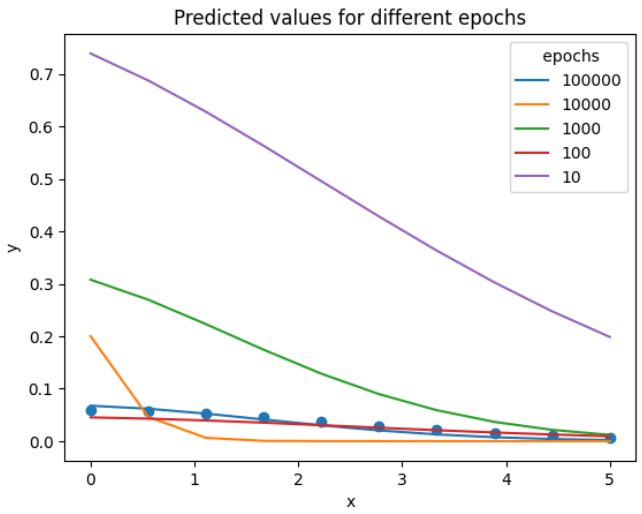

Figure 6. Sensitivity of epochs

Figure 6 shows the sensitivity of epochs. The epochs was varied by setting the learning rate to 0.001. Clearly, the curve fits better for higher epochs, especially for an epoch value of 100000.

Figure 7. Optimized fit with epoch = 100000, η = 0.01

Based on the sensitivity analysis the model was optimized with an epoch value of 100000 and learning rate of 0.01. Figure 7 shows the optimized fit.

IV. Conclusion

In summary, the hyperparameters control how much the weights of the model are updated during training. The values of the hyperparameters can vary widely based on the problem, dataset, and optimization algorithm. Enough epochs allow the model to learn the meaningful patterns without overfitting. Too few epochs will underfit the curve and too high epochs will overfit. An optimal epochs for the given problem was identified and the model was hence optimized. Similarly, an optimal learning rate is important for convergence and stability of the model, to ensure it is small enough to update the weights delicately and large enough not to slow the training significantly.

References

[1] Professor M. Khalid Jawed, MechAE 263F Course modules Link: https://bruinlearn.ucla.edu/courses/193842/modules Portion of this code and the helper functions or files have been referenced from the class resources on Bruinlearn.