1 Introduction Machine Learning

To provide a comprehensive introduction to the foundational concepts of machine learning, we propose presenting an illustrative problem that serves as a case study. This example will delineate key terminologies and demonstrate the sequential process inherent in machine learning systems. Through this example, the theoretical framework and operational procedures will be examined, offering insights into algorithmic design, model training, and evaluation.

1.1 Example: Simple Regression

Consider the following *dataset* of input-output pairs:

$$\begin{aligned} 1 &\implies 1 \\ 2 &\implies 4 \\ 3 &\implies 9 \\ 4 &\implies 16 \\ 5 &\implies 25 \end{aligned}$$

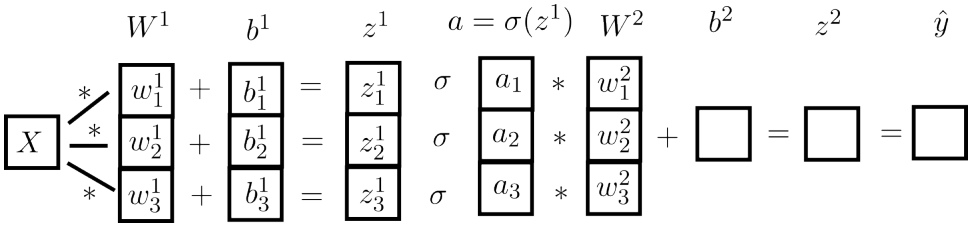
From a mathematical standpoint, this dataset exhibits a clear and deterministic pattern. Upon examination, one can infer that the output y is the square of the input x, i.e.

$$y = x^2 \tag{1}$$

This conclusion is reached by recognizing that each output corresponds exactly to the square of its respective input value. Such an inference illustrates a fundamental example of function approximation in the realm of machine learning, where the objective is to model the relationship between inputs and outputs. In this instance, the underlying function is explicitly known and can be directly derived from the provided data. On the other hand, when dealing with datasets that do not exhibit a readily discernible analytical relationship, numerical methods offer a robust alternative for modeling the underlying phenomenon. In such cases, rather than deducing a closed-form expression from first principles, one can employ data-driven approaches to approximate the functional mapping between inputs and outputs.

1.1.1 Solution Proposal

Lets assume that we a semantic structure to calculate the output of an any given input for a given problem like following:



Here $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ represent *real-valued parameters*, σ represents a function, X represents a vector of inputs and y represents a vector of outputs. If we try to obtain explicit formulation of the given schema, the following steps should be followed:

$$f(x) = \sum[\sigma(\mathbf{z}^{(1)}) * \mathbf{W}^{(2)}] + \mathbf{b}^{(2)} = \hat{y} \tag{2}$$

$$f(x) = \sum[\sigma(\mathbf{W}^{(1)} * x + \mathbf{b}^{(1)}) * \mathbf{W}^{(2)}] + \mathbf{b}^{(2)} = \hat{y} \tag{3}$$

$$f(x) = \sigma(W_1^1 * x + b_1^1) * W_1^{(2)} + \sigma(W_1^{(1)} * x + b_2^{(1)}) * W_2^{(2)} + \sigma(W_3^{(1)} * x + b_3^{(1)}) * W_3^{(2)} + b_2 = z^{(2)} = \hat{y} \tag{4}$$

In this formulation, σ denotes the *activation function*, which is typically chosen to be non-linear. The activation function introduces nonlinearity into the network, thereby enabling it to learn complex

relationships between the input features and their corresponding outputs. $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ serve as the weight parameters, governing the multiplicative interactions with the input vectors and the outputs of the activation function, respectively. Meanwhile, $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ are the bias terms added to these linear transformations. The total operation of taking an input and producing an output using a single weights and biases with/without activation function is called a layer.

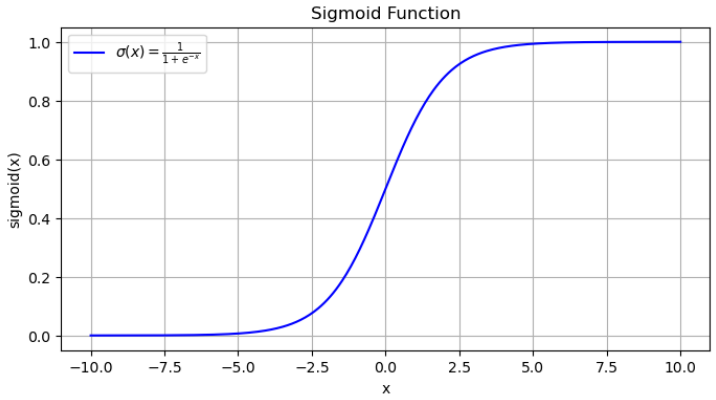
$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{5}$$

$$\mathbf{a} = \sigma(z) \tag{6}$$

where $\mathbf{z}$ represents the pre-activation output, $\mathbf{a}$ activation output.

In this formulation, the first step involves selecting the type of activation function to be employed. As this remains an active research area, numerous options exist. One of the most widely utilized activation functions is the sigmoid function, which can be defined as:

$$sigmoid = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$



This function is particularly notable for its ability to map any real-valued input into the interval $(0, 1)$, making it suitable for modeling logistic-like relationships in various machine learning tasks. Derivation of the sigmoid function is

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \sigma(x)(1 - \sigma(x)) \tag{8}$$

The subsequent phase involves determining the initial values for the weights and biases. In the absence of a predefined methodology, these parameters are typically initialized using random values. However, relying solely on random initialization is insufficient for accurately modeling complex phenomena. Consequently, it becomes necessary to iteratively refine these parameters in accordance with the specific problem at hand—a process known as learning.

1.1.2 Learning

To facilitate learning, it is imperative to provide a metric that quantifies the quality of the model's performance. This is achieved by comparing the model's predicted outputs with the corresponding actual outputs. The discrepancy between these values is quantified using a *loss function*, which serves as a critical measure of model accuracy and guides the subsequent adjustments to the model parameters.

The selection of a loss function is inherently discretionary, as any function that quantifies the deviation between predicted and actual outputs can be employed. The literature presents a plethora of loss functions, such as mean squared error (MSE), mean absolute error (MAE), and cross entropy, among others. In this problem, the mean squared error is adopted as the loss function.

$$L = \frac{1}{n}\sum(\hat{y} - y)^2 \quad L(W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}, X) \tag{9}$$

where $\hat{y}$ is the predicted output, y is the actual output and n is input count. From definition we can see that the loss function is a function of the weights and biases, as well as the input data.

In order to enhance the model's predictive capabilities, it is necessary to devise a strategy that updates the weights and biases in a manner that systematically reduces the loss function. Within this framework, the learning process can be interpreted as an iterative procedure aimed at minimizing the loss by appropriately adjusting the model parameters. This principle forms the cornerstone of modern machine learning algorithms, ensuring that the model progressively aligns more closely with the underlying data distribution.

1.1.3 Gradient Descent

Minimizing the loss function is fundamental to the learning process because it ensures that the model's parameters are refined to produce increasingly accurate predictions. The loss function quantifies the discrepancy between predicted and actual outputs; consequently, reducing this discrepancy translates to better alignment between the model and the true data-generating process. By systematically minimizing the loss, the model iteratively adapts its weights and biases to capture the underlying patterns in the data, thereby improving its generalization capabilities and overall performance.

Minimizing the loss function is fundamental to the learning process because it ensures that the model's parameters are refined to produce increasingly accurate predictions. The loss function quantifies the discrepancy between predicted and actual outputs; consequently, reducing this discrepancy translates to better alignment between the model and the true data-generating process. By systematically minimizing the loss, the model iteratively adapts its weights and biases to capture the underlying patterns in the data, thereby improving its generalization capabilities and overall performance. An *epoch*, defined as one complete pass through the entire training dataset, is a critical concept in this process. Multiple epochs allow the model to progressively *fine-tune* its parameters, as each pass provides an opportunity to reduce the error further and consolidate learning.

A crucial aspect of this iterative minimization process involves computing the derivative (or gradient) of the loss function with respect to the weights and biases. By examining how small changes in these parameters affect the loss, one can determine the direction in which the loss decreases most rapidly. This gradient-based approach is commonly implemented via *gradient descent*, an iterative optimization algorithm that updates model parameters by moving them in the direction of the steepest descent of the loss function. Importantly, the magnitude of each update is scaled by a factor known as the *learning rate*, a hyperparameter that controls the step size during the optimization process. This parameter is critical for balancing the convergence speed and stability of the learning process. Such an approach underpins modern optimization strategies in machine learning, enabling the systematic refinement of model parameters to achieve progressively lower loss values.

A fundamental element underpinning the computation of these derivatives is the *chain rule*, a core concept in calculus. In multi-layered models, the loss function is a composite of several functions representing successive layers of transformations. The chain rule facilitates the differentiation of this composite function by decomposing it into a product of simpler derivatives corresponding to each layer. This decomposition is crucial for efficiently propagating gradients backward through the network—a process known as *backpropagation*—which allows for precise adjustments to individual weights and biases. Without the chain rule, accurately computing how changes in any given parameter influence the overall loss would be considerably more complex, if not intractable, thereby impeding the optimization process.

If we write relations of each layer from 5 and 6 we can get following equations:

$$L = \frac{1}{2n}\sum(z^{(2)} - y)^2 \tag{10}$$

$$z^{(2)} = W^{(2)}a + b^{(2)} \tag{11}$$

$$a = \sigma(z^{(1)}) \tag{12}$$

$$z^{(1)} = W^{(1)}x + b^{(1)} \tag{13}$$

Following calculation steps from the last to the first, we can obtain derivate of the loss function with respect to the weights and biases. Firstly we need to calculate the derivate of the loss function with respect to the output of the last layer with using 10

$$\frac{\partial L}{\partial z^{(2)}} = \frac{2}{n}\sum(z^{(2)} - y) \tag{14}$$

Derivative of the loss function with respect to the weights and biases of the second layer can be calculated as follows using 11

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(2)}} = \frac{2a}{n} \sum (z^{(2)} - y) \tag{15}$$

$$\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial b^{(2)}} = \frac{2}{n} \sum (z^{(2)} - y) \tag{16}$$

To calculate gradients of first layer we first need to calculate the gradient of the activation from 11.

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a} = \frac{2W^{(2)}}{n} \sum (z^{(2)} - y) \tag{17}$$

Derivative of activation function is with using 8

$$\frac{\partial a}{\partial z^{(1)}} = \sigma(z^{(1)})(1 - \sigma(z^{(1)})) = a(1 - a) \tag{18}$$

Hence derivative of the loss function with respect to $z^{(1)}$ is

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z^{(1)}} = \frac{2W^{(2)} a(1 - a)}{n} \sum (z^{(2)} - y) \tag{19}$$

From 13 we can write derivatives of the first layer as:

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}} = \frac{2W^{(2)} a(1 - a)x}{n} \sum (z^{(2)} - y) \tag{20}$$

$$\frac{\partial L}{\partial b^{(1)}} = \frac{\partial L}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial b^{(1)}} = \frac{2W^{(2)} a(1 - a)}{n} \sum (z^{(2)} - y) \tag{21}$$

$$\tag{22}$$

After calculation of the derivatives of the loss function with respect to the weights and biases, we can update them using the following equations:

$$W^{(1)} = W^{(1)} - \eta \frac{\partial L}{\partial W^{(1)}} \tag{23}$$

$$b^{(1)} = b^{(1)} - \eta \frac{\partial L}{\partial b^{(1)}} \tag{24}$$

$$W^{(2)} = W^{(2)} - \eta \frac{\partial L}{\partial W^{(2)}} \tag{25}$$

$$b^{(2)} = b^{(2)} - \eta \frac{\partial L}{\partial b^{(2)}} \tag{26}$$

$$\tag{27}$$

where η is the learning rate. Since the objective is to minimize the loss function, it is imperative that the parameter updates occur in the direction opposite to that of the gradient. This counter-gradient adjustment ensures that each update effectively reduces the loss, thereby promoting convergence towards a minimum.

Algorithm 1: Training a Two-Layer Neural Network (Batch Size = 5, Input Dimension = 1)

Input: $X \in \mathbb{R}^{5\times 1}$, $y \in \mathbb{R}^{5\times 1}$, initial parameters $W^{(1)} \in \mathbb{R}^{1\times 3}$, $b^{(1)} \in \mathbb{R}^{1\times 3}$, $W^{(2)} \in \mathbb{R}^{3\times 1}$, $b^{(2)} \in \mathbb{R}$, learning rate η, number of epochs N

Output: Trained parameters $W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$

for *epoch = 1 to* N **do**

$z^{(1)} \leftarrow X \cdot W^{(1)} + b^{(1)}$ // Compute hidden layer pre-activation (result: 5×3)

$a^1 \leftarrow \sigma(z^{(1)})$ // Apply sigmoid activation elementwise: $\sigma(z) = \frac{1}{1+e^{-z}}$

$z^{(2)} \leftarrow a^1 \cdot W^{(2)} + b^{(2)}$ // Compute output layer pre-activation (result: 5×1)

$\hat{y} \leftarrow z^{(2)}$ // Predicted outputs

$L \leftarrow \frac{1}{2\cdot 5}\sum_{i=1}^{5}(y_i - \hat{y}_i)^2$ // Mean Squared Error (MSE)

$dL/d\hat{y} \leftarrow \frac{1}{5}(\hat{y} - y)$ // Gradient of loss w.r.t. $\hat{y}$

$dL/dW^{(2)} \leftarrow a^{1\top} \cdot (dL/d\hat{y})$ // Gradient w.r.t. $W^{(2)}$

$dL/db^{(2)} \leftarrow \sum(dL/d\hat{y})$ // Gradient w.r.t. $b^{(2)}$ (sum over samples)

$dL/da^1 \leftarrow (dL/d\hat{y}) \cdot (W^{(2)})^\top$ // Backpropagate error to hidden layer

$da^1/dz^{(1)} \leftarrow a^1 \circ (1 - a^1)$ // Elementwise derivative of sigmoid ($\circ$ denotes Hadamard product)

$dL/dz^{(1)} \leftarrow dL/da^1 \circ (da^1/dz^{(1)})$

$dL/dW^{(1)} \leftarrow X^\top \cdot (dL/dz^{(1)})$ // Gradient w.r.t. $W^{(1)}$

$dL/db^{(1)} \leftarrow \sum(dL/dz^{(1)})$ // Gradient w.r.t. $b^{(1)}$ (sum over samples)

$W^{(1)} \leftarrow W^{(1)} - \eta \cdot (dL/dW^{(1)})$

$b^{(1)} \leftarrow b^{(1)} - \eta \cdot (dL/db^{(1)})$

$W^{(2)} \leftarrow W^{(2)} - \eta \cdot (dL/dW^{(2)})$

$b^{(2)} \leftarrow b^{(2)} - \eta \cdot (dL/db^{(2)})$

return $W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$

1.1.4 Numerical Example

Input and output data are already is given as follow:

$$X = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{Bmatrix} \tag{28}$$

$$Y = \begin{Bmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \end{Bmatrix} \tag{29}$$

$$W^{(1)} = \begin{Bmatrix} 0.8 \\ 0.6 \\ -1.3 \end{Bmatrix} \tag{30}$$

$$b^{(1)} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{31}$$

$$W^{(2)} = \begin{Bmatrix} 3.4 \\ -2.1 \\ 1.3 \end{Bmatrix} \tag{32}$$

$$b^{(2)} = \{0\} \tag{33}$$

$$\eta = 0.03 \tag{34}$$

Epoch 1

Pass Forward

$$z^{(1)} = \begin{bmatrix} 0.8 & 0.6 & -1.3 \\ 1.6 & 1.2 & -2.6 \\ 2.4 & 1.8 & -3.9 \\ 3.2 & 2.4 & -5.2 \\ 4.0 & 3.0 & -6.5 \end{bmatrix} \tag{35}$$

$$a = \begin{bmatrix} 0.69 & 0.646 & 0.214 \\ 0.832 & 0.769 & 0.069 \\ 0.917 & 0.858 & 0.02 \\ 0.961 & 0.917 & 0.005 \\ 0.982 & 0.953 & 0.002 \end{bmatrix} \tag{36}$$

$$z^{(2)} = \begin{Bmatrix} 1.268 \\ 1.305 \\ 1.341 \\ 1.349 \\ 1.34 \end{Bmatrix} \tag{37}$$

$$\tag{38}$$

$$\begin{aligned} L &= \frac{1}{2 \times 5} \sum (y^* - y)^2 \\ &= \frac{(25 - 1.34)^2 + (16 - 1.349)^2 + (9 - 1.341)^2 + (4 - 1.305)^2 + (1 - 1.268)^2}{10} \\ &= 84.044 \end{aligned} \tag{39}$$

Calculation of the gradients

$$\frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} -9.278 \\ -8.888 \\ -0.079 \end{Bmatrix} \tag{40}$$

$$\frac{\partial L}{\partial b^{(2)}} = -9.68 \tag{41}$$

$$\frac{\partial L}{\partial a} = \begin{bmatrix} 0.183 & -0.113 & 0.07 \\ -1.833 & 1.132 & -0.701 \\ -5.208 & 3.217 & -1.991 \\ -9.963 & 6.154 & -3.809 \\ -16.089 & 9.937 & -6.151 \end{bmatrix} \tag{42}$$

$$\frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} -4.585 \\ 5.67 \\ -0.32 \end{Bmatrix} \tag{43}$$

$$\frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} -1.27 \\ 1.48 \\ -0.10 \end{Bmatrix} \tag{44}$$

Backpropagation (Adjust Weights and Biases)

$$W^{(1)} = W^{(1)} - \eta \frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} 0.8 \\ 0.6 \\ -1.3 \end{Bmatrix} - 0.03 \begin{Bmatrix} -4.585 \\ 5.67 \\ -0.32 \end{Bmatrix} = \begin{Bmatrix} 0.934 \\ 0.43 \\ -1.29 \end{Bmatrix} \tag{45}$$

$$b^{(1)} = b^{(1)} - \eta \frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} - 0.03 \begin{Bmatrix} 1.27 \\ 1.48 \\ -0.10 \end{Bmatrix} = \begin{Bmatrix} 0.038 \\ -0.045 \\ 0.003 \end{Bmatrix} \tag{46}$$

$$W^{(2)} = W^{(2)} - \eta \frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} 3.4 \\ -2.1 \\ 1.3 \end{Bmatrix} - 0.03 \begin{Bmatrix} -9.278 \\ -8.888 \\ -0.079 \end{Bmatrix} = \begin{Bmatrix} 3.678 \\ -1.833 \\ 1.302 \end{Bmatrix} \tag{47}$$

$$b^{(2)} = b^{(2)} - \eta \frac{\partial L}{\partial b^{(2)}} = 0 - 0.03(-9.68) = 0.29 \tag{48}$$

Epoch 2

Pass Forward

$$z^{(1)} = \begin{bmatrix} 0.976 & 0.385 & -1.287 \\ 1.913 & 0.815 & -2.578 \\ 2.851 & 1.245 & -3.868 \\ 3.788 & 1.675 & -5.158 \\ 4.726 & 2.104 & -6.448 \end{bmatrix} \tag{49}$$

$$a = \begin{bmatrix} 0.726 & 0.595 & 0.216 \\ 0.871 & 0.693 & 0.071 \\ 0.945 & 0.776 & 0.02 \\ 0.978 & 0.842 & 0.006 \\ 0.991 & 0.891 & 0.002 \end{bmatrix} \tag{50}$$

$$z^{(2)} = \begin{Bmatrix} 2.152 \\ 2.317 \\ 2.371 \\ 2.351 \\ 2.304 \end{Bmatrix} \tag{51}$$

$$\tag{52}$$

$$\begin{aligned} L &= \frac{1}{2 \times 5} \sum (y^* - y)^2 \\ &= \frac{(25 - 2.304)^2 + (16 - 2.351)^2 + (9 - 2.371)^2 + (4 - 2.317)^2 + (1 - 2.152)^2}{10} \\ &= 74.95 \end{aligned} \tag{53}$$

Calculation of the gradients

$$\frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} -8.548 \\ -7.47 \\ -0.024 \end{Bmatrix} \tag{54}$$

$$\frac{\partial L}{\partial b^{(2)}} = -8.70 \tag{55}$$

$$\frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} -2.46 \\ 8.12 \\ -0.24 \end{Bmatrix} \tag{56}$$

$$\frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} -0.585 \\ 1.923 \\ -0.042 \end{Bmatrix} \tag{57}$$

Backpropagation (Adjust Weights and Biases)

$$W^{(1)} = W^{(1)} - \eta \frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} 0.934 \\ 0.43 \\ -1.29 \end{Bmatrix} - 0.03 \begin{Bmatrix} -2.46 \\ 8.12 \\ -0.24 \end{Bmatrix} = \begin{Bmatrix} 1.011 \\ 0.186 \\ -1.283 \end{Bmatrix} \tag{58}$$

$$b^{(1)} = b^{(1)} - \eta \frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} 0.038 \\ -0.045 \\ 0.003 \end{Bmatrix} - 0.03 \begin{Bmatrix} -0.585 \\ 1.923 \\ -0.042 \end{Bmatrix} = \begin{Bmatrix} 0.056 \\ -0.102 \\ 0.004 \end{Bmatrix} \tag{59}$$

$$W^{(2)} = W^{(2)} - \eta \frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} 3.678 \\ -1.833 \\ 1.302 \end{Bmatrix} - 0.03 \begin{Bmatrix} -8.548 \\ -7.47 \\ -0.024 \end{Bmatrix} = \begin{Bmatrix} 3.935 \\ -1.609 \\ 1.303 \end{Bmatrix} \tag{60}$$

$$b^{(2)} = b^{(2)} - \eta \frac{\partial L}{\partial b^{(2)}} = 0.29 - 0.03(-8.70) = 0.55 \tag{61}$$

Epoch 3

Pass Forward

$$z^{(1)} = \begin{bmatrix} 1.067 & 0.084 & -1.279 \\ 2.079 & 0.27 & -2.562 \\ 3.09 & 0.457 & -3.845 \\ 4.101 & 0.643 & -5.128 \\ 5.113 & 0.829 & -6.411 \end{bmatrix} \tag{62}$$

$$a = \begin{bmatrix} 0.744 & 0.521 & 0.218 \\ 0.889 & 0.567 & 0.072 \\ 0.956 & 0.612 & 0.021 \\ 0.984 & 0.655 & 0.006 \\ 0.994 & 0.696 & 0.002 \end{bmatrix} \tag{63}$$

$$z^{(2)} = \begin{Bmatrix} 2.924 \\ 3.229 \\ 3.357 \\ 3.375 \\ 3.344 \end{Bmatrix} \tag{64}$$

$$\tag{65}$$

$$\begin{aligned} L &= \frac{1}{2 \times 5} \sum (y^* - y)^2 \\ &= \frac{(25 - 3.344)^2 + (16 - 3.375)^2 + (9 - 3.357)^2 + (4 - 3.229)^2 + (1 - 2.924)^2}{10} \\ &= 66.449 \end{aligned} \tag{66}$$

Calculation of the gradients

$$\frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} -7.719 \\ -5.248 \\ 0.027 \end{Bmatrix} \tag{67}$$

$$\frac{\partial L}{\partial b^{(2)}} = -7.754 \tag{68}$$

$$\frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} -1.53 \\ 12.303 \\ -0.155 \end{Bmatrix} \tag{69}$$

$$\frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} -0.217 \\ 2.729 \\ 0.013 \end{Bmatrix} \tag{70}$$

Backpropagation (Adjust Weights and Biases)

$$W^{(1)} = W^{(1)} - \eta \frac{\partial L}{\partial W^{(1)}} = \begin{Bmatrix} 1.011 \\ 0.186 \\ -1.283 \end{Bmatrix} - 0.03 \begin{Bmatrix} -1.53 \\ 12.303 \\ -0.155 \end{Bmatrix} = \begin{Bmatrix} 1.057 \\ -0.183 \\ -1.278 \end{Bmatrix} \tag{71}$$

$$b^{(1)} = b^{(1)} - \eta \frac{\partial L}{\partial b^{(1)}} = \begin{Bmatrix} 0.038 \\ -0.045 \\ 0.003 \end{Bmatrix} - 0.03 \begin{Bmatrix} -0.217 \\ 2.729 \\ 0.013 \end{Bmatrix} = \begin{Bmatrix} 0.062 \\ -0.184 \\ 0.004 \end{Bmatrix} \tag{72}$$

$$W^{(2)} = W^{(2)} - \eta \frac{\partial L}{\partial W^{(2)}} = \begin{Bmatrix} 3.678 \\ -1.833 \\ 1.302 \end{Bmatrix} - 0.03 \begin{Bmatrix} -7.719 \\ -5.248 \\ 0.027 \end{Bmatrix} = \begin{Bmatrix} 4.166 \\ -1.452 \\ 1.302 \end{Bmatrix} \tag{73}$$

$$b^{(2)} = b^{(2)} - \eta \frac{\partial L}{\partial b^{(2)}} = 0.55 - 0.03(-7.754) = 0.784 \tag{74}$$

Epoch 4

Pass Forward

$$z^{(1)} = \begin{bmatrix} 1.12 & -0.367 & -1.275 \\ 2.177 & -0.55 & -2.553 \\ 3.234 & -0.733 & -3.832 \\ 4.291 & -0.915 & -5.11 \\ 5.349 & -1.098 & -6.389 \end{bmatrix} \tag{75}$$

$$a = \begin{bmatrix} 0.754 & 0.409 & 0.218 \\ 0.898 & 0.366 & 0.072 \\ 0.962 & 0.325 & 0.021 \\ 0.986 & 0.286 & 0.006 \\ 0.995 & 0.25 & 0.002 \end{bmatrix} \tag{76}$$

$$z^{(2)} = \begin{Bmatrix} 3.615 \\ 4.089 \\ 4.349 \\ 4.487 \\ 4.57 \end{Bmatrix} \tag{77}$$

$$\tag{78}$$

$$\begin{aligned} L &= \frac{1}{2 \times 5} \sum (y^* - y)^2 \\ &= \frac{(25 - 4.57)^2 + (16 - 4.487)^2 + (9 - 4.349)^2 + (4 - 4.089)^2 + (1 - 3.615)^2}{10} \\ &= 57.843 \end{aligned} \tag{79}$$

- In epoch 1, the loss was recorded at 84.044, while by epoch 4, it had decreased to 57.843. This observation suggests that the model experiences a rapid reduction in error during the initial stages of training, indicative of effective parameter adjustments early in the learning process.

End of training

The training process was continued for 350 epochs. When the loss function is plotted over the epochs, as depicted in the subsequent figures, it is evident that the loss consistently decreases over time. This steady reduction in loss indicates that the optimization algorithm is effectively fine-tuning the model parameters. Despite the modest size of the dataset, the final model output is considered a good fit, demonstrating the network's ability to capture the underlying patterns in the data.

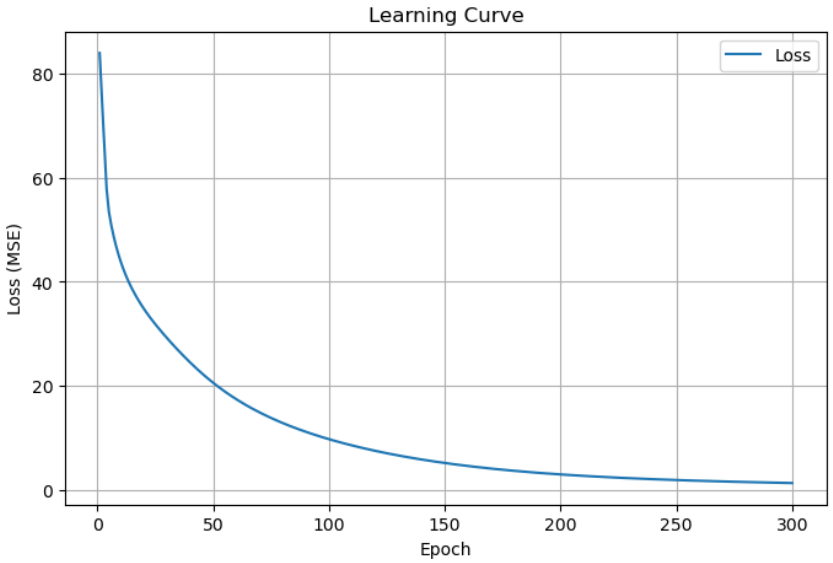


Figure 1: Learning curve for the numerical example

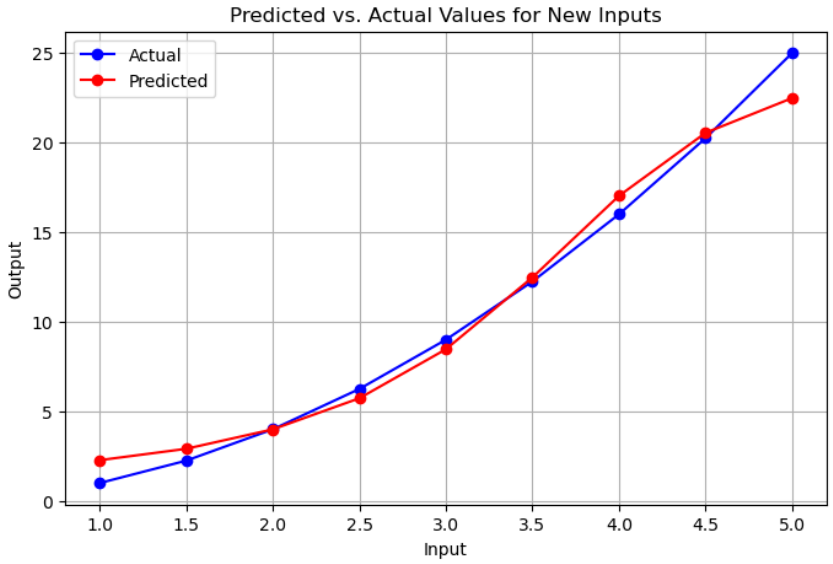


Figure 2: Output of the model

After training, the network converges to a set of optimized parameters. Specifically, the weights and biases have been fine-tuned over the 350 epochs to minimize the loss function. The final learned

parameters, which include the weights and biases are as follows:

$$W^{(1)} = \begin{Bmatrix} 1.961 \\ -1.357 \\ -1.923 \end{Bmatrix} \tag{80}$$

$$b^{(1)} = \begin{Bmatrix} -7.322 \\ 3.239 \\ -0.602 \end{Bmatrix} \tag{81}$$

$$W^{(2)} = \begin{Bmatrix} 17.154 \\ -5.354 \\ 0.552 \end{Bmatrix} \tag{82}$$

$$b^{(2)} = \{6.80\} \tag{83}$$

These values reflect the adjustments made during training to best approximate the underlying function, thereby yielding a model that is well-fitted even with the limited dataset.