Department of Electrical and Electronics Engineering

EEE 443: Neural Networks

Class Project IV Report

Name and Surname: Ali Aral Takak

Student ID: 22001758

Department: EEE

Instructor: Erdem Koyuncu

Problem

The objective of the fourth course project is to implement a simple neural network to accomplish the task of curve fitting using a dataset generated from a given mathematical function. The dataset consists of 300 points, where each input value x_i is drawn uniformly from $[0, 1]$ and each corresponding target value d_i is computed as:

$$d_i = \sin(20x_i) + 3x_i + v_i$$

Where v_i can be considered as additive noise sampled from the interval $[-0.1, 0.1]$. A feedforward neural network with a $1 \times 24 \times 1$ architecture is trained using backpropagation to minimize the Mean-squared Error (MSE) between the network's predictions and the target values. The hidden layer uses the tanh activation function, while the output neuron applies a linear activation. Gradient descent is employed for optimization, with an adaptive learning rate adjustment method to also employ stable convergence.

Implementation and Results

The pseudocode given below explains the complete implementation for the task mention in the previous chapter.

- **Step 1: Initialize Parameters**
    1. Set the number of hidden neurons, $N = 24$.
    2. Define the learning rate η, such as $\eta = 0.01$
    3. Set the number of training epochs, 5000 in our case.
    4. Initialize the weight matrices:
        1. W_1 as a random $(N \times 1)$ matrix.
        2. W_2 as a random $(1 \times N)$ matrix.
    5. Initialize the bias vectors.
        1. B_1 as a zero $(N \times 1)$ vector.
        2. B_2 as a zero (1×1) vector.

- **Step 2: Generate Dataset**
    1. Draw $n = 300$ random numbers x_i uniformly from $[0, 1]$.
    2. Draw $n = 300$ random numbers v_i uniformly from $[-0.1, 0.1]$.
    3. Compute the values using $d_i = \sin(20x_i) + 3x_i + v_i$
- **Step 3: Train the Neural Network using Backpropagation**
    1. For each epoch from 1 to the total number of epochs:
        1. Initialize Mean Squared Error (MSE) as 0.
        2. For each data point (x_i, d_i) from 1 to 300:
            - **Forward Pass:**
                - Compute hidden layer activation $A_1 = \tanh(Z_1)$ and $Z_1 = W_1 \times x_i + B_1$.
                - Compute output layer activation $y_i = Z_2$ and $Z_2 = W_2 \times A_1 + B_2$.
            - **Compute Error:**
                - Compute error as $Error = d_i - y_i$.
                - Adjust Mean-squared Error as $MSE = MSE + Error^2$.
            - **Backpropagation:**
                - Compute gradient for W_2 as $\Delta W_2 = -2 \times Error \times A_1^T$.
                - Compute gradient for B_2 as $\Delta B_2 = -2 \times Error$
                - Compute error propagated back to the hidden layer as $\Delta Z_1 = W_2^T \times (-2 \times Error) \times (1 - tanh^2(Z_1))$
                - Compute gradient for W_1 as $\Delta W_1 = \Delta Z_1 \times x_i^T$.
                - Compute gradient for B_1 as $\Delta B_1 = \Delta Z_1$.
            - **Update Weights and Biases:**
                - $W_1 = W_1 - \eta \times \Delta W_1$
                - $W_2 = W_2 - \eta \times \Delta W_2$
                - $B_1 = B_1 - \eta \times \Delta B_1$
                - $B_2 = B_2 - \eta \times \Delta B_2$
        3. Compute average Mean-squared Error over all data points.

4. Execute adaptive learning adjustment if Mean-squared error increases compared to the previous epoch as $\eta = 0.9 \times \eta$.

- **Step 4: Visualize the Training Process and Results**
  1. Plot the Mean-squared Error vs. Epochs to observe the learning process.
  2. Compute the final fitted function $f(x, w_0)$ using the trained weights.
  3. Plot the fitted function over the original curve.

We initially set our hyperparameters as 5000 epochs and a learning rate of $\eta = 0.01$. The figure given below displays the generated data points and the curve:

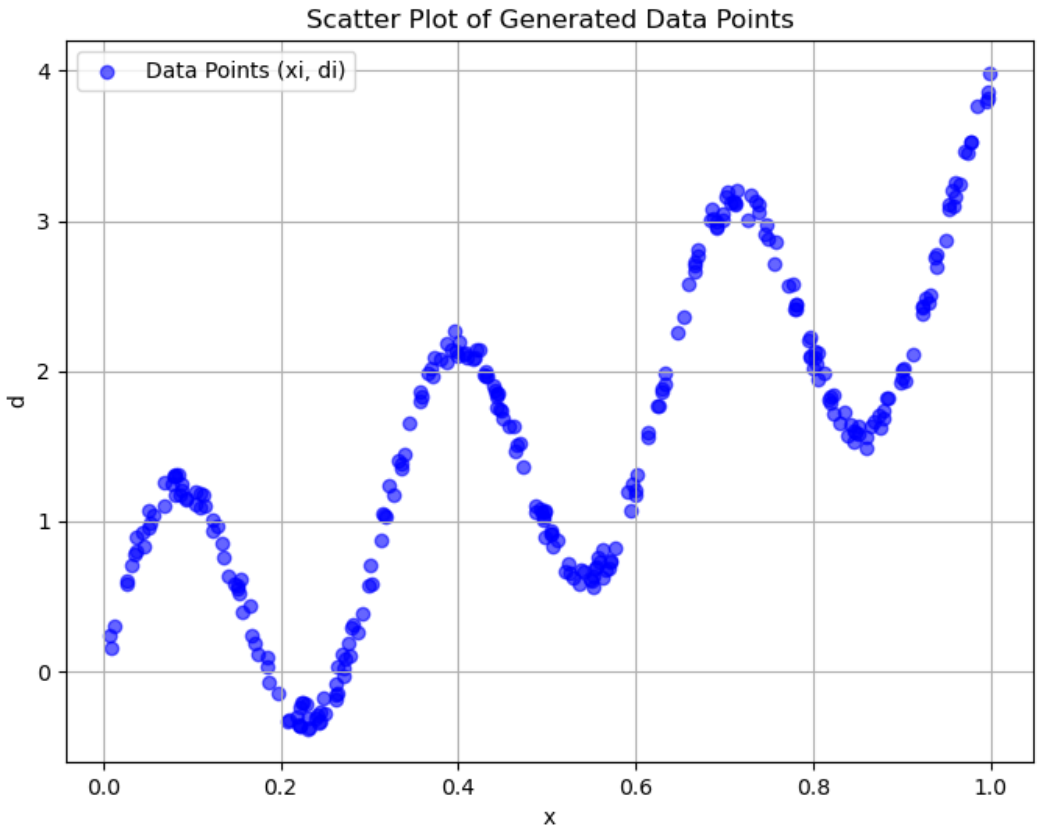

Figure 1: *Generated data points and the curve.*

The figure given below displays the Mean-squared Error versus epoch plot of the Neural Network:

Figure 2: *MSE versus epoch chart for epoch=5000.*

The figure given below displays the curve fitting executed using the trained Neural Network, over the original data points:

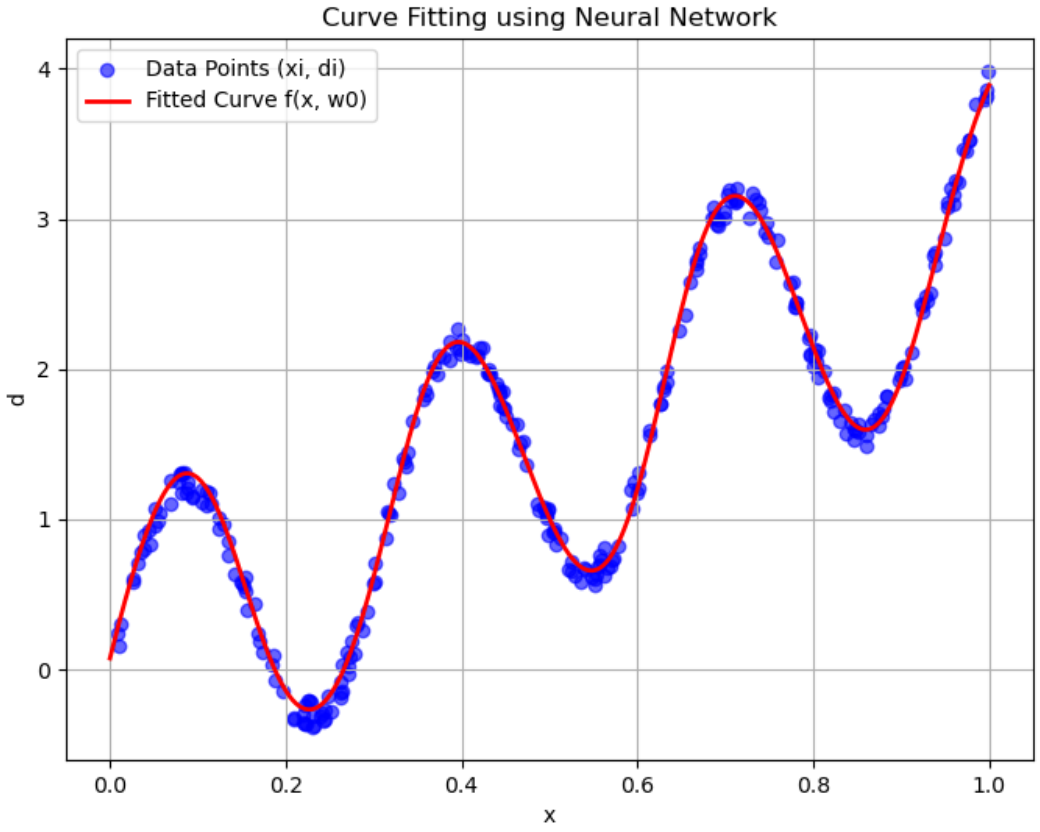

Figure 3: *Curve fitting using the trained Neural Network.*

Conclusion

This project successfully implemented a feedforward neural network to approximate a nonlinear function using a dataset generated from a combination of a sinusoidal term, a linear component, and additive noise. The network architecture consisted of one input neuron, 24 hidden neurons with tanh activation, and one output neuron with a linear activation function. Training was performed using the backpropagation algorithm, whereas an adaptive learning rate ensured stable convergence. Results yielded that the error reduces significantly over time, which indicates the successful adjustment of network parameters. The final fitted curve demonstrated that the designed model has the capabilities and effectiveness for nonlinear curve fitting.