**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GOVERNMENT ENGINEERING COLLEGE THRISSUR**

**MACHINE LEARNING ASSIGNMENT SEMESTER 7**

Submitted By: Alna Elsa Tom

Roll No: TCR22CS008

**TITLE**

**Multi-Output Neural Network for Non-Linear Transformations**

**OBJECTIVE**

To design and train a feedforward neural network that learns to compute three non-linear output functions based on the given input values (x₁, x₂, x₃):

* **Output 1:** x₁ × x₂ × x₃
* **Output 2:** 2 × (x₁ + x₂ + x₃)
* **Output 3:** 3 × (x₁ + x₂ + x₃)

This project aims to explore the capability of neural networks to model complex mathematical relationships using multiple activation functions and output neurons.

**ALGORITHM**

1. **Initialize Parameters:**
   * Set all weights to **1.0** and biases to zero.
   * Define learning rate **η = 0.001** and error threshold **0.001**.
   * Number of epochs: **10** (can be increased if needed).
2. **Forward Propagation:**
   * **Input Layer (Layer 0):** 3 neurons with **Leaky ReLU** activation.
   * **Hidden Layer (Layer 1):** 3 neurons with **Sigmoid** activation.
   * **Output Layer (Layer 2):** 3 neurons with **Leaky ReLU** activation.
   * Compute neuron outputs layer by layer using weighted sums and activations.
3. **Error Calculation:**
   * Compute error = (Target Output – Predicted Output).
   * Use **Mean Squared Error (MSE)** for evaluating performance.
4. **Backward Propagation:**
   * Compute error gradients for each layer.
   * Update weights using:  
     w(new) = w(old) + η × error × input
5. **Repeat:**
   * Perform multiple epochs until the total error falls below the threshold or the maximum number of epochs is reached.
6. **Testing:**
   * Evaluate the trained model using new input samples to verify accurate prediction of all three output functions.

**NETWORK STRUCTURE**

| **LAYER TYPE** | **NO. OF NEURONS** | **ACTIVATION FUNCTION** |
| --- | --- | --- |
| Input Layer | 3 | Leaky ReLU |
| Hidden Layer | 3 | Sigmoid |
| Output Layer | 3 | Leaky ReLU |

**TRAINING PARAMETERS**

* **Weights:** 1.0 (for all connections)
* **Learning Rate:** 0.001
* **Epochs:** 10
* **Error Function:** Mean Squared Error (MSE)
* **Error Threshold:** 0.001

**CHANGES MADE**

* Updated dataset values to represent new non-linear output relationships.
* Added multiple outputs:
  + (x₁ × x₂ × x₃), 2 × (x₁ + x₂ + x₃), 3 × (x₁ + x₂ + x₃)
* Modified activations for diverse learning (Leaky ReLU and Sigmoid).
* Standardized all weights to 1.0 for controlled convergence testing.
* Reduced learning rate for stability and gradual learning.
* Implemented error threshold stopping condition.

SAMPLE INPUT AND OUTPUT

| **Input (x₁, x₂, x₃)** | **Expected Output** | **Predicted Output (Example)** | **Error** |
| --- | --- | --- | --- |
| (1, 2, 3) | (6, 12, 18) | (5.97, 11.98, 17.90) | (0.03, 0.02, 0.10) |
| (2, 3, 4) | (24, 18, 27) | (23.91, 17.95, 26.88) | (0.09, 0.05, 0.12) |

**Training Successful**

**RESULT**

The neural network successfully learned the multi-output mapping functions involving both product and summation terms. The predicted outputs closely matched the target values, confirming that the network effectively modeled non-linear relationships using backpropagation and proper activation selections.

**CONCLUSION**

This experiment demonstrates that even a small feedforward neural network with mixed activation functions can model complex, non-linear transformations. Fine-tuning parameters like learning rate, activation choice, and epoch count helps achieve precise convergence and minimal error.