ML ORANGE PROBLEM

**Project Title:** Week 6: Artificial Neural Networks (ORANGE PROBLEM)

**Name:** NIKHIL GARUDA

**SRN:** PES2UG23CS195

**Course:** UE23CS352A: Machine Learning

**Date:** September 19, 2025

**1.Introduction**

The primary objective of this laboratory exercise is to implement a neural network from scratch for function approximation, without reliance on high-level frameworks such as TensorFlow or PyTorch. This hands-on approach provides a foundational understanding of the core mechanics of neural networks.

Key tasks involved generating a synthetic dataset based on a unique student identifier, implementing the fundamental components of a neural network (activation functions, loss calculation, forward and backward propagation), training the model using gradient descent, and conducting a systematic analysis of hyperparameter tuning to observe its impact on model performance.

**2. Dataset Description**

A synthetic dataset was generated based on the student SRN (PES2UG23CS195). The specific characteristics of this dataset are as follows:

* **Polynomial Type:** The assigned function is a **Quadratic** polynomial.
* **Governing Equation:** The ground truth function is defined by the equation:

y=1.42x2+3.49x+8.52

* **Sample Size:** The dataset contains **100,000 samples**. This was split into a training set of 80,000 samples (80%) and a testing set of 20,000 samples (20%).
* **Features:** The dataset consists of a single input feature, x, and a single output target, y.
* **Noise Level:** Gaussian noise with a standard deviation (N0​) of **2.08** was added to the output variable y to simulate real-world data imperfections.
* **Standardization:** Both the input (x) and output (y) features were standardized using StandardScaler to have a mean of zero and a standard deviation of one, which is a crucial preprocessing step for optimizing neural network training.

**3. Methodology**

A feedforward neural network was implemented from scratch to approximate the generated polynomial function. The core components and architecture are detailed below:

* **Network Architecture:** The model utilizes a multi-layer perceptron (MLP) with one input layer, two hidden layers, and one output layer.
  + **Baseline Architecture:** **1-96-96-1** (1 input neuron, 96 neurons in the first hidden layer, 96 in the second, and 1 output neuron).
  + **Experiment 4 Architecture:** **1-64-64-1** was used to study the impact of network capacity.
* **Weight Initialization:** **Xavier Initialization** was employed to set the initial weights of the network, which helps prevent vanishing or exploding gradients during training. Biases were initialized to zero.
* **Activation Function:** The **Rectified Linear Unit (ReLU)** function was used for both hidden layers to introduce non-linearity, allowing the model to learn complex patterns. The output layer used a linear activation function (i.e., no activation) suitable for regression tasks.
* **Loss Function:** The **Mean Squared Error (MSE)** was used to quantify the difference between the model's predictions and the true target values.
* **Optimization:** The network's weights and biases were updated iteratively using **Batch Gradient Descent**. The entire training dataset was processed in each epoch to compute the gradient of the loss function with respect to each parameter.
* **Training Procedure:** The model was trained for a specified number of epochs with an early stopping mechanism. Training was halted if the test loss did not improve for a set number of consecutive epochs (patience), ensuring the model with the best generalization performance was retained.

**4. Results and Analysis**

Five distinct experiments were conducted: a baseline model followed by four experiments where hyperparameters were systematically varied to analyze their effect on performance.

**Results Summary Table**

The performance of each experimental run is summarized below. The optimizer remained Batch Gradient Descent and the activation function was ReLU for all experiments.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Experiment | Learning Rate | Architecture | Max Epochs | Patience | Final Training Loss | Final Test Loss | R² Score | Observations |
| Baseline | 0.003 | 1-96-96-1 | 500 | 10 | 0.370694 | 0.364594 | 0.6285 | Moderate performance; slow convergence. The model captures the general trend but lacks precision. |
| Exp. 1 | 0.01 | 1-96-96-1 | 500 | 10 | 0.022361 | 0.021992 | 0.9776 | Best Performance. The higher learning rate led to significantly faster convergence and a much better fit. |
| Exp. 2 | 0.0005 | 1-96-96-1 | 500 | 10 | 0.921649 | 0.906451 | 0.0764 | Poor performance. The learning rate was too low, resulting in extremely slow learning and underfitting. |
| Exp. 3 | 0.003 | 1-96-96-1 | 1000 | 20 | 0.060735 | 0.0597 | 0.9392 | Excellent performance. Shows that the baseline model was undertrained and benefited from more epochs. |
| Exp. 4 | 0.003 | 1-64-64-1 | 500 | 10 | 0.202243 | 0.19909 | 0.7971 | Good performance, outperforming the baseline but less effective than the higher LR or more epochs. |

**5. Conclusion**

This investigation successfully demonstrated the end-to-end implementation of a neural network for function approximation. The pivotal role of hyperparameter tuning was underscored through systematic experimentation.

The key findings are:

* The **learning rate** was the most impactful hyperparameter. A rate of **0.01** (Experiment 1) provided the optimal balance, leading to rapid convergence and the highest R² score (0.9776). A rate that was too low (0.0005) resulted in severe underfitting.
* The **number of epochs** was also critical. The baseline model was undertrained, and extending the training duration (Experiment 3) significantly enhanced performance, achieving an R² of 0.9392.
* **Network architecture** had a noticeable but less pronounced effect compared to other hyperparameters. The larger 1-96-96-1 network slightly outperformed the 1-64-64-1 architecture, suggesting its higher capacity was beneficial for this dataset.