Neural Network Regression on Polynomial Data

Name: Mohammed Ehan Sheikh
SRN: PES2UG23CS345
Section: F
Course: Machine Learning
Date: 19 SEP 2025

1. Introduction

Purpose of the Lab

This lab focuses on implementing a neural network from scratch to perform regression on a synthetically generated polynomial dataset. The objectives include:
• Implementing forward and backward propagation
• Applying Xavier initialization for weight initialization
• Training the network using mini-batch gradient descent with Adam optimization
• Experimenting with different hyperparameters to understand their effects on model performance

Tasks Performed

1. Generated a polynomial dataset based on student SRN
2. Implemented activation functions (ReLU and Tanh) and their derivatives
3. Implemented Xavier initialization for weight initialization
4. Built forward and backward propagation functions
5. Trained the neural network with various hyperparameter configurations
6. Analyzed results and compared different experimental setups

2. Dataset Description

Type of Polynomial Assigned

The polynomial assigned is a quadratic function:
$y = 1.06x^2 + 6.32x + 9.97$

Number of Samples and Features

• Number of samples: 100,000
• Number of features: 1 (univariate input)

Noise Level

- Noise standard deviation: 2.10 (Gaussian noise)

3. Methodology

Neural Network Architecture

- Input layer: 1 neuron
- Hidden layer 1: 32 neurons
- Hidden layer 2: 72 neurons
- Output layer: 1 neuron

Activation Functions

- ReLU and Tanh (for experimentation)

Weight Initialization

- Xavier initialization for weights, biases initialized to zero.

Optimization

- Adam optimizer with learning rate 0.005 (assigned) and other hyperparameters (β_1=0.9, β_2=0.999, ε=1e-8).
- Mini-batch gradient descent with batch size 32.

Training

- Epochs: 10 (baseline), up to 40 for experiments.
- Early stopping with patience of 10 epochs

4. Results and Analysis

Training Loss Curve

The model achieved rapid convergence with very low training and test losses.

Final Test MSE

- Baseline Model (ReLU, Adam, LR=0.001, Batch Size=32): 0.000005

Plot of Predicted vs. Actual Values

The predicted values closely match the actual values, indicating excellent model performance.

Discussion on Performance

- The model achieves very low training and test loss, indicating good generalization.
- The predicted values closely match the actual values, as seen in the scatter plot.
- No significant overfitting or underfitting is observed; the model performs well on both training and test data.

Experiment	Learning Rate	Epochs	Activation	Train Loss	Test Loss	Train Acc	Test Acc
Baseline	0.001	10	ReLU	0.000017	0.000005	0.994	0.994
Exp1_HigherLR	0.005	19*	ReLU	0.000081	0.000007	0.987	0.987
Exp2_LargerBatch	0.001	20	ReLU	0.000006	0.000030	0.973	0.975
Exp3_MoreEpochs	0.001	23*	ReLU	0.000034	0.000002	0.988	0.988
Exp4_TanhActivation	0.001	20	Tanh	0.000042	0.000162	0.944	0.945

*Early stopping occurred

Performance Metrics

- **R^2 Score:** 1.0000 (perfect fit)
- **Final Test Loss:** 0.000005
- **Final Test Accuracy:** 99.4%

Prediction Test

For input x = 90.2:
- **Neural Network Prediction:** 9,202.94
- **Ground Truth (formula):** 9,201.77
- **Absolute Error:** 1.16
- **Relative Error:** 0.013%

Conclusion

The neural network successfully approximates the quadratic polynomial function with high accuracy. The baseline model (ReLU activation, Adam optimizer, LR=0.001) performs exceptionally well, achieving very low MSE and high accuracy.

Hyperparameter experiments show that:
- Higher learning rates (0.005) can achieve similar performance but may be less stable
- Larger batch sizes improve stability but may slightly reduce final accuracy
- More training epochs can further improve performance when early stopping is used effectively
- ReLU activation consistently outperforms Tanh activation for this regression task

The model generalizes excellently to unseen data, with $R^2 = 1.0000$ indicating near-perfect fit to the underlying quadratic relationship. The narrow-to-wide architecture (32→72 neurons) assigned based on the SRN proves highly effective for this polynomial regression task.