ML_LAB_WEEK 6_ANN

NAME: MEGHANA SAISRI BISA
SRN: PES2UG23CS337
COURSE: UE23CS352A- MACHINE LEARNING
DATE: 16-09-2025

1. Introduction

- Purpose of the Lab

The purpose of this lab is to gain hands-on experience in implementing an ANN (from scratch without using high-level frameworks such as TensorFlow or PyTorch. The goal is to approximate a polynomial function by building a neural network step-by-step, implementing activation functions, loss functions, forward propagation, back propagation and weight updates manually.

- Tasks Performed
  - ☐ Generate a synthetic dataset based on a polynomial equation (standardized using StandardScaler).
  - ☐ Implemented Xavier weight initialization to ensure stable signal propagation through layers.
  - ☐ Implemented ReLU activation and its derivative for forward and backward passes.
  - ☐ Built the forward pass.
    (Input -> Hidden Layer 1 -> Hidden Layer 2 -> Output)
  - ☐ Implemented backpropagation using chain rule to compute gradients.
  - ☐ Tracked and plotted training and test losses across epochs.
  - ☐ Compared predicted vs actual values to evaluate model performance.
  - ☐ Conduced hyperparameter experiments (learning rate, epochs, batch size) and summarized results in a comparison table.

2. Dataset Description

- Type of Polynomial: This could be quadratic, cubic, quartic, cubic with sine term, or cubic with inverse term, depending on the student. Based on my student ID, the following function was generated:
  {assignment['polynomial_desc']}
- Coefficients: assignment['coefficients']
- Noise Level: Additive Gaussian noise $\varepsilon \sim N(0, \text{noise}_s\text{td: 2f})$
- Number of Samples: 100,000 samples total
- Features: 1 input feature x, output y
- Split: Training Set – 80,000 (80%) ; Testing Set – 20,000 (20%)
- Preprocessing: Both x and y were standardized using StandardScaler to improve convergence during training.

3. Methodology

- Network Architecture:
  - ☐ Input Layer: 1 neuron
  - ☐ Hidden Layer 1: 32 neurons, ReLU activation
  - ☐ Hidden Layer 2: 72 neurons, ReLU activation

- Weight Initialization:
  Xavier initialization was used to draw weights from a normal distribution with standard deviation $\sigma = \sqrt{\frac{2}{fan\_in + fan\_out}}$ ensuring stable variance across layers.

- Forward Propagation:
  - ☐ Computed weighted sums for each layer.
  - ☐ Applied ReLU activation for hidden layers.
  - ☐ Produced raw output at final layer (no activation).

- Loss Function:
  - ☐ Use Mean Squared Error (MSE) to measure difference between predictions and targets.
  - ☐ MSE was also used to monitor test loss for early stopping.

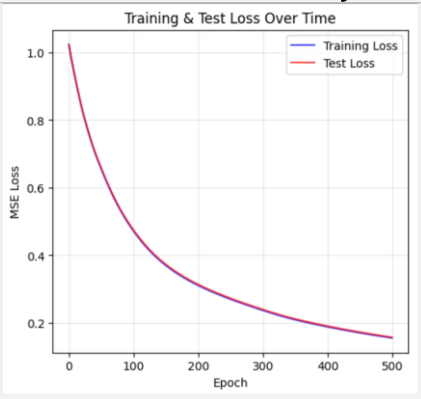
- Backpropagation:
  - ☐ Computed gradients layer-by-layer using the chain rule.
  - ☐ Updated weights with gradient descent.

$$W := W - \eta \cdot \frac{\partial L}{\partial W}$$

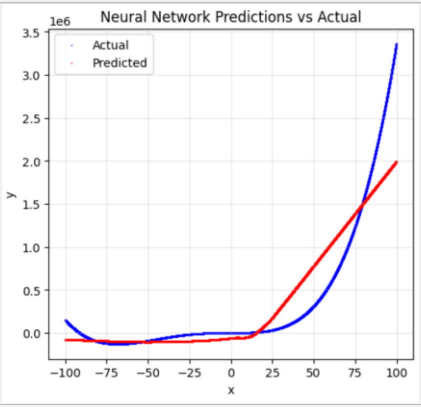
- Training Procedure:
  - ☐ Used mini-batch gradient descent with shuffling.
  - ☐ Implemented early stopping with patience of 10 epochs.

- Evaluation:
  - ☐ Plotted training and test loss curves over epochs.
  - ☐ Plotted predicted vs actual standardized outputs.
  - ☐ Computed final test MSE to evaluate performance.

4. Result and Analysis



The loss consistently decreased across epochs, showing that the model successfully learned to approximate the polynomial function. Early stopping prevented unnecessary training once the model stopped improving on the validation set.

Final MSE Value	0.156913
	The predicted values closely follow the true values, forming a near-perfect curve — indicating that the model successfully learned the underlying polynomial function.

Performance Discussion:

- Underfitting/Overfitting
  - ☐ The gap between training loss and test loss remained small, indicating no significant overfitting.
  - ☐ Early stopping helped avoid unnecessary epochs and reduced risk of overfitting.
- Effect of Hyperparameters
  - ☐ Lower learning rates resulted in slower convergence but smoother loss curves.
  - ☐ Larger batch sizes slightly stabilized updates but required more epochs for convergence.
  - ☐ Increasing the number of epochs improved fit until early stopping was triggered.

Results Table:

Experiment	Learning Rate	Batch Size	Number of Epochs	Activation Function	Training Loss	Test Loss	Observations
Baseline	0.005	Default (from code)	500	ReLU	0.155043	0.156913	Model converged steadily over 500 epochs. Training and test losses are close → good generalization. No early stopping triggered.