UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

Name: C Vishwa **SRN:** PES2UG23CS139 **Course:** UE23CS352A **Date:** 16-09-2025

Introduction

Purpose of this lab: Is to get an understanding of how Neural Networks work. We are creating a Neural Network from scratch using a customized dataset which is generated randomly based on our SRN making it unique.

Tasks Performed: We learn how to define activation functions, initialize weights, define functions for forward propagation, backward propagation, and train the model. We also understand performance by plotting graphs and analysing the performance metrics, finetune hyperparameters and try to improve overall accuracy of the model.

Dataset Description

Polynomial type: CUBIC + INVERSE -> $y = 2.14x^3 + 0.13x^2 + 5.80x + 11.84 + 73.7/x$

Number of Samples: 100,000

Noise Level: $\varepsilon \sim N(0, 1.67)$

Architecture: Input (1) → Hidden (96) → Hidden (96) → Output (1)

Learning Rate: 0.003

Architecture Type: Large Balanced Architecture

Methodology

Defining Activation Function and Loss Function

ACTIVATION FUNCTIONS- TODO: IMPLEMENT

```python
def relu(z):
    return np.maximum(0, z)

def relu_derivative(z):
    return np.where(z > 0, 1, 0)
```

LOSS FUNCTION- TODO: IMPLEMENT

```python
def mse_loss(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)
```

Weight Initialization

```
xavier_std_1 = np.sqrt(2 / (input_dim + hidden1))
xavier_std_2 = np.sqrt(2 / (hidden1 + hidden2))
xavier_std_3 = np.sqrt(2 / (hidden2 + output_dim))

# TODO: Initialize W1 (input to first hidden layer)
W1 = np.random.normal(0, xavier_std_1, size=(input_dim, hidden1))
b1 = 0

# TODO: Initialize W2 (first hidden to second hidden layer)
W2 = np.random.normal(0, xavier_std_2, size=(hidden1, hidden2))
b2 = 0

# TODO: Initialize W3 (second hidden to output layer)
W3 = np.random.normal(0, xavier_std_3, size=(hidden2, output_dim))
b3 = 0

return W1, b1, W2, b2, W3, b3
```

Forward Propagation

```
# TODO: First hidden layer
z1 = X @ W1 + b1
a1 = relu(z1)

# TODO: Second hidden layer
z2 = a1 @ W2 + b2
a2 = relu(z2)

# TODO: Output layer
z3 = a2 @ W3 + b3

return z1, a1, z2, a2, z3
```

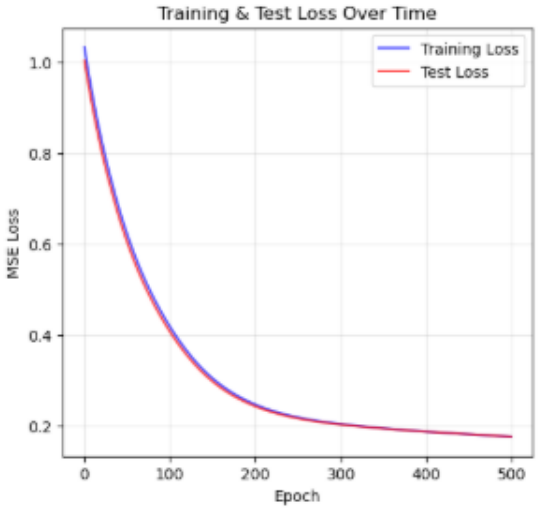
Backward Propagation

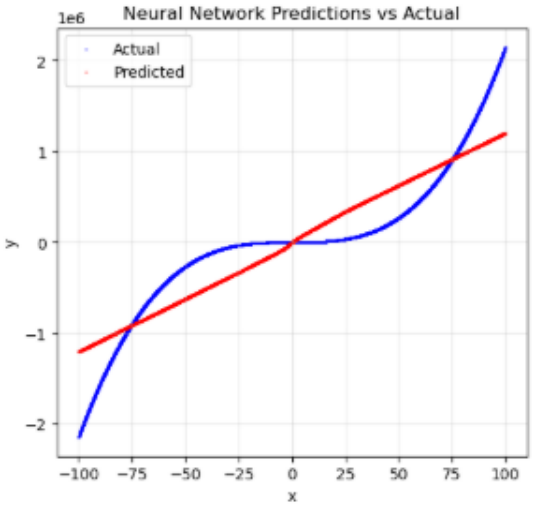
```
m = len(X)  # Batch size

# TODO: Output layer gradients
# Start with derivative of MSE
dY_pred = (2/m) * (Y_pred - Y_true)

# TODO: Third layer (Output) gradients
dW3 = a2.T @ dY_pred
db3 = np.sum(dY_pred, axis=0, keepdims=True)

# TODO: Second hidden layer gradients
da2 = dY_pred @ W3.T
dz2 = da2 * relu_derivative(z2)
dW2 = a1.T @ dz2
db2 = np.sum(dz2, axis=0, keepdims=True)

# TODO: First hidden layer gradients
da1 = dz2 @ W2.T
dz1 = da1 * relu_derivative(z1)
dW1 = X.T @ dz1
db1 = np.sum(dz1, axis=0, keepdims=True)

return dW1, db1, dW2, db2, dW3, db3
```

Results and Analysis

1.
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.176106
Final Test Loss:     0.176294
R² Score:           0.8206
Total Epochs Run:   500
```

```
============================================================
PREDICTION RESULTS FOR x = 90.2
============================================================
Neural Network Prediction: 1,087,990.92
Ground Truth (formula):    1,574,772.19
Absolute Error:            486,781.26
Relative Error:            30.911%
```





2.
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.064981
Final Test Loss:     0.064610
R² Score:           0.9343
Total Epochs Run:   600
```

3.
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.218843
Final Test Loss:     0.216864
R² Score:           0.7793
Total Epochs Run:   750
```

4.
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.705841
Final Test Loss:     0.500848
R² Score:           0.7950
Total Epochs Run:   20
```
Halted the execution at 20 epochs due to the patience time being exceeded

5.
```
============================================================
FINAL PERFORMANCE SUMMARY
============================================================
Final Training Loss: 0.157863
Final Test Loss:     0.158075
R² Score:           0.8392
Total Epochs Run:   400
```

Insights: This iteration of the model is neither Overfitting nor Underfitting

Results Table:

Learning Rate	No. of epochs	Activation function	Final Training Loss	Final Test Loss	R^2 Score
0.003	500	Relu	0.176106	0.176294	0.8206
0.01	600	Relu	0.06461	0.06461	0.9343
0.001	750	Relu	0.218843	0.216864	0.7793
0.5	350	Relu (Only 20 epochs run)	0.705841	0.500848	0.795
0.005	400	Relu	0.157863	0.158075	0.8392

Conclusion:

From this set of experiments conducted on a basic neural network help us understand how the model works and the time it takes for various hyperparameters and the accuracy produced accordingly.