Task 2 Report: Linear Regression Using Multi-Layer Neural Network from Scratch

1. Introduction

This report documents the implementation of a linear regression model using a multi-layer neural network built entirely from scratch in Python.

The aim of this task is to predict the median value of owner-occupied homes (MEDV) from the Boston Housing Dataset using two input features:

Crime Rate (CRIM) and Average Number of Rooms (RM).

This experiment helps in understanding how neural networks can be used for regression problems and how different optimization techniques influence training and performance.

2. Dataset Description

The Boston Housing Dataset consists of housing-related attributes collected from various suburbs of Boston.

For this task:

- Input Features:
  - CRIM: Per capita crime rate by town
  - RM: Average number of rooms per dwelling
- Target Variable:
  - MEDV: Median value of owner-occupied homes

Only the above two features were selected to keep the model simple and interpretable.

3. Data Preprocessing

Before training the neural network, the data was preprocessed as follows:

• Feature normalization was applied using mean and standard deviation scaling to bring CRIM and RM onto a similar scale.

• The dataset was randomly split into:

- 80% training data

- 20% testing data

Normalization ensured faster convergence and improved numerical stability during training.

4. Neural Network Architecture

A fully connected feedforward neural network was implemented with the following architecture:

• Input Layer:

- 2 neurons corresponding to CRIM and RM

• Hidden Layer 1:

- 5 neurons

- ReLU activation function

• Hidden Layer 2:

- 3 neurons

- ReLU activation function

• Output Layer:

- 1 neuron

- Linear activation function to predict continuous house prices

This architecture allows the model to learn non-linear relationships between the input features and the target.

5. Training Methodology

The network was trained using forward propagation and backpropagation.

Mean Squared Error (MSE) was used as the loss function, defined as the average of the squared differences between predicted and actual values.

The model was trained for 1000 epochs, and the loss was recorded at each epoch to analyze convergence behavior.

Different learning rates (0.01 and 0.001) were experimented with to observe their effect on training stability and convergence speed.

6. Optimization Techniques

Three optimization methods were implemented and compared:

a) Basic Gradient Descent:

Parameters were updated using the negative gradient of the loss function.

This optimizer showed slow convergence and higher final loss.

b) Momentum:

Momentum introduced a velocity term that accumulated past gradients.

This resulted in faster convergence and smoother loss reduction compared to basic gradient descent.

c) Adam Optimizer:

Adam combined momentum and adaptive learning rates.

It demonstrated the fastest convergence and achieved the lowest training and testing loss.

Among all optimizers, Adam provided the most stable and efficient training.

7. Results

After training, the model was evaluated on the test dataset.

• Test Mean Squared Error (MSE):

The MSE value indicated that the model predictions were reasonably close to the actual MEDV values.

Lower MSE values confirmed better regression performance.

• Predicted vs Actual Plot:

The scatter plot of predicted versus actual MEDV values showed points closely aligned along the diagonal.

This indicates that the model learned a strong relationship between input features and house prices.

These results demonstrate that the neural network successfully captured the underlying trend in the data.

8. Interpretation of Results

• Higher RM values generally corresponded to higher predicted house prices.

• Higher CRIM values negatively influenced house prices.

• Adam optimizer minimized oscillations during training and reached optimal weights faster.

• Lower learning rates produced smoother loss curves but required more epochs to converge.

9. Bonus Experiments

• Additional Hidden Layer:

A third hidden layer with 2 neurons was added to the network.

This slightly improved training accuracy but increased model complexity with marginal gains.

• L2 Regularization:

L2 regularization (weight decay) was implemented to penalize large weights.

This reduced overfitting and improved generalization on test data.

10. Conclusion

This task successfully demonstrated the implementation of a multi-layer neural network for linear regression from scratch.

The experiment highlighted the importance of data preprocessing, optimizer selection, and learning rate tuning.

Advanced optimizers such as Adam significantly improved convergence speed and final performance.

Overall, the neural network provided accurate predictions of house prices using only two input features.