AIM-829 NLP
Assignment 2
Implementation of Multi-Layered Perceptron from scratch and Sentiment Analysis: using libraries.

Aditya Saraf

March 17, 2025

1 Introduction

This assignment consists of two primary tasks:

- Implementing a Multi-Layer Perceptron (MLP) from Scratch for the XOR Logic Gate: We built an MLP in Python without relying on external deep learning libraries. The purpose was to demonstrate an understanding of the underlying algorithms for forward propagation, backpropagation, and weight updates by solving the classic XOR problem.
- Implementing an MLP (with Libraries) for Sentiment Analysis: In this task, we employed standard Python libraries (such as TensorFlow/Keras) to build and train a neural network for classifying sentiment. The dataset contains text data with 279 unique sentiment classes. We followed standard vectorization methods to extract vocabulary and preprocessed the data for training.

2 Assumptions

- The corpus provided us consist only of one dataset respectively, therefore we have to divide the same data for training and testing.
- As the dataset being small the neural network could not be trained properly as it's always leading to overfitting.

3 Dataset Details

3.1 Task 1 – XOR Logic Gate

- **Dataset:** The XOR dataset consist of 1000 rows of X1 and X2 consist of real numbers between 0 and 1, and label for output of XOR operation between X1 and X2.

3.2 Task 2 – Sentiment Analysis

- **Dataset:** The dataset consist of 732 rows. Each row contains a text sample, other related columns and its corresponding sentiment tag. The dataset has 279 unique sentiment classes.
- **Preprocessing:** We loaded the dataset using Python libraries, explored its structure, and split the data into text (`df_X`) and sentiment labels (`df_Y`). Tokenization was applied to build the vocabulary and convert the text into numerical features.

4 Methodology

4.1 Task 1: MLP from Scratch for XOR

- **Architecture:** A neural network was built using classes with different layers and activation function (e.g., sigmoid or tanh).
- **Implementation:** The MLP was implemented from scratch in Python. The forward pass computes activations, and the backward pass computes gradients for weight updates using mean squared error as the loss function.
- **Results:** The network was successfully trained on the XOR dataset, correctly learning the XOR logic.

4.2 Task 2: MLP with Libraries for Sentiment Analysis

- **Preprocessing:**
  - Un-necessary columns were removed to train the model. Just used 'Text' for input and 'Sentiment' for output.
  - Text in `df_X` was tokenized and padded, building a vocabulary of the whole dataset.
  - Sentiment labels in `df_Y` were encoded as integers (from 0 to 278).
- **Network Architecture:**
  - **Embedding Layer:** Maps words to dense vectors.
  - **Flatten Layer:** Converts the output of the embedding layer into a single vector.
  - **Dense Layers:** Several fully-connected layers with ReLU or sigmoid activation to learn complex features.
  - **Output Layer:** A Dense layer with 279 neurons and softmax activation to produce probability distributions over sentiment classes.
- **Loss Function and Optimizer:**
  - **Loss Function:** `sparse_categorical_crossentropy` was used since the labels are integer encoded.

  - **Optimizer:** Adam was selected for its adaptive learning rates and robust performance.

- **Training and Validation:**
  - The model was trained on the training set and validated on a hold-out set.
  - Training metrics such as accuracy were monitored, and the process was visualized using matplotlib and seaborn.

- **Evaluation:**
  - The model was evaluated on the test set using accuracy.

5 Experimental Results

5.1 Task 1 (XOR)

The MLP implemented from scratch successfully learned the XOR function. After several epochs, the model correctly classified all XOR inputs, validating our forward and backward propagation implementations, with an accuracy : 98.2 %.

5.2 Task 2 (Sentiment Analysis)

- **Training Performance:** Training and validation accuracy were tracked over epochs. The best model achieved a test accuracy of 18.23 % after 10 epochs.

6 Discussion

- **Reason for low accuracy in Task 2:** As the dataset was too small to train neural network with various sentiments having only 1 - 5 entries leading to not able to train the model properly, leading to overfitting. We can't remove these entries as removing this would make the dataset more small.

- **Model Architecture:** For Task 2, multiple architectures were experimented with. The chosen architecture, including an embedding layer followed by dense layers, balanced complexity and generalization well.

- **Loss Function and Optimizer:** The use of `sparse_categorical_crossentropy` and Adam contributed to robust and fast convergence.

- **Future Work:** Future improvements include exploring alternative architectures (e.g., convolutional or recurrent neural networks) and further hyperparameter tuning.

7 Conclusion

This assignment allowed us to delve into the fundamentals of neural networks by implementing an MLP from scratch for the XOR logic gate and applying deep learning libraries to a multi-class sentiment analysis task. By leveraging standard vectorization,

careful network design, and adaptive optimizers like Adam, we achieved promising results in sentiment classification. Future work will explore advanced architectures and further tuning to enhance performance.

8 Team-wise Contribution

As a sole team member of the team, I individually handled all the work with some help from [1].

[1] to understand and write code for MLP from scratch, referred : `https://www.youtube.com/watch?v=pauPCy_s0Ok`