Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Course No: CSE 4204

Course Title: Sessional Based on CSE 4203

Experiment No. 3

Name of the Experiment: Design and implementation of Multi-layer Neural Networks algorithm.

Submitted by:

Ashraf-Ul-Alam

Roll: 1803070

Section: B

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Submitted to:

Rizoan Toufiq

Assistant Professor

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Date of Submission: 02 December 2023

Contents

1 Introduction

This report focuses on implementing the Multi-layer Neural Networks algorithm, employing Back-propagation learning. It investigates the application of these networks in analyzing the "Indian Liver Patient Records"[1] dataset and tackling the XOR problem. By examining both practical datasets and a fundamental logical challenge, this report aims to showcase the efficacy and versatility of multi-layer neural networks in diverse scenarios.

2 Dataset

2.1 Dataset Analysis

This analysis began by loading the dataset from the "indian_liver_patient.csv" [1] file. The dataset contains information on various attributes related to liver health, such as age, gender, bilirubin levels, liver enzymes, and more. The dataset has 583 entries and 11 columns.

- Age: An integer representing the patient's age.
- Gender: Categorical feature (0 for female, 1 for male).
- Total Bilirubin, Direct Bilirubin, Alkaline Phosphotase, Alamine Aminotransferase, Aspartate Aminotransferase: Numeric measurements.
- Total Proteins, Albumin, Albumin and Globulin Ratio: Numeric measurements.
- Dataset: The target variable (1 for liver disease, 2 for no liver disease).

2.2 Feature Processing

The original dataset contained the 'Gender' feature as a categorical variable with string values representing male and female. To make this feature compatible with the perceptron learning algorithm, which requires numerical input, the 'Gender' data was transformed into a numeric format using Scikit-Learn's Label Encoder, assigning '0' for female and '1' for male. Simultaneously, the 'Dataset' column is transformed to have 0 represent the presence of liver disease and 1 indicate the absence of liver disease.
This encoding step is crucial because it converts categorical data into a numerical format that neural networks can understand. It's essential for the algorithm to process this data accurately and seamlessly, ensuring it integrates well and performs effectively within the model.

2.3 Handling Missing Values

The missing values within the dataset were also checked and it was found that the "Albumin_and_Globulin_Ratio" column had four missing values. To address this, the missing values were filled with the mean of the column.

2.4 Correlation Analysis

A correlation analysis was performed To ensure that the selected features were suitable for the perceptron learning algorithm. The goal was to visually represent any significant connections between variables using a heatmap [1]. Recognizing these relationships is important for a perceptron model because it depends on the independence of features to make accurate predictions. The findings revealed no noteworthy correlations, suggesting that each variable independently affects the prediction of liver disease. This independence is advantageous for the perceptron algorithm as it reduces the chance of multicollinearity, leading to more dependable and understandable model results.

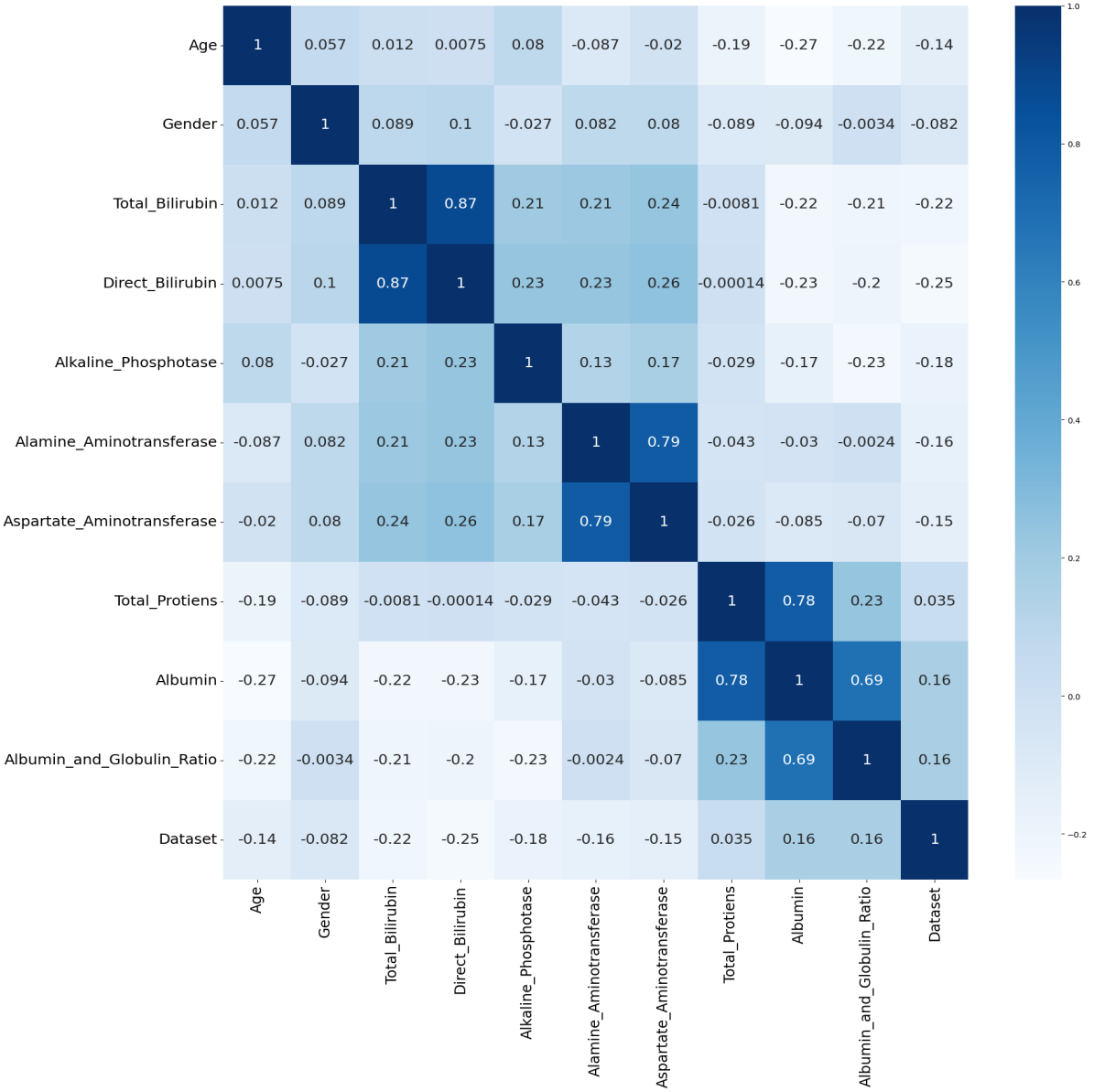

Figure 1: Correlation Heatmap

2.5 Data Type Conversion

To maintain consistency and avoid data loss, all the features were converted to the float data type.

2.6 Data Imbalance

A significant class imbalance can be observed in the target variable ("Dataset") with more instances of one class (0) compared to the other (1). To address this imbalance, Synthetic Minority Over-sampling Technique (SMOTE) was applied to oversample the minority class. The following table [1] illustrates the class distribution before and after applying SMOTE:

Table 1: Class Distribution Before and After SMOTE

Class	**Count Before SMOTE**	**Count After SMOTE**
0	416	416
1	167	416

In the original dataset, class 0 (indicating patients with liver disease) had 416 instances, while class 1 (indicating patients without liver disease) had only 167 instances. After applying SMOTE, both classes have an equal number of instances (416), effectively addressing the class imbalance issue.

Balancing the dataset through SMOTE ensures that the machine learning model is trained on a more representative dataset, preventing bias towards the majority class and improving its ability to make accurate predictions for both classes.

2.7 Z-score Normalization using StandardScaler

The dataset preprocessing involves employing 'StandardScaler' from 'sklearn.preprocessing' to perform Z-score normalization. This process standardizes each feature to possess a mean of 0 and a standard deviation of 1. By centering the data around 0 and scaling it using standard deviations, this method ensures that all features contribute equally to prevent any single feature from dominating the learning process. Z-score normalization is critical for enhancing model convergence rates and maintaining fair feature contributions, particularly in machine learning algorithms reliant on gradient descent or distance-based calculations.

2.8 Train-Test-Validation Split

The training, testing, and validation split was performed using the train_test_split method from sklearn.model_selection in Python. The dataset was preprocessed with StandardScaler to standardize the features.

The data was split into training and testing sets initially using train_test_split. Subsequently, the training set was further divided into training and validation subsets to optimize the model's performance. The test_size parameter was set to 0.15 for both the initial test split and the subsequent division to maintain a 15% proportion for the testing and validation sets.

3 Methodology

The Multi-layer Perceptron (MLP) is a type of artificial neural network composed of multiple layers of interconnected nodes, capable of learning and modeling complex relationships within data. This algorithm outlines the training process of an MLP, where weights and biases are adjusted iteratively to minimize errors between predicted and actual outputs.

3.1 Algorithm: Multi-layer Perceptron

The algorithm utilizes forward propagation to compute the output of each layer, backpropagation to calculate errors, and weight updates based on these errors using a specified learning rate. The sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ introduces non-linearity, aiding in capturing complex patterns in data.

Initialization:

- Randomly initialize weights W_{ij} and biases θ with small values.
- Define the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$.

Training Loop:

- For each epoch in range(epochs):
  - **Forward Propagation:**
    - Calculate hidden layer output: hidden_layer_output $= \sigma(X_{\text{train}} \cdot W_{\text{input\_hidden}})$.
    - Calculate output layer output: output_train $= \sigma(\text{hidden\_layer\_output} \cdot W_{\text{hidden\_output}})$.
  - **Error Calculation:**
    - Calculate output error: output_error_train $= y_{\text{train}} -$ output_train.
    - Calculate hidden layer error: hidden_error_train $=$ output_error_train $\cdot W^T_{\text{hidden\_output}}$.
  - **Backpropagation:**
    - Compute output layer delta: output_delta_train $=$ learning_rate $\times$ output_error_train $\times$ output_train $\times (1 -$ output_train$)$.
    - Compute hidden layer delta: hidden_delta_train $=$ learning_rate $\times$ hidden_error_train $\times$ $\sigma'(\text{hidden\_layer\_output})$.

- **Update Weights:**
  - Update weights for hidden-to-output layer: $W_{\text{hidden\_output}} += \text{hidden\_layer\_output}^T \cdot$ output_delta_train.
  - Update weights for input-to-hidden layer: $W_{\text{input_hidden}} += X_{\text{train}}^T \cdot$hidden_delta_train.
- **Evaluation:**
  - Calculate training and validation loss.
  - Calculate training and validation accuracies.
  - Print epoch number, training loss, training accuracy, validation loss, and validation accuracy.

Output:

- Return lists containing training and validation accuracies, as well as losses, for analysis and visualization.

In this algorithm, σ' represents the derivative of the sigmoid activation function, denoted as $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$. The term hidden_layer_output^T signifies the transpose of the hidden layer output matrix, while X_{train}^T refers to the transpose of the training input data matrix.

These variables and functions play crucial roles in the computation of layer outputs, error calculations, and weight adjustments during the training process of the MLP. Through iterations and periodic evaluations, the model aims to minimize loss and improve accuracy in predicting output values.

4 Results Analysis

4.1 Training Setup

The construction and training of the MLP involved setting the hidden layer dimension to 4 and configuring the output dimension as 1. Training spanned 550 epochs with a learning rate of 0.0001 for weight updates. These hyperparameters play a critical role in shaping the model's architecture and influencing its learning process. The MLP was trained on (*X_train*, *y_train*) with validation against (*X_val*, *y_val*). The resulting loss and accuracy histories were observed, emphasizing the significance of hyperparameter tuning in achieving optimal model performance and generalization on unseen data.

4.2 Model Training Performance

- **Loss Curves**

  Both the training and validation loss consistently decreased with each epoch shown in Figure 2. Notably, the training loss consistently remained lower than the validation loss.

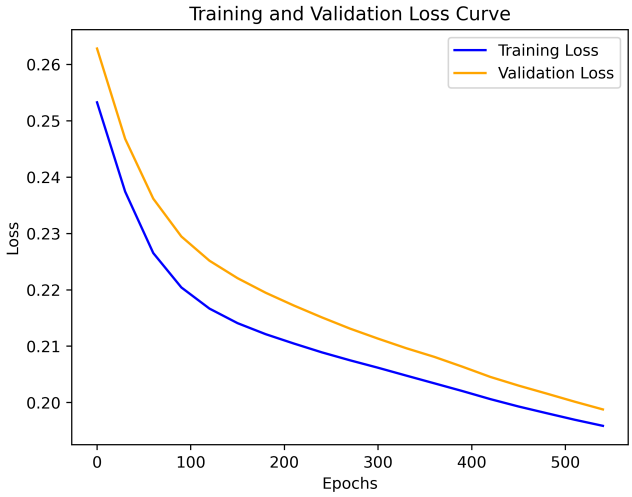

Figure 2: Loss Curves

This trend suggests that while the model generalized well, it might have slightly overfit the training data.

- **Accuracy Curves**

  The accuracy curves for both training and validation data displayed an upward trend across epochs shown in Figure 3. Initially, the validation accuracy lagged slightly behind the training accuracy, indicating the model's superior performance on the training data. However, towards the end of training epochs, the validation accuracy caught up, showcasing the model's improvement in predicting unknown data.

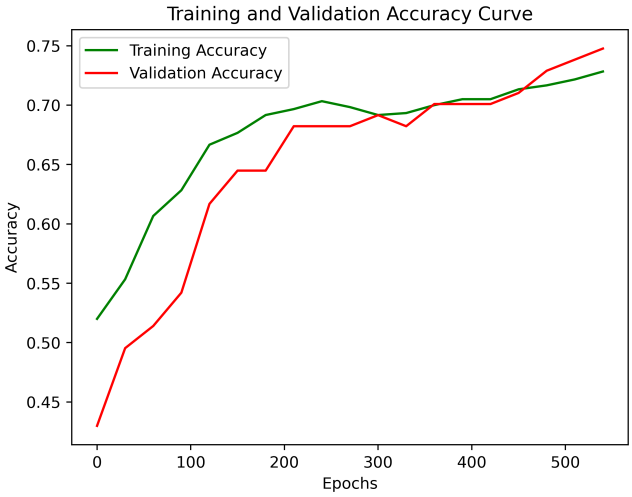

Figure 3: Accuracy Curves

4.3 Test Set Performance

The model exhibited a test set accuracy of 76.00%, indicating a reasonable level of performance on unseen data. While the accuracy is decent, there might be room for further improvement depending on specific application requirements.

4.4 Confusion Matrix

In the Figure 4 confusion matrix provides insights into the model's predictions compared to actual values in the test set.

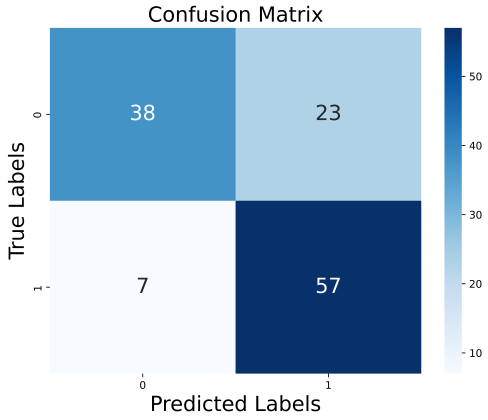

Figure 4: Confusion Matrix

- True Positives (TP): The model correctly identified 57 cases as the absence of liver disease.
- True Negatives (TN): It accurately recognized 38 cases as the presence of liver disease.
- False Positives (FP): The model incorrectly labeled 23 cases as the absence of liver disease when it was present.
- False Negatives (FN): It erroneously identified 7 cases as the presence of liver disease when it was absent.

Here, 0 represents the presence of liver disease, and 1 indicates the absence. The matrix reveals that the model had more false predictions for the absence of liver disease (false positives) than false predictions for the presence of liver disease (false negatives). This suggests a slight bias towards predicting the absence of liver disease.

Overall, while the model demonstrated promising performance, further fine-tuning could enhance accuracy and balance predictions for both classes in the dataset.

5 Applying Multi-layer Neural Networks algorithm to Solve the XOR Problem

The XOR problem poses a challenge in binary classification due to its non-linear nature. It involves two input variables and requires the output to be 1 if the inputs differ and 0 if they're the same. Traditional linear classifiers fail to accurately separate these classes due to their non-linear relationship.

5.1 Dataset for XOR Problem

The dataset for the XOR problem comprises the following input-output pairs, represented in Table [2]:

Table 2: XOR Dataset

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

5.2 Experiment and Results

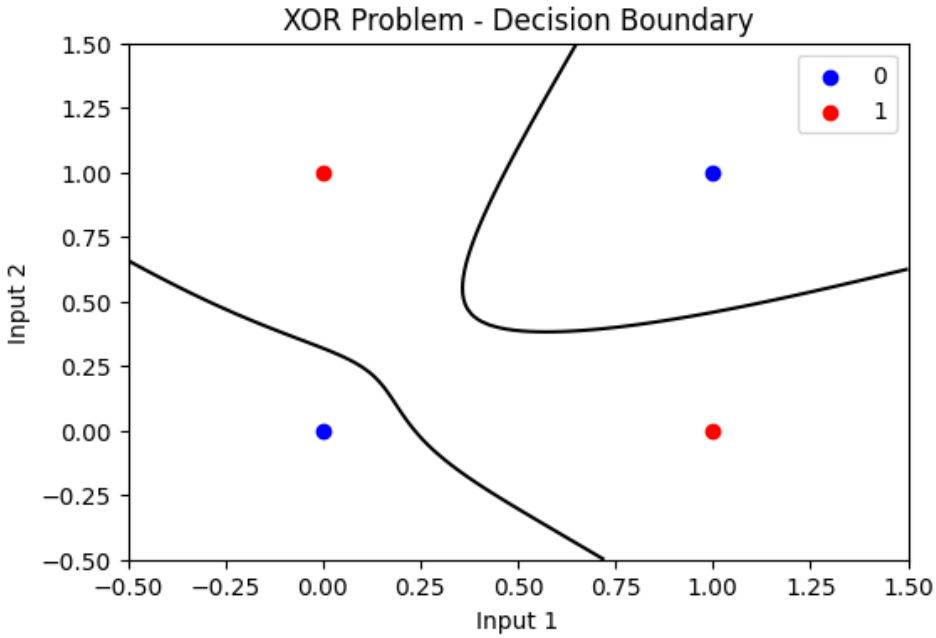

Figure 5: Decision boundary of the Multi-layer perceptron for the XOR problem.

A Multi-layer Neural Network solves the XOR problem by employing hidden layers and non-linear activation functions like the sigmoid. Through these mechanisms, the MLP learns intricate patterns in the data, creating non-linear decision boundaries. In the XOR example, the plotted decision boundary in Figure 5 showcases how the MLP effectively separates the 0s and 1s, demonstrating its capability to grasp complex, non-linear relationships between input features. This ability to capture non-linear structures underscores the power of neural networks in addressing intricate classification tasks.

6 Conclusion

The MLP model showcased promising predictive abilities for diagnosing liver disease. It exhibited continual improvement in accuracy throughout training, achieving a commendable 76.00% accuracy on unseen data. However, slight indications of overfitting were observed, with the model showing a tendency to predict the absence of liver disease more often than its presence. A closer look at the confusion matrix reveals a tendency to predict absence rather than presence of liver disease. Fine-tuning to balance predictions for both classes could further optimize its reliability. Addressing this bias and refining the model's generalization could further enhance its reliability for real-world applications.

References

[1] UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/ [Accessed: November 2, 2023].