University of Tripoli Faculty of Engineering

EEE 569: Introduction to deep learning

Assignment 1 Part (B)

Reem Ben Guma

Introduction:

This part of the homework focuses on implementing a multi-layer perceptron to solve classification problems, using it with different types of inputs, and automating the creation of neural network architecture.

Task 1- XOR Problem:

XOR Problem Generate a dataset consisting of two classes, each with 50% of the total samples. Each class should be drawn from two Gaussian distributions, resulting in a total of four Gaussian distributions. Position the means of the distributions such that the classes are not linearly separable, creating an XOR-like problem. Train a logistic regression model on the generated dataset. Evaluate the performance of the logistic regression model and comment on its ability to solve the XOR problem.

Data given the following normal distribution parameters:

- Class 1: $\sum = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, half of the samples has mean of $\binom{3}{10}$ $other\ half\ has\ mean\ of\ \binom{3}{10}$
- Class 2: $\sum = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, half of the samples has mean of $\binom{3}{3}$ $other\ half\ has\ mean\ of\ \binom{10}{10}$

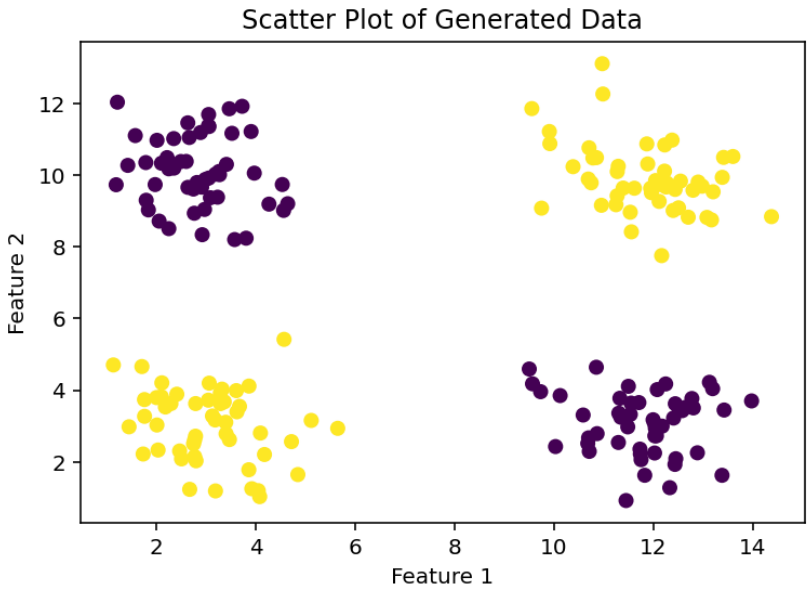


Figure 1: scatter plot of data set.

Running the logistic regression classifier, several times produces accuracies of {62%, 50%, 44%, 38%, 58%, 54%, 30%}, the average accuracy of 48%, which is very bad.

Decision boundary:

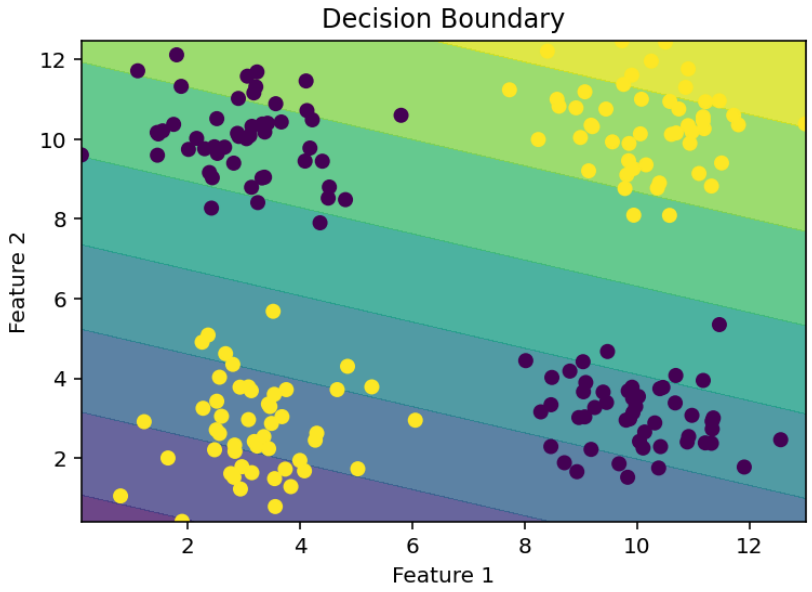


Figure 2: Decision boundary using logistic regression.

Since the logistic regression is a linear classifier, it cannot work well in this problem because there is no straight line that can split feature space into two regions where each class (or most of its samples) has its own region.

Task 2- Multi-Layer Perceptron:

In this task, you will design and implement a multi-layer perceptron (MLP) using the Linear class you developed in Assignment 1, Part A. The MLP will be applied to the dataset generated in Task 1, and its performance will be evaluated.

1. Construct a multi-layer perceptron with the following architecture:
    - Depth: 2 (i.e., two hidden layers)
    - Width: 20 (i.e., 20 neurons in each hidden layer)
    - Activation function: Sigmoid
2. Utilize the Linear class you implemented in Assignment 1, Part A, as the building block for the MLP.
3. Train the MLP on the dataset generated in Task 1.
4. Evaluate the performance of the MLP on the same dataset. And comment on its ability to solve the XOR problem.

Running the multilayer perceptron with the given network design several times produces accuracies of {100%, 100%, 96%, 98%,98}, the average accuracy of 98.4%.

Decision boundary:

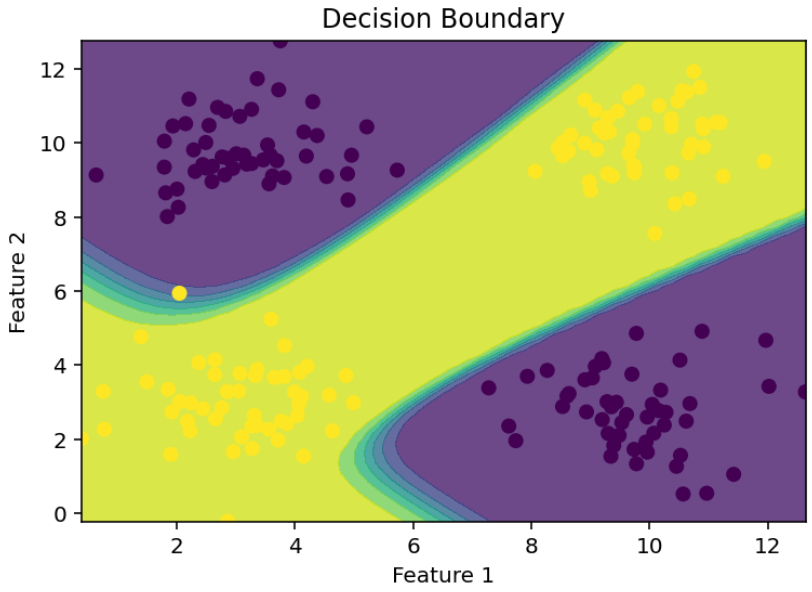


Figure 3: Decision boundary using MLP.

As expected, MLP performs much better than logistic regression. It can learn the nonlinearity of the data, resulting in a more flexible and accurate decision boundary that better separates the classes.

Task 3- Code Refactoring and Automation:

In this task, you will improve the quality and maintainability of the code by automating repetitive tasks and reducing manual effort. Specifically, you will focus on automating the creation of the neural network architecture and related data structures.

1. Automate the creation of parameter nodes, such as weights and biases, within the Linear class.
2. Automate the creation of the graph and trainable lists.

The creation of a neural network is enhanced, so the use of MLP needs few steps:

- Create model: model = MLP(input_channels, hidden_layers, width, number_of_classes)
- Train model: model.train(X, y, batch_size, epochs, learning_rate)
- Make predictions: predictions = model.predict(X)

Where the MLP class works with both classification problems, binary classification with a Sigmoid activation unit in the output layer and binary cross-entropy loss as a cost function, and multiclass classification using the SoftMax activation function in the output layer and cross-entropy loss as a cost function.

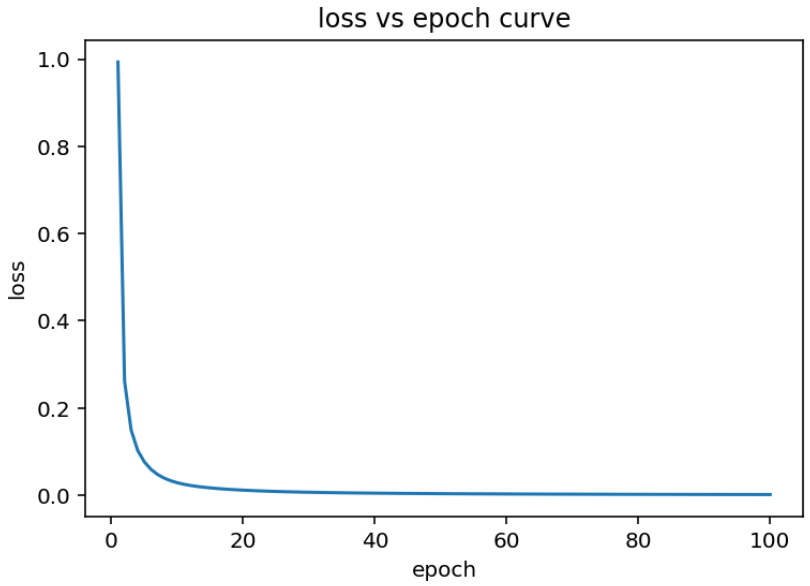
Task 4- Handwritten Digit Classification using MNIST Dataset:

In this task, we aim to develop a neural network model to classify handwritten digits using a simplified version of the MNIST dataset. We will utilize the scikit-learn library to load the dataset.

The input data is a 64-dimensional vector, which is a rearrangement of the 8x8 images into a single vector. Task Requirements:

1. Split the input data into training and testing sets using a 60%-40% split.
2. Implement a multi-class classification model using a neural network with one hidden layer of size 64.
3. Implement the SoftMax activation function at the output layer and the Cross-Entropy loss function.
4. Create a one-hot vector target for the multi-class classification problem.
5. Train the network using the training data and evaluate its performance on the testing data. You should get an accuracy greater than 80% (the higher the better).

The figure below shows loss of training process.



With an average accuracy of 93.2%.