Nonlinear Classification of Double Moon Dataset

Vatsal J Patel, Latika Dekate, Lakshminarayaa Rajamannar
Email: vpate60@asu.edu, ldekate@asu.edu, lrajamannar@asu.edu

***Abstract*—This report describes how the multilayer perceptron (MLP) neural network is constructed using PyTorch to solve the double moon dataset's nonlinear binary classification task. The architecture of neural network consists of a hidden layer of 16 neurons that we have done using the ReLU activation function, and then one output neuron using the sigmoid activation. The Adam optimizer of learning rate 0.001 and weight decay 1e-5 was utilized to train the model for 2000 epochs. The solution helped us find nonlinear decision boundary with a validation accuracy of 98.75%. Learning curves and decision boundary visualization that we have achieved proves successful learning and outstanding generalization with no overfitting.**

***Index Terms*—Multilayer Perceptron (MLP), Nonlinear Classification, Double Moon Dataset, PyTorch, Adam Optimizer, Supervised Learning, Neural Networks.**

I. INTRODUCTION

The issue of nonlinearly separable data is a common machine learning problem. The "double moon" data set, with two overlapping crescent moons, that we are provided with, is a classic test data set for comparing algorithm capacity and to learn complex decision boundaries. Linear classifiers cannot do this, which calls for nonlinear models such as Multilayer Perceptrons (MLPs). This report showcases the design and testing of an MLP designed to classify this dataset according to the requirements for the assignment. The requirements were that a feedforward neural network with a single hidden layer should be used, and then train the network with the an optimizer, validation accuracy should be greater than 95%, that should show both the learning process and the final decision boundary.

The full implementation code is accessible at the following link: https://github.com/vpate160/2-Moons-classifier-Team7/blob/main/nonlinear_classification.ipynb

II. PROBLEM CONTEXT AND DATASET

The task involves binary classification of 2D data points belonging to two distinct, nonlinearly separable classes arranged in a double moon pattern. The issue is trying to come up with a way to create a model that can determine the curved boundary between the classes.

The training set that was used here (2halfmoonsTrain.csv) contains 800 samples. There are about two feature values (X and Y coordinates) and the class label per sample. The labels were remapped to 0, 1 to match the binary cross-entropy loss. The data was then split into a training set (640 samples, 80%) and a validation set (160 samples, 20%) using sampling to maintain the initial class balance.

III. METHODOLOGY

A. Network Architecture

An MLP was implemented using PyTorch's nn.Module. The architecture, designed for simplicity while providing sufficient nonlinear capacity, is detailed in Table I.

TABLE I
MLP ARCHITECTURE

Layer	Type	Input Size	Output Size	Activation	Parameters
Hidden (fc1)	`nn.Linear`	2	16	ReLU	48
Output (fc2)	`nn.Linear`	16	1	Sigmoid	17
Total					**65**

The forward pass is defined as:

$$h = \text{ReLU}(XW_1 + b_1) \tag{1}$$

$$\hat{y} = \sigma(hW_2 + b_2) \tag{2}$$

where $X \in \mathbb{R}^{N\times 2}$ is the input data batch, $W_1 \in \mathbb{R}^{2\times 16}, b_1 \in \mathbb{R}^{16}$ are the weights and bias of the hidden layer, $W_2 \in \mathbb{R}^{16\times 1}, b_2 \in \mathbb{R}$ are the weights and bias of the output layer, ReLU is the Rectified Linear Unit activation function ($f(x) = \max(0, x)$), and σ is the sigmoid function ($\sigma(z) = 1/(1 + e^{-z})$).

B. Training Configuration

The network was trained using the hyperparameters specified in Table II.

TABLE II
HYPERPARAMETERS

Parameter	Value
Loss Function	BCELoss
Optimizer	Adam
Learning Rate	0.001
Weight Decay (L2)	1e-5
Number of Epochs	2000
Batch Size	640 (Full)
Random Seed	42

Binary Cross-Entropy (BCE) loss was used:

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3}$$

The Adam optimizer [**?**] was chosen for its efficiency. A small L2 weight decay was added for regularization. Training was performed for 2000 epochs. Reproducibility was ensured by setting random seeds.

IV. RESULTS

A. Setup

The model was trained using PyTorch version 2.8.0+. The data was then split, converted to tensors, and fed into the training loop accordingly. Training and validation loss (BCE) and accuracy were recorded successfully.

B. Results Analysis

After 2000 epochs, the model achieved the performance summarized in Table III.

TABLE III
FINAL PERFORMANCE METRICS

Metric	Training	Validation
Accuracy (%)	95.16	98.75
Correct Samples	609/640	158/160
BCE Loss	0.1013	0.0852

The learning curves (Fig. 1) show smooth convergence. We also noticed, the validation accuracy slightly exceeds the training accuracy and then the final validation loss is lower than the training loss, indicating a good generalization without overfitting.

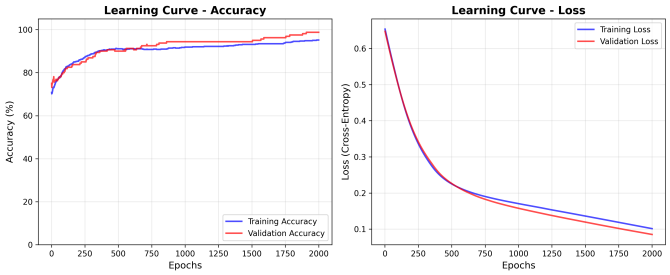

Fig. 1. Learning curves showing training (blue) and validation (red) accuracy (top) and BCE loss (bottom) over 2000 epochs. The model demonstrates stable convergence and good generalization.

The learned nonlinear decision boundary is visualized in Fig. 2.

The boundary effectively captures the nonlinear structure, separating most of the points properly.

V. DISCUSSIONS AND CONCLUSIONS

The MLP trained model provided with the nonlinear boundary of the double moon dataset with a validation accuracy of 98.75%. The single hidden layer with 16 neurons was enough to get the result we were looking for. The Adam optimizer provided stable and effective convergence, following the requirements.

The result confirms the application of simple MLPs to nonlinear tasks and the effectiveness of the Adam optimizer successfully. All the targets of the assignment were met as mentioned.

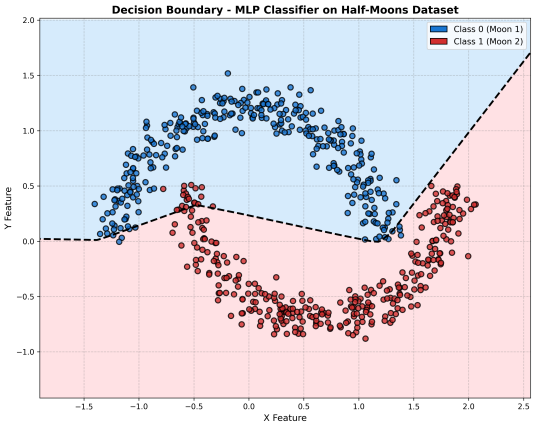

Fig. 2. Decision boundary learned by the MLP classifier. Background color indicates predicted class (Blue: Class 0, Red: Class 1). Dashed line is the $P = 0.5$ boundary. Training data points are overlaid.

APPENDIX A

Following link provided is the GitHub repository of this project assignment. This link includes implementation code, optimization comparison, and images achieved:

https://github.com/vpate160/2-Moons-classifier-Team7