Implementing a Single-Layer Perceptron from Scratch

Abdullah Chaudhary[1 [21i−0844]]

National University of Computers and Emerging Sciences, FAST
i210844@nu.edu.pk

Abstract. This paper presents a manual implementation of a single-layer perceptron for binary classification. It covers dataset generation, forward propagation, weight updates, decision boundary visualization, and performance evaluation. The implementation avoids deep learning frameworks, offering insights into perceptron mechanics. Experiments highlight the model's strengths and limitations, especially for linearly separable data.

Keywords: Perceptron, Machine Learning, Neural Networks, Binary Classification

1 Introduction

1.1 Objective

The primary aim is to implement a perceptron model for binary classification manually, adjusting weights and biases based on training data. Key steps include dataset generation, visualization, model training, and performance evaluation.

1.2 Instructions

The implementation follows these guidelines:

- Use NumPy for numerical computations and Matplotlib for visualization.
- Implement perceptron functionality manually, avoiding machine learning frameworks.
- Generate and visualize a synthetic dataset with 500 samples and binary labels.
- Evaluate the perceptron's classification performance on test data.

2 Dataset Generation and Preprocessing

A synthetic dataset of 500 samples was generated with two features. Binary labels were assigned based on a linear relationship. The dataset was split into training (80%) and testing (20%) subsets.

$$X = [X_1, X_2], \quad y \in \{0, 1\} \tag{1}$$

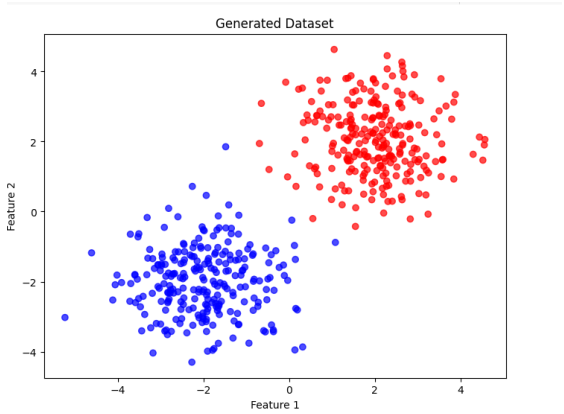

Fig. 1. Visualization of the synthetic dataset. Training and testing data are shown in distinct colors.

3 Perceptron Implementation

3.1 Forward Propagation

The perceptron calculates the weighted sum of inputs and applies a step activation function:

$$y_{pred} = f(WX + b), \quad \text{where } f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

3.2 Training Process

The perceptron underwent training for 20 epochs. Errors per epoch were recorded, and training concluded when misclassifications ceased.

4 Visualization and Evaluation

4.1 Error over Epochs

Training progress was visualized by plotting total errors per epoch.

4.2 Decision Boundary Visualization

The decision boundary was plotted, illustrating the model's ability to separate classes.

4.3 Accuracy on Test Data

The perceptron achieved an accuracy of **95%** on the test dataset, demonstrating its effectiveness for linearly separable data.

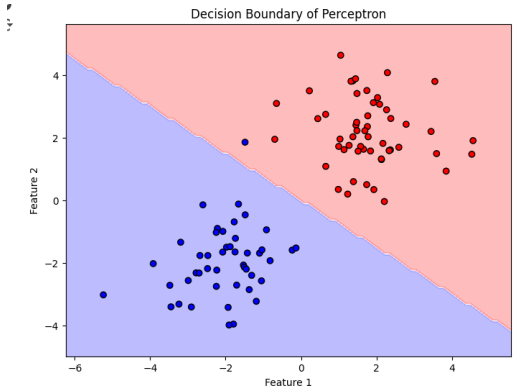

Fig. 2. Error reduction over epochs. The model converges after a few iterations.

5 Results and Discussion

5.1 Performance on Linearly Separable Data

- The perceptron successfully classified linearly separable data.
- Training convergence was rapid, with errors diminishing over a few epochs.

5.2 Limitations on Non-Linearly Separable Data

- The perceptron fails on non-linearly separable datasets due to its linear decision boundary.
- Addressing these limitations requires advanced models, such as multi-layer perceptrons.

5.3 Comparison with Modern Classifiers

- Compared to logistic regression and SVMs, the perceptron lacks robustness and flexibility.
- Future work may explore neural networks with hidden layers for improved performance.

6 Conclusion

This report details the implementation of a single-layer perceptron, highlighting its strengths in simple binary classification tasks and limitations for complex patterns. Future improvements could involve introducing non-linearity and deeper architectures.

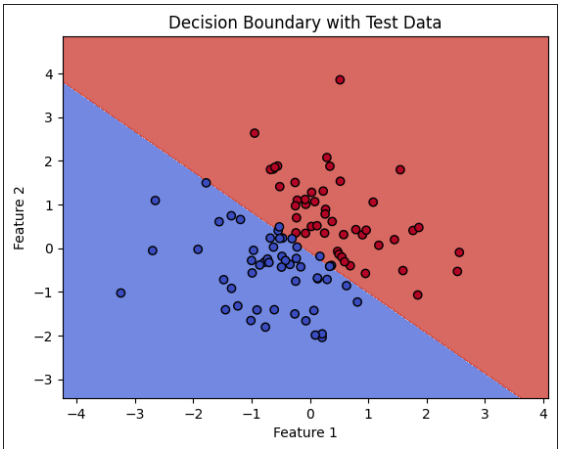

Fig. 3. Decision boundary visualization with test data. Misclassified points are minimal.

References

1. Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review **65**(6), 386–408 (1958)
2. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)