Implementation of the Perceptron Learning Algorithm for Classification of the Iris Dataset

Mridunja Raman[1]

[1] Computer Science and Engineering, Indian Institute of Technology, Delhi.

*email address: mridunjaraman@gmail.com

Keywords: Perceptron, Machine Learning, Binary Classification, Iris Dataset, Supervised Learning, Python, NumPy, Decision Boundary.

Abstract. This report gives the details of the Perceptron Learning Algorithm from scratch for binary classification on the Iris dataset. The model built is capable of distinguishing between Iris-setosa and Iris-versicolor specimens based on their sepal and petal length. The implementation was carried out in Python, utilizing only NumPy for numerical operations and Matplotlib for visualization, deliberately avoiding high-level machine learning libraries. The trained model successfully demonstrated a clear linear decision boundary, validating a foundational understanding of the algorithm's mechanics.

1 Introduction

The Perceptron is a foundational algorithm in the history of machine learning and represents one of the earliest models of a supervised learning algorithm for binary classification. It serves as the basis for more complex neural networks.

This project implements the Perceptron algorithm from scratch to provide a foundational understanding of its core mechanics, such as the weight update rule, which are often abstracted by high-level libraries. The classic Iris dataset was selected as an ideal, simple testbed for this fundamental classification algorithm.

2 Data and Methods

2.1 The Perceptron Algorithm

The Perceptron model makes predictions based on a linear combination of input features. The core components of the algorithm are:

1. Net Input Function: The net input z is calculated as the dot product of the input feature vector X and the weight vector w, plus a bias term b:

$$z = \mathbf{w}^T\mathbf{X} + b$$

2. Weight Update Rule: The model learns by adjusting its weights after each sample. If a prediction is incorrect, the weights are updated according to the Perceptron learning rule:

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

where is the learning rate, $y^{(i)}$ is the true class label, and $\hat{y}^{(i)}$ is the predicted class label. The bias is updated similarly. This process is repeated for a set number of epochs.

Article version: 7th October 2025 h08:18 CET

2.2 Dataset

The project uses the Iris dataset, focusing on two linearly separable classes: Iris-setosa and Iris-versicolor. Two features were selected for classification and visualization:

- Sepal Length (cm)
- Petal Length (cm)

2.3 Implementation Details

The algorithm was encapsulated within a Perceptron class in Python. This class contains methods for fit (to train the model on data), predict (to make predictions), and net_input. The training was performed with a learning rate of 0.1 for 10 epochs.

3 Results

3.1 *Model Convergence*

The plot below tracks the number of misclassifications (errors) during each epoch of the training process. The model demonstrates rapid learning, with the number of errors dropping to zero by the 6th epoch, indicating that the algorithm has successfully converged and found a linear decision boundary that separates the two classes.

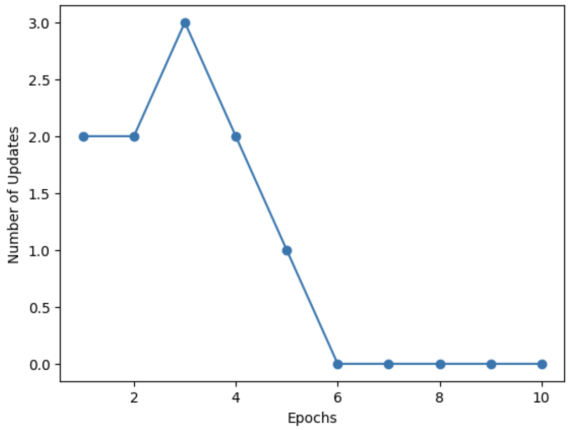

Figure 1: Model Convergence

3.2 *Decision Boundary*

The plot below visualizes the final decision boundary learned by the model. The straight line effectively separates the Iris-setosa and Iris-versicolor data points, confirming that the Perceptron successfully learned the linear separability of the classes.

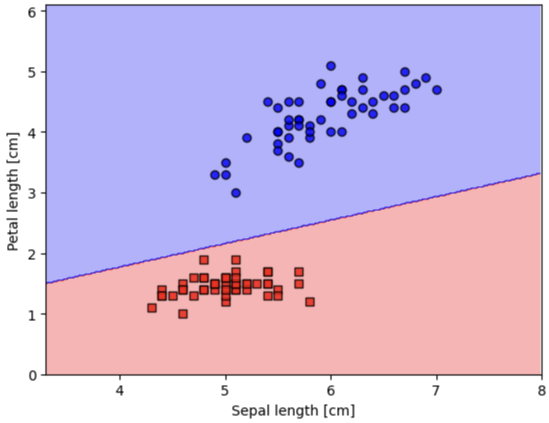

Figure 2: Decision Boundary

4 Conclusion

The from-scratch implementation of the Perceptron algorithm was successfully validated on the Iris dataset. The model learned to effectively classify two linearly separable classes, as evidenced by its rapid convergence and the clear decision boundary in the results. This project confirms a strong understanding of foundational machine learning principles that are central to modern neural networks.