Mathematical Analysis of a Neural Network for Logistic Regression

Peterson Carara Junior

June, 2025

Introduction

This work aims to mathematically explain the foundations of training a neural network applied to binary logistic regression, whose objective is to classify asteroids as **hazardous** or **non-hazardous** based on astronomical data from the *Near Earth Objects (NEO)* dataset.

Problem Description and Dataset

The problem consists of predicting the variable `hazardous` (binary: 0 for non-hazardous, 1 for hazardous) based on variables such as:

- `est_diameter_min`, `est_diameter_max`: Estimated minimum and maximum diameter of the asteroid;
- `relative_velocity`: Relative velocity to the planet;
- `miss_distance`: Approach distance in km;
- `absolute_magnitude`: Absolute magnitude of the celestial body.

After preprocessing, the dataset contains 5 normalized input variables for the neural network.

Neural Network Architecture

The implemented neural network consists of:

- Fully connected hidden layers with **ReLU** activation function: $f(x) = \max(0, x)$;
- An output layer with **sigmoid** activation:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

  which maps the output to the interval $[0, 1]$, interpreted as probability.

Cost Function: Binary Cross-Entropy

The loss function used was binary cross-entropy, given by:

$$\mathcal{L}(y, \hat{y}) = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right]$$

Where:

- $y \in \{0, 1\}$: true label;
- $\hat{y} \in (0, 1)$: neural network output (probability).

Optimization: Gradient Descent

During training, the network weights are adjusted to minimize the loss function using the gradient descent algorithm.

Mathematical Definition

Given a weight vector $\vec{w}$ and a cost function $\mathcal{L}(\vec{w})$, the weight update rule is given by:

$$\vec{w}_{t+1} = \vec{w}_t - \eta \cdot \nabla\mathcal{L}(\vec{w}_t)$$

Where:

- η: learning rate (hyperparameter automatically tuned);
- $\nabla\mathcal{L}$: gradient of the cost function.

The gradient indicates the direction of greatest increase of the loss. Subtracting the gradient means moving in the direction of greatest decrease, seeking the **minimum of the cost function**.

Hyperparameter Tuning with Random Search

We used the `KerasTuner` library with Random Search to find the best values for:

- Number of hidden layers (between 1 and 3);
- Number of neurons per layer (between 8 and 64);
- Learning rate $\eta \in [10^{-4}, 10^{-2}]$.

Results

Classification Report

```
accuracy:       0.91
precision:      0.92 (Non-Hazardous), 0.73 (Hazardous)
recall:         0.99 (Non-Hazardous), 0.14 (Hazardous)
```

Confusion Matrix

$$\begin{bmatrix} 16346 & 93 \\ 1480 & 249 \end{bmatrix}$$

AUC

Area Under the ROC Curve (AUC) = 0.91

Graphs

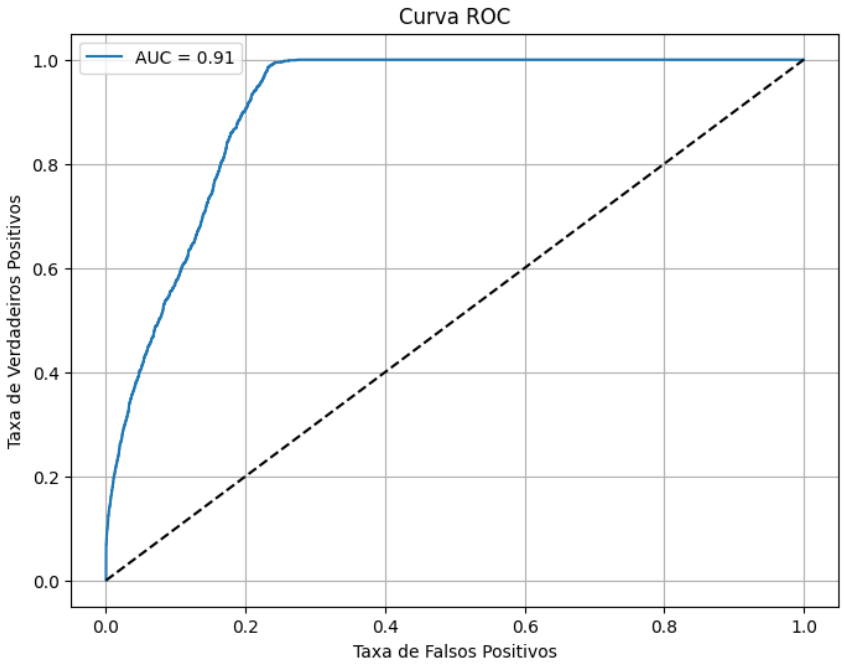

Figure 1: ROC Curve

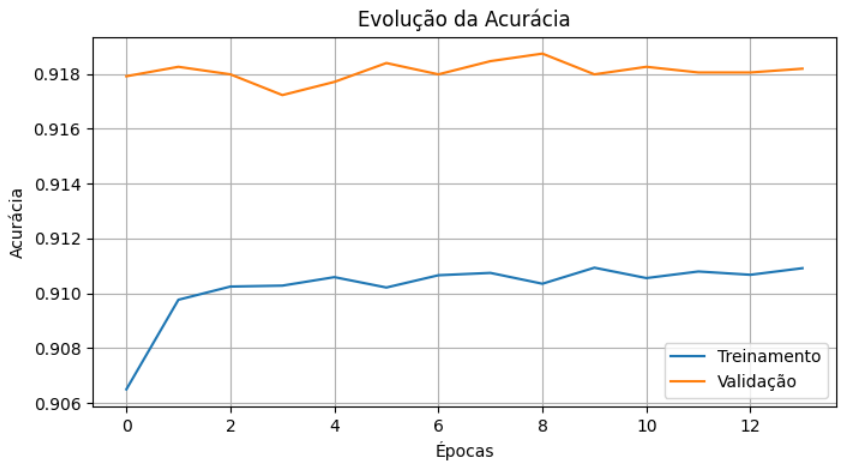

Figure 2: Accuracy Evolution During Training

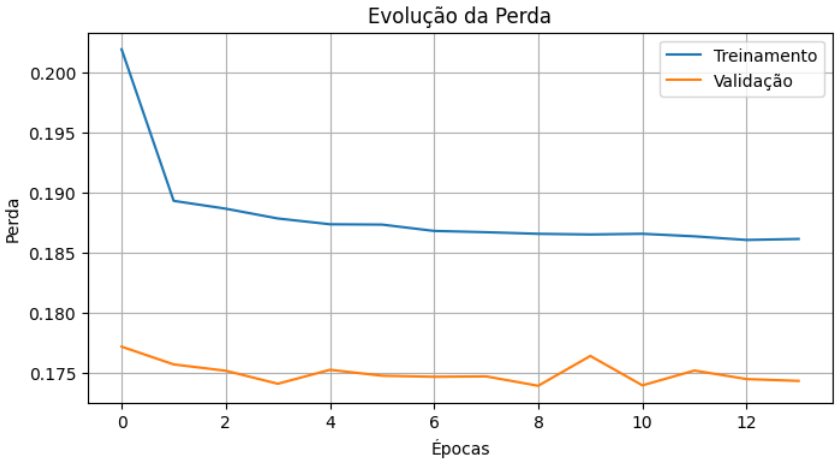

Figure 3: Loss Evolution During Training

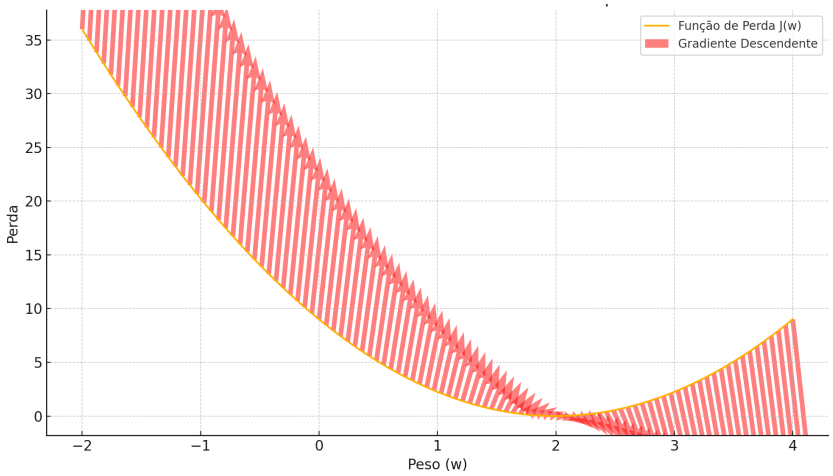

Figure 4: Gradient vs Dataset Graph

Final Considerations

The logistic regression implemented through a neural network proved to be efficient in classifying asteroids, despite the dataset imbalance. The use of gradient descent allowed finding a minimum for the cost function, and hyperparameter tuning enhanced the network's ability to learn patterns from the data. From a mathematical perspective, training the network involves vector calculus and partial derivatives to find the minimum of a nonlinear function.