Análise Matemática de Rede Neural para Regressão Logística

Peterson Carara Junior

Junho, 2025

Introdução

Este trabalho visa explicar matematicamente os fundamentos do treinamento de uma rede neural aplicada à regressão logística binária, cujo objetivo é classificar asteroides como **perigosos** ou **não perigosos** com base em dados astronômicos do dataset *Near Earth Objects (NEO)*.

Descrição do Problema e do Dataset

O problema consiste em prever a variável `hazardous` (binária: 0 para não perigoso, 1 para perigoso) com base em variáveis como:

- `est_diameter_min`, `est_diameter_max`: Diâmetro estimado mínimo e máximo do asteroide;
- `relative_velocity`: Velocidade relativa ao planeta;
- `miss_distance`: Distância de aproximação em km;
- `absolute_magnitude`: Magnitude absoluta do corpo celeste.

Após limpeza, o dataset apresenta 5 variáveis de entrada normalizadas para entrada da rede neural.

Arquitetura da Rede Neural

A rede neural implementada é composta por:

- Camadas ocultas densamente conectadas com função de ativação **ReLU**: $f(x) = \max(0, x)$;
- Uma camada de saída com ativação **sigmoide**:

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

que mapeia a saída para o intervalo $[0, 1]$, interpretada como probabilidade.

Função de Custo: Entropia Cruzada (Binary Cross-Entropy)

A função de perda usada foi a entropia cruzada binária, dada por:

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

Onde:

- $y \in \{0, 1\}$: rótulo verdadeiro;
- $\hat{y} \in (0, 1)$: saída da rede neural (probabilidade).

Otimização: Gradiente Descendente

Durante o treinamento, os pesos da rede são ajustados para minimizar a função de perda usando o algoritmo do gradiente descendente.

Definição Matemática

Dado um vetor de pesos $\vec{w}$ e uma função de custo $\mathcal{L}(\vec{w})$, a atualização dos pesos é dada por:

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

Onde:

- η: taxa de aprendizado (hiperparâmetro ajustado automaticamente);
- $\nabla\mathcal{L}$: gradiente da função de custo.

O gradiente indica a direção de maior crescimento da perda. Subtrair o gradiente implica mover na direção de maior declínio, buscando o **mínimo da função de custo**.

Ajuste de Hiperparâmetros com Random Search

Utilizamos a biblioteca `KerasTuner` com Random Search para encontrar os melhores valores de:

- Número de camadas ocultas (entre 1 e 3);
- Número de neurônios por camada (entre 8 e 64);
- Taxa de aprendizado $\eta \in [10^{-4}, 10^{-2}]$.

Resultados

Relatório de Classificação

```
accuracy:       0.91
precision:      0.92 (Não Perigoso), 0.73 (Perigoso)
recall:         0.99 (Não Perigoso), 0.14 (Perigoso)
```

Matriz de Confusão

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

AUC

Área sob a curva ROC (AUC) = 0.91

Gráficos

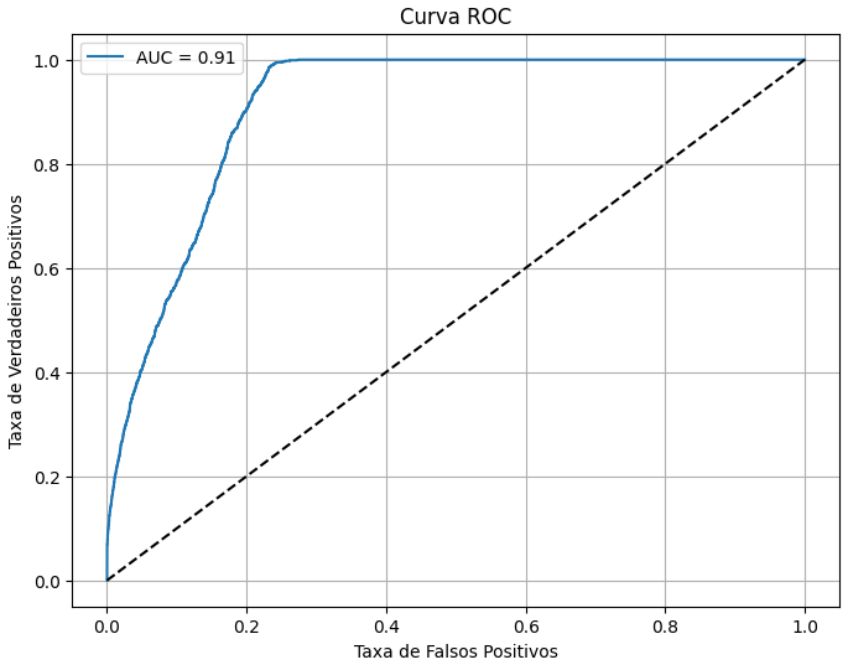

Figure 1: Curva ROC

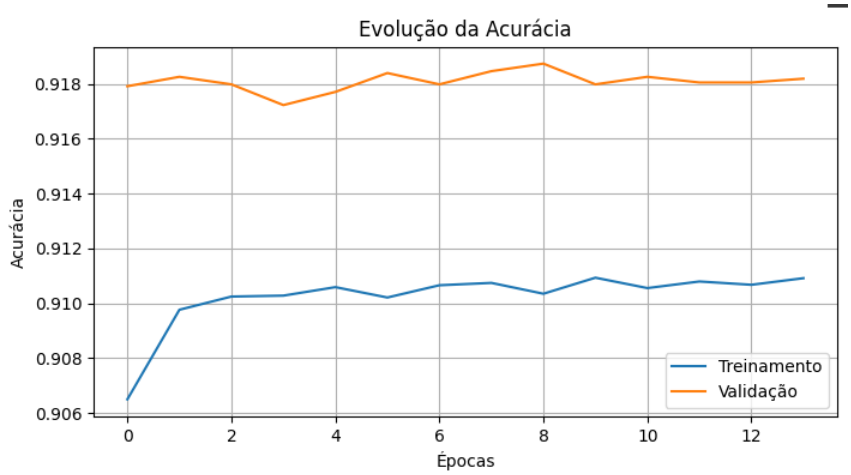

Figure 2: Evolução da Acurácia durante o Treinamento

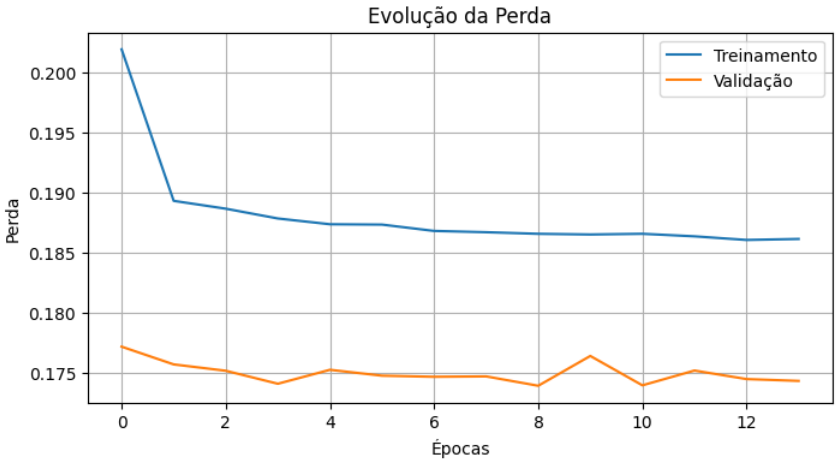

Figure 3: Evolução da Perda (Loss) durante o Treinamento

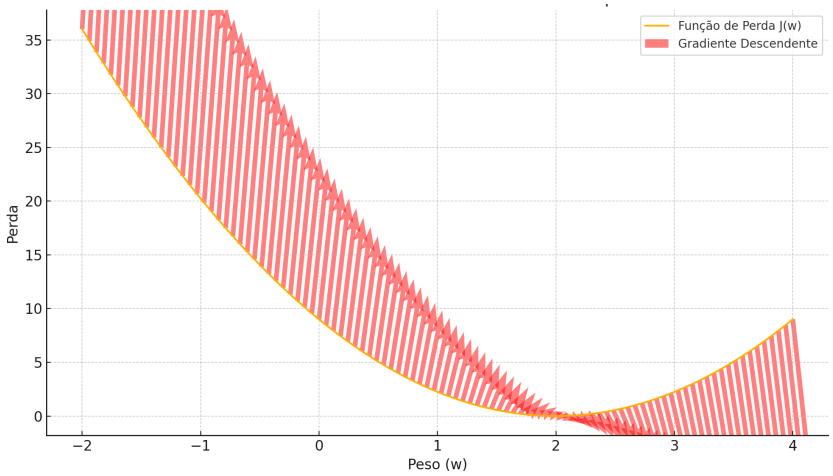

Figure 4: Grafico Gradeiente Vs Dataset

Considerações Finais

A regressão logística implementada por rede neural mostrou-se eficiente em classificar asteroides, apesar do desbalanceamento do dataset. O uso do gradiente descendente permitiu encontrar um mínimo para a função de custo, e o ajuste de hiperparâmetros potencializou a capacidade da rede de aprender padrões nos dados. Do ponto de vista matemático, o treinamento da rede envolve cálculo vetorial e derivadas parciais para encontrar o mínimo de uma função não linear.