

Classificação de dados utilizando redes neurais - Uma aplicação de machine learning supervisionado em C++

Bacharelando: Rodrigo Berino Pereira

Docente: Prof. Msc. Éder Augusto Penharbel

INSTITUTO FEDERAL CATARINENSE -
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO - 5^a Fase

20 de fevereiro de 2026



Objetivos

- Introduzir os conceitos de AI, Machine Learning e Redes Neurais Artificiais.
- Utilizar um exemplo prático de aprendizado supervisionado para resolver problemas de classificação de dados.

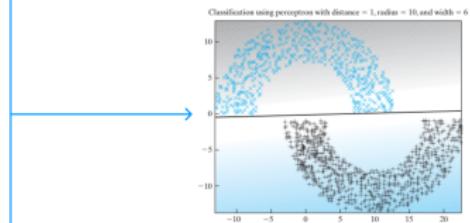


Figura: Fontes principais utilizadas neste projeto.

Agenda

- 1 Introdução
- 2 Conceitos Fundamentais
- 3 Apresentação do Problema
- 4 Solução Implementada
- 5 Conclusão

Introdução

Diferenciação de conceitos utilizados diariamente.

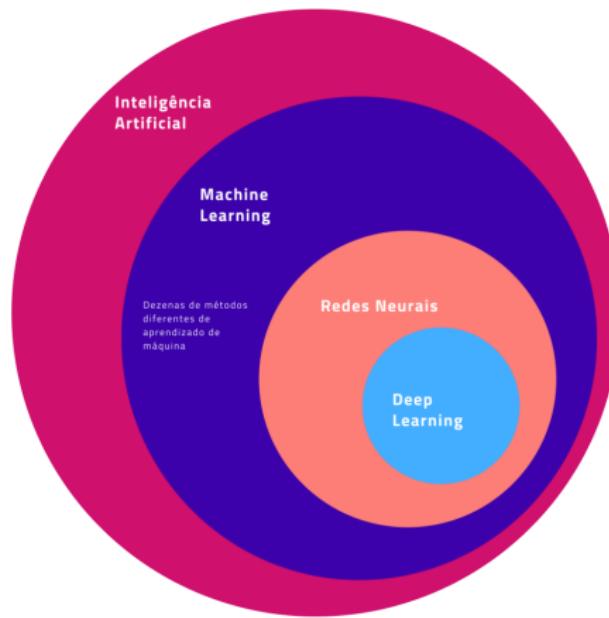


Figura: Diagrama - razor.com.br

Inteligência Artificial

Área da ciência da computação que busca desenvolver sistemas capazes de realizar tarefas que normalmente exigiriam inteligência humana.

- Processamento de Linguagem Natural
- Visão Computacional
- Sistemas Especialistas



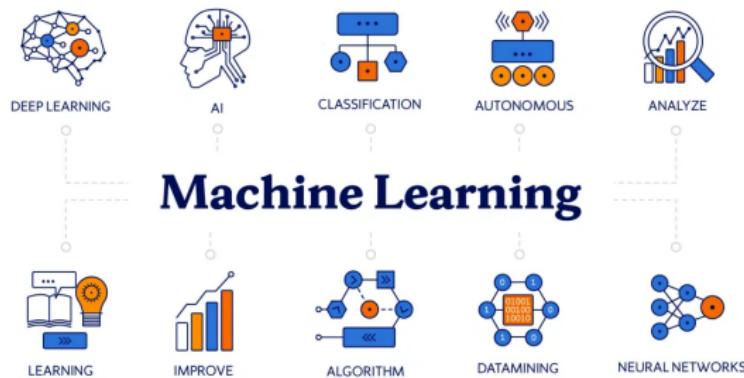
Inteligência Artificial

Sim, o ChatGPT é um exemplo de inteligência artificial (IA). Mais especificamente, é um modelo de linguagem grande (LLM) desenvolvido pela OpenAI que utiliza técnicas de aprendizado profundo para gerar texto semelhante ao humano.



Aprendizado de Máquina (Machine Learning)

- Técnica para ensinar computadores a reconhecer padrões em dados.
- Classificação, regressão, agrupamento.
- Tipos principais: supervisionado, não supervisionado e por reforço.



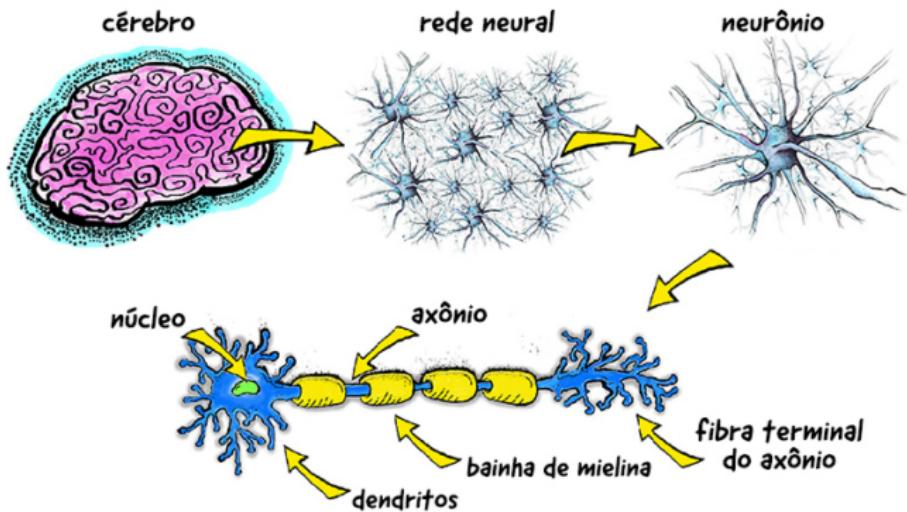
Redes Neurais Artificiais



Figura: Os seres humanos buscam exemplos na natureza.

Redes Neurais Artificiais

Modelos computacionais inspirados no funcionamento do cérebro humano, compostos por unidades de processamento interconectadas (neurônios artificiais) que trabalham em conjunto para resolver problemas.



Redes Neurais Artificiais

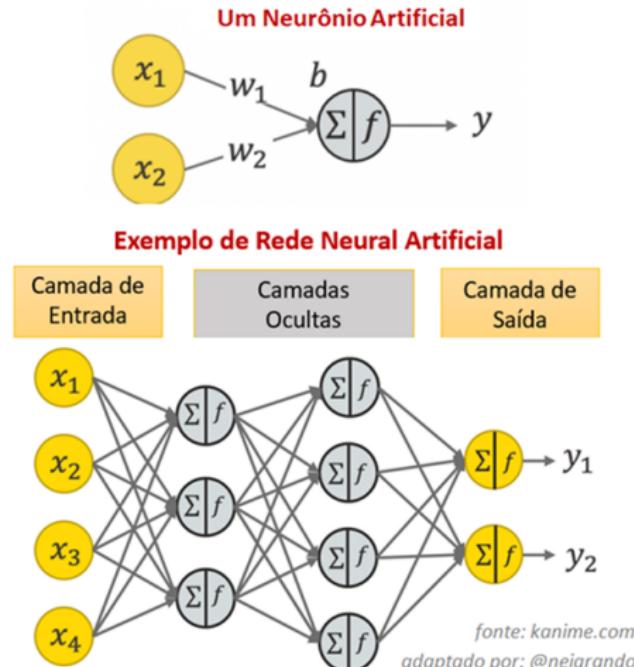


Figura: Exemplificação de um modelo de rede neural.

O Perceptron

Modelo mais simples de neurônio artificial, desenvolvido por Frank Rosenblatt em 1958. É a unidade básica de uma rede neural capaz de classificar padrões linearmente separáveis.

Ideal para:

- Classificações binárias (0,1)
- Reconhecimento de padrões
- Modelagens Lineares



FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

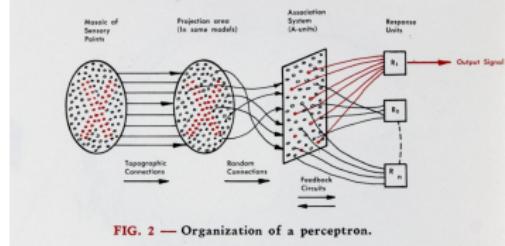


FIG. 2 — Organization of a perceptron.

Figura: Frank Rosenblatt, 1958

O Perceptron

O perceptron precisa de inúmeras entradas, passar por um processo de soma, iniciar uma função de ativação e ajustar o treinamento.

Funcionamento:

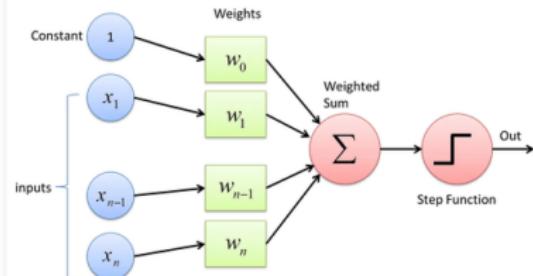
Recebe entradas multiplicadas por pesos

1 $x_1w_1 + x_2w_2 + \dots + x_nw_n + b$

2 Calcula soma ponderada (produto escalar)

Aplica função de ativação (ex: degrau)

3 $y = f(\sum x_i w_i + b)$



Treinamento:

Ajuste iterativo dos pesos para minimizar o erro:

$$w_i = w_i + \eta(t - y)x_i$$

Onde: η = taxa de aprendizado, t = valor alvo, y = saída atual

O que é uma matriz?

- Uma matriz é uma tabela retangular de números organizada em linhas e colunas.
- Notação geral: $\mathbf{A}_{m \times n}$ indica uma matriz com m linhas e n colunas.
- Exemplo:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad (2 \times 3)$$

Tipos de Matrizes

- Matriz quadrada: número de linhas = número de colunas.
- Matriz linha: apenas uma linha.
- Matriz coluna: apenas uma coluna.
- Matriz identidade: matriz quadrada com 1s na diagonal principal.

Adição de Matrizes

- Só é possível somar matrizes de mesma dimensão.
- Soma-se elemento a elemento.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Multiplicação de Matrizes

- Só é possível multiplicar $\mathbf{A}_{m \times n} \cdot \mathbf{B}_{n \times p}$.
- O número de colunas da primeira deve ser igual ao número de linhas da segunda.
- O resultado será uma matriz $m \times p$.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} (1 \cdot 5 + 2 \cdot 7) & (1 \cdot 6 + 2 \cdot 8) \\ (3 \cdot 5 + 4 \cdot 7) & (3 \cdot 6 + 4 \cdot 8) \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Perceptron: Operação com Matrizes (1/2)

- Uma camada de Perceptron realiza uma operação linear entre os pesos e as entradas:

$$\mathbf{z} = \mathbf{w} \cdot \mathbf{x} + \mathbf{b}$$

- Onde:

- \mathbf{x} : vetor de entrada ($n \times 1$)
- \mathbf{w} : matriz de pesos ($m \times n$)
- \mathbf{b} : vetor de bias ($m \times 1$)
- \mathbf{z} : vetor de saída linear ($m \times 1$)

Exemplo:

$$\mathbf{W} = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} = \begin{bmatrix} (1)(3) + (-1)(1) \\ (2)(3) + (0)(1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 + 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$$

Perceptron: Operação com Matrizes (2/2)

- Após a operação linear, aplica-se uma função de ativação elemento a elemento:

$$\mathbf{a} = f(\mathbf{z})$$

- No Perceptron clássico, usa-se a função **degrau** (step):

$$f(z_i) = \begin{cases} 1, & \text{se } z_i \geq 0 \\ 0, & \text{se } z_i < 0 \end{cases}$$

- Exemplo aplicado ao vetor $\mathbf{z} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$:

$$\mathbf{a} = f(\mathbf{z}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

O Perceptron

O perceptron precisa de inúmeras entradas, passar por um processo de soma, iniciar uma função de ativação e ajustar o treinamento.

Funcionamento:

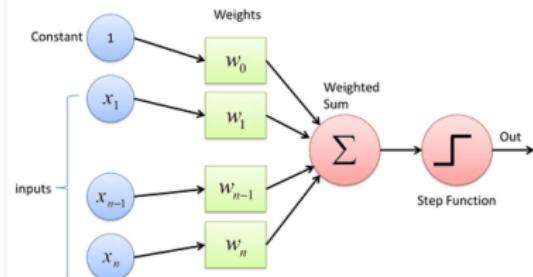
Recebe entradas multiplicadas por pesos

1 $x_1w_1 + x_2w_2 + \dots + x_nw_n + b$

2 Calcula soma ponderada (produto escalar)

Aplica função de ativação (ex: degrau)

3 $y = f(\sum x_i w_i + b)$



Treinamento:

Ajuste iterativo dos pesos para minimizar o erro:

$$w_i = w_i + \eta(t - y)x_i$$

Onde: η = taxa de aprendizado, t = valor alvo, y = saída atual

O desafio

Criar uma biblioteca de machine learning chamada IFML, buscando implementar uma biblioteca de redes neurais.

The screenshot shows a GitHub repository page for 'ifml'. At the top, there's a dropdown menu set to 'stable' and a search bar containing 'ifml'. Below the search bar, there's a button labeled 'Encontrar arquivo' and a dropdown menu labeled 'Código'. A commit history table follows, listing five commits:

Nome	Último commit	Última atualização
.vscode	Refactor - Corrections in perceptron neural model	3 meses atrás
cmake	[fix] som2 with lbase and lhum	2 meses atrás
examples/ifml	fix: improve Adaline learning and add noise cancellation...	1 semana atrás
include/ifml	test: increase sample size in uniform noise test to ensur...	2 dias atrás

Below the commit history, there's a link to 'README.md'. The main content area contains the following text:

IFML - A Machine Learning library based in C++, by IFC-Blumenau

IFML is a Machine Learning tool to support the scientific community, in data analysis. The key feature of this solution is it is solving different kinds of mathematical and statistical problems and allows the user to deal with numerous tasks using this machine learning program. This library was made using C++, because it allows the development impact of data conversion between different languages used in the MLModel, allowing have direct access to core algorithms and raw data.

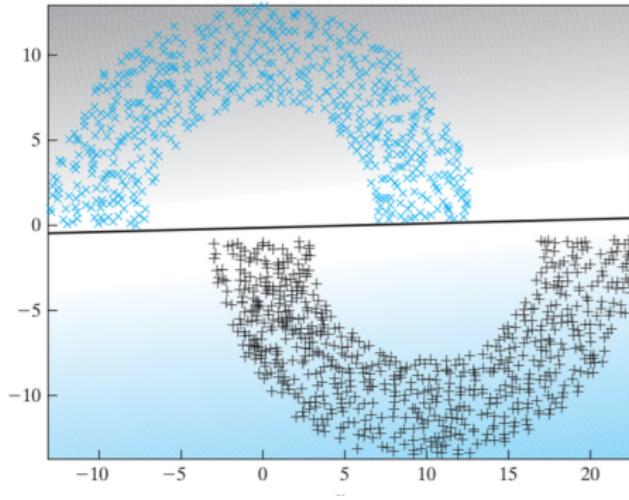
About data

This directory contains some algorithms that allows the dev team to access the training data used for machine learning model development. The data is sourced from third-party platforms like [dados.gov.br](#) and has already been prepared for training. Some of the tools what we have include is: parsing data formats to C++ data structures, read and

Figura: IFML - Biblioteca de redes neurais do IFC - Blumenau

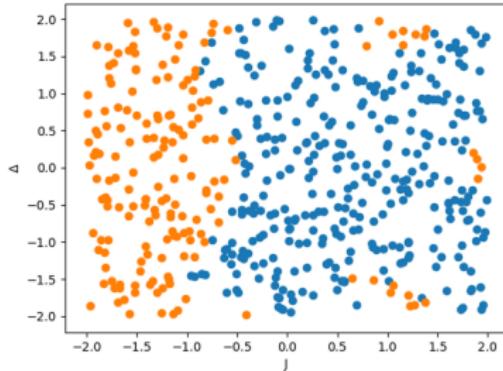
Classificação de dados

Classification using perceptron with distance = 1, radius = 10, and width = 6



- Problema **sintético** usado para testar algoritmos de **classificação não linear**.
- Consiste em duas classes com formato de **meias-luas opostas**.
- Parâmetros ajustáveis: **raio (r)**, **largura (w)** e **distância (d)**.

Uso em pesquisas



- Base de dados grande e desorganizada.
- Redes neurais multicamadas (MLPs).
- Algoritmos genéticos e meta-heurísticas.
- **Overfitting e underfitting.**
- Fronteiras de decisão não lineares.

Foco

Criar um algoritmo para classificação de dados utilizando C++ e Python para plotar os gráficos de dados arbitrários. Esse programa pode ser utilizado para classificar dados.

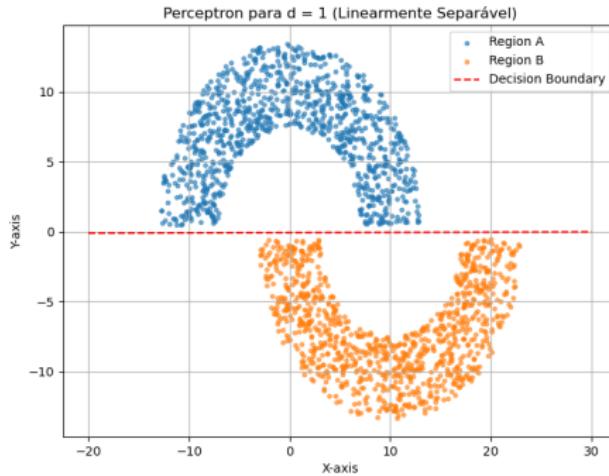


Figura: Exemplos de Aplicações Reais utilizando o Dataset Double Moon

Relação com problemas reais

Embora seja um problema artificial, o Double Moon simula desafios de classificação em muitos domínios:

Aplicação Real	Semelhança com Double Moon
Diagnóstico médico	Sintomas sobrepostos entre doenças
Reconhecimento de fala	Fonemas com variações acústicas
Segurança em redes	Atividades legítimas e maliciosas
Satélites e geociênciа	Fronteiras suaves entre regiões

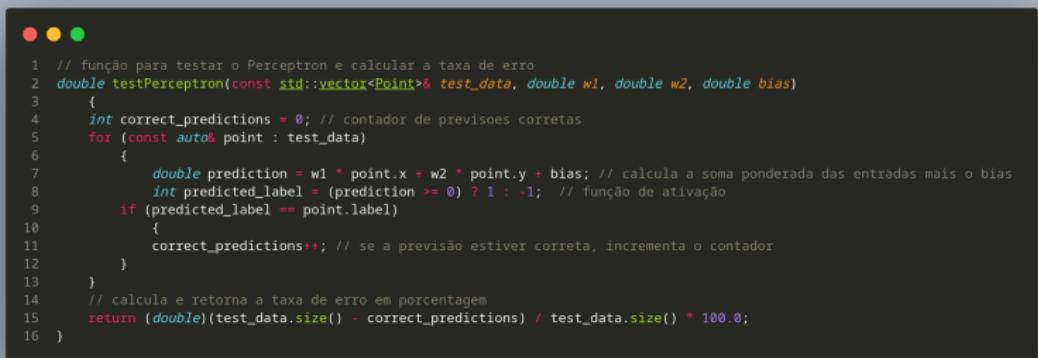
Figura: Exemplos de Aplicações Reais com Similaridade ao Dataset Double Moon

Pontuando o Double Moon

- O Double Moon é um problema didático e científico importante.
- Serve como base para **testes controlados de classificadores**.
- Auxilia no entendimento das **capacidades e limitações de redes neurais**.
- Prepara algoritmos para problemas do mundo real com dados **não linearmente separáveis**.

Tecnologias utilizadas

- Linguagem: c++
- Classe ‘Perceptron’ com funções de ativação e treinamento
- Treinamento supervisionado com dados rotulados
- Dados gerados em um arquivo de texto

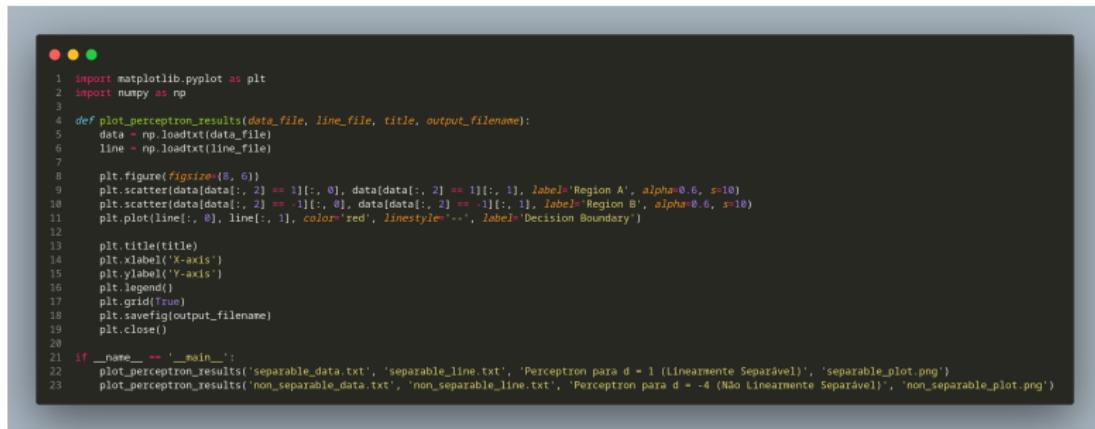


```
1 // função para testar o Perceptron e calcular a taxa de erro
2 double testPerceptron(const std::vector<Point>& test_data, double w1, double w2, double bias)
3 {
4     int correct_predictions = 0; // contador de previsões corretas
5     for (const auto& point : test_data)
6     {
7         double prediction = w1 * point.x + w2 * point.y + bias; // calcula a soma ponderada das entradas mais o bias
8         int predicted_label = (prediction >= 0) ? 1 : -1; // função de ativação
9         if (predicted_label == point.label)
10         {
11             correct_predictions++; // se a previsão estiver correta, incrementa o contador
12         }
13     }
14     // calcula e retorna a taxa de erro em porcentagem
15     return (double)(test_data.size() - correct_predictions) / test_data.size() * 100.0;
16 }
```

Figura: perceptron-clasification.cpp

Tecnologias utilizadas

- Linguagem: python e matplotlib
- plotagem do gráfico com os dados gerados



```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 def plot_perceptron_results(data_file, line_file, title, output_filename):
 5     data = np.loadtxt(data_file)
 6     line = np.loadtxt(line_file)
 7
 8     plt.figure(figsize=(8, 6))
 9     plt.scatter(data[data[:, 2] == 1][:, 0], data[data[:, 2] == 1][:, 1], label='Region A', alpha=0.6, s=10)
10     plt.scatter(data[data[:, 2] == -1][:, 0], data[data[:, 2] == -1][:, 1], label='Region B', alpha=0.6, s=10)
11     plt.plot(line[:, 0], line[:, 1], color='red', linestyle='--', label='Decision Boundary')
12
13     plt.title(title)
14     plt.xlabel('X-axis')
15     plt.ylabel('Y-axis')
16     plt.legend()
17     plt.grid(True)
18     plt.savefig(output_filename)
19     plt.close()
20
21 if __name__ == '__main__':
22     plot_perceptron_results('separable_data.txt', 'separable_line.txt', 'Perceptron para d = 1 (Linearmente Separável)', 'separable_plot.png')
23     plot_perceptron_results('non_separable_data.txt', 'non_separable_line.txt', 'Perceptron para d = -4 (Não Linearmente Separável)', 'non_separable_plot.png')
```

Figura: plot-data.py

Considerações Finais

- Para criar uma IA, é necessário um conhecimento e utilização de conceitos de redes neurais.
- O Perceptron é uma solução eficiente para classificação binária.
- Sua implementação em C++ permite controle total sobre o processo de treinamento.

Obrigado!

Dúvidas?