

**Universidade Pitágoras Unopar**

**Anhanguera**

**ENGENHARIA DE SOFTWARE**

**Elisson Nadson Souza Marques - RA 4372521401**

**PORTFÓLIO - RELATÓRIO DE AULA PRÁTICA**

NOME DA DISCIPLINA: Fundamentos da Inteligência Artificial

**Elisson Nadson Souza Marques - RA 4372521401**

**PORTFÓLIO - RELATÓRIO DE AULA PRÁTICA**

NOME DA DISCIPLINA: Fundamentos da Inteligência Artificial

Trabalho de portfólio apresentado como requisito parcial para a obtenção de pontos para a média semestral

Orientador: Vinicius Camargo Prattes

# Introdução

Neste trabalho, explorei os conceitos fundamentais de Inteligência Artificial (IA), focando no desenvolvimento de redes neurais artificiais.

Meu objetivo foi entender a aplicação prática dessas tecnologias por meio da implementação de um modelo de rede neural simples, utilizando a função sigmoide como ativação.

O trabalho também teve como meta reforçar meu conhecimento sobre aprendizado supervisionado e análise de erros.

A IA tem se tornado cada vez mais presente em nosso cotidiano, seja em assistentes virtuais, sistemas de recomendação ou no reconhecimento de padrões complexos.

Aqui, o foco foi compreender como redes neurais artificiais, inspiradas pelo funcionamento do cérebro humano, podem aprender a partir de dados e tomar decisões.

# Desenvolvimento

Primeiramente, foi implementada uma rede neural de camada única em Python.

Para isso, utilizei as bibliotecas **NumPy** e **Matplotlib**. Abaixo, explico as etapas do processo e apresento o código desenvolvido:

1. **Importação das Bibliotecas e Configuração Inicial:**

As bibliotecas necessárias foram importadas, incluindo funções para calcular a ativação sigmoide e sua derivada, que são essenciais para o treinamento da rede.

import numpy as np

import matplotlib.pyplot as plt

def sigmoide(x):

return 1 / (1 + np.exp(-x))

def derivada\_sigmoide(x):

return x \* (1 - x)

1. **Inicialização dos Pesos e Dados:** Os pesos sinápticos foram inicializados aleatoriamente, e os dados de entrada e saída foram definidos.

def inicializar\_pesos(num\_entradas):

np.random.seed(1)

return 2 \* np.random.random((num\_entradas, 1)) - 1

entradas = np.array([[0, 0, 1],

[1, 1, 1],

[1, 0, 1],

[0, 1, 1]])

saidas = np.array([[0], [1], [1], [0]])

1. **Treinamento da Rede Neural:** O treinamento foi realizado utilizando a técnica de retropropagação. A rede ajustou os pesos iterativamente para minimizar o erro quadrático médio.

def treinar\_rede(entradas, saidas, num\_iteracoes, taxa\_aprendizado):

num\_amostras, num\_entradas = entradas.shape

pesos = inicializar\_pesos(num\_entradas)

erros = []

for iteracao in range(num\_iteracoes):

saida\_camada\_1 = sigmoide(np.dot(entradas, pesos))

erro = saidas - saida\_camada\_1

erros.append(np.mean(erro\*\*2))

ajustes = derivada\_sigmoide(saida\_camada\_1) \* erro

pesos += taxa\_aprendizado \* np.dot(entradas.T, ajustes)

return pesos, erros

1. **Avaliação e Visualização dos Resultados:** Após o treinamento, os pesos finais foram exibidos, e o erro foi plotado ao longo das iterações para análise.

pesos, erros = treinar\_rede(entradas, saidas, 10000, 0.1)

plt.plot(erros)

plt.title("Erro durante o treinamento")

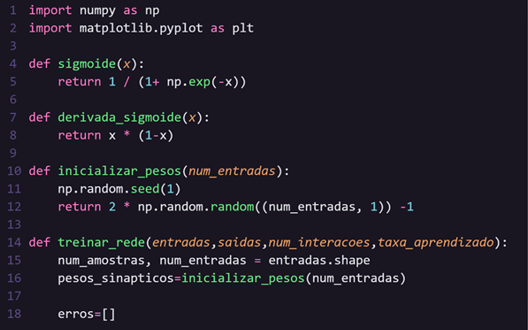
plt.xlabel("Número de Iterações")

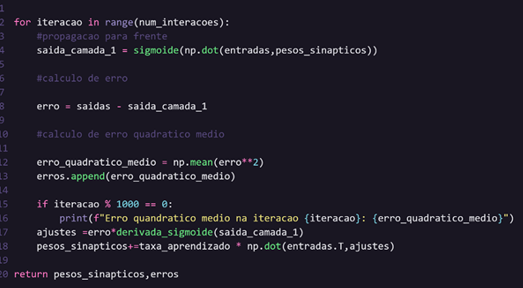
plt.ylabel("Erro Quadrático Médio")

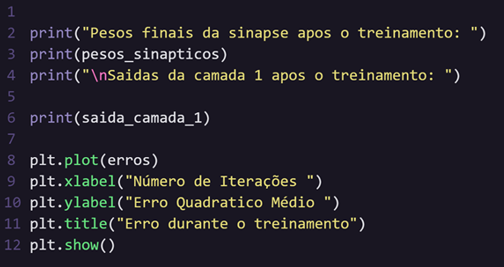
plt.show()

# Imagens

1. **Código em Python**  
   As seguintes imagens ilustram a implementação e execução do código:
   * Importação de bibliotecas e definição de funções:
   * Treinamento e ajuste de pesos:







# Conclusão

Por meio deste projeto, entendi como as redes neurais artificiais funcionam e como elas podem ser treinadas para solucionar problemas.

A implementação prática reforçou a importância de técnicas como retropropagação e a análise de erros, fundamentais para o aprendizado de máquina.

# Referências

1. Brasil Escola. "Inteligência Artificial". Disponível em: <https://brasilescola.uol.com.br>
2. Amazon Web Services. "Redes Neurais". Disponível em: <https://aws.amazon.com>