Lab. 1: O Primeiro Neurônio e o Versionamento

Aléssio Miranda Júnior

Agosto - 2026/2

Lab. 01: O Primeiro Neurônio e o Versionamento

⋆ Objetivo do Laboratório

Bem-vindo ao seu primeiro dia prático. A missão de hoje é estabelecer a base da sua carreira como Engenheiro de Sistemas Inteligentes. Faremos duas coisas que são vitais para o mercado de trabalho:

1. Montar um fluxo corporativo de versionamento de código no GitLab.
2. Programar a matemática "crua" de um neurônio artificial utilizando apenas Python puro e Álgebra Linear, entendendo que a Inteligência Artificial não possui mágica, apenas matrizes.

—

1 Parte 1: Ambiente e Cultura Git

Na indústria tecnológica de alto impacto, nenhum código sobrevive isolado na máquina de um único engenheiro.

A. Criando o Repositório Institucional a partir do Template

1. Acesse o **GitLab Institucional** em `https://git.juninho.com.br`.
2. Navegue até o repositório modelo público: `https://git.juninho.com.br/cefet-ia2-2026/modulo-1/ia2-modulo1-template`.
3. Clique no botão **Fork** no canto superior direito para criar a sua cópia pessoal do projeto no GitLab.
4. Clone o seu Fork no seu computador de trabalho:

```
git clone https://git.juninho.com.br/<seu-usuario>/ia2-modulo1-template.git
cd ia2-modulo1-template
```

B. Clonando e Configurando na Máquina Local

1. Abra o terminal do computador do laboratório.
2. Clone o seu novo repositório: `git clone URL_DO_SEU_REPOSITORIO`.
3. Entre na pasta gerada: `cd ia2-2025-SeuNome`.
4. Abra a pasta no seu editor de código (como o VS Code): `code .`.

—

2 Parte 2: O Desafio do Perceptron

Nós não vamos usar *TensorFlow* ou bibliotecas avançadas prontas. Você vai construir o "cérebro" de um neurônio na unha utilizando a biblioteca `NumPy`.

Problema A: O Robô Jardineiro

Contexto: A prefeitura de *TechVille* instalou um robô autônomo na praça central para regar o jardim. A bateria é cara, então ele só deve ligar a mangueira quando realmente for necessário. Os engenheiros da prefeitura já calibraram os sensores e determinaram os pesos ideais — a sua missão é **traduzir esse projeto em código**.

Modelo de Entrada (X): Vetor com 3 leituras dos sensores do robô (0 ou 1):

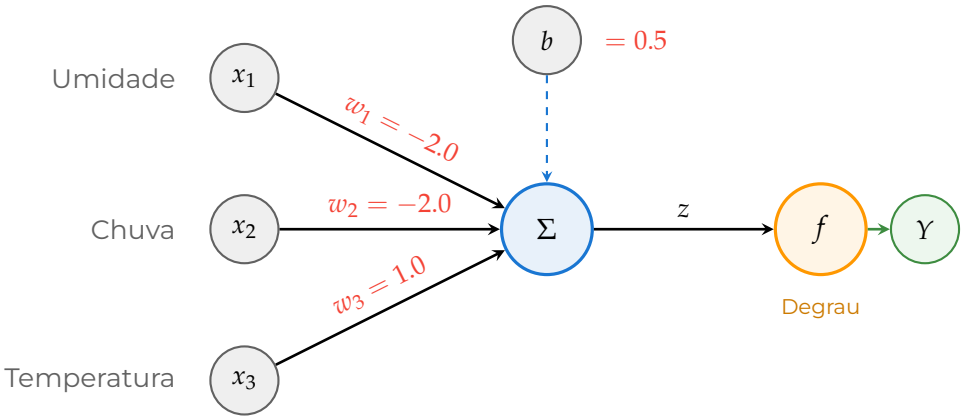
- x_1: Umidade do Solo (0 = Seco, 1 = Molhado)
- x_2: Previsão de Chuva (0 = Céu Limpo, 1 = Tempestade)
- x_3: Temperatura (0 = Frio, 1 = Quente)

Modelo de Saída (Y): Decisão binária via Função Degrau (limiar $\theta = 1.0$):

- $Y = 1$ $\rightarrow$ Ligar a mangueira
- $Y = 0$ $\rightarrow$ Ficar em repouso

A. O Projeto do Neurônio

Os engenheiros entregaram a você o seguinte diagrama com os pesos já calibrados. O neurônio calcula a soma ponderada $z = \mathbf{x} \cdot \mathbf{w} + b$ e aplica a função degrau para decidir.



$$z = x_1 \cdot (-2.0) + x_2 \cdot (-2.0) + x_3 \cdot (1.0) + 0.5$$

$$Y = \begin{cases} 1, & \text{se } z \geq 1.0 \\ 0, & \text{caso contrário} \end{cases}$$

B. Tabela-Verdade Esperada

Com os pesos do diagrama acima, o comportamento esperado do robô para **todas** as combinações possíveis dos sensores é:

x_1 **(Umidade)**	x_2 **(Chuva)**	x_3 **(Temp.)**	$z = \mathbf{x} \cdot \mathbf{w} + b$	Y **(Decisão)**
0 (Seco)	0 (Limpo)	0 (Frio)	0.5	0 — Repouso
0 (Seco)	0 (Limpo)	1 (Quente)	1.5	accentgreen!15**1 — Regar!**
0 (Seco)	1 (Chuva)	0 (Frio)	-1.5	0 — Repouso
0 (Seco)	1 (Chuva)	1 (Quente)	-0.5	0 — Repouso
1 (Molhado)	0 (Limpo)	0 (Frio)	-1.5	0 — Repouso
1 (Molhado)	0 (Limpo)	1 (Quente)	-0.5	0 — Repouso
1 (Molhado)	1 (Chuva)	0 (Frio)	-3.5	0 — Repouso
1 (Molhado)	1 (Chuva)	1 (Quente)	-2.5	0 — Repouso

Observe: o robô só liga a mangueira no **único** cenário em que faz sentido — solo seco, sem previsão de chuva, e dia quente. Os pesos negativos em w_1 e w_2 funcionam como “penalidades” que inibem o disparo quando já está molhado ou quando vai chover.

C. Implementando o Código

Agora traduza o diagrama acima para código. Crie o arquivo `src/neuronio.py` e implemente as duas funções. O `numpy` aplica o produto escalar sem precisarmos de laços `for`:

```
1 import numpy as np
2
3 def funcao_degrau(soma):
4     """Função de Ativação Degrau (Step Function).
```

```
    Retorna 1 se soma >= limiar, 0 caso contrário."""
    return 1 if soma >= 1.0 else 0

def neuronio(entradas, pesos, vies):
    """Equação do Perceptron: Y = f(X . W + b)"""
    soma_ponderada = np.dot(entradas, pesos) + vies
    return funcao_degrau(soma_ponderada)

if __name__ == "__main__":
    # Pesos calibrados pelos engenheiros (do diagrama)
    W = np.array([-2.0, -2.0, 1.0])
    b = 0.5

    # Teste: Solo Seco(0), Sem Chuva(0), Quente(1) -> deve regar (1)
    X = np.array([0, 0, 1])
    decisao = neuronio(X, W, b)
    print(f"Cenário {X} -> Decisão: {decisao}")

    # Teste: Solo Molhado(1), Sem Chuva(0), Quente(1) -> repouso (0)
    X2 = np.array([1, 0, 1])
    decisao2 = neuronio(X2, W, b)
    print(f"Cenário {X2} -> Decisão: {decisao2}")
```

—

3 Parte 3: Garantia de Qualidade (Testes Unitários)

Em sistemas reais de Inteligência Artificial, a predição deve ser rigorosamente testada antes da entrega. Os testes já estão prontos no arquivo `tests/test_perceptron.py` do repositório. Abra-o e observe a estrutura:

```
import numpy as np
from src.neuronio import neuronio

# Pesos do diagrama do Robô Jardineiro
W = np.array([-2.0, -2.0, 1.0])
b = 0.5

def test_regar_cenario_ideal():
    """Solo Seco, Sem Chuva, Quente -> deve regar (1)"""
    assert neuronio(np.array([0, 0, 1]), W, b) == 1

def test_repouso_solo_molhado():
    """Solo Molhado, Sem Chuva, Quente -> repouso (0)"""
    assert neuronio(np.array([1, 0, 1]), W, b) == 0

def test_repouso_chuva():
    """Solo Seco, Chuva, Quente -> repouso (0)"""
    assert neuronio(np.array([0, 1, 1]), W, b) == 0
```

```
19
20 def test_repouso_frio():
21     """Solo Seco, Sem Chuva, Frio -> repouso (0)"""
22     assert neuronio(np.array([0, 0, 0]), W, b) == 0
```

1. Rode os testes no terminal com o comando: `pytest tests/ -v`.
2. Se a tela exibir **4 passed** em verde, significa que o seu neurônio reproduz corretamente a tabela-verdade do diagrama.

—

4 Parte 4: Integração Contínua (A Entrega)

Sua implementação está pronta e validada localmente. Agora vamos enviar o código para o servidor e deixar o pipeline de CI/CD confirmar que tudo está correto.

1. No terminal, adicione os arquivos à zona de preparação:
   `git add src/neuronio.py`.
2. Empacote a versão com um comentário claro:
   `git commit -m "Lab 01: Implementação do Perceptron Jardineiro"`.
3. Envie a atualização para o servidor remoto: `git push origin main`.

Critério de Avaliação: O pipeline de CI/CD do GitLab irá interceptar o seu *push*, criará um ambiente em nuvem invisível, e executará o `pytest` automaticamente. Você terá nota máxima ao obter o "Selo Verde" de aprovação na plataforma.

✔ Takeaways

Parabéns! Você acabou de implementar a matemática fundamental de um neurônio artificial — o mesmo bloco que, empilhado milhões de vezes, forma o ChatGPT e os sistemas de visão computacional modernos. Na próxima aula, vamos empilhar vários desses neurônios para criar uma **Rede Neural Multicamadas (MLP)** e resolver problemas que um único Perceptron não consegue.