

Laboratório de Fusão e
Inteligência Artificial Aplicada
(LaFIAA)

Introdução ao Aprendizado Supervisionado



Instituto de Pesquisas
da Marinha

Prof. Pablo Rangel

*Encarregado da Divisão de Desenvolvimento, Operações e Qualidade
Coordenador do LaFIAA*



Neurônio Biológico



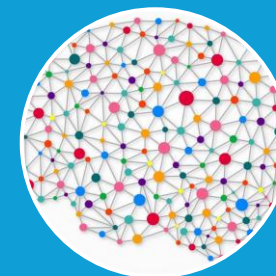
O cérebro humano
como inspiração



Todo ser humano
aprende a
reconhecer objetos,
pessoas e
situações.

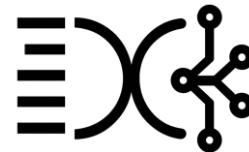


Você não nasce com
esse conhecimento
— você observa,
tenta, erra e corrige.
Aos poucos, seu
cérebro aprende
padrões.



Como um modelo
computacional pode
fazer o mesmo?



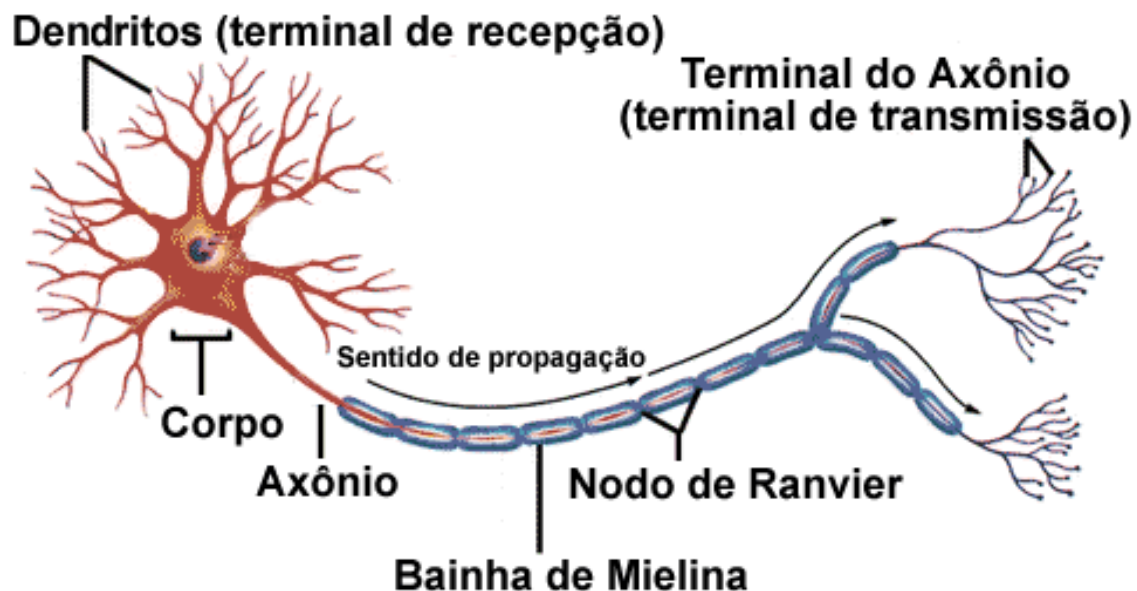


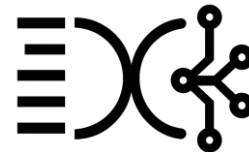
Neurônio Biológico

Dendritos recebem
sinais de outros
neurônios

Neurônio agrega os
sinais recebidos de
todos os neurônios

Axônio transmite
impulso se limiar
for superado

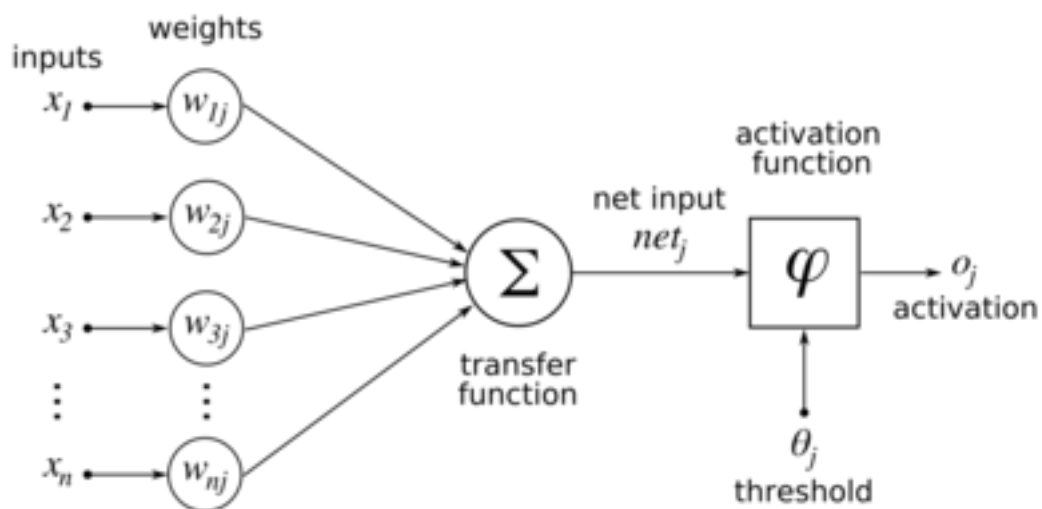


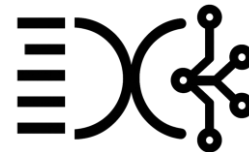


Neurônio Artificial (McCulloch & Pitts, 1943)

Na década de 1940, McCulloch e Pitts propuseram um modelo matemático do neurônio.

Já nos anos 1950, Frank Rosenblatt criou o Perceptron, o primeiro modelo de rede capaz de aprender tarefas simples.





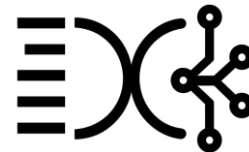
Representação de Entradas e Atributos

Entradas são um conjunto de atributos (a) que você deseja que uma rede neural aprenda.

Por exemplo, você gostaria de aprender sobre o perfil socioeconômico de uma população. A amostra pode ter como base o salário bruto e a idade.

Como estão em domínio de valores distintos, os atributos das amostras são convertidas para um intervalo entre 0 e 1

$$f(a) = \frac{valor - min}{max - min}$$



Representação de Entradas e Atributos

Seja Pessoa 1 x_1 e
Pessoa 2 x_2 .

x_1 possui 49 anos
e salário de R\$
20.000,00.

x_2 possui 24 anos
e salário de
R\$5.000,00.

Convertemos os
atributos
utilizando o
método Min-Max.

Seja 18 anos e 120
anos as idades
mínima e máxima.

Seja R\$1.200,00 e
R\$30.000,00 os
salários mínimo e
máximos.

$$X_1^{idade} = \frac{49-18}{120-18} = \frac{31}{102} \approx 0,30$$

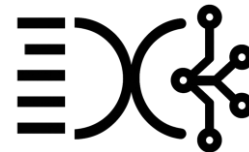
$$X_1^{salario} = \frac{20000-1200}{30000-1200} = \frac{18800}{28800} \approx 0,65$$

$$X_1 = [0.3, 0.65]$$

$$X_2^{idade} = \frac{24-18}{120-18} = \frac{6}{102} \approx 0,05$$

$$X_2^{salario} = \frac{5000-1200}{30000-1200} = \frac{3800}{28800} \approx 0,13$$

$$X_2 = [0.05, 0.13]$$



Representação de Neurônios: inicialização

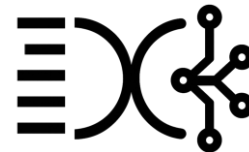
Neurônio N é representado pelos pesos.

Requer vetor de pesos w com o quantitativo de valores para cada amostra existente.

A inicialização pode ser com valores pré-fixados ou por função randômica r .

Pré-fixado: $w_i = [0, 0, 0, 0]$.

Randômico : $w_i = [r(0, 1), r(0, 1), r(0, 1), r(0, 1)]$.



Representação de Neurônios: Soma

x_i : i -ésimo atributo a ser considerada no aprendizado.

w_i : i -ésimo peso do neurônio.

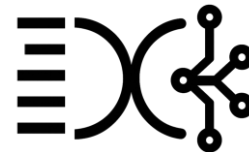
b : valor de ajuste para ativação.

ϕ : função de ativação.

y : saída do neurônio.

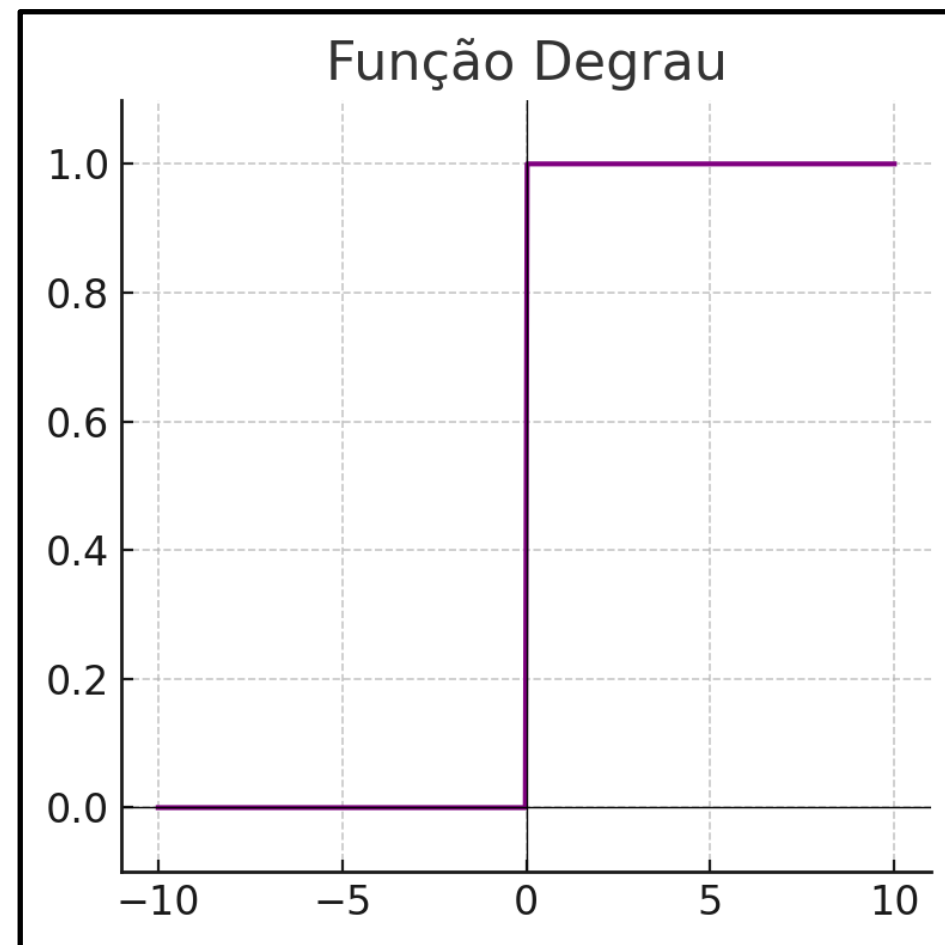
$$z = \sum_{i=1}^n w_i x_i + b$$

$$y = \phi(z)$$

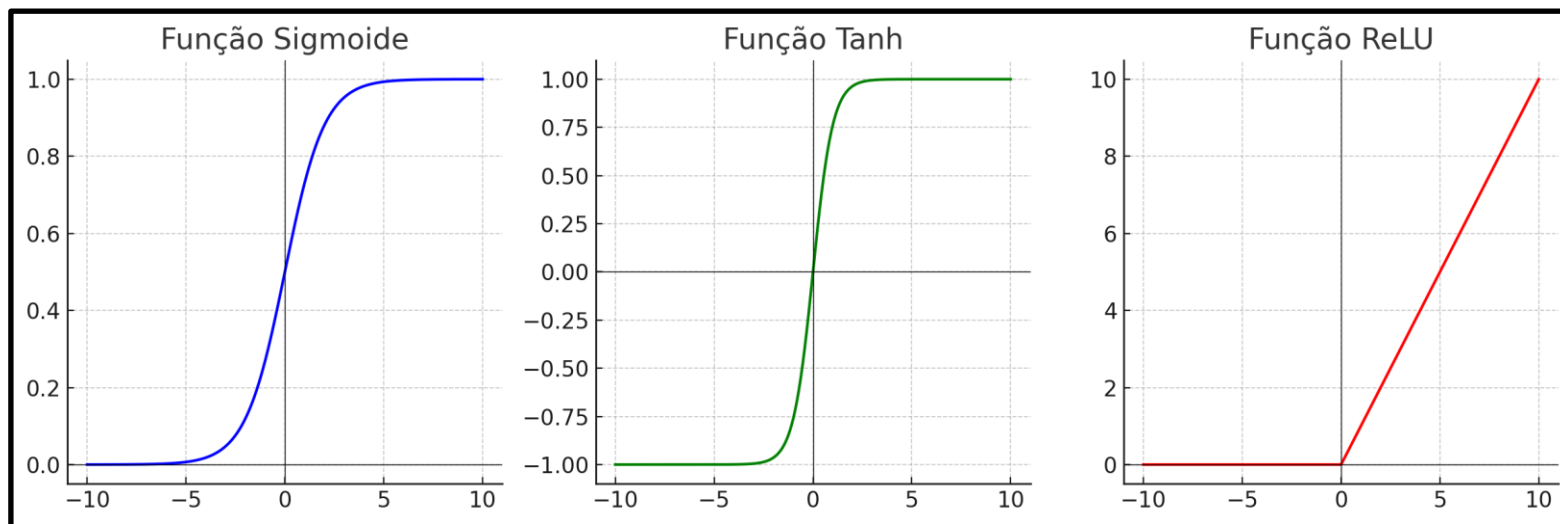


Representação de Neurônios: Ativação

$$\phi(z) = \begin{cases} 1, z \geq 0 \\ 0, \text{caso contrário} \end{cases}$$



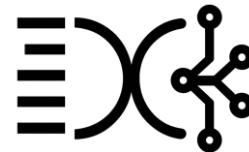
Representação de Neurônios: Ativação



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

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

$$\phi(z) = \max(0, z)$$



O Perceptron de Rosenblatt (1958): Modelo

Modelo de rede neural de camada única, usada para classificação binária.

Requer exemplos de entrada x e a respectiva saída esperada (*ground truth*) g e consecutivas iterações.

É limitada à classificação binária, não sendo capaz de lidar com problemas não-lineares

Ajusta os pesos em função do erro ϵ um parâmetro η que indica a taxa de aprendizado.

$$\epsilon = (g - y)$$

$$w_i = w_i + \epsilon \cdot x_i \cdot \eta$$

$$b = b + \eta \cdot \epsilon$$

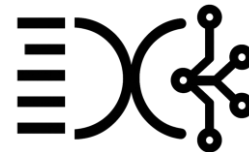
O Perceptron de Rosenblatt (1958): Exemplo

Um exemplo com
a porta AND.

Sejam 4 entradas
 x para $0 \leq k \leq 4$
com 2 atributos
para $0 \leq i \leq 2$.

Assim, temos x_{ki} .

$x_{0,0}$	0	$x_{0,1}$	0	g_k	0
$x_{1,0}$	0	$x_{1,1}$	1	g_k	0
$x_{2,0}$	1	$x_{2,1}$	0	g_k	0
$x_{3,0}$	1	$x_{3,1}$	1	g_k	1



O Perceptron de Rosenblatt (1958): Código

Paradigma
Orientado a Objetos

Linguagem Python.

[https://github.com/pablorangel82/
introducao_aprendizado_supervisionado](https://github.com/pablorangel82/introducao_aprendizado_supervisionado)

O Perceptron de Rosenblatt (1958): Limitações

Um exemplo com a
porta XOR.

Sejam k entradas
($0 \leq k \leq 4$) cada
uma com i atributos
($0 \leq i \leq 2$).

Assim, temos x_{ki} .

$x_{0,0}$	0	$x_{0,1}$	0	d_k	0
$x_{1,0}$	0	$x_{1,1}$	1	d_k	1
$x_{2,0}$	1	$x_{2,1}$	0	d_k	1
$x_{3,0}$	1	$x_{3,1}$	1	d_k	0

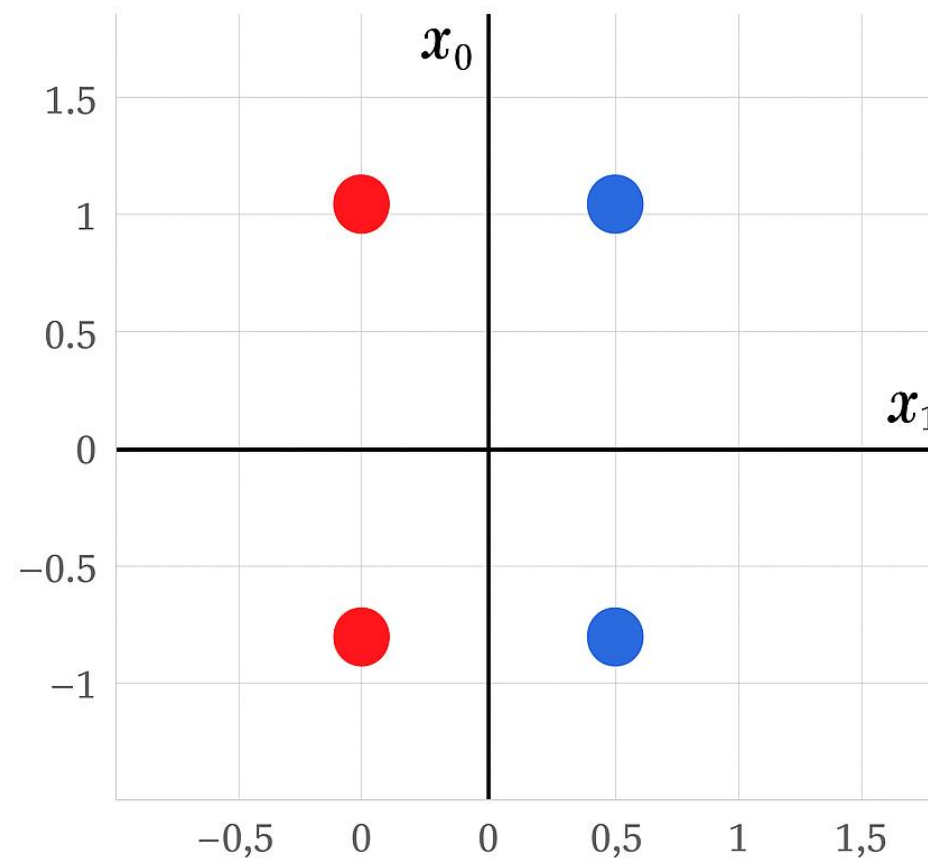
O Perceptron de Rosenblatt (1958): Limitações

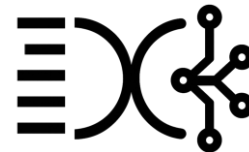
Um exemplo com a porta XOR.

Sejam k entradas ($0 \leq k \leq 4$) cada uma com i atributos ($0 \leq i \leq 2$).

Assim, temos x_{ki} .

XOR – Não linearmente separável





Uma possibilidade...

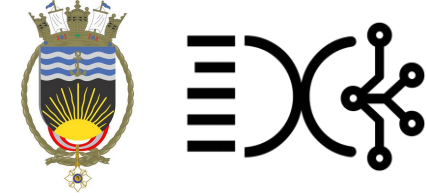
Podemos expressar o
XOR como uma
expressão
combinatória entre
AND e OR

$$\oplus (x_1, x_2) = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2)$$

Precisamos de uma
arquitetura que
possibilite essa
solução...

$x_{0,0}$	0	$x_{0,1}$	0	$(0 \vee 0) \wedge \neg(0 \wedge 0)$	d_0	0
$x_{1,0}$	0	$x_{1,1}$	1	$(0 \vee 1) \wedge \neg(0 \wedge 1)$	d_1	1
$x_{2,0}$	1	$x_{2,1}$	0	$(1 \vee 0) \wedge \neg(1 \wedge 0)$	d_2	1
$x_{3,0}$	1	$x_{3,1}$	1	$(1 \vee 1) \wedge \neg(1 \wedge 1)$	d_3	0

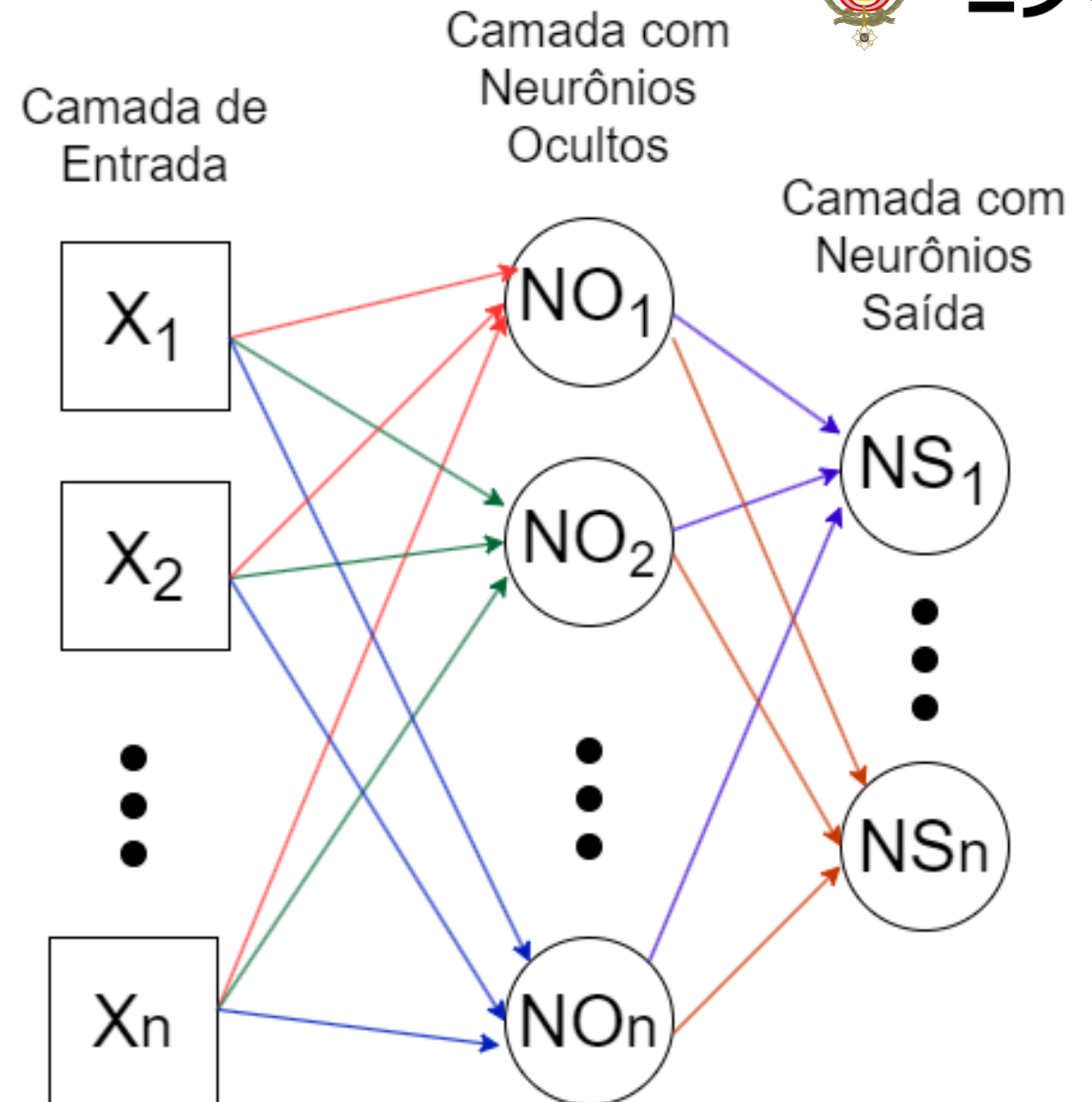
MultiLayer Perceptron (MLP)

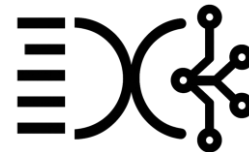


Entradas x apresentadas para cada camada oculta da rede.

As saídas dos neurônios da camada oculta servirão de entrada para as camadas seguintes (*forward pass*).

Retropropagar o erro e ajustar os pesos desde a camada de saída até a primeira camada oculta (*backward pass*).





MultiLayer Perceptron (MLP): *forward pass*

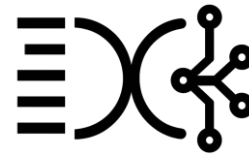
Considere o neurônio de uma camada existente.

Uma entrada é representada como x .

Considere o i -ésimo peso w ou atributo de uma entrada.

b : valor de ajuste para ativação.

$$z = b + \sum_i w_i \cdot x_i$$
$$y = \phi(z) = \frac{1}{1 + e^{-z}}$$



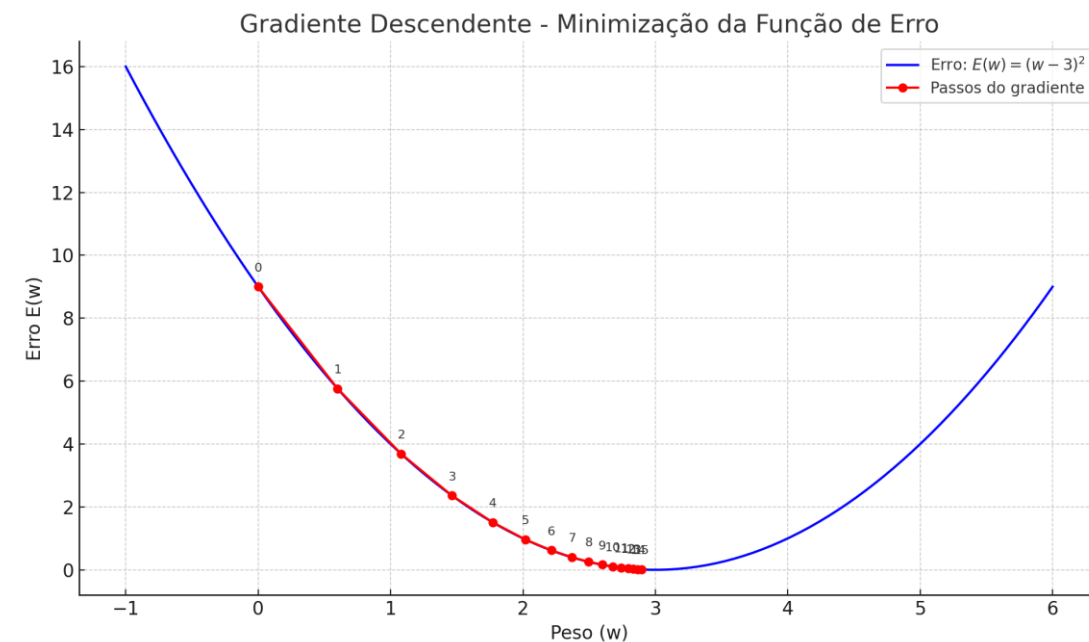
MultiLayer Perceptron (MLP): *backward pass*

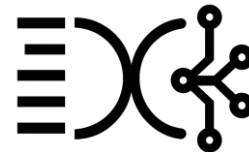
Como tratar erros descobertos nas camadas e ajustar os pesos retroativamente?



Algoritmo de Backpropagation.

$$\textit{Gradiente} : \frac{\partial \epsilon}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w}$$





MultiLayer Perceptron (MLP): *backward pass*

Como calcular a
primeira derivada
parcial?

$$E^{saida} = \frac{1}{2}(g - y^{saida})^2$$

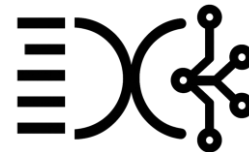
$$(g - y^{saida})^2 = \mu^2$$

$$\epsilon = \frac{\partial E}{\partial y} = \frac{1}{2} \cdot 2(g - y^{saida}) \cdot \frac{d\mu}{dy}$$

$$\epsilon = \frac{\partial E}{\partial y} = \frac{1}{2} \cdot 2(g - y^{saida}) \cdot -1 = -(g - y^{saida})$$

$$\epsilon^{saida} = y^{saida} - g$$

$$Gradiente : \frac{\partial \epsilon}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w}$$

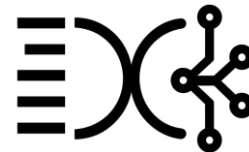


MultiLayer Perceptron (MLP): *backward pass*

Como calcular a
segunda derivada
parcial?

$$y^{saida} = \phi(z^{saida}) = \frac{1}{1 + e^{-z^{saida}}}$$
$$\frac{\partial y}{\partial z} = \frac{d\phi(z)}{dz} = \phi(z)(1 - \phi(z))$$
$$\delta^{saida} = y^{saida}(1 - y^{saida}) \cdot \epsilon^{saida}$$

$$Gradiente : \frac{\partial \epsilon}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w}$$

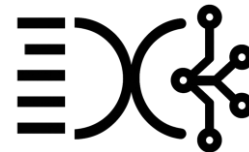


MultiLayer Perceptron (MLP): *backward pass*

Como calcular a
terceira derivada
parcial?

$$\begin{aligned}\frac{d}{dz} &= y^{saida} (1 - y^{saida}) = \delta^{saida} \\ \epsilon^{oculto} &= y^{oculto} (1 - y^{oculto}) \\ \delta^{oculto} &= \delta^{saida} \cdot w^{saida} \cdot \epsilon^{oculto}\end{aligned}$$

$$Gradiente : \frac{\partial \epsilon}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w}$$



MultiLayer Perceptron (MLP): *backward pass*

Com base no erro calculado retroativamente, vamos ajustar os pesos das camadas!

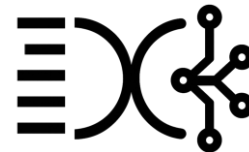
Como ajustar os pesos?

$$w_i^{saida} = w_i^{saida} - (\eta \cdot \delta^{saida} * y_i^{oculto})$$

$$b_i^{saida} = b_i^{saida} - (\eta \cdot \delta^{saida})$$

$$w_i^{oculto} = w_i^{oculto} - (\mu \cdot \delta^{oculto} * x_i)$$

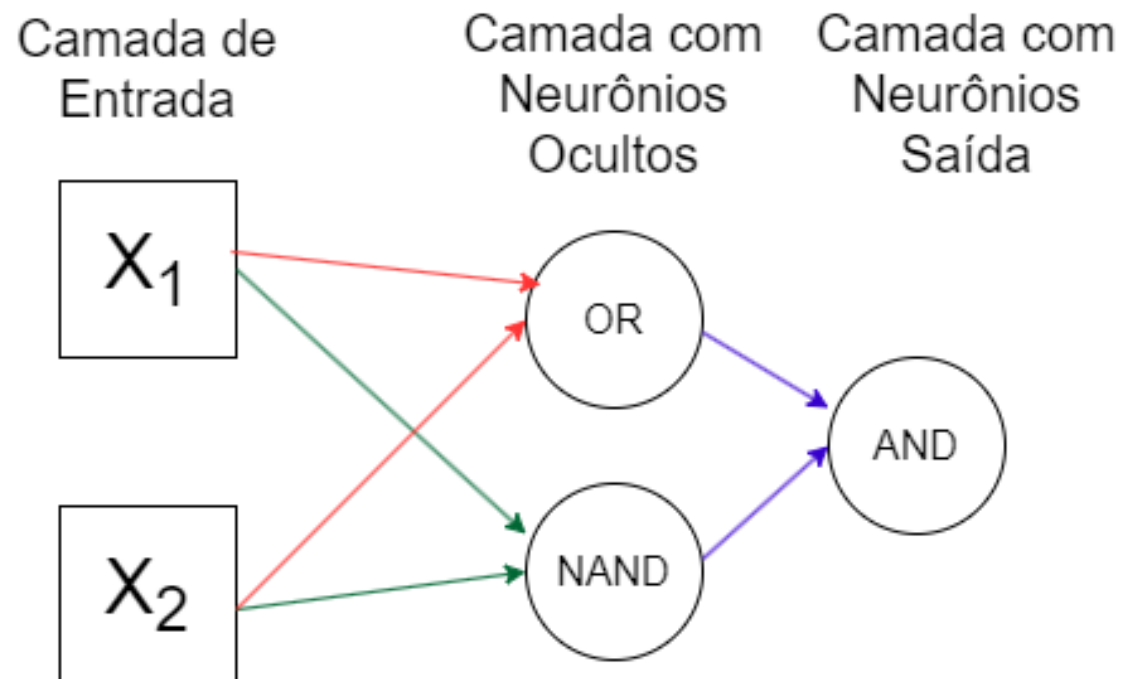
$$b_i^{oculto} = b_i^{oculto} - (\eta \cdot \delta^{oculto})$$

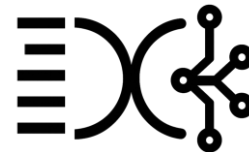


MultiLayer Perceptron (MLP): resolvendo o XOR



Na camada oculta, ter um neurônio para a operação OR e um neurônio para operação NOT AND

Na camada saída, ter um neurônio para agregação dos resultados da camada oculta (AND).





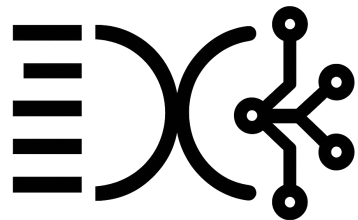
MultiLayer Perceptron (MLP): Código



Paradigma
Orientado a Objetos

Linguagem Python.

[https://github.com/pablorangel82/
introducao_aprendizado_supervisionado](https://github.com/pablorangel82/introducao_aprendizado_supervisionado)



**Laboratório de Fusão e
Inteligência Artificial Aplicada
(LaFIAA)**



**Instituto de Pesquisas
da Marinha**

Dúvidas?

pablo.rangel@marinha.mil.br