Redes Neurais

alura

Começando pela História...

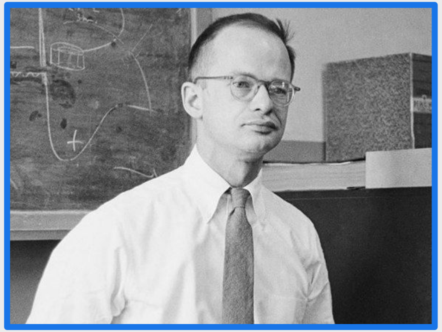
1943

Warren McCulloch e Walter Pitts

Publicação do 1° artigo descrevendo o funcionamento de um neurônio em uma arquitetura.

Warren McCulloch

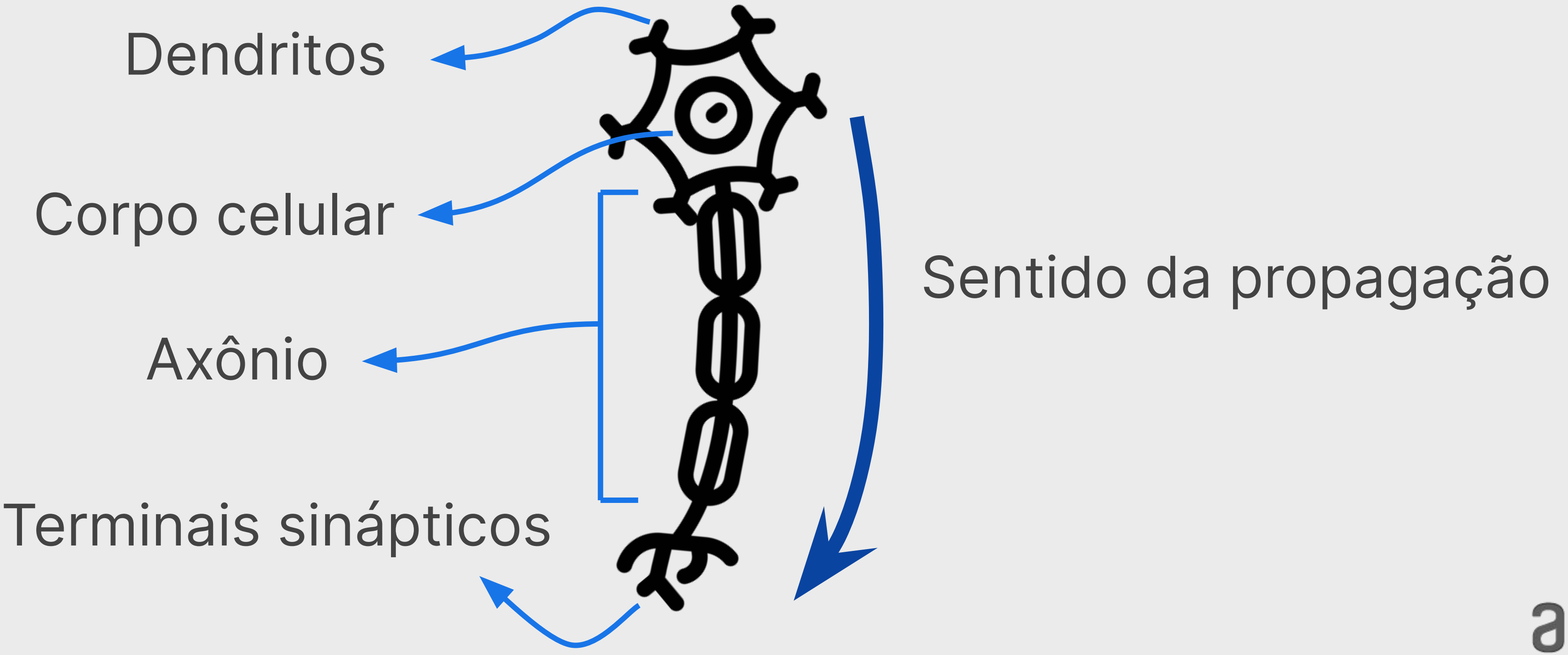
Neurocientista



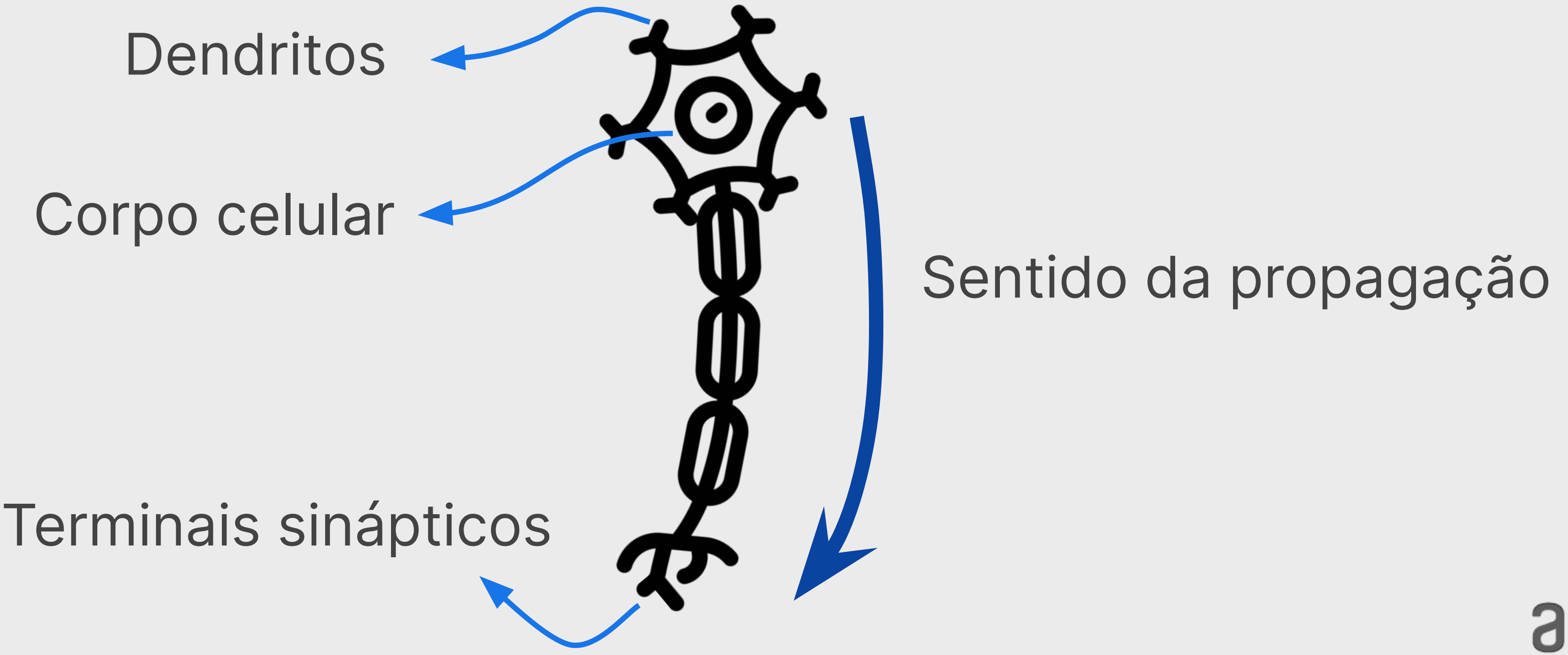
Walter Pitts

Matemático

Funcionamento de um neurônio



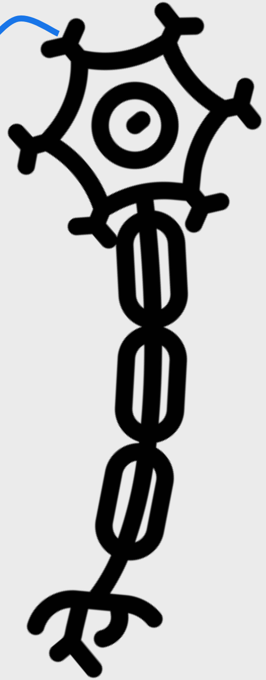
Funcionamento de um neurônio



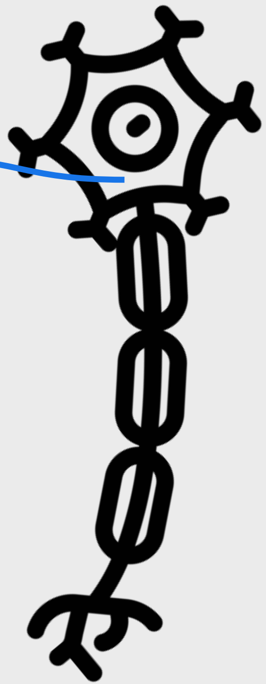
Funcionamento de um neurônio

Dendritos

Recebem os sinais sinápticos



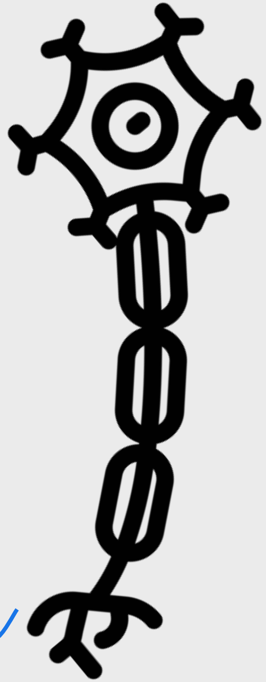
Funcionamento de um neurônio



Corpo celular

Processa os sinais recebidos

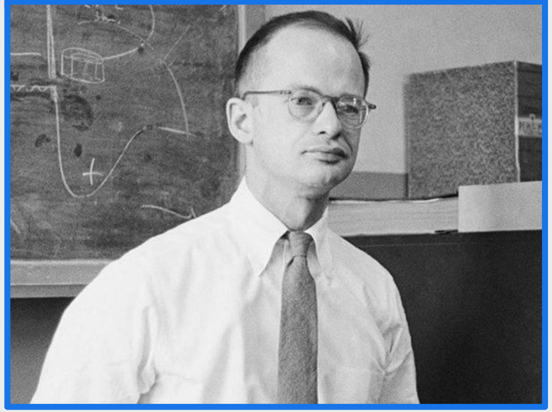
Funcionamento de um neurônio



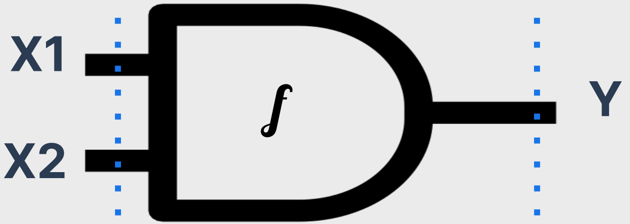
Terminais sinápticos

Enviam o sinal para outros neurônios

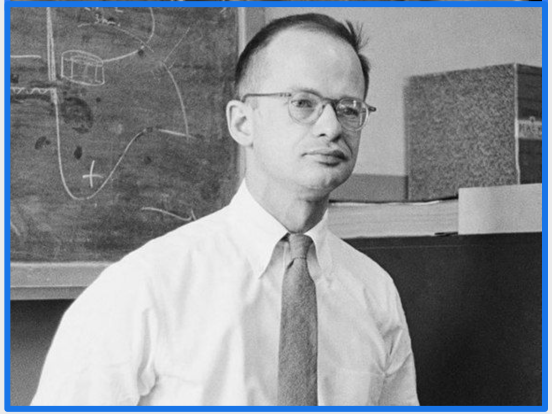
Funcionamento de um neurônio artificial

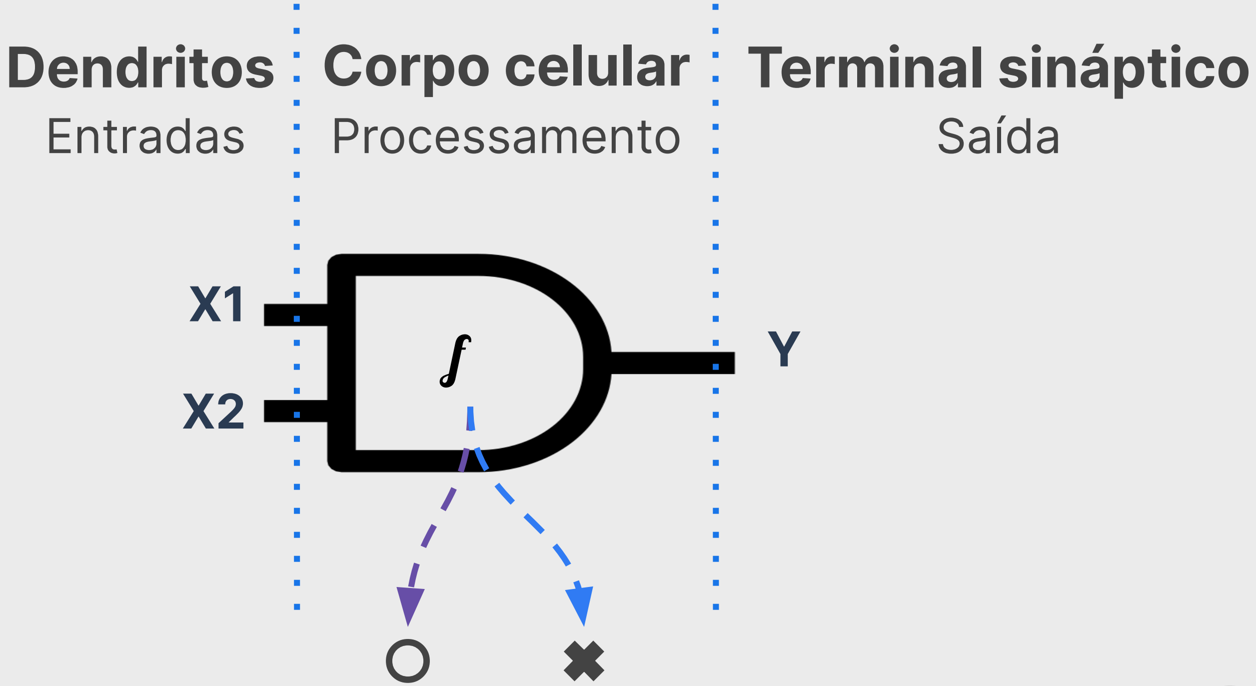


Dendritos	Corpo celular	Terminal sináptico
Entradas	Processamento	Saída

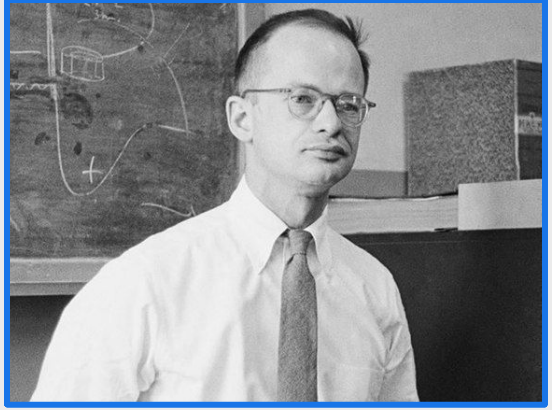


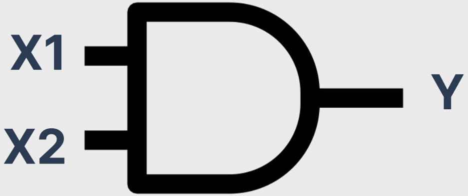
Funcionamento de um neurônio artificial



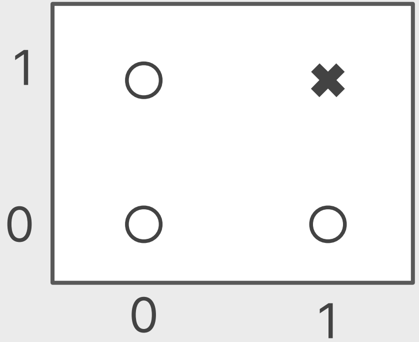


Funcionamento de um neurônio artificial



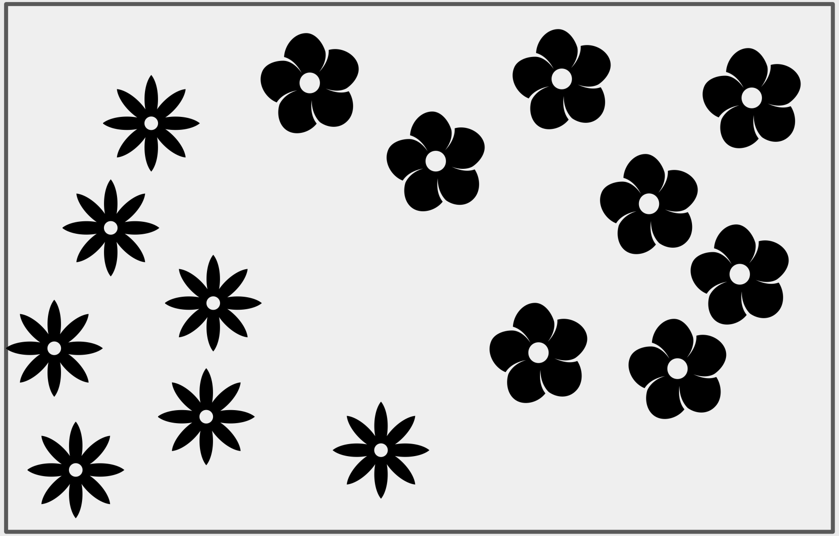


X1	X2	Y
0	0	○
0	1	○
1	0	○
1	1	✖



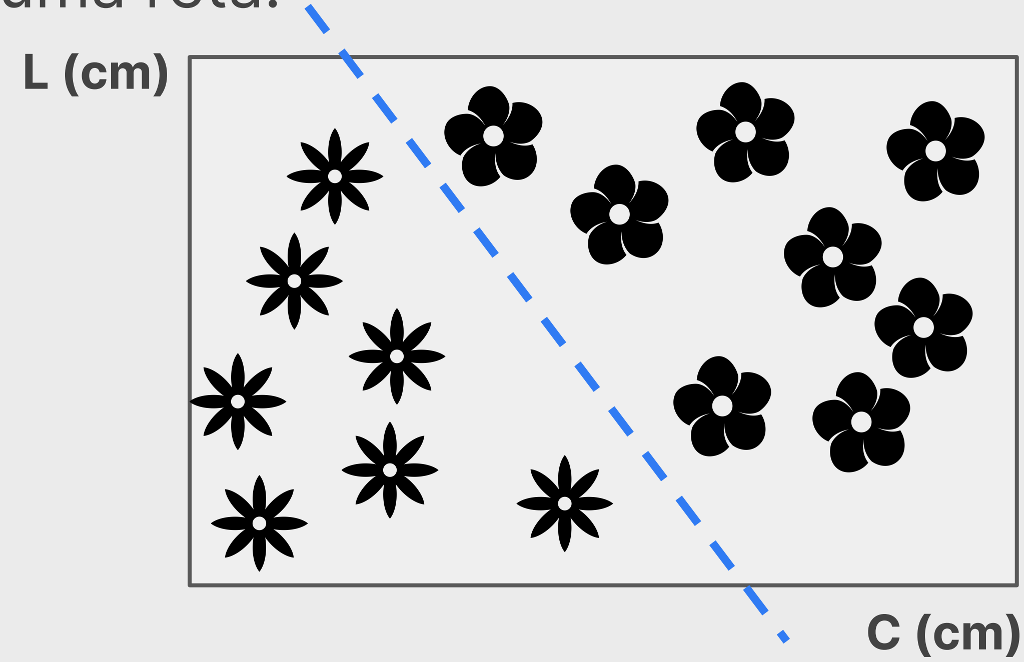
O problema da classificação

Como ensinar uma máquina a dizer se uma flor é do tipo 1 ( ) ou do tipo 2 ( )?

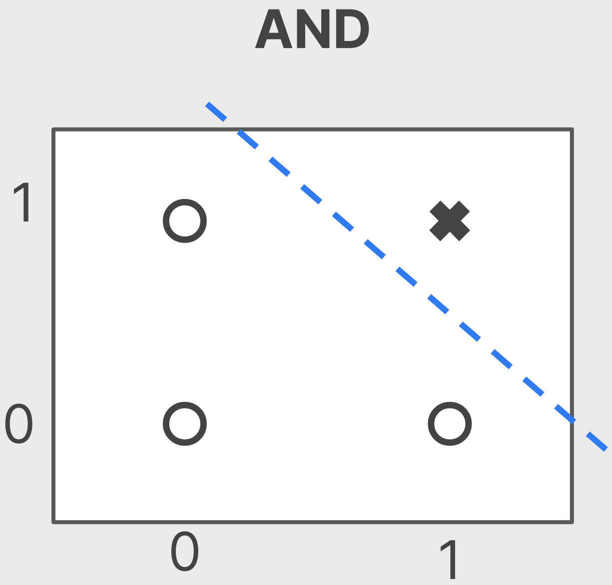


O problema da classificação

Se os dois tipos são linearmente separados, podemos traçar uma reta.



O problema da classificação



Retomando a História...

Frank Rosenblatt
Psicólogo

1957

Neurônio artificial

O **Perceptron** foi a arquitetura computacional ***melhorada*** que imitou o neurônio biológico e conseguia aprender

Retomando a História...

1957

Neurônio artificial

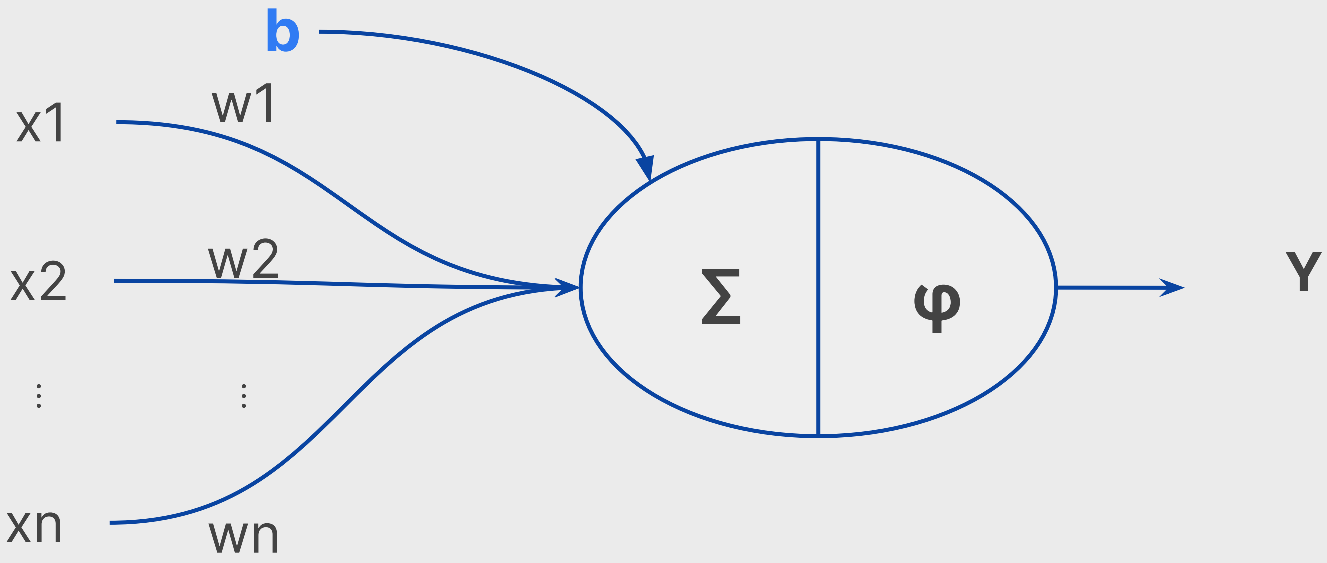
Frank Rosenblatt
Psicólogo

O **Perceptron** foi a arquitetura computacional ***melhorada*** que imitou o neurônio biológico e conseguia aprender

Considerado o **pai do Deep Learning**

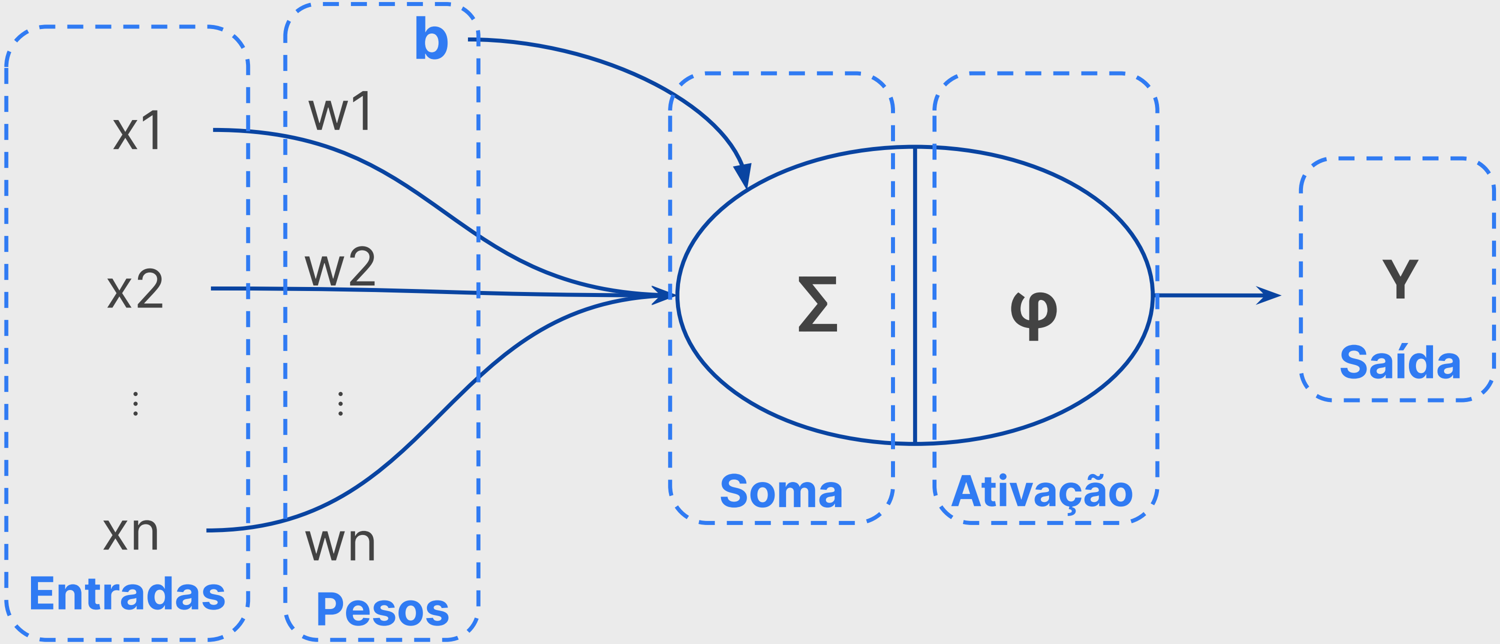
Perceptron

Modelo neural que associa pesos às entradas do modelo.



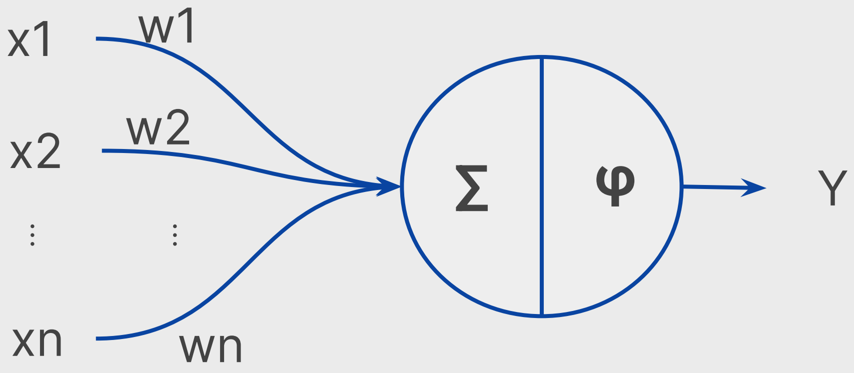
Perceptron

Modelo neural que associa pesos às entradas do modelo.



Perceptron

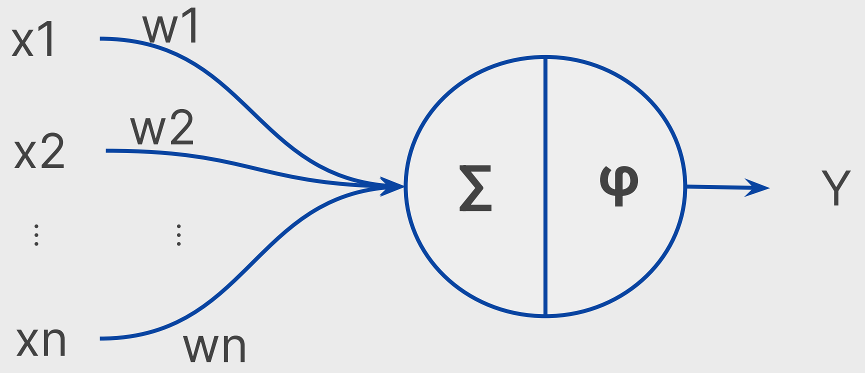
A função soma (Σ) no Perceptron segue um modelo matemático linear que se expressa da seguinte forma:



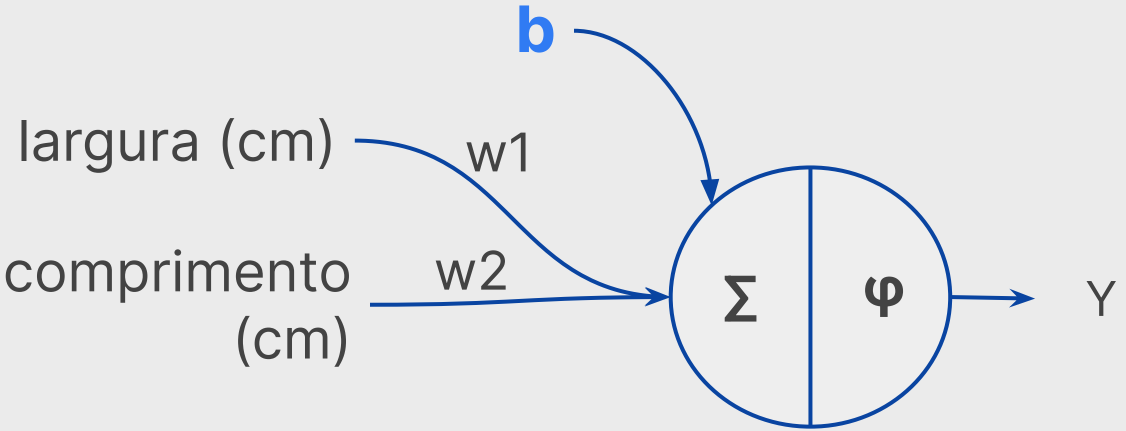
$$\sum = x_1 \times w_1 + x_2 \times w_2 + \cdots + x_n \times w_n$$

Perceptron

O resultado da função Σ é passado para a **função de ativação (φ)** que irá retornar um resultado correspondente a função utilizada.



Perceptron



$$y = -x \cdot \frac{w_1}{w2} - \frac{b}{w_2}$$

Perceptron

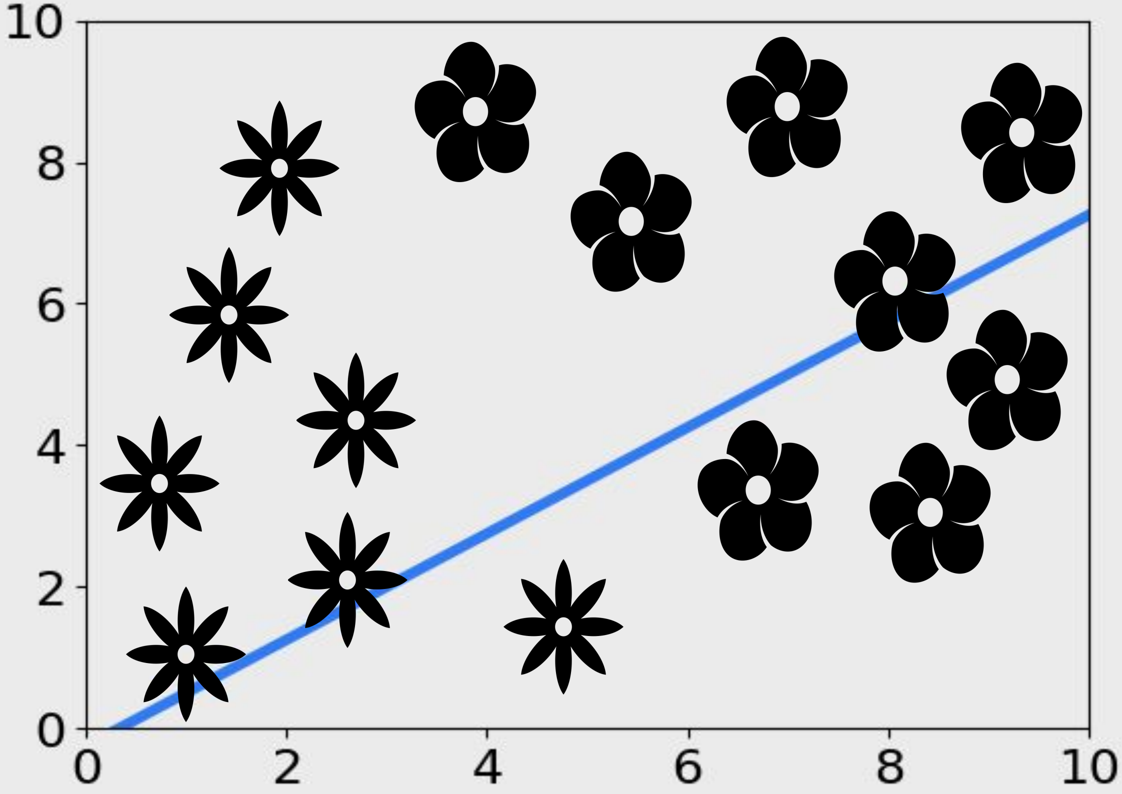
$$y = -x \cdot \frac{w_1}{w2} - \frac{b}{w_2}$$

w1 = - 3

w2 = 4

b = 1

Perceptron

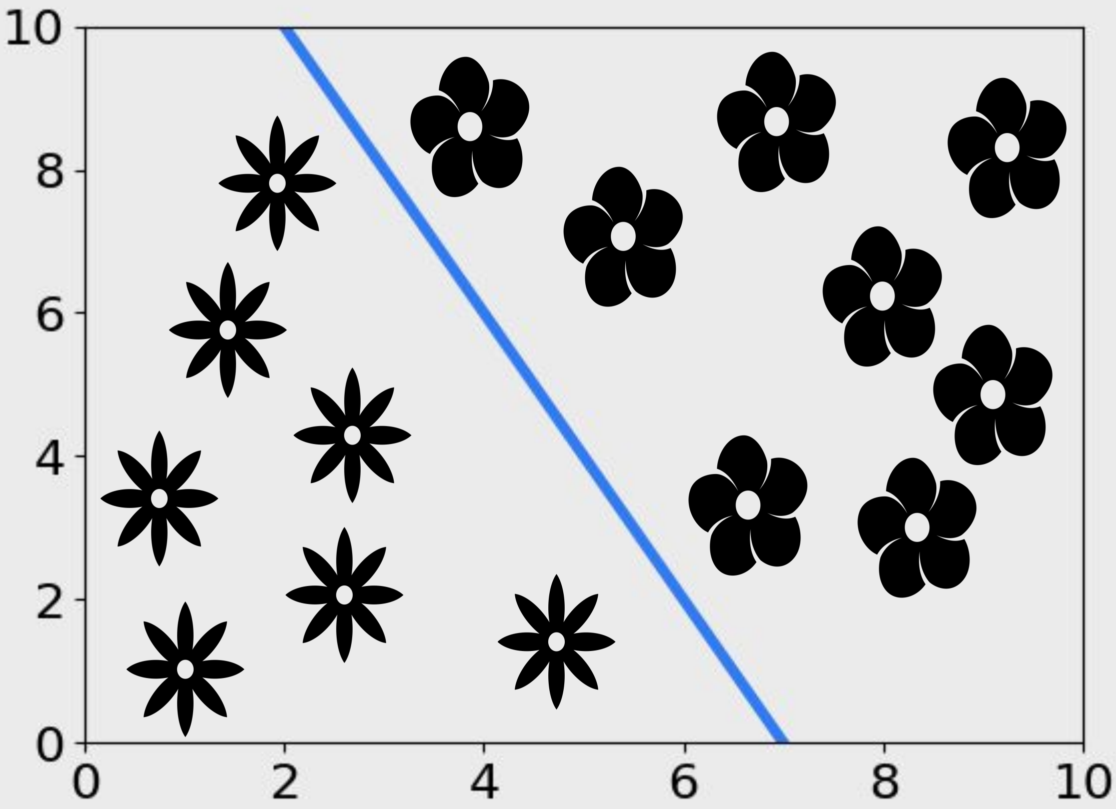


w1 = - 3

w2 = 4

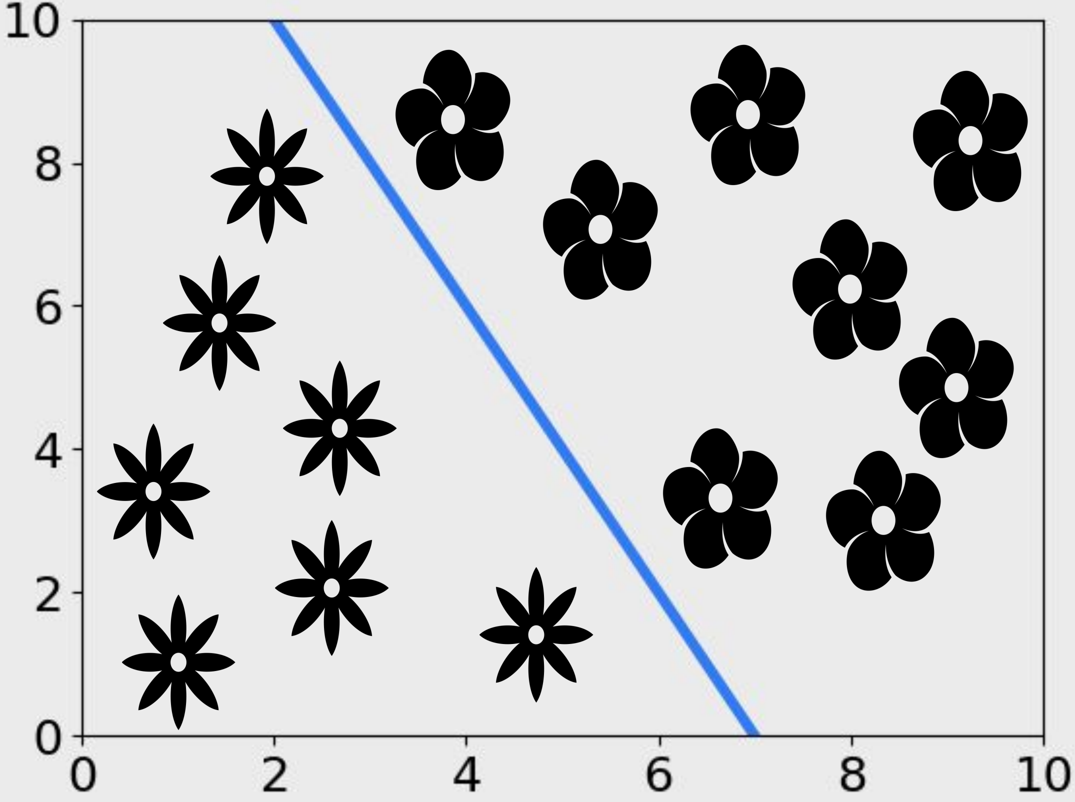
b = 1

Perceptron



w1 = 4
w2 = 2
b = -28

Perceptron



Retomando a História...

Marvin Minsky
Cientista da Computação

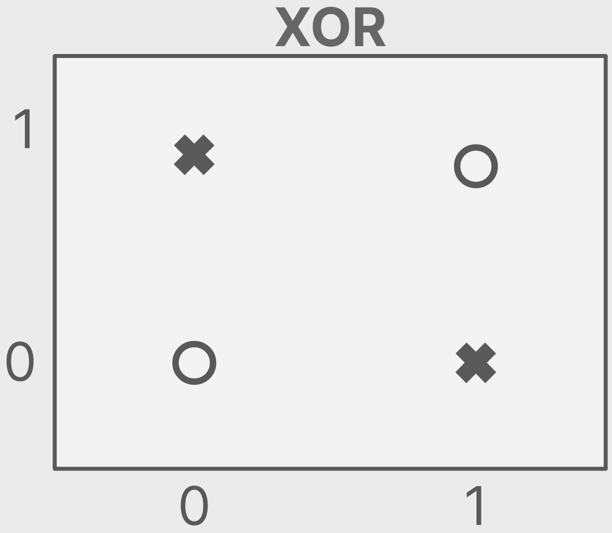
1969
Problema do XOR

O Perceptron foi provado como uma arquitetura computacional incapaz de solucionar problemas de separação de dados não-lineares ou que demandam de mais uma reta de separação.

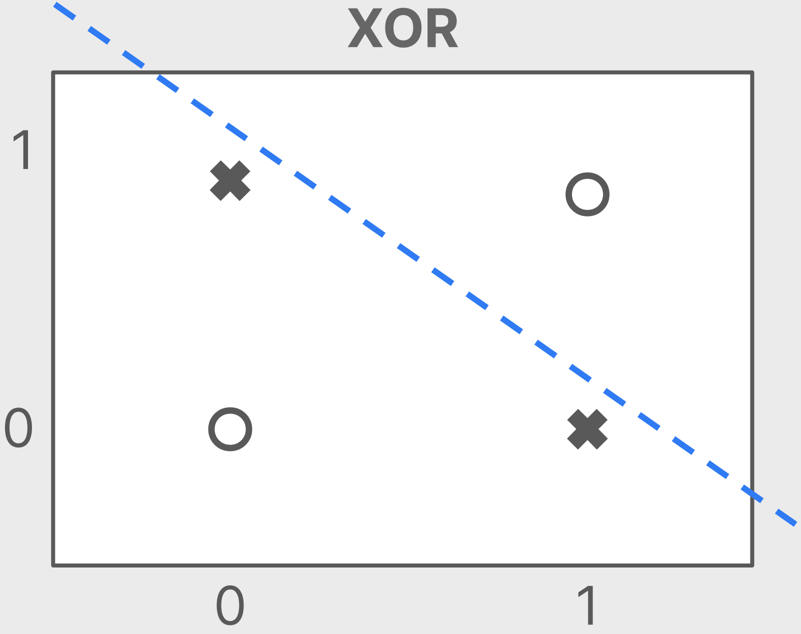
Seymour A Papert
Cientista da Computação

Problema XOR

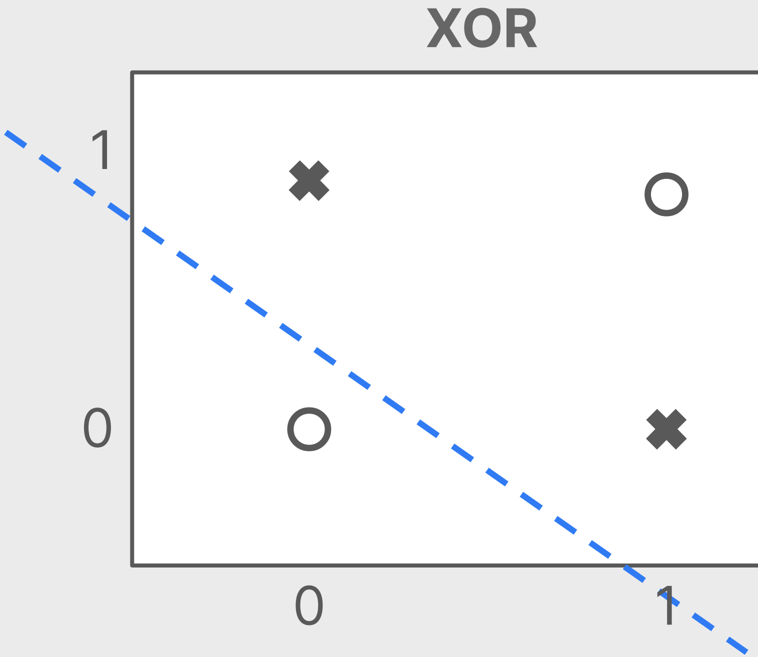
X1	X2	Y
0	0	○
0	1	✖
1	0	✖
1	1	○



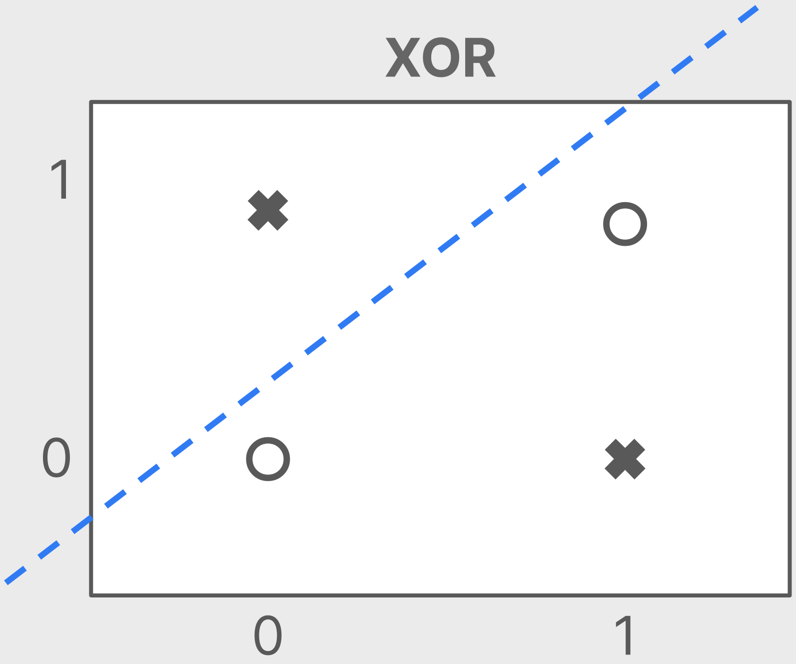
Problema XOR



Problema XOR

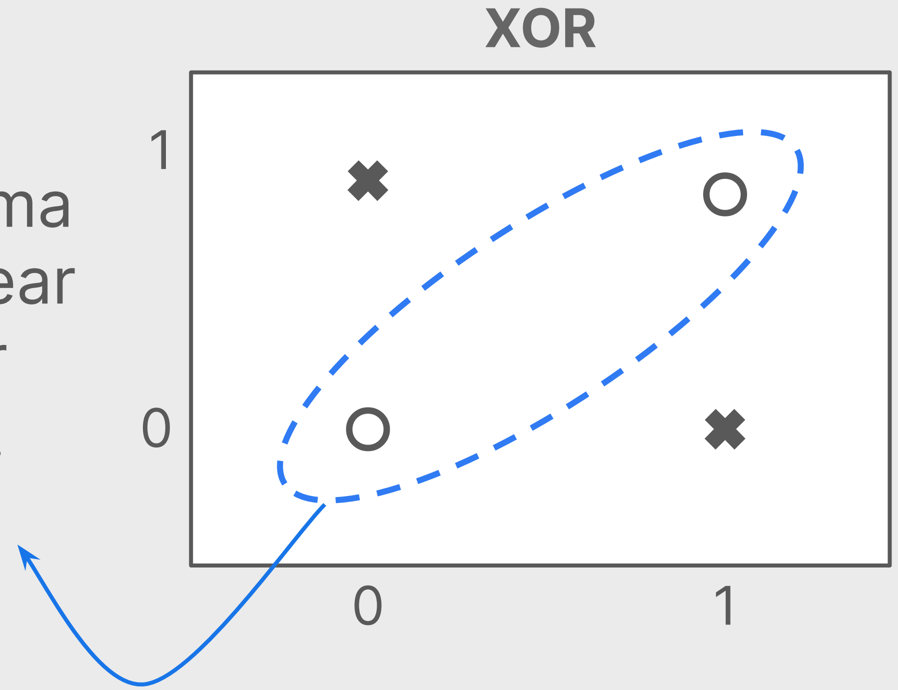


Problema XOR



Problema XOR

É necessária uma função não-linear para solucionar esse problema.



Retomando a História...

1969 — Inverno da IA — 1986

Retomando a História...

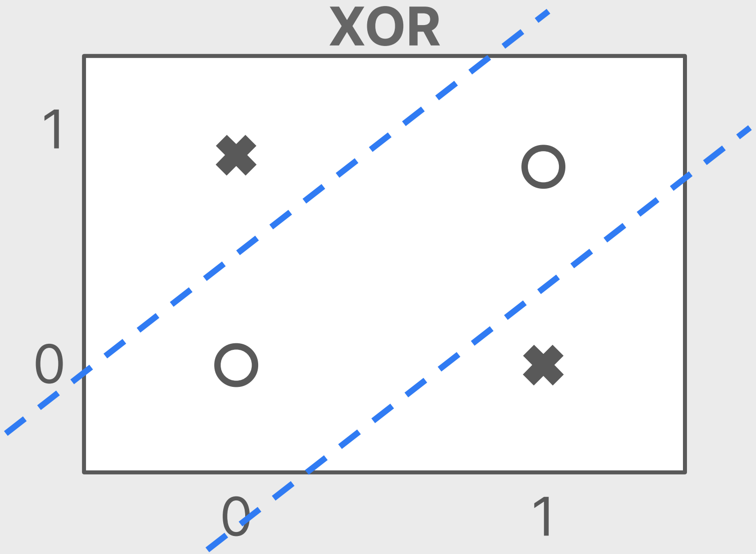
1986

Multilayer Perceptron

Geoffrey Hinton
Cientista da Computação

É proposto um novo método de aprendizado, com a adição de novas camadas de neurônios e a retropropagação.

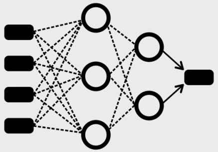
Retomando a História...



1986

Multilayer Perceptron

É proposto um novo método de aprendizado, com a adição de novas camadas de neurônios e a retropropagação.

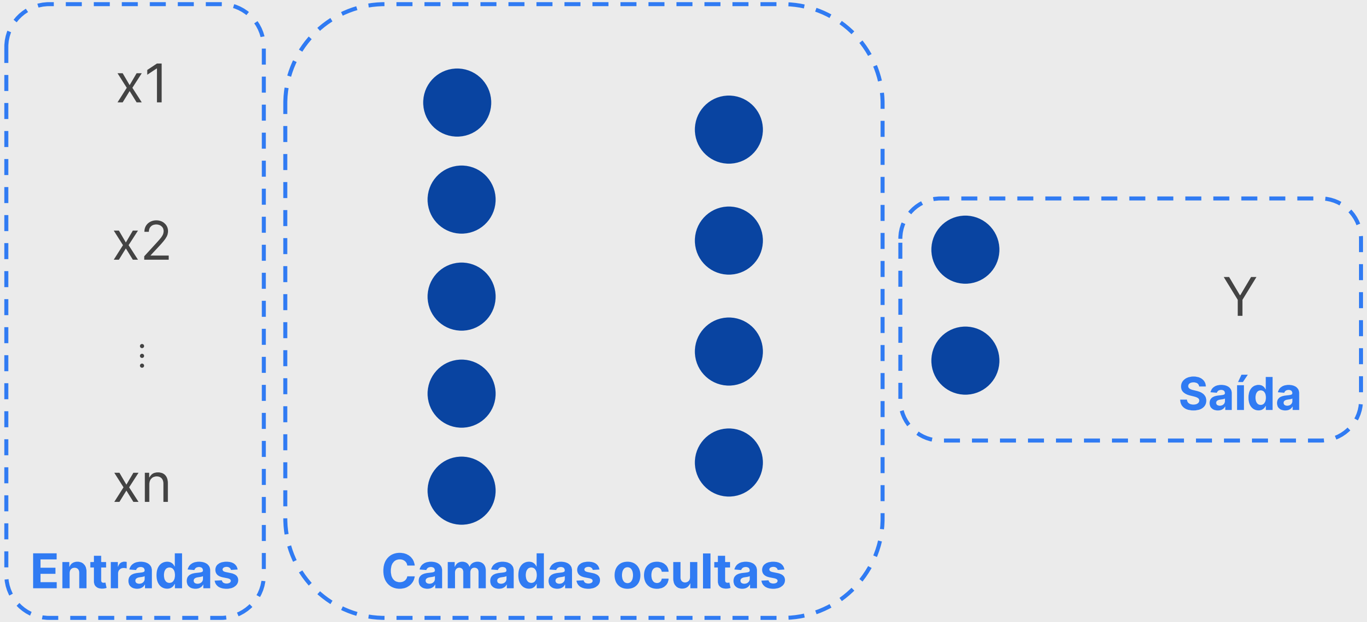


MLP

Múltiplas camadas de Perceptron

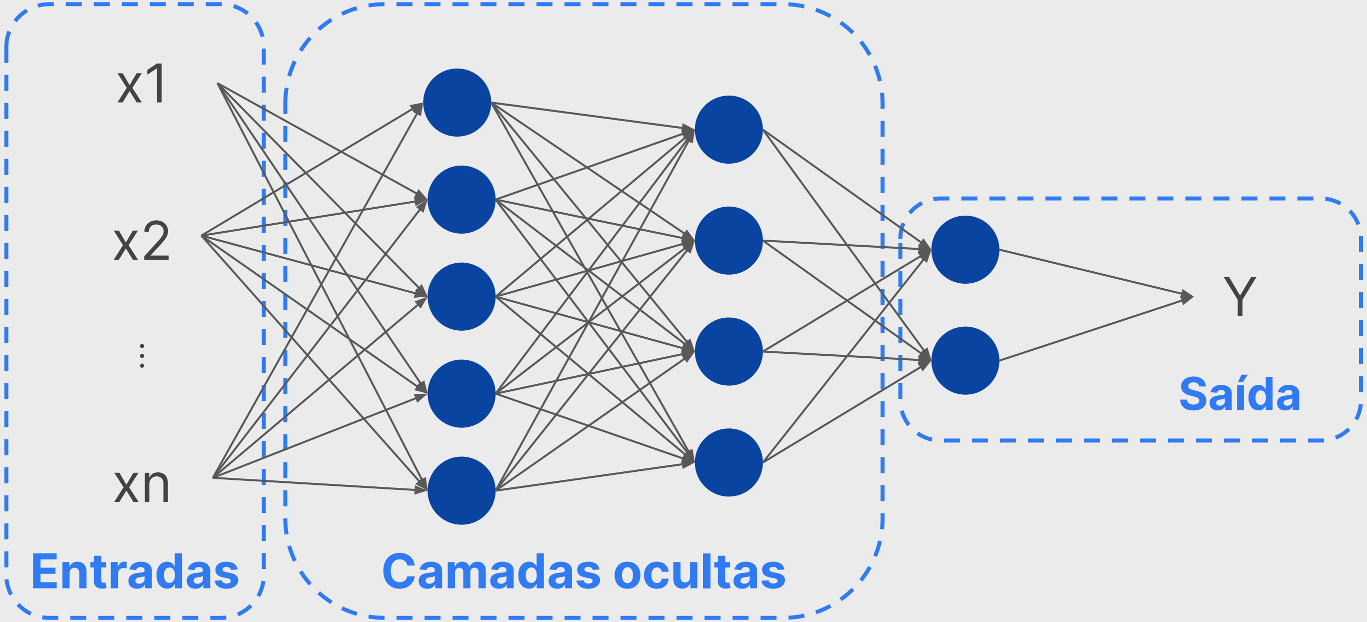
MLP

Múltiplas camadas de Perceptron



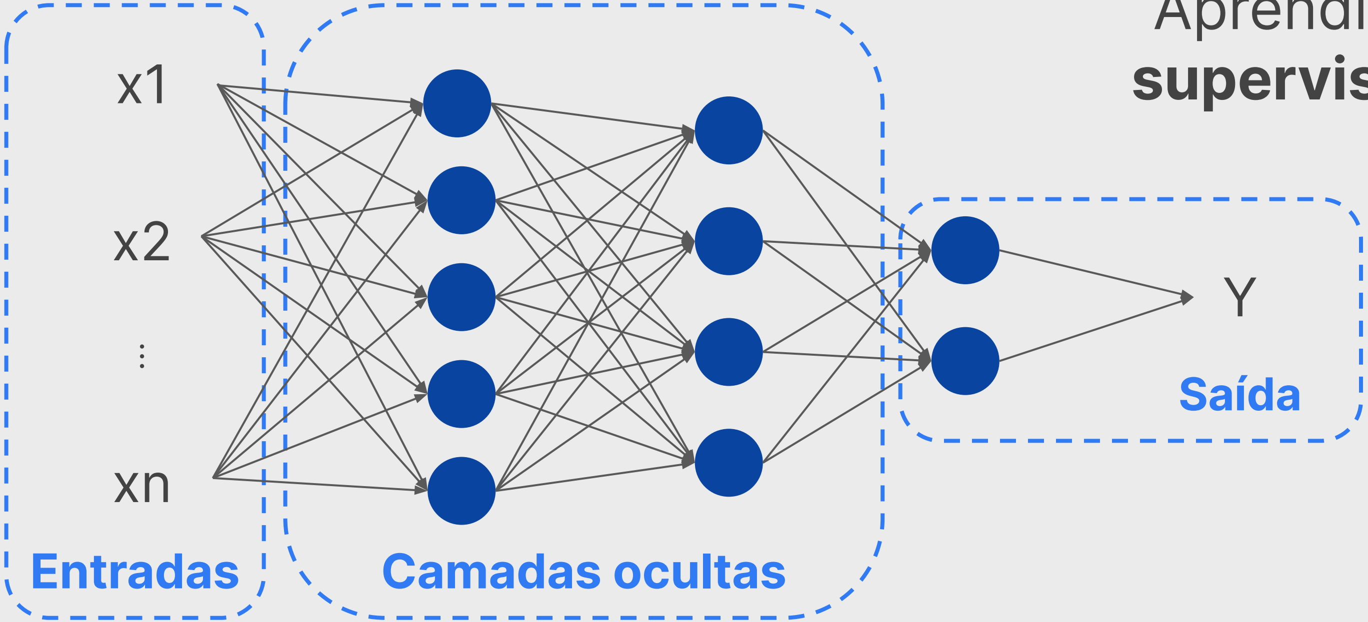
MLP

Caminho total do dado



MLP

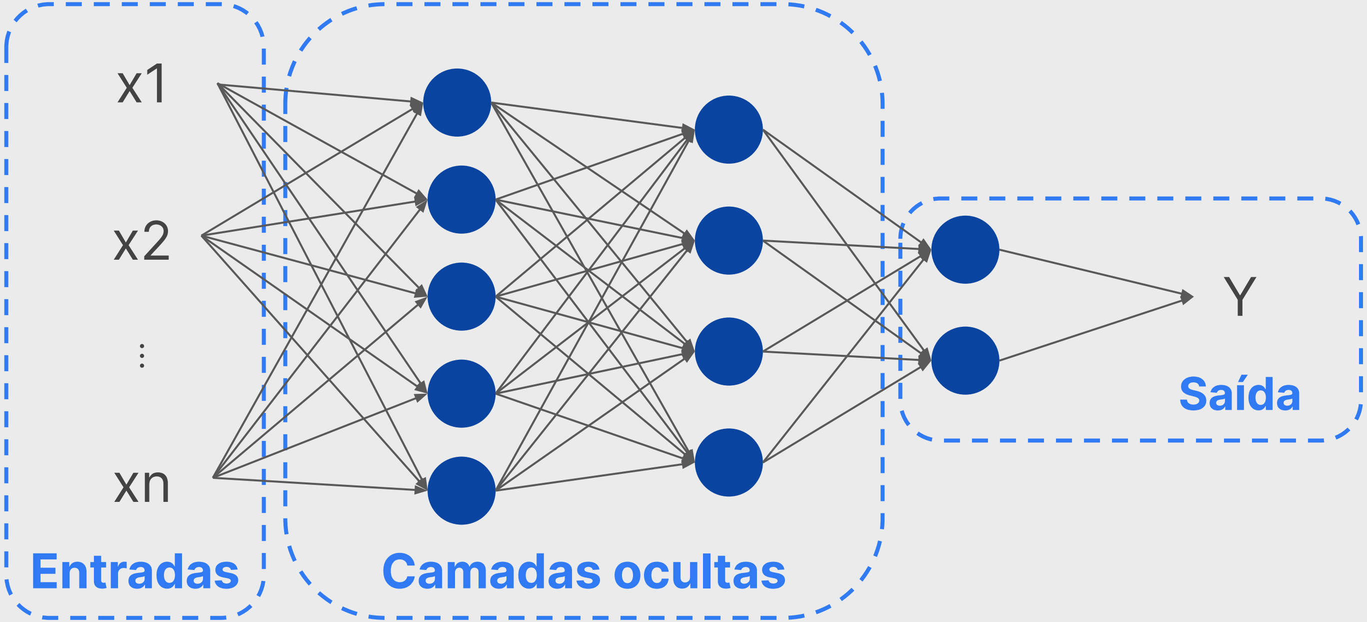
Caminho total do dado



Aprendizado é **supervisionado**

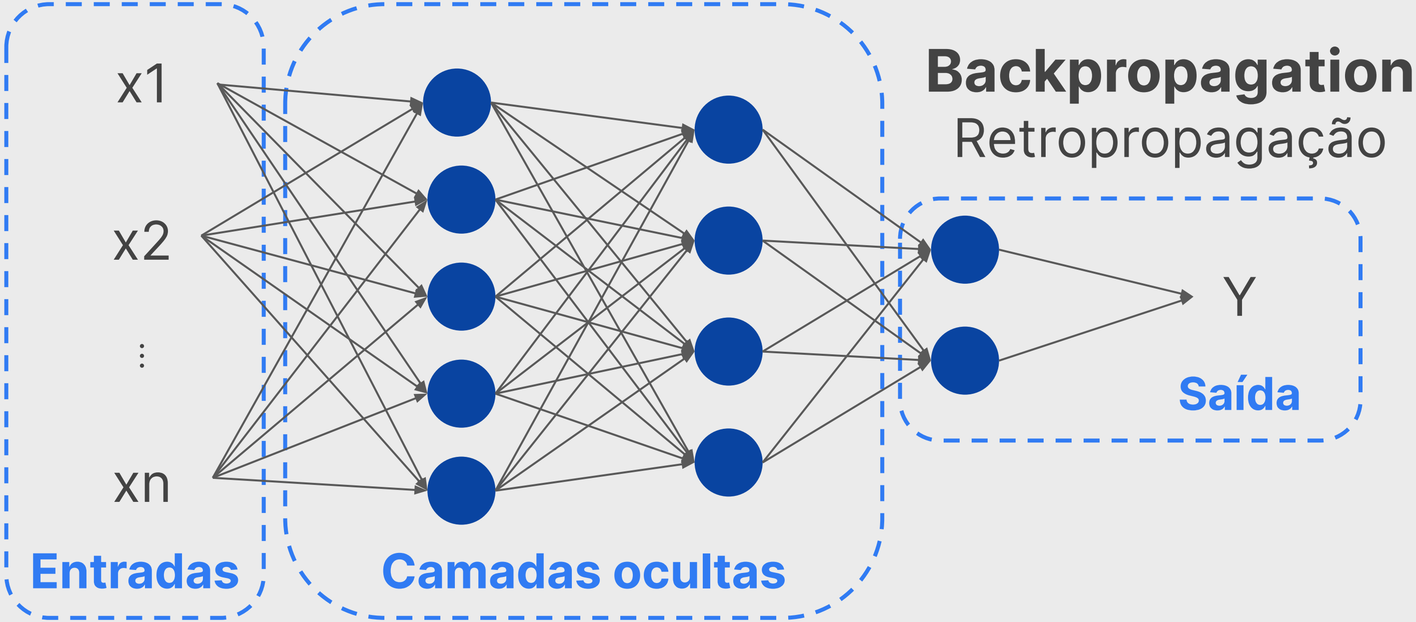
MLP

O aprendizado se inicia pela estrutura *feedfoward*, **mas não para nisso**!

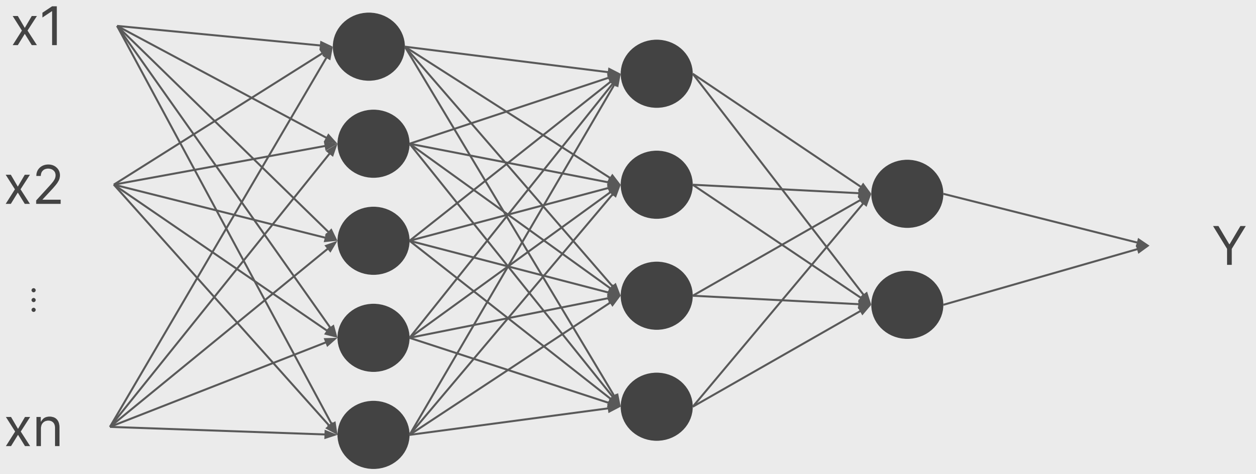


MLP

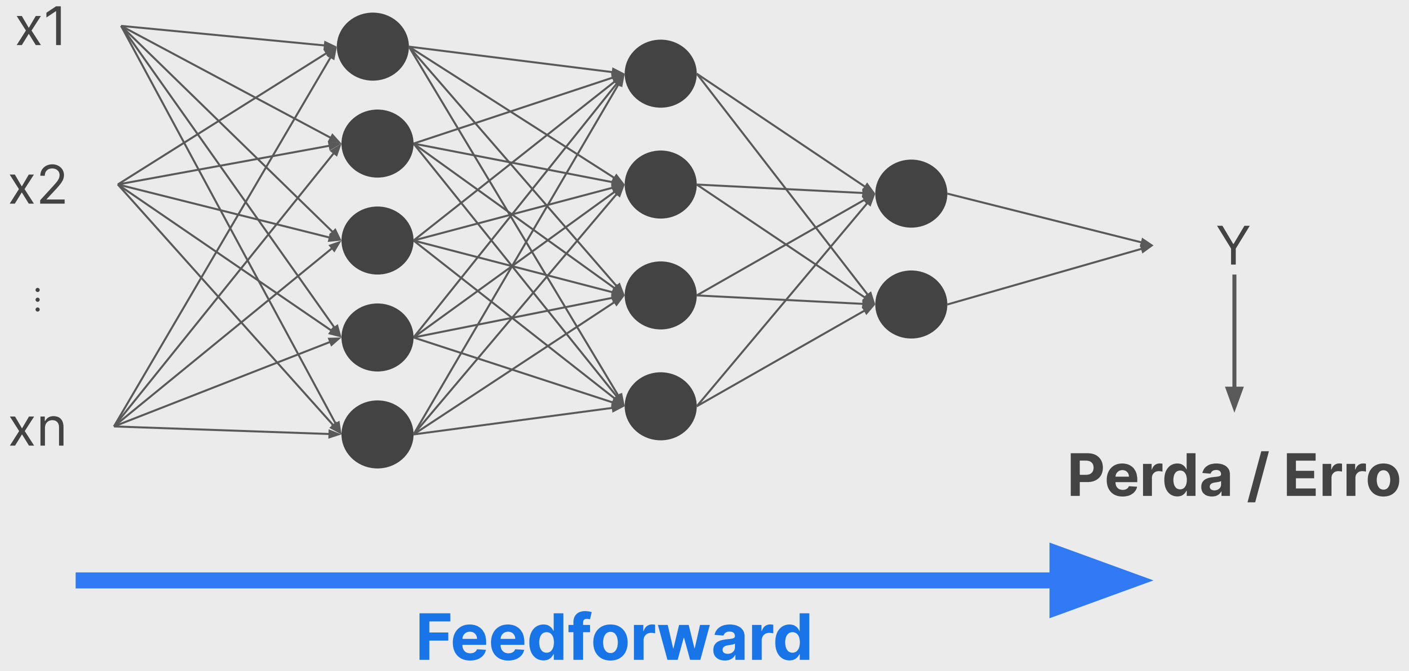
O aprendizado se inicia pela estrutura *feedfoward*, **mas não para nisso**!



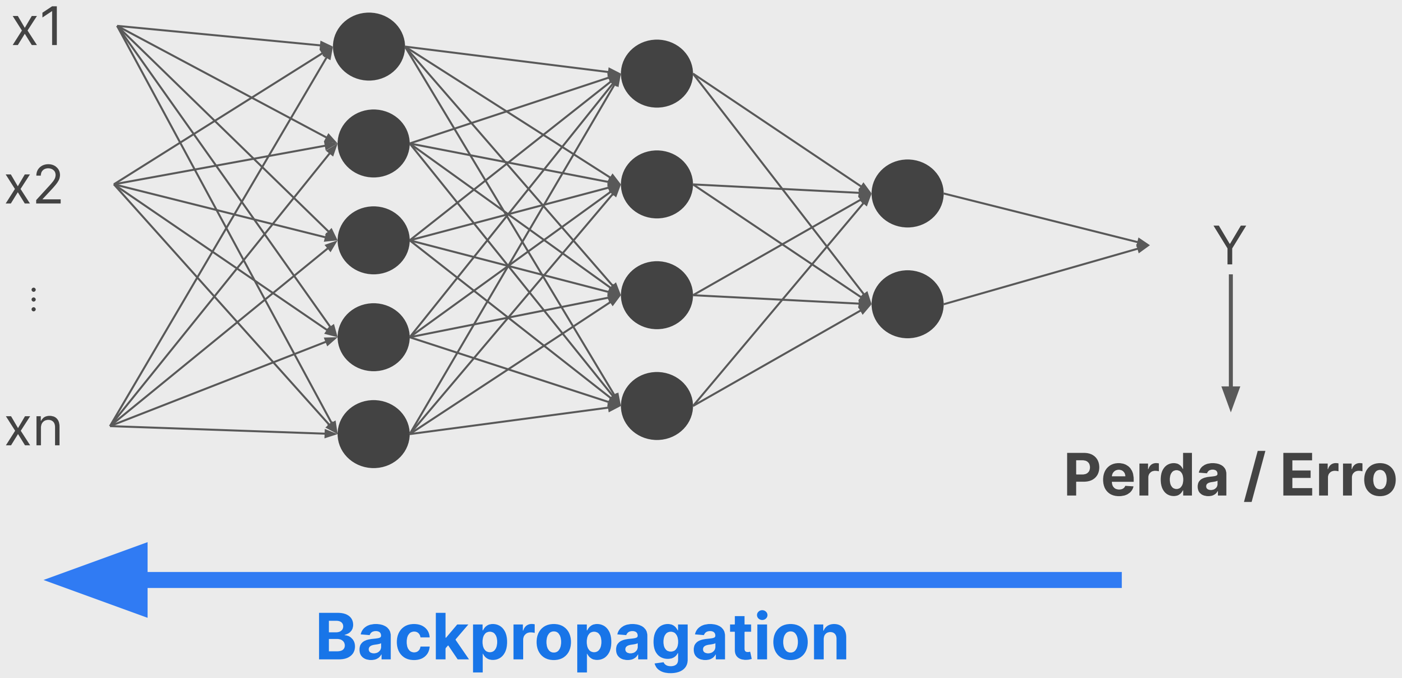
MLP - aprendizado



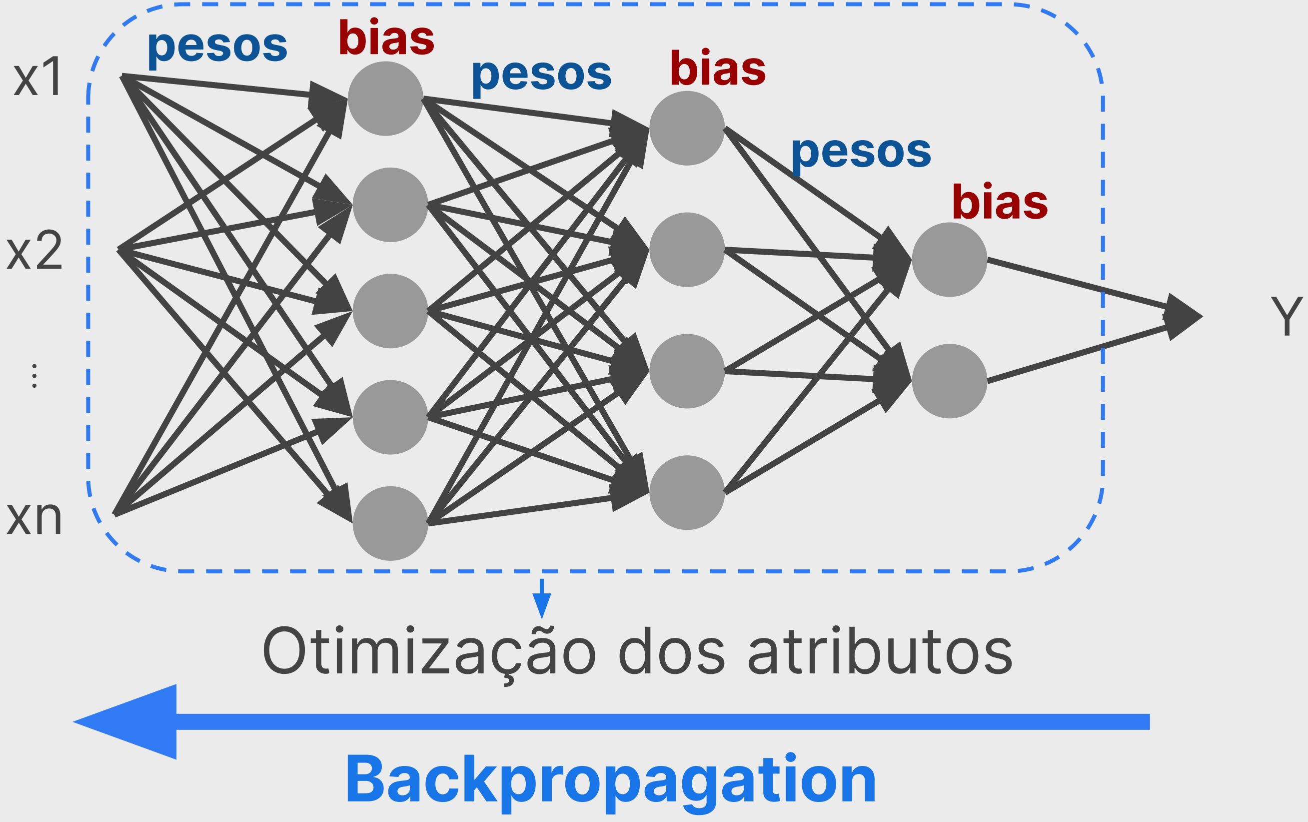
MLP - aprendizado



MLP - aprendizado



MLP - aprendizado



MLP - aprendizado

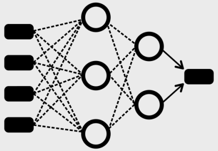
Conjunto de entradas

1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

Aprendizado

Feedforward

Backpropagation



MLP - aprendizado

Conjunto de entradas

1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

Aprendizado

Feedforward

Backpropagation

MLP - aprendizado

Conjunto de entradas

1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

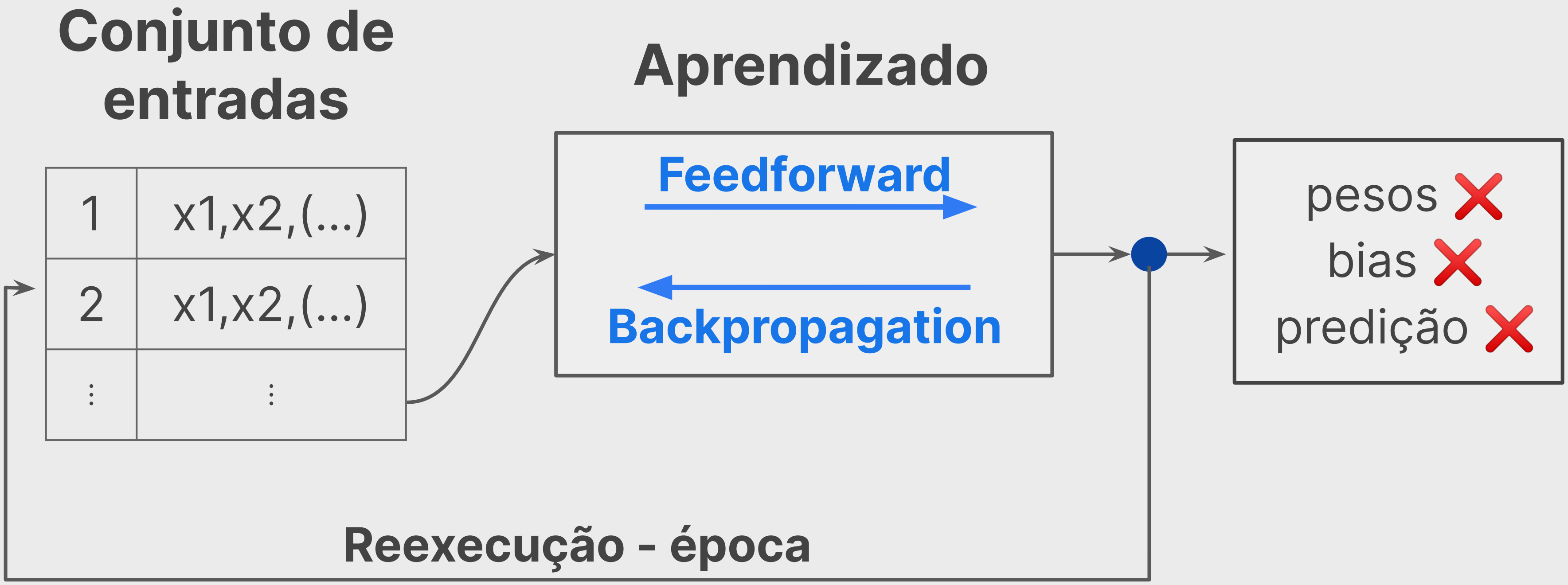
Aprendizado

Feedforward

Backpropagation

pesos ❌
bias ❌
predição ❌

MLP - aprendizado



MLP - aprendizado

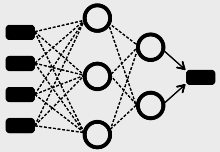
Conjunto de entradas

1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

Aprendizado

Feedforward →

← **Backpropagation**



MLP - aprendizado

Conjunto de entradas

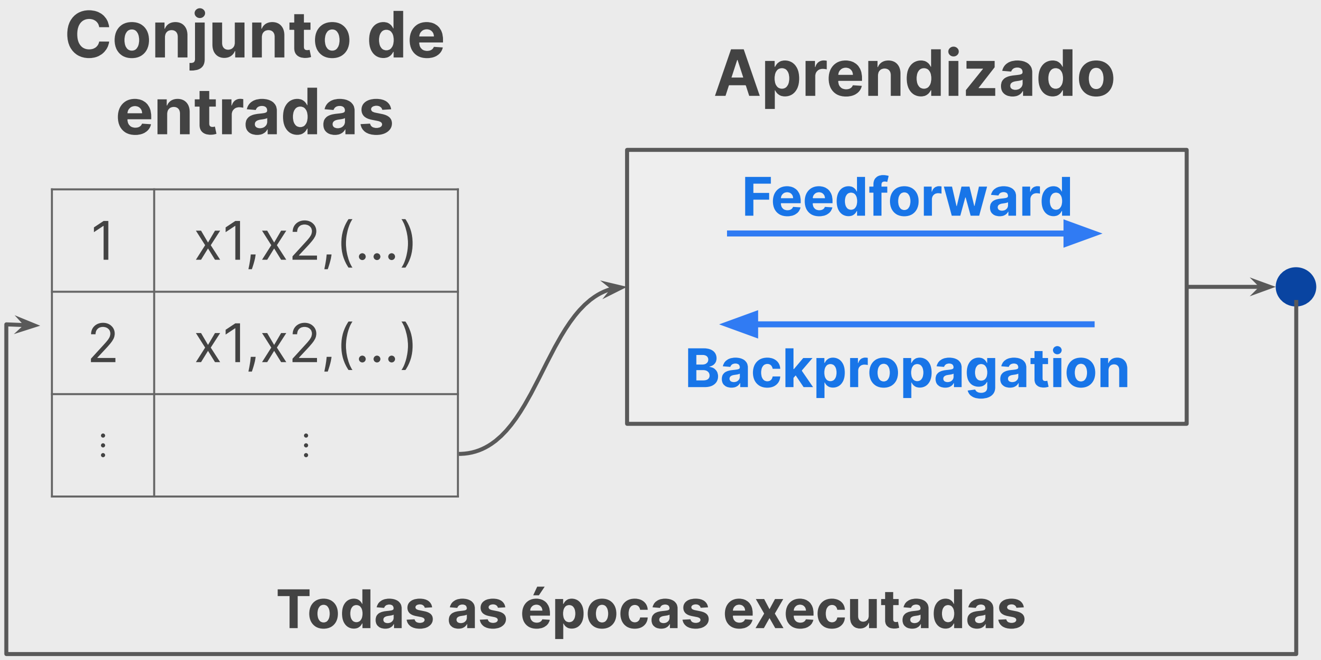
1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

Aprendizado

Feedforward

Backpropagation

MLP - aprendizado



MLP - aprendizado

Conjunto de entradas

1	x1,x2,(...)
2	x1,x2,(...)
⋮	⋮

Aprendizado

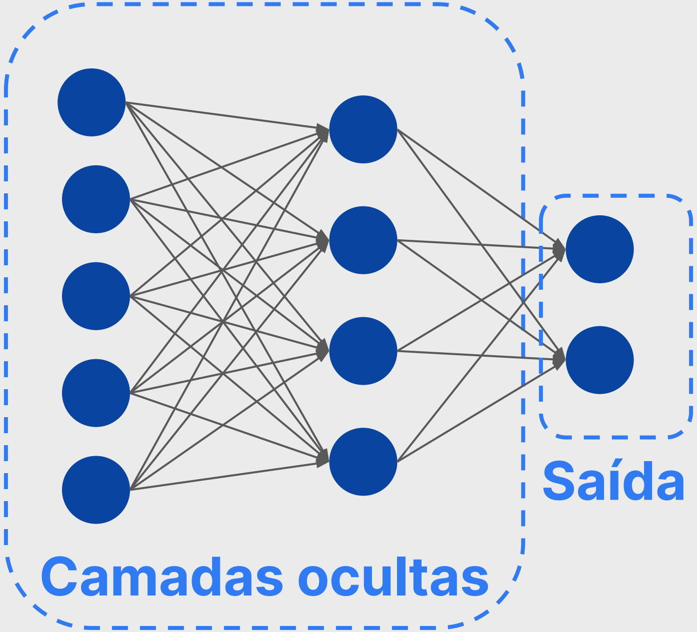
Feedforward →

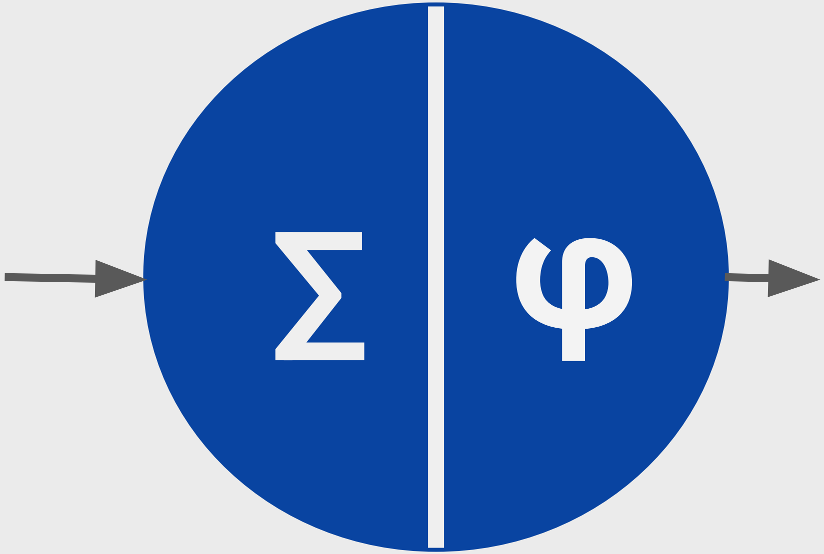
← **Backpropagation**

Modelo treinado

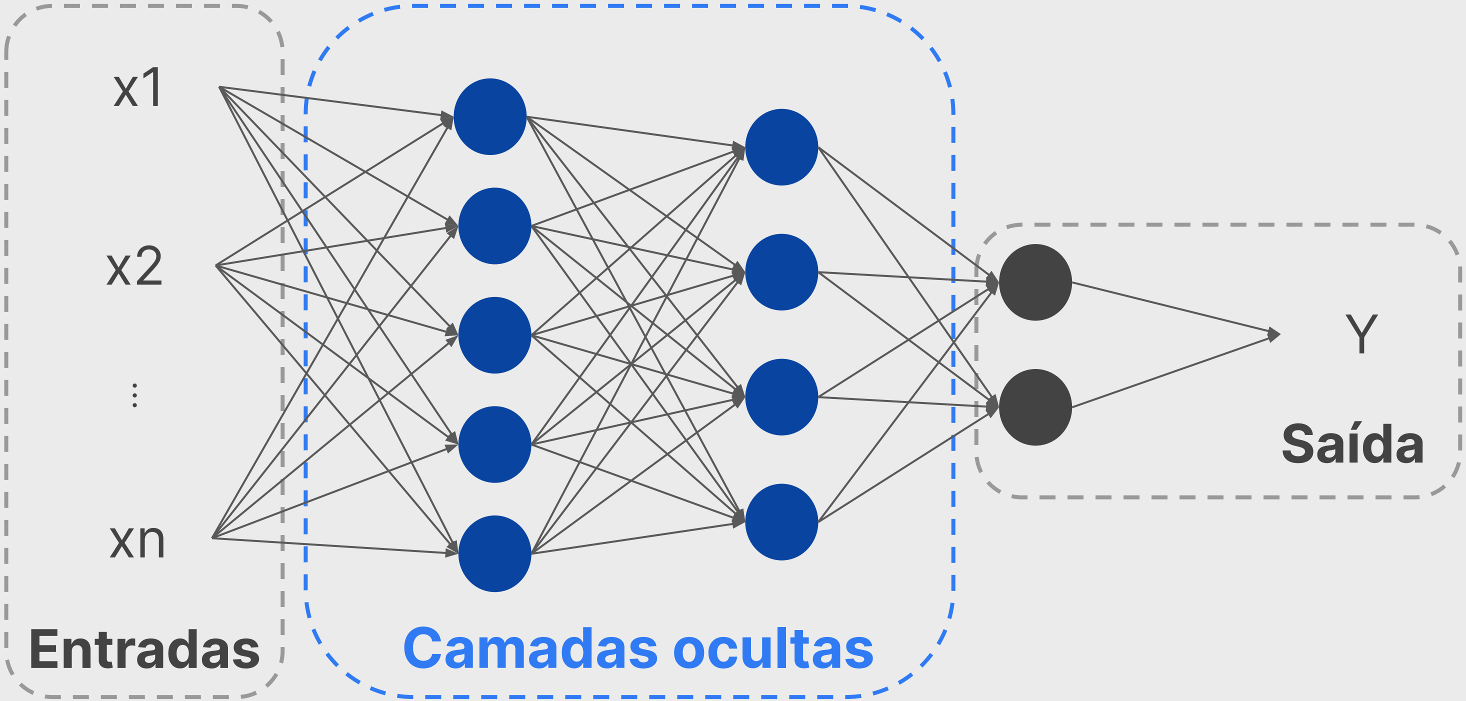
pesos ✅
bias ✅
predição ✅

Camadas

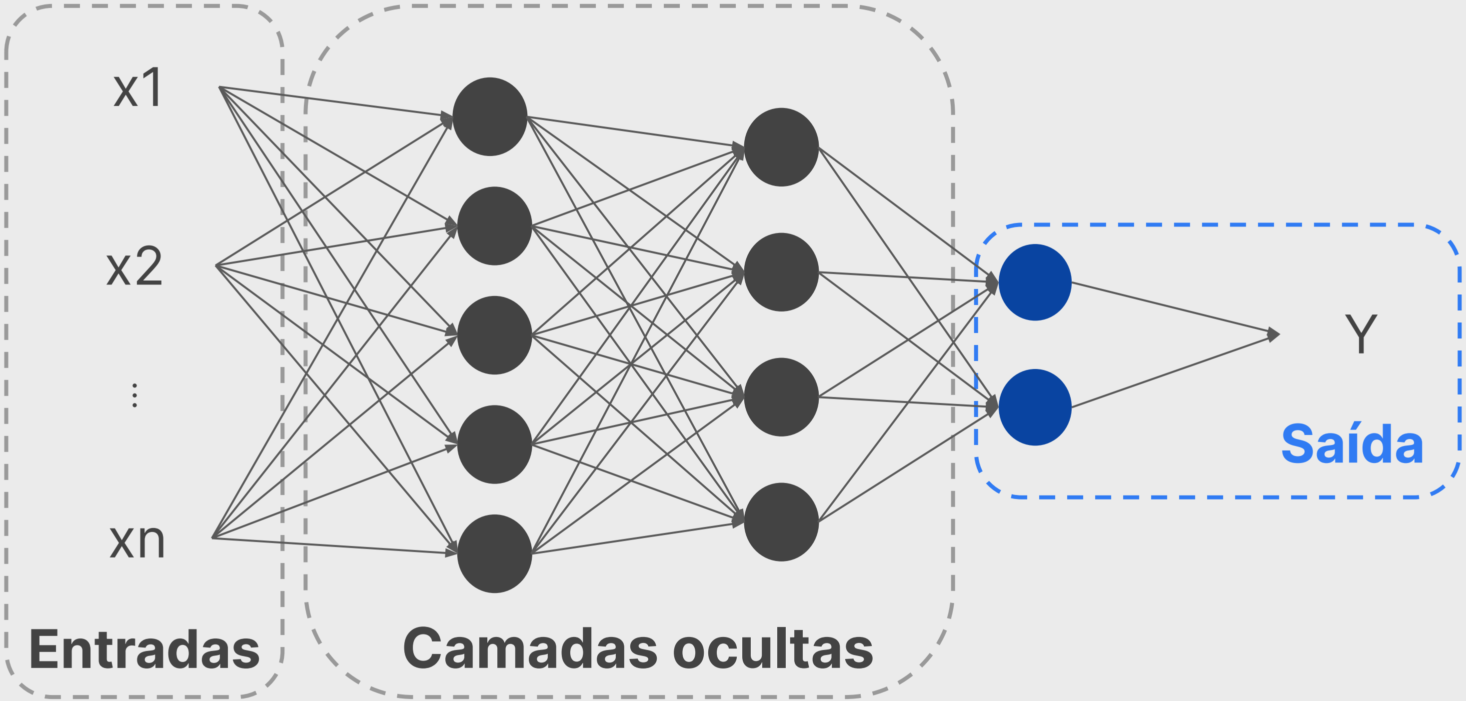




φ - Camada oculta



φ - Camada saída



Funções φ(x)

As funções de ativação mais comuns e utilizadas são:

Funções φ(x)

As funções de ativação mais comuns e utilizadas são:

Sigmoid - sigmóide / logística

Funções $\varphi(x)$

As funções de ativação mais comuns e utilizadas são:

Sigmoid - sigmóide / logística

Tanh - tangente hiperbólica

Funções $\varphi(x)$

As funções de ativação mais comuns e utilizadas são:

- **Sigmoid** - sigmóide / logística
- **Tanh** - tangente hiperbólica
- **ReLU** - unidade linear retificada

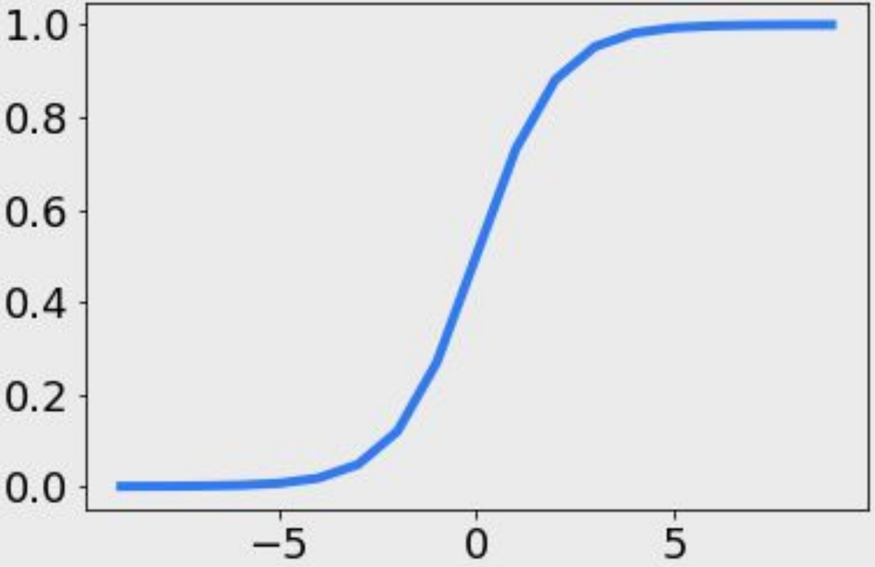
Funções φ(x)

As funções de ativação mais comuns e utilizadas são:

- **Sigmoid** - sigmóide / logística
- **Tanh** - tangente hiperbólica
- **ReLU** - unidade linear retificada
- **Softmax**

Funções φ(x)

Sigmoid

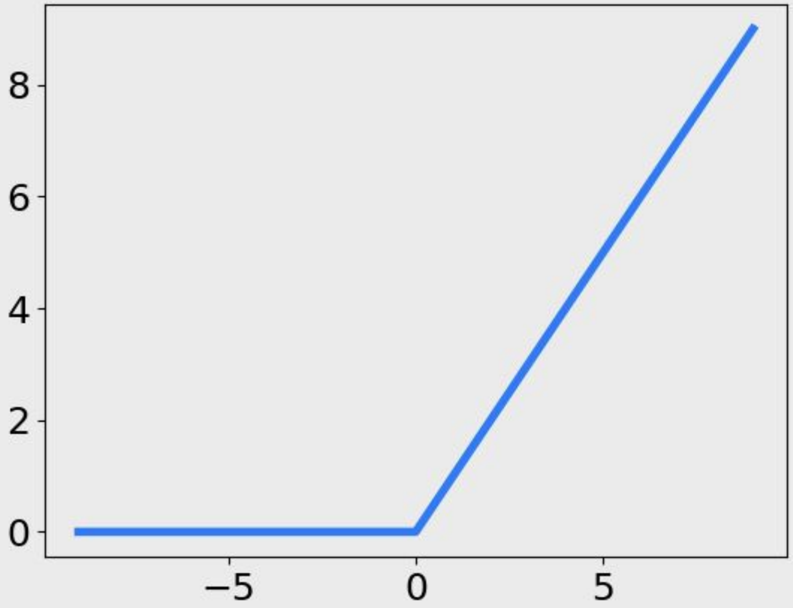


$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

Utilizada na camada de **saída** ou na **oculta**

Funções φ(x)

ReLU



$$\varphi(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Utilizada na camada **oculta**

Funções φ(x)

Softmax

$$\varphi_i = \left[\frac{exp(x_i \cdot w_i)}{\sum_{n=1}^{j} exp(x_n \cdot w_n)} \right]$$

Utilizada na camada de **saída**