

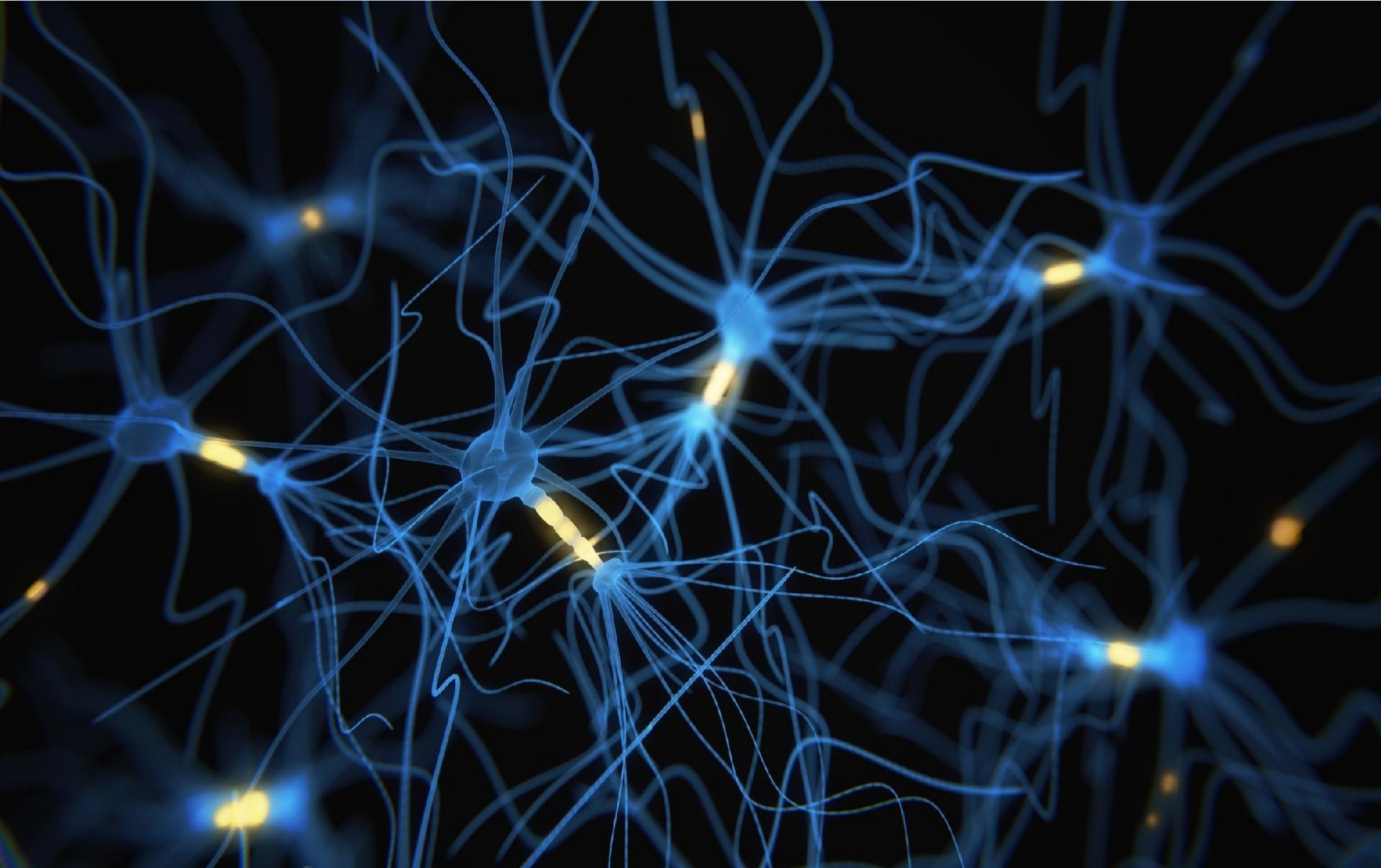


**Universidade Federal do Piauí**  
Laboratório de Inteligência Artificial - LINA

# Introdução à Deep Learning

**Bruno Vicente Alves de Lima**

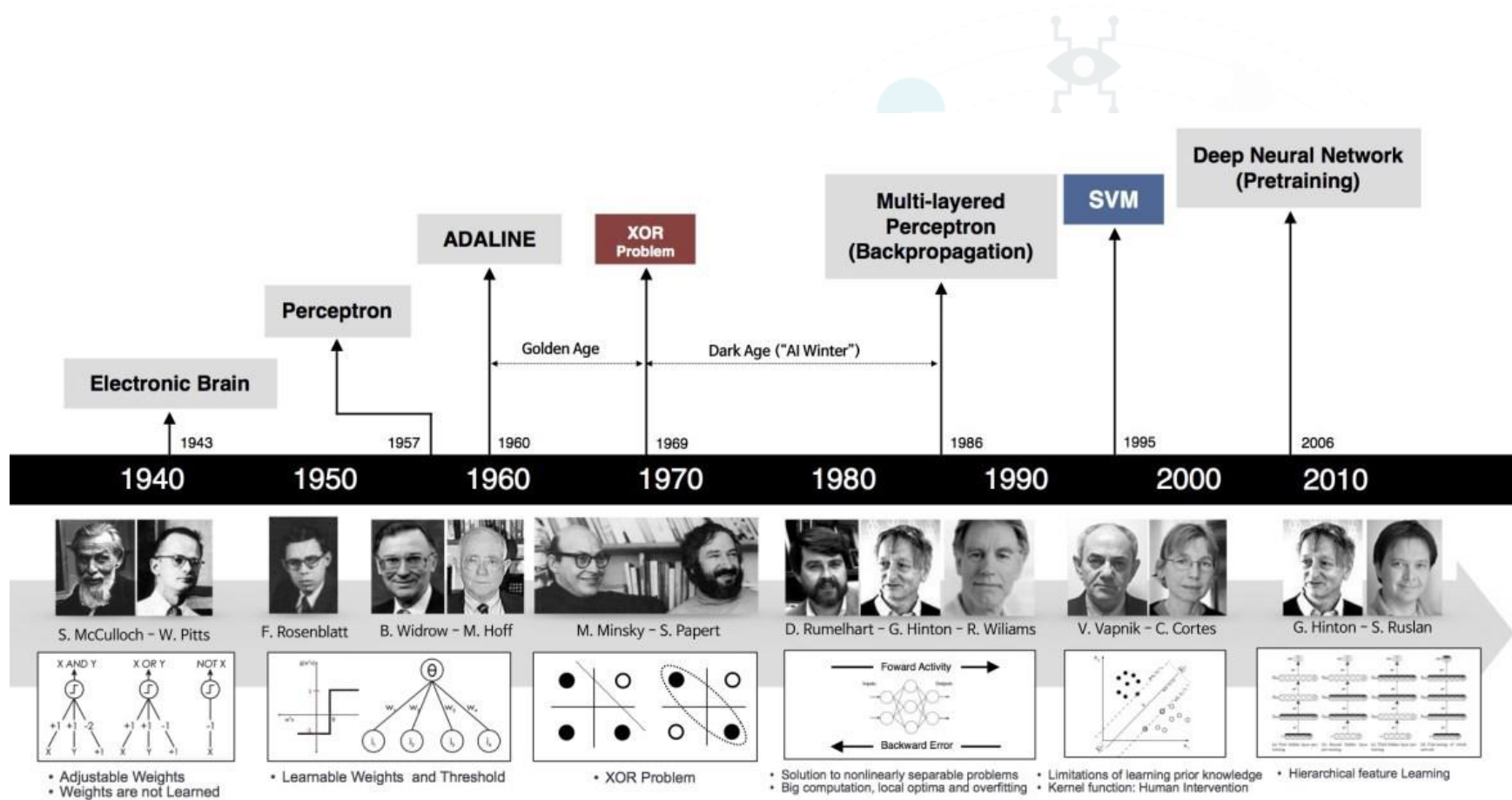
# Introdução à Redes Neurais Artificiais



# Introdução à Redes Neurais Artificiais

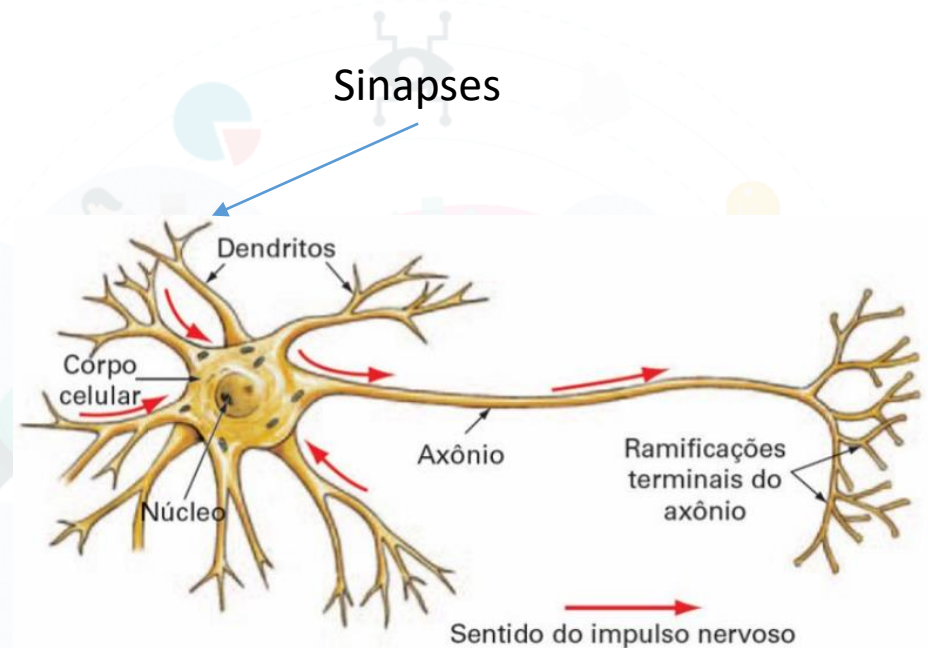
Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência.

# Contexto Histórico



# Neurônio Biológico

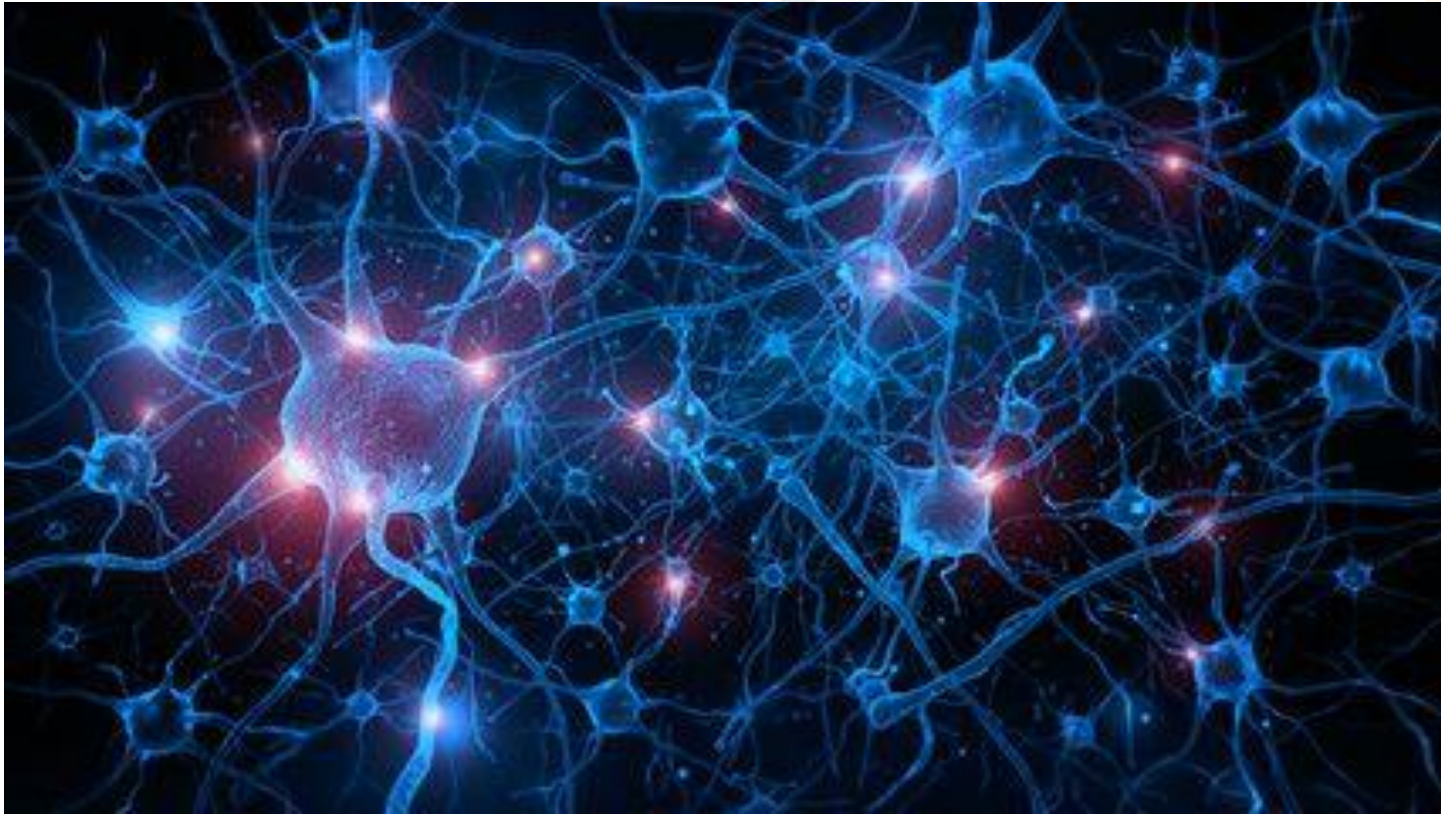
- Compõe o cérebro para processar informações;
- **Dendritos** – Recebem substâncias químicas que controlam o potencial elétrico na célula;
- **Axônios** – Transmite o impulso elétrico de um neurônio para o outro;
- **Sinapses** – Ponto de conexão entre um neurônio e outro. Regula a intensidade do impulso elétrico.





# Neurônio Biológico

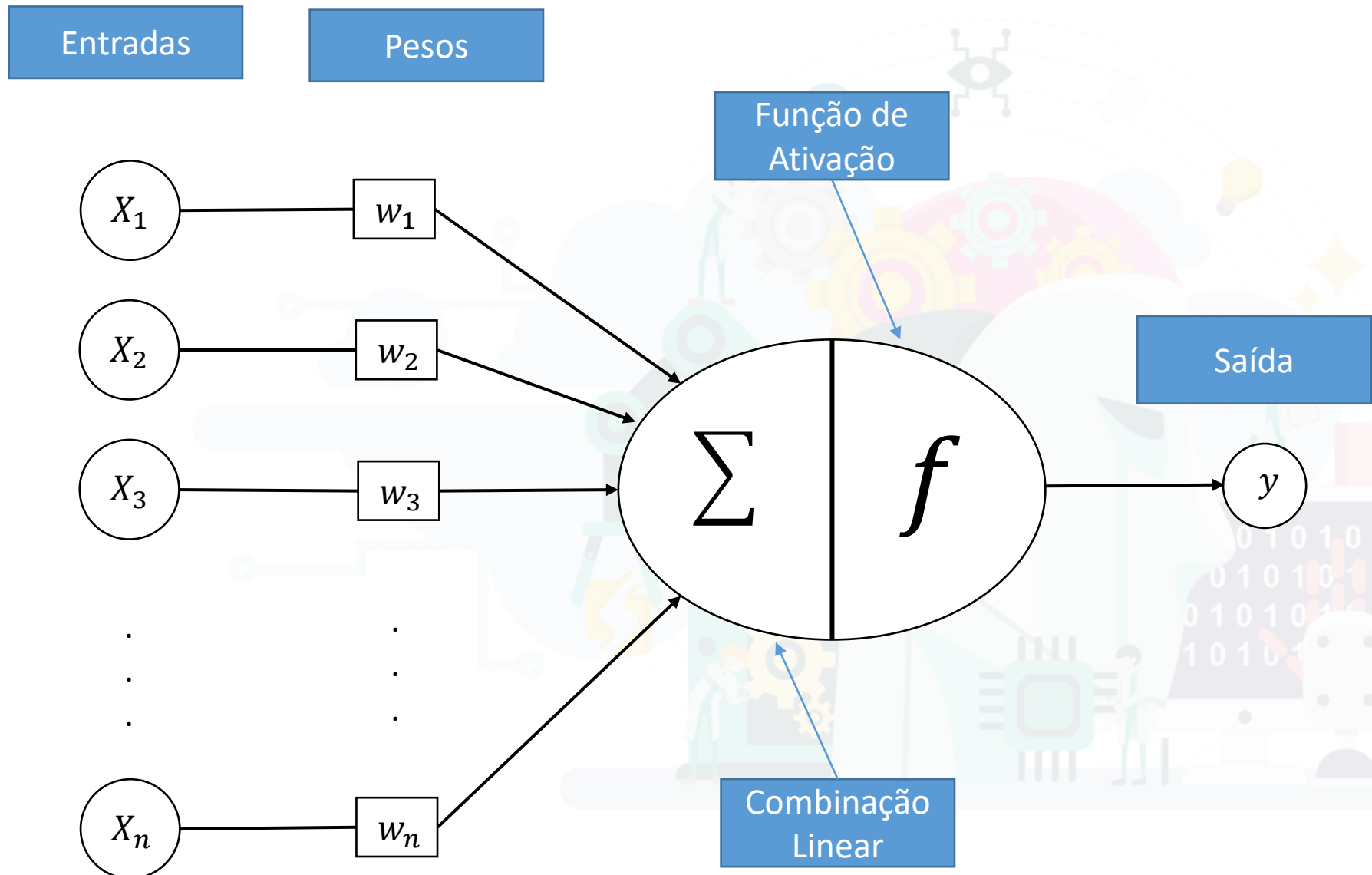
## Rede Neural Biológica



# Neurônio Artificial - Perceptron

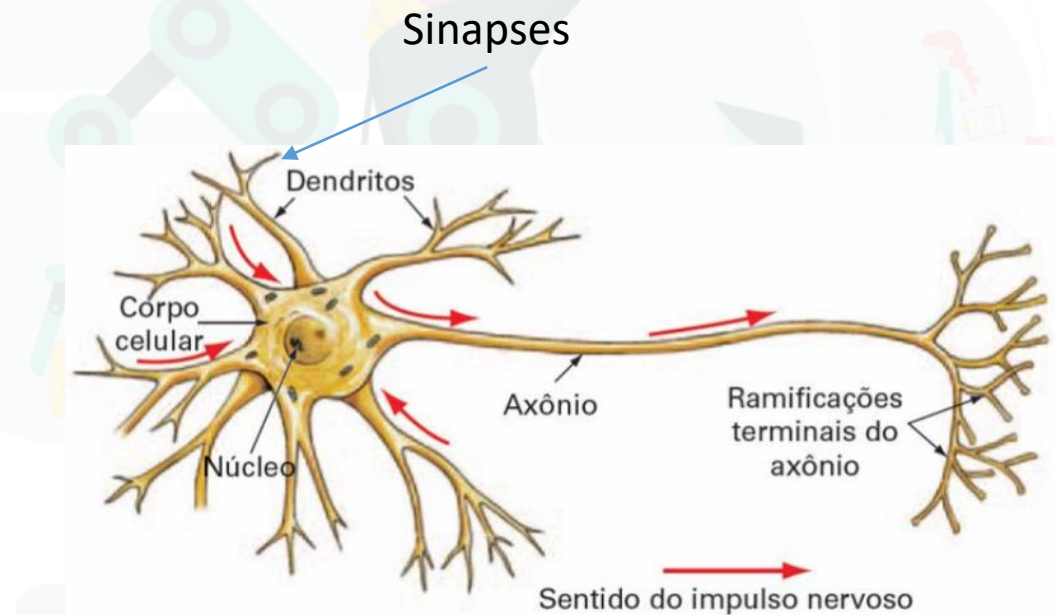
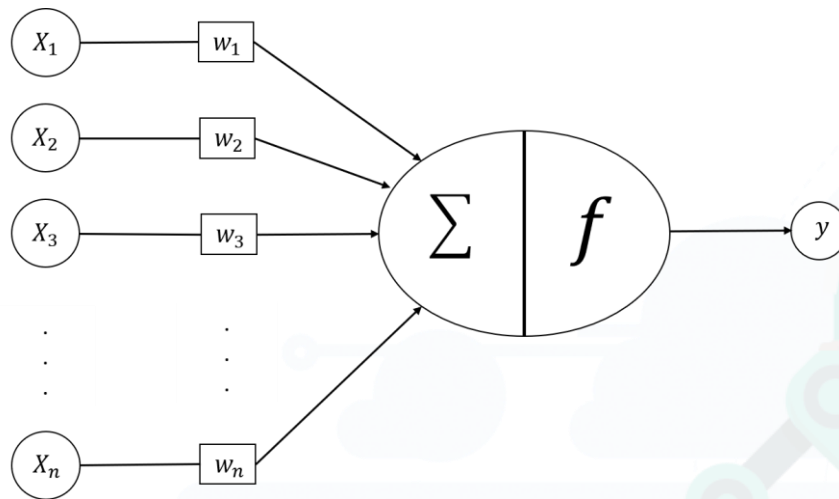
- Desenvolvido inicialmente por McCulloch e Pitts em 1943;
- Frank Rosenblatt criou Perceptron em 1958;
- Base das redes neurais artificiais.

# Neurônio Artificial - Perceptron

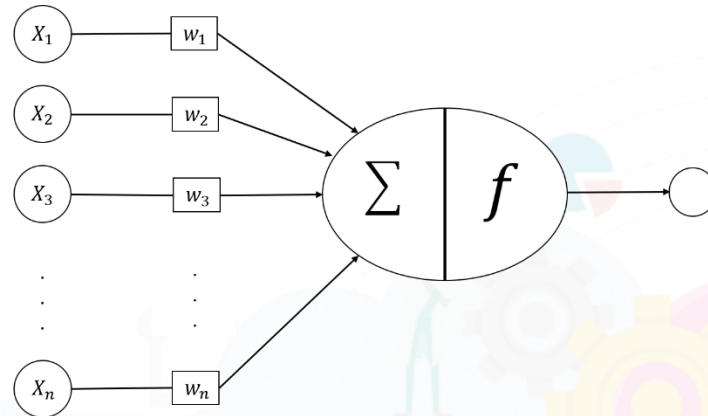




# Neurônio Artificial x Neurônio Biológico



# Neurônio Artificial - Perceptron



$$\sum_{i=1}^n x_i \cdot w_i$$

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + \dots + x_n w_n = x^T \cdot w$$

$$y = f(x^T \cdot w)$$

ou

$$y = f(x \cdot w^T)$$

# Neurônio Artificial - Perceptron

## Entradas:

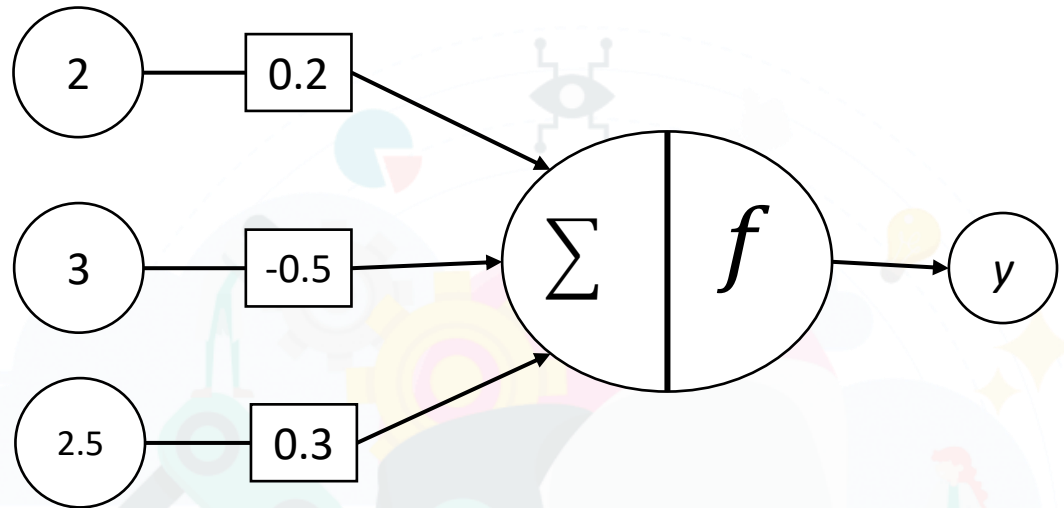
- $x_1 = 2$ ,
- $x_2 = 3$
- $x_3 = 2.5$

## Pesos:

- $w_1 = 0.2$
- $w_2 = -0.5$
- $w_3 = 0.3$

## Função de Ativação

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



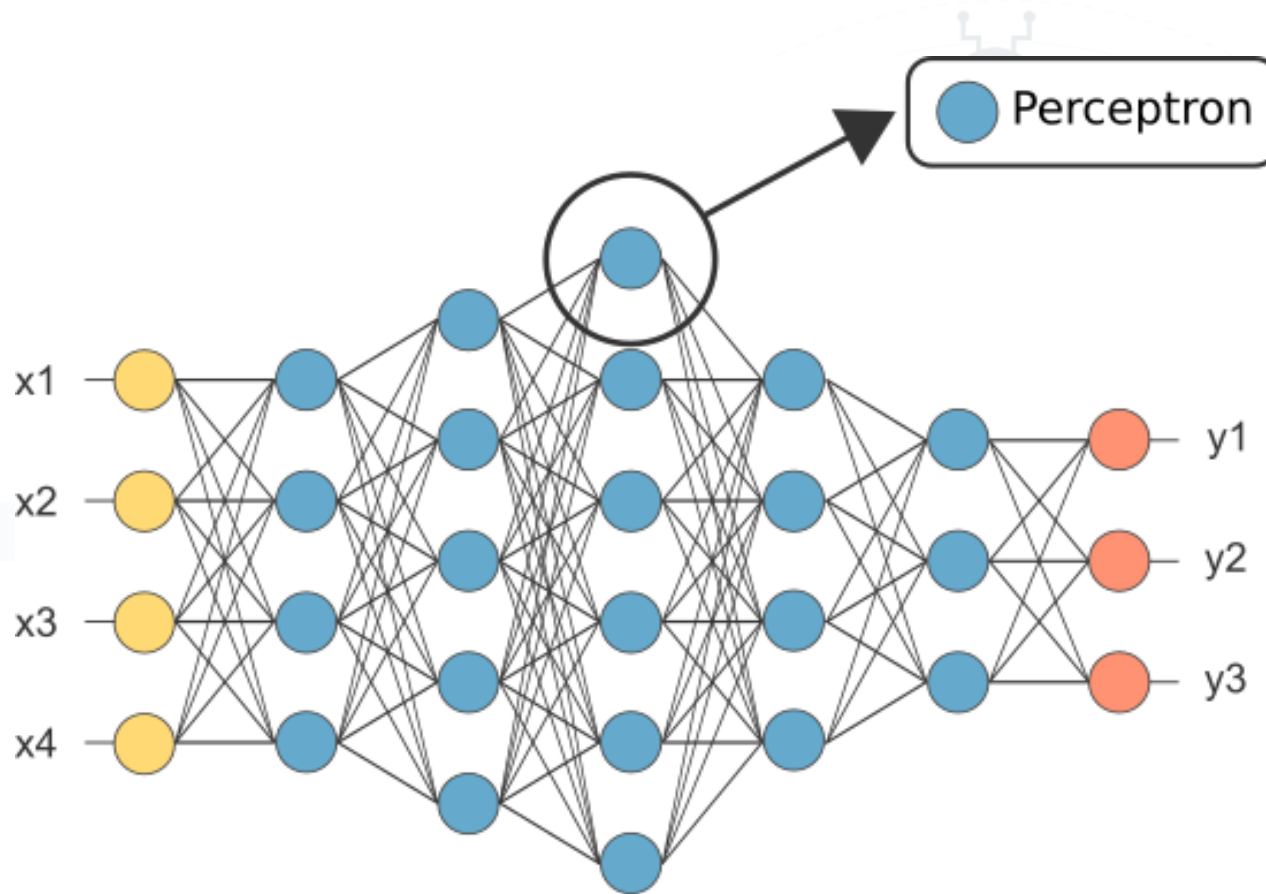
$$f(x_1w_1 + x_2w_2 + x_3w_3)$$

$$f(2 * 0.2 + 3 * (-0.5) + 2.5 * 0.3)$$

$$f(0.4 + (-1.5) + 2.8)$$

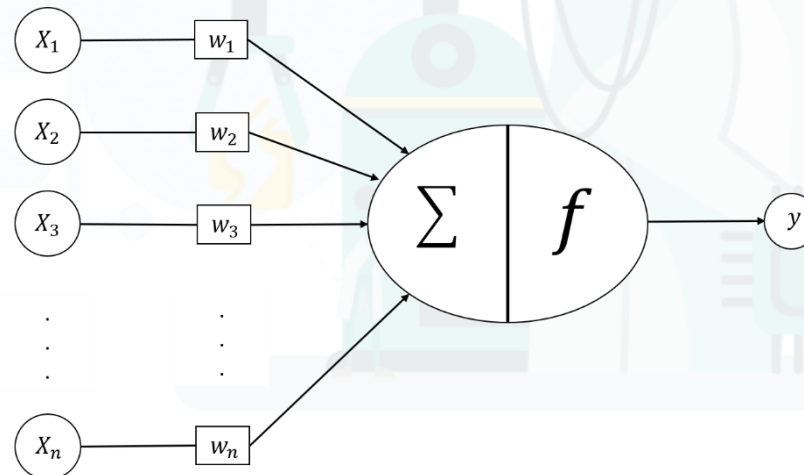
$$f(1.7) = 1$$

# Rede Neural Artificial



# Rede Perceptron – Correção de Pesos

- Para o perceptron responder corretamente, deve ter os pesos “ideais”;
- Após a correção dos pesos, o neurônio emite resposta correta para qualquer nova entrada;
- Para obter esses pesos, no perceptron, utiliza-se um ajuste de peso, por meio da Regra Delta;





# Rede Perceptron – Correção de Pesos

- Define-se aleatoriamente os pesos iniciais;

- Algoritmo baseado no erro

$$e = y - y_{pred}$$

- Onde  $y_{pred}$  é a resposta emitida pelo perceptron.

- Atualização dos pesos

$$w_{i+1} = w_i + (t * x_i * e)$$

- Onde  $t$  é a taxa de aprendizagem.

# Algoritmo de Treinamento

🌐 Enquanto não atingir condição de parada:

- Para cada Amostra de Treinamento  $x$ :
  - Calcular a saída com os pesos atuais para  $x$ ;
  - Calcular o erro;
  - Para cada peso do perceptron:
    - Atualiza usando a regra Delta

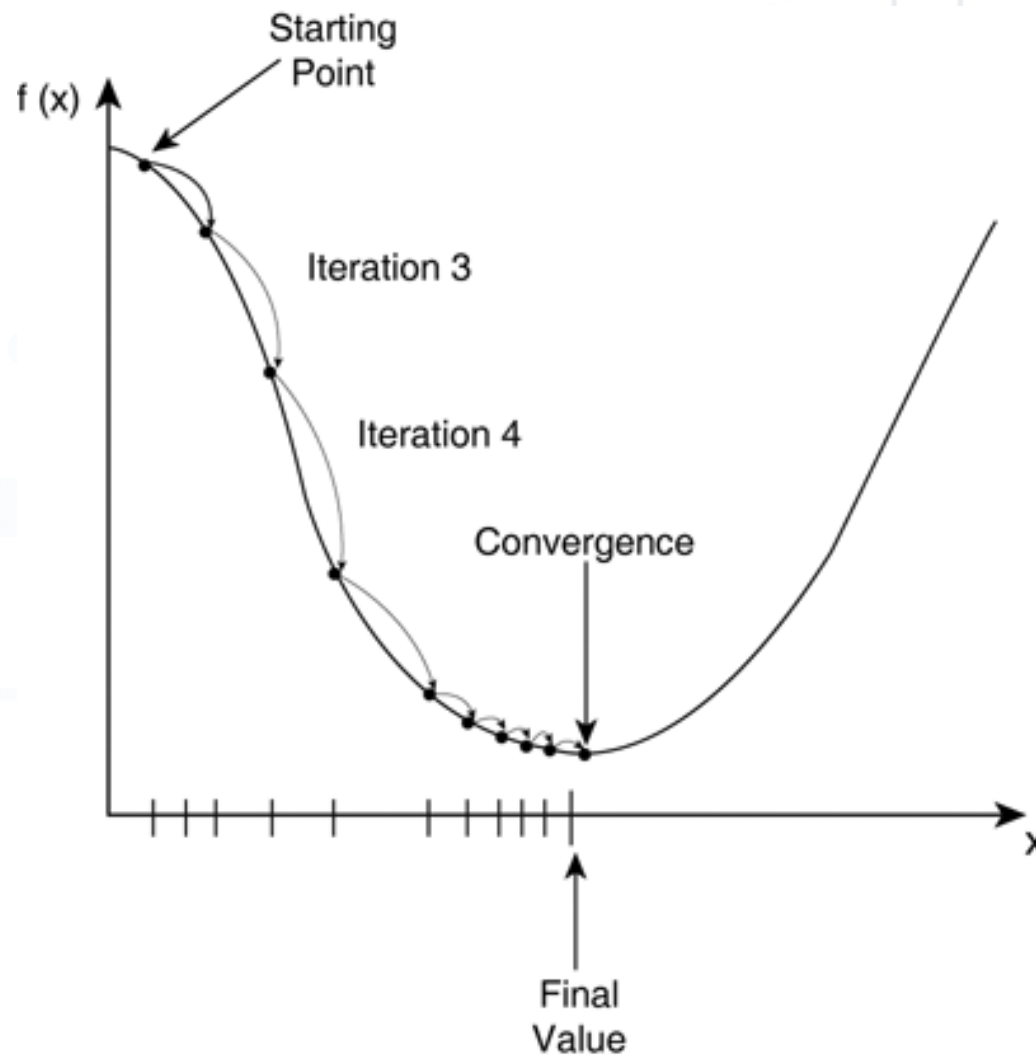
# Taxa de Aprendizagem

- Dita o “quanto” a rede vai aprender em uma correção de peso;
- Influencia no tempo de treinamento de uma rede neural;
- Geralmente varia entre 0.1 e 1.0;

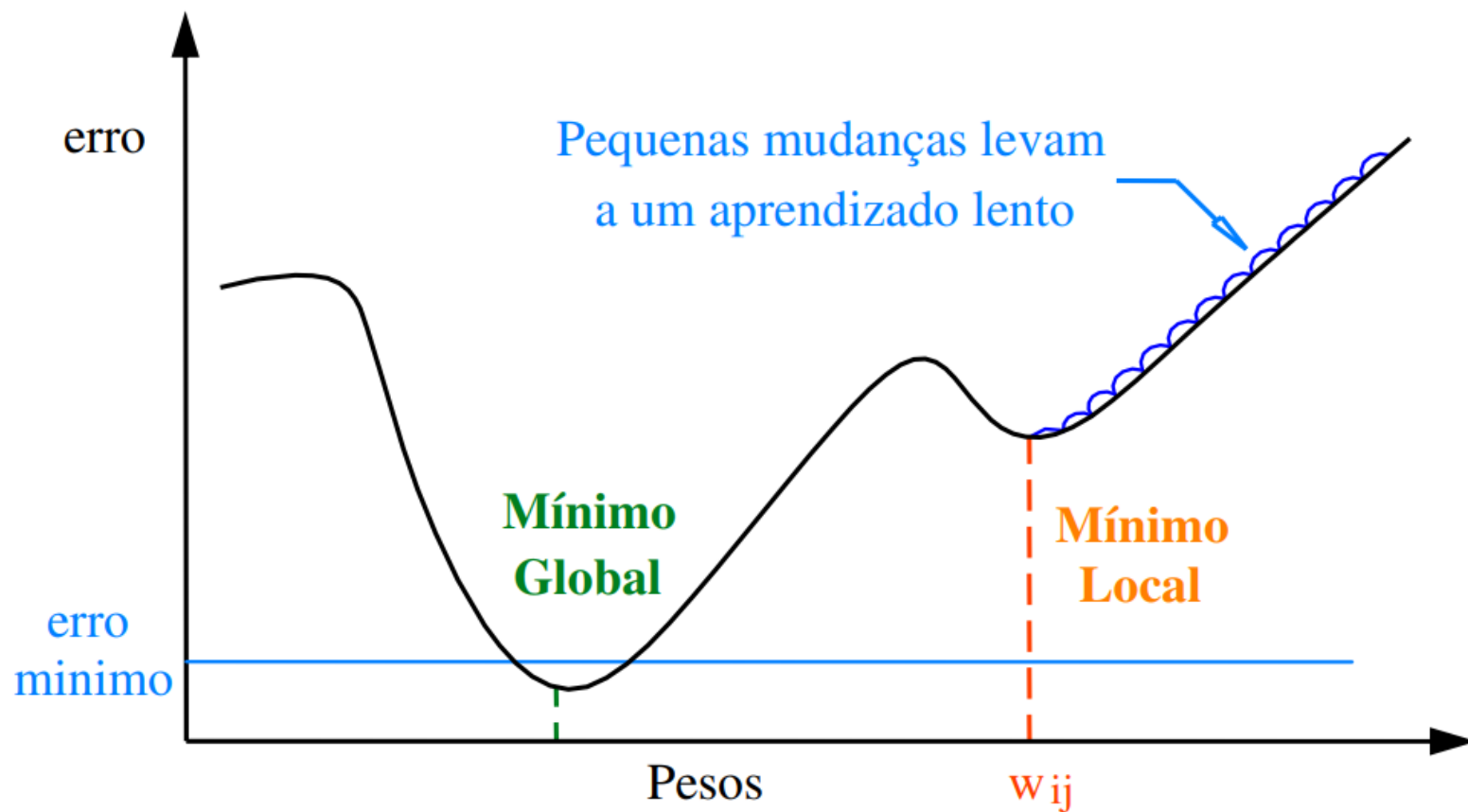
$$w_{i+1} = w_i + (\textcolor{red}{t} * x_i * e)$$

Taxa de Aprendizagem

# Taxa de Aprendizagem

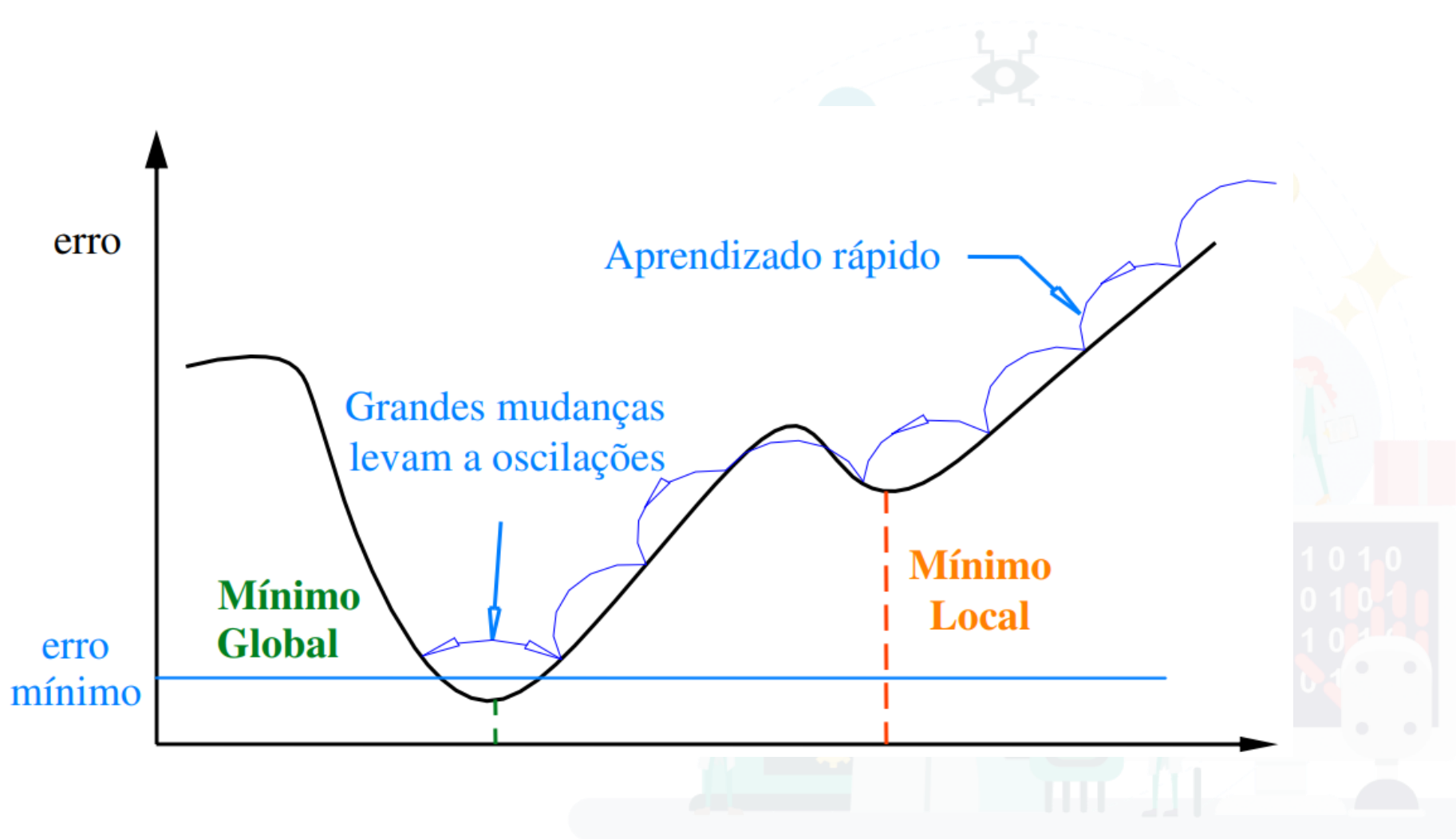


# Taxa de Aprendizagem





# Taxa de Aprendizagem



# Rede Perceptron – Correção de Pesos

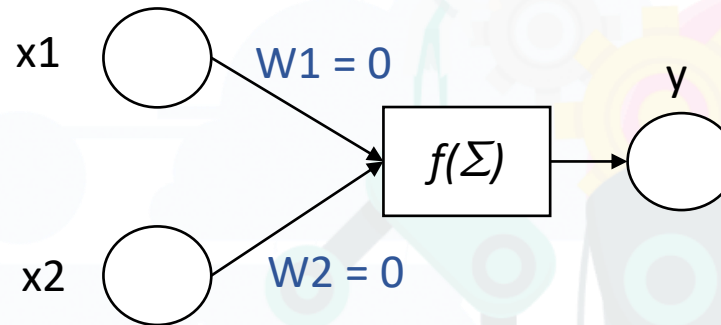
## Exemplo: Tabela Verdade OU

P	Q	P ou Q
V	V	V
V	F	V
F	V	V
F	F	F

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

# Rede Perceptron – Correção de Pesos

## 🌐 Inicialização do Perceptron



Pesos Iniciais

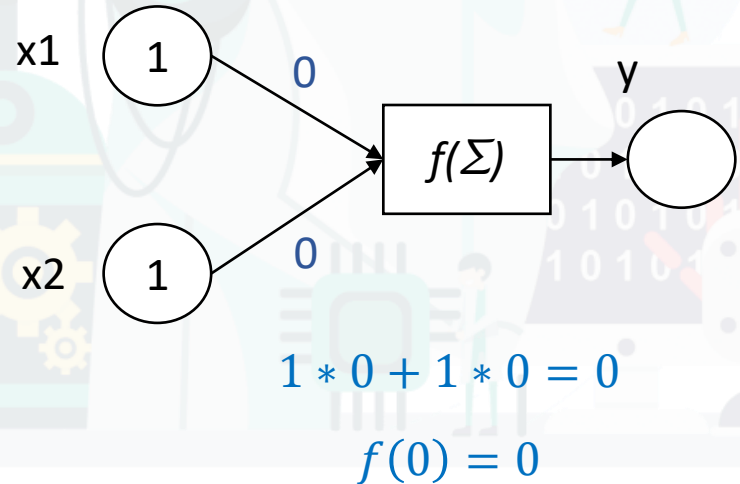
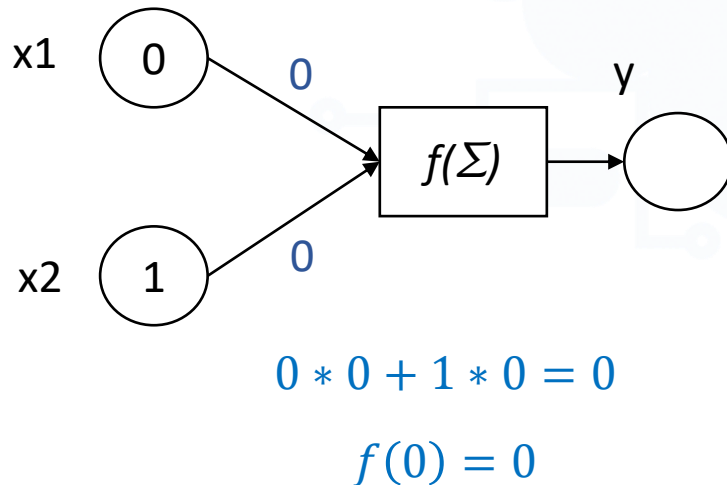
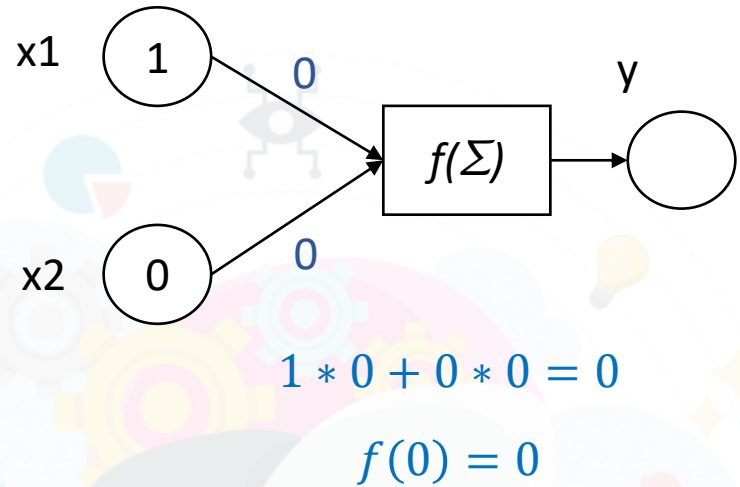
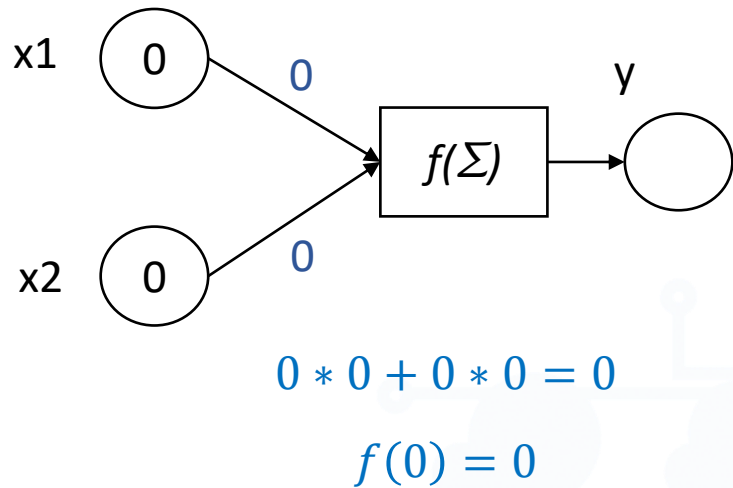
$$w_1 = 0$$

$$w_2 = 0$$

Função de Ativação

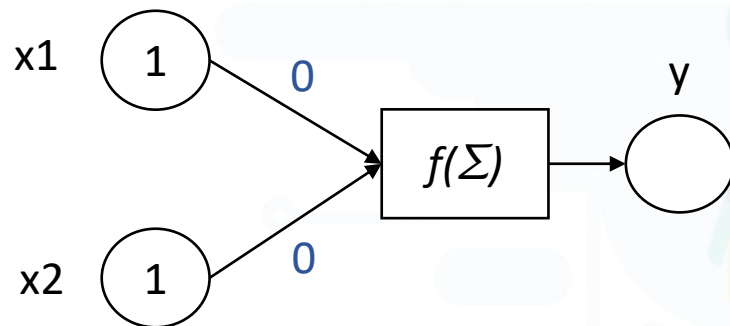
$$f(x) = \begin{cases} 0, & x < 1 \\ 1, & x \geq 1 \end{cases}$$

# Rede Perceptron – Correção de Pesos



# Rede Perceptron – Correção de Pesos

X1	X2	Y	Y_pred	Erro
0	0	0	0	$0 - 0 = 0$
0	1	0	0	$0 - 0 = 0$
1	0	0	0	$0 - 0 = 0$
1	1	1	0	$1 - 0 = 1$



$$1 * 0 + 1 * 0 = 0$$

$$f(0) = 0$$

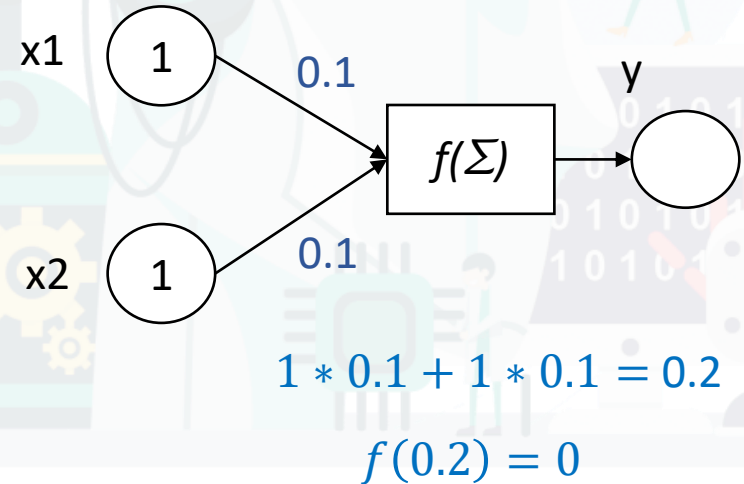
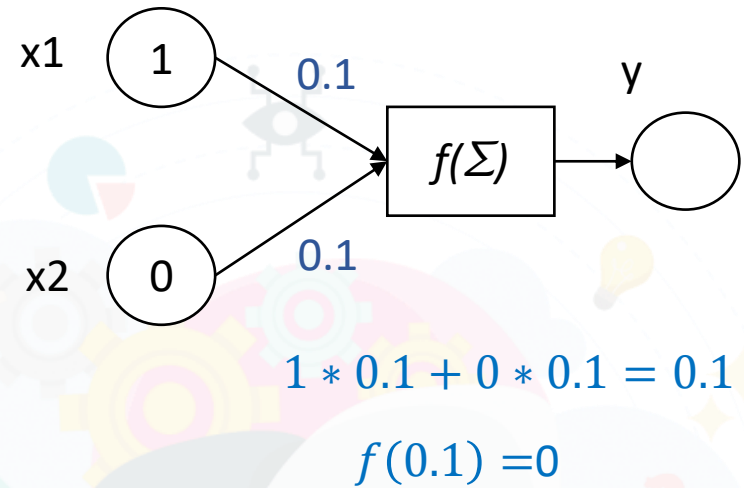
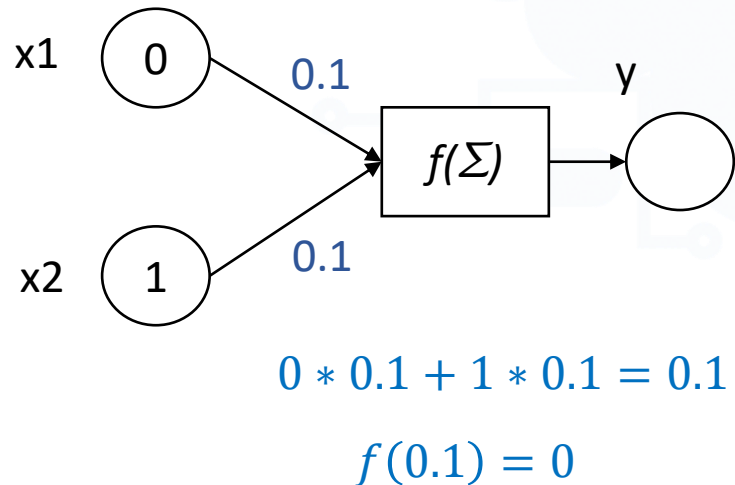
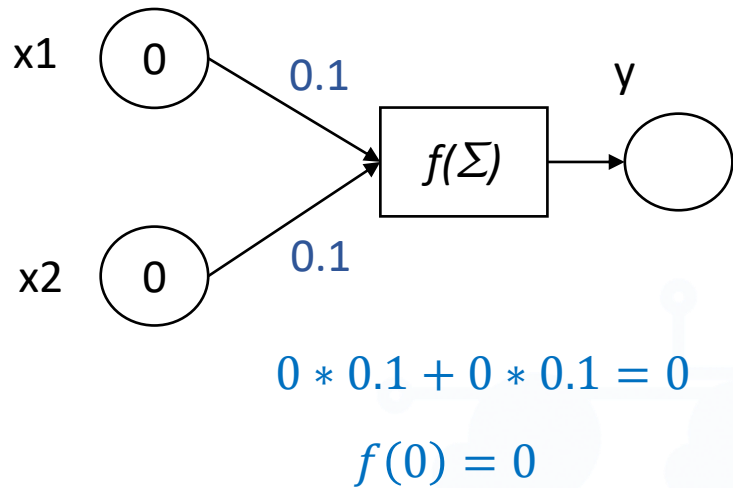
$$w_{i+1} = w_i + (t * x_i * e)$$

$$w_1 = 0 + (0.1 * 1 * 1) = 0.1$$

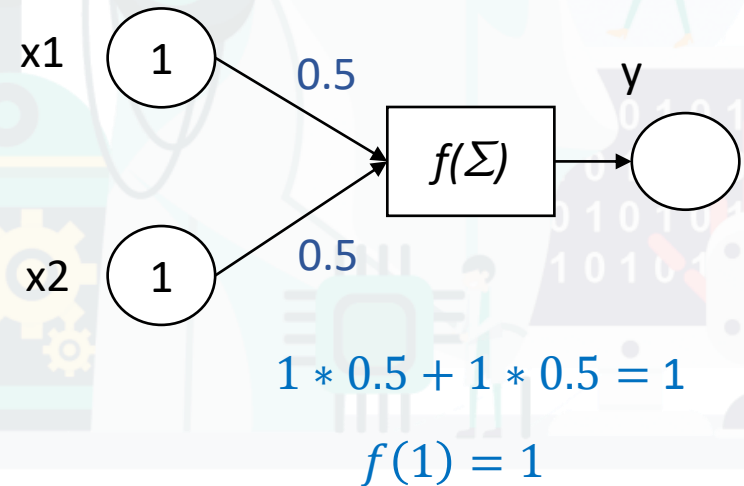
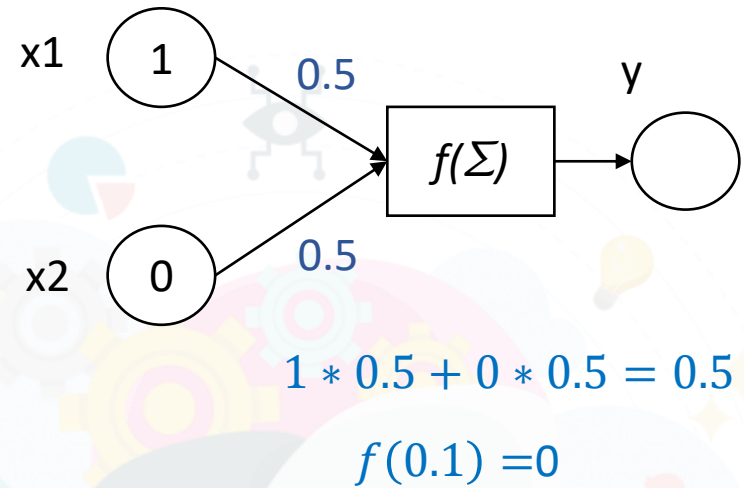
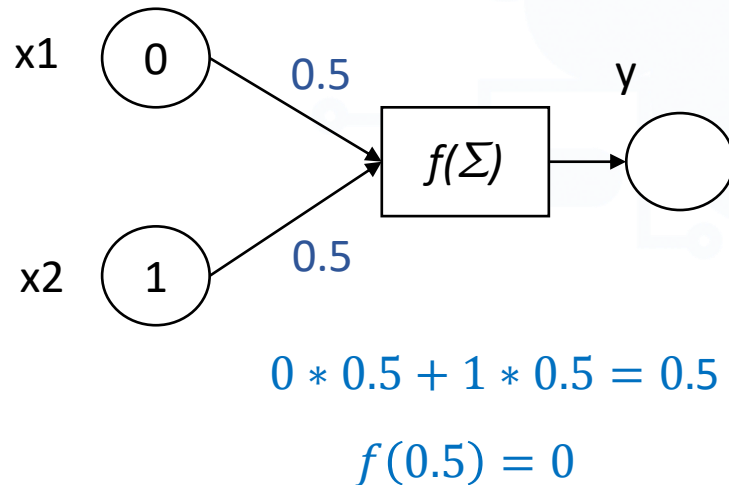
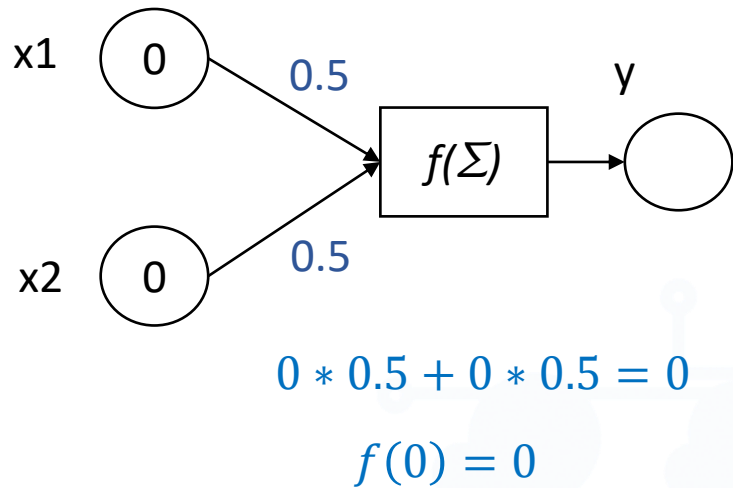
$$w_2 = 0 + (0.1 * 1 * 1) = 0.1$$



# Rede Perceptron – Correção de Pesos



# Rede Perceptron – Correção de Pesos



# Treinamento Perceptron

- Treinar um perceptron significa ajustar os pesos para que este responda corretamente para todas as entradas;
- Os pesos iniciais influenciam no tempo de treinamento;

# Implementação

## 🌐 Hora de Prática

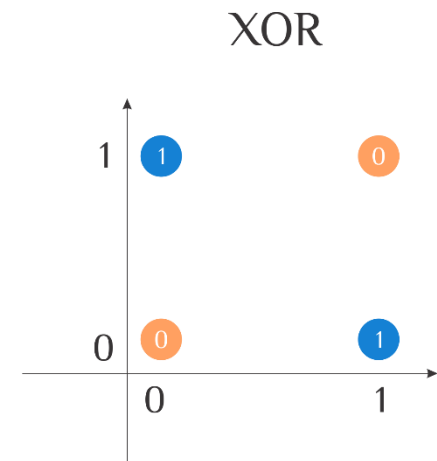
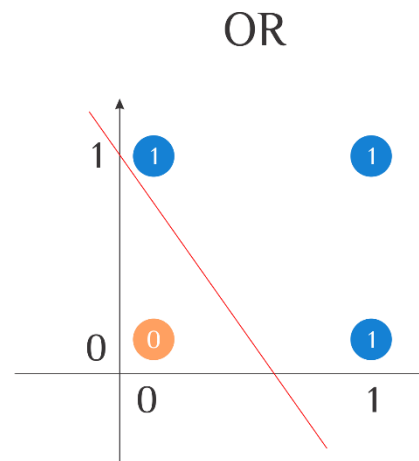
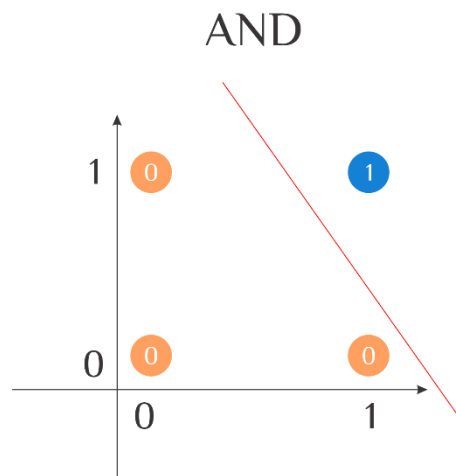
➤ Implementação de um neurônio artificial em Python.

P	Q	P ou Q
V	V	V
V	F	V
F	V	V
F	F	F

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

# Rede Perceptron

- Pode conter mais de um neurônio;
- Resolve apenas problemas lineares





# Funções de Ativação

- São essenciais para dar capacidade representativa às redes neurais artificiais;
- Introduz componente de não linearidade à rede;
- Decide a ativação ou não de um neurônio.

# Funções de Ativação

- Considere a Equação:

$$y = f(x_1w_1 + x_2w_2 + x_3w_3)$$

- Removendo a função  $f$ :

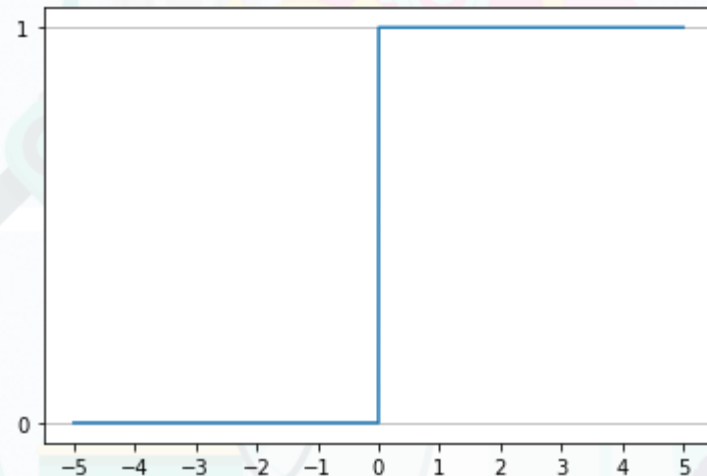
$$y = x_1w_1 + x_2w_2 + x_3w_3$$

- *Resta apenas um modelo de regressão linear*

# Funções de Ativação

## 🌐 Função Degrau

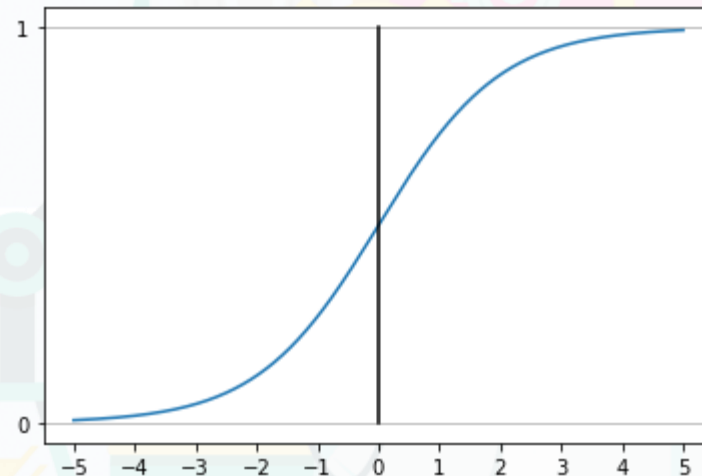
$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$



# Funções de Ativação

## 🌐 Função Sigmoid

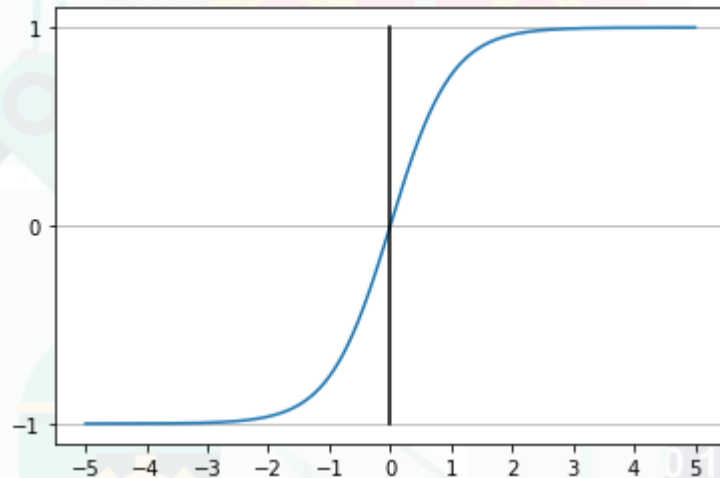
$$f(x) = \frac{1}{1 + e^{-x}}$$



# Funções de Ativação

## Função Tangente Hiperbólica

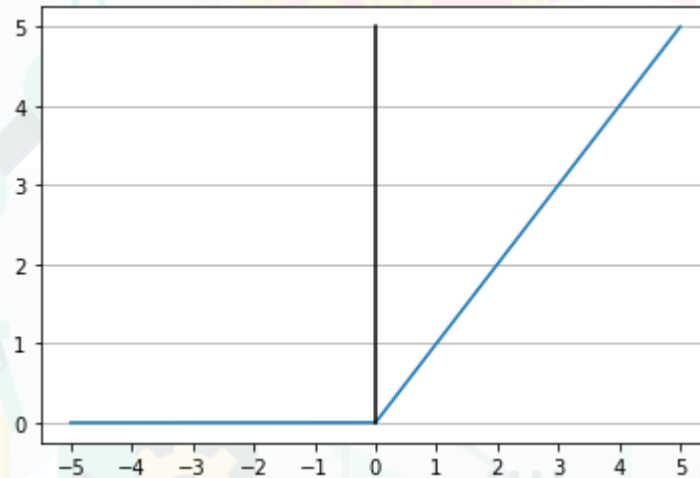
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



# Funções de Ativação

## 🌐 Função Relu

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



# Funções de Ativação

## ● Função Softmax

$$f(x) = \frac{e^x}{\sum_i^n e^{x_i}}$$

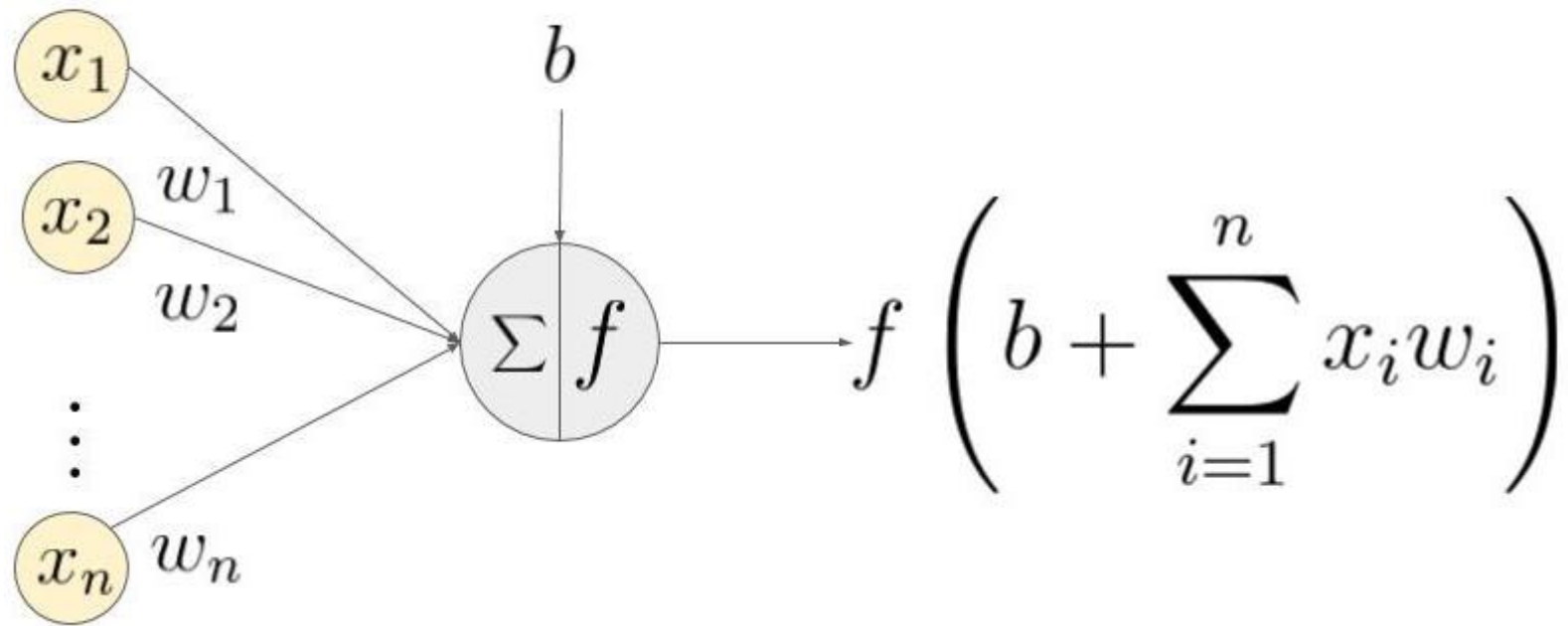
● Utilizada na camada de saída.



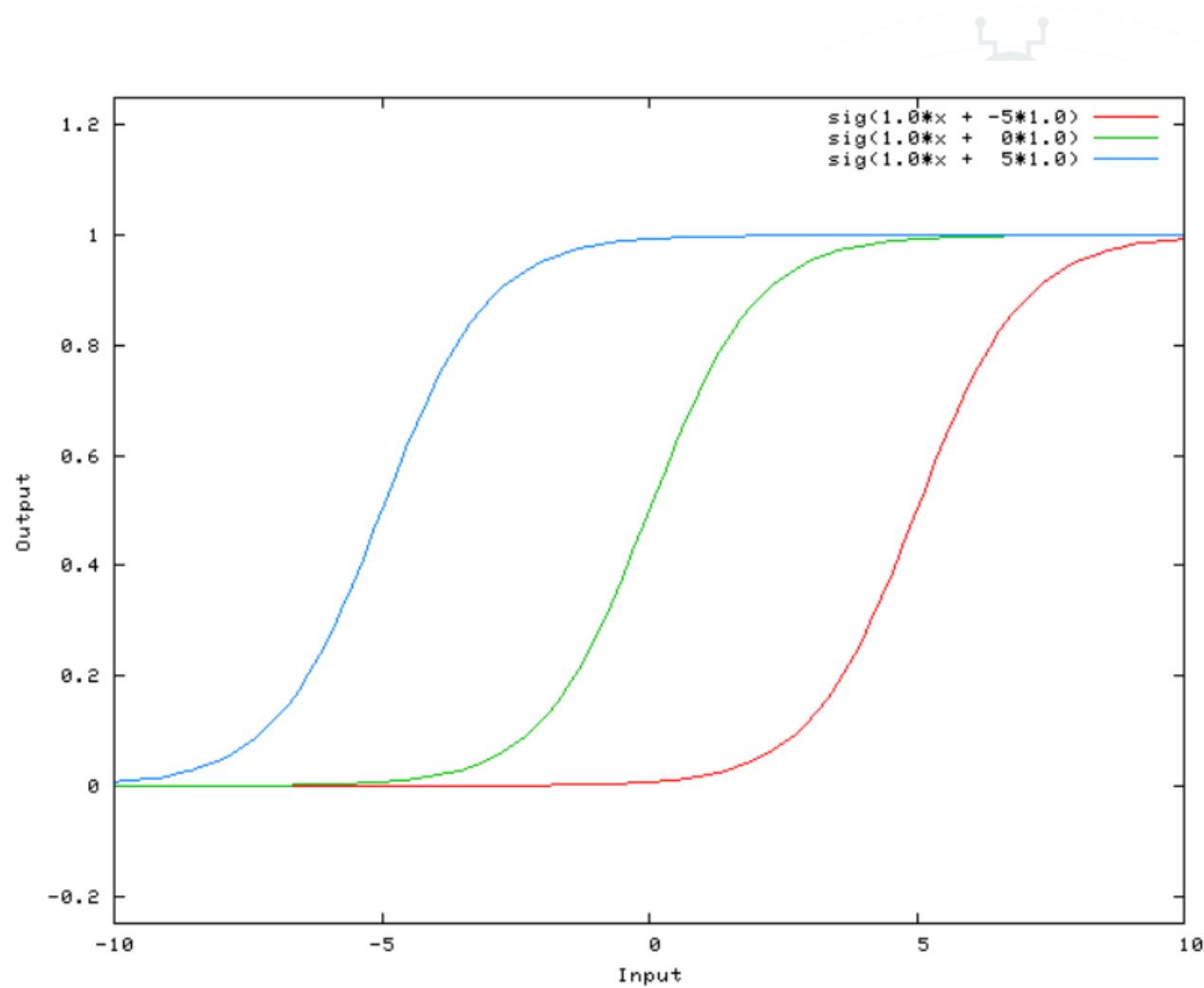
# Bias

- Também chamada viés ou limiar;
- O Bias permite flexibilidade de ajuste da RNA;
- Valor do bias varia entre -1 ou 1;
- Possui um peso associado que também é ajustado no treinamento.

# Bias

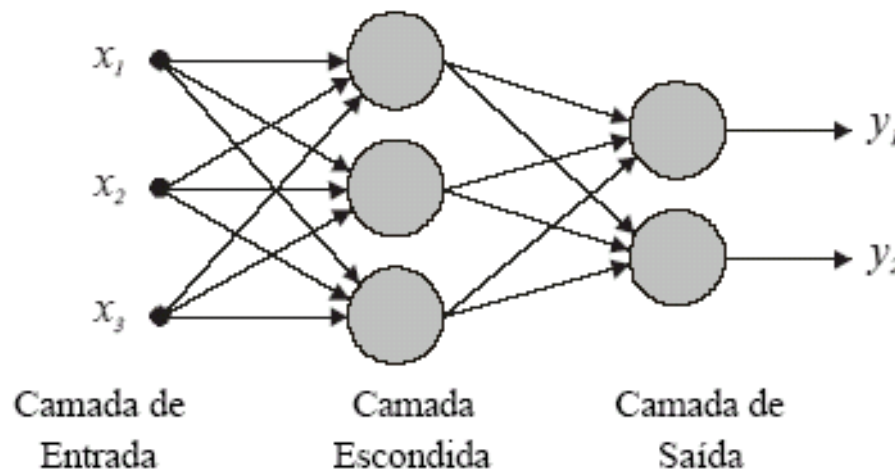


# Bias



# Rede Perceptron Multicamada

- Arquitetura onde os neurônios são organizados em pelo menos uma camada de entrada, uma camada oculta e uma camada de saída;



# Rede Perceptron Multicamada

- Arquitetura *feedforward*;
- As limitações do perceptron deixam de existir;
- Aprendizado supervisionado e não supervisionado;
- Pode ser Rasa ou Profunda;

# Treinamento de uma MLP

- Significa Ajuste dos pesos (parâmetros) da rede;
- Algoritmo *Backpropagation*;
- Alguns **hiperparâmetros** são determinados por tentativa e erro:
  - Configuração da Rede
  - Taxa de Aprendizado
  - Momentum
  - Tamanho do Lote

# Algoritmo BackPropagation

Realiza o treinamento de uma Rede Neural Artificial;

O objetivo do backpropagation é otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas para as saídas.

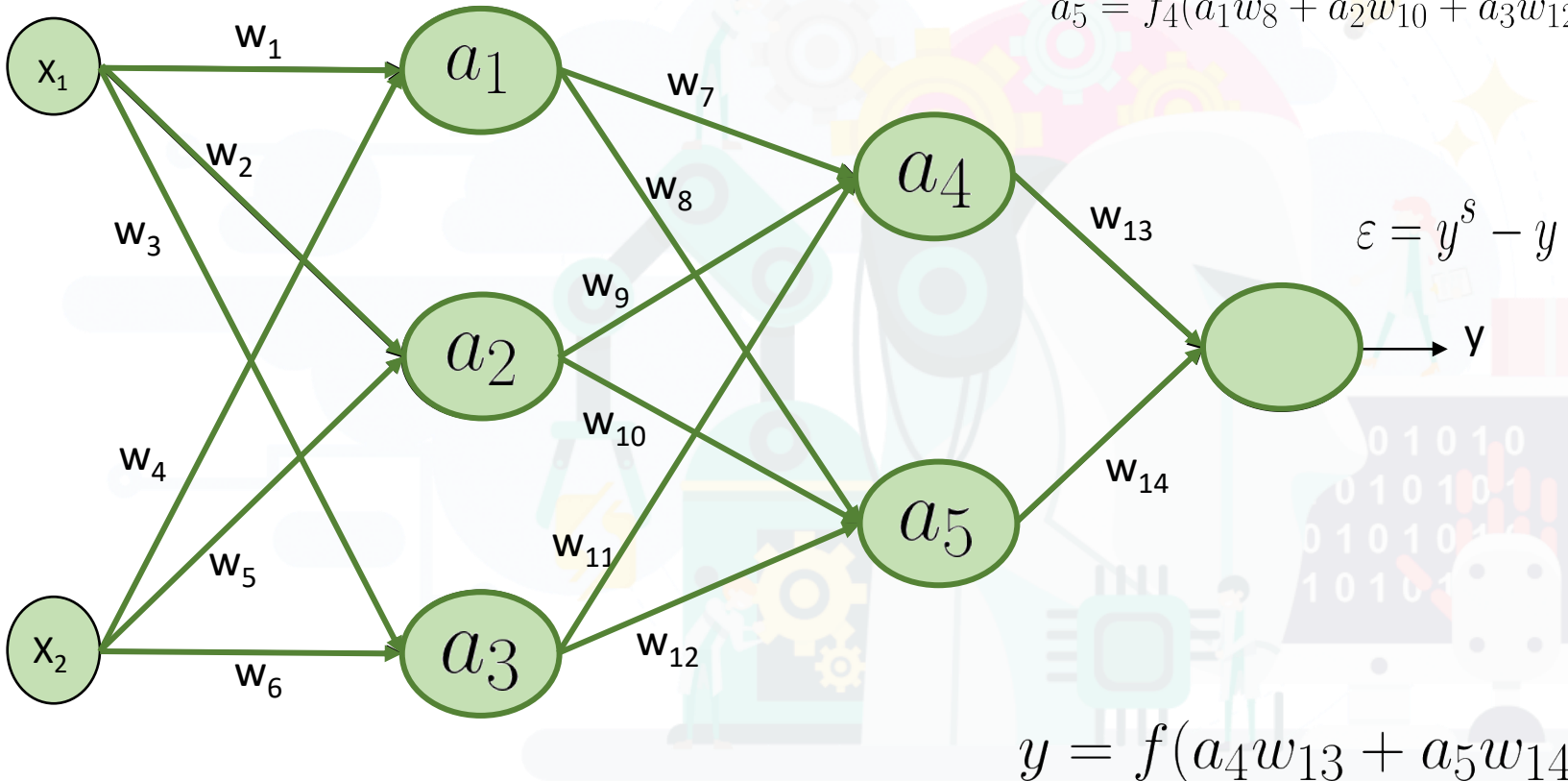


# Algoritmo Backpropagation

- “Retropropagação do Erro”;
- Corrige os erros da uma MLP por meio da descida do gradiente;
- Realizado em duas fases:
  - Forwad - Propagação
  - Backforwad - Retropropagação

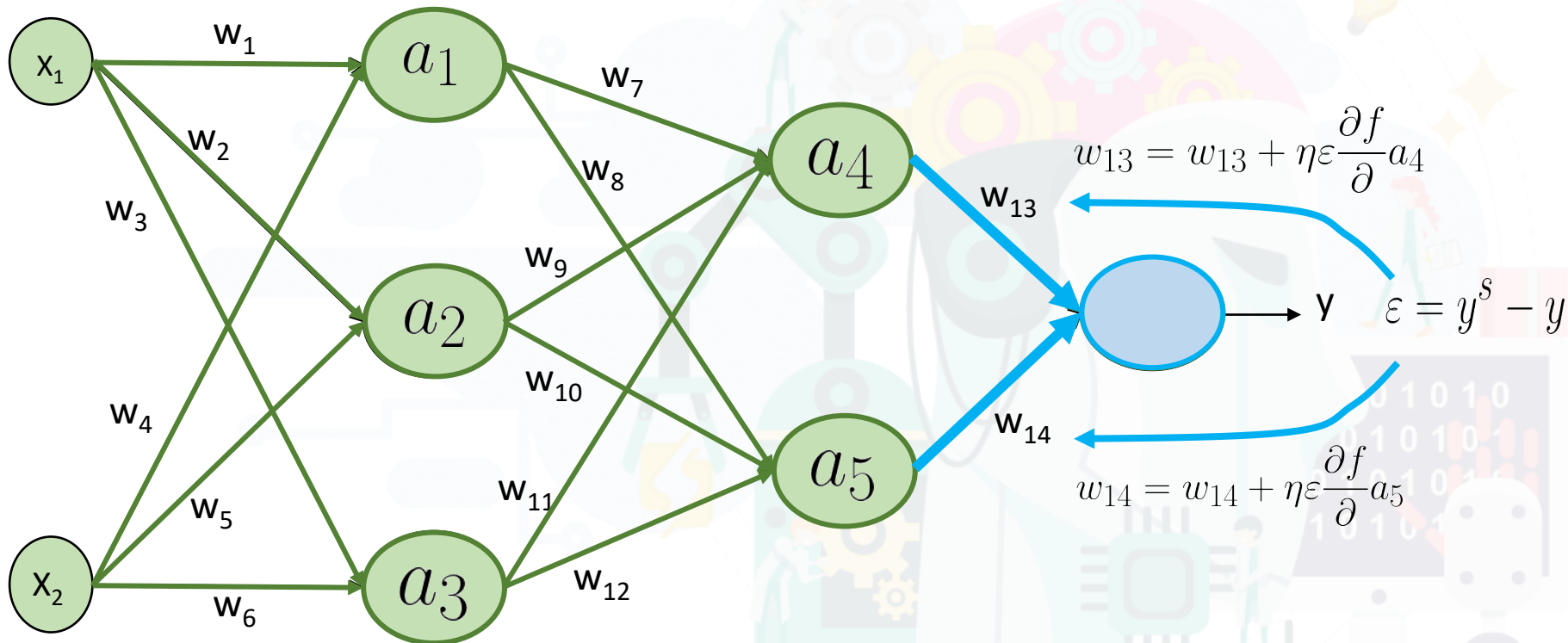
# Algoritmo Backpropagation

## 🌐 Fase Forward



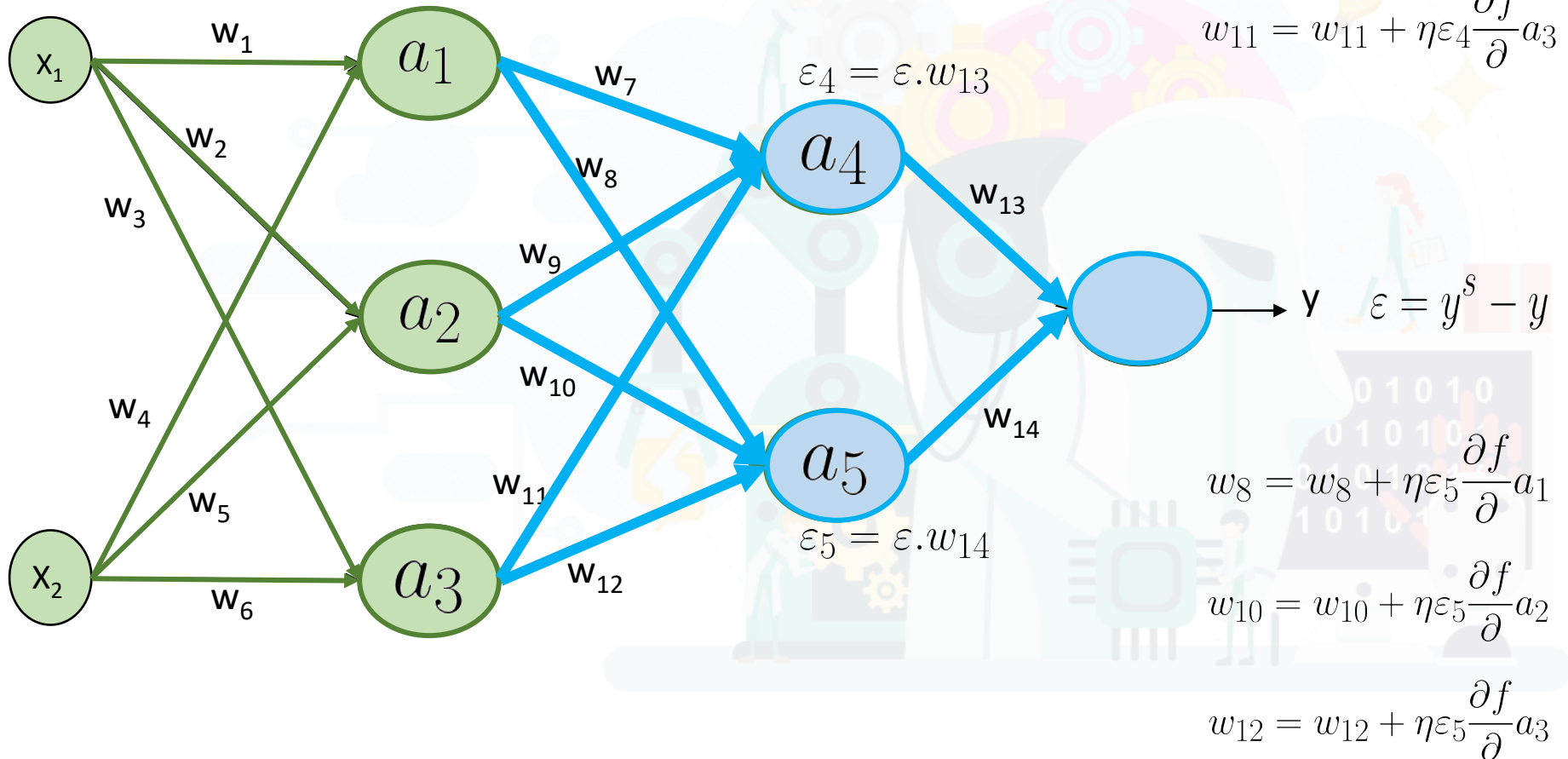
# Algoritmo Backpropagation

## 🌀 Fase Backward



# Algoritmo Backpropagation

## 🌀 Fase Backward



# Algoritmo Backpropagation

## 🌀 Fase Backward

$$\varepsilon_1 = \varepsilon_4 \cdot w_7 + \varepsilon_5 \cdot w_8$$

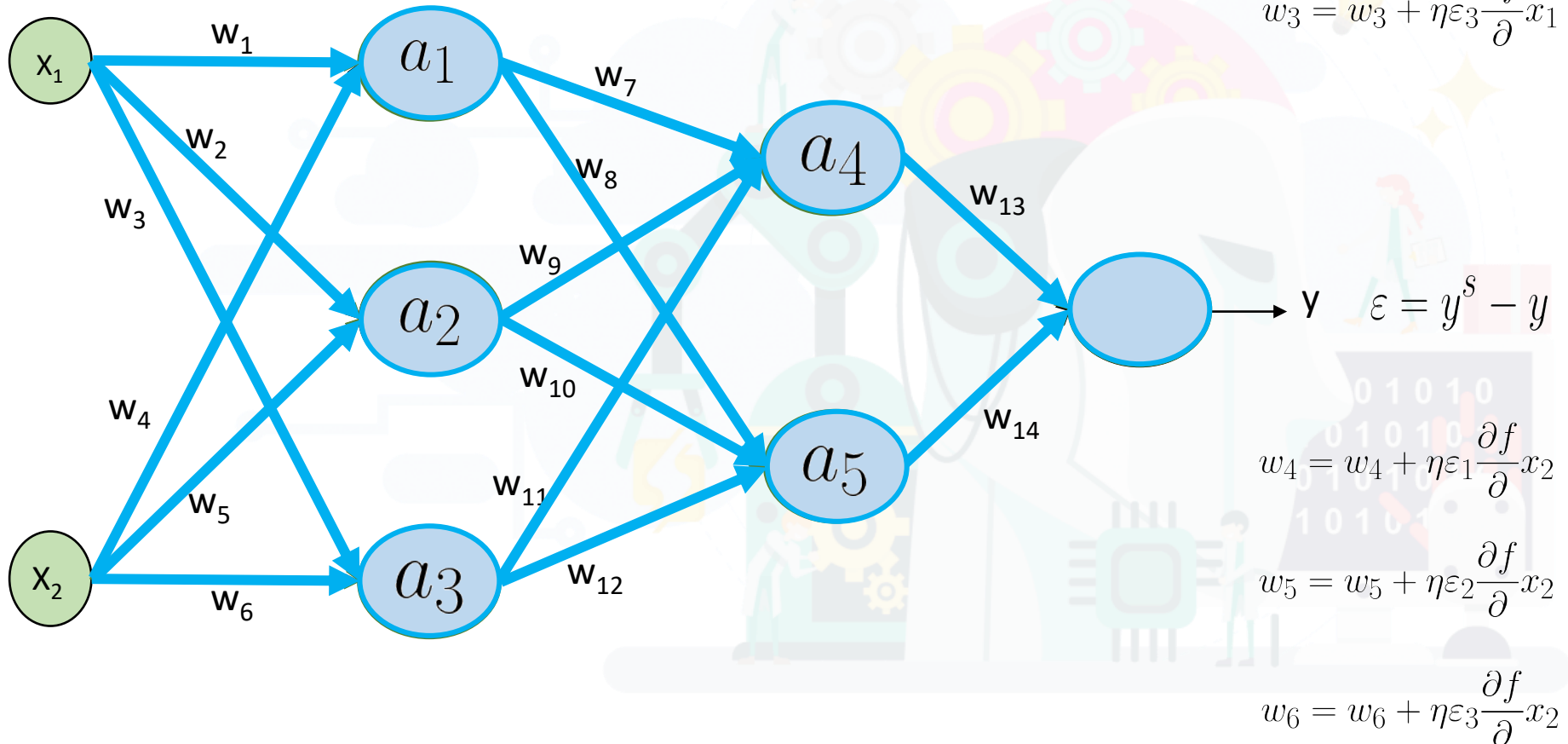
$$\varepsilon_2 = \varepsilon_4 \cdot w_9 + \varepsilon_5 \cdot w_{10}$$

$$\varepsilon_3 = \varepsilon_4 \cdot w_{11} + \varepsilon_5 \cdot w_{12}$$

$$w_1 = w_1 + \eta \varepsilon_1 \frac{\partial f}{\partial x_1}$$

$$w_2 = w_2 + \eta \varepsilon_2 \frac{\partial f}{\partial x_1}$$

$$w_3 = w_3 + \eta \varepsilon_3 \frac{\partial f}{\partial x_1}$$



# Algoritmo Backpropagation

- Por X épocas:
- Passo 1 – Inicialização dos Pesos e Hiperparâmetros;
- Passo 2 - Fase Forward
  - Para cada entrada  $X$ , calcular a saída  $y$ ;
  - Calcular o erro da saída da rede
- Passo 3 – Fase backward
  - Efetuar os cálculos de atualização dos Pesos da rede da camada saída até chegar a camada de entrada.

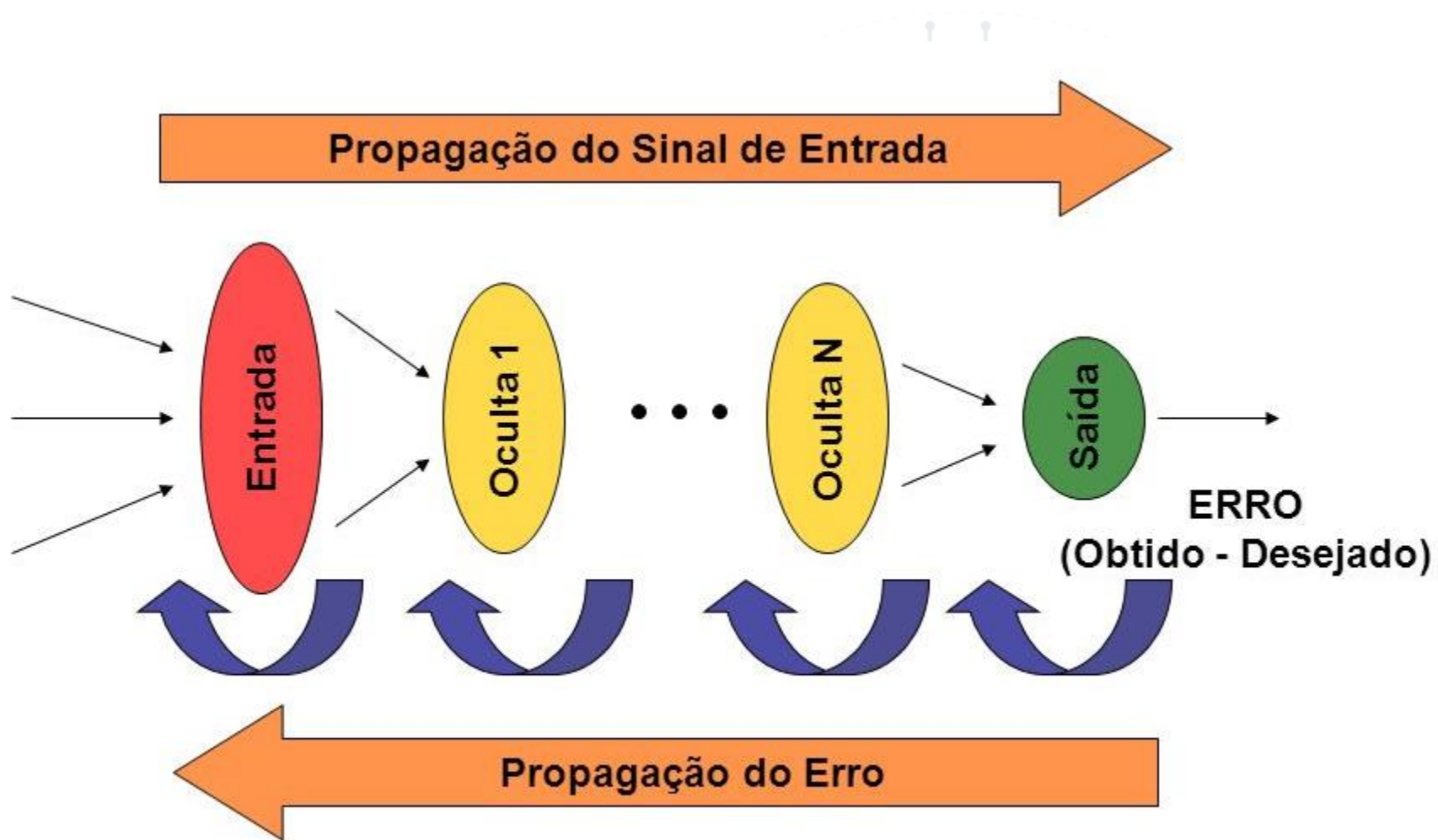
# Algoritmo BackPropagation

- As fases forward e backward são realizadas para todas as amostras de entrada do conjunto de treino;
- Condições de Parada:
  - Erro aceitável;
  - Quantidade de épocas
- **Época** – Quando processo de forward e backward realizado com todas as entradas é caracterizado uma época.



# Algoritmo Backpropagation

Redes Neurais





**Universidade Federal do Piauí**  
Laboratório de Inteligência Artificial - LINA

# Introdução à Deep Learning

**Bruno Vicente Alves de Lima**