

Fundamentos e Redes Neurais Clássicas

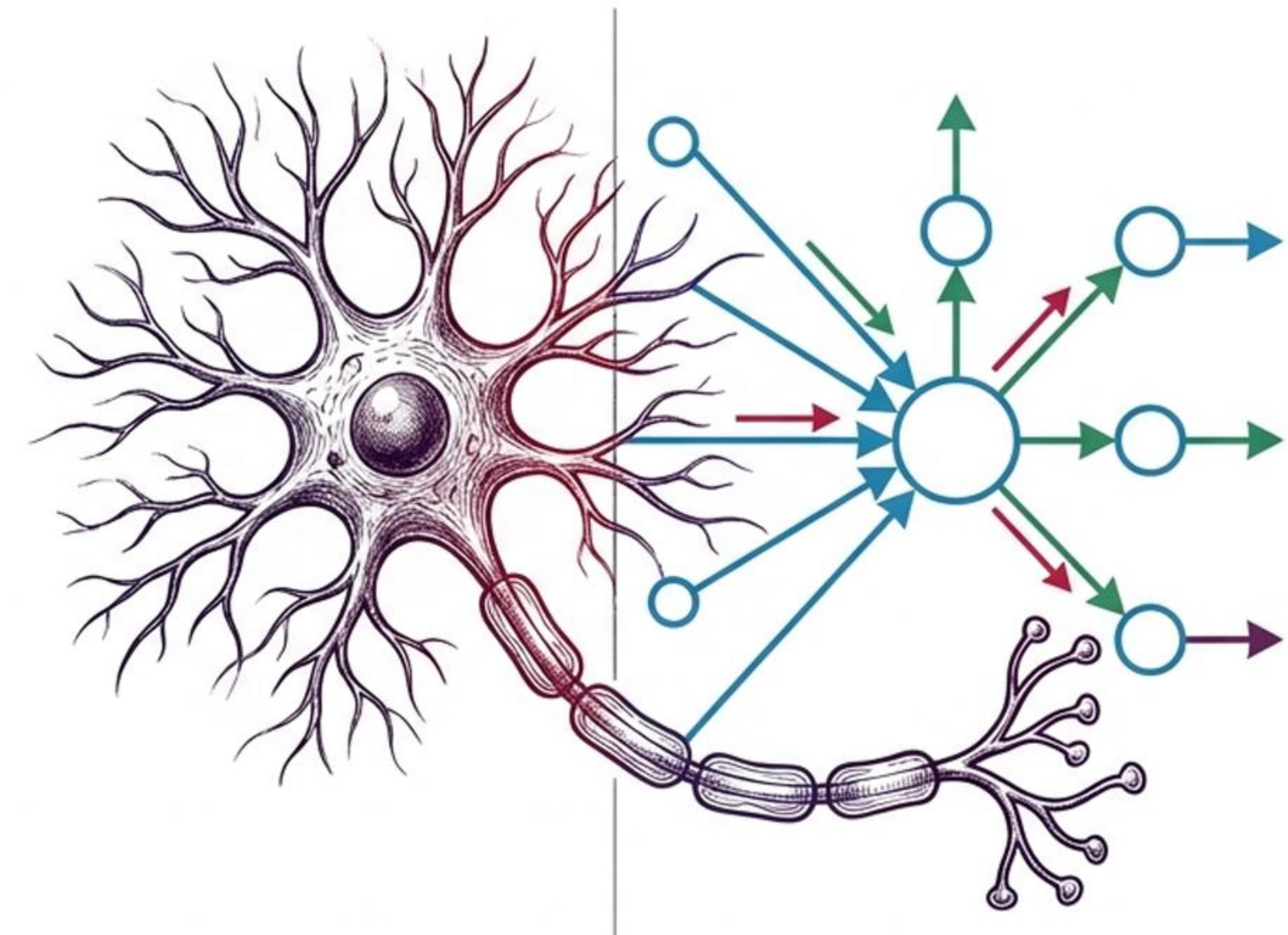
Perceptron e MLP

Disciplina Redes Neurais e Deep Learning
Curso Ciência de Dados e Inteligência Artificial 2026-2



Fundamentos de Redes Neurais Artificiais

Do Perceptron ao MLP: Uma Abordagem Matemática e Arquitetural



Introdução Teórica e Formulação Matemática

A Inspiração Biológica: O Computador Paralelo Massivo

O Cérebro vs. Von Neumann

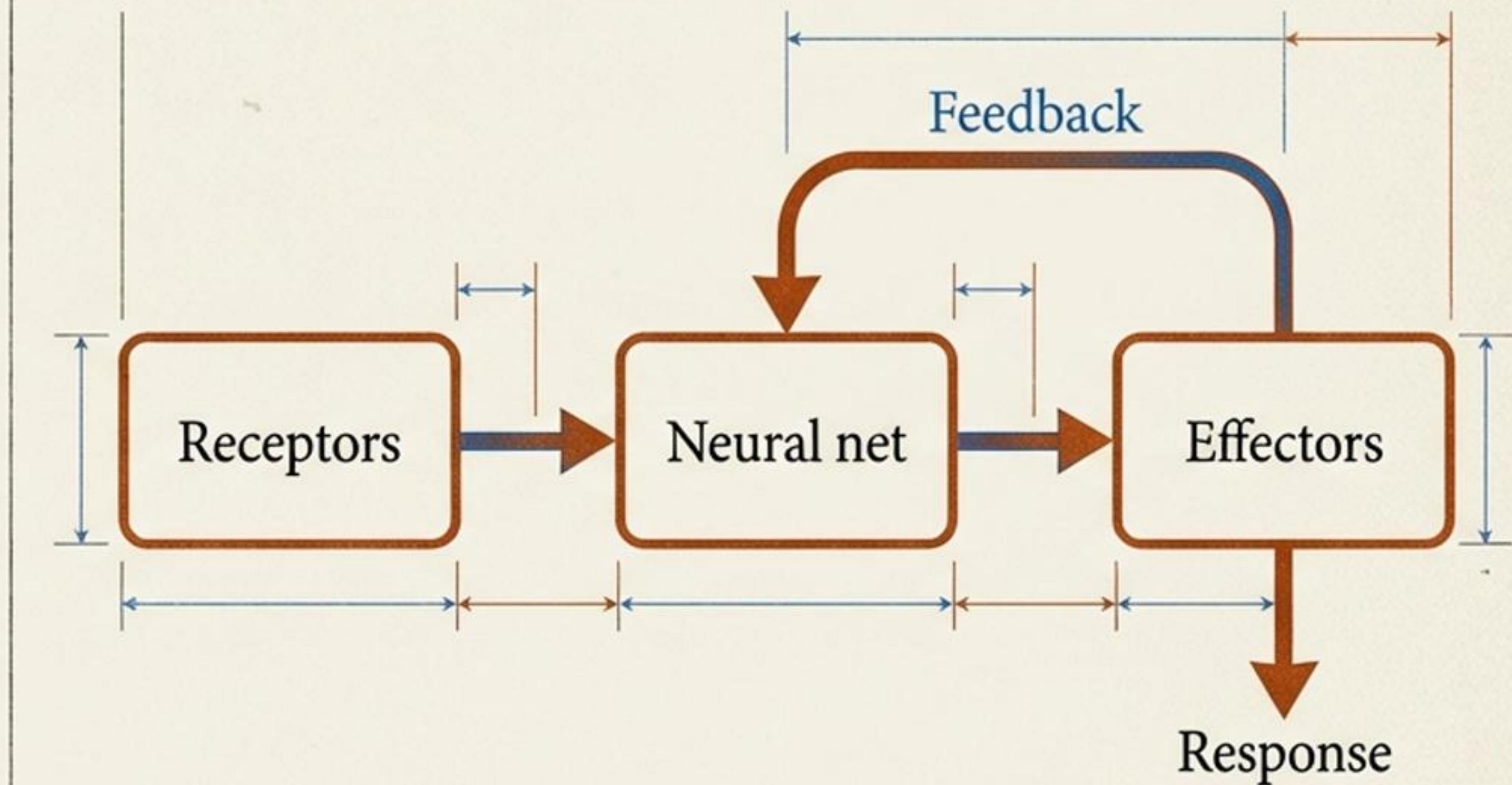
Enquanto computadores digitais operam de forma serial, o cérebro é um processador altamente complexo, não-linear e paralelo.

Velocidade e Eficiência

O cérebro realiza tarefas de reconhecimento perceptual (como identificar um rosto) em aproximadamente 100-200 ms. Computadores convencionais levariam muito mais tempo para tarefas de menor complexidade.

Plasticidade

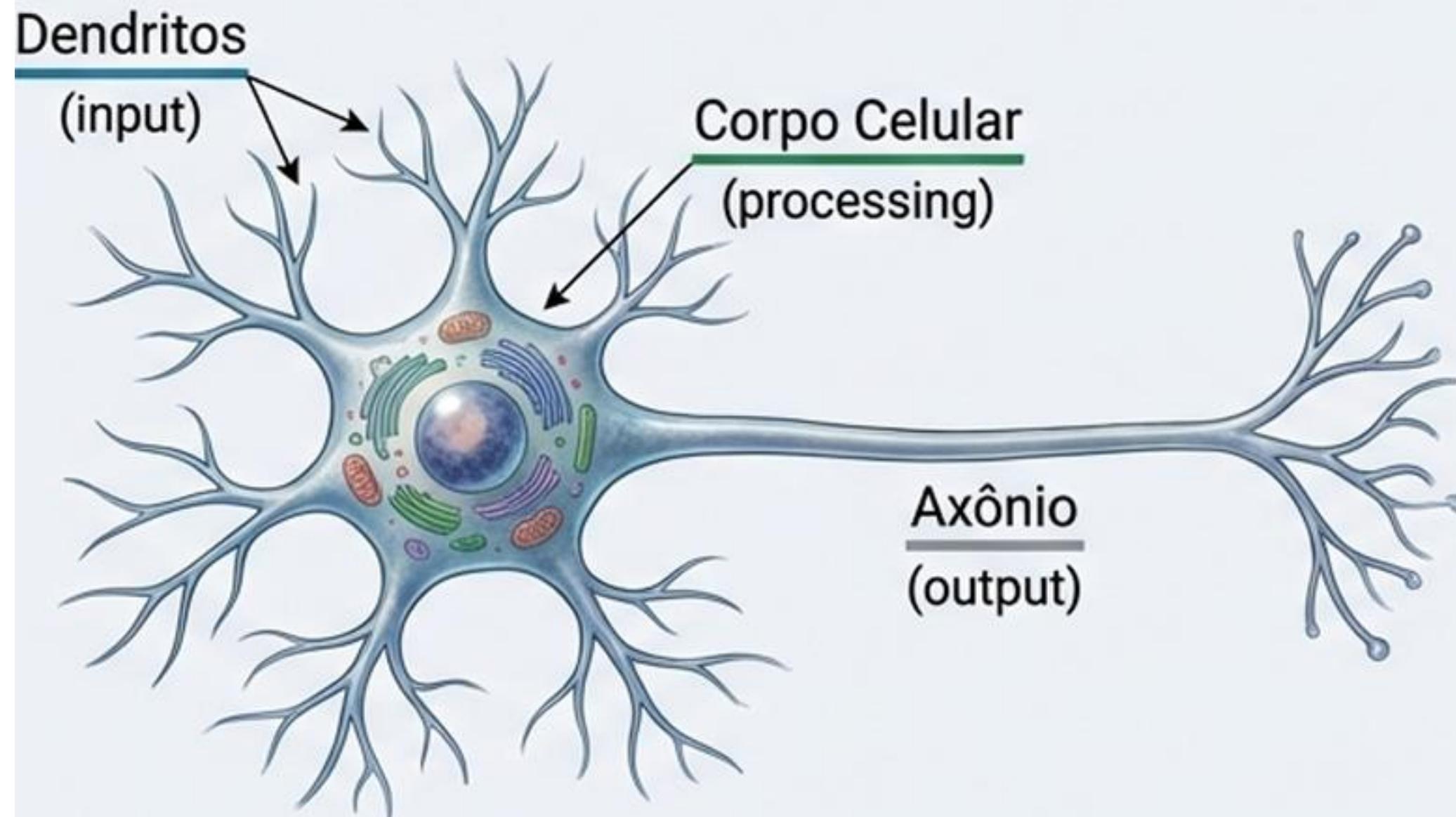
O diferencial biológico é a capacidade de adaptar-se ao ambiente. A criação de novas conexões sinápticas (ou a modificação das existentes) é a base física do aprendizado.



Block diagram representation of nervous system.

Da Biologia à Abstração Matemática

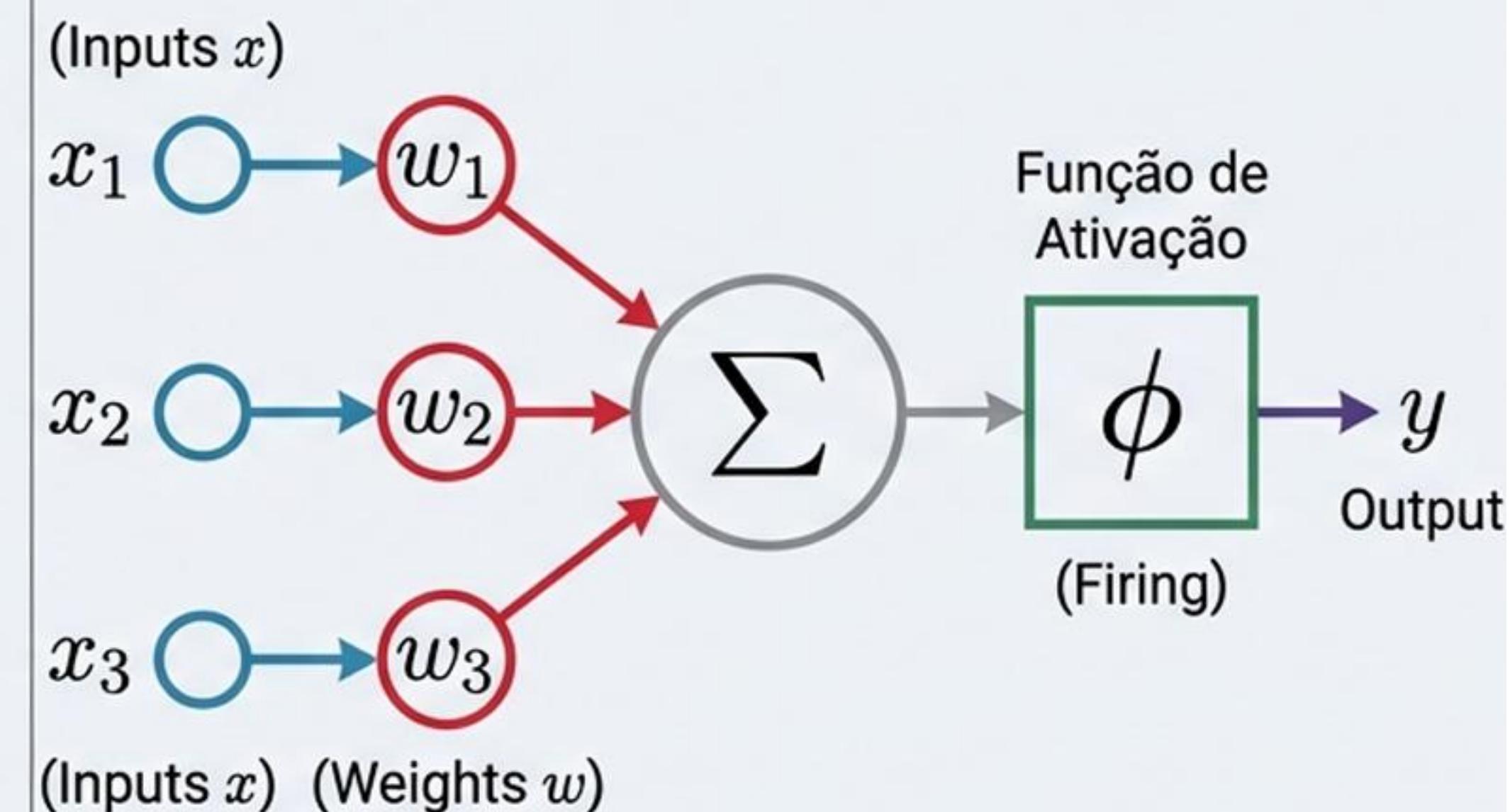
The Biological Model



Neurônio Biológico: Unidade fundamental. Sinais recebidos pelos dendritos são somados no corpo celular.

Sinapse: Conexão plástica. Aprendizado = Regra de Hebb ("Células que disparam juntas, permanecem conectadas").

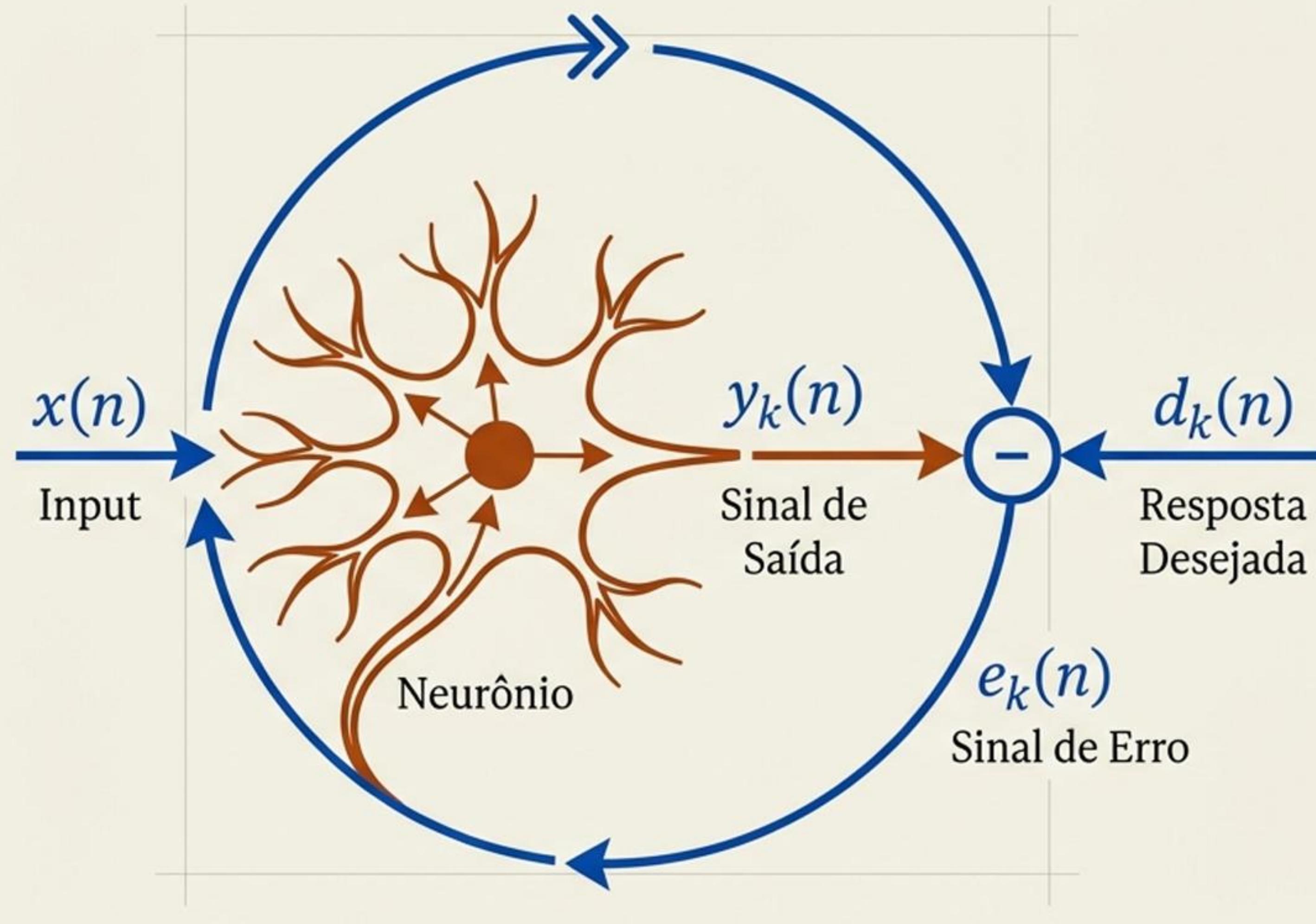
The Artificial Model



Neurônio Artificial: Simplificação algébrica.
Analogias:

- Sinais Eletroquímicos → Números (*Inputs x*)
- Força Sináptica → Pesos (w)
- Disparo (*Firing*) → Função de Ativação (ϕ)

O Algoritmo de Aprendizado: Correção de Erro

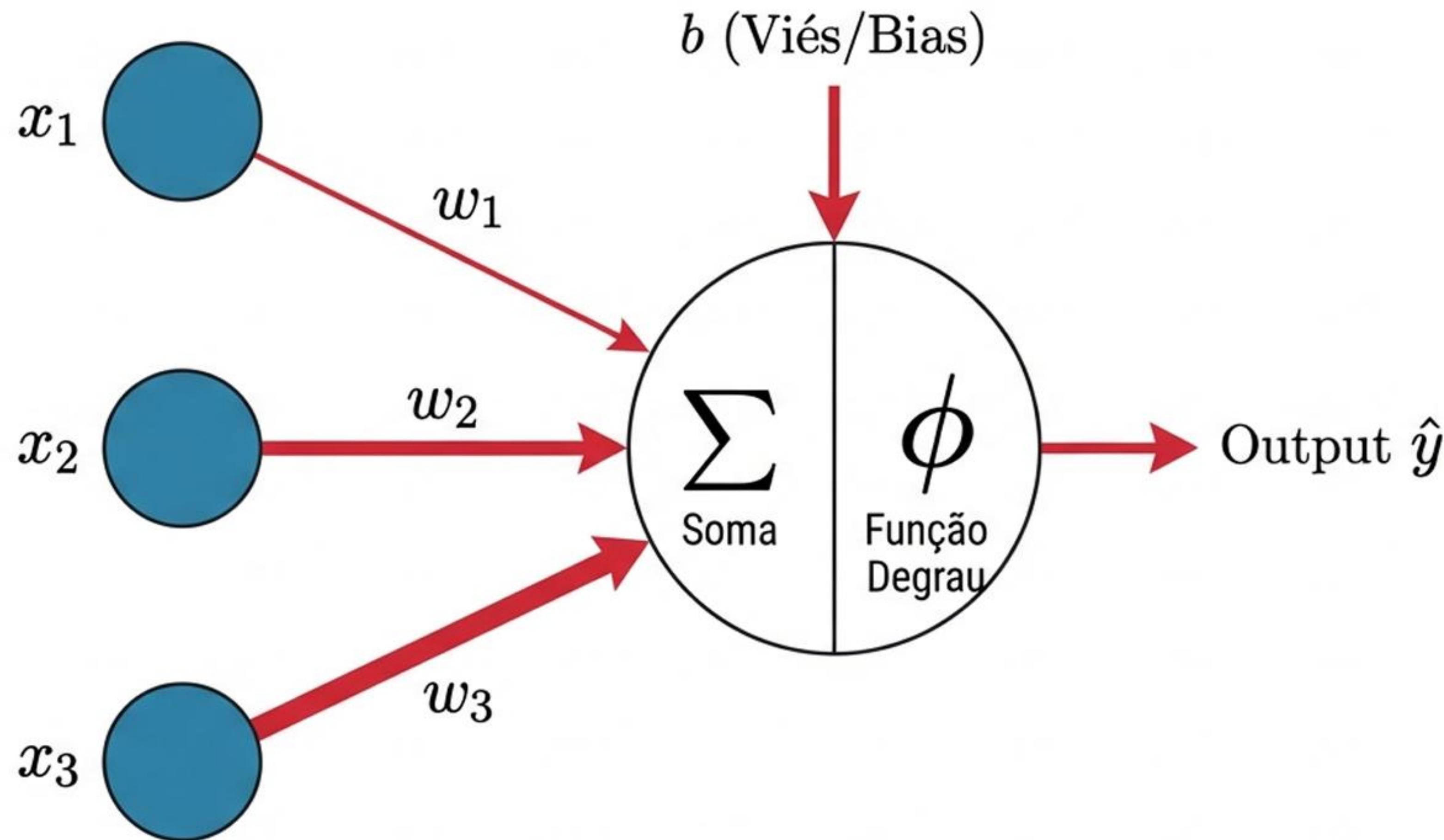


O Ciclo de Aprendizado:

1. A rede gera um sinal de saída $y_k(n)$.
2. Compara-se com a resposta desejada $d_k(n)$.
3. Gera-se um **Sinal de Erro**:
$$e_k(n) = d_k(n) - y_k(n).$$

Objetivo: Minimizar a função de custo, aproximando a saída real da desejada passo a passo.

O Perceptron: A Unidade Lógica Fundamental



**Intuição:

O neurônio é um mecanismo de votação ponderada. Ele toma uma decisão binária baseada na soma de evidências.

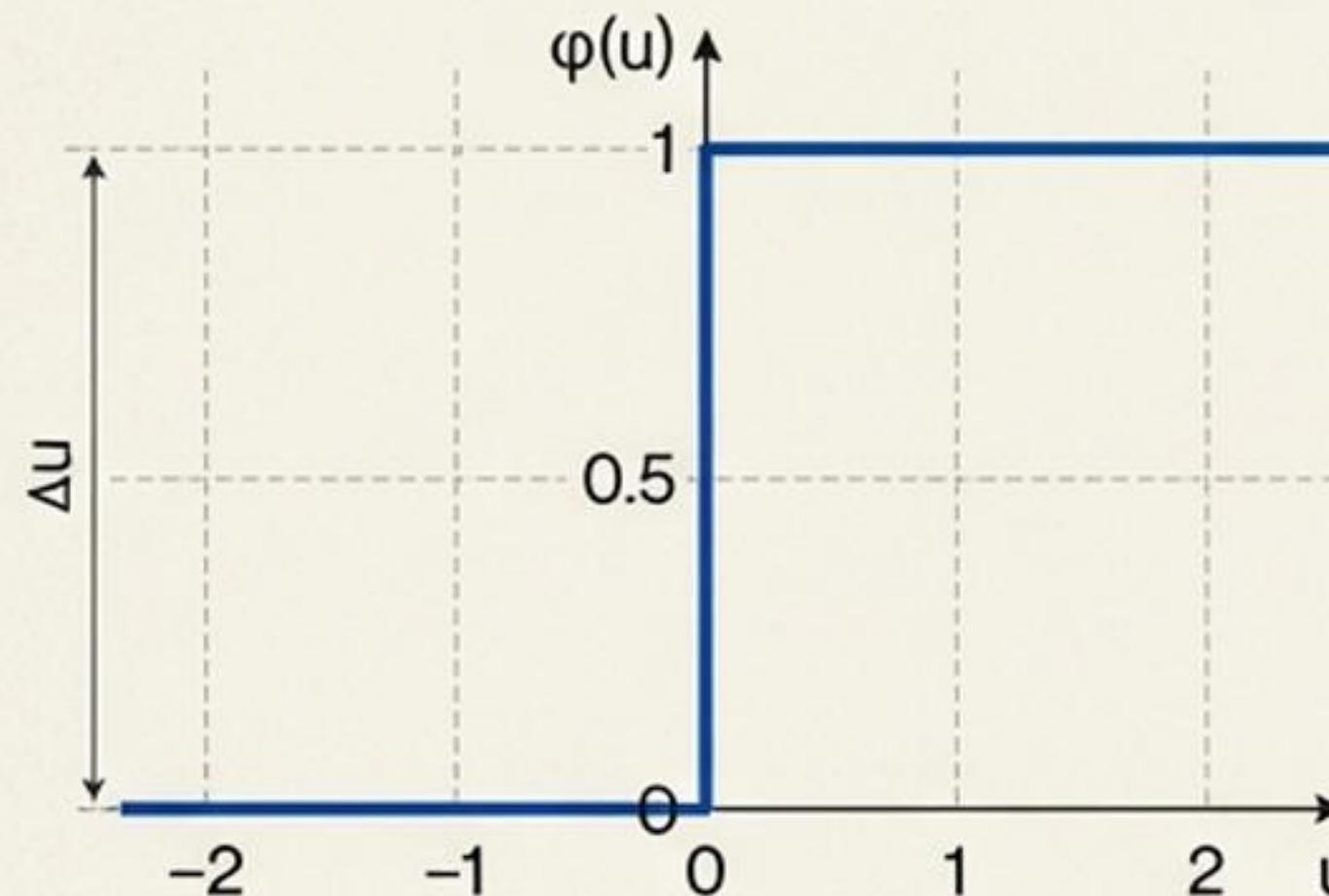
Componentes:

1. **Entradas (x):** Dados brutos.
2. **Pesos (w):** Importância de cada entrada (linha vermelha).
3. **Viés (b):** Ajuste do limiar de ativação.
4. **Soma:** Agregação linear ($\sum w \cdot x$).

Funções de Ativação: A Decisão de Disparar

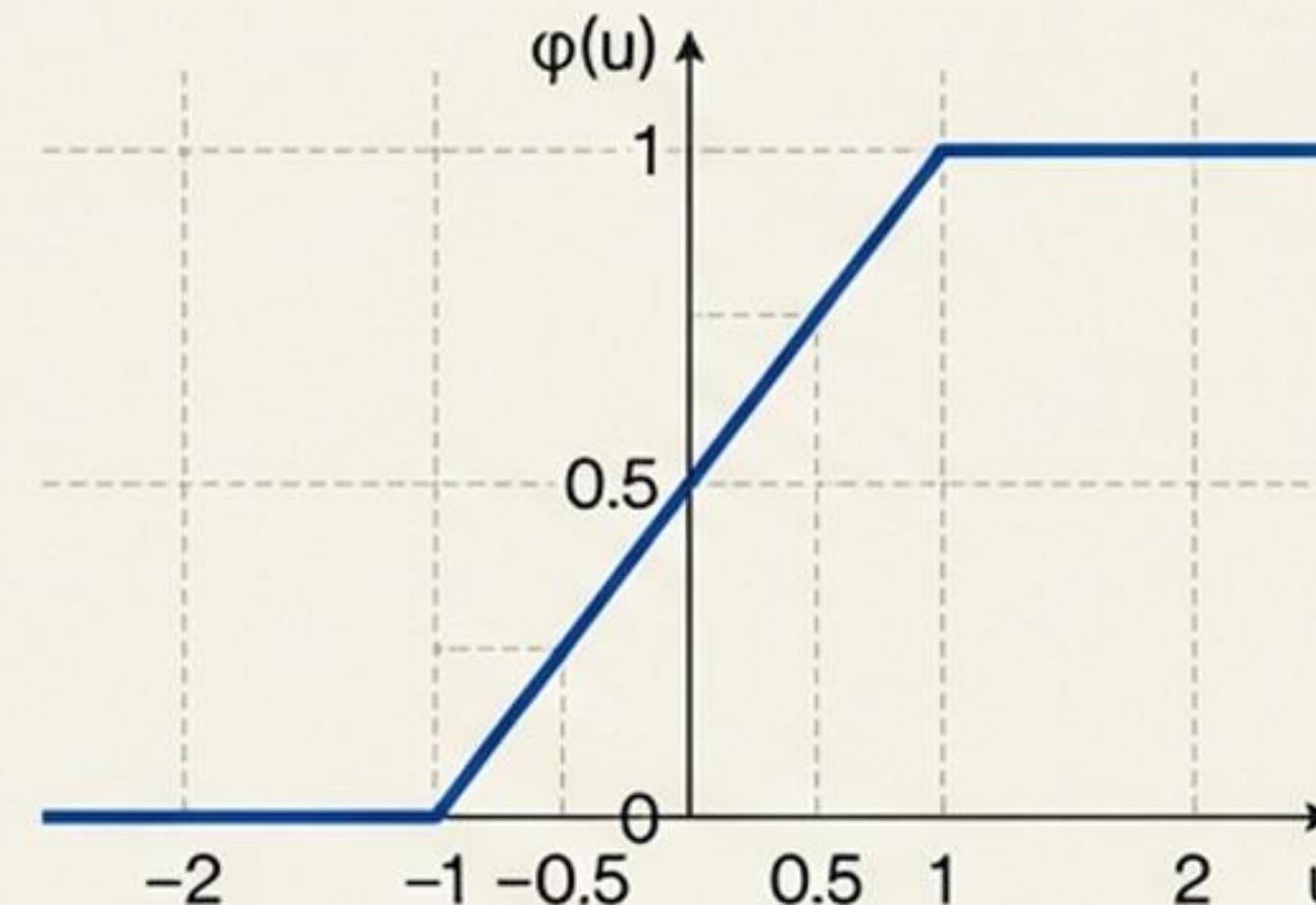
A função de ativação introduz a não-linearidade essencial para a rede.

Função Limiar (Threshold)



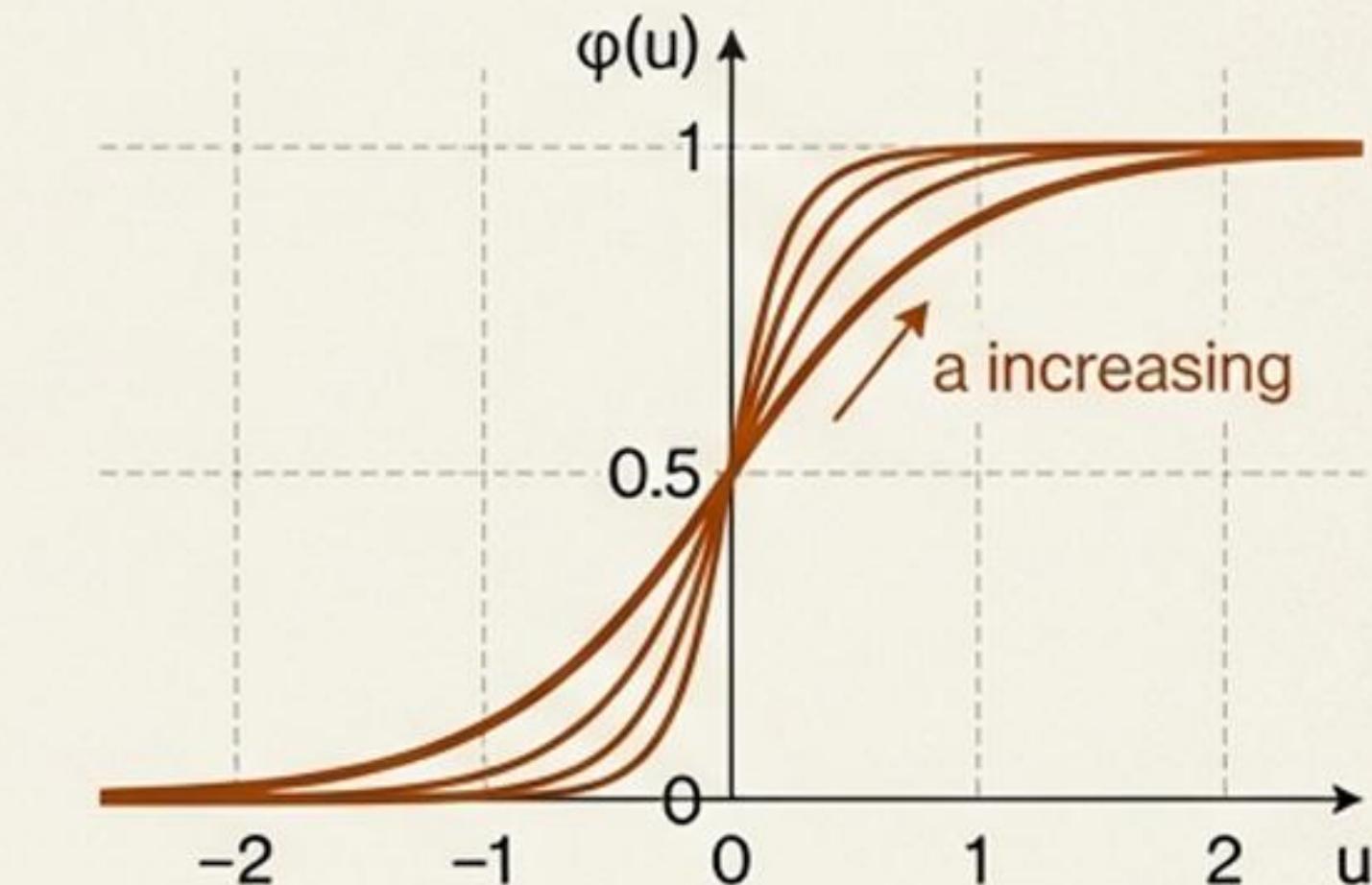
1. Limiar: Modelo McCulloch-Pitts.
Tudo ou nada (0 ou 1).

Linear por Partes



2. Linear por Partes: Introduz uma
região linear antes da saturação.

Sigmoide (Logística)



3. Sigmoide: Estritamente crescente.
Balanceia comportamento linear e
não-linear.

INSIGHT: A diferenciabilidade da função Sigmoide é crucial para a teoria das redes neurais, mas sua saturação nas pontas (valores 0 ou 1) cobrará um preço alto em redes profundas.

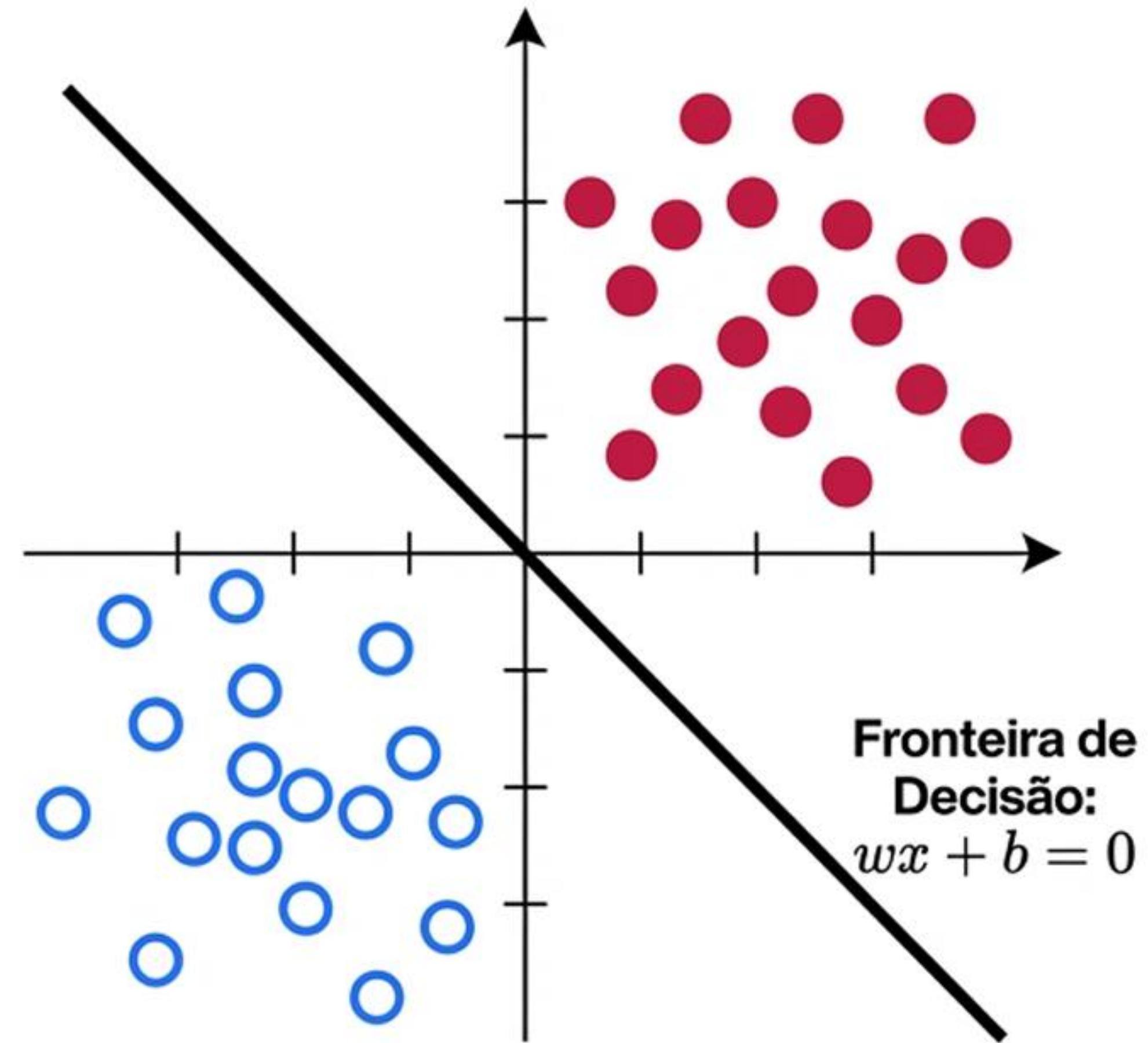
Formulação Matemática do Perceptron

1. Combição Linear (Soma Ponderada):

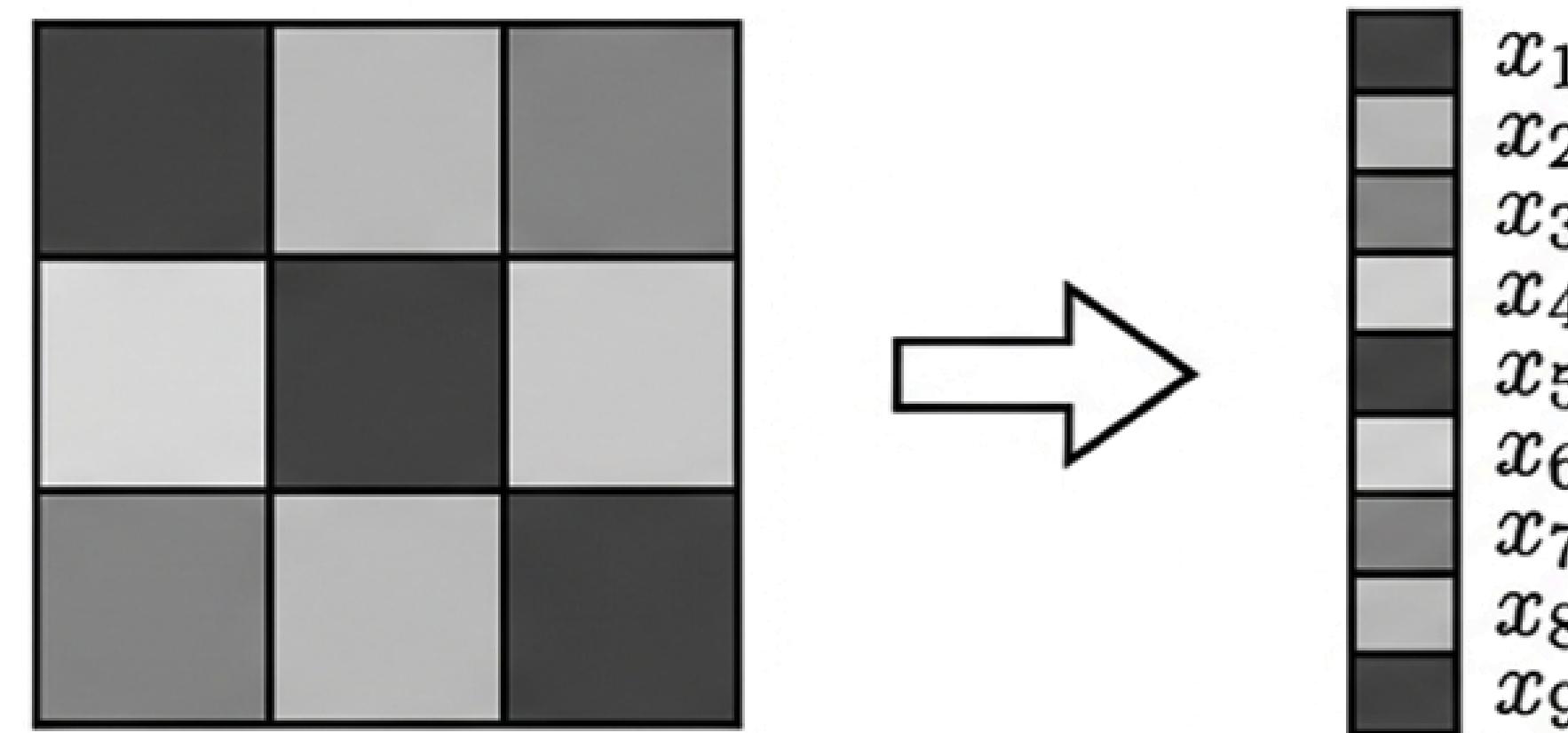
$$z = \sum_{i=1}^n w_i x_i + b$$

2. Decisão (Função de Ativação):

$$\hat{y} = \phi(z) = \begin{cases} 1 & \text{se } z \geq 0 \\ 0 & \text{se } z < 0 \end{cases}$$



Notação Vetorial e Eficiência Computacional



Flattening (Achatamento): Matriz → Vetor

Para eficiência, substituímos *loops* por Álgebra Linear.

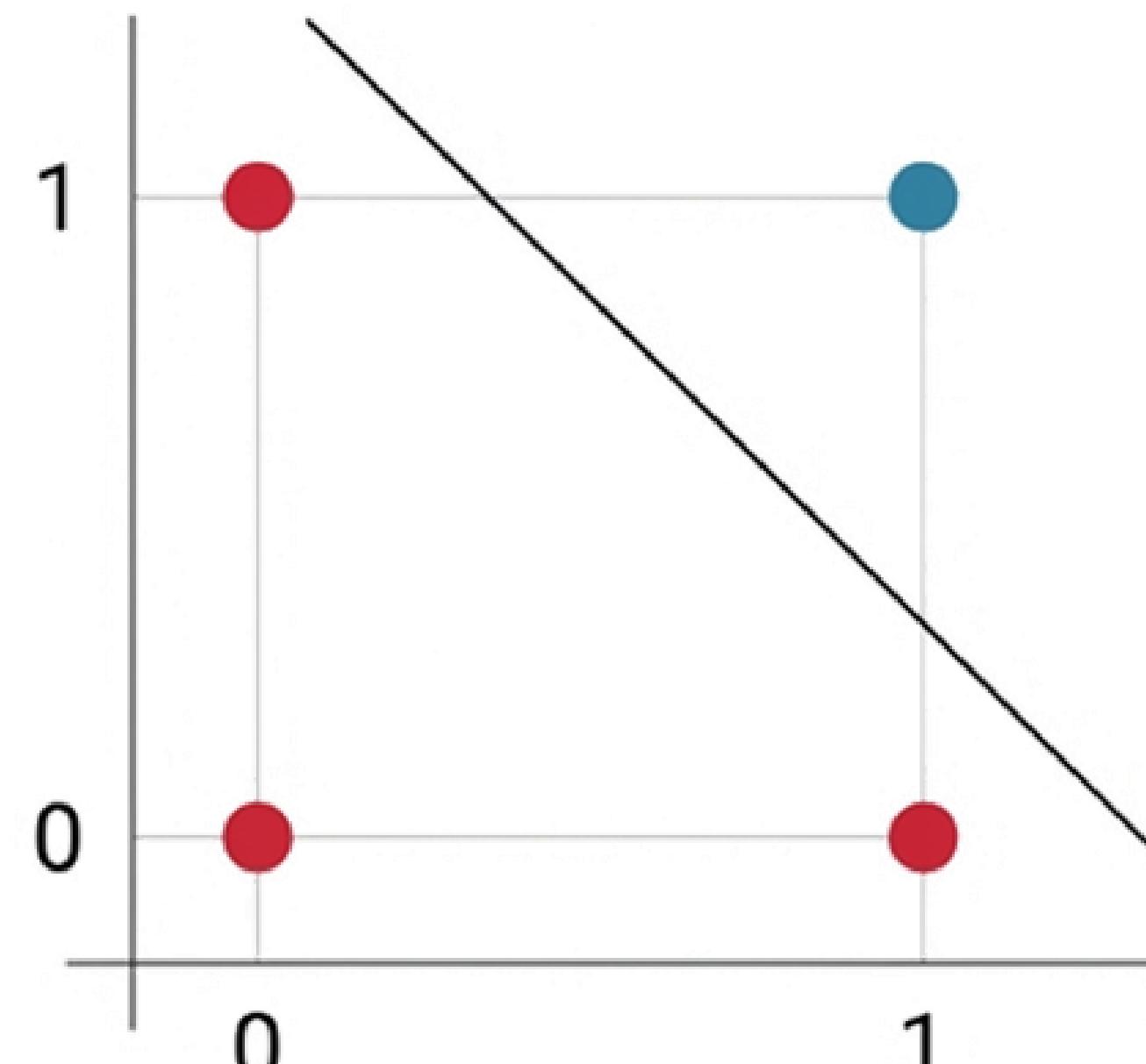
$$\mathbf{x} \text{ (Input)} \quad \mathbf{w} \text{ (Weights)}$$
$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$z = \mathbf{w}^T \mathbf{x} + b$$

O Produto Escalar (Dot Product) permite computação paralela massiva em GPUs. 

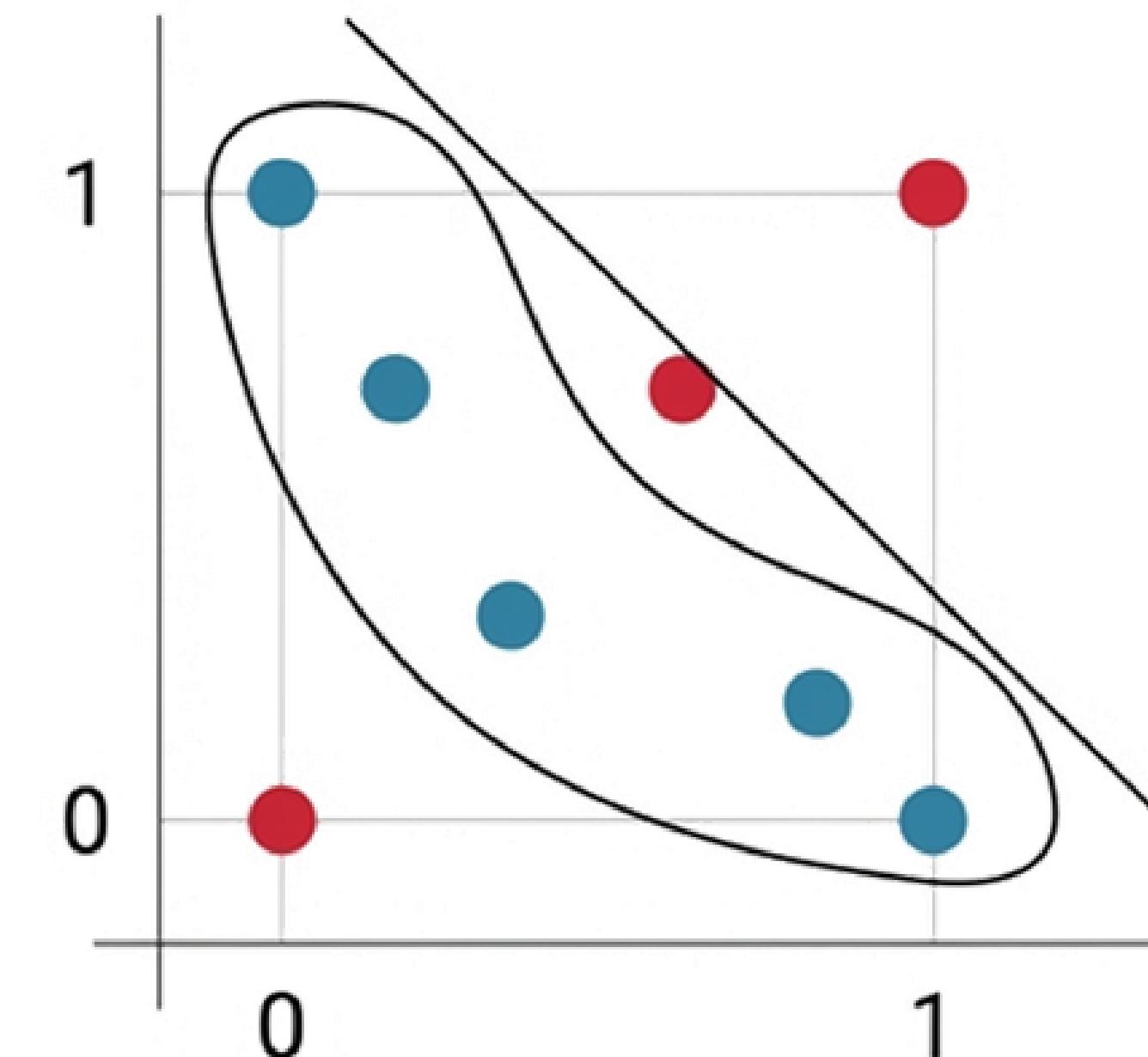
Limitações Lineares e o Problema do XOR

Problema Linear (AND)



Linearmente Separável

Problema Não-Linear (XOR)



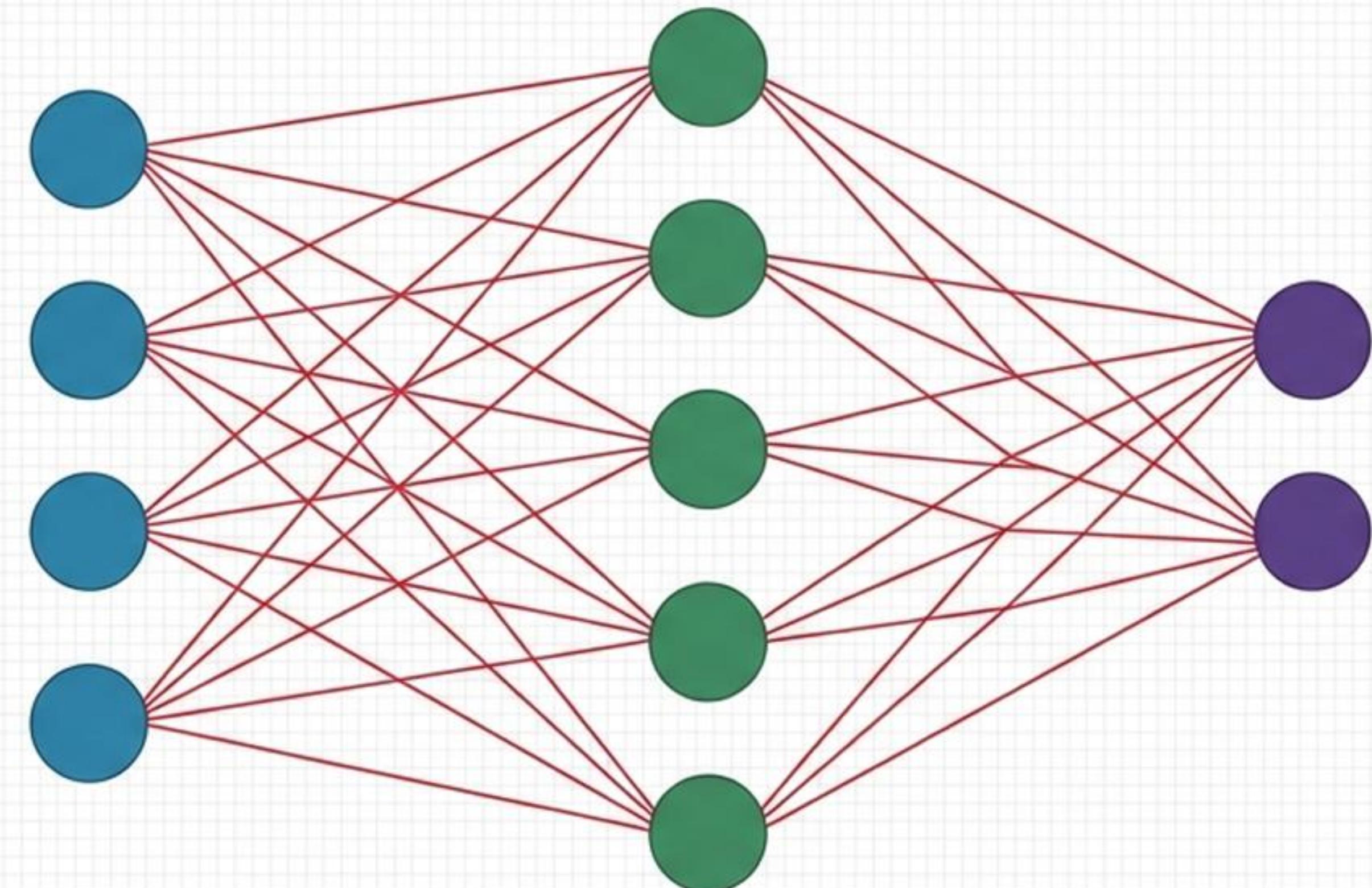
Não Linearmente Separável (XOR)

O Obstáculo: Um único Perceptron só desenha retas. Ele falha em lógica simples como 'Ou Exclusivo' (XOR).

A Solução: Empilhar neurônios em camadas (MLP) para criar superfícies de decisão complexas.

Multilayer Perceptron (MLP): Arquitetura

****Input Layer****
(Entrada - Vetor de Características)



****Hidden Layer****
(Camada Oculta - Extração de Features)

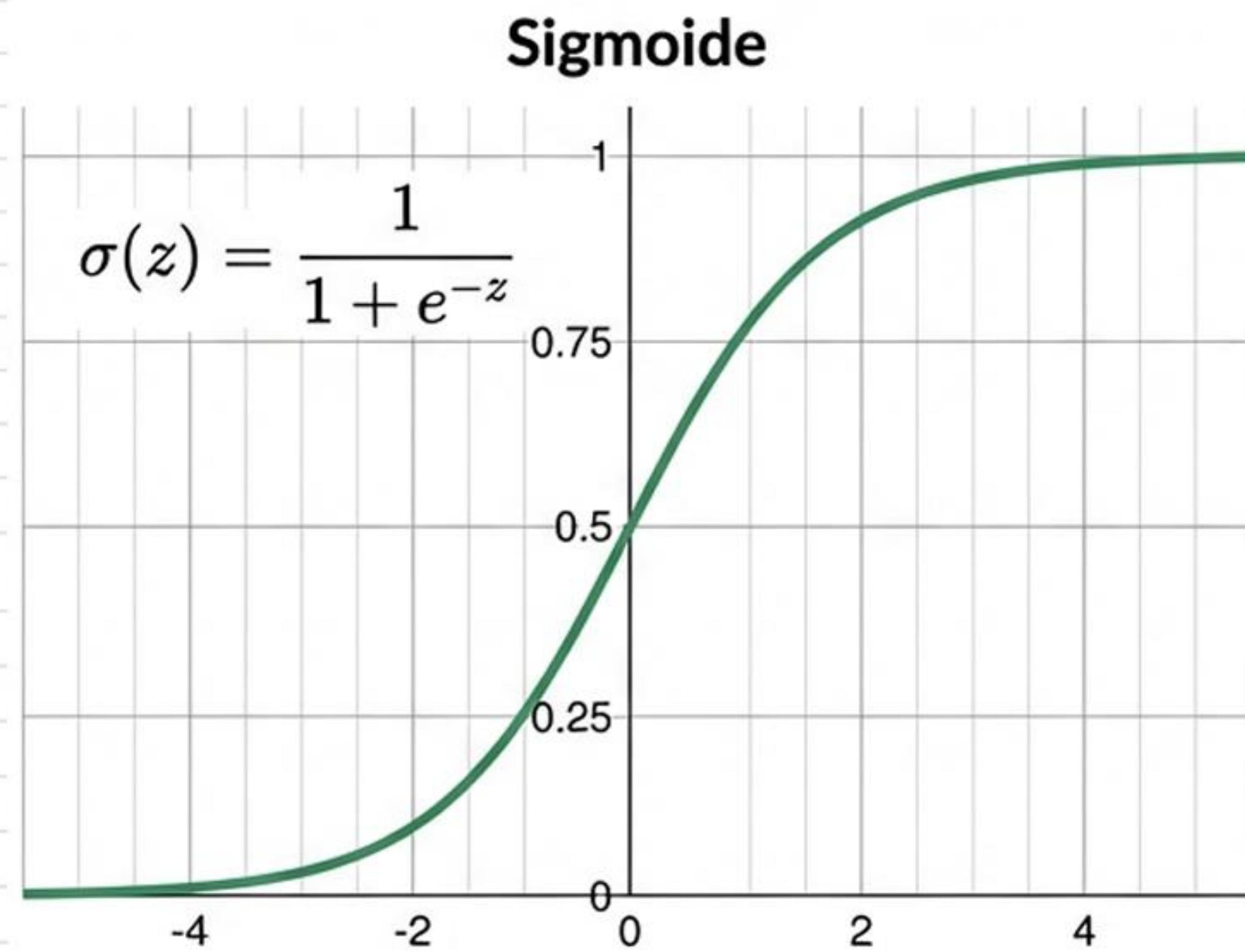
****Output Layer****
(Saída - Predição)

- * **Estrutura Densa (Fully Connected):**

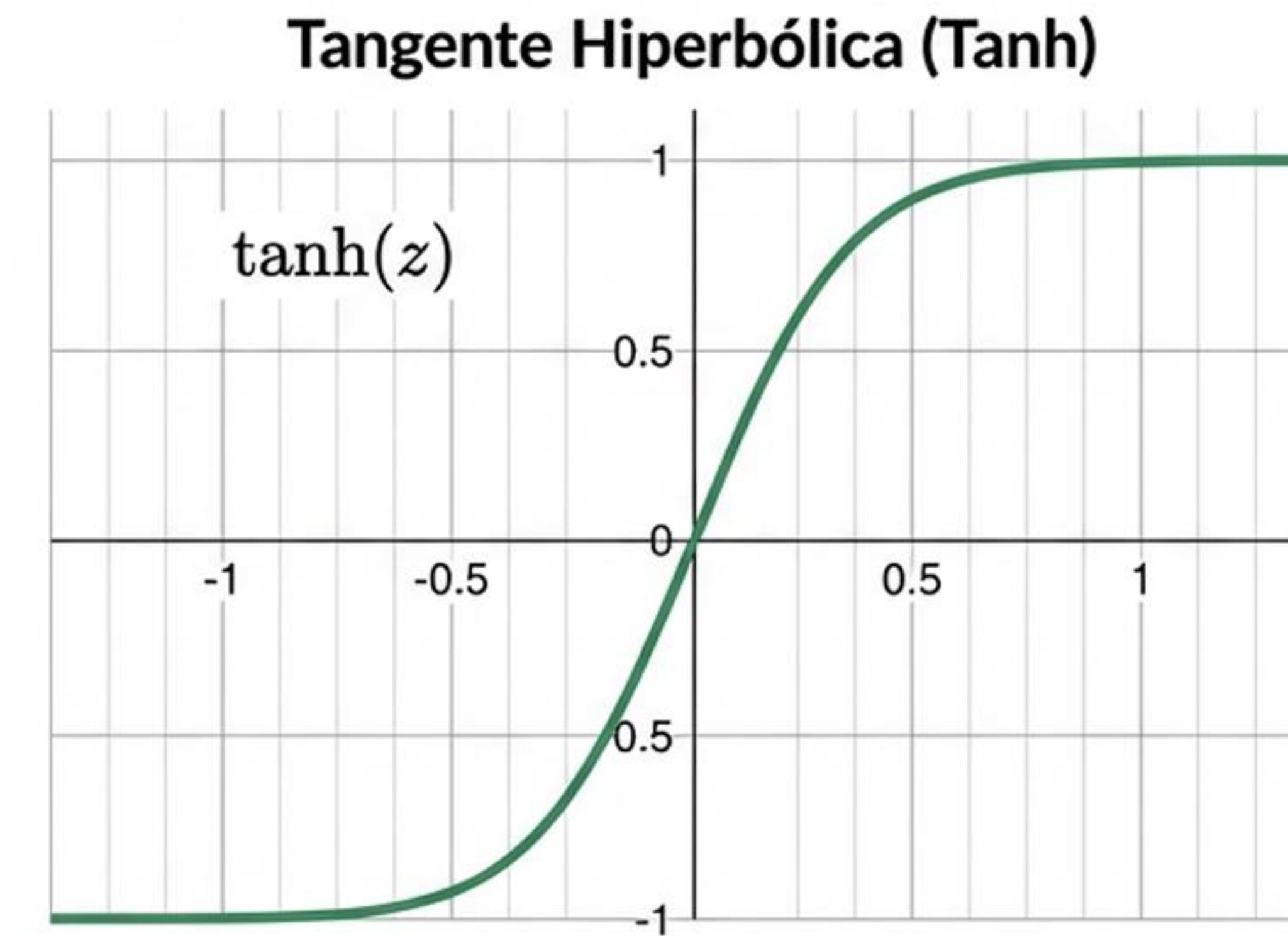
- * **Deep Learning:** Refere-se a redes com múltiplas camadas ocultas empilhadas.
- * **Hierarquia:** Camadas iniciais detectam padrões simples; camadas profundas combinam esses padrões em conceitos abstratos.

Funções de Ativação: A Fonte da Não-Linearidade

Sem funções de ativação (ϕ), uma rede neural seria apenas uma regressão linear gigante.
 $\text{Linear}(\text{Linear}(x)) = \text{Linear}(x)$ A ativação quebra a linearidade.



Output: $(0, 1)$ - Interpretação probabilística.



Output: $(-1, 1)$ - Centrada em zero.

Ativações Modernas: ReLU e Softmax

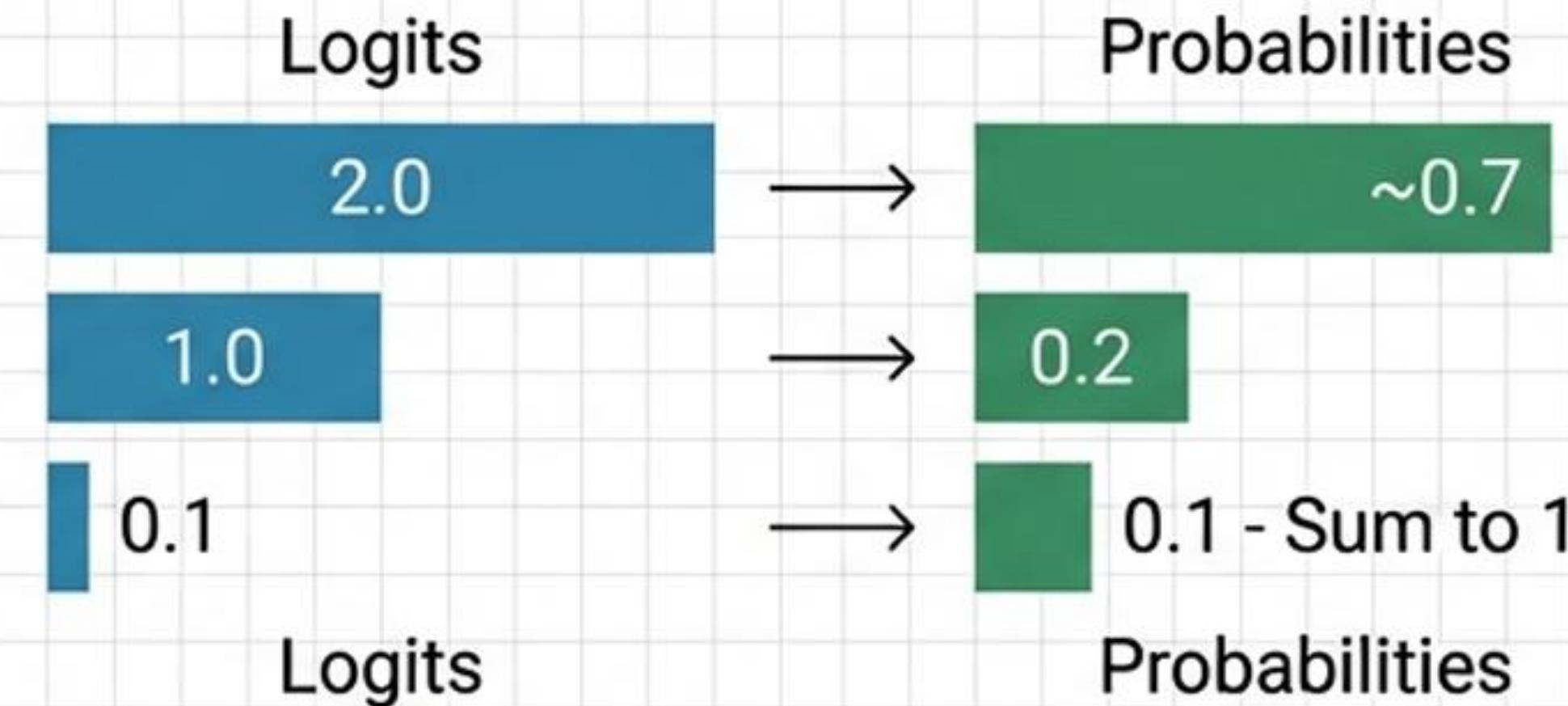
ReLU (Rectified Linear Unit): Padrão da indústria para camadas ocultas.



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

Eficiência computacional e evita desaparecimento de gradiente.

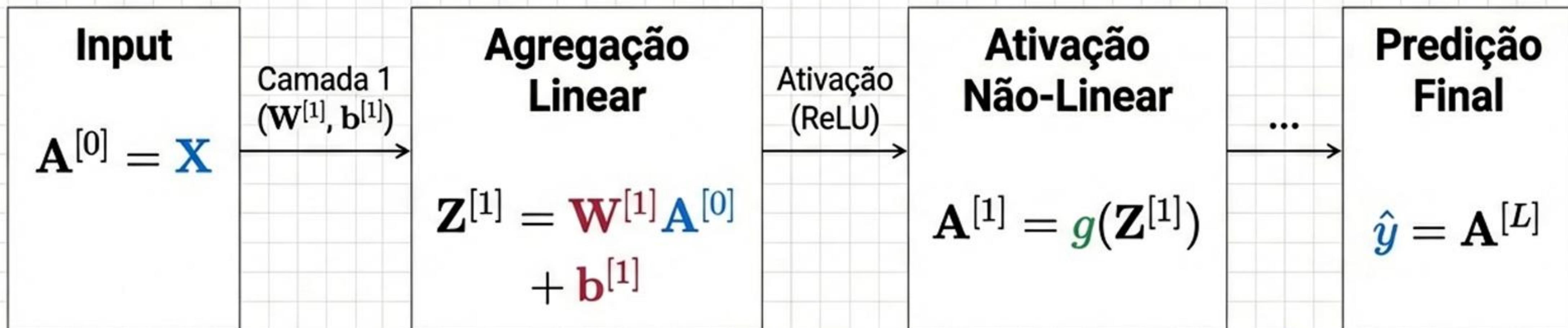
Softmax: Exclusiva para Camada de Saída (Classificação Multiclasse).



$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

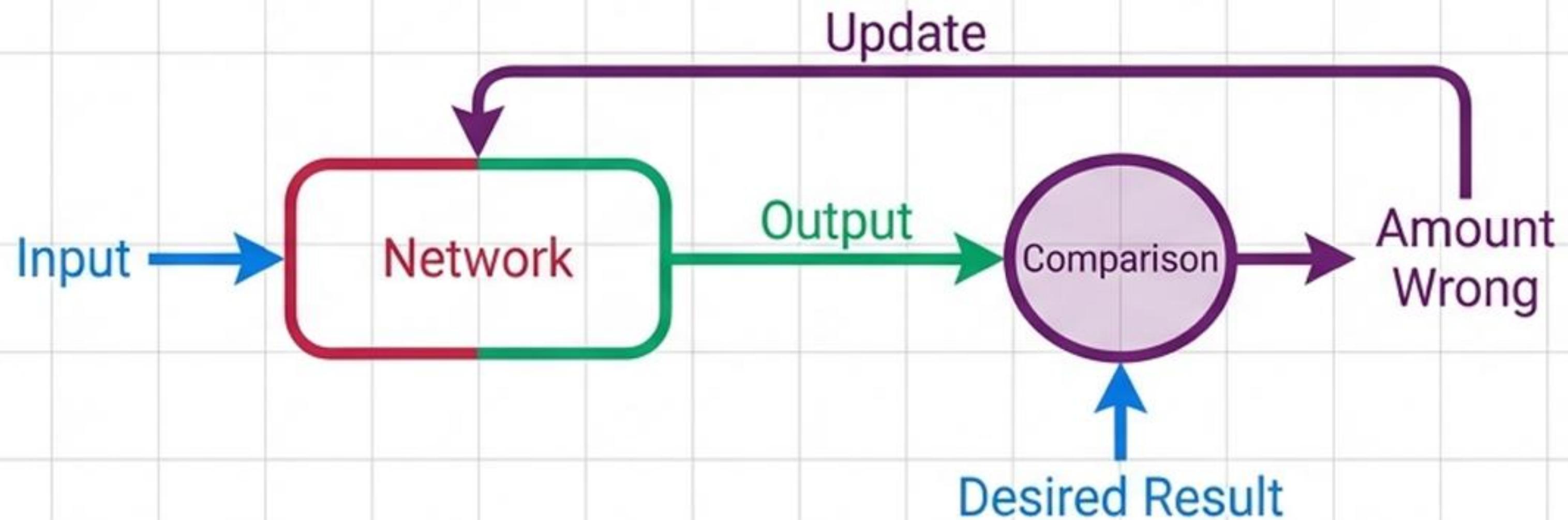
Transforma scores brutos em distribuição de probabilidade.

Etapa Forward: Propagação Matricial por Camadas



Onde [l] representa o índice da camada.

Funções de Perda (Loss Functions)



Definição

Quantifica a distância entre a predição da rede (\hat{y}) e o valor real (y). O objetivo do treino é minimizar \mathcal{L} .

MSE (Regressão)

Erro Quadrático Médio. Para prever valores contínuos.

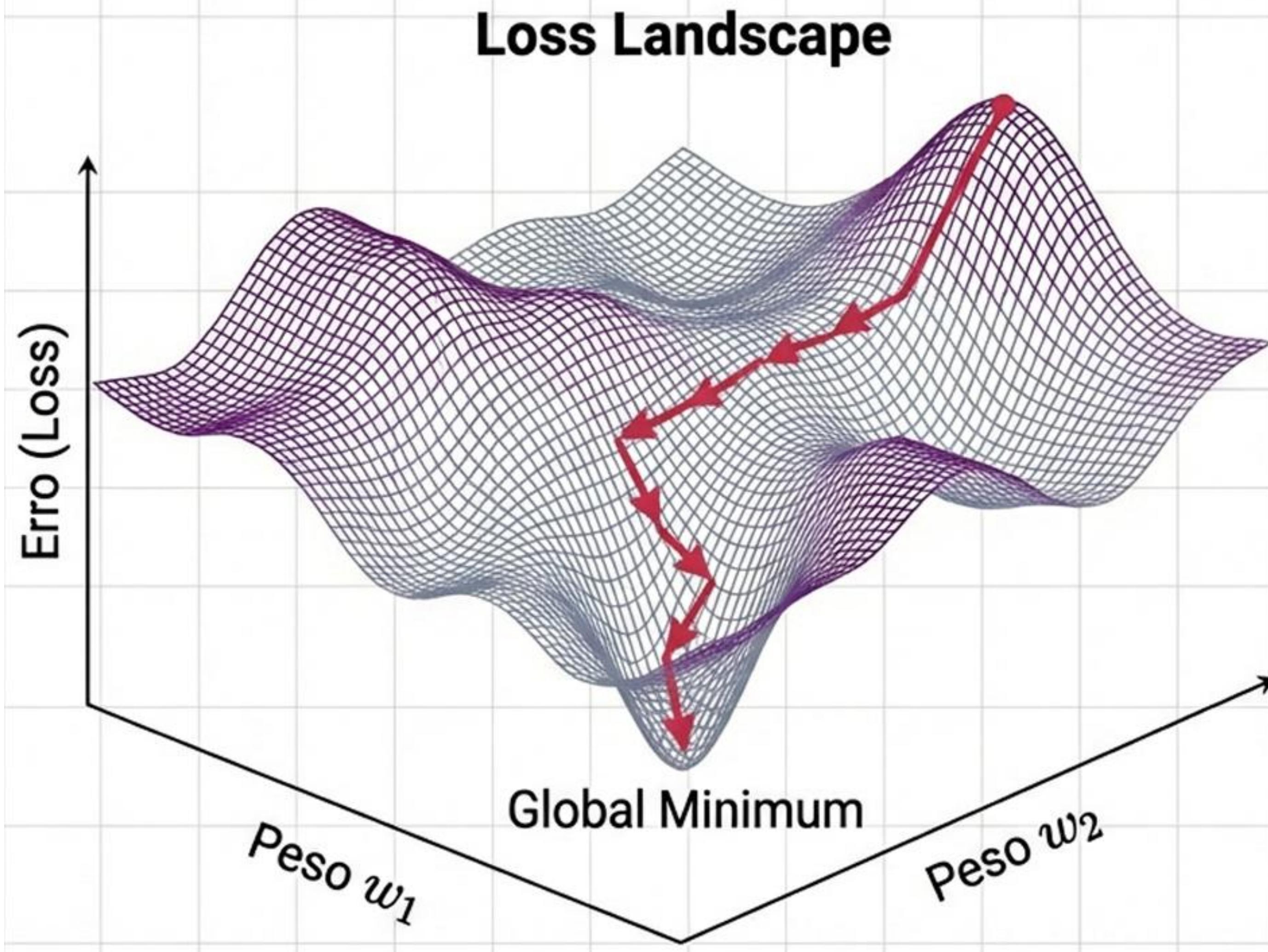
$$\mathcal{L} = \frac{1}{n} \sum (y - \hat{y})^2$$

Cross-Entropy (Classificação)

Entropia Cruzada. Para categorização.

$$\mathcal{L} = - \sum y \log(\hat{y})$$

Otimização: O Gradiente Descendente



Descendo a Montanha:

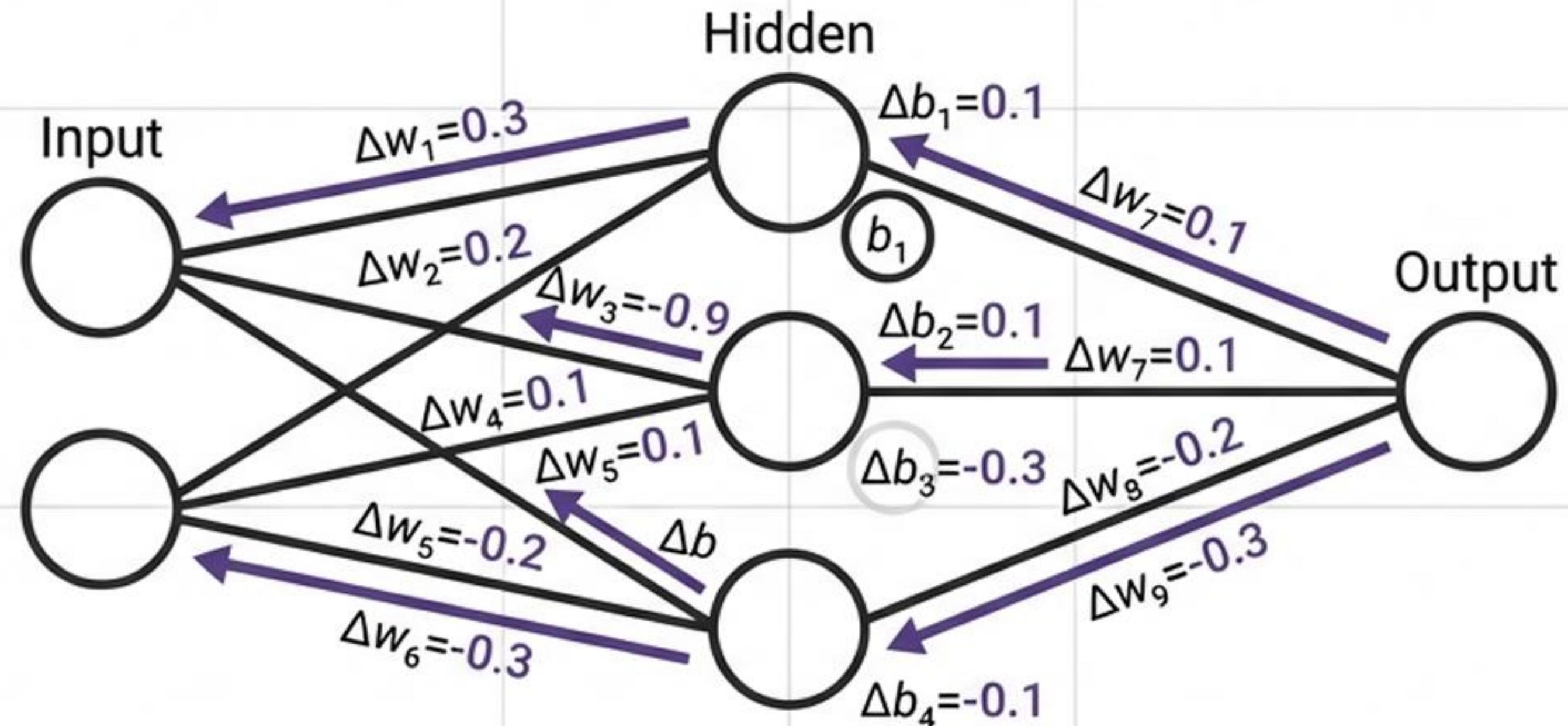
Calculamos a inclinação (gradiente) e damos um passo na direção oposta para reduzir o erro.

Regra de Atualização:

$$\theta_{novo} = \theta_{antigo} - \eta \cdot \nabla J(\theta)$$

- * θ : Parâmetros (w, b)
- * ∇J : Gradiente (Direção da subida)
- * η : Taxa de Aprendizado (Tamanho do passo)

Backpropagation: A Regra da Cadeia



****Como atribuir “culpa” ao erro?****

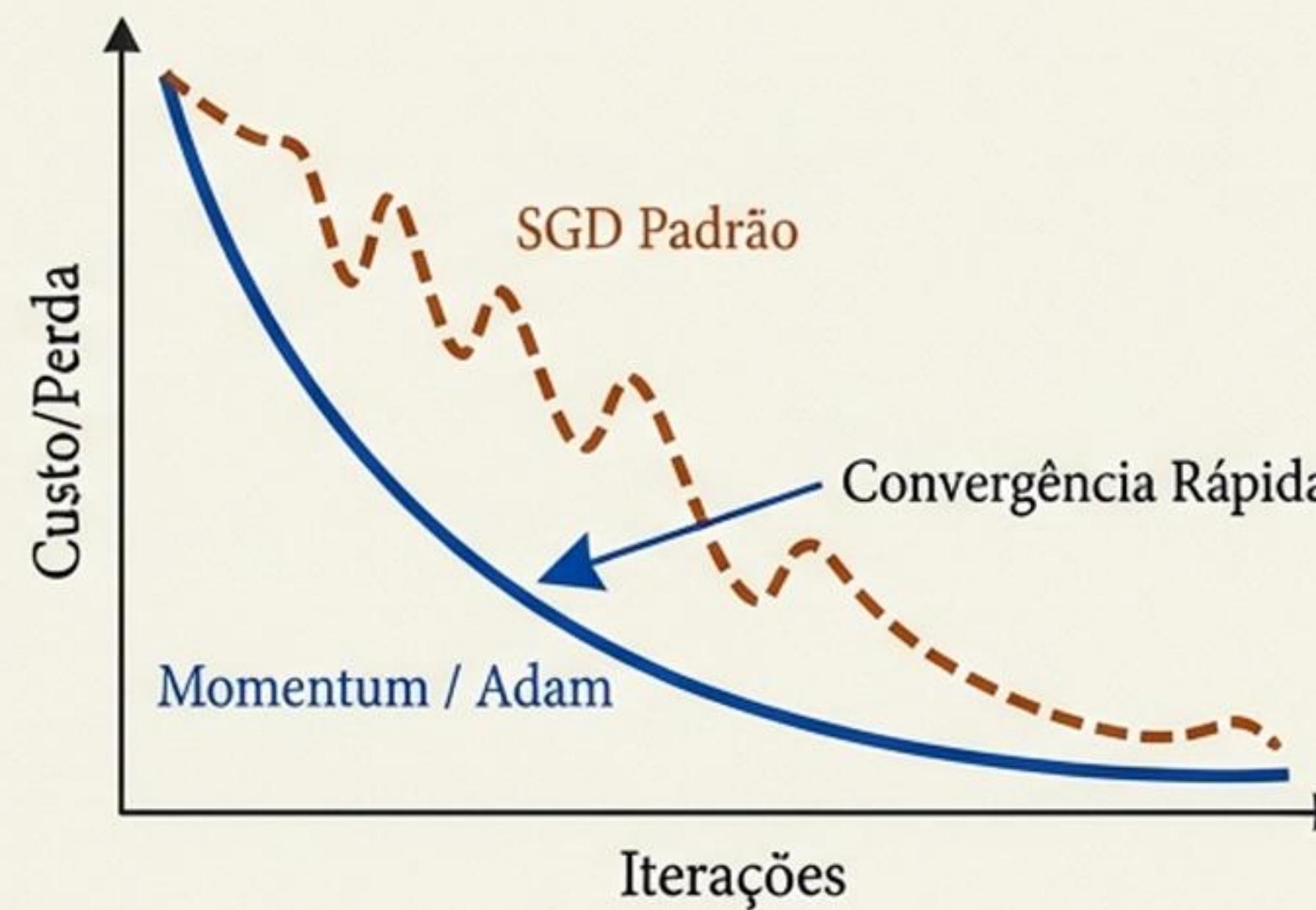
Propagamos o gradiente da saída para a entrada usando a Regra da Cadeia (Cálculo).

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

Erro → Derivada da Ativação → Entrada do Neurônio.

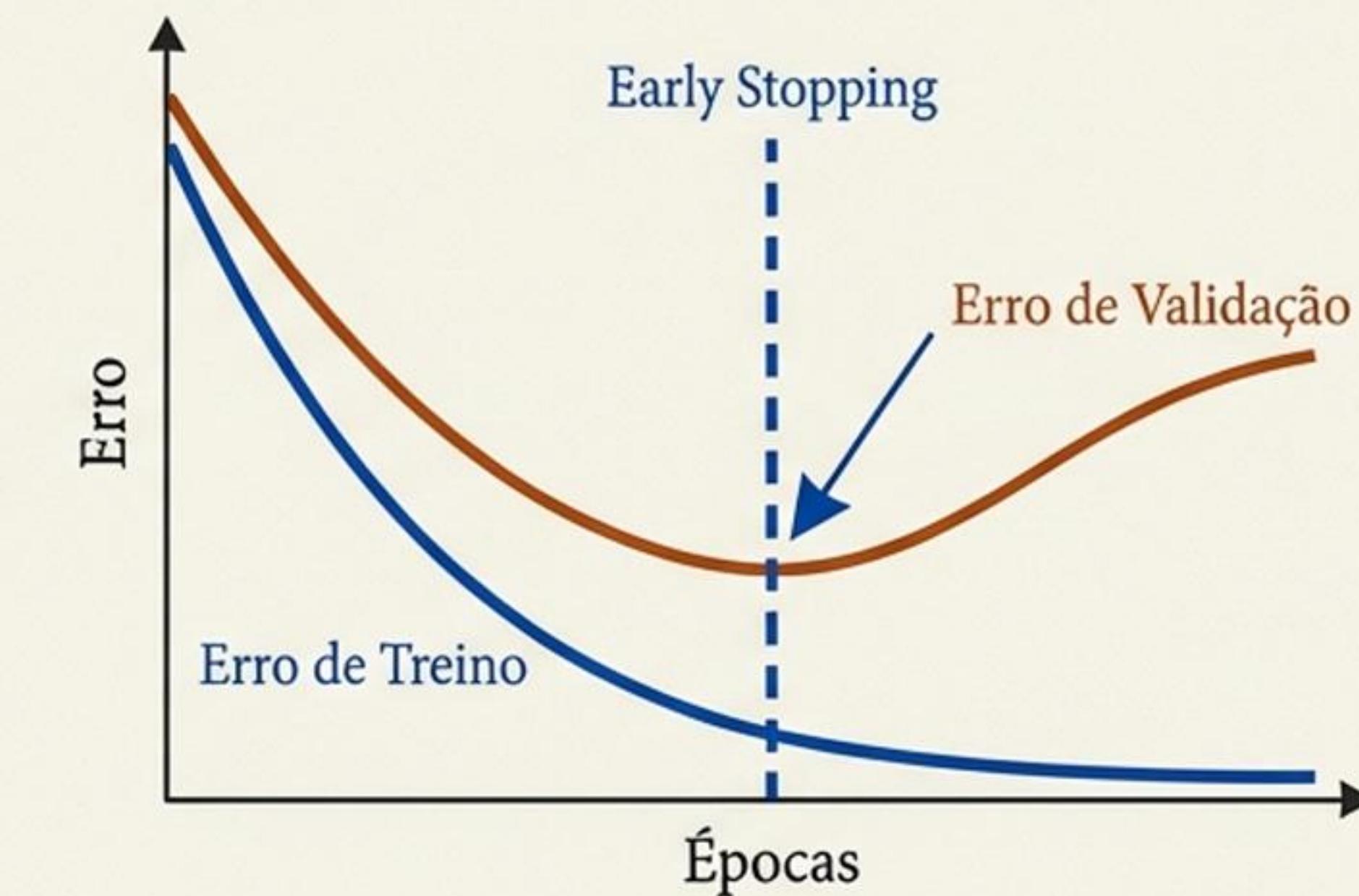
O Arsenal Moderno: Otimizadores e Regularização

Acelerando a Convergência



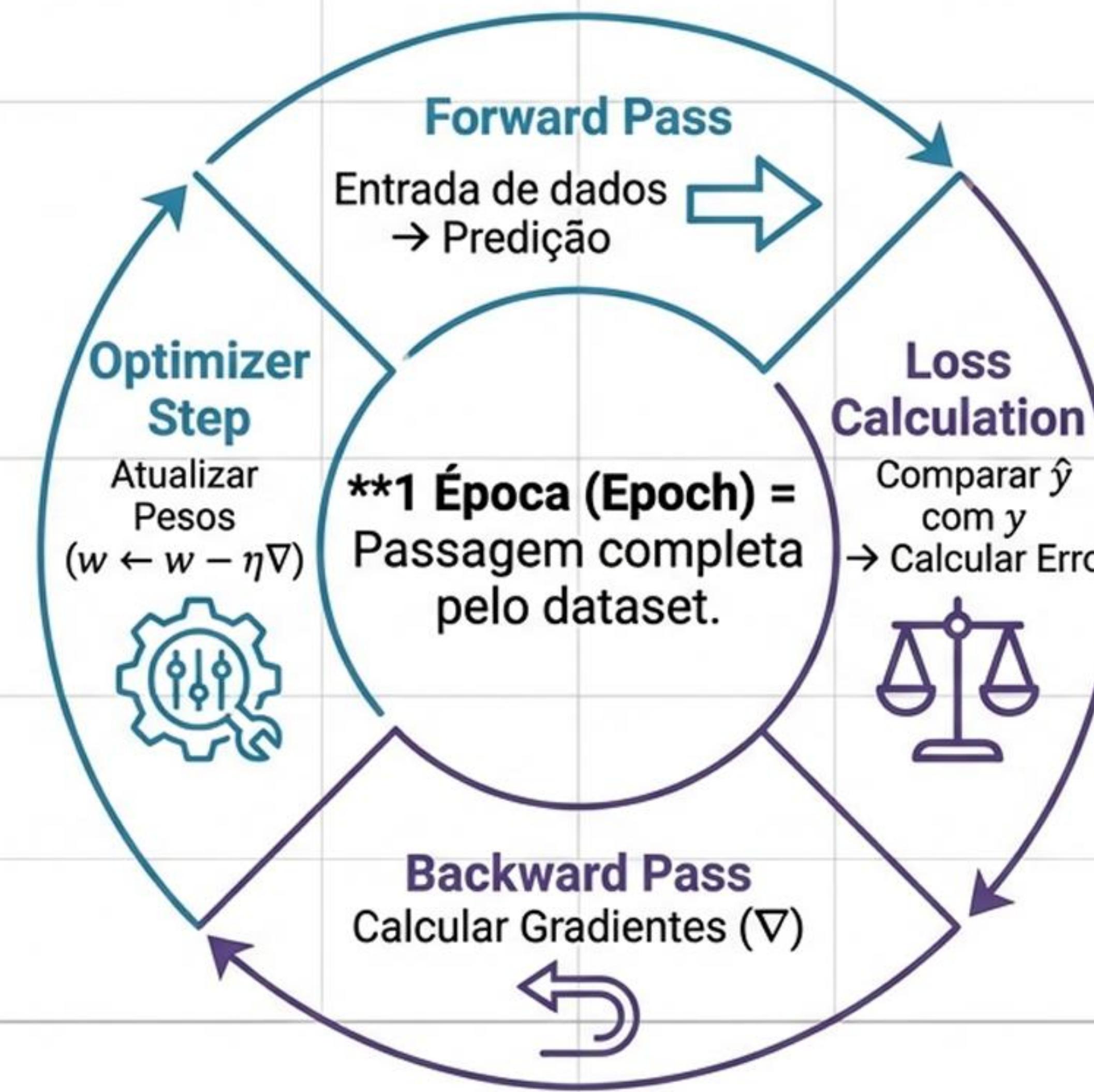
Otimizadores modernos aceleram drasticamente o aprendizado.

Evitando Overfitting



Early Stopping e Regularização penalizam a complexidade excessiva para garantir generalização.

O Ciclo de Treinamento (Epochs)



Conclusão: A Emergência da Inteligência

Resumo Estrutural:

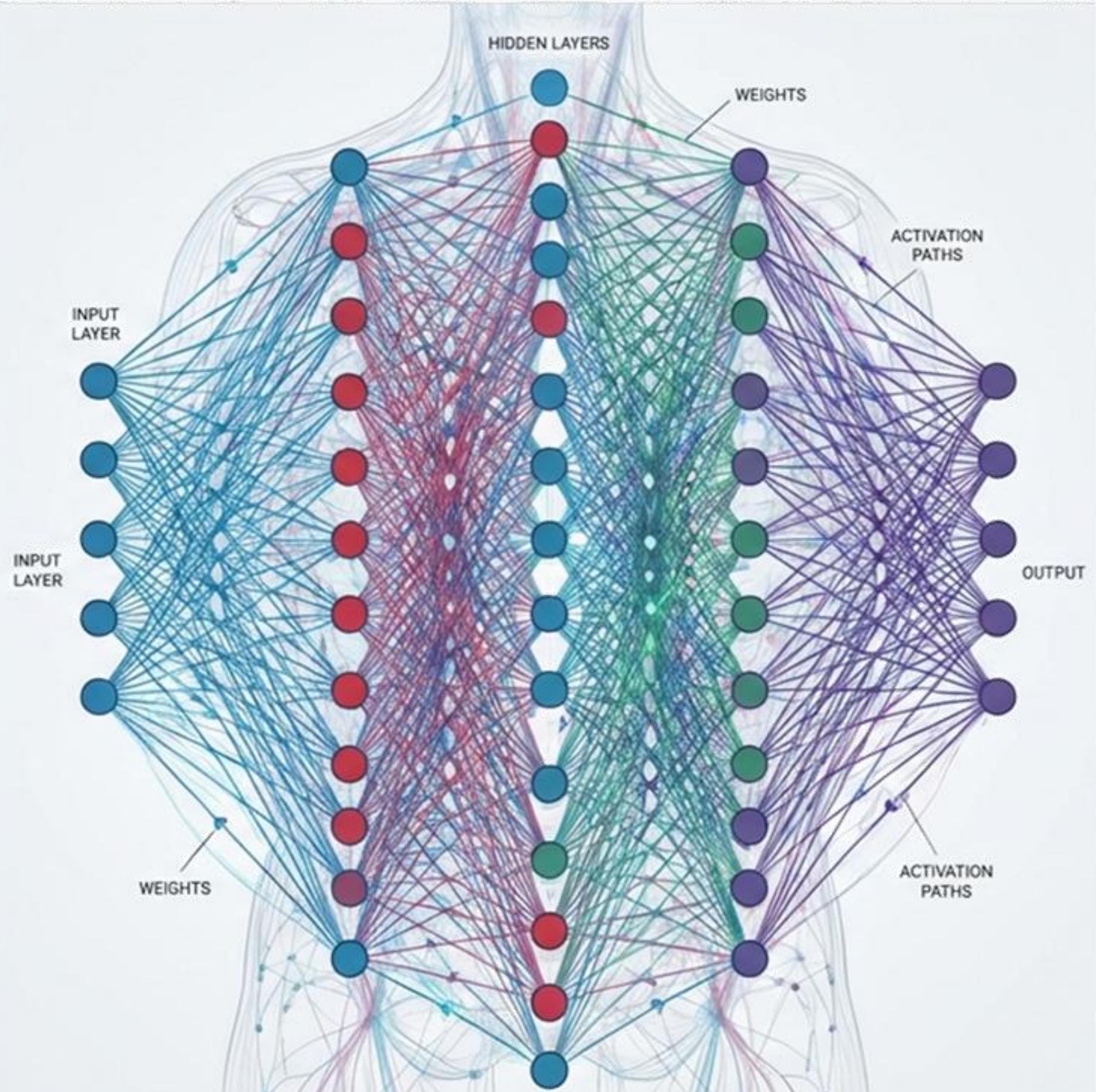
- Perceptron: A célula base. Classificador linear simples.
- MLP: A arquitetura. Camadas ocultas e não-linearidade (ReLU) permitem modelar qualquer função complexa.

O Segredo:

A "inteligência" não é mágica. Ela emerge da otimização iterativa (Gradient Descent) de milhões de parâmetros simples (somas e multiplicações), guiados por dados.

Próximos Passos:

CNNs (Visão), RNNs/Transformers
(Linguagem).





IBMEC.BR

 /IBMEC

 IBMEC

 @IBMEC_OFICIAL

 @IBMEC

