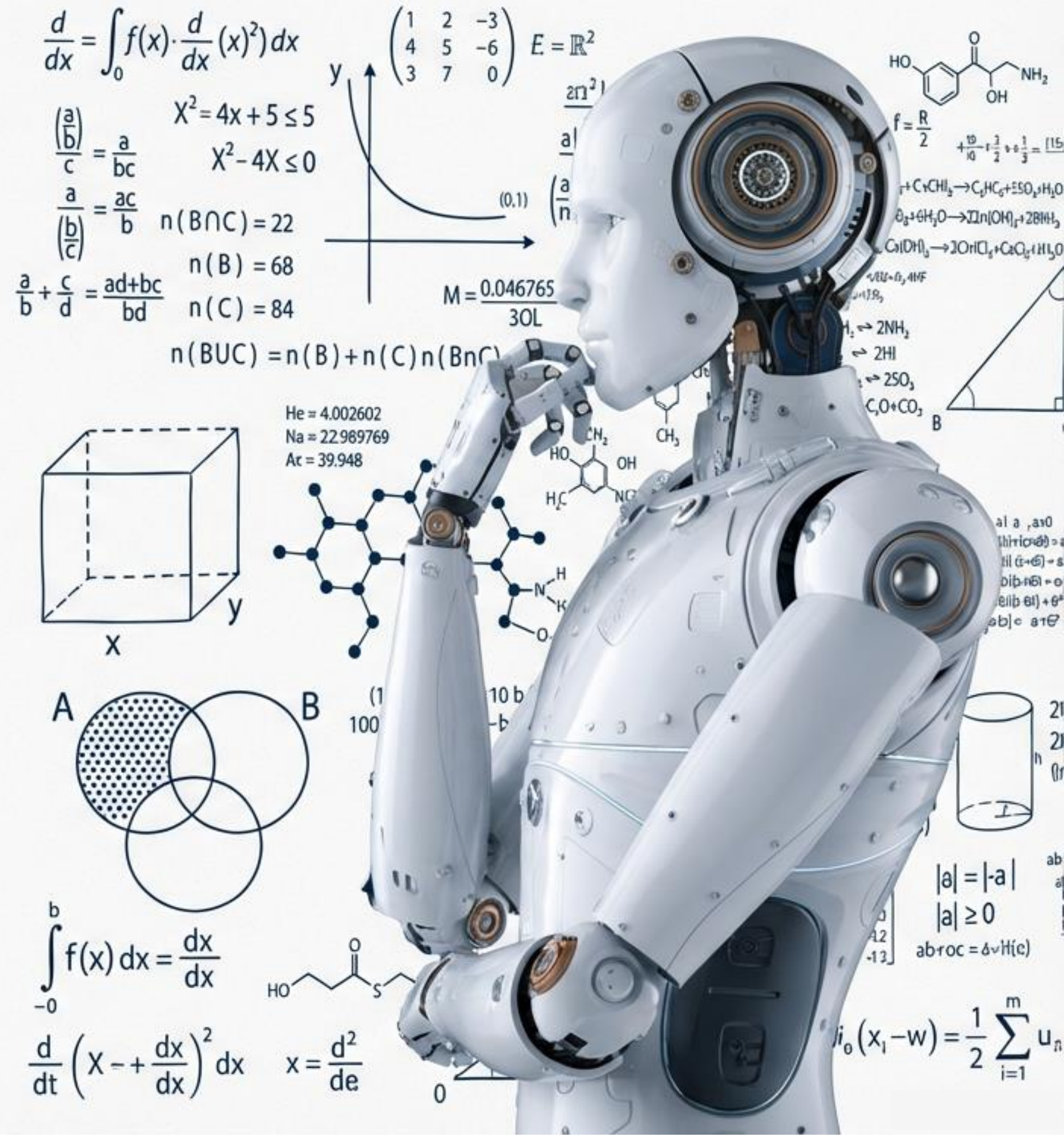


# Aprendizado de Máquina: Dos Fundamentos às Redes Profundas

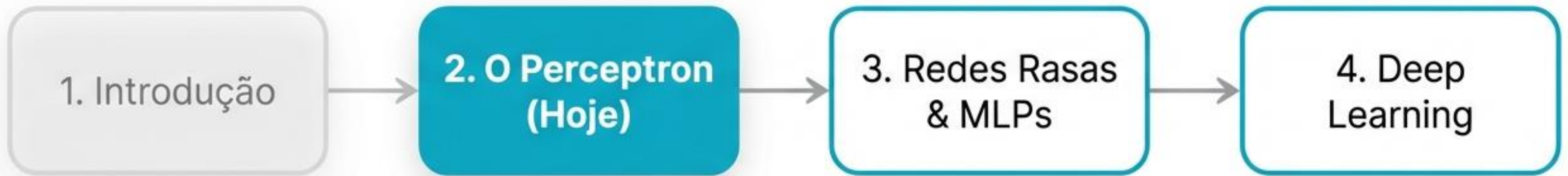
## Aula 2

Pontifícia Universidade Católica de São Paulo | Ciência de Dados e Inteligência Artificial  
Prof. Dr. Rooney R. A. Coelho





# O Caminho do Aprendizado: Onde Estamos



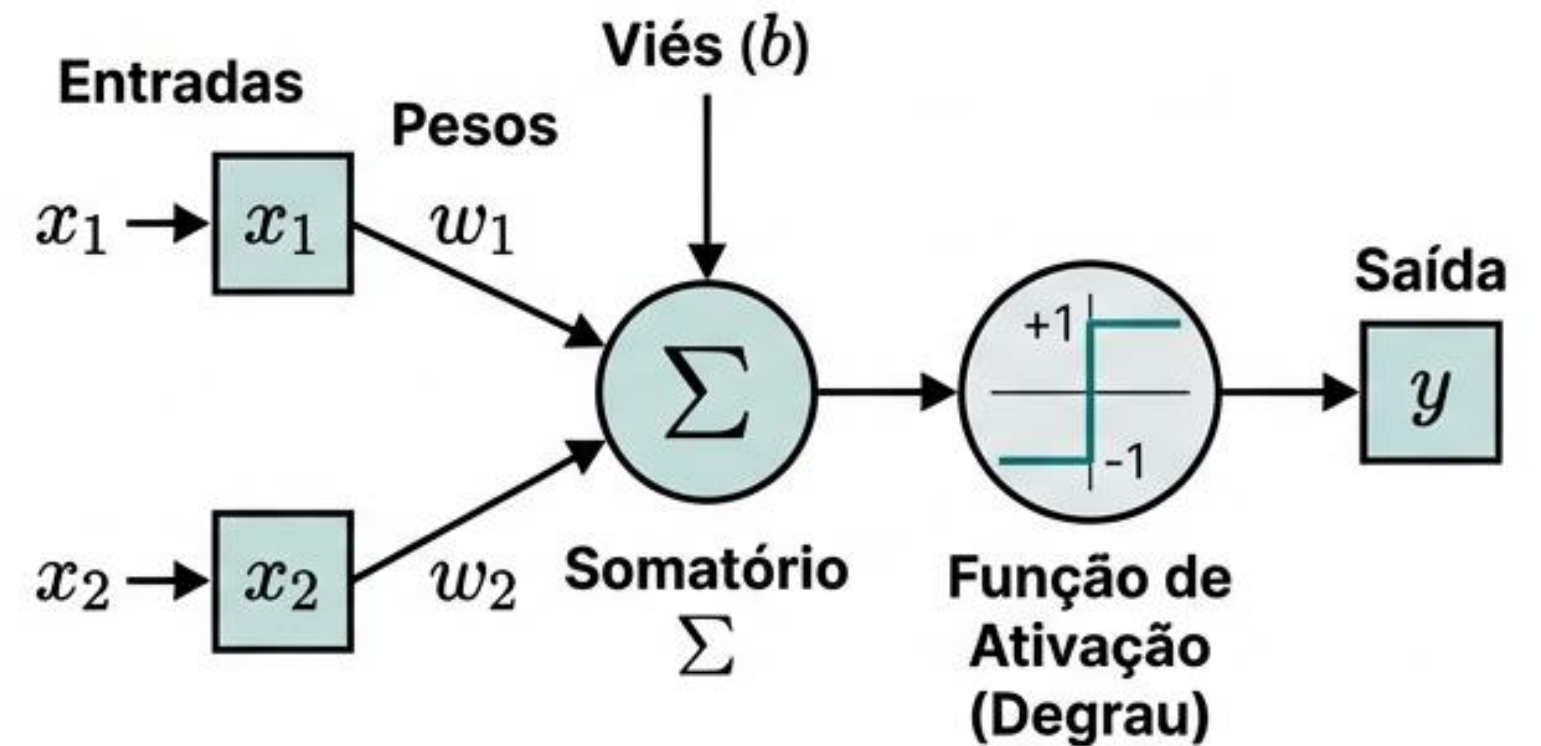
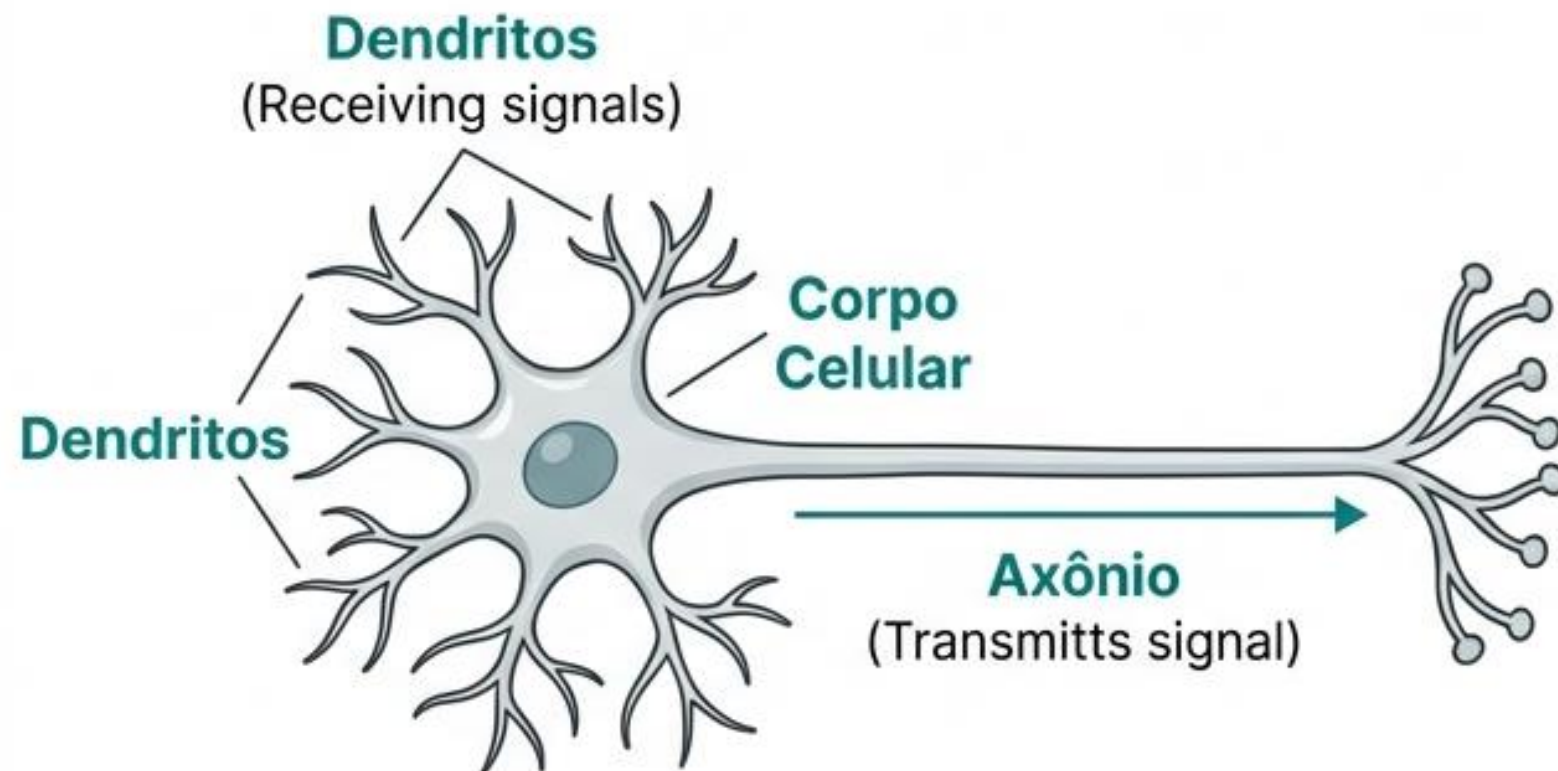
## Objetivo da Aula

Entender o funcionamento matemático e as limitações do neurônio artificial básico. É o estudo da 'unidade atômica' da IA.

## Por que isso importa

O Perceptron é o bloco construtor fundamental. Sem dominá-lo, é impossível compreender arquiteturas avançadas como Redes Convolucionais ou Transformadores.

# Do Biológico ao Matemático



**Combinador Linear:** Soma ponderada das entradas ( $x \cdot w$ ).

**Viés ( $b$ ):** Ajuste externo que desloca a decisão.

**Limitador Rígido:** Função de ativação (Degrau) que define a classe (+1 ou -1).

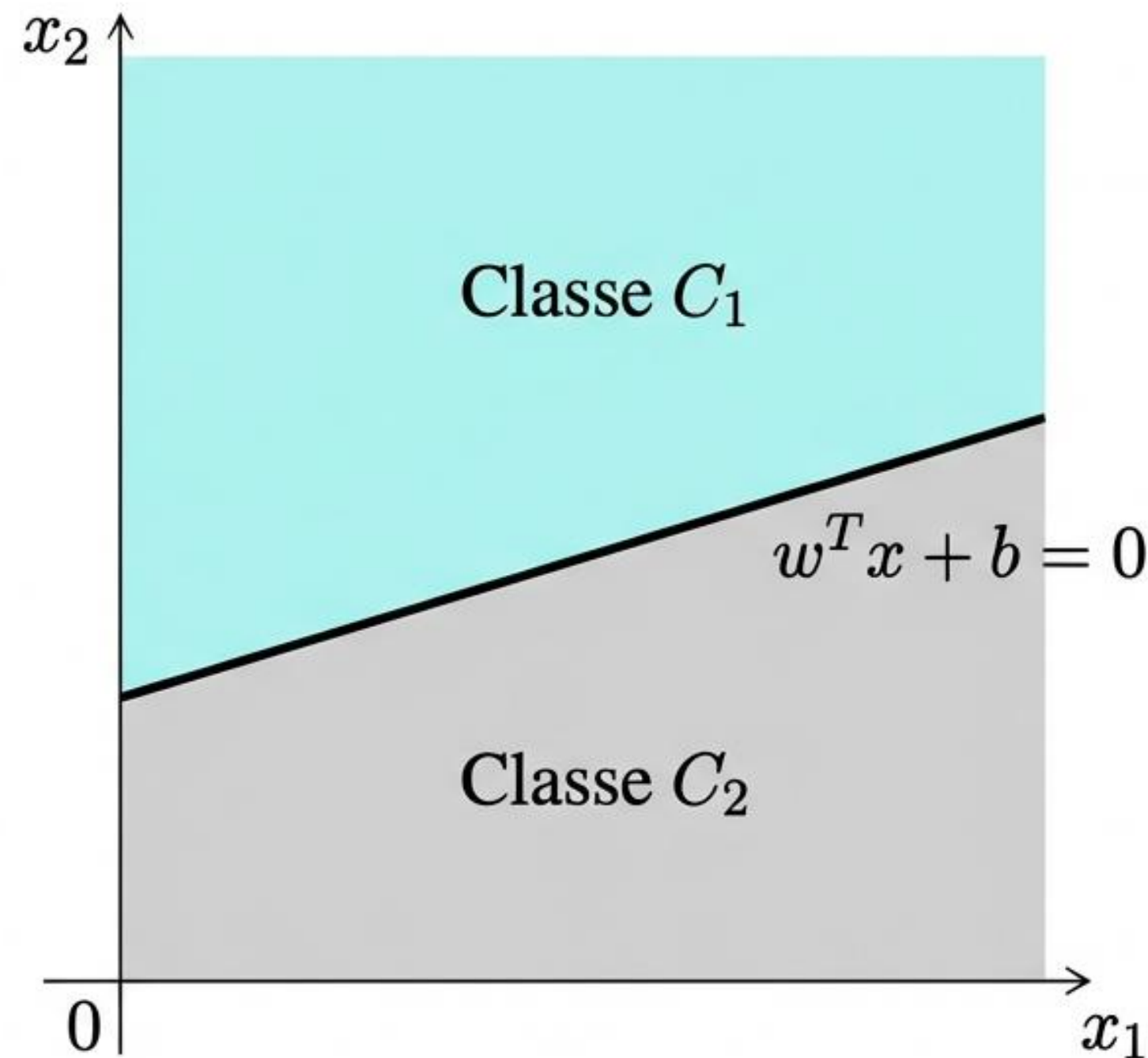
# A Matemática da Decisão

Campo Local:  $v = \sum_{i=1}^m w_i x_i + b$

Regra de Decisão:

Se  $v > 0 \rightarrow y = +1$  (Classe  $C_1$ )

Se  $v \leq 0 \rightarrow y = -1$  (Classe  $C_2$ )





# O Algoritmo de Aprendizado

Aprendizado Supervisionado via Correção de Erro

**Taxa de aprendizado**  
(tamanho do passo)

**O Erro**  
(Desejado - Atual)

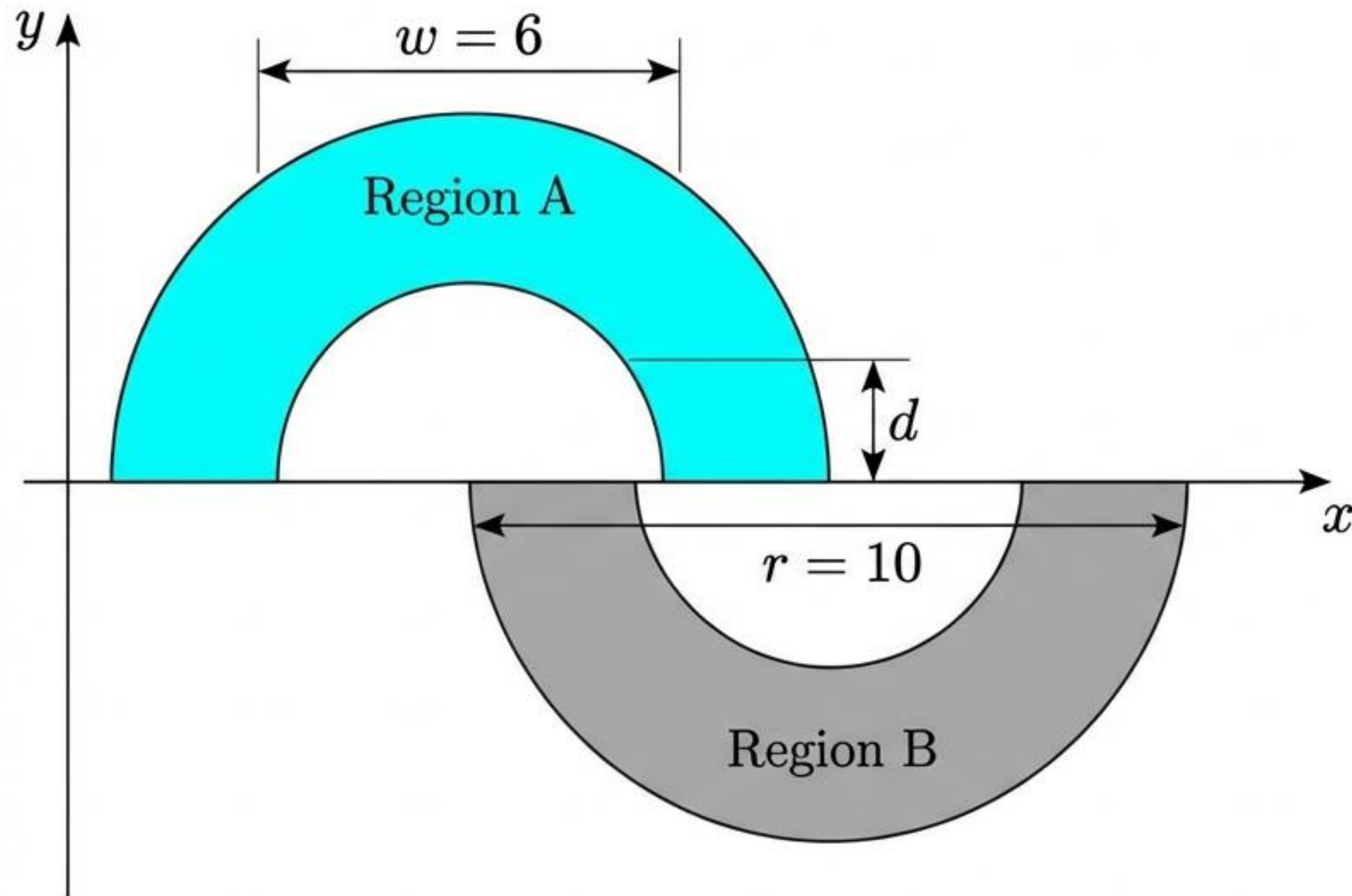
$$w(n + 1) = w(n) + \eta [d(n) - y(n)] x(n)$$

**Novo peso ajustado**

**A entrada que**  
causou o erro

**Lógica: Se o erro for zero, o peso não muda. Se houver erro, movemos o peso na direção da entrada.**

# Estudo de Caso: As “Duas Luas” (Double Moon)



**Objetivo:** Testar a capacidade de classificação binária.

**Dataset:** 1.000 amostras de treino, 2.000 de teste.

**Parâmetro Crítico ( $d$ ):**

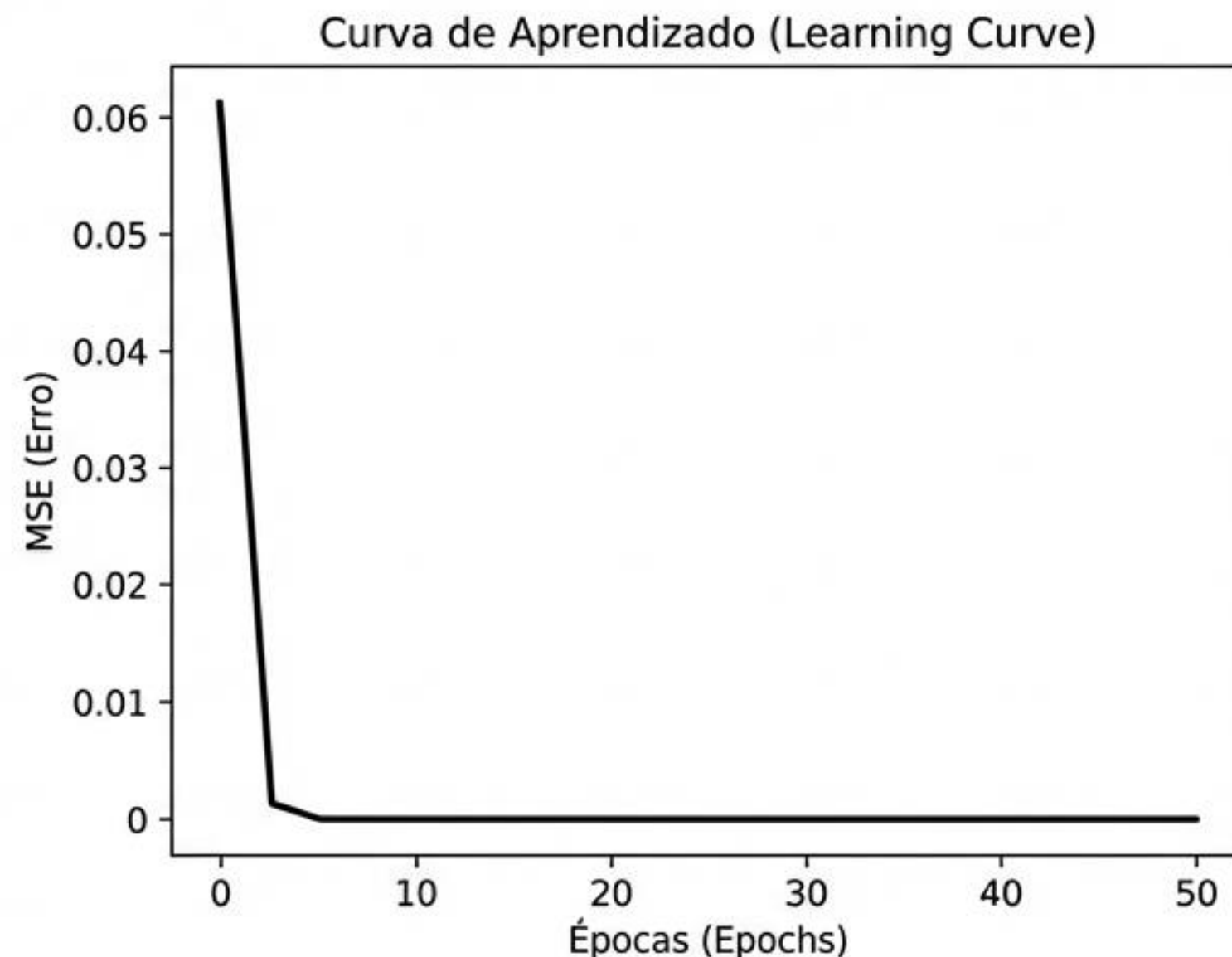
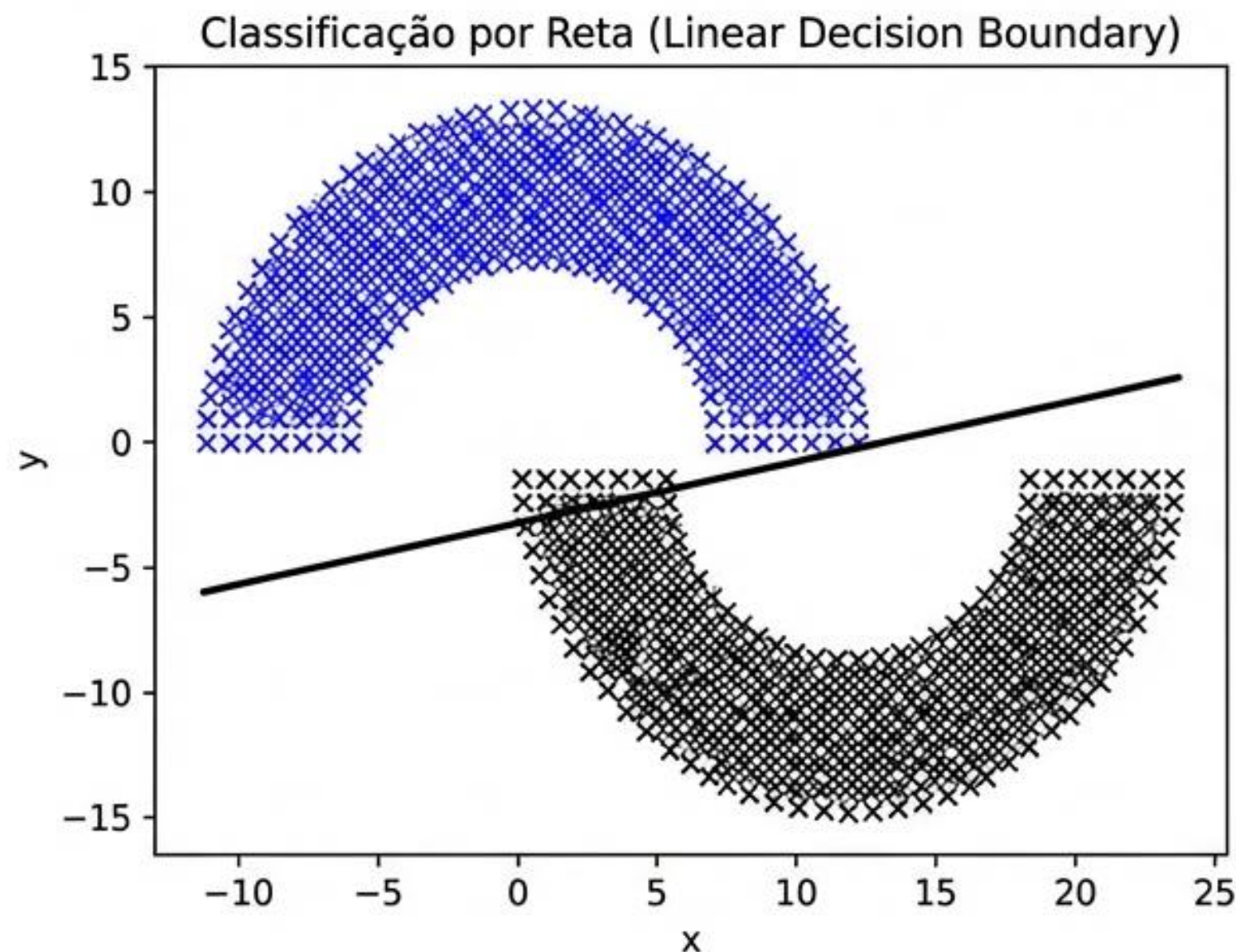
Se  $d > 0$ : Separação clara.

Se  $d < 0$ : Sobreposição (Overlap).



# Cenário 1: Sucesso na Separabilidade Linear

Distância  $d = 1$  (Luas Separadas)

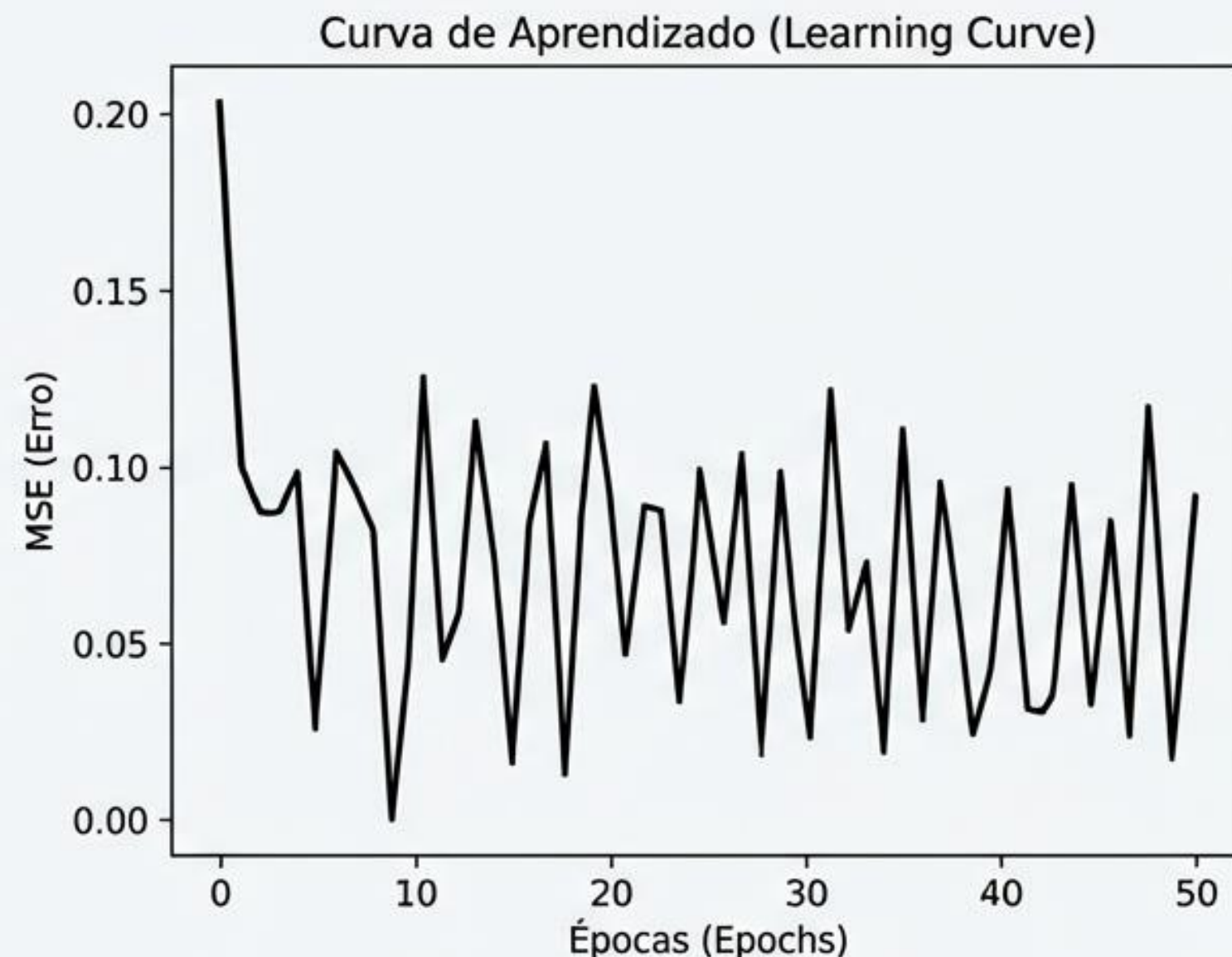
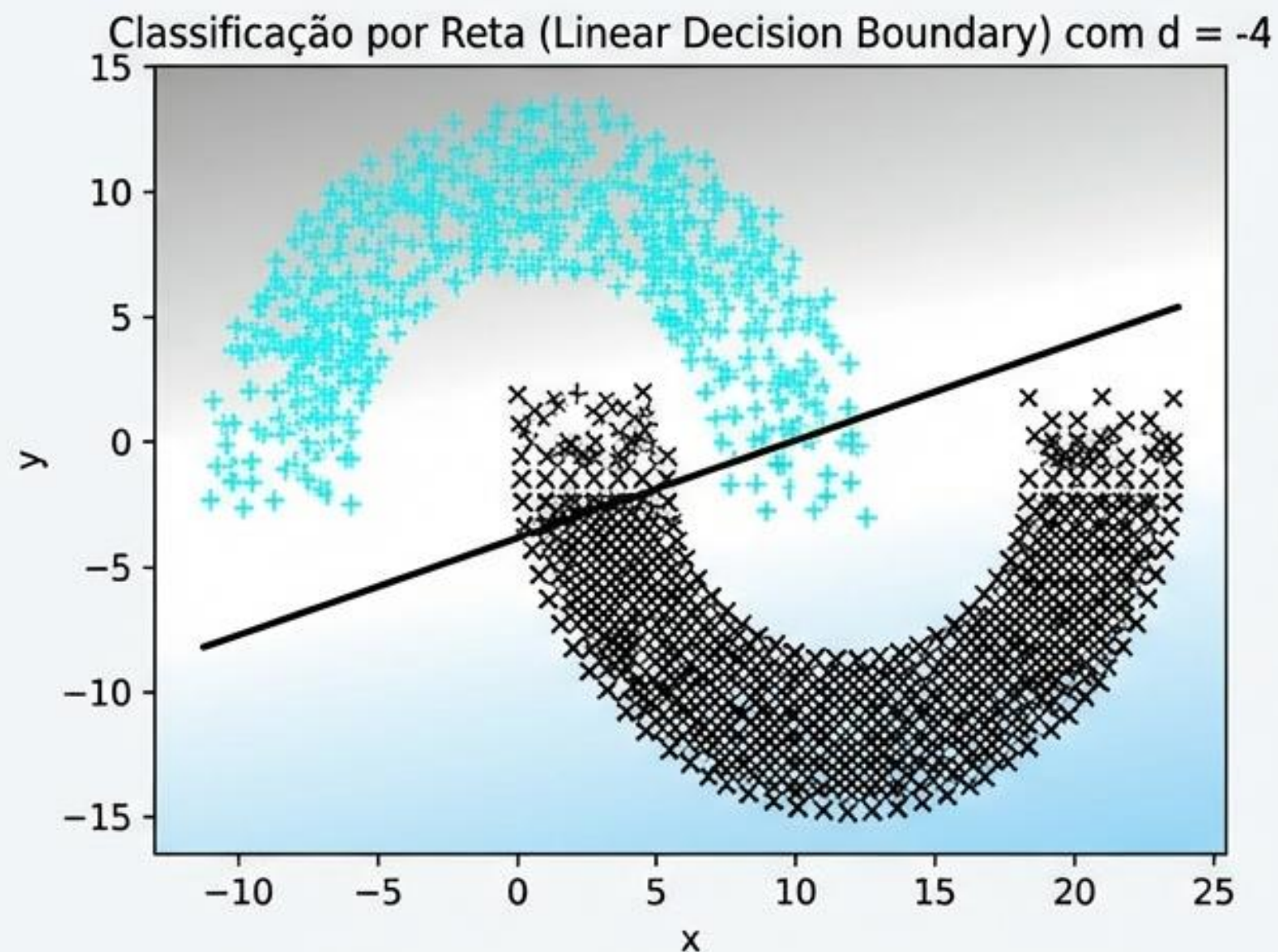


**Resultado:** Convergência rápida em 3 épocas. Como existe um espaço vazio, a reta separa as classes perfeitamente.



# Cenário 2: Falha na Não-Linearidade

Distância  $d = -4$  (Luas Sobrepostas)



**Resultado:** O algoritmo oscila infinitamente. Não existe reta capaz de separar Azul de Preto quando há sobreposição. O Perceptron falha.



# O Teorema de Convergência do Perceptron

## O Teorema:

O algoritmo de aprendizado de correção de erro *garante* encontrar uma solução em um número finito de passos...

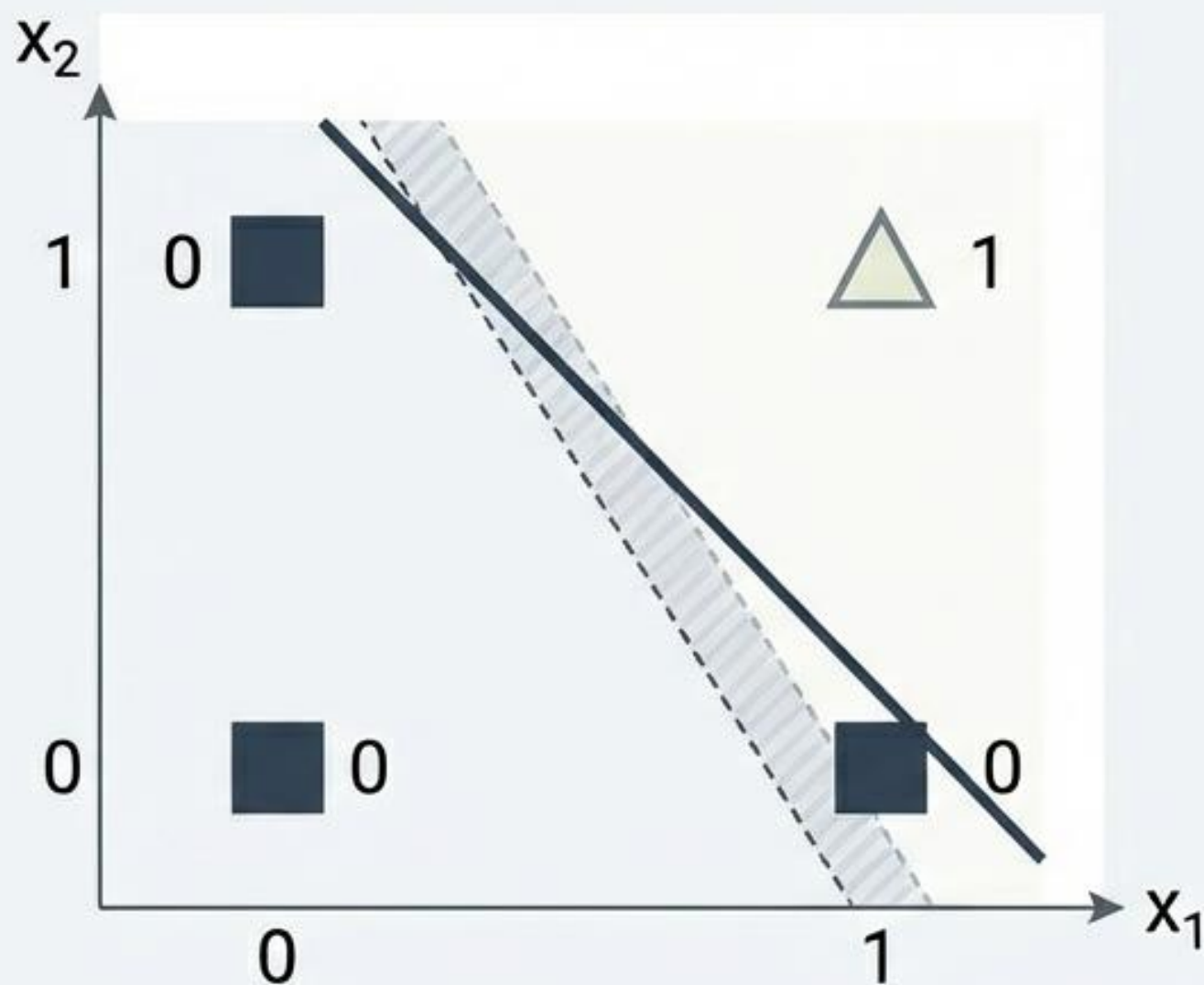
**...SE E SOMENTE SE** os dados forem linearmente separáveis.



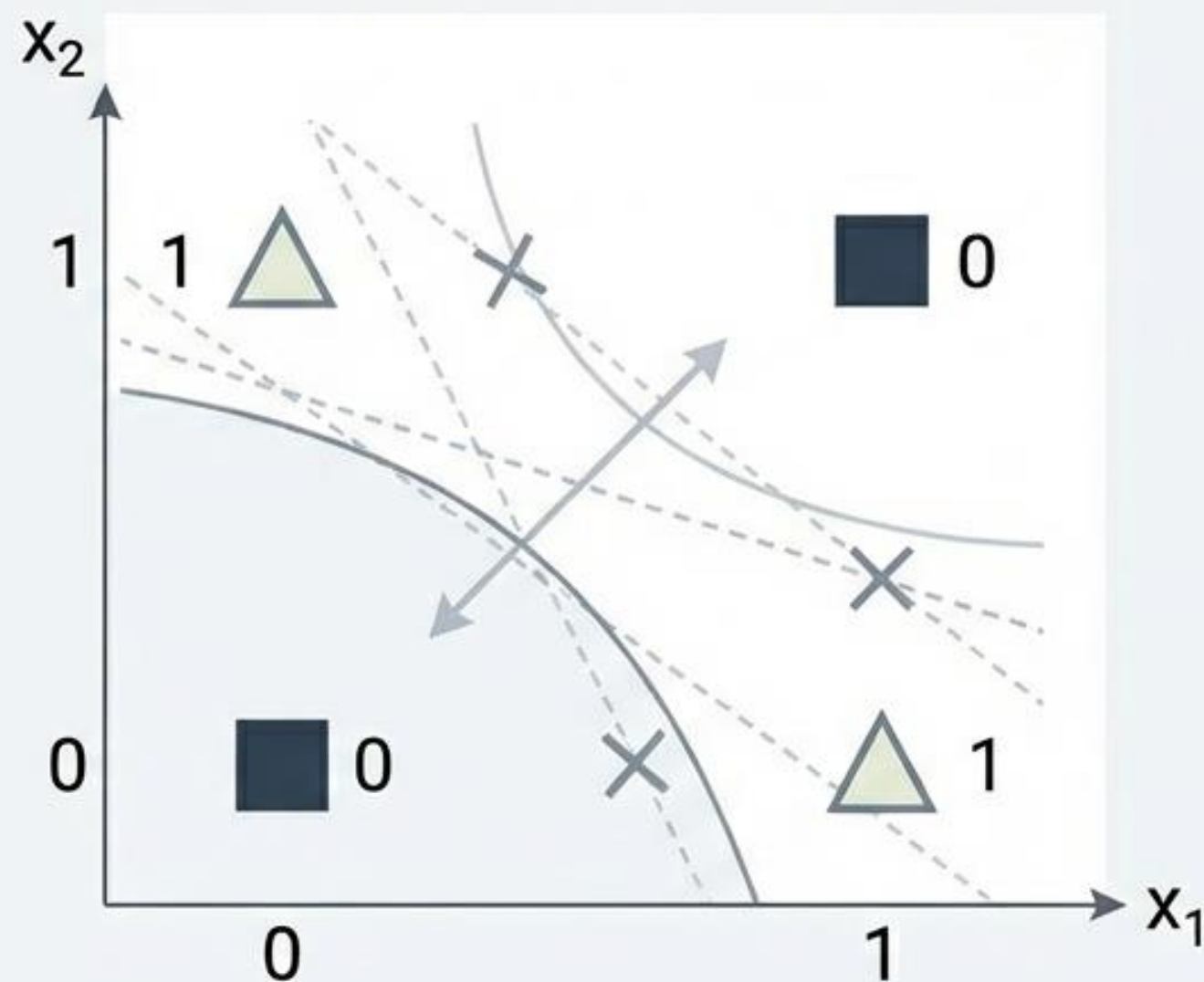
**Implicação Prática:** Se os dados se sobrepõem (como no caso  $d = -4$ ) ou possuem geometria complexa (XOR), o Perceptron padrão entrará em loop eterno. Critérios de parada são obrigatórios.

# Limitações Históricas: O Problema XOR

Linearmente Separável (AND/OR)



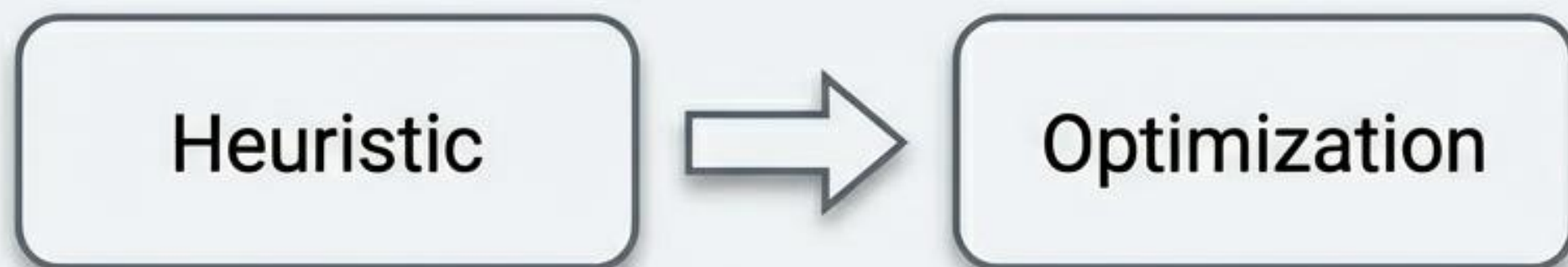
Não-Linear (XOR)



**Crítica de Minsky & Papert (1969):** A incapacidade de resolver o XOR paralisou a pesquisa em IA por anos (o 'Inverno da IA').



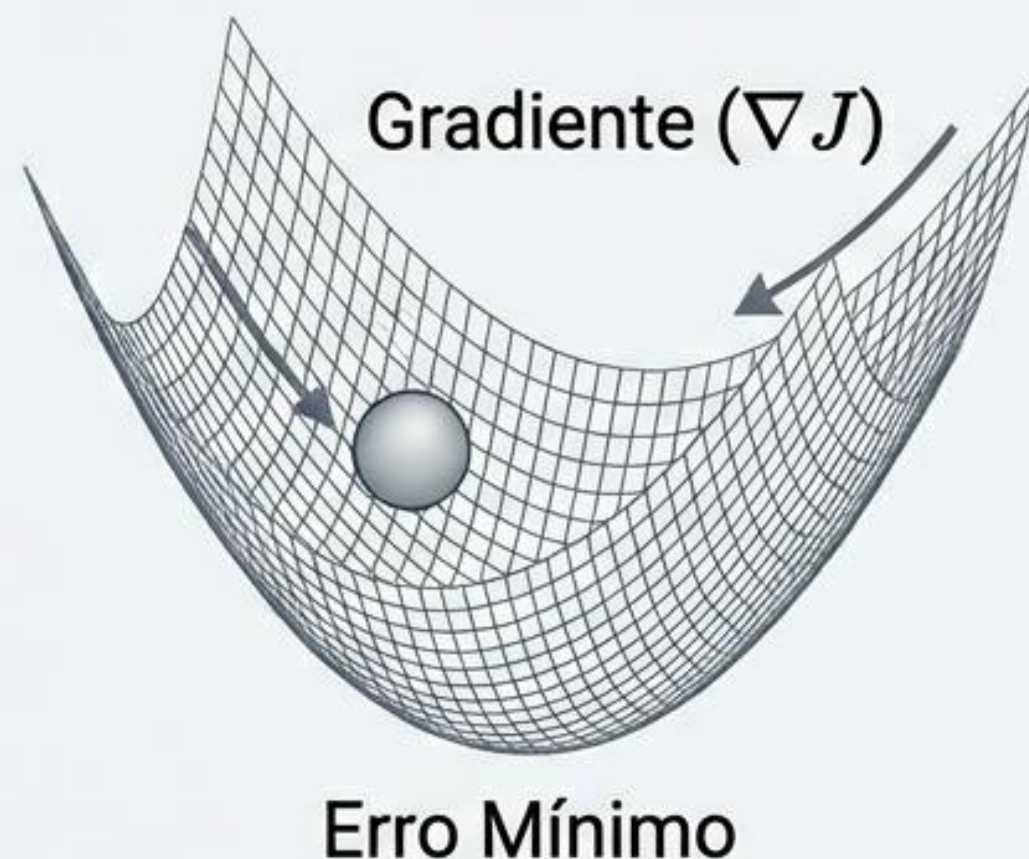
# Evolução 1: Introduzindo a Função de Custo



Para estabilizar o aprendizado, saímos da simples "correção de erro" para a "minimização de custo".

$$J(\mathbf{w}) = \sum \underbrace{\left( -\mathbf{w}^T \mathbf{x}(n) d(n) \right)}$$

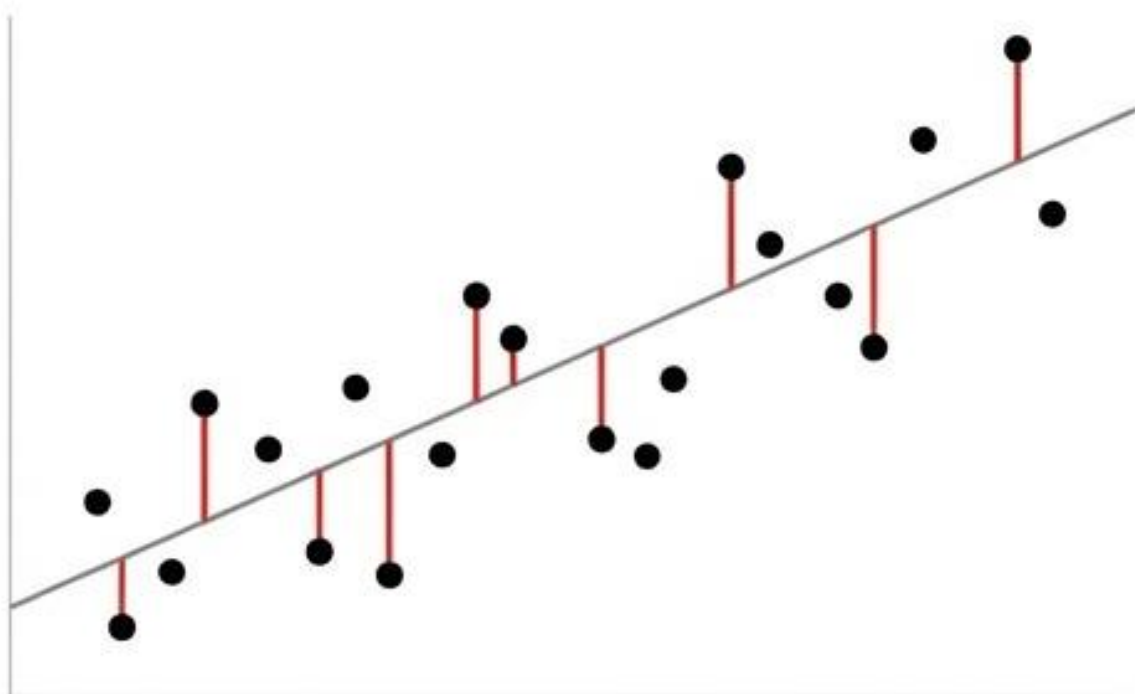
Para o conjunto de amostras erradas



Isso habilita o uso do **Gradiente Descendente**, a base matemática de todo o Deep Learning moderno.

# Funções de Perda (Loss Functions)

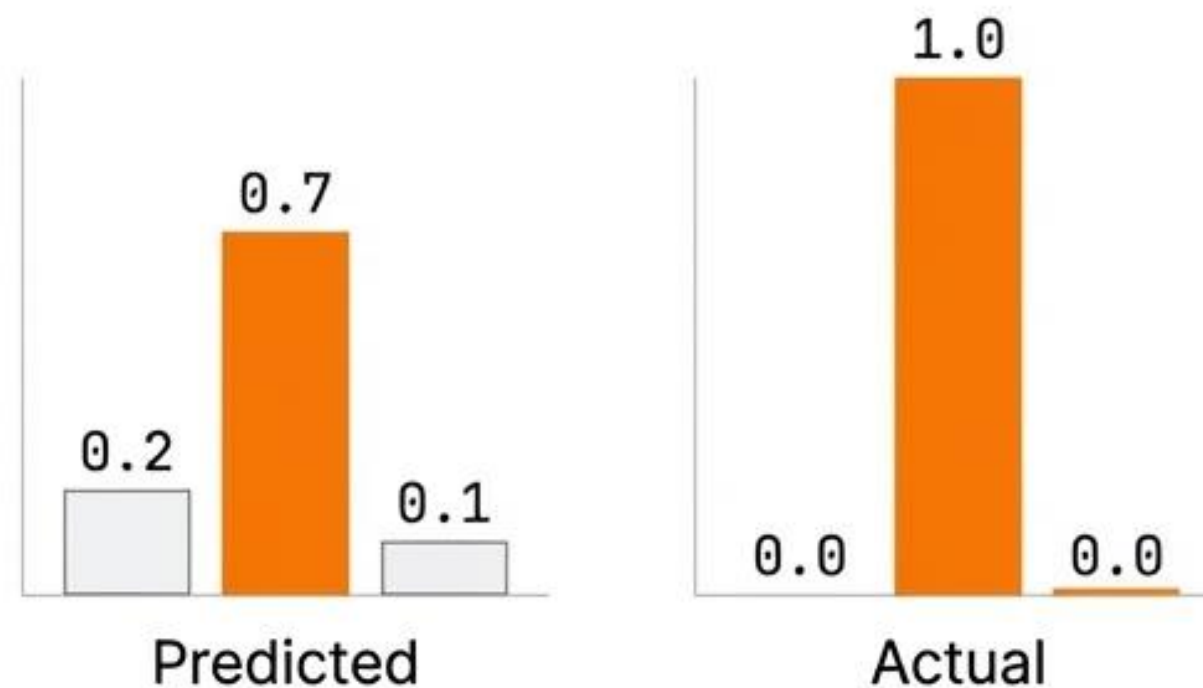
## Regressão



MSE (Erro Quadrático Médio)

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

## Classificação



Cross-Entropy (Entropia Cruzada)

$$H(p, q) = - \sum p(x) \log q(x)$$



# Evolução 2: Aprendizado em Lote (Batch)

## Online vs. Batch

### Perceptron Online



Atualiza pesos a cada amostra.  
Rápido, mas ruidoso e instável.

### Perceptron Batch

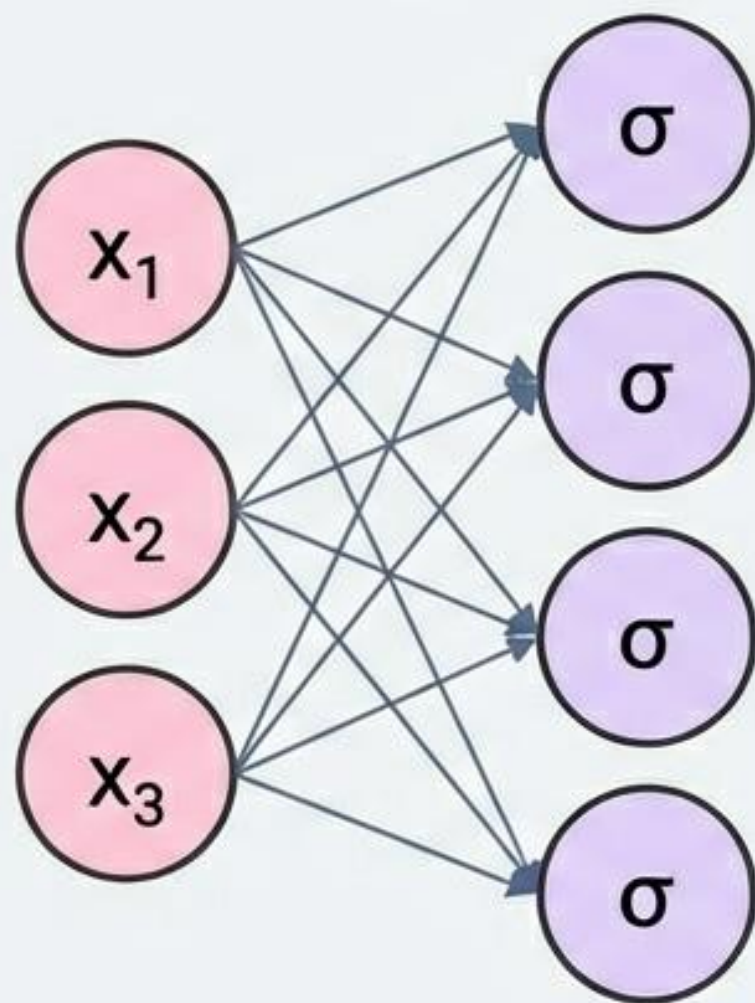


Calcula o erro de *todas* as amostras,  
depois atualiza. Estimativa  
estável do gradiente.

$$w(n+1) = w(n) + \eta \sum x(n)d(n)$$

# Evolução 3: Vetorização e Matrizes

## Transição para o Código Moderno (TensorFlow/PyTorch)



Matriz de Pesos  $W^{(1)}$


$3 \times 4$

$3 \times 4$

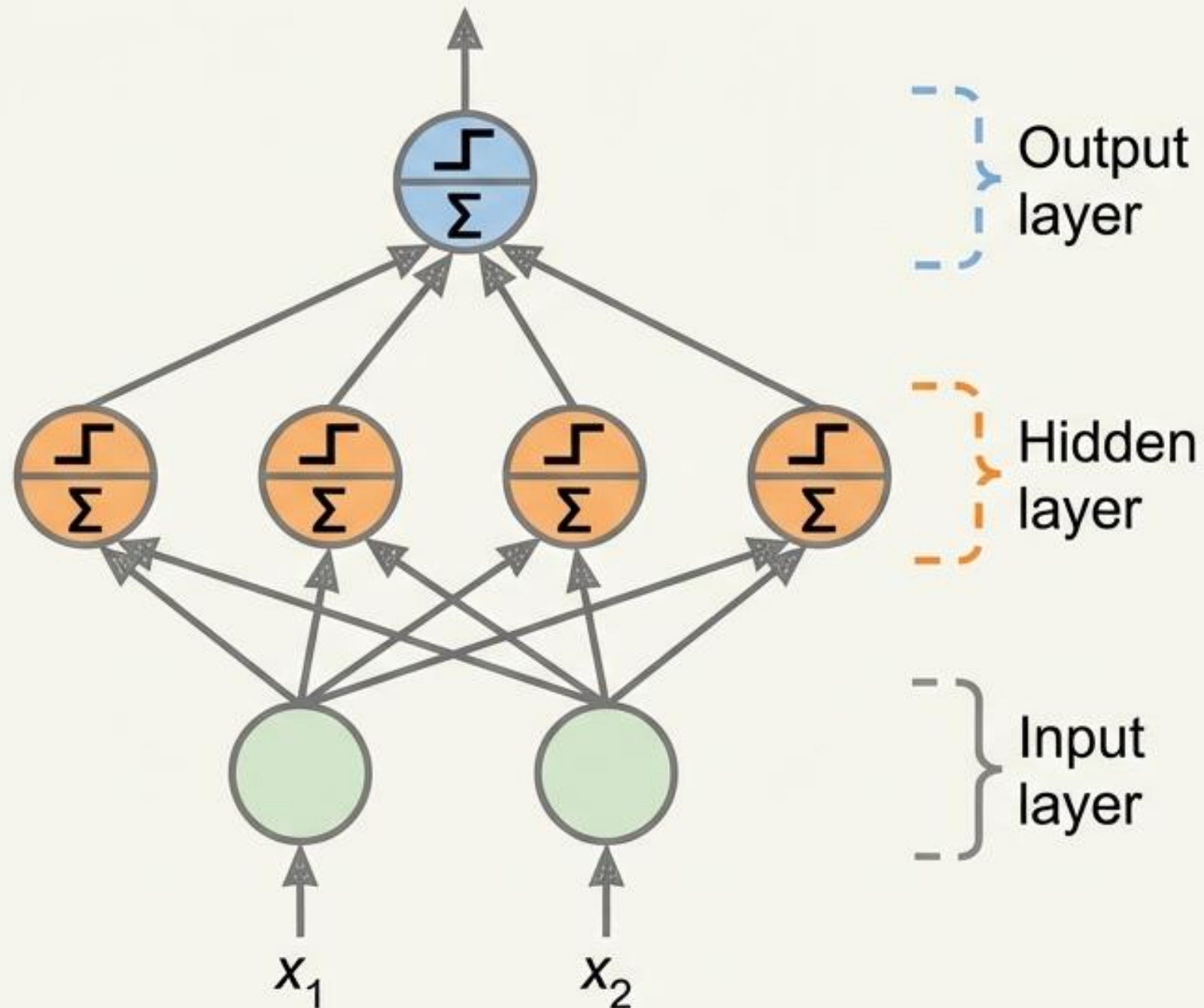
$$Z = X = X \cdot W + b$$

$$A = \sigma(Z)$$

Em frameworks modernos, substituímos laços `for` por Álgebra Linear. Isso define uma 'Dense Layer'.



# A Solução para a Não-Linearidade: MLP



**Como resolver o XOR e as Duas Luas?**

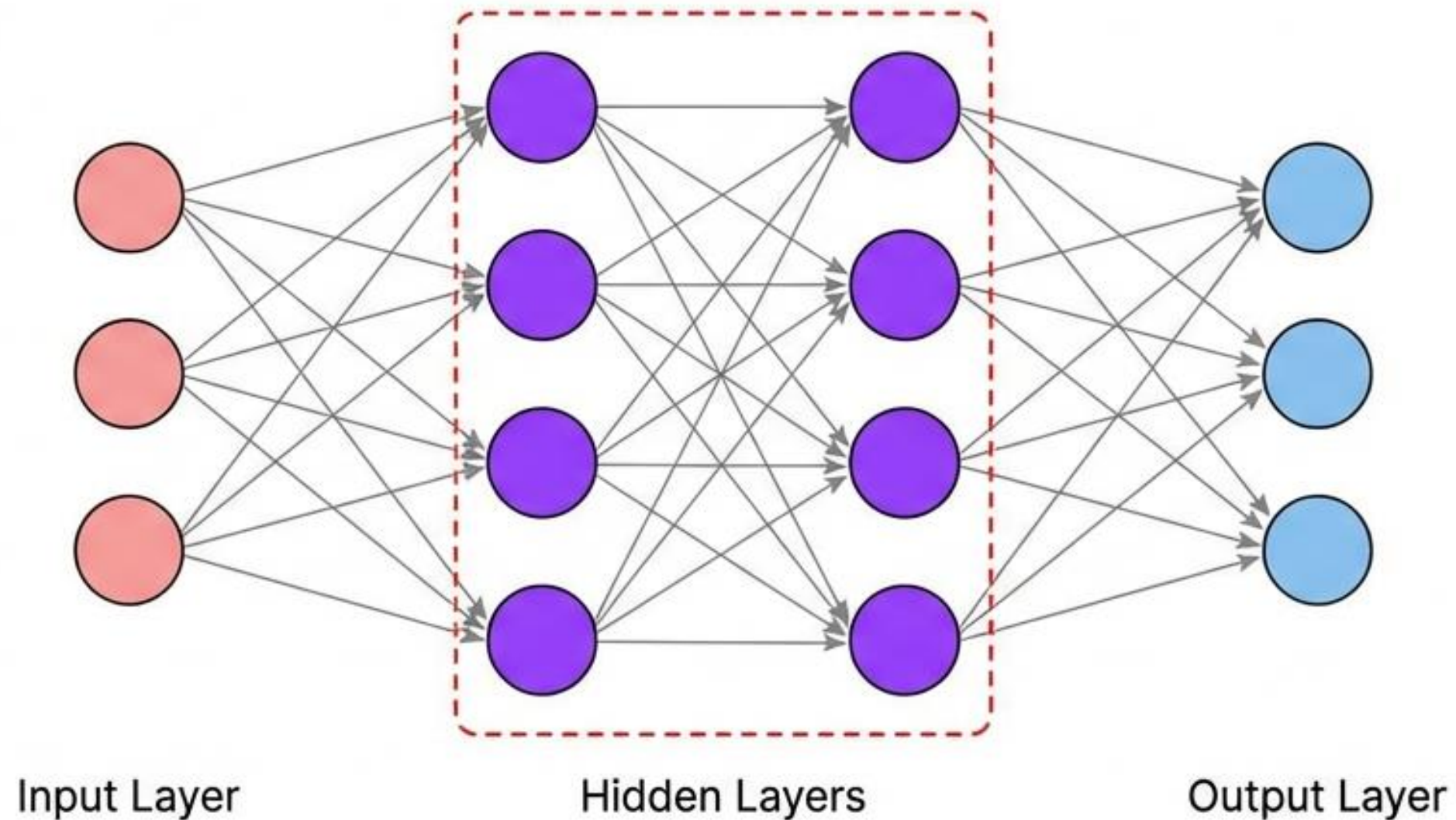
Adicionamos **Camadas Ocultas**. Elas distorcem o espaço de entrada, tornando os dados linearmente separáveis para a camada final.

**Resultado:** O nascimento das Redes Neurais Profundas (Deep Learning).

# A Revolução do MLP (Multilayer Perceptron)

**Camadas Ocultas:** Onde a rede aprende hierarquias e características complexas.

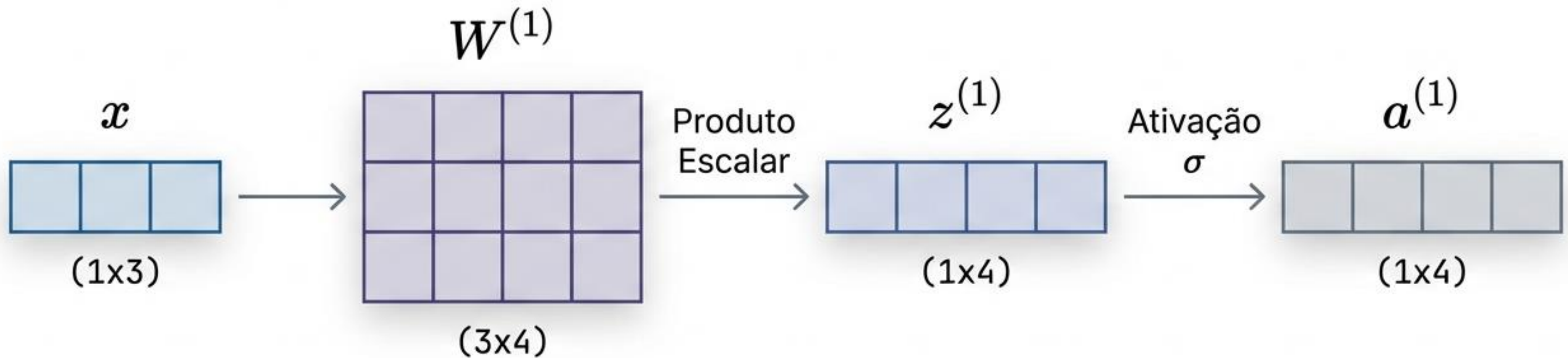
**Teorema da Aproximação Universal:** Com neurônios suficientes, esta arquitetura pode aproximar qualquer função contínua.





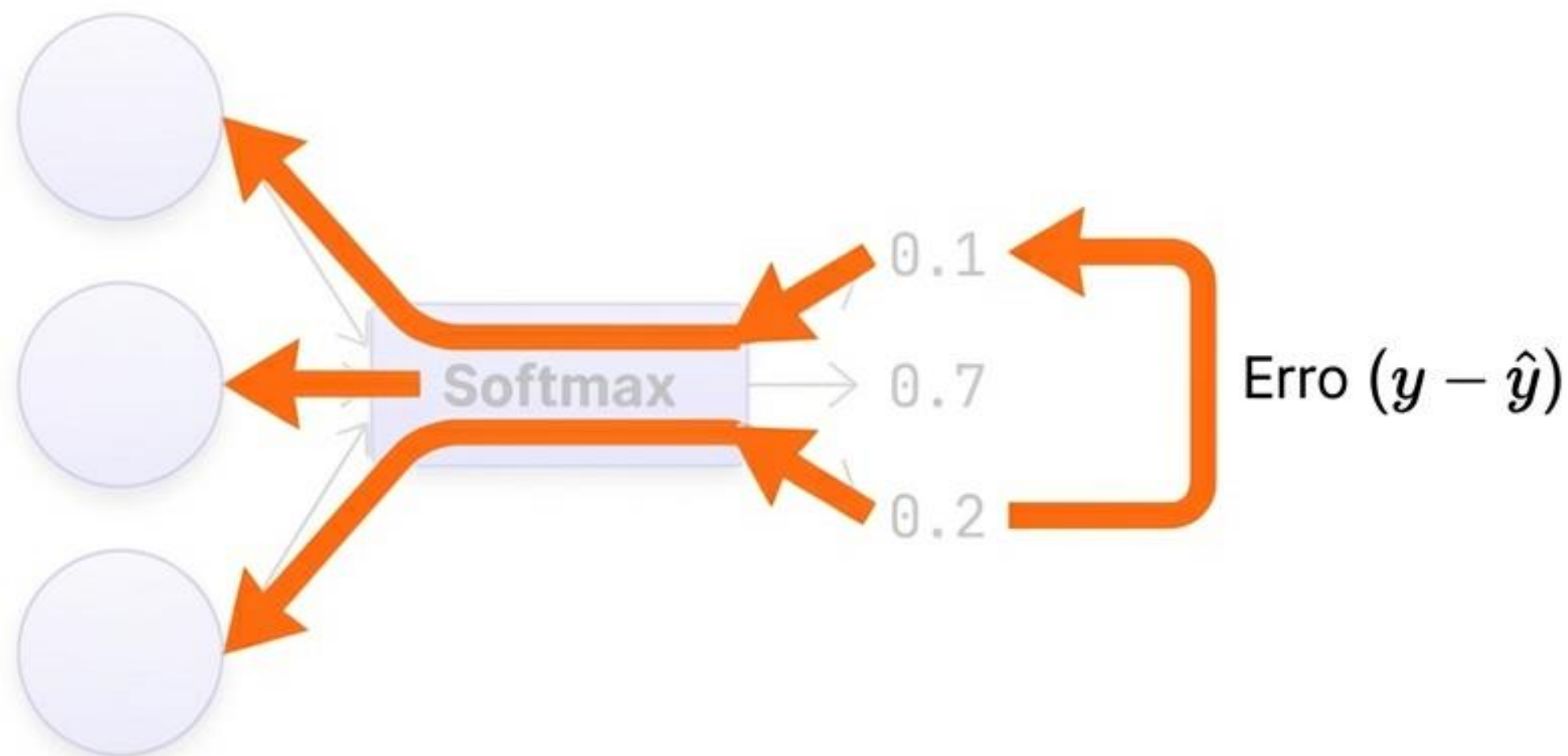
# O Fluxo de Dados (Forward Pass)

Os dados viajam da esquerda para a direita, sofrendo transformações dimensionais a cada matriz de pesos.  $z^{(l)} = a^{(l-1)} \cdot W^{(l)}$



# Backpropagation: A Atribuição de Crédito

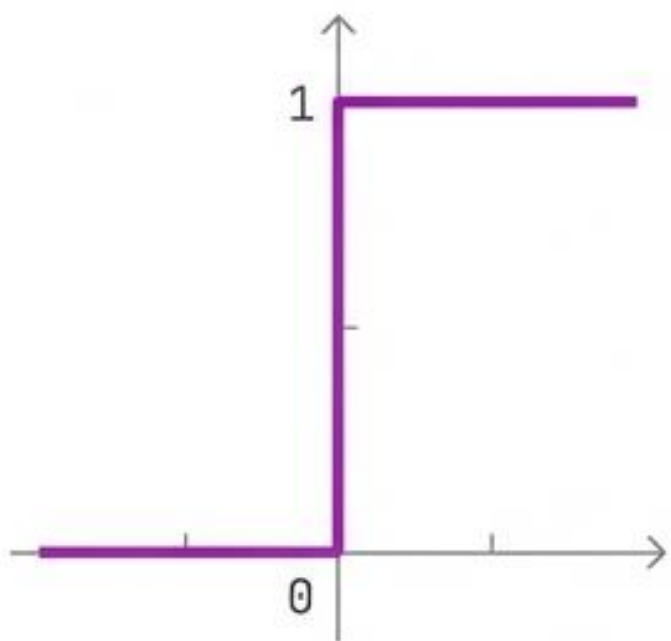
Como a rede aprende? Calculamos o erro na saída e o “retropropagamos” pela rede. O objetivo é descobrir a responsabilidade de cada neurônio no erro final e ajustar seus pesos.



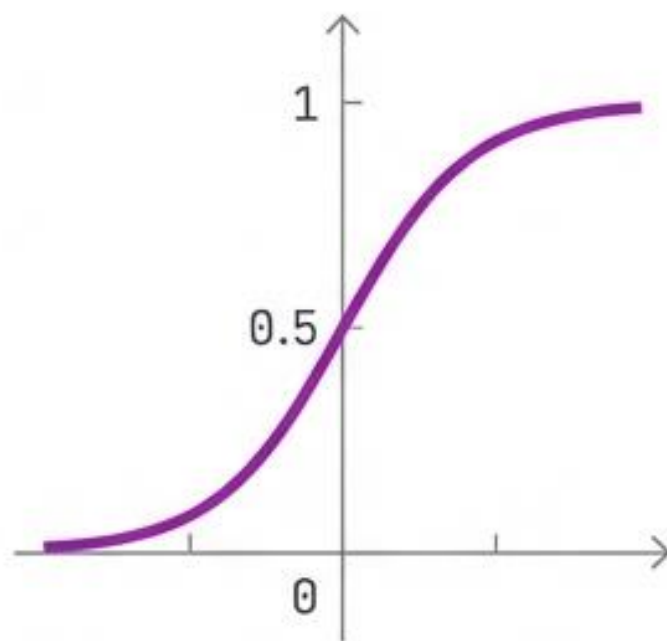


# Funções de Ativação: A Centelha da Não-Linearidade

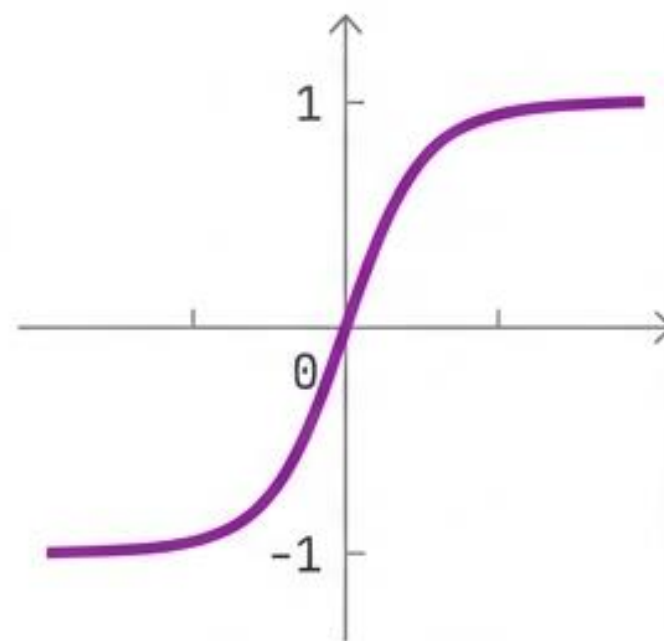
Sem essas funções, a rede seria apenas uma regressão linear gigante. **ReLU** é a escolha padrão atual: eficiente e evita o desaparecimento do gradiente.



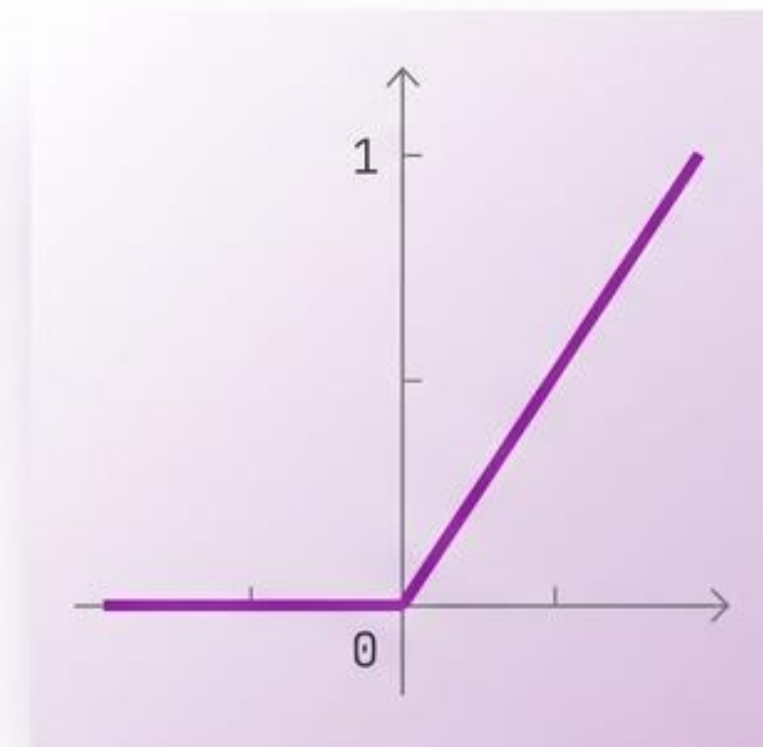
Step (Degrau)



Sigmoide



Tanh

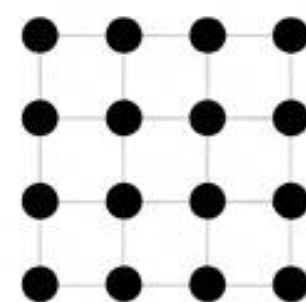
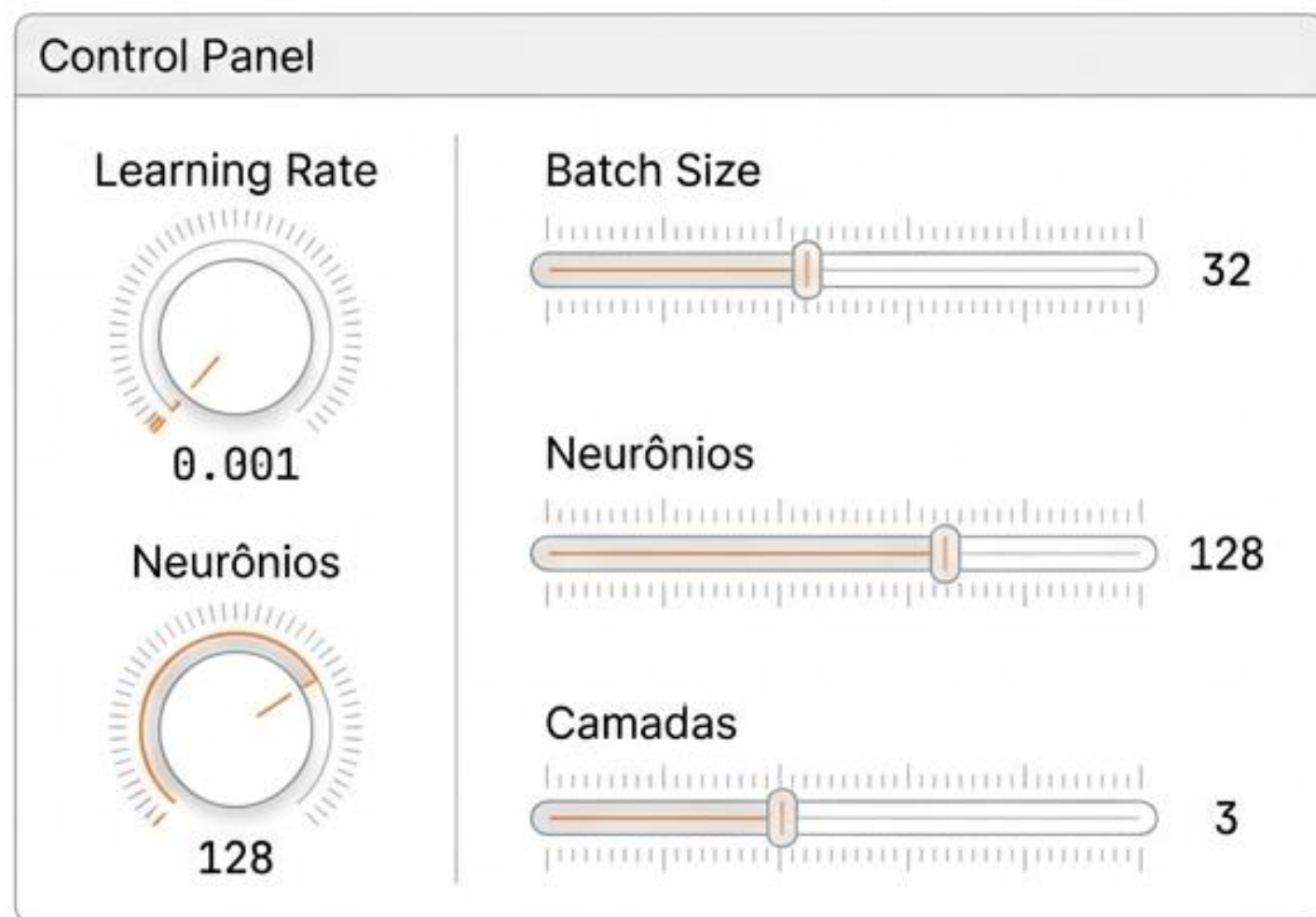


ReLU (Padrão)

# Ajuste Fino de Hiperparâmetros

Parâmetros que não são aprendidos pela rede, mas definidos pelo engenheiro.

**Batch Size:** O equilíbrio entre velocidade e generalização (ex: 32).



Grid Search



Random Search

Mais Eficiente



# Resumo e Próximos Passos

- **O Perceptron:** É um classificador linear robusto, mas limitado.
- **Teorema de Convergência:** Garante sucesso *apenas* em dados separáveis (Ex:  $d = 1$ ).
- **A Falha:** Falha em dados complexos ou sobrepostos (Ex: XOR,  $d = -4$ ).
- **A Evolução:** Funções de Custo, Batch Learning e Matrizes preparam o terreno para redes modernas.
- **A Solução:** Para problemas reais, precisamos de profundidade (MLPs).

**Obrigado!**