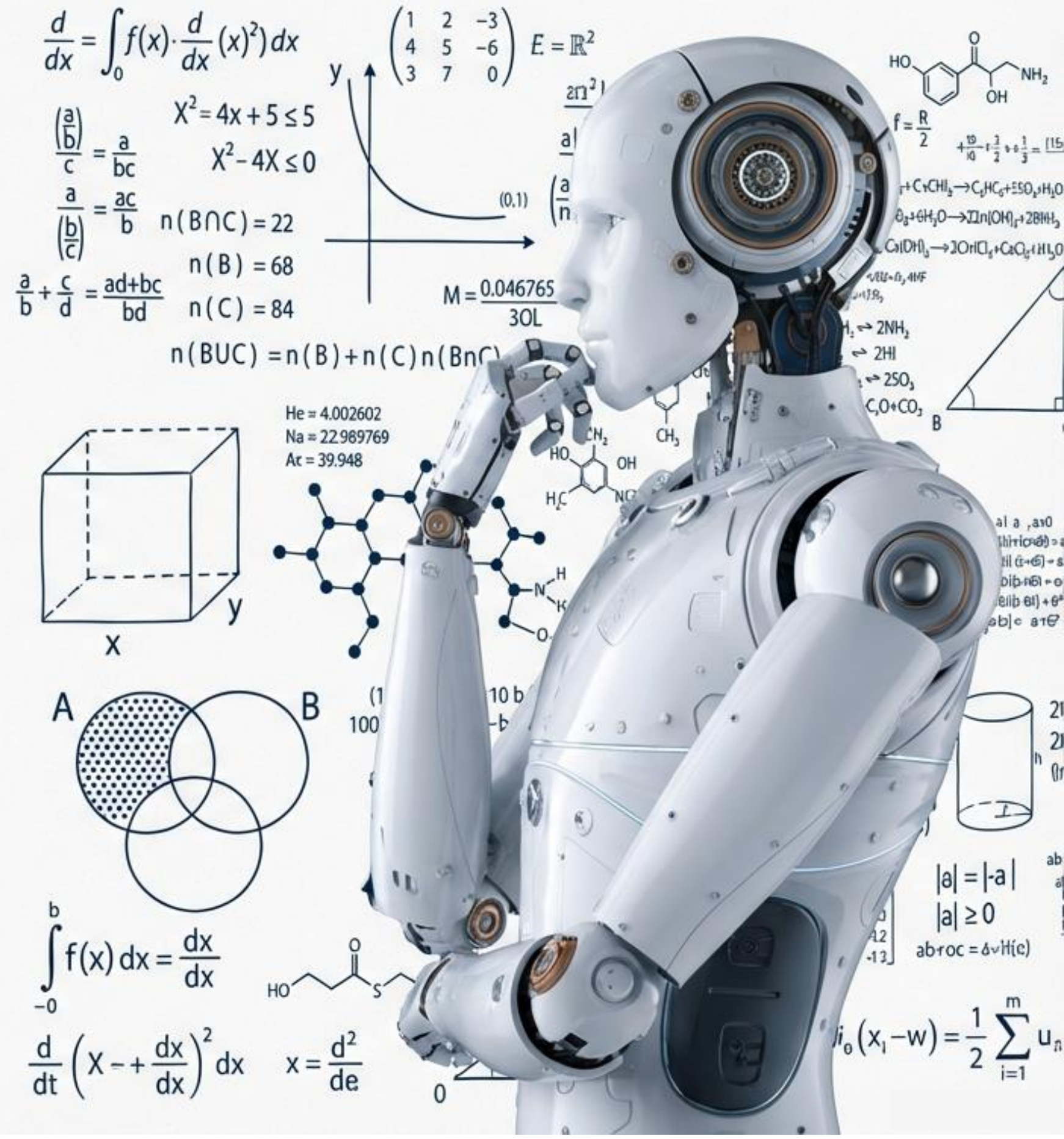


# Aprendizado de Máquina: Dos Fundamentos às Redes Profundas

## Aula 1

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



# Objetivos do Curso

- **Objetivo Geral:**
  - Fornecer uma introdução teórica e prática sobre redes neurais e aprendizado de máquina.
  - Familiarizar o aluno com as principais ferramentas e técnicas utilizadas na área, com foco em frameworks modernos como TensorFlow e PyTorch.
- **Objetivos Específicos (Atualizado):**
  - Compreender abordagens fundamentais: Gradiente Descendente, Propagação (Forward/Backward) e Regularização.
  - Dominar arquiteturas de redes supervisionadas (Perceptrons, CNNs, RNNs) e não supervisionadas (Autoencoders, RBMs).
- **Projetos Extensionistas:** Desenvolvimento de diagnósticos e oficinas para comunidades de software livre e instituições com necessidade de apoio social.

# O que é Aprendizado de Máquina?

- **Definições Clássicas:**

- "O campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados" — Arthur Samuel, 1959.
- "Um programa de computador aprende com a experiência  $E$ , em relação a uma tarefa  $T$  e uma medida de desempenho  $P$ , se seu desempenho em  $T$ , medido por  $P$ , melhora com a experiência  $E$ " — Tom Mitchell, 1997.

- **Por que usar ML?**

- Substituir longas listas de regras manuais por modelos que aprendem padrões.
- Resolver problemas complexos onde abordagens tradicionais falham (ex: reconhecimento de fala).
- Adaptar-se a ambientes flutuantes e novos dados.

# Tipos de Aprendizado de Máquina

- **Aprendizado Supervisionado:**
  - O conjunto de treinamento inclui os rótulos (soluções desejadas).
  - **Tarefas:** Classificação (ex: filtro de spam) e Regressão (ex: prever valores numéricos).
- **Aprendizado Não Supervisionado:**
  - O sistema tenta aprender sem um "professor"; os dados não possuem rótulos.
  - **Aplicações:** Clusterização, Detecção de Anomalias e Visualização.
  - **Modelos:** Inclui Autoencoders e Máquinas Restritas de Boltzmann (RBMs).
- **Aprendizado por Reforço:**
  - Um agente observa o ambiente, executa ações e recebe recompensas ou penalidades.

# Redes Neurais Artificiais (ANNs)

- **Inspiração Biológica:**

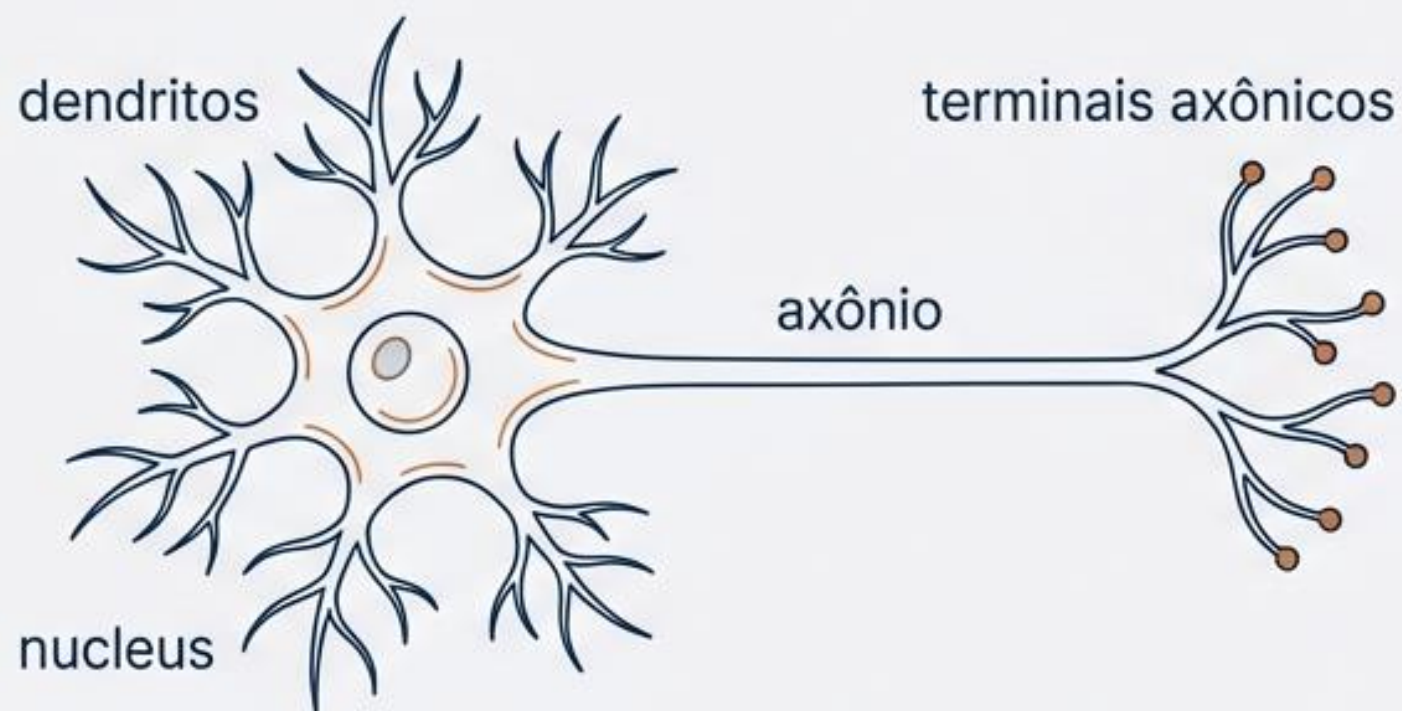
- Baseadas na arquitetura do cérebro, mas evoluíram para modelos matemáticos e computacionais distintos.
- Constituídas por neurônios artificiais conectados em camadas.

- **O Perceptron:**

- Arquitetura simples inventada por Frank Rosenblatt (1957).
- Baseado na Unidade Lógica de Limiar (TLU): soma ponderada das entradas seguida de uma função de passo.
- Limitação: Resolve apenas problemas linearmente separáveis (ex: falha no problema XOR).

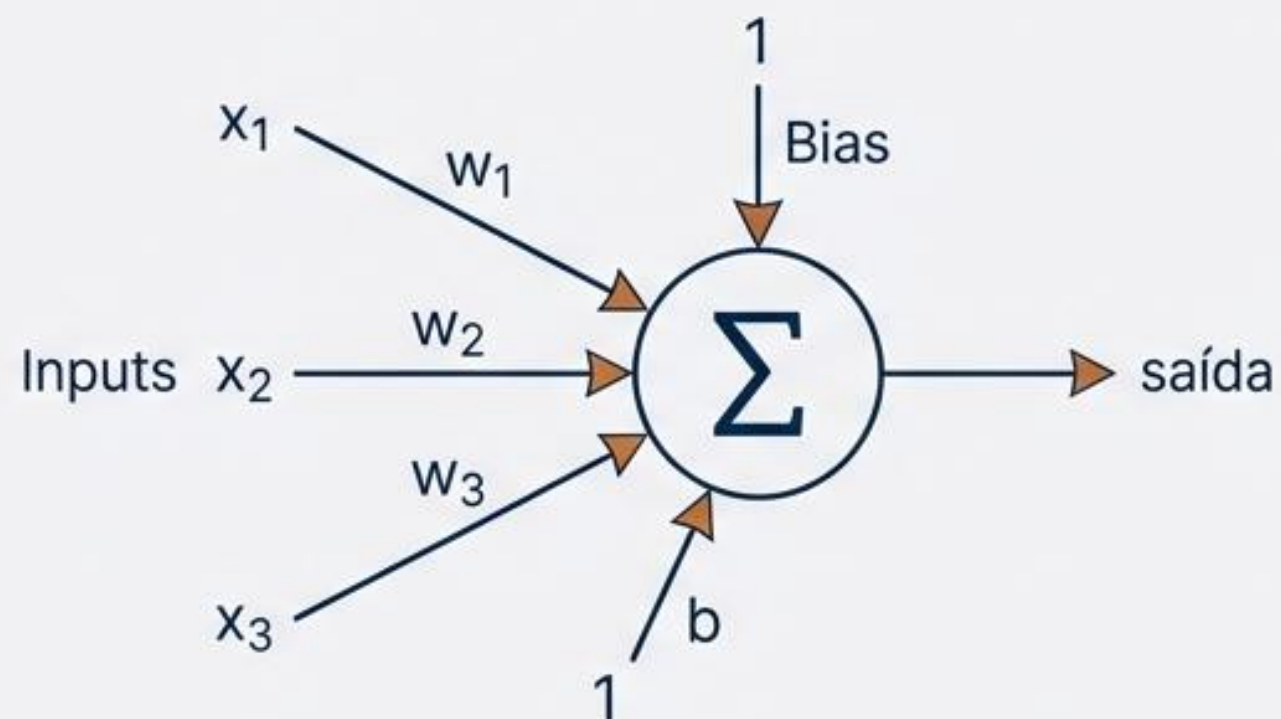
# A Inspiração Biológica e o Modelo Matemático

## Biológico



O cérebro humano contém bilhões de neurônios. Sinais elétricos viajam dos dendritos (entrada) pelo axônio até os terminais (saída).

## Artificial - Modelo McCulloch-Pitts



$$v = \sum w_i x_i + b$$

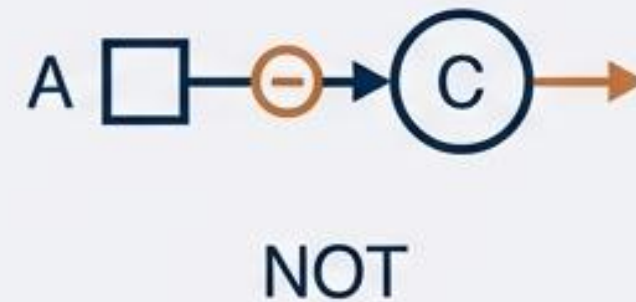
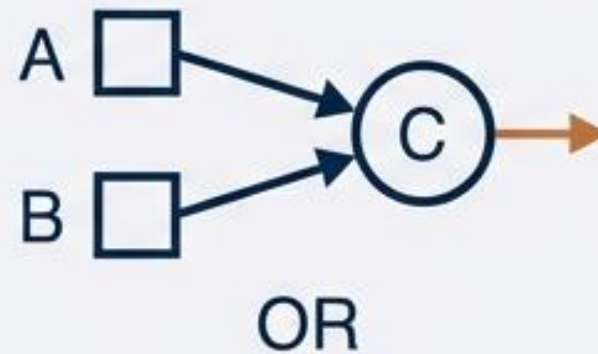
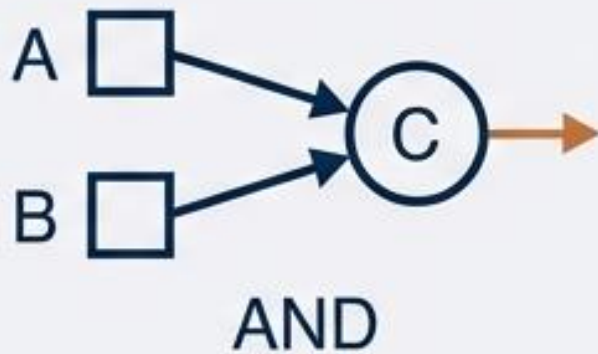
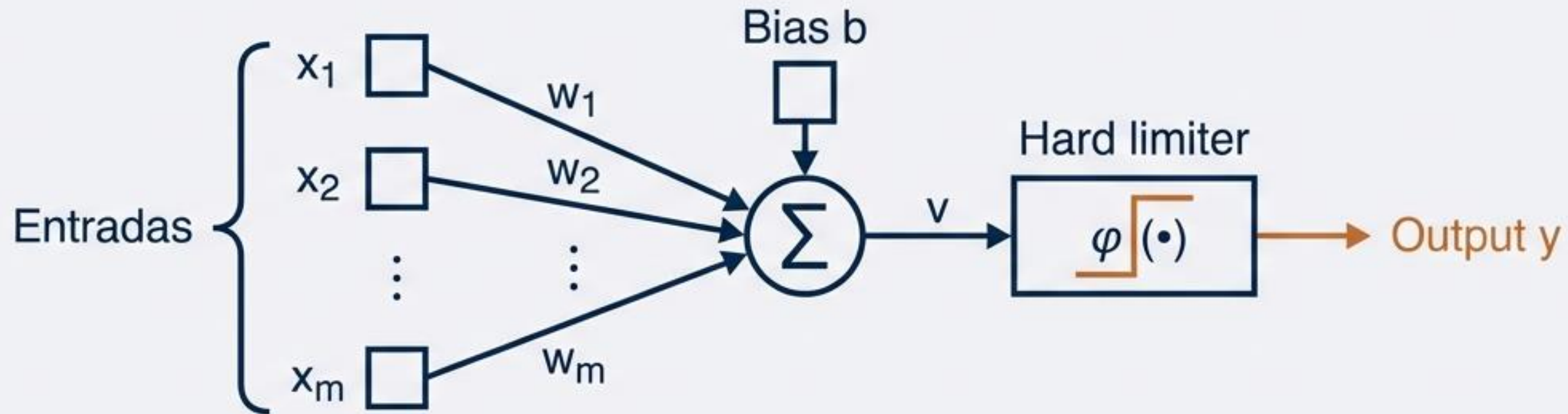
$x_i$ : Entradas (Inputs)

$w_i$ : Pesos Sinápticos (Importância)

$b$ : Viés (Bias - Ajuste de fronteira)

Simulação matemática do comportamento biológico.

# O Perceptron de Rosenblatt: Lógica e Vetores

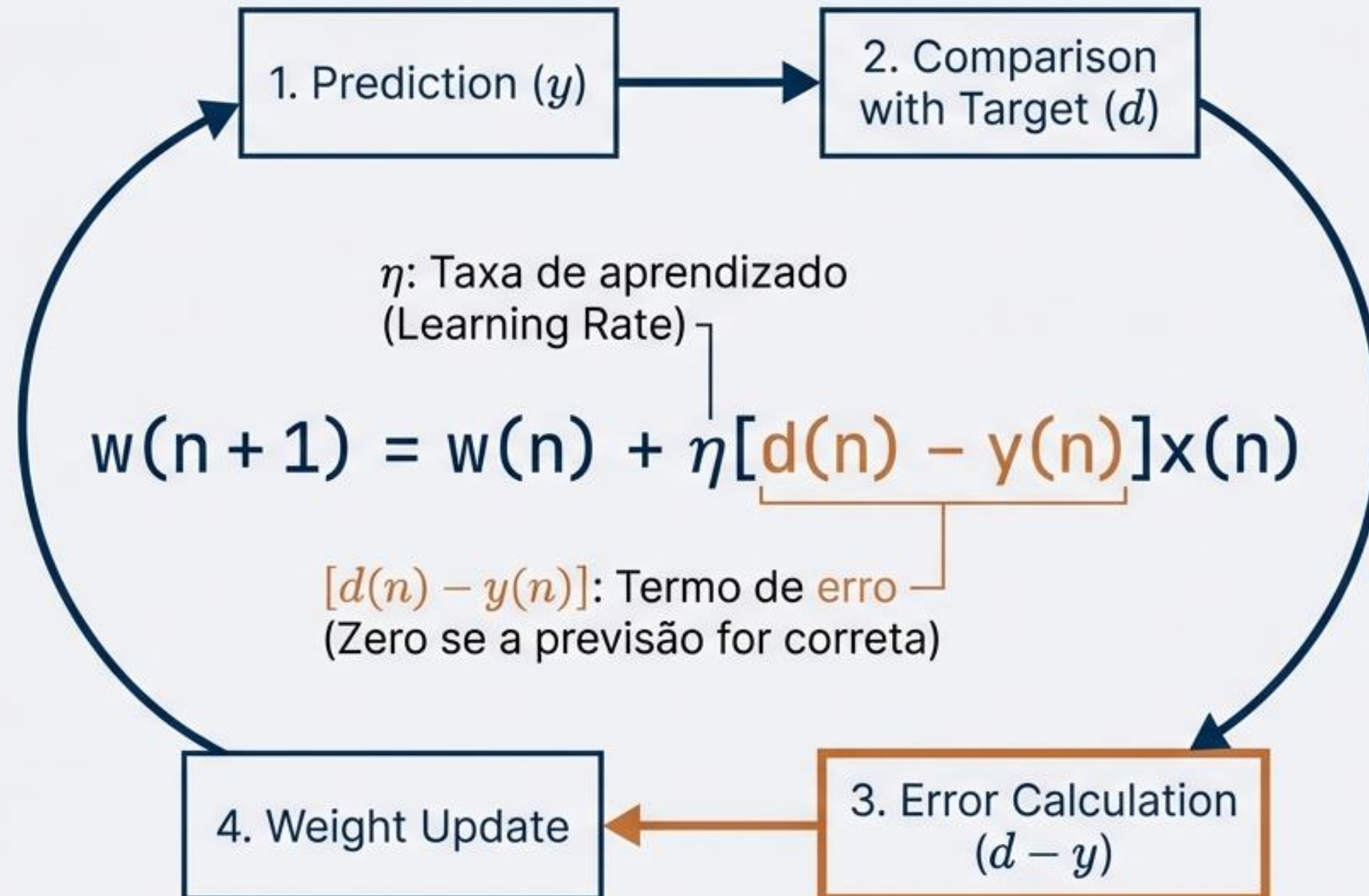


**Conceito Central:** Um combinador linear seguido por um limitador rígido (função degrau).

**Notação Vetorial:**  
$$y(n) = \text{sgn}[w^T(n)x(n)]$$

**Lógica:** Um único neurônio resolve problemas linearmente separáveis (portas lógicas).

# O Algoritmo de Aprendizado: Correção de Erro



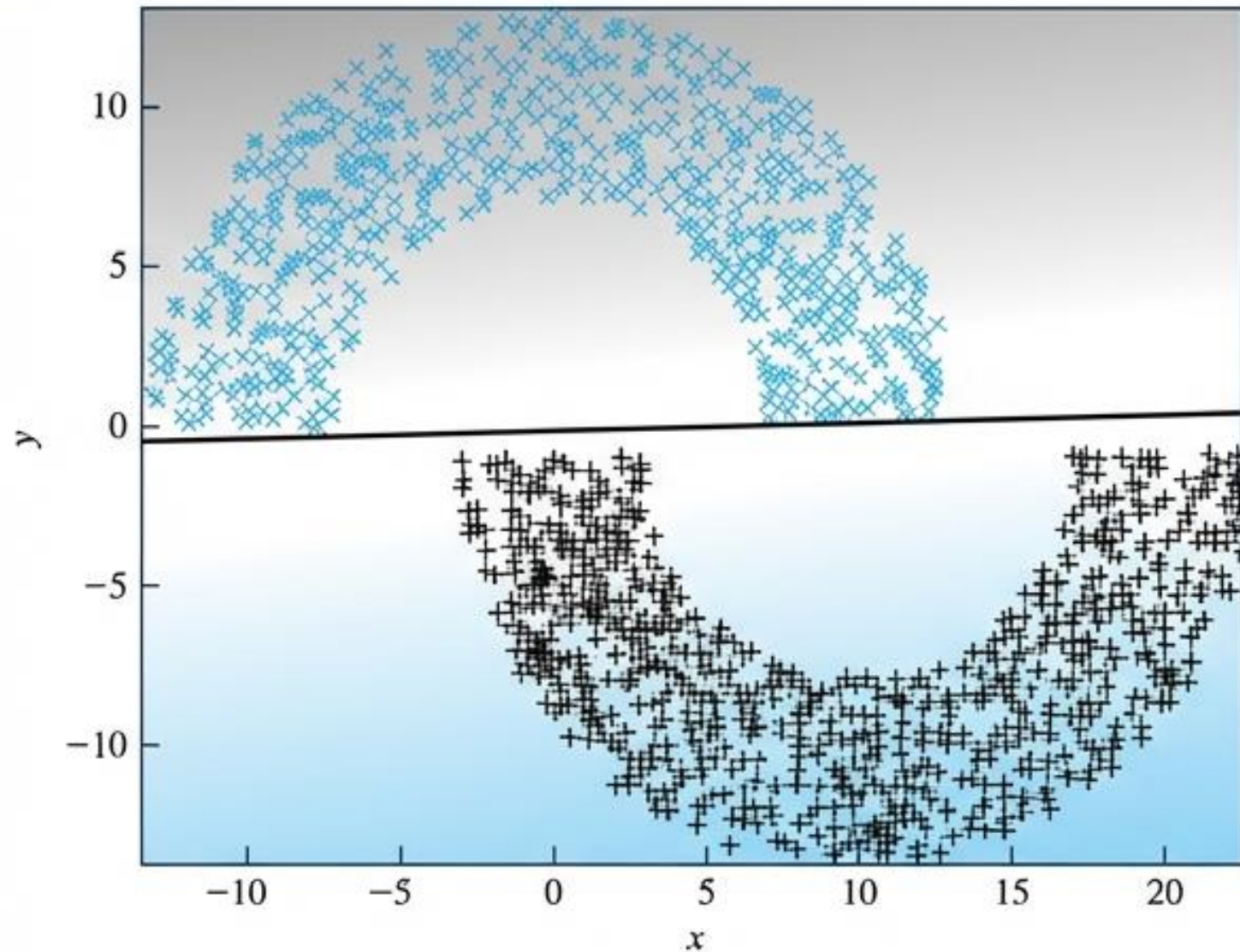
## Estratégias de Aprendizado:

- **Amostral (Online):**  
Ajuste a cada exemplo. Mais rápido, maior oscilação.
- **Em Lote (Batch):**  
Acumula erros do conjunto. Maior estabilidade, convergência suave.

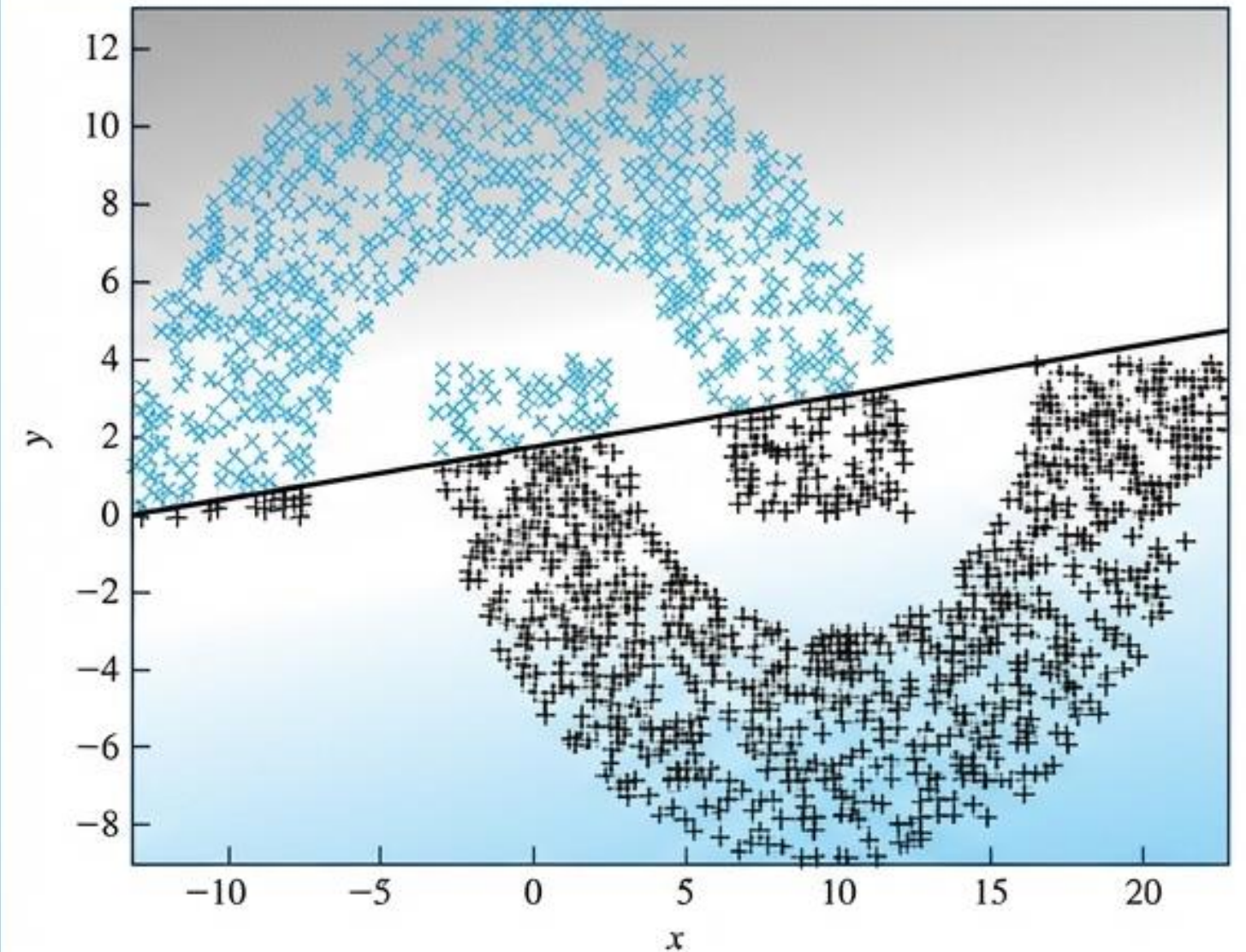
**Objetivo:** Minimizar a função de custo  $J(w)$  via Gradiente Descendente.

# O Limite da Linearidade: O Experimento 'Duas Luas'

Linearmente Separável: Convergência em 3 iterações



Não-Linear (Sobreposição): Falha na Convergência



Conclusão Crítica: O Perceptron simples falha em problemas não-lineares. Ele **oscila indefinidamente tentando traçar uma reta** onde uma **curva** é necessária.

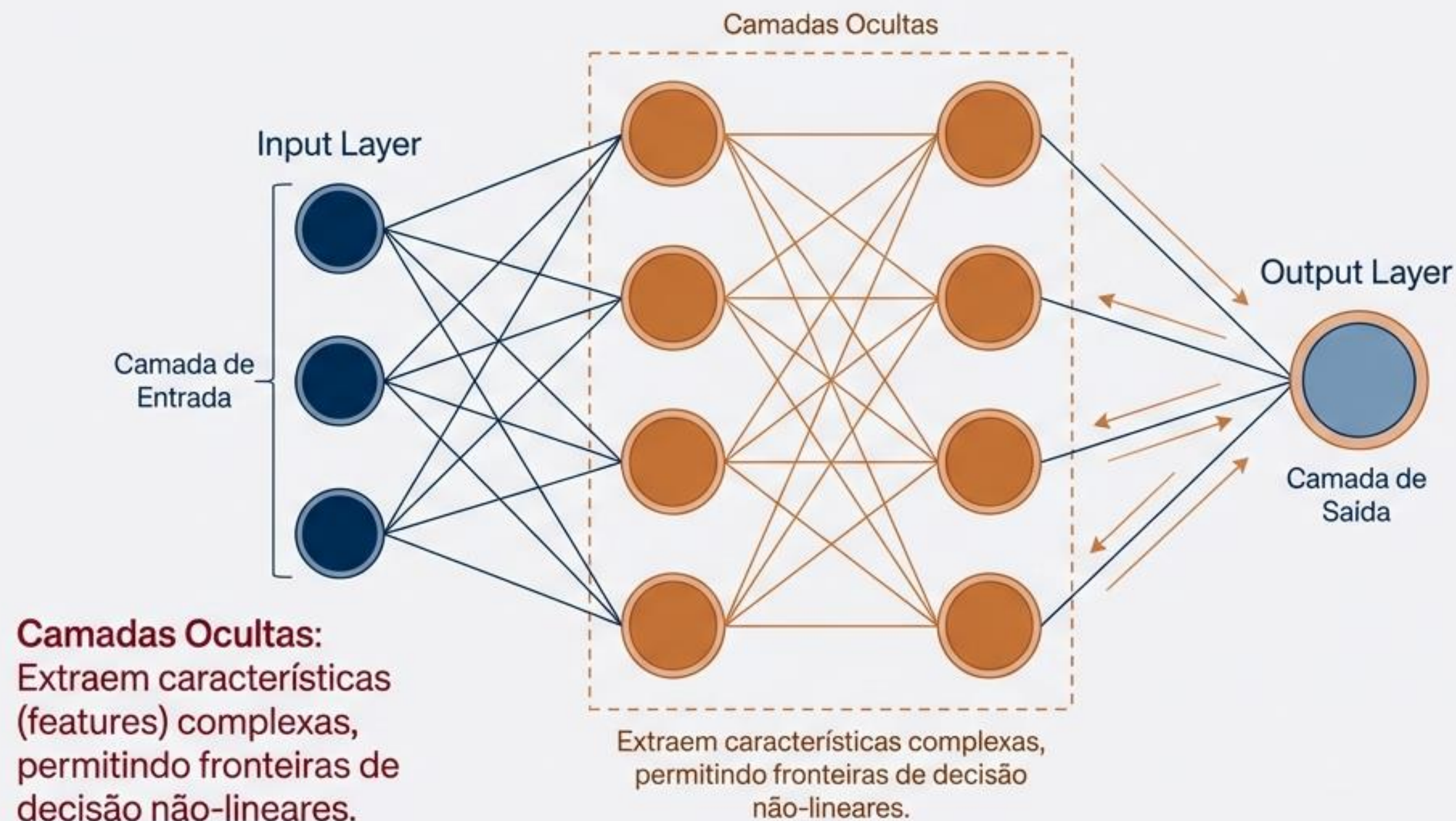
# Perceptron Multicamadas (MLP) e Deep Learning

- **Estrutura do MLP:**
  - Composto por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.
  - Quando possui múltiplas camadas ocultas, é denominado Rede Neural Profunda (DNN).
- **Componentes Essenciais:**
  - **Funções de Ativação:** Introduzem não-linearidade. Exemplos: Sigmoid, Tanh, ReLU (Rectified Linear Unit) e Softmax.
  - **Tensores:** Estruturas de dados fundamentais para manipulação em frameworks como TensorFlow.

# Treinamento de Modelos

- **O "Motor" do Aprendizado:**
  - **Função de Perda (Loss Function):** Mede o quão distante a previsão do modelo está do resultado real (ex: MSE para regressão, Cross-Entropy para classificação).
  - **Gradiente Descendente:** Algoritmo de otimização que ajusta os pesos iterativamente para minimizar a função de perda.
- **Algoritmo de Backpropagation:**
  - Processo eficiente que calcula o gradiente do erro em relação a cada parâmetro da rede em duas passagens (frente e trás).
  - • • **Otimizadores:** Variantes para acelerar o treino (ex: SGD, Adam, RMSProp).

# A Solução: Perceptron Multicamadas (MLP)

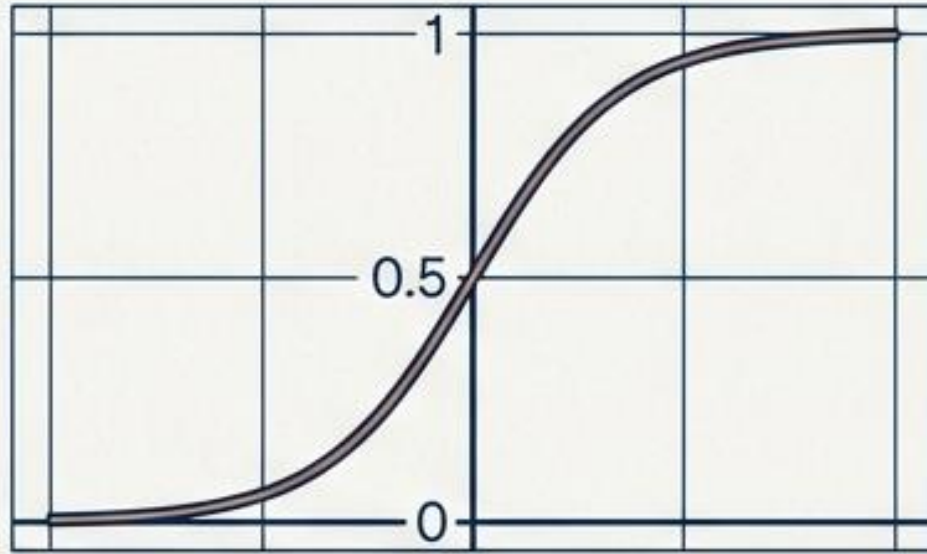


## The Engine

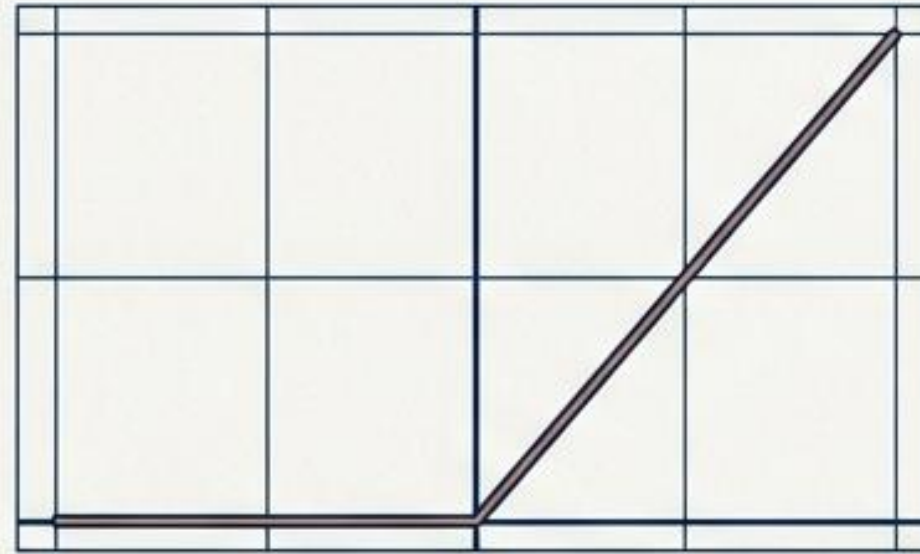
- **Backpropagation (Retropropagação):**
  1. **Forward Pass:** O dado flui e gera uma previsão.
  2. **Cálculo do Erro:** Comparação com o valor real.
  3. **Backward Pass:** O erro retorna pela rede.
  4. **Atualização:** Pesos ajustados via Gradiente Descendente.

$$w(n+1) = w(n) + \eta * \text{Error}$$

# Componentes Críticos: Ativação e Hiperparâmetros

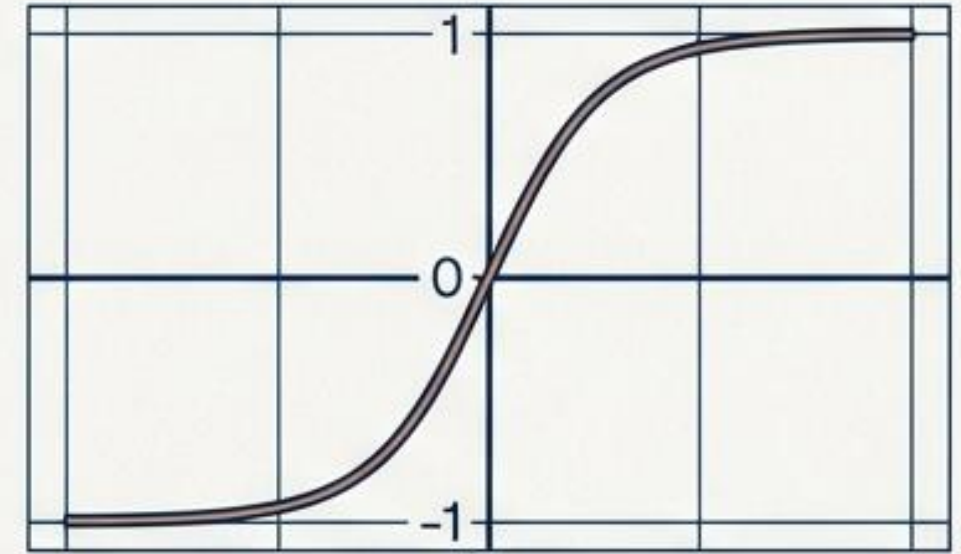


Sigmoid



ReLU

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



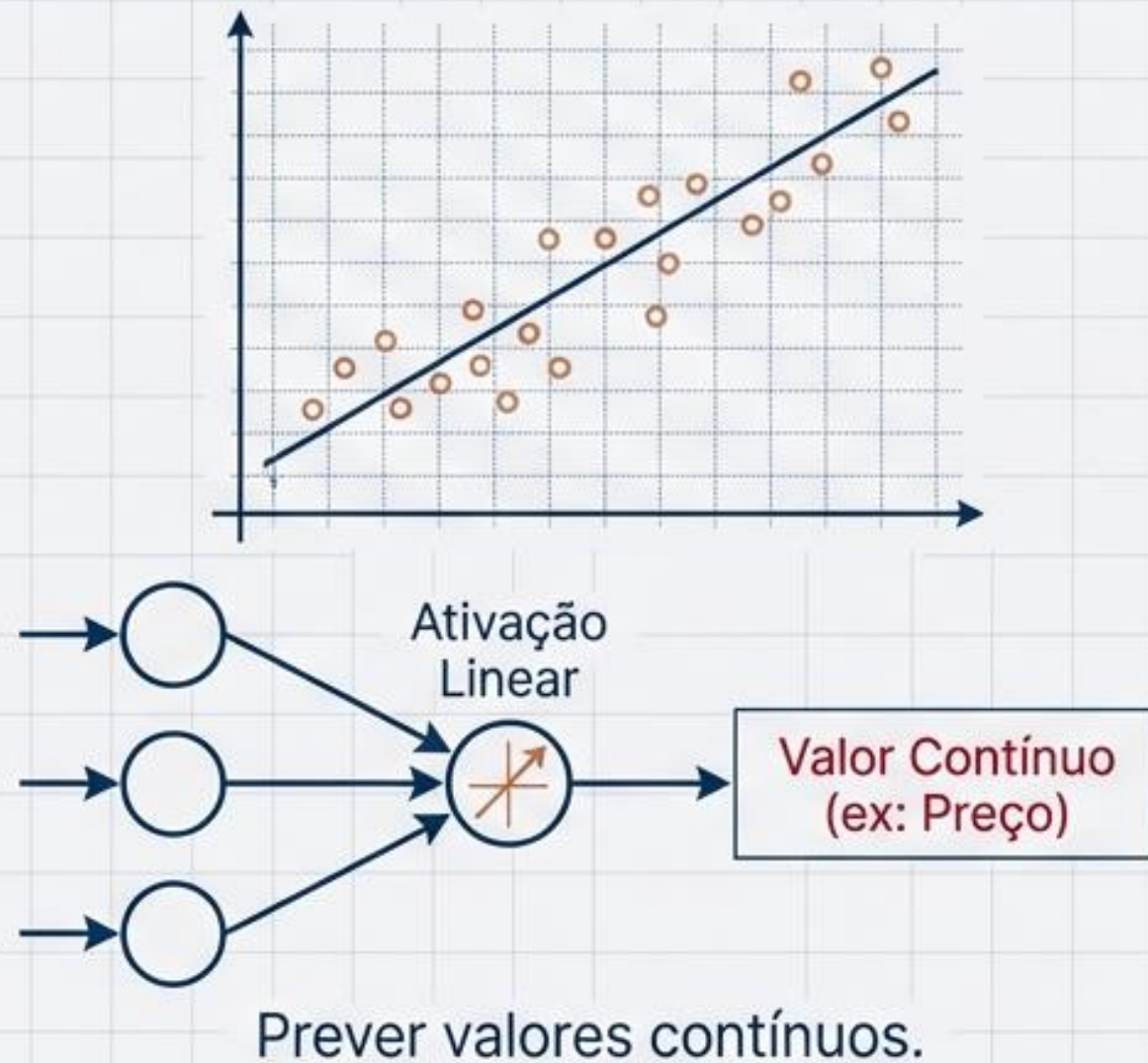
Tanh

**Hiperparâmetros** (Os “botões” de controle):

- **Learning Rate:** Tamanho do passo de ajuste. (Alto = Oscilação | Baixo = Lentidão)
- **Batch Size:** Número de exemplos processados antes de atualizar os pesos.
- **Epochs:** Ciclos completos através de todo o dataset.

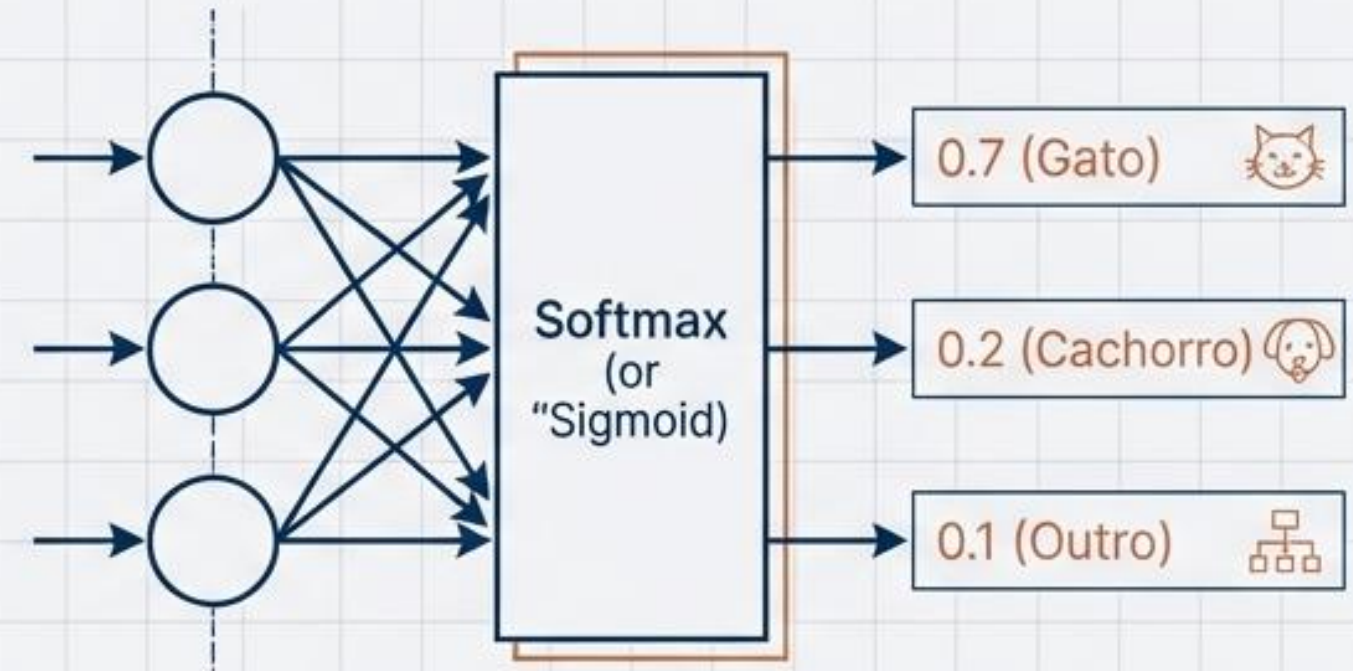
# Aplicações do MLP: Regressão vs. Classificação

## Regressão



Loss: **MSE** (Mean Squared Error)

## Classificação



Prever categorias (ex: Gato/Cachorro).

Ativação: **Softmax (Multiclasse)** ou **Sigmoid** (Binária).

Loss: **Cross-Entropy** (Entropia Cruzada)

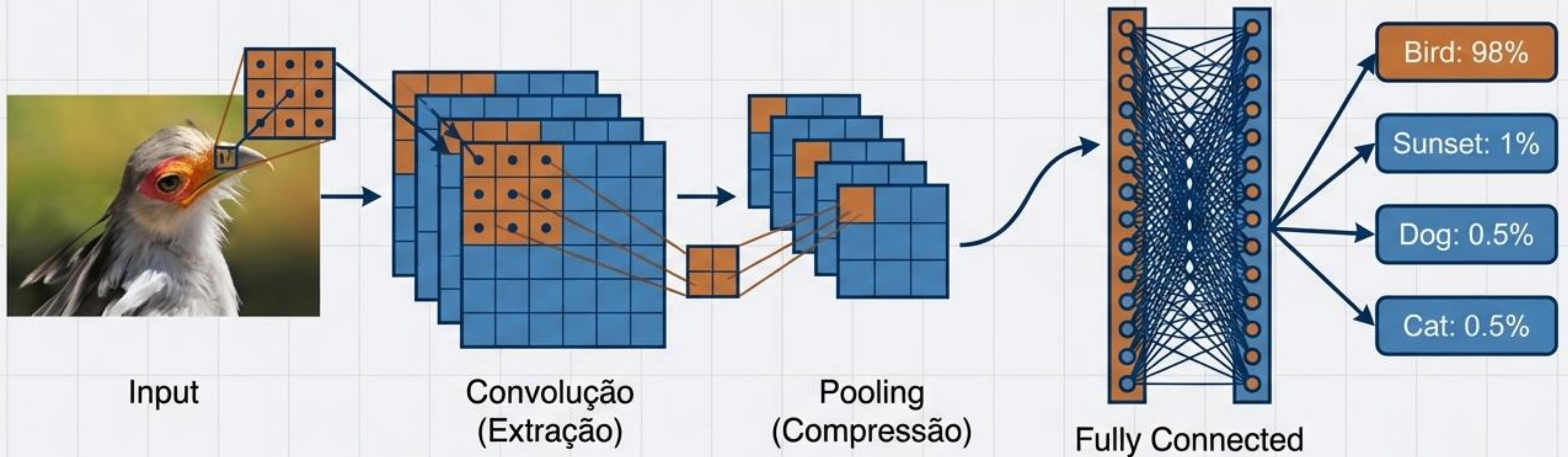
# Técnicas de Otimização e Regularização

- **Melhorando o Desempenho:**
- **Batching:** Divisão dos dados em pequenos lotes (mini-batches) para treinamento mais eficiente e estável.
- **Regularização:** Técnicas para evitar *overfitting* (quando o modelo decora os dados de treino mas não generaliza).
  - Exemplos: Dropout (desligar neurônios aleatoriamente), Regularização L1/L2.
- **Ajuste de Hiperparâmetros:** Taxa de aprendizado, número de camadas e número de neurônios.

# Arquiteturas Especializadas - CNNs

- **Redes Neurais Convolucionais (CNNs):**
- Foco: Processamento de dados em grade, como imagens.
- **Camadas Principais:**
  - *Convolução*: Extração de características locais (bordas, texturas) usando filtros.
  - *Pooling*: Redução da dimensionalidade e invariância a pequenas translações.
- **Aplicações:** Classificação de imagens, detecção de objetos, segmentação semântica.

# Visão Computacional: Redes Neurais Convolucionais (CNN)



**Inspiração:** Córtex Visual. Filtros detectam bordas, texturas e formas.

**Arquitetura:** Input → Convolução (Extração) → Pooling (Compressão) → Classificação.

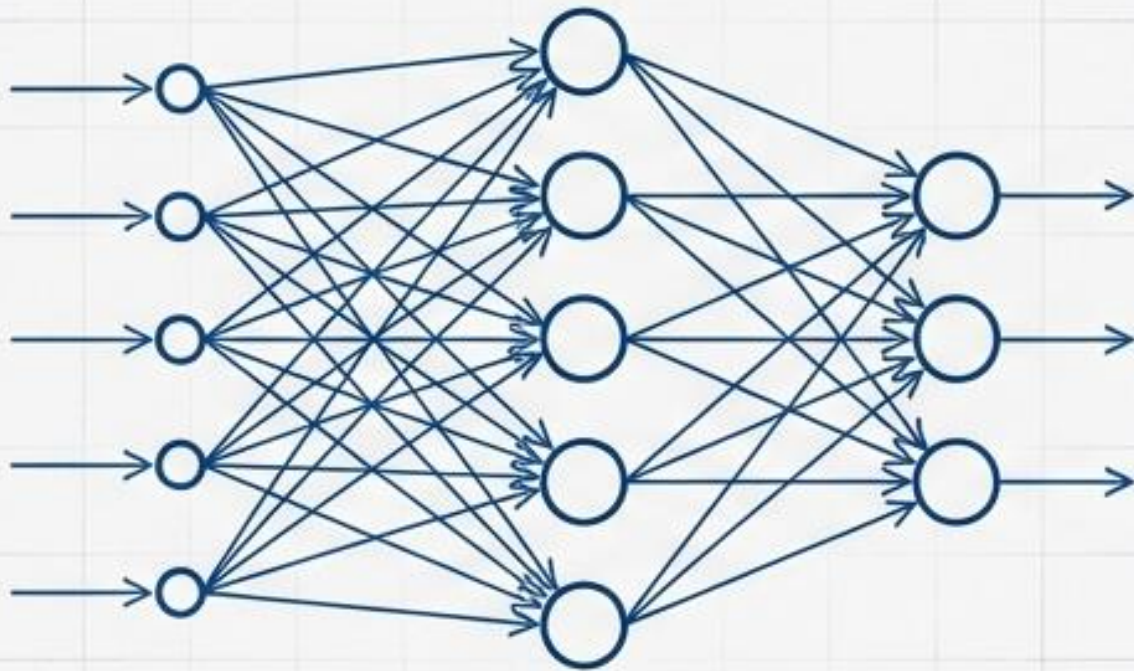
**Aplicação:** Processamento de imagens e vídeos.

# Arquiteturas Especializadas - RNNs

- **Redes Neurais Recorrentes (RNNs):**
- Foco: Processamento de sequências (séries temporais, texto, áudio).
- **Características:** Possuem conexões "para trás" (loops), permitindo memória de estados anteriores.
- **Desafios:** Problema do gradiente que desaparece/explode em sequências longas.
- **Soluções:** Células LSTM (Long Short-Term Memory) e GRU.

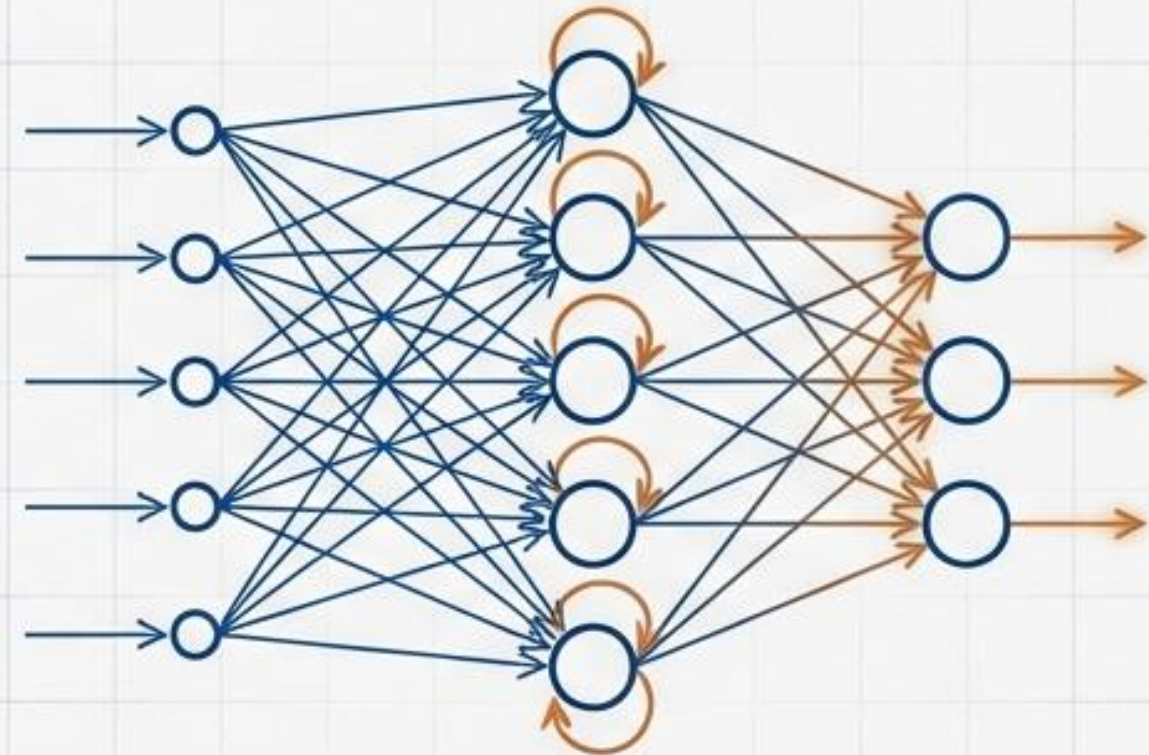
# Processamento de Sequências: Redes Recorrentes (RNN)

## Feed-Forward (Padrão)



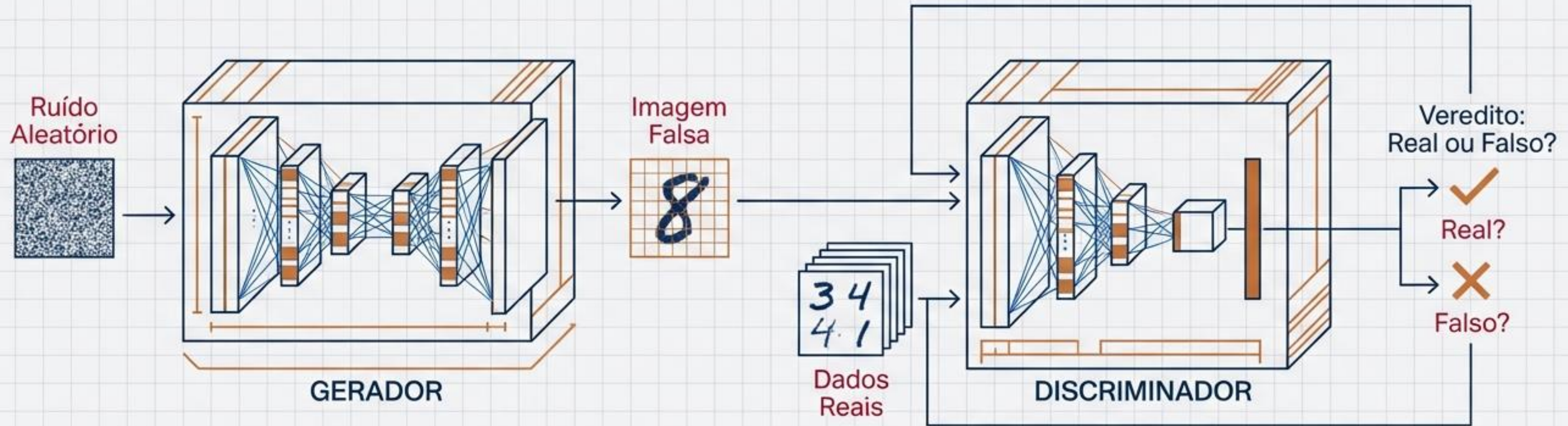
O sinal flui estritamente da entrada para a saída. Sem memória de estados anteriores.

## Recurrent (RNN)



**O Diferencial: Memória.** A saída anterior alimenta a próxima entrada (**Looping Arrow**).  
**Ideal para dados temporais:** Texto, Áudio, Séries Temporais (Bolsa de Valores).  
**Evolução:** LSTM & GRU (Arquiteturas que resolvem o problema de memória curta).

# Criatividade Artificial: Redes Generativas Adversárias (GANs)



O Duelo: Dois modelos competindo.

- **Gerador:** Tenta criar falsificações perfeitas.
- **Discriminador:** O detetive que tenta identificar a fraude.

Objetivo: **Equilíbrio de Nash** (O falso torna-se indistinguível do real).

# Aprendizado por Reforço (Reinforcement Learning)

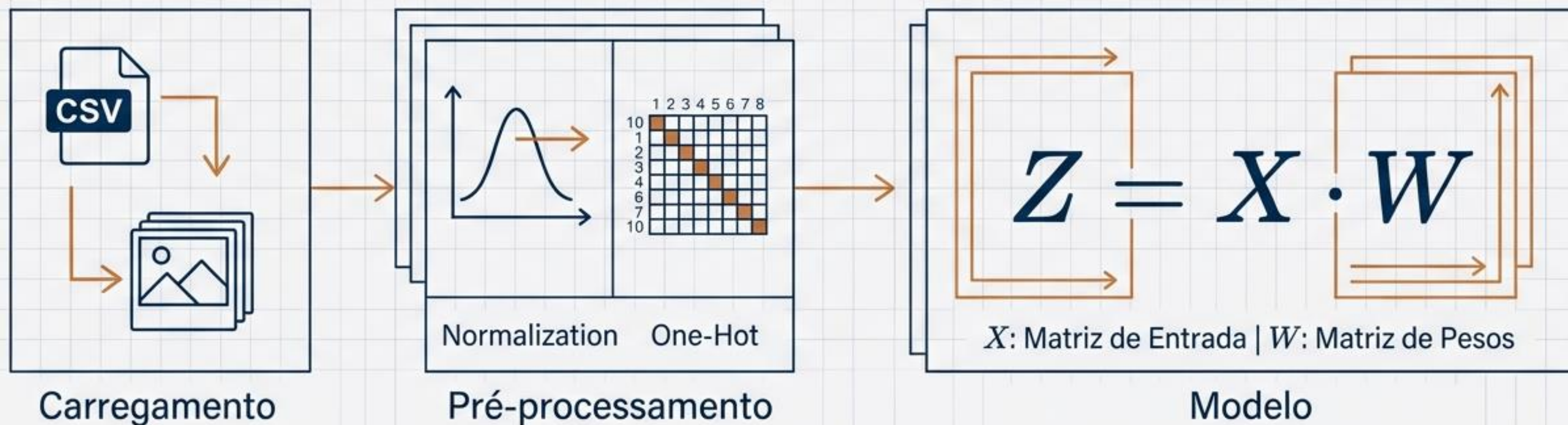


- **Conceito:** Aprender por tentativa e erro, maximizando recompensas futuras.
- **Elementos:** Agente, Ambiente, Ação, Recompensa.
- **Base Matemática:** Processo de Decisão de Markov (MDP).

# Ferramentas e Avaliação

- **Frameworks de Desenvolvimento:**
- **TensorFlow & Keras:** API de alto nível para construção rápida e escalável de modelos.
- **PyTorch:** Biblioteca dinâmica popular para pesquisa e produção.
- **Métricas de Avaliação:**
- **Acurácia:** Taxa de acertos gerais.
- **Precisão e Recall:** Importantes para classes desbalanceadas.
- **Curva ROC e AUC:** Avaliação de trade-offs em classificação binária.

# Implementação Prática: Frameworks e Dados

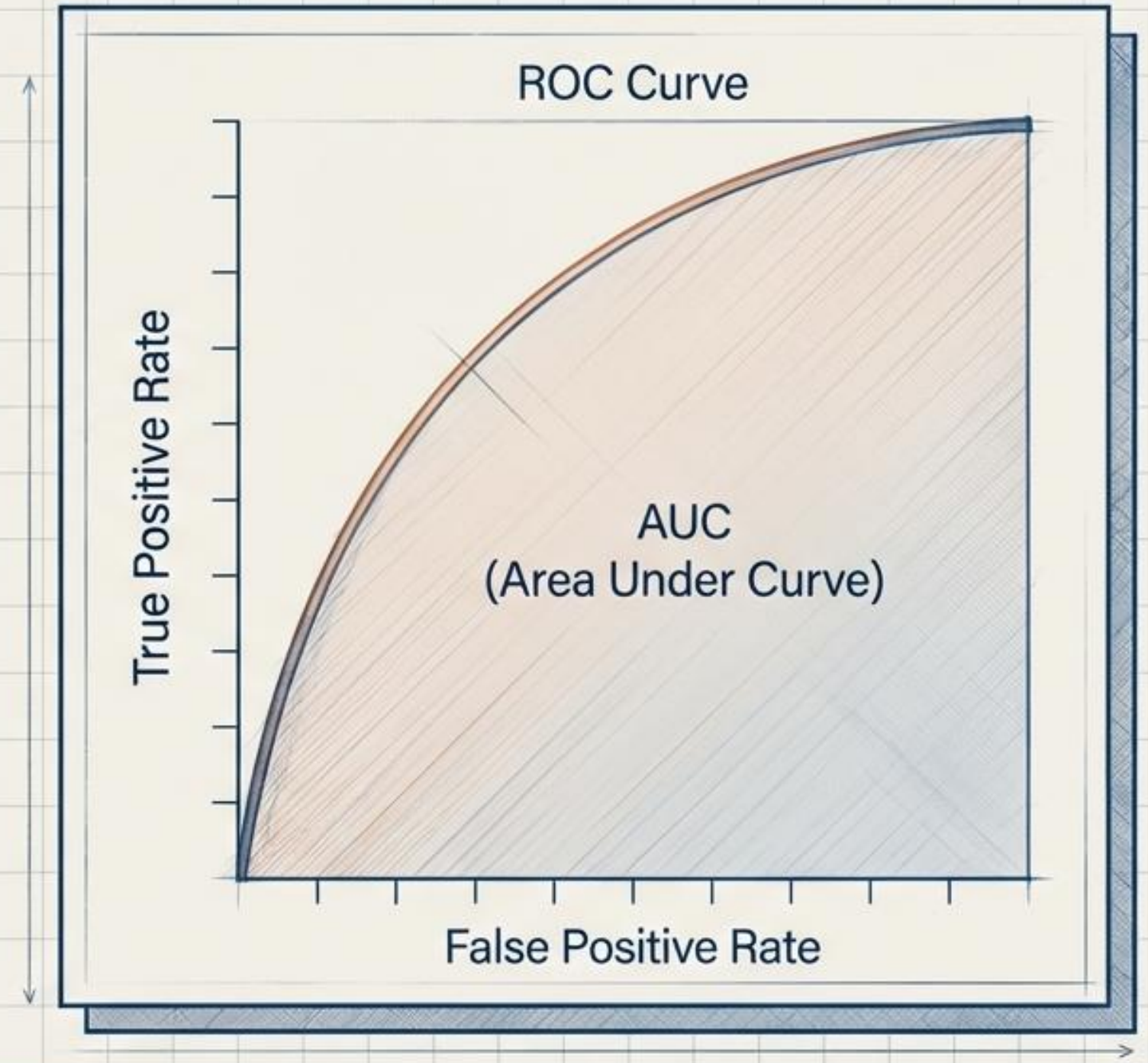


Ferramentas: TensorFlow e PyTorch (Padrões da Indústria).

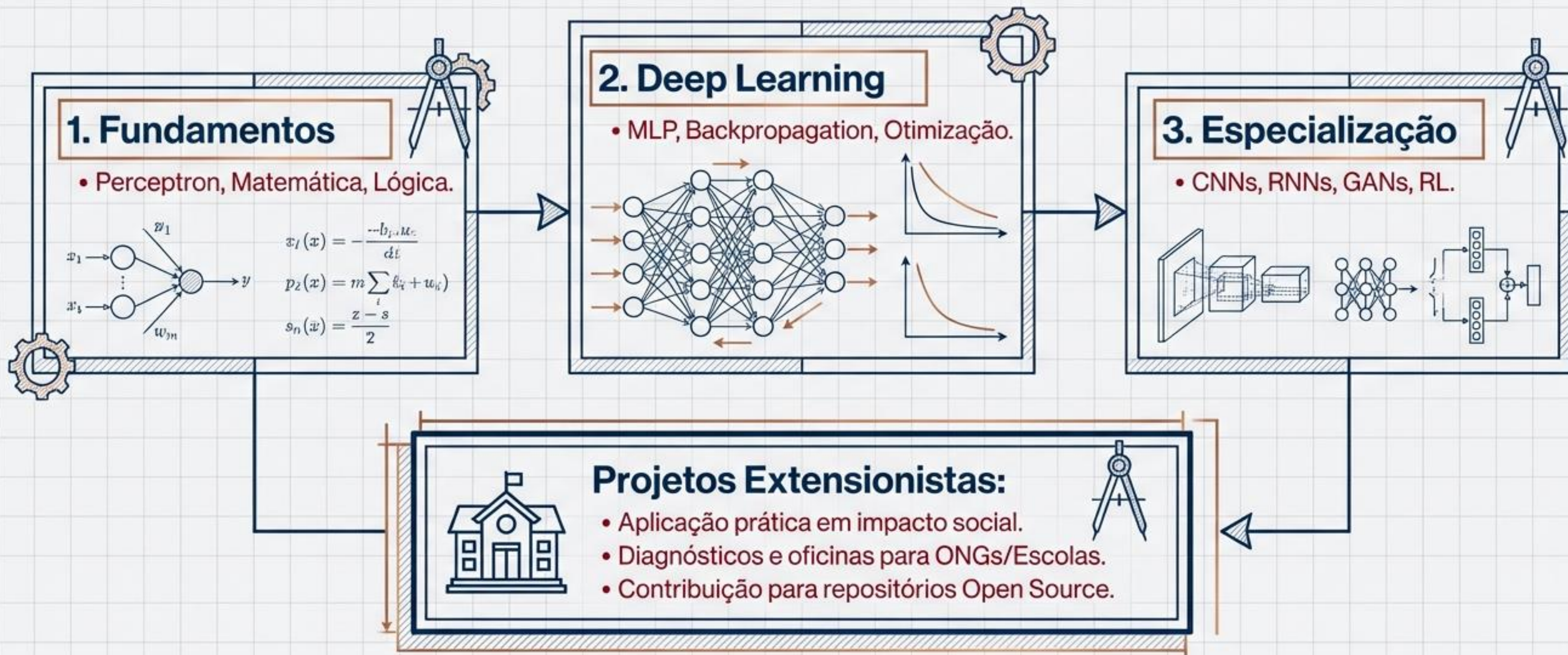
Processo: O dado deve ser normalizado e transformado em tensores antes do treinamento.

# Avaliação de Modelos: Métricas de Sucesso

- Acurácia: % de acertos totais.
- Precisão: Qualidade dos positivos encontrados.
- Recall: Quantidade de reais positivos detectados.
- F1-Score: Média harmônica (Precisão + Recall).



# Roteiro do Curso e Próximos Passos



**Objetivo Final: Unir rigor teórico com transformação prática.**

**Obrigado!**