

Inteligência Artificial Aplicada à Física

Fundamentos e Aplicações com Redes Neurais

AULA 1

Alexandre Suaide

Escola Jayme Tiomno
IFUSP 2025

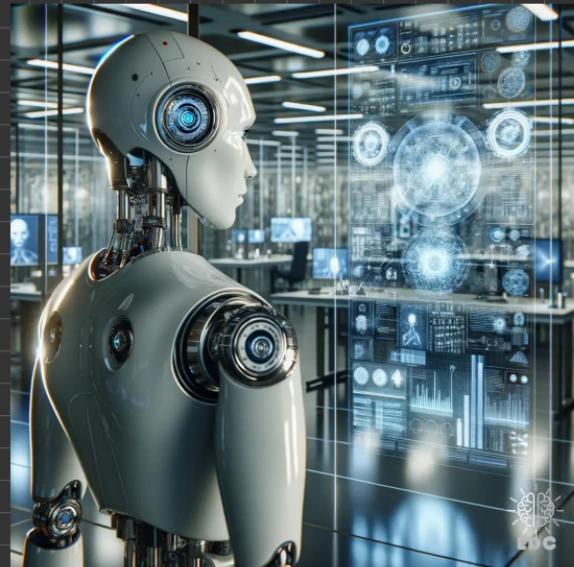
Estrutura desse mini-curso

Apresentar algumas noções básicas de IA, com foco em redes neurais, em problemas típicos de física (experimental ou teórica)

- 5 aulas de 2 horas, aproximadamente
 - Aula 1 - Introdução à IA e ML
 - Aula 2 - Fundamentos de redes neurais
 - Aula 3 - Redes neurais convolucionais (CNN)
 - Aula 4 - Redes recorrentes (RNN) e séries temporais
 - Aula 5 - Redes informadas por física (PINN)
- Sempre que possível teremos uma atividade prática

O que é Inteligência Artificial

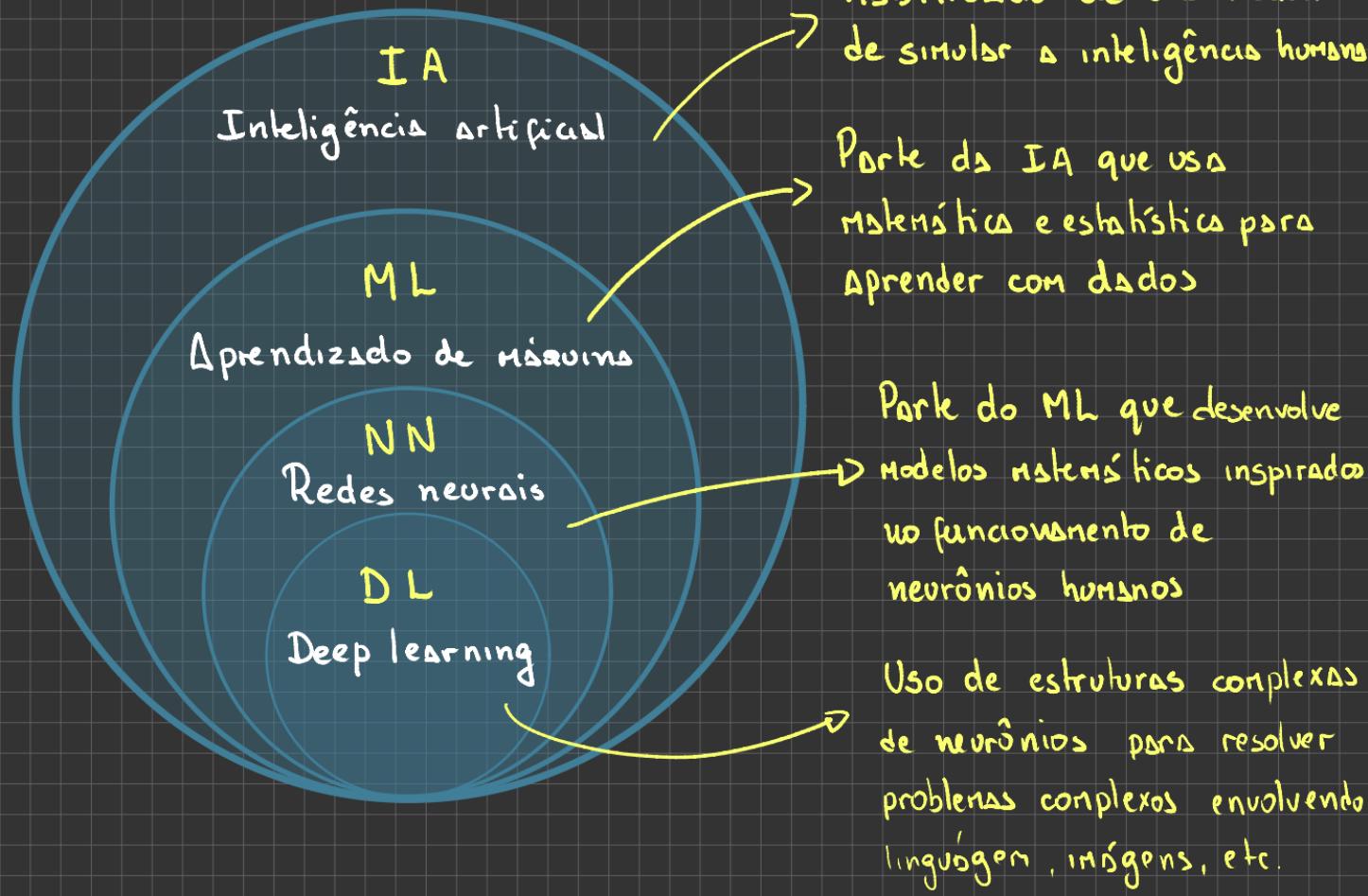
"Capacidade de uma máquina ou sistema computacional habilidades cognitivas humanas, como raciocínio, Aprendizado, planejamento e resolução de problemas" - Google IA



Machine Learning (ML) Aprendizado de máquinas

"Área da inteligência artificial que se concentra no desenvolvimento de algoritmos que os sistemas aprendem com os dados sem serem explicitamente programados"

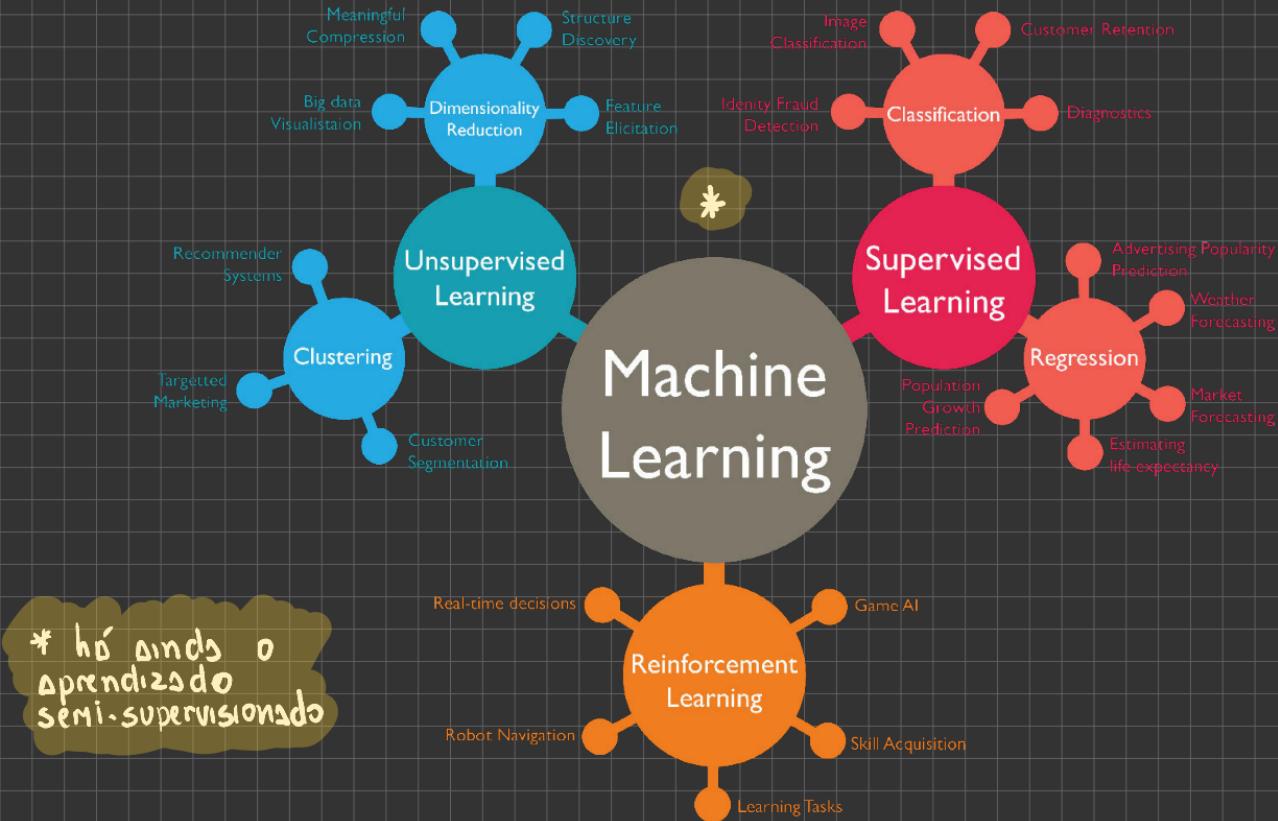
Parece próximo do
nossa dia-a-dia!



Tipos de Aprendizado de Máquina

- Aprendizado não supervisionado
 - Dados não rotulados
 - Descobrir agrupamentos, similaridades
 - Ex: Agrupamento de galáxias por similaridade, separação de materiais por semelhanças químico-física, etc
- Aprendizado supervisionado
 - Dados rotulados
 - Aprender um método que mapeie entrada → saídas
 - Ex: Identificação de pontícuas, evolução temporal de sistemas, identificação de fases da matéria, etc
- Aprendizado por reforço
 - tentativas e erro → recompensas
 - Ex: robôs aprendendo a andar, jogar xadrez, etc

Tipos de Aprendizado de Máquina



Redes Neurais Artificiais (NN)

- Inspiração biológica - Neurônio humano

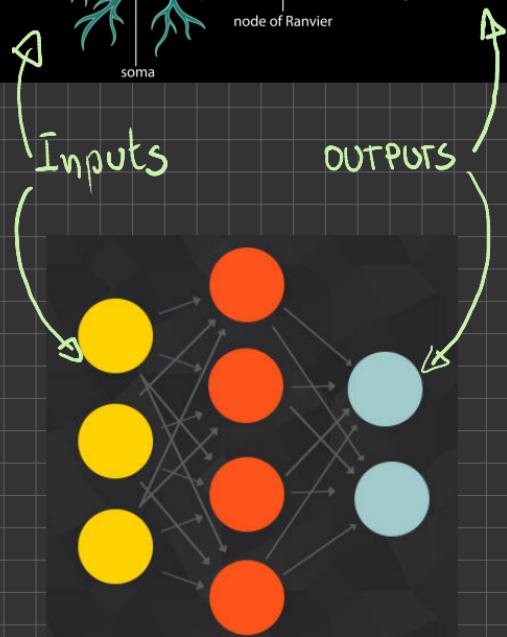
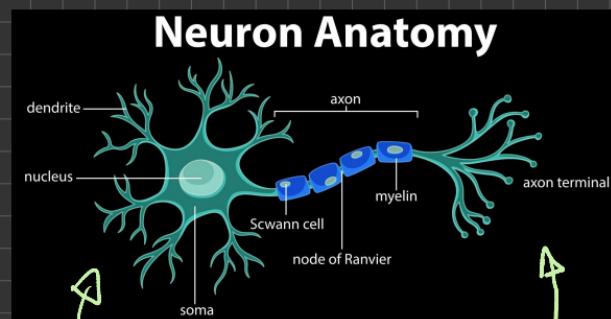
Cérebro humano

$\sim 10^{11}$ neurônios

$\sim 10^3$ conexões / neurônio

\downarrow
 $\sim 10^{14}$ conexões

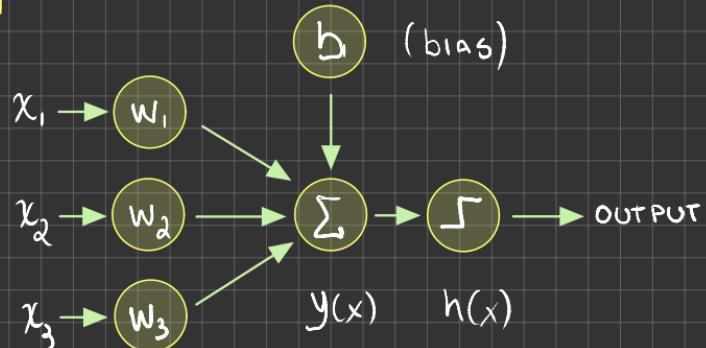
GPT-4 $\Rightarrow 10^{14}$ parâmetros



Tudo começou com o Perceptron

$$y(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

$$h(\vec{x}) = \begin{cases} 1 & \text{se } y(\vec{x}) > 0 \\ 0 & \text{se } y(\vec{x}) \leq 0 \end{cases}$$



Criado em ~1943

MARK I perceptron
Machine - 1957

400 input (20×20)

512 hidden

8 outputs



O Perceptron de 1943

McCulloch - Pitts

↳ Primeiro modelo matemático
de um neurônio

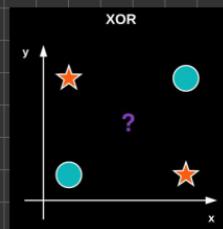
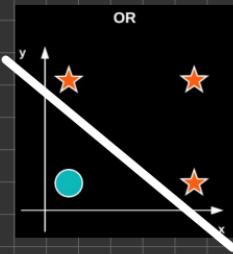
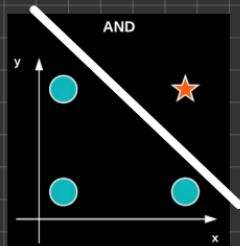
Inputs booleanos (0 ou 1)

Pesos iguais em todas
conexões (w)

$h(x)$ feito na não

Rosenblatt

Diferentes pesos (w)

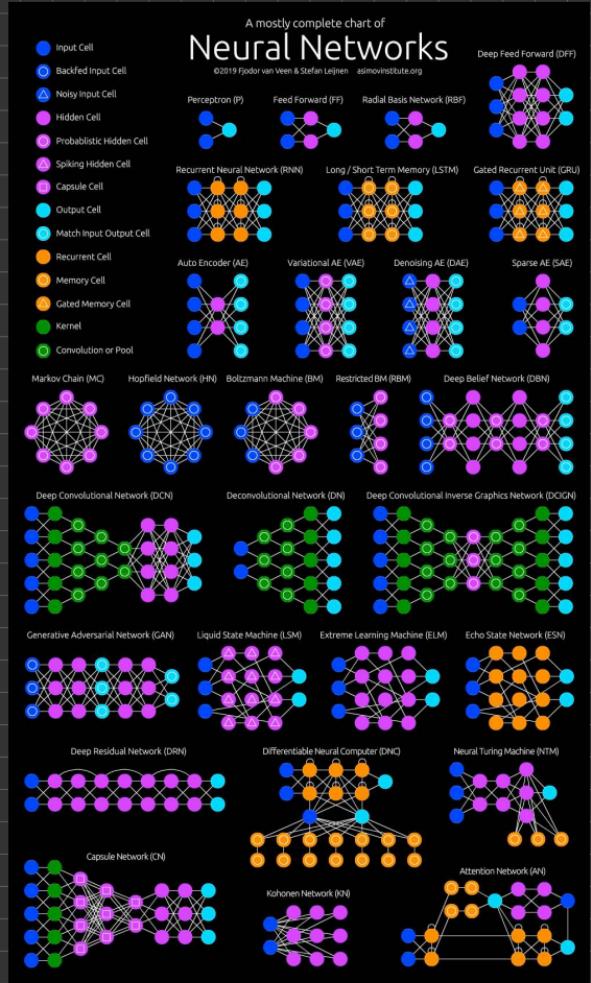


Esse modelo resulta em uma

Rede linear \rightarrow NÃO APRENDE \Rightarrow
XOR

Ideia ficou esquecida até
anos 80 / 90

Tipo de Rede	Características	Principais Aplicações
MLP (Perceptron Multicamadas)	Rede totalmente conectada, camadas densas, funções de ativação não lineares.	Classificação, regressão, previsão de séries temporais simples.
CNN (Convolutional Neural Networks)	Camadas de convolução e pooling, extração de padrões locais.	Reconhecimento de imagens, visão computacional, diagnóstico médico por imagem.
RNN (Recurrent Neural Networks)	Conexões recorrentes, memória de curto prazo.	Texto, fala, séries temporais.
LSTM / GRU	Variantes de RNN com portas para memória de longo prazo.	Tradução, chatbots, previsão de sequências longas.
Transformers	Mecanismo de atenção, paralelismo, dependências globais.	Modelos de linguagem (ChatGPT, BERT), tradução automática, IA gerativa multimodal.
Autoencoders	Estrutura encoder-decoder, aprendizado não supervisionado.	Redução de dimensionalidade, detecção de anomalias, compressão de dados.
GANs (Generative Adversarial Networks)	Duas redes (gerador e discriminador) competindo.	Geração de imagens realistas, deepfakes, arte e design.
Deep Reinforcement Learning	Combina aprendizado por reforço com redes profundas.	Jogos (AlphaGo), robótica autônoma, otimização de processos.
SNNs (Spiking Neural Networks)	Neurônios disparam pulsos (spikes), inspiradas no cérebro.	Computação neuromórfica, robótica de baixo consumo energético.
Redes Híbridas / Modulares	Combinação de arquiteturas (ex.: CNN+RNN, Transformer multimodal).	Sistemas multimodais (texto+imagem+áudio), medicina de precisão, recomendações.



Uma rede precisa aprender → treinamento

Independentemente da arquitetura, toda rede começa seu conhecimento

- CONEXÕES ALEATÓRIAS ENTRE NEURÔNIOS

Treinar uma rede significa determinar como os neurônios se conectam de forma eficiente para a realização da tarefa

- Determinar pesos, biases, etc

"Assim como um aluno aprende estudando, fazendo exercícios, etc
uma rede aprende realizando previsões e comparando-as às respostas corretas"

O que é necessário p/ treinar uma rede

• Dataset

- Conjunto de dados cuja entrada e saída (resultado) sejam conhecidos

	wolf		dog
	wolf		dog
	wolf		dog
	wolf		wolf

- O dataset é normalmente dividido em 3 conjuntos
 - treino (~70%) - Usado para ajustar os pesos
 - validação (~15%) - Usado durante o treino p/ verificar se a rede está aprendendo de forma geral
 - teste (~15%) - Usado apenas no final p/ avaliar o desempenho global

Preparação dos dados

- **NORMALIZAÇÃO** - Colocar todas entradas na mesma escala
Ex: valores entre 0 e 1 para intensidade de cor
- **Balanceamento** - Verificar se há quantidade semelhante de dados de mesmas categorias
Ex: nº lobos ~ nº cães
 - Evita que a rede se especialize em uma categoria
- **Data Augmentation** - Gerar novos dados p/ generalizar ainda mais
Ex: imagens → girar, espelhar, mudar escala, etc

O processo de treinamento

1. FORWARD pass

A rede recebe uma entrada e gera uma saída

2. Cálculo do erro

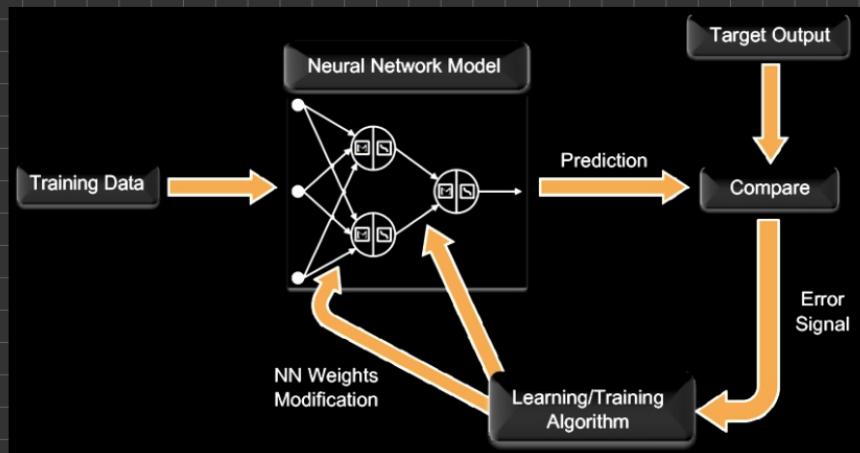
A saída é comparada com a resposta correta e um erro é computado

3. Backpropagation

O erro é propagado de volta pela rede, ajustando os pesos p/ minimizá-lo usando algum algoritmo de otimização

4. EPOCHs

O processo é repetido inúmeras vezes até a rede aprender



O software p/ aprendizado de máquina

- Python é uma linguagem de programação
 - Amplamente usada em ML
 - Algunas características
 - Interpretada
 - Orientada a objetos
 - Escalável - para qualquer tamanho de projeto
 - tipagem dinâmica
 - tratamento de exceções
 - Coletor de lixo
 - Muita, mas muita mesmo, informação disponível
 - tem também o R, JAVA, C++
 - Não vamos usar, mas é possível

Um porém em relação ao Python

- O desenvolvimento de algoritmos de ML é preferencialmente escrito em linguagens compiladas

→ Performance

Ex: tensorflow → c++

- Esses algoritmos, etc → bibliotecas

Numpy - dados e operações c/ matrizes. Matemática

Pandas - tabelas, datasets, planilhas

Scipy - Métodos numéricos p/ ciência (ajustes, interpolações, EDO, etc)

Matplotlib - Gráficos

Scikit-learn Bibliotecas de ML genéricas

KERAS - Interface com tensorflow (GOOGLE)

PyTorch - " " torch (META)

Ferramentas que vamos utilizar

- Interface para as atividades

- Preferencialmente Google colab

<https://colab.google.com>

↳ Se você tiver um notebook com jupyter instalado
pode usar também

- Repositório c/ atividades, aulas, datasets

- Github

<https://github.com/suaide/jt2025>