

Introdução à Redes Neurais

Panorama, conceitos e prática

Enzo L. Fernandes

Universidade Estadual Paulista – *Campus* Botucatu

November 29, 2021

1 Aprendizado de Máquina

2 Conceitos

3 Redes Neurais

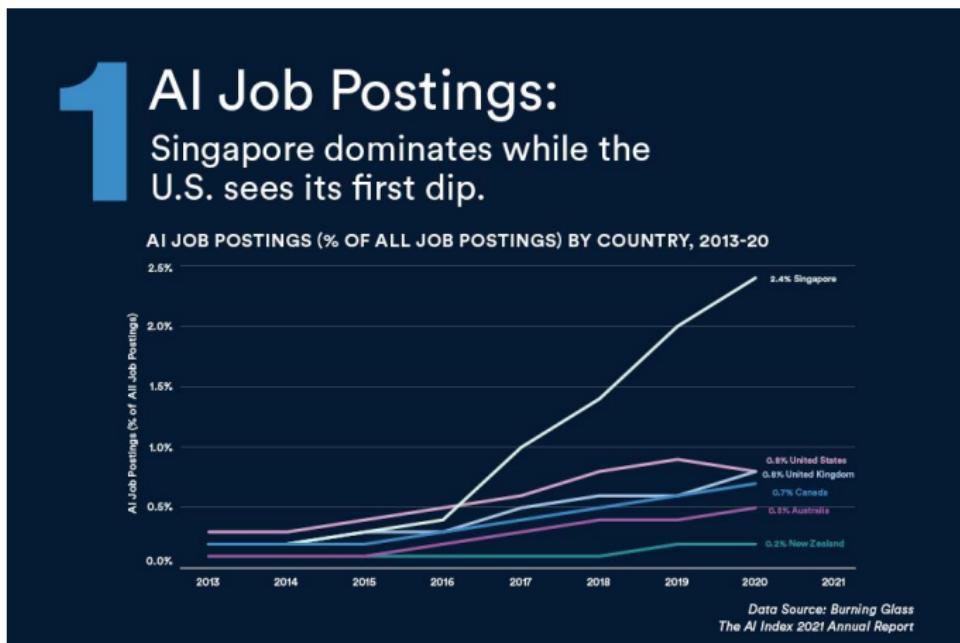
Parte 1: Aprendizado de Máquina

Demandas e Interesse

- Automação
- Extração de Conhecimento
- Processamento de alto volume de dados
- Estado da Arte em diversas Áreas de Pesquisa
- Crescentes investimentos no Indústria

Demanda e Interesse

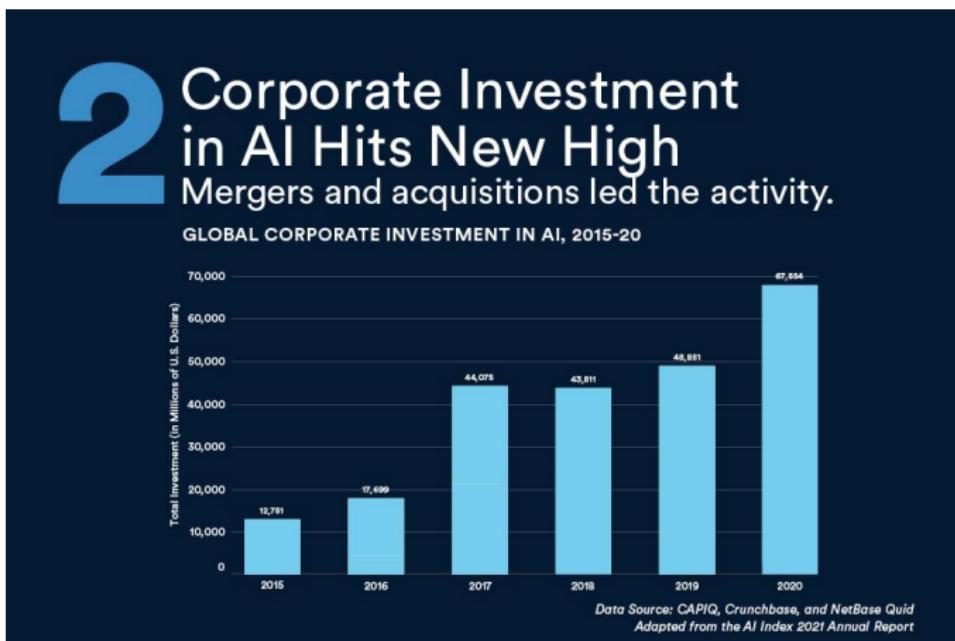
Segundo o 2021 AI Index (Stanford University):



Fonte: <https://hai.stanford.edu/news/state-ai-10-charts>

Demanda e Interesse

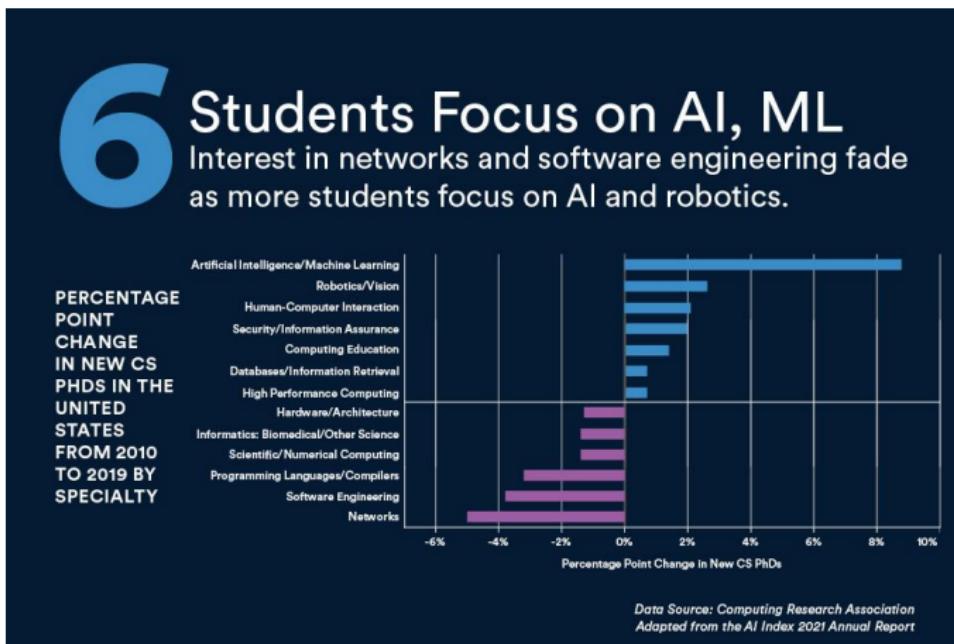
Segundo o *2021 AI Index* (Stanford University):



Fonte: <https://hai.stanford.edu/news/state-ai-10-charts>

Demanda e Interesse

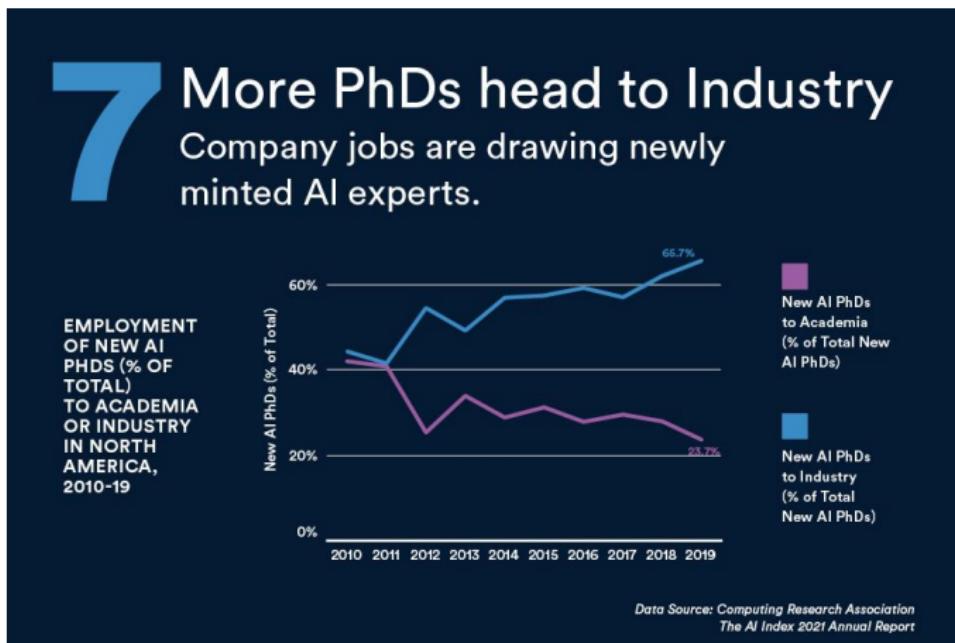
Segundo o *2021 AI Index* (Stanford University):



Fonte: <https://hai.stanford.edu/news/state-ai-10-charts>

Demanda e Interesse

Segundo o *2021 AI Index* (Stanford University):



Fonte: <https://hai.stanford.edu/news/state-ai-10-charts>

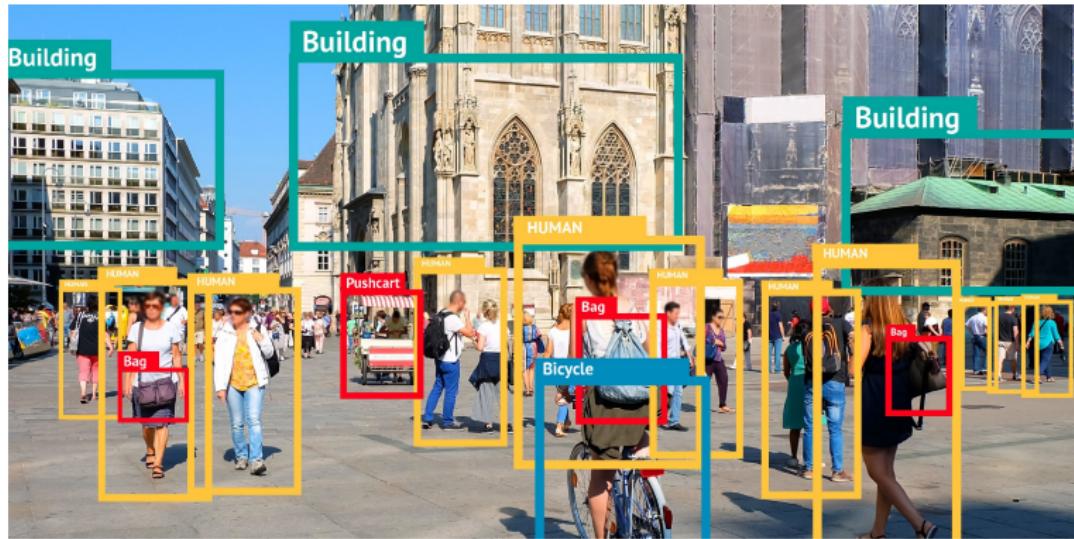
Aplicações

Carros autônomos e navegação inteligente



Aplicações

Detecção de objetos e segmentação de imagens (Visão Computacional)



Aplicações

Robótica

Aplicações

Geração de conteúdo (*Google Deep Dream*)



Parte 2: Conceitos

Aprendizado de Máquina significa utilizar algoritmos para resolver problemas *sem que eles tenham sido explicitamente programados para resolvê-los.*

Exemplo

Deseja-se *classificar* uma planta a partir de uma imagem em duas classes:
margarida ou girassol.

Abordagens possíveis:

- Criar um conjunto de regras
- Contratar especialistas
- ... ?
- Machine Learning!



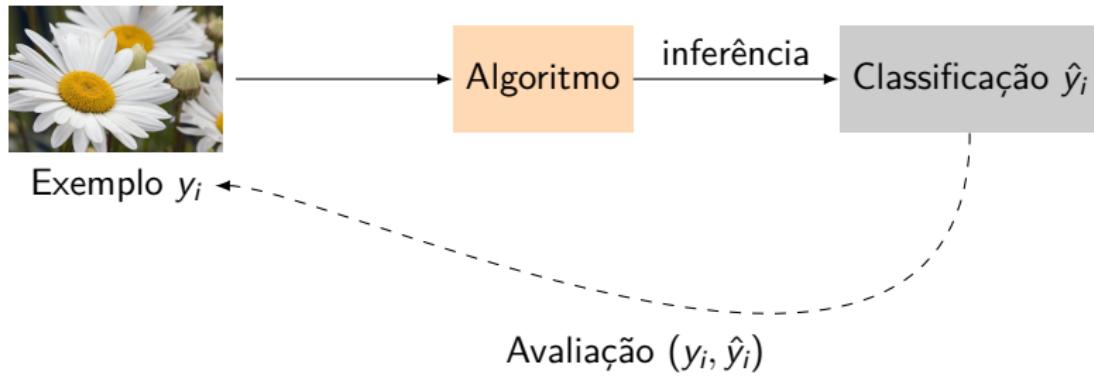
(a) Margarida



(b) Girassol

Mas então, como isso funciona?

Exemplo com uma instância de treinamento:



$$\text{Avaliação } (y_i, \hat{y}_i) = \begin{cases} \hat{y}_i = y_i, & \checkmark \\ \hat{y}_i \neq y_i, & \times \rightarrow \text{aplicar correções...} \end{cases}$$

O que é Aprendizado de Máquina?

“Um programa é dito aprender a partir da **experiência E**, em relação a uma classe de **tarefas T**, com medida de **desempenho P**, se seu desempenho em **T**, medido por **P**, melhora com **E**.”

– **Tom Mitchell**

No exemplo do Girassol/Margarida teremos que:

E: imagens de flores, cada qual com o respectivo rótulo;

T: associar corretamente cada imagem à respectiva espécie (rótulo);

P: quantidade de imagens classificadas corretamente.

Exemplo

Deseja-se estimar o preço de uma casa a partir de informações obtidas na tabela abaixo:

Metros Quadr.	Andares	nº Banheiros	Esquina	DDD	Preço (K R\$)
210	2	4	"Não"	15	350
315	2	5	"Sim"	11	515
120	1	2	"Não"	14	250
80	1	1	"Não"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
190	1	2	"Não"	15	220

Exemplo

Deseja-se estimar o preço de uma casa a partir de informações obtidas na tabela abaixo:

Metros Quadr.	Andares	nº Banheiros	Esquina	DDD	Preço (K R\$)
210	2	4	"Não"	15	350
315	2	5	"Sim"	11	515
120	1	2	"Não"	14	250
80	1	1	"Não"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
190	1	2	"Não"	15	220

- E:** informações sobre características de residências;
T: estimar o valor de uma residência dadas as suas características;
P: diferença entre o valor estimado e o valor real da residência.

Exemplo

Notação

Cada **coluna** da tabela corresponde a uma **atributo** ou *feature*.

Metros Quadr.	Andares	nº Banheiros	Esquina	DDD	Preço (K R\$)
210	2	4	"Não"	15	350
315	2	5	"Sim"	11	515
120	1	2	"Não"	14	250
80	1	1	"Não"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
190	1	2	"Não"	15	220

Exemplo

Notação

Cada **linha** da tabela corresponde a uma **amostra** (instância ou *sample*).

Metros Quadr.	Andares	nº Banheiros	Esquina	DDD	Preço (K R\$)
210	2	4	"Não"	15	350
315	2	5	"Sim"	11	515
120	1	2	"Não"	14	250
80	1	1	"Não"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
190	1	2	"Não"	15	220

Exemplo

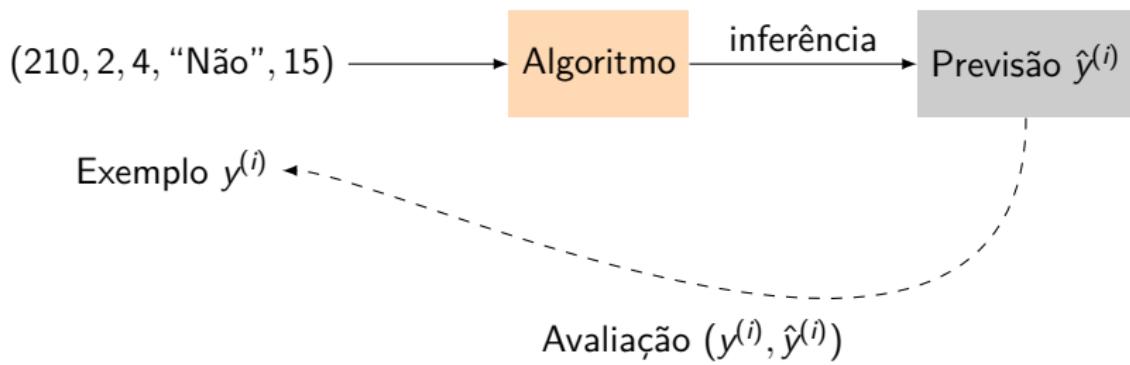
Notação

Atributo Meta (ou *target*) é o atributo que se deseja prever utilizando os modelos de Machine Learning.

Metros Quadr.	Andares	nº Banheiros	Esquina	DDD	Preço (K R\$)
210	2	4	"Nao"	15	350
315	2	5	"Sim"	11	515
120	1	2	"Nao"	14	250
80	1	1	"Nao"	11	180
900	3	7	"Sim"	11	850
245	2	3	"Sim"	14	450
215	1	3	"Sim"	11	310
190	1	2	"Não"	15	220

Exemplo

Exemplo com a primeira instância de treinamento da tabela de dados:



$$\text{Avaliação } (\mathbf{y}, \hat{\mathbf{y}}) = \text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

Formalismos

Definição

Utilizar Aprendizado de Máquina Supervisionado significa obter uma função-hipótese $h_{\theta}(\cdot)$ que aproxima uma função $f(\cdot)$ que descreve um fenômeno de interesse tal que:

$$f(x) = y$$

tomando como entradas x e y conhecidos *a priori*, mas sem que h_{θ} seja determinada explicitamente¹.

¹O modelo obtido pelos processos de treinamento é a função.

Formalismos: Entrada e Saída

Os dados obtidos obtidos *a priori* – também chamados de *dataset* – correspondem à:

- \mathbf{x} : matriz $m \times n$ em que cada linha corresponde a uma amostra de entrada e cada coluna um atributo.
- \mathbf{y} : matriz $m \times k$ em que cada linha corresponde ao valor esperado (ou valores esperados quando $k > 1$) para cada linha (amostra) em \mathbf{x} .

Ou seja, $h_{\theta}(\cdot)$ será tal que:

$$h_{\theta}: \mathbb{R}^n \rightarrow \mathbb{R}^k$$

Formalismos: Entrada e Saída

Em cada matriz \mathbf{x} e \mathbf{y} temos elementos $e_j^{(i)}$ nos quais:

- i corresponde à i -ésima amostra do conjunto e;
- j corresponde à j -ésimo atributo da amostra.

Assim temos:

$$\mathbf{x} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ \vdots & & \ddots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1^{(1)} & \dots & y_k^{(1)} \\ y_1^{(2)} & \dots & y_k^{(2)} \\ \vdots & \ddots & \vdots \\ y_1^{(m)} & \dots & y_k^{(m)} \end{bmatrix}$$

Formalismos: Treinamento

O processo de treinamento é feito ajustando os valores dos *parametros internos* θ em função de uma *função de custo* C que compara os valores previstos \hat{y} e os valores esperados y .

C depende do algoritmo utilizado e do tipo de tarefa realizada.

- regressão – e.g. Erro Quadrático Médio
- classificação – e.g. Entropia Cruzada

Desta forma o treinamento consiste em otimizar os valores dos parâmetros internos θ de modo a minimizar C :

$$\min_{\theta} C(\theta)$$

Tipos de Aprendizado

Os tipos de aprendizado podem ser divididos em:

- **Aprendizagem Supervisionada**
 - Regressão
 - Classificação
- **Aprendizagem Não-Supervisionada**
 - Redução de Dimensionalidade
 - Análise de Agrupamento
- **Aprendizagem por Reforço**
 - Agentes Autônomos

Pipeline de Projetos utilizando Machine Learning

- ① Coleta de Dados
- ② Preprocessamento
 - Limpeza dos Dados
 - *Feature Engineering*
 - *Feature Selection*
 - Normalização
- ③ Seleção dos Algoritmos de Treinamento
- ④ Ajuste de Hiperparâmetros
- ⑤ Validação e Avaliação de Performance
- ⑥ *Deployment*

Custo/Tempo de Desenvolvimento

De longe, as tarefas mais custosas estão em 1 e 2. As demais podem ser semi-automatizadas utilizando bibliotecas voltadas para Machine Learning.

Parte 3: Redes Neurais

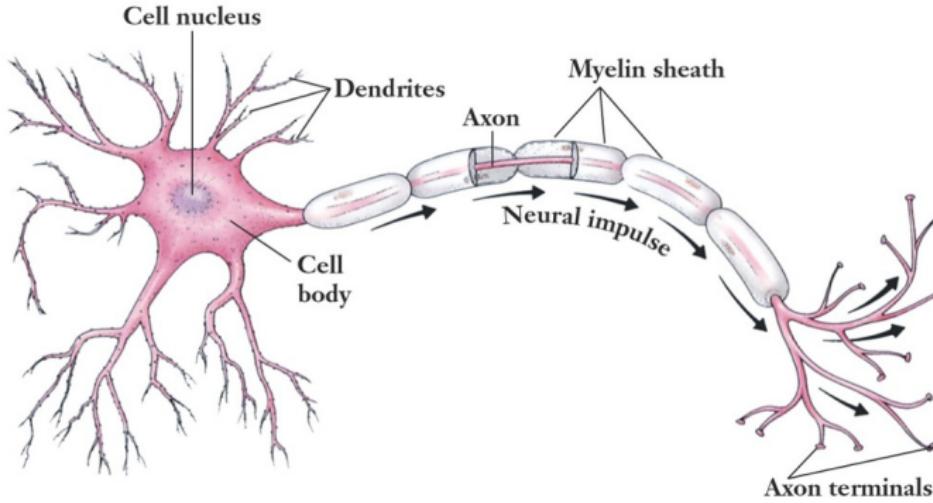
Redes Neurais

- Estado da Arte em diferentes aplicações e áreas de estudo:
 - Processamento de Linguagem Natural
 - Classificação de Videos e Imagens
- Surgiram como uma forma tentar emular o funcionamento do cérebro biológico
- Foram introduzidas em meados de 1950, mas caíram no esquecimento devido ao alto poder computacional necessário para treinar os modelos
- Com o aprimoramento do Hardware e das técnicas de treinamento as Redes Neurais “ressurgiram” em meados de 2000
- São indicadas para problemas envolvendo **alto volume de dados e muitos atributos**

Inspiração

Inspiração em *neurônios biológicos*

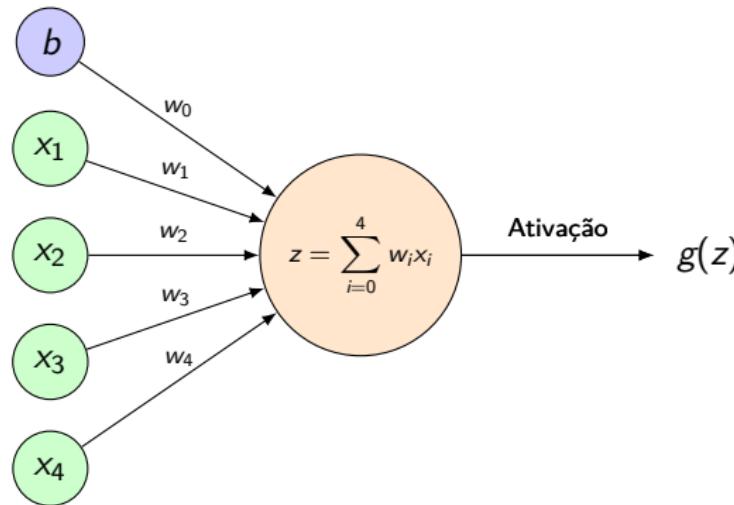
- ① Entrada
- ② Propagação de um impulso
- ③ Saída



Representação

A menor unidade de uma rede neural: **perceptron**².

Entrada Processamento Saída



²Também chamado de “Neurônio”.

Funções de Ativação

Diferentes *funções de ativação* podem ser utilizadas nas saídas dos *perceptrons*:

Logistica (*Sigmoid*)

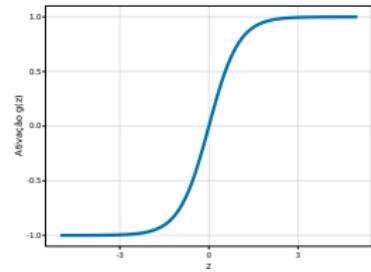
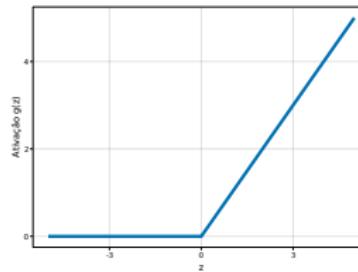
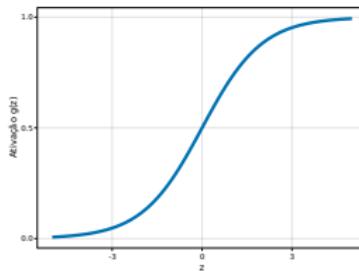
$$g(z) = \frac{1}{1 + \exp(-z)}$$

Rectified-Linear (ReLU)

$$g(z) = \max\{0, z\}$$

Tangente Hiperbólica

$$g(z) = \tanh(z)$$

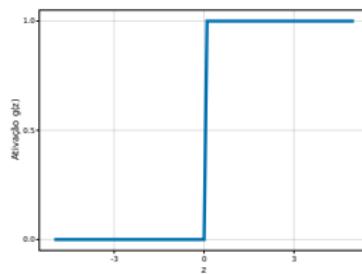


Funções de Ativação

Diferentes *funções de ativação* podem ser utilizadas nas saídas dos *perceptrons*:

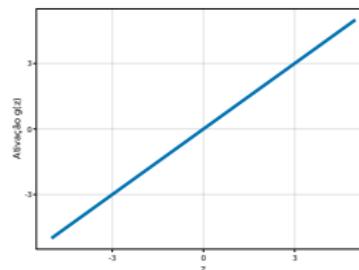
Limiar (*Step-Function*)

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



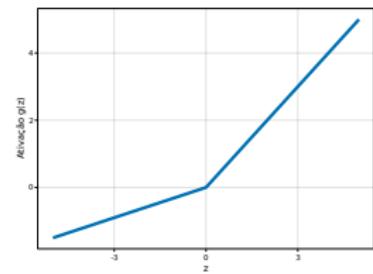
Linear (*Identity*)

$$g(z) = z$$



Leaky-ReLU

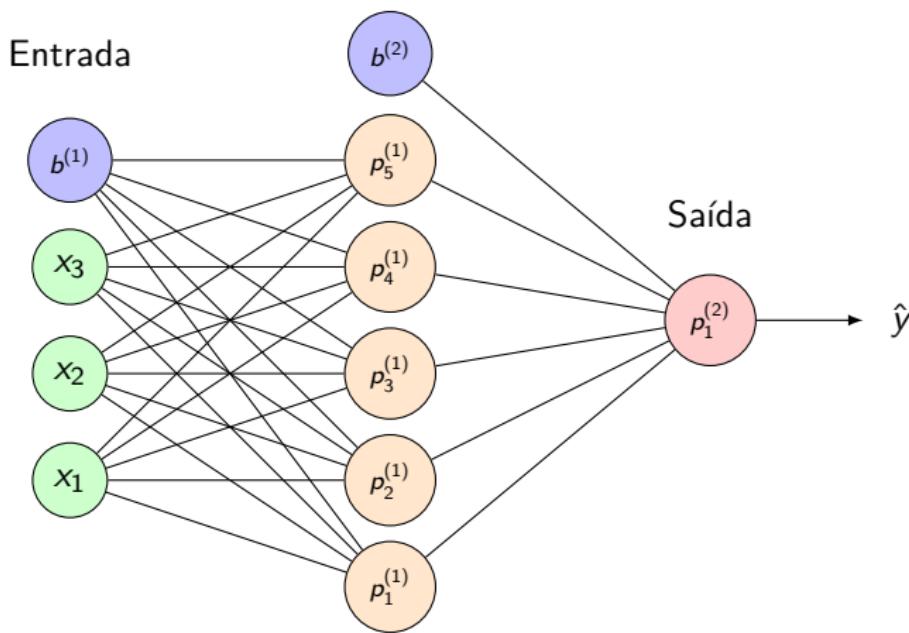
$$g(z) = \begin{cases} z \geq 0, & z \\ z < 0, & az \end{cases}$$



Estrutura Geral

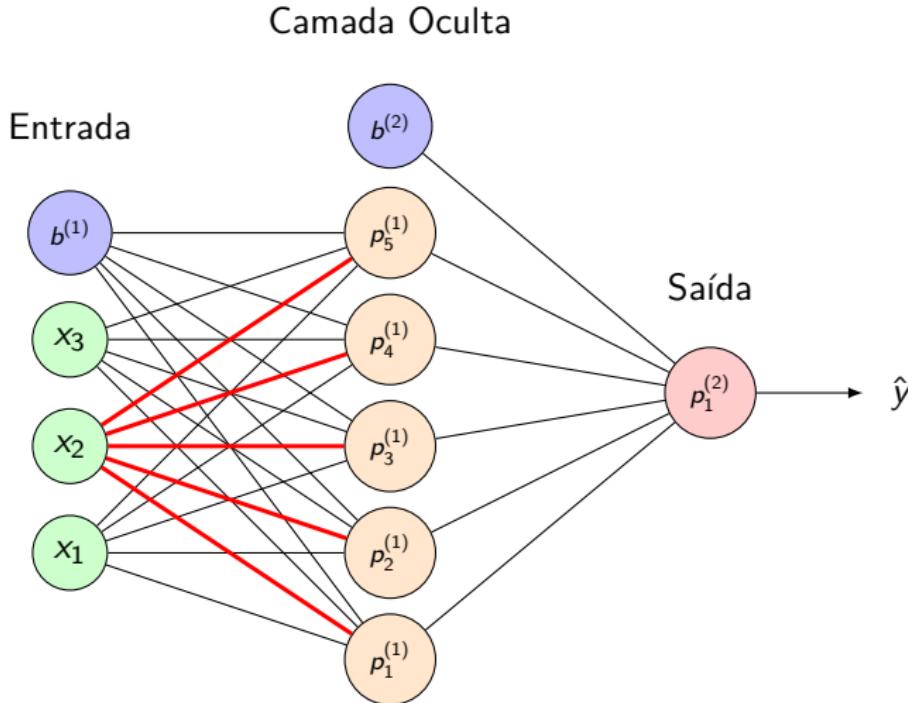
Uma Rede Neural é composta pela concatenação de diversos *perceptrons*:

Camada Oculta



Estrutura Geral

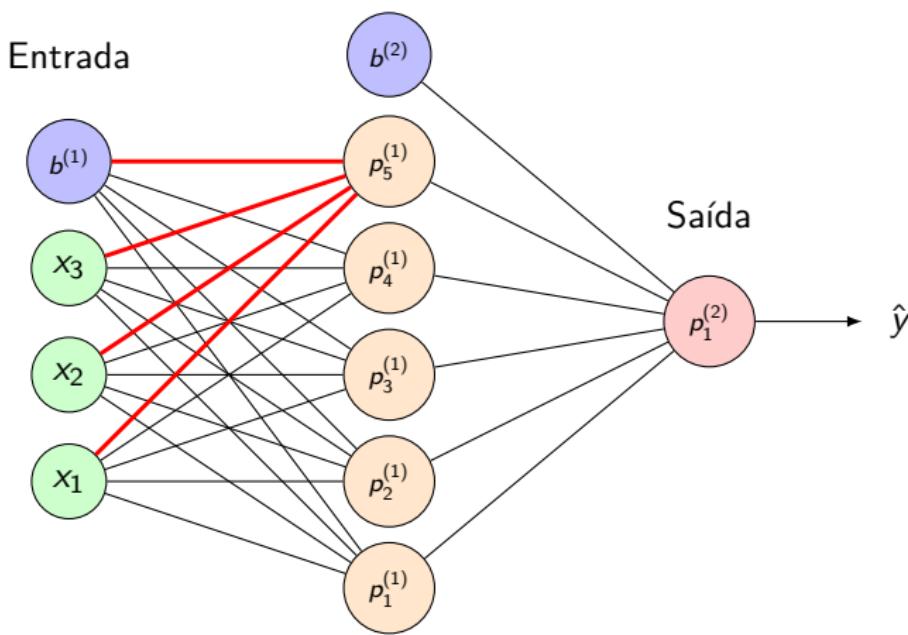
Cada atributo é conectado a cada neurônio da próxima camada:



Estrutura Geral

Cada neurônio recebe como entrada todas as saídas da camada anterior:

Camada Oculta



Camadas da Rede Neural

Disposição das camadas, comumente chamada de “Arquitetura” ou “Topologia” da rede neural.

① Camada de Entrada

- Primeira camada da rede neural
- As ativações são os valores dos atributos das amostras
- Número de neurônios = número de atributos na amostra³

② Camada Oculta:

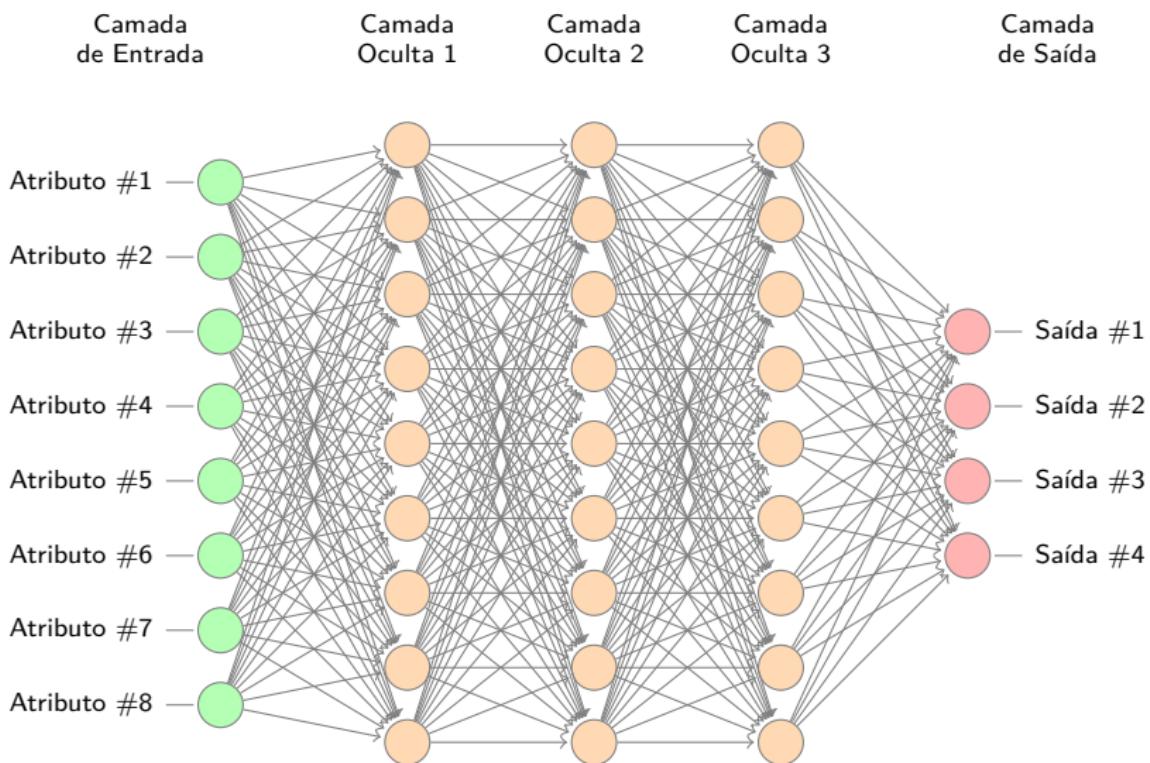
- Camada (ou camadas) entre a camada de entrada e saída
- Contém um número arbitrário k de neurônios cada
- Diferentes valores de k podem ser escolhidas para cada camada
- Parte do nosso trabalho consiste em encontrar valores ótimos de k

③ Camada de Saída:

- Última camada da rede
- Número de neurônios = número de atributos esperados na saída

³Para efeitos de implementação não são considerados os *bias* quando é definida a entrada da rede.

Exemplo



Treinamento

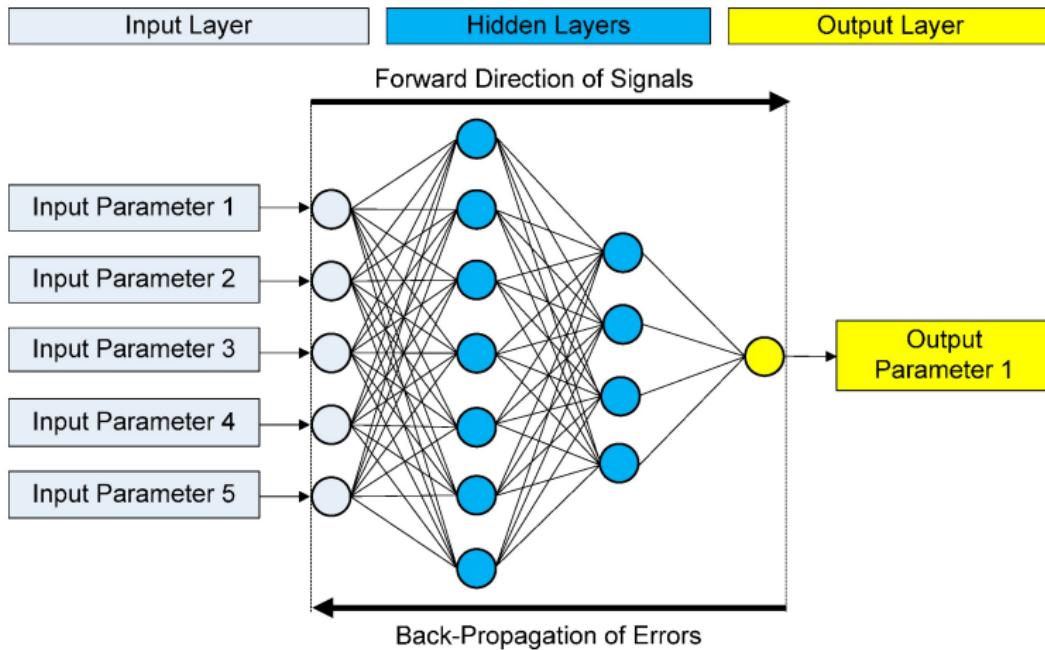
As Redes Neurais são treinadas de forma iterativa em 2 etapas que se repetem:

- ① **Feed-Forward:** as amostras são processadas da camada de entrada até a camada de saída da rede e um valor de saída é produzido.
- ② **Back-Propagation:** o valor produzido no passo de *Feed-Forward* é comparado com o valor esperado e são calculados os erros com relação a cada uma dos neurônios. Os valores de w são ajustados utilizando a derivada parcial da *função de custo* com relação ao peso w (qual é a “participação” de cada peso w no erro final da rede).

Inferência

Ao final do treinamento, apenas o processo de *Feed-Forward* é realizado na inferência em novas amostras.

Treinamento



Treinamento: Termos Recorrentes

Alguns dos termos normalmente utilizados no contexto de treinamento de Redes Neurais:

- **Batch Size**
- **Época**
- **Otimizador / Solver**
- **Custo / Loss**

Termos Recorrentes: *Batch Size*

Batch Size: quantidade de amostras processadas ao mesmo tempo pela rede durante uma iteração do *Feed-Forward*.

Exemplo

Um conjunto de dados que tem 20.000 amostras e será treinado com um *batch size* de 50 terá:

$$\frac{\text{Nº de Amostras}}{\text{batch size}} = \frac{20.000}{50} = 400 \text{ iterações/batches}$$

O ajuste dos pesos será feito ao final de cada iteração com base na média dos erros de cada uma das suas amostras do *batch*.

Quando todas as iterações do conjunto de dados forem realizadas uma **época** terá se passado.

Termos Recorrentes: Época

Época: uma iteração completa em todas os dados de treinamento (seja lá qual for o *batch size*).

Exemplo

Podemos pensar que o treinamento é como uma Série do Netflix em que:

- cada episódio é um *batch*;
- cada temporada é uma **época**.

Mas, infelizmente, as temporadas seriam todas iguais. Seria uma série meio chata...

Termos Recorrentes: Otimizador/*Solver*

Otimizador/*Solver*: método de atualização dos valores de w durante o *Back-Propagation*. Alguns dos mais comuns são:

- **Adam:** *Adaptive Momentum Optimizer*
- **SGD:** *Stochastic Gradient Descent*
- **RMSprop:** *Stochastic Gradient Descent*
- **Nadam:** ADAM + *Nesterov Momentum*

Termos Recorrentes: Custo/Loss

Função de Custo/Loss: função de custo com relação ao qual os parâmetros do modelo serão ajustados.

As funções variam dependendo do tipo de problema – Classificação ou Regressão

Regressão:

- Mean Squared Error
- Mean Absolute Error
- Huber Loss
- Mean Squared Logarithmic Error

Classificação:

- Entropia Cruzada
- Entropia Binária
- Focal Loss
- Label Smoothing

Implementação

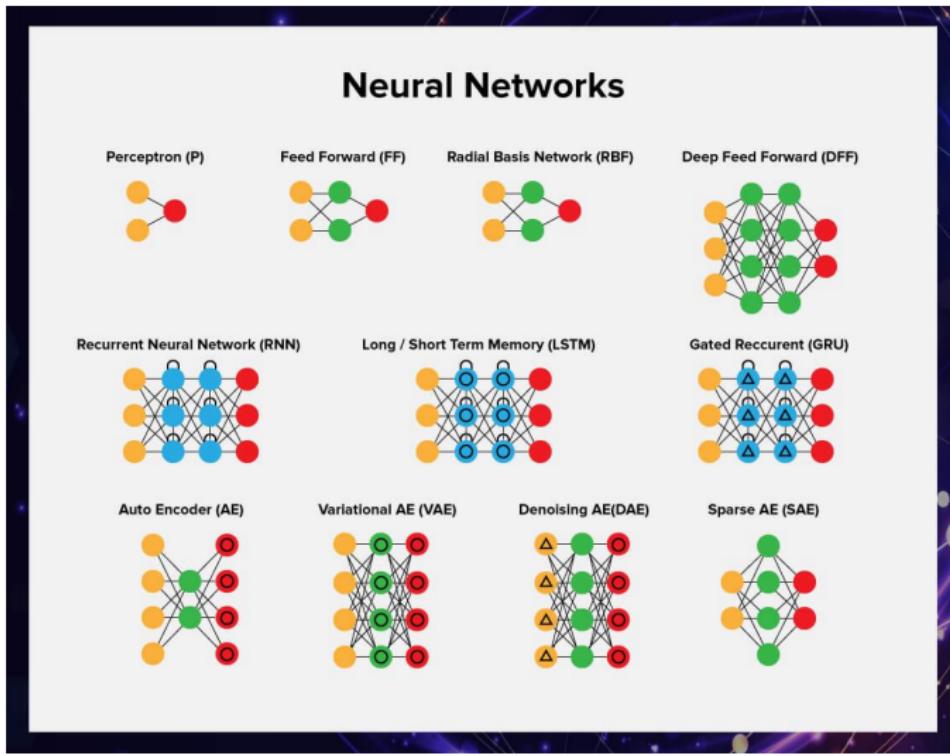
Não há necessidade de implementar *do zero* (e nem é aconselhável); são empregadas implementações prontas para uso disponibilizadas por diferentes bibliotecas:

- **Scikit-Learn**
- **Tensorflow/Keras**
- **PyTorch**
- **Flux**

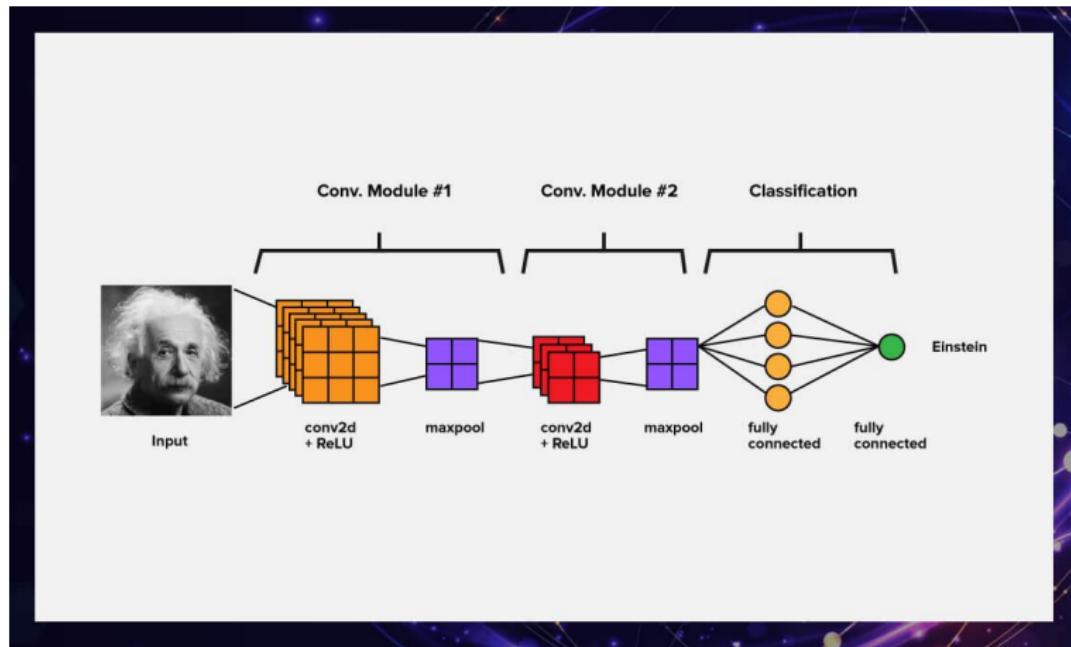
Vantagens:

- Auditabilidade (*Open-Source*)
- Credibilidade
- Reprodutibilidade
- Foco na aplicação do método (e não na sua implementação)
- Custo zero!

Tipos de Redes Neurais



Tipos de Redes Neurais



Links Úteis

- *Deep Learning and Neural Networks Guide.*
- *Activation Functions Explained.*
- *Tensorflow Playground.*
- *A Comprehensive Guide to Types of Neural Networks.*