∇ data

Redes Neurais

Enzo Tonon Morente
in \enzotm

Presença

- Linktree: Presente na bio do nosso instagram
- Presença ficará disponível até 1 hora antes da próxima aula
- É necessário 70% de presença para obter o certificado

Presença

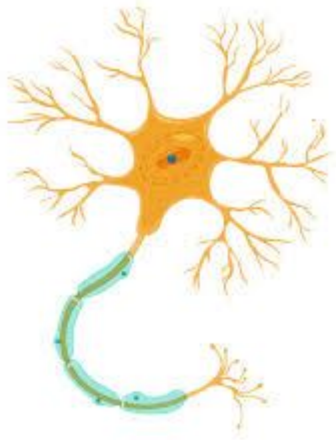
Introdução

Inspiração Biológica

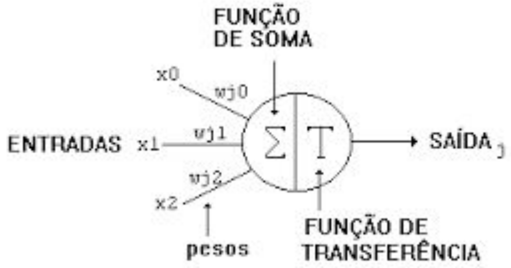
- Capacidade adaptativa do aprendizado humano
- Interação entre neurônios

Inspiração Biológica

- Apenas inspiração, não reflete diretamente os processos que ocorrem no cérebro

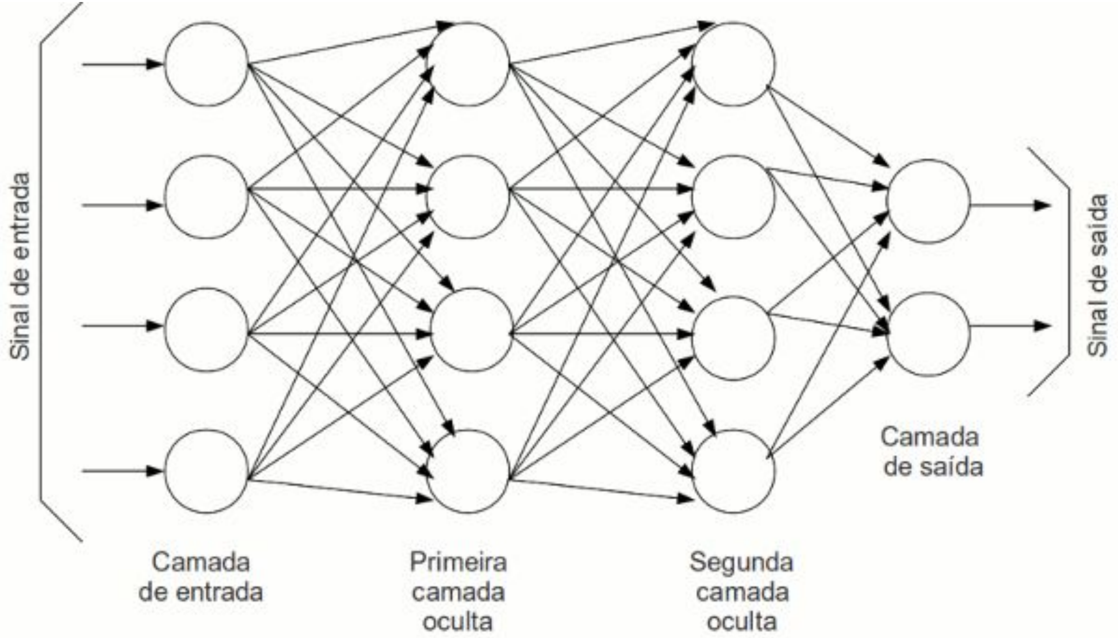


≠



Tipos

- Multi-Layer Perceptron (MLP)

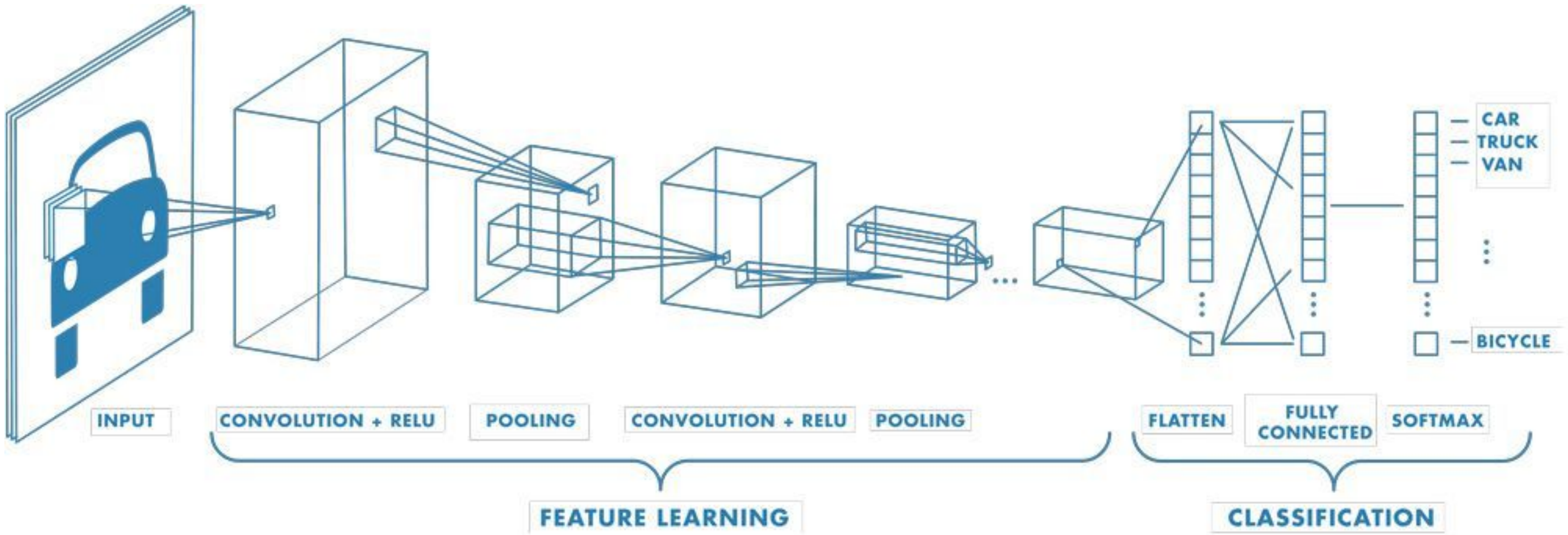


Tipos

- Multi-Layer Perceptron (MLP)
  - Versátil
  - Simples
  - Pode ser utilizado em conjunto com várias arquiteturas

Tipos

- Convolutional Neural Networks (CNN)

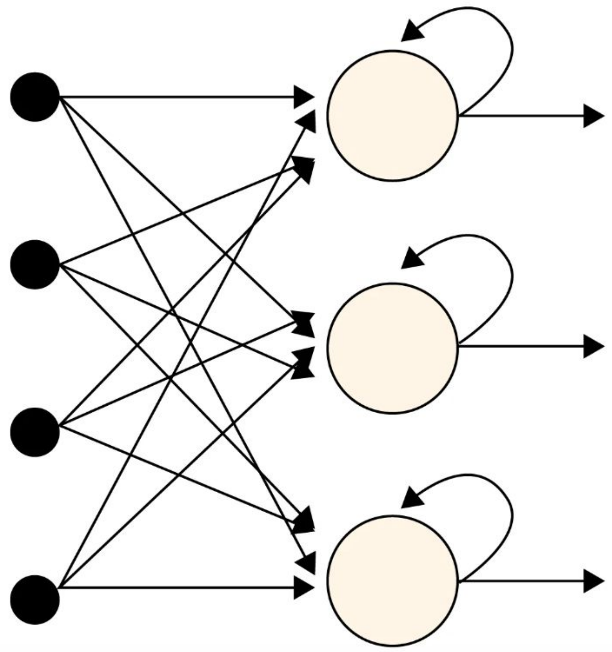


Tipos

- Convolutional Neural Networks (CNN)
    - Utilizada principalmente em imagens
    - Pode ser usada para reconhecimento de rostos, dígitos e várias outras tarefas

Tipos

- Recurrent Neural Networks (RNN)



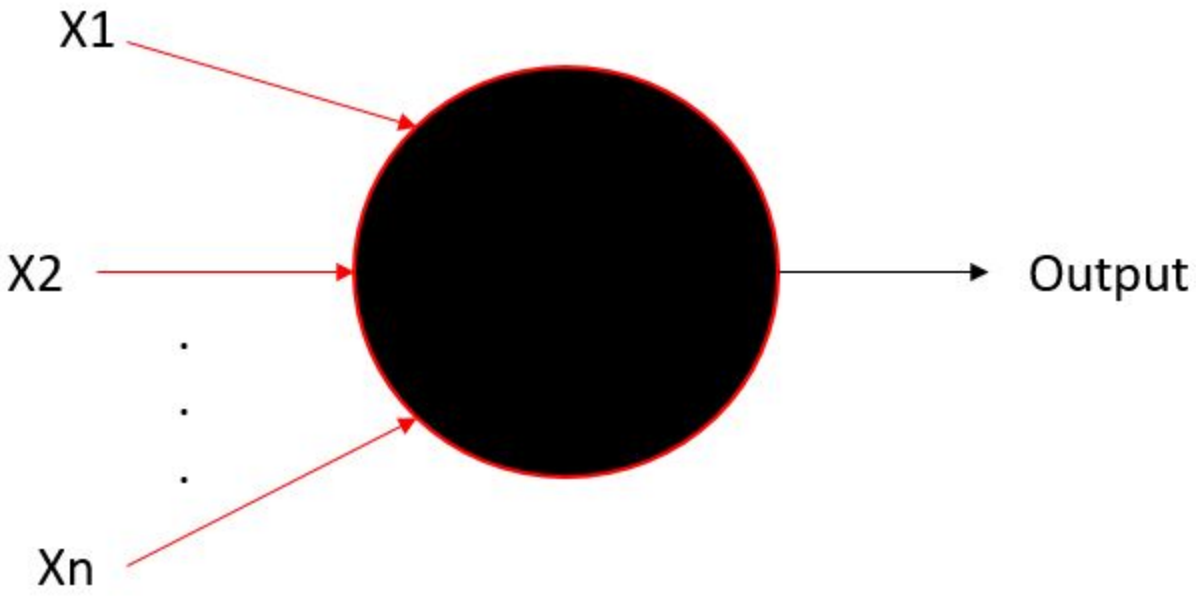
Tipos

- Recurrent Neural Networks (RNN)
  - Utilizada principalmente dados sequenciais
  - Pode ser usada para processamento de texto, valores de ações e outras tarefas

Neurônio Simples

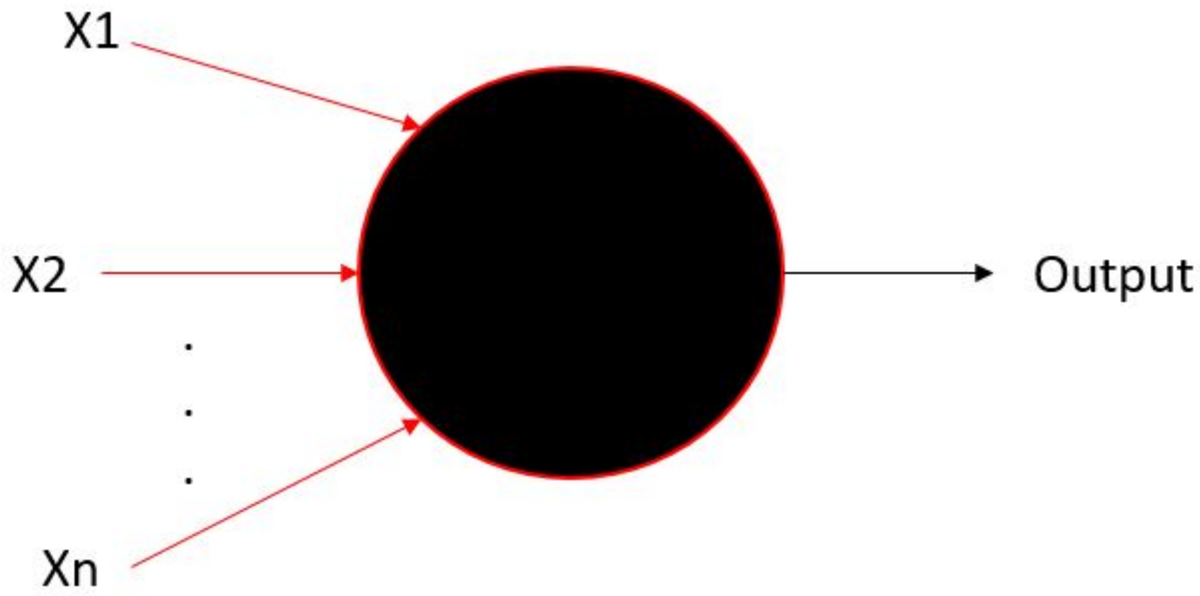
- O neurônio clássico das redes neurais possui as seguintes características:
  - Pode receber vários dados (inputs)
  - Processa esses dados internamente
  - Tem apenas um resultado (output)

Neurônio Simples



Perceptron

$$z = W \cdot x + b$$



Perceptron

- Cada neurônio é uma função linear (perceptron)
- Uma rede neural é um conjunto de neurônios (sempre linear)

$$\text{Camada 1:} \quad z_1 = W_1 x + b_1$$

$$\text{Camada 2:} \quad z_2 = W_2 z_1 + b_2$$

$$z = W_n(...W_2(W_1 x + b_1) + b_2...) + b_n = W'x + b'$$

E a linearidade?

- Resolvemos anteriormente com funções kernels
- Iremos agora introduzir o conceito de funções de ativação
    - Introdução à não-linearidade

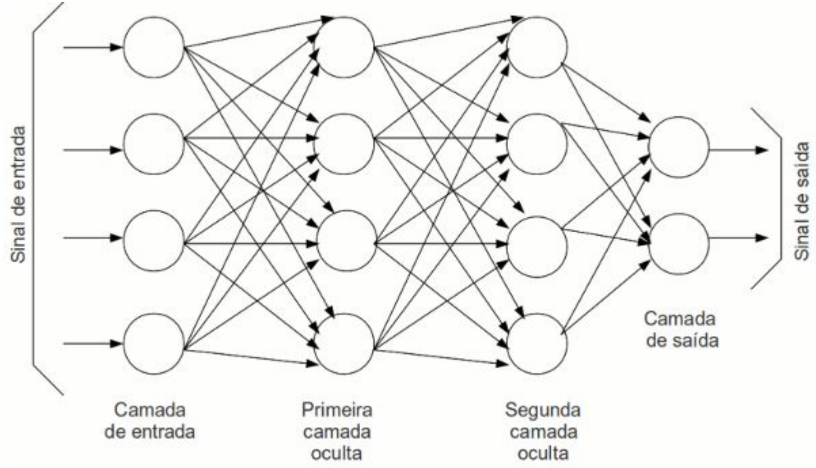
$$z = W \cdot x + b \qquad a = \sigma(z)$$

- O valor de alpha serve como entrada para a próxima camada

E a linearidade?

- O valor de alpha serve como entrada para a próxima camada

$$z = W \cdot x + b \qquad\qquad a = \sigma(z)$$



Funções de Ativação

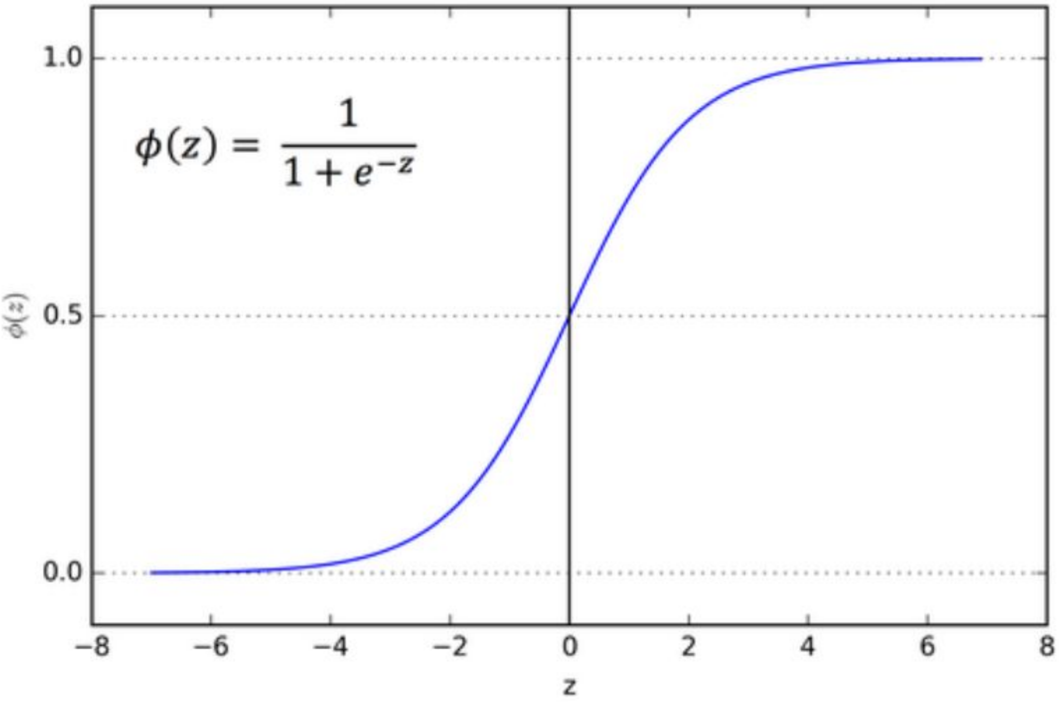
ReLU (Rectified Linear Unit): $\sigma(z) = \max(0, z)$

Sigmoid: $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

tanh: $\sigma(z) = \tanh(z)$

∇

Funções de Ativação



Teorema da Aproximação Universal

"Uma rede neural com pelo menos **uma camada escondida** e **ativação não-linear** (como ReLU, sigmoid, etc.) pode aproximar **qualquer função contínua** com precisão arbitrária, desde que tenha **neurônios suficientes**."

∇

Treinamento

Backpropagation

- Algoritmo para treinamento de redes neurais
    - calcula os gradientes da função de perda com relação a todos os parâmetros da rede (pesos e biases), aplicando a regra da cadeia (chain rule) do cálculo diferencial.
- O objetivo do backpropagation é ajustar os parâmetros da rede neural (os pesos e biases de cada camada) para minimizar a função de perda.
    - Para isso é necessário saber como cada peso e bias influência na perda final

Gradiente Descendente

- Atualização

$$w_{11} := w_{11} - \eta \cdot \frac{\partial L}{\partial w_{11}}$$

Backpropagation

- Para isso é necessário saber como cada peso e bias influência na perda final

$$\frac{\partial L}{\partial w_{ij}^{(l)}}, \quad \frac{\partial L}{\partial b_j^{(l)}}$$

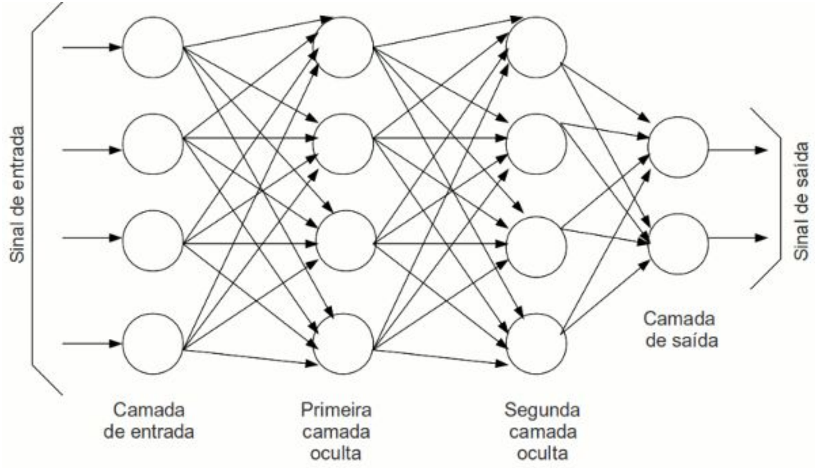
L: função de perda

$w_{ij}^{(l)}$: peso da camada l, que conecta o neurônio i da camada anterior ao neurônio j da camada atual

$b_j^{(l)}$: bias do neurônio j da camada l

∇

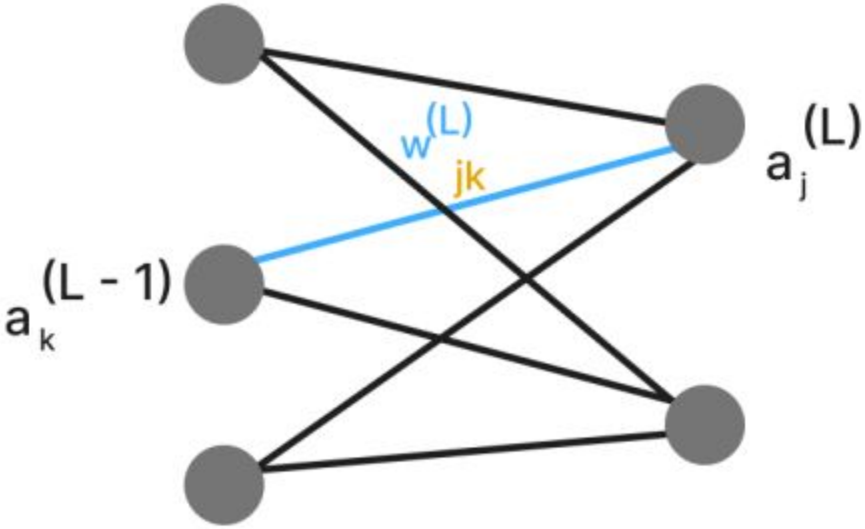
Backpropagation



$$\frac{\partial L}{\partial w_{ij}^{(l)}}, \quad \frac{\partial L}{\partial b_j^{(l)}}$$

Lousa

Backpropagation



Backpropagation

$$L = \sum_{j=0}^{n} (a_j^{(L)} - y_j)^2$$

∇

Backpropagation

$$\frac{\partial L}{\partial W_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial W_{jk}^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial L}{\partial a_j^{(L)}}$$

∇

Backpropagation

$$\frac{\partial L}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial L}{\partial a_j^{(L)}}$$

∇

Backpropagation

$$\frac{\partial \mathcal{C}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \sigma' \left(z_j^{(l)} \right) \frac{\partial L}{\partial a_j^{(l)}}$$

$$\frac{\partial \mathcal{C}}{\partial a_j^{(l)}} \begin{cases} \sum_{j=0}^{n} w_{jk}^{(l+1)} \sigma' \left(z_j^{(l+1)} \right) \frac{\partial L}{\partial a_j^{(l+1)}} \\ \textbf{ou} \quad 2 \left(a_j^{(L)} - y_j \right) \end{cases}$$

Prática

data@icmc.usp.br

@data.icmc

/c/DataICMC

/icmc-data

data.icmc.usp.br

obrigado!