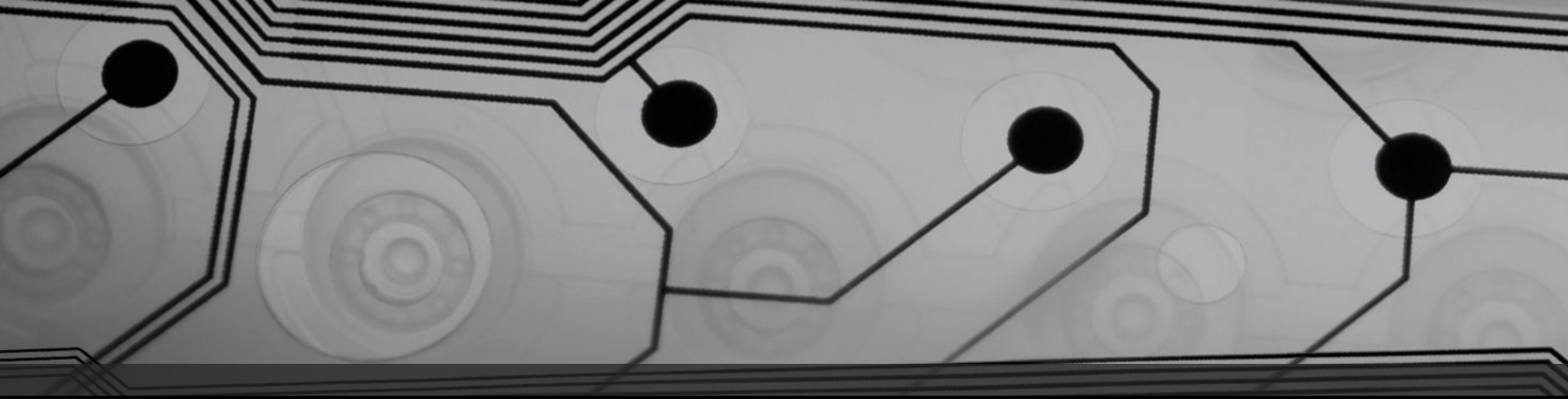


Introdução a Redes Neurais

Imergindo em uma popular área da Inteligência Artificial

Apresentações

Antes de tudo, é um prazer conhecê-los :)

A cara de sono é característica...

Olá!
Sou o **Minho**, estudante de Engenharia de Computação pela Universidade Federal do Ceará (UFC).

Na área de Inteligência Artificial, sou pesquisador principalmente em *Statistical Machine Learning*.

Contatos:

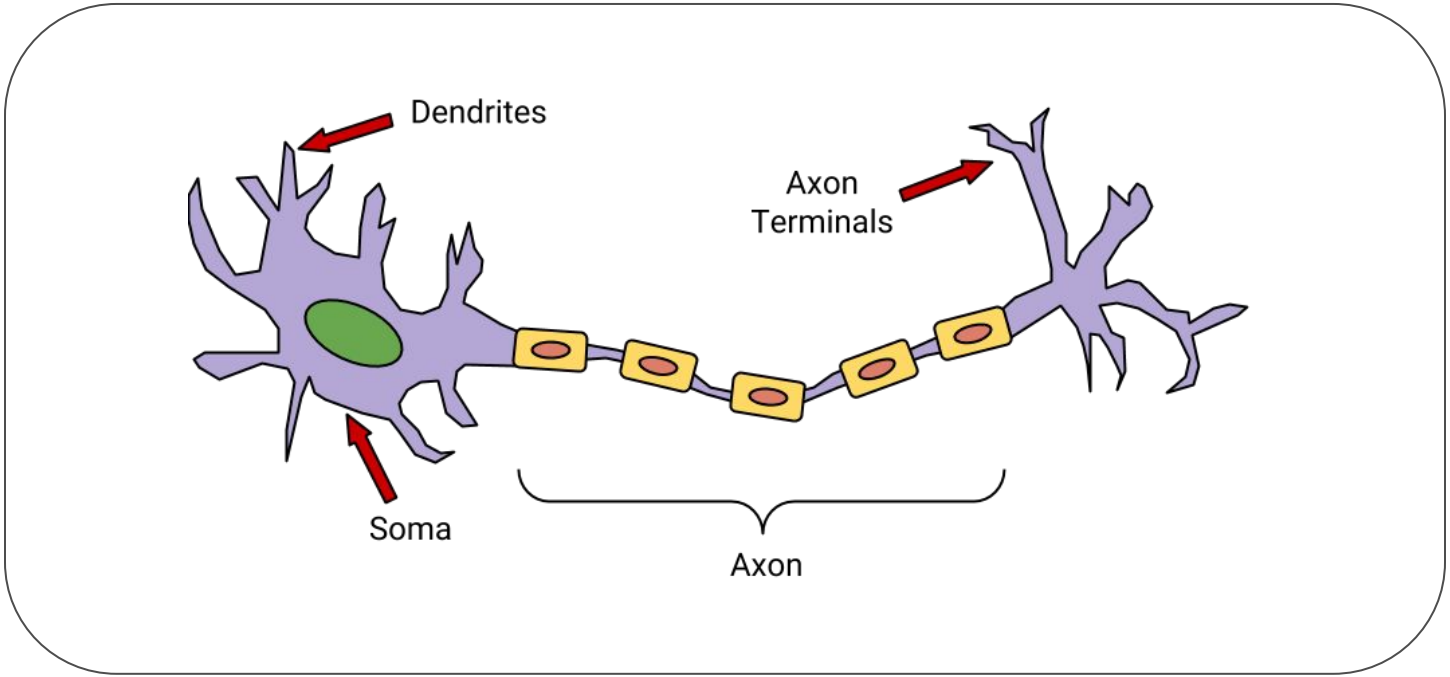
- **Email:** minhotmog@gmail.com
- **Github:** github.com/TioMinho
- **Telegram:** @katchau
- **Twitter:** @minhomenezes

corgis >>> other dogs >>>>> cats

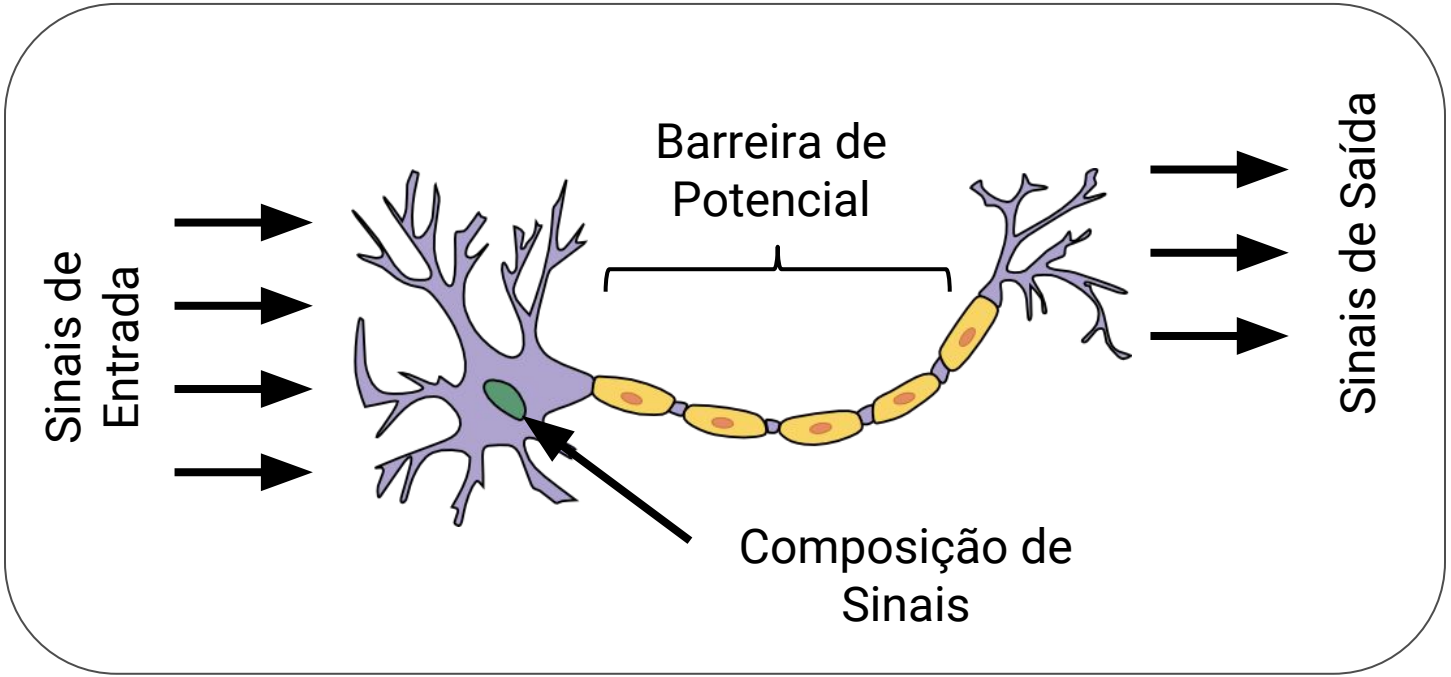
Redes Neurais: **Representação**

O que é uma Rede Neural? Como representá-la? O que ela processa?

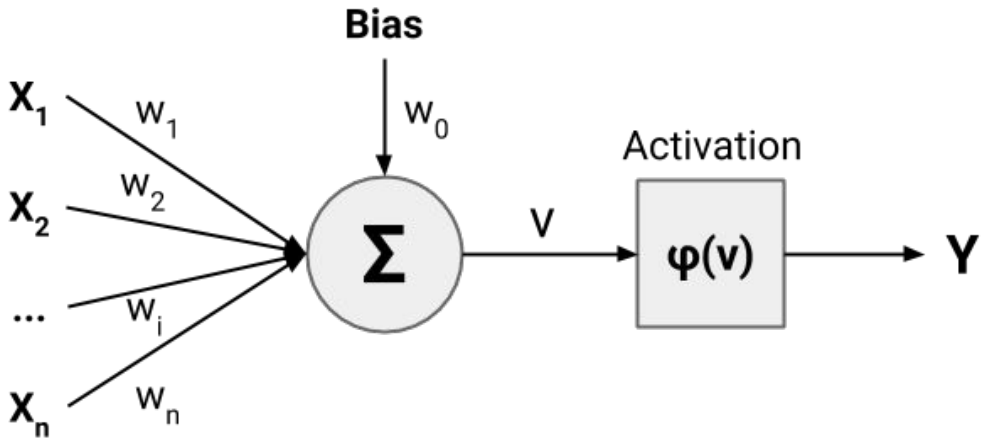
Modelo do Neurônio Biológico



Modelo do Neurônio Biológico



Neurônio McCulloch-Pitts



Neurônio - Computação

- A computação, ou processamento, que um Neurônio faz em uma entrada para avançar um sinal ao próximo estágio é dividida entre duas partes:
  - **Potencial de Rede:** os sinais de entrada são ponderados por cada peso e então somados, juntamente com o peso associado ao bias.

$$S_{\text{net}} = W_0 + \sum_i W_i X_i$$

  - **Ativação:** o potencial de rede é aplicado sobre uma função de ativação que determinará se o *potencial de rede* é suficiente para avançar o sinal.

$$\hat{y} = \varphi(S_{\text{net}})$$

Neurônio - Forma Matricial

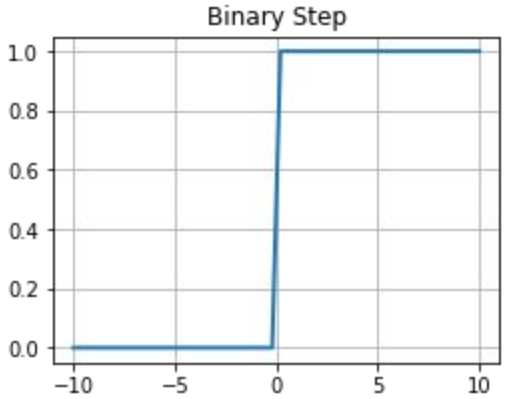
- Podemos “forçar” um atributo unitário (igual a 1) na Entrada para obter uma representação matricial da Rede Neural:

$$\begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} ; \quad \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ \vdots \\ X_n \end{bmatrix} ; \quad \left[Y_{\text{out}}\right]$$

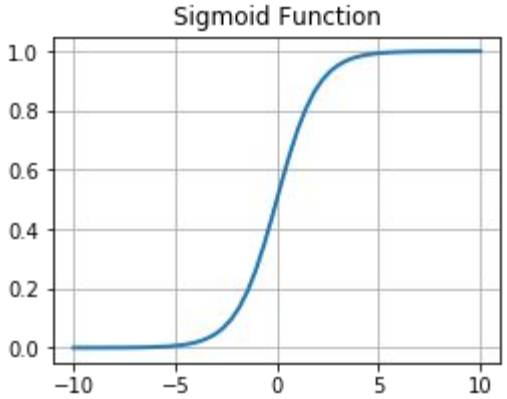
- As computações, então, se tornam simples operações matriciais:

$$\begin{cases} S_{\text{net}} & = W^T X \\ \left[Y_{\text{out}}\right] & = \varphi(S_{\text{net}}) = \varphi(W^T X) \end{cases}$$

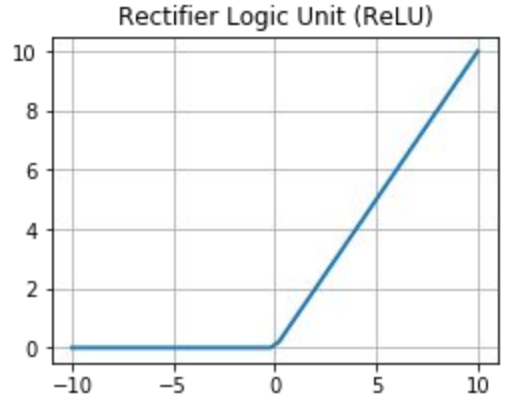
Neurônio - Funções de Ativação



$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

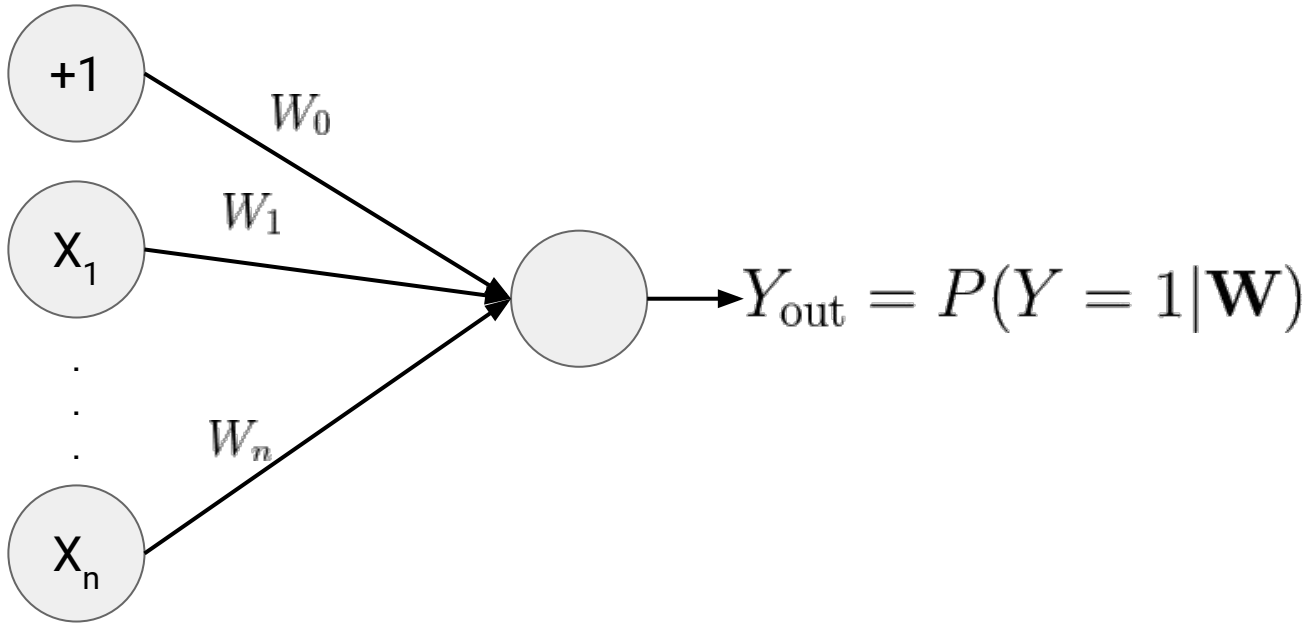


$$f(x) = \frac{1}{1 + e^{-x}}$$

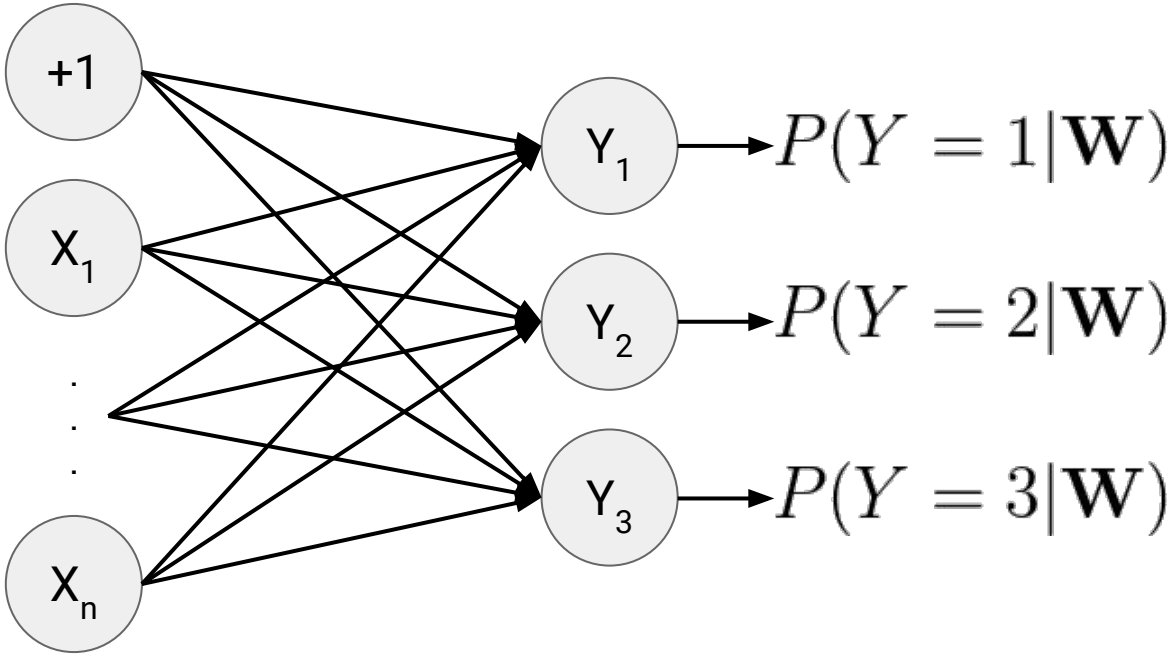


$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

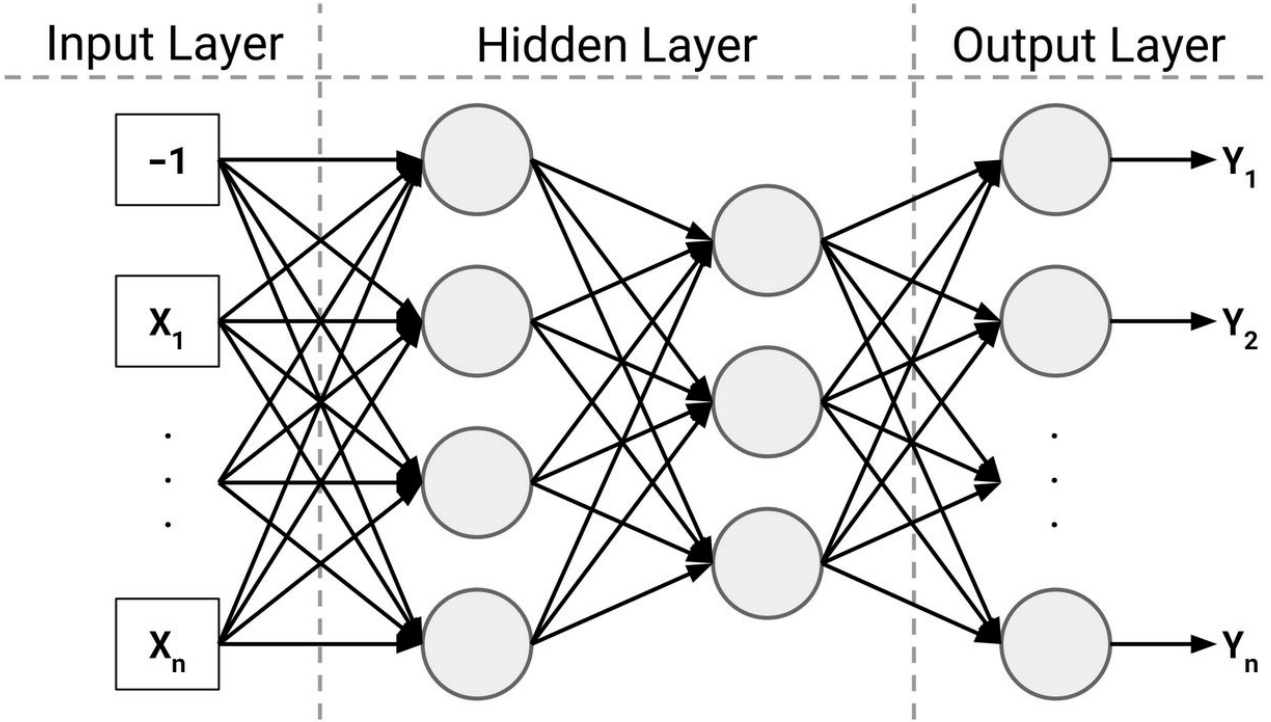
Rede Neural Camada Única Densa (Singlelayer Perceptron Binário)



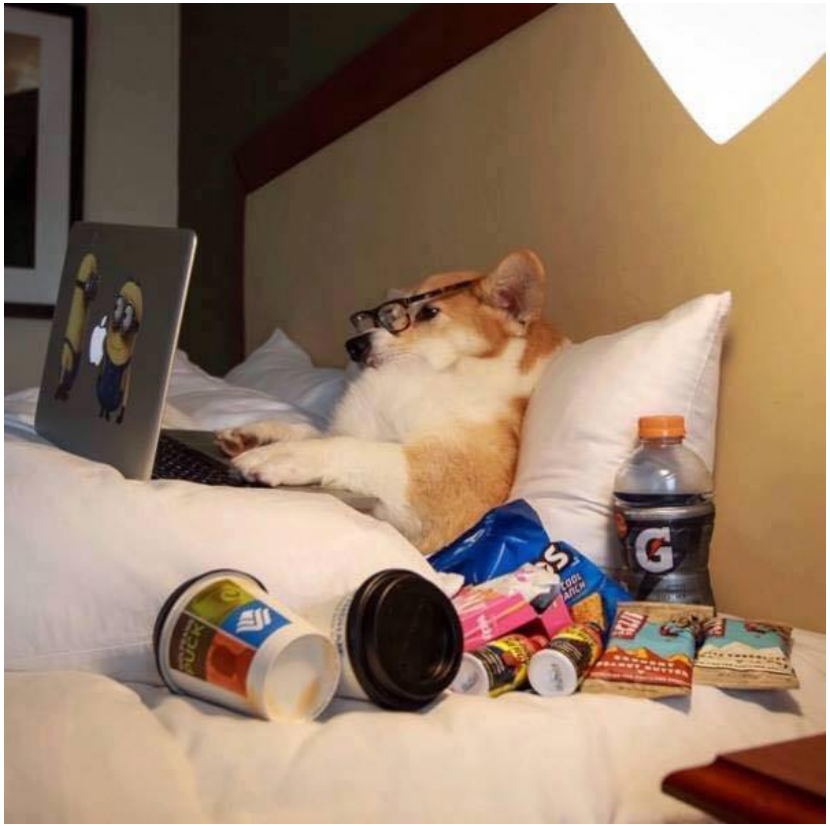
Rede Neural Camada Única Densa (Singlelayer Perceptron Multiclasse)



Rede Neural Multicamada Densa (Multilayer Perceptron)



!! Hands-On !!



Redes Neurais: **Aprendizagem**

Como avaliar a qualidade da rede? Como ajustar os pesos para uma condição ótima?

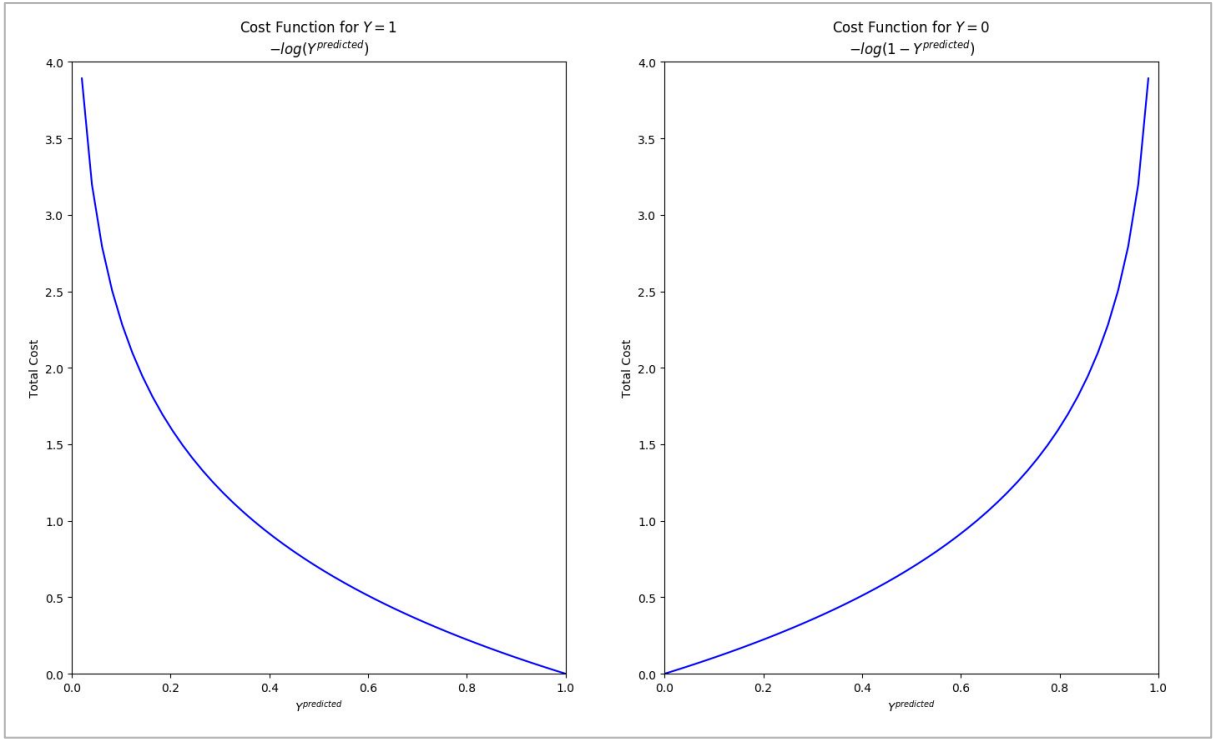
Treinando uma SLP

- Uma Rede Neural é descrita, praticamente, pelo valor dos pesos de suas conexões!

- Para ajustar a Rede Neural a alguma aplicação, é necessário realizar algum algoritmo de Aprendizagem Supervisionada, que observe dados já classificados e compare com a interpretação da Rede Neural.

- Podemos definir uma função de custo para problemas de Classificação e a otimização conhecida como **Gradiente Descendente!**

Função de Custo de Classificações



Função de Custo de Classificações

- Podemos fundir as duas funções anteriores em uma só:

$$\mathcal{L}(W) = -\sum y \; log(h(W)) + (1-y) \; (1 - log(h(W)))$$

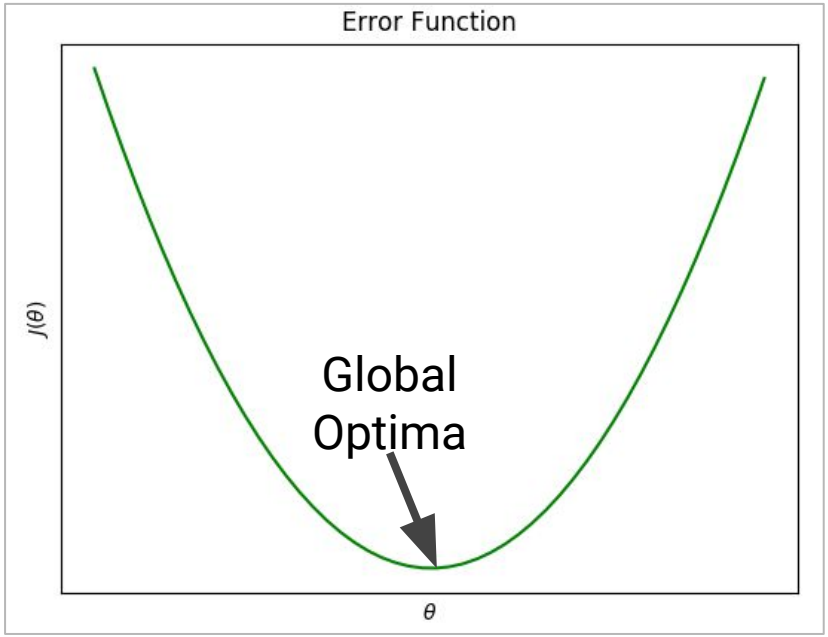
- Onde:

$$h(W) = \frac{1}{1 + e^{-W^T X}}$$

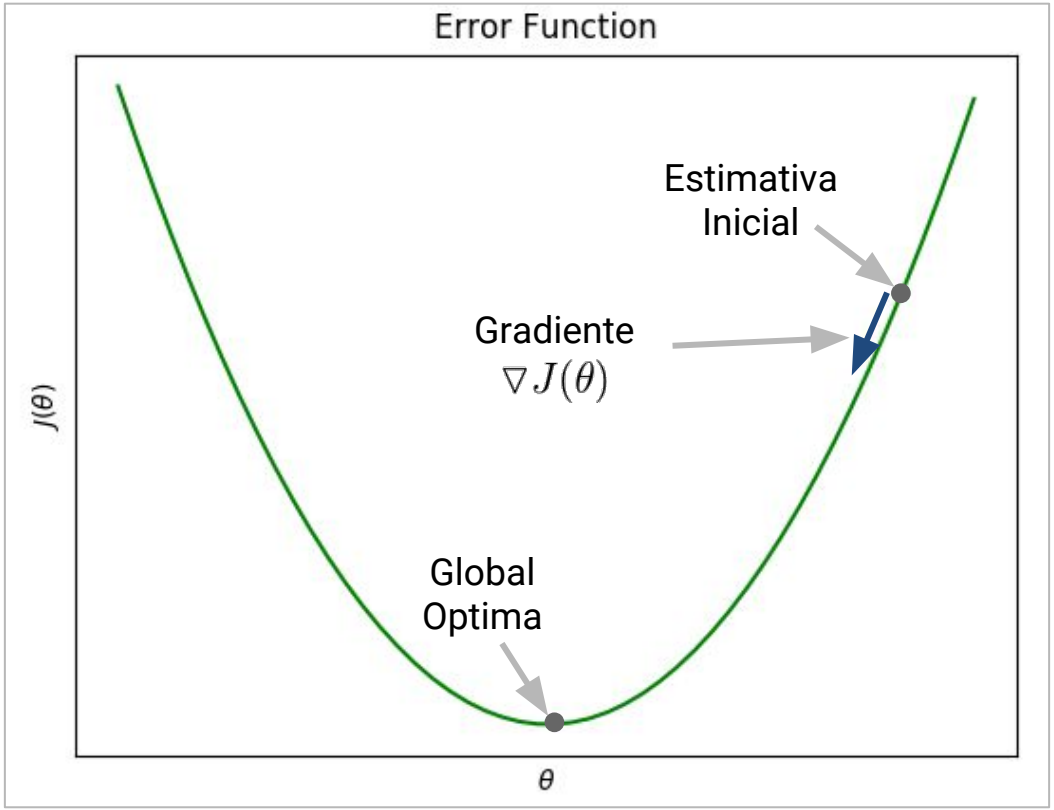
Função de Custo de Classificações

A Função de Erro é uma função **convexa**!

*tridimensional



Gradiente Descendente

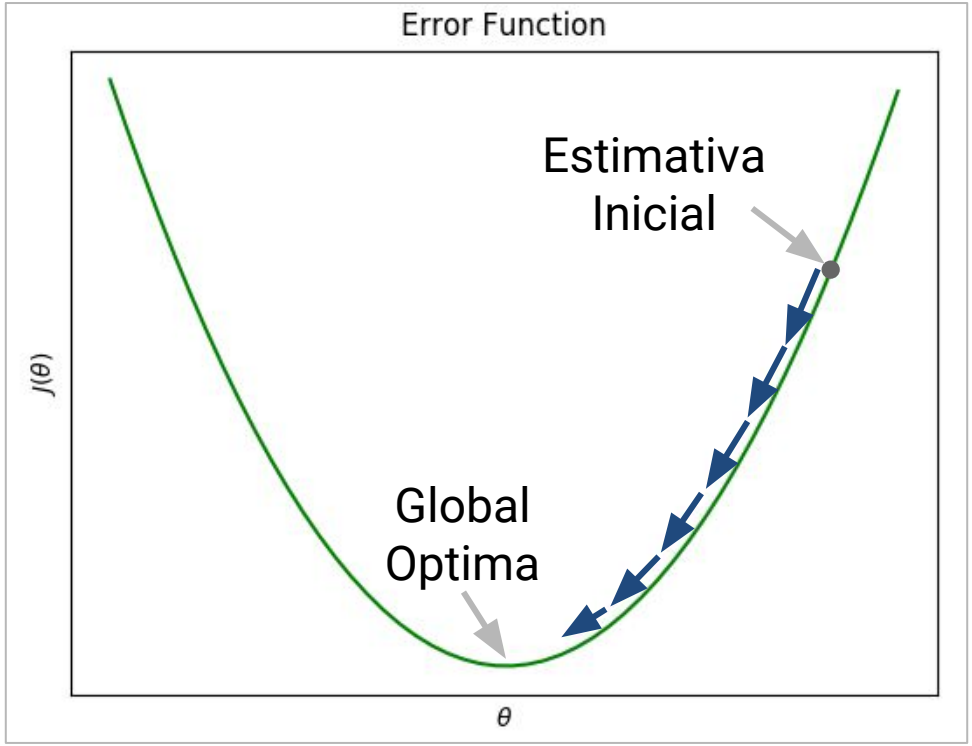


Gradiente Descendente

- Quando utilizamos a função de ativação **Sigmoide**, as derivadas da função de custo mostrada anteriormente tem um formato canônico muito simples:

$$\frac{\partial}{\partial W}(\mathcal{L}(\mathbf{W})) = \frac{1}{m}\sum_{i} E_i X_i$$

Gradiente Descendente



Objetivo de Otimização

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

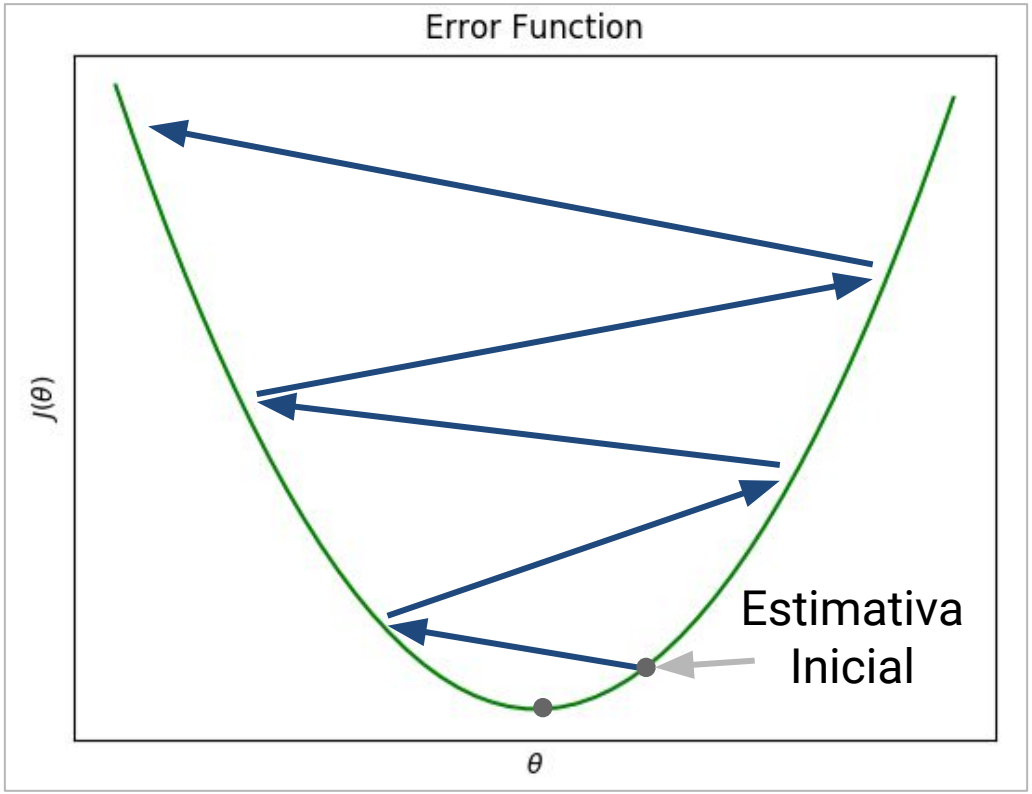
Algoritmo:

enquanto nao converge:

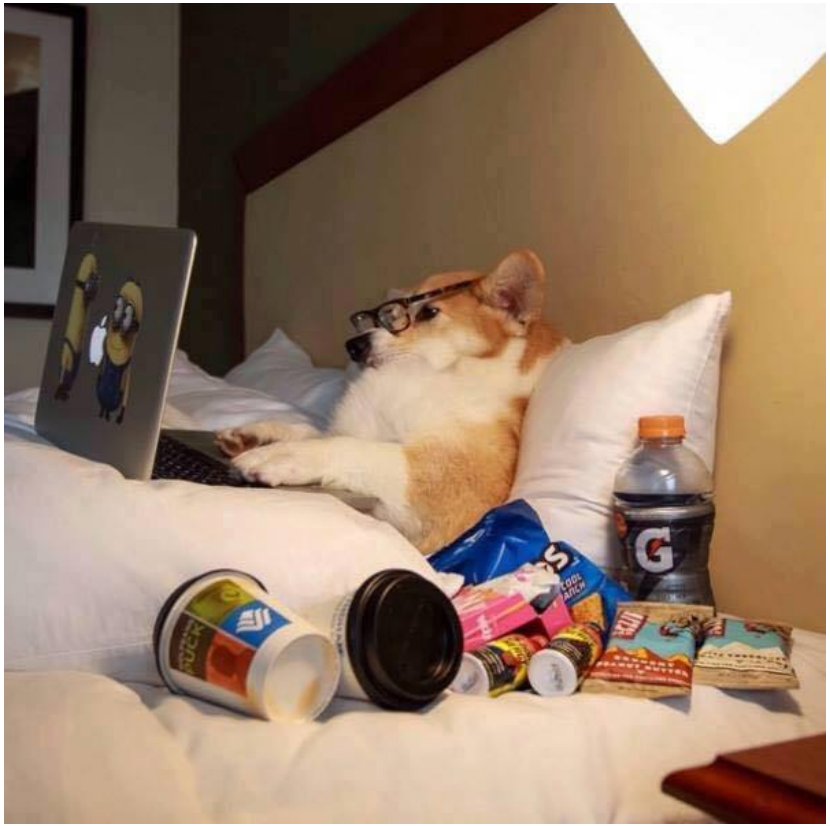
$$\theta_0 := \theta_0 - \frac{\alpha}{m}\sum(h(\theta) - y)$$

$$\theta_i := \theta_i - \frac{\alpha}{m}\sum(h(\theta) - y)x_i$$

Gradiente Descendente



!! Hands-On !!



Por hoje é só, pessoal!
Dúvidas?