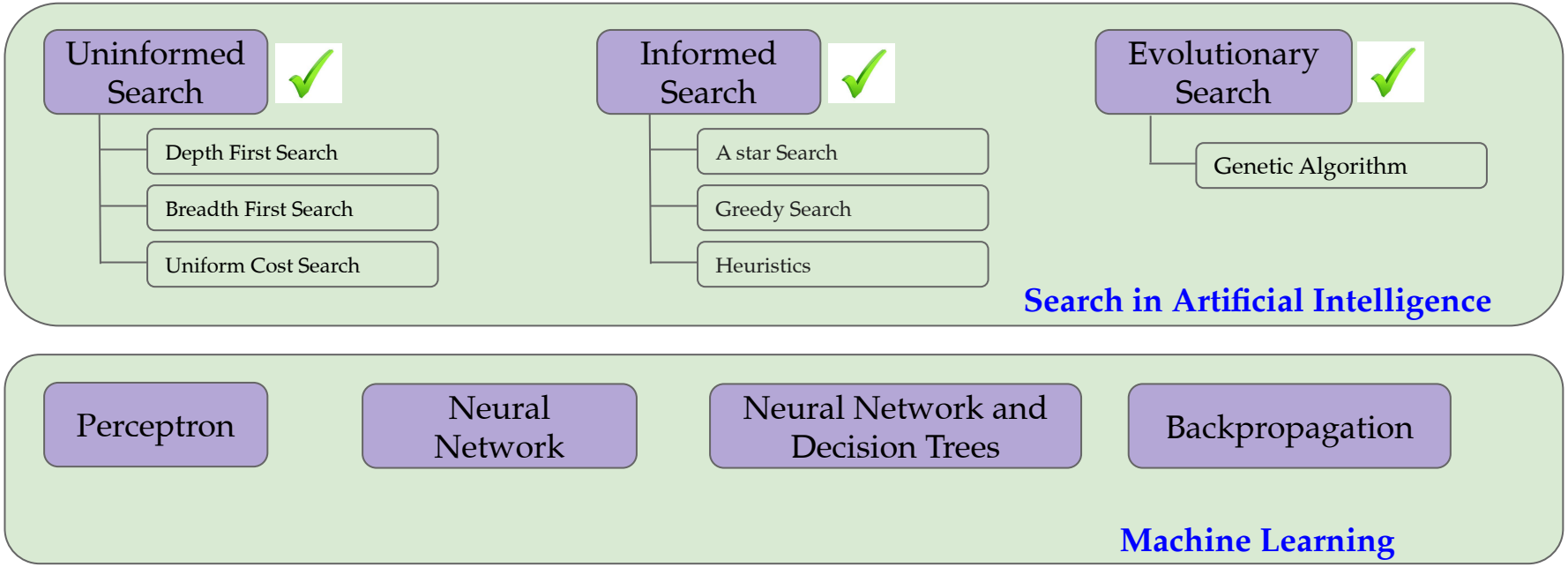
Perceptron

Prof. Dr. Eduardo Noronha
Inteligência Artificial Aplicada
Instituto Federal de Goiás (IFG)

Topics Overview

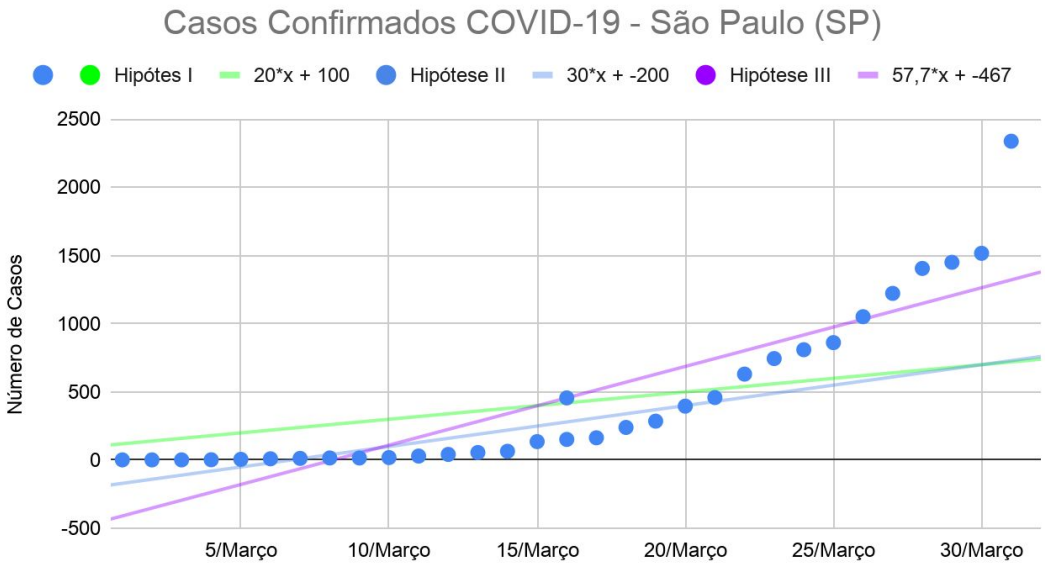


Agenda

- Review of Linear Regression Models
  - Weight
  - Linear Regression
  - Loss
  - Weight Space
  - Gradient Descent
  - Learning Rate
  - Weight Update (Perceptron Learning Rule)
  - Batch Gradient Descent
  - Stochastic Gradient Descent
  - Training Curve

Escolhendo as Melhores Hipóteses

(Least Squares Method)



Hipótese	w1	w0	Erro
I	20	100	**201.410**
II	30	-200	**178.876**
III	57,72	-467	**86.264**

Otimizando as Variáveis de Decisão

(Least Squares Method)

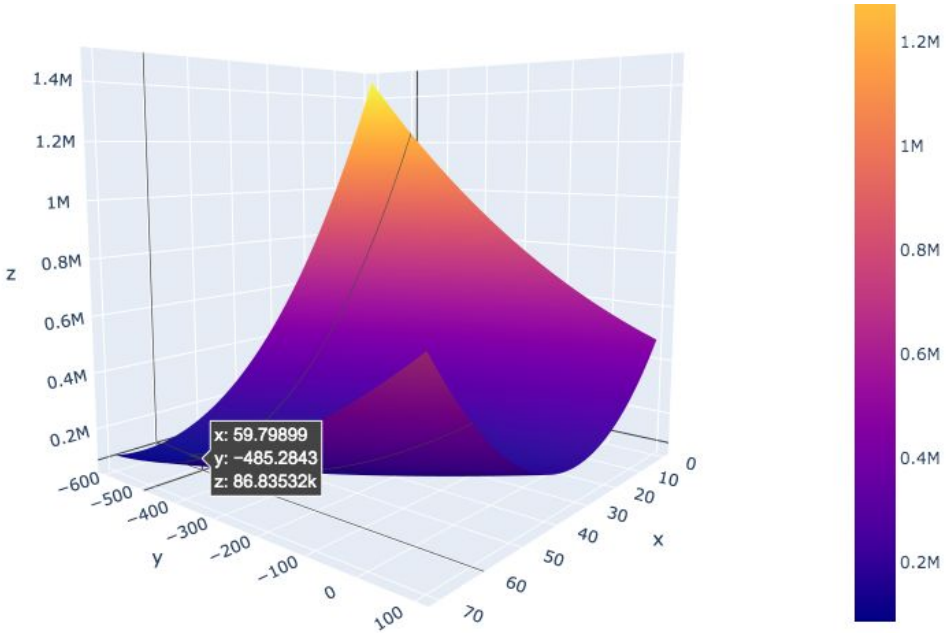
$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - h_i)^2$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w1 * x_i + w0))^2$$

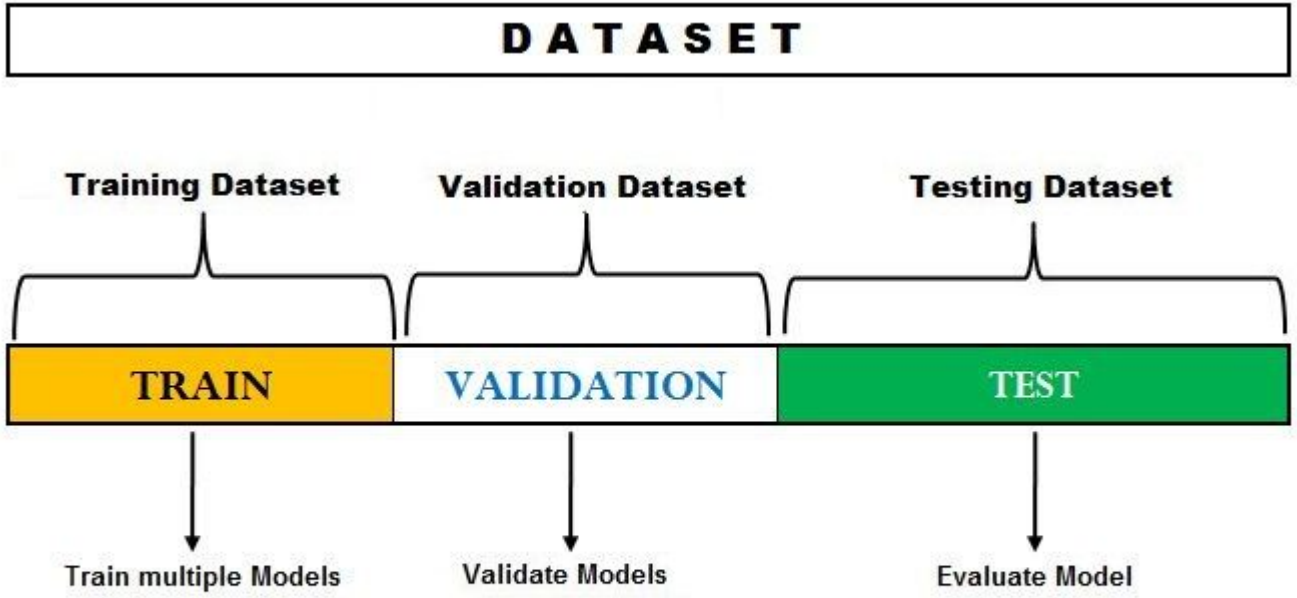
Para minimizar o Erro:

$$w1 = \frac{\partial E}{\partial w1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = 0$$

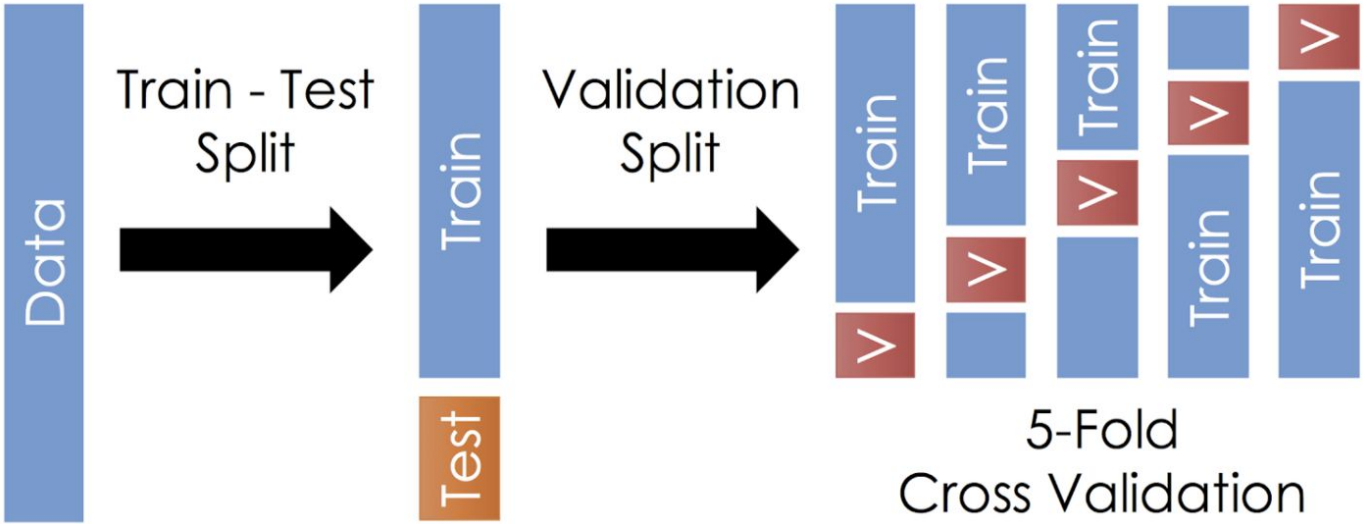
$$w_0 = \frac{\partial E}{\partial w0} = \bar{y} - w_1 * \bar{x} = 0$$



Validação da Hipótese

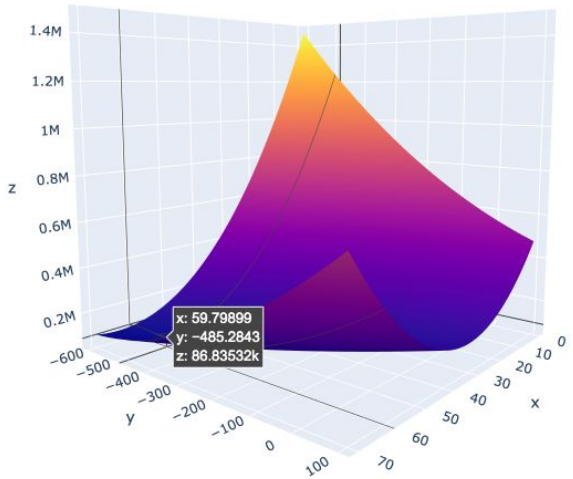


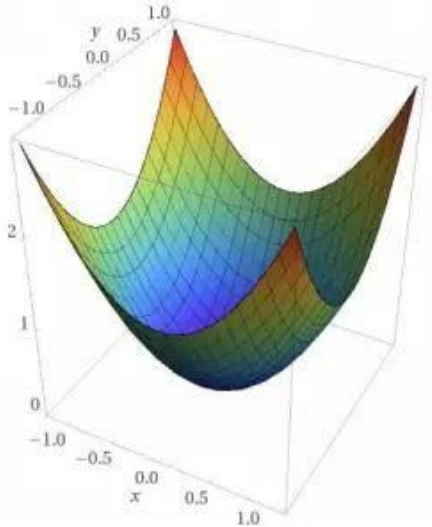
Holdout e K-Fold Cross Validation

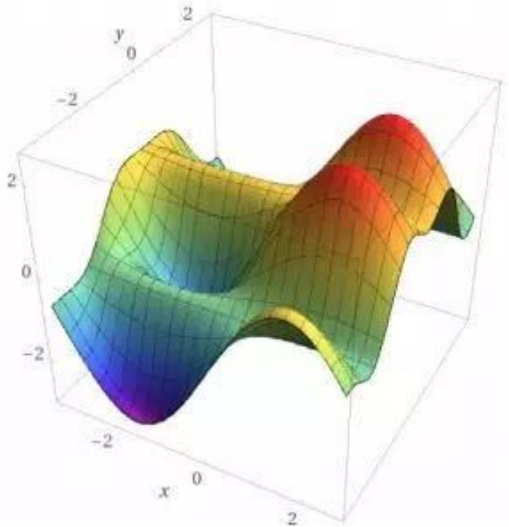


Fonte: https://medium.com

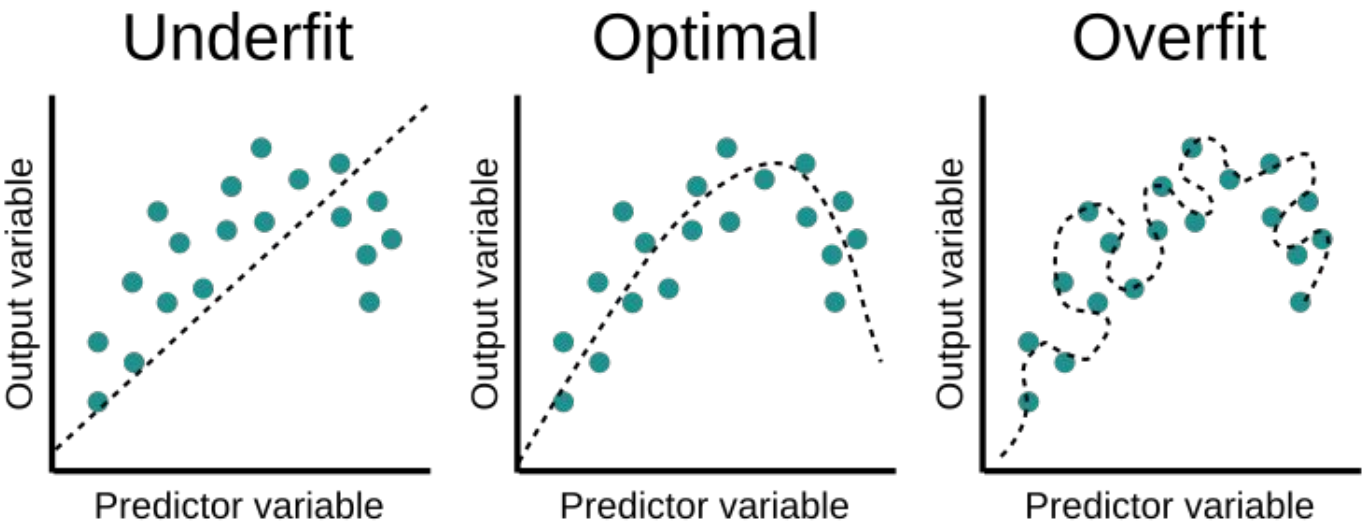
Weight space





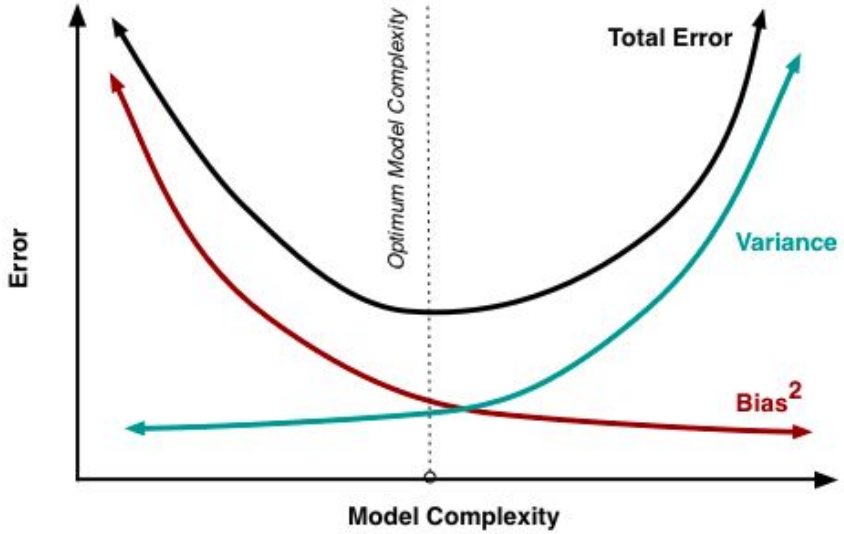


Overfitting e Underfitting



Fonte: https://www.educative.io

Overfitting e Underfitting



Fonte: https://dev.to

O que vimos de inteligência artificial até agora?

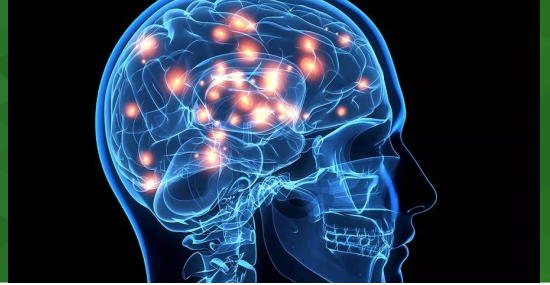
Redes Neurais Artificiais

- Baseada no cérebro humano
- Conexionismo
- Sistemas de Processamento Paralelo e Distribuído
  - Nodos (Sistemas de processamento)
  - Computam funções (geralmente não lineares)
- Não algorítmico
- Dispostas em uma ou mais camadas
- Interligadas por grande número de conexões

Redes Neurais Artificiais

- Fase de Aprendizagem
  - Capacidade de aprendizagem por meio de exemplos
  - Exemplos são repassados e representação é automaticamente gerada
  - Mapeadores universais de funções multivariáveis
  - Capacidade de auto organização
  - Processamento temporal

Motivação para as RNA's

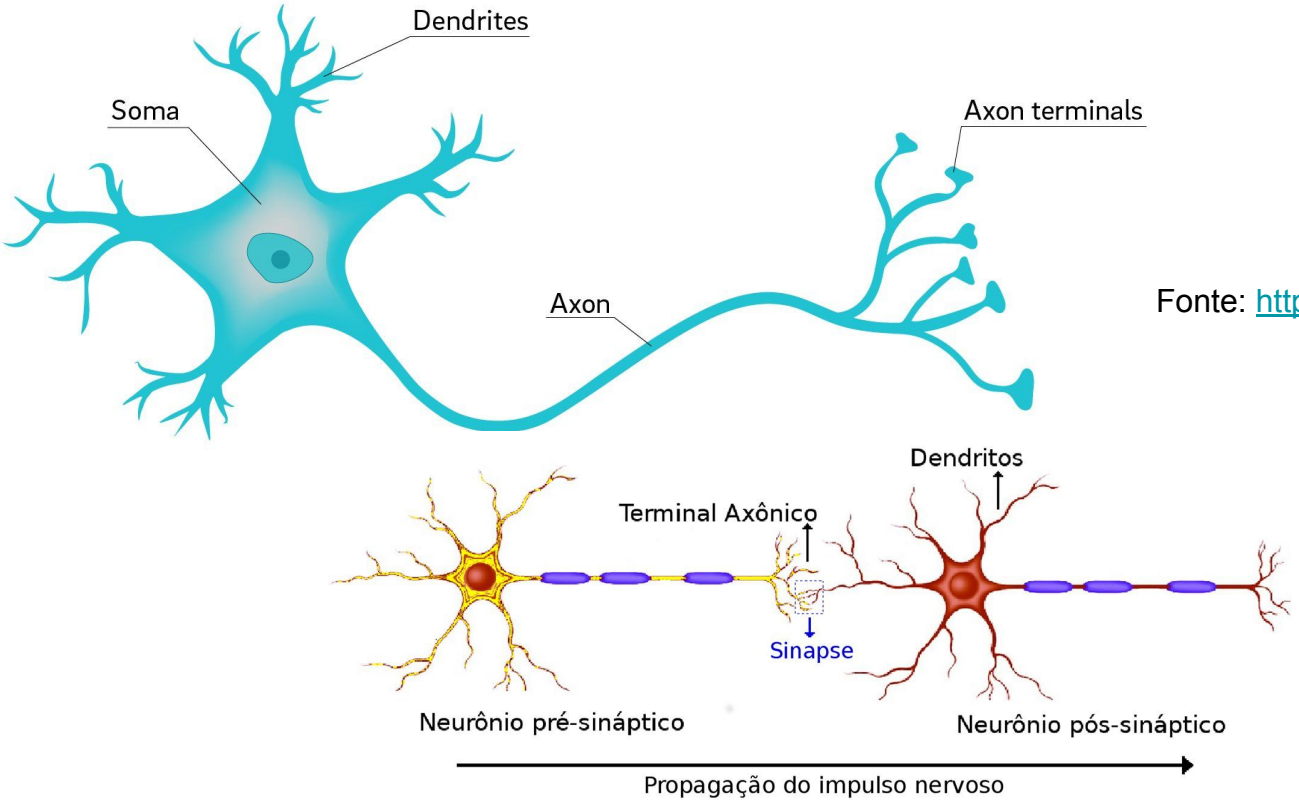


- Cérebro humano: 10^{11} nodos
- Milhares de conexões contínuas e paralelas entre eles
- Cérebro: emoção, pensamento, cognição (percepção, a atenção, associação, memória, raciocínio, juízo, imaginação, pensamento e linguagem)

Neurônios Biológicos

- Dendritos: Recebem as informações, ou impulsos nervosos, oriundas de outros nodos, e conduzi-las até o corpo celular.
- Corpo da Célula: Processar e produzir novas informações/impulsos
- Axônio: Transmitir as informações/impulsos a outro neurônio.
- Sinapse: Ponto de encontro entre a terminação axônica de um neurônio e o dendrito de outro.

Neurônio



Fonte: https://medicalxpress.com/

INSTITUTO FEDERAL
Goiás

Comunicação no Cérebro

Sinais Químicos: Por meio das Sinapses

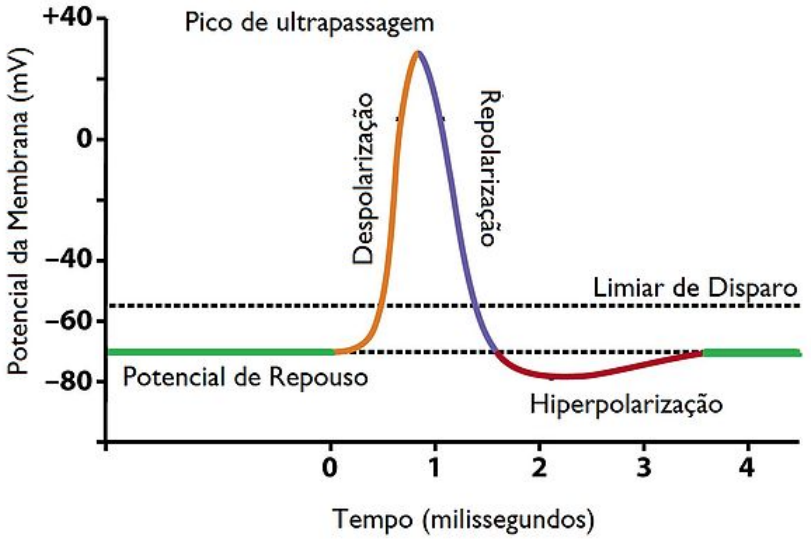
Sinais Elétricos: Dentro do neurônio

O corpo combina os sinais recebidos e, se o valor resultante for acima do limiar de excitação do neurônio, um impulso elétrico é produzido e propagado através do axônio para os nodos seguintes.

Existe uma diferença de potencial (em volts) entre o interior (concentração de potássio) e o exterior (concentração de sódio) do neurônio. Para que a sinapse ocorra é necessário haver um aumento do potencial elétrico de -70mv (potencial de repouso) para -50mv (potencial de ação).

Comunicação no Cérebro

- Uma sinapse poderá ser inibitória ou excitatória. Isso vai depender do acúmulo das entradas inibitórias e excitatórias, medido pelo corpo em um intervalo de tempo.
- Depois de gerar um impulso, o neurônio entre em um período de refração (período em que não poderá ser novamente estimulado)




SINAPSE
QUÍMICA
EDU

1900
TriodeVaccumTube
1920
First ever robot
1938
Digital Logic
1948
Cybernetics

1956
AI term born

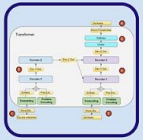
"A paper called the **Dartmouth Proposal** is presented in which, for the first time, the AI definition is used."

1980
WABOT-2
1982
Hopfield ANN
1986
Parallel Processing
1989
Convolutional NN

1997
Deep Blue

"IBM's Deep Blue defeats chess world champion Garry Kasparov."

2003-2005
DARPA Challenges
2006
Deep Belief ANN
2008
IBM Blue Brain
2010
Apple Siri



2017
Transformers

"
Attention
Is
All
You
Need
"

2017
WaveNet
2019
GPT-1
2020
BERT LM
2020
GPT-3

2022
ChatGPT

"OpenAI ChatGPT became the fastest growing consumer software application in history just after 2 months!"

2025
deepseek

The chinesse **deepseek** overpasses OpenAI in 2 weeks!

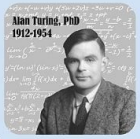
1st Winter

2nd Winter

1900 - 1950	1950 - 1973	1980 - 1987	1993 - 2000	2000 - 2012	2012 - 2017	2017 - 2020	2020 - 2025	2025 - ...

1950
Turing Test

"A computer would deserve to be called intelligent if it could deceive a human into belivign it was human"



1957
First Perceptron
1957
First AI program
1965
First Ind. Robot
1965
Eliza ChatBot!

1986
Back propagation

"Multi-layer networks could learn non-linear mappings, significantly contributing to the field's progress"

1994
Robocup
1997
LSTM ANN
1999
Sony AIBO
2000
Kismet Robot

2012
AlexNet

"A Revolutionary Deep Learning Architecture significantly outperforms ImageNet"

2013
DeepMind
2014
Generative AN
2015
Tensorflow
2016
Alpha Go

2020
DALL-E

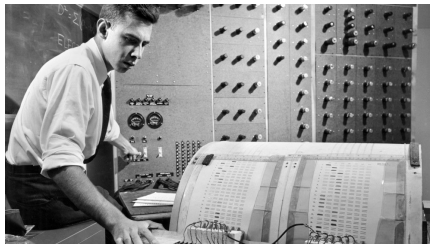
"generating images from descriptions, showcases the power of AI in cross-modal tasks and creative content generation"

2020
Tesla Self drive
2021
Regulation & Ethics
2024
Sora LVM
2024
EU AI Act

+2025
AGI
ASI
Singularity
SuperHumans
&
Beyond

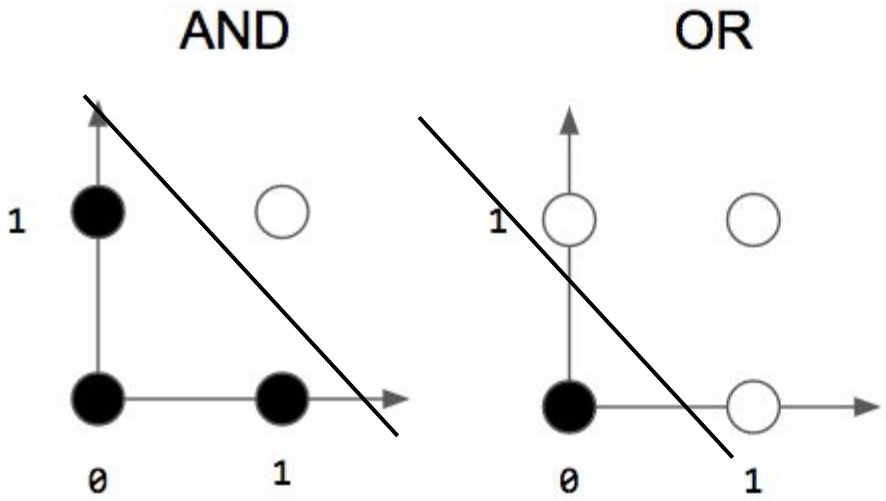
INSTITUTO FEDERAL
Goiás
Câmpus Goiânia

Perceptron

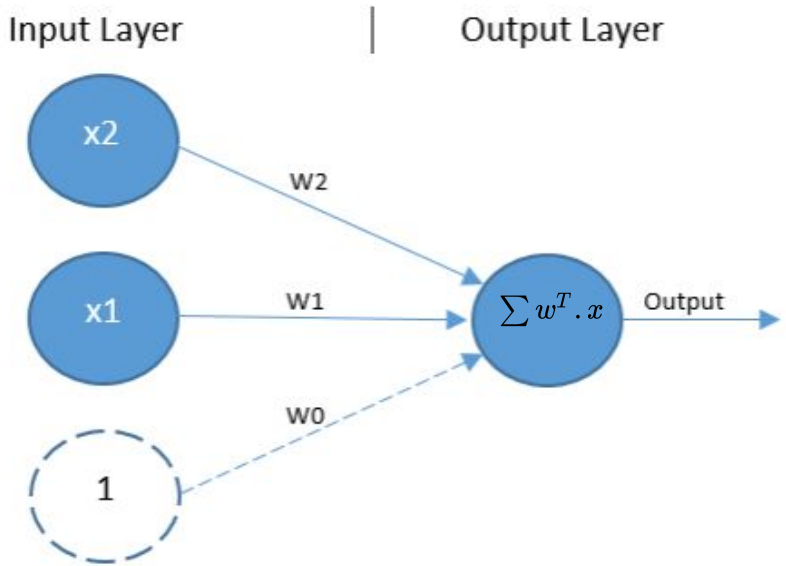


Frank Rosenblatt, Perceptron (1957, 1962): Inspirado pela maneira como os neurônios trabalham juntos no cérebro. O perceptron é uma rede neural de camada única - um algoritmo que classifica a entrada em categorias possíveis. A rede neural faz uma previsão de, direita ou esquerda; ou cachorro ou gato, etc.

Classificador Perceptron



Rede Perceptron



$$\begin{cases} y = 1 \ \ se \ \ \sum_{i=0}^{n} w_i \cdot x_i \geq 0 \\ y = 0 \ \ se \ \ \sum_{i=0}^{n} w_i \cdot x_i < 0 \end{cases}$$

ou

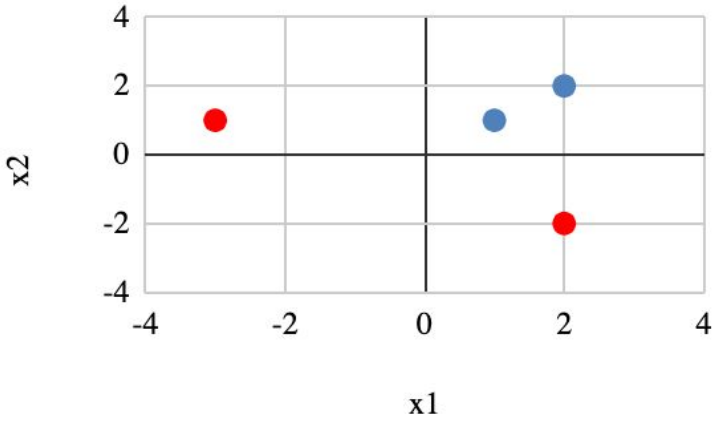
$$\begin{cases} y = 1 \ \ se \ \ w_0 + \sum_{i=1}^{n} w_i \cdot x_i \geq 0 \\ y = 0 \ \ se \ \ w_0 + \sum_{i=1}^{n} w_i \cdot x_i < 0 \end{cases}$$

Exemplo:

Dado um conjunto de vetores de entrada $\{x^1,..., x^P\}$, e um conjunto de rótulos (labels) desejáveis $\{d^1,..., d^P\}$, nós queremos encontrar um algoritmo que, começando de algum **vetor de peso *w*** inicial, ele irá modificá-lo baseado nos exemplos fornecidos produzindo o conjunto de pesos w que **classifica corretamente todos os exemplos**.

Os exemplos são apresentados um por um em cada passo, e uma **regra de atualização de peso** é aplicada. Uma vez que todos os exemplos são apresentados, o algoritmo percorre novamente todos os exemplos, até a convergência.

Evento	x_1	x_2	d
1	1	1	1
2	2	2	1
3	-3	1	0
4	2	-2	0



Parâmetros e Entradas

$$y = w^T . x$$

0 Bias = 1

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \quad [\, x_0 x_1 x_2 \,]$$

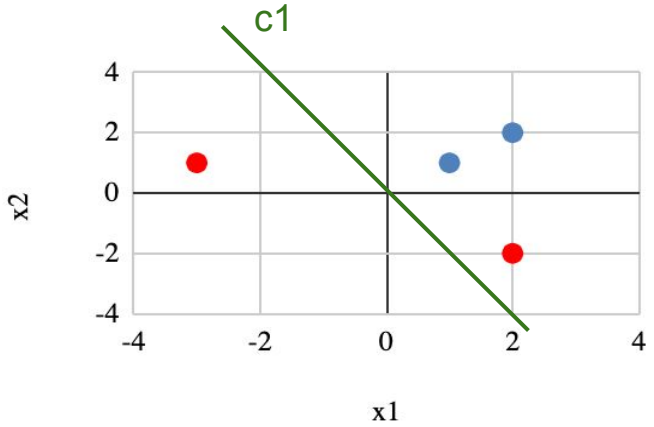
$$\begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix} [\, 1, x_1, x_2 \,]$$

Classificador → $(0 * 1) + (1 * x_1) + (0.5 * x2) = 0$

Classificador → $x_2 = -2x_1$

x1=0, x2=0
x1=1, x2=-2

(0, 0) e (1, -2)



$$\begin{cases} w_0.1 + w_1.x1 + w_2.x2 > 0 \\ \text{Classificador} \rightarrow x_1 + 0.5x_2 > 0 \\ \text{Classificador} \rightarrow x_2 > -2x_1 \end{cases}$$

$$\begin{cases} w_0.1 + w_1.x1 + w_2.x2 \leq 0 \\ \text{Classificador} \rightarrow x_1 + 0.5x_2 \leq 0 \\ \text{Classificador} \rightarrow x_2 \leq -2x_1 \end{cases}$$

INSTITUTO FEDERAL
Goiás

Verificação das classificações

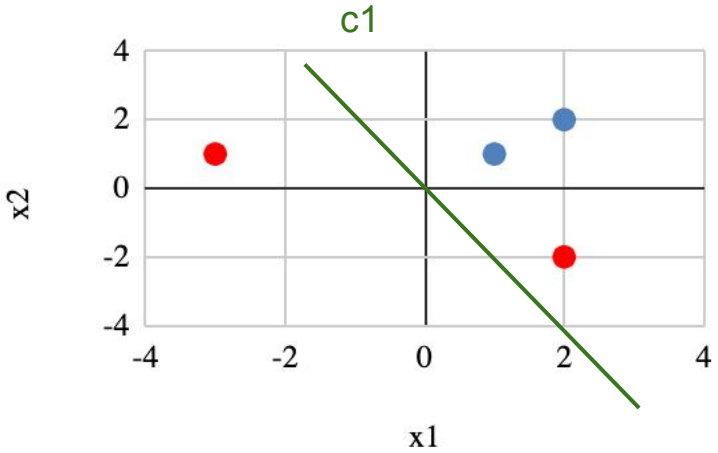
Verificando ponto P(-3,1)

$$\begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix} [1, -3, 1] \qquad 0 - 3 + 0.5 = -2.5 \leq 0 \qquad \text{y=0}$$

Verificando ponto P(2,-2)

$$\begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix} [1, 2, -2] \qquad 0 + 2 - 1 = 1 \geq 0 \qquad \textbf{y=1}$$

$$\begin{cases} w^T . x \geq 0 (y = 1) \\ w^T . x < 0 (y = 0) \end{cases}$$

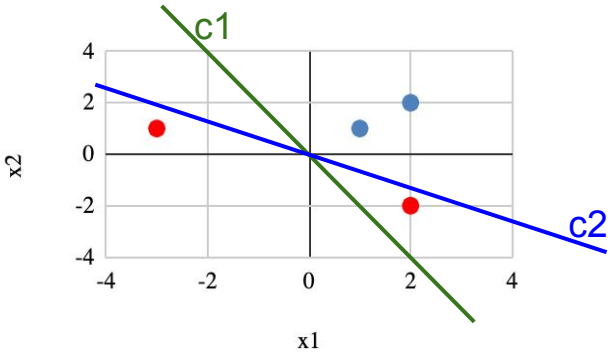


Atualização dos Pesos - Stochastic Gradient Descent (SGD)

Atualizar os pesos para o ponto **(2, -2)**

$$\begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix} [1, 2, -2]$$

$$\begin{cases} w^T . x \geq 0 (y = 1) \\ w^T . x < 0 (y = 0) \end{cases}$$



$$w_i = w_i + \Delta w_i$$

Learning rate

Erro

$$\Delta w_i = \alpha . (d - y) . x_i$$

$$\Delta w_0 = 0,2 . (0 - 1) . 1 = -0,2$$

$$\Delta w_1 = 0,2 . (0 - 1) . 2 = -0,4$$

$$\Delta w_2 = 0,2 . (0 - 1) . -2 = 0,4$$

$$w_0 = w_0 + \Delta w_0 = 0 - 0,2 = -0,2$$

$$w_1 = w_1 + \Delta w_1 = 1 - 0,4 = 0,6$$

$$w_2 = w_2 + \Delta w_2 = 0,5 + 0,4 = 0,9$$

Novo Classificador

$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, x_1, x_2]$$

$$x_2 = -0,67 x_1 + 0,22$$

(0, 0.22)
(4, -2.46)

INSTITUTO FEDERAL
Goiás

Atualização dos Pesos - Stochastic Gradient Descent (SGD)

Atualizar os pesos passando por todos os pontos (Época).

Evento	x_1	x_2	d
1	1	1	1
2	2	2	1
3	-3	1	0
4	2	-2	0

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \alpha.(d - y).x_i$$

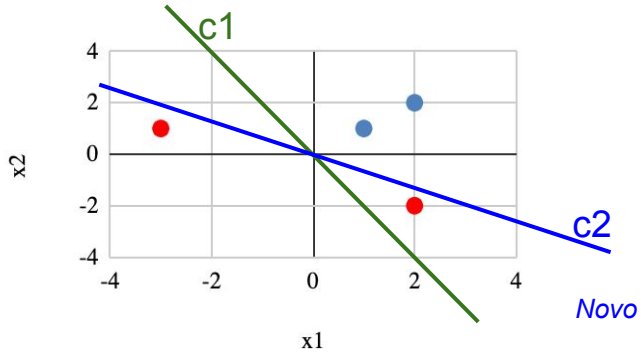
Learning rate

Erro

$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, x_1, x_2]$$

$$x_2 = -0,67x_1 + 0,22$$

(0, 0.22)
(4, -2.46)



Novo Classificador

Verificação das Classificações

Verificando ponto P(-3,1)

$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, -3, 1]$$

$$(-0,2 \cdot 1) + (0,6 \cdot -3) + (0,9 \cdot 1) = -1,1 < 0$$ y=0

$$\begin{cases} w^T . x \geq 0 (y = 1) \\ w^T . x < 0 (y = 0) \end{cases}$$

Verificando ponto P(2,-2)

$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, 2, -2]$$

$$(-0,2 \cdot 1) + (0,6 \cdot 2) + (0,9 \cdot -2) = -0,8 < 0$$ y=0

Verificando ponto P(1,1)

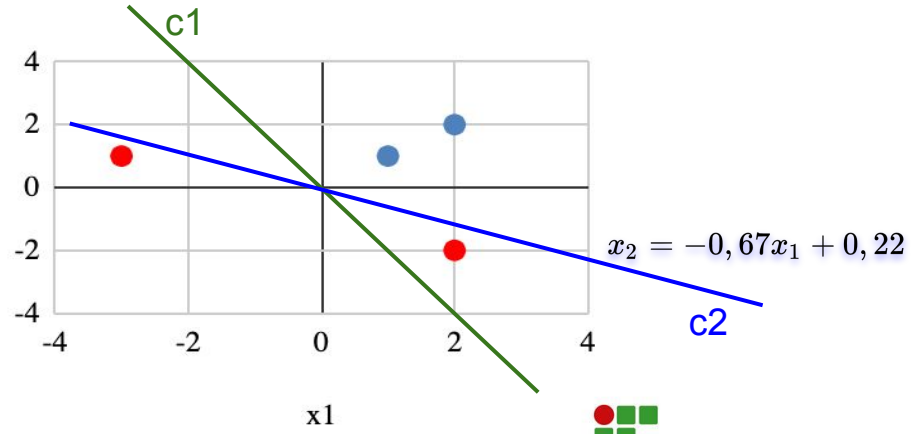
$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, 1, 1]$$

$$(-0,2 \cdot 1) + (0,6 \cdot 1) + (0,9 \cdot 1) = 1,3 > 0$$ y=1

Verificando ponto P(2,2)

$$\begin{bmatrix} -0,2 \\ 0,6 \\ 0,9 \end{bmatrix} [1, 2, 2]$$

$$(-0,2 \cdot 1) + (0,6 \cdot 2) + (0,9 \cdot 2) = 2,8 > 0$$ y=1



INSTITUTO FEDERAL
Goiás

Algoritmo Perceptron

$$W_j \leftarrow W_j + \alpha \times I_j \times Err$$

Perceptron learning rule:

1. Start with random weights, $\mathbf{w} = (w_1, w_2, \ldots, w_n)$.
2. Select a training example $(\mathbf{x}, y) \in S$.
3. Run the perceptron with input $\mathbf{x}$ and weights $\mathbf{w}$ to obtain g
4. Let α be the training rate (a user-set parameter).

$$\forall w_i,\ w_i \leftarrow w_i + \Delta w_i,$$

where

$$\Delta w_i = \alpha(y - g(in))g'(in)x_i$$

5. Go to 2.

Epoch → cycle through the examples

Epochs are repeated until some stopping criterion is reached—typically, that the weight changes have become very small.

The stochastic gradient method selects examples randomly from the training set rather than cycling through them.

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \alpha.(d - y).x_i$$

Aprendendo com Perceptron

Load and return the digits dataset (classification).

Each datapoint is a 8x8 image of a digit.

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

Parâmetros

sklearn.linear_model.Perceptron

class sklearn.linear_model.**Perceptron**(*, *penalty=None*, *alpha=0.0001*, *l1_ratio=0.15*, *fit_intercept=True*, *max_iter=1000*, *tol=0.001*, *shuffle=True*, *verbose=0*, *eta0=1.0*, *n_jobs=None*, *random_state=0*, *early_stopping=False*, *validation_fraction=0.1*, *n_iter_no_change=5*, *class_weight=None*, *warm_start=False*) [source]

```
from sklearn.datasets import load_digits
from sklearn.linear_model import Perceptron
X, y = load_digits(return_X_y=True)
clf = Perceptron(tol=1e-3, random_state=0)
clf.fit(X, y)
clf.score(X, y)
```

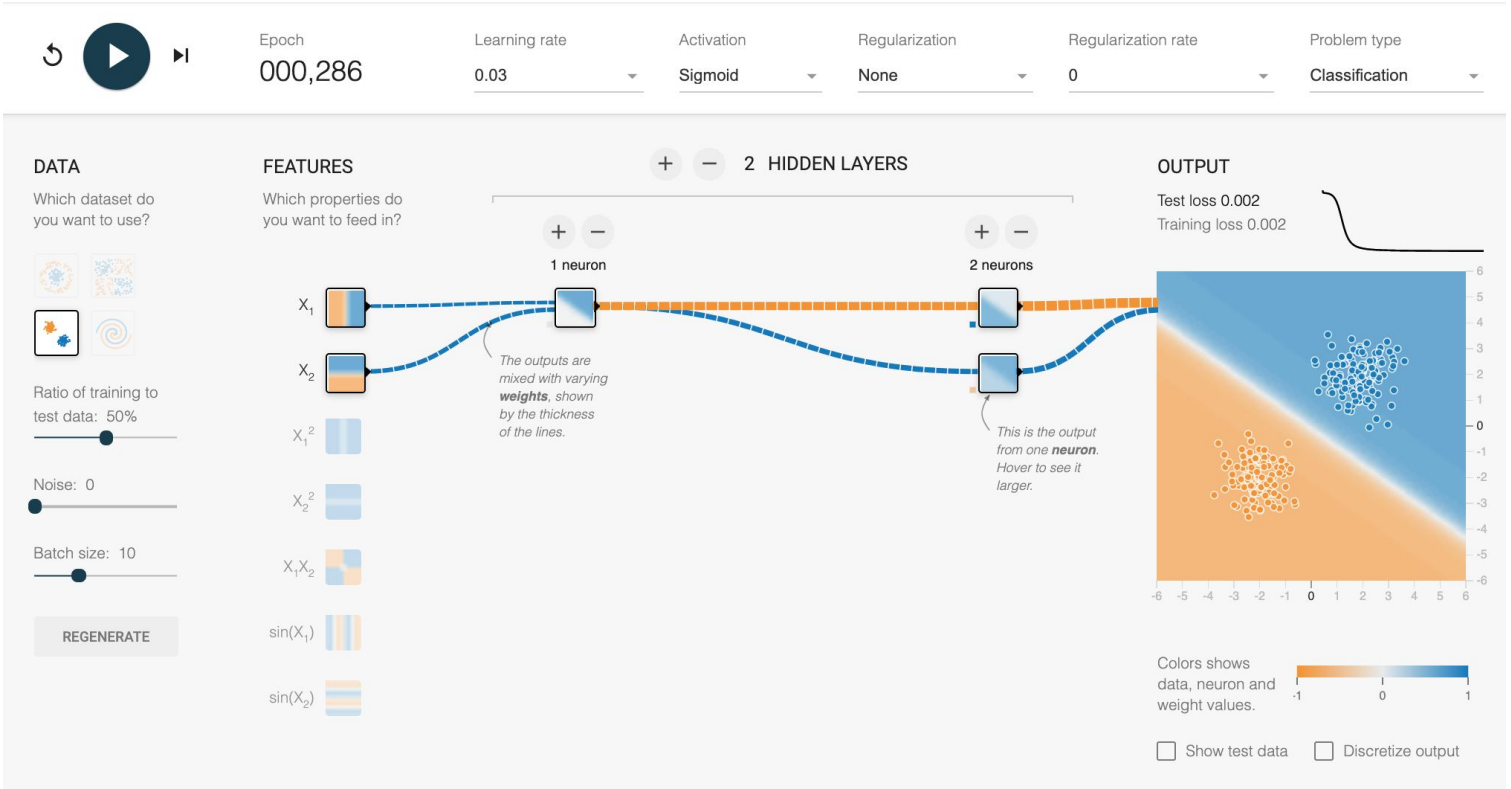
```
0.9393433500278241
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

Exercícios

- Implementar o perceptron e testar para os problemas do **AND** e **OR**
- Testar bibliotecas do Perceptron para o Dataset (load Digits)
  - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html
  -
-

Playground - Tensorflow



Regularization

$$
\begin{aligned}
Cost(h) &= EmpLoss(h) + \lambda\, Complexity(h) \\
\hat{h}^* &= \operatorname*{argmin}_{h \in \mathcal{H}} Cost(h)\ .
\end{aligned}
$$

Tipos de Erros

1. Mean Absolute Error (MAE) is the mean of the absolute value of the errors. It is calculated as:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|$$

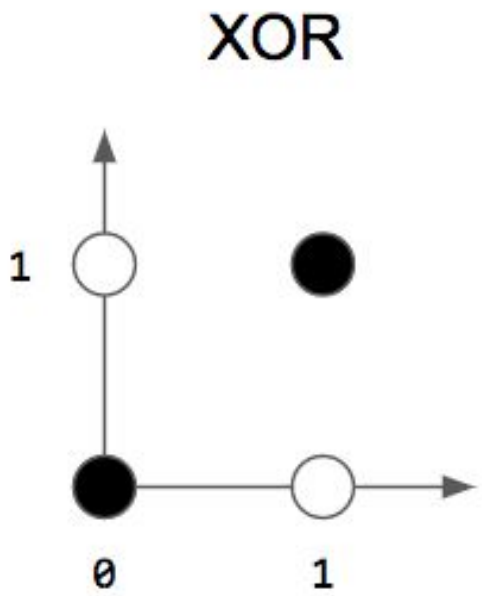
2. Mean Squared Error (MSE) is the mean of the squared errors and is calculated as:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|^2$$

3. Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

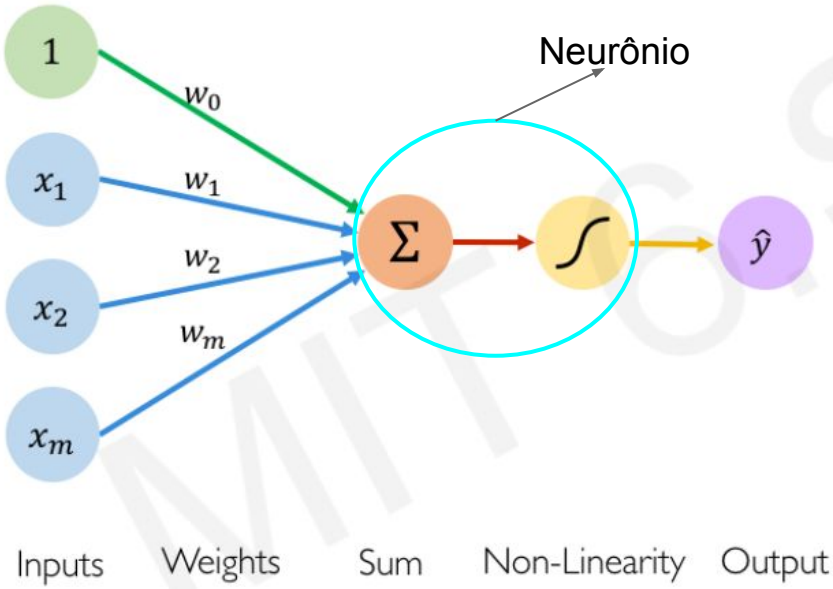
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|^2}$$

E o XOR?



INSTITUTO FEDERAL
Goiás

Inserção da Não Linearidade

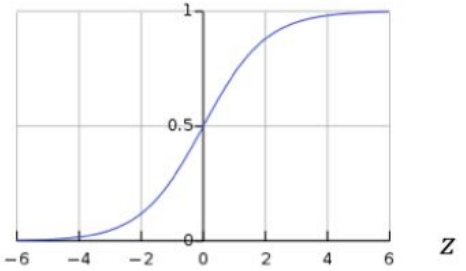


Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W}\right)$$

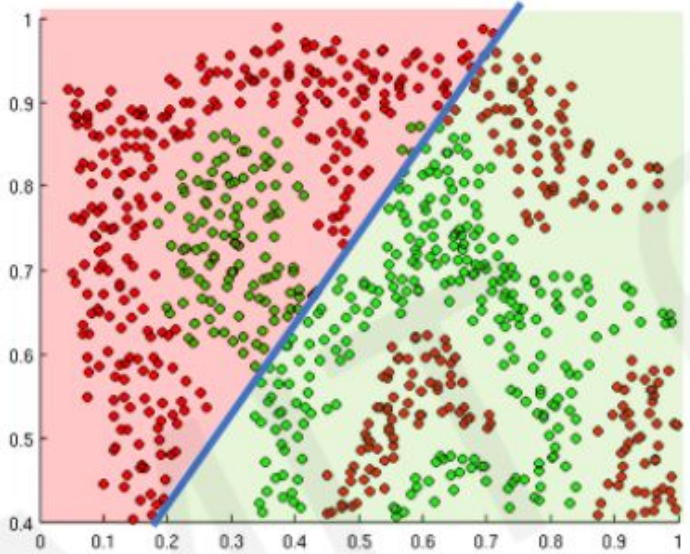
- Example: sigmoid function

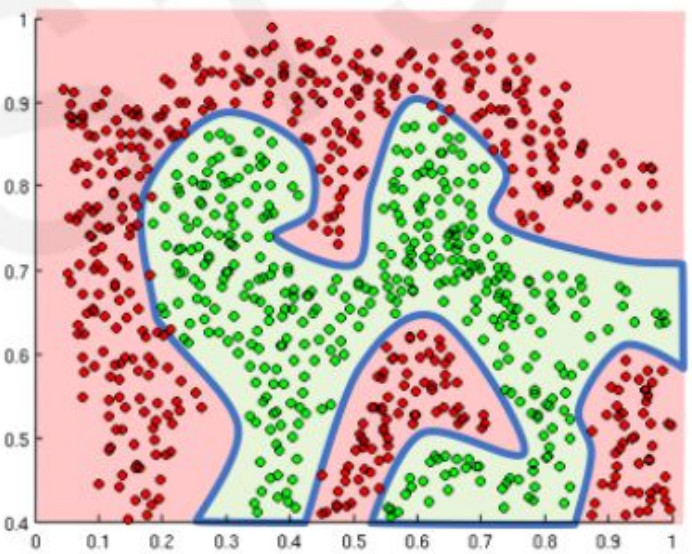
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Fonte: https://introtodeeplearning.com/

Importância da função de ativação



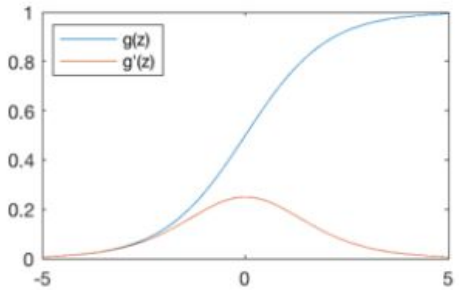


Fonte: https://introtodeeplearning.com/

INSTITUTO FEDERAL
Goiás

Funções de ativação

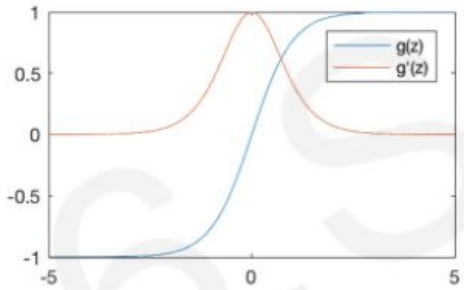
Sigmoid Function



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

$$g'(z) = g(z)(1 - g(z))$$

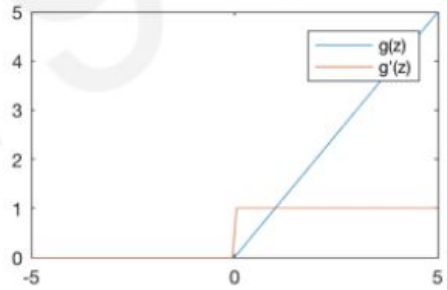
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)

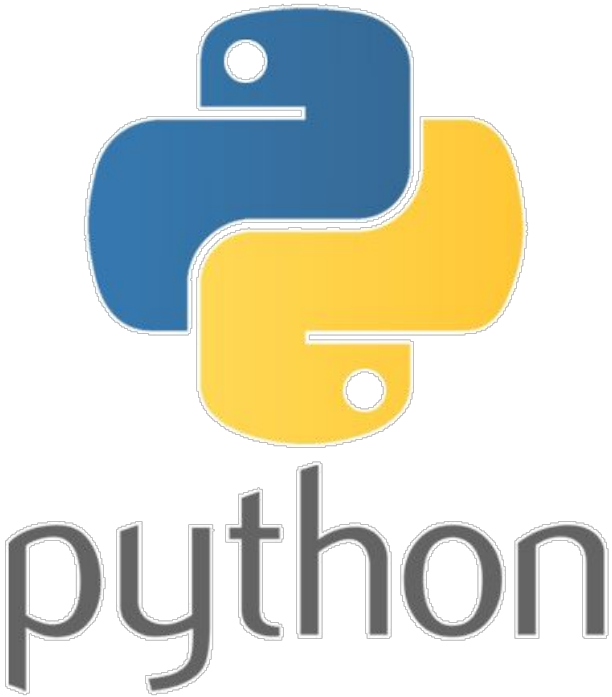


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Fonte: https://introtodeeplearning.com/

Vamos para um exemplo prático em Python!



INSTITUTO FEDERAL
Goiás

Desafio da semana

- Leitura do Capítulo 18 (Artificial Intelligence)
- Pesquisa Aplicada: prever o preço do Bitcoin com base na variação do preço de outras criptomoedas (Escolher pelo menos 3 - https://coinmarketcap.com/)
-