Redes Neurais Artificiais - Parte 1

Exemplos e Aplicações em Python

Lucas Migliorin da Rosa

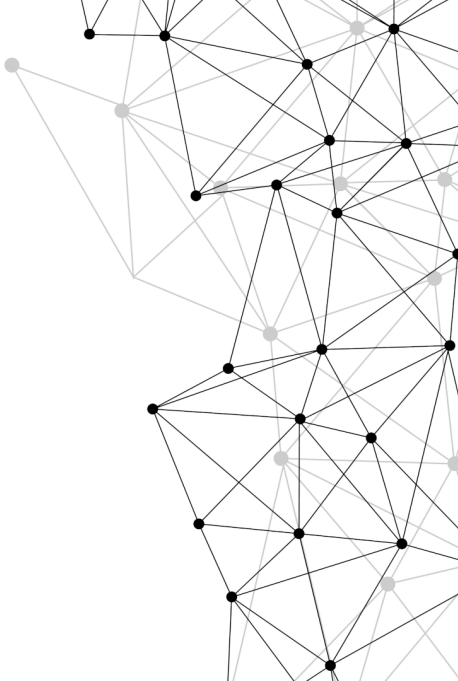


Table of Contents

1 Breve Introdução

O que é um neurônio?

1 Breve Introdução

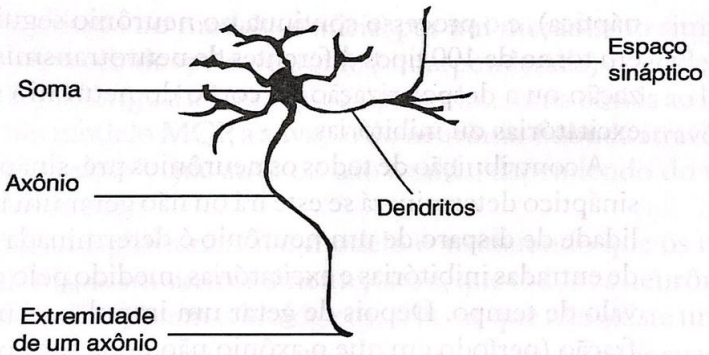


Figure: Componentes do neurônio biolôgico

Zona de ativação

1 Breve Introdução

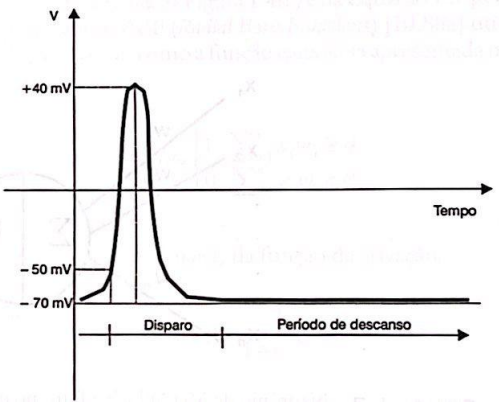


Figure: Potencial de ação de um neurônio

Table of Contents

2 Componentes Neurônio

Abstração e interpretações

2 Componentes Neurônio

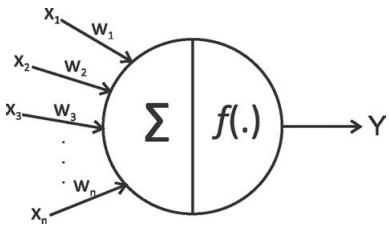


Figure: Modelo inicial proposto por McCulloch e Pitts em 1943

1. Entradas de X_i à X_n
2. Pesos de W_i à W_n

Visualização Matemática

2 Componentes Neurônio

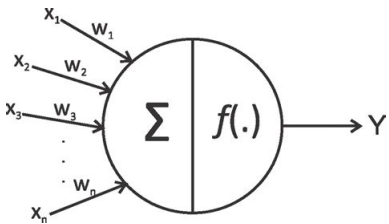


Figure: Modelo inicial proposto por McCulloch e Pitts em 1943

- $x = \sum_{i=1}^{N} X_i * W_i$
- $x = \begin{bmatrix} X_1, & X_2, & \cdots & X_n \end{bmatrix} * \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix}$

Visualização Matemática

2 Componentes Neurônio

Código em Python

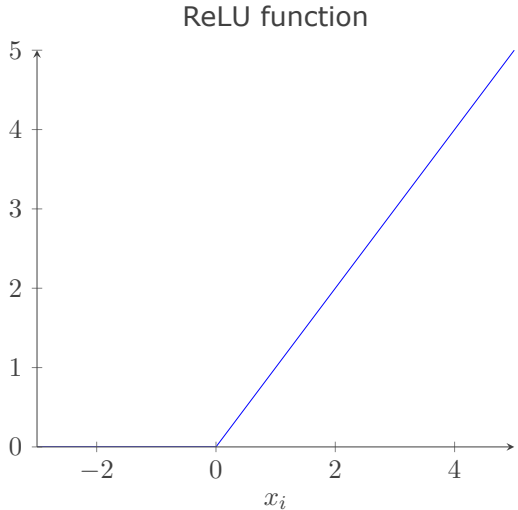
```
>>> import numpy as np
>>> inputs_neuron = np.array(
[0.120, 0.635, -0.219, 1.7632])
>>> weights_neuron = np.array([[0.123],
                               [-0.323],
                               [-1.231],
                               [2.322]])
>>> result = inputs_neuron @ weights_neuron
>>> print(result) # Produto vetorial
array([4.1733944])
```

- $x = \sum_{i=1}^{N} X_i * W_i$
- $x = \begin{bmatrix} X_1, & X_2, & \cdots & X_n \end{bmatrix} * \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix}$

Table of Contents

Funções Básicas

3 Funções de Ativação



$$ReLU(x_i) = \begin{Bmatrix} 0, & x_i < 0 \\ x_i, & x_i \geq 0 \end{Bmatrix} \quad (1)$$

Funções Básicas

3 Funções de Ativação

Código em Python

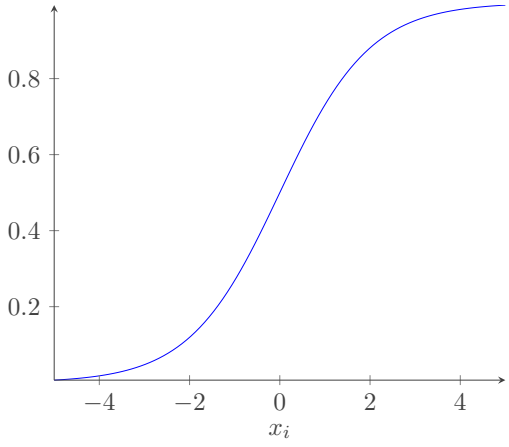
```
>>> import numpy as np
>>> result = np.array([4.1733944, -1.2121])
>>> relu = lambda x: x if x >= 0 else 0
>>> print(relu(result[0]))
[4.1733944]
>>> print(relu(result[1]))
[0]
```

$$ReLU(x_i) = \begin{cases} 0, & x_i < 0 \\ x_i, & x_i \geq 0 \end{cases}$$

Funções Básicas

3 Funções de Ativação

Sigmoid function



$$Sigmoid(x_i) = \frac{1}{1 + e^{-x_i}} \qquad (2)$$

Funções Básicas

3 Funções de Ativação

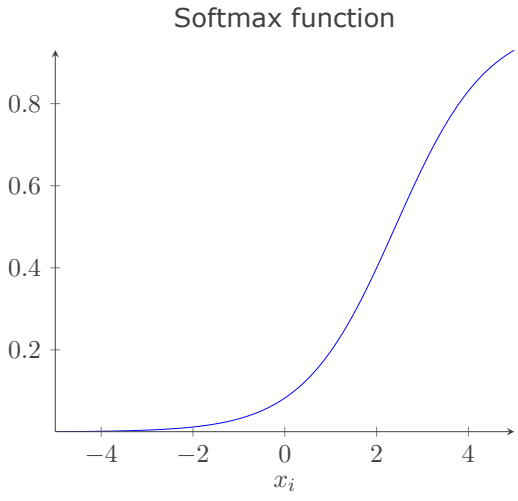
Código em Python

```
>>> import numpy as np
>>> result = np.array([4.1733944, -1.2121])
>>> sigmoid = lambda x: 1/(1 + np.exp(-x))
>>> print(sigmoid(result))
[0.98483366 0.22932969]
```

$$Sigmoid(x_i) = \frac{1}{1 + e^{-x_i}}$$

Funções Básicas

3 Funções de Ativação



$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \qquad (3)$$

Funções Básicas

3 Funções de Ativação

Código em Python

```
>>> import numpy as np
>>> result = np.array([4.1733944, -1.2121])
>>> softmax =
        lambda x,x2: np.exp(x)/(sum(np.exp(x2)))
>>> print(softmax(result[0],result))
[0.9954383300532891]
>>> print(softmax(result[1],result))
[0.0045616699467109]
```

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Table of Contents

Fluxo da Informação

4 Recapitulando

$$x = \sum_{i=1}^{N} X_i * W_i \longrightarrow \{Softmax(x), ReLU(x), Sigmoid(x)\}$$

Aplicando em Python

4 Recapitulando

Escolhendo a função Sigmoidal como função de ativação

Código em Python

```
>>> import numpy as np
>>> inputs_neuron = np.array([0.120, 0.635, -0.219, 1.7632])
>>> weights_neuron = np.array([[0.123],[-0.323],[-1.231],[2.322]])
>>>
>>> result = inputs_neuron @ weights_neuron
>>> sigmoid = lambda x: 1/(1 + np.exp(-x))
>>>
>>> result_final = sigmoid(result)
>>> print(result_final)
>>> 0.98483366
```

Fim da parte 1!!