

RNA - Atividade

Representação matemática das Funções de Ativação mais usadas

1) Criar no Google Colab o notebook: **RNA – Funções de Ativação.ipynb**

2) Importar a biblioteca matemática **numpy**.

3) Para cada função de ativação a seguir, criar:

a – uma área de texto informando em quais tipos de problemas aquela função é mais utilizada;

b – incluir nessa área de texto o gráfico da função (estão anexos a seguir);

c – uma célula com a representação daquela função na linguagem Python;

d – uma chamada para execução da função, com o valor do parâmetro;

e – a impressão do valor retornado pela função.

4) Enviar o link do notebook do Colab, pelo privado, até o dia 03/06.



+ Code + Text



{x}



▼ RNA - Funções de Ativação

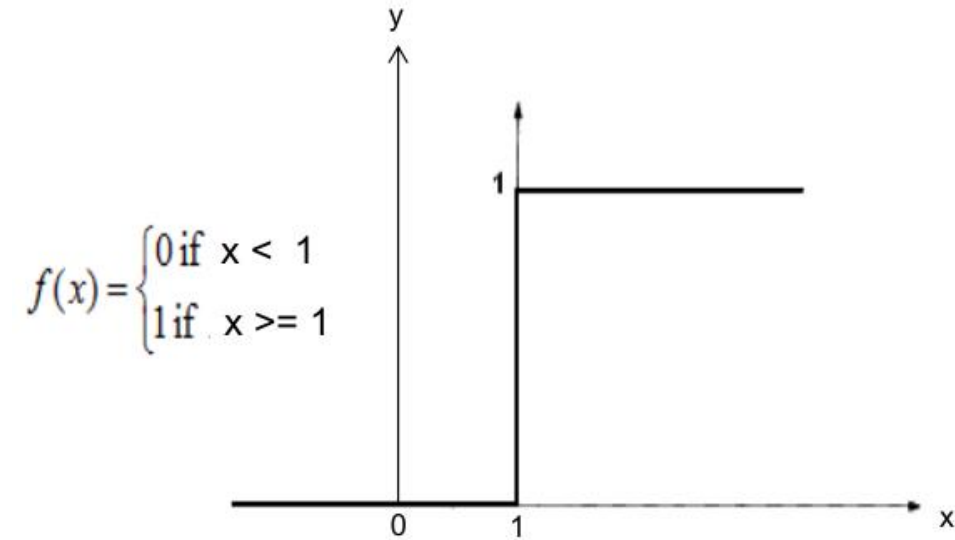


✓ RNA - Funções de Ativação

```
▶ # importação da biblioteca matemática  
import numpy as np
```

Exemplo: função Degrau

Step - função Degrau



Usada em problemas linearmente separáveis

Step - função Degrau

```
# função Degrau
def stepFunction(soma):
    if (soma >= 1):
        return 1
    return 0

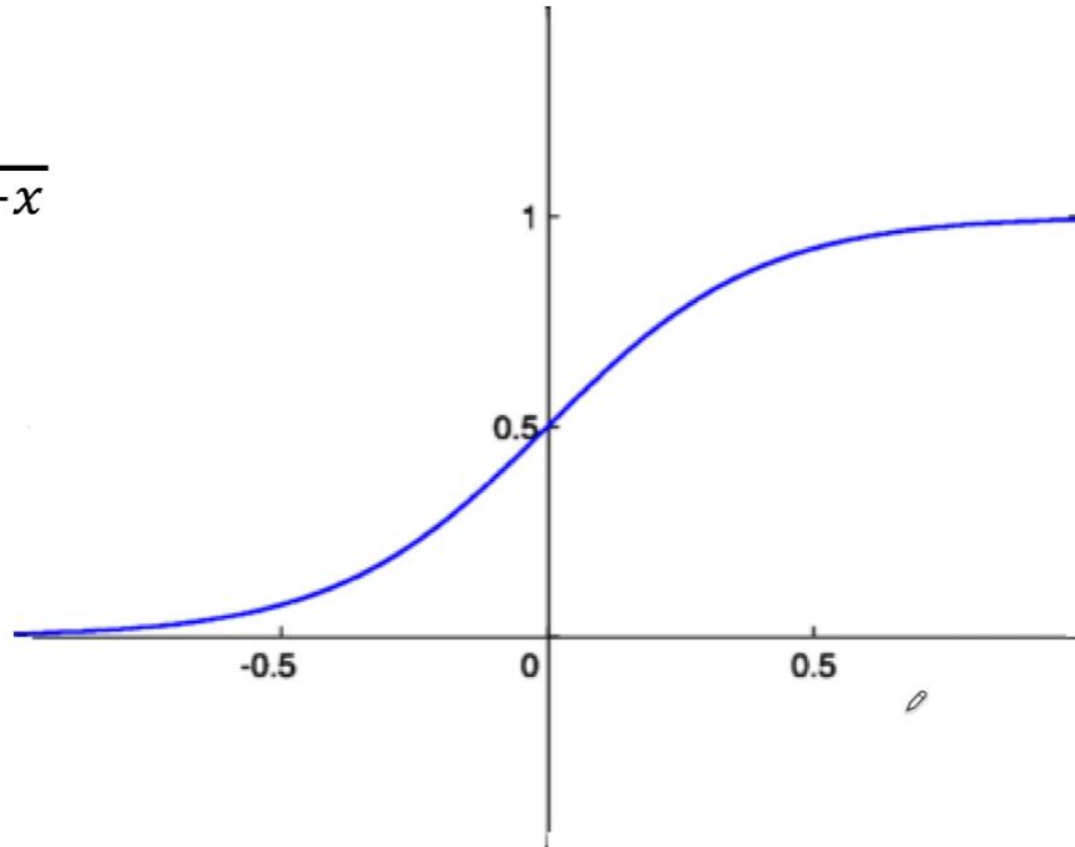
# chamada das funções
step = stepFunction(-1)
print(step)
```

0

Fazer para as seguintes funções de Ativação:

Sigmoid - função Sigmoide

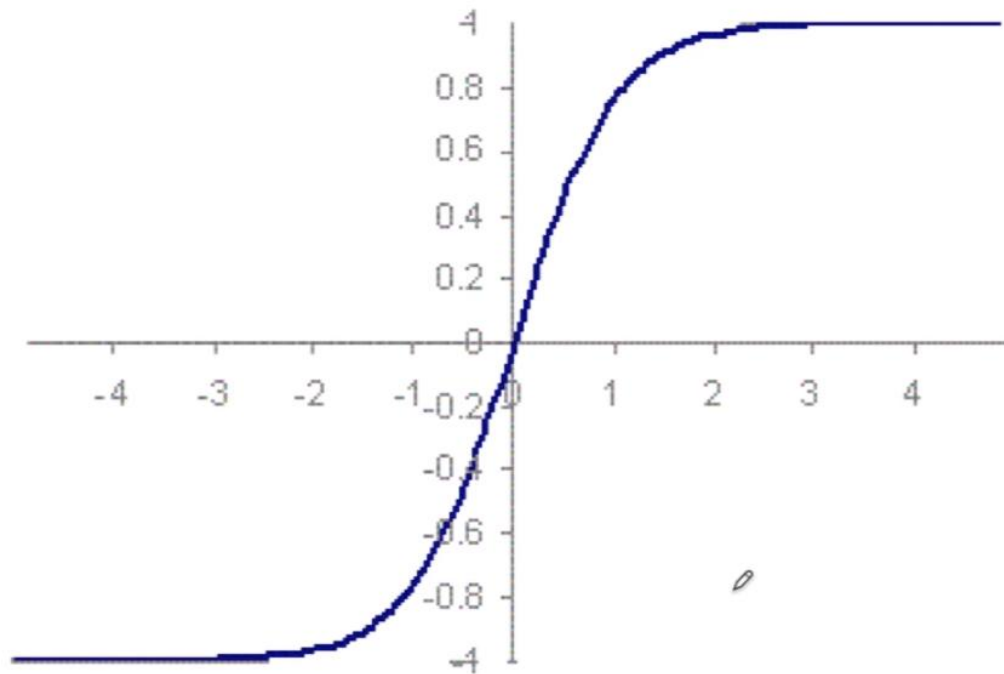
$$y = \frac{1}{1 + e^{-x}}$$



Valores entre 0 e 1

Hyperbolic tangent - função Tangente Hiperbólica

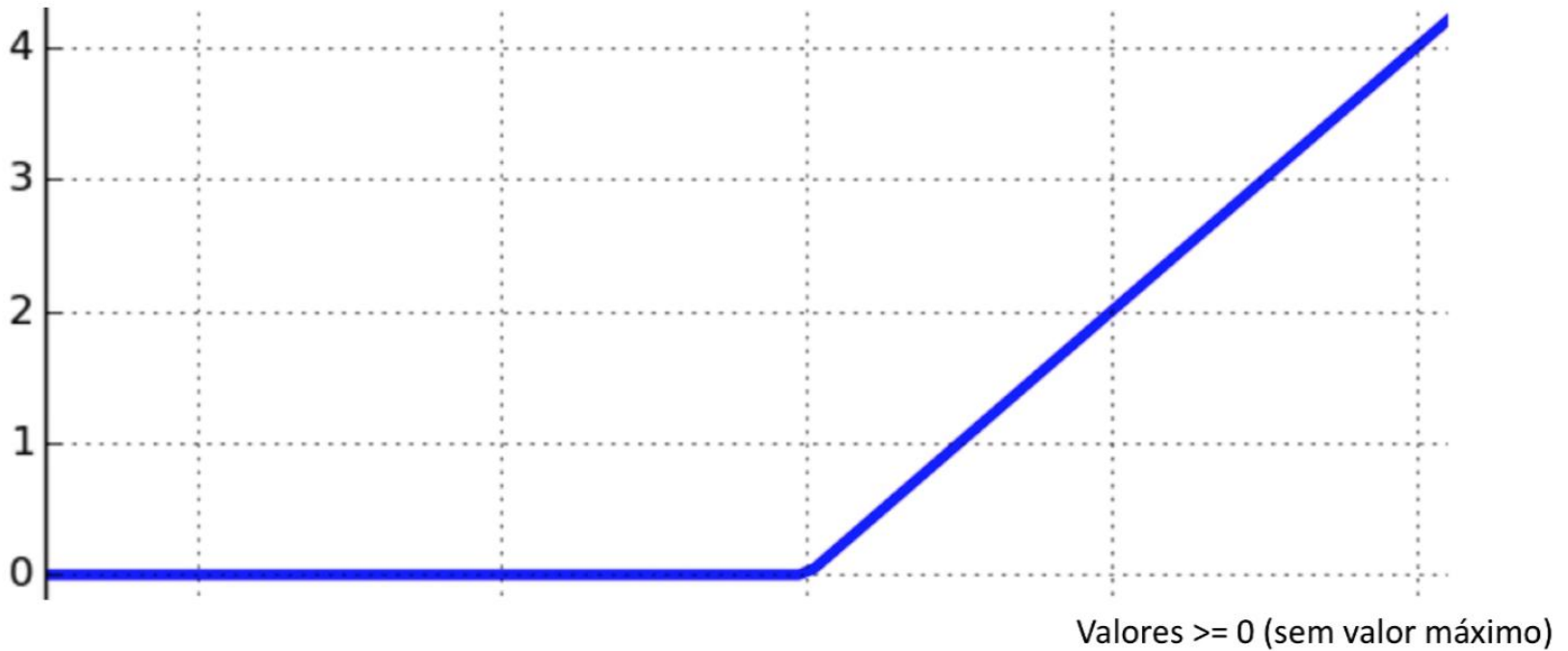
$$Y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



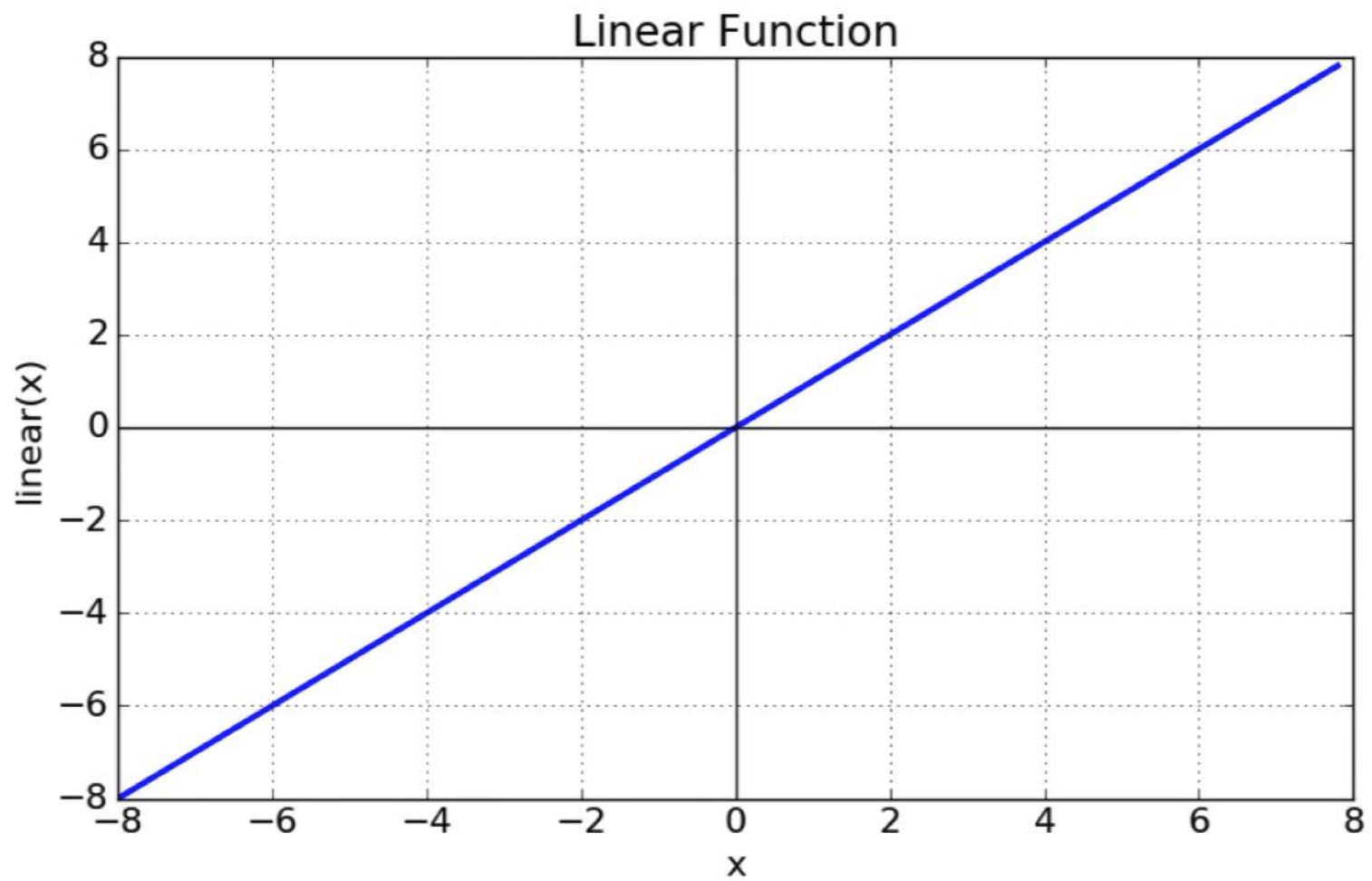
Valores entre -1 e 1

ReLU - Rectified Linear Units

$$Y = \max(0, x)$$

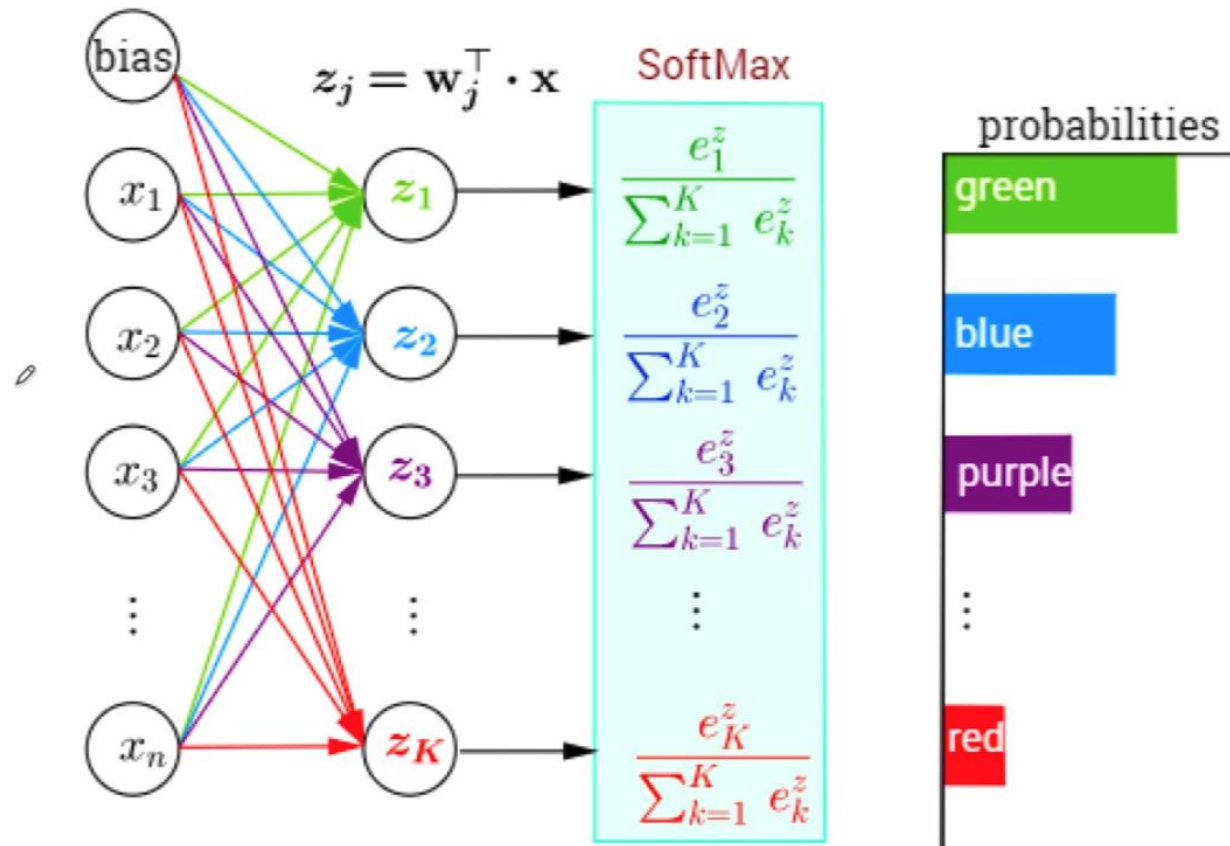


Linear



Softmax

$$Y = \frac{e(x)}{\sum e(x)}$$



Fonte de consultas

K

Keras Documentation

Home

Keras: The Python Deep Learning library

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Configuring your Keras backend

Support

Why this name, Keras?

Why use Keras

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

Layers

About Keras layers

Core Layers

Convolutional Layers

Recurrent Layers

Keras: The Python Deep Learning library



You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of **TensorFlow**, **CNTK**, or **Theano**. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at **Keras.io**.

Keras is compatible with: **Python 2.7-3.6**.

Guiding principles

Keras: The Python Deep Learning library



You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at [Keras.io](#).

Keras is compatible with: **Python 2.7-3.6**.

Guiding principles

[FAQ](#)

[Models](#)

[About Keras models](#)

[Sequential](#)

[Model \(functional API\)](#)

[Layers](#)

[About Keras layers](#)

[Core Layers](#)

[Convolutional Layers](#)

[Pooling Layers](#)

[Locally-connected Layers](#)

[Recurrent Layers](#)

[Embedding Layers](#)

[Merge Layers](#)

[Advanced **Activations** Layers](#)

[Normalization Layers](#)

[Noise layers](#)

[Layer wrappers](#)

[Writing your own Keras layers](#)

[Preprocessing](#)

[Sequence Preprocessing](#)

[Text Preprocessing](#)

[Image Preprocessing](#)

[Losses](#)

[Metrics](#)

[Optimizers](#)

[Activations](#)



Home

Why use Keras

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

Layers

About Keras layers

Core Layers

Convolutional Layers

Pooling Layers

Locally-connected Layers

Recurrent Layers

Embedding Layers

Merge Layers

Advanced Activations Layers

Normalization Layers

Noise layers

Layer wrappers

- **alpha**: Slope of the negative part. Defaults to zero.
- **max_value**: Maximum value for the output.

Returns

The (leaky) rectified linear unit activation: x if $x > 0$, $\alpha * x$ if $x < 0$. If **max_value** is defined, the result is truncated to this value.

tanh

```
tanh(x)
```

Hyperbolic tangent activation function.

sigmoid

```
sigmoid(x)
```

Sigmoid activation function.

hard_sigmoid

```
hard_sigmoid(x)
```

Hard sigmoid activation function.

Faster to compute than sigmoid activation.

Arguments

- **x**: Input tensor.