Atividade: Neurônio Artificial

INSTITUTO FEDERAL DE MINAS GERAIS *Departamento de Engenharia e Computação*

Aluno: César Nogueira Rodrigues

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Exercício 1: Neurônio

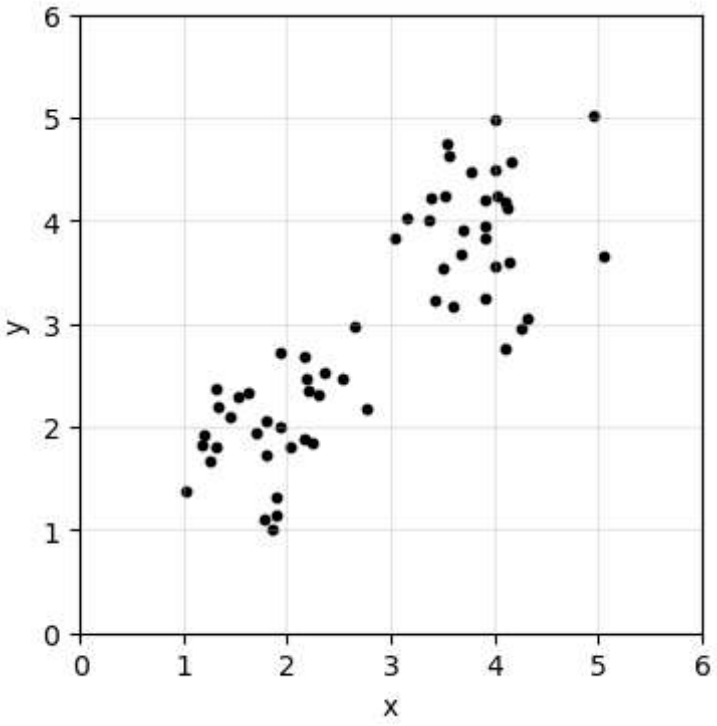
In [12]:
```
def neuronio(x, y, w0, w1, w2, bias):
    u = (w1 * x) + (w2 * y) - (w0 * bias)
    return 1 if u > 0 else 0
```

In [13]:
```
dados = pd.read_csv("amostrabivariada.csv", sep=";", decimal=",")

dados["x"] = pd.to_numeric(dados["x"], errors="coerce")
dados["y"] = pd.to_numeric(dados["y"], errors="coerce")

dados.head()
```

Out[13]:

	x	y
0	1.183988	1.832880
1	1.523565	2.293337
2	2.199241	2.342880
3	2.768052	2.179136
4	2.165374	1.888445

In [14]:
```
plt.figure(figsize=(4, 4))
plt.scatter(dados["x"], dados["y"], c="black", s=10)
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True, alpha=0.3)
plt.show()
```



In [15]:
```
w0 = -6
w1 = -1
w2 = -1
bias = 1
```

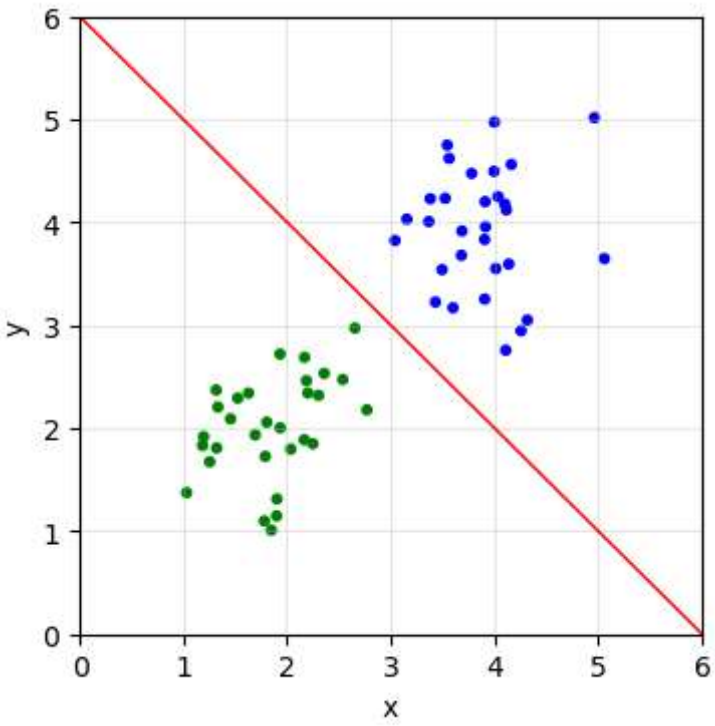
In [16]:
```
classificacoes = []
for i, linha in dados.iterrows():
    classe = neuronio(linha["x"], linha["y"], w0, w1, w2, bias)
    classificacoes.append(classe)

dados["classe"] = classificacoes
dados.head()
```

Out[16]:

	x	y	classe
0	1.183988	1.832880	1
1	1.523565	2.293337	1
2	2.199241	2.342880	1
3	2.768052	2.179136	1
4	2.165374	1.888445	1

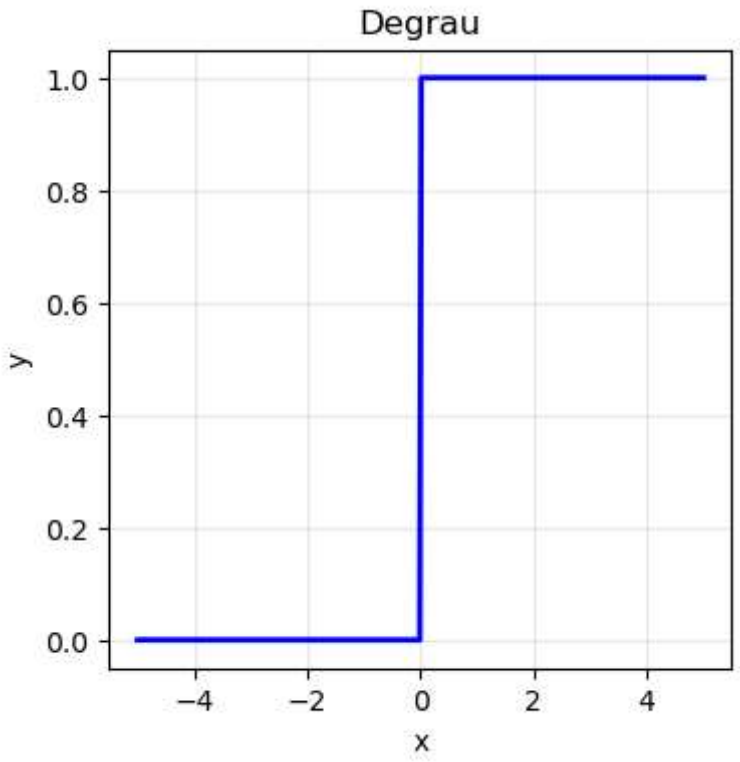
In [17]:
```
plt.figure(figsize=(4, 4))
cores = ["blue" if c == 0 else "green" for c in dados["classe"]]
plt.scatter(dados["x"], dados["y"], c=cores, s=10)

eixox = np.linspace(0, 6, 100)
eixoy = (w0 * -1) - eixox
plt.plot(eixox, eixoy, "-r", linewidth=1)

plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.grid(True, alpha=0.3)
plt.show()
```

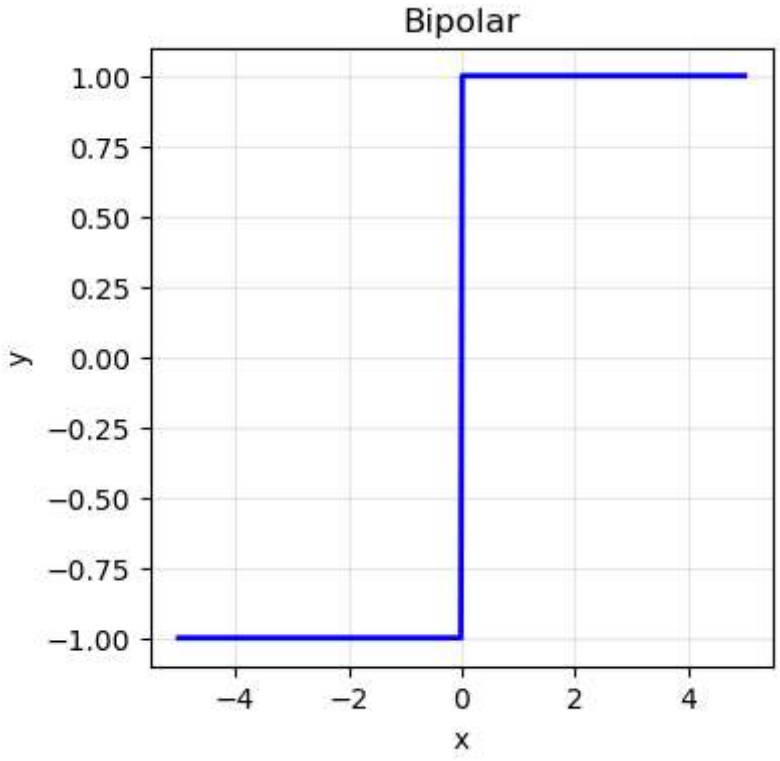


In [ ]:
```
x_valores = np.linspace(-5, 5, 400)
def plota_grafico(x, y, titulo):
    plt.figure(figsize=(4, 4))
    plt.plot(x, y, '-b', linewidth=2)
    plt.title(titulo)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True, alpha=0.3)
    plt.show()
```

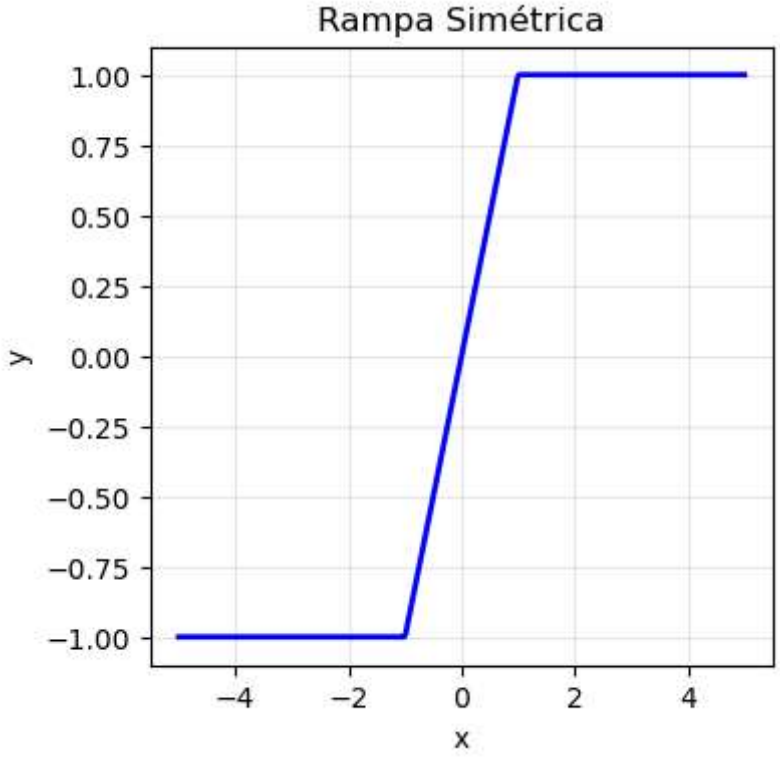
In [29]:
```
def degrau(x):
    return np.where(x >= 0, 1, 0)


y_val = degrau(x_valores)

plota_grafico(x_valores, y_val,'Degrau')
```



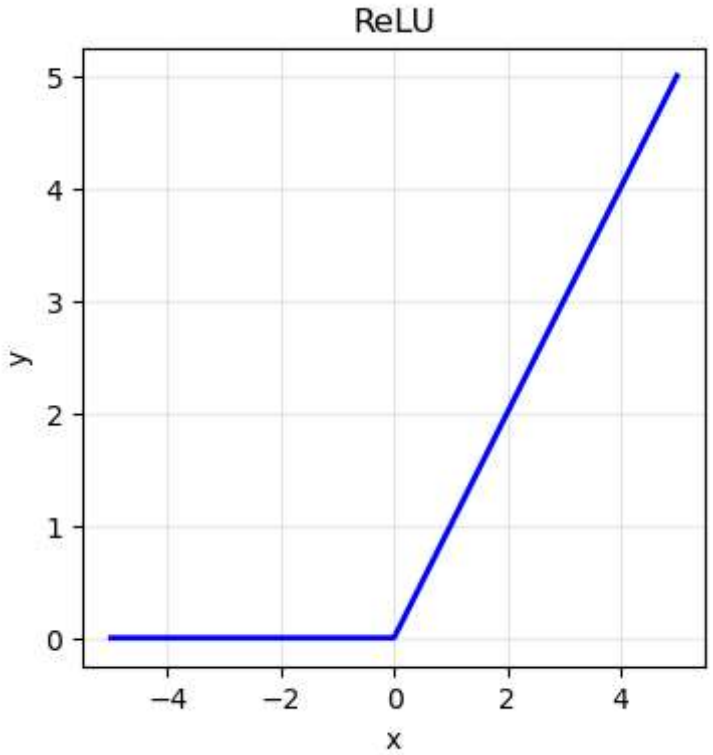
In [31]:
```
def bipolar(x):
    return np.where(x >= 0, 1, -1)

y_degrau = bipolar(x_valores)
plota_grafico(x_valores, y_degrau, 'Bipolar')
```



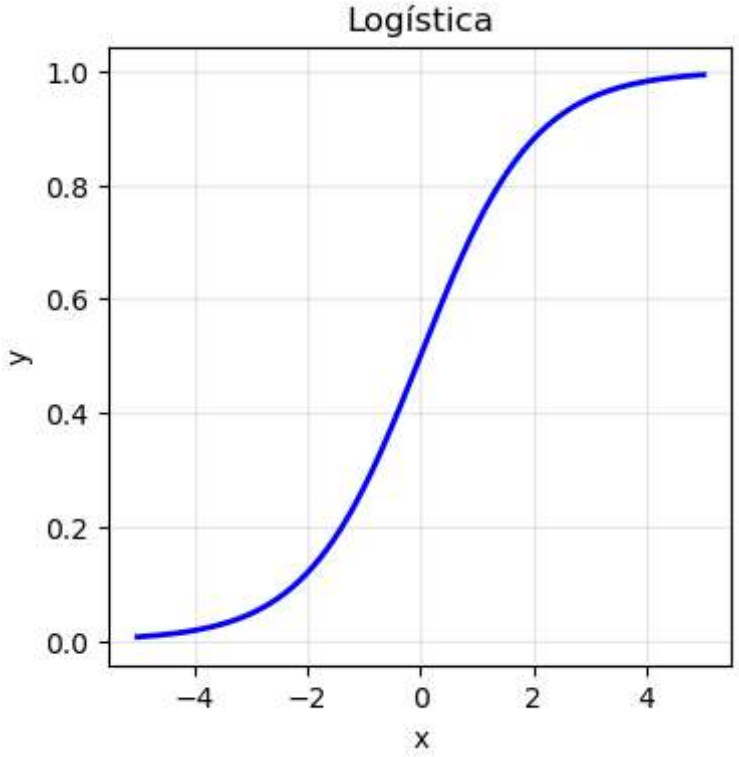
In [32]:
```
def rampa_simetrica(x):
    return np.where(x > 1, 1, np.where(x < -1, -1, x))
y_rampa_simetrica = rampa_simetrica(x_valores)
plota_grafico(x_valores, y_rampa_simetrica, "Rampa Simétrica")
```



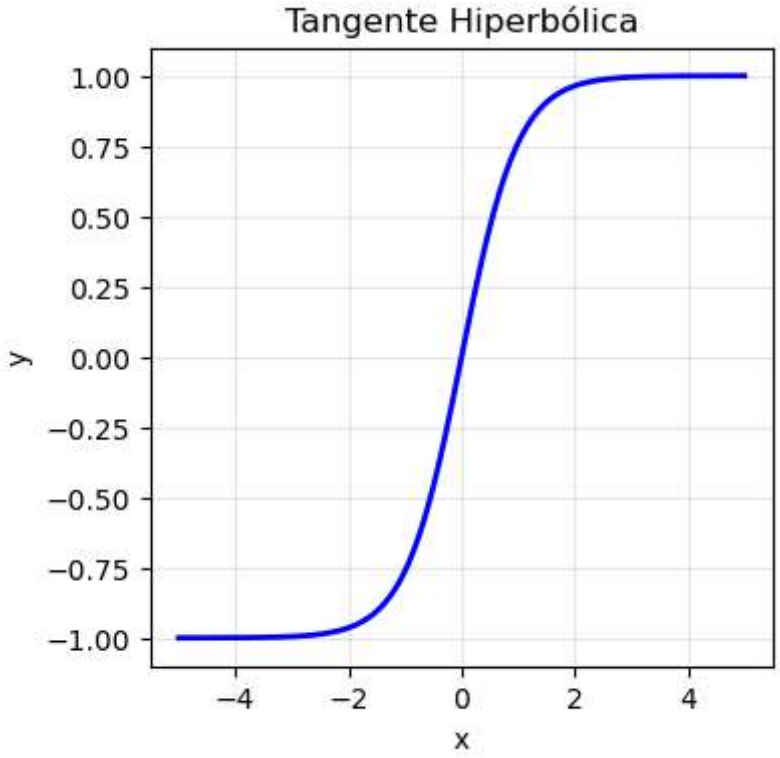
In [33]:
```
def relu(x):
    return np.where(x > 0, x, 0)
y_relu = relu(x_valores)
plota_grafico(x_valores, y_relu, "ReLU")
```



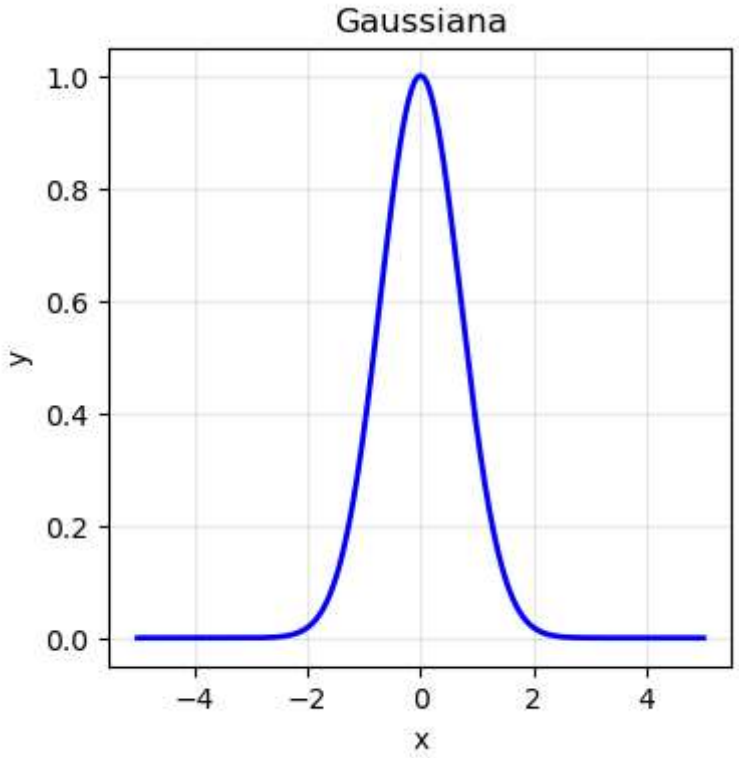
In [34]:
```
def logistica(x):
    return 1 / (1 + np.exp(-x))
y_sigmoide = logistica(x_valores)
plota_grafico(x_valores, y_sigmoide, 'Logística')
```



In [35]:
```
def tangente_hiperbolica(x):
    return np.tanh(x)
y_tanh = tangente_hiperbolica(x_valores)
plota_grafico(x_valores, y_tanh, 'Tangente Hiperbólica')
```



In [36]:
```
def gaussiana(x):
    return np.exp(-(x**2))
y_gaussiana = gaussiana(x_valores)
plota_grafico(x_valores, y_gaussiana, "Gaussiana")
```



In [37]:
```
def linear(x, a=1):
    return a * x
y_linear = linear(x_valores)
plota_grafico(x_valores, y_linear, 'Identidade')
```