REDES NEURAIS ARTIFICIAIS

INTRODUÇÃO

Zenilton K. G. Patrocínio Jr
zenilton@pucminas.br

Rede Neural Artificial (sem metáfora cognitiva)

(**Antes**) Função de predição:

$$f = Wx$$

Rede Neural Artificial (sem metáfora cognitiva)

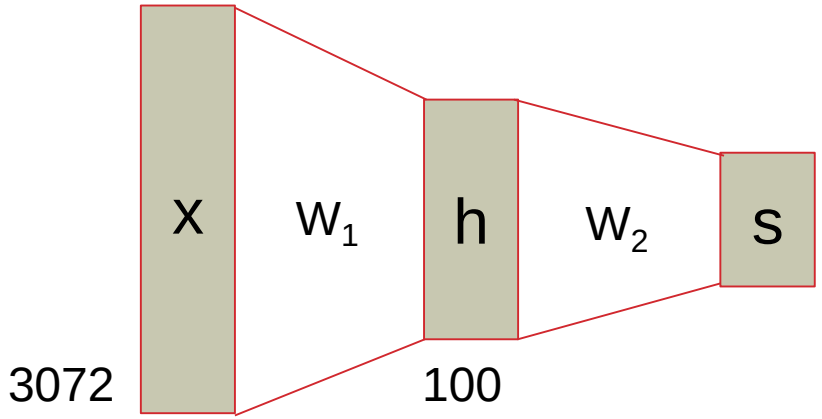
(**Antes**) Função de predição: $f = Wx$

(**Agora**) Rede neural de 2 camadas: $f = W_2 \max(0, W_1 x)$

Rede Neural Artificial (sem metáfora cognitiva)

(**Antes**) Função de predição: $f = Wx$

(**Agora**) Rede neural de 2 camadas: $f = W_2 \max(0, W_1 x)$



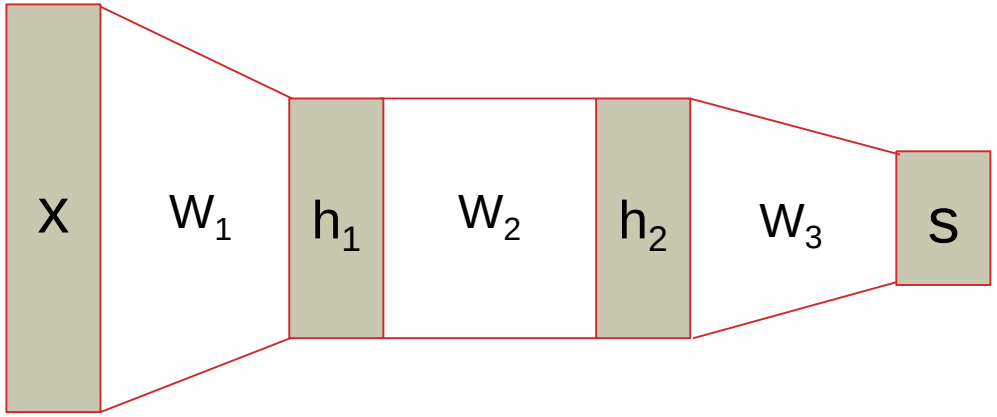
Rede Neural Artificial (sem metáfora cognitiva)

(**Antes**) Função de predição: $f = Wx$

(**Agora**) Rede neural de 2 camadas: $f = W_2 \max(0, W_1 x)$

ou Rede neural de 3 camadas:

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$



Código para Treino de Rede Neural de 2 Camadas

```
1  import numpy as np
2  from numpy.random import randn
3
4  N, D_in, H, D_out = 64, 1000, 100, 10
5  x, y = randn(N, D_in), randn(N, D_out)
6  w1, w2 = randn(D_in, H), randn(H, D_out)
7
8  for t in range(2000):
9    h = 1 / (1 + np.exp(-x.dot(w1)))
10   y_pred = h.dot(w2)
11   loss = np.square(y_pred - y).sum()
12   print(t, loss)
13
14   grad_y_pred = 2.0 * (y_pred - y)
15   grad_w2 = h.T.dot(grad_y_pred)
16   grad_h = grad_y_pred.dot(w2.T)
17   grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19   w1 -= 1e-4 * grad_w1
20   w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

```
1   import numpy as np
2   from numpy.random import randn
3
4   N, D_in, H, D_out = 64, 1000, 100, 10
5   x, y = randn(N, D_in), randn(N, D_out)
6   w1, w2 = randn(D_in, H), randn(H, D_out)
7
8   for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

```
1  import numpy as np
2  from numpy.random import randn
3
4  N, D_in, H, D_out = 64, 1000, 100, 10
5  x, y = randn(N, D_in), randn(N, D_out)
6  w1, w2 = randn(D_in, H), randn(H, D_out)
7
8  for t in range(2000):
9    h = 1 / (1 + np.exp(-x.dot(w1)))
10   y_pred = h.dot(w2)
11   loss = np.square(y_pred - y).sum()
12   print(t, loss)
13
14   grad_y_pred = 2.0 * (y_pred - y)
15   grad_w2 = h.T.dot(grad_y_pred)
16   grad_h = grad_y_pred.dot(w2.T)
17   grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19   w1 -= 1e-4 * grad_w1
20   w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

```
1  import numpy as np
2  from numpy.random import randn
3
4  N, D_in, H, D_out = 64, 1000, 100, 10
5  x, y = randn(N, D_in), randn(N, D_out)
6  w1, w2 = randn(D_in, H), randn(H, D_out)
7
8  for t in range(2000):
9    h = 1 / (1 + np.exp(-x.dot(w1)))
10   y_pred = h.dot(w2)
11   loss = np.square(y_pred - y).sum()
12   print(t, loss)
13
14   grad_y_pred = 2.0 * (y_pred - y)
15   grad_w2 = h.T.dot(grad_y_pred)
16   grad_h = grad_y_pred.dot(w2.T)
17   grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19   w1 -= 1e-4 * grad_w1
20   w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

```
1  import numpy as np
2  from numpy.random import randn
3
4  N, D_in, H, D_out = 64, 1000, 100, 10
5  x, y = randn(N, D_in), randn(N, D_out)
6  w1, w2 = randn(D_in, H), randn(H, D_out)
7
8  for t in range(2000):
9    h = 1 / (1 + np.exp(-x.dot(w1)))
10   y_pred = h.dot(w2)
11   loss = np.square(y_pred - y).sum()
12   print(t, loss)
13
14   grad_y_pred = 2.0 * (y_pred - y)
15   grad_w2 = h.T.dot(grad_y_pred)
16   grad_h = grad_y_pred.dot(w2.T)
17   grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19   w1 -= 1e-4 * grad_w1
20   w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

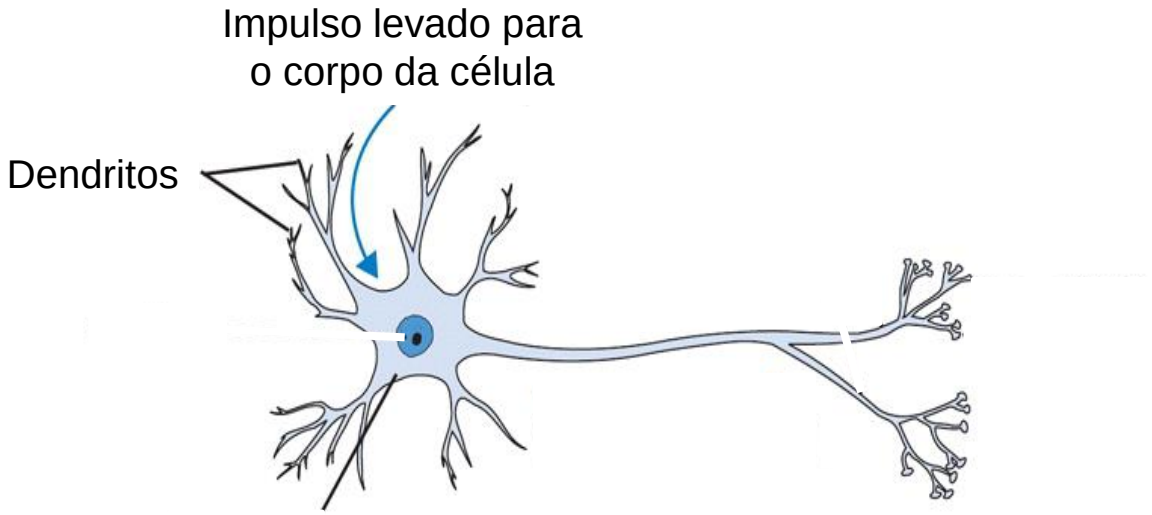
```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```

Código para Treino de Rede Neural de 2 Camadas

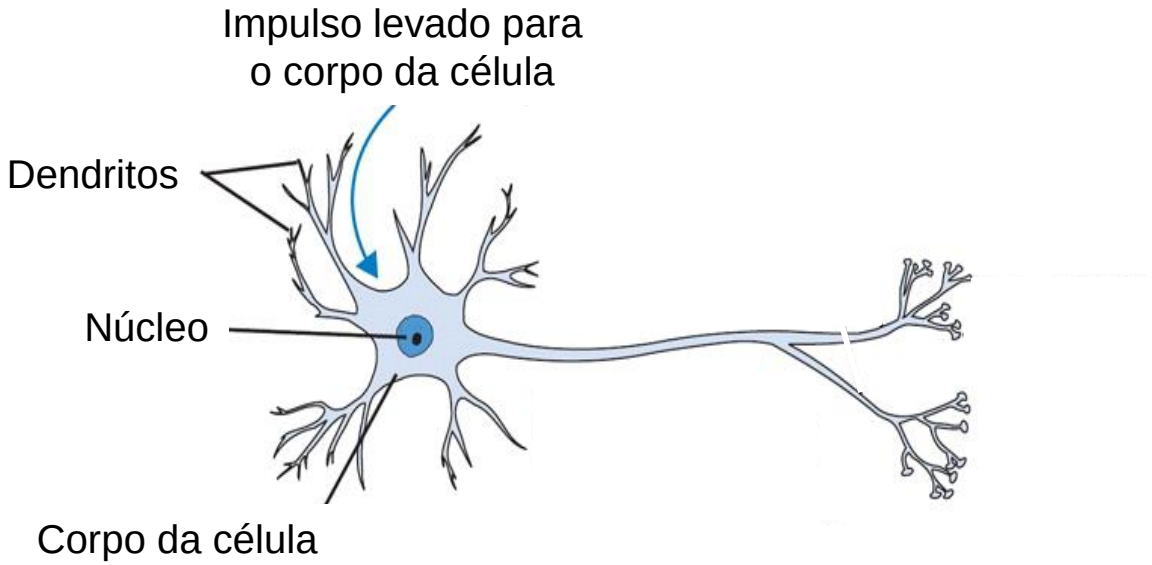
```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```

Apenas ~20 linhas!

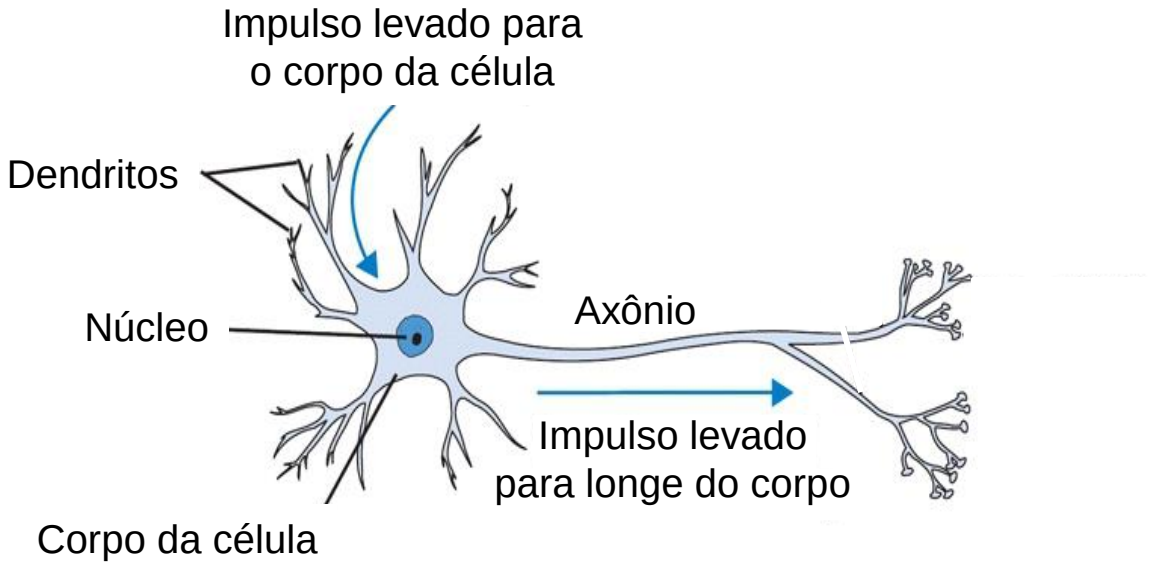
Inspiração Biológica



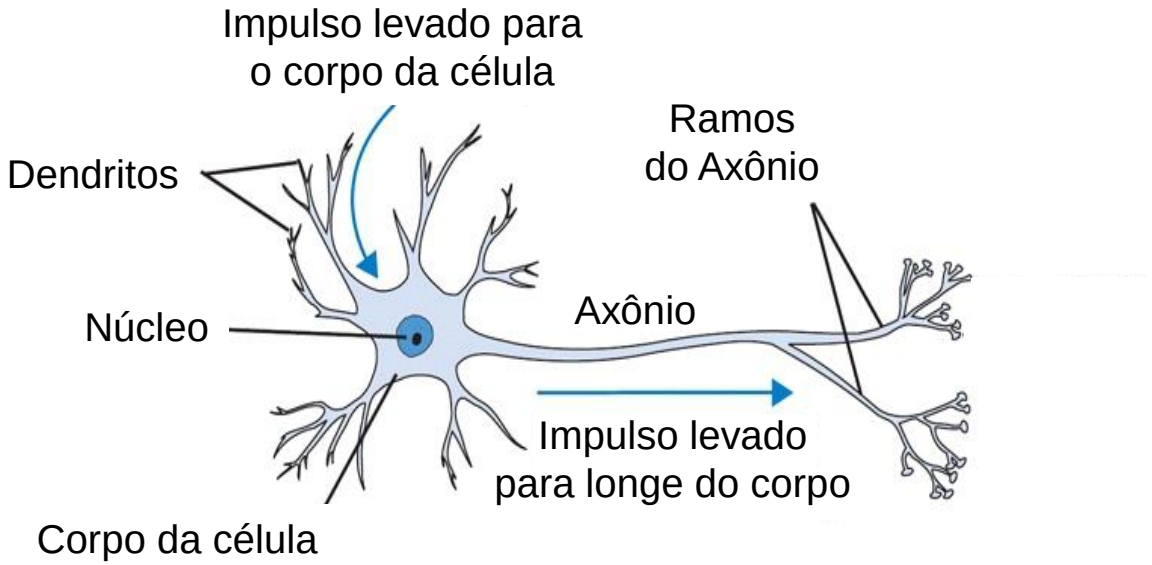
Inspiração Biológica



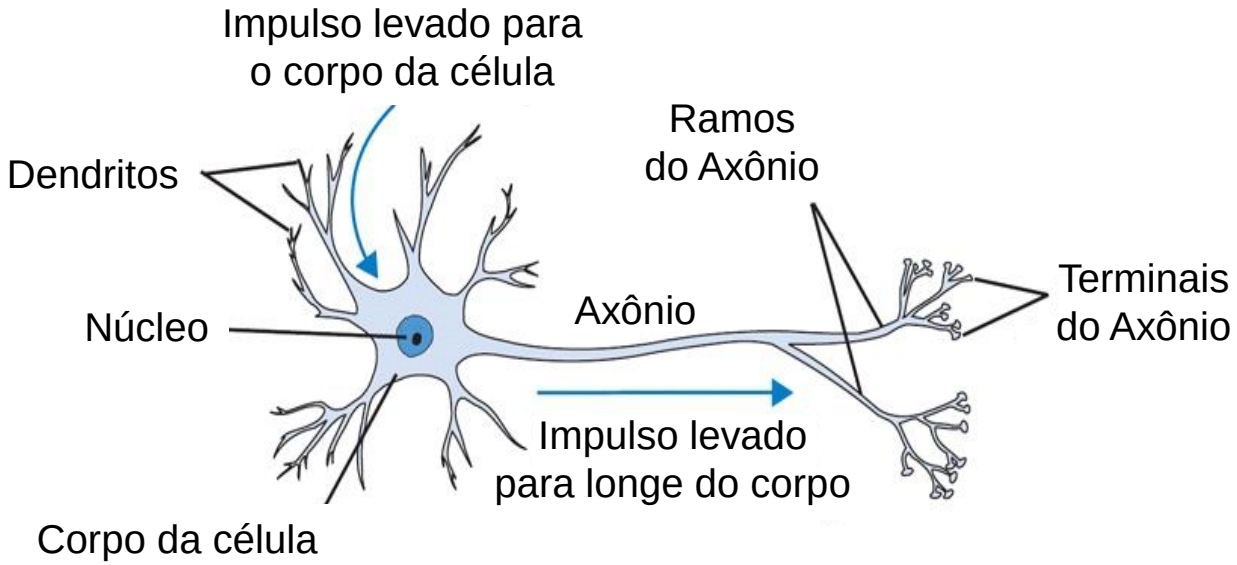
Inspiração Biológica



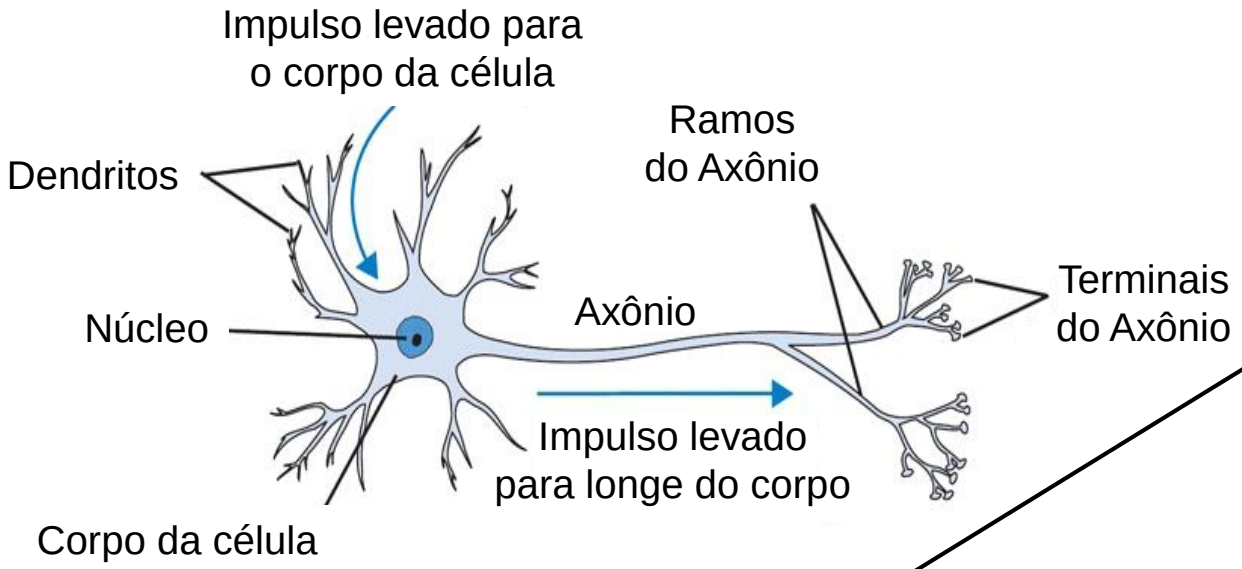
Inspiração Biológica

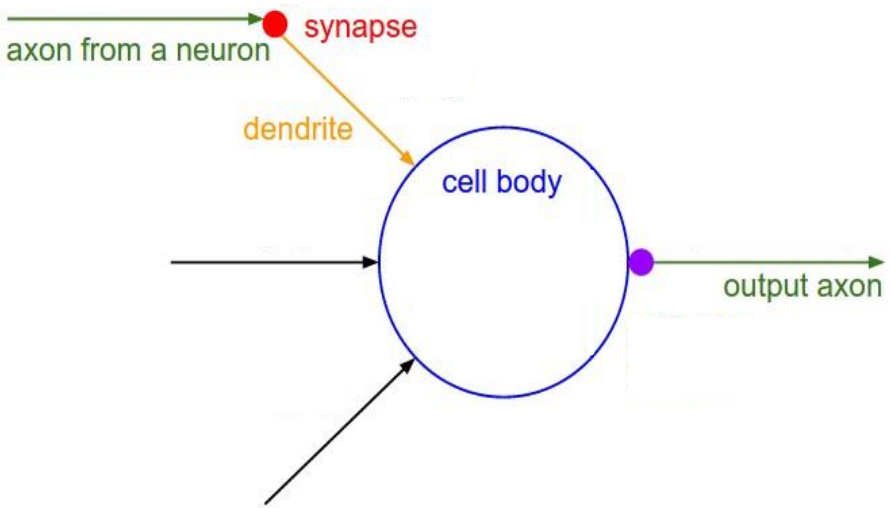


Inspiração Biológica

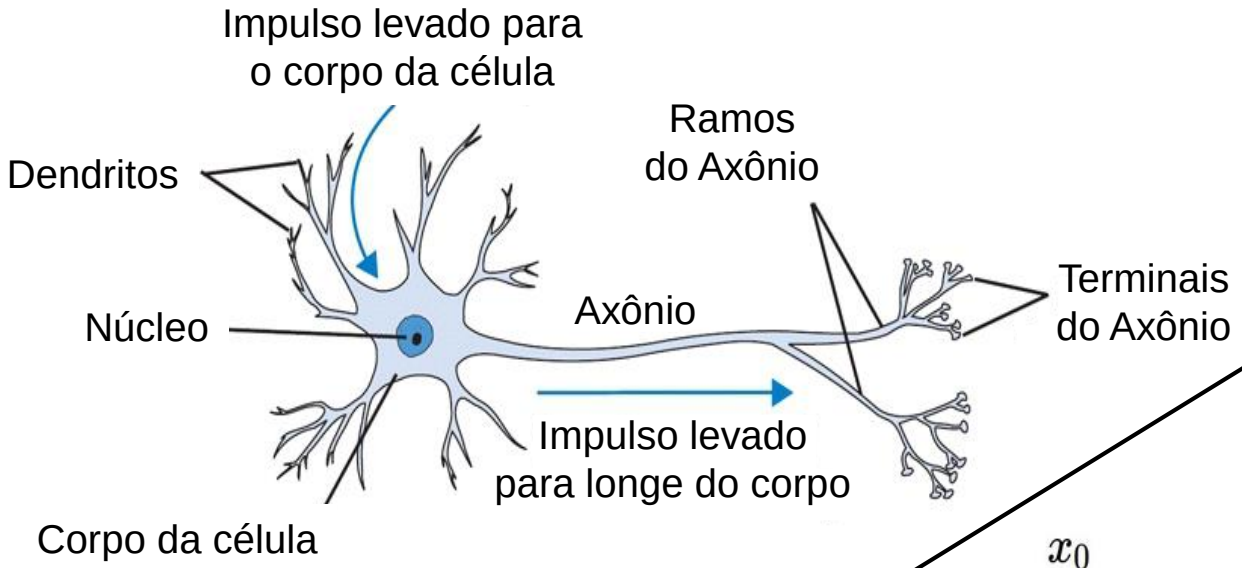


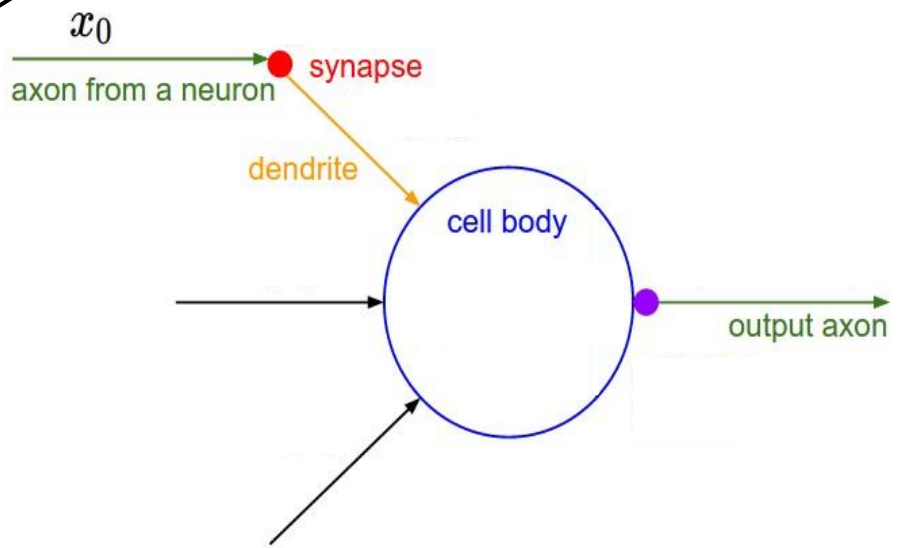
Inspiração Biológica



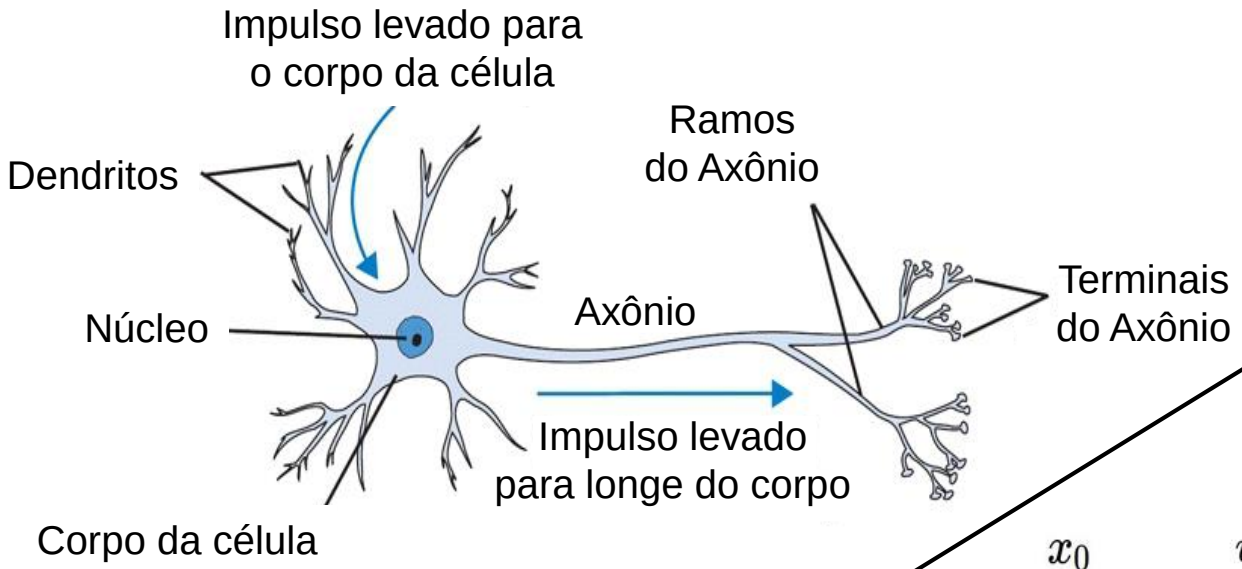


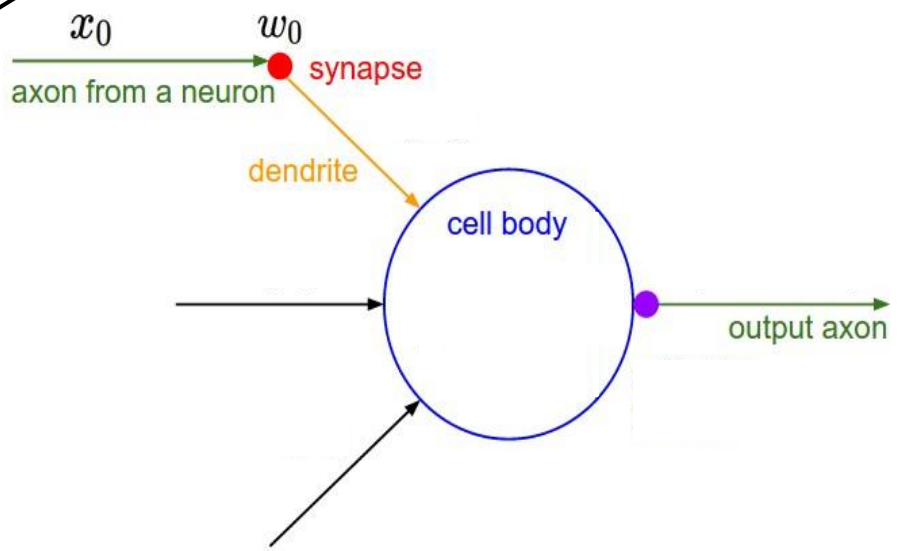
Inspiração Biológica



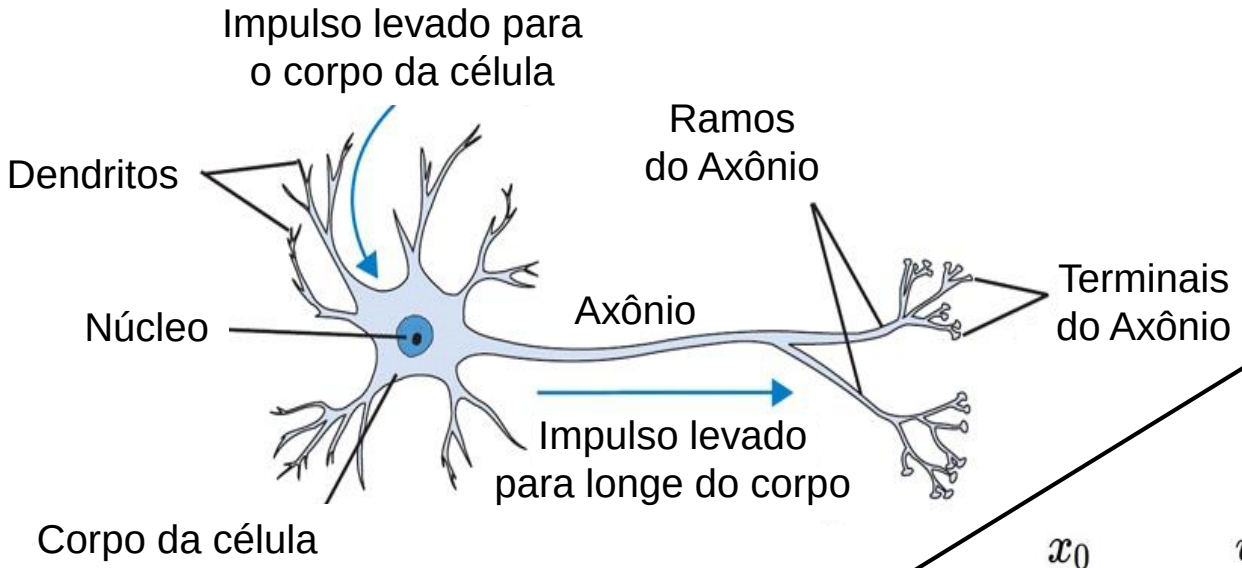


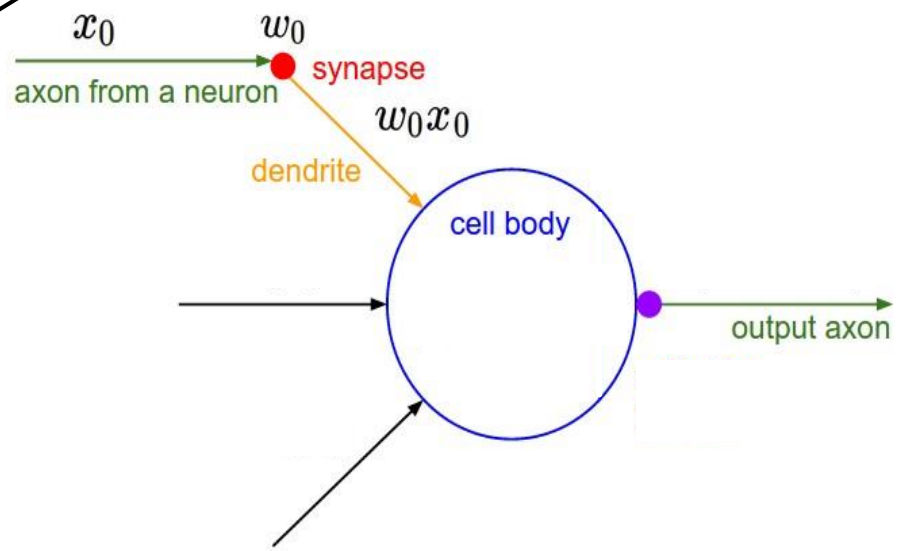
Inspiração Biológica



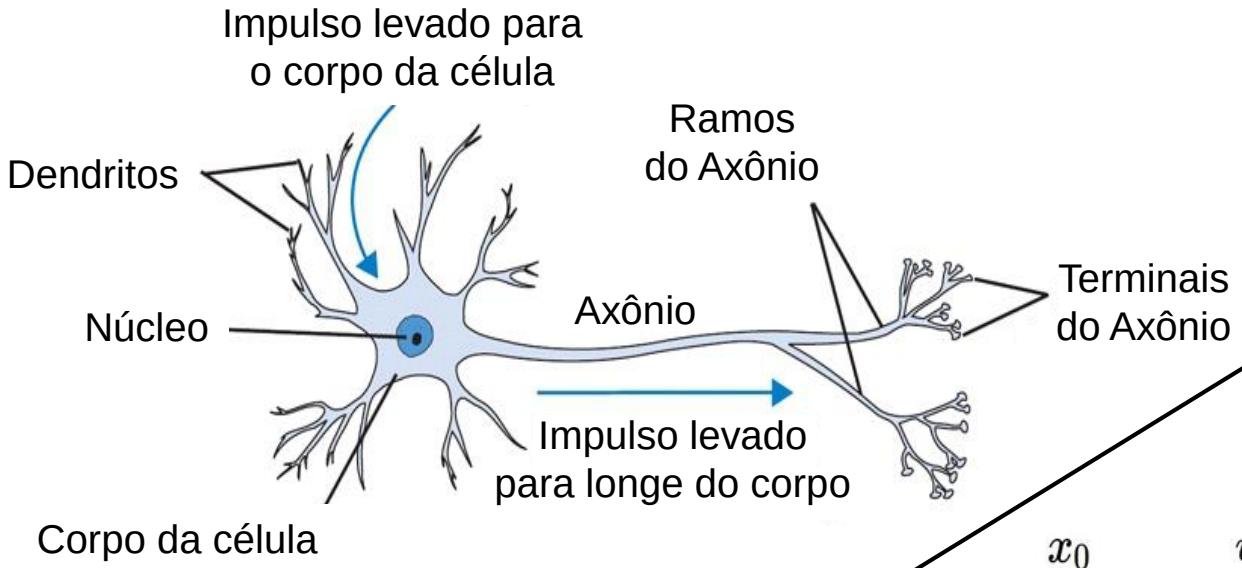


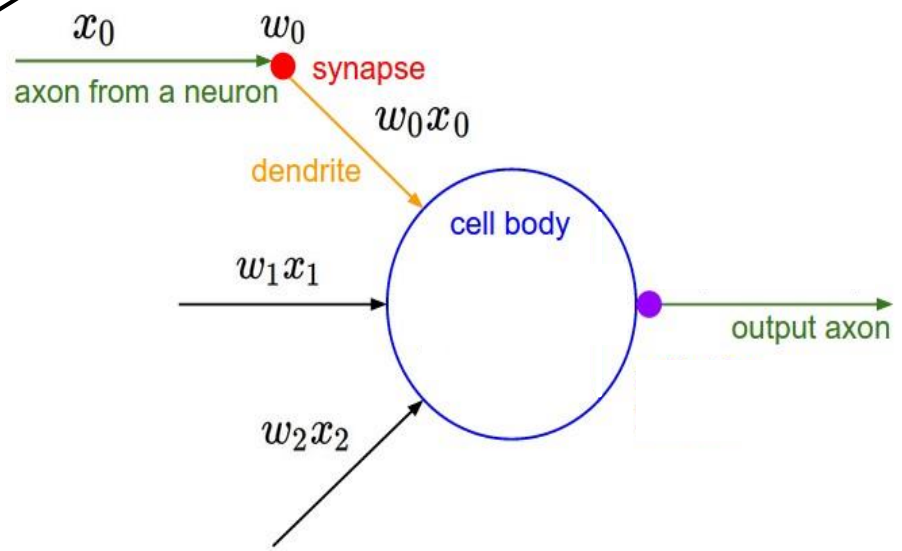
Inspiração Biológica



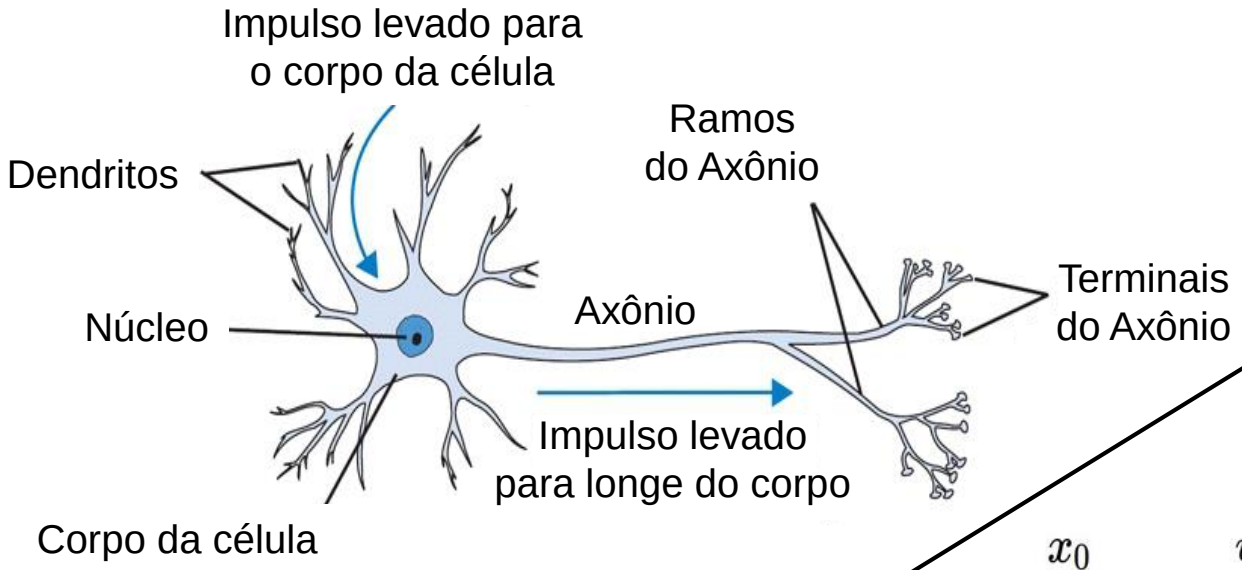


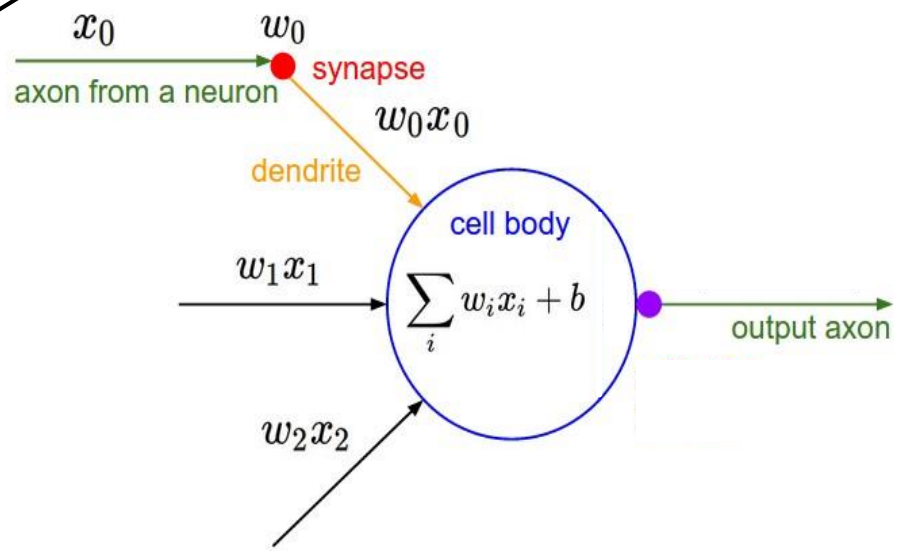
Inspiração Biológica



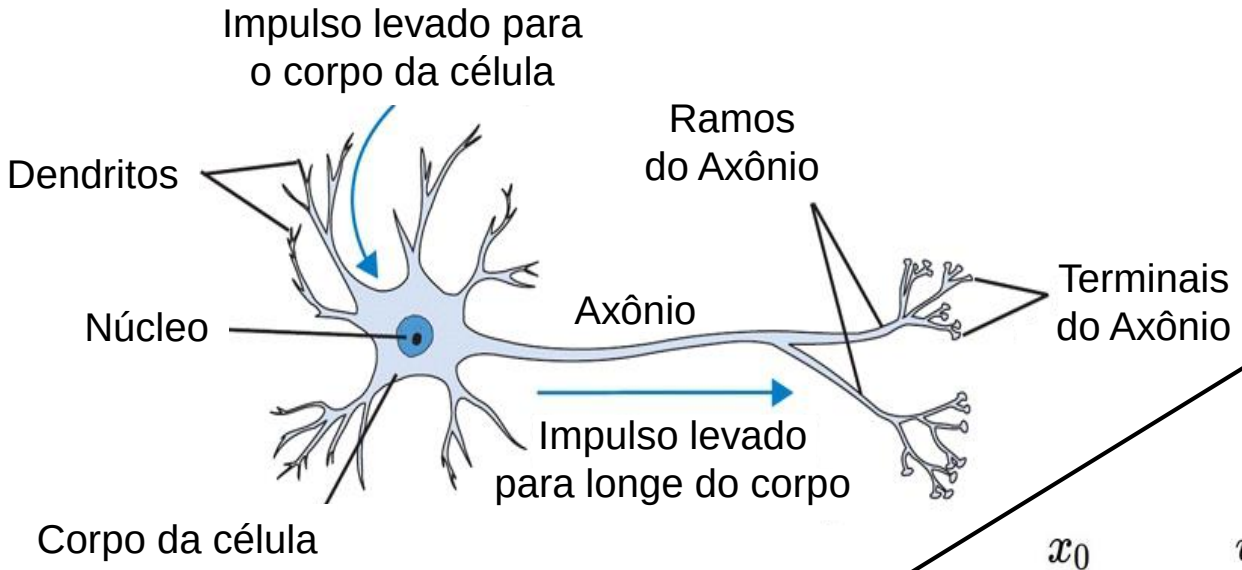


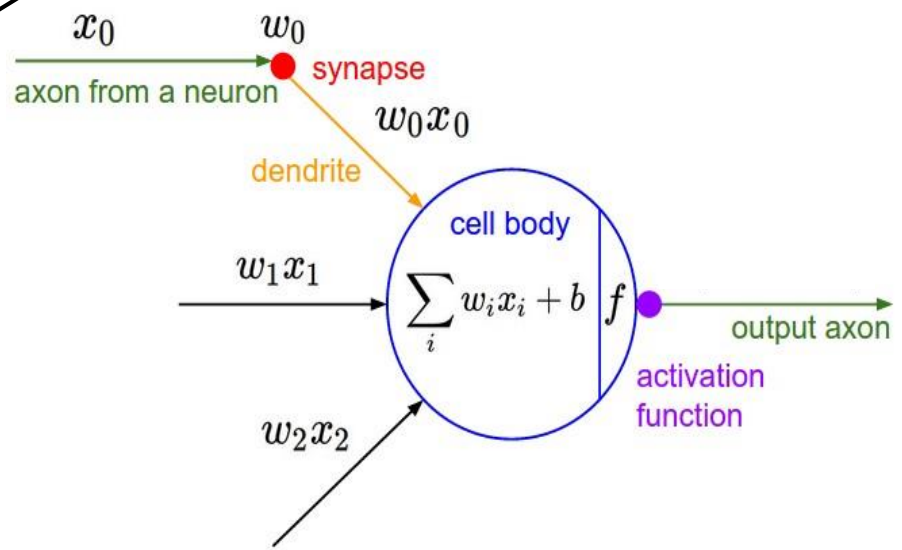
Inspiração Biológica



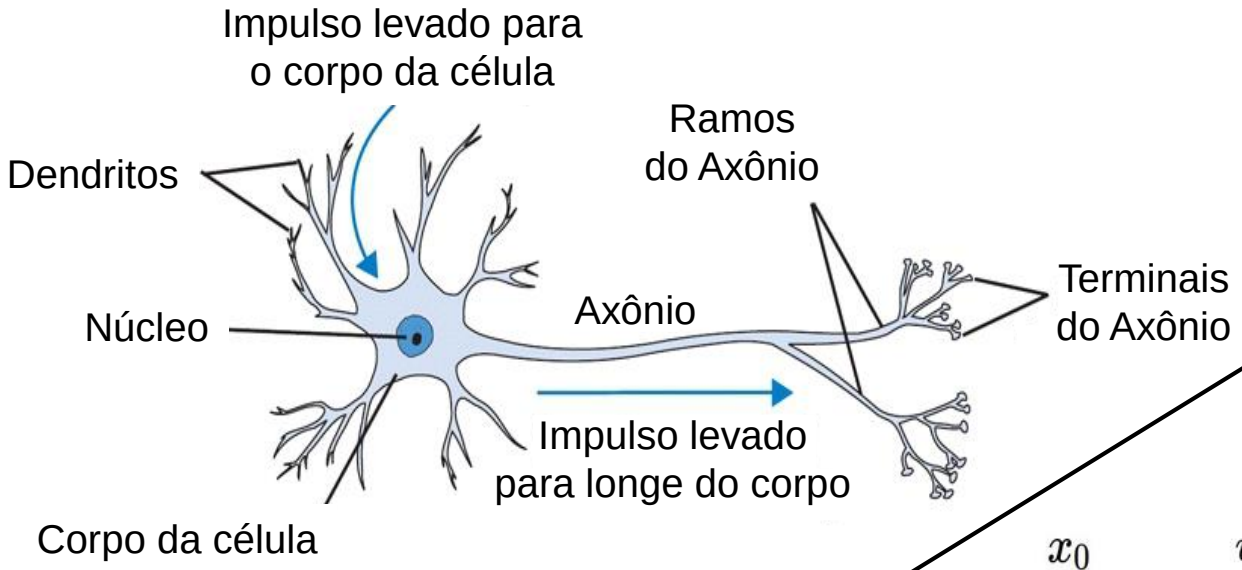


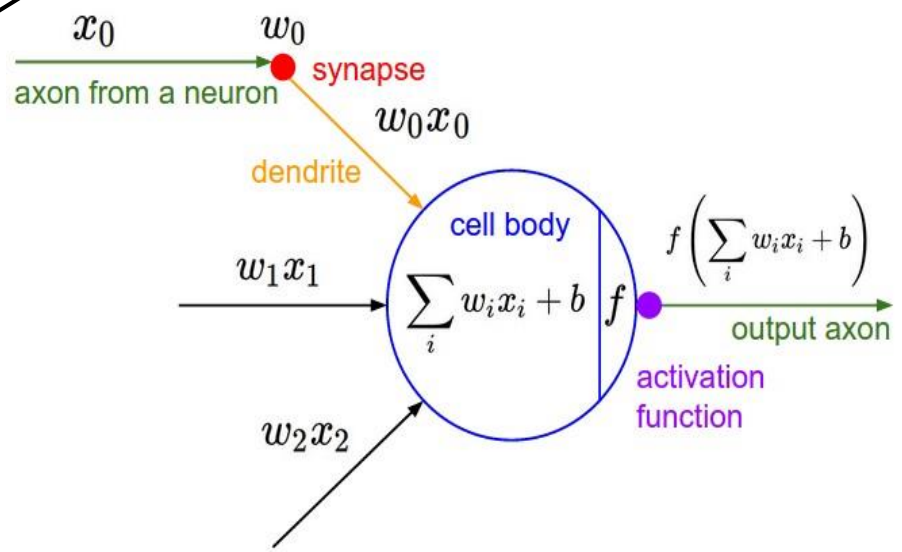
Inspiração Biológica



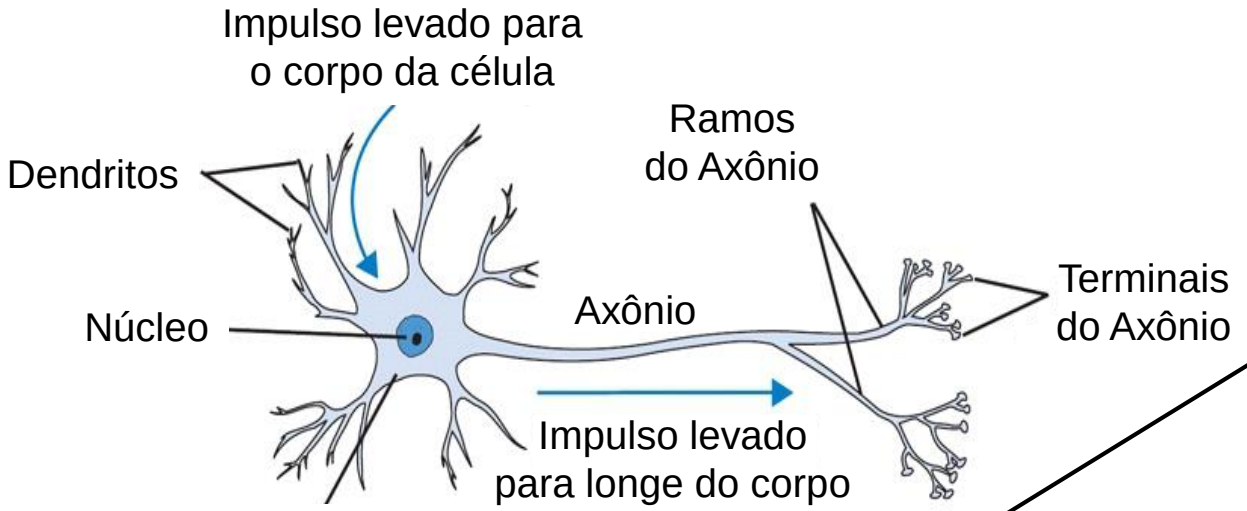


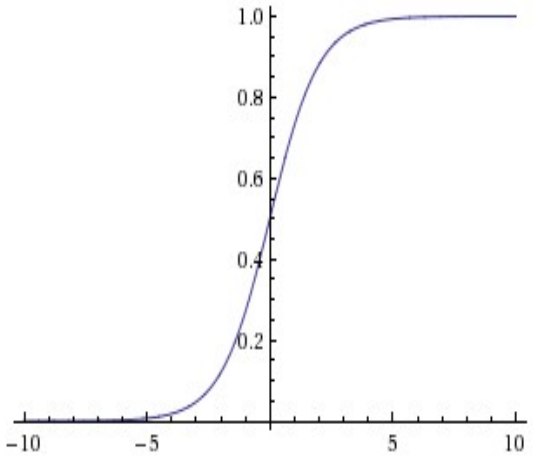
Inspiração Biológica





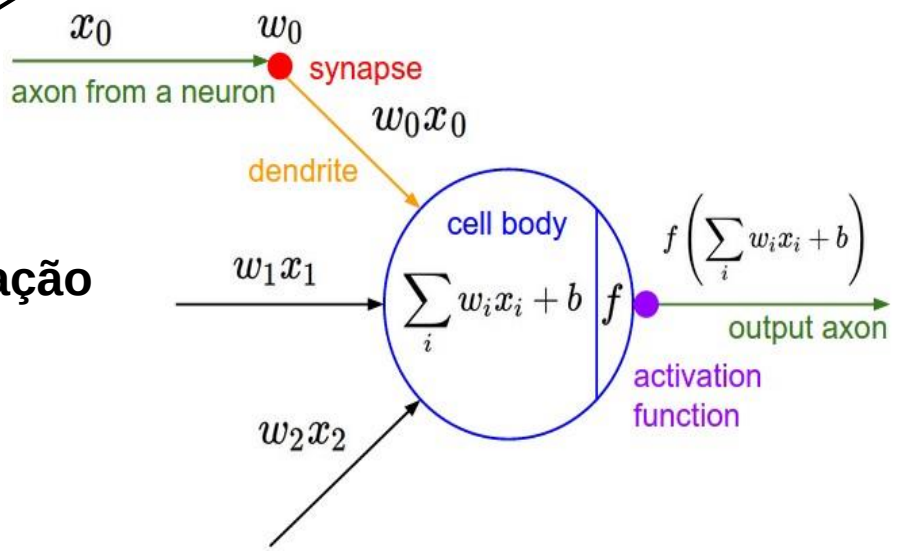
Inspiração Biológica





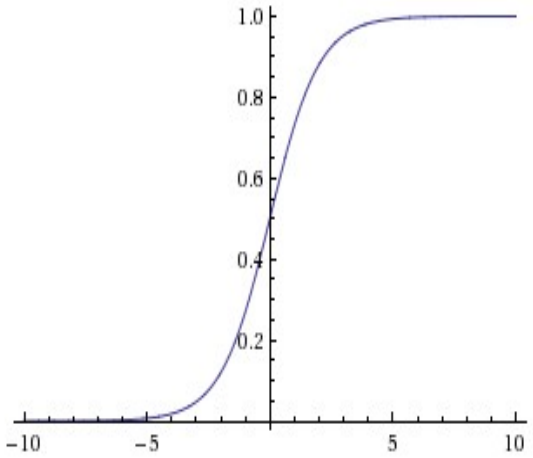
Função de ativação sigmoide

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



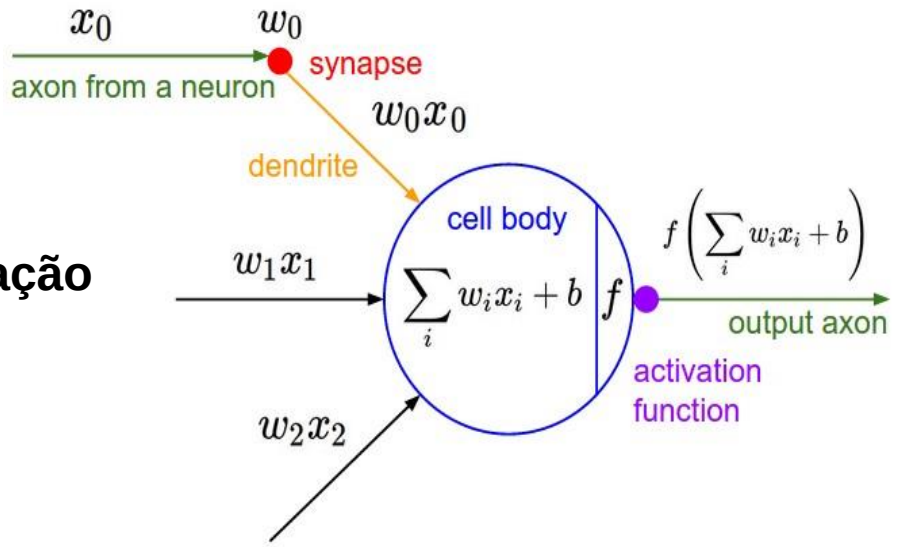
Inspiração Biológica

```python
class Neuron:
  # ...
  def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
    return firing_rate
```



Função de ativação sigmoide

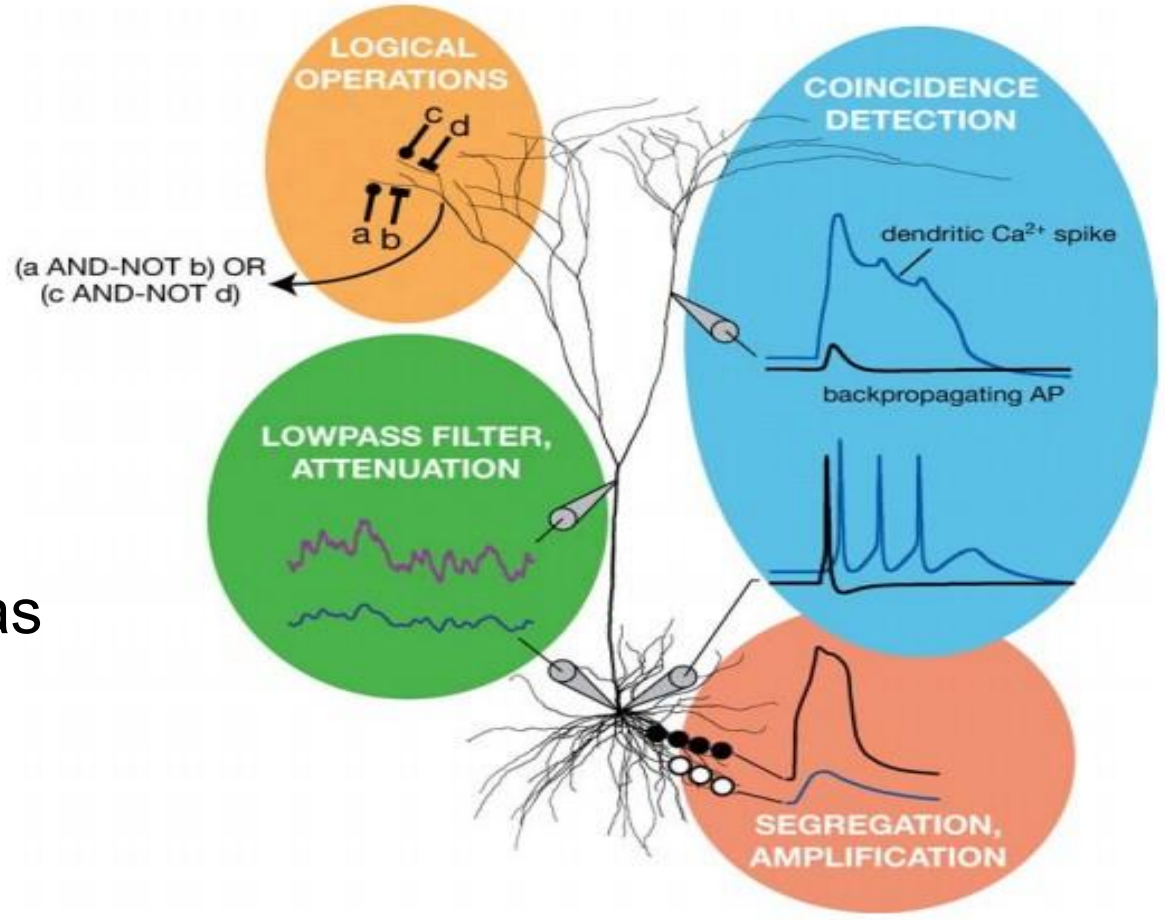
$$\frac{1}{1+e^{-x}}$$



Cuidados com Analogias

Neurônio biológicos:

- Vários tipos diferentes
- Dendritos pode realizar computações não-lineares
- Sinapses não representam apenas um “simples peso” mas sim um complexo sistema dinâmico não-linear

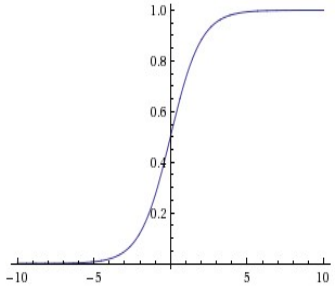


London, M., & Häusser, M. Dendritic computation. *Annual Review of Neuroscience*, *28*: 503-532, (2005).

Algumas Funções de Ativação

Sigmoid

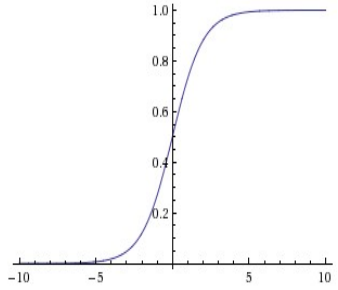
$$\sigma(x) = 1/(1 + e^{-x})$$



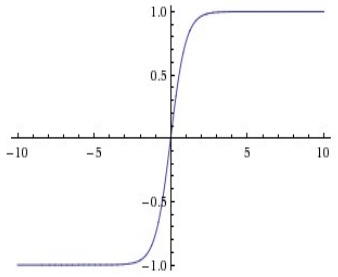
Algumas Funções de Ativação

Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$



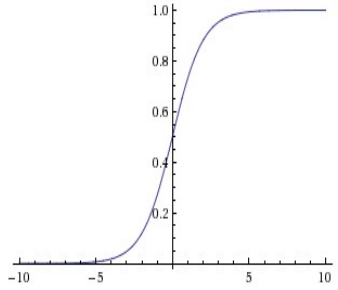
Tanh tanh(x)



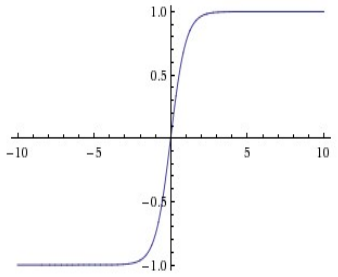
Algumas Funções de Ativação

Sigmoid

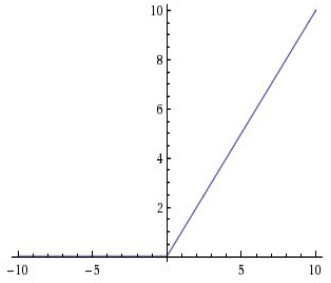
$$\sigma(x) = 1/(1 + e^{-x})$$



Tanh tanh(x)



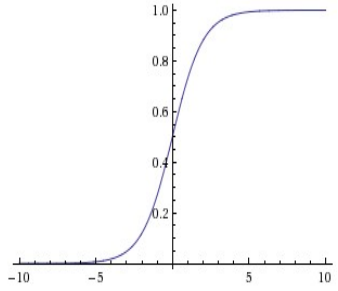
ReLU max(0,x)



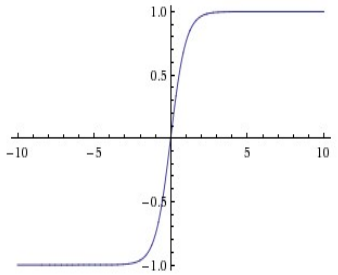
Algumas Funções de Ativação

Sigmoid

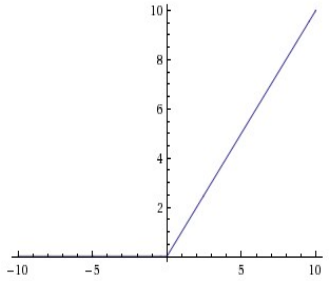
$$\sigma(x) = 1/(1 + e^{-x})$$



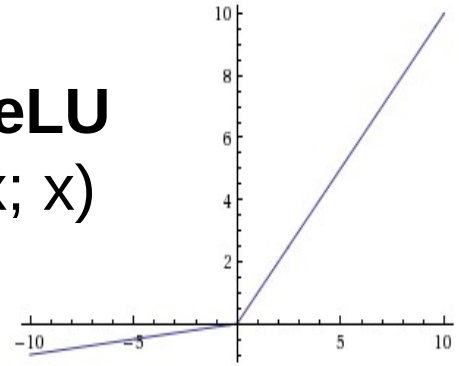
Tanh tanh(x)



ReLU max(0,x)



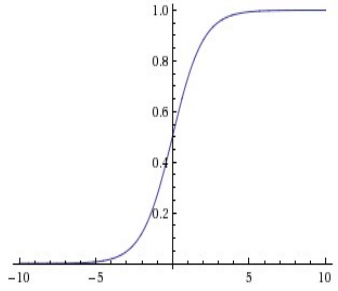
Leaky ReLU
max(0,1x; x)



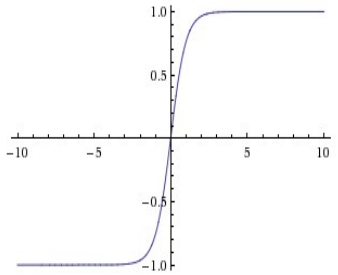
Algumas Funções de Ativação

Sigmoid

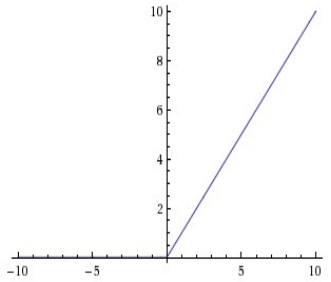
$$\sigma(x) = 1/(1 + e^{-x})$$



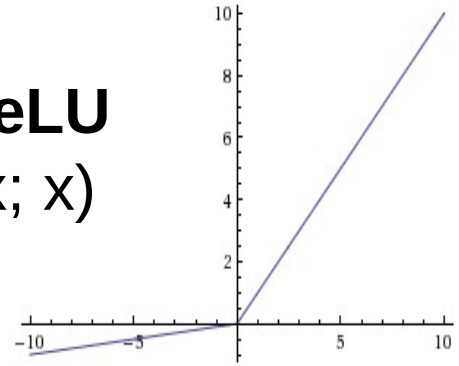
Tanh tanh(x)



ReLU max(0,x)

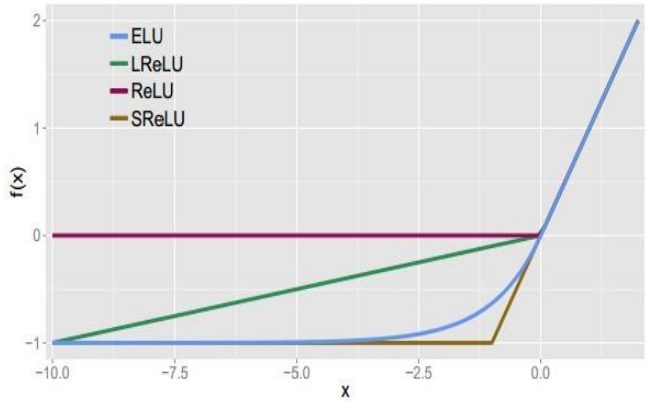


Leaky ReLU
max(0,1x; x)



ELU

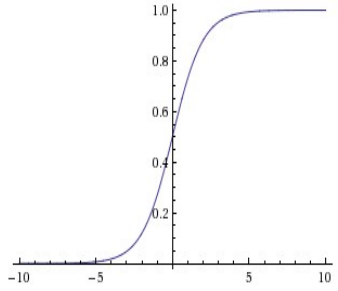
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$



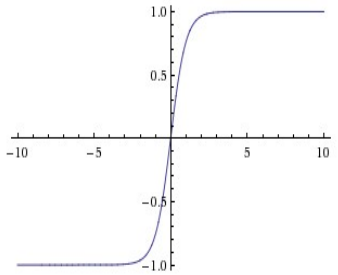
Algumas Funções de Ativação

Sigmoid

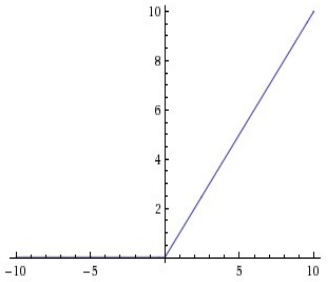
$$\sigma(x) = 1/(1 + e^{-x})$$



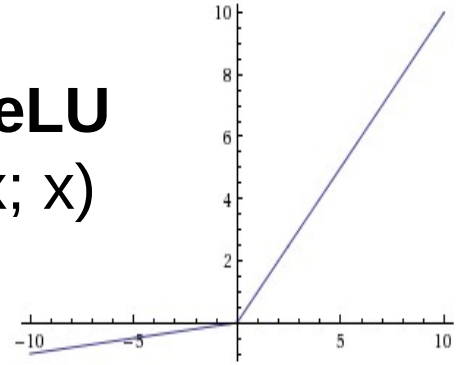
Tanh tanh(x)



ReLU max(0,x)

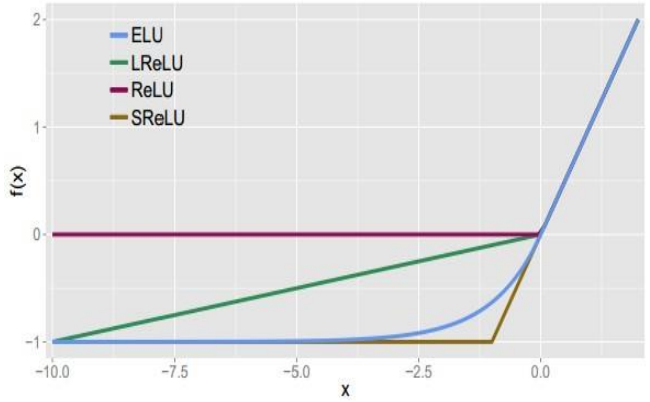


Leaky ReLU
max(0,1x; x)



ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}$$



Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$