INF721

2023/2

Aprendizado em Redes Neurais Profundas

A5: Multilayer Perceptron (MLP)

Logística

Avisos

- Teste T1: Regressão Logística será corrigido até o final de semana

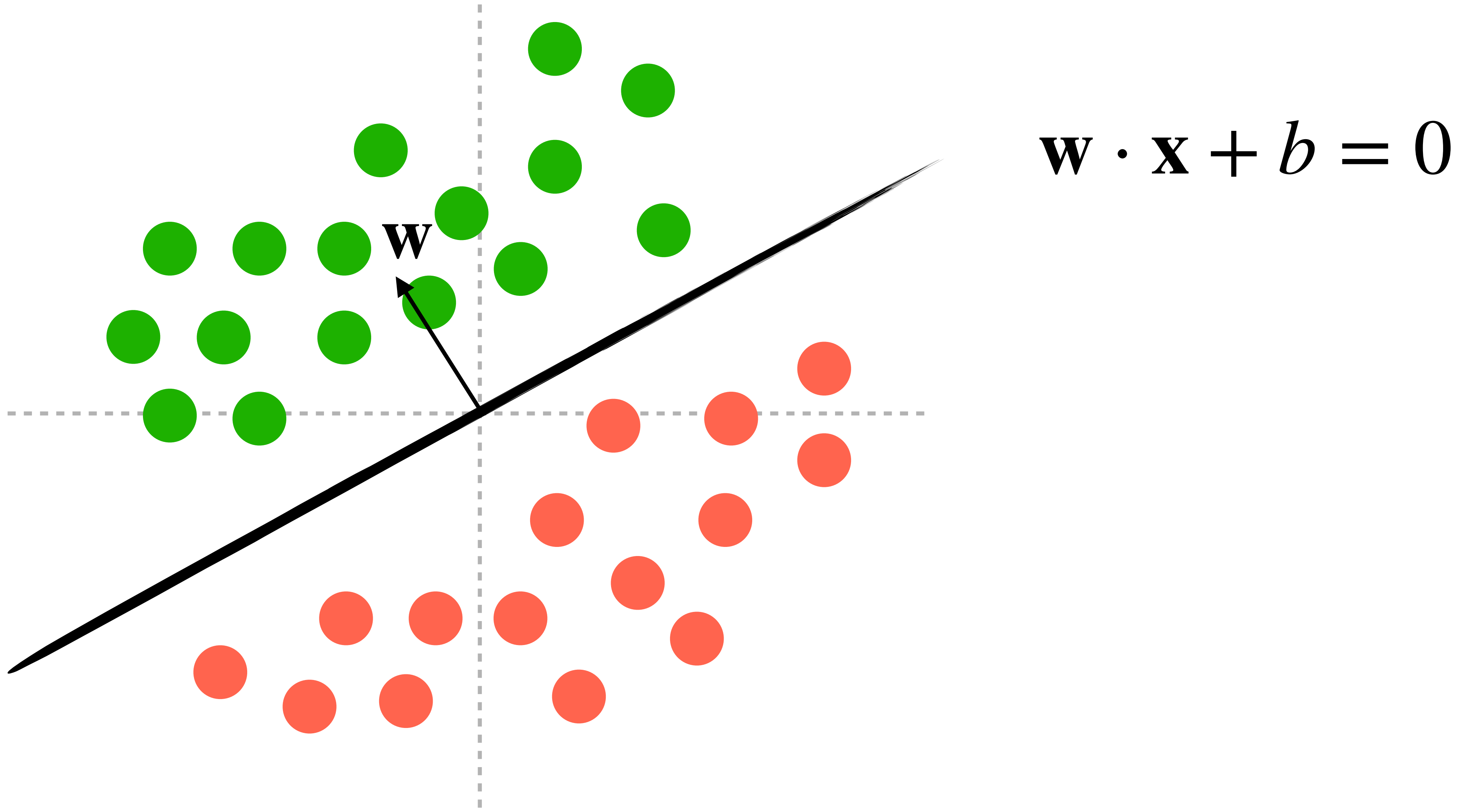
Última aula

- Regressão Logística em Numpy
- Vetorização
- Gradientes da Regressão Logística

Plano de Aula

- Problemas linearmente separáveis
- Perceptron
- Problemas linearmente não-sepáveis
- Multilayer Perceptron (MLP)
  - Intuição e formalização
  - Propagação das entradas (forward pass)

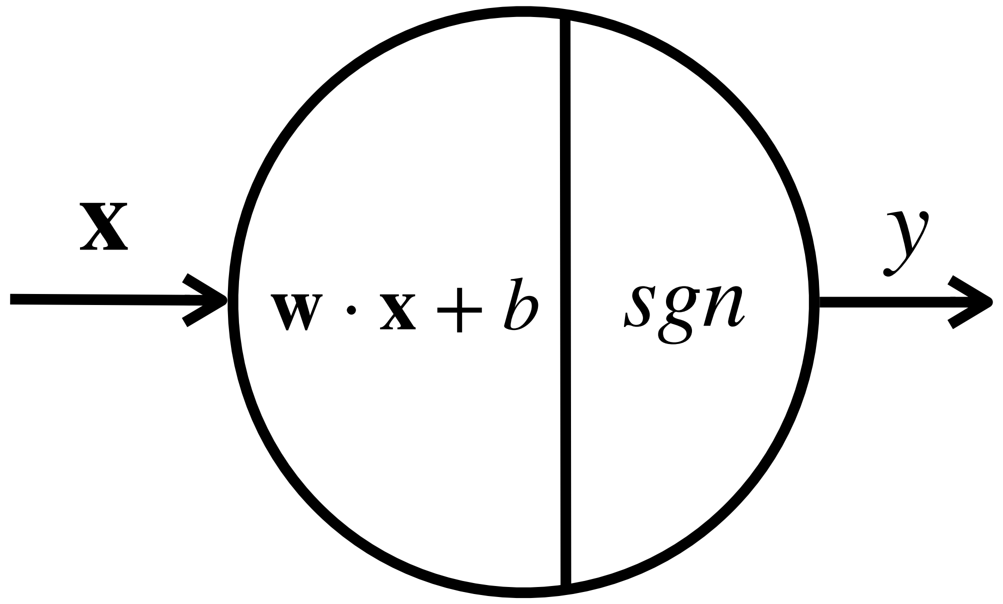
Problemas Linearmente Separáveis



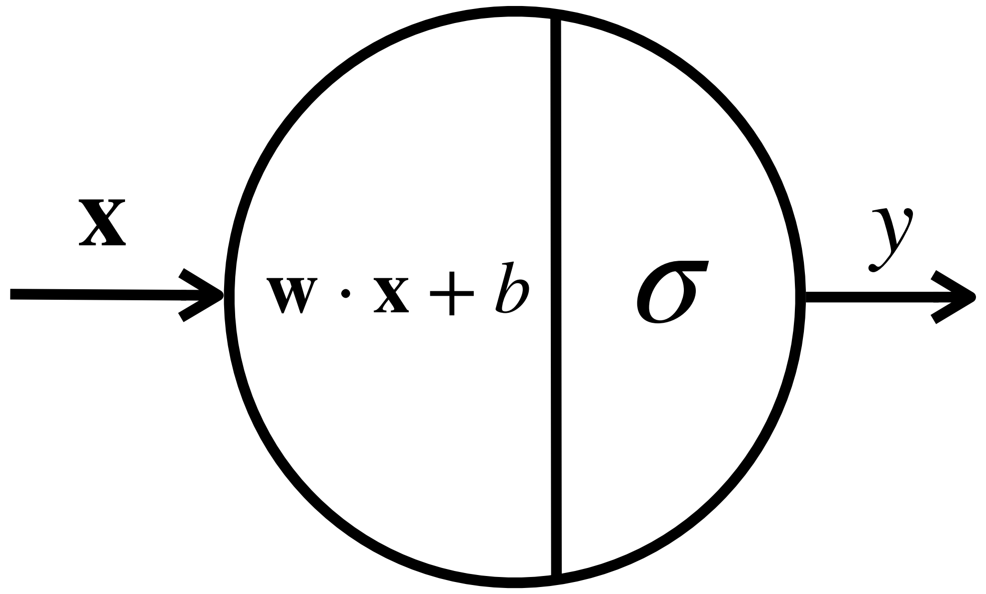
UFV

Problemas Linearmente Separáveis

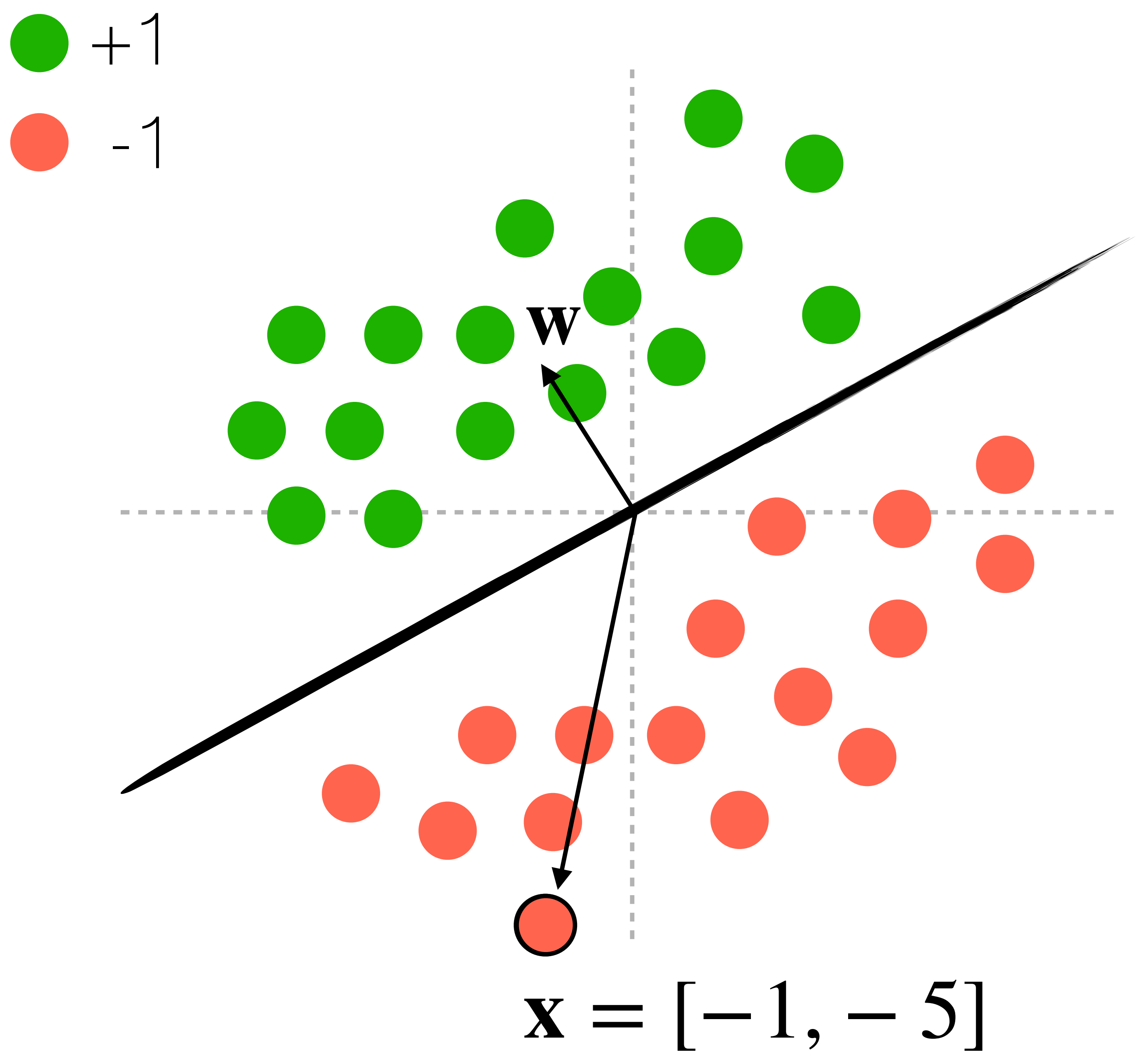
Perceptron



Regressão Logística



Ambos aprendem apenas fronteiras de decisão lineares!



$$sgn(z) = \begin{cases} 1, & z > 0 \\ -1, & z < 0 \end{cases}$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$h(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\mathbf{w} = [-2,1]$$

$$b = 0$$

$$h(\mathbf{x}) = sgn(-2 \cdot -1 + 1 \cdot (-5) + 0)$$

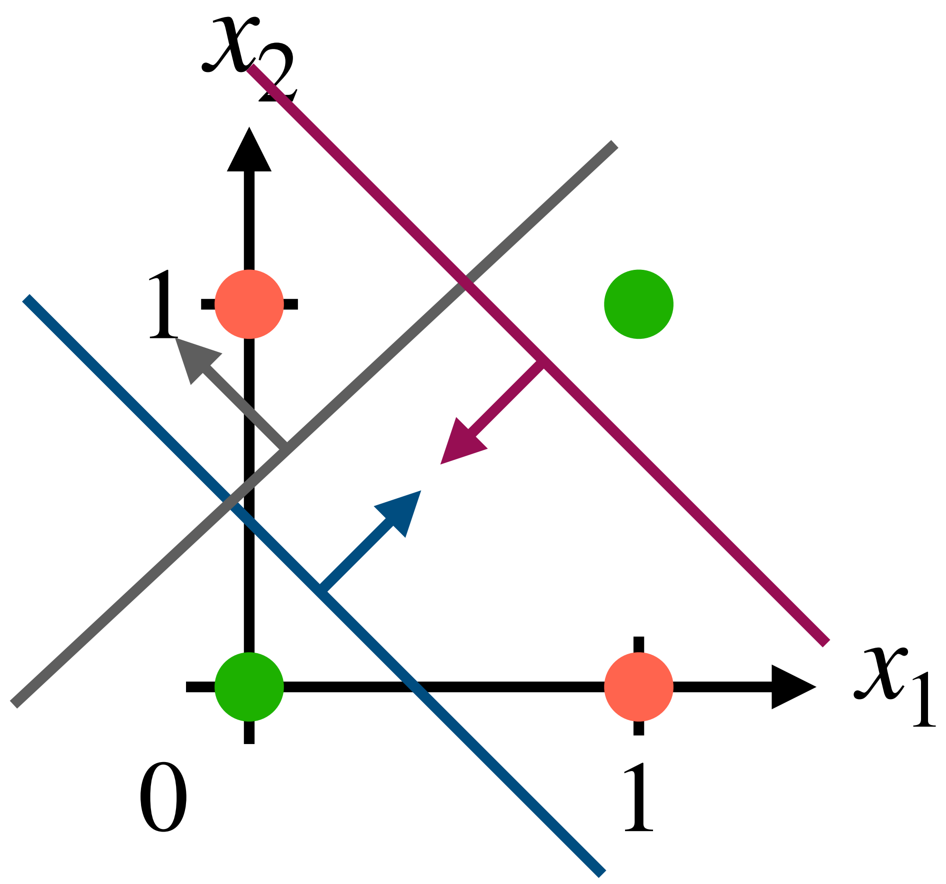
$$h(\mathbf{x}) = sgn(-3)$$

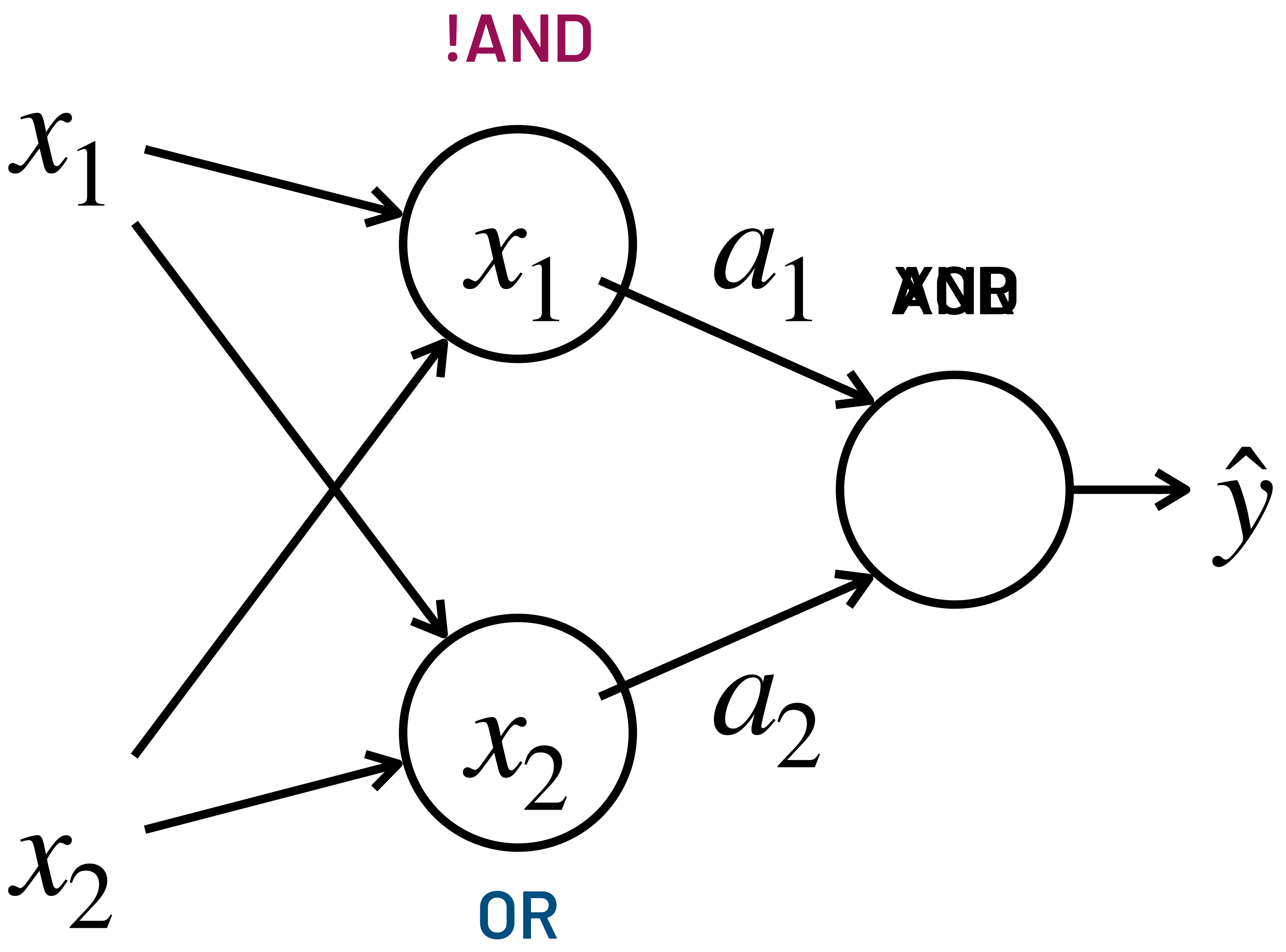
$$h(\mathbf{x}) = -1$$

Problemas Não-linearmente Separáveis

$$f(x_1, x_2) = x_1 \text{ XOR } x_2$$

$x_2 \backslash x_1$	0	1
0	0	1
1	1	0

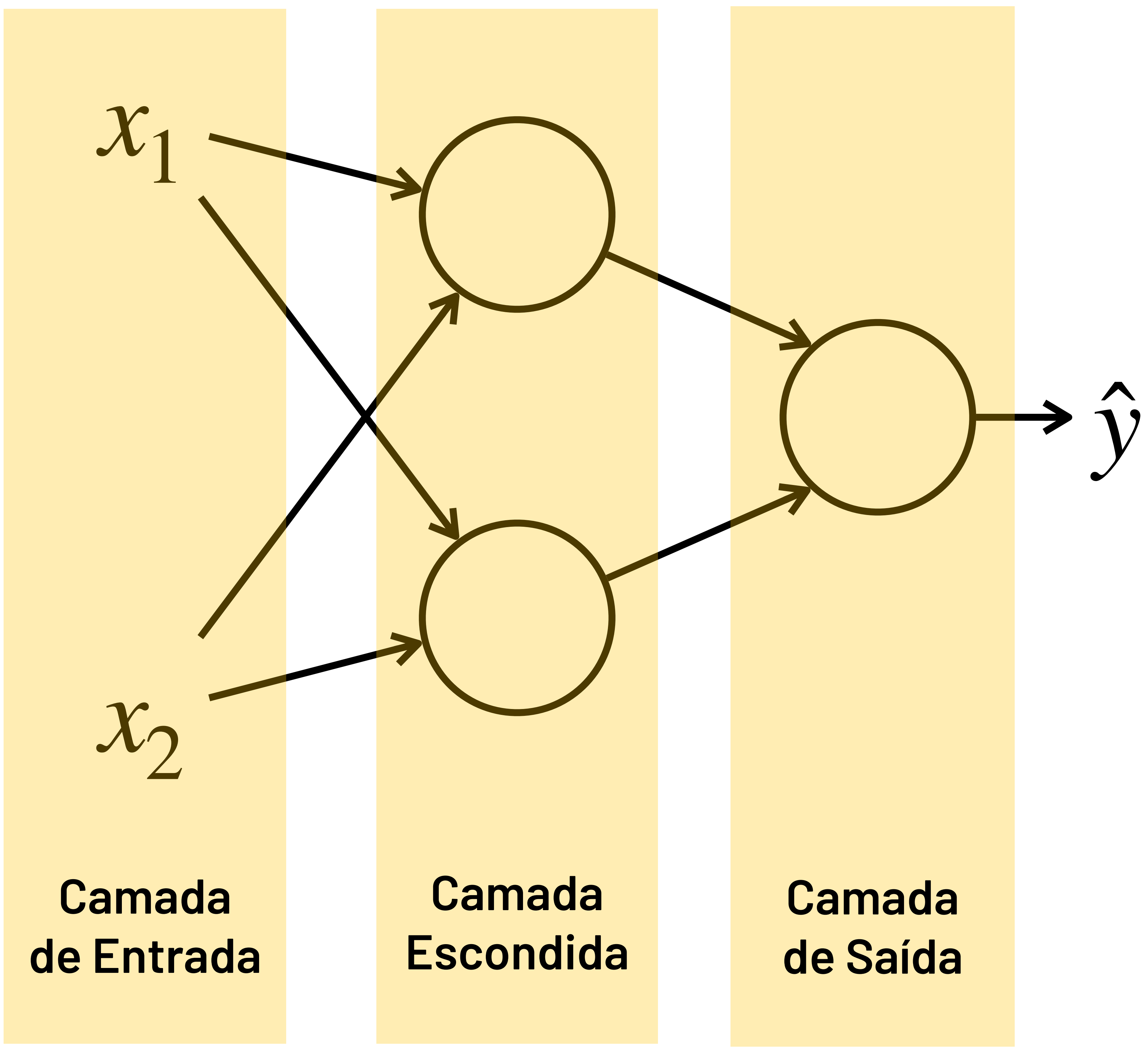




RNAs aprendem representações intermediárias $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ dos dados de entrada $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, chamadas **representações latentes**, que podem tornar um problema não-linearmente separável em linearmente separável!

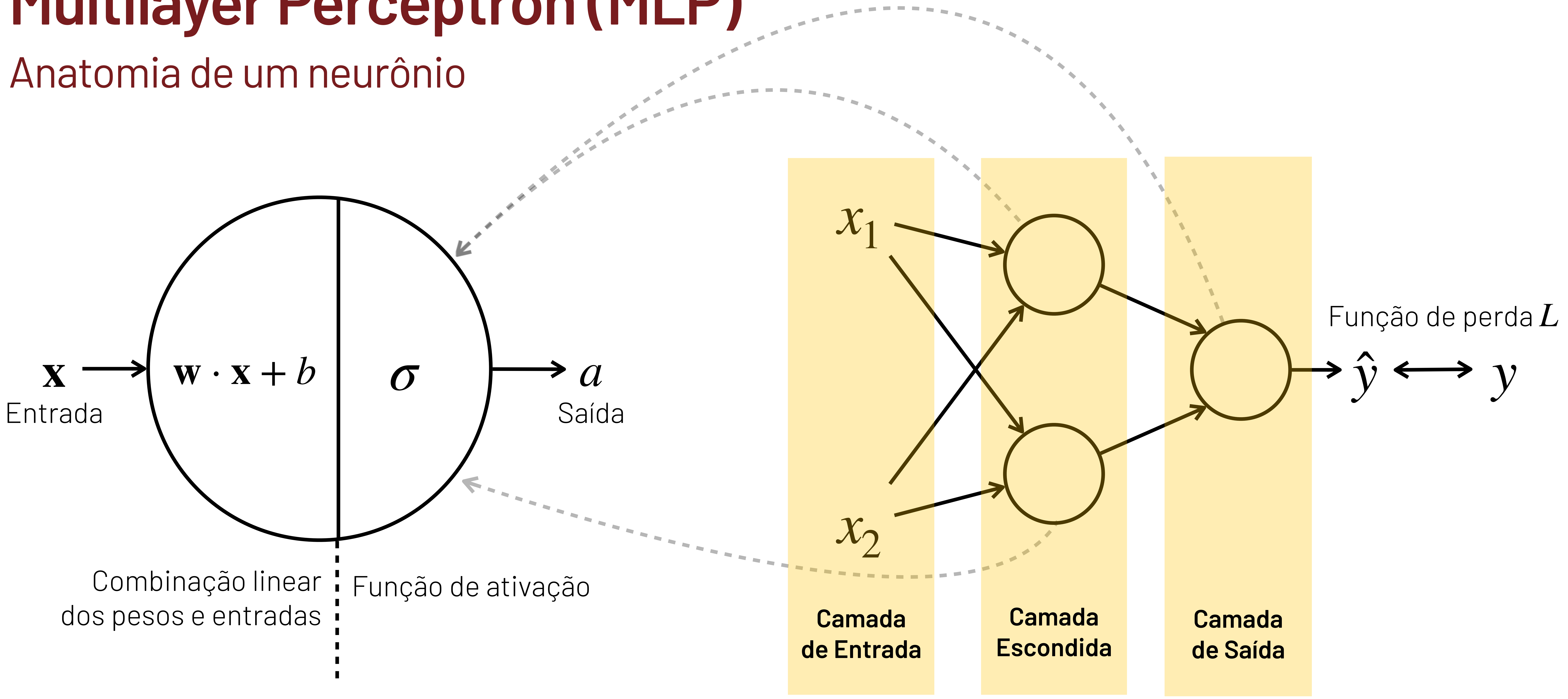
Multilayer Perceptron (MLP)

Algoritmo de classificação (binária ou multiclasse) e regressão que conecta neurônios artificiais em camadas para resolver problemas linarmente não-separáveis



Multilayer Perceptron (MLP)

Anatomia de um neurônio



UFV

Propagação da Entrada (Forward Pass)

Para um exemplo $\mathbf{x}$

$$a_1 = \sigma(w_{11}^{[1]}x_1 + w_{12}^{[1]}x_2 + b_1^{[1]})$$

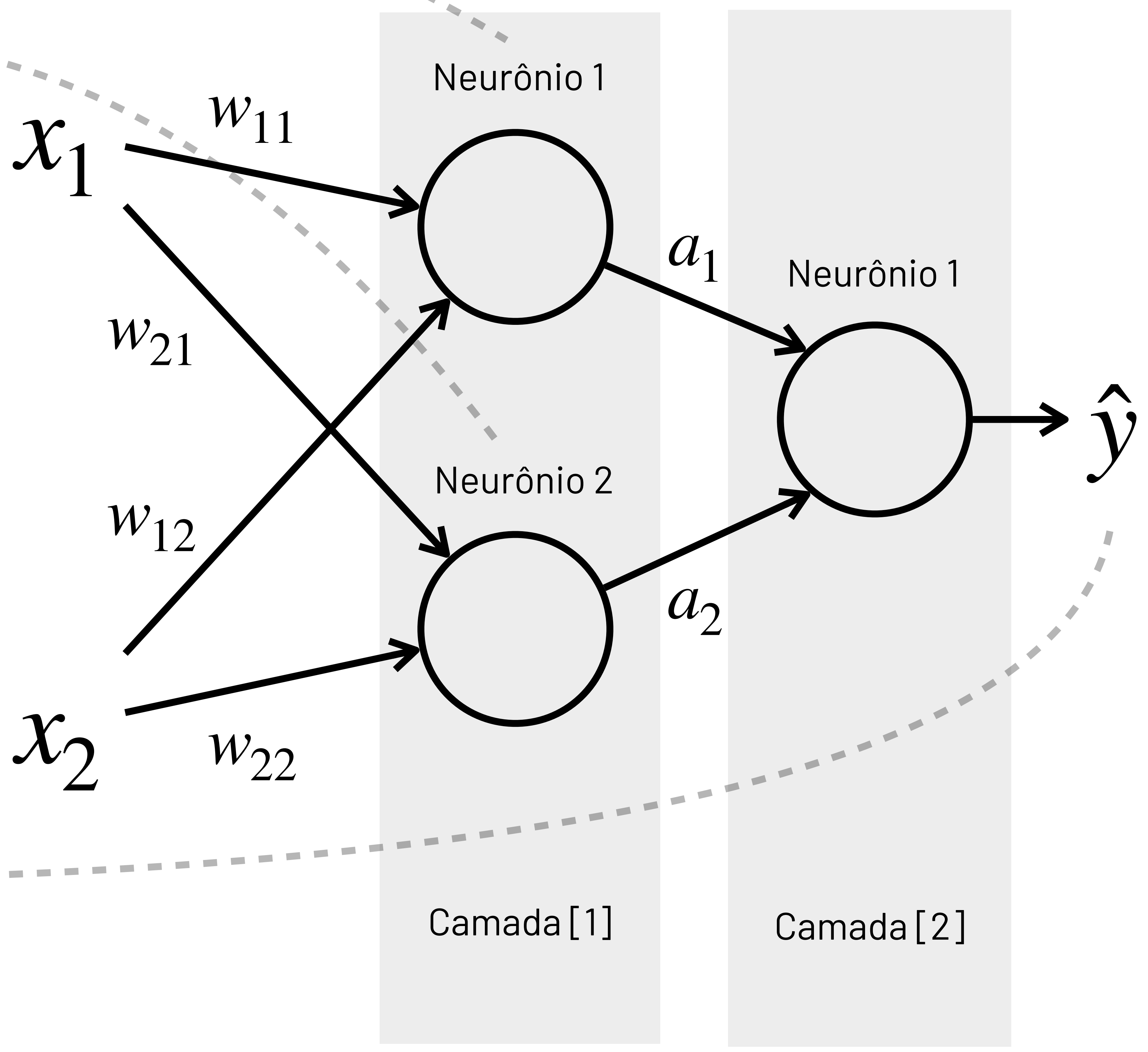
$$a_2 = \sigma(w_{21}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]})$$

$$\mathbf{a}^{[1]} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \sigma(\begin{bmatrix} w_{11}^{[1]}x_1 + w_{12}^{[1]}x_2 + b_1^{[1]} \\ w_{21}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]} \end{bmatrix})$$

$$= \sigma(\begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}) = \underline{\sigma(W^{[1]}\mathbf{x} + \mathbf{b}^{[1]})}$$

$$\hat{y} = \sigma(w_{11}^{[2]}a_1 + w_{12}^{[2]}a_2 + b_1^{[2]})$$

$$\hat{y} = \sigma(\begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + b_1^{[2]}) = \underline{\sigma(W^{[2]}\mathbf{a} + b_1^{[2]})}$$



UFV

Propagação da Entrada (Forward Pass)

Para o conjunto de dados X com n exemplos

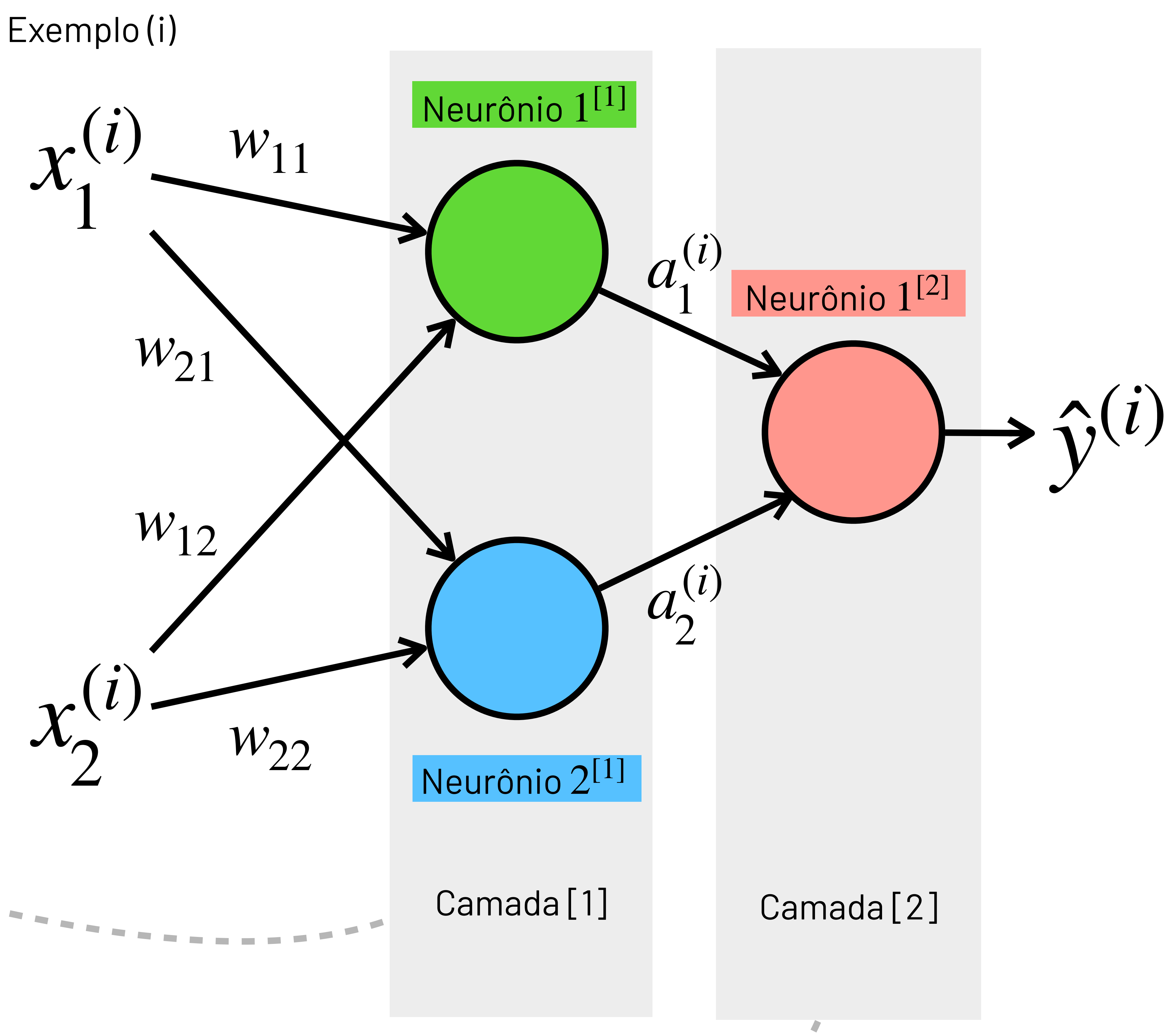
$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \end{bmatrix}$$

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad \mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}$$

$$\underline{A^{[1]} = \sigma(W^{[1]}X + \mathbf{b}^{[1]})} = \sigma\left(\begin{bmatrix} a_1^{(1)} & a_1^{(2)} & \dots & a_1^{(n)} \\ a_2^{(1)} & a_2^{(2)} & \dots & a_2^{(n)} \end{bmatrix}\right)$$

$$W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix}$$

$$\underline{\hat{Y} = \sigma(W^{[2]}A^{[1]} + b^{[2]})} = \begin{bmatrix} \hat{y}^{(1)} & \hat{y}^{(2)} & \dots & \hat{y}^{(n)} \end{bmatrix}$$



UFV

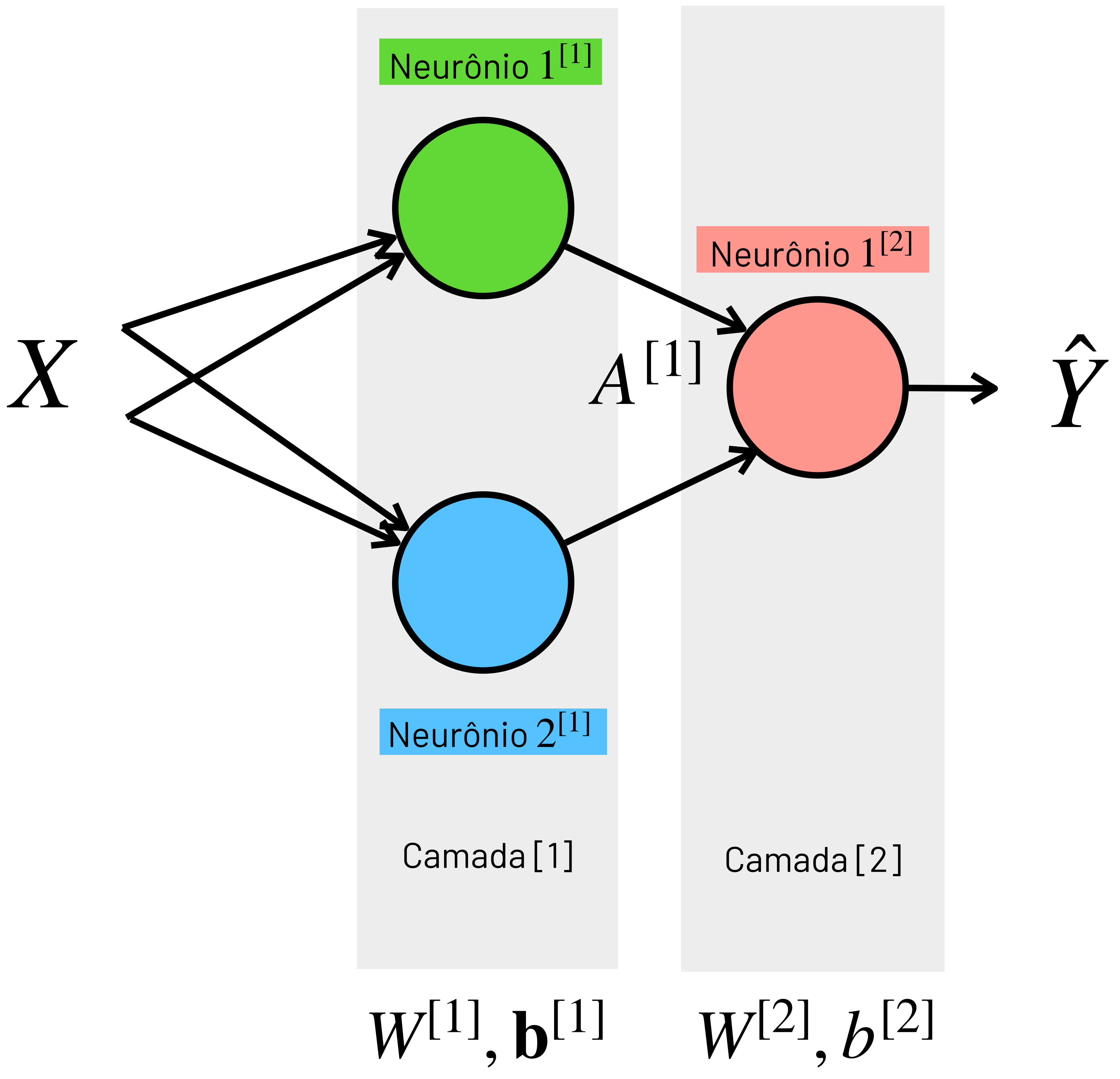
Hipótese

Hipótese

$$Z^{[1]} = W^{[1]}X + \mathbf{b}^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$$
$$\hat{Y} = \sigma(Z^{[2]})$$

$$h(\mathbf{x}) = \sigma(W^{[2]} \cdot \sigma(W^{[1]}X + \mathbf{b}^{[1]}) + b^{[2]}$$
$$h(\mathbf{x}) = \underline{\sigma(W^{[2]} \cdot h^{[1]}(X) + b^{[2]})}$$

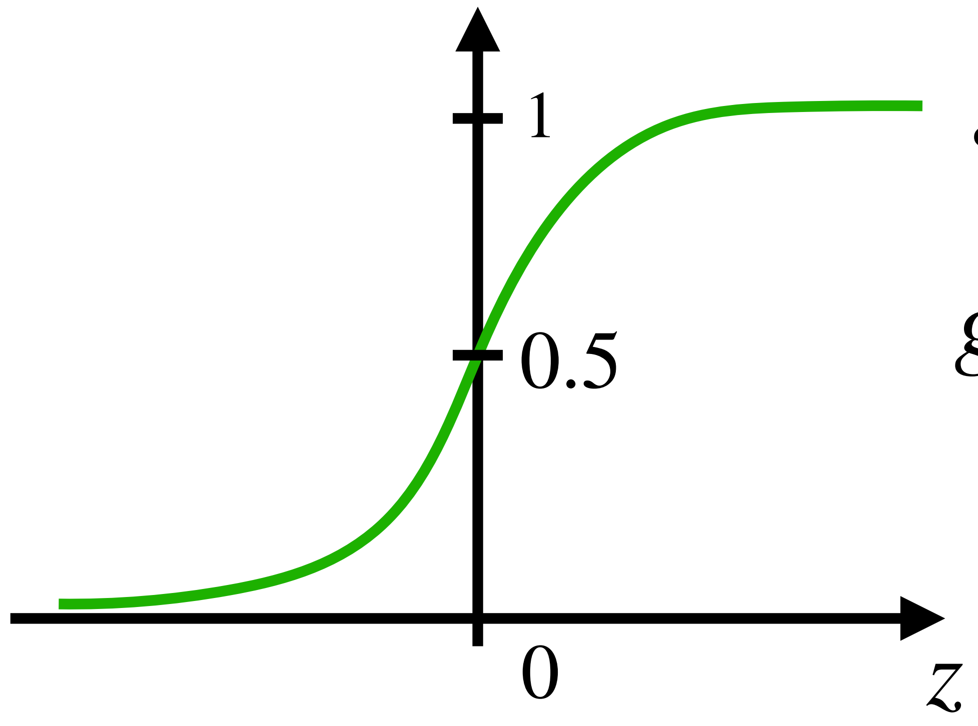
MLPs aprendem funções compostas!



UFV

Funções de Ativação

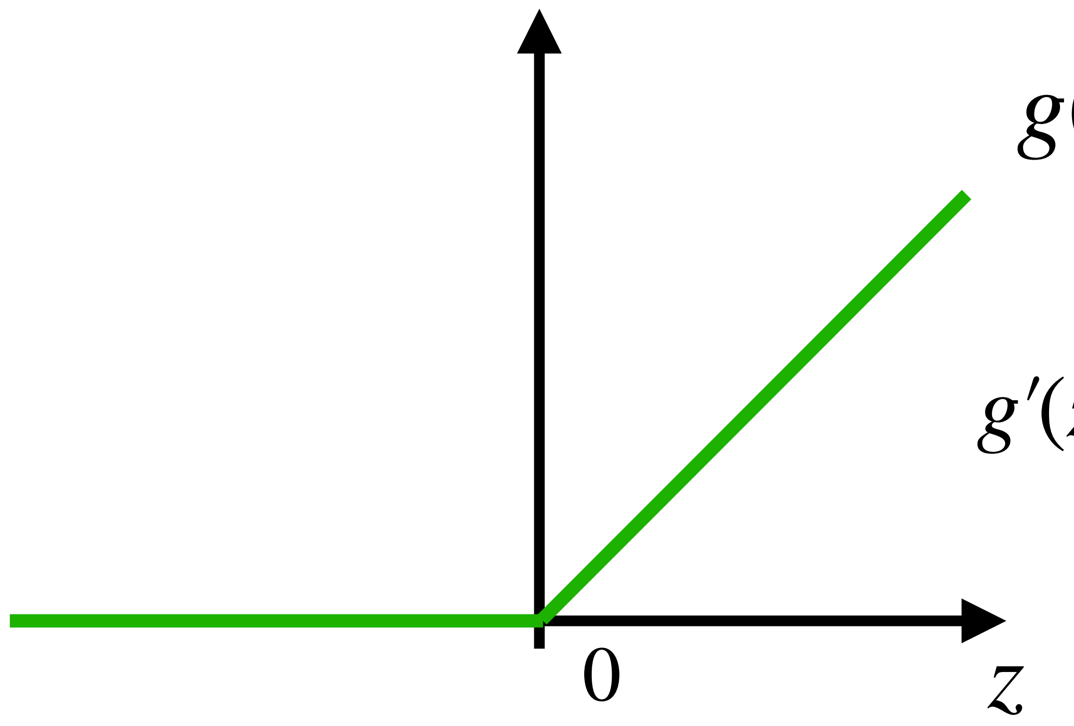
Logística (sigmoide)



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

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

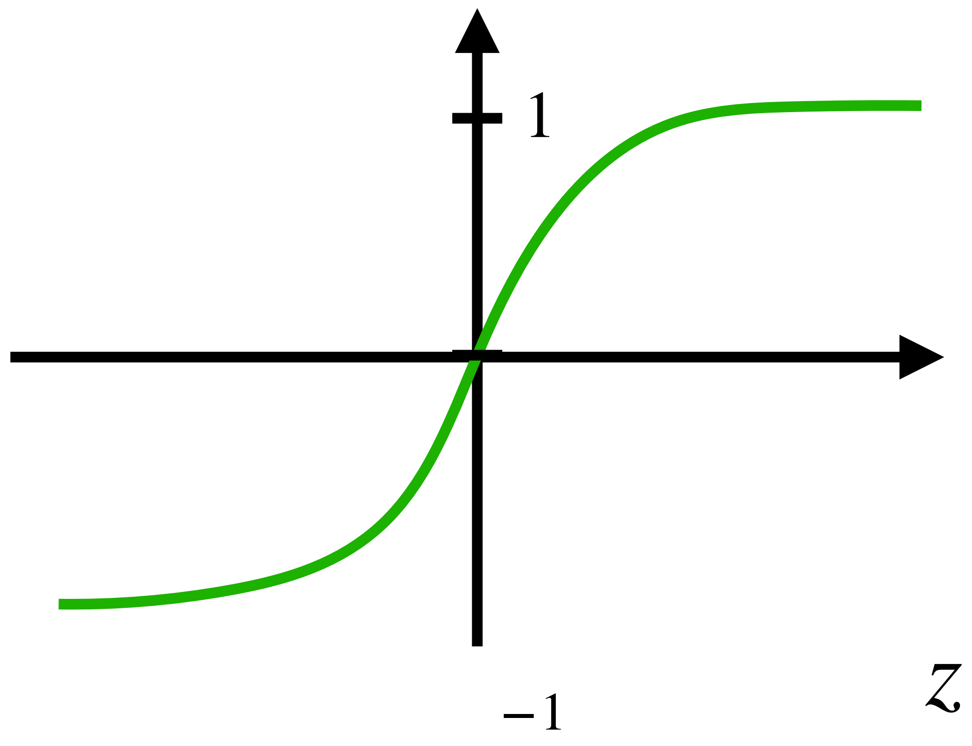
Unidade Linear Retificada (ReLU)



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

$$g'(z) = \begin{cases} 1 \text{ se } z < 0 \\ 0 \text{ se } z > 0 \\ \nexists, \text{ se } z = 0 \end{cases}$$

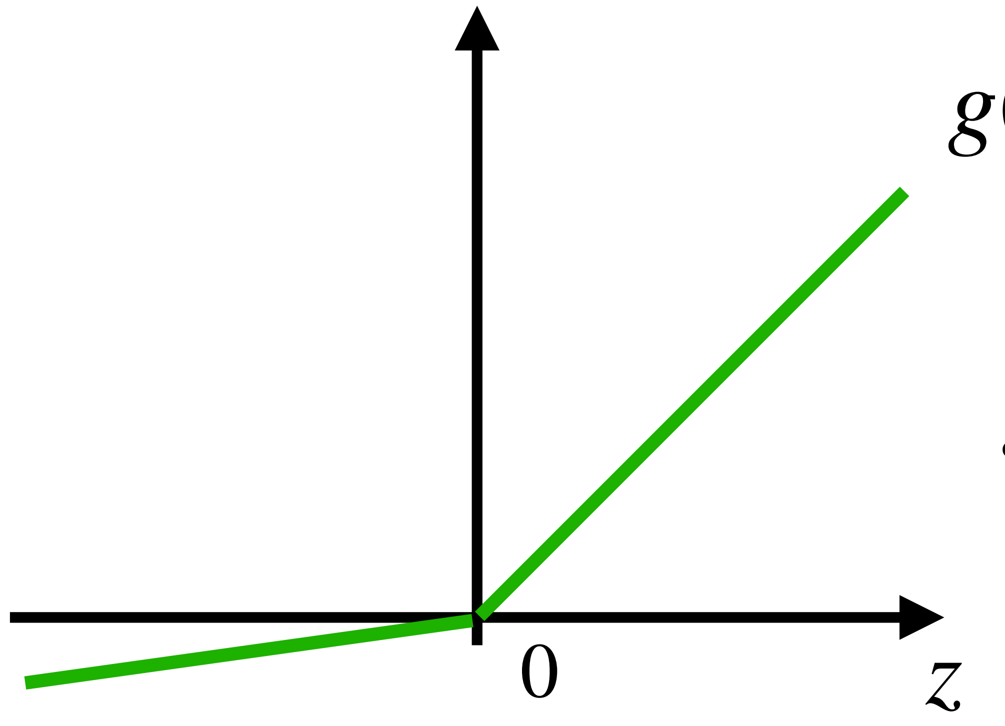
Tangete Hiperbólica



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

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

Leaky ReLU



$$g(z) = max(0.01z, z)$$

$$g'(z) = \begin{cases} 1 \text{ se } z < 0 \\ 0.01 \text{ se } z > 0 \\ \nexists, \text{ se } z = 0 \end{cases}$$

UFV

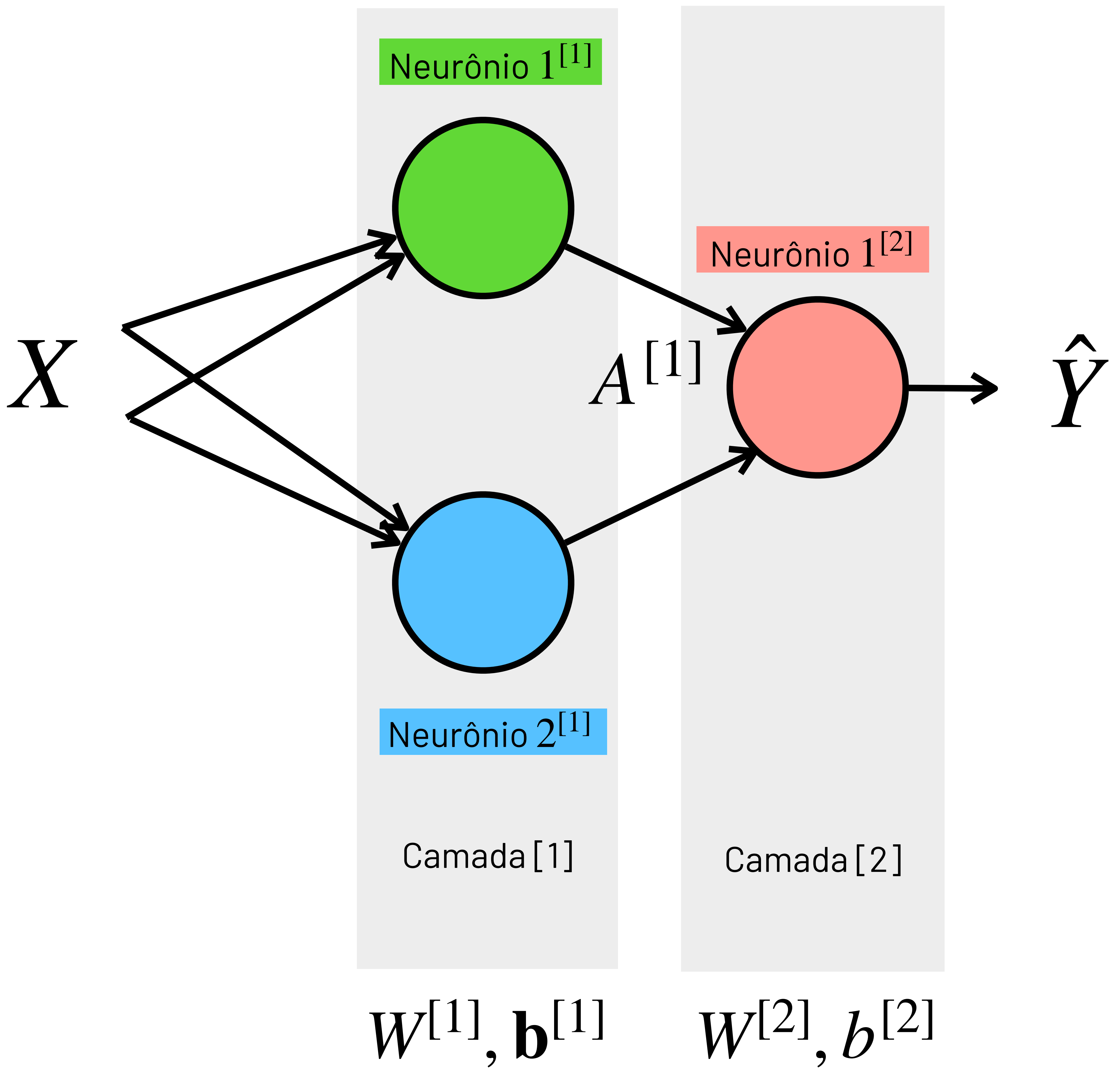
Porque precisamos de funções de ativação não lineares?

$$Z^{[1]} = W^{[1]}X + \mathbf{b}^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$$
$$\hat{Y} = \sigma(Z^{[2]})$$

Se utilizarmos funções de ativação lineares, a hipótese será linear!

$$h(\mathbf{x}) = \cancel{\sigma}(W^{[2]} \cdot \cancel{g}(W^{[1]} \cdot \mathbf{x} + \mathbf{b}^{[1]}) + b^{[2]})$$
$$h(\mathbf{x}) = W^{[2]} \cdot (W^{[1]} \cdot \mathbf{x} + \mathbf{b}^{[1]}) + b^{[2]}$$
$$h(\mathbf{x}) = \underbrace{(W^{[2]} \cdot W^{[1]})}_{W'} \cdot \mathbf{x} + \underbrace{(W^{[2]} \cdot \mathbf{b}^{[1]}) + b^{[2]}}_{b'}$$

$$h(x) = W' \cdot \mathbf{x} + b'$$



Próxima aula

A7: MLP em Numpy

Aula prática sobre implementação de redes neurais profundas com Numpy.