INF721

2023/2

UFV

Aprendizado em Redes Neurais Profundas

A6: Multilayer Perceptron (MLP)

Logística

Avisos

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

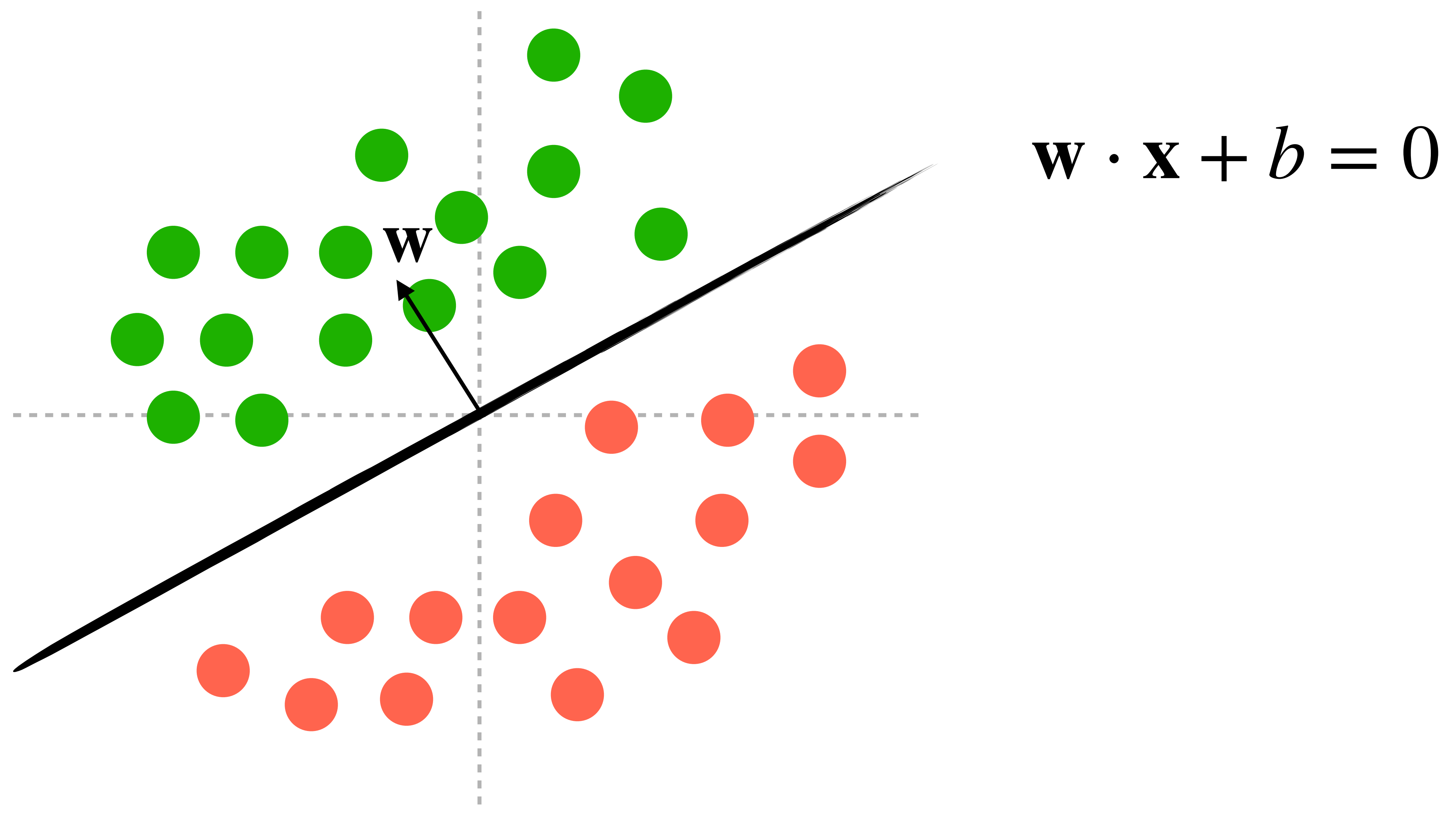
Última aula

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

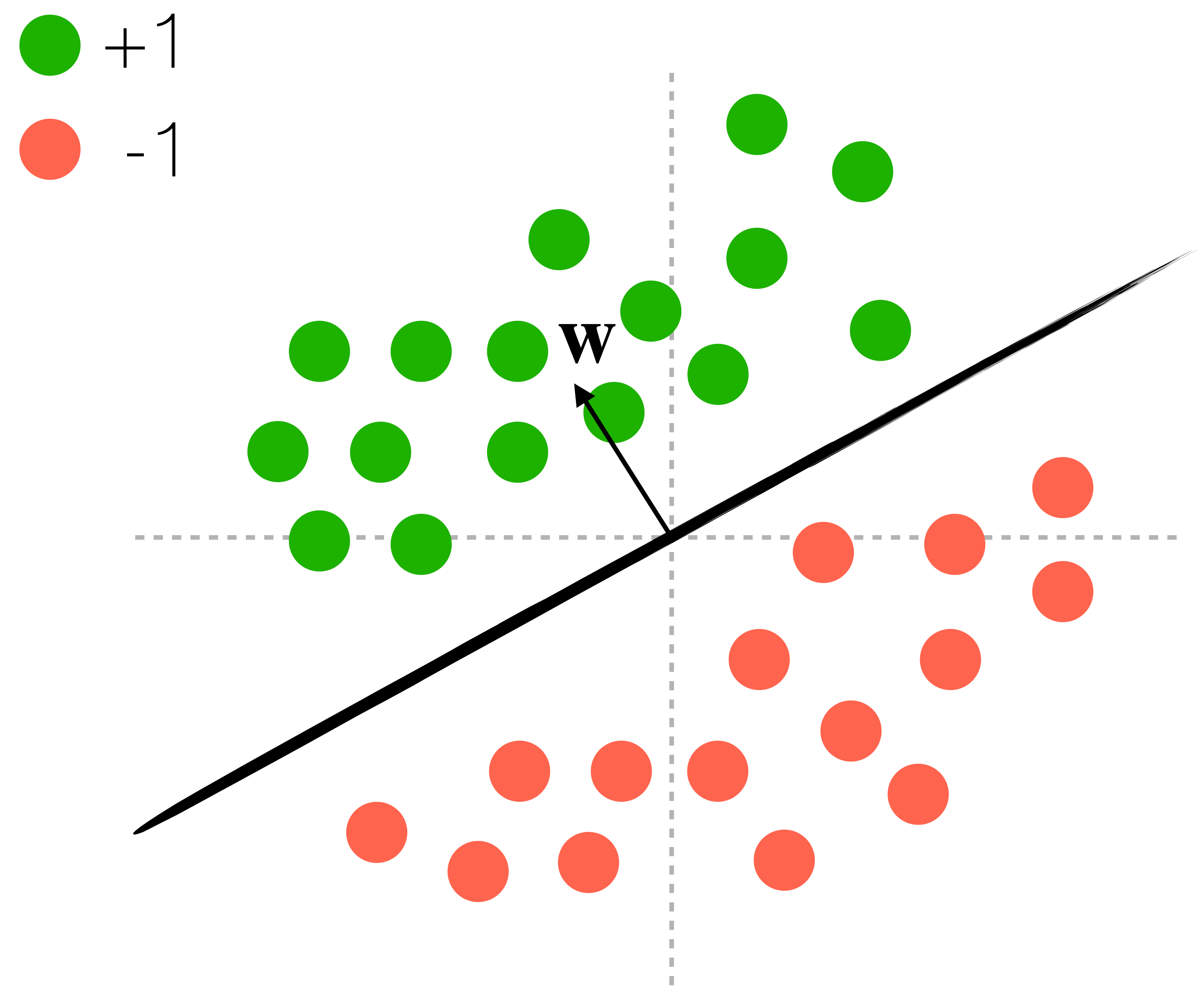
Plano de Aula

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

Problemas Linearmente Separáveis



Problemas Linearmente Separáveis

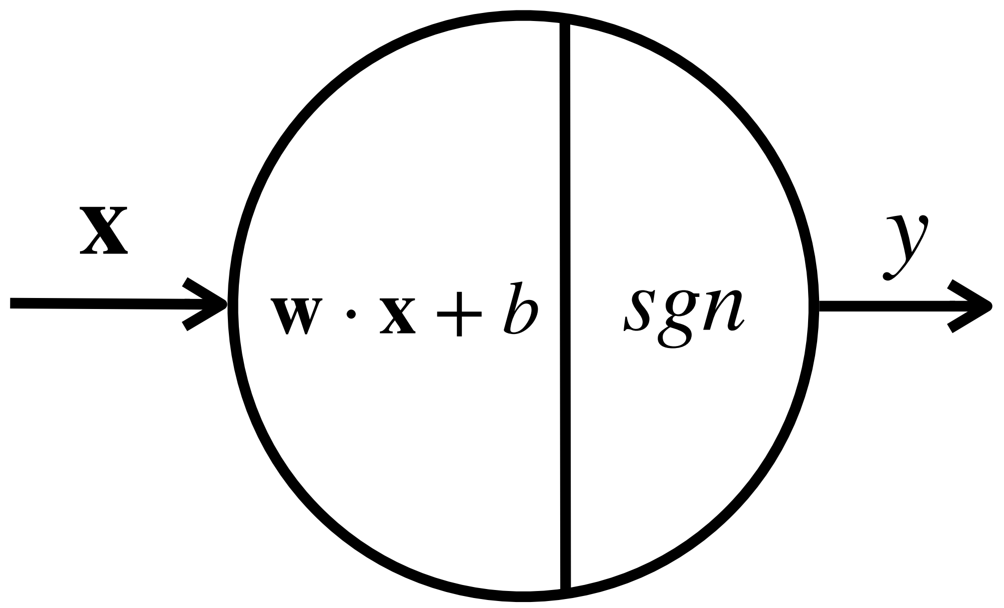


$$sgn(z) = \begin{cases} 1, & z > 0 \\ -1, & z < 0 \end{cases}$$

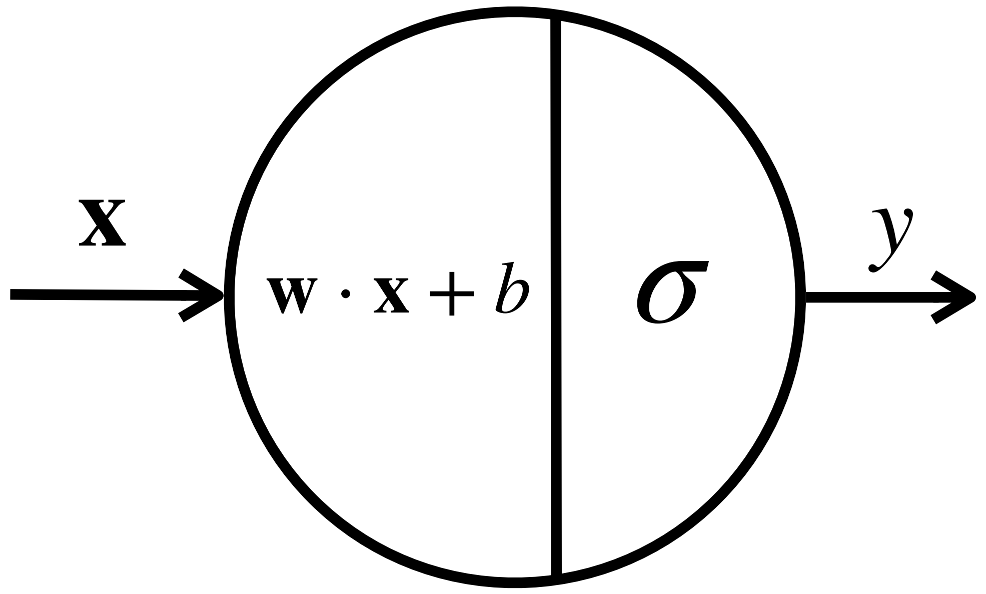
$$h(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b)$$

Problemas Linearmente Separáveis

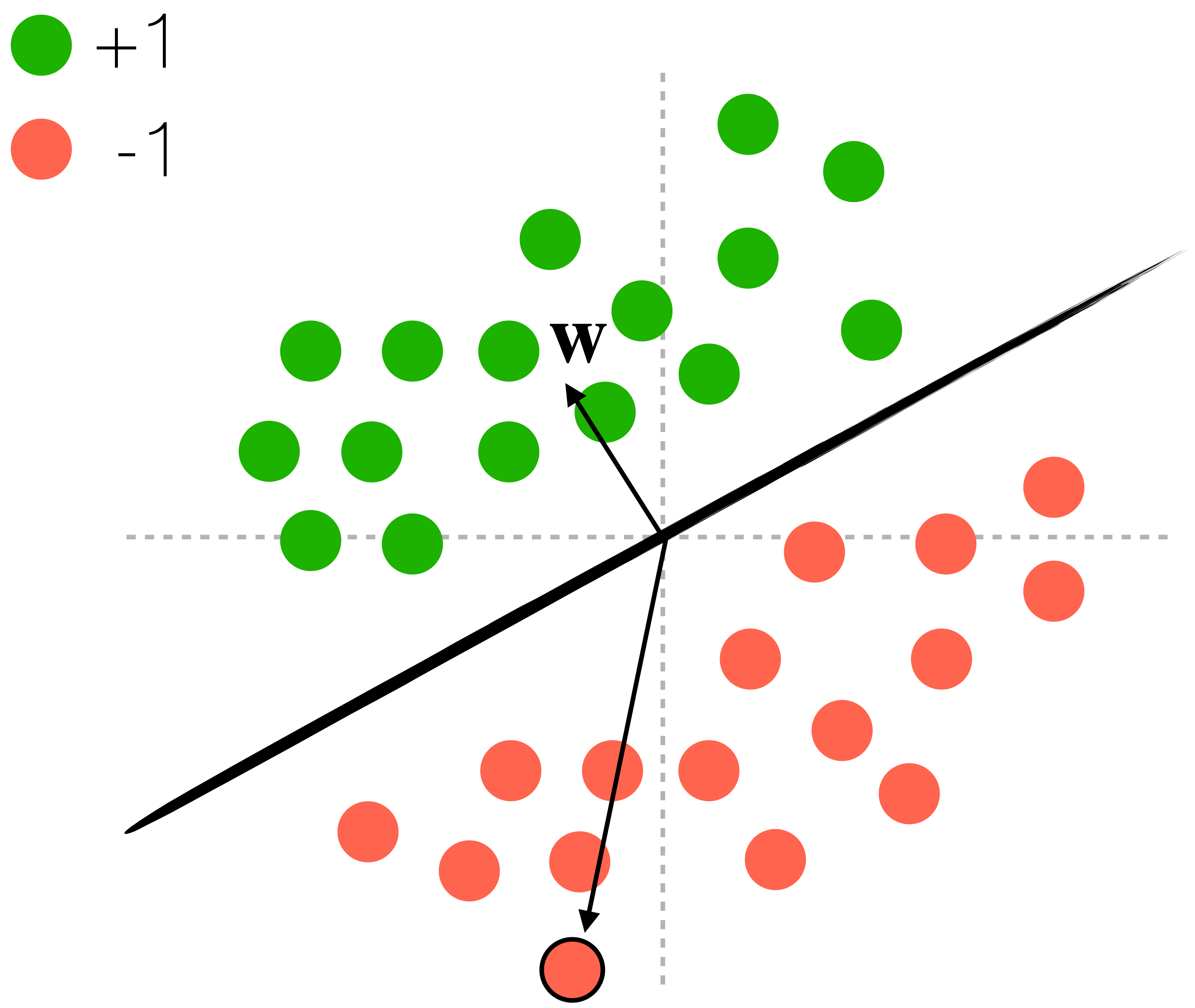
Perceptron



Regressão Logística



Ambos aprendem apenas fronteiras de decisão lineares!



$\mathbf{x} = [-1, -5]$

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

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

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

$$b = 0$$

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

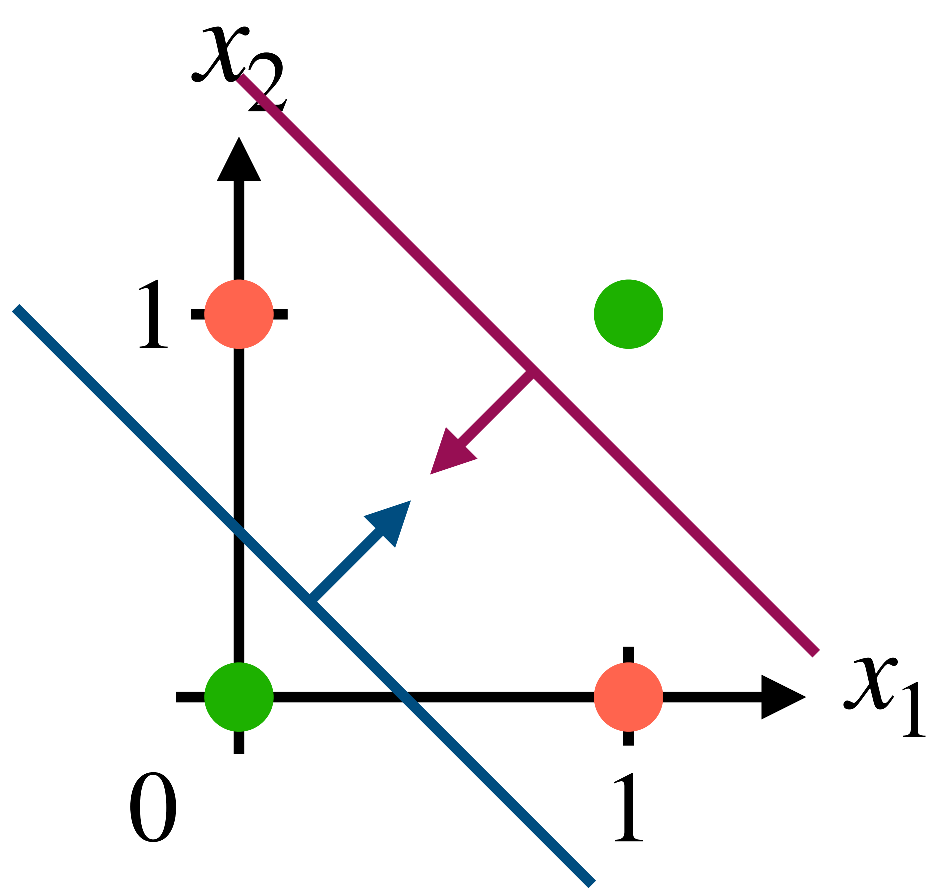
$$h(\mathbf{x}) = sgn(-3)$$

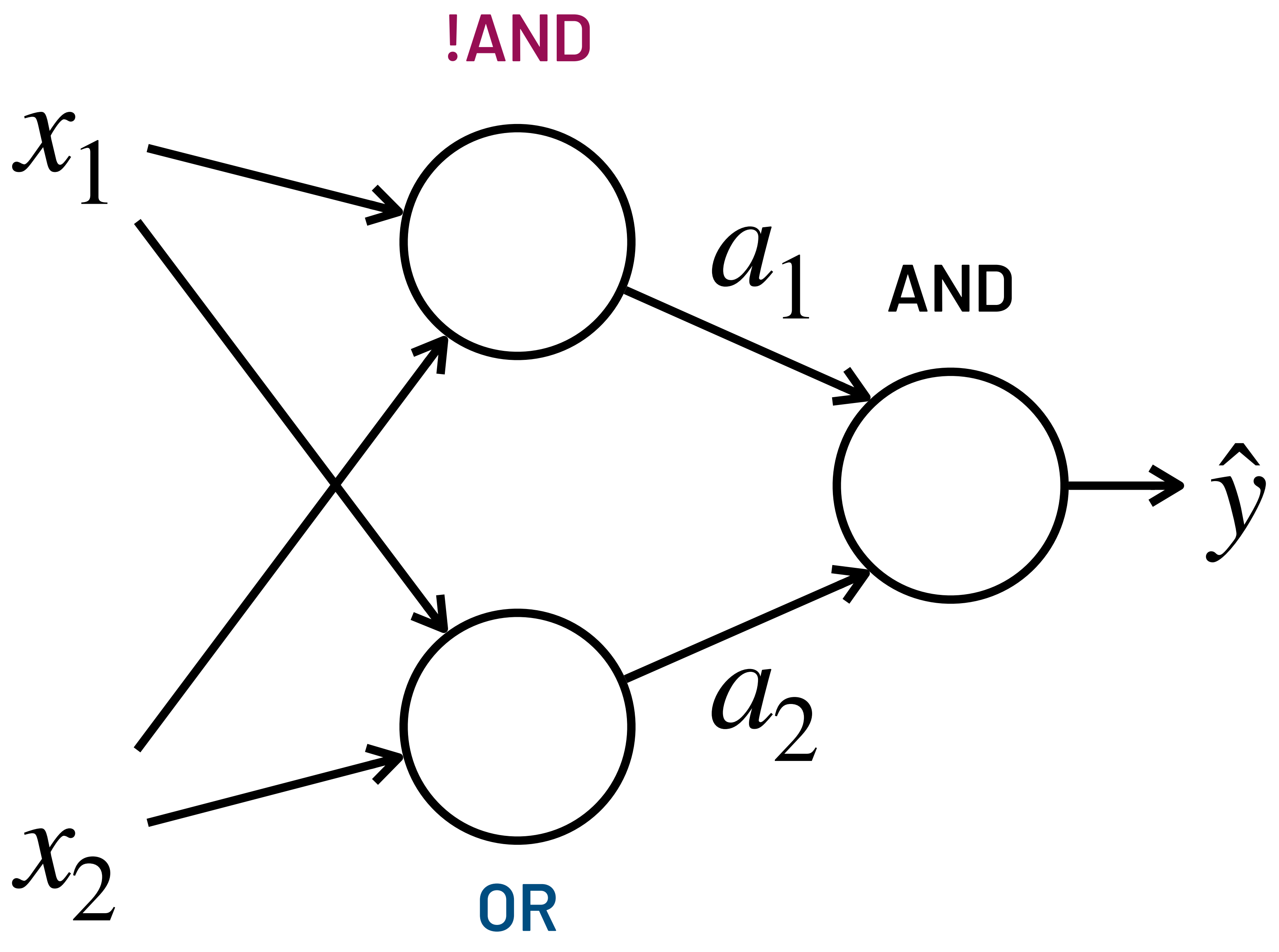
$$h(\mathbf{x}) = -1$$

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

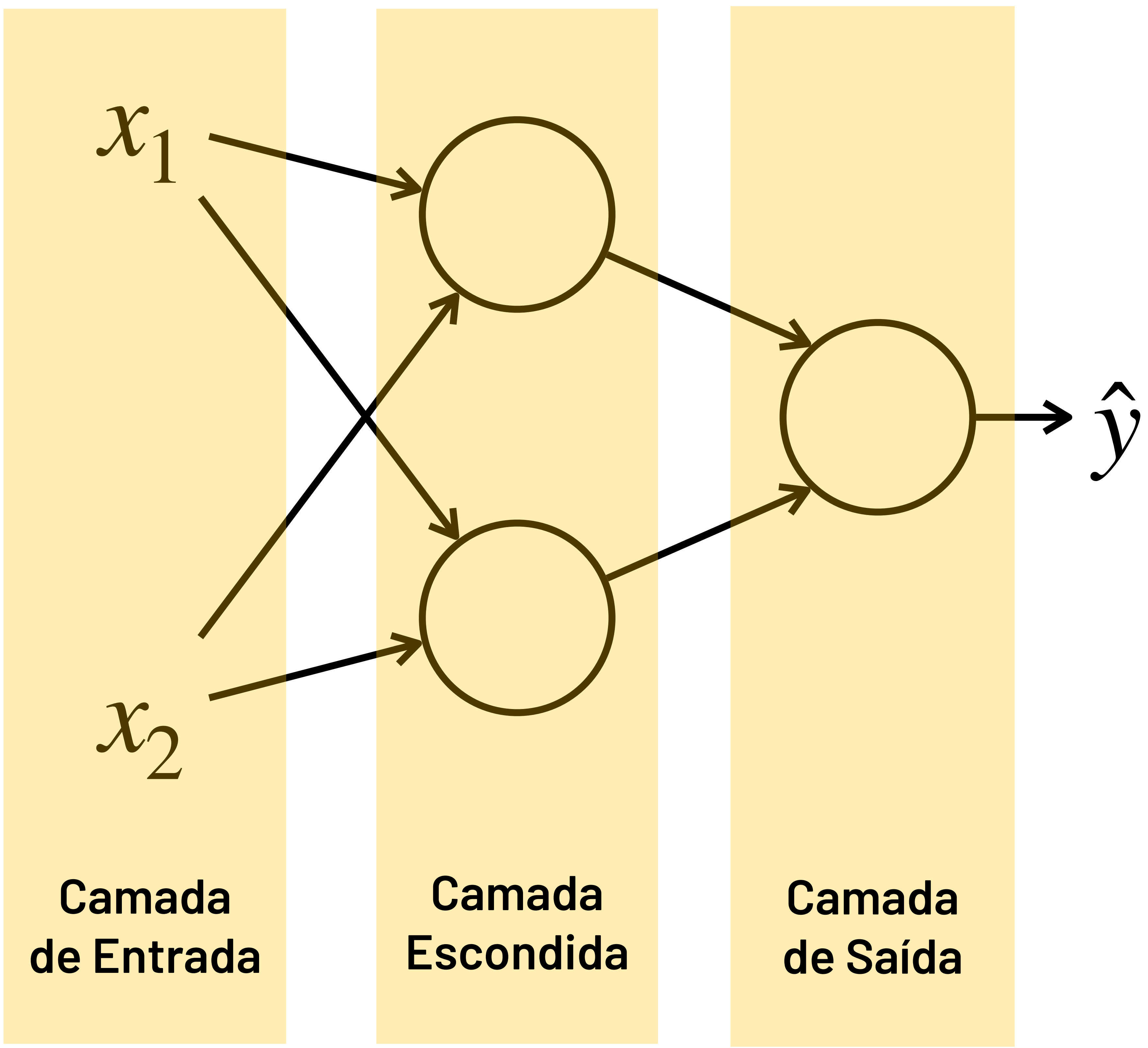




RNAs aprendem representações intermediárias $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ dos dados de entrada $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, chamadas **representações latentes**, que podem tornar um problema não-linearmente separável em linearmente separável!

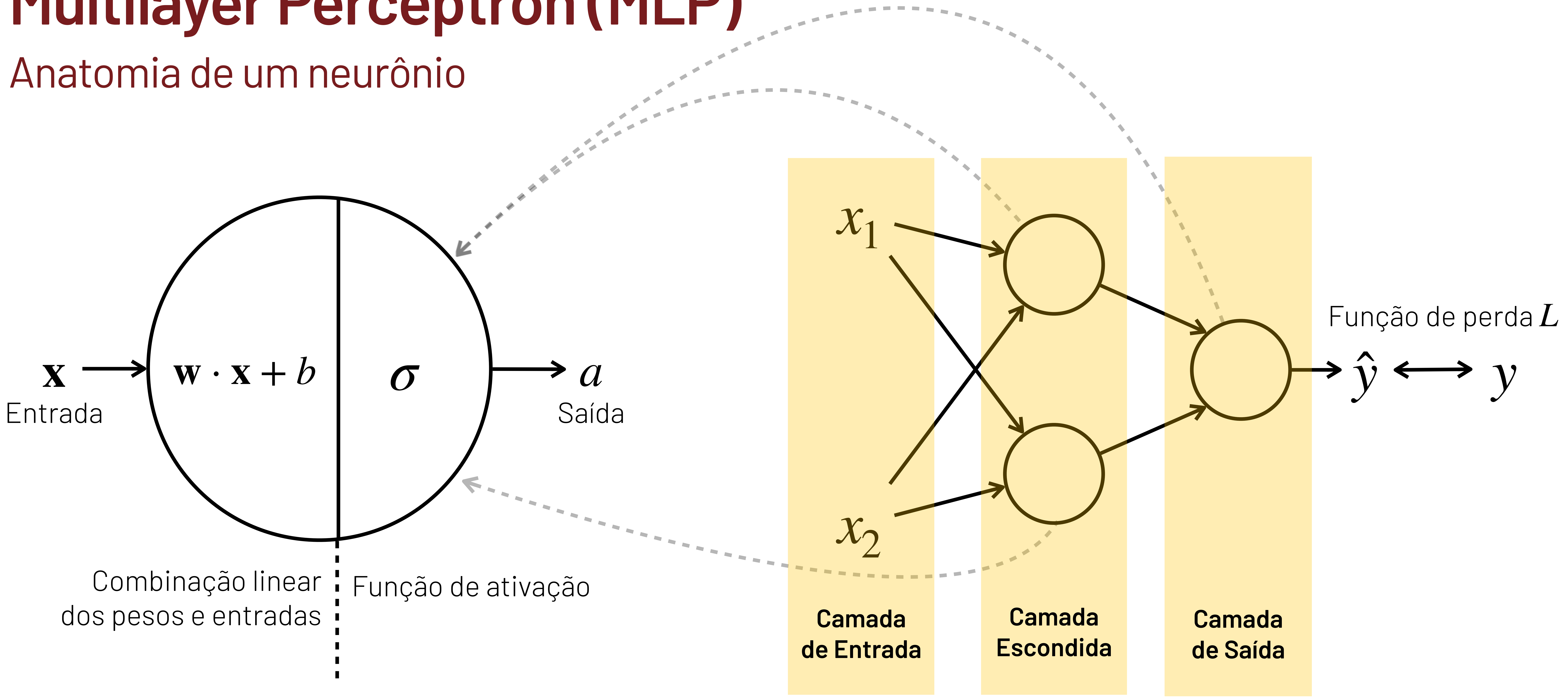
Multilayer Perceptron (MLP)

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



Propagação da Entrada (Forward Pass)

Para um exemplo $\mathbf{x}$

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

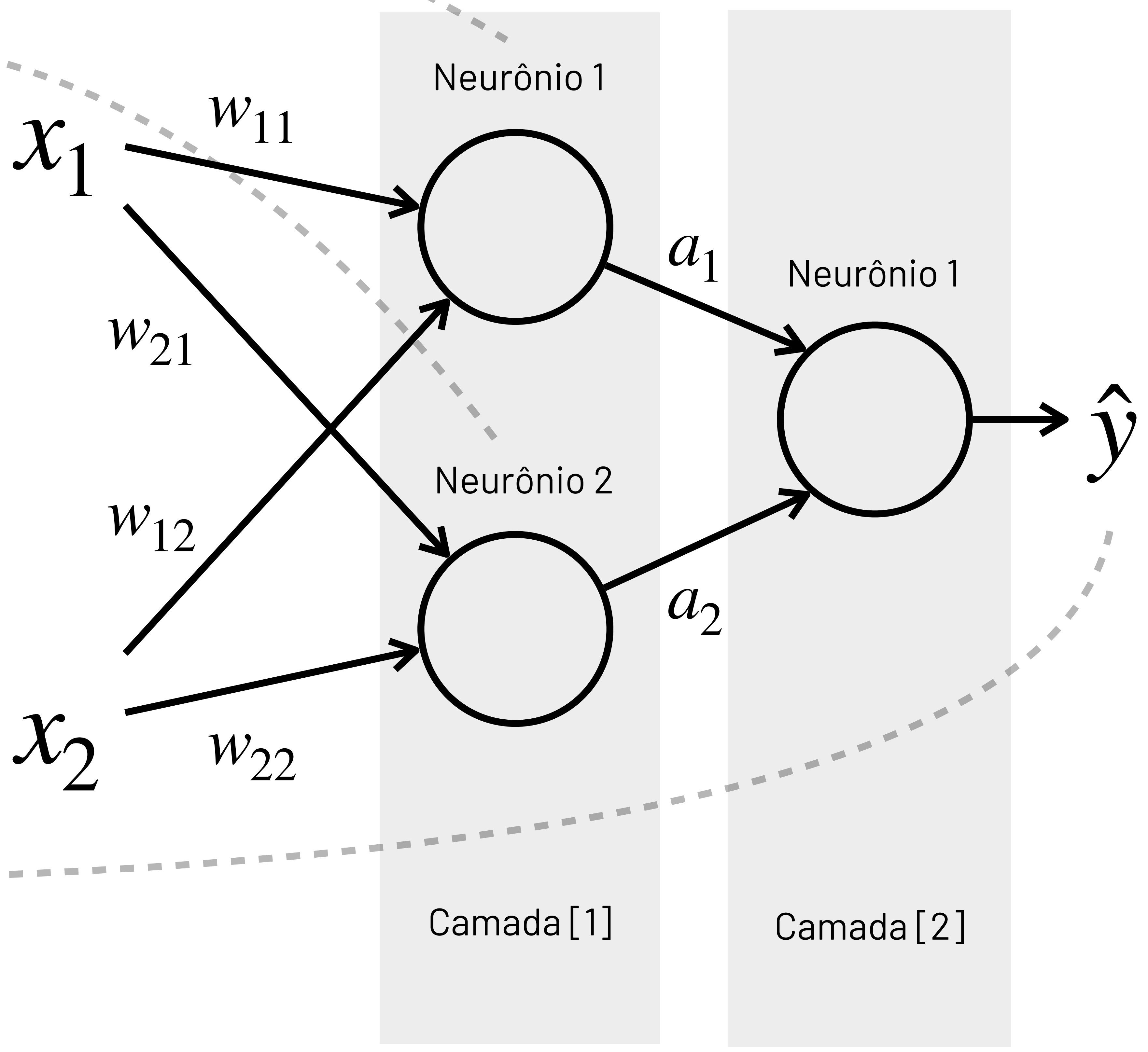
$$a_2 = \sigma(w_{21}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]})$$

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

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

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

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



UFV

Propagação da Entrada (Forward Pass)

Para o conjunto de dados X com n exemplos

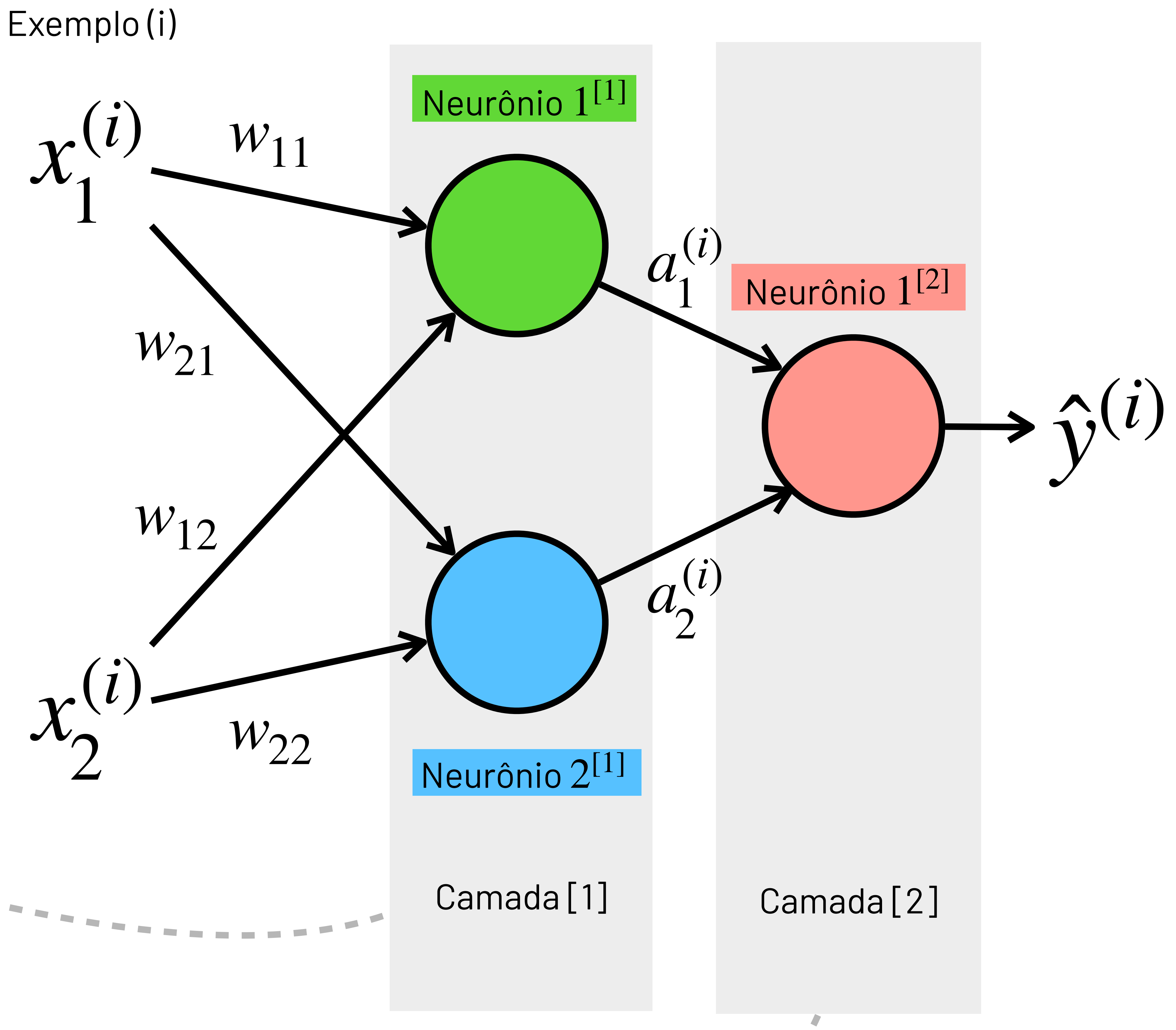
$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \end{bmatrix}$$

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

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

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

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



UFV

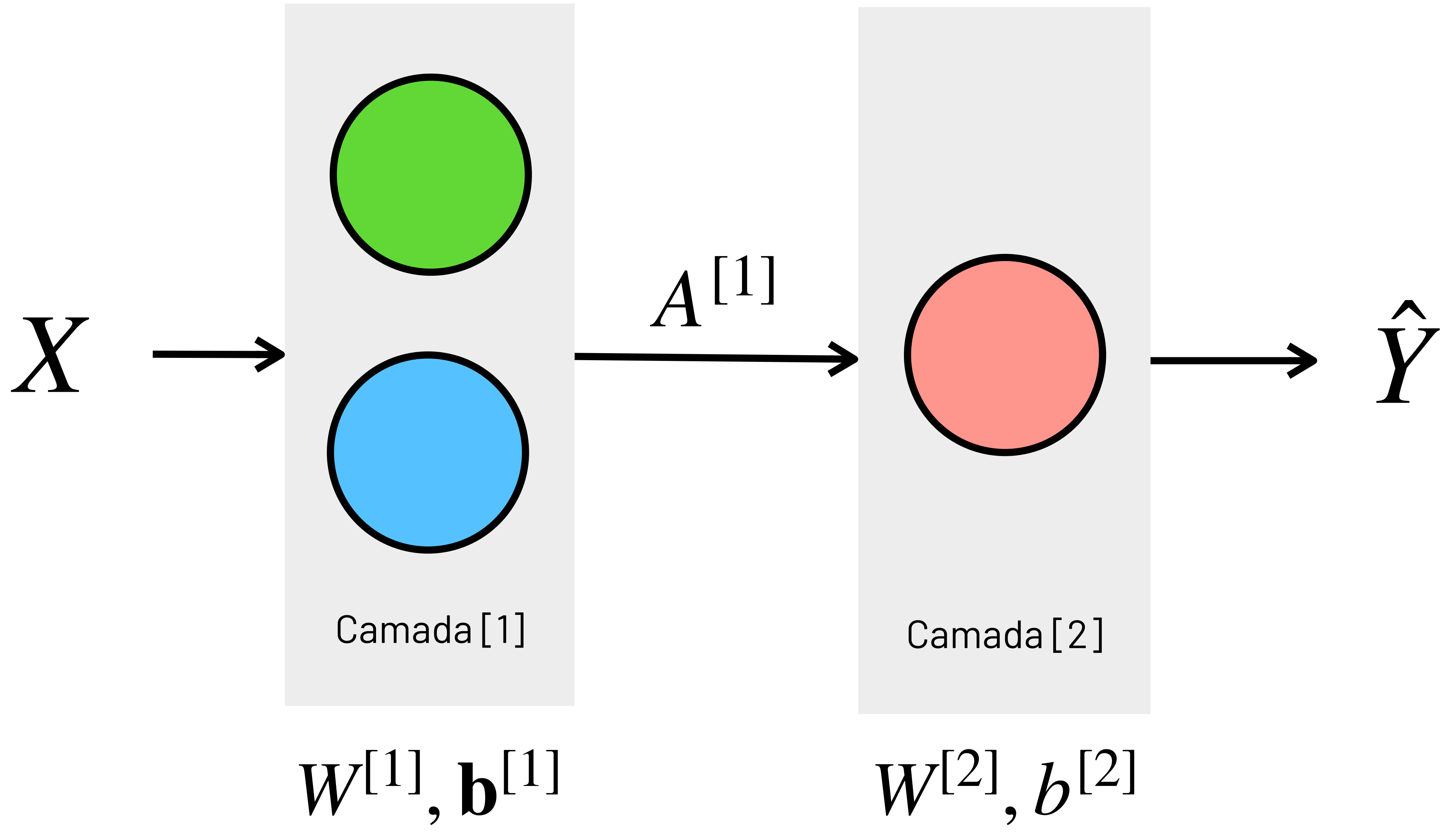
Hipótese

Hipótese

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

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

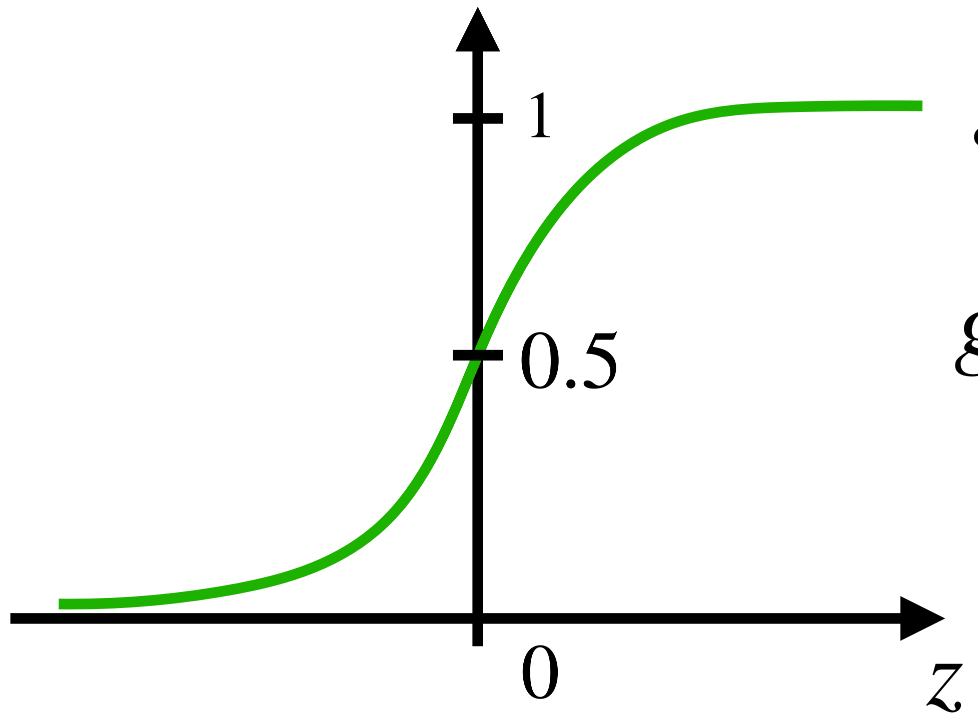
MLPs aprendem funções compostas!



UFV

Funções de Ativação

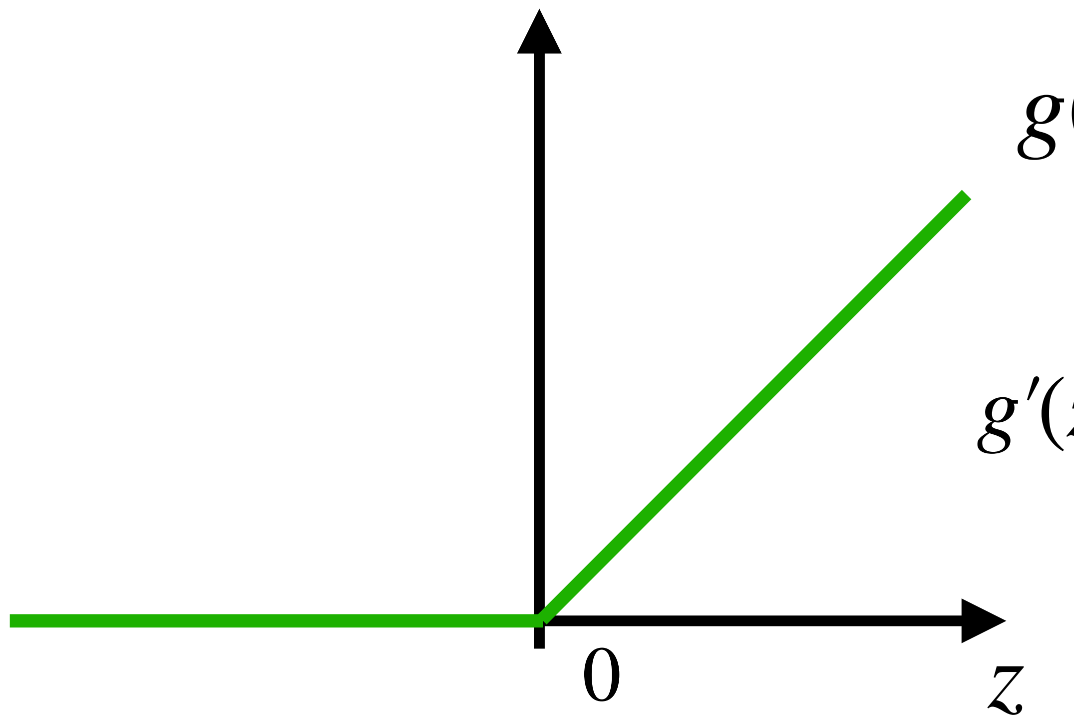
Logística (sigmoide)



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

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

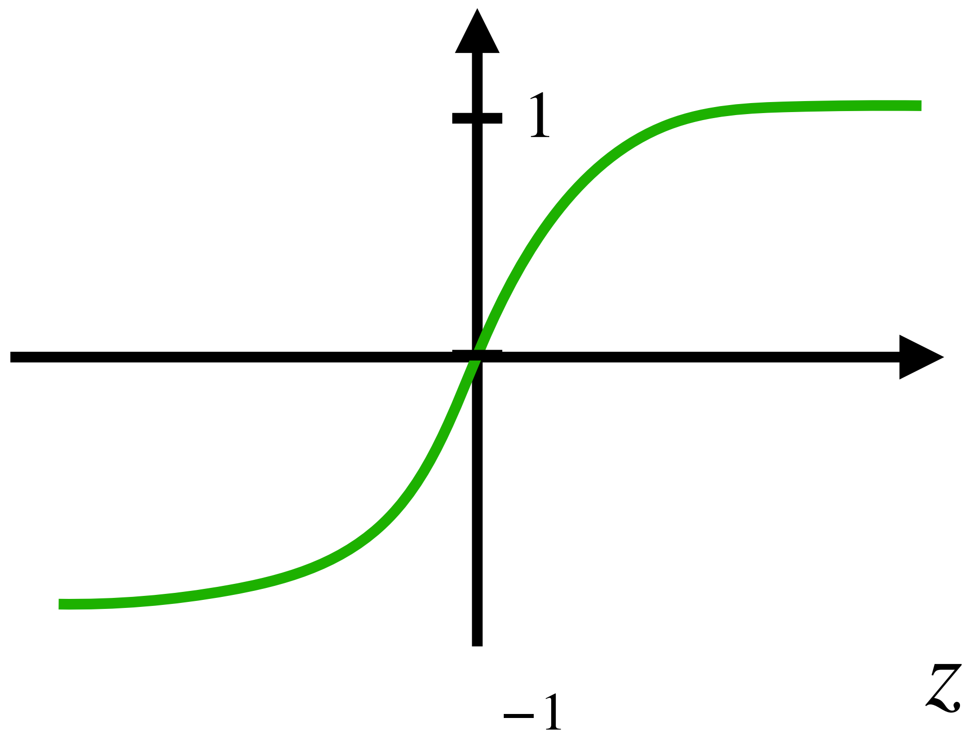
Unidade Linear Retificada (ReLU)



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

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

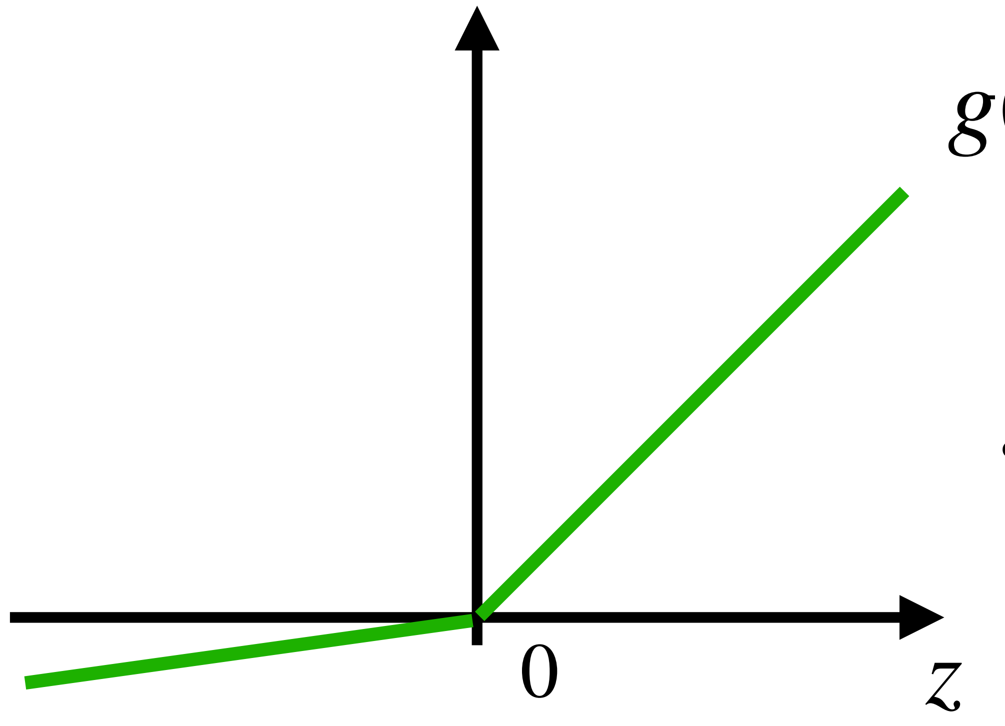
Tangete Hiperbólica



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

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

Leaky ReLU



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

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

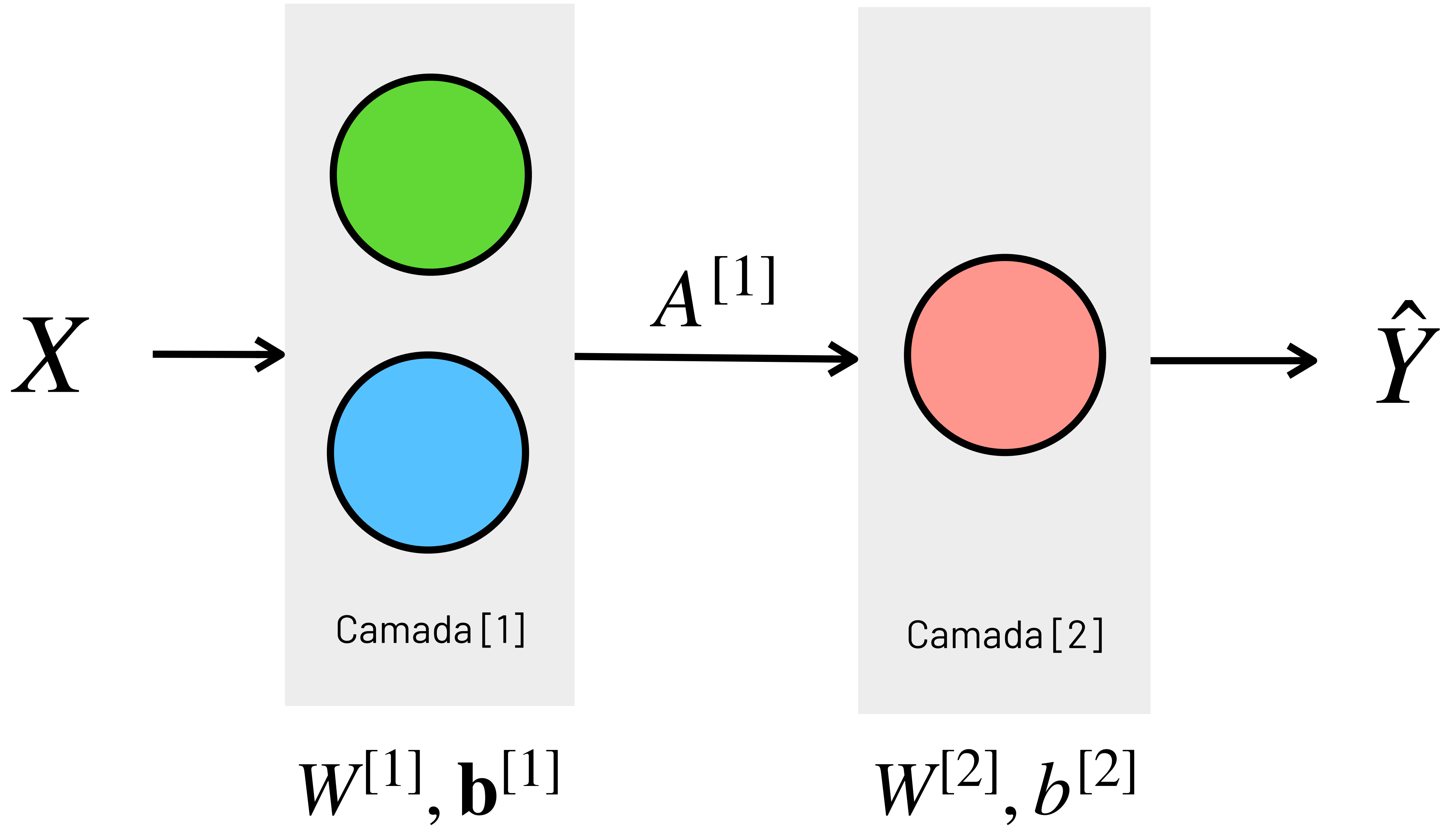
Porque precisamos de funções de ativação não lineares?

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

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

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

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



Inicialização de pesos na MLP

Em RNAs com pelo menos 1 camada escondidada (MLPs), temos que inicializar os pesos com valores aleatórios próximos de zero.

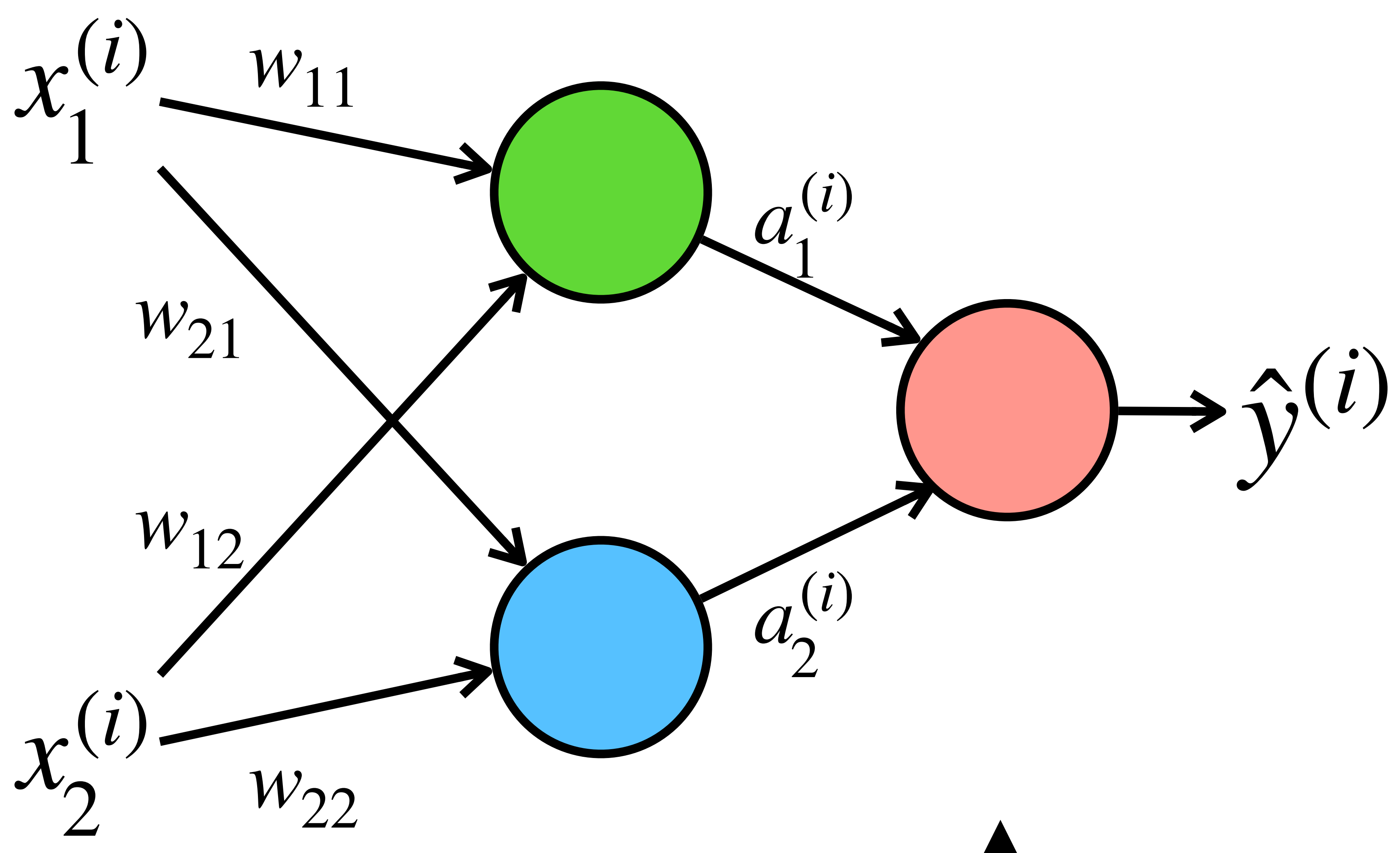
Se inicializarmos com zeros, os neurônios da camada escondida serão iguais!

$$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

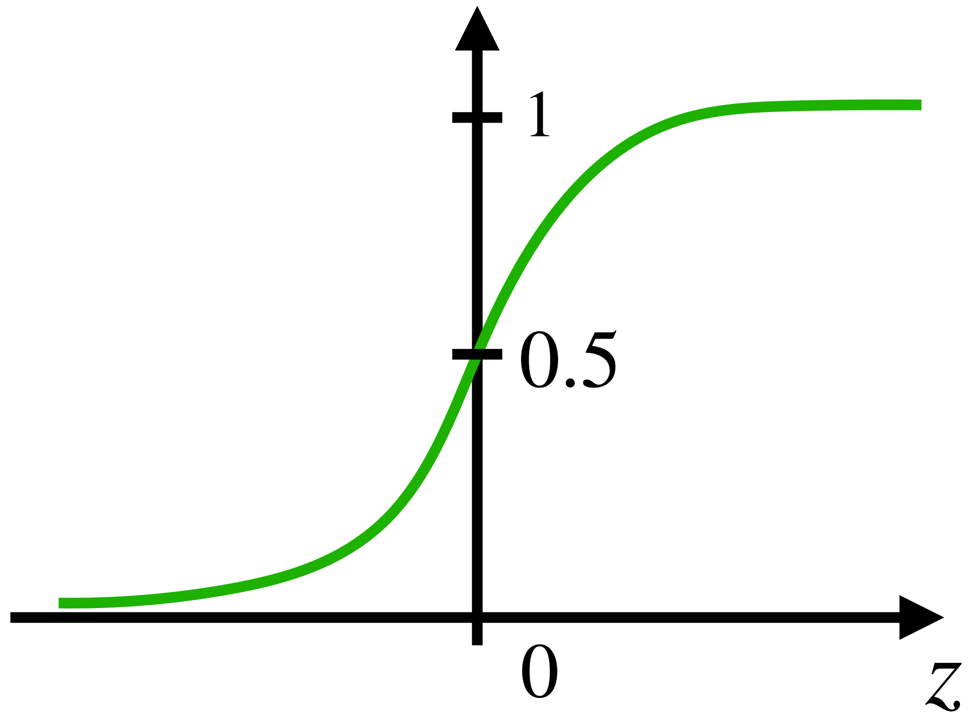
$$W^{[2]} = [0 \quad 0]$$

$$a_1^{(i)} = a_2^{(i)} \longrightarrow dZ_1^{[1]} = dZ_2^{[1]}$$

$$dW = \begin{bmatrix} u & u \\ u & u \end{bmatrix}$$



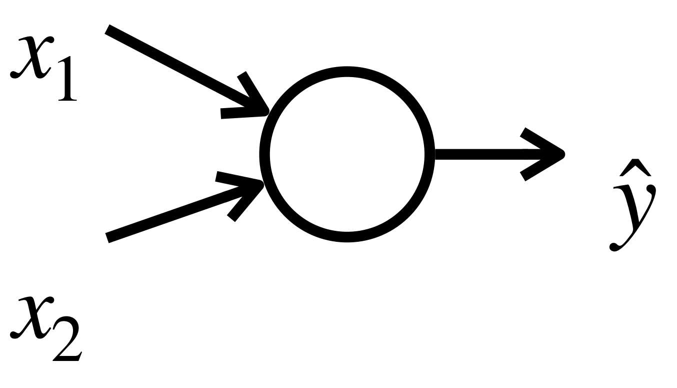
Ns regiões próximas de zero o gradiente é maior!



Redes Neurais Artificiais Profundas

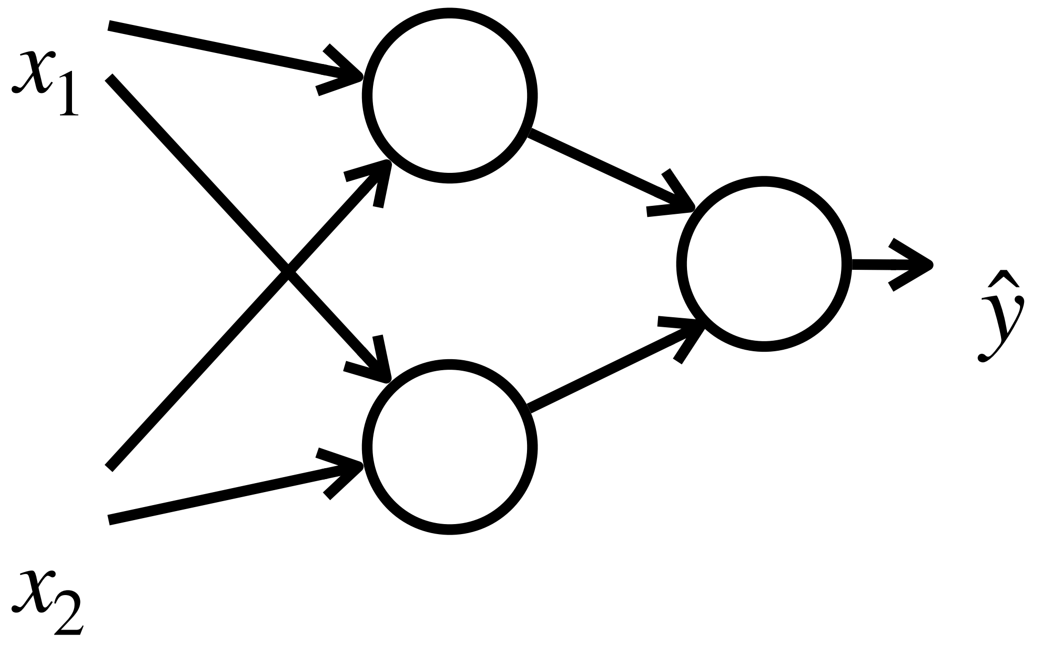
Regressão Logística

RNA de 1 camada (raza)



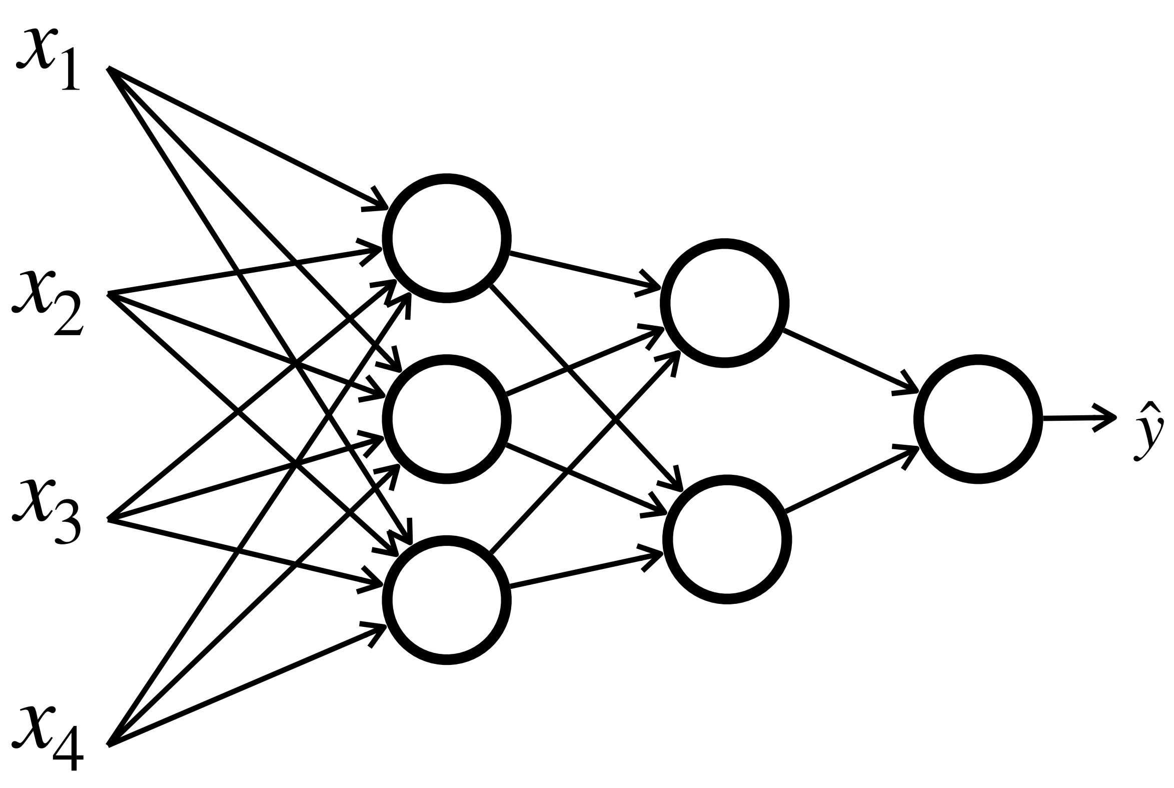
1 camada oculta

RNA de 2 camadas (raza)



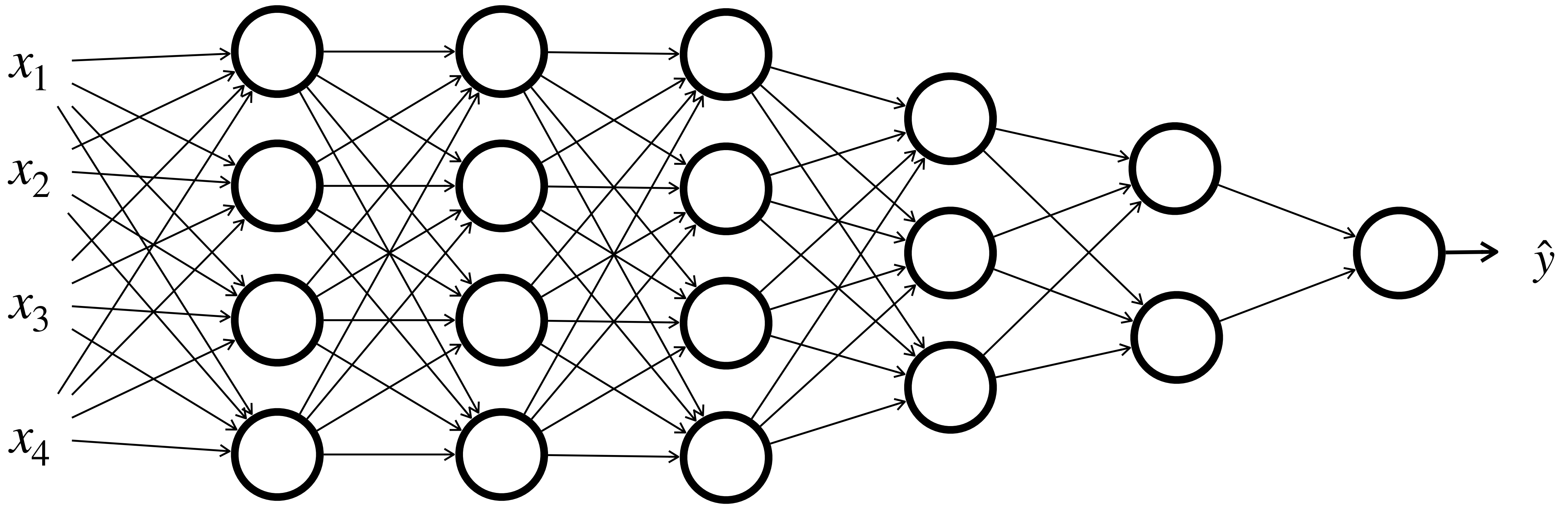
2 camadas ocultas

RNA de 3 camadas (raza)



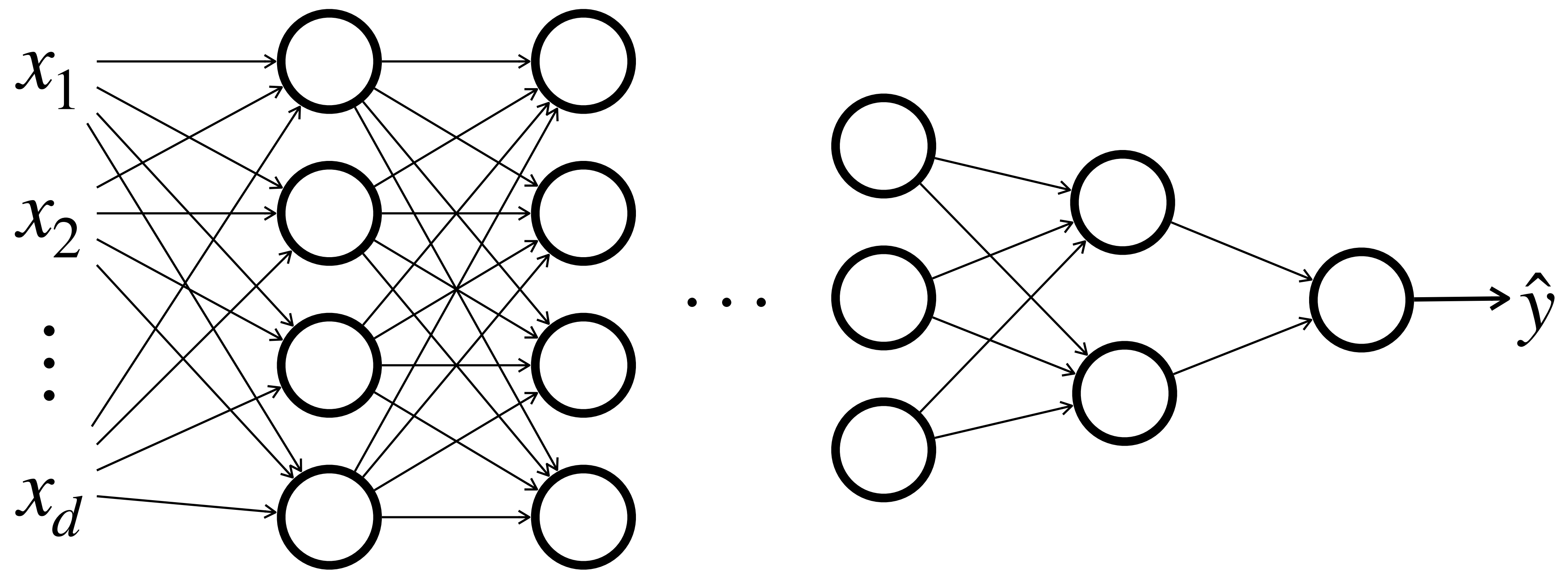
5 camadas ocultas

RNA de 6 camadas (profunda)



Redes Neurais Artificiais Profundas

RNA de L camadas



Para um exemplo $\mathbf{x}$:

$$\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = g(\mathbf{z}^{[1]})$$
$$\mathbf{z}^{[2]} = W^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$
$$\mathbf{a}^{[2]} = g(\mathbf{z}^{[2]})$$
$$\cdots$$
$$\mathbf{z}^{[L]} = W^{[L]}\mathbf{a}^{[L-1]} + b^{[L]}$$
$$\hat{y} = \sigma(\mathbf{z}^{[L]})$$

Regra geral:

$$\mathbf{z}^{[l]} = W^{[l]}\mathbf{a}^{[l-i]} + \mathbf{b}^{[l]}$$
$$\mathbf{a}^{[l]} = g(\mathbf{z}^{[l]})$$

Vetorizado

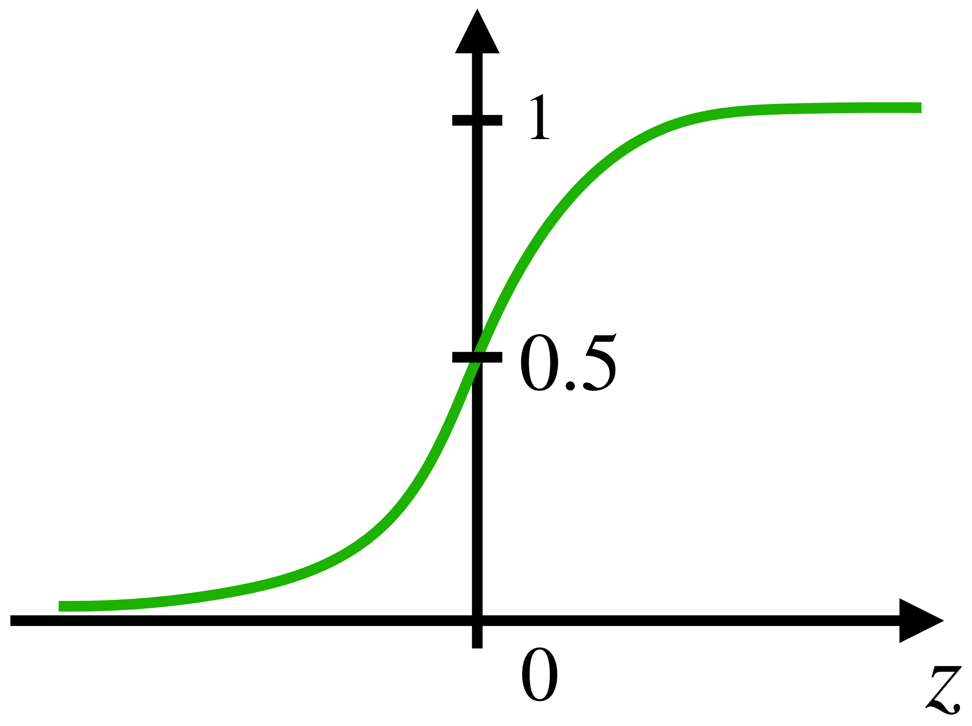
$$Z^{[l]} = W^{[l]}A^{[l-1]} + \mathbf{b}^{[l]}$$
$$\mathbf{A}^{[l]} = g(Z^{[l]})$$

$$A^{[0]} = X$$
$$A^{[L]} = \hat{Y}$$

UFV

Funções de ativação na camada de saída

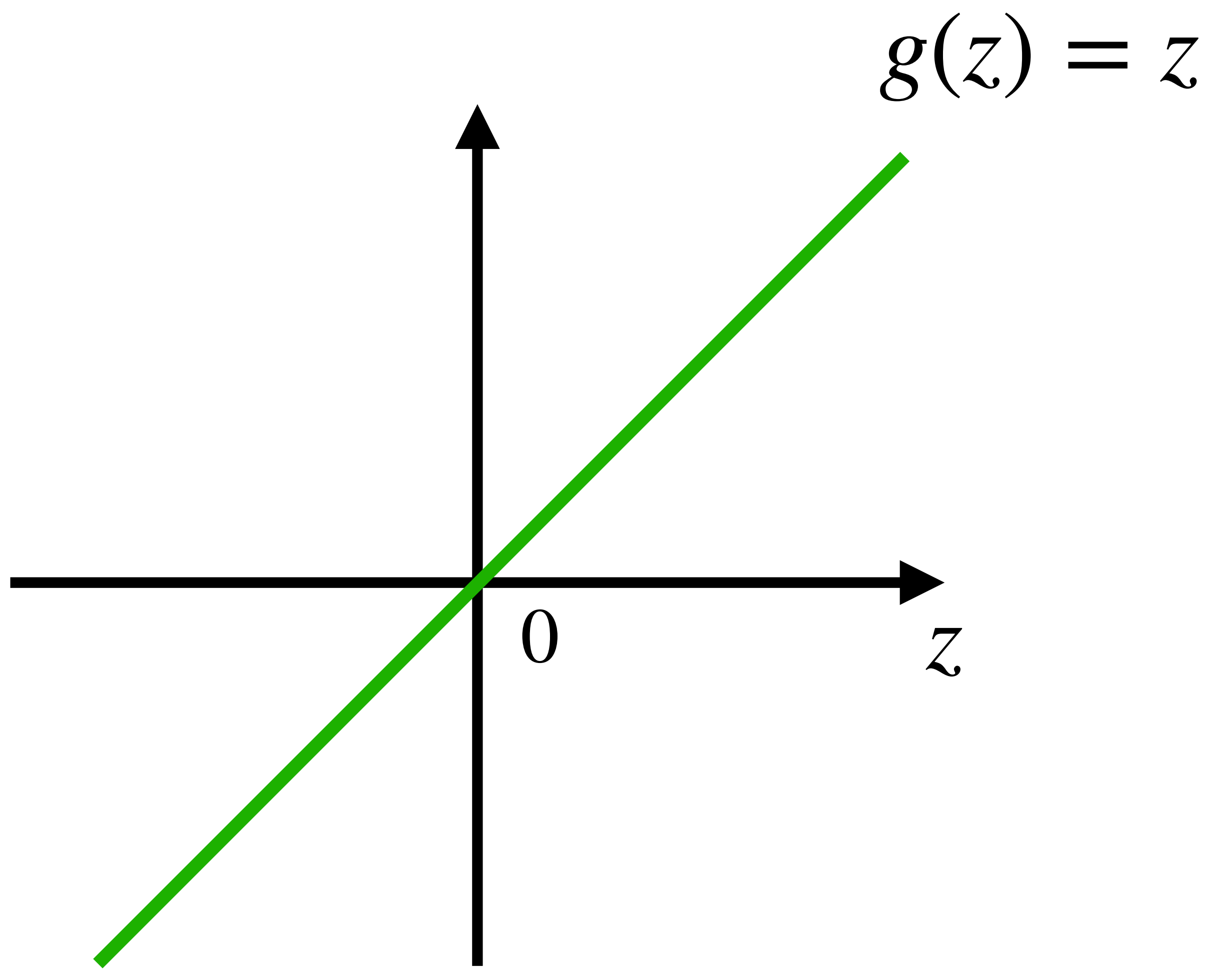
Logística (sigmoide)



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

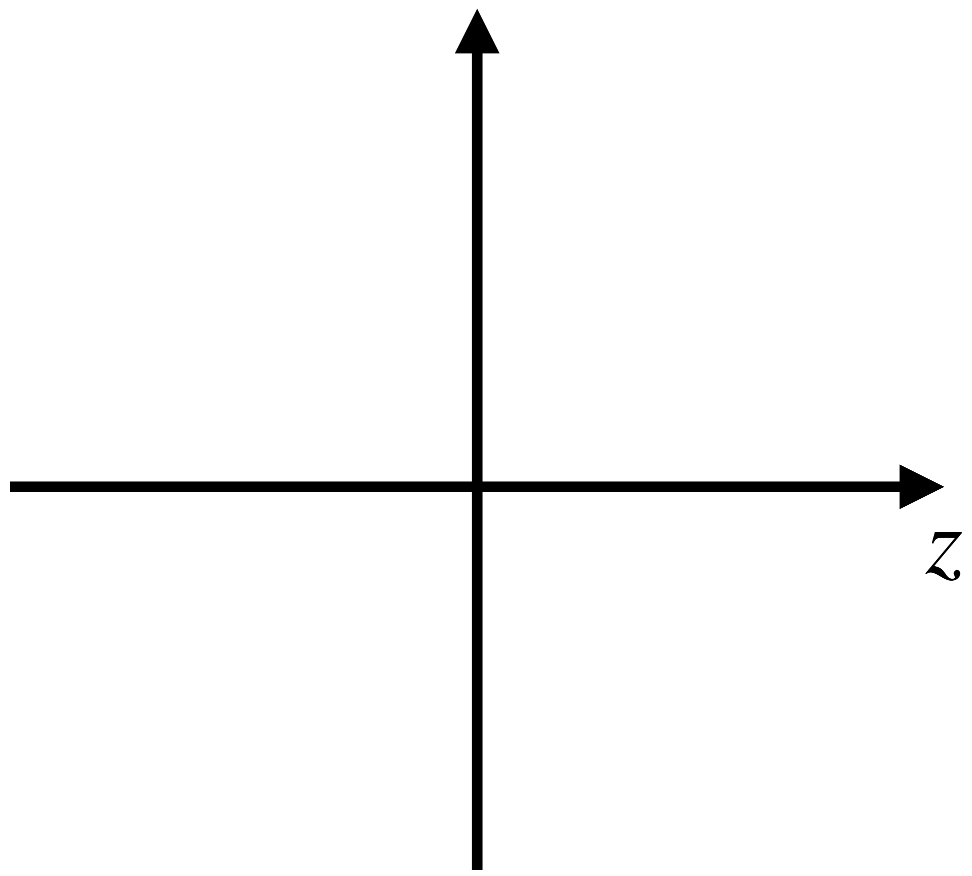
Classificação Binária

Linear



Regressão

?



Classificação Multiclasse

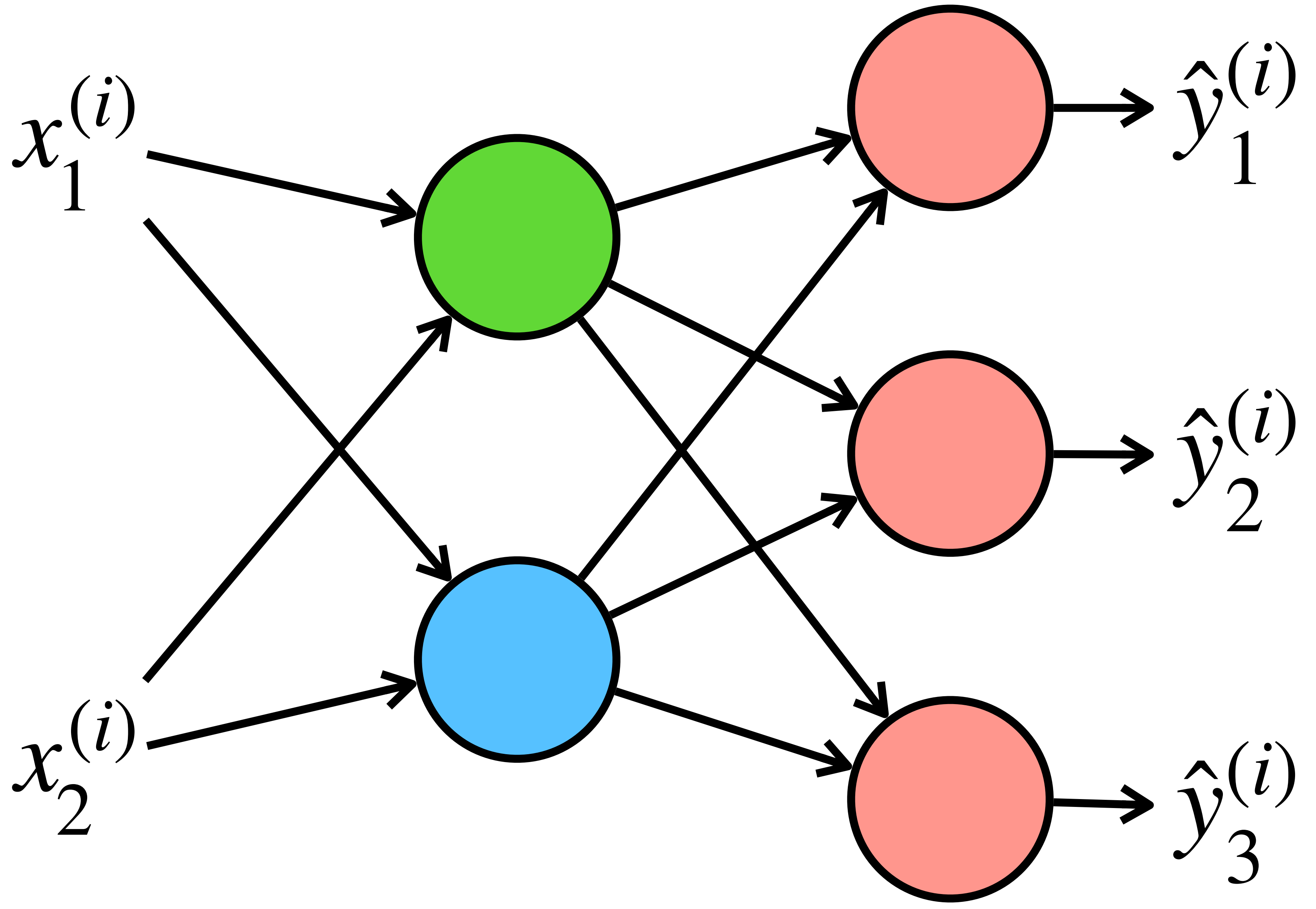
Função de ativação softmax para classificação multiclasse

Hipótese

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

Softmax

$$g(z) = \frac{e^z}{\sum_{j=1}^{C} e_j^z}$$



$$Z^{[2]} = \begin{bmatrix} 5 \\ 2 \\ -1 \end{bmatrix} \qquad e^z = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \end{bmatrix}$$

$$\sum_{j=i}^{C} e_i^z = 156.17$$

$$\hat{y}^{(i)} = \begin{array}{cl} \begin{bmatrix} 0.531 \\ 0.238 \\ 0.229 \end{bmatrix} & \begin{array}{l} \text{Classe 0} \\ \text{Classe 1} \\ \text{Classe 2} \end{array} \end{array}$$

Distribuição de Probabilidades

UFV

Próxima aula

A7: MLP em Numpy

Aula prática sobre implementação de redes neurais profundas com Numpy.