

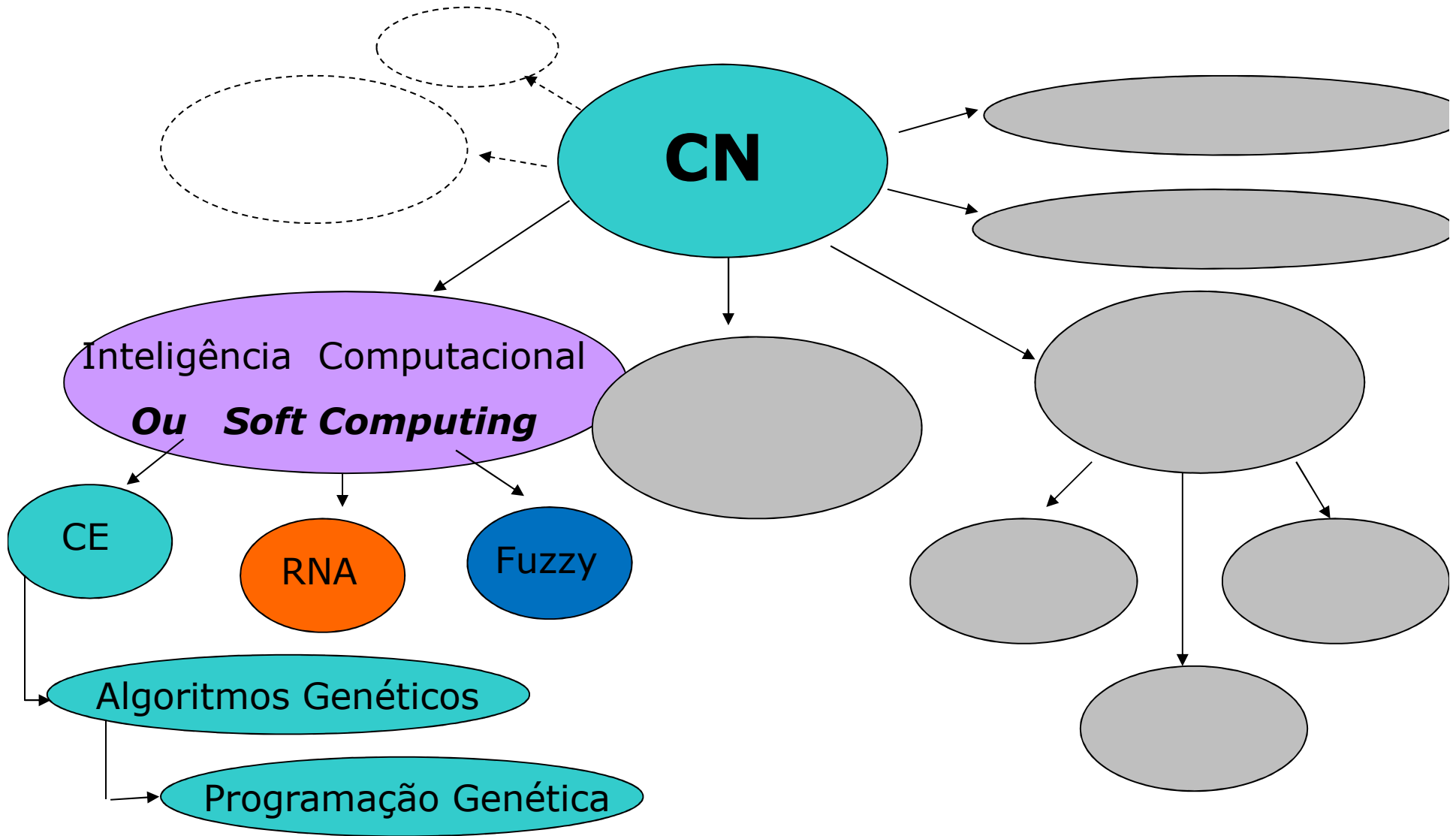
---

# SISTEMAS INTELIGENTES 1

## **Computação Natural**

(Soft Computing)

# Esquema Geral da Computação Natural

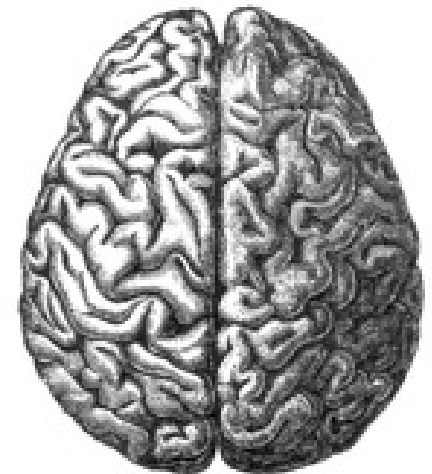
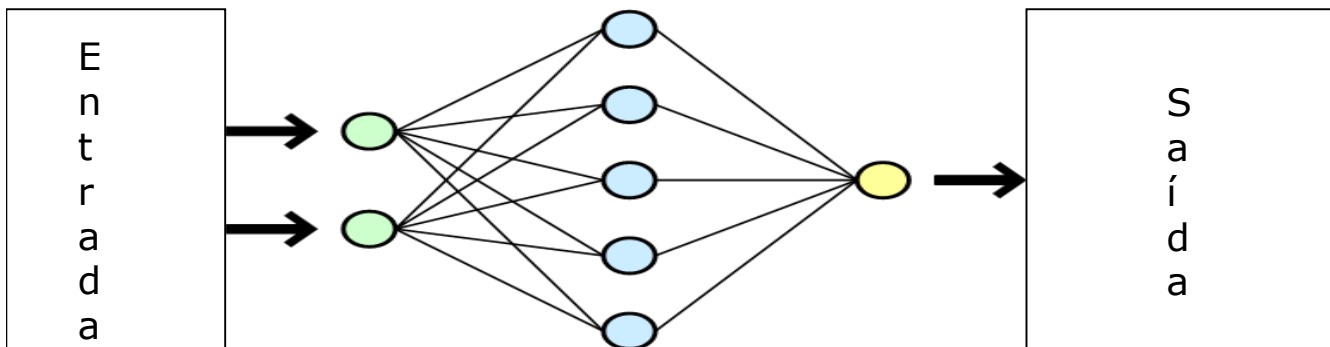


# Redes Neurais Artificiais

---

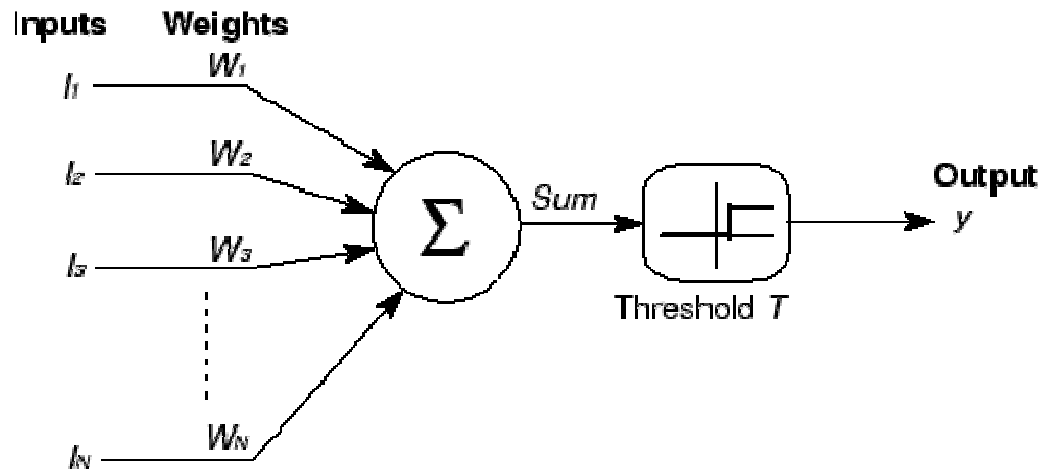
Redes Neurais (RN): Inspiradas no Funcionamento do Cérebro Humano

Uma coleção **massivamente paralela** de **unidades** de processamento pequenas e **simples**, onde as **interligações** formam a maior parte da “**inteligência**” da rede



# Redes Neurais: modelo do neurônio

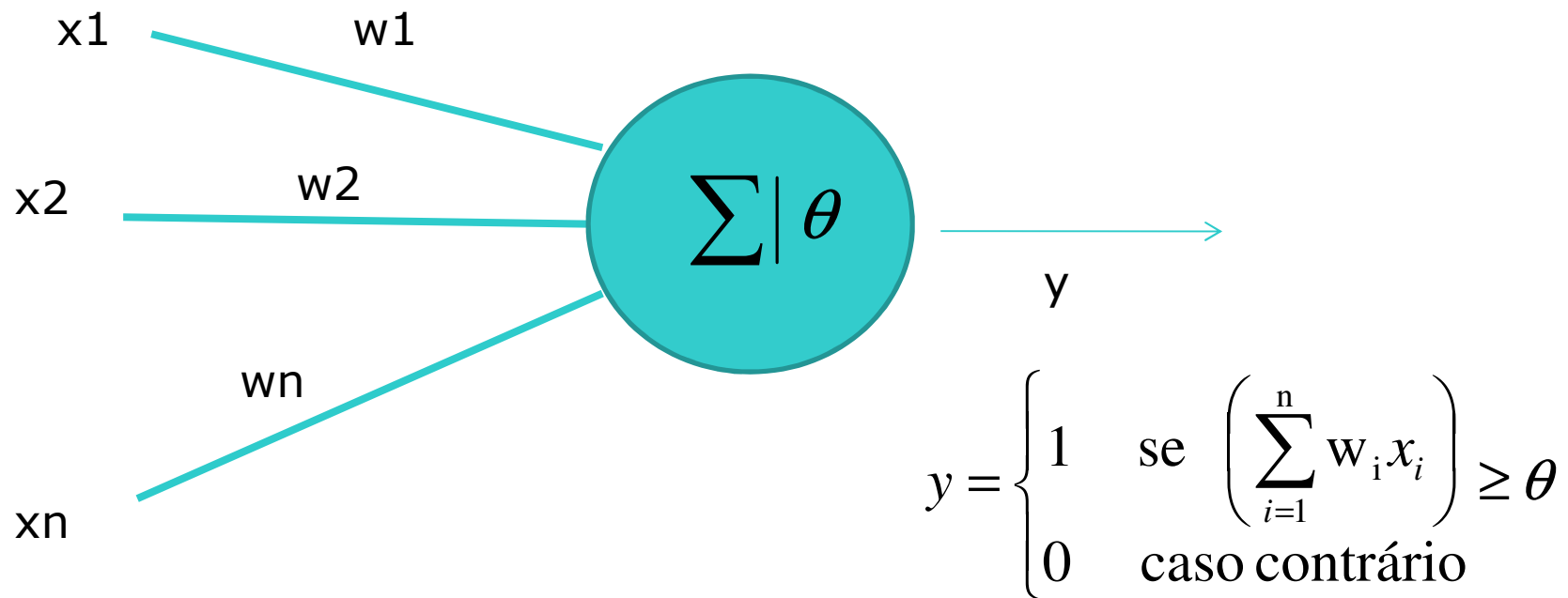
Os primeiros trabalhos na área datam de 1943: McCulloch e Pitts desenvolveram o primeiro modelo matemático do neurônio



$$y = \text{Sinal}(\text{Soma}(I_j * W_j)), j = 1, \dots, N$$

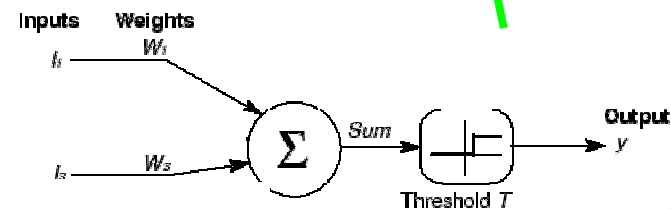
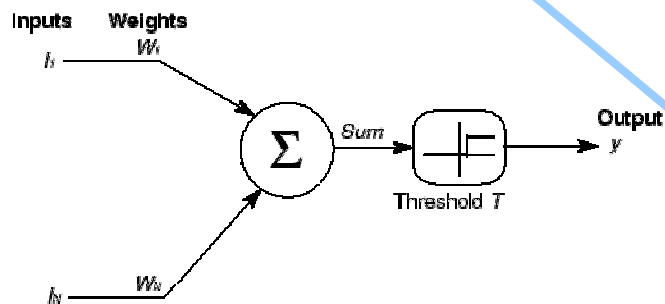
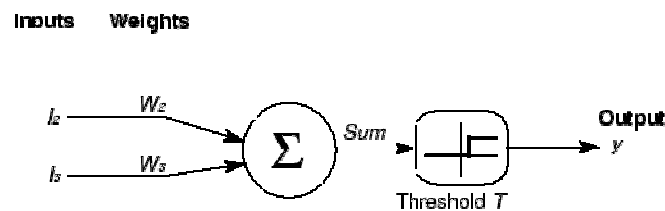
# Redes Neurais: modelo do neurônio MCP

Neurônio: unidade de processamento simples com limiar (função de ativação degrau)



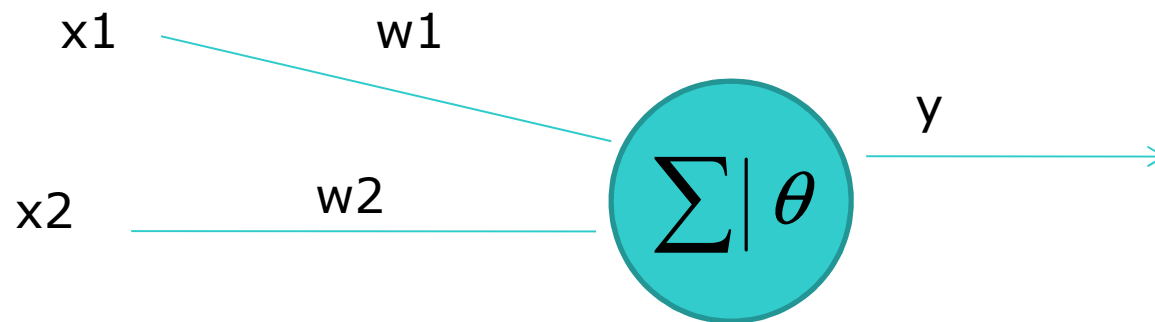
# RNAs: Neurônio MCP (partição do espaço)

Neurônios com diferentes pesos: partição do espaço de entradas



## RNAs: Neurônio MCP (partição do espaço)

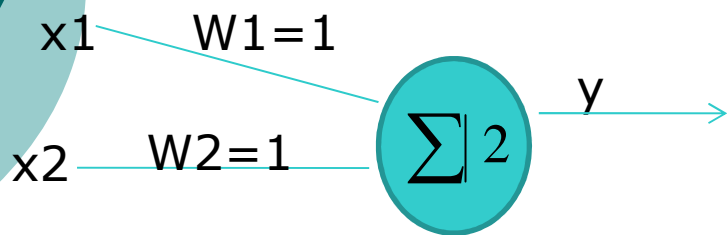
Considerando o caso particular de 1 neurônio com 2 entradas



$$y = \begin{cases} 1 & \text{se } \left( \sum_{i=1}^n w_i x_i \right) \geq \theta \\ 0 & \text{caso contrário} \end{cases} \longrightarrow y = \begin{cases} 1 & \text{se } w_1 x_1 + w_2 x_2 \geq \theta \\ 0 & \text{caso contrário} \end{cases}$$

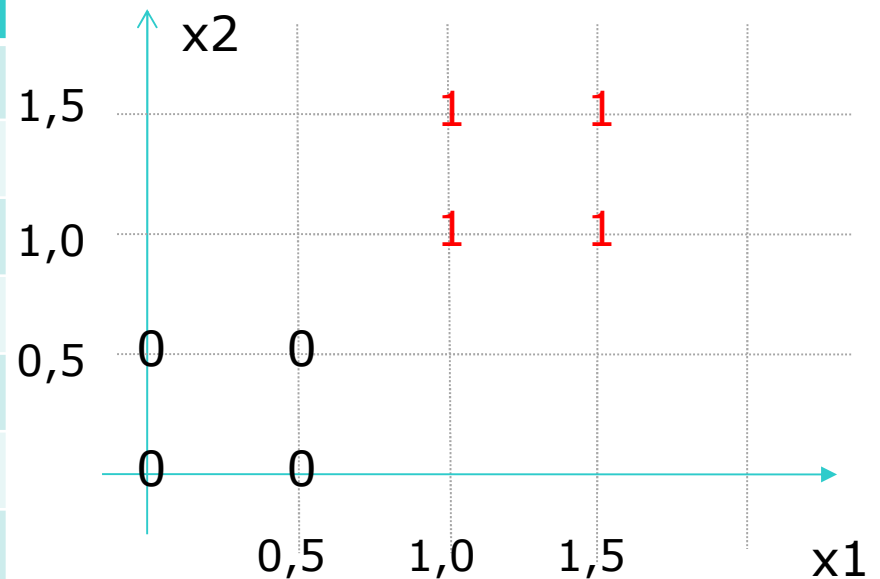
# RNAs: Neurônio MCP (partição do espaço)

Considerando  $w_1=w_2=1$  e  $\Theta=2$



$$y = \begin{cases} 1 & \text{se } 1 * x_1 + 1 * x_2 \geq 2 \\ 0 & \text{caso contrário} \end{cases}$$

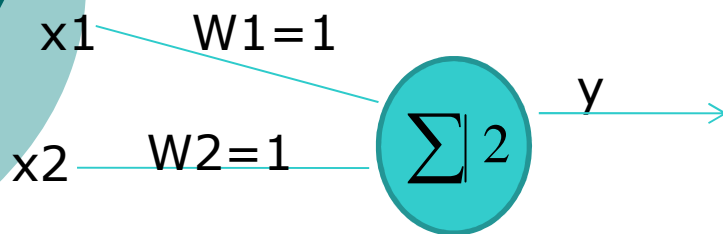
x1	x2	y
0	0	0
0	0,5	0
0,5	0	0
0,5	0,5	0
1	1	1
1	1,5	1
1,5	1	1
1,5	1,5	1





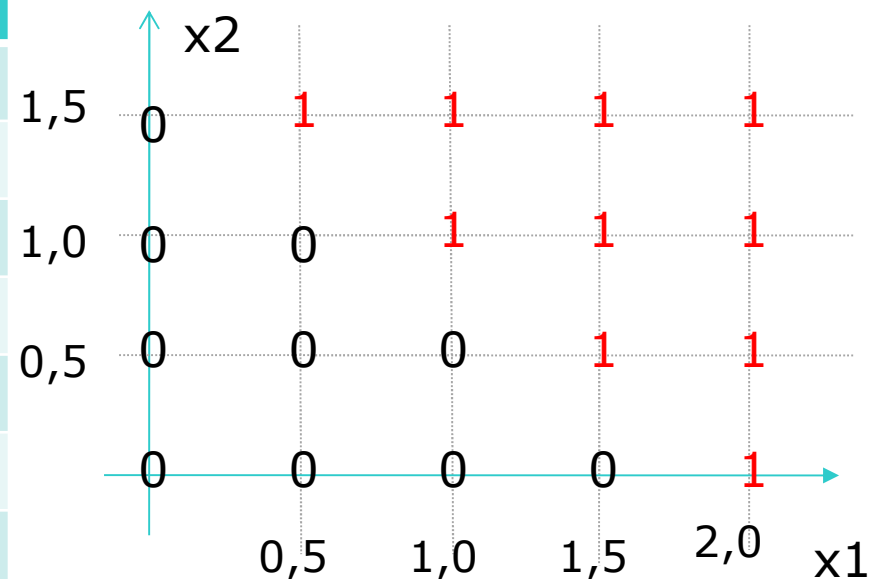
# RNAs: Neurônio MCP (partição do espaço)

Aumentando-se o número de padrões de entrada (x1,x2)



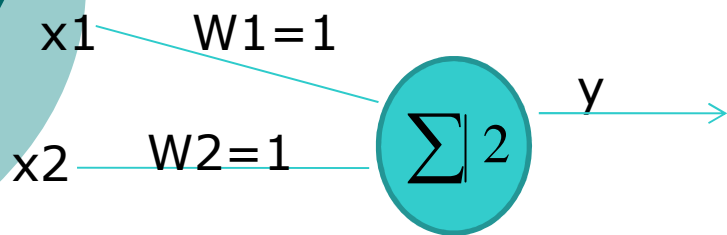
$$y = \begin{cases} 1 & \text{se } 1 * x_1 + 1 * x_2 \geq 2 \\ 0 & \text{caso contrário} \end{cases}$$

x1	x2	y
0	0	0
...	...	...
1,0	0	0
1,5	0	0
0,5	1,5	1
1,0	1,0	1
1,5	0,5	1
2,0	1,5	1



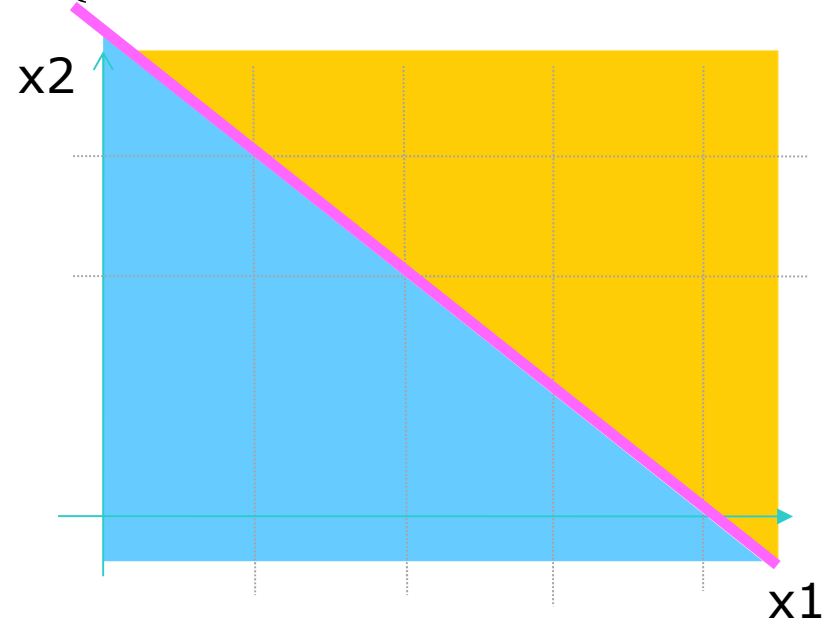
# RNAs: Neurônio MCP (partição do espaço)

Para infinitos padrões de entrada  $(x_1, x_2)$



$x_1$	$x_2$	$y$
0,99	0,80	0
...	...	...
1,99	0	0
0,5	1,52	1
1,2	1,0	1
1,57	0,58	1
2,01	1,5	1
...	...	...

$$y = \begin{cases} 1 & \text{se } 1 * x_1 + 1 * x_2 \geq 2 \\ 0 & \text{caso contrário} \end{cases}$$



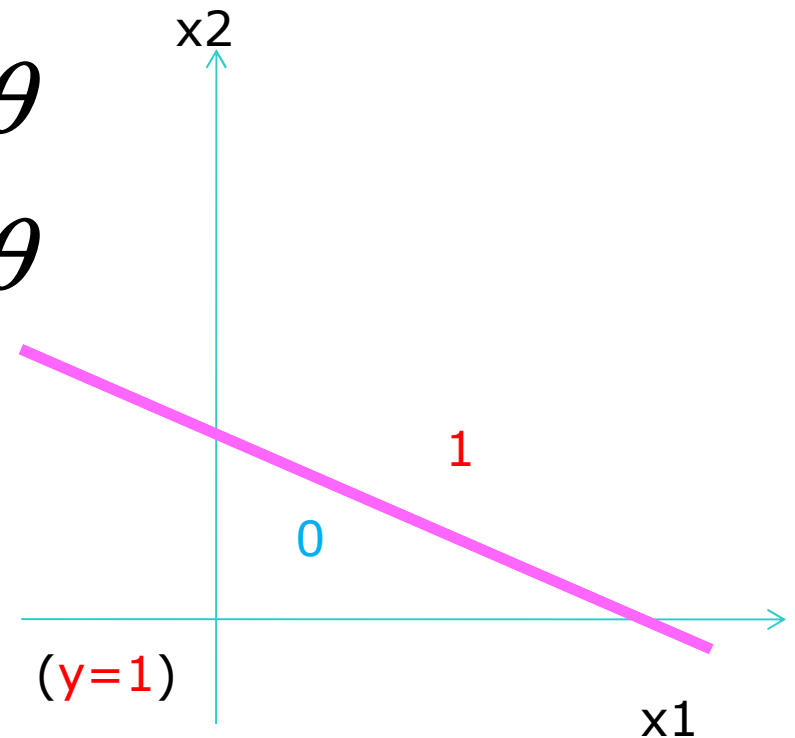
## RNAs: Neurônio MCP (partição do espaço)

Saída ( $y$ ) é uma função da combinação linear das entradas ( $\mathbf{x}$ )

$$y = \begin{cases} 1 & \text{se } w_1x_1 + w_2x_2 \geq \theta \\ 0 & \text{se } w_1x_1 + w_2x_2 < \theta \end{cases}$$

Na condição **limite** (mudança do grau de ativação) temos:

$$w_1x_1 + w_2x_2 = \theta \begin{cases} \rightarrow \Sigma > \theta \text{ (} y=1 \text{)} \\ \rightarrow \Sigma < \theta \text{ (} y=0 \text{)} \end{cases}$$

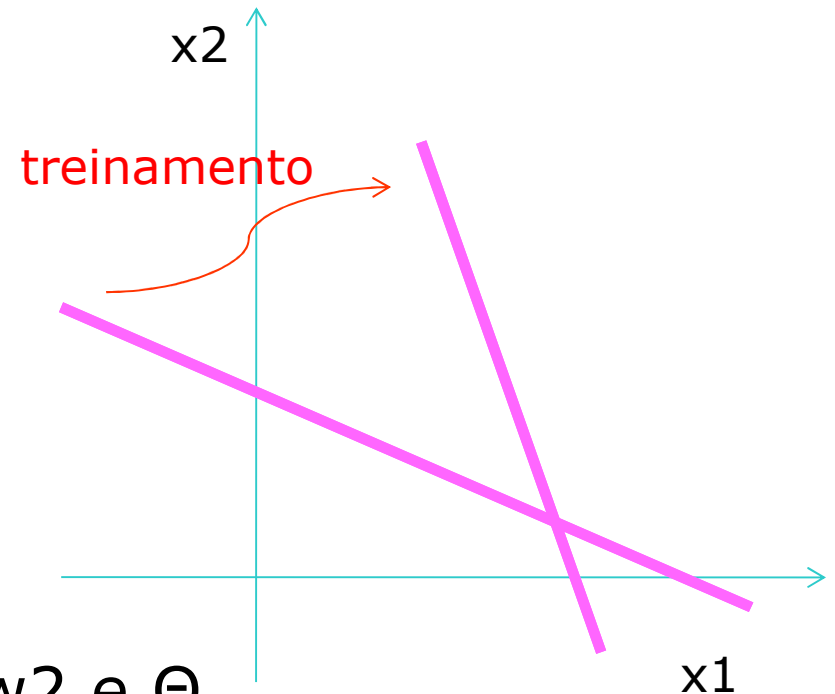


## RNAs: Pesos do Neurônio MCP (treinamento)

$$w_1 x_1 + w_2 x_2 = \theta$$

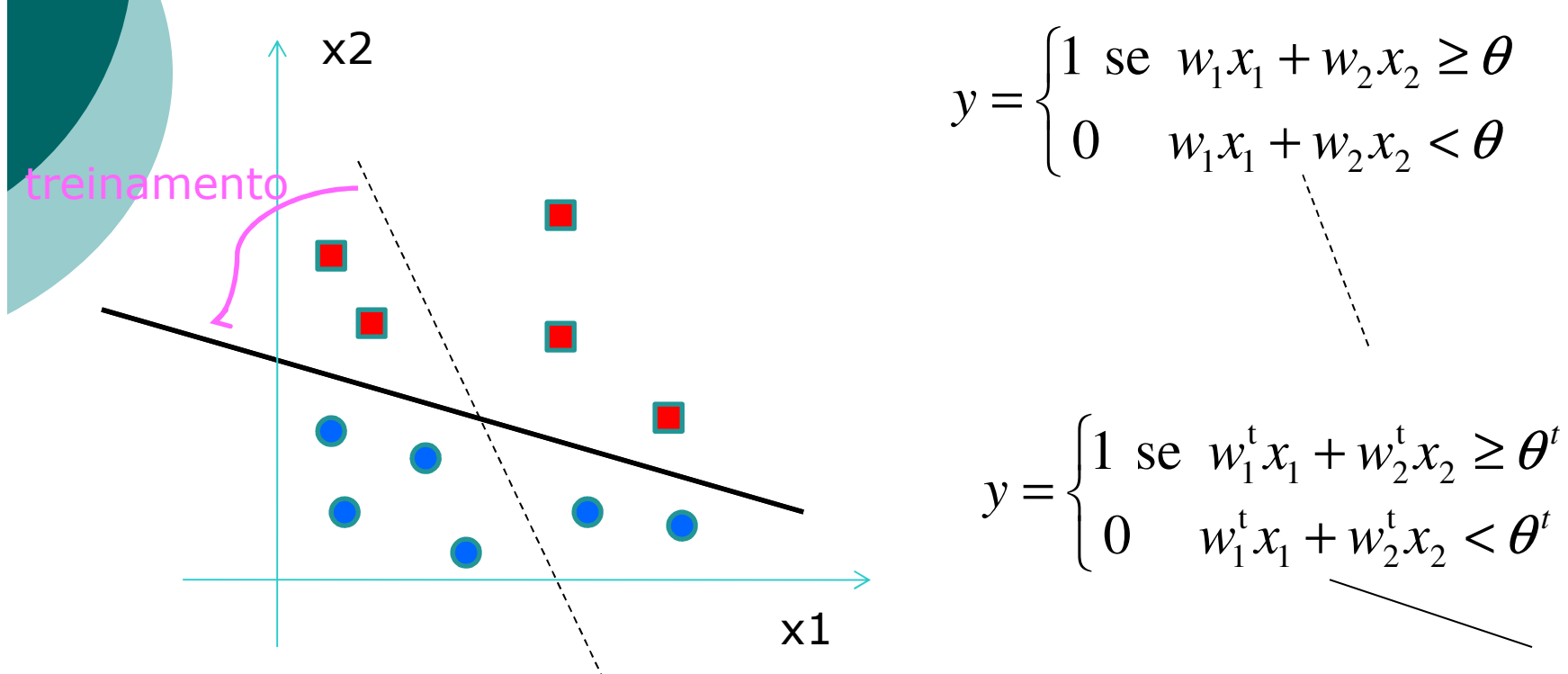
**ou**

$$x_2 = -\left(\frac{w_1}{w_2}\right)x_1 + \left(\frac{\theta}{w_2}\right)$$



A alteração dos parâmetros  $w_1$ ,  $w_2$  e  $\theta$  (**treinamento**) modifica a posição da reta e portanto da partição no espaço

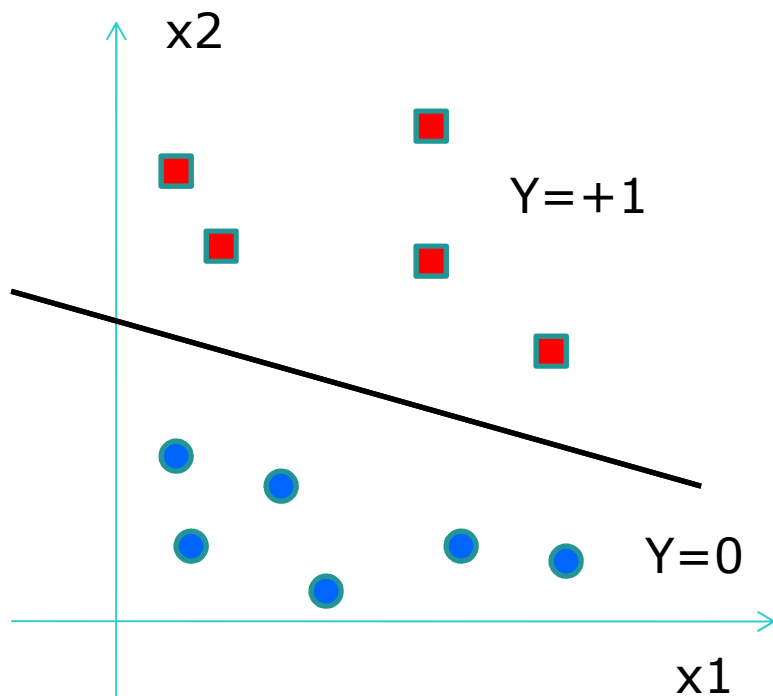
# RNAs: Neurônio MCP para Classificação



A alteração dos parâmetros  $\mathbf{w}$  e  $\theta$  (treinamento) modifica a posição da reta e portanto da partição no espaço de entrada.

# Problema de Classificação

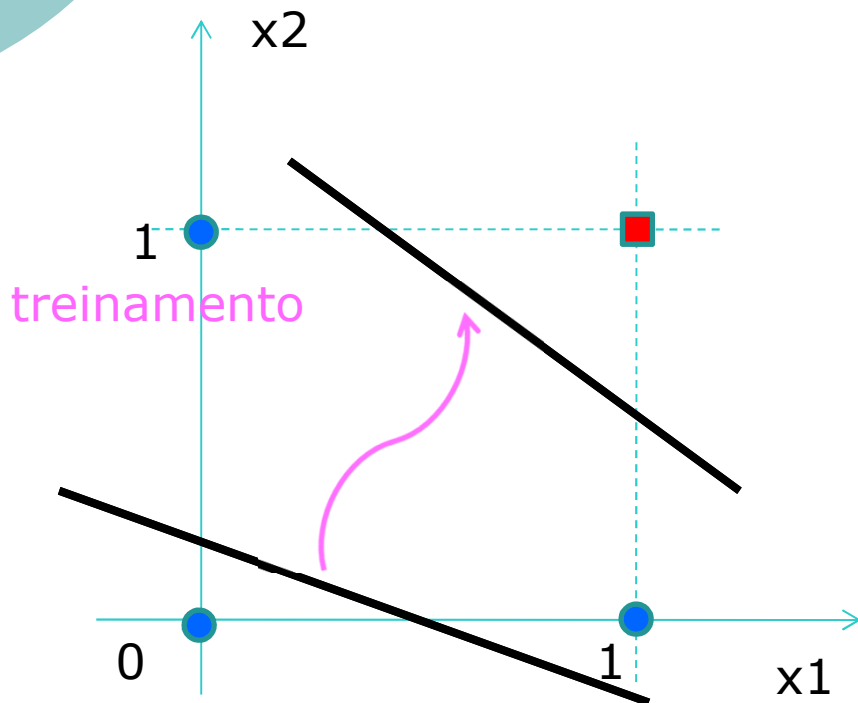
---



Supondo que o problema fosse linearmente separável, como seria para o caso do Resgate de robôs?

# RNAs: Neurônios MCP(resolução de problemas)

Os modelos iniciais se restringiam à solução de problemas simples (linearmente separáveis) como o AND Lógico.

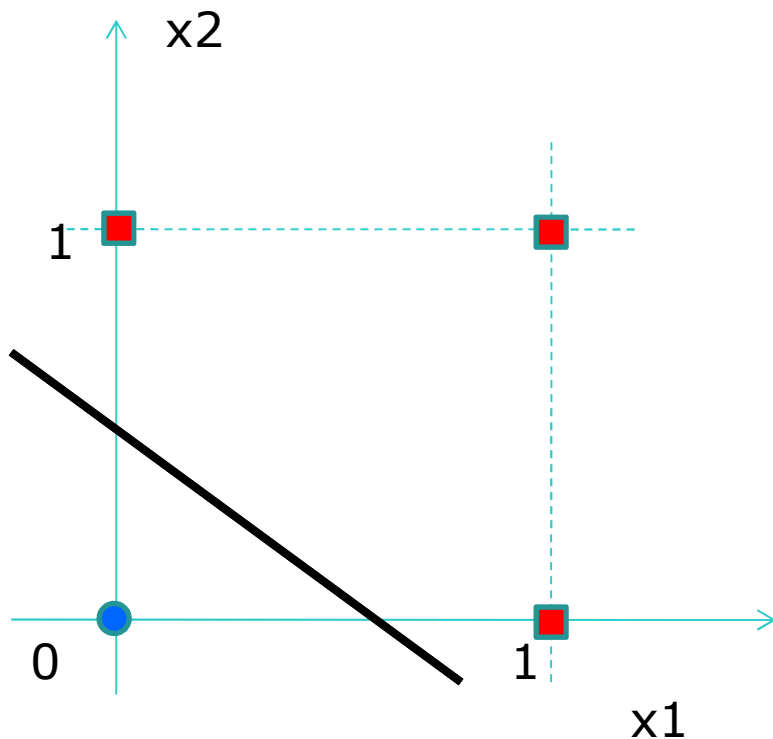


AND Lógico

x1	x2	y
1	1	1
0	1	0
1	0	0
0	0	0

# RNAs: Neurônios (resolução de problemas)

OR lógico: linearmente separável



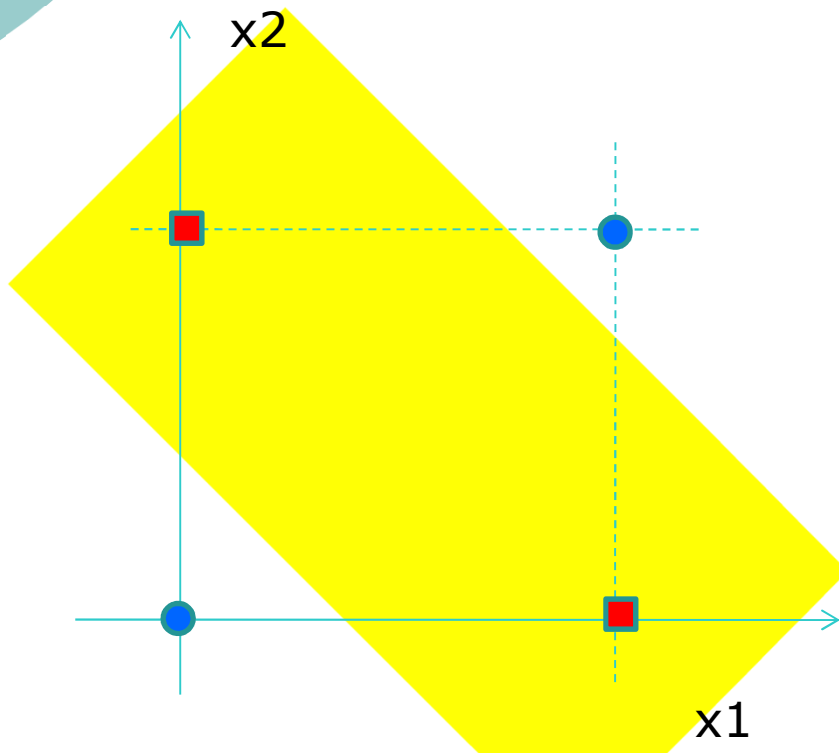
OR Lógico

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1



# RNAs: Neurônio MCP (resolução de problemas)

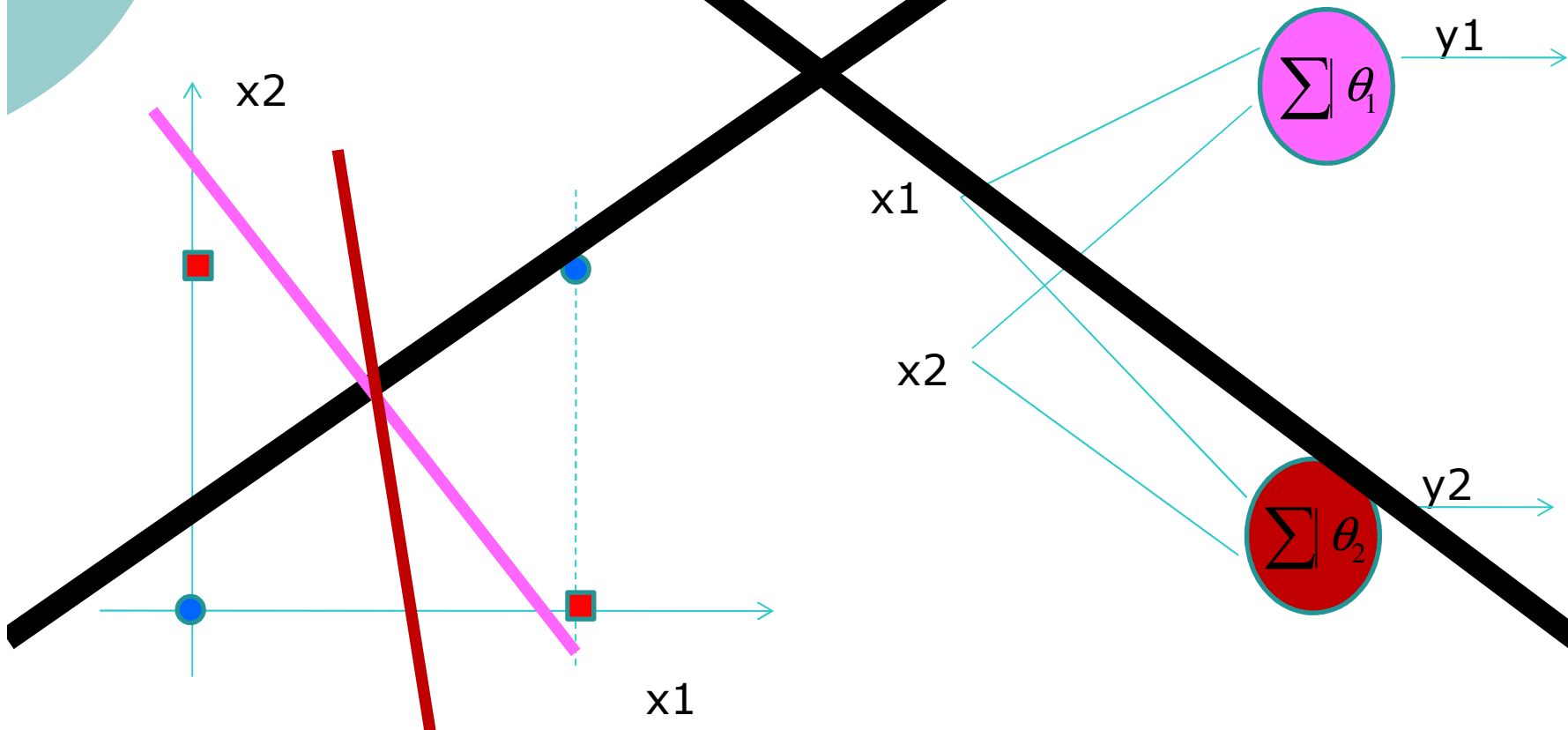
XOR lógico: não linearmente separável



XOR Lógico		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

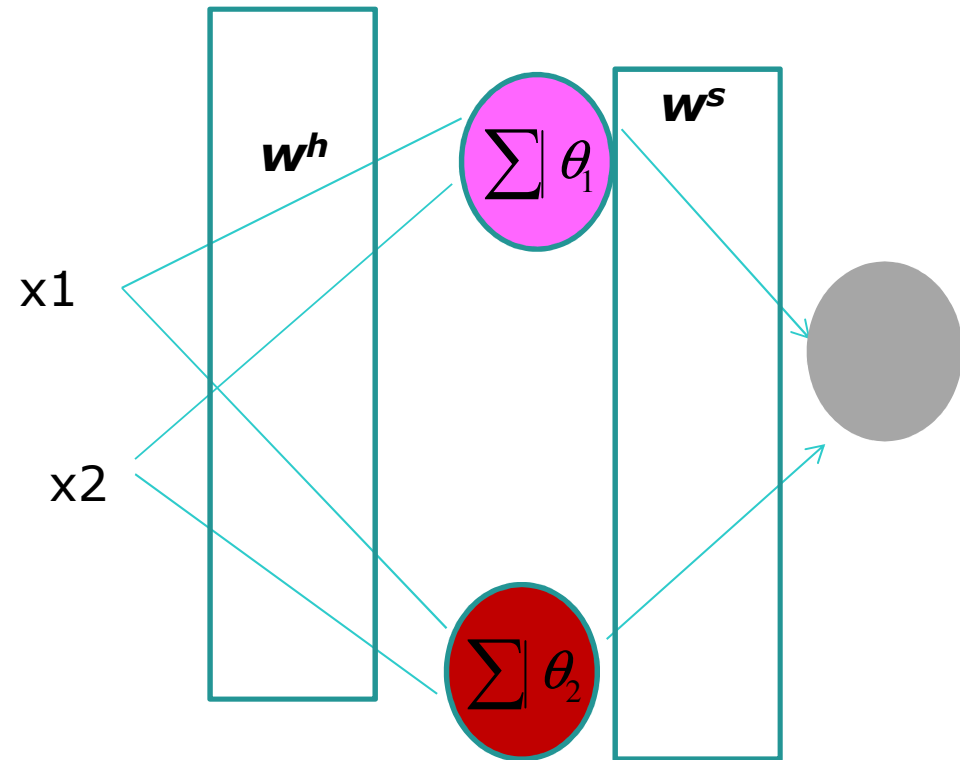
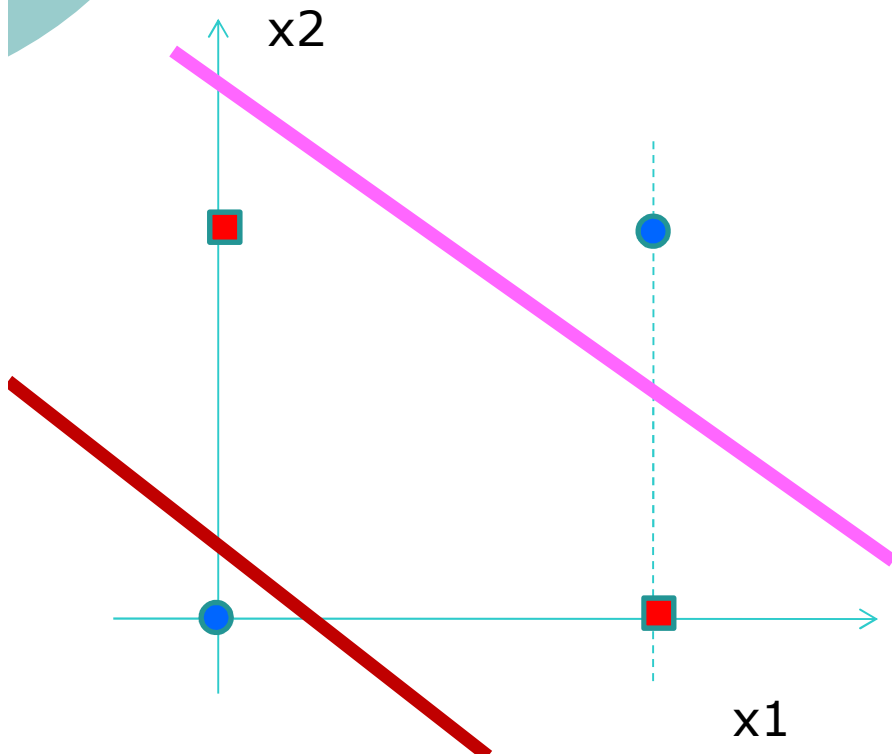
# RNAs: Neurônio MCP (resolução de problemas)

XOR lógico: Mais neurônios ???



# RNAs: Neurônio MCP (resolução de problemas)

XOR lógico: Mais camadas?  
Como ajustar os pesos  $w^h$  ?





# RNAs: Resolução de problemas

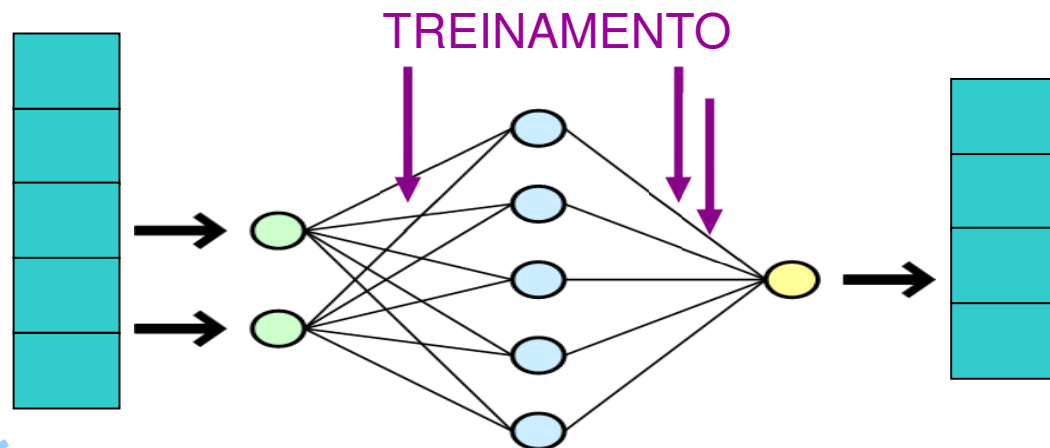
---

A descoberta da **limitação** dos neurônios (ou redes de uma única camada) na resolução de **problemas não-linearmente separáveis** trouxe um desânimo à comunidade.

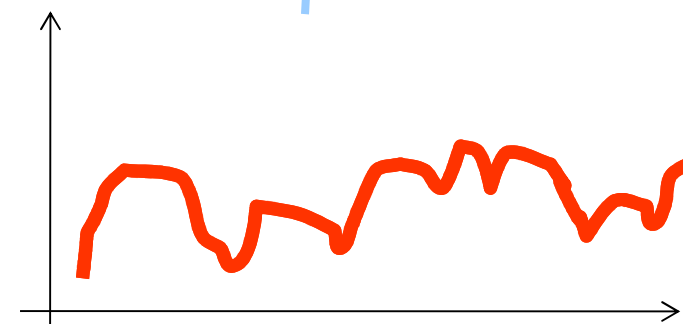
Este período de descrença (conhecido como **idade das trevas**) teve início em 1969 e durou até o princípio dos anos 80.

O **fim da idade das trevas** foi marcado por fatos importantes como a descoberta de algoritmos de **treinamento** para redes **multicamadas** – backpropagation.

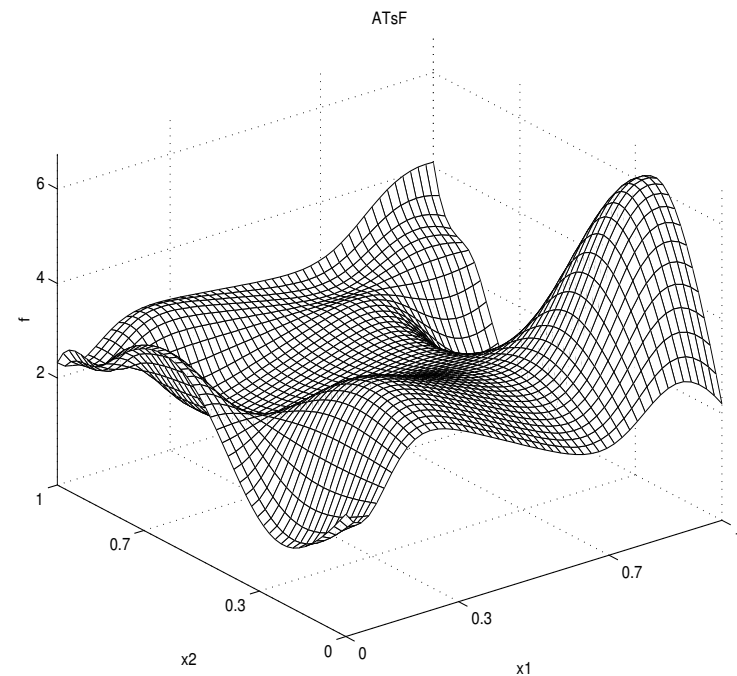
# Solução por Redes Neurais (RN)



Classificação



Previsão de Séries Temporais



Aproximação de Funções



# Redes Neurais Artificiais

---

- Modelo do neurônio
- Topologia da rede
- Treinamento



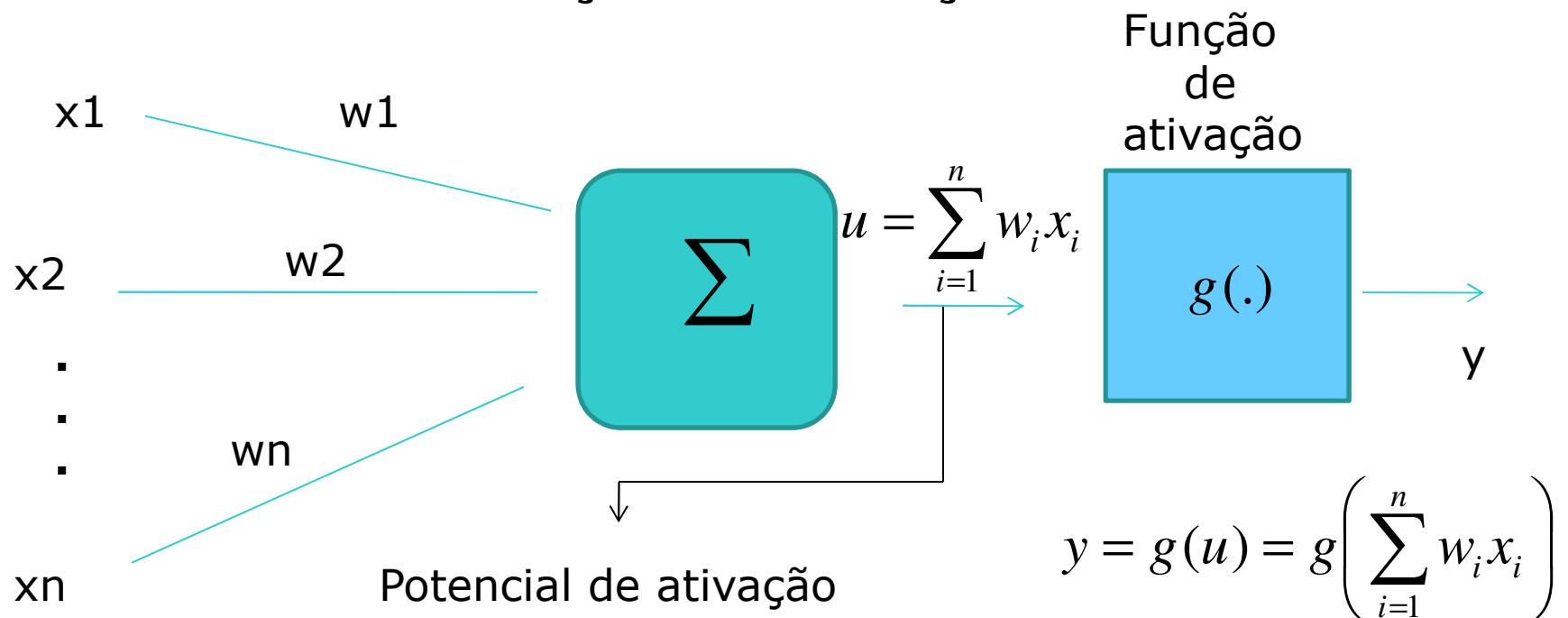
# Redes Neurais Artificiais

---

- **Modelo do neurônio: função de ativação**
- Arquitetura da rede
- Treinamento

# Redes Neurais: Funções de Ativação

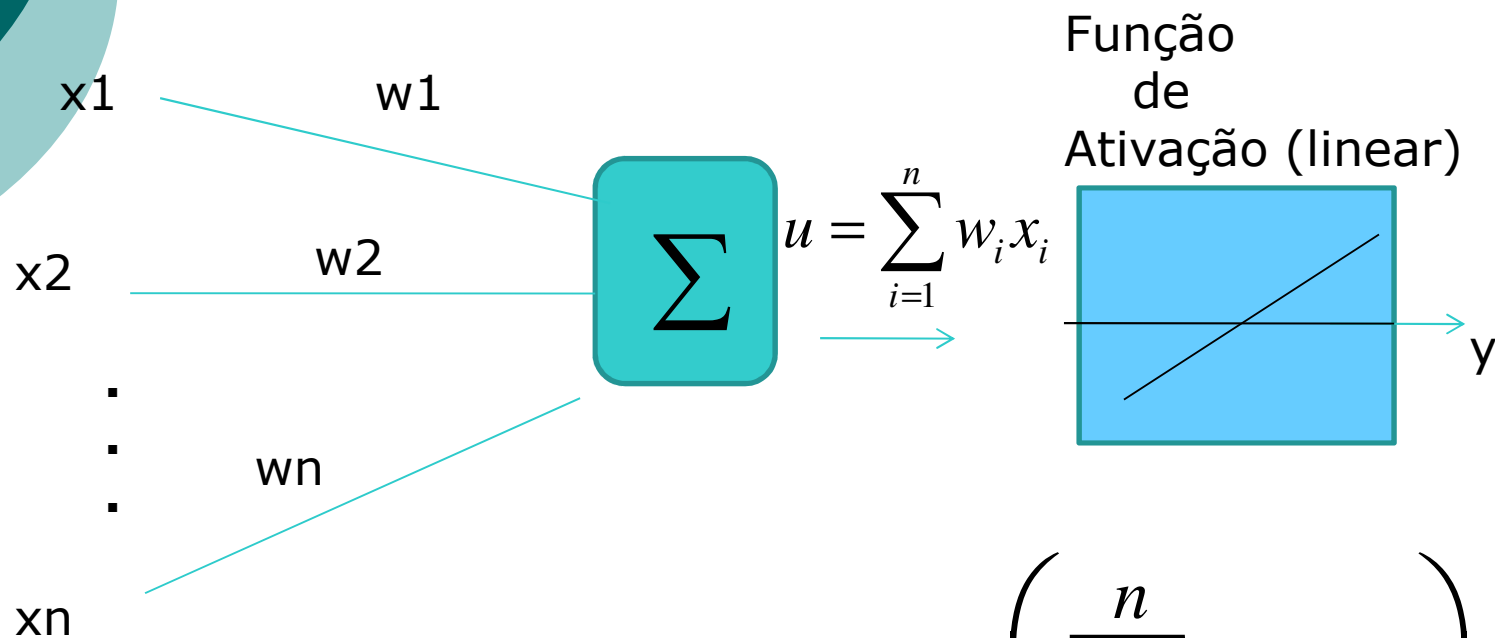
A partir do Modelo MCP original, foram derivados vários outros modelos de neurônios que permitem uma saída qualquer através de **diferentes funções de ativação**





# Redes Neurais: Funções de Ativação

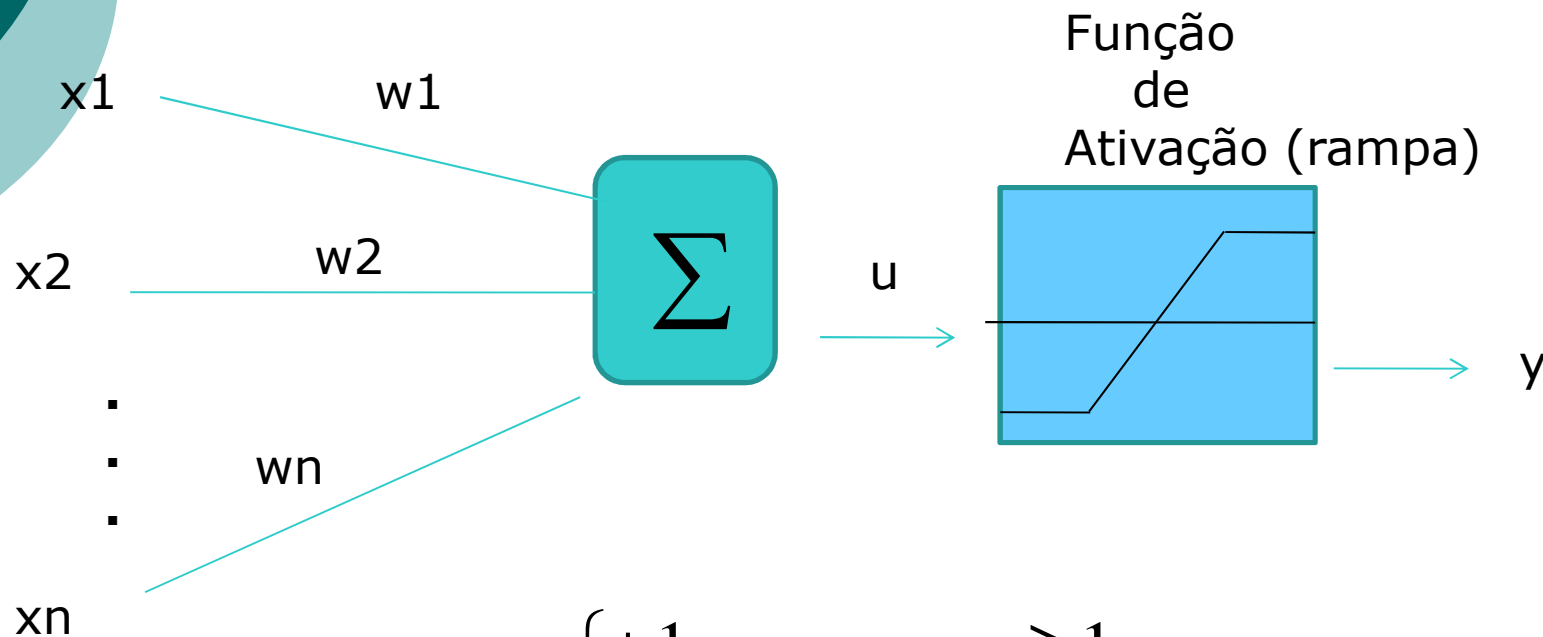
$f$  = função linear



$$y = \gamma \left( \sum_{i=1}^n w_i x_i \right) = \gamma u$$

# Redes Neurais: Funções de Ativação

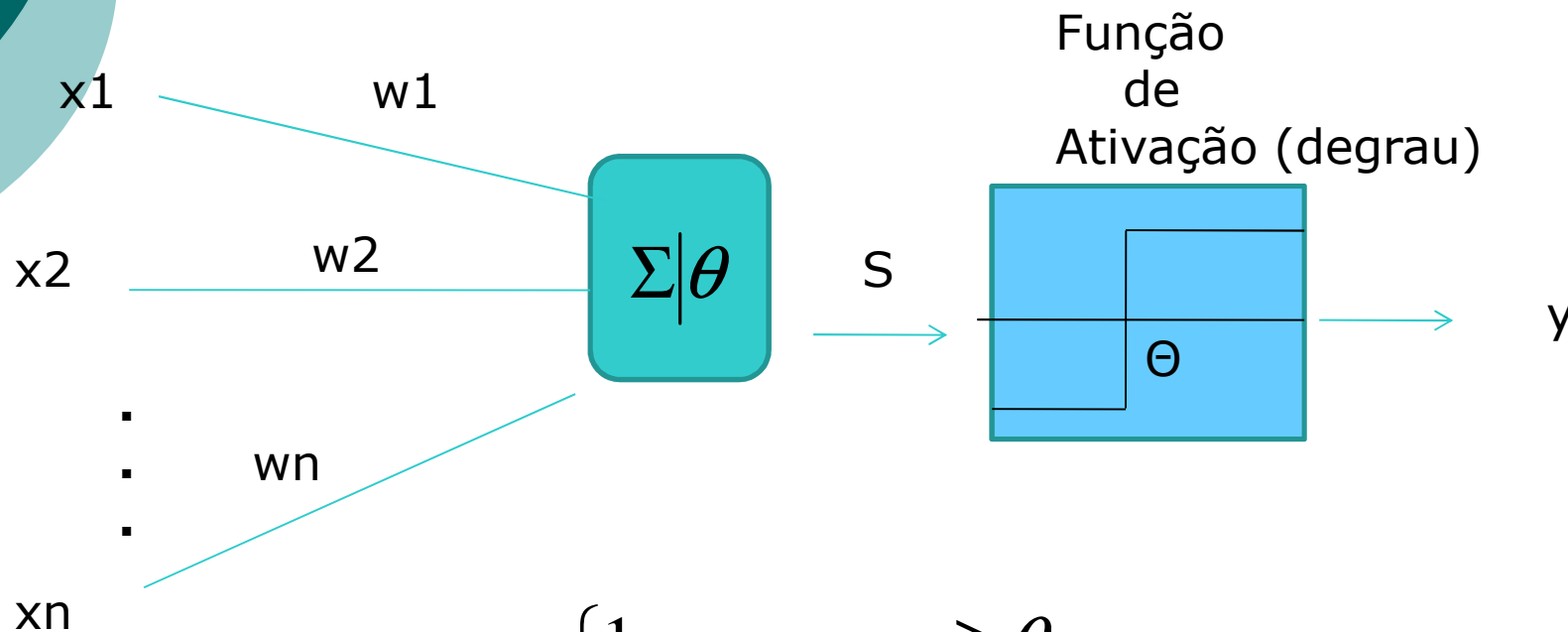
$f$  = função rampa



$$y = \begin{cases} +1 & \text{se } u \geq 1 \\ u & \text{se } |u| < 1, \\ -1 & \text{se } u \leq -1 \end{cases} \quad \text{onde } u = \left( \sum_{i=1}^n w_i x_i \right)$$

# Redes Neurais: Funções de Ativação

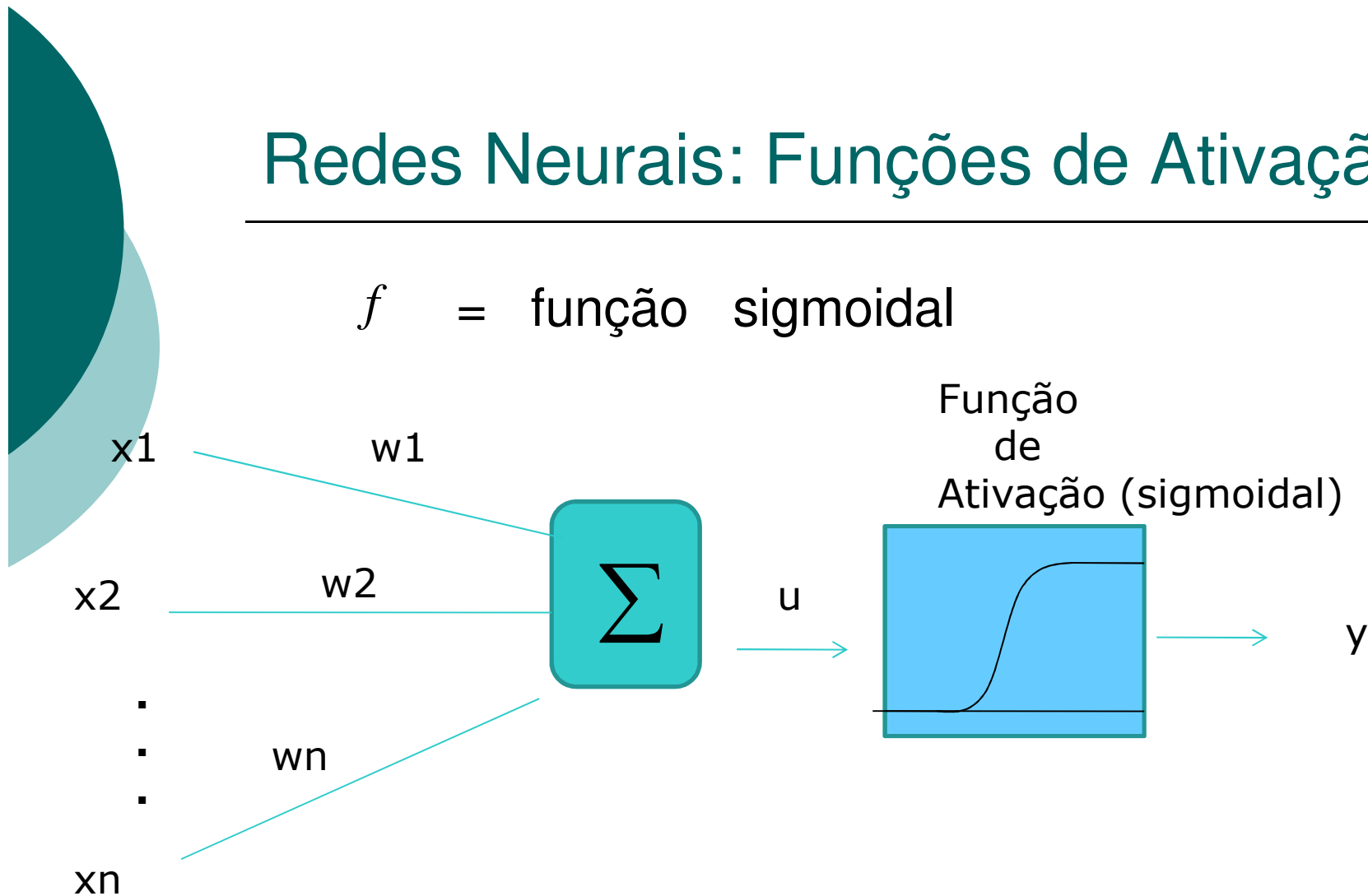
$f$  = função degrau



$$y = \begin{cases} 1 & \text{se } u \geq \theta \\ 0 & \text{se } u < \theta \end{cases}, \quad \text{onde } u = \left( \sum_{i=1}^n w_i x_i \right)$$

# Redes Neurais: Funções de Ativação

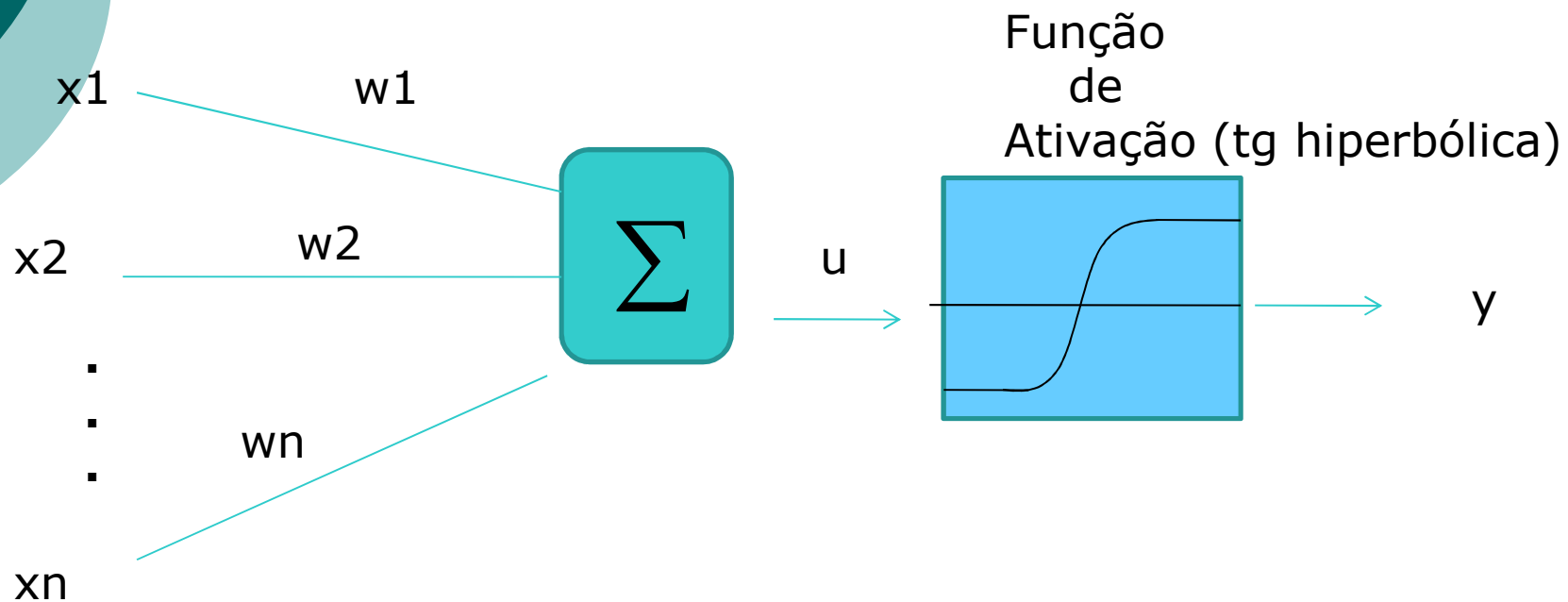
$f$  = função sigmoidal



$$y = \frac{1}{1 + e^{(-u)}} \quad \text{onde} \quad u = \left( \sum_{i=1}^n w_i x_i \right)$$

# Redes Neurais: Funções de Ativação

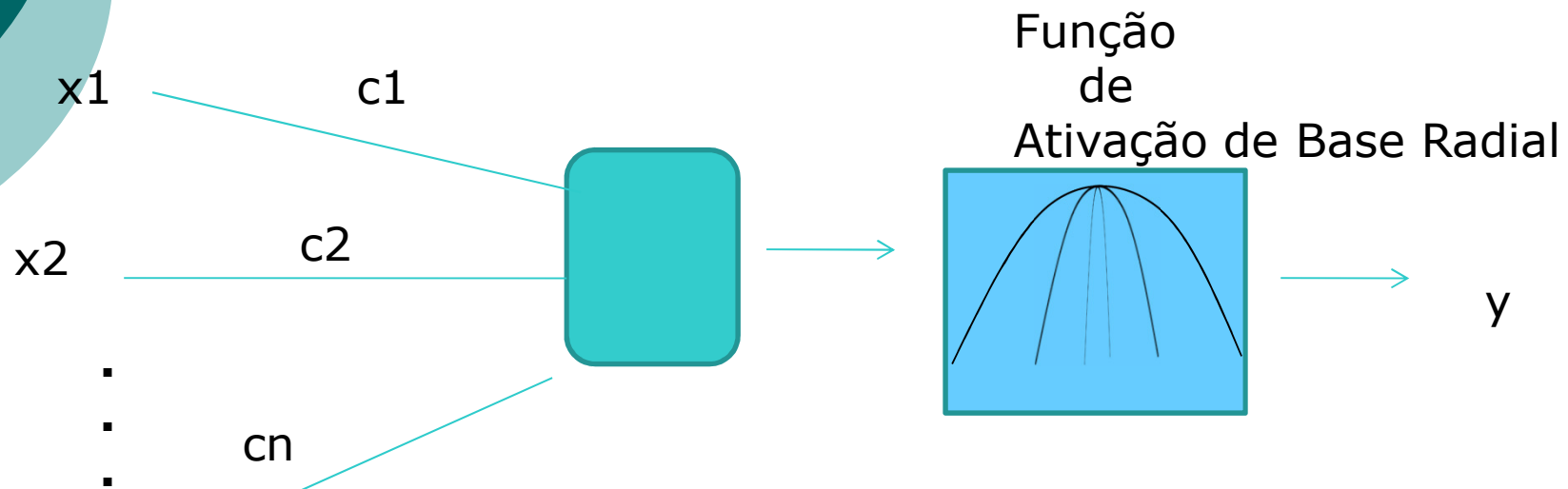
$f$  = função tangente hiperbólica



$$y = \frac{e^{(u)} - e^{(-u)}}{e^{(u)} + e^{(-u)}}, \quad \text{onde } u = \left( \sum_{i=1}^n w_i x_i \right)$$

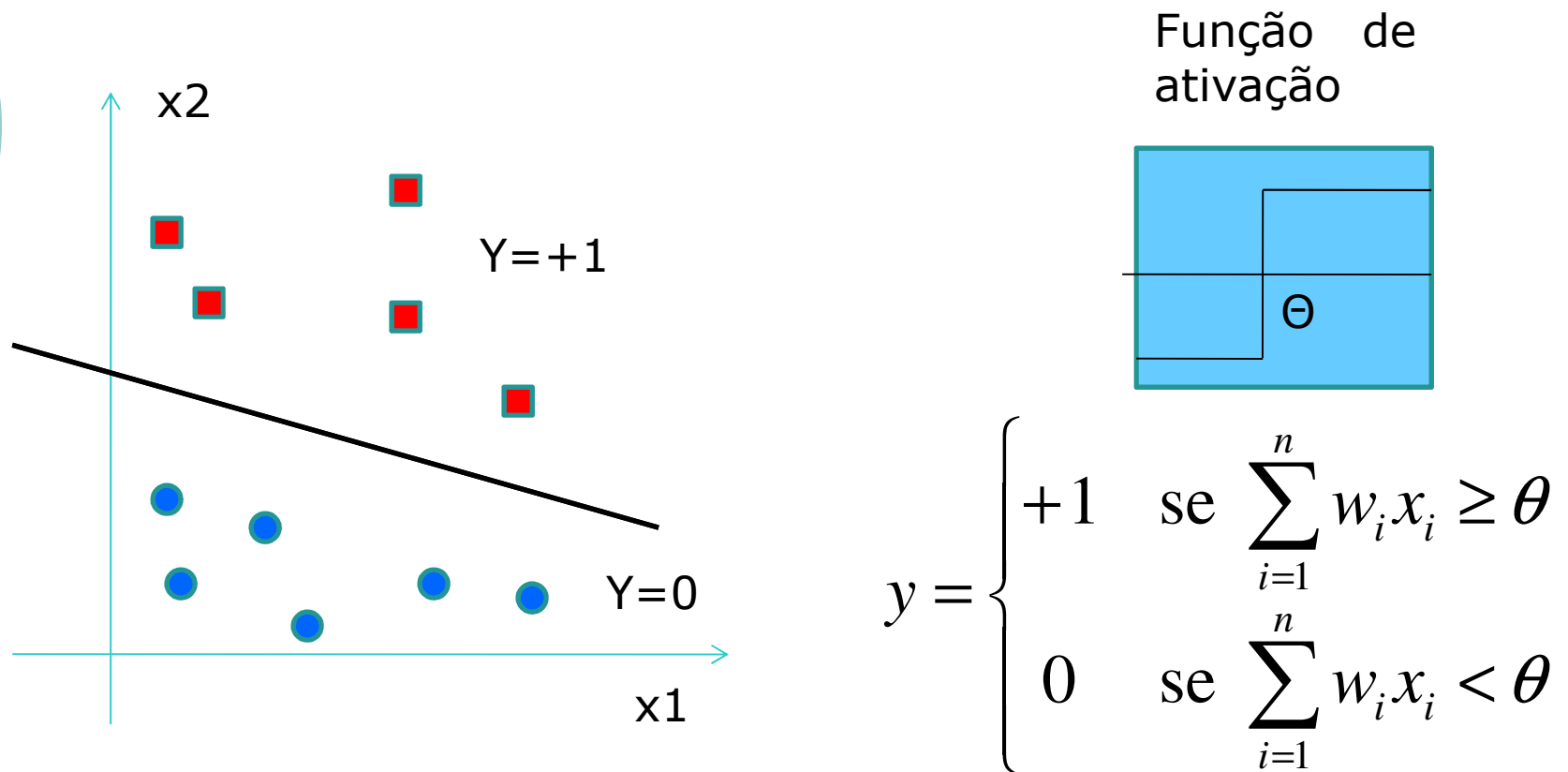
# Redes Neurais: Funções de Ativação

$f$  = função de base radial



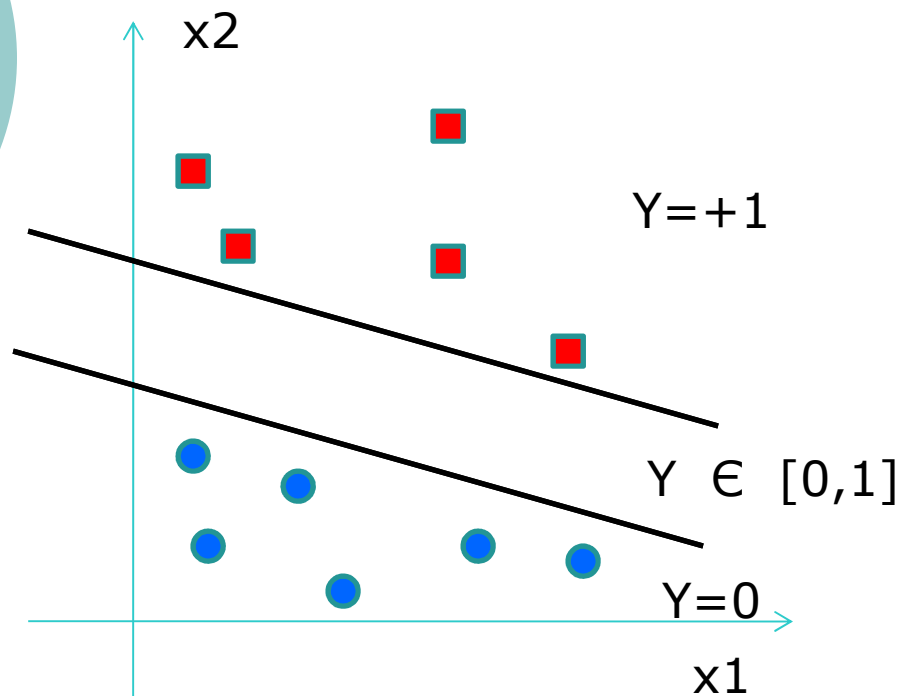
$$y = e^{\left( -\frac{(x_1 - c_1)^2}{\sigma_1} - \frac{(x_2 - c_2)^2}{\sigma_2} - \dots - \frac{(x_n - c_n)^2}{\sigma_n} \right)}$$

# RNAs: Neurônio com ativação degrau

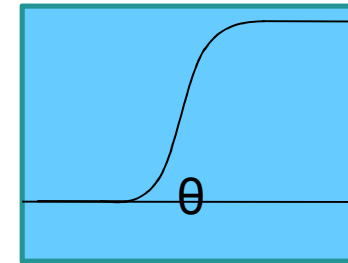


O ajuste dos parâmetros  $\mathbf{w}$  e  $\Theta$  (treinamento) altera a posição da reta (**hiperplano** para mais de duas entradas) e portanto da partição no espaço de entrada (**N-dimensional**)

# RNAs: Neurônio com ativação sigmoideal



Função de ativação

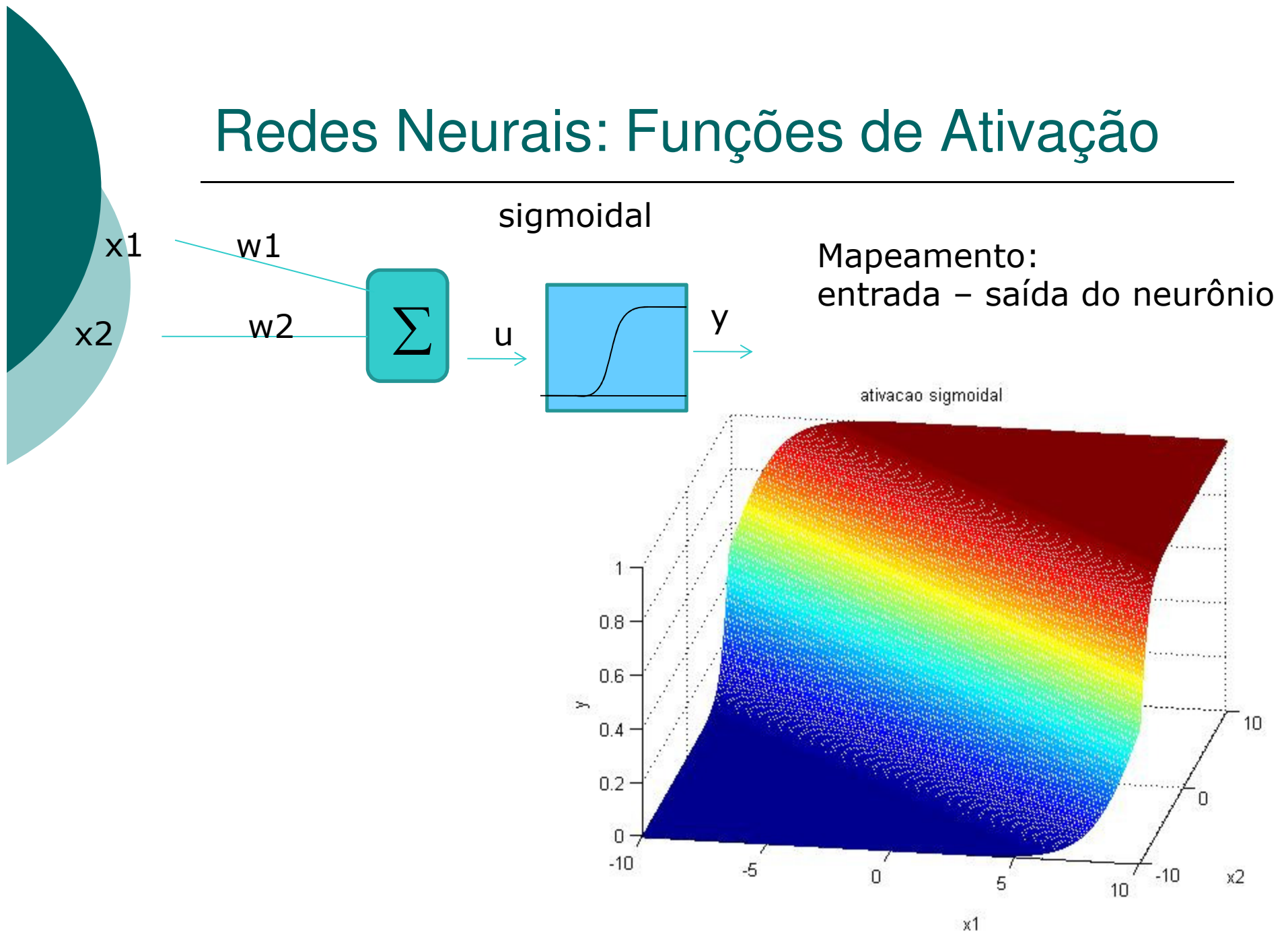


$$y = \frac{1}{1 + e^{(-\gamma \sum_{i=0}^n w_i x_i)}}$$

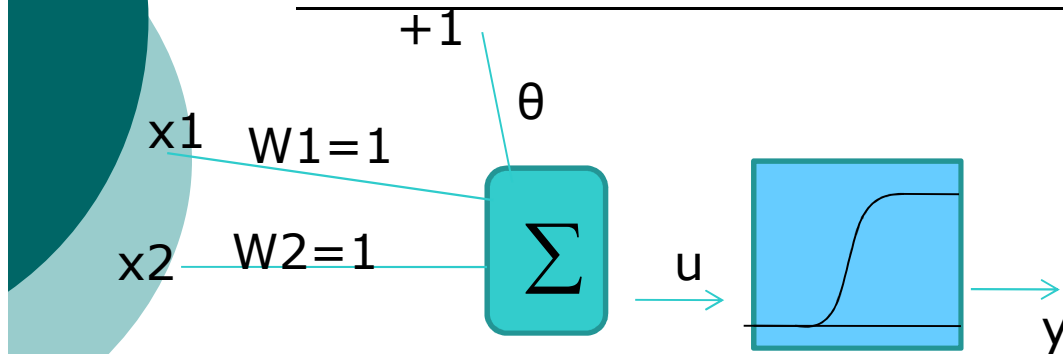
O ajuste dos parâmetros  $\mathbf{w}$  e  $\theta$  (treinamento) altera o mapeamento entrada saída



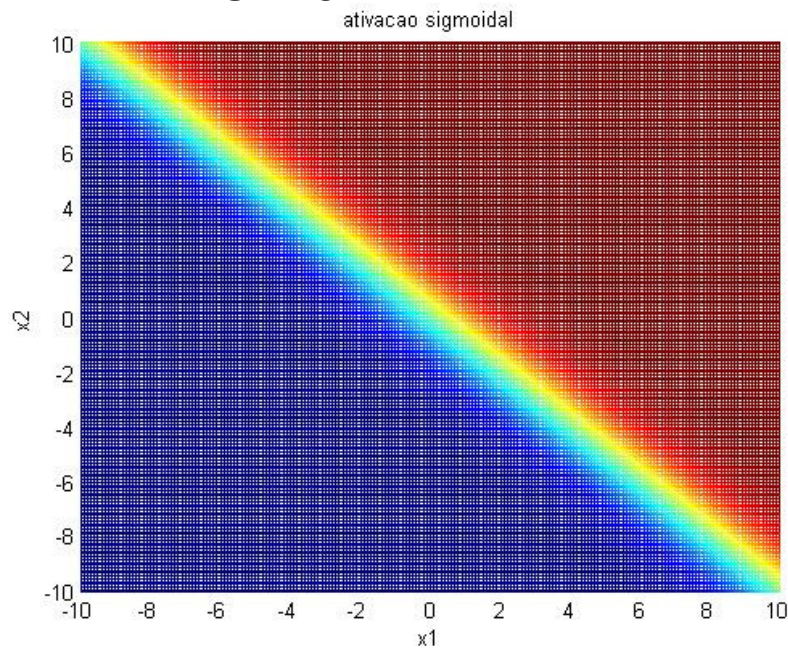
# Redes Neurais: Funções de Ativação



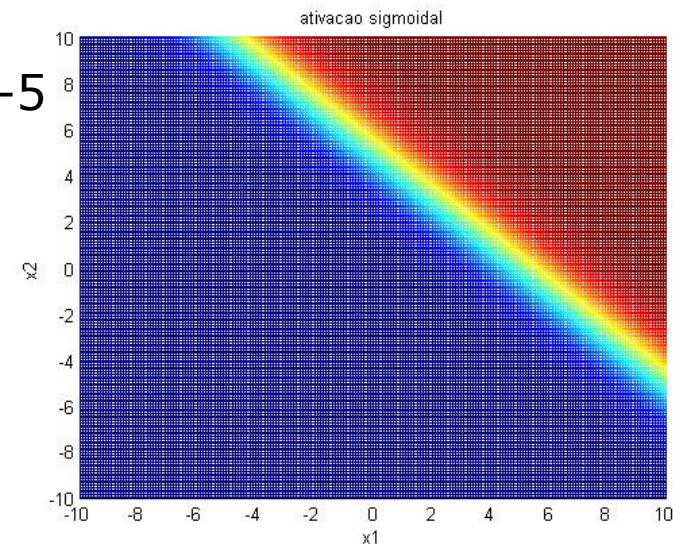
# Redes Neurais: Funções de Ativação



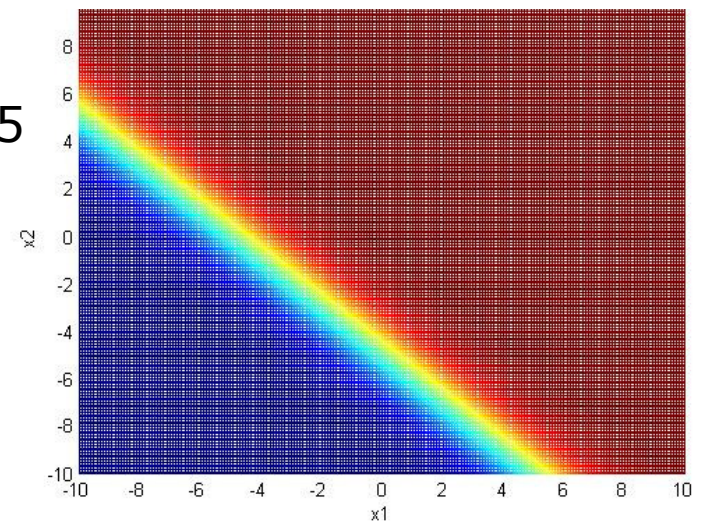
$\Theta = 0$



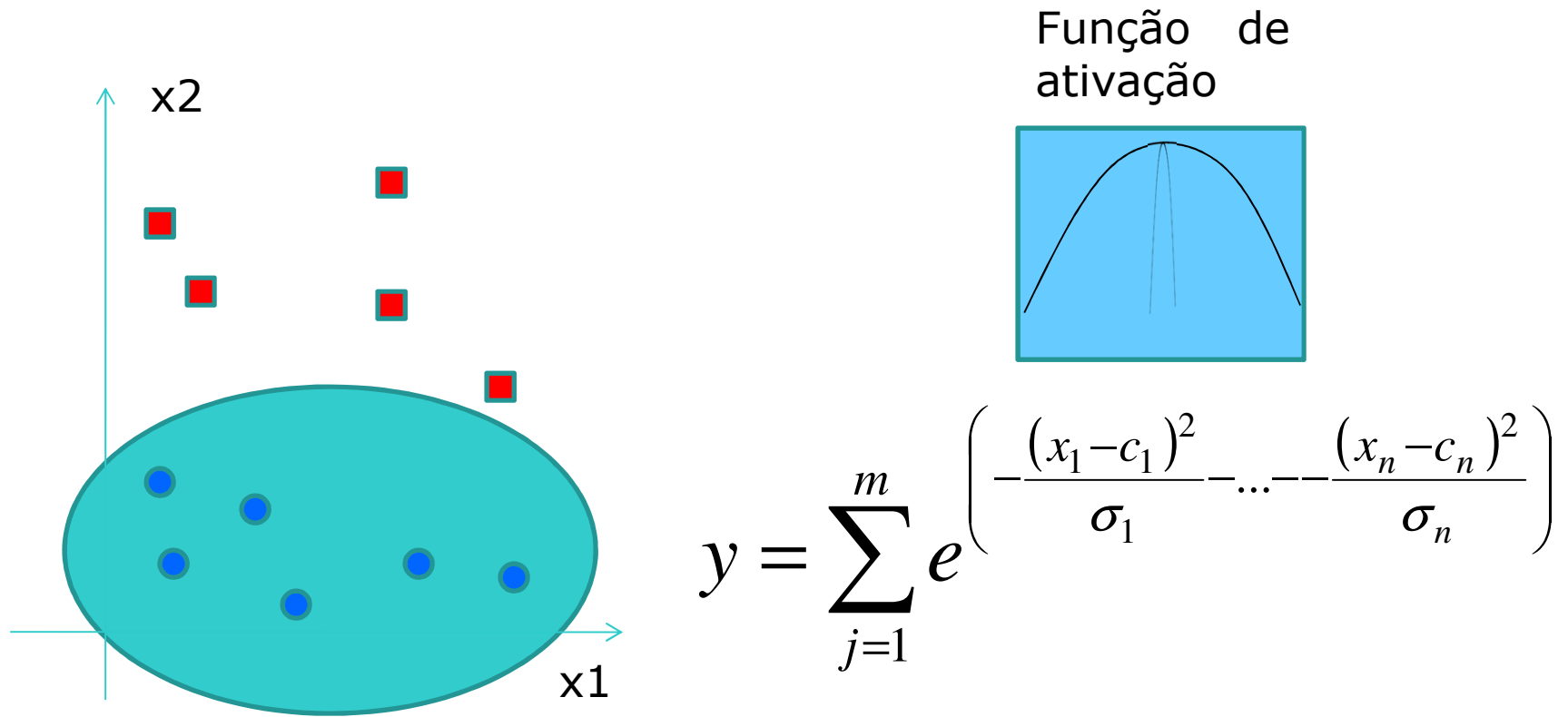
$\Theta = -5$



$\Theta = +5$

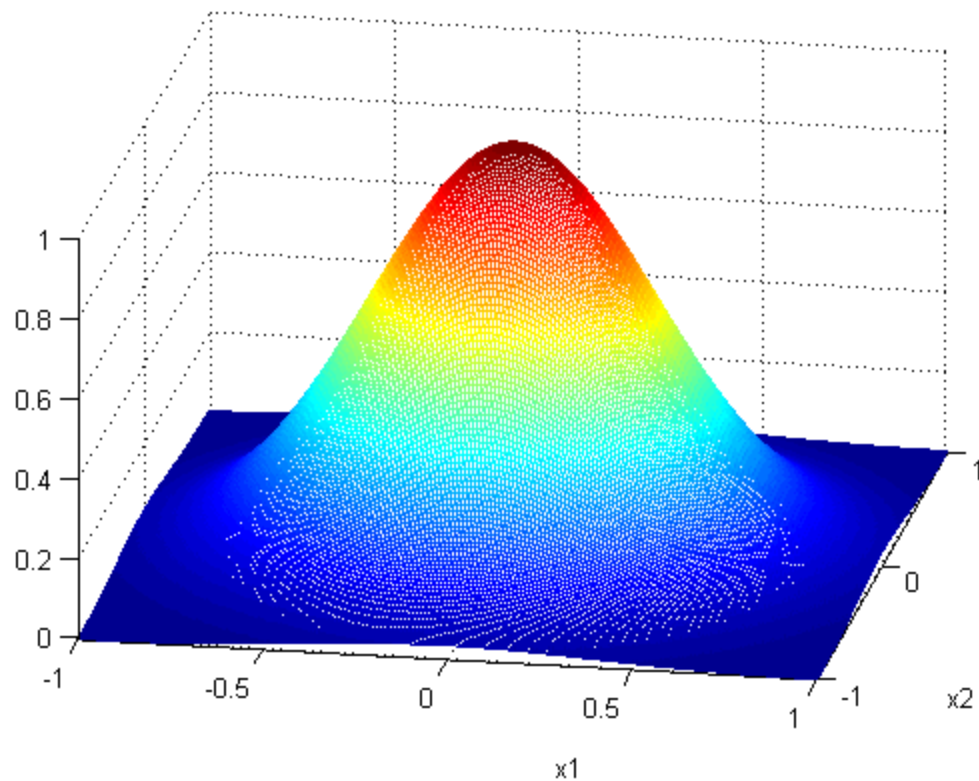
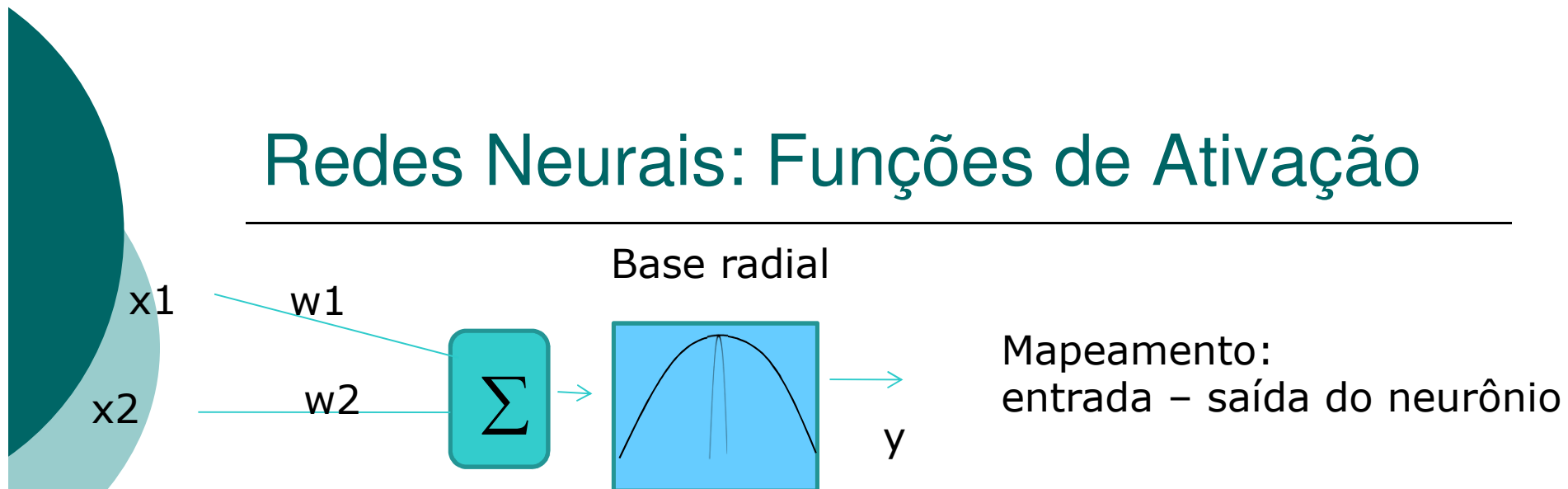


# RNAs: Neurônio com ativação base radial



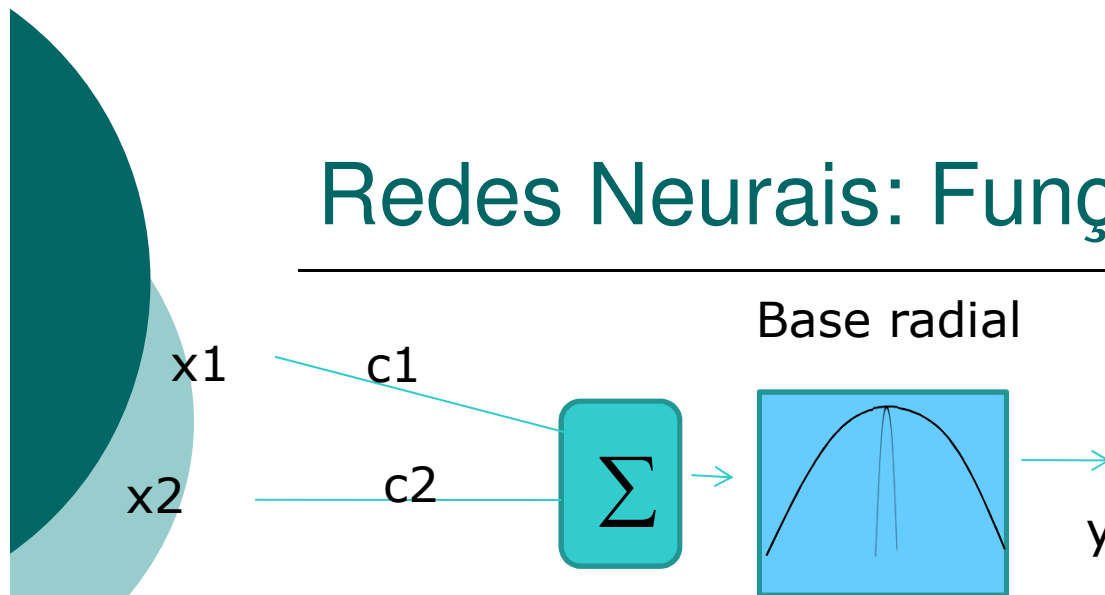
O ajuste dos parâmetros  $\mathbf{c}$  (treinamento) altera a posição da elipse (**hiperelipsóide**) e portanto o agrupamento no espaço de entrada (**N-dimensional**)

# Redes Neurais: Funções de Ativação



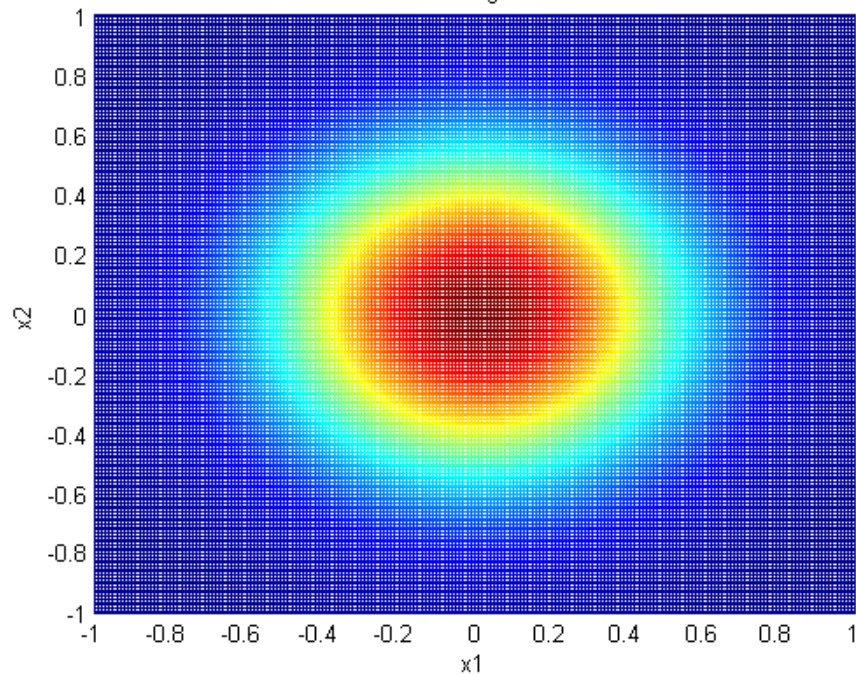


# Redes Neurais: Funções de Ativação

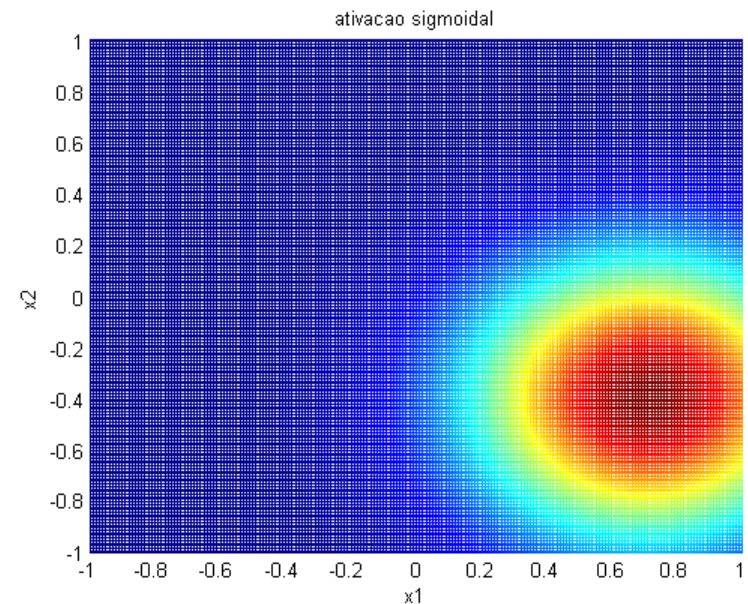


$$c_1 = c_2 = 0$$

ativacao sigmoidal



$$c_1 = 0,7$$
$$c_2 = -0,4$$





# Redes Neurais Artificiais

---

- Modelo do neurônio: função de ativação
- **Topologia da rede**
- Treinamento



# Redes Neurais Artificiais

---

- Modelo do neurônio
- **Topologia da rede**
- **Diferentes modelos:**
  - função de ativação
  - estrutura
- Treinamento



# Redes Neurais Artificiais: Arquitetura

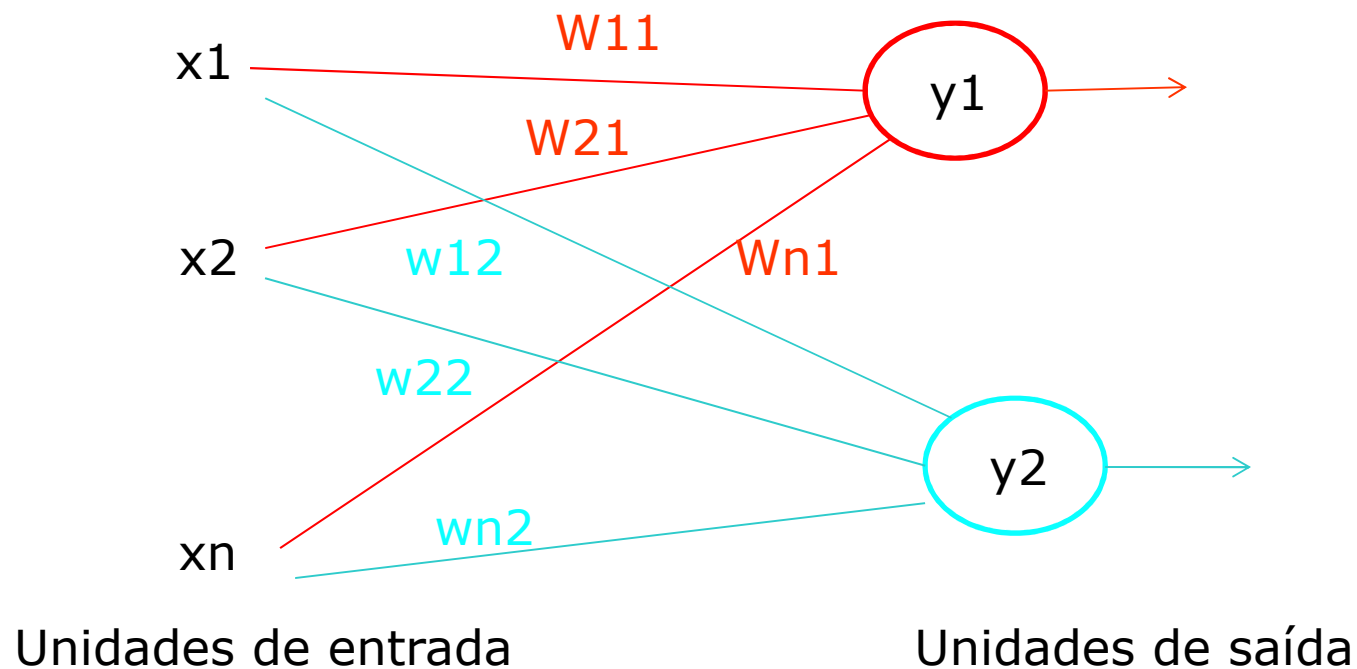
---

- Parâmetros que definem a arquitetura:
  - Número de camadas
    - Número de neurônios em cada camada
  - Tipo de conexão entre os neurônios
  - Conectividade da rede



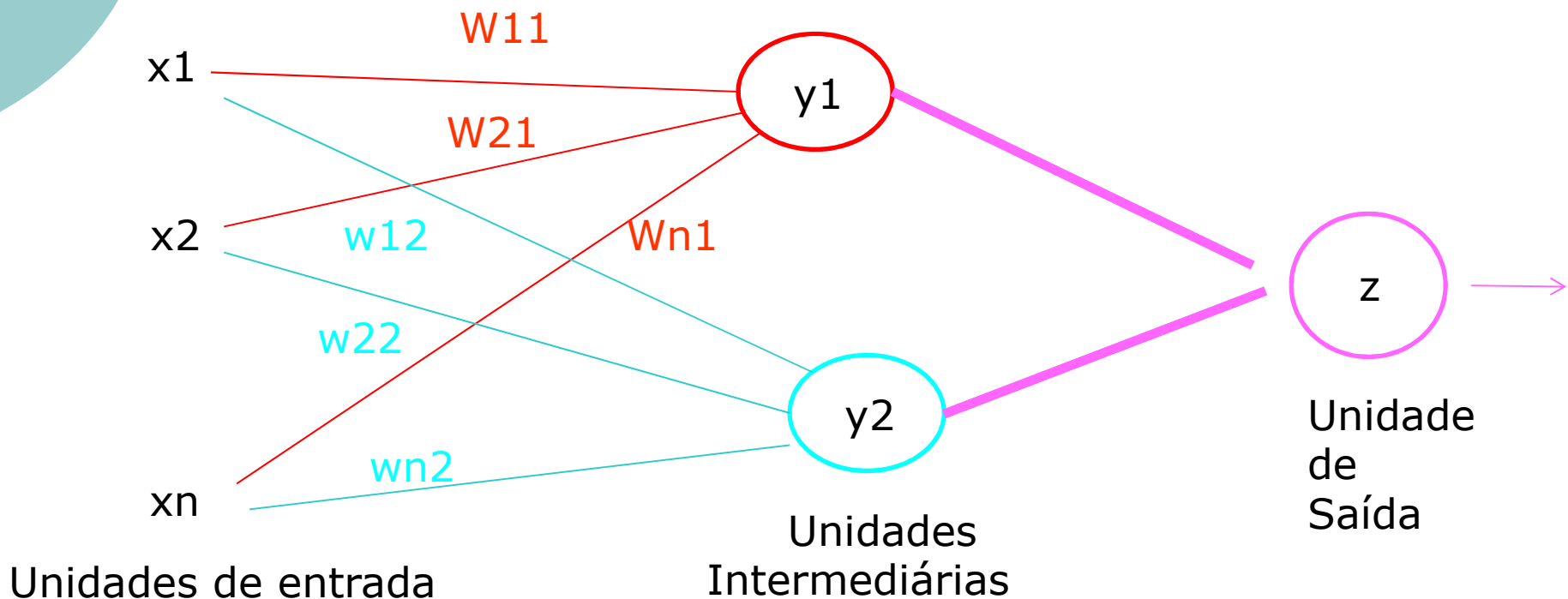
# Redes Neurais Artificiais: Arquitetura

Número de camadas: **Rede de Camada Única (Perceptron)**



# Redes Neurais Artificiais: Arquitetura

- Número de camadas: **Rede de Múltiplas Camadas (Multilayer Perceptron – MLP)**





# Redes Neurais Artificiais: Arquitetura

---

## Tipos de Conexões:

Redes do tipo **feedforward** (acíclicas): A saída de um neurônio na  $k$ -ésima camada **não** pode ser usada como entrada de neurônios em camadas de índices menores ou iguais a  $k$  (não há recorrência).

- Redes do tipo feedback (cíclicas ou **recorrentes**): A saída de algum neurônio na  $k$ -ésima camada **é** usada como entrada de neurônios em camadas de índices menores ou iguais a  $k$ 
  - autômatos: saída final única com recorrência
  - auto-associativas: todas as ligações são cíclicas



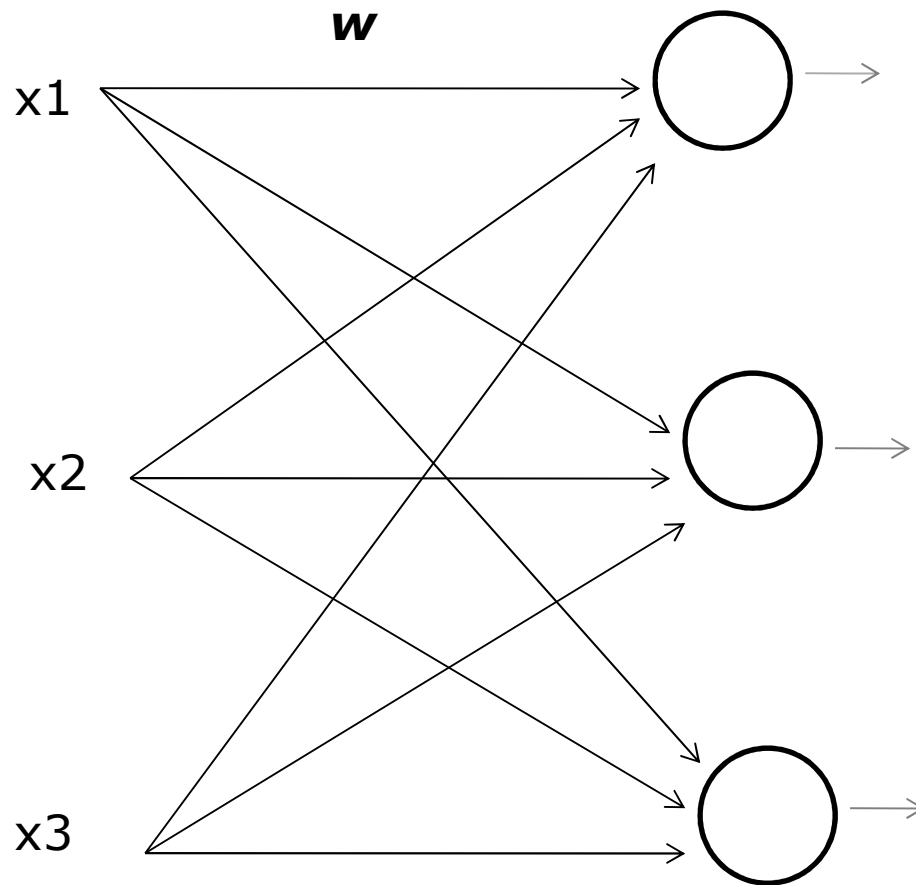
# Redes Neurais Artificiais: Arquitetura

---

- **Conectividade:**
  - Parcialmente conectada
  - Completamente conectada

# Redes Neurais Artificiais: Arquitetura

---



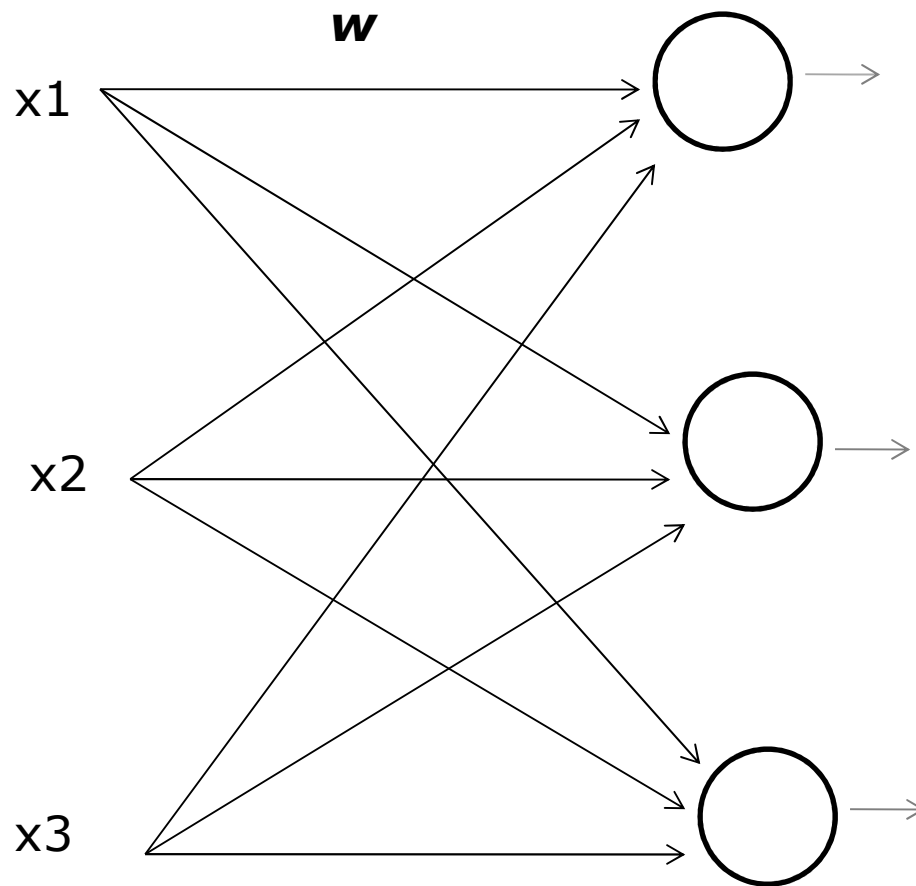
Quantas camadas?

Feedforward ou  
recorrente?

Parcial ou totalmente  
conectada?

# Redes Neurais Artificiais: Arquitetura

---



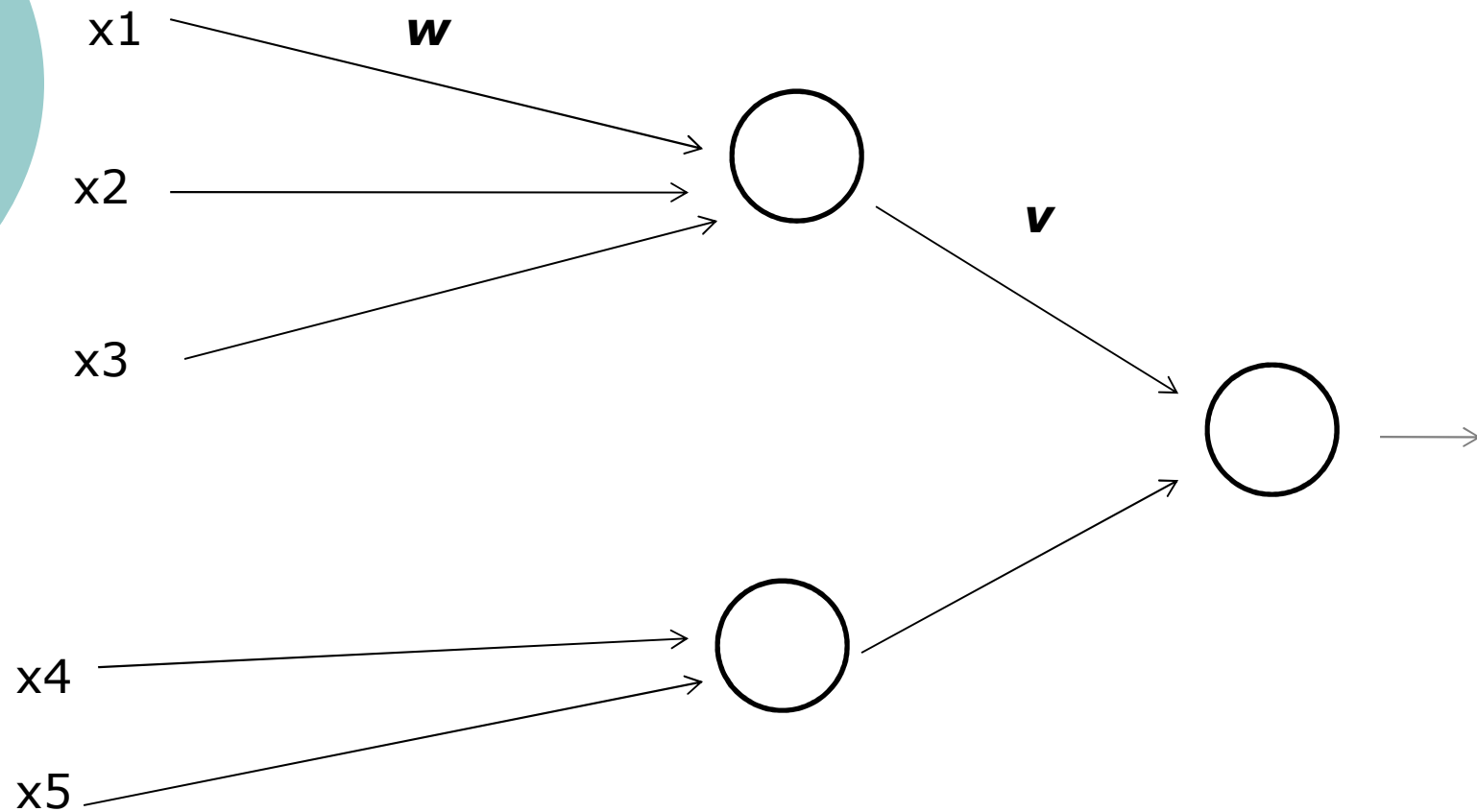
Rede de  
Camada Única

Feedforward

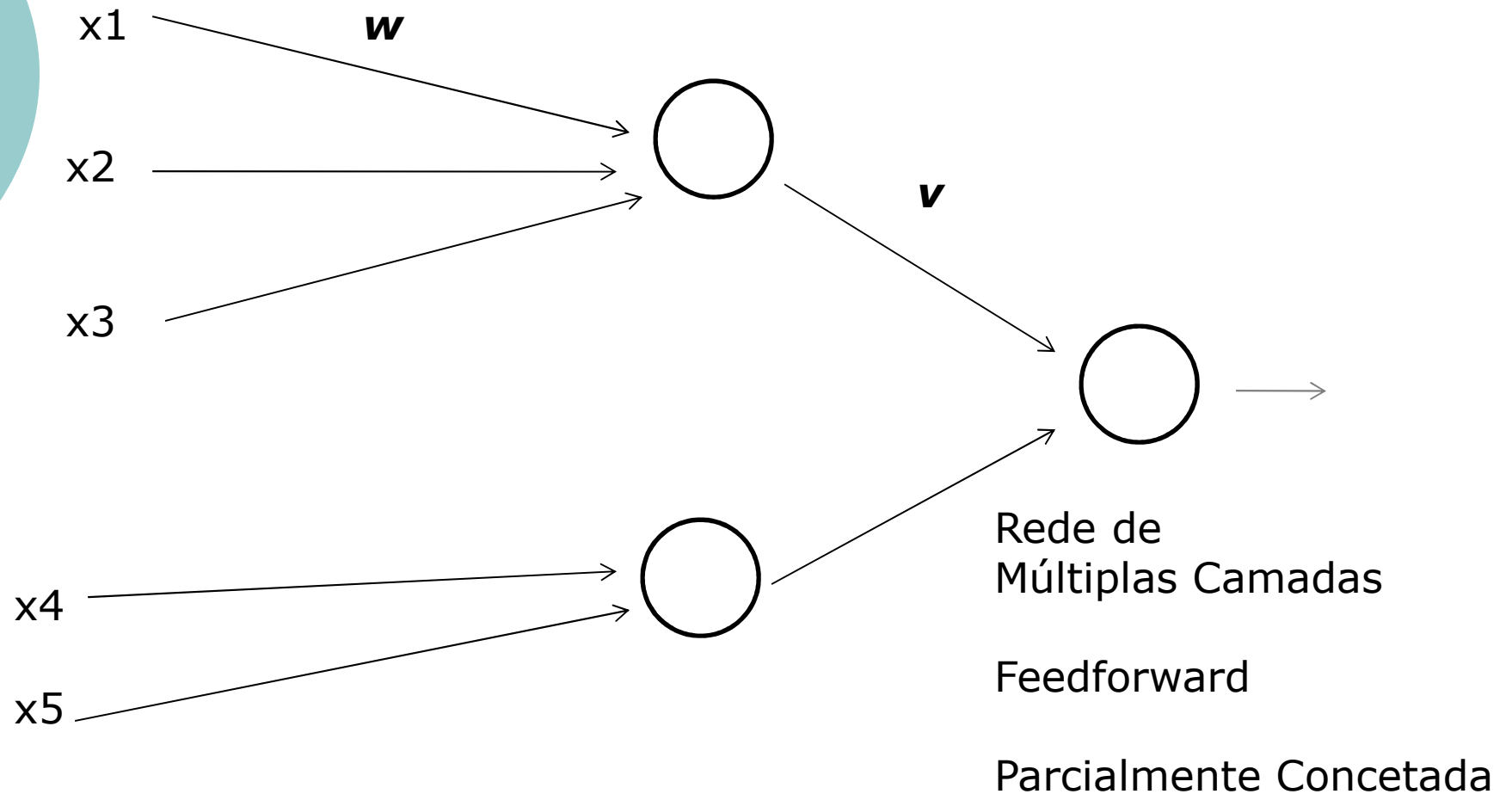
Complemente  
Conectada

Exemplo:  
**Perceptron**

# Redes Neurais Artificiais: Arquitetura



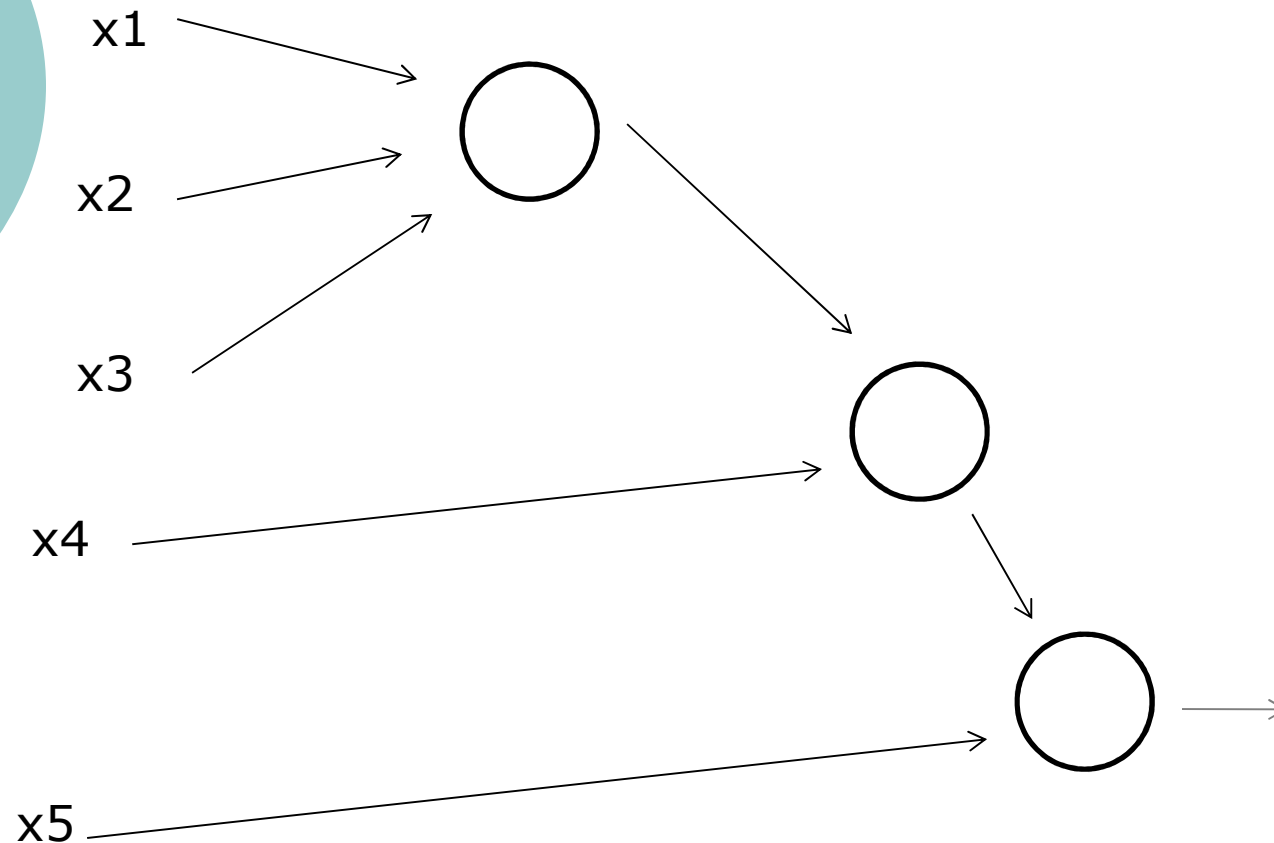
# Redes Neurais Artificiais: Arquitetura



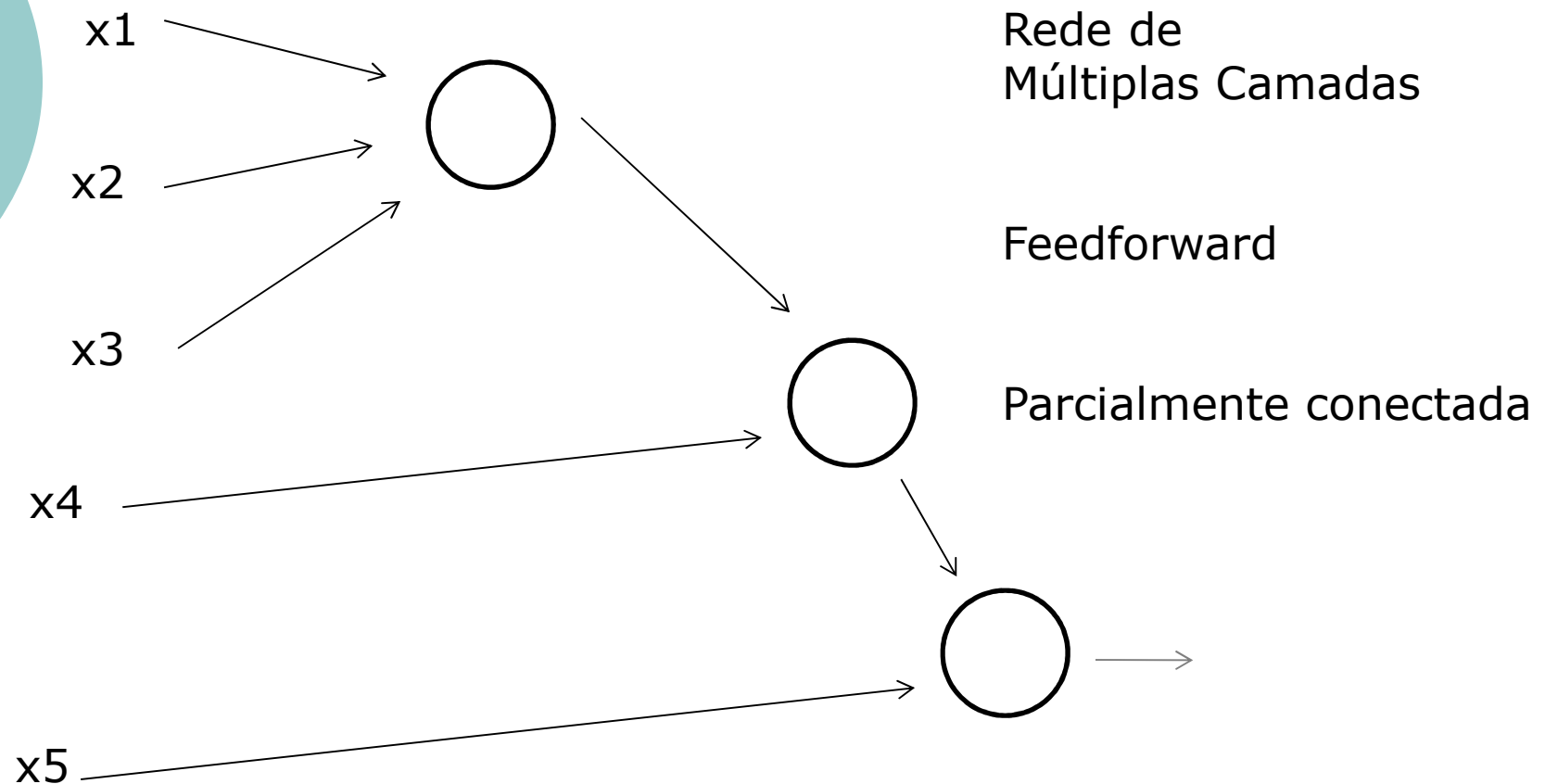


# Redes Neurais Artificiais: Arquitetura

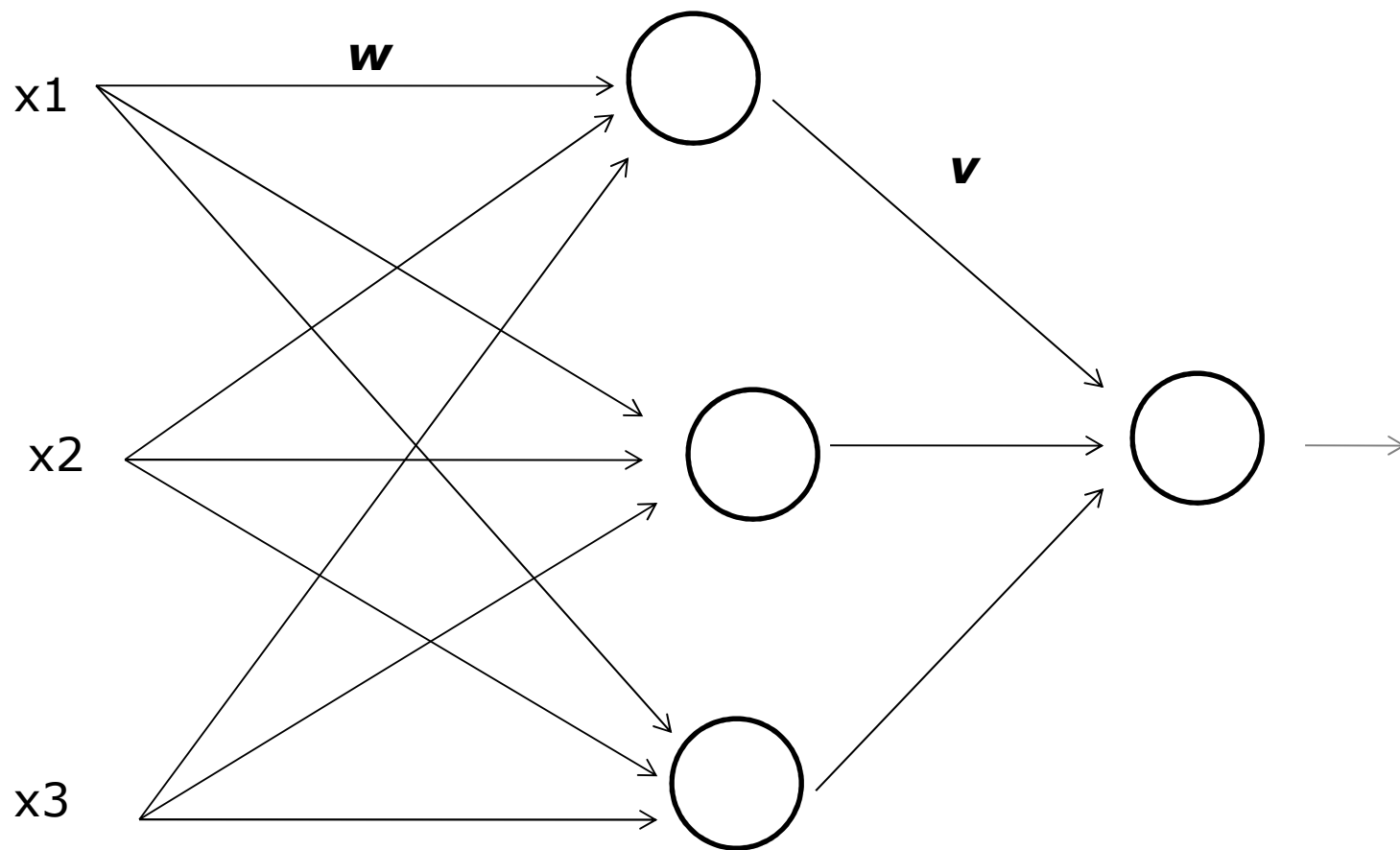
---



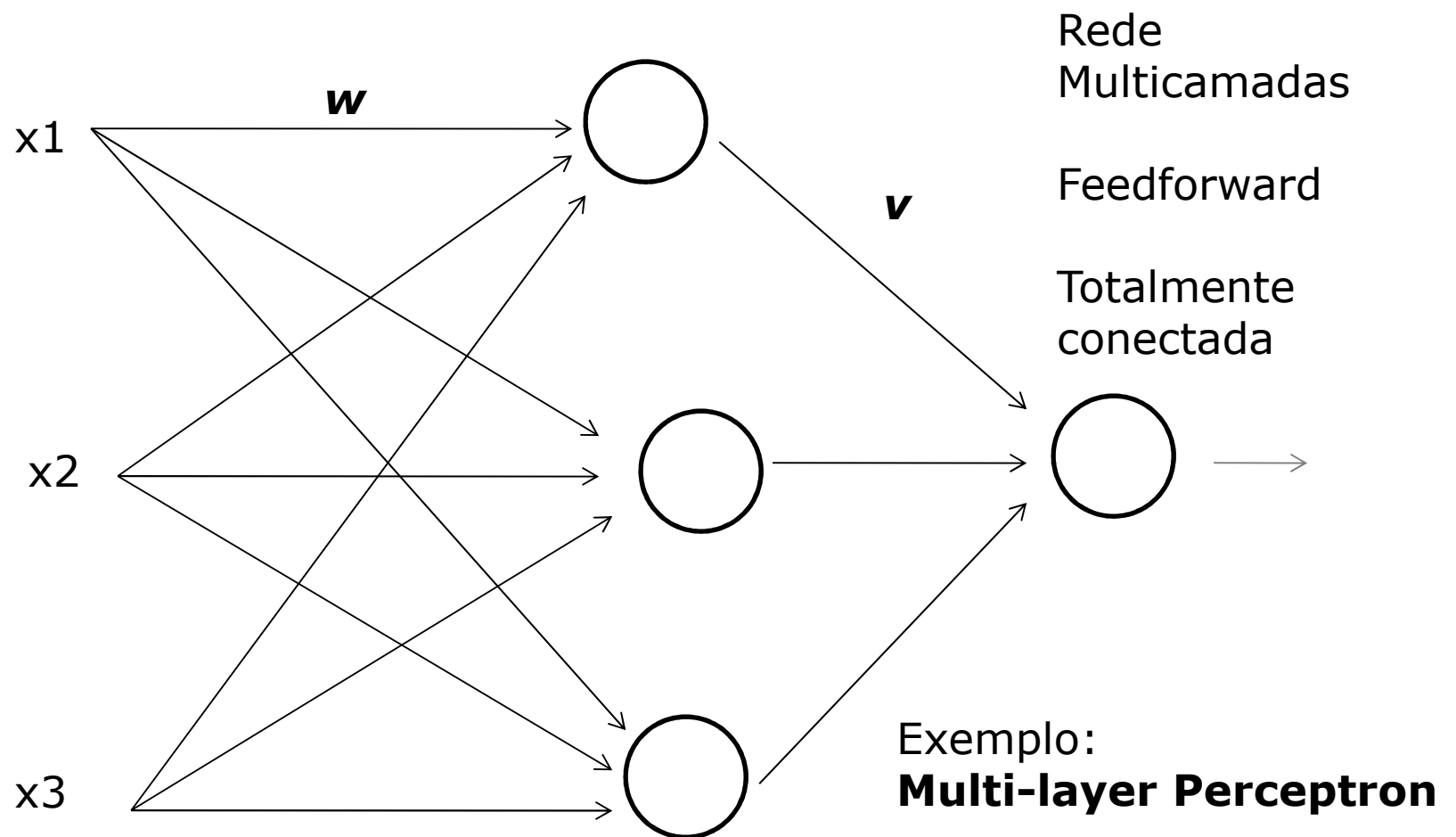
# Redes Neurais Artificiais: Arquitetura



# Redes Neurais Artificiais: Arquitetura

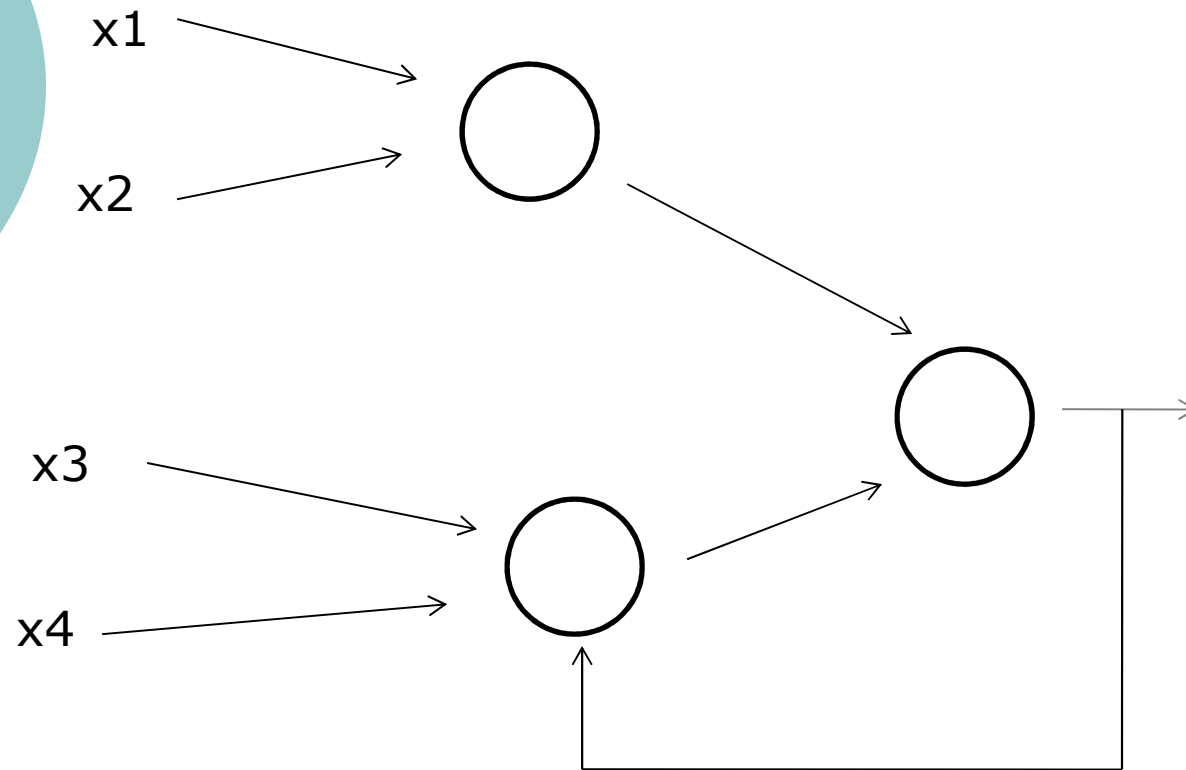


# Redes Neurais Artificiais: Arquitetura



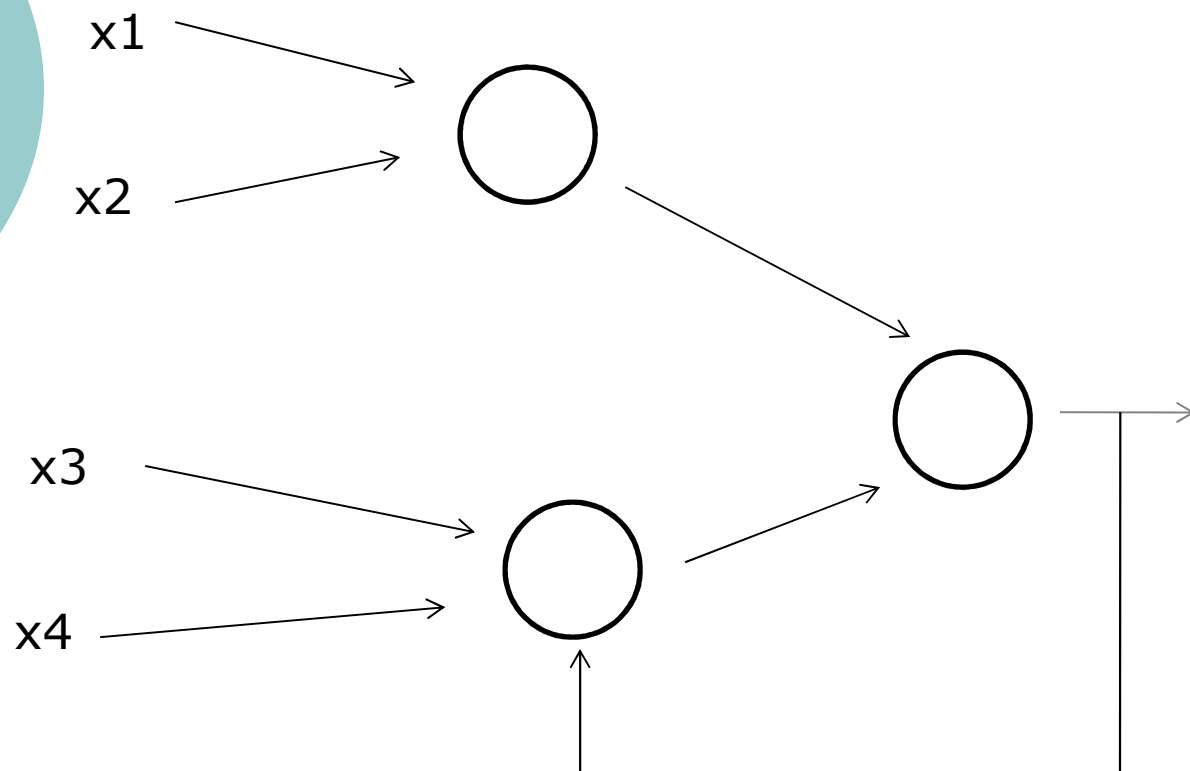
# Redes Neurais Artificiais: Arquitetura

---



# Redes Neurais Artificiais: Arquitetura

---



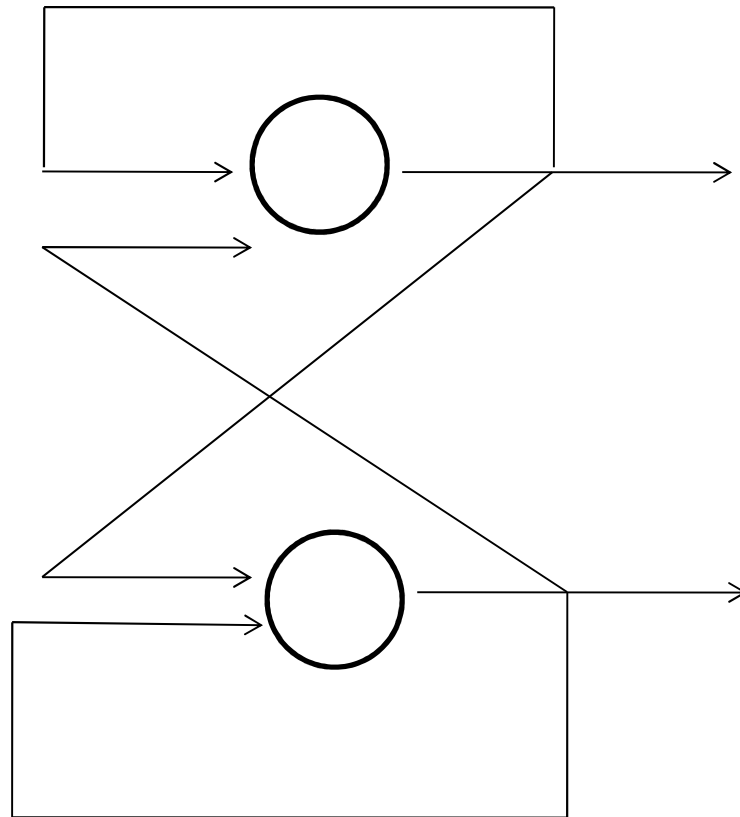
Rede de  
Múltiplas Camadas

Recorrente

Parcialmente  
Conectada

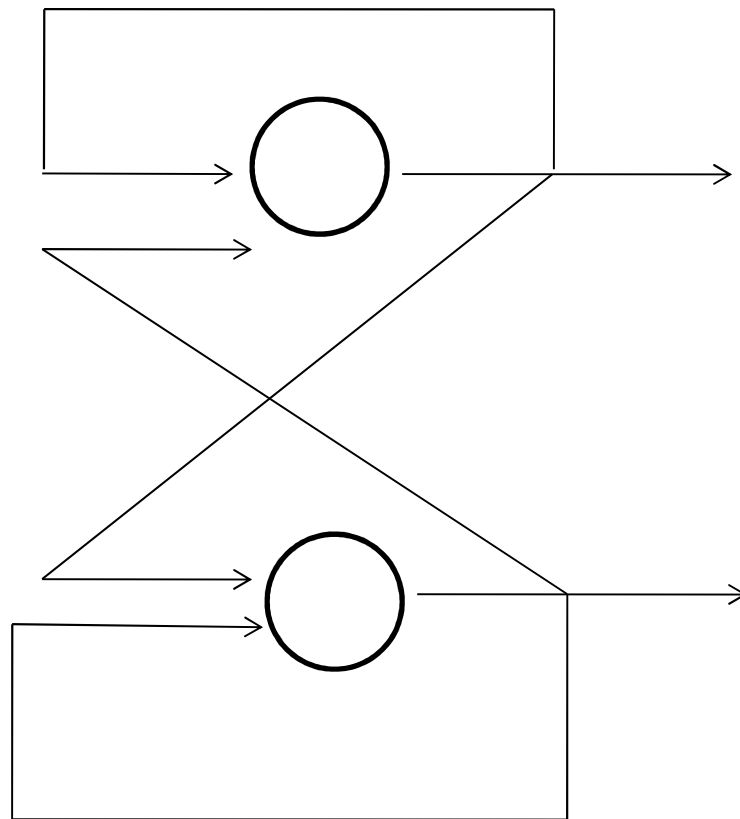
# Redes Neurais Artificiais: Arquitetura

---



# Redes Neurais Artificiais: Arquitetura

---



Rede de  
Camada Única

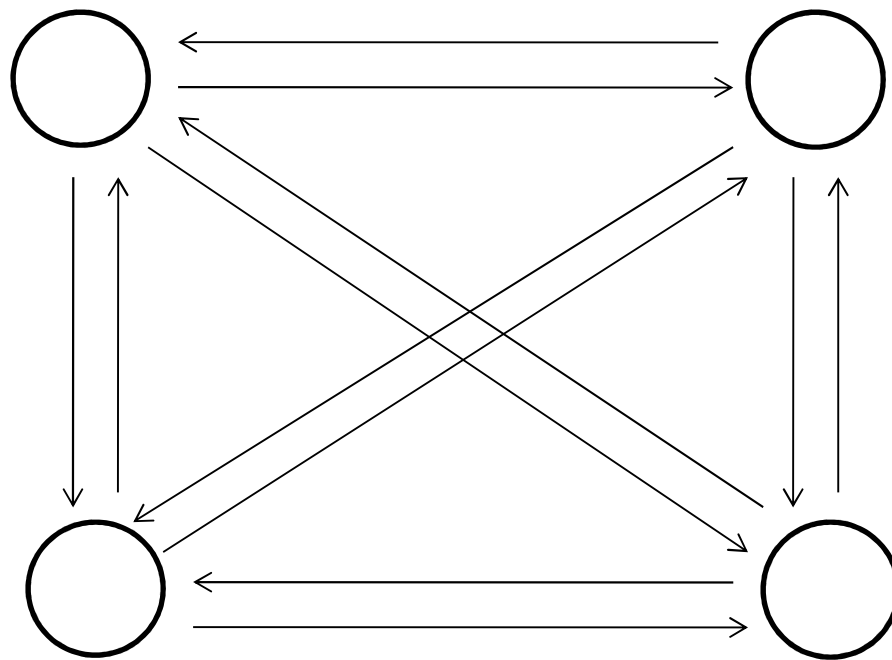
Recorrente

Completamente  
Conectada



# Redes Neurais Artificiais: Arquitetura

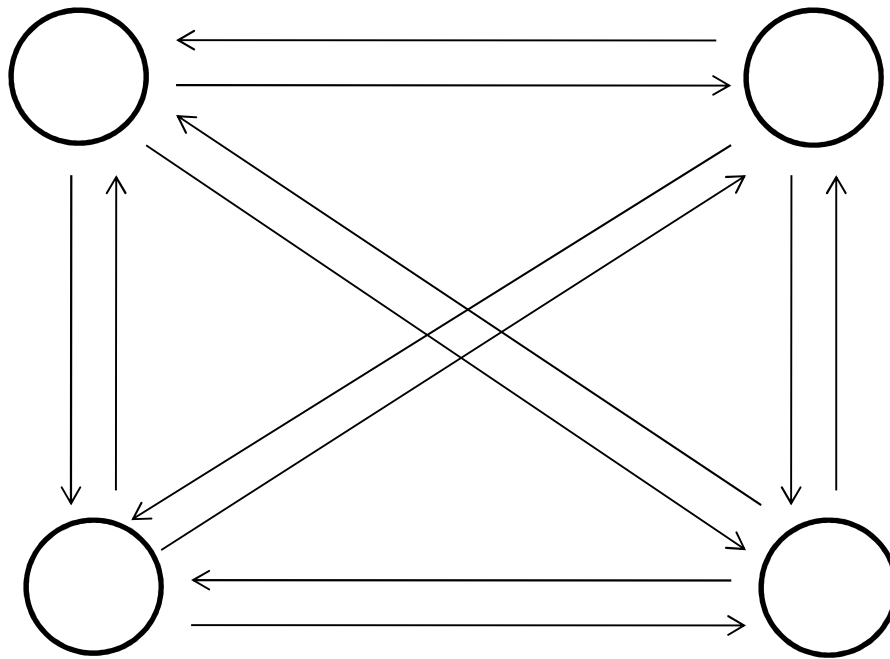
---



?????

# Redes Neurais Artificiais: Arquitetura

---



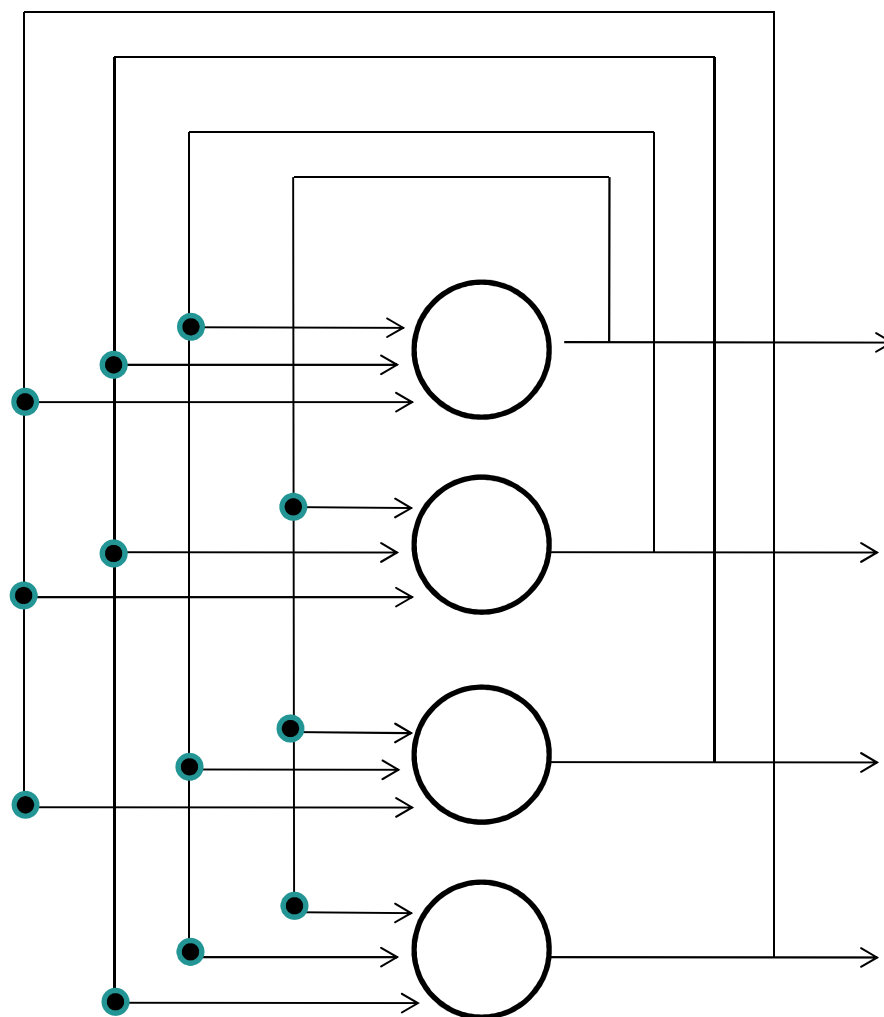
Rede de Camada Única

Recorrente

Parcialmente Conectada

**Rede de Hopfield**  
(com 4 neurônios)

# Redes Neurais Artificiais: Arquitetura



Outra forma de  
visualizar a

Rede de Hopfield

Mostrada  
anteriormente



# Redes Neurais Artificiais (Modelos)

---

## **Redes Feedforward:**

Perceptron

**Multilayer Perceptron (MLP)**

Redes de Funções de Base Radial (RBF)

## **Redes Associativas e Recorrentes**

Hopfield

## **Redes Auto-organizáveis**

Mapas de Kohonen



# Redes Neurais Artificiais

---

- Modelo do neurônio: função de ativação
- Arquitetura da rede
- **Treinamento**



# Redes Neurais Artificiais: Treinamento

---

A etapa de **treinamento** ou aprendizado consiste em um processo iterativo de **ajuste dos parâmetros** da rede.

Normalmente são **ajustados os pesos** das conexões os quais guardam (ao final do treinamento) **o conhecimento** que a rede adquiriu do ambiente no qual está operando.



# Aprendizado Conexionista

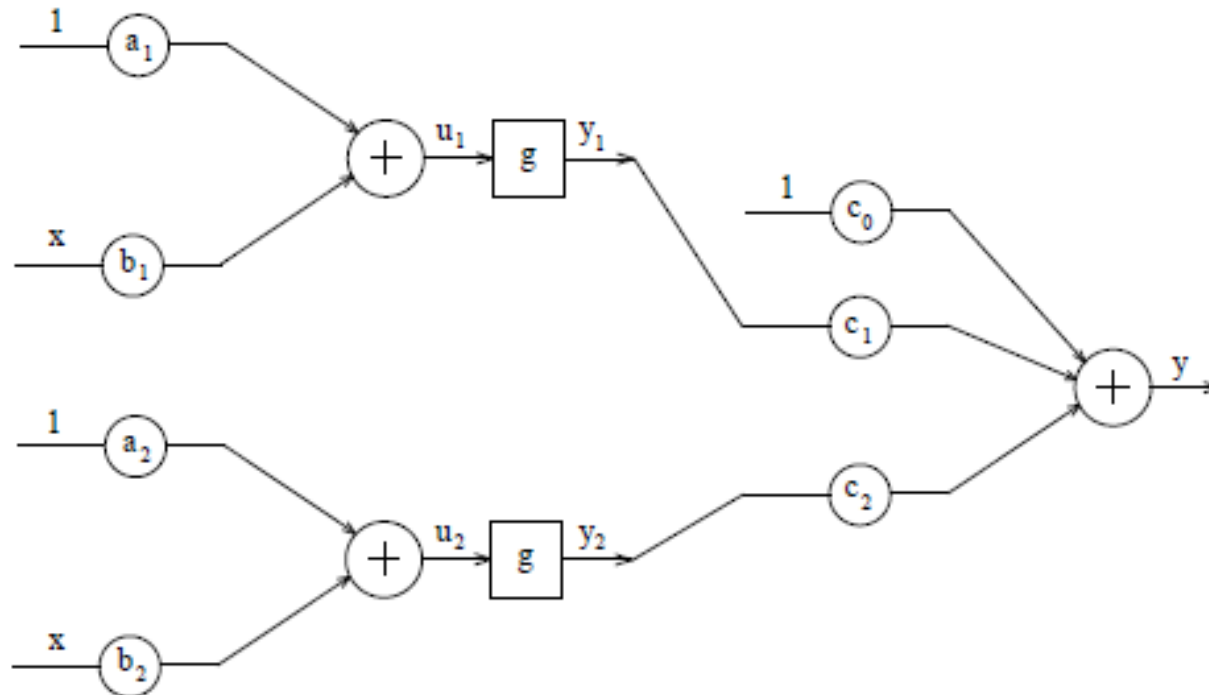
## O papel dos pesos

# Aprendizado Conexionista

## O papel dos pesos

$$y = c_0 + \sum_{n=1}^p c_n g(b_n x + a_n)$$

$g$  : função simóide



$$y = c_0 + c_1 g(b_1 x + a_1) + c_2 g(b_2 x + a_2) \Rightarrow \begin{cases} a : \text{deslocamento no eixo x} \\ b : \text{inclinação da sigmóide} \\ c : \text{amplitude da sigmóide} \end{cases}$$

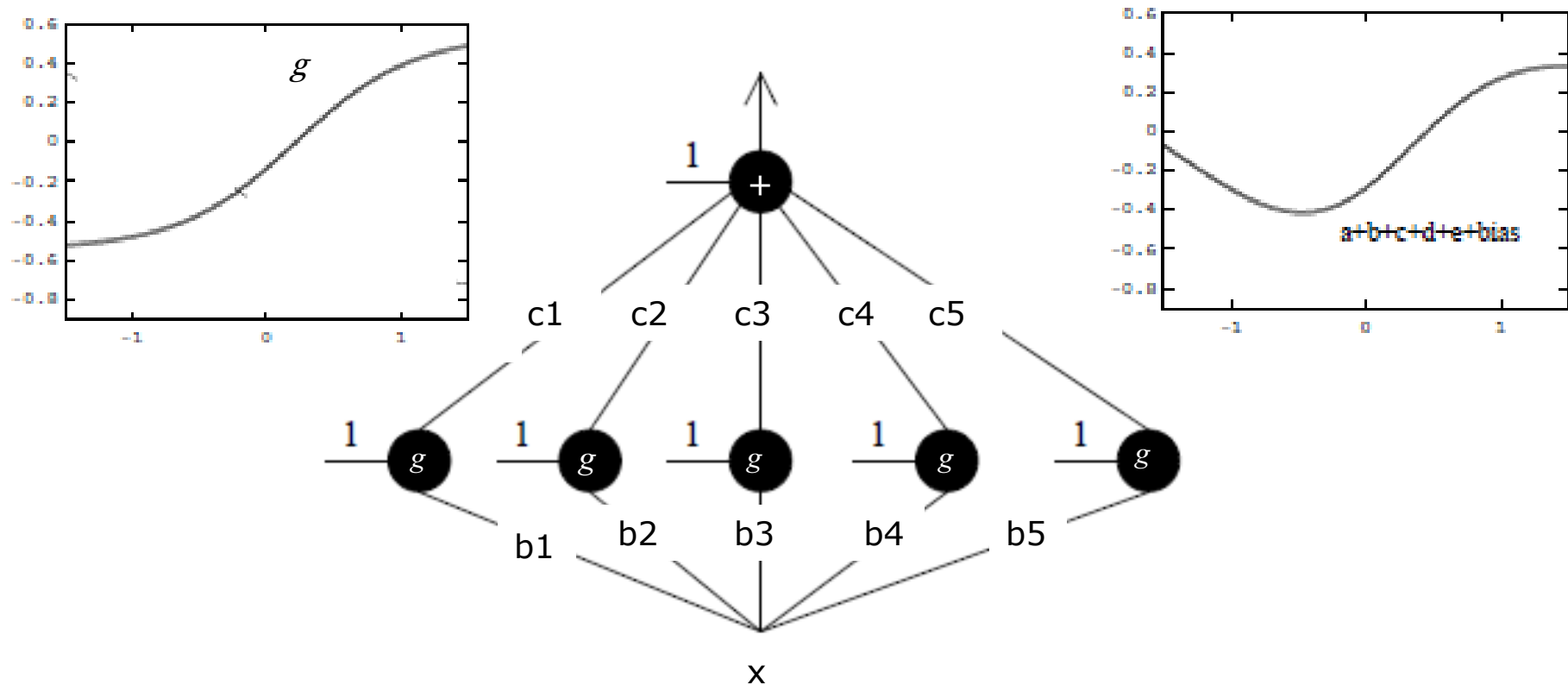


# Aprendizado Conexionista: pesos

Exemplo: Forma “construtiva” de aproximação de um mapeamento não-linear

# Aprendizado Conexionista: pesos

**Exemplo:** Forma “construtiva” de aproximação de um mapeamento não-linear



$$f(w) = \underbrace{c_1 g(b_1 x + a_1)}_a + \underbrace{c_2 g(b_2 x + a_2)}_b + \underbrace{c_3 g(b_3 x + a_3)}_c + \underbrace{c_4 g(b_4 x + a_4)}_d + \underbrace{c_5 g(b_5 x + a_5)}_e + \underbrace{c_0}_{\text{bias}}$$

# Aprendizado Conexionista: pesos

$$f(w) = \underbrace{c_1 g(b_1 x + a_1)}_a + \underbrace{c_2 g(b_2 x + a_2)}_b +$$

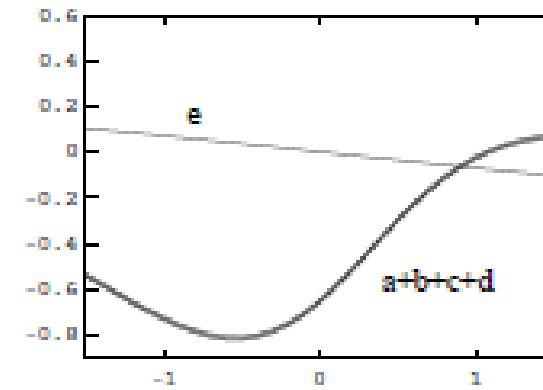
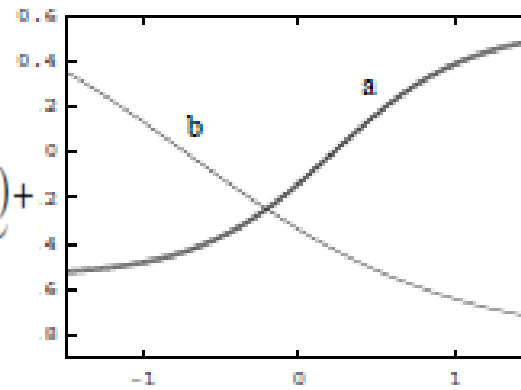
$$+ \underbrace{c_3 g(b_3 x + a_3)}_c + \underbrace{c_4 g(b_4 x + a_4)}_d +$$

$$+ \underbrace{c_5 g(b_5 x + a_5)}_e + \underbrace{c_0}_{bias}$$

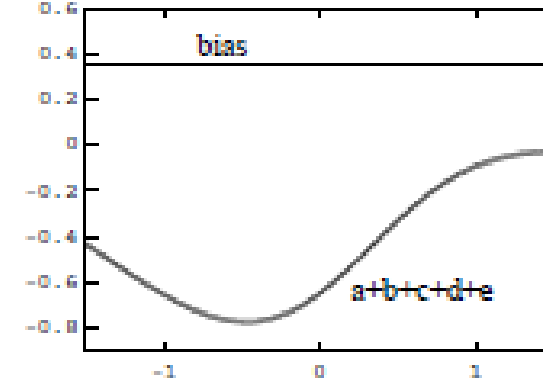
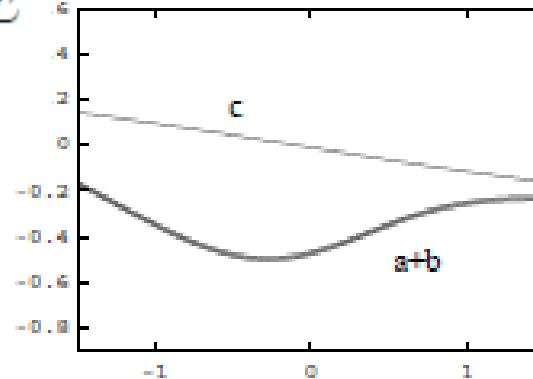
...

# Aprendizado Conexionista: pesos

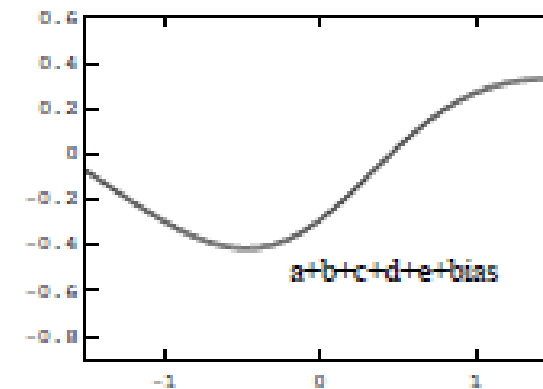
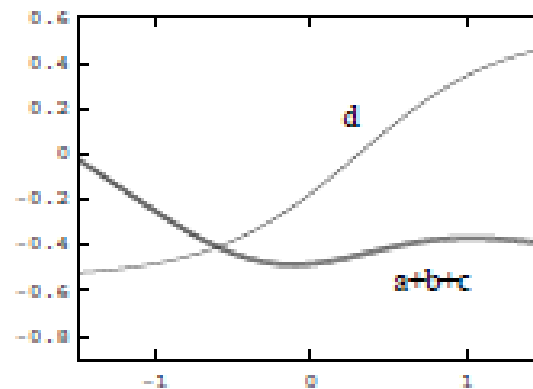
$$f(w) = \underbrace{c_1 g(b_1 x + a_1)}_a + \underbrace{c_2 g(b_2 x + a_2)}_b +$$



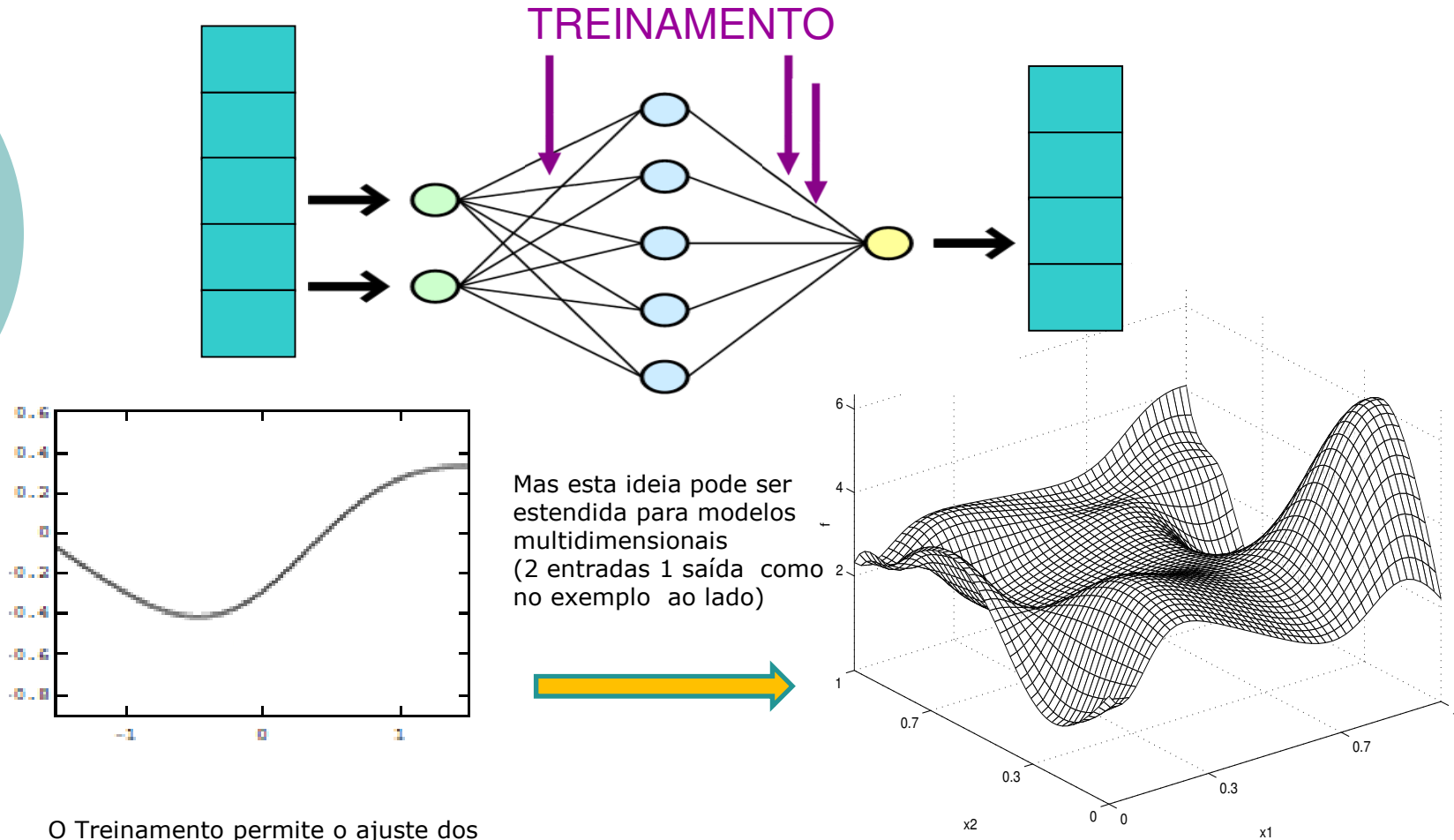
$$+ \underbrace{c_3 g(b_3 x + a_3)}_c + \underbrace{c_4 g(b_4 x + a_4)}_d +$$



$$+ \underbrace{c_5 g(b_5 x + a_5)}_e + \underbrace{c_0}_{bias}$$



# Solução por Redes Neurais (RN)

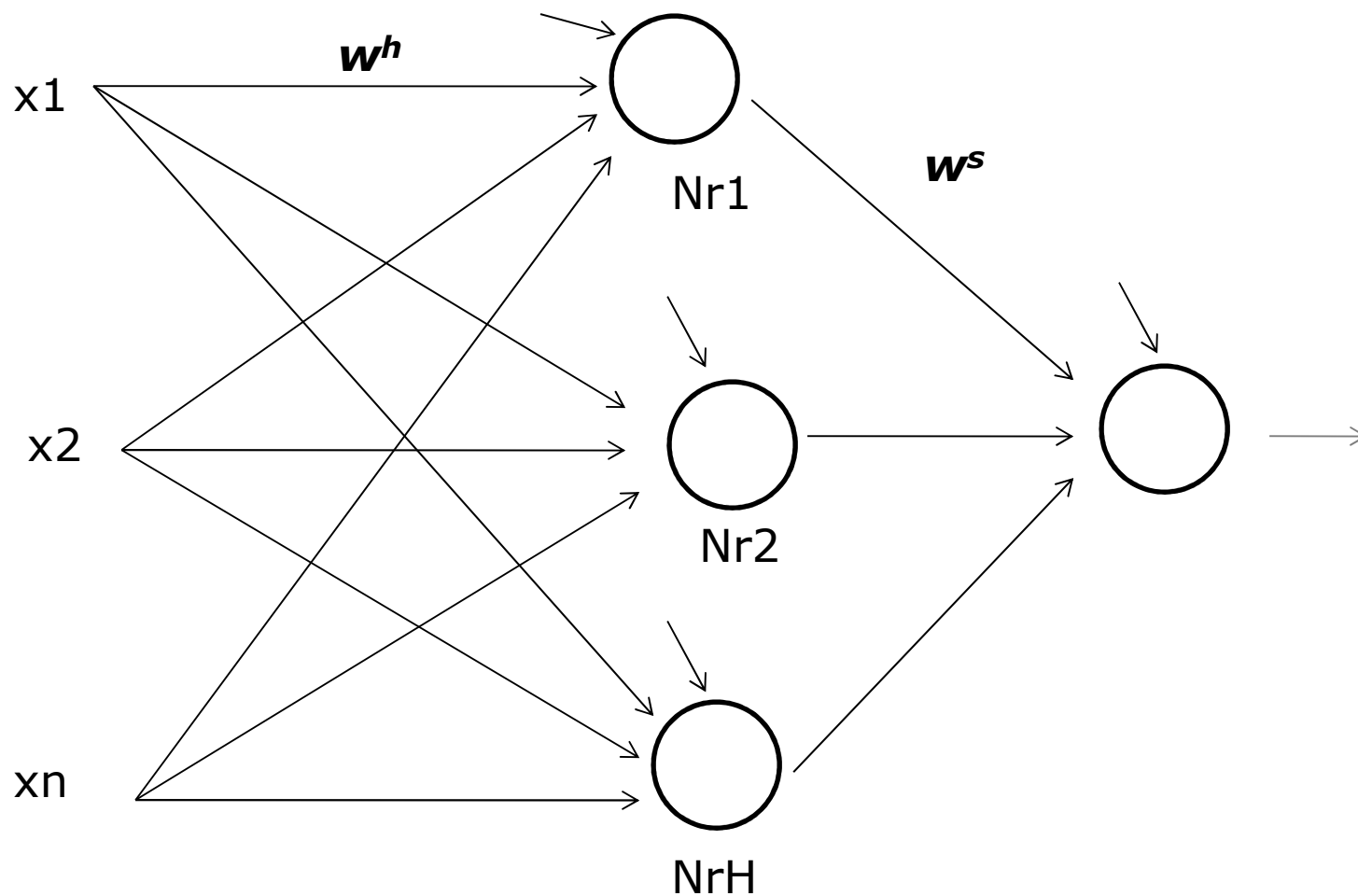


O Treinamento permite o ajuste dos parâmetros (pesos e bias) o qual modela a curva do mapeamento entrada – saída (1 entrada e 1 saída no exemplo acima)

Aproximação de Funções

## Exercício de Fixação

Considere a rede de duas camadas , completamente conectada e funcao  $g = \text{sigmoide}$





## Exercício de Fixação - Continuação

---

Calcular a saída da rede  $(N_{ent}, N_{term}, N_{saida}) = (2, 2, 1)$ ,  
 $g = \text{sigmóide}$  para os seguintes pares de entrada

x1	x2	Y
0.9	0.9	
0.1	0.9	
0.9	0.1	
0.1	0.1	

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

$$\text{onde } u = \left( \sum_{i=0}^n w_i x_i \right), \quad x_0 = 1$$

a) Considerando todos os pesos iguais a 1

b) Considerando todos os pesos como valores aleatórios no intervalo  $[-1, 1]$