Inteligência Artificial

Redes Neuronais Artificiais (Parte III)

Paulo Moura Oliveira

Departamento de Engenharias

Gabinete F2.15, ECT-1

UTAD

email: oliveira@utad.pt

RetroPropagação – BackPropagation

✓ A apresentação que se segue maioritariamente o texto do livro [2]*

✓ Até este ponto vimos que a:

- Regra do Perceptrão e a Regra Delta, são modelos com poder de adaptação.
- No entanto: com uma só unidade de aprendizagem **só é possível classificar funções linearmente separáveis**.

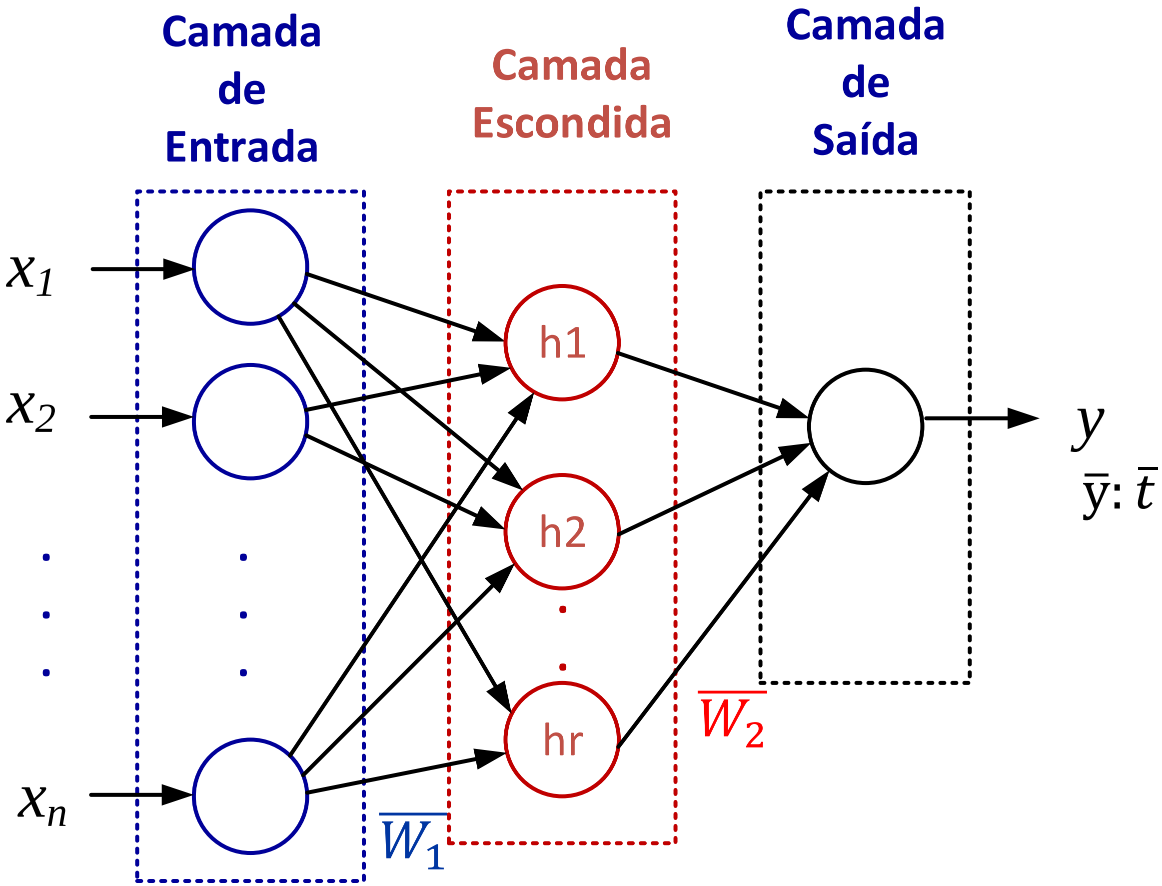
Precisamos de mais de que uma unidade integradas em Redes e com várias camadas: **Redes Multicamada.**

*[2] Lucci S. e Kopec D., (2016), Artificial Intelligence in the 21st Century, A Living Introduction, Mercury leArning And inforMAtion.

Rede Neuronal Multicamada-*Feedforward*

✓ Uma representação genérica:

n - r - m:



✓ Notação:

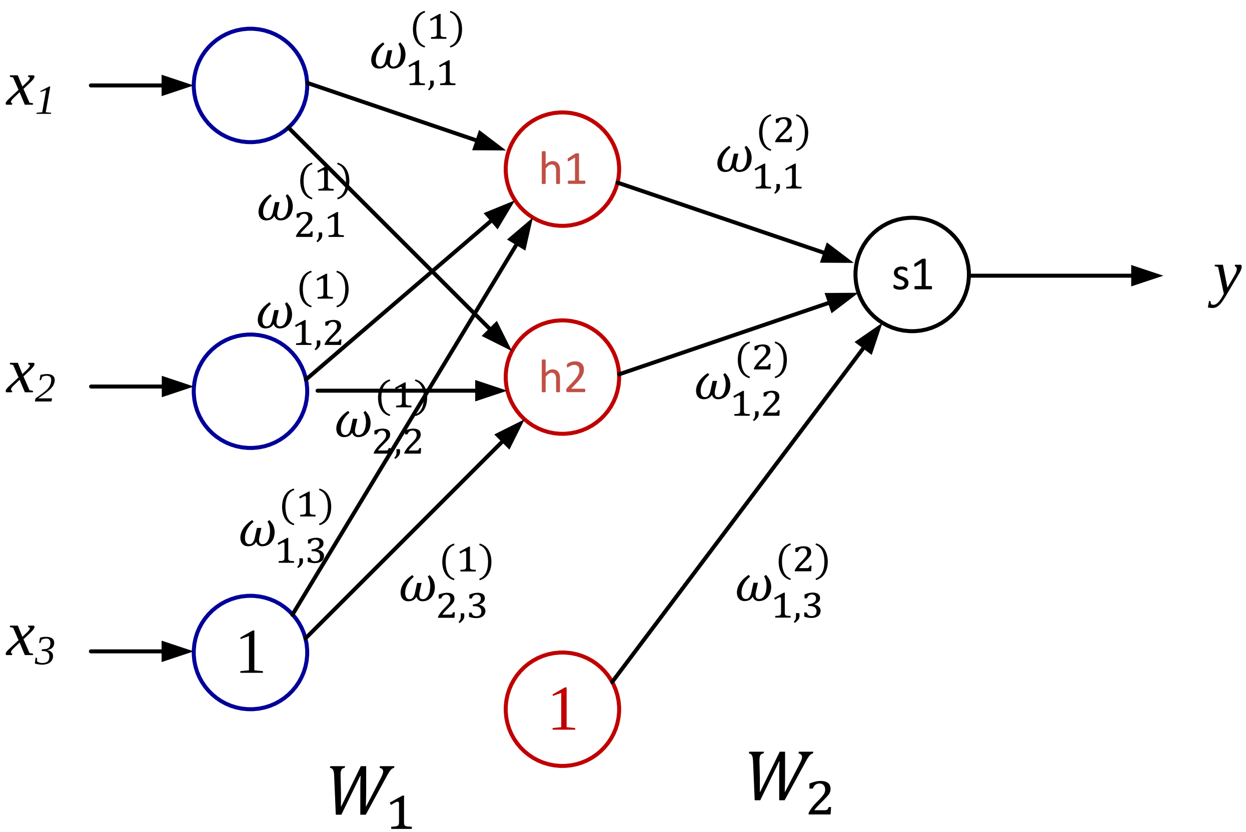
- ***n***- número de neurónios de entrada
- ***r***- número de neurónios escondidos
- ***m***- número de neurónios de saída (neste caso m=1)
- $\overline{W_1}$ **matriz** de pesos que ligam a camada de entrada à escondida.
- $\overline{W_2}$ **matriz** de pesos que ligam a camada escondida à de saída.
- $\bar{y}$ saída da rede (**vetor**)
- $\bar{t}$ saída desejada (**vetor** alvo ou *target*)

Rede Neuronal Multicamada-*Feedforward*

- ✓ Consideremos a seguinte rede 3-2-1:

$\overline{W_1}$ matriz: $r_h \; x \; n_i$ = 2 x 3

$\overline{W_2}$ matriz: $m_o \; x \; r_h$ = 1 x 3



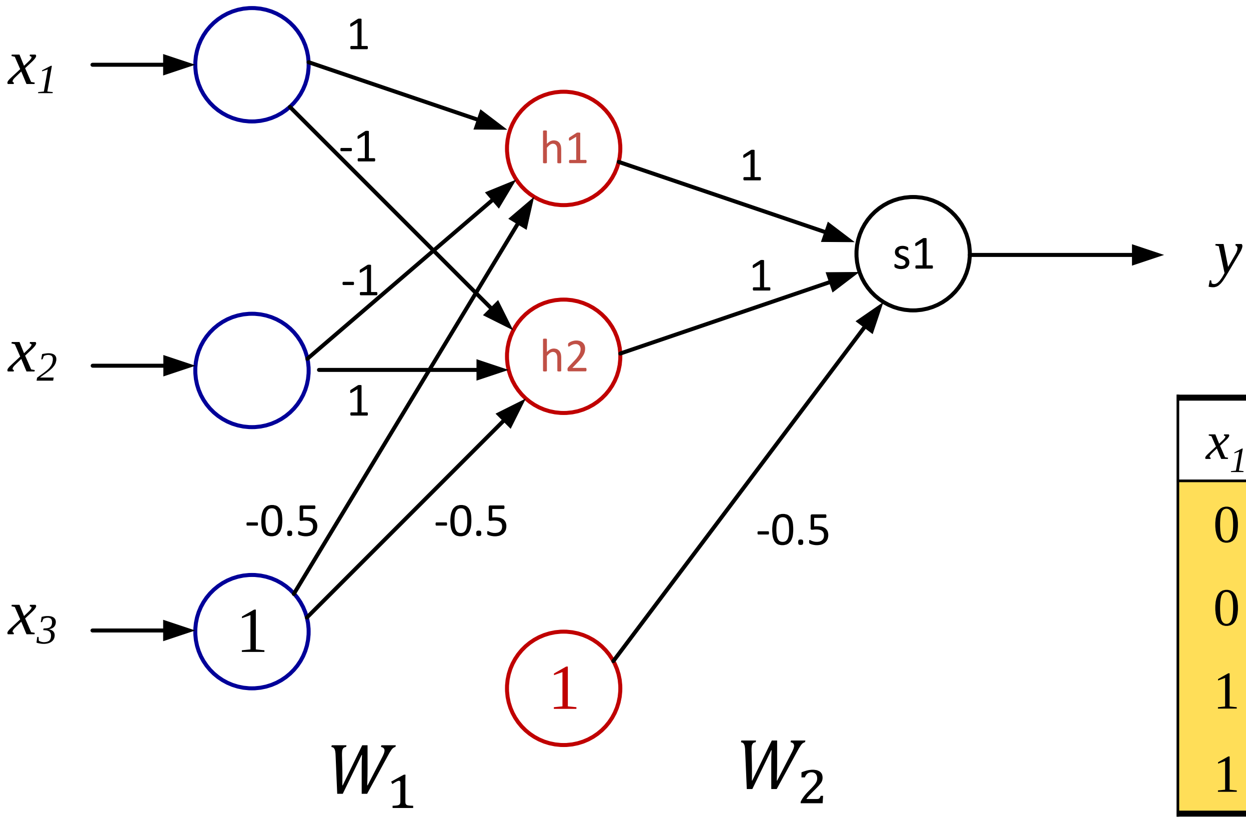
$$W_1 = \begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} & \omega_{13}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} & \omega_{23}^{(1)} \end{bmatrix}$$

$$W_2 = \begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} & \omega_{13}^{(2)} \end{bmatrix}$$

Rede Neuronal Multicamada-*Feedforward*

Função XOR

✓ A mesma rede com pesos utilizando a função de ativação de **Heaviside**. Pretende-se mostrar que consegue **classificar a função XOR**



$$W_1 = \begin{bmatrix} 1 & -1 & -0.5 \\ -1 & 1 & -0.5 \end{bmatrix}$$

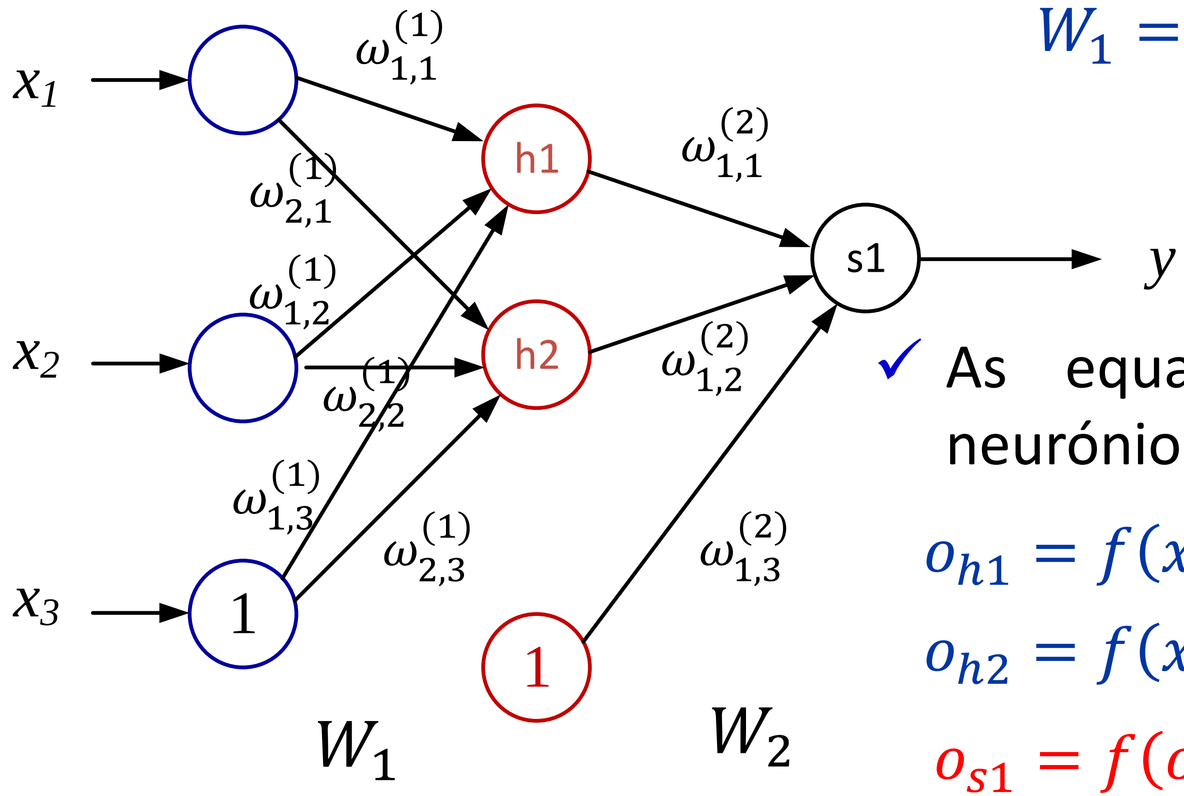
$$W_2 = \begin{bmatrix} 1 & 1 & -0.5 \end{bmatrix}$$

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

Rede Neuronal Multicamada-*Feedforward*

Função XOR

✓ Vamos utilizar a seguinte notação:



$$W_1 = \begin{bmatrix} \omega_{h11} & \omega_{h12} & \omega_{h13} \\ \omega_{h21} & \omega_{h22} & \omega_{h23} \end{bmatrix}$$

$$W_2 = \begin{bmatrix} \omega_{s11} & \omega_{s12} & \omega_{s13} \end{bmatrix}$$

✓ As equações da saída dos vários neurónios são as seguintes:

$$o_{h1} = f(x_1\omega_{h11} + x_2\omega_{h12} + \omega_{h13})$$

$$o_{h2} = f(x_1\omega_{h21} + x_2\omega_{h22} + \omega_{h23})$$

$$o_{s1} = f(o_{h1}\omega_{s11} + o_{h2}\omega_{s12} + \omega_{s13})$$

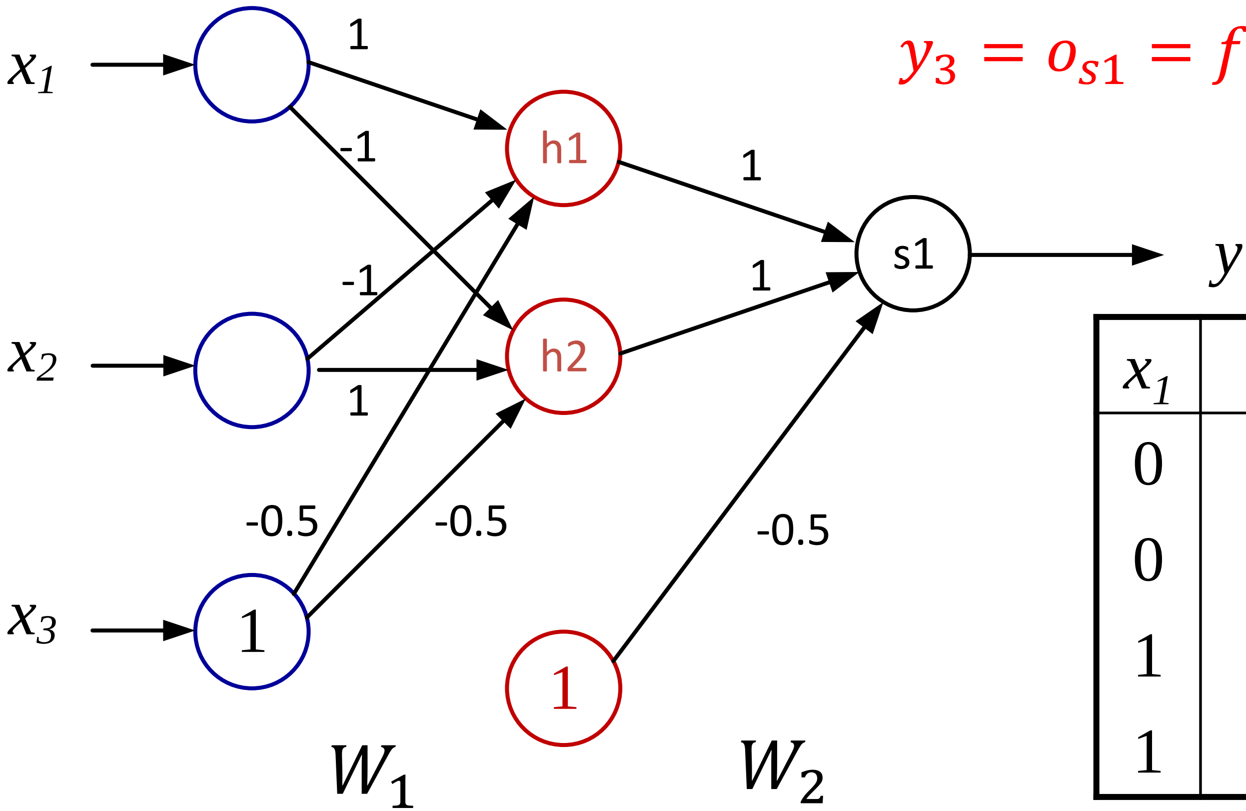
Rede Neuronal Multicamada-*Feedforward*

Função XOR

$$y_1 = o_{h1} = f(x_1\omega_{h11} + x_2\omega_{h12} + \omega_{h13})$$

$$y_2 = o_{h2} = f(x_1\omega_{h21} + x_2\omega_{h22} + \omega_{h23})$$

$$y_3 = o_{s1} = f(o_{h1}\omega_{s11} + o_{h2}\omega_{s12} + \omega_{s13})$$



x_1	x_2	x_3	y_1	y_2	y
0	0	1	**0**	**0**	**0**
0	1	1	**0**	**1**	**1**
1	0	1	**1**	**0**	**1**
1	1	1	**0**	**0**	**0**

Função XOR

- ✓ Podemos determinar as equações das retas para cada neurónio:

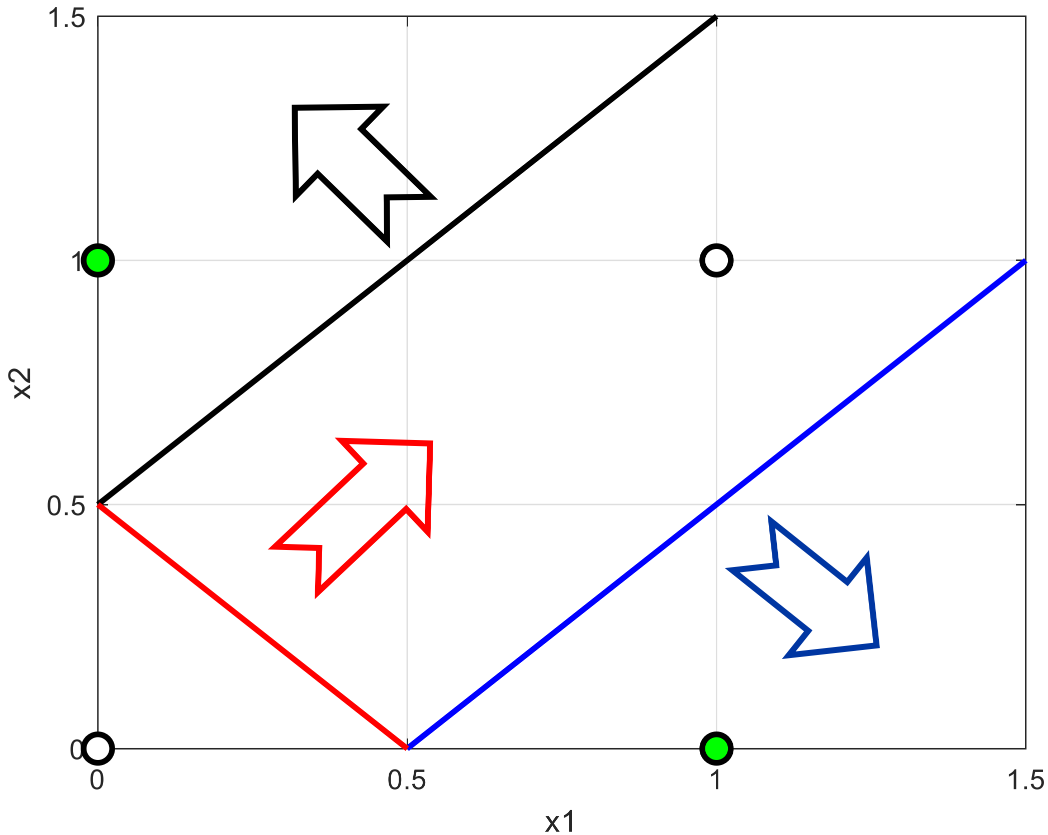
$$x_1\omega_{h11} + x_2\omega_{h12} + \omega_{h13} = 0 \Rightarrow x_2 = -\frac{\omega_{h11}}{\omega_{h12}}x_1 - \frac{\omega_{h13}}{\omega_{h12}}$$

$$x_1\omega_{h21} + x_2\omega_{h22} + \omega_{h23} = 0 \Rightarrow x_2 = -\frac{\omega_{h21}}{\omega_{h22}}x_1 - \frac{\omega_{h23}}{\omega_{h22}}$$

$$o_{h1}\omega_{s11} + o_{h2}\omega_{s12} + \omega_{s13} = 0 \Rightarrow o_{h2} = -\frac{\omega_{s11}}{\omega_{s12}}o_{h1} - \frac{\omega_{s13}}{\omega_{s12}}$$

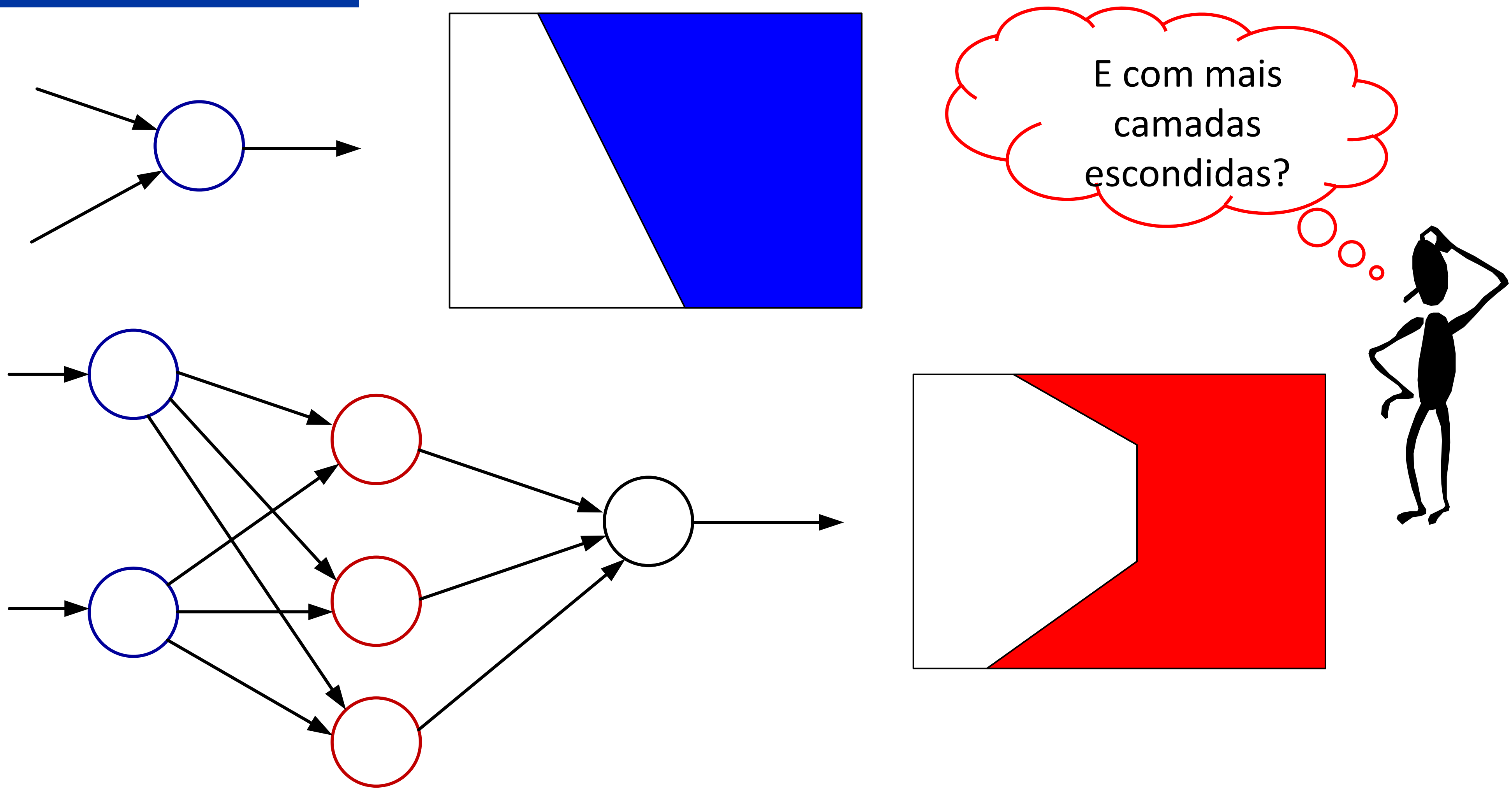
Função XOR

- Com estas equações podemos gerar uma figura para visualizar o resultado da classificação:



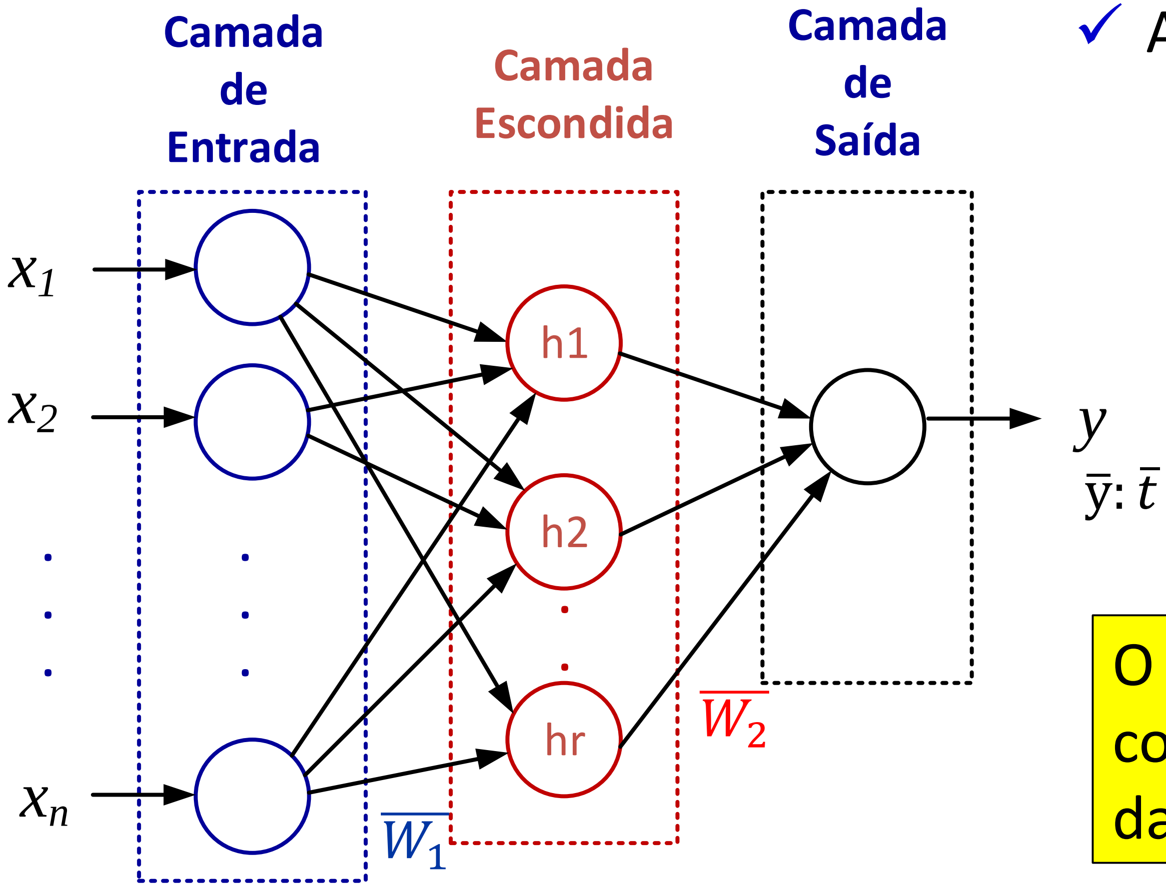
Rede Neuronal Multicamada-*Feedforward*

Regiões de Decisão



Problema de Aprendizagem da BPN

- Considere a seguinte rede n-r-m:



- Amostra de dados (padrões) i:

$$\bar{x_i} = (x_1, x_2, \cdots, x_n)$$

$$\boxed{1 < i \leq \mathrm{N}}$$

$$\boxed{E = \frac{1}{N}\sum_{i=1}^{N} e^i}$$

O **processo de aprendizagem** consiste no ajuste dos pesos, W, da rede **minimizando o erro E**.

- Número total de pesos: $l = (n \times r + r \times m)$

Problema de Aprendizagem da BPN

- Para cada neurónio j calcula-se:

$$f\left(g(\overline{x})\right) = f(x.w)$$

$$\downarrow$$

  - ✓ Pesos w gerados aleatoriamente com valores pequenos.

- A RN implementa uma **função composta: F**.

- O **Problema de Aprendizagem** consiste em **encontrar um conjunto de pesos w** de forma a que **F se aproxime o mais possível** a uma **função desejada Fd**.

✓ **Como se define Fd?**

Problema de Aprendizagem da BPN

- ✓ Fd é fornecida implicitamente a partir do conjunto de dados apresentados à entrada da RN:

$$\{(\overline{x_1}, \overline{t_1}), (\overline{x_2}, \overline{t_2}), \cdots, (\overline{x_N}, \overline{t_N})\}$$

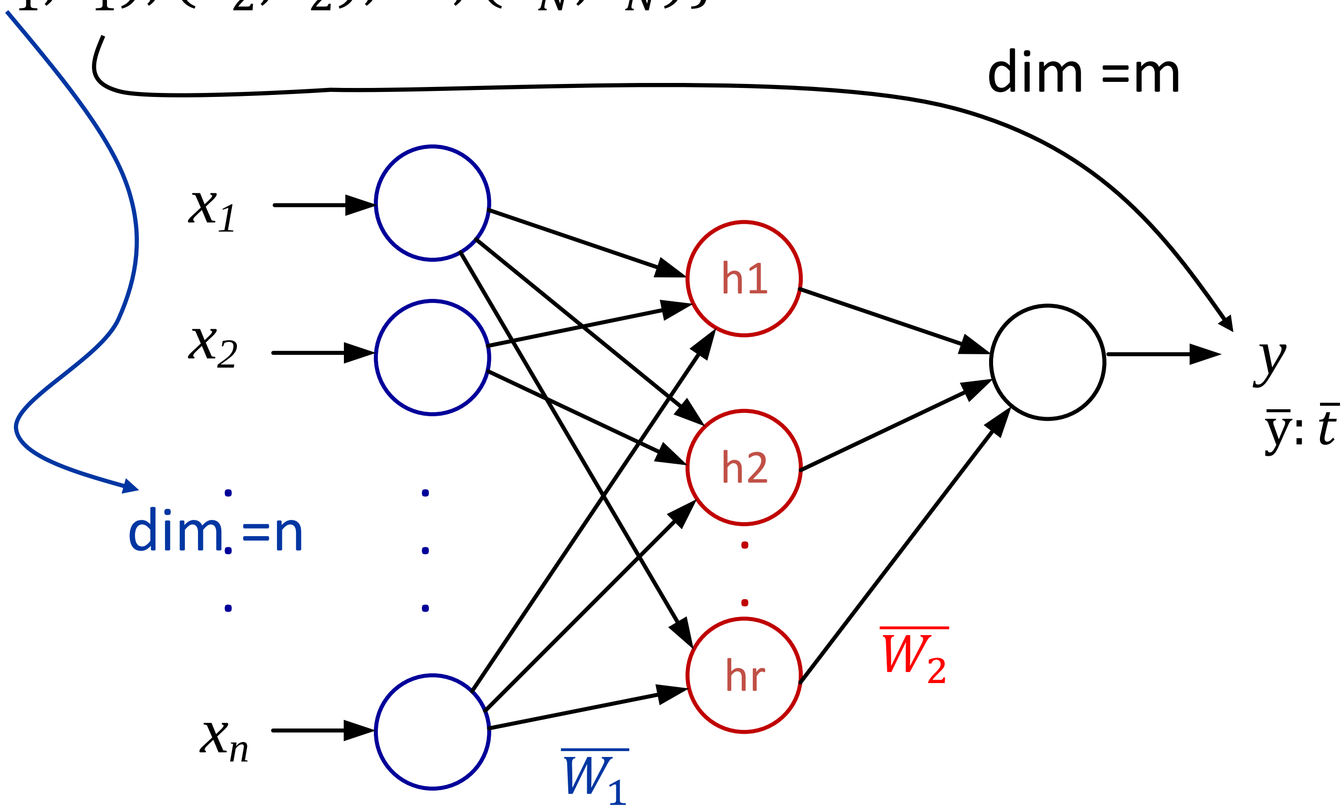
$$\overline{x_i} \rightarrow RN \rightarrow \overline{y_i}$$

$\overline{y_i}$ *comparada com* $\overline{t_i}$

- ✓ O objetivo da regra de aprendizagem é tentar fazer com que :

$$\overline{y_i} = \overline{t_i}$$

- ✓ **De que forma?**



RetroPropagação – BackPropagation

Problema de Aprendizagem da BPN

$$\overline{y_i} = \overline{t_i}$$

Minimizando a função do erro da RN:

$$E = \frac{1}{2}\sum_{i=1}^{N}(\overline{y_i} - \overline{t_i})^2$$

Usando o método do Gradiente Descendente

$$\nabla E = \left(\frac{\partial E}{\partial \omega_1}, \frac{\partial E}{\partial \omega_2}, \cdots, \frac{\partial E}{\partial \omega_l}\right)$$

Os pesos são atualizados com:

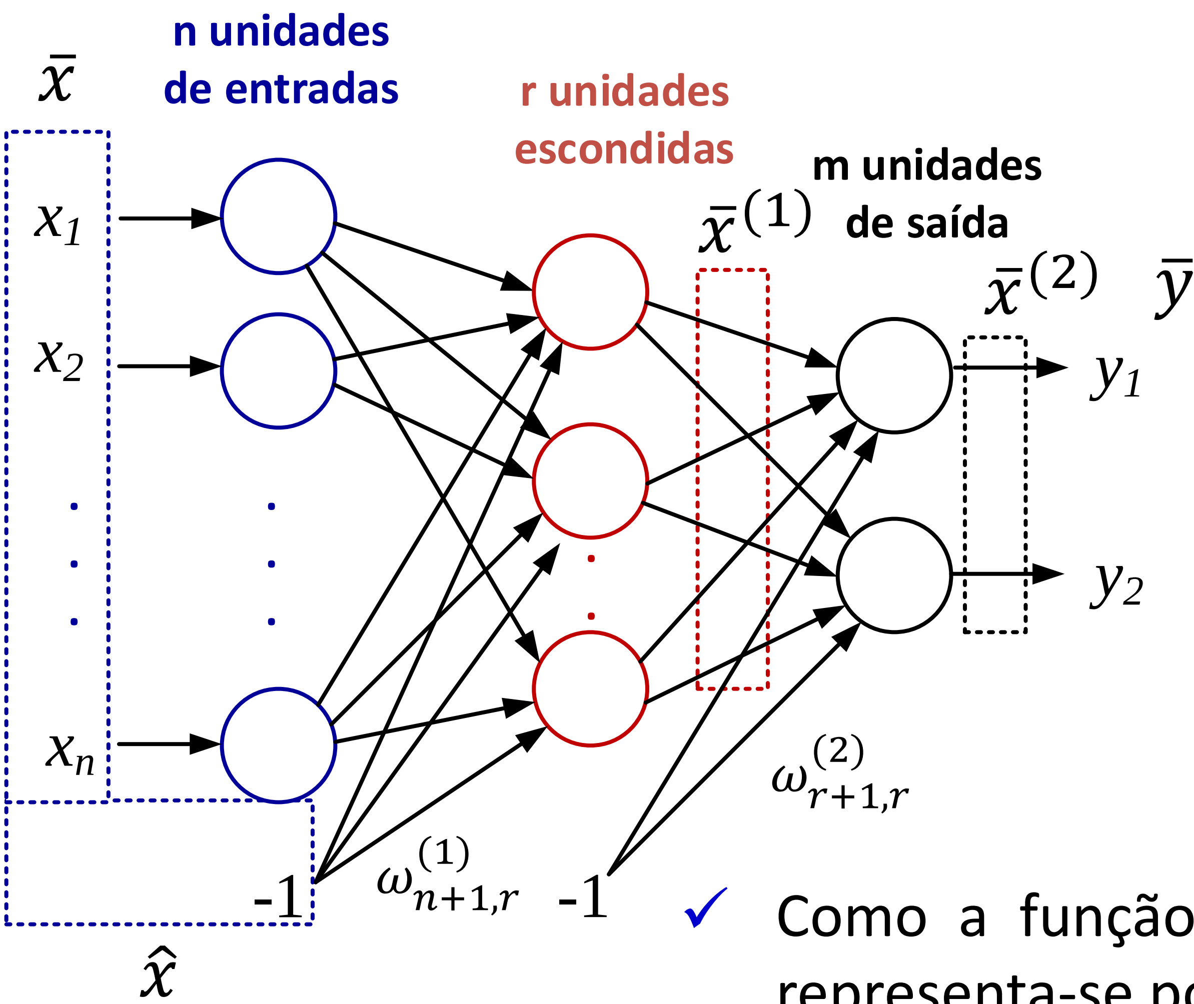
$$\Delta\omega_i = \alpha\left(\frac{\partial E}{\partial \omega_1}\right) \mathrm{i} = 1,2, \ldots l \quad 0 < \alpha \leq 1$$

Até que o erro seja nulo (realisticamente: aproximadamente nulo) :

$$\nabla E = 0$$

RetroPropagação – BackPropagation

Problema de Aprendizagem da BPN



$$\bar{x} = (x_1, x_2, \cdots, x_n)$$

$$\hat{x} = (x_1, x_2, \cdots, x_n, -1)$$

- ✓ **Excitação (entrada) da unidade escondida j:**

$$g(h_j) = \sum_{i=1}^{n+1} \hat{x}_i . \hat{\omega}^{(1)}_{ij}$$

- ✓ **Saída da unidade escondida j:**

$$x_j^{(1)} = f\left(g(h_j)\right)$$

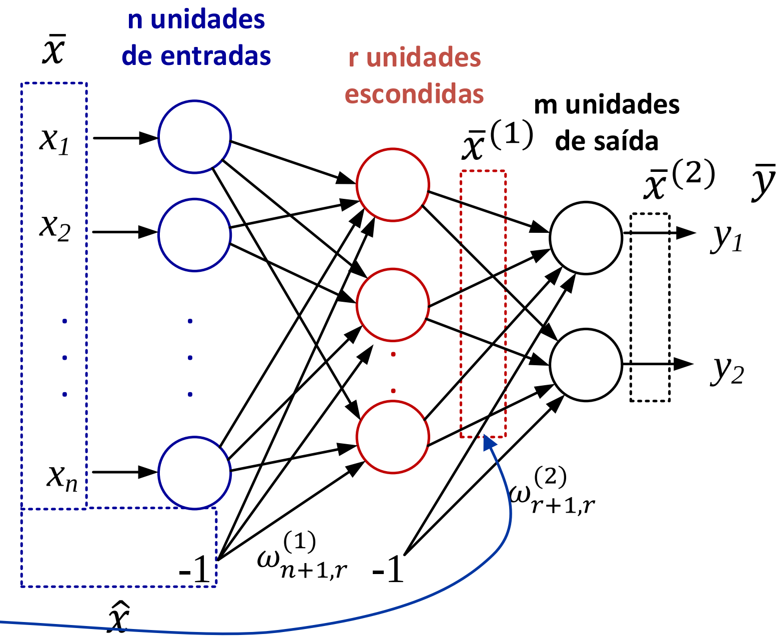
- ✓ Como a função f é uma sigmoide, representa-se por s: $x_j^{(1)} = s\left(g(h_j)\right)$

utad

RetroPropagação – BackPropagation

Problema de Aprendizagem da BPN

$$x_j^{(1)} = s\left(g(h_j)\right) \quad g(h_j) = \sum_{i=1}^{n+1} \hat{x}_i . \hat{\omega}_{ij}^{(1)}$$



$$\boxed{\therefore\ x_j^{(1)} = s\left(\sum_{i=1}^{n+1} \hat{x}_i . \hat{\omega}_{ij}^{(1)}\right)}$$

- ✓ **Entrada de todas as unidades da camada escondida:** $= \hat{x}\,\hat{\omega}_i$
- ✓ **Saída de todas as unidade da camada escondida:**

$$\boxed{\bar{x}^{(1)} = s(\hat{x} . \hat{\omega}_1\,)}$$

- ✓ **Entrada das unidades da camada de saída:**

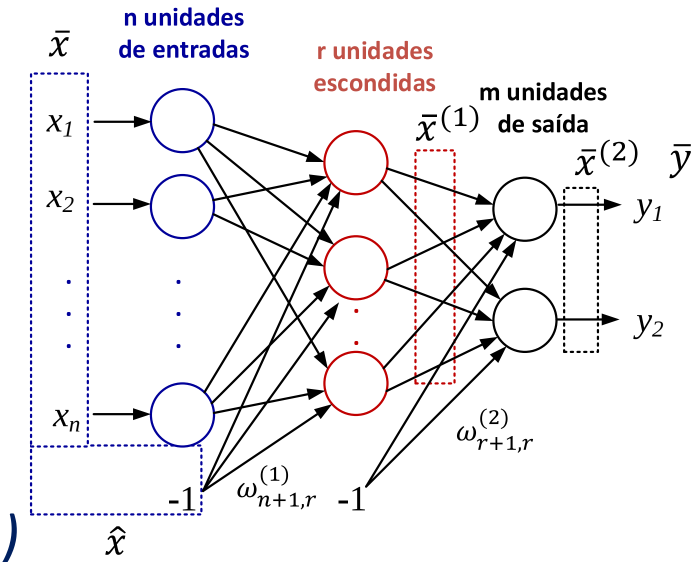
$$\boxed{\bar{x}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, \cdots, x_r^{(1)}, -1\right)}$$

- ✓ **Saída da camada de saída:**

$$\bar{x}^{(2)} = s\left(\hat{x}^{(1)} . \hat{\omega}_2\,\right)$$

RetroPropagação – BackPropagation

Algoritmo da RetroPropagação:



t = 1   // Contador de épocas

***while**(!(critério de paragem))*

Propagação Feedforward   / (1)

RetroPropagação para a camada de saída   / (2)

RetroPropagação para a camada escondida / (3)

Atualização dos pesos   / (4)

t=t+1

end while

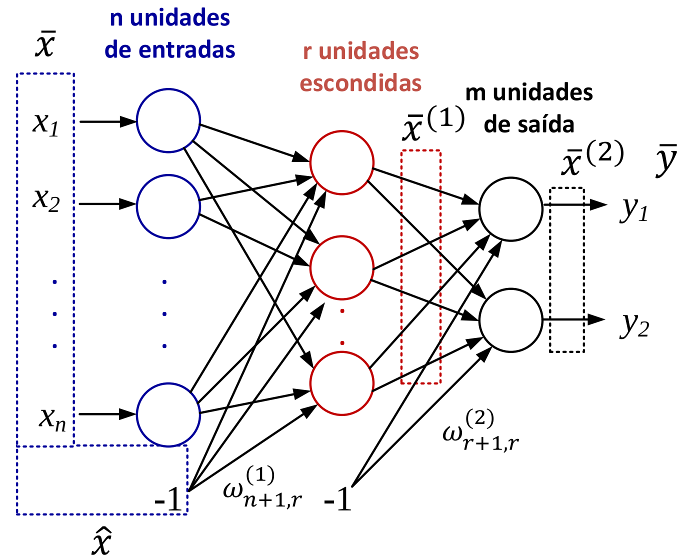
✓ Exemplos de critérios de paragem:

- Número de épocas;
- **E** abaixo de um limiar.

utad

RetroPropagação – BackPropagation

Algoritmo da RetroPropagação:



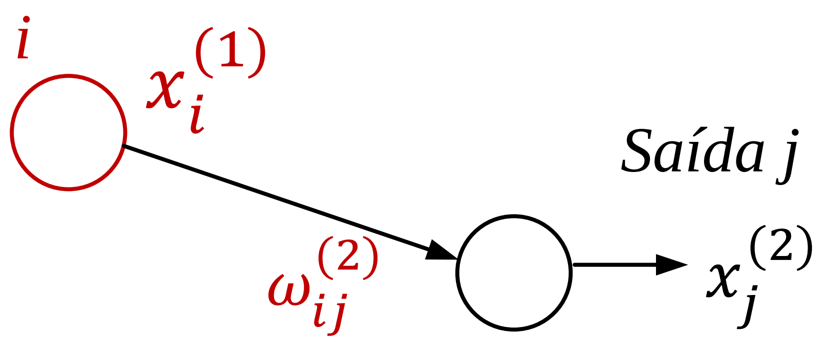
1. Propagação Feedforward :

✓ Padrão é apresentado à rede para cálculos das saídas:

$$\bar{x} \rightarrow RN \rightarrow \bar{x}^{(1)} \rightarrow \bar{x}^{(2)}$$

2. RetroPropagação do Erro para a camada de saída :

Escondido



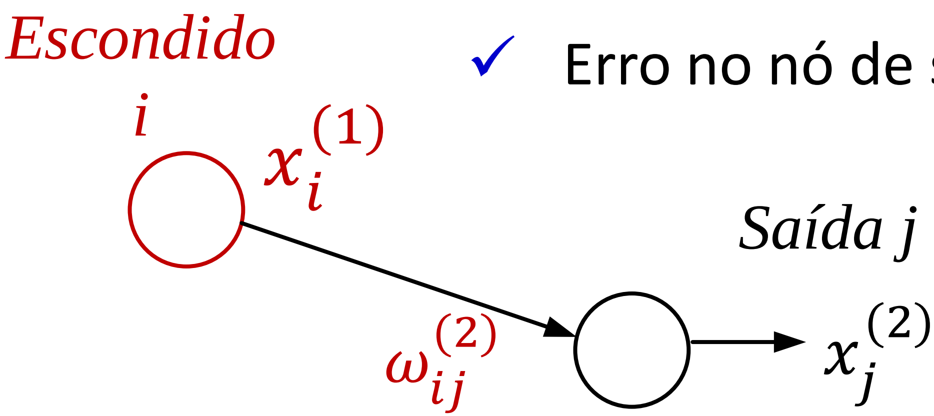
✓ Erro no nó de saída j:

$$\boxed{E = \frac{1}{2}\left(t_j - x_j^{(2)}\right)^2}$$

$$\boxed{-\frac{\partial E}{\partial \omega_{ij}^{(2)}} = ?}$$

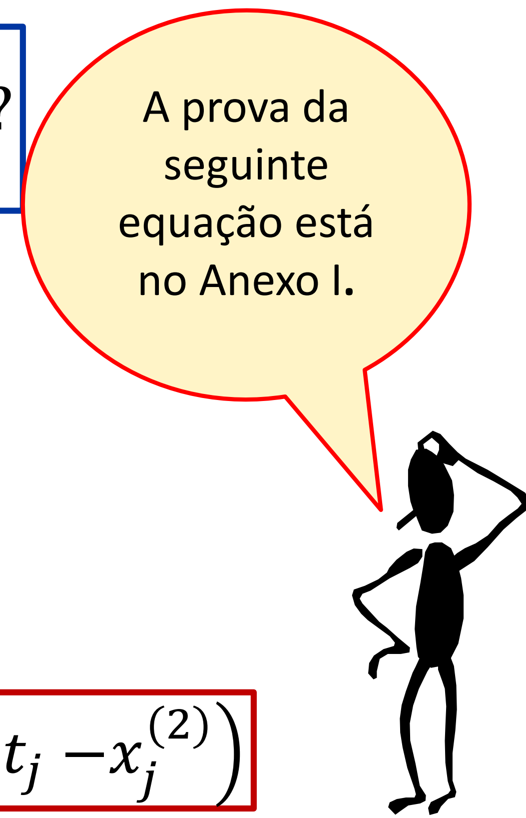
Algoritmo da RetroPropagação:

2. RetroPropagação do Erro para a camada de saída :



✓ Erro no nó de saída j: $$E = \frac{1}{2}\left(t_j - x_j^{(2)}\right)^2$$

$$-\frac{\partial E}{\partial \omega_{ij}{}^{(2)}} = ?$$



$$-\frac{\partial E}{\partial \omega_{ij}{}^{(2)}} = x_j^{(2)}\left(1 - x_j^{(2)}\right)\left(t_j - x_j^{(2)}\right) x_i^{(1)}$$

$$-\frac{\partial E}{\partial \omega_{ij}{}^{(2)}} = \delta_j^{(2)} x_i^{(1)}$$

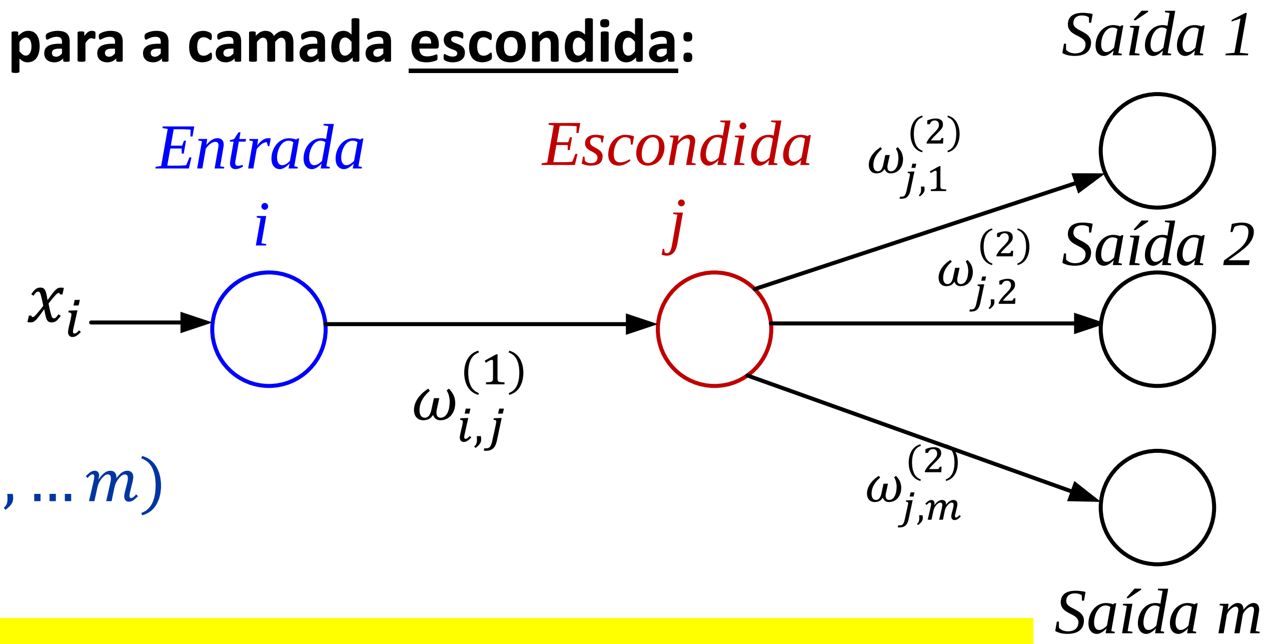
$$\delta_j^{(2)} = x_j^{(2)}\left(1 - x_j^{(2)}\right)\left(t_j - x_j^{(2)}\right)$$

RetroPropagação – BackPropagation

Algoritmo da RetroPropagação:

3. RetroPropagação do Erro para a camada escondida:

$$\boxed{\frac{\partial E}{\partial \omega_{ij}{}^{(1)}} = ?}$$



✓ Cada unidade escondida

$$j \xrightarrow[\omega_{j,q}^{(2)}]{} q = (1,2, \dots m)$$

✓ Pode-se provar que erro retroprogagado na unidade escondida j:

$$\boxed{-\frac{\partial E}{\partial \omega_{ij}{}^{(1)}} = \delta_j^{(1)} x_i}$$

$$\boxed{\delta_j^{(1)} = -x_j^{(1)} \left(1 - x_j^{(1)}\right) \sum_{q=1}^{m} \omega_{j,q}^{(2)} \delta_j^{(2)}}$$

Algoritmo da RetroPropagação:

4. Atualização dos pesos:

✓ Para os pesos conectando as **unidades escondidas às unidades de saída:**

$$\boxed{\omega_{i,j}^{(2)}(t+1) = \omega_{i,j}^{(2)}(t) + \Delta\omega_{i,j}^{(2)}(t) = \omega_{i,j}^{(2)}(t) + \alpha x_i^{(1)}(t)\delta_j^{(2)}(t)}$$

$i = 1,2,\dots,r+1$ j$= 1,2,\dots,m$

✓ Para os pesos conectando as **unidades de entrada às unidades escondidas**:

$$\boxed{\omega_{i,j}^{(1)}(t+1) = \omega_{i,j}^{(1)}(t) + \Delta\omega_{i,j}^{(1)}(t) = \omega_{i,j}^{(1)}(t) + \alpha x_i(t)\delta_j^{(1)}(t)}$$

$i = 1,2,\dots,n+1$ j$= 1,2,\dots,r$

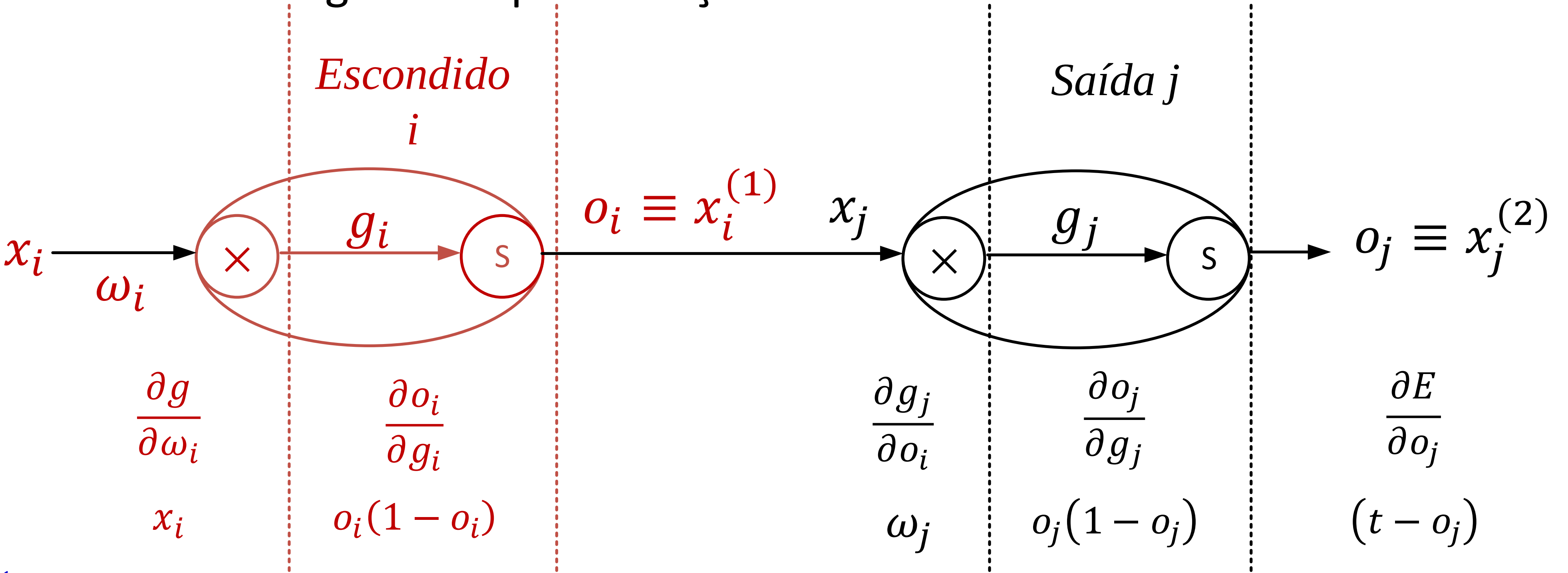
✓ Considerando o erro para todas as N amostras:

$$\Delta\omega_{i,j}^{(1)} = \Delta_1\omega_{i,j}^{(1)} + \Delta_2\omega_{i,j}^{(1)} + \cdots \Delta_N\omega_{i,j}^{(1)}$$

RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

✓ Considere a seguinte representação **de dois neurónios**:



✓ Pretende-se calcular os valores da Retropropagação considerando:

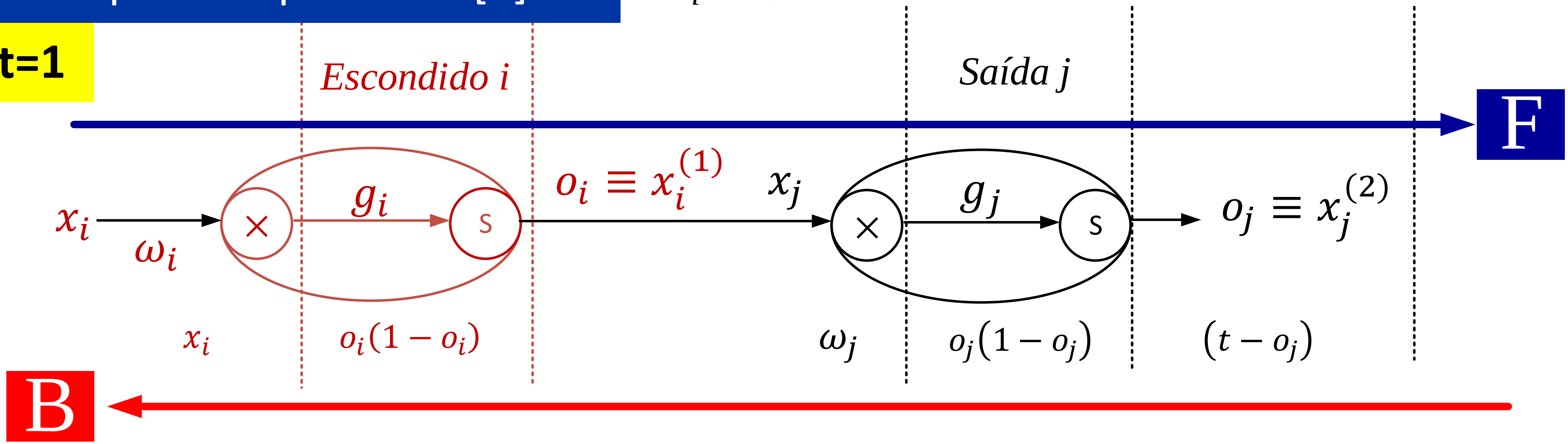
$x_i = 1$, $t = 1$ e $\alpha = 8$

*[3] Berwick B., (2020), MIT Classes Notes. http://web.mit.edu/6.034/wwwbob/, acedido em 25-3-2020

RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]: $x_i = 1$, $t=1$ e $\alpha=8$ $\quad s(0) = 0.5$

t=1



Passo 1: *Forward Propagation (F)*

$g_i = x_i \times \omega_i = 1\times 0=0$ $\quad o_i = x_j = s(0) = 0.5$ $\quad g_j = x_j \times \omega_j = 0.5\times 0=0$ $\quad o_j = s(0) = 0.5$

Passo 2: *Backward Propagation (B), Calcular δ_j para a camada de saída*

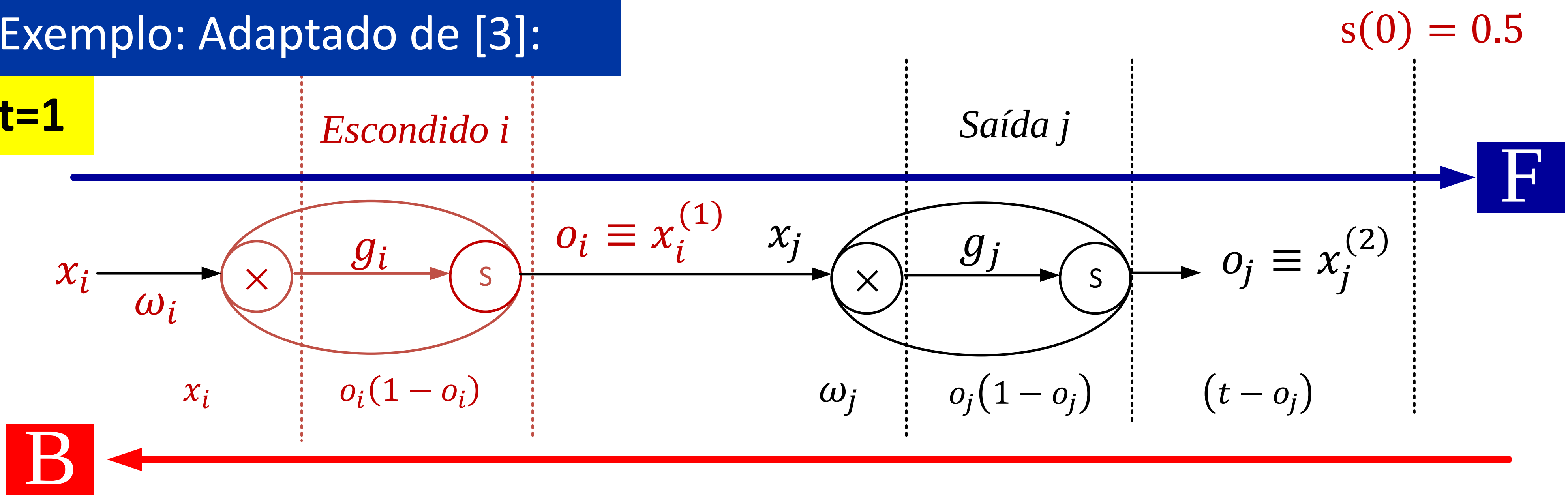
$$\delta_j = o_j \times (1-o_j)(t-o_j) = 0.5\times(1-0.5)(1-0.5)=0.125$$

$$\Delta\omega_j = \alpha \times x_j \times \delta_j = 8\times 0.5\times 0.125 = 0.5$$

RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

t=1



Passo 3: *Backward Propagation (B), Calcular δ_i para a camada escondida*

$$\delta_i = o_i \times (1-o_i)\omega_j\delta_j = 0.5\times(1-0.5)\times 0\times 0.125 = 0$$

$$\Delta\omega_i = \alpha \times x_i \times \delta_i = 8\times 1\times 0 = 0$$

$$g_i = x_i \times \omega_i = 1\times 0 = 0$$

$$o_i = x_j = s(0) = 0.5$$

Passo 4: *Calcular os novos pesos*

$$\omega_i = \omega_i + \Delta\omega_i = 0 + 0 = 0$$

$$\omega_j = \omega_j + \Delta\omega_j = 0 + 0.5 = 0.5$$

RetroPropagação – BackPropagation

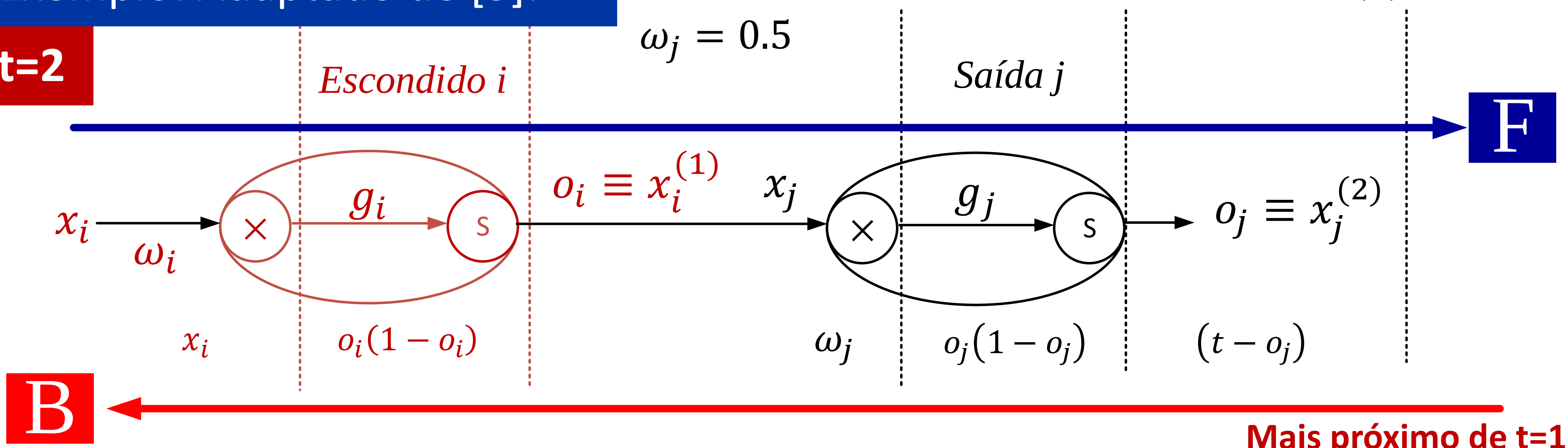
Exemplo: Adaptado de [3]:

t=2

$\omega_i = 0$

$\omega_j = 0.5$

$s(0) = 0.5$



Mais próximo de t=1

Passo 1: *Forward Propagation (F)*

$g_i = x_i \times \omega_i = 1\times0=0$ $\quad o_i = x_j = s(0) = 0.5$ $\quad g_j = x_j \times \omega_j = 0.5\times0.5=0.25$ $\quad o_j = s(0.25) = 0.5622$

Passo 2: *Backward Propagation (B), Calcular δ_j para a camada de saída*

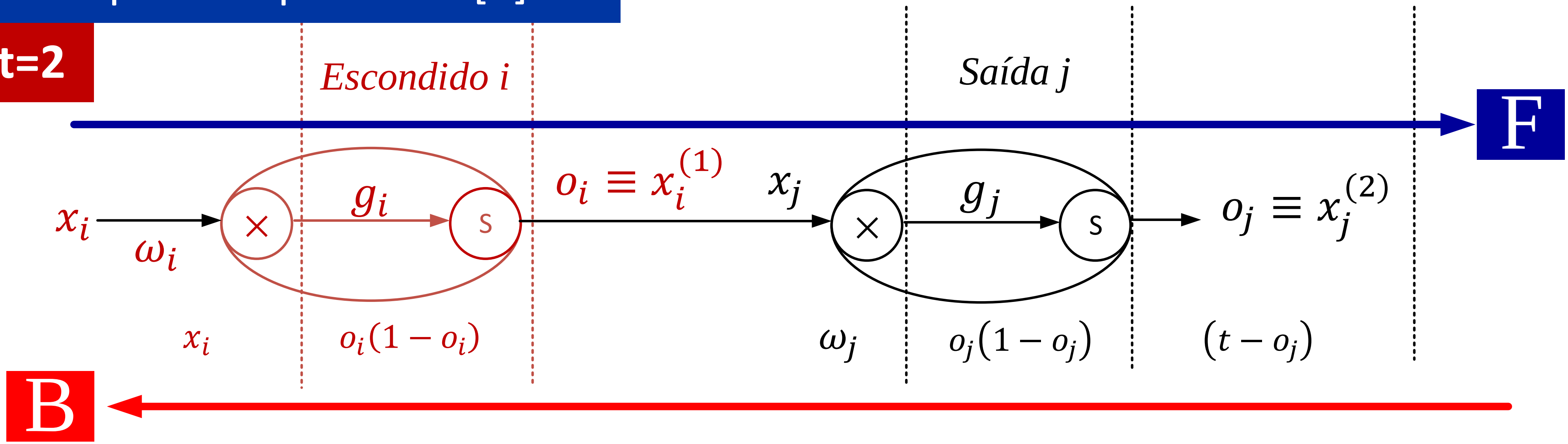
$$\delta_j = o_j \times (1-o_j)(t-o_j) = 0.5622\times(1-0.5622)(1-0.5622)=0.10776$$

$$\Delta\omega_j = \alpha \times x_j \times \delta_j = 8\times0.5\times0.10776 = 0.43104$$

RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

t=2



Passo 3: ***Backward Propagation (B), Calcular δ_i para a camada escondida***

$$\delta_i = o_i \times (1-o_i)\omega_j\delta_j = 0.5\times(1-0.5)\times 0.5\times 0.10776 = 0.01347$$

$$g_i = x_i \times \omega_i = 1\times 0 = 0$$

$$\Delta\omega_i = \alpha \times x_i \times \delta_i = 8\times 1\times 0.01347 = 0.1072$$

$$o_i = x_j = s(0) = 0.5$$

Passo 4: ***Calcular os novos pesos***

$$\omega_i = \omega_i + \Delta\omega_i = 0 + 0.1072 = 0.0172 \qquad \omega_j = \omega_j + \Delta\omega_j = 0.5 + 0.43104 = 0.943104$$

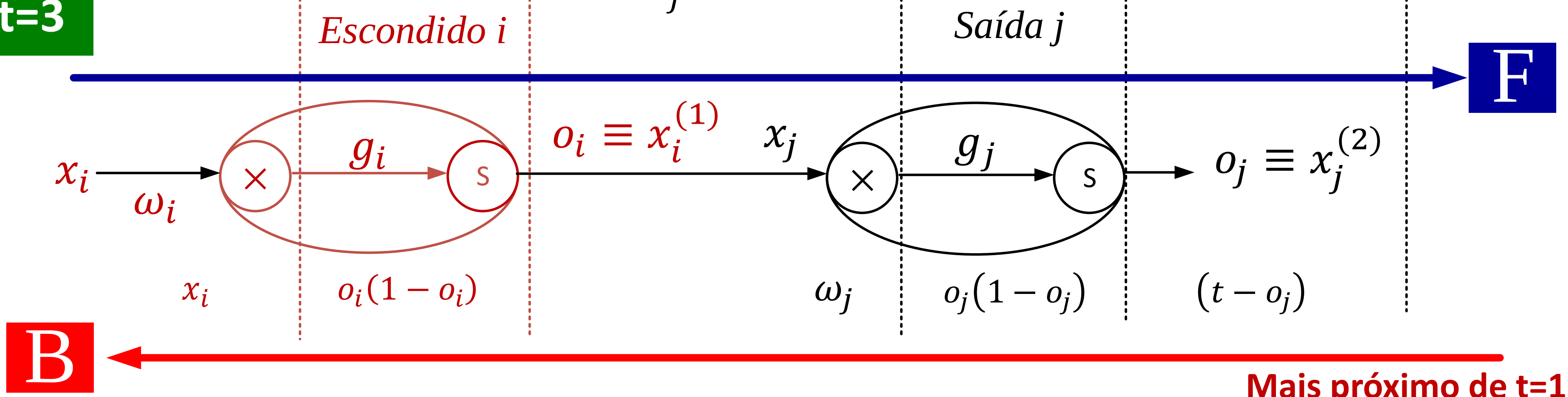
RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

t=3

$\omega_i = 0.1072$

$\omega_j = 0.94310$



Mais próximo de t=1

Passo 1: *Forward Propagation (F)*

$g_i = x_i \times \omega_i = 1 \times 0.1072 = 0.1072$

$o_i = x_j = s(0.1072) = 0.5267$

$g_j = x_j \times \omega_j = 0.5267 \times 0.9431 = 0.4968$

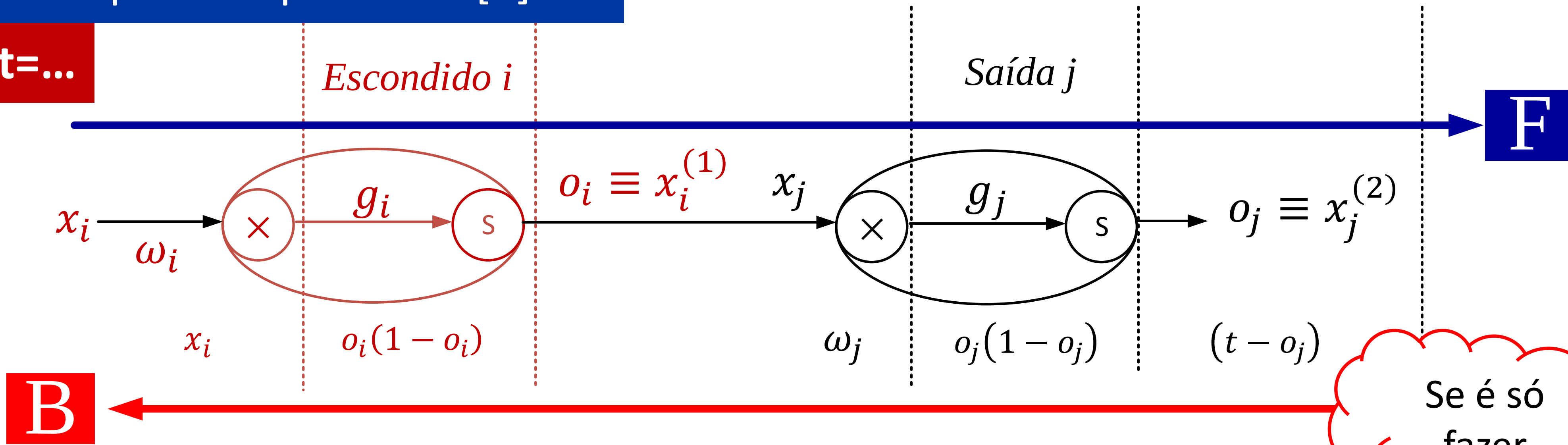
$o_j = s(0.4968) = 0.62171$

E assim sucessivamente até à iteração t=….

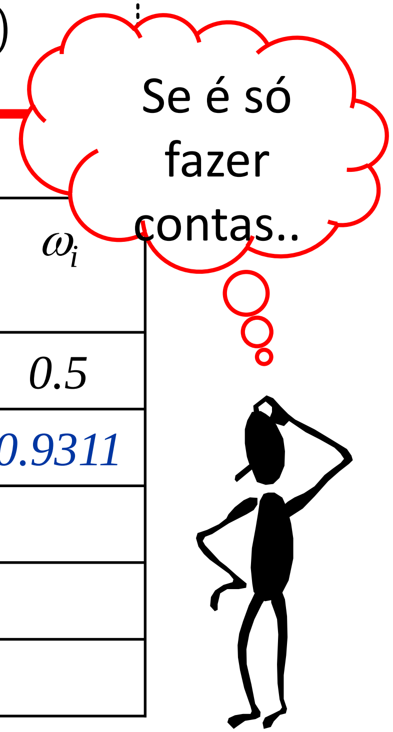
RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

t=...



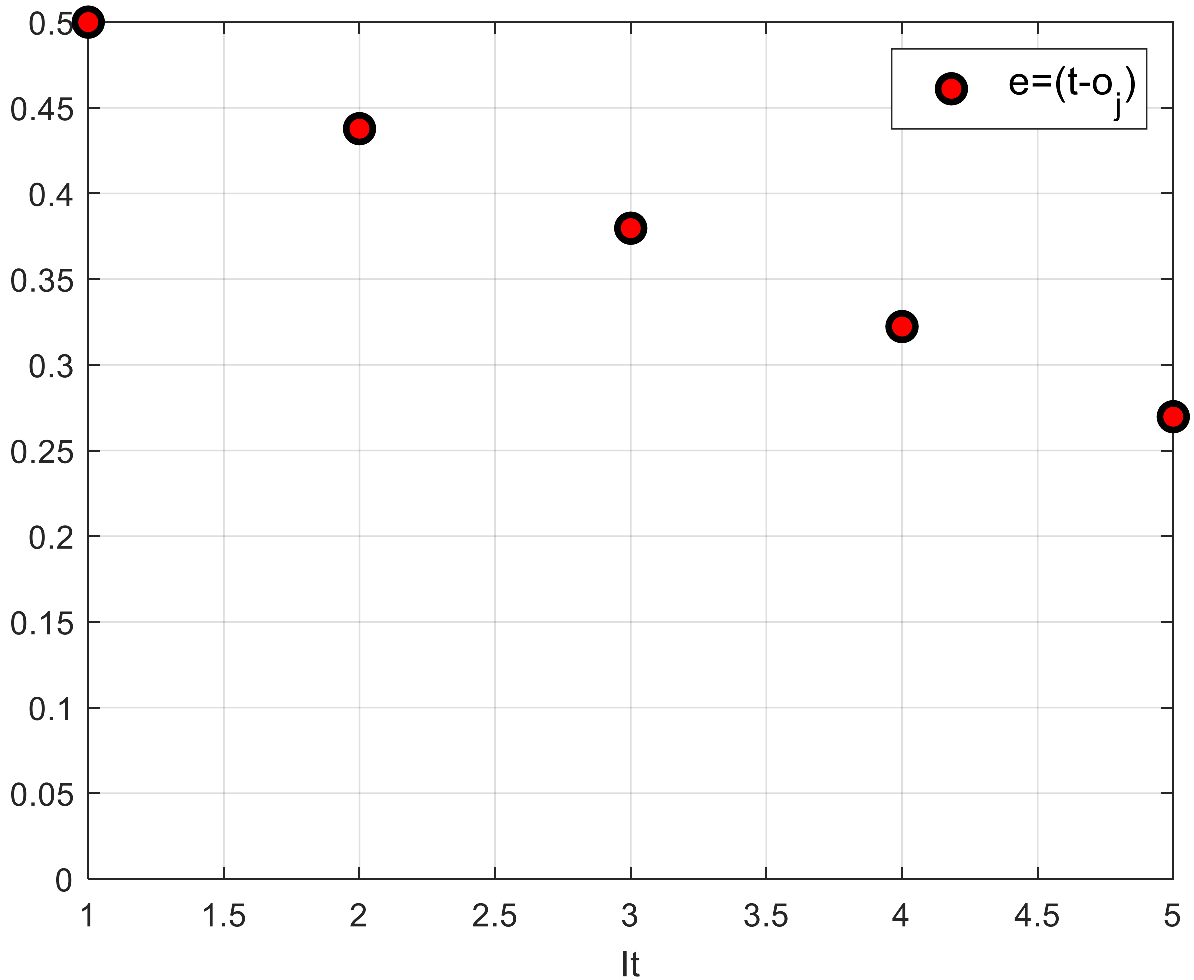
It	g_i	$o_i = x_j$	g_j	o_j	δ_j	$\Delta\omega_j$	δ_i	$\Delta\omega_i$	ω_j	ω_i
1	*0*	*0.5*	*0*	*0.5*	*0.125*	*0.5*	*0*	*0*	*0*	*0.5*
2	*0*	*0.5*	*0.25*	*0.5622*	*0.1078*	*0.4311*	*0.0135*	*0.1078*	*0.178*	*0.9311*
3										
4										
5										

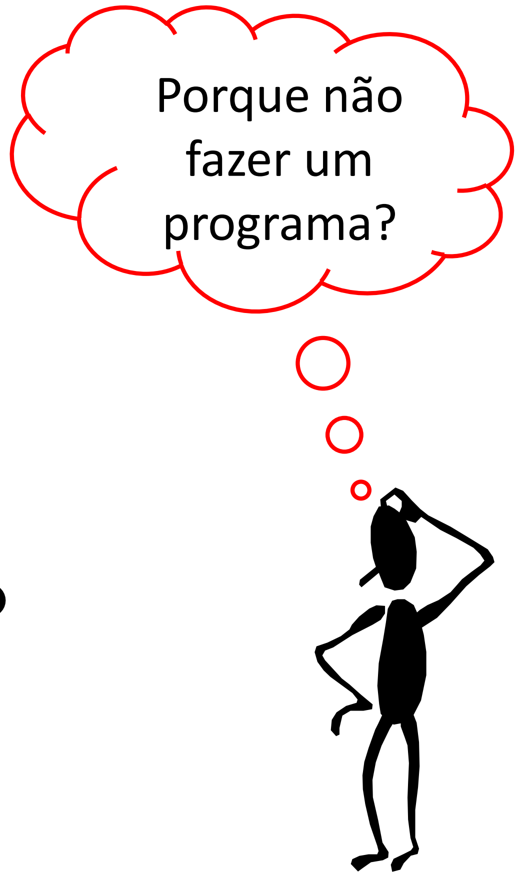


RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

t=5

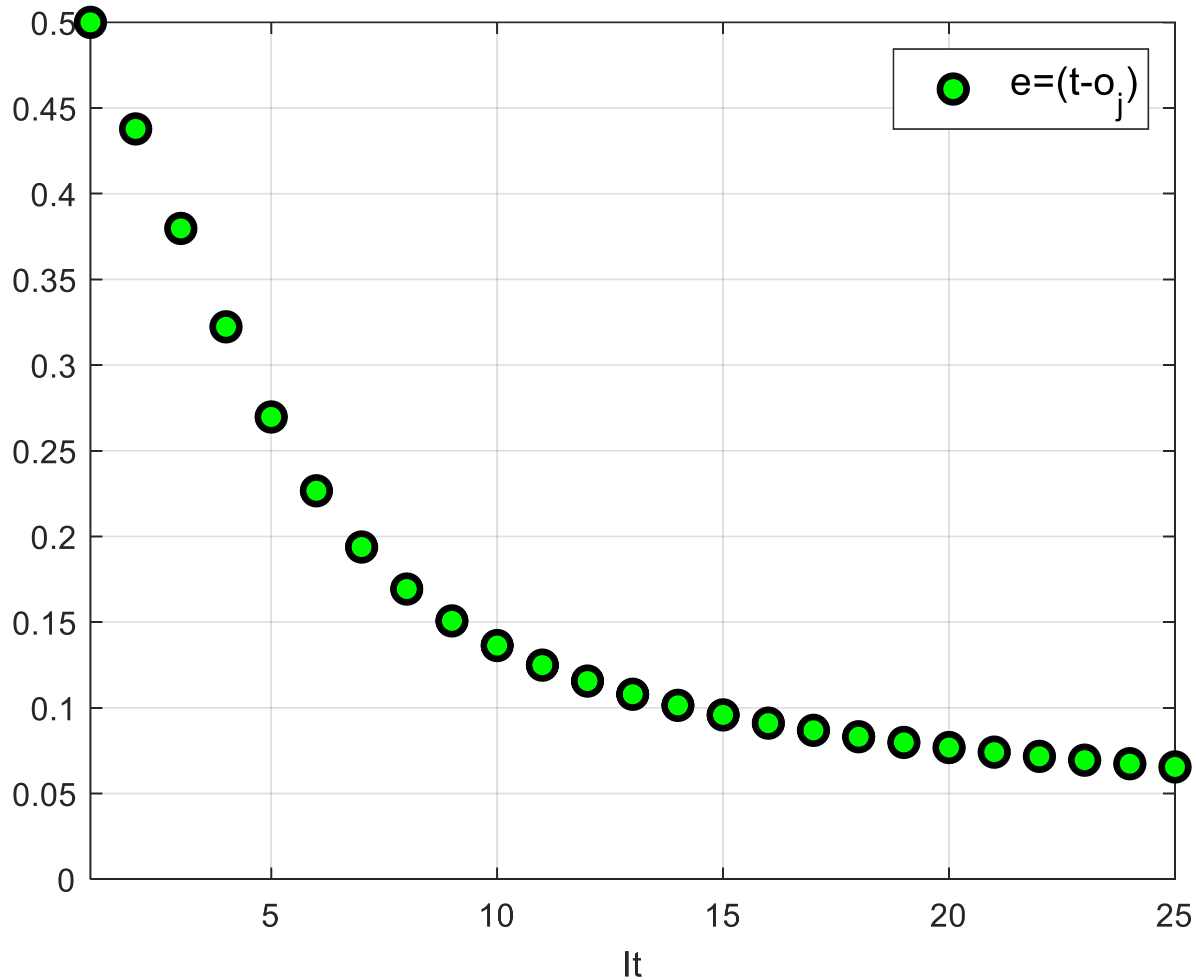


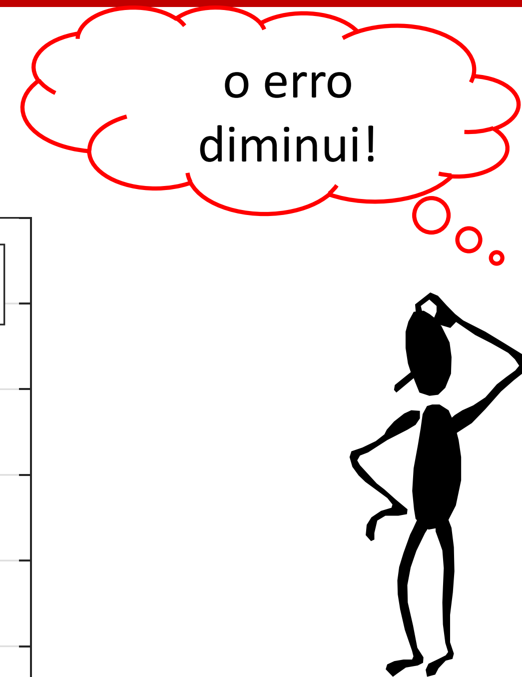


RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

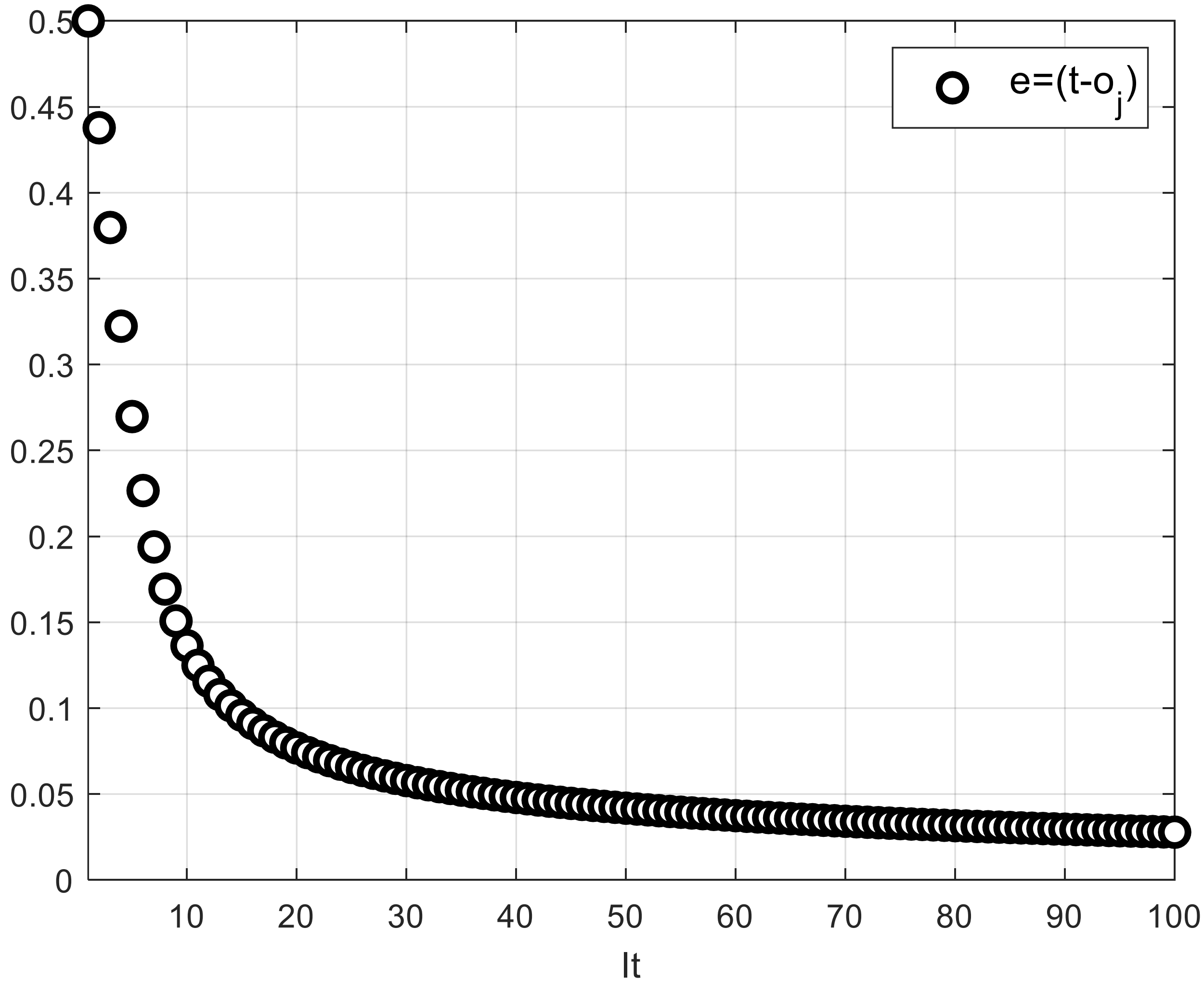
t=25





Exemplo: Adaptado de [3]:

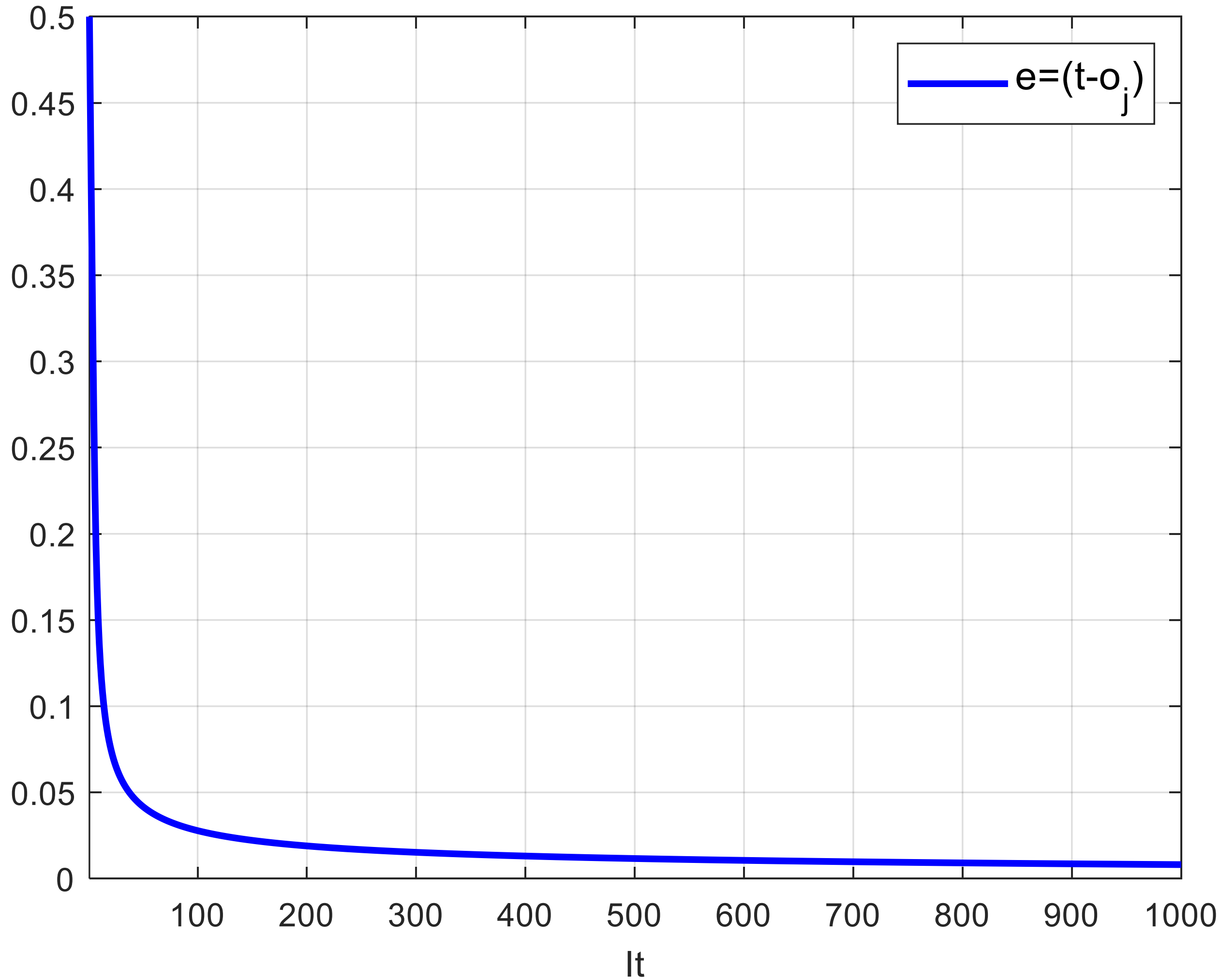
t=100



RetroPropagação – BackPropagation

Exemplo: Adaptado de [3]:

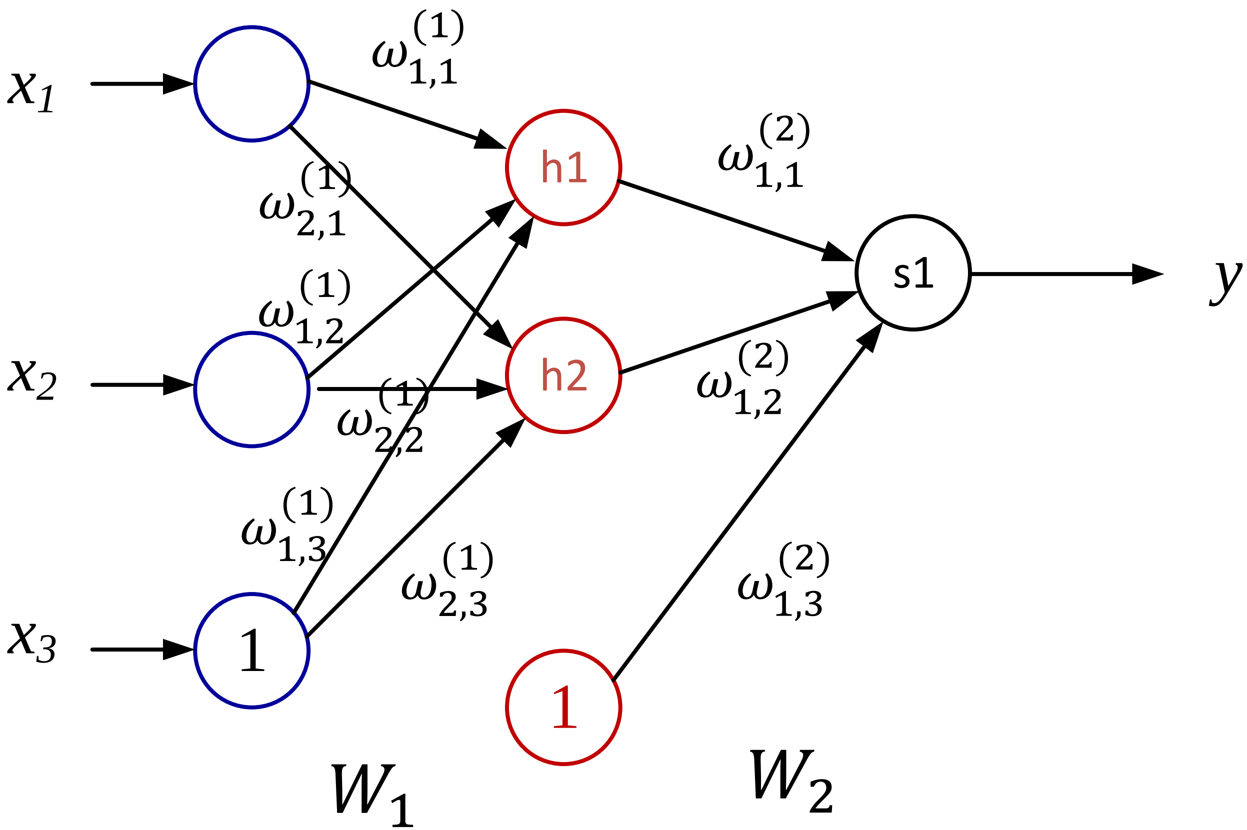
t=1000



RetroPropagação – BackPropagation

Exemplo: Problema XOR

✓ Considere a seguinte rede *Feedforward* para aprender a função lógica XOR com duas entradas: x1 e x2. A terceira entrada representa a entrada de bias.

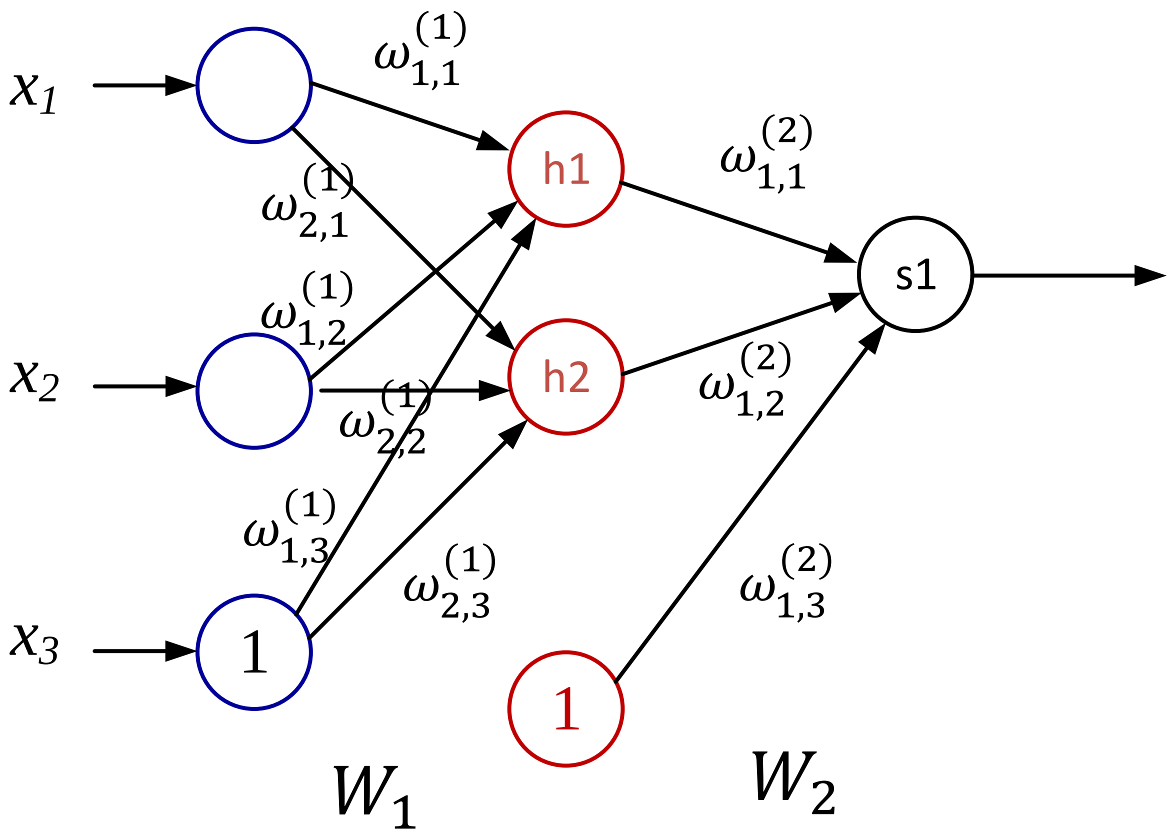


x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

RetroPropagação – BackPropagation

Exemplo: Problema XOR

✓ **<u>Por conveniência</u>** adotou-se a seguinte representação para os pesos e para as matrizes da camada de entrada e de saída:



$$W_1 = \begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} & \omega_{13}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} & \omega_{23}^{(1)} \end{bmatrix}$$

$$dim(W_1) - 2\times 3$$

$$r_h \, x \, n_i$$

$$W_2 = \begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} & \omega_{13}^{(2)} \end{bmatrix}$$

$$\dim(W_2) - 1x3$$

$$m_o \, x \, r_h$$

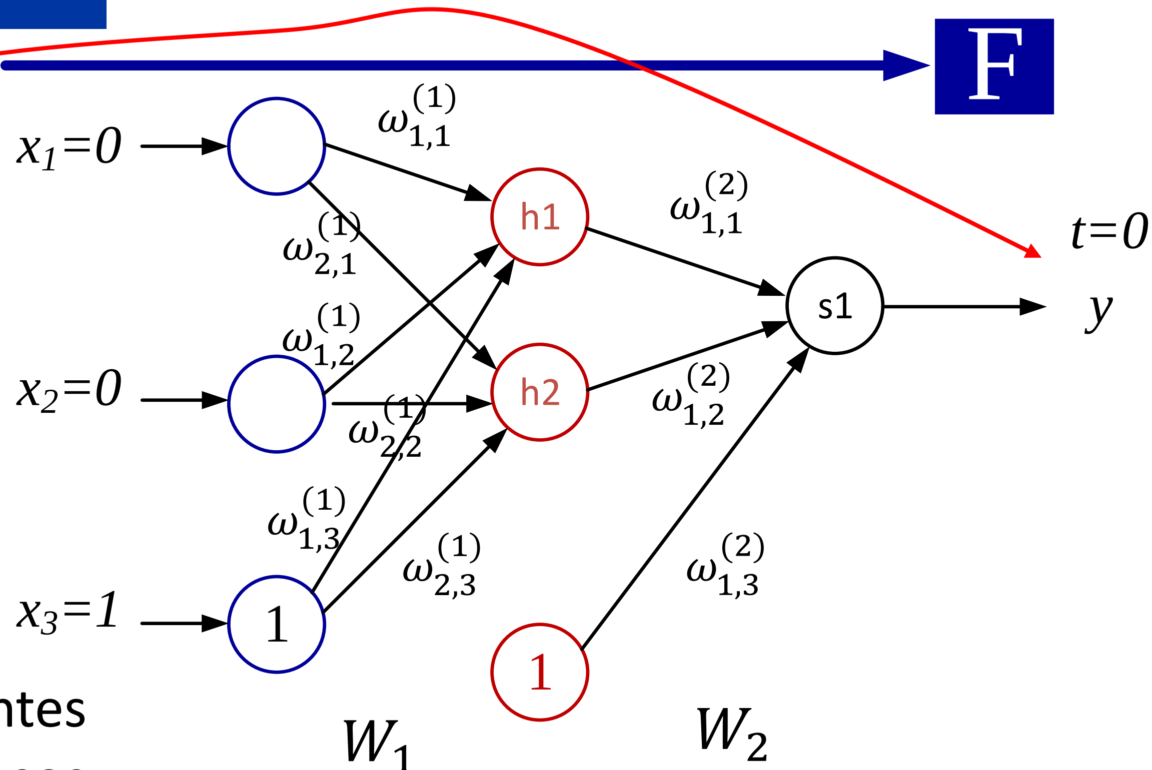
RetroPropagação – BackPropagation

Exemplo: Problema XOR

t=1, p=1

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

α=0.9, f- sigmoide



- ✓ Considerem-se os seguintes valores para os pesos inicializados aleatoriamente entre -1 e 1 :

$$W_1 = \begin{bmatrix} 0.5099 & -0.1152 & -0.2815 \\ -0.5144 & 0.3756 & 0.4727 \end{bmatrix} \quad W_2 = \begin{bmatrix} -0.2106 & 0.3668 & 0.4081 \end{bmatrix}$$

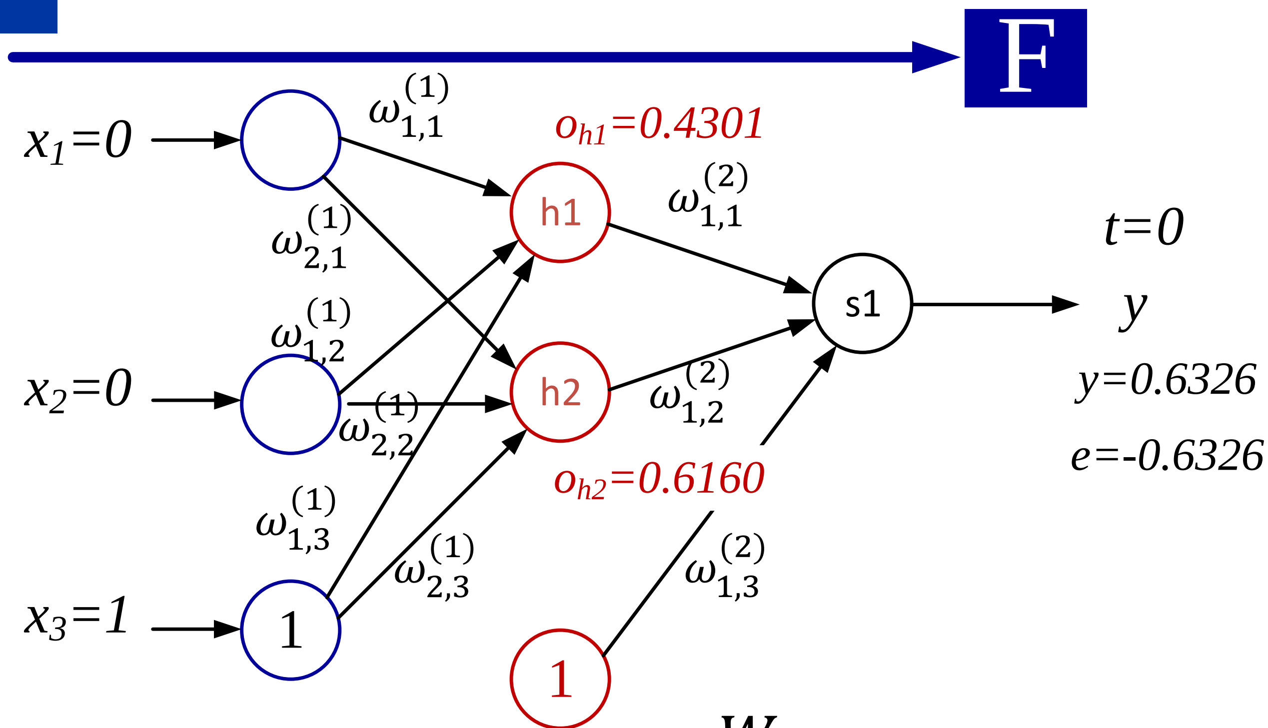
RetroPropagação – BackPropagation

Exemplo: Problema XOR

t=1, p=1

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

α=0.9



W_1 W_2

$$o_{h1} = f(g_{h1}) = f(\omega_{11}.x_1 + \omega_{12}.x_2 + \omega_{13}.x_3)$$

$$o_{h1} = f(\omega_{11}.0 + \omega_{12}.0 + \omega_{13}.1)$$

$$o_{h1} = f(-0.2815) = 0.4301$$

$$W_1 = \begin{bmatrix} 0.5099 & -0.1152 & -0.2815 \\ -0.5144 & 0.3756 & 0.4727 \end{bmatrix}$$

$$W_2 = [-0.2106 \quad 0.3668 \quad 0.4081]$$

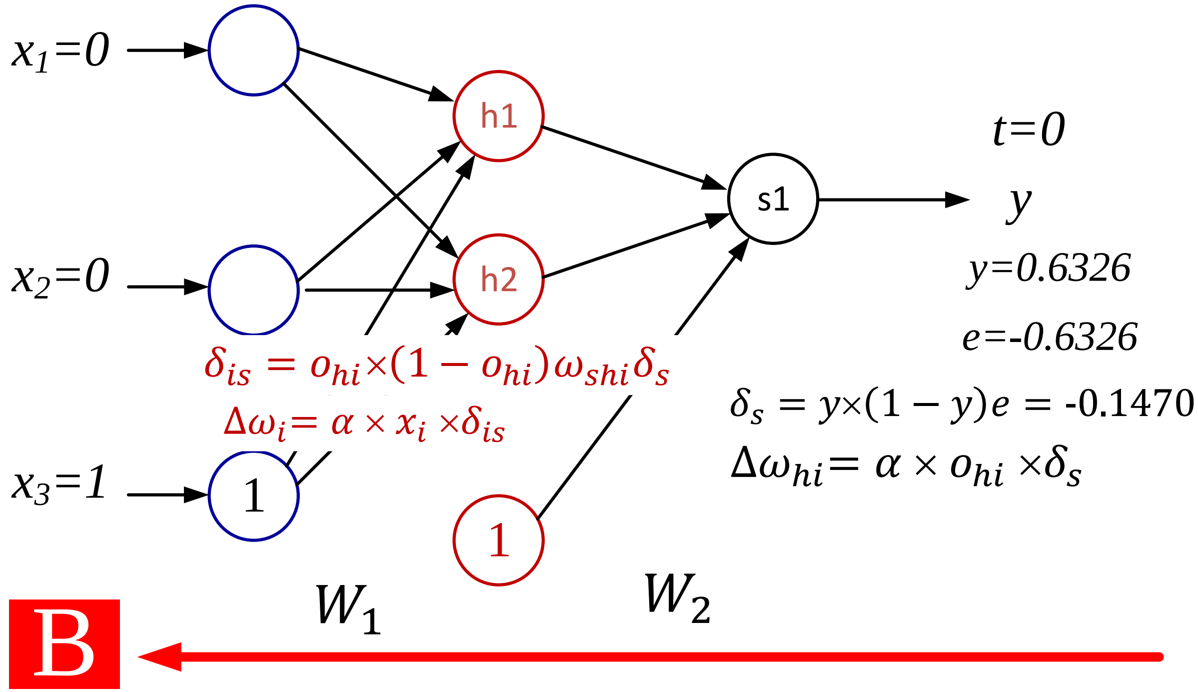
RetroPropagação – BackPropagation

Exemplo: Problema XOR

t=1, p=1

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

α=0.9



$$W_{1old} = \begin{bmatrix} 0.5099 & -0.1152 & -0.2815 \\ -0.5144 & 0.3756 & 0.4727 \end{bmatrix}$$

$$\delta W_1 = \begin{bmatrix} 0 & 0 & 0.0068 \\ 0 & 0 & -0.0115 \end{bmatrix}$$

$$W_{1new} = \begin{bmatrix} 0.5099 & -0.1152 & -0.2747 \\ -0.5144 & 0.3756 & 0.4612 \end{bmatrix}$$

$$W_{2\_old} = [-0.2106 \quad 0.3668 \quad 0.4081]$$

$$\delta W_2 = [-0.0569 \quad -0.0815 \quad -0.1323]$$

$$W_{2\_new} = [-0.2675 \quad 0.2853 \quad 0.2758]$$

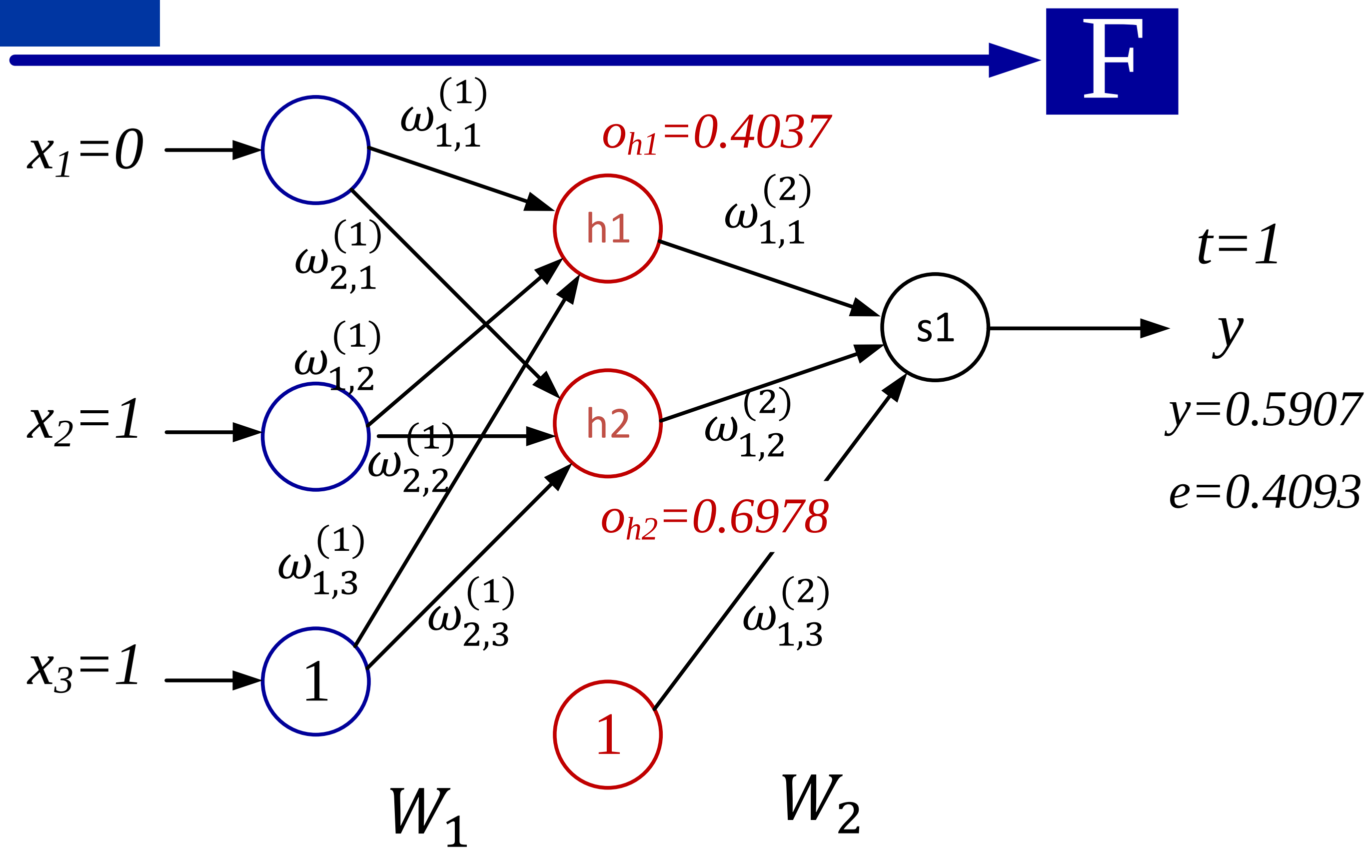
RetroPropagação – BackPropagation

Exemplo: Problema XOR

t=1, p=2

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

α=0.9



$$W_1 = \begin{bmatrix} 0.5099 & -0.1152 & -0.2747 \\ -0.5144 & 0.3756 & 0.4612 \end{bmatrix}$$

$$W_{2\_new} = [-0.2675 \quad 0.2853 \quad 0.2758]$$

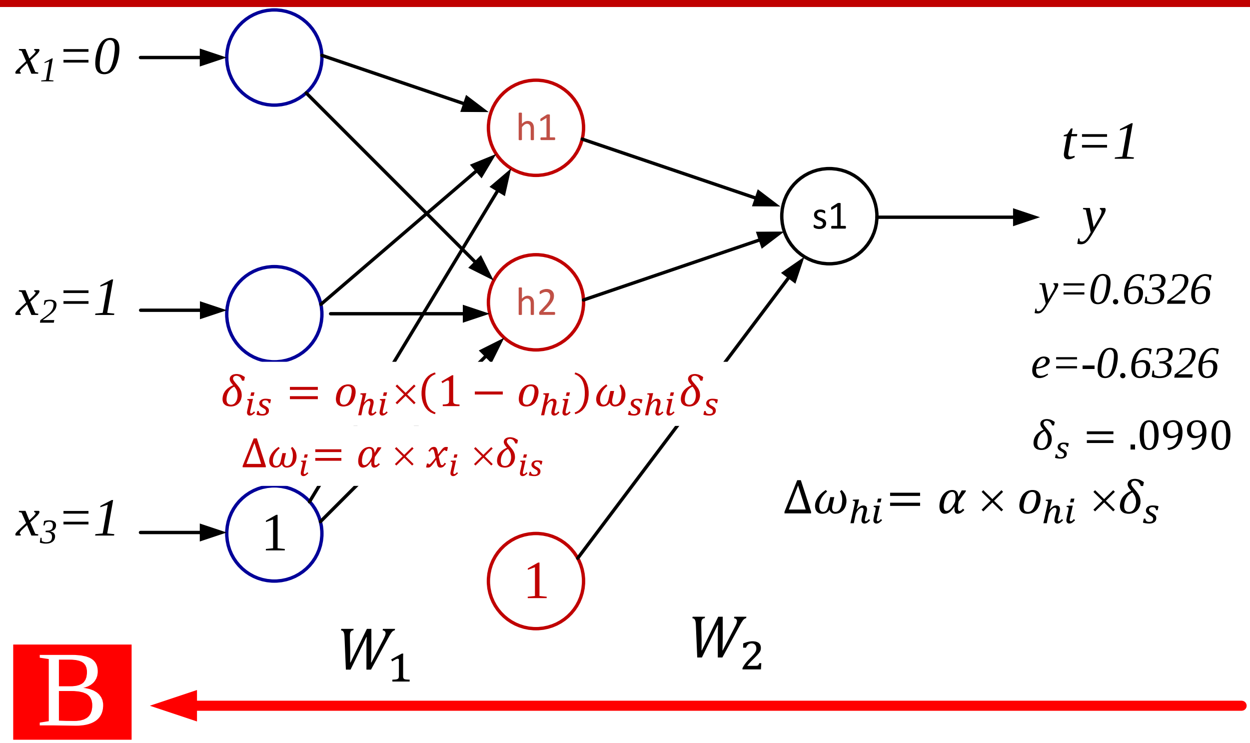
RetroPropagação – BackPropagation

Exemplo: Problema XOR

t=1, p=2

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

α=0.9



$$W_{1old} = \begin{bmatrix} 0.5099 & -0.1152 & -0.2747 \\ -0.5144 & 0.3756 & 0.4612 \end{bmatrix}$$

$$\delta W_1 = \begin{bmatrix} 0 & -0.0057 & -0.0057 \\ 0 & 0.0054 & 0.0054 \end{bmatrix}$$

$$W_{1new} = \begin{bmatrix} 0.5099 & -0.1209 & -0.2804 \\ -0.5144 & 0.3810 & 0.4666 \end{bmatrix}$$

$$W_{2\_old} = [-0.2675 \quad 0.2853 \quad 0.2758]$$

$$\delta W_2 = [0.0360 \quad 0.0621 \quad 0.0890]$$

$$W_{2\_new} = [-0.2315 \quad 0.3475 \quad 0.3648]$$

t=1, p=3,

t=1,p=4,

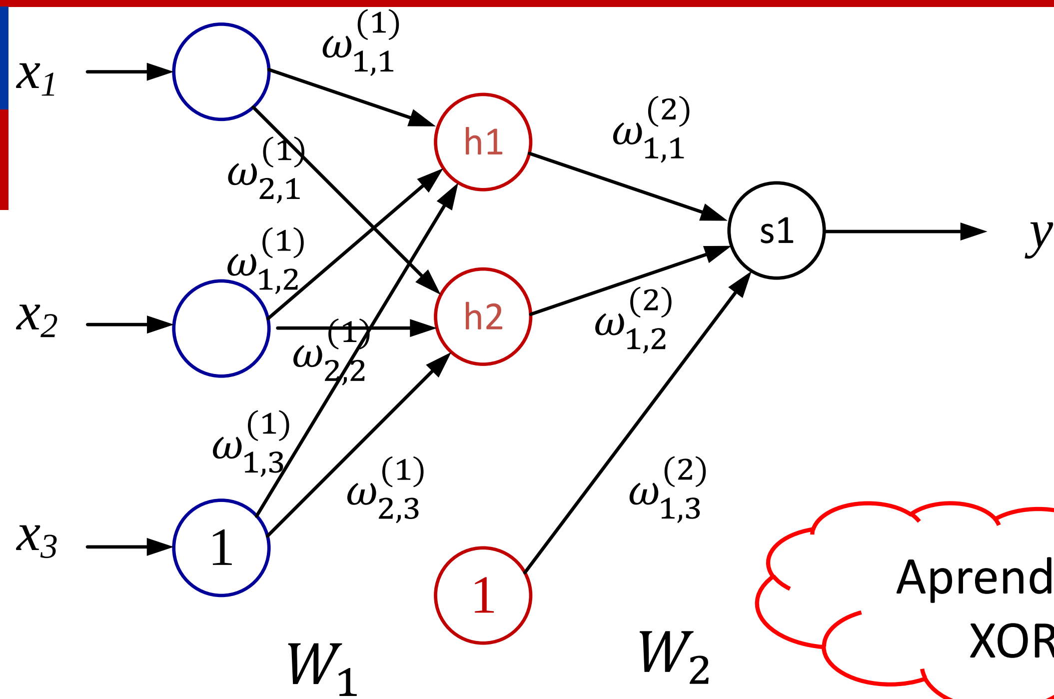
t=2,p=1, etc. etc.

RetroPropagação – BackPropagation

Exemplo: Problema XOR

e.g. Ao fim de 10000 épocas

x_1	x_2	x_3	t
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0



$$W_1 = \begin{bmatrix} -4.9783 & -4.9982 & 7.4323 \\ -6.6192 & -6.7284 & 2.7621 \end{bmatrix}$$

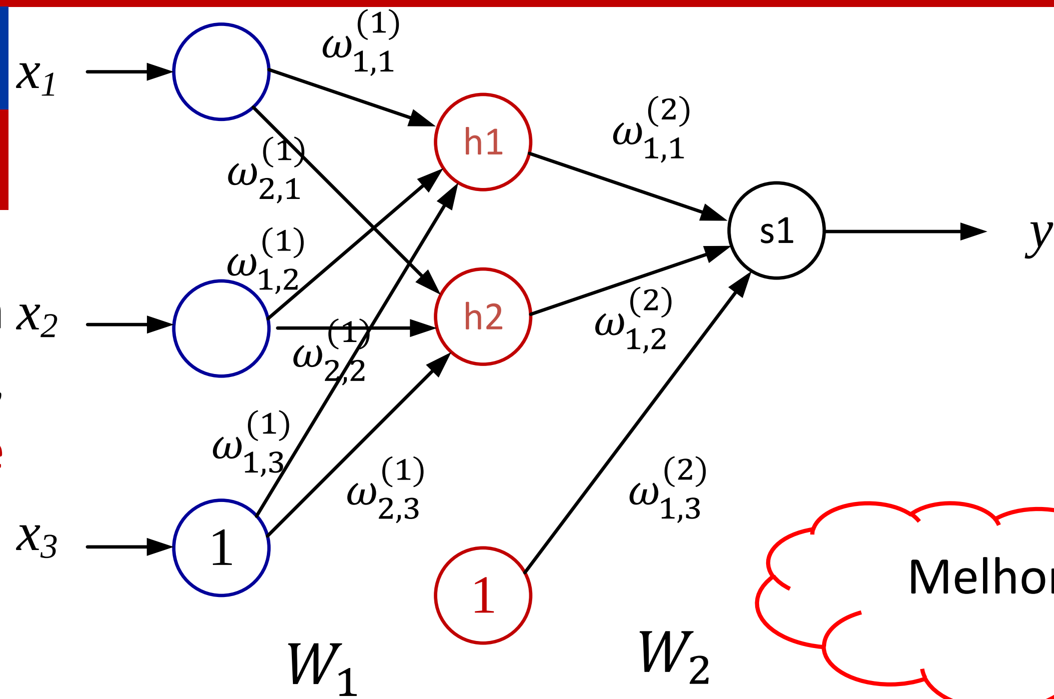
$$W_2 = [10.3739 \quad -10.4966 \quad 4.9413]$$

x_1	x_2	x_3	y
0	0	1	0.0116
0	1	1	0.9879
1	0	1	0.9878
1	1	1	0.0150

RetroPropagação – BackPropagation

Exemplo: Problema XOR

e.g. Ao fim de 10000 épocas

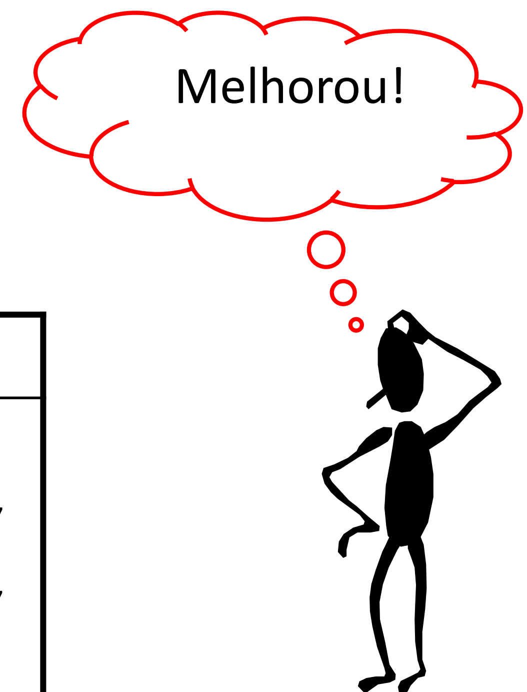


- ✓ Repetindo o processo com os mesmos pesos iniciais, mas usando a função de ativação ReLU.

$$W_1 = \begin{bmatrix} 0.7696 & -0.6634 & -0.0218 \\ -0.7788 & 0.6872 & 0.1301 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1.5907 & 1.4555 & -0.1898 \end{bmatrix}$$

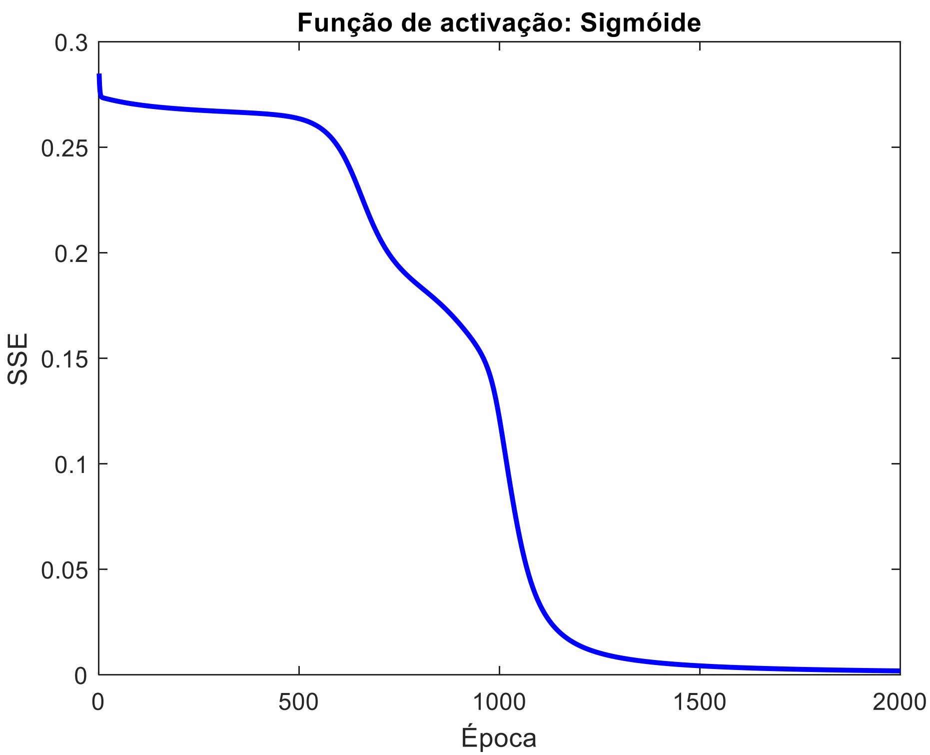
x_1	x_2	x_3	y
0	0	1	0
0	1	1	0.9997
1	0	1	0.9997
1	1	1	0.0005

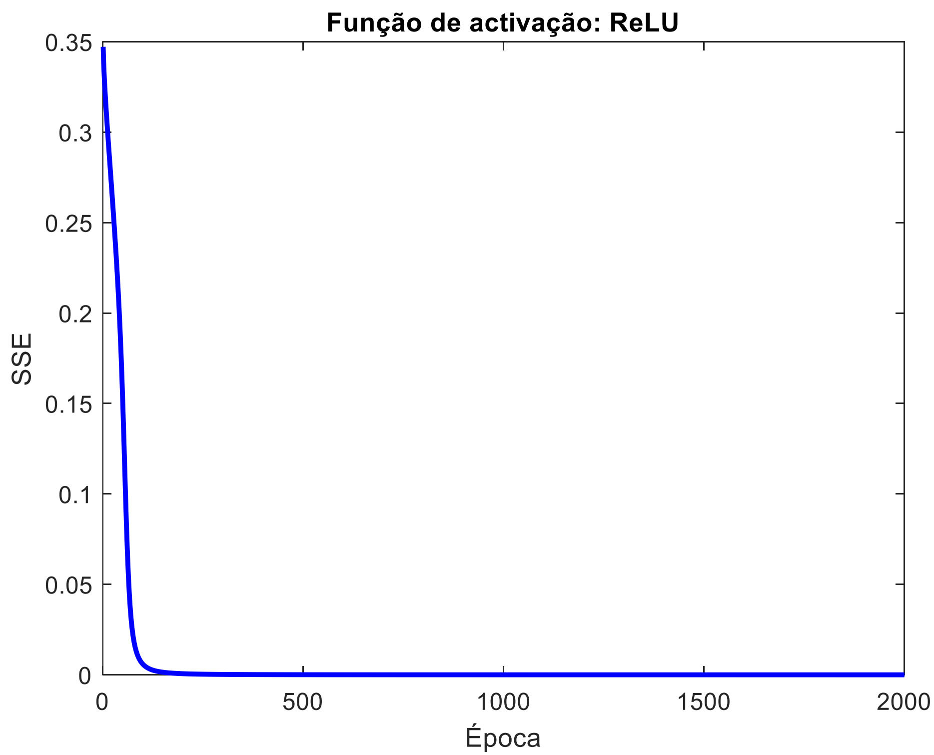


RetroPropagação – BackPropagation

Exemplo: Problema XOR

Comparação entre a evolução do MSE em 2000 épocas:





Momento

- Várias novas regras do Gradiente Descendente têm vindo a ser propostas para efetuar a atualização dos pesos, com algumas vantagens, tais como:
  - Maior estabilidade;
  - Maior eficácia no processo de treino da RN.
- Umas dessas técnicas otimizadas mais antigas com grande popularidade é conhecida por ***Momento (Momentum)*** e representa-se por **m**.
- Relembremos a velha conhecida **regra delta:**

$$\boxed{\omega_{novo} = \omega_{velho} + \alpha\, \delta x}$$

$$\boxed{\omega_{novo} = \omega_{velho} + \Delta\omega}$$

Regra delta com *momento*

$$\boxed{m_{novo} = \Delta\omega + \beta m_{velho}}$$

$$\boxed{\omega_{novo} = \omega_{velho} + m_{novo}}$$

$$0 < \beta < 1$$

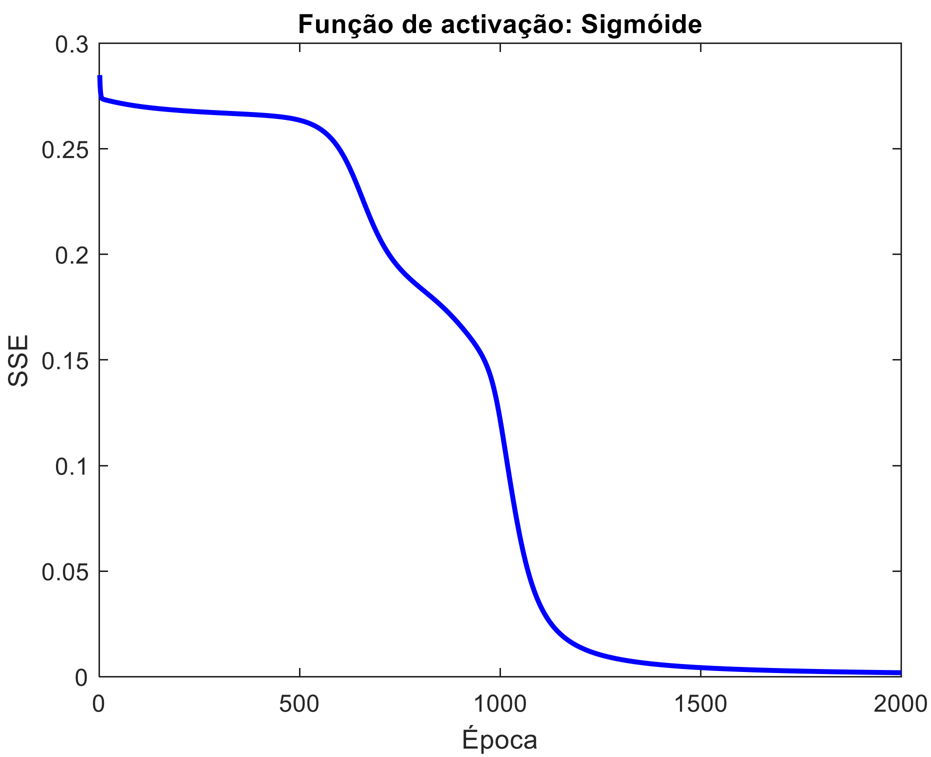
Momento

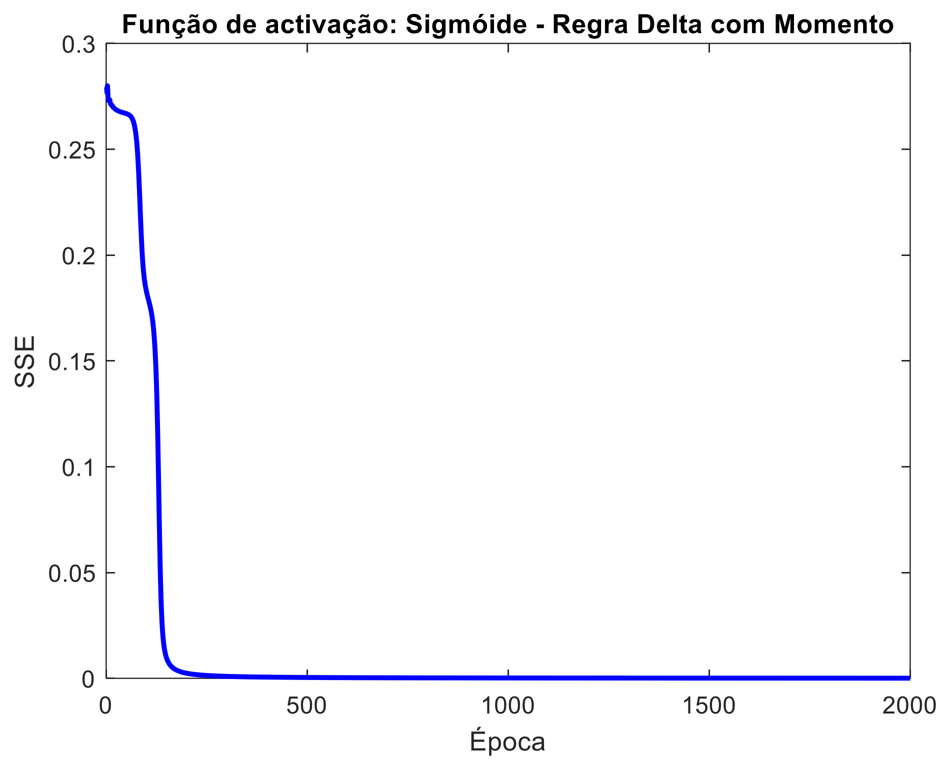
- O momento, por analogia com o momento da Física, tem como efeito fazer com que **atualização dos pesos na época t**, leve em consideração o **valor dos pesos em época anteriores**.
- Como o valor de $\beta<1$, a influência das atualizações mais antigas vai diminuindo no tempo.

Momento

Considere-se o problema XOR

Comparação entre a evolução do MSE em 2000 épocas, usando a função de ativação sigmoidal. Regra delta sem e com momento



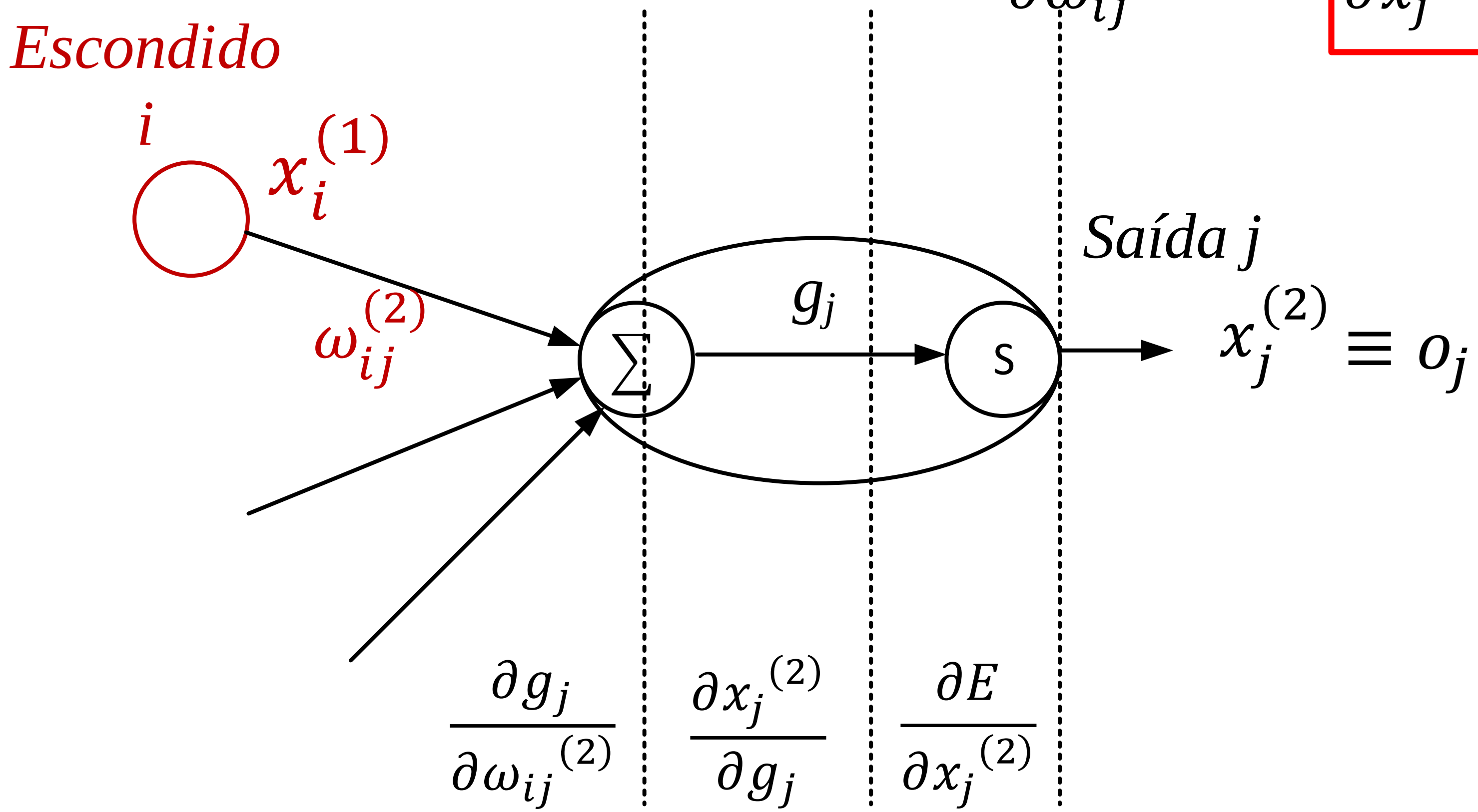


Algoritmo da RetroPropagação:

2. RetroPropagação do Erro para a camada de saída :

✓ Famosa regra em cadeia:

$$-\frac{\partial E}{\partial \omega_{ij}^{(2)}} = -\boxed{\frac{\partial E}{\partial x_j^{(2)}}}\boxed{\frac{\partial x_j^{(2)}}{\partial g_j}}\boxed{\frac{\partial g_j}{\partial \omega_{ij}^{(2)}}}$$



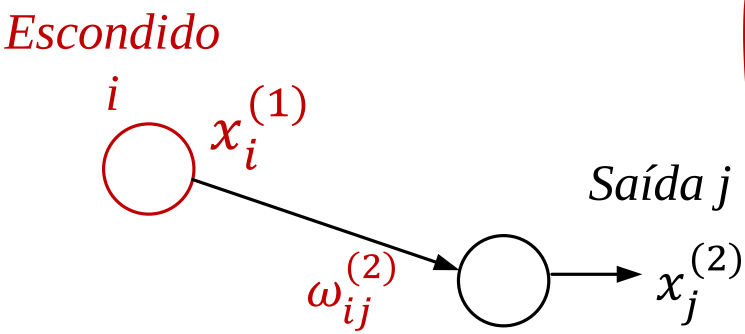
Anexo 1

Algoritmo da RetroPropagação:

De ***:(ver no 59) $\frac{\partial}{\partial x} s(x) = s(x)(1 - s(x))$

$$\frac{\partial}{\partial g_j} s(g_j) = \frac{\partial x_j^{(2)}}{\partial g_j} = x_j^{(2)} \left(1 - x_j^{(2)}\right)$$

2. RetroPropagação do Erro para a camada de saída :



✓ Famosa regra em cadeia:

$$-\frac{\partial E}{\partial \omega_{ij}^{(2)}} = \boxed{-\frac{\partial E}{\partial x_j^{(2)}}} \boxed{\frac{\partial x_j^{(2)}}{\partial g_j}} \boxed{\frac{\partial g_j}{\partial \omega_{ij}^{(2)}}}$$

$$-\frac{\partial E}{\partial x_j^{(2)}} = \left(t_j - x_j^{(2)}\right)$$

$$\frac{\partial x_j^{(2)}}{\partial g_j} = x_j^{(2)} \left(1 - x_j^{(2)}\right)$$

$$\frac{\partial g_j}{\partial \omega_{ij}^{(2)}} = \frac{\partial \left(x_1^{(1)} \omega_{ij}^{(2)}\right)}{\partial \omega_{ij}^{(2)}} = x_i^{(1)}$$

$$-\frac{\partial E}{\partial \omega_{ij}^{(2)}} = x_j^{(2)} \left(1 - x_j^{(2)}\right) \left(t_j - x_j^{(2)}\right) x_i^{(1)}$$

$$\frac{\partial E}{\partial \omega_{ij}^{(2)}} = \delta_j^{(2)} \, x_i^{(1)}$$