

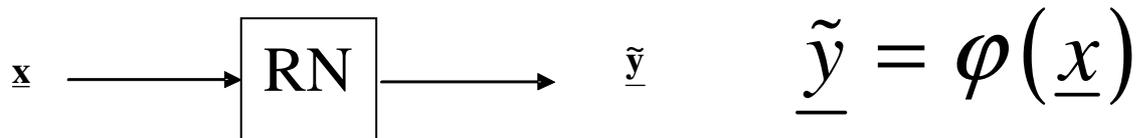
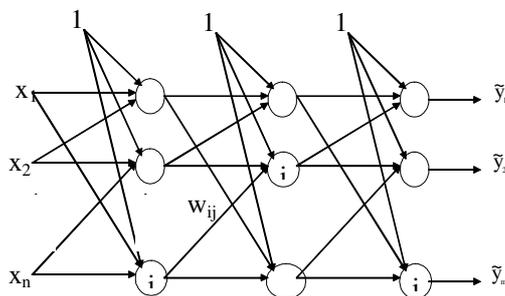
# Redes Neurais Feedforward

## Aprendizado (Treinamento) Backpropagation

### Aproximador Universal

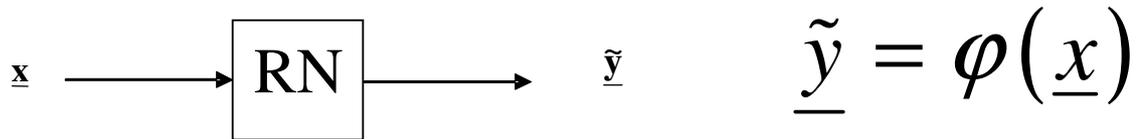
### Redes Neurais *FeedForward*

#### Redes “*Backpropagation*”



**Mapeador não linear**

**Aproximador Universal !**



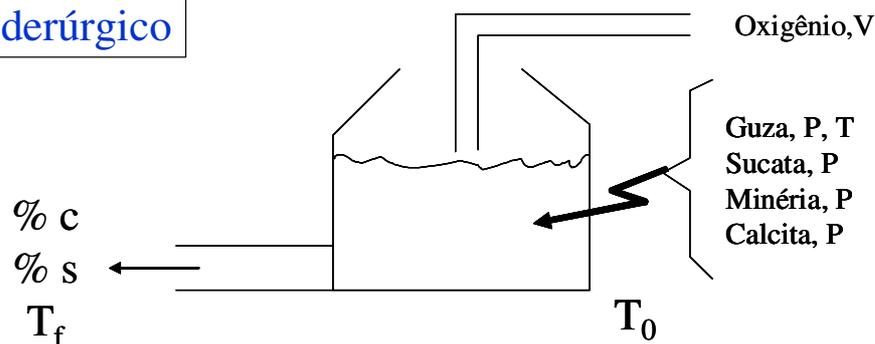
Para que serve ?

Aplicações:

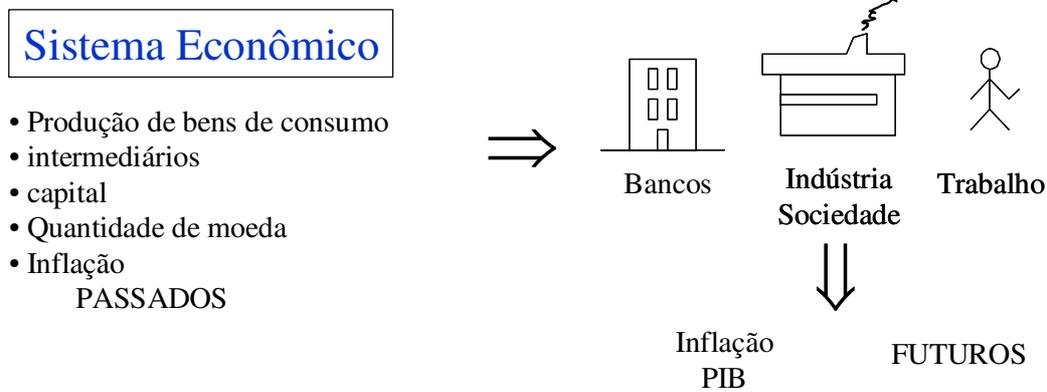
Simuladores;  
Controladores;  
Classificadores;  
Reconhecimento de padrões;  
Filtragem não linear;  
etc...

Simulação de Sistemas

Conversor Siderúrgico



# Simulação de Sistemas



## Simulação de Sistemas

### Sistema Real, Planta

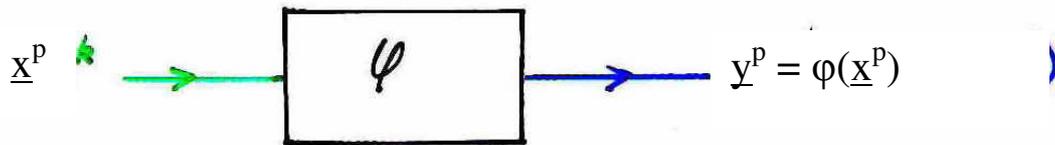
$$(\mathbf{x}^p, \mathbf{y}^p) \quad p = 1, 2, \dots, P$$



### Simulador



## Sistema à emular



### Conhecimento do Sistema

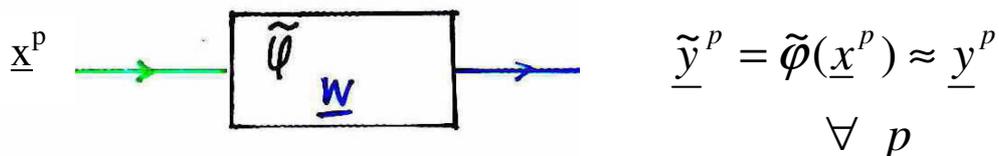
**Fenomenológico – equações – modelo caixa branca - NÃO**

**Funcional – relação entrada/saída – modelo caixa preta**

### Pares entrada-saída

$$(\underline{x}^p, \underline{y}^p) \quad p = 1, \dots, P$$

## Modelo Matemático Fenomenológico / Numérico - Simulador



**Modelo Fenomenológico – Caixa branca**

**Modelo Numérico Geral (e.g. rede neural) – Caixa Preta**

**Ajuste dos parâmetros w ?**

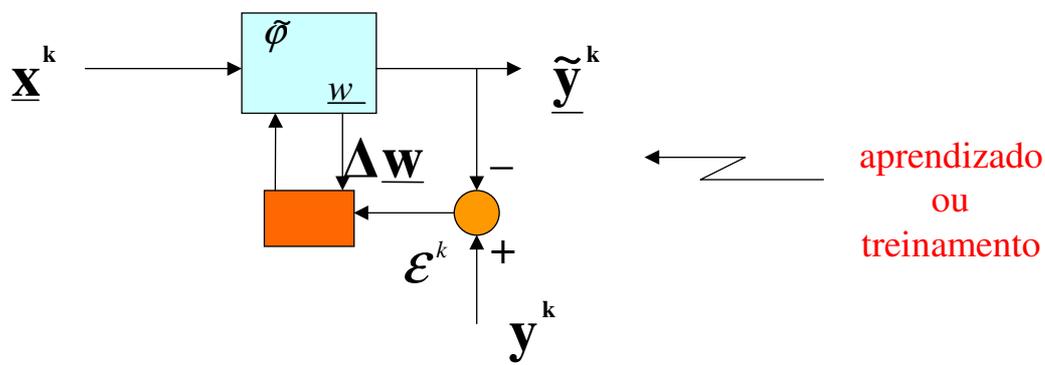
# Simulação de Sistemas

## Sistema Real, Planta

$$(\underline{\mathbf{x}}^k, \underline{\mathbf{y}}^k) \quad k = 1, 2, \dots, P$$



## Simulador



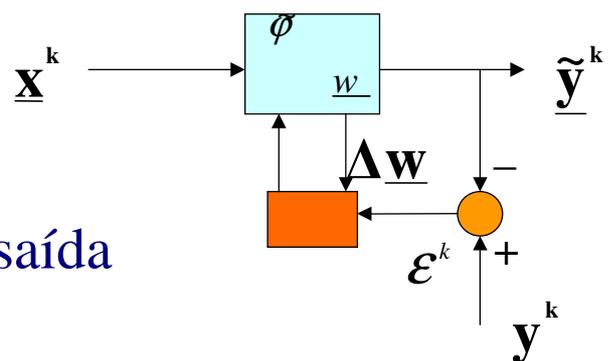
## Aprendizado

(ou treinamento, ou ajuste de parâmetros)

**como um processo de otimização**

(visto em detalhes em CPE 723 - apostila disponível no site)

Aprendizado =  
minimização do erro na saída



Erro a minimizar:

$$\mathcal{E}^{2^k} = \|\underline{\mathbf{y}}^k - \tilde{\underline{\mathbf{y}}}^k\|^2 = \sum_{l=1}^m (\underline{\mathbf{y}}_l^k - \tilde{\underline{\mathbf{y}}}_l^k)^2$$

Erro médio quadrático:

$$F_0 = E(\mathcal{E}^{2^k}) = \frac{1}{P} \sum_{k=1}^P \mathcal{E}^{2^k}$$

$$F_0 = E(\mathcal{E}^{2^k}) = F_0(\underline{\mathbf{w}}) \geq 0$$

**Treinamento, Aprendizado: Minimizar o erro na saída**

Função objetivo à minimizar:

$$F_0 = E(\mathcal{E}^{2^k}) = F_0(\underline{\mathbf{w}}) \geq 0$$

**Problema:**

Como encontrar o vetor  $\underline{\mathbf{w}}_0$  que minimize  $F_0(\underline{\mathbf{w}})$ ?

$$F_0(\underline{\mathbf{w}}_0) \leq F_0(\underline{\mathbf{w}}) \quad \forall \underline{\mathbf{w}} \neq \underline{\mathbf{w}}_0$$

## Gradiente:

$$F_0(w_1, w_2, \dots) = F(\underline{\mathbf{w}})$$

$$\underline{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \end{bmatrix} \quad \nabla_{\underline{\mathbf{w}}} F_0 = \underline{\nabla} = \begin{bmatrix} \frac{\partial F_0}{\partial w_1} \\ \frac{\partial F_0}{\partial w_2} \\ \vdots \end{bmatrix}$$

## Propriedades do Gradiente:

$$1. \underline{\mathbf{w}} \rightarrow \underline{\mathbf{w}} + \Delta \underline{\mathbf{w}} \Rightarrow F_0 \rightarrow F_0 + \Delta F_0$$

$$\text{se } \Delta \underline{\mathbf{w}} = \alpha \underline{\nabla} \Rightarrow \Delta F_0 \text{ é máximo}$$

$$\text{então se } \Delta \underline{\mathbf{w}} = -\alpha \underline{\nabla} \Rightarrow \Delta F_0 \text{ é mínimo}$$

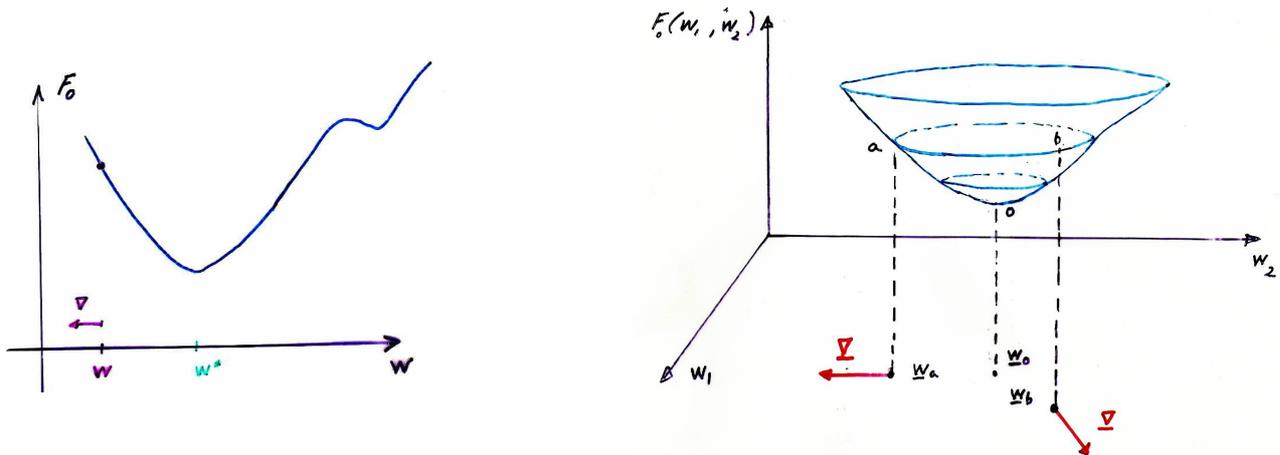
$$\text{i.e. } \Delta F_0 < 0 \text{ e } |\Delta F_0| \text{ é máximo}$$

$$2. \underline{\nabla}|_{\underline{\mathbf{w}}_0} = \underline{\mathbf{0}}$$

# Gradiente descendente

## ou descida por gradiente

$$\underline{w} \longrightarrow \Delta \underline{w} = -\alpha \nabla(\underline{w}) \longrightarrow \underline{w} = \underline{w} + \Delta \underline{w}$$



Complexidade de cálculo:

$$\Delta w_{ij} = -\alpha \frac{\partial F}{\partial w_{ij}}$$

Fim do processo:

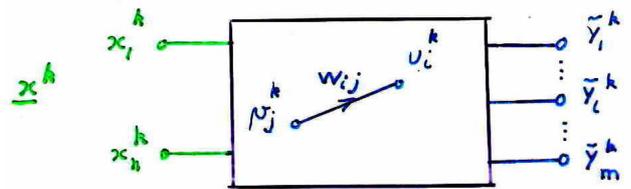
$$\nabla \underline{w}_{otimo} = \underline{0} \quad \longrightarrow \quad \Delta \underline{w}_{otimo} = \underline{0}$$

**Acréscimo a aplicar em cada sinapse:**

$$F_0 = E(\epsilon^{2p}) = F_0(\underline{\mathbf{w}}) \qquad \Delta w_{ij} = -\alpha \frac{\partial F_0(w_{ij})}{\partial w_{ij}}$$

**Como calcular**

$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} \quad ????$$



$$1 - F_0(w_{ij}) = E_k [(\epsilon^k)^2] = \frac{1}{K} \sum_{k=1}^K (\epsilon^k)^2$$

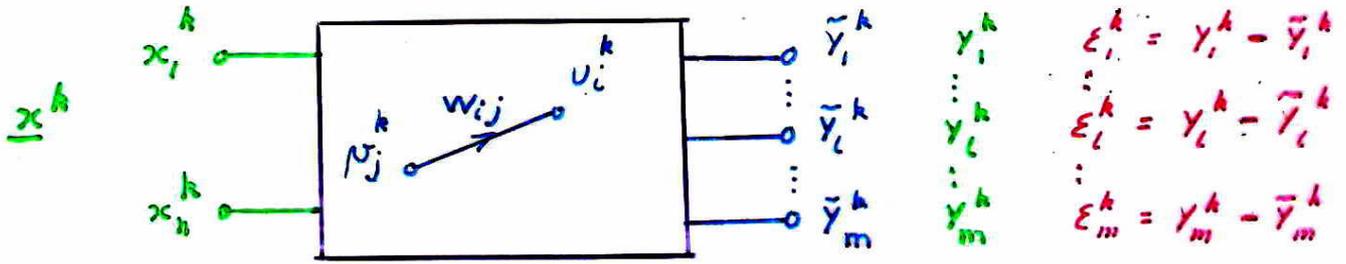
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{\partial E_k [(\epsilon^k)^2]}{\partial w_{ij}} = E_k \left[ \frac{\partial (\epsilon^k)^2}{\partial w_{ij}} \right]$$

**Propriedade importante:**

o gradiente do valor esperado do erro quadrático é igual ao valor esperado do gradiente do erro quadrático

**cada par entrada-saída é tratado isoladamente.**

## 2 – Cálculo de $\frac{\partial(\epsilon^k)^2}{\partial w_{ij}}$ para cada par entrada – saída k



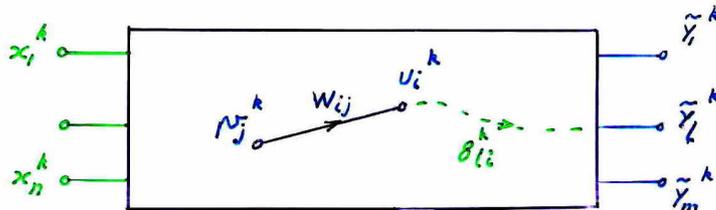
$$(\epsilon)^2 = \sum_{l=1}^m (\epsilon_l)^2 = \sum_{l=1}^m (y_l - \tilde{y}_l)^2$$

$$\frac{\partial(\epsilon)^2}{\partial w_{ij}} \quad ????$$

$$\frac{\partial(\epsilon)^2}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{l=1}^m \epsilon_l^2 = \sum_{l=1}^m \frac{\partial \epsilon_l^2}{\partial w_{ij}} = 2 \sum_{l=1}^m \epsilon_l \frac{\partial \epsilon_l}{\partial w_{ij}} =$$

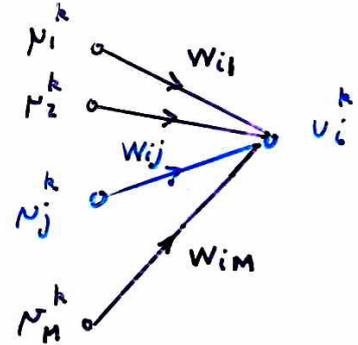
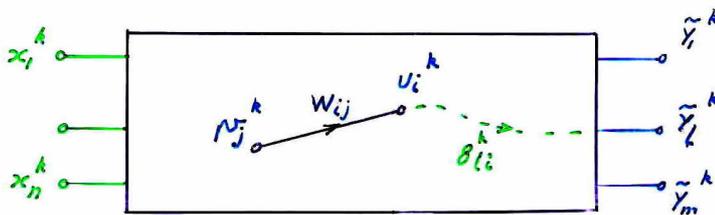
$$= 2 \sum_{l=1}^m \epsilon_l \frac{\partial (y_l - \tilde{y}_l)}{\partial w_{ij}} = -2 \sum_{l=1}^m \epsilon_l \frac{\partial \tilde{y}_l}{\partial w_{ij}} = -2 \sum_{l=1}^m \epsilon_l \frac{\partial \tilde{y}_l}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}}$$

$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li}$$



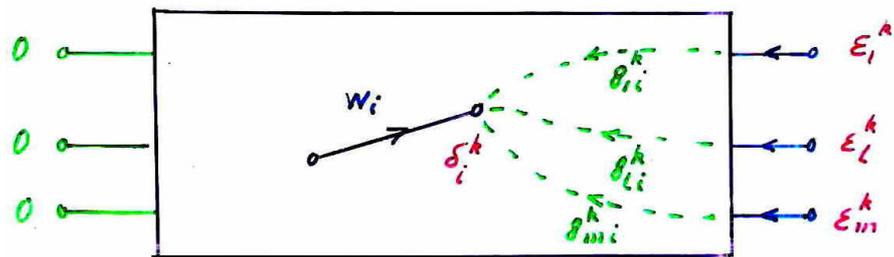
$$\frac{\partial(\epsilon)^2}{\partial w_{ij}} = -2 \sum_{l=1}^m \epsilon_l \frac{\partial \tilde{y}_l}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}} = -2 \sum_{l=1}^m \epsilon_l g_{li} v_j$$

$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li} \quad u_i = \sum_{m=1}^M w_{mj} v_j \quad \frac{\partial u_i}{\partial w_{ij}} = v_j$$



$$\frac{\partial(\epsilon)^2}{\partial w_{ij}} = -2 \sum_{l=1}^m \epsilon_l g_{li} v_j = -2 v_j \sum_{l=1}^m \epsilon_l g_{li} = -2 v_j \delta_i$$

$$\delta_i = \sum_{l=1}^m \epsilon_l \frac{\partial \tilde{y}_l}{\partial u_i} = \sum_{l=1}^m \epsilon_l g_{li}$$



$\delta_i$  é o erro retropropagado da saída até a extremidade da sinapse

### 3 - como:

$$\Delta w_{ij} = -\alpha \frac{\partial F_0(w_{ij})}{\partial w_{ij}}$$

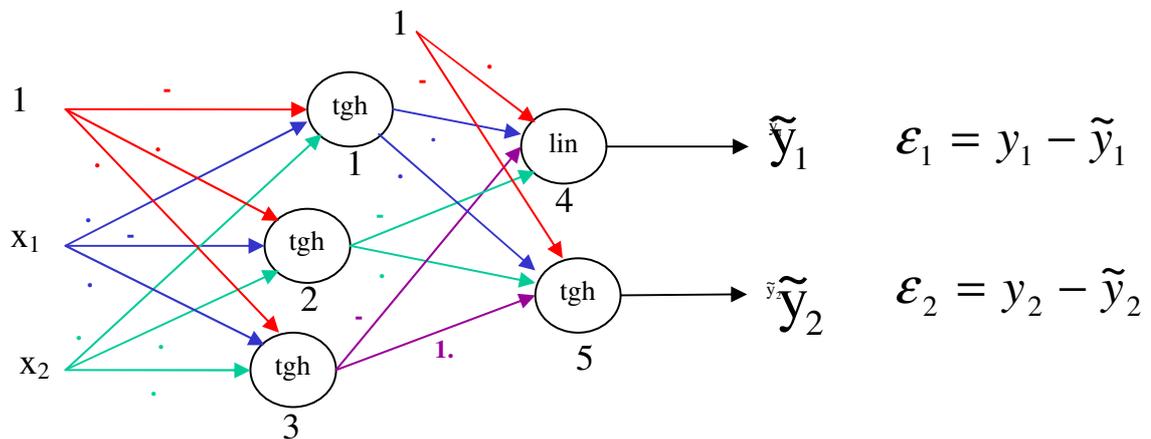
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{1}{P} \sum_{p=1}^P \frac{\partial}{\partial w_{ij}} (\epsilon^{2^p})$$

$$\frac{\partial (\epsilon)^2}{\partial w_{ij}} = -2 v_j \delta_i$$

$$\Delta w_{ij} = 2\alpha \frac{1}{P} \sum_{p=1}^P v_j \delta_i|_p$$

## Error Backpropagation -Princípio do Algoritmo:

### 1 - Rede Original:



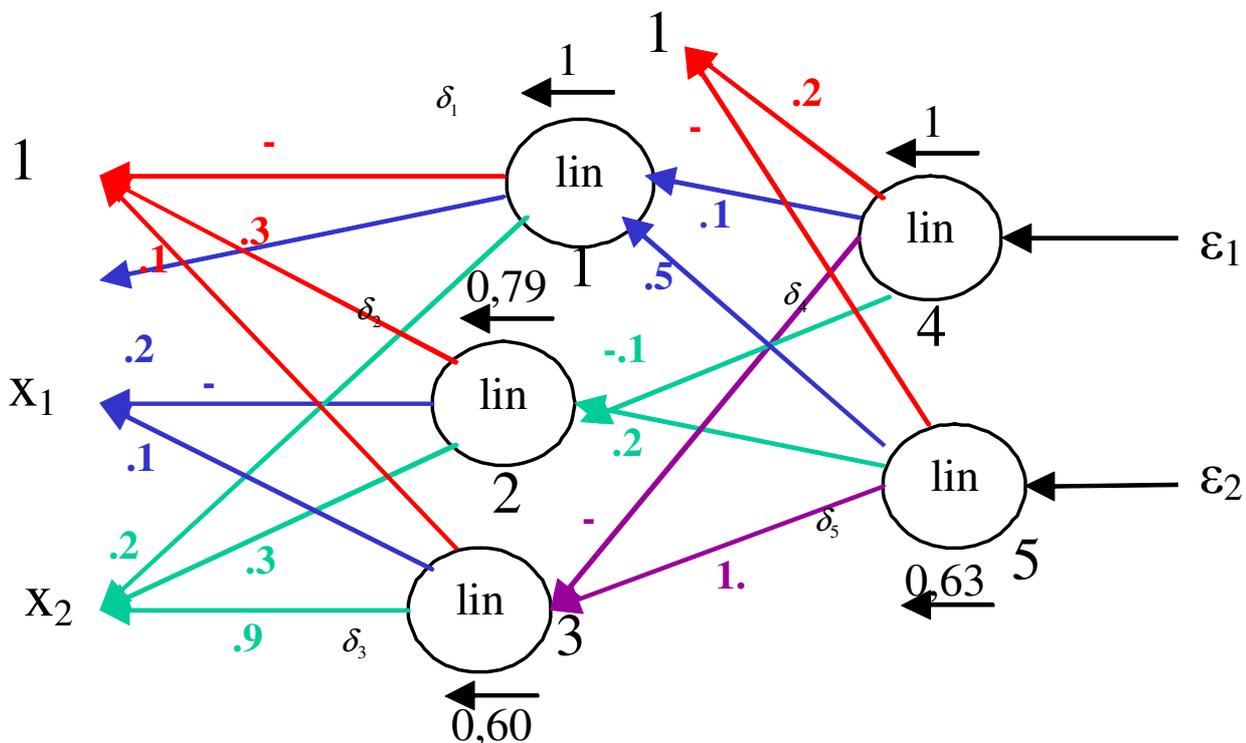
**Propagar o sinal na rede original e conservar o valor do sinal  $v_j$  na entrada de cada sinapse  $w_{ij}$ . Calcular o erro em cada saída.**

## 2 – Criar uma “Rede Associada” a partir da Rede Original mesma arquitetura, mas

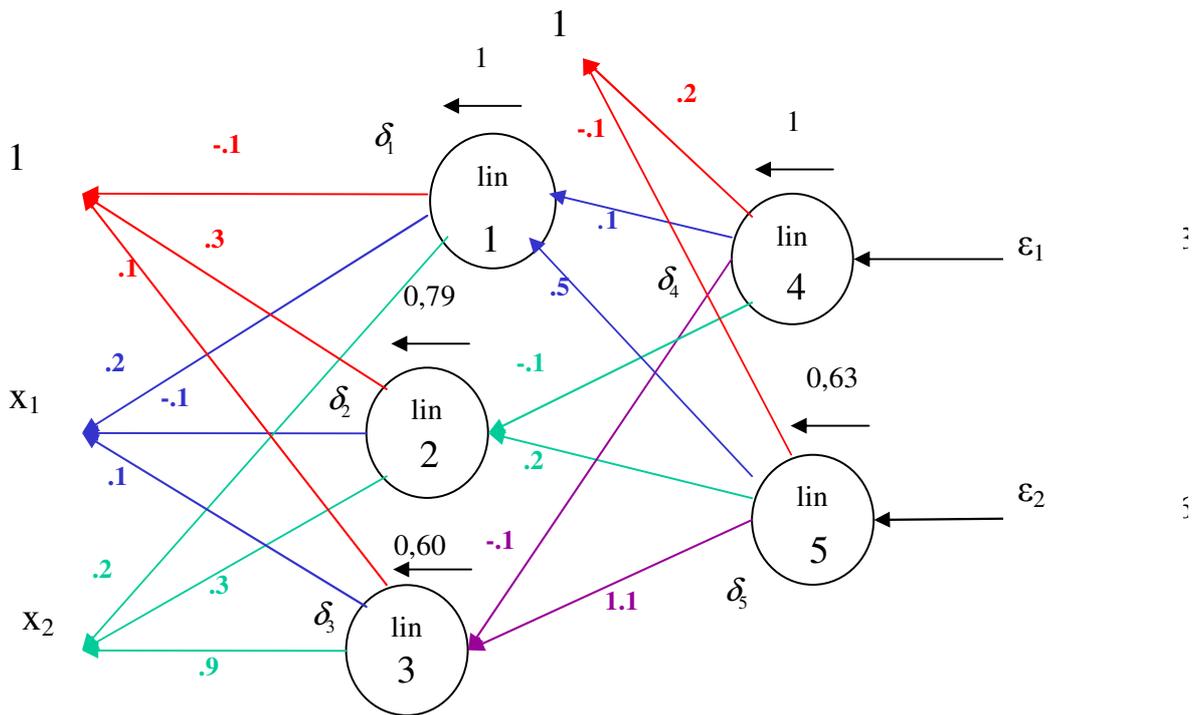
### 2.1 - “Linearizar” (os neurônios) da rede original

Rede original		Rede Associada
Neurônio não linear	>>>	Neurônio linear
$v_0 = \text{tgh}(u_0)$	>>>	$v = (1-v_0^2) u$
Neurônio linear	>>>	Neurônio linear
$v_0 = u_0$	>>>	$v = u$

### 2.2- Inverter todos os sentidos de transmissão (dos neurônios e sinapses)



**3 – (Retro)propagar o erro  $\epsilon_i$  em cada saída através da rede “associada” e conservar o valor do sinal  $\delta_i$  de erro retropropagado que chega na saída (original) de cada sinapse  $w_{ij}$ .**



**4 – O acréscimo para cada sinapse  $w_{ij}$  para o par  $p$  entrada-saída em operação é dado por:**

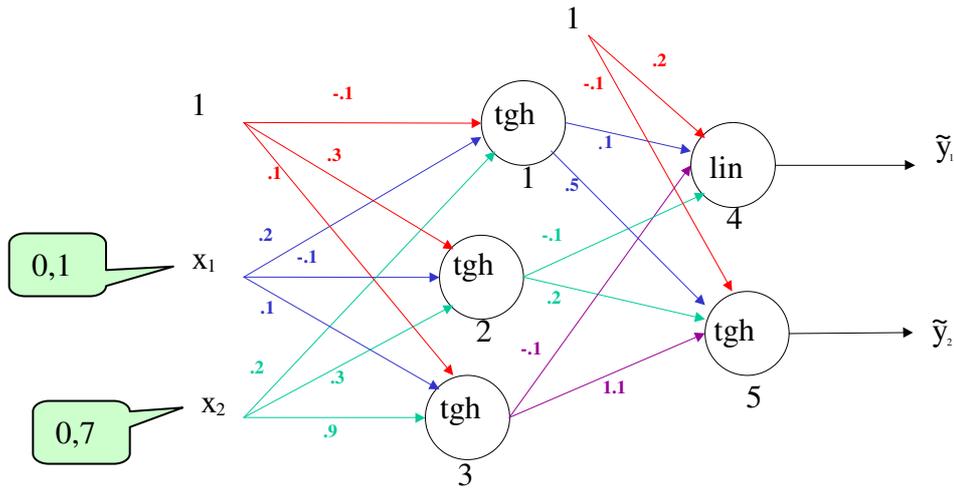
$$\Delta_p w_{ij} = 2 \alpha v_j \delta_i$$

**5 – O acréscimo a ser aplicado na sinapse  $w_{ij}$  é o valor esperado dos acréscimos calculados para todos os pares entrada-saída.**

$$\Delta w_{ij} = E_p(\Delta_p w_{ij}) = 2 \alpha E_p(v_j \delta_i | p)$$

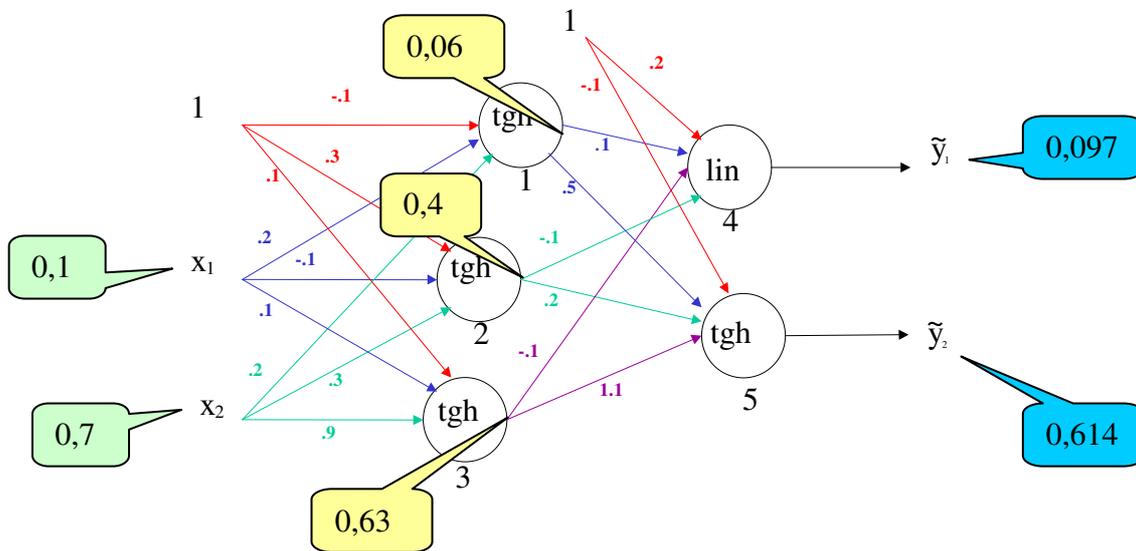
**Processo em Batelada – Batch**

### Exemplo anterior:



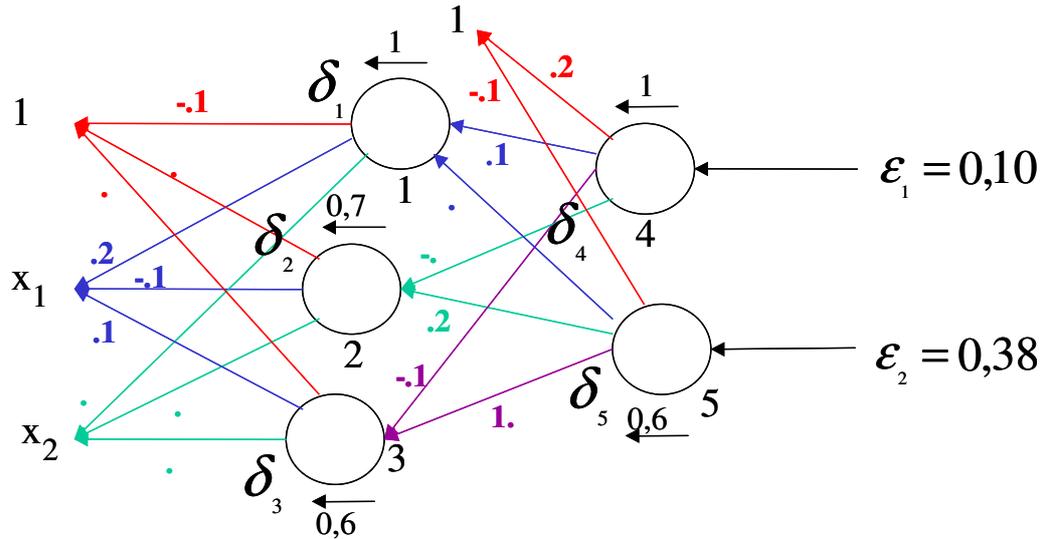
$$\underline{\mathbf{x}} = \begin{bmatrix} 0,1 \\ 0,7 \end{bmatrix} \quad \tilde{\mathbf{y}} = ? \quad \mathbf{y} = \begin{bmatrix} 0,2 \\ 0,1 \end{bmatrix}$$

### Signal Feedforward



$$\underline{\mathbf{x}} = \begin{bmatrix} 0,1 \\ 0,7 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 0,2 \\ 1 \end{bmatrix} \quad \tilde{\mathbf{y}} = \begin{bmatrix} 0,097 \\ 0,614 \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} 0,103 \\ 0,386 \end{bmatrix}$$

### Rede associada e retropropagação do erro:



$$\delta_5 = (0,386)(0,63) = 0,24 \quad \delta_4 = (0,103)(1) = 0,103$$

$$\delta_3 = [(0,103)(-0,1) + (0,24)(1,1)](0,60) = 0,152$$

$$\delta_2 = [(0,103)(-0,1) + (0,24)(0,2)](0,79) = 0,030$$

$$\delta_1 = [(0,103)(0,1) + (0,24)(0,5)](1) = 0,130$$

### Treinamento Regra Delta

Atualização dos valores das sinapses:

$$\Delta w_{ij} = 2 \alpha v_j \delta_i$$

$$\Delta b_4 = (2)(0,1)(1)(0,103) = 0,021$$

$$b_4 \rightarrow 0,2 + 0,021 = 0,221$$

$$\Delta b_1 = (2)(0,1)(1)(0,130) = 0,026$$

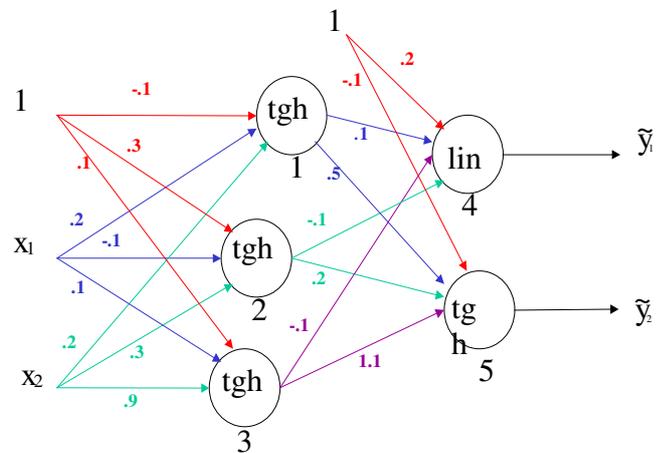
$$b_1 \rightarrow -0,1 + 0,026 = -0,074$$

$$\Delta w_{41} = (2)(0,1)(0,06)(0,103) = 0,001$$

$$w_{41} \rightarrow 0,1 + 0,001 = 0,101$$

$$\Delta w_{3b} = (2)(0,1)(0,7)(0,152) = 0,021$$

$$w_{3b} \rightarrow 0,9 + 0,021 = 0,921$$



## Algoritmos BP para cálculo do acréscimo na sinapse $w_{ij}$ devido ao par $p$ , $\Delta_p w_{ij}$

para dois casos específicos:

### Redes com uma única camada

Definição das variáveis:

N entradas:  $x_1, x_2, \dots, x_N$  e  $x_0$  (bias) = 1

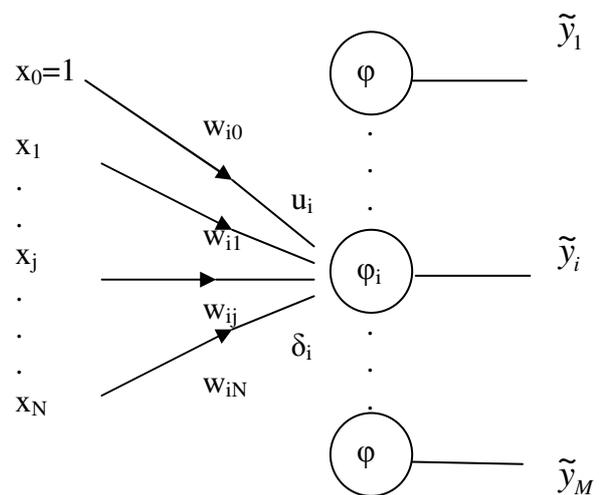
$w_{ij}$  - sinapse conectando a entrada  $j$   
ao neurônio  $i$

M neurônios com função de ativação  
 $\varphi(\cdot)$  linear ou tgh

$u_i$  - excitação interna do neurônio  $i$

$\tilde{y}_i$  - saída do neurônio  $i$

### Rede Original



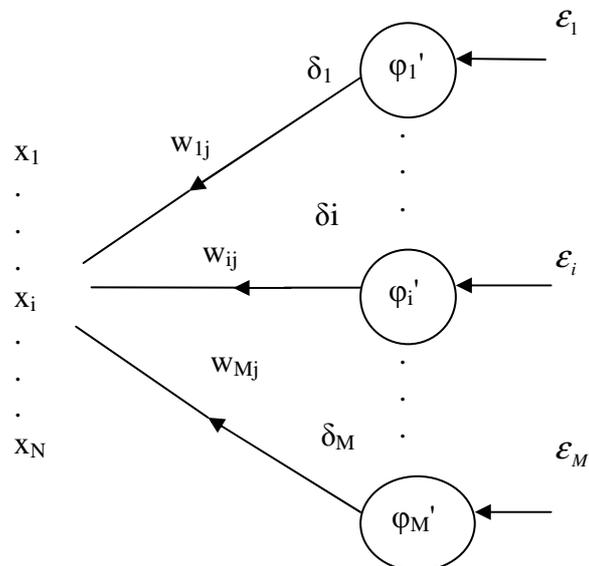
Definição das variáveis:

$\epsilon_i$  - erro na saída do neurônio  $i$

$\delta_i$  - erro retropropagado até  
a entrada do neurônio  $i$

$\varphi_i'$  - valor numérico da derivada de  
 $\varphi_i$  no ponto de operação

### Rede Associada



## Algoritmo BP acréscimo $\Delta w_{ij}^p$ - Rede 1 camada

Para o par entrada-saída p  $(\underline{x}, \underline{y})$

Signal feedforward: Propague o sinal para a frente, calcule a saída e o erro em cada saída:

$$u_i = \sum_{j=0}^N w_{ij} x_j$$

$$\tilde{y}_i = \varphi_i(u_i) = \begin{cases} u_i & \text{neurônio linear} \\ \text{tgh}(u_i) & \text{neurônio tgh} \end{cases}$$

$$\varepsilon_i = y_i - \tilde{y}_i$$

Calcule o erro retropropagado até a entrada de cada neurônio:

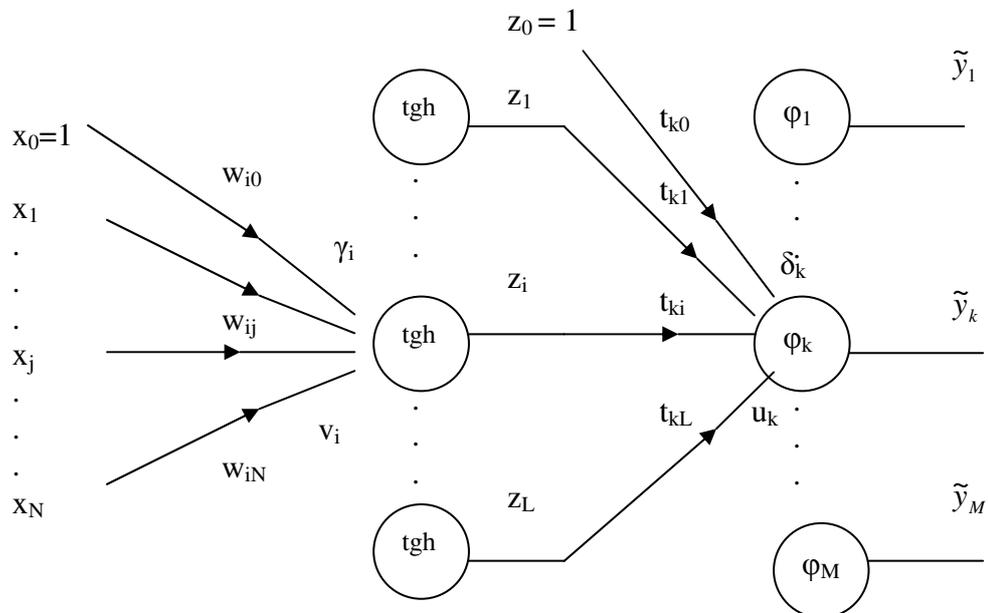
$$\delta_i = \begin{cases} \varepsilon_i & \text{neurônio linear} \\ (1 - \tilde{y}_i^2) \varepsilon_i & \text{neurônio tgh} \end{cases}$$

Calcule o acréscimo em cada sinapse devido ao par p:

$$\Delta w_{ij}^p = 2\alpha x_j \delta_i$$

*fim do algoritmo*

## Redes com duas camadas - Rede Original



Definição das variáveis:

N entradas:  $x_1, x_2, \dots, x_N$  e  $x_0$  (bias) = 1

L neurônios na primeira camada com função de ativação tgh

M neurônios na camada de saída com função de ativação  $\phi(\cdot)$  linear ou tgh

$w_{ij}$  - sinapse conectando a entrada j ao neurônio i da camada intermediária

$t_{ki}$  - sinapse conectando a saída  $z_i$  do neurônio i da camada intermediária com a entrada do neurônio k da camada de saída.

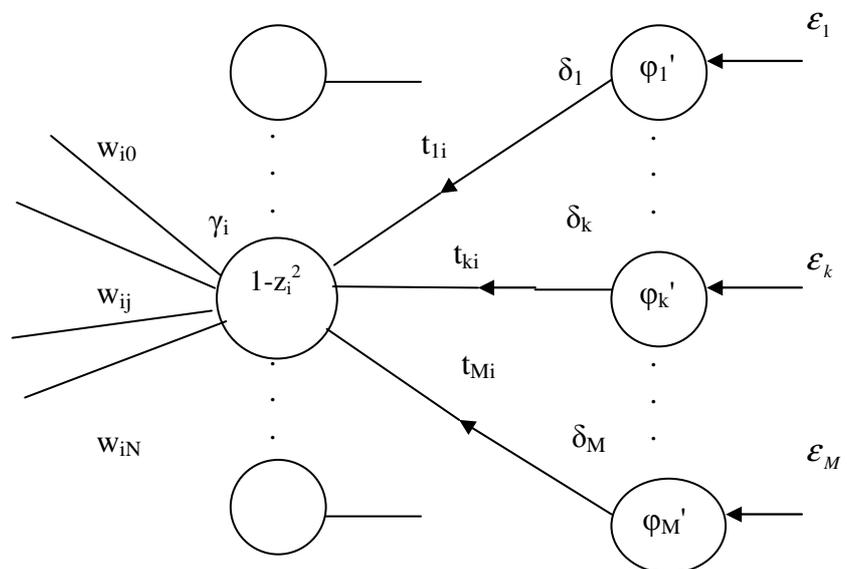
$v_i$  - excitação interna do neurônio i da primeira camada

$z_i$  - saída do neurônio i da camada intermediária.  $z_0$  (bias) = 1

$u_k$  - excitação interna do neurônio k da camada intermediária

$\tilde{y}_k$  - saída do neurônio k da camada de saída

## Rede Associada



Definição das variáveis:

$\epsilon_k$  - erro na saída do neurônio da camada de saída

$\delta_k$  - erro retropropagado até a entrada do neurônio k da camada de saída

$\phi_i'$  - valor numérico da derivada de  $\phi_i$  no ponto de operação

**Algoritmo BP acréscimos  $\Delta w_{ij}^p$  e  $\Delta t_{ki}^p$  - Rede com 2 camadas**

Para o par entrada-saída p ( $\underline{x}$ ,  $y$ )

Signal feedforward: Propague o sinal para a frente, calcule a saída e o erro em cada saída:

Para todos os neurônios da primeira camada  $i = 1, \dots, L$

$$v_i = \sum_{j=0}^N w_{ij} x_j \quad x_0 = 1$$

$$z_i = \text{tgh}(v_i) \quad z_0 = 1$$

Para todos os neurônios da segunda camada  $k = 1, \dots, M$

$$u_k = \sum_{i=0}^L t_{ki} z_i \quad z_0 = 1$$

$$\tilde{y}_k = \varphi_k(u_k) = \begin{cases} u_k & \text{neurônio linear} \\ \text{tgh}(u_k) & \text{neurônio tgh} \end{cases}$$

$$\mathcal{E}_k = y_k - \tilde{y}_k$$

Retropropagação do erro

Para todo neurônio da segunda camada,  $k = 1, \dots, M$ :

$$\delta_k = \begin{cases} \mathcal{E}_k & \text{neurônio linear} \\ (1 - \tilde{y}_k^2) \mathcal{E}_k & \text{neurônio tgh} \end{cases}$$

Para todo neurônio da primeira camada,  $i = 1, \dots, L$

$$\gamma_i = (1 - z_i^2) \sum_{k=1}^M t_{ki} \delta_k$$

Acréscimo nas sinapses da primeira camada devido ao par p:

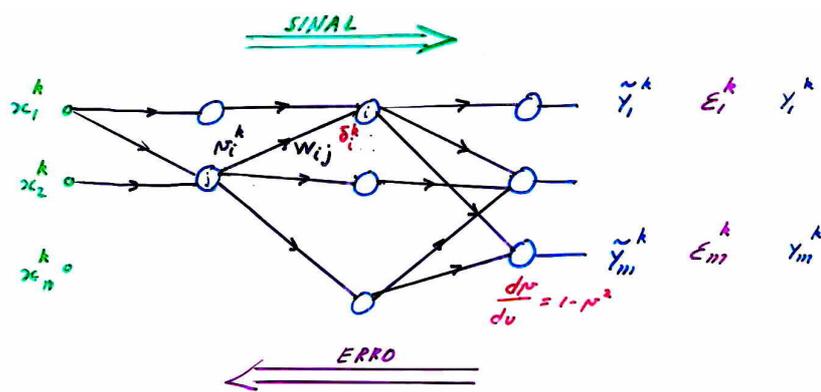
$$\Delta w_{ij}^p = 2 \alpha x_j \gamma_i \quad \forall j = 0, 1, \dots, N \quad e \quad i = 1, 2, \dots, L$$

Acréscimo nas sinapses da segunda camada devido ao par p:

$$\Delta t_{ki}^p = 2 \alpha z_i \delta_k \quad \forall i = 0, 1, \dots, L \quad e \quad k = 1, 2, \dots, M$$

*fim do algoritmo*

## Error Backpropagation - Resumo:



$$\frac{\partial E_i^k}{\partial w_{ij}} = -2 p_j^k \delta_i^k$$

$$\nabla F_0 = \left[ \frac{\partial F_0}{\partial w_{ij}} \right] = -2 E(p_j \delta_i)$$

$$\Delta w_{ij} = 2 \alpha E(p_j \delta_i)$$