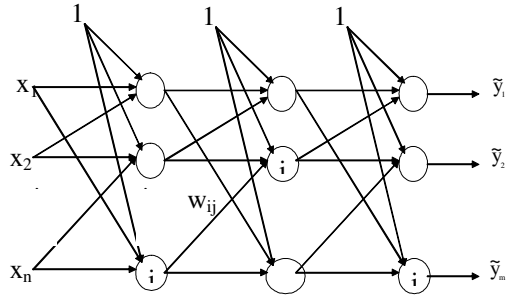
Redes Neurais Feedforward

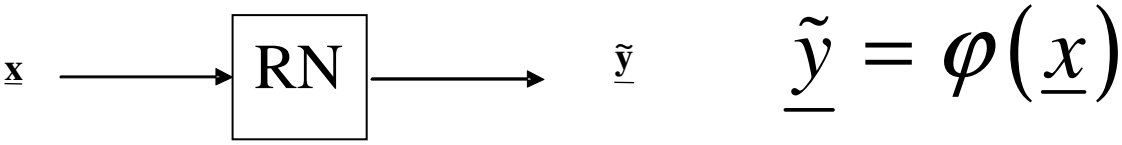
Aprendizado (Treinamento) Backpropagation

Aproximador Universal

Redes Neurais *FeedForward*

Redes "*Backpropagation*"

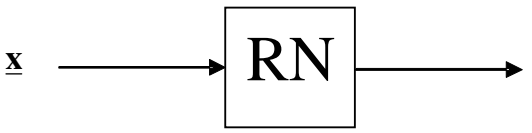




$$\underline{\tilde{y}} = \varphi\left(\underline{x}\right)$$

Mapeador não linear

Aproximador Universal !



$$\tilde{\underline{y}} = \varphi(\underline{x})$$

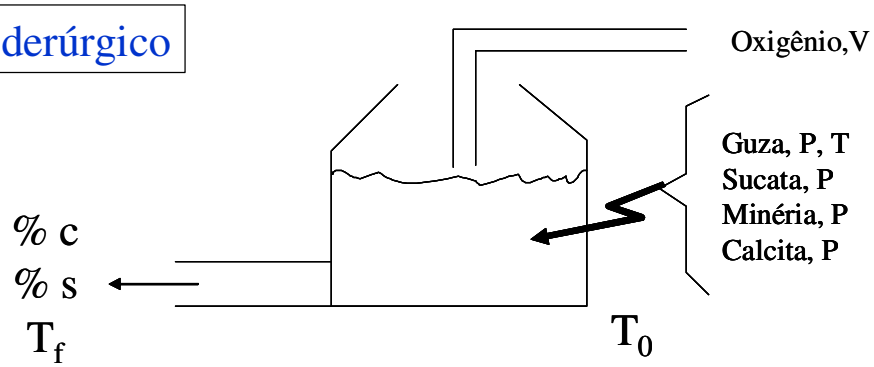
Para que serve ?

Aplicações:

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

Simulação de Sistemas

Conversor Siderúrgico

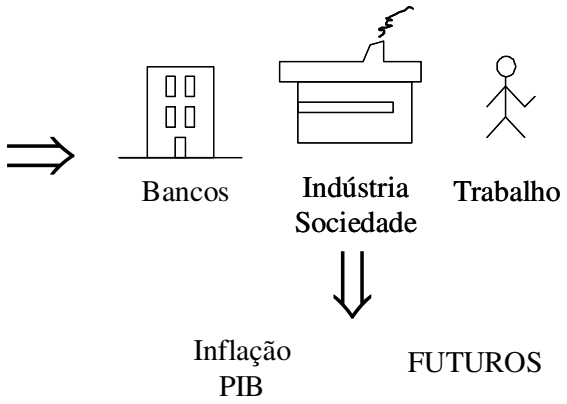


Simulação de Sistemas

Sistema Econômico

- Produção de bens de consumo
- intermediários
- capital
- Quantidade de moeda
- Inflação

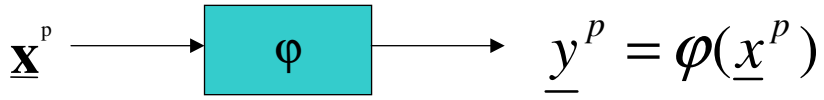
PASSADOS



Simulação de Sistemas

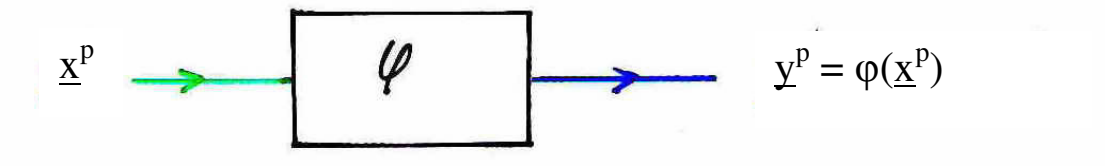
Sistema Real, Planta

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



Simulador

Sistema à emular



Conhecimento do Sistema

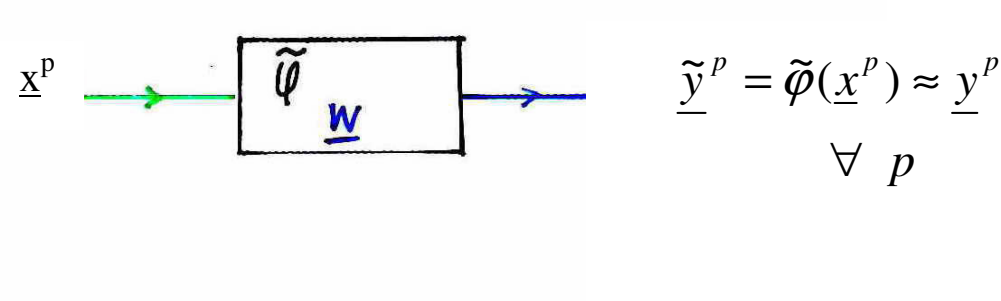
Fenomenológico – equações – modelo caixa branca - NÃO

Funcional – relação entrada/saída – modelo caixa preta

Pares entrada-saída

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

Modelo Matemático Fenomenológico / Numérico - Simulador



Modelo Fenomenológico – Caixa branca

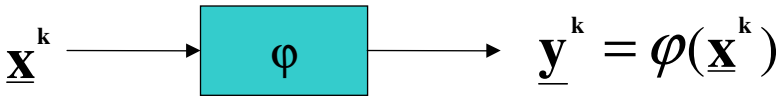
Modelo Numérico Geral (e.g. rede neural) – Caixa Preta

Ajuste dos parâmetros $\underline{w}$?

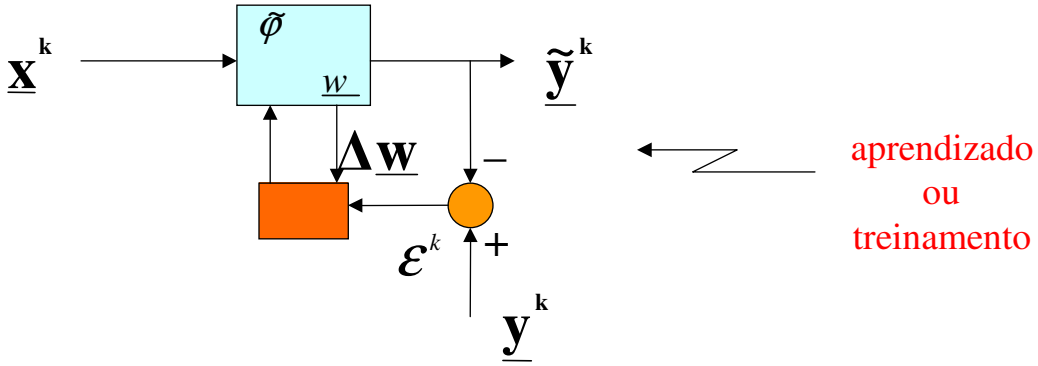
Simulação de Sistemas

Sistema Real, Planta

$$(\mathbf{x}^k, \mathbf{y}^k) \qquad k = 1,2,\ldots,P$$



Simulador

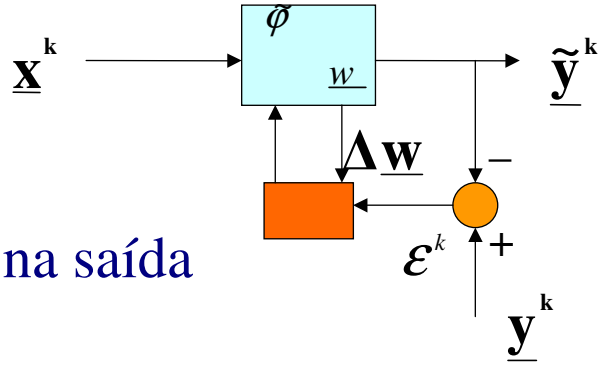


Aprendizado

(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:

$$\varepsilon^{2^k} = \left\| \underline{\mathbf{y}}^{\mathbf{k}} - \underline{\tilde{\mathbf{y}}}^{\mathbf{k}} \right\|^2 = \sum_{l=1}^{m} (\underline{\mathbf{y}}_{\mathbf{l}}^{\mathbf{k}} - \underline{\tilde{\mathbf{y}}}_{\mathbf{l}}^{\mathbf{k}})^2$$

Erro médio quadrático:

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

$$\mathrm{F}_0 = \mathrm{E}(\varepsilon^{2^k}) = \mathrm{F}_0(\underline{\mathbf{w}}) \geq 0$$

Treinamento, Aprendizado: Minimizar o erro na saída

Função objetivo à minimizar:

$$\mathrm{F}_0 = \mathrm{E}(\varepsilon^{2^k}) = \mathrm{F}_0(\underline{\mathbf{w}}) \geq 0$$

Problema:

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

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

Gradiente:

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

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

Propriedades do Gradiente:

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

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

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

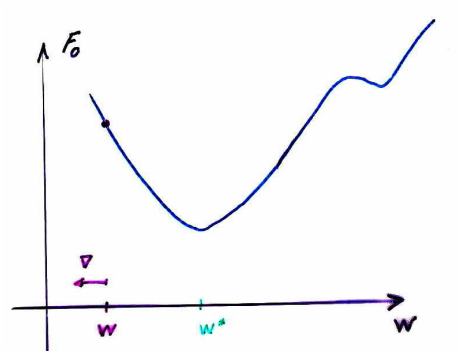
i.e. $\Delta F_0 < 0$ e $|\Delta F_0|$ é máximo

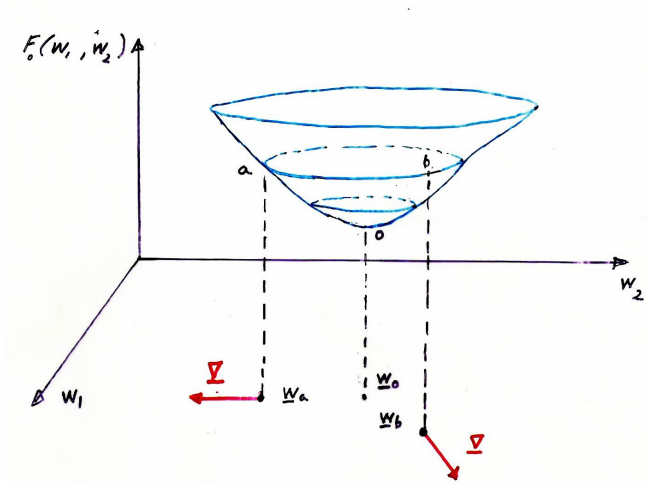
2. $\underline{\nabla}\big|_{\underline{\mathbf{w}_0}} = \underline{\mathbf{0}}$

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:

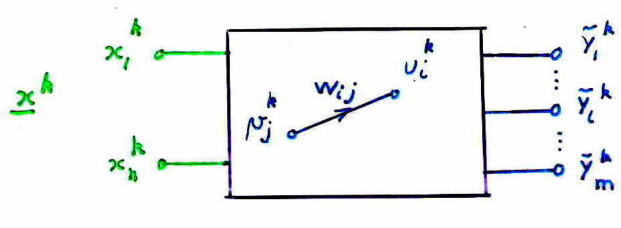
$$\nabla \underline{w}_{otimo} = \underline{0} \quad \Rightarrow \quad \Delta \underline{w}_{otimo} = \underline{0}$$

Acréscimo a aplicar em cada sinapse:

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

Como calcular

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



$$1 - F_0(w_{ij}) = \underset{k}{E}\left[(\varepsilon^k)^2\right] = \frac{1}{K}\sum_{k=1}^{K}(\varepsilon^k)^2$$

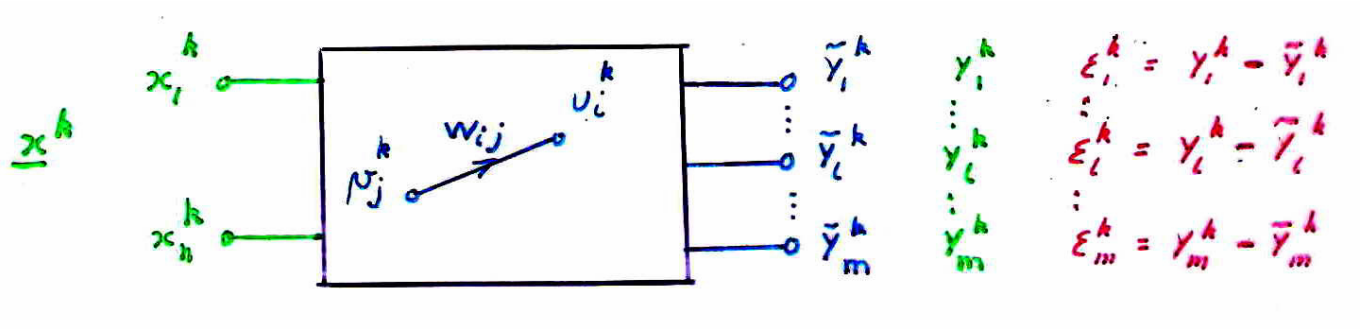
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{\partial \underset{k}{E}\left[(\varepsilon^k)^2\right]}{\partial w_{ij}} = \underset{k}{E}\left[\frac{\partial(\varepsilon^k)^2}{\partial w_{ij}}\right]$$

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 $\frac{\partial(\varepsilon^k)^2}{\partial w_{ij}}$ para cada par entrada – saída k



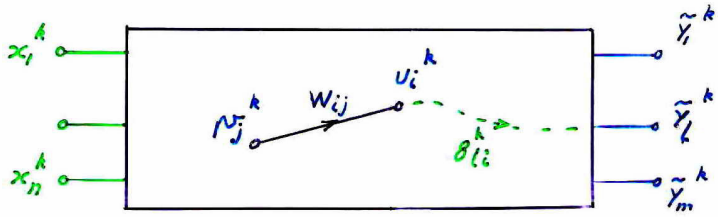
$$(\varepsilon)^2 = \sum_{l=1}^{m} (\varepsilon_l)^2 = \sum_{l=1}^{m} (y_l - \tilde{y}_l)^2$$

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

$$\frac{\partial(\varepsilon)^2}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{l=1}^{m} \varepsilon_l^{\,2} = \sum_{l=1}^{m} \frac{\partial \varepsilon_l^{\,2}}{\partial w_{ij}} = 2\sum_{l=1}^{m} \varepsilon_l \frac{\partial \varepsilon_l}{\partial w_{ij}} =$$

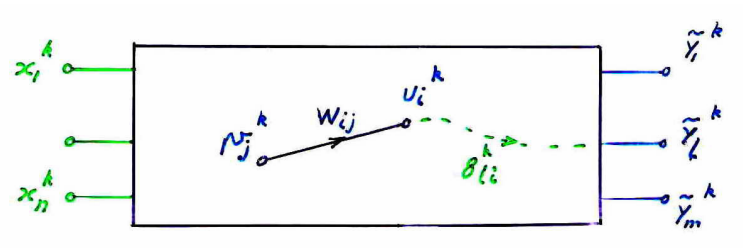
$$= 2\sum_{l=1}^{m} \varepsilon_l \frac{\partial(y_l - \tilde{y}_l)}{\partial w_{ij}} = -2\sum_{l=1}^{m} \varepsilon_l \frac{\partial \tilde{y}_l}{\partial w_{ij}} = -2\sum_{l=1}^{m} \varepsilon_l \frac{\partial \tilde{y}_l}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}}$$

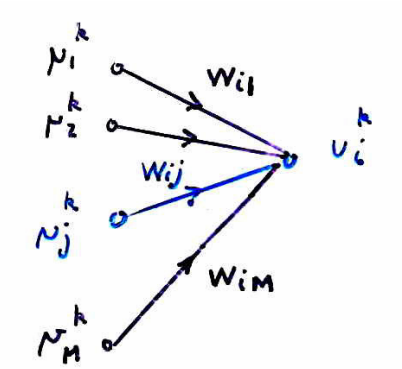
$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li}$$



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

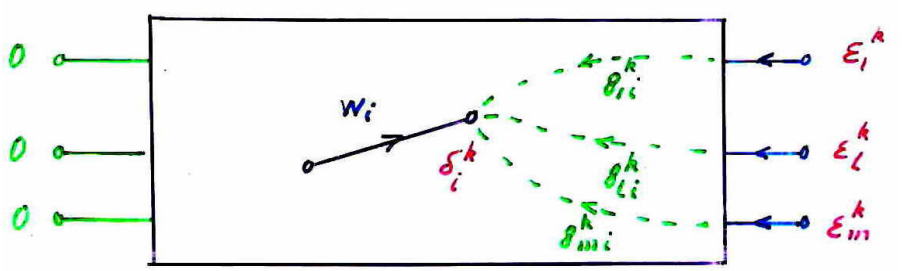
$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li} \qquad u_i = \sum_{m=1}^{M} w_{mj} v_j \qquad \frac{\partial u_i}{\partial w_{ij}} = v_j$$





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

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



δ_i é o erro retropropagado da saída até a extremidade da sinapse

3 - como:

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

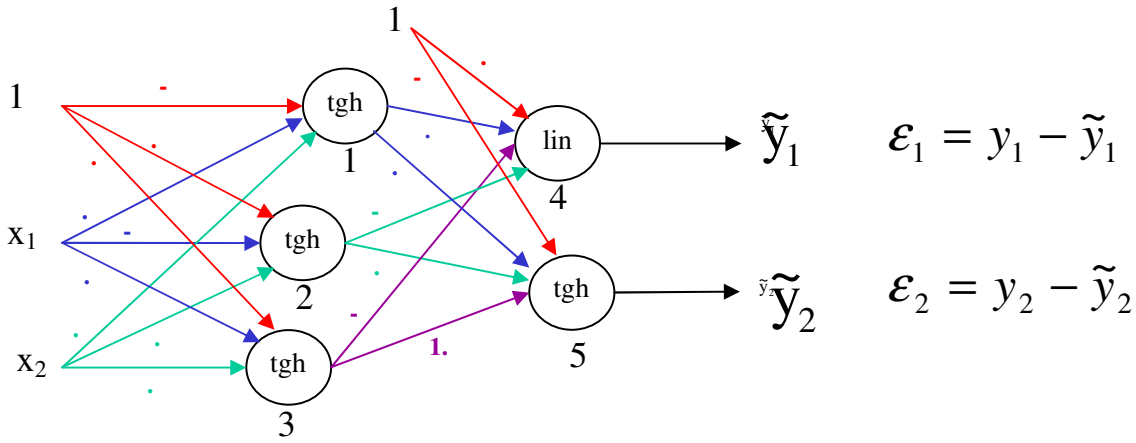
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{1}{P}\sum_{p=1}^{P} \frac{\partial}{\partial w_{ij}}(\varepsilon^{2^p})$$

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

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

Error Backpropagation -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.

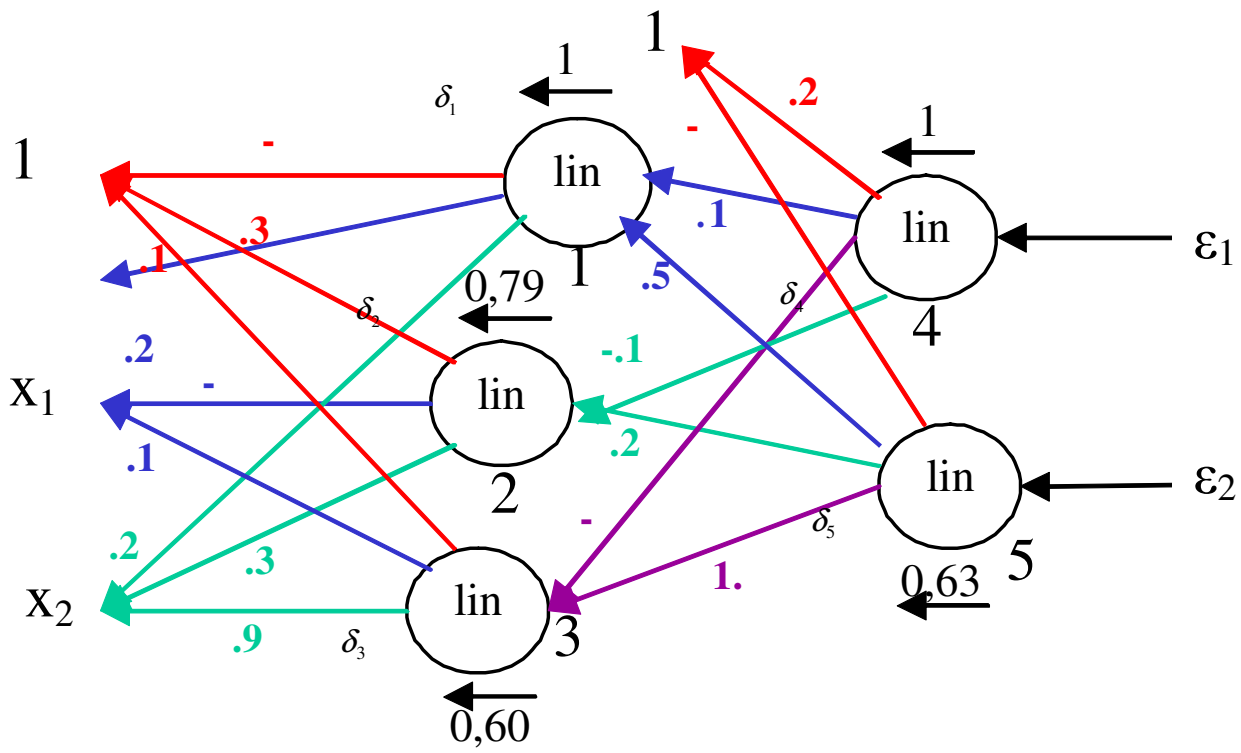
2 – Criar uma "Rede Associada" a partir da Rede Original

mesma arquitetura, mas

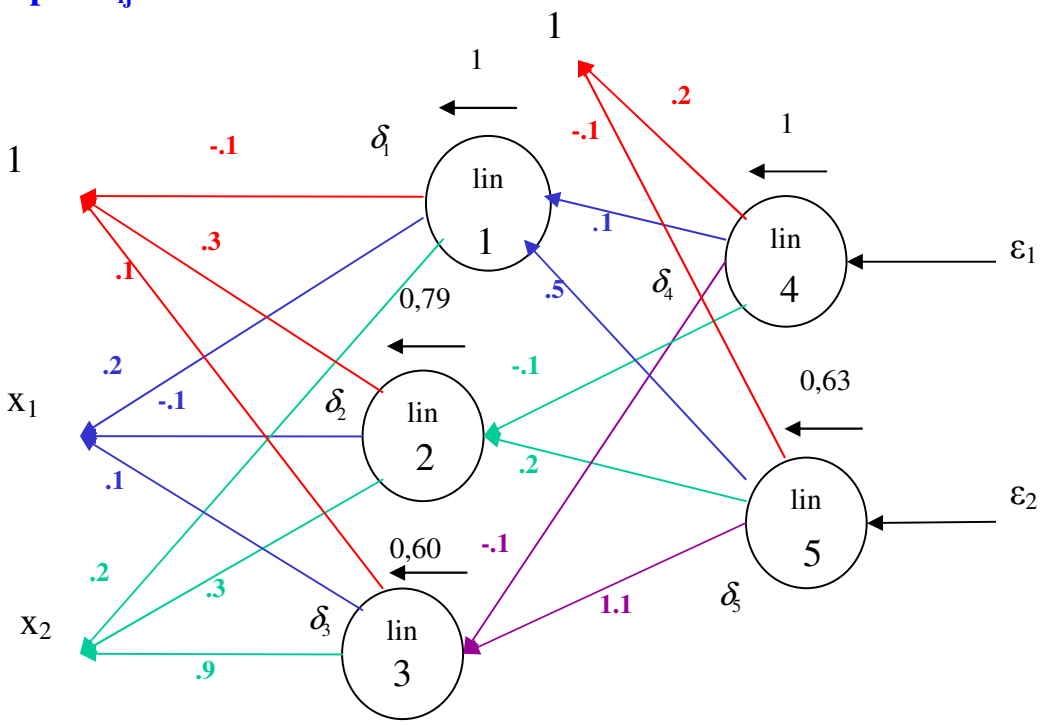
2.1 - "Linearizar" (os neurônios) da rede original

Rede original		Rede Associada
Neurônio não linear	**>>>**	**Neurônio linear**
$v_0 = tgh(u_0)$	>>>	$v = (1-v_0^2)\, u$
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 ε_l em cada saída através da rede "associada" e conservar o valor do sinal δ_i de erro retropropagado que chega na saída (original) de cada sinapse w_{ij}.



4 – O acréscimo para cada sinapse wij para o par p entrada-saída em operação é dado por:

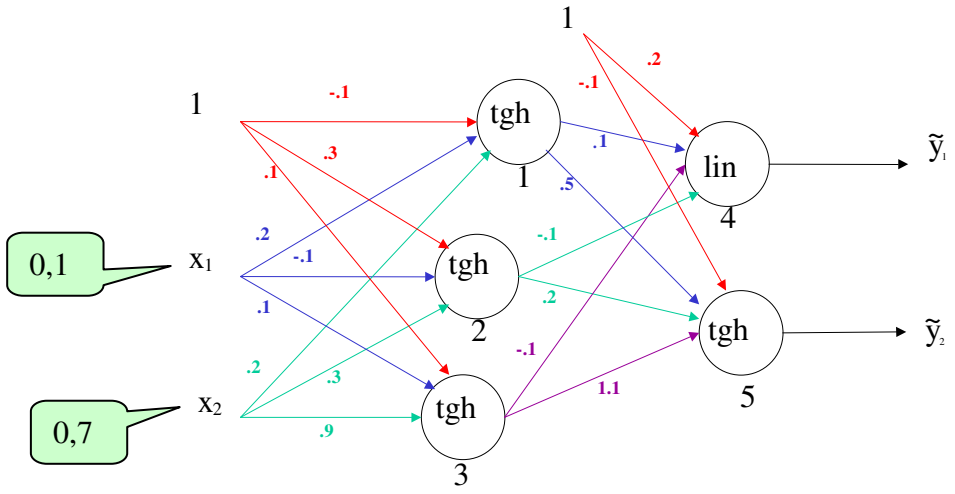
$$\Delta_p w_{ij} = 2\,\alpha\, v_j\, \delta_i$$

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

$$\Delta w_{ij} = \underset{p}{E}\left(\Delta_p w_{ij}\right) = 2\,\alpha\,\underset{p}{E}\left(v_j \delta_i\Big|_p\right)$$

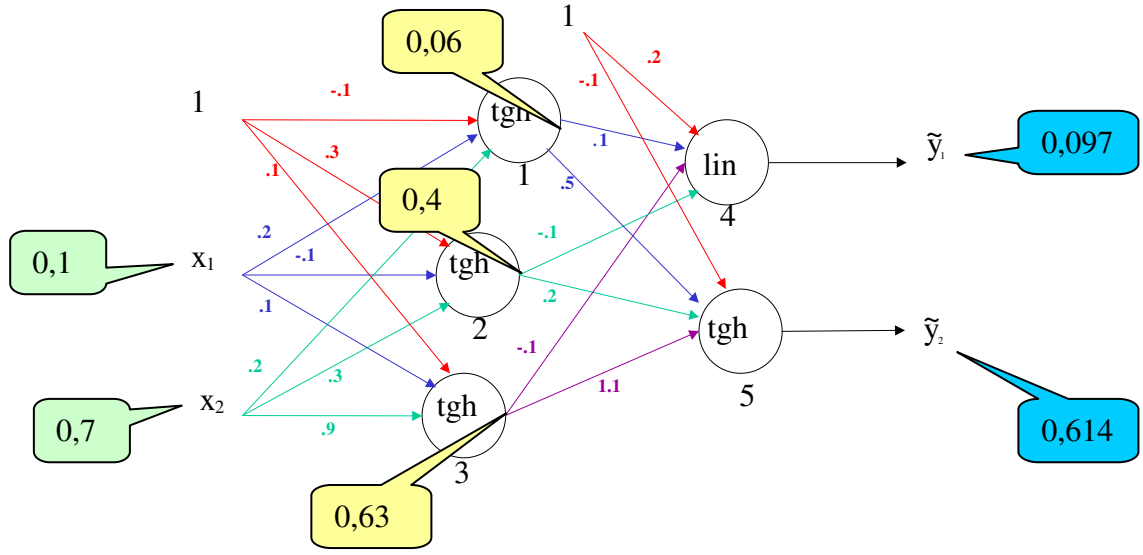
Processo em Batelada – Batch

Exemplo anterior:



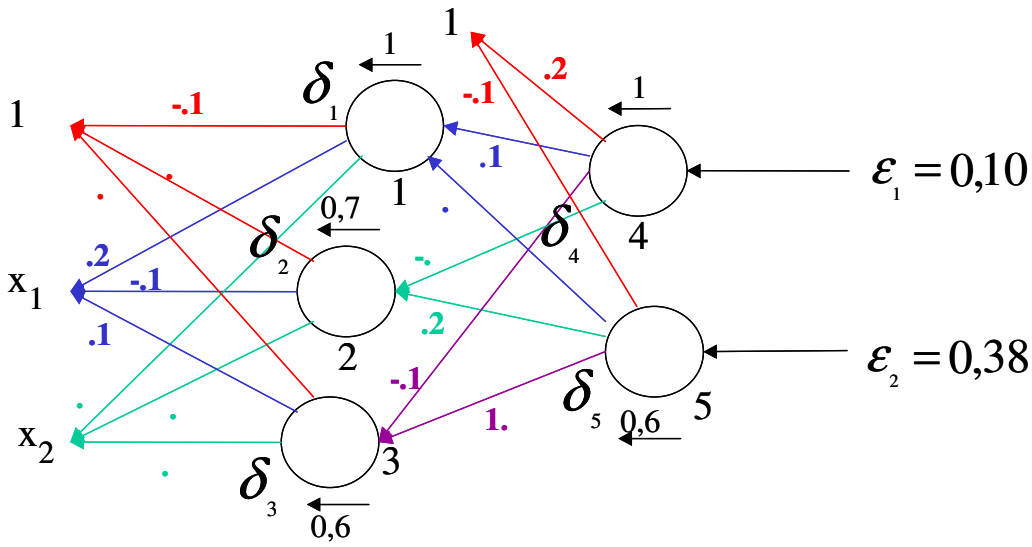
$$\underline{\mathbf{x}} = \begin{bmatrix} 0,1 \\ 0,7 \end{bmatrix} \qquad \tilde{\mathbf{y}} = ? \qquad \mathbf{y} = \begin{bmatrix} 0,2 \\ 0,1 \end{bmatrix}$$

Signal Feedforward



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

Rede associada e retropropagação do erro:



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

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

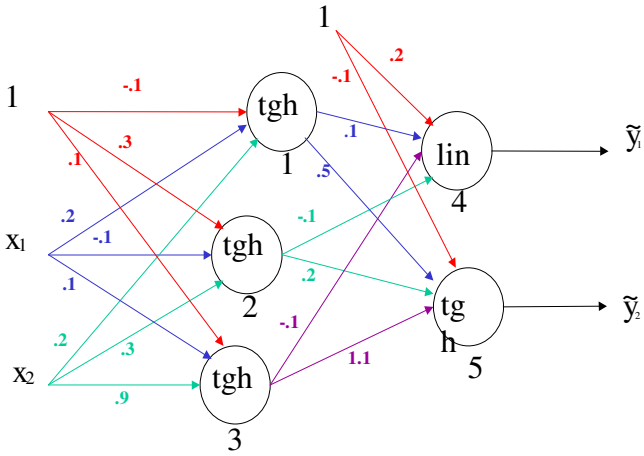
$$\delta_2 = [(0,103)(-0,1) + (0,24)(0,2)](0,79) = 0,030$$

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

Treinamento Regra Delta

Atualização dos valores das sinapses:

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



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

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

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

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

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

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

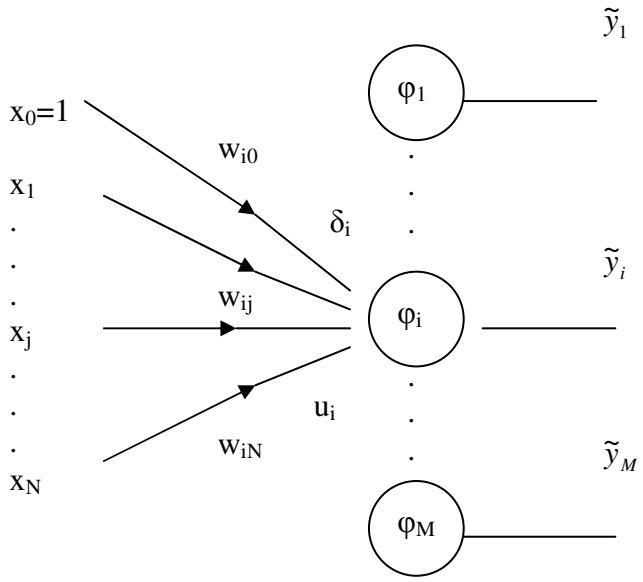
$$\Delta w_{3b} = (2)(0,1)(0,7)(0,152) = 0,021$$

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

Algorítmos específicos:

Algorítmo para Redes com uma Camada

Formulário para Treinamento Batelada



Para cada par entrada-saída p = 1,...,P

Para todos os neurônios i = 1,...,M

Signal feedforward:

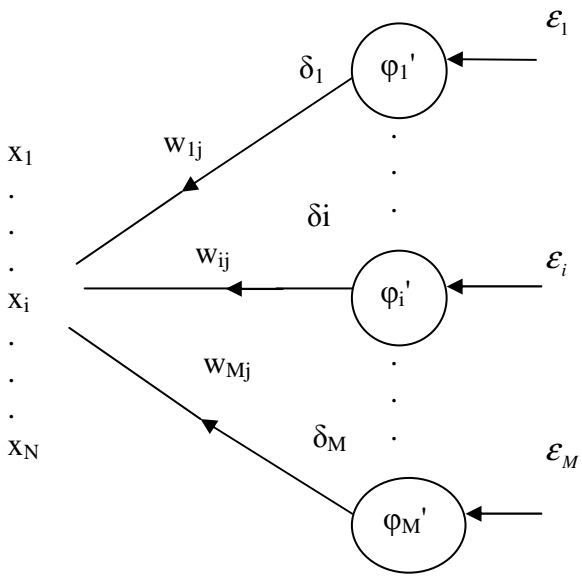
$$u_i = \sum_{j=0}^{N} w_{ij} x_j$$

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

Erro na saída:

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

Rede associada:



Erro retropropagado até a saída das sinapses:

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

Acréscimo nas sinapses devido ao par p:

Para todo i = 1,...,M e j = 0,1,...,N

$$\Delta w_{ij}(p) = 2\alpha\, x_j \delta_i$$

Outro par

Acréscimo nas sinapses:

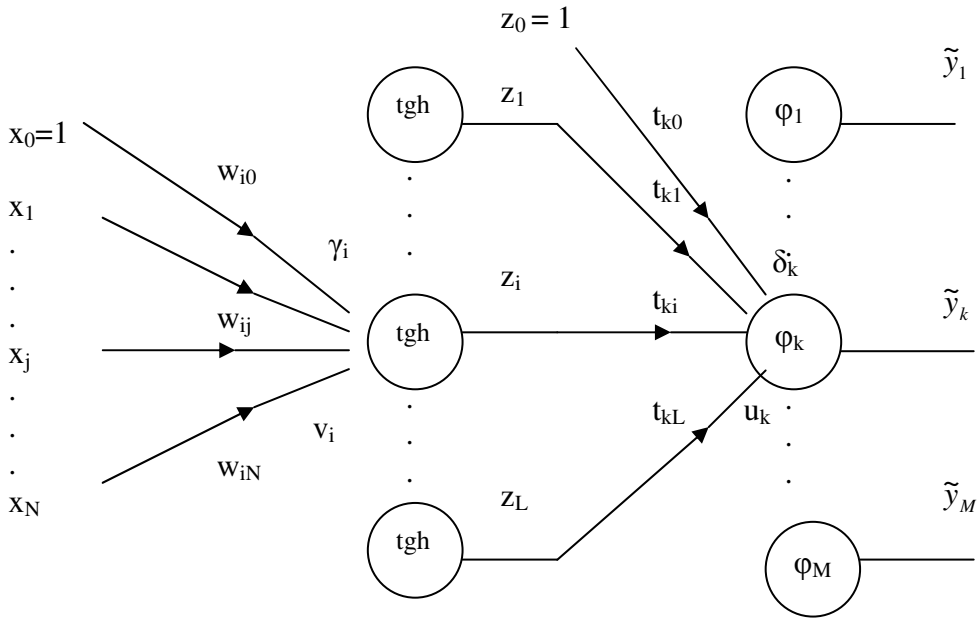
Para todo i = 1,...,M e j = 0,1,...,N

$$\Delta w_{ij} = E\left[\Delta w_{ij}(p)\right] = \frac{1}{P}\sum_{p=1}^{P} \Delta w_{ij}(p)$$

Reiniciar até que a condição de parada esteja satisfeita

Algorítmo para Redes com duas camadas

Formulário para Treinamento Batelada



Para cada par entrada-saída p = 1,...,P

Signal feedforward:

Para todos os neurônios da primeira camada i = 1,...,L

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

$$z_i = tgh\,(v_i) \qquad z_0 = 1$$

Para todos os neurônios da segunda camada k = 1,..., M

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

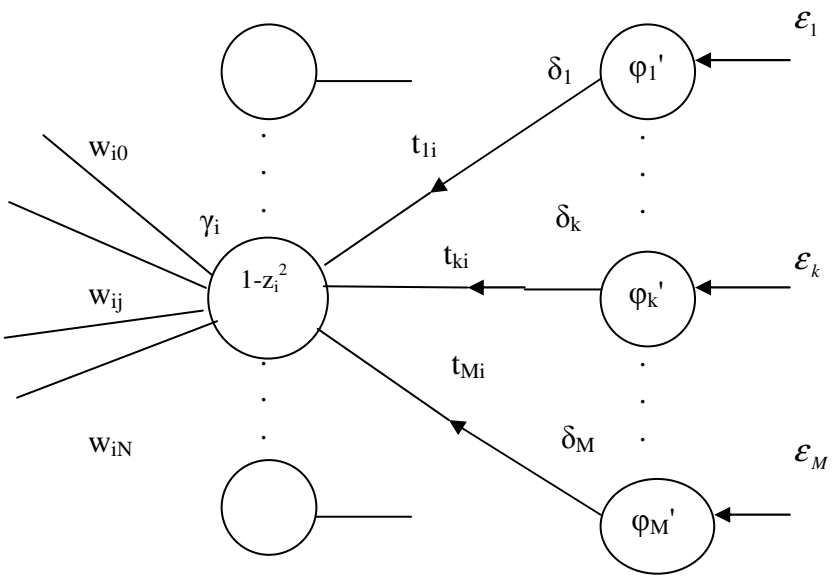
$$\tilde{y}_k = \varphi_k(u_k) = \begin{cases} u_k & \textit{neurônio linear} \\ tgh\,(u_k) & \textit{neurônio tgh} \end{cases}$$

Erro na saída:

Para todos os neurônios da segunda camada k = 1,..., M

$$\varepsilon_k = y_k - \tilde{y}_k$$

Rede associada:



Retropropagação do erro

Para todo neurônio da segunda camada, k = 1,...,M:

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

Para todo neurônio da primeira camada, i = 1,...,L

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

Acréscimo nas sinapses da primeira camada devido ao par p:

Para todo j = 0,1,...,N e i = 1,...,L

$$\Delta w_{ij}(p) = 2\,\alpha\, x_j \gamma_i$$

Acréscimo nas sinapses da segunda camada devido ao par p:

Para todo i = 0,1,...,L e k = 1,...,M

$$\Delta t_{ki}(p) = 2\alpha\, z_i\, \delta_k$$

Outro par

Acréscimo nas sinapses da primeira camada:

Para todo j = 0,1,...,N e i = 1,...,L

$$\Delta w_{ij} = E[\Delta w_{ij}(p)] = \frac{1}{P}\sum_{p=1}^{P} \Delta w_{ij}(p)$$

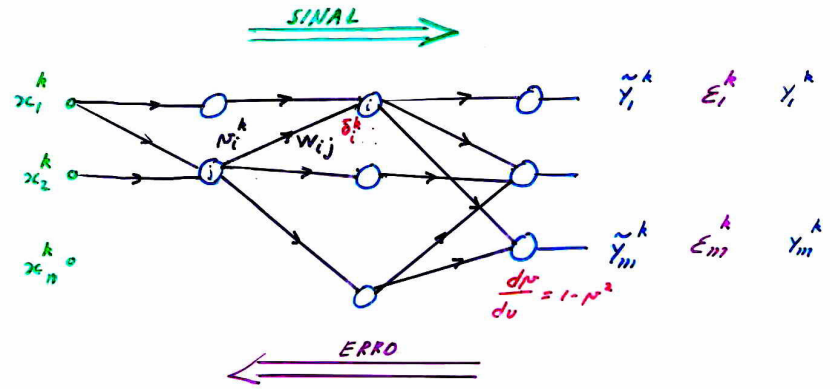
Acréscimo nas sinapses da segunda camada:

Para todo i = 0,1,...,L e k = 1,...,M

$$\Delta t_{ki} = E[\Delta t_{ki}(p)] = \frac{1}{P}\sum_{p=1}^{P} \Delta t_{ki}(p)$$

Reiniciar até que a condição de parada esteja satisfeita.

Error Backpropagation - Resumo:



$$\frac{\partial \mathcal{E}^{l\,k}}{\partial w_{ij}} = -2\, y_j^k\, \delta_i^k$$

$$\nabla F_o = \frac{\partial F_o}{\partial w_{ij}} = -2\, E(y_j\, \delta_i)$$

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