Redes Neurais Feedforward

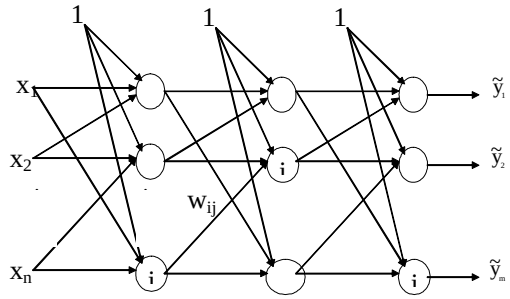
Aprendizado (Treinamento)

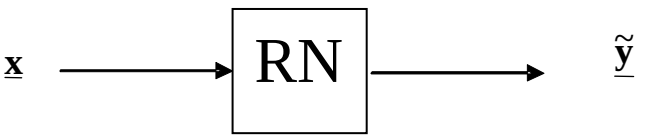
Backpropagation

Aproximador Universal

Redes Neurais *FeedForward*

Redes "*Backpropagation*"

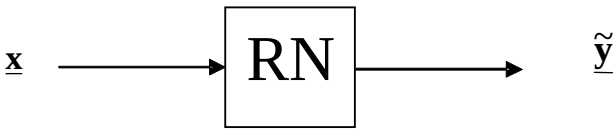




$$\underline{\tilde{y}} = \varphi\left(\underline{x}\right)$$

Mapeador não linear

Aproximador Universal !



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

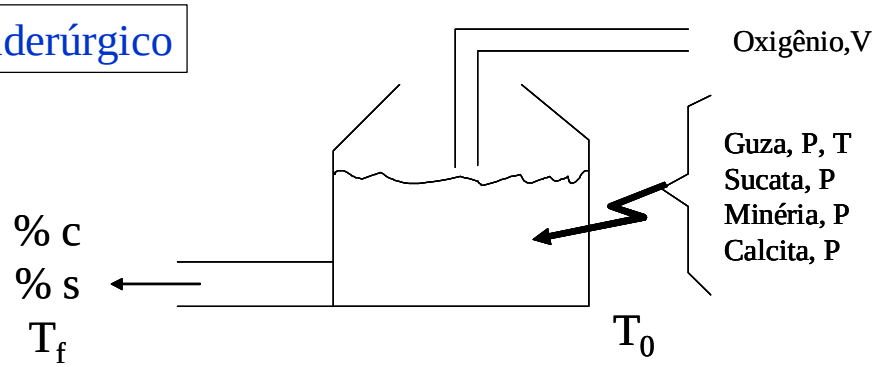
Para que serve ?

Aplicações:

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

Simulação de Sistemas

Conversor Siderúrgico

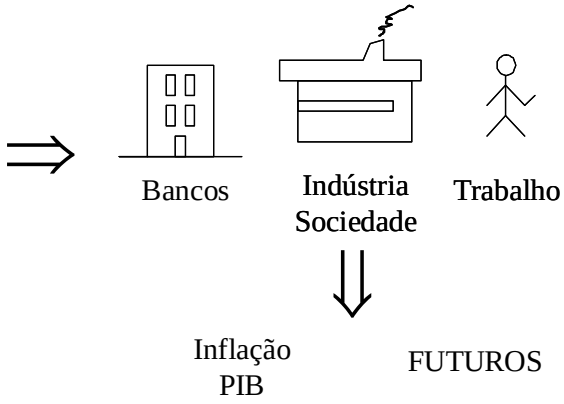


Simulação de Sistemas

Sistema Econômico

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

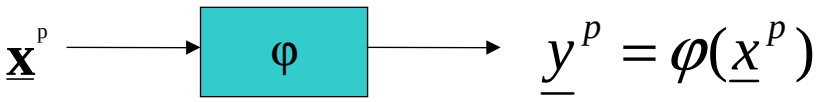
PASSADOS



Simulação de Sistemas

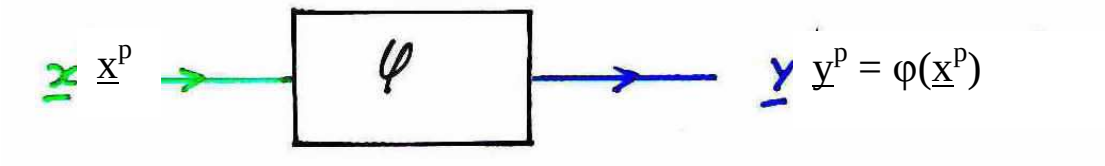
Sistema Real, Planta

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



Simulador

Sistema à emular



Conhecimento do Sistema

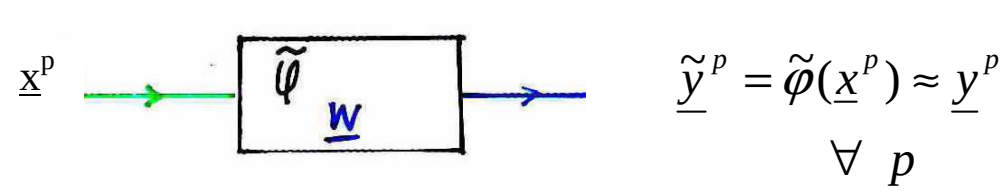
Fenomenológico – equações – modelo caixa branca - NÃO

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

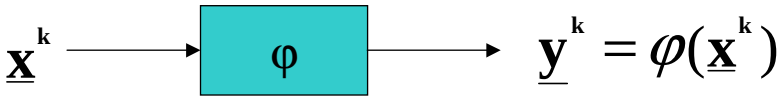
Modelo Numérico Geral (e.g. rede neural) – Caixa Preta

Ajuste dos parâmetros $\underline{w}$?

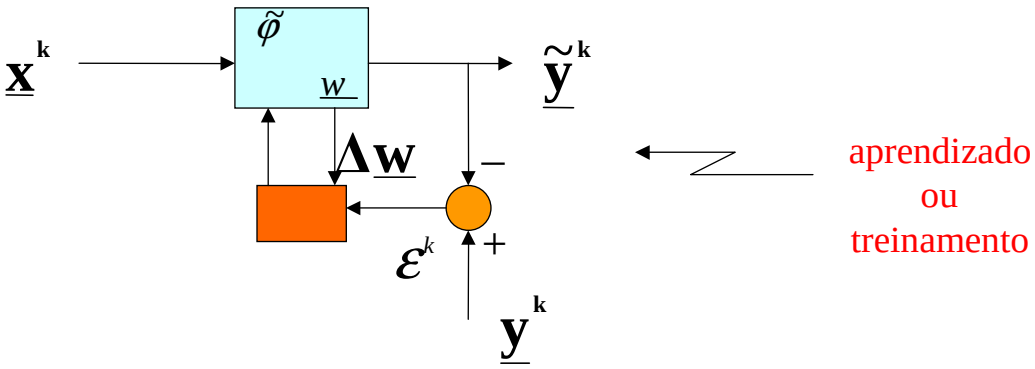
Simulação de Sistemas

Sistema Real, Planta

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



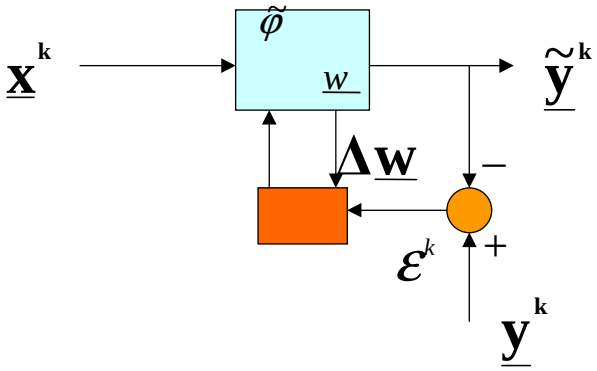
Simulador



Aprendizado

(ou treinamento, ou ajuste de parâmetros)

como um processo de otimização



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:

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

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

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

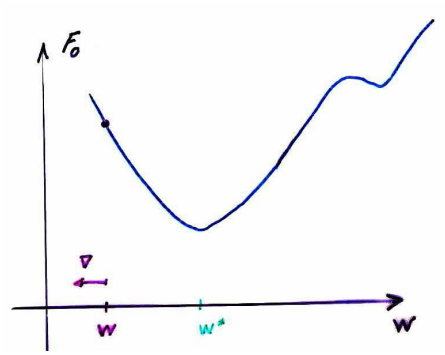
i.e. $\Delta F_0 < 0$ e $|\Delta F_0|$ é máximo

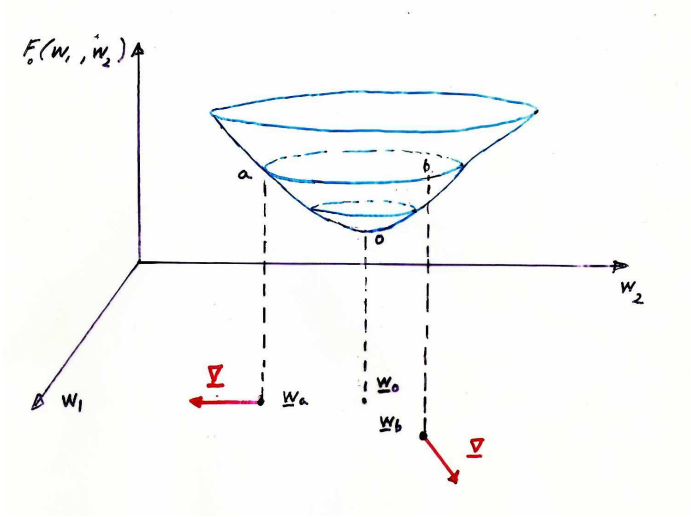
2. $\underline{\nabla}\big|_{\underline{\mathbf{w}_0}} = \underline{\mathbf{0}}$

Gradiente descendente

ou descida por gradiente

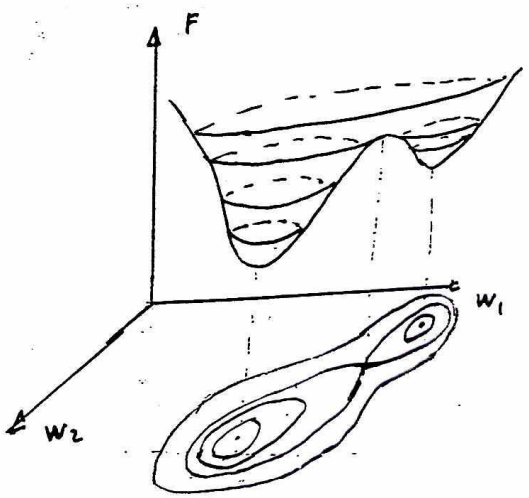
$$\underline{w} \longrightarrow \Delta\underline{w} = -\alpha \nabla(\underline{w}) \longrightarrow \underline{w} = \underline{w} + \Delta\underline{w}$$

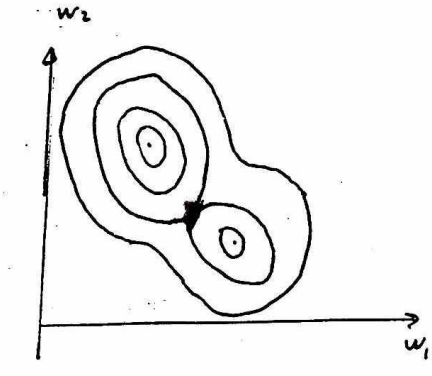




Visualização

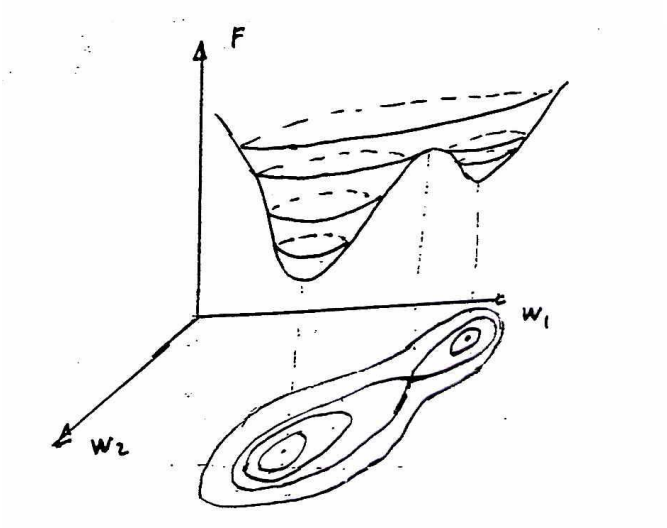
por Curvas de Nível

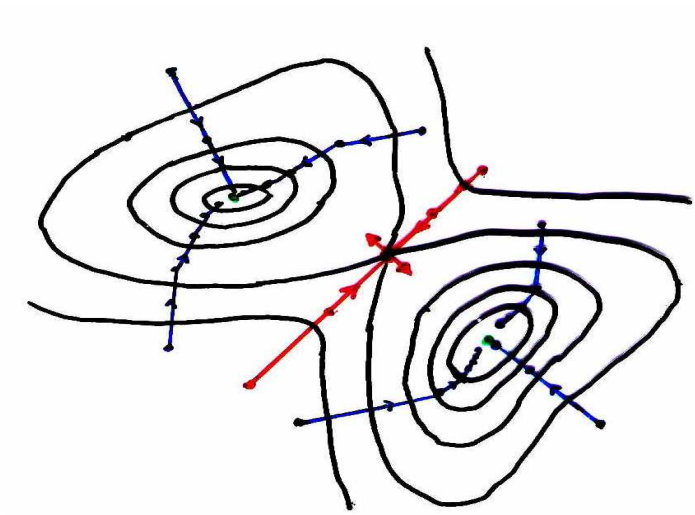




Gradiente descendente ou descida por gradiente

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





sela instável **mínimos locais**

Complexidade de cálculo:

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

Fim do processo:

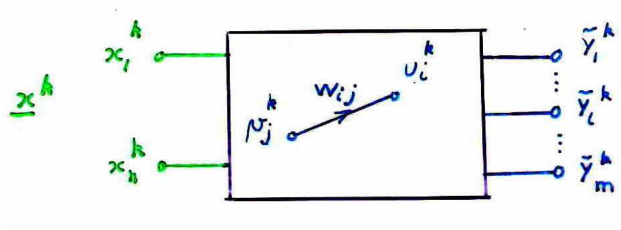
$$\nabla \underline{w}_{otimo} = \underline{0} \quad \Rightarrow \quad \Delta \underline{w}_{otimo} = \underline{0}$$

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

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



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

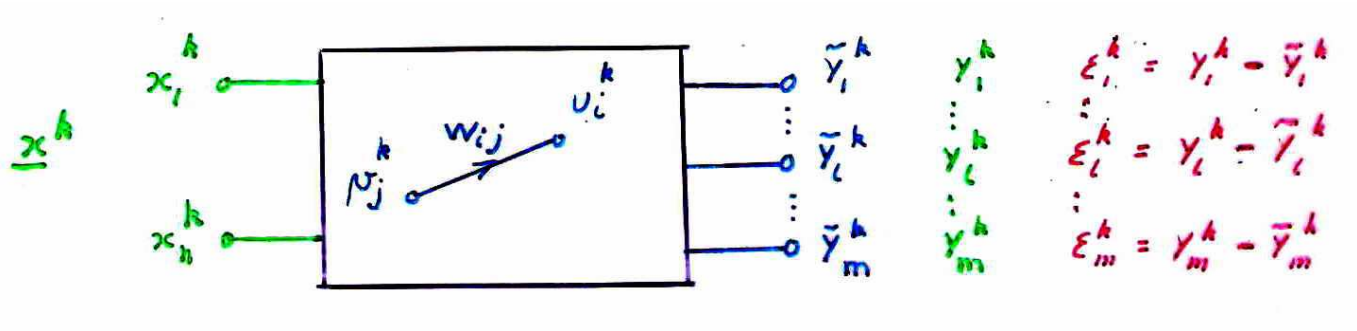
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{\partial \underset{k}{E}\left[(\varepsilon^k)^2\right]}{\partial w_{ij}} = \underset{k}{E}\left[\frac{\partial(\varepsilon^k)^2}{\partial w_{ij}}\right]$$

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



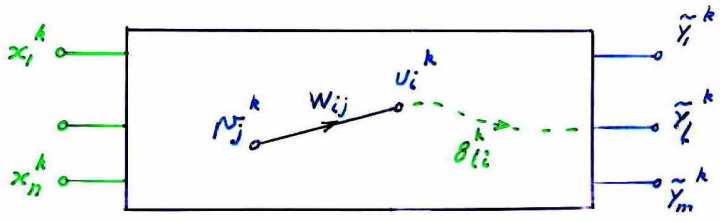
$$(\varepsilon)^2 = \sum_{l=1}^{m} (\varepsilon_l)^2 = \sum_{l=1}^{m} (y_l - \tilde{y}_l)^2$$

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

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

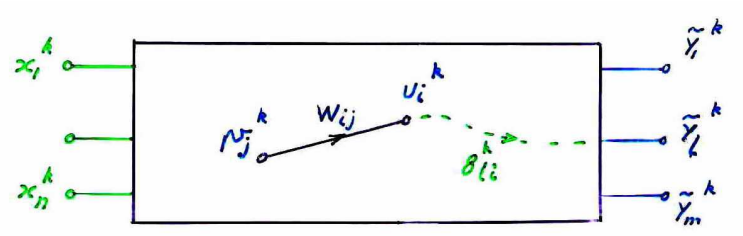
$$= 2\sum_{l=1}^{m} \varepsilon_l \frac{\partial(y_l - \tilde{y}_l)}{\partial w_{ij}} = -2\sum_{l=1}^{m} \varepsilon_l \frac{\partial \tilde{y}_l}{\partial w_{ij}} = -2\sum_{l=1}^{m} \varepsilon_l \frac{\partial \tilde{y}_l}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}}$$

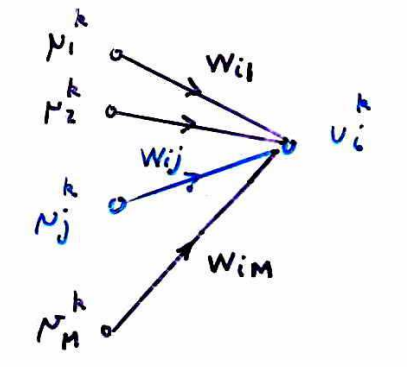
$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li}$$



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

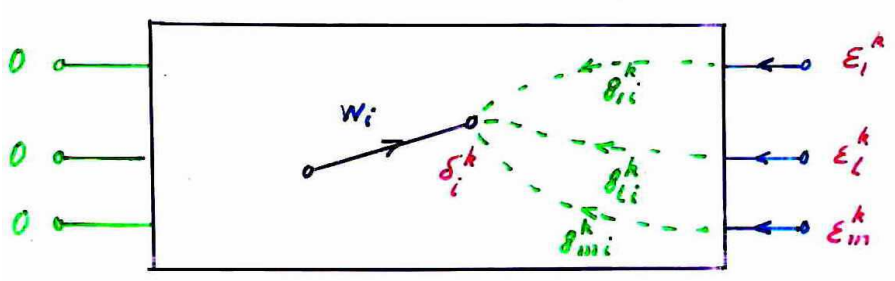
$$\frac{\partial \tilde{y}_l}{\partial u_i} = g_{li} \qquad u_i = \sum_{m=1}^{M} w_{mj} v_j \qquad \frac{\partial u_i}{\partial w_{ij}} = v_j$$





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

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



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

3 - como:

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

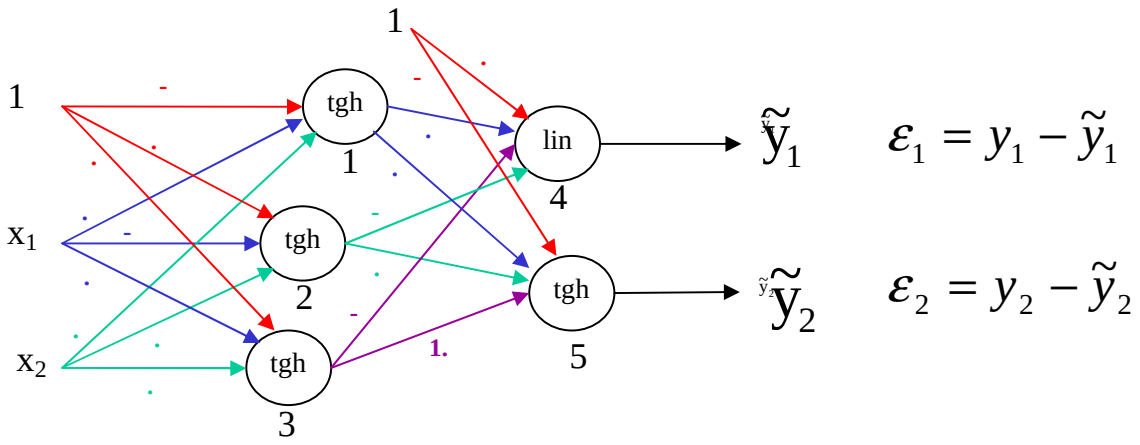
$$\frac{\partial F_0(w_{ij})}{\partial w_{ij}} = \frac{1}{P}\sum_{p=1}^{P} \frac{\partial}{\partial w_{ij}}(\varepsilon^{2\,p})$$

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

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

Error Backpropagation -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.

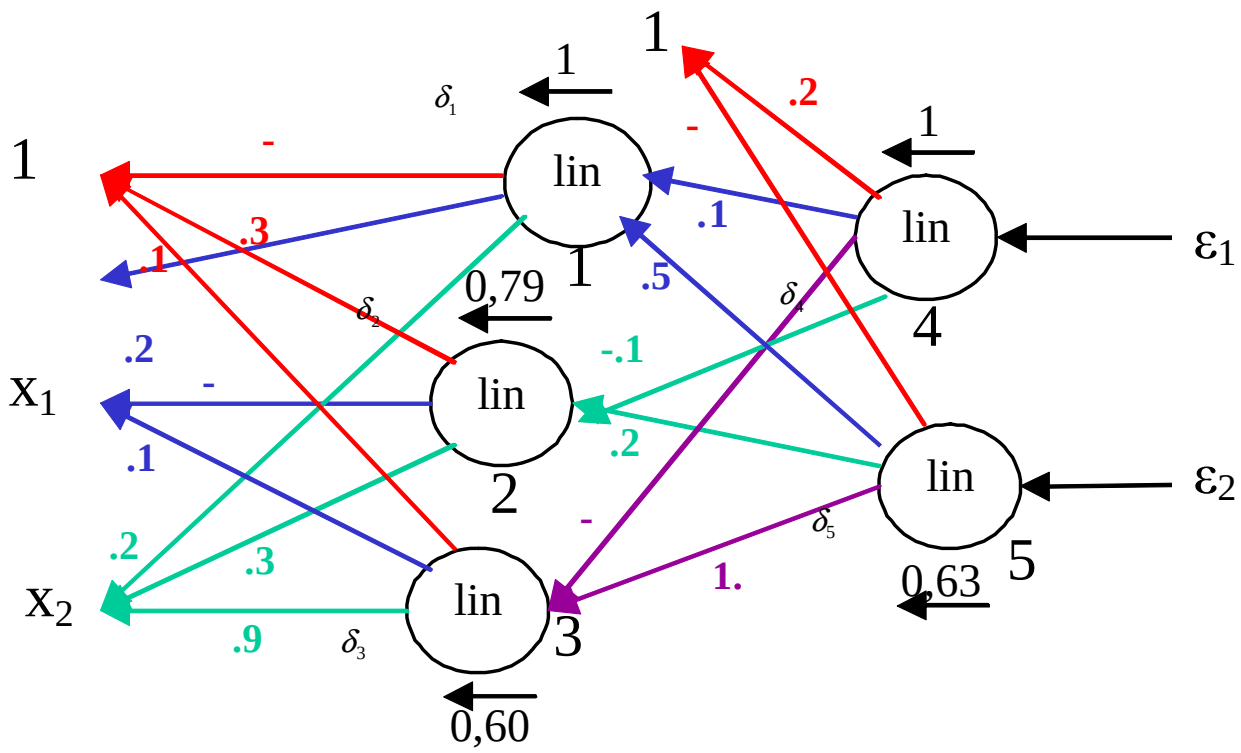
2 – Criar uma "Rede Associada" a partir da Rede Original

mesma arquitetura, mas

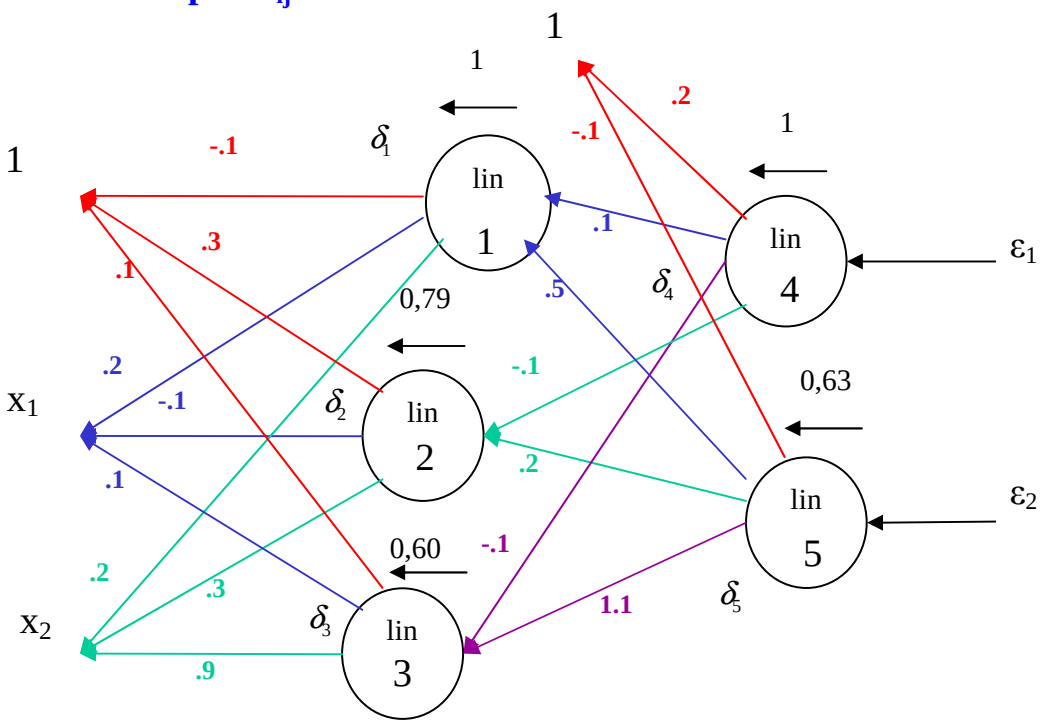
2.1 - "Linearizar" (os neurônios) da rede original

Rede Original	→	Rede Associada
Neurônio não linear $v_0 = tgh(u_0)$	→	**Neurônio linear** $v = (1-v_0^2)\,u$
Neurônio linear $v_0 = u_0$	→	**Neurônio linear** $v = u$

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:

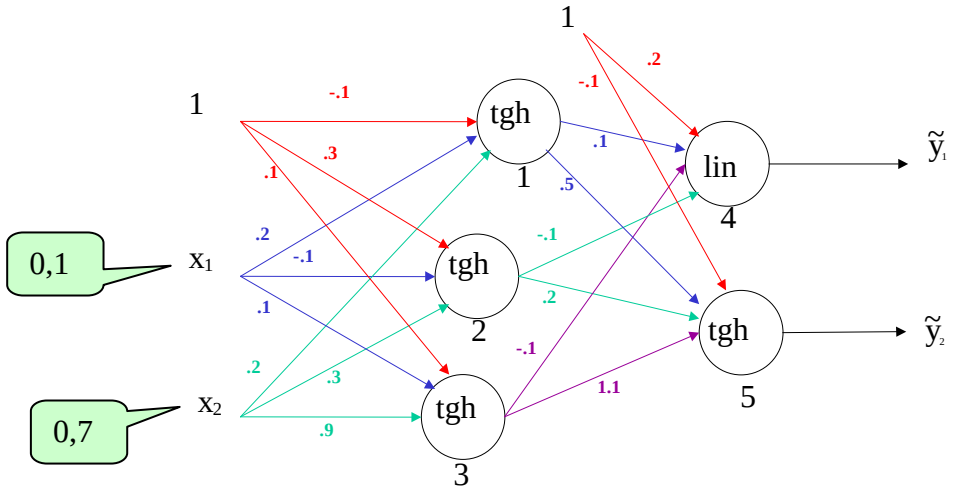
$$\Delta_p w_{ij} = 2\,\alpha\, v_j\, \delta_i$$

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

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

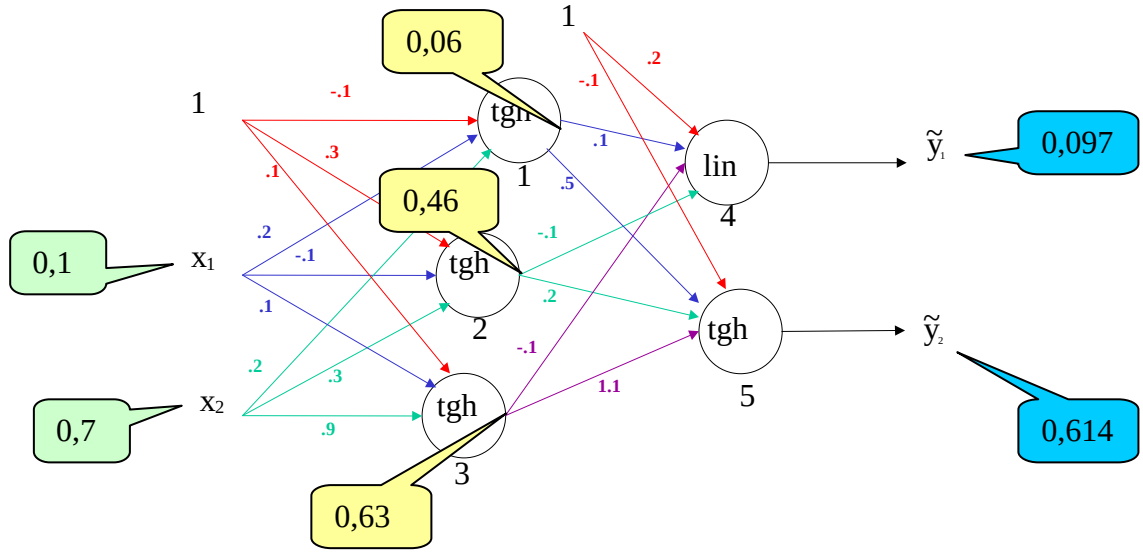
Processo em Batelada – Batch

Exemplo anterior:



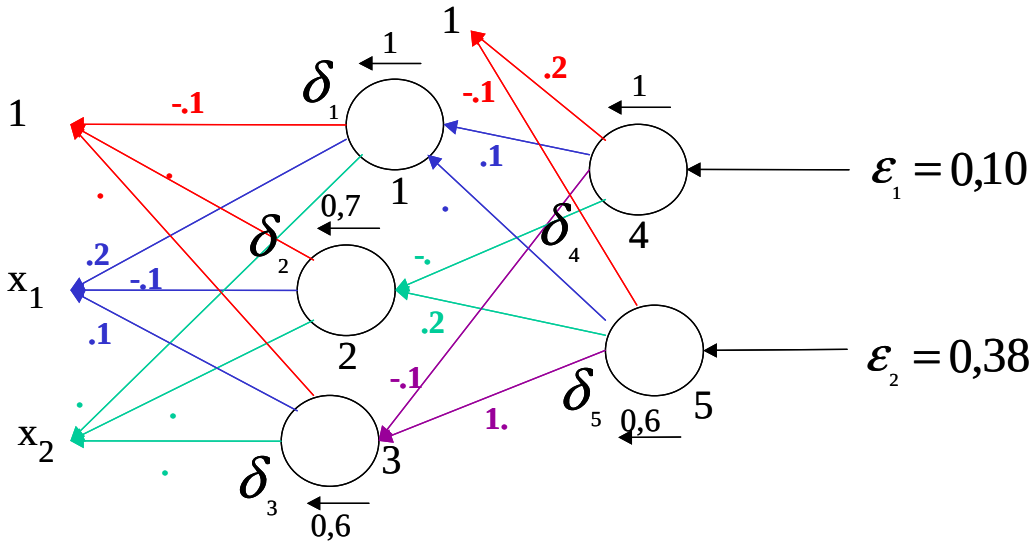
$$\underline{\mathbf{x}} = \begin{bmatrix} 0,1 \\ 0,7 \end{bmatrix} \qquad \tilde{\mathbf{y}} = ? \qquad \mathbf{y} = \begin{bmatrix} 0,2 \\ 0,1 \end{bmatrix}$$

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

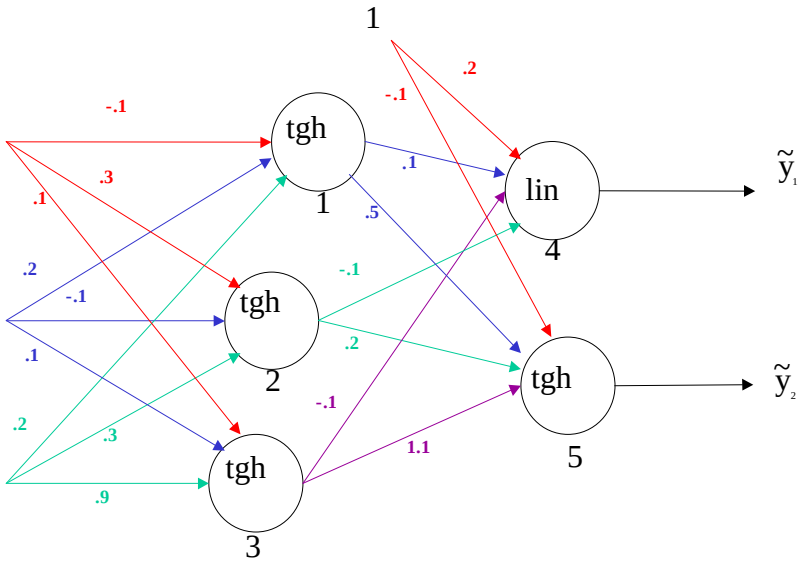
Atualização dos valores das sinapses:

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

$\Delta b_4 = (2)(0$

$\Delta b_1 = (2)(0$

$\Delta w_{41} = (2)($



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

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

Algorítmos 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

Rede Original

Definição das variáveis:

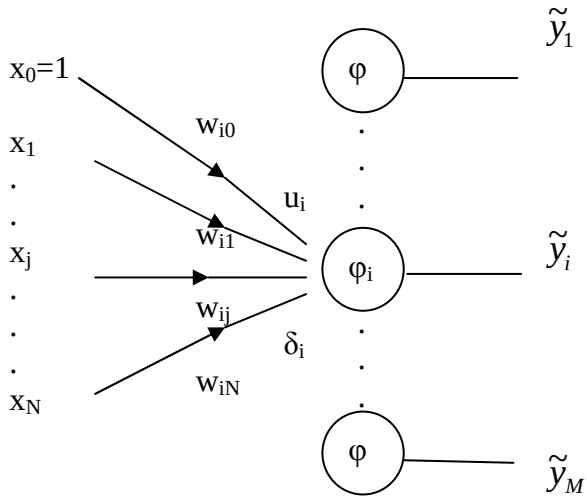
N entradas: $x_1, x_2, ..., x_N$ e x_0 (bias) = 1

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

M neurônios com função de ativação φ(.) linear ou tgh

u_i - excitação interna do neurônio i

$\tilde{y}_i$ - saída do neurônio i



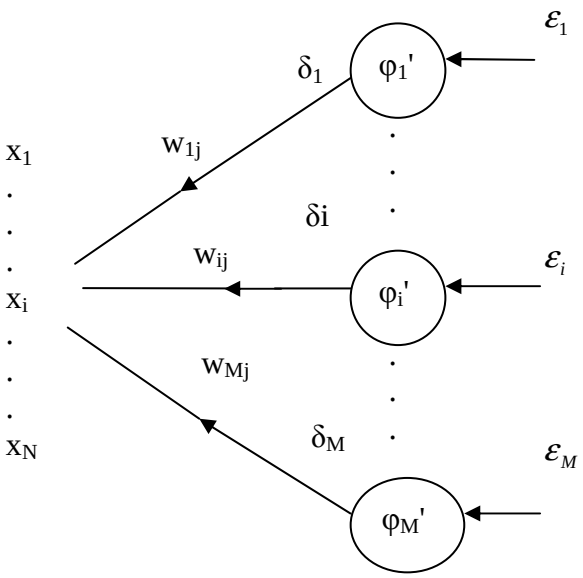
Definição das variáveis:

Rede Associada

ε_i - erro na saída do neurônio i

δ_i - erro retropropagado até a entrada do neurônio i

φ_i' - valor numérico da derivada de φ_i no ponto de operação



Algorítmo BP acréscimo Δw_{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:

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

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

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

Calcule o erro retropropagado até a entrada de cada neurônio:

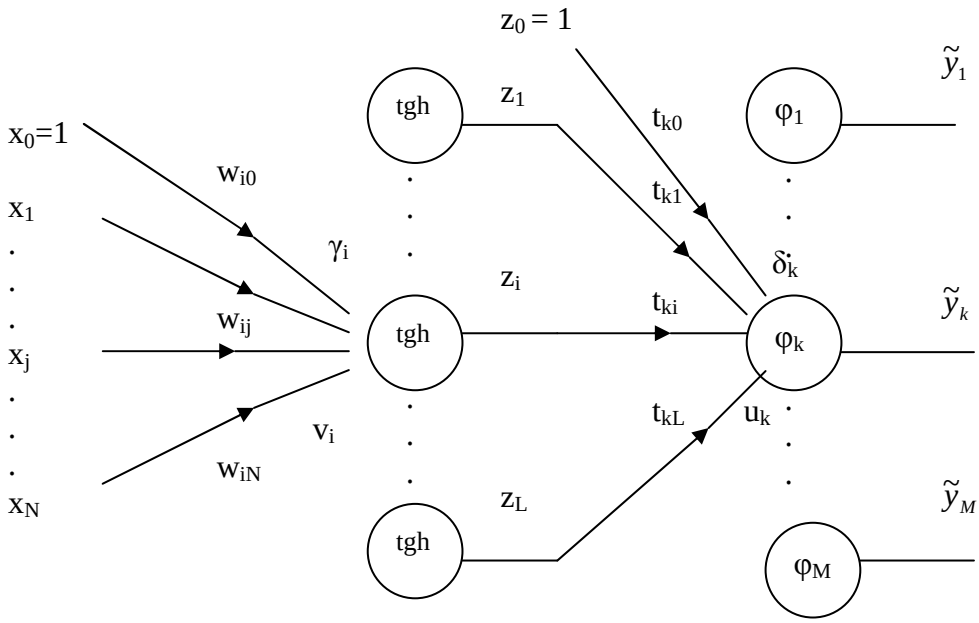
$$\delta_i = \begin{cases} \varepsilon_i & \textit{neurônio linear} \\ \left(1 - \tilde{y}_i^2\right)\varepsilon_i & \textit{neurônio tgh} \end{cases}$$

Calcule o acéscimo em cada sinapse devido ao par p:

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

fim do algorítmo

Redes com duas camadas - Rede Original



Definição das variáveis:

N entradas: $x_1, x_2, ..., x_N$ 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 φ(.) 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.

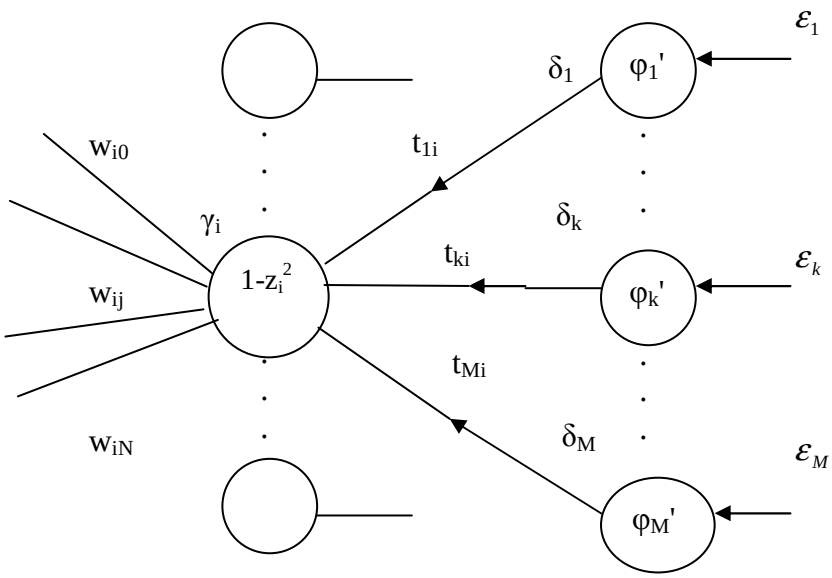
v_i - excitação interna do neurônio i da primeira camada

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:

ε_k - erro na saída do neurônio da camada de saída

δ_k - erro retropropagado até a entrada do neurônio k da camada de saída

φ_i' - valor numérico da derivada de φ_i no ponto de operação

Algorítmo BP acréscimos Δw_{ij}^p e Δt_{ki}^p - Rede com 2 camadas

Para o par entrada-saída p (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,...,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

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

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

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

Retropropagação do erro

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

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

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:

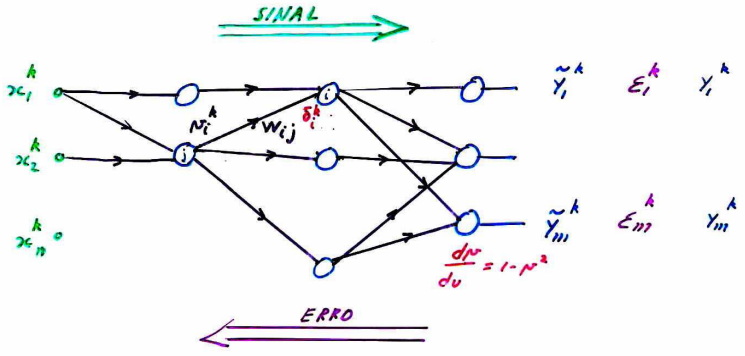
$$\Delta w_{ij}^p = 2\,\alpha\, x_j \gamma_i \qquad \forall\, j = 0,1,....,N \quad e \quad i = 1,2,...,L$$

Acréscimo nas sinapses da segunda camada devido ao par p:

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

fim do algorítmo

Error Backpropagation - Resumo:



$$\frac{\partial \varepsilon^{2\,k}}{\partial w_{ij}} = -2\,\rho_j^k\,\delta_i^k$$

$$\nabla F_0 = \frac{\partial F_0}{\partial w_{ij}}\Big] = -2\;E(\rho_j\,\delta_i)\Big]$$

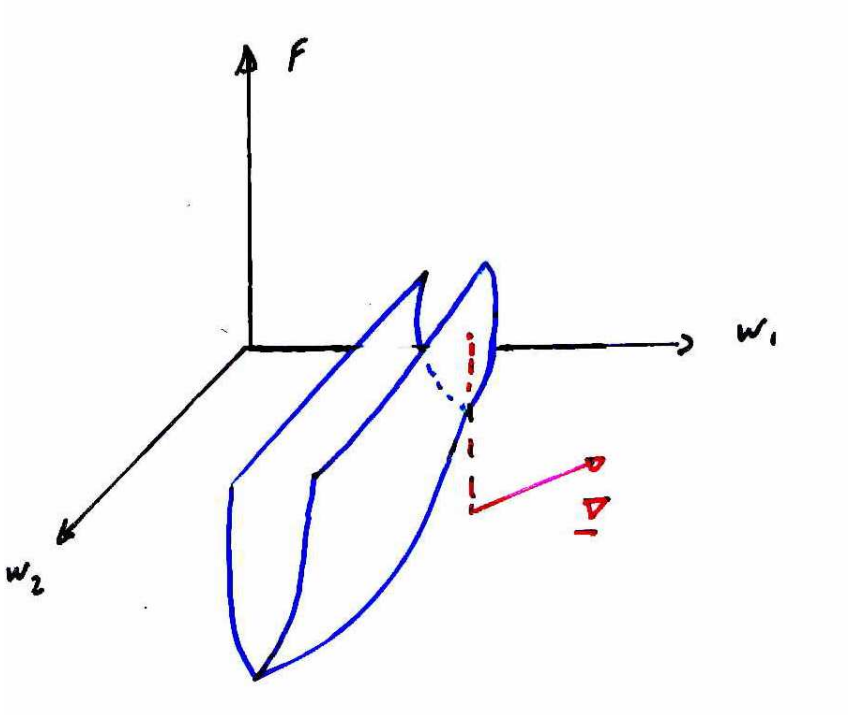
$$\Delta w_{ij} = 2\,\alpha\; E(\rho_j\,\delta_i)$$

Método de Primeira Ordem Aprimorado

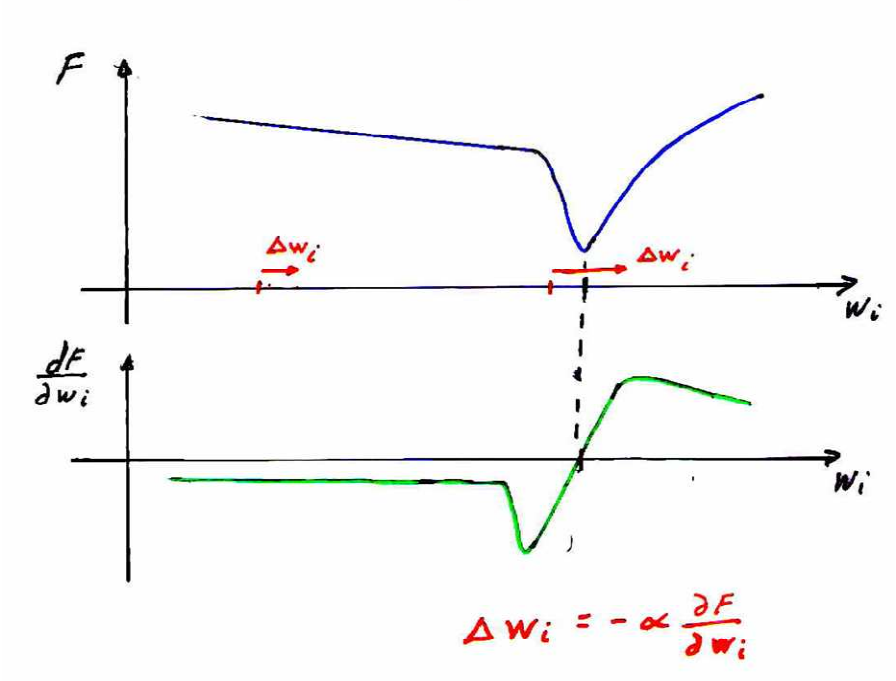
Resilient Backpropagation

Acelerando o processo

passo variável
por sinapse



Acelerando o processo – passo variável no tempo



Resilient Backpropagation (modificado)

(Silva e Almeida, Super SAB, etc.)

α variável $\begin{cases} \text{por variável } w_i \\ \text{por passo de treinamento n} \end{cases}$

$$\Delta w_i(n) = - \alpha_i(n) \frac{\partial F}{\partial w_i}(n)$$

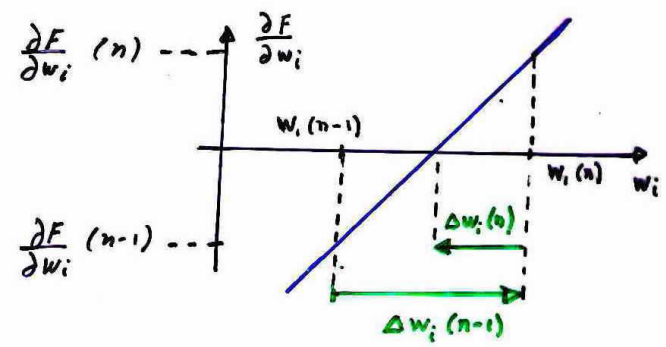
$$\alpha_i(n) = \begin{cases} a\,\alpha_i(n-1) & \text{se } \operatorname{sign}\frac{\partial F}{\partial w_i}(n) = \operatorname{sign}\frac{\partial F}{\partial w_i}(n-1) \\ b\,\alpha_i(n-1) & \text{se } \operatorname{sign}\frac{\partial F}{\partial w_i}(n) \neq \operatorname{sign}\frac{\partial F}{\partial w_i}(n-1) \end{cases}$$

se $\alpha_i(n) > \alpha^*$ faça $\alpha = \alpha^*$

$\alpha_i(0) \approx .05 \quad \alpha^* \approx .3 \quad a \approx 1.05 \quad b \approx .9$

Outra alternativa

$$b = \frac{\frac{\partial F}{\partial w_i}(n-1)}{\frac{\partial F}{\partial w_i}(n-1) - \frac{\partial F}{\partial w_i}(n)}$$



BP Resiliente - Algorítmo

Inicializar: $n = 1;\quad \alpha_i(1) = .1;\quad \alpha_{\max} \approx .5;\quad a \approx 1.05;\quad b \approx .9;\quad \frac{\partial F}{\partial w_i}(0) = 0^+;$

$w_i(1) \in [+.2, -.2]$ *randômicos*;

Até que o critério de parada esteja satisfeito

Passo de treinamento n

Cálculo das derivadas por sinapse

para cada par $(\underline{x}^p, \underline{y}^p)$ p = 1, …, P

calcular $\underline{\tilde{y}}^p = \varphi(\underline{x}^p)$ $\varepsilon_m^p = y_m^p - \tilde{y}_m^p$ e $\frac{\partial \tilde{y}_m^p}{\partial w_i}$ $\forall$ m, i

$$\frac{\partial \varepsilon^{p2}}{\partial w_i} = -2 \sum_{m=1}^{M} \varepsilon_m^p \frac{\partial y_m^p}{\partial w_i} \qquad \forall i$$

outro par

calcular $\frac{\partial F}{\partial w_i}(n) = E_p \frac{\partial \varepsilon^{p2}}{\partial w_i}$

Cálculo do acréscimo

para i = 1, 2, …

$$\alpha_i(n) = \begin{cases} a\,\alpha_i(n-1) & \text{se } \operatorname{sign}\frac{\partial F}{\partial w_i}(n) = \operatorname{sign}\frac{\partial F}{\partial w_i}(n-1) \\ & \textit{mas se } \alpha_i(n) > \alpha_{\max} \textit{ faça } \alpha_i(n) = \alpha_{\max} \\ \dfrac{\frac{\partial F}{\partial w_i}(n-1)}{\frac{\partial F}{\partial w_i}(n-1) - \frac{\partial F}{\partial w_i}(n)}\alpha_i(n-1) & \text{se } \operatorname{sign}\frac{\partial F}{\partial w_i}(n) \neq \operatorname{sign}\frac{\partial F}{\partial w_i}(n-1) \end{cases}$$

$$\Delta w_i(n) = -\alpha_i(n) \frac{\partial F}{\partial w_i}(n)$$

Atualização das sinapses

$$w_i(n+1) = w_i(n) + \Delta w_i(n)$$

n = n + 1

fim do algorítmo

Métodos de Segunda Ordem

necessitam das derivadas de segunda ordem e da inversão da Hessiana !

$$\underline{d} = -\underline{H}^{-1}\underline{\nabla}$$

Métodos de 2ª Ordem

Newton

Newton Amortecido

Levemberg-Marquadt

Quasi Newton

BFGS

Gradiente Conjugado

Comentários finais

Primeira ordem

Passo fixo (“vanilla”) – **convergência usualmente muito lenta**

BP Resiliente (Silva & Almeida, SuperSAB) – **bom compromisso entre estabilidade e rapidez na convergência.**

Segunda ordem – rápidos, mas mais complexos e com risco de instabilidade na convergência

Levenberg – Marquardt (Newton amortecido)

BFGS (quase Newton)