SCC0270/SCC5809 - Redes Neurais

Aula 3 - *Multi-Layer Perceptron* (MLP)

Profa. Dra. Roseli Aparecida Francelin Romero
SCC - ICMC - USP

2018

Sumário

Perceptron multicamadas

- Redes de apenas uma camada representam somente funções linearmente separáveis.
- Redes de múltiplas camadas solucionam essa restrição.
- O desenvolvimento do algoritmo *backpropagation* foi um dos motivos para o ressurgimento da área de redes neurais [Rumelhart *et. al*, 1986].

Sumário

Modelo de rede neural com múltiplas camadas.

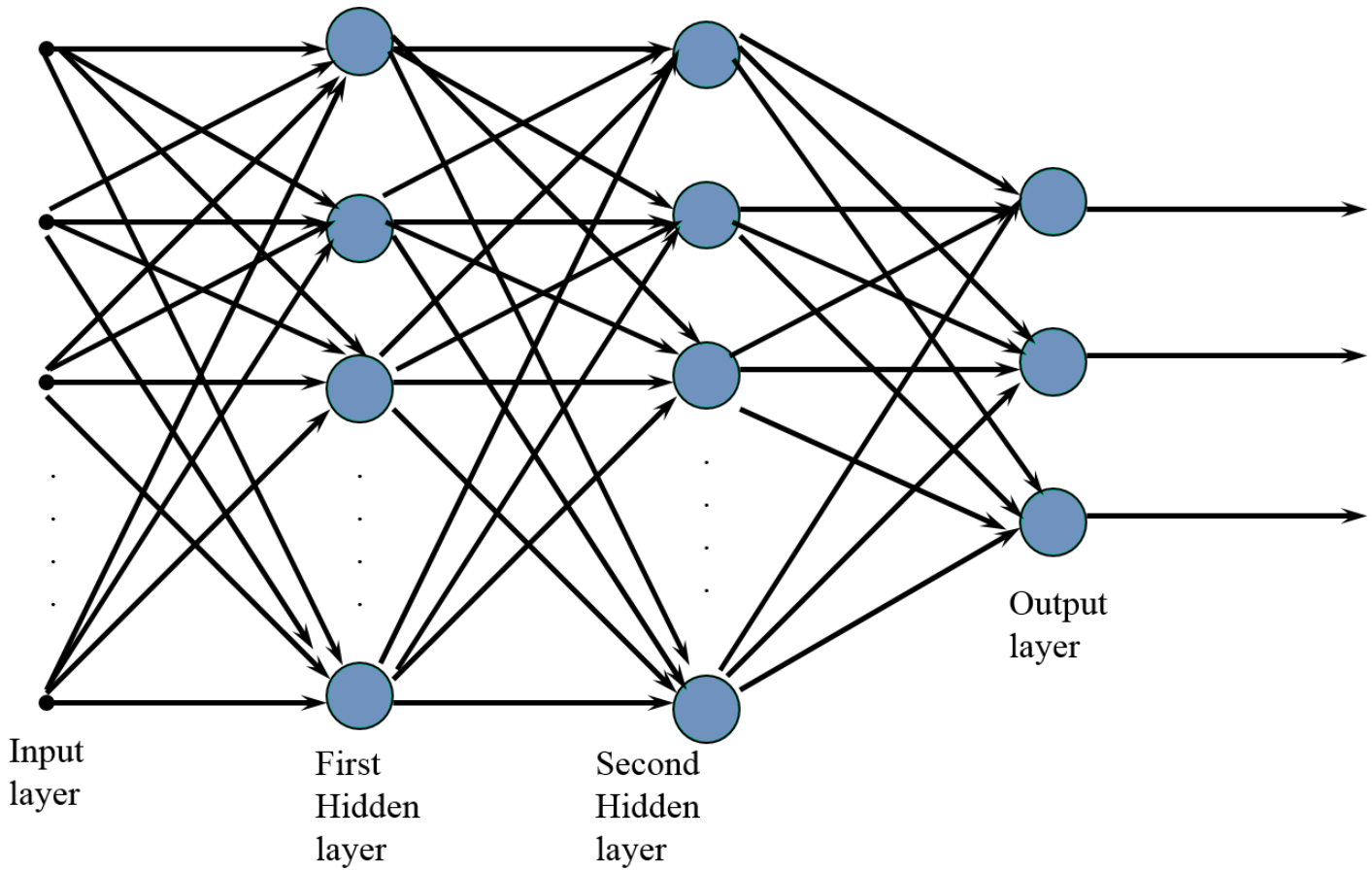


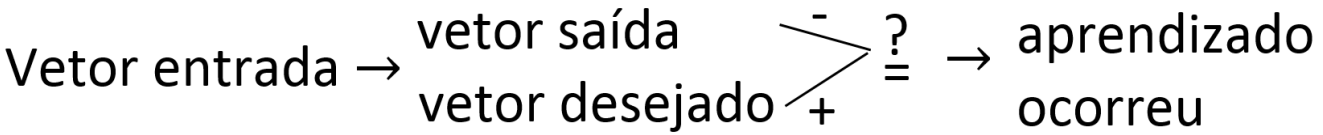
Figura 1: Rede neural *feed-forward* com múltiplas camadas.

Sumário

Sumário

Aprendizado da rede

- O esquema de aprendizado da rede pode ser descrito do seguinte modo:



Aprendizado da rede

- Caso contrário, os pesos são modificados para minimizar o erro:

$$E(w) = \sum_{p=1}^{N} E_p(w)$$

onde N é o no. total de padrões e E_p é o erro quadrático referente a cada par p apresentado à rede, sendo dado por:

$$E_p = \frac{1}{2} \sum_{j} (t_{pj} - y_{pj})^2$$

onde:

- t_{pj}: j-ésima componente do vetor saída desejada.
- y_{pj}: j-ésima componente do vetor obtido pela rede.

Aprendizado da rede

Pesos (*Gradient Descent Method*)

$$w_{ji}(k+1) = w_{ji}(k) - \eta \frac{\partial E_p(w)}{\partial w_{ji}}\Big|_{w(k)}$$

Onde η é uma constante positiva (velocidade de aprendizado).

- Calculando a derivada parcial do E_p, tem-se:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial v_{pj}} \cdot \frac{\partial v_{pj}}{\partial w_{ji}}$$

- Para se calcular $\frac{\partial E_p}{\partial y_{pj}}$, dois casos devem ser considerados:

Aprendizado da rede

1. Neurônio j está na camada de saída.

$$\frac{\partial E_p}{\partial y_{pj}} = -(t_{pj} - y_{pj})$$

$$\therefore \frac{\partial E_p}{\partial w_{ji}} = \underbrace{-(t_{pj} - y_{pj}) \cdot \overbrace{y_{pj}(1 - y_{pj})}^{\frac{\partial y_{pj}}{\partial v_{pj}}}}_{\delta_{pj}} \cdot y_{pi}$$

$$\boxed{\frac{\partial E_p}{\partial w_{ji}} = -\delta_{pj} \cdot y_{pi}} \rightarrow \text{erro na camada de saída}$$

onde $-\delta_{pj} = \frac{\partial E_p}{\partial v_{pj}}$

Aprendizado da rede

2. Neurônio j está na camada oculta (escondida).
   - Nesse caso, não se conhece a expressão do erro.
   - Para obtermos $\frac{\partial E_p}{\partial y_{pj}}$, usamos mais uma vez a **regra da cadeia**.

$$\frac{\partial E_p}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot \frac{\partial v_{pk}}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot \frac{\partial \left(\sum_j w_{kj} y_{pj} \right)}{\partial y_{pj}}$$

$$= \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot w_{kj} = \sum_k \left(-\delta_{pk} \cdot w_{kj} \right)$$

$$\therefore \boxed{\frac{\partial E_p}{\partial w_{ji}} = \left(\sum_k (-\delta_{pk} w_{kj}) \right) \cdot y_{pj}(1 - y_{pj}) \cdot y_{pi}}$$

erro na camada oculta

Aprendizado da rede

- **Observação:** os erros são computados no sentido *backward*. O erro foi chamado de *back-propagado* $\rightarrow$ algoritmo de aprendizado **backpropagation** (BP).

Algoritmo *Backpropagation*

- **Inicialização:** pesos iniciados com valores aleatórios e pequenos ($[-1, +1]$).
- **Treinamento - Repita:**
  - Considere um novo padrão de entrada x_i e seu respectivo vetor de saída t_i desejado do conjunto de treinamento.
  - **Repita:**
    - Apresentar o par (x_i, t_i). **(modo padrão)**
    - Calcular as saídas dos processadores, começando da primeira camada escondida até a camada de saída.
    - Calcular o erro na camada de saída.
    - Atualizar os pesos de cada processador, começando pela camada de saída, até a camada de entrada.
  - **Até que o erro quadrático médio para esse padrão seja** $<= tol1$.
- **Até que o erro quadrático médio seja** $<= tol2$ **para todos os padrões do conjunto de treinamento.**

Sumário

Processo de aprendizado

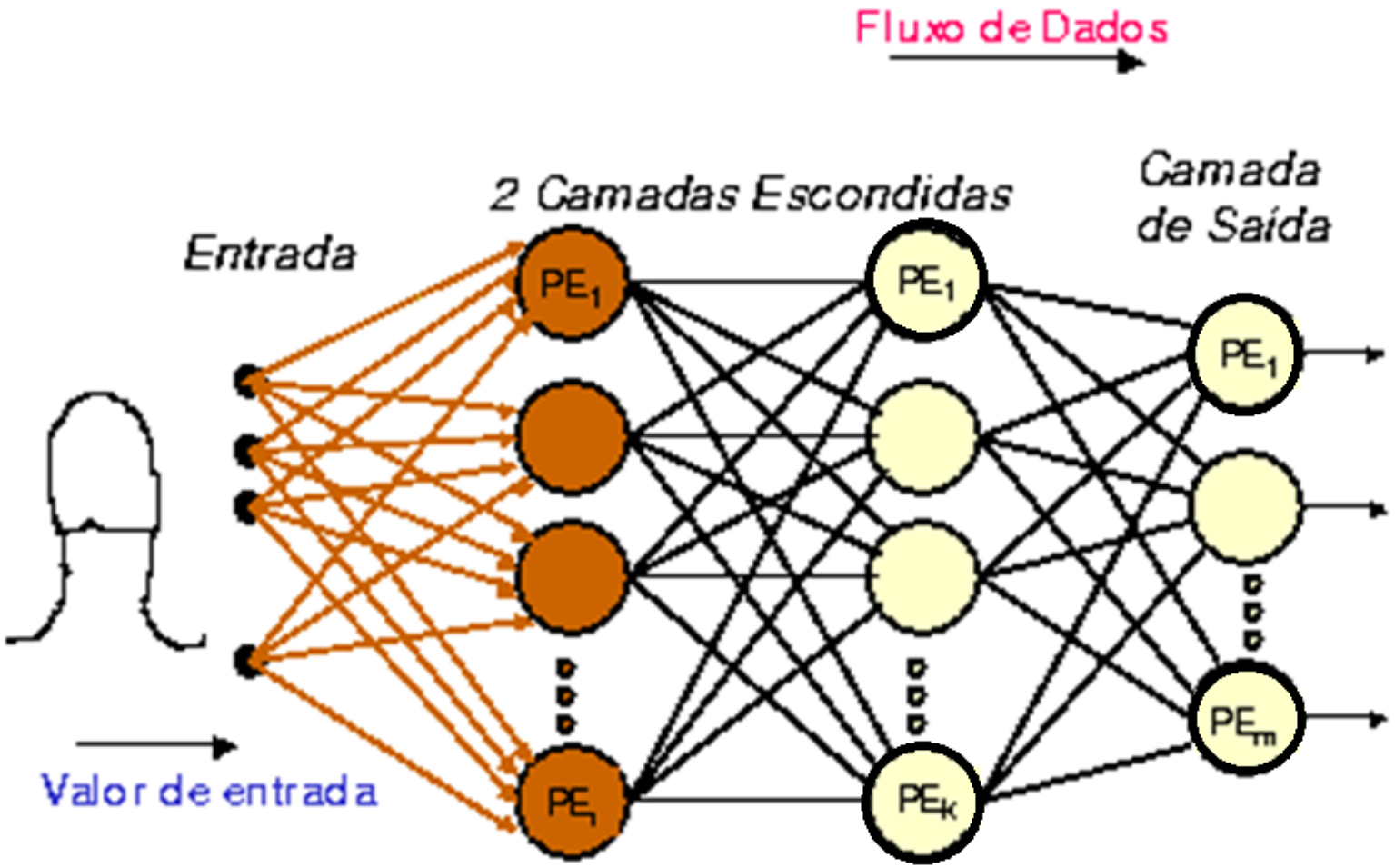


Figura 2: *Feed-forward* (fase 1), primeira camada escondida.

Processo de aprendizado

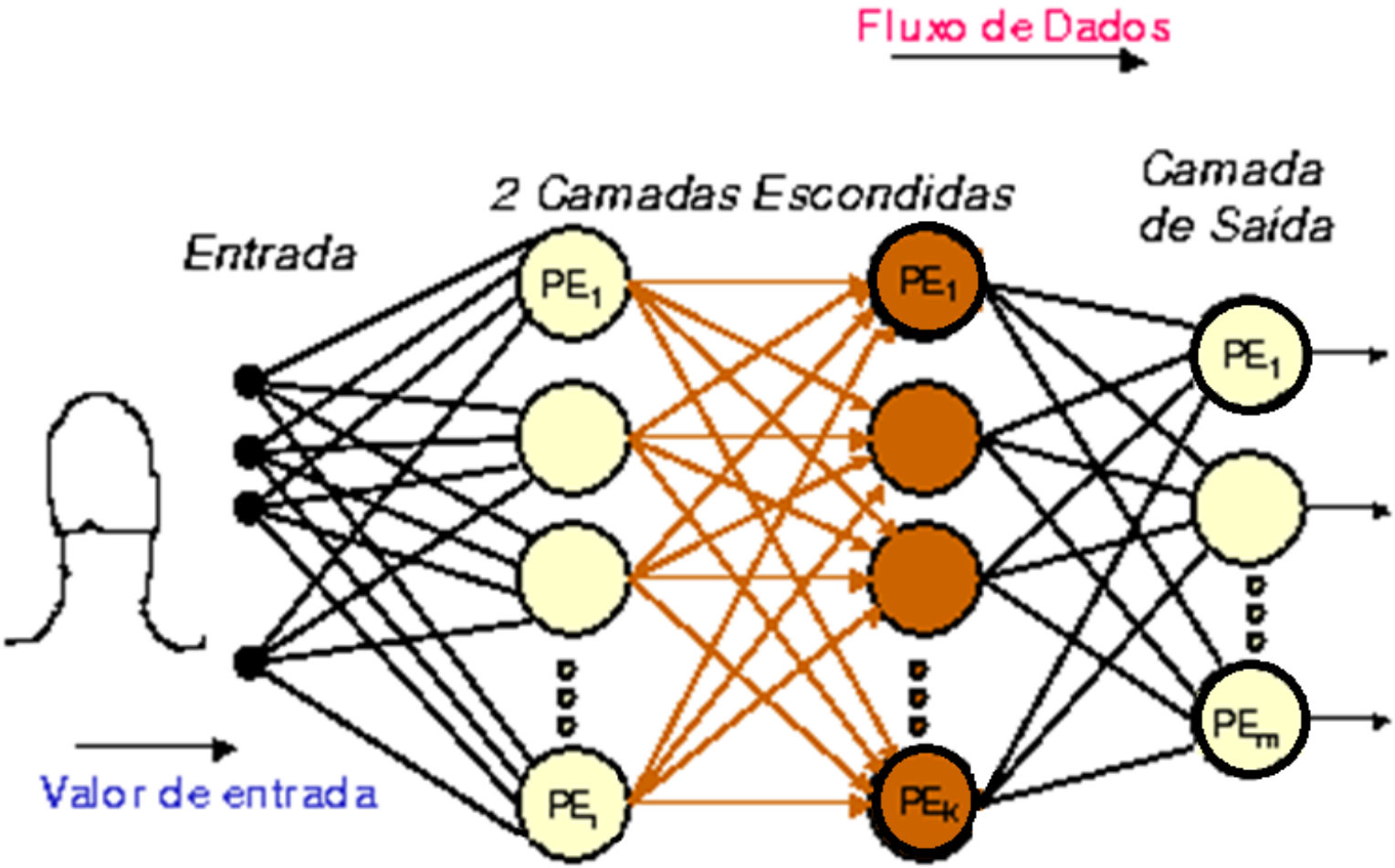


Figura 3: *Feed-forward* (fase 1), segunda camada escondida.

Processo de aprendizado

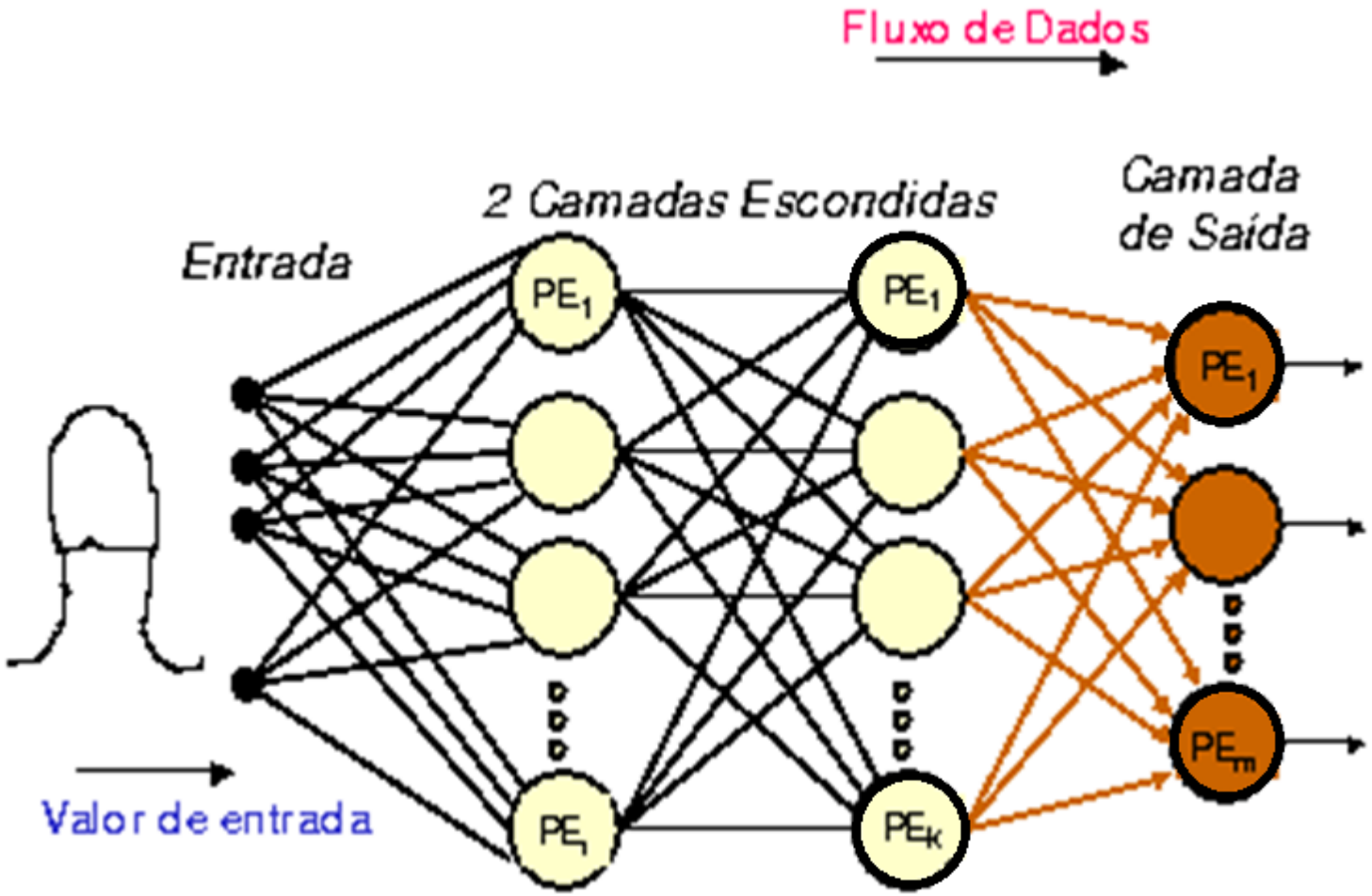


Figura 4: *Feed-forward* (fase 1), camada de saída.

Processo de aprendizado

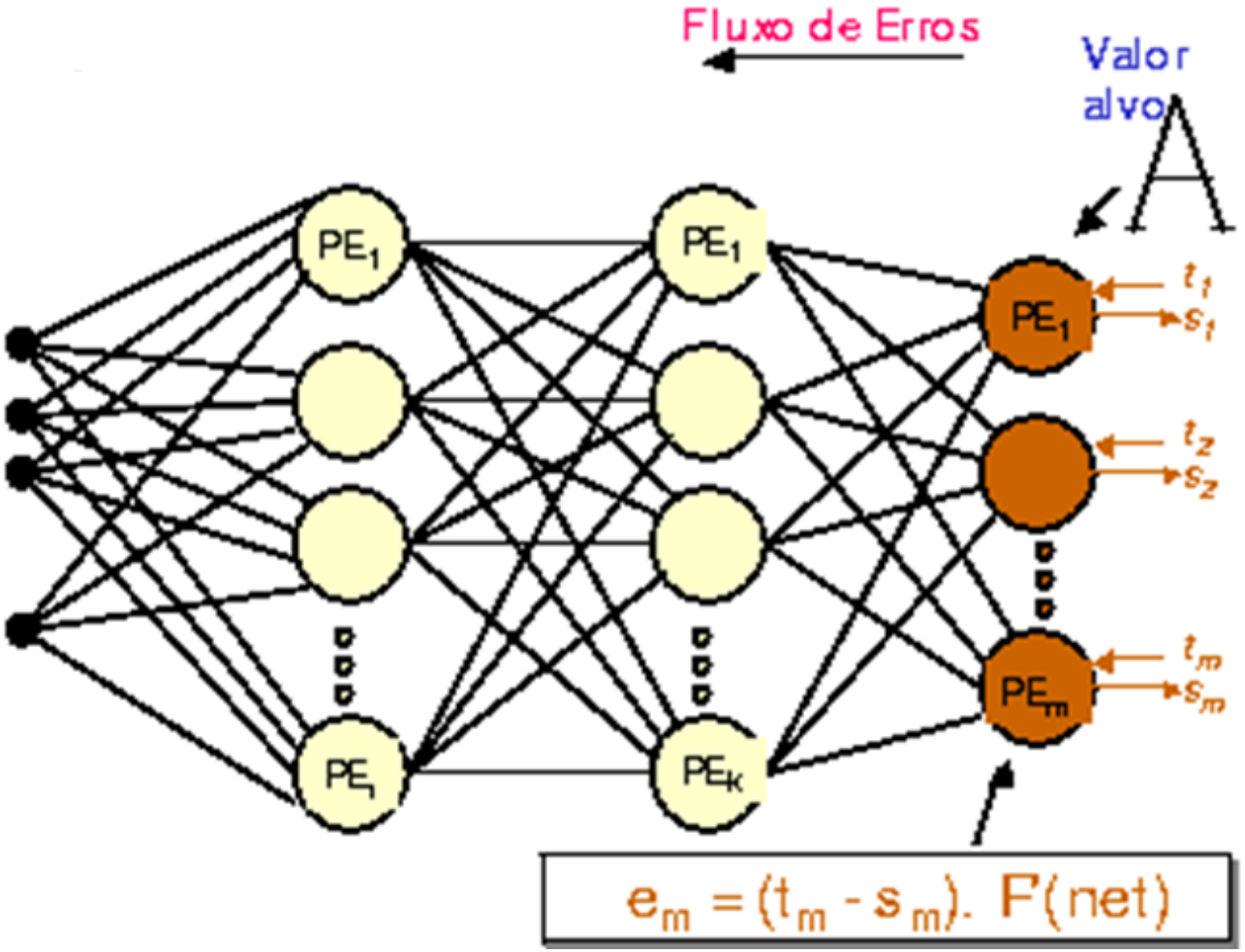


Figura 5: *Feed-backward* (fase 2), cálculo do erro da camada de saída.

Processo de aprendizado

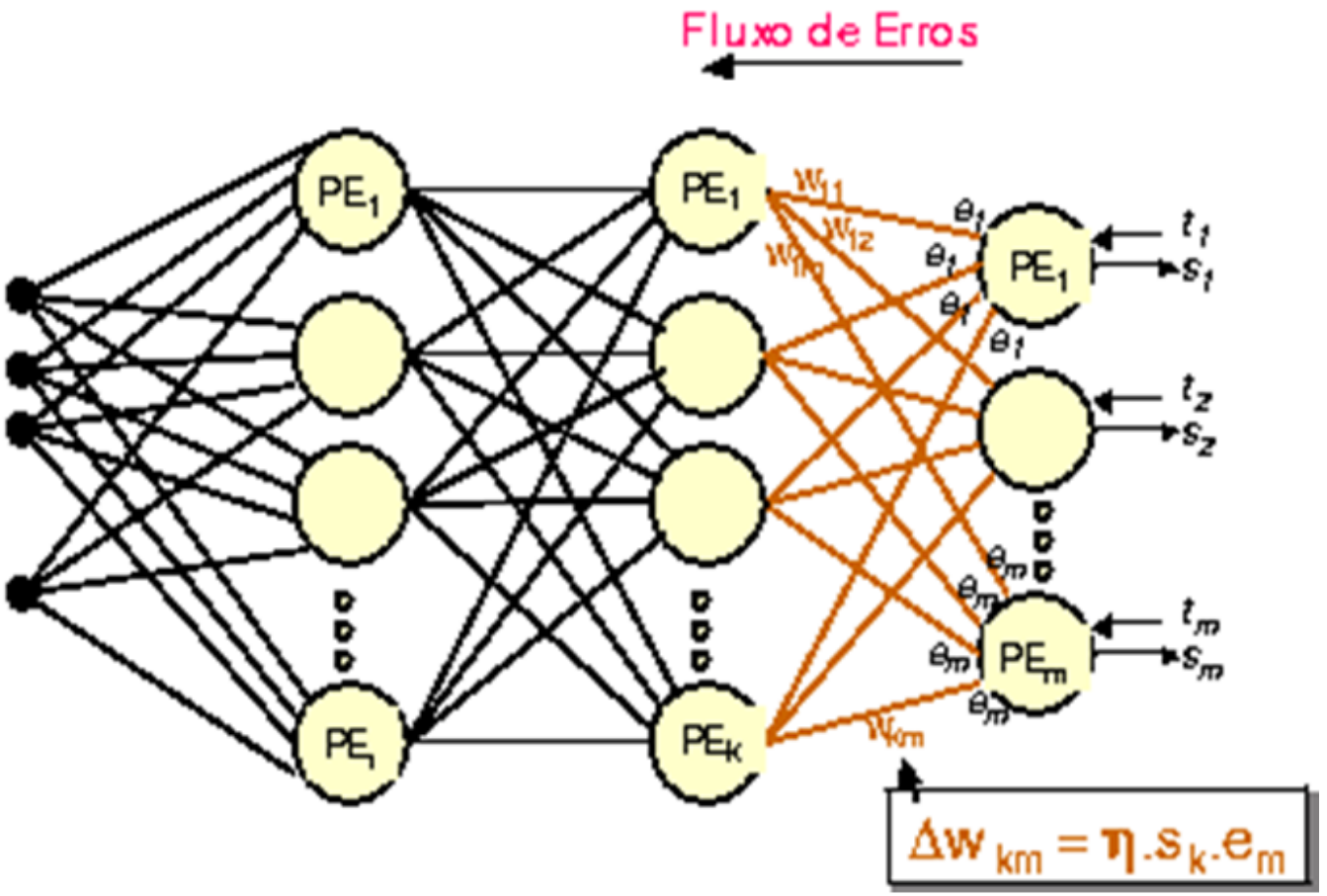


Figura 6: *Feed-backward* (fase 2), atualização dos pesos da camada de saída.

Processo de aprendizado

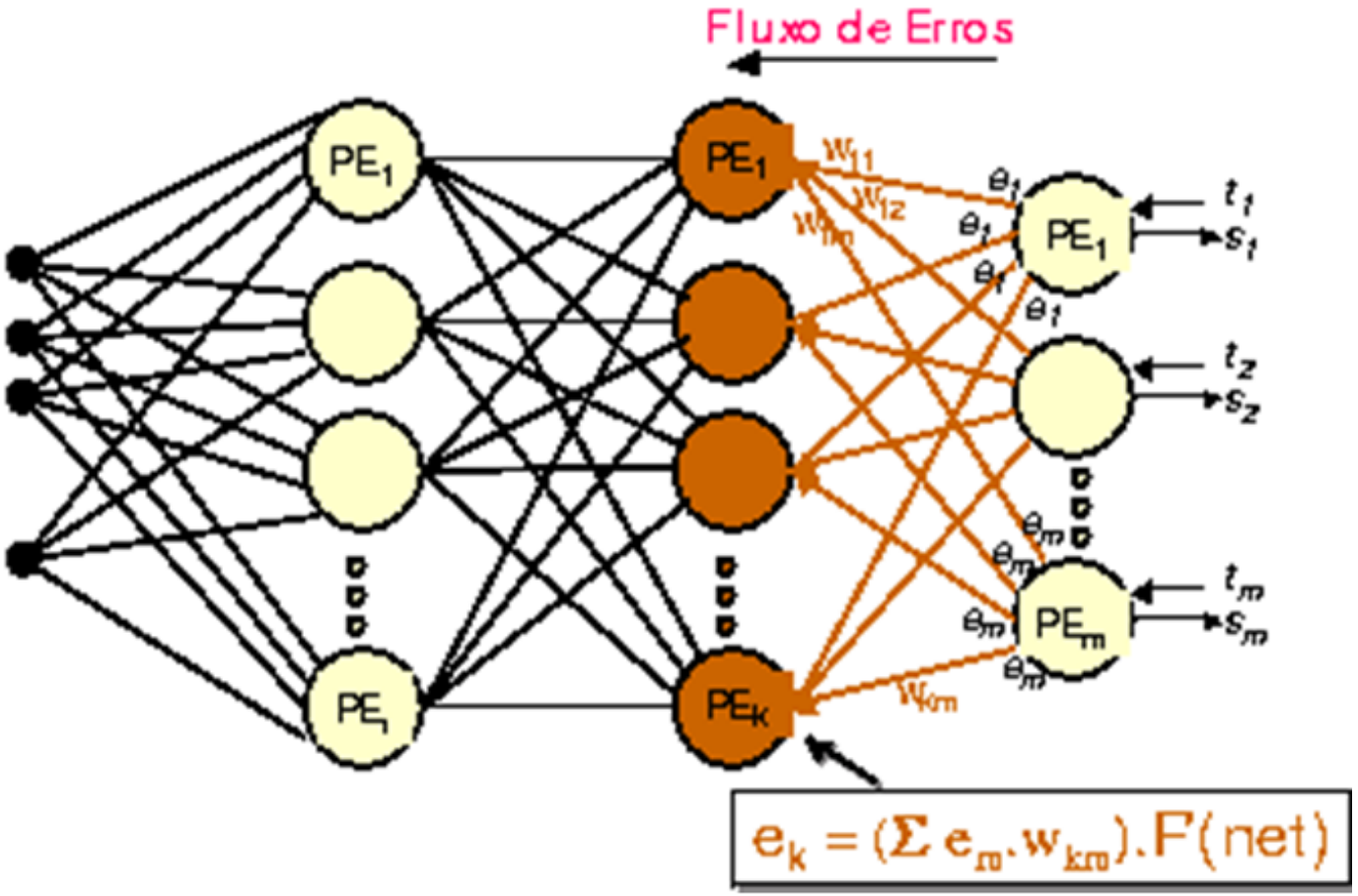


Figura 7: *Feed-backward* (fase 2), cálculo do erro da segunda camada escondida.

Processo de aprendizado

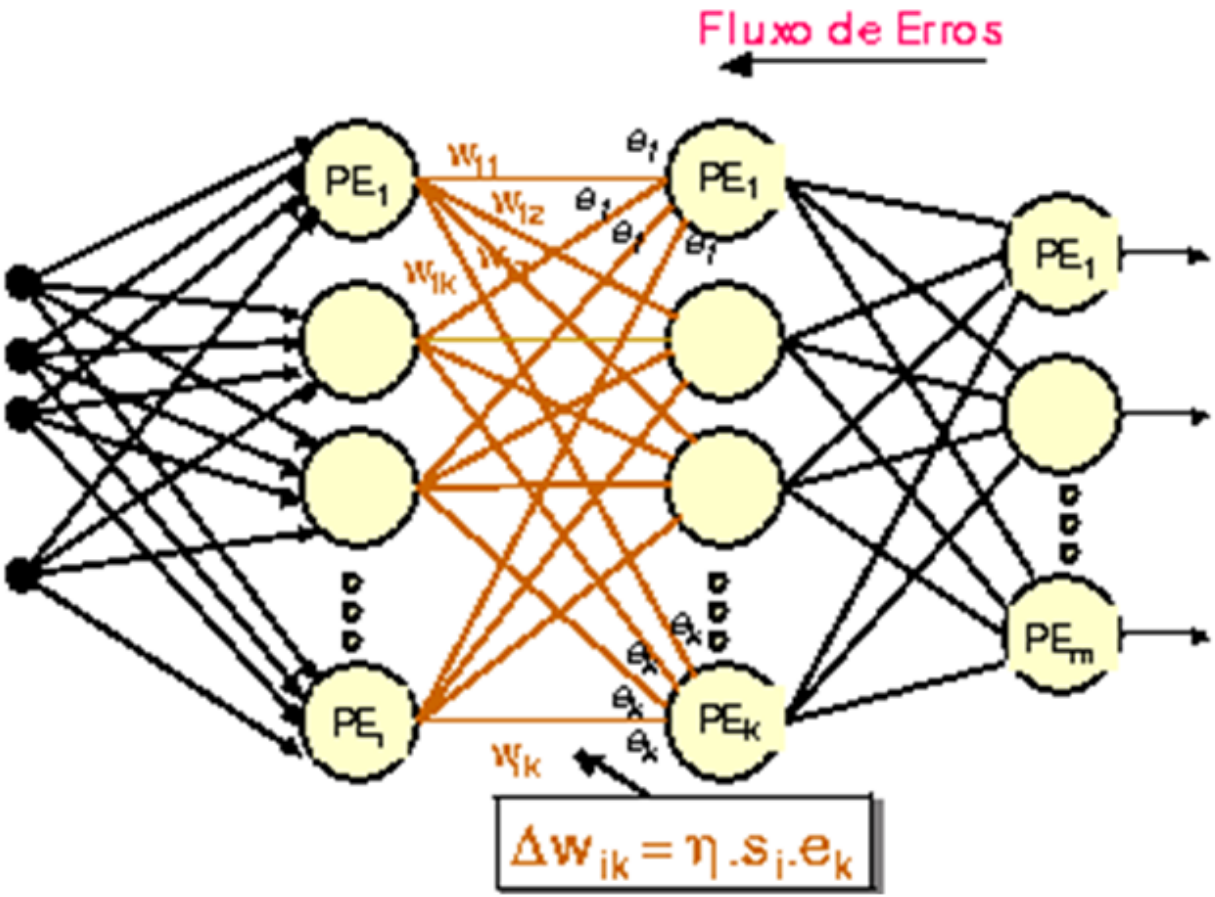


Figura 8: *Feed-backward* (fase 2), atualização dos pesos da segunda camada escondida.

Processo de aprendizado

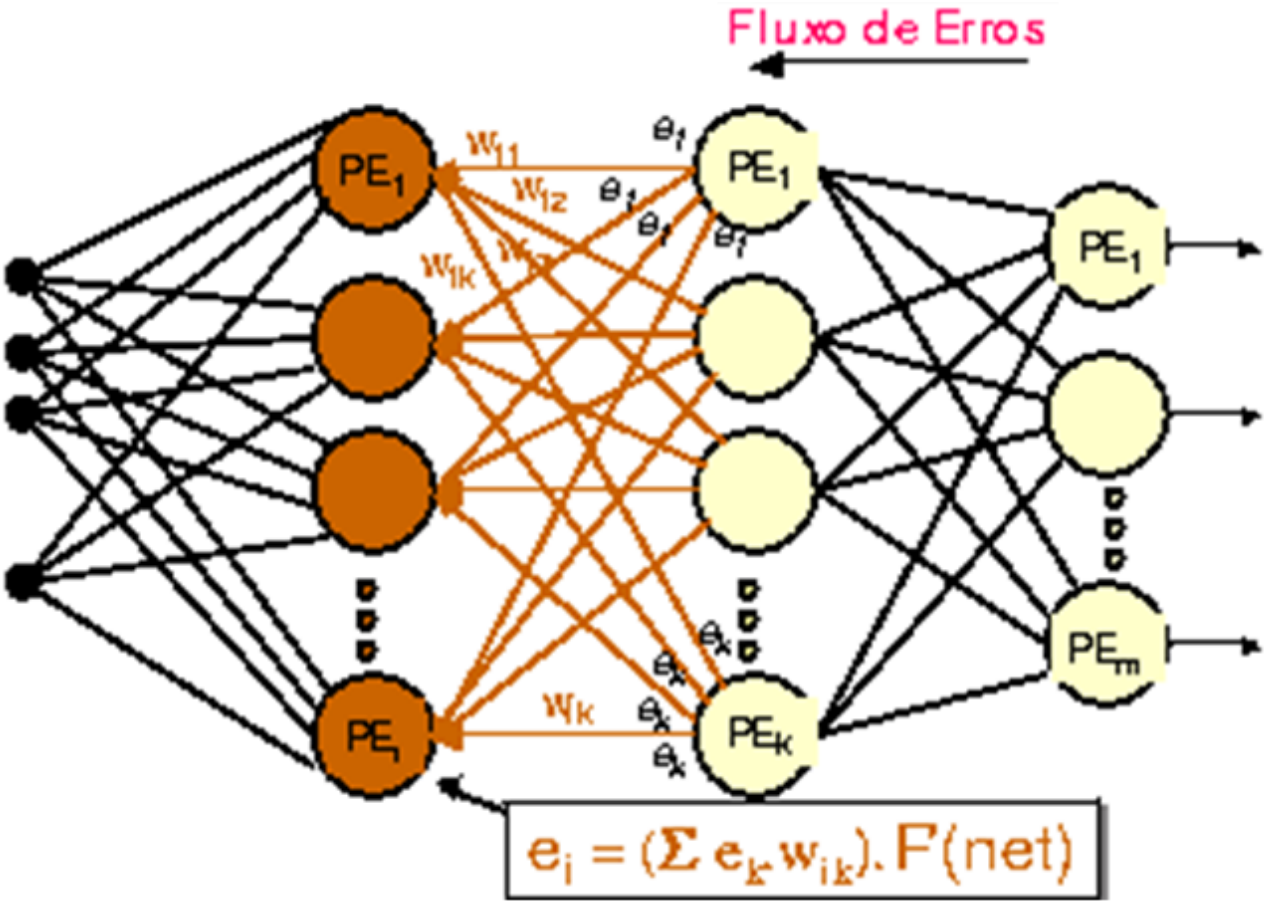


Figura 9: *Feed-backward* (fase 2), cálculo do erro da primeira camada escondida.

Processo de aprendizado

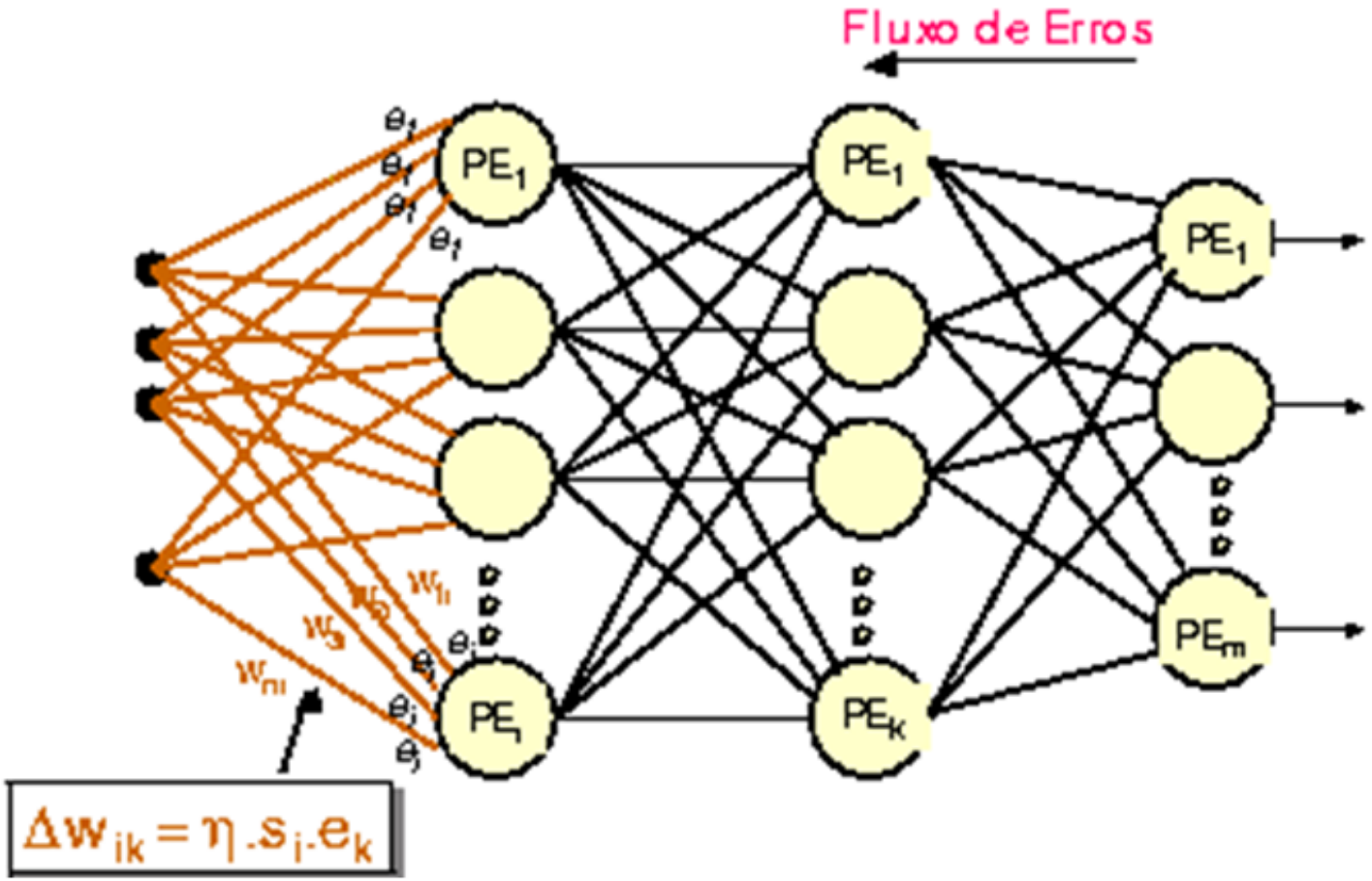


Figura 10: *Feed-backward* (fase 2), atualização dos pesos da primeira camada escondida.

Processo de aprendizado

- Este procedimento de aprendizado é repetido diversas vezes, até que, **para todos processadores de camada de saída e para todos padrões de treinamento**, o erro seja menor do que o especificado.

Sumário

Exemplo - XOR

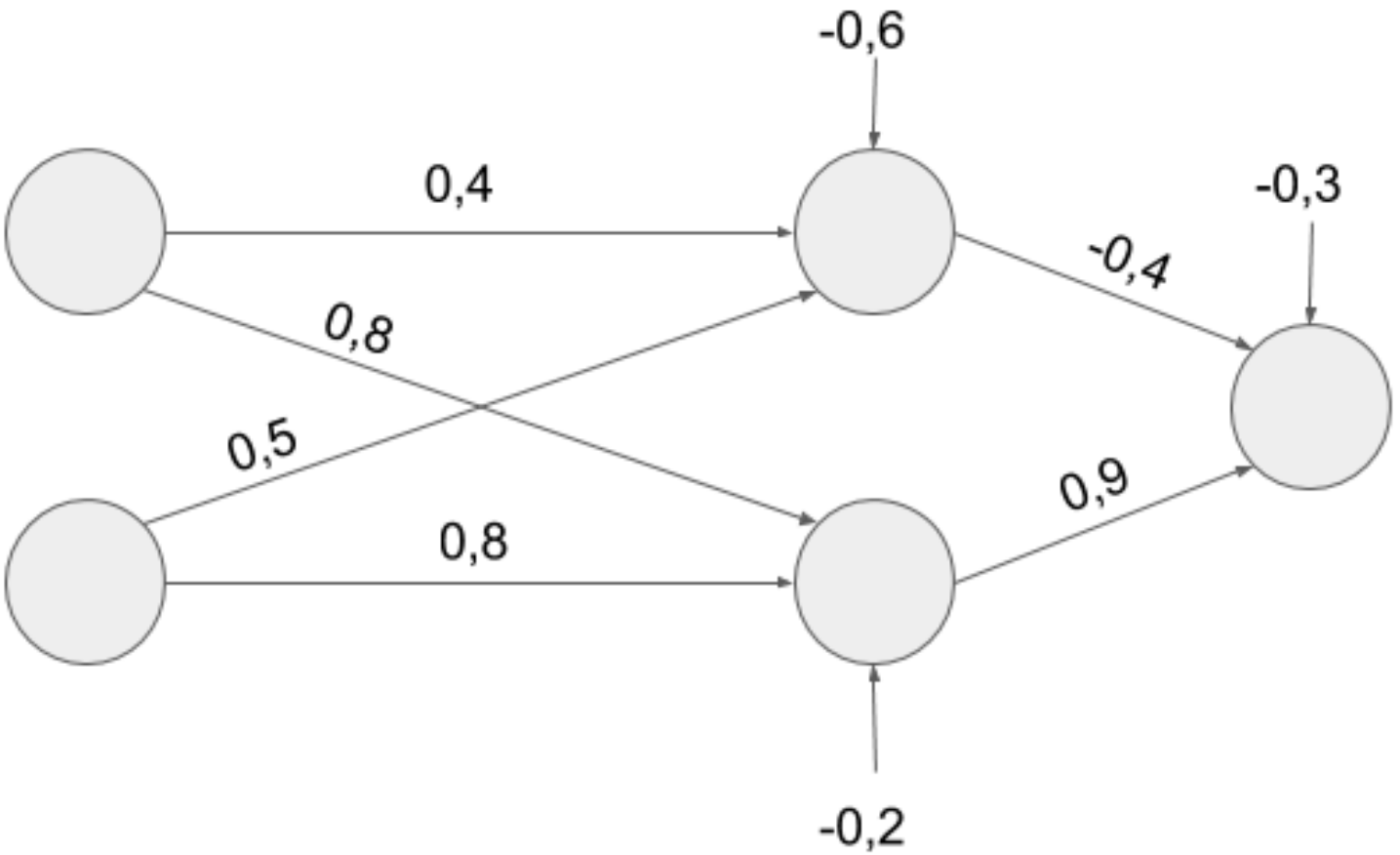


Figura 11: Rede neural inicial. Atualizar os pesos.

Exemplo - XOR

Taxa aprendizado	0,5					
	t	0	1	2	3	4
Entrada	x_1	1	0	0	1	
	x_2	1	0	1	0	
Saída desejada	y	0	0	1	1	
Pesos	$w_{\theta 1}^{h1}$	-0,6				
	w_{11}^{h1}	0,4				
	w_{21}^{h1}	0,5				
	$w_{\theta 2}^{h1}$	-0,2				
	w_{12}^{h1}	0,8				
	w_{22}^{h1}	0,8				
	$w_{\theta 1}^{out}$	-0,3				
	w_{11}^{out}	-0,4				
	w_{21}^{out}	0,9				
Camada h_1	$v_1^{h1}(x)$					
	$v_2^{h1}(x)$					
	$f[v_1^{h1}(x)]$					
	$f[v_2^{h1}(x)]$					
Camada *out* (saída)	v_1^{out}					
	$y' = f[v_1^{out}]$					

Camada oculta h1 - forward

$$v_1^{h1}(\mathbf{x}_{t=0}) = 1 \cdot w_{\theta 1}^{h1}(0) + x_1(0) \cdot w_{11}^{h1}(0) + x_2(0) \cdot w_{21}^{h1}(0)$$
$$= 1 \cdot -0.6 + 1 \cdot 0.4 + 1 \cdot 0.5 = \mathbf{0.3}$$

$$v_2^{h1}(\mathbf{x}_{t=0}) = 1 \cdot w_{\theta}^{h1}(0) + x_1(0) \cdot w_{12}^{h1}(0) + x_2(0) \cdot w_{22}^{h1}(0)$$
$$= 1 \cdot -0.2 + 1 \cdot 0.8 + 1 \cdot 0.8 = \mathbf{1.4}$$

$$f[v_1^{h1}(\mathbf{x}_{t=0})] = \frac{1}{1 + e^{-v_1^{h1}(\mathbf{x}_{t=0})}} = \frac{1}{1 + e^{0.3}} = \mathbf{0.5744}$$

$$f[v_2^{h1}(\mathbf{x}_{t=0})] = \frac{1}{1 + e^{-v_2^{h1}(\mathbf{x}_{t=0})}} = \frac{1}{1 + e^{1.4}} = \mathbf{0.8022}$$

Camada de saída - forward

$$v_1^{out}(\mathbf{x}_{t=0}) = 1 \cdot w_{\theta 1}^{out}(0) + f[v_1^{h1}(\mathbf{x}_{t=0})] \cdot w_{11}^{out}(0) + f[v_2^{h1}(\mathbf{x}_{t=0})] \cdot w_{21}^{out}(0)$$

$$= 1 \cdot -0.3 + 0.5744 \cdot (-0.4) + 0.8022 \cdot 0.9 = \mathbf{0.1922}$$

$$y' = f[v_1^{out}(h1)](\mathbf{x}_{t=0}) = \frac{1}{1 + e^{-v_1^{out}(\mathbf{x}_{t=0})}} = \frac{1}{1 + e^{0.1922}} = \mathbf{0.5479}$$

Exemplo - XOR

Taxa aprendizado	0,5					
	t	0	1	2	3	4
Entrada	x_1	1	0	0	1	
	x_2	1	0	1	0	
Saída desejada	y	0	0	1	1	
Pesos	$w_{\theta 1}^{h1}$	-0,6				
	w_{11}^{h1}	0,4				
	w_{21}^{h1}	0,5				
	$w_{\theta 2}^{h1}$	-0,2				
	w_{12}^{h1}	0,8				
	w_{22}^{h1}	0,8				
	$w_{\theta 1}^{out}$	-0,3				
	w_{11}^{out}	-0,4				
	w_{21}^{out}	0,9				
Camada h_1	$v_1^{h1}(x)$	0,3				
	$v_2^{h1}(x)$	1,4				
	$f[v_1^{h1}(x)]$	0,5744				
	$f[v_2^{h1}(x)]$	0,8022				
Camada *out* (saída)	v_1^{out}	0,1922				
	$y' = f[v_1^{out}]$	0,5479				

Backpropagation

Camada de saída

$$w_{ji}(t) = w_{ji}(t-1) - \eta \cdot (t_{pj} - y_{pj}) \cdot y_{pj}(1 - y_{pj}) \cdot y_{pi}$$

$$w_{\theta 1}^{out}(t=1) = -0.3 + \overbrace{0.5}^{\eta} \cdot \overbrace{(0 - 0.5479) \cdot 0.5479(1 - 0.5479)}^{-\delta_{pj} = -0.1357} \cdot 1 = \mathbf{-0.3679}$$

$$w_{11}^{out}(t=1) = -0.4 + 0.5 \cdot (-0.1357) \cdot \overbrace{0.5744}^{y_{pi} = f[v_1^{h1}(x)]} = \mathbf{-0.4390}$$

$$w_{21}^{out}(t=1) = 0.9 + 0.5 \cdot (-0.1357) \cdot \overbrace{0.8022}^{y_{pi} = f[v_2^{h1}(x)]} = \mathbf{0,8456}$$

Exemplo - XOR

Taxa aprendizado	0,5					
	t	0	1	2	3	4
Entrada	x_1	1	0	0	1	
	x_2	1	0	1	0	
Saída desejada	y	0	0	1	1	
Pesos	$w_{\theta 1}^{h1}$	-0,6				
	w_{11}^{h1}	0,4				
	w_{21}^{h1}	0,5				
	$w_{\theta 2}^{h1}$	-0,2				
	w_{12}^{h1}	0,8				
	w_{22}^{h1}	0,8				
	$w_{\theta 1}^{out}$	-0,3	-0.3679			
	w_{11}^{out}	-0,4	-0.4390			
	w_{21}^{out}	0,9	0,8456			
Camada h_1	$v_1^{h1}(x)$	0,3				
	$v_2^{h1}(x)$	1,4				
	$f[v_1^{h1}(x)]$	0,5744				
	$f[v_2^{h1}(x)]$	0,8022				
Camada *out* (saída)	v_1^{out}	0,1922				
	$y' = f[v_1^{out}]$	0,5479				

Backpropagation

Camada oculta

$$w_{ji}(t) = w_{ji}(t-1) - \eta \cdot \left(\sum_k (-\delta_{pk} w_{kj}) \right) \cdot y_{pj}(1 - y_{pj}) \cdot y_{pi}$$

$$w_{\theta 1}^{h1} = -0.6 + 0.5 \cdot \overbrace{(-0.1357)}^{-\delta_{pj}} \cdot \overbrace{(-0.4)}^{w_{11}^{out}(t=0)} \cdot \overbrace{0.5744}^{y_{pj}=f[v_1^{h1}(x)]} \cdot (1-0.5744) \cdot 1 = \mathbf{-0.5934}$$

$$w_{11}^{h1} = 0.4 + 0.5 \cdot (-0.1357) \cdot (-0.4) \cdot 0.5744 \cdot (1-0.5744) \cdot \overbrace{1}^{y_{pi}=x_1} = \mathbf{0.4066}$$

$$w_{21}^{h1} = 0.5 + 0.5 \cdot (-0.1357) \cdot (-0.4) \cdot 0.5744 \cdot (1-0.5744) \cdot \overbrace{1}^{y_{pi}=x_2} = \mathbf{0.5066}$$

Backpropagation

$$w_{\theta 2}^{h1} = -0.2 + 0.5 \cdot (-0.1357) \cdot \overbrace{0.9}^{w_{12}^{out}(t=0)} \cdot \overbrace{0.8022}^{y_{pj}=f[v_2^{h1}(x)]} \cdot (1-0.8022) \cdot 1 = \mathbf{-0,2097}$$

$$w_{12}^{h1} = 0.8 + 0.5 \cdot (-0.1357) \cdot 0.9 \cdot 0.8022 \cdot (1 - 0.8022) \cdot 1 = \mathbf{0,7903}$$

$$w_{22}^{h1} = 0.8 + 0.5 \cdot (-0.1357) \cdot 0.9 \cdot 0.8022 \cdot (1 - 0.8022) \cdot 1 = \mathbf{0,7903}$$

Exemplo - XOR

Taxa aprendizado	0,5					
	t	0	1	2	3	4
Entrada	x_1	1	0	0	1	
	x_2	1	0	1	0	
Saída desejada	y	0	0	1	1	
Pesos	$w_{\theta 1}^{h1}$	-0,6	-0,5934			
	w_{11}^{h1}	0,4	0,4066			
	w_{21}^{h1}	0,5	0,5066			
	$w_{\theta 2}^{h1}$	-0,2	-0,2097			
	w_{12}^{h1}	0,8	0,7903			
	w_{22}^{h1}	0,8	0,7903			
	$w_{\theta 1}^{out}$	-0,3	-0.3679			
	w_{11}^{out}	-0,4	-0.4390			
	w_{21}^{out}	0,9	0,8456			
Camada h_1	$v_1^{h1}(x)$	0,3				
	$v_2^{h1}(x)$	1,4				
	$f[v_1^{h1}(x)]$	0,5744				
	$f[v_2^{h1}(x)]$	0,8022				
Camada *out* (saída)	v_1^{out}	0,1922				
	$y' = f[v_1^{out}]$	0,5479				

Backpropagation

- Completa-se uma **época** ao se atualizarem todos os exemplos de treinamento uma vez.
  - $(0, 0) \rightarrow 0$
  - $(0, 1) \rightarrow 1$
  - $(1, 0) \rightarrow 1$

Exemplo - XOR

Taxa aprendizado	0,5					
	t	0	1	2	3	4
Entrada	x_1	1	0	0	1	
	x_2	1	0	1	0	
Saída desejada	y	0	0	1	1	
Pesos	$w_{\theta 1}^{h1}$	-0,6	-0,5934	-0,5876	-0,5951	-0,6018
	w_{11}^{h1}	0,4	0,4066	0,4066	0,4066	0,4000
	w_{21}^{h1}	0,5	0,5066	0,5066	0,4991	0,4991
	$w_{\theta 2}^{h1}$	-0,2	-0,2097	-0,2217	-0,2092	-0,1968
	w_{12}^{h1}	0,8	0,7903	0,7903	0,7903	0,8027
	w_{22}^{h1}	0,8	0,7903	0,7903	0,8028	0,8028
	$w_{\theta 1}^{out}$	-0,3	-0,3679	-0,4255	-0,3594	-0,2969
	w_{11}^{out}	-0,4	-0,4390	-0,4595	-0,4278	-0,3995
	w_{21}^{out}	0,9	0,8456	0,8197	0,8619	0,9020
Camada h_1	$v_1^{h1}(x)$	0,3	-0,5934	-0,0809	-0,1885	
	$v_2^{h1}(x)$	1,4	-0,2097	0,5686	0,5811	
	$f[v_1^{h1}(x)]$	0,5744	0,3559	0,4798	0,4530	
	$f[v_2^{h1}(x)]$	0,8022	0,4478	0,6384	0,6413	
Camada *out* (saída)	v_1^{out}	0,1922	-0,1455	-0,1226	-0,0005	
	$y' = f[v_1^{out}]$	0,5479	0,4637	0,4694	0,4999	

Sumário

Teorema da Aproimação Universal (TAU)

- Qual é o número mínimo de camadas em uma MLP que fornece uma aproximação para qualquer mapeamento contínuo?
- Cybenko [1989] mostrou pela primeira vez que uma rede com **uma única camada intermediária** é suficiente para aproximar uniformemente qualquer função contínua definida em um hipercubo unitário.

Teorema da Aproximação Universal (TAU)

- **Teorema:** seja $g(\cdot)$ uma função contínua limitada estritamente crescente. Seja I_p um hipercubo unitário p-dimensional e $C(I_p)$ o espaço das funções contínuas em I_p. Então, dada qualquer função $f \in C(I_p)$ e $\epsilon > 0$, existe um inteiro M e constantes reais α_i, θ_i e w_{ji}, onde $i = 1, 2, \cdots, M$ e $j = 1, 2, \cdots, p$, tal que se pode definir:

$$F(x_1, \cdots, x_p) = \sum \alpha_i g\left(\sum w_{ji} x_j - \theta_i\right) \qquad (1)$$

com

$$|F(x_1, \cdots, x_p) - f(x_1, \cdots, x_p)| < \epsilon \qquad \{x1, \cdots, x_p\} \in I_p$$

Teorema da Aproimação Universal (TAU)

- As funções *sigmoid* ou logística são contínuas, estritamente crescentes e limitadas, portanto satisfazem as condições impostas para a função $g(\cdot)$.
- A equação (1) representa a saída da MLP.
  - A rede tem p nós de entrada e uma única camada intermediária de M nós.
- O neurônio i tem pesos $w_{1i}, \cdots, w_{pi}$ e limiar θ_i.
- A saída da rede é uma combinação linear das saídas dos neurônios intermediários com α_i.

Teorema da Aproimação Universal (TAU)

- Trata-se de um teorema de **existência**, visto que fornece uma justificativa para a aproximação de funções contínuas. $\Longrightarrow$ SUFICIENTE
- Entretanto, ele não afirma que uma única camada é um número **ótimo**.
- Na prática, nem sempre se dispõe de uma função contínua e nem de uma camada intermediária de tamanho qualquer.
- Chester [1990] e Funahashi [1989] defendem o uso de duas camadas intermediárias, tornando a aproximação mais maleável.

Teorema da Aproimação Universal (TAU)

- Características locais são extraídas na primeira camada.
  - Alguns neurônios na primeira camada são usados para particionar o espaço em várias regiões, e outros aprendem as características locais daquelas regiões.
- Características globais são extraídas na segunda camada.
  - Um neurônio na segunda camada combina as saídas de neurônios da primeira que estão operando numa região particular do espaço de entrada e assim aprende características globais daquela região.

Sumário

Sumário

Velocidade de aprendizado

- O algoritmo BP fornece uma aproximação para a trajetória no espaço dos pesos.
- Quanto menor o valor de η, menores as mudanças nos pesos e mais suave será a trajetória.
  - **Aprendizado lento.**
- Se η é muito grande, o aprendizado torna-se rápido, porém a rede pode tornar-se **instável**.

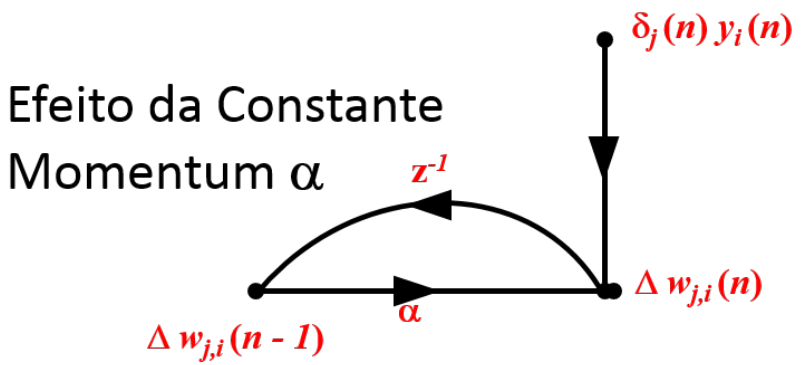
Sumário

Efeito da constante α

- É um método simples de aumentar a velocidade do aprendizado e evitar o perigo de instabilidade, como mostrado por Rumelhart *et al.*, 1986.

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \qquad (2)$$

- Onde α é geralmente um número positivo chamado **constante momentum**.



A equação (α) é chamada *REGRA DELTA GENERALIZADA*. Se α =0 $\Rightarrow$ REGRA DELTA

Efeito da constante α

- Vamos considerar uma série de tempo com índice t (de 0 a n).
- A equação 2 pode ser vista como uma equação diferencial de primeira ordem em relação a $\Delta w_{ji}(n)$. Resolvendo:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) \tag{3}$$

- Que representa uma série de tempo comprimido $n+1$. Mas:

$$\delta_j(n) y_i(n) = -\frac{\partial E(n)}{\partial w_{ji}(n)}$$

$$\therefore \Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)} \tag{4}$$

Efeito da constante α

1. O ajuste atual $\Delta w_{ji}(n)$ representa a soma de uma série temporal ponderada exponencialmente convergente $\Longrightarrow$ $0 \leq |\alpha| < 1$
2. Quando $\frac{\partial E(t)}{\partial w_{ji}(t)}$ tem o mesmo sinal algébrico em iterações consecutivas, então a série cresce em magnitude e os pesos são ajustados por uma quantidade grande. Portanto, o BP tende a acelerar a "descida"nas regiões de descida da superfície do erro.
3. Quando $\frac{\partial E(t)}{\partial w_{ji}(t)}$ tem sinais opostos em iterações sucessivas, então a série diminui em magnitude, e $\Delta w_{ji}(n)$ é atualizado por uma quantidade pequena. Então, a inclusão do termo *momentum* tem o **efeito de estabilização** nas direções em que o sinal oscila.

Efeito da constante α

- Portanto, o termo *momentum* pode ter efeitos benéficos no comportamento do aprendizado do algoritmo. Ele pode evitar que o processo termine em um mínimo local na superfície do erro.
- **Observação:** o parâmetro η foi considerado constante.
  1. η_{ji} **dependente da conexão**: fatos interessantes ocorrem se η_{ji} é tomado diferente em diferentes partes do algoritmo.
  2. **Restringir o número de pesos a serem ajustados**: $\eta_{ji} = 0$ para o peso w_{ji}.

Efeito da constante α

- Modo segundo o qual as camadas ocultas são interconectadas: no procedimento, supomos que cada camada recebe entradas apenas das unidades da **camada anterior**.
- Não existe uma razão para isso. Se esse não for o caso, existem dois tipos de sinais de erro:
  - Um sinal de erro que resulta de uma comparação direta do sinal de saída daquele neurônio como uma resposta desejada.
  - Um sinal de erro que é passado através de outras unidades cuja ativação ele afeta.'

Sumário

Modos de treinamento

- Aprendizado BP resulta de muitas apresentações de um conjunto de treinamento de exemplos.
- Uma apresentação **completa** do conjunto de treinamento corresponde a 1 ciclo (*epoch*).
- O processo de aprendizado é repetido ciclo após ciclo, até que os pesos sinápticos e níveis *threshold* se **estabilizem**.
- Tomar os pesos em uma forma aleatória $\rightarrow$ pesquisa no espaço dos pesos **estocástica**.

Modo padrão

- **(1) Modo padrão:**
  - Atualização nos pesos é feita após a apresentação de cada exemplo de treinamento.
  - Um ciclo consistindo de N exemplos de treinamento, arranjados na ordem:

$$\{[\mathbf{x_1}, \mathbf{d_1}], [\mathbf{x_2}, \mathbf{d_2}], \cdots, [\mathbf{x_N}, \mathbf{d_N}]\}$$

  - $[\mathbf{x_1}, \mathbf{d_1}] \rightarrow$ cálculos*forward/backward* e atualização dos pesos.
  - $[\mathbf{x_2}, \mathbf{d_2}] \rightarrow$ cálculos*forward/backward* e atualização dos pesos.

$$\vdots$$

  - $[\mathbf{x_N}, \mathbf{d_N}] \rightarrow$ cálculos*forward/backward* e atualização dos pesos.

Modo padrão

- Dessa forma, a variação média nas mudanças dos pesos é:

$$\Delta\hat{w}_{ji} = \frac{1}{N}\sum_{n=1}^{N}\Delta w_{ji}(n)$$

$$= -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial E(n)}{\partial w_{ji}(n)} \implies$$

$$\boxed{\Delta\hat{w}_{ji} = -\frac{\eta}{N}\sum_{n=1}^{N} e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}(n)}} \tag{5}$$

Modo *batch*

- **(2) Modo batch**:
  - Atualização dos pesos é feita depois da apresentação de todos os exemplos de treinamento que constituem um ciclo.
  - Para um ciclo particular, função custo com o erro quadrático médio:

$$\mathcal{E}_{av} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n) \tag{6}$$

  - Onde C denota o conjunto de índices correspondentes aos neurônios da camada de saída e e_j é o sinal do erro do neurônio j correspondente ao exemplo de treinamento w.

$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{E}_{av}}{\partial w_{ji}} \implies$$

$$\boxed{\Delta w_{ji} = \frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}} \tag{7}$$

Modos de treinamento - comparação

- Claramente, $\Delta\hat{w}_{ji}$ é diferente de Δw_{ji}.
  - $\Delta\hat{w}_{ji}$ representa uma **estimativa** de Δw_{ji}.
- Do ponto de vista *online*, o **modo padrão** é preferido. Além disso, os exemplos de treinamento são aleatoriamente apresentados (atualização nos pesos é **estocástica**) $\rightarrow$ menos provável o algoritmo BP estacionar em um mínimo local.
- Por outro lado, o **modo batch** fornece uma estimativa mais precisa do **vetor gradiente**.
- De qualquer forma, a eficiência dos dois modos depende do problema que se tem em mãos (Hertz, 1991).

Sumário

Critério de parada

- Não se pode, em geral, mostrar a convergência do algoritmo BP, tampouco existem critérios bem definidos para encerrar seu processamento.
- Para formular um critério, devem-se considerar propriedades de mínimo local ou global da superfície de erro.

Critério de parada

- Seja $\mathbf{w}^*$ o vetor mínimo local ou global.

1. Uma condição necessária para $\mathbf{w}^*$ ser mínimo:
   - O gradiente (derivada de primeira ordem) da superfície de erro em relação a $\mathbf{w}$ seja zero em $\mathbf{w} = \mathbf{w}^*$, isto é, $\nabla g(w) = 0$ em $\mathbf{w} = \mathbf{w}^*$.
   - Diz-se que o algoritmo BP convergiu se a norma do vetor gradiente é menor que um certo ϵ pequeno arbitrário.
2. Função custo $\mathcal{E}_{av}(w)$ é estacionária em $\mathbf{w} = \mathbf{w}^*$.
   - Diz-se que o algoritmo BP convergiu se a taxa de mudança no erro quadrático médio por ciclo é suficientemente pequena.
   - Tipicamente, são consideradas pequenas taxas de mudanças no erro de 0.1% a 1% ou de 0.01%.

Critério de parada

- Kramer e Sangiovanni-Vicentelli(1989) sugerem um critério de convergência:

O algoritmo BP termina no vetor peso w_{final} *quando* $\| g(w_{final}) \| \leq \varepsilon$, onde ε é suficiente pequeno, ou $|| \varepsilon_{av}(\text{ final }) || \leq \tau$ onde τ é suficiente pequeno.

Sumário

Generalização

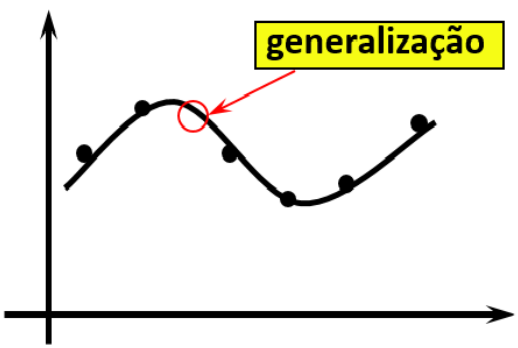
Aprendizado **BP**

conjunto treinamento + algoritmo BP ⇒ pesos sinápticos

GENERALIZAR

"GENERALIZAÇÃO": termo da psicologia.

Processo de aprendizado pode ser visto como um Método de Aproximação de Funcões



: efeito de uma boa aproximação não linear dos dados de entrada, tamanho e eficiência do conjunto treinamento, arquitetura da rede, complexidade física do problema

Complexidade da rede

- **Problema:** determinar o melhor número de nós na camada intermediária.
- Estatisticamente, esse problema é equivalente a determinar o tamanho do conjunto de parâmetros usado para modelar o conjunto de dados. Existe um limite no tamanho da rede.
- Esse limite deve ser tomado lembrando que é melhor treinar a rede para **produzir a melhor generalização** do que treinar a rede para representar perfeitamente um conjunto de dados.
- Isso pode ser feito usando **validação cruzada**.

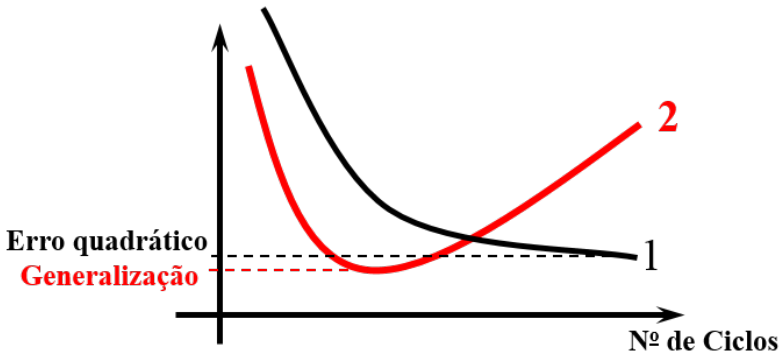
Validação cruzada

- Conjunto de dados:
  - Treinamento (75%)
  - Teste (25%)
- Conjunto de treinamento:
  - Um subconjunto para validação do modelo.
  - Um subconjunto para treinamento.
- Validar o modelo em um conjunto diferente do usado para estimá-lo.

Validação cruzada

- Usa-se o subconjunto de validação para avaliar o desempenho de diferentes candidatos do modelo (diferentes topologias) e, então, escolhe-se uma delas.
- O modelo escolhido é treinado sobre o conjunto de treinamento inteiro e a capacidade de generalização é medida no conjunto de teste.
- A validação cruzada pode ser usada para decidir quando o treinamento de uma rede deve ser encerrado.

Tamanho do conjunto de treinamento



Curva 1: poucos parâmetros (*underfitting*)

Curva 2: muitos parâmetros (*overfitting*)

- Em ambos os casos:
  1. O desempenho do erro na generalização exibe um mínimo.
  2. O mínimo no caso *overfitting* é menor e mais definido.
- Pode-se obter boa generalização se a rede é projetada com muitos neurônios, contanto que o treinamento seja cessado após um número de ciclos correspondente ao mínimo da curva do **erro** obtida na **validação cruzada**.

Sumário

Normalização

Normalização

Consiste em modificar o processo de aprendizado, de modo a favorecer soluções com determinadas propriedades em detrimento de outras.

- Principais técnicas de normalização:
  - Normalização L1
  - Normalização L2
  - *Early-stopping*

Normalização

Normalização L1/L2

Consiste em colocar um termo adicional na função de perda, punindo parametrizações com pesos mais elevados.

Normalização L1/L2

- Se $\mathcal{L}(\theta, \mathcal{D})$ é a função perda (e.g. erro quadrático médio), θ é o conjunto de parâmetros livres e $\mathcal{D}$ é um exemplo de treinamento, então a função de perda regularizada será:

$$E(\theta, \mathcal{D}) = \mathcal{L}(\theta, \mathcal{D}) + \lambda R(\theta)$$

- Ou, no nosso caso:

$$E(\theta, \mathcal{D}) = \mathcal{L}(\theta, \mathcal{D}) + \lambda ||\theta||_p$$

- Onde:

$$||\theta||_p = \left(\sum_{j=0}^{|\theta|} |\theta_j|^p \right)^{\frac{1}{p}}$$

- Que é a norma L_p de θ.

Normalização L1/L2

- α é um hiper-parâmetro (i.e. um parâmetro configurado manualmente) que controla a importância relativa do parâmetro de regularização.
- Valores comuns são $p = 1$ ou $p = 2$, daí o nome **L1/L2**.
- Adicionar um termo de regularização tende a produzir um mapeamento mais suave, uma vez que valores elevados dos parâmetros são penalizados.
- Pelo princípio da Navalha de Occam, será obtida uma função mais simples.
  - Não significa necessariamente que vai generalizar bem.
  - Empiricamente, constata-se que realizar esse tipo de normalização no contexto de redes neurais melhora a generalização, principalmente em bases de dados pequenas.

Normalização

Early-stopping

Consiste em adicionar um coeficiente de paciência e avaliar as parametrizações frequentemente em um conjunto de validação. Quando a melhoria for insignificante ou negativa, retornar a melhor parametrização encontrada até então e encerrar o treinamento.

Early-stopping

- A técnica de *early stopping* requer que particionemos os exemplos em três conjuntos: de treinamento, de validação e de teste.
- O conjunto de treinamento é usado para aplicação de gradiente descendente estocástico na aproximação diferenciável da função perda.
- Enquanto se executa o gradiente descendente, o conjunto de validação é periodicamente consultado para verificar como o modelo está se desempenhando.
- Pode ser adicionado um coeficiente de **paciência**, segundo o qual o algoritmo pára assim que não houver melhora significativa em sucessivas avaliações com o conjunto de validação.

Sumário

Inicialização

- O primeiro passo do algoritmo BP é a inicialização da rede.
- Uma boa escolha para os parâmetros livres (pesos sinápticos e *threshold*) podem contribuir significativamente no sucesso do aprendizado.

Inicialização

- **Informação disponível**
- **Nenhuma informação disponível?**
  - Pesos inicializados aleatoriamente, isto é, inicializar os pesos com valores uniformemente distribuídos em um intervalo pequeno.
- **Escolha errada $\Longrightarrow$ saturação prematura**
  - Esse fenômeno se refere a uma situação na qual o erro quadrático permanece constante por um período de tempo, porém continua a diminuir depois que este período é concluído.

Inicialização

- Fatos interessantes podem ocorrer:

1. Suponha que, para um particular padrão de treinamento, o nível de ativação interna de um neurônio saída tenha um valor cuja magnitude é grande (como a função é *sigmoid*, trata-se de um caso em que $y = 1$ ou $y = -1$). Em tal caso , diz-se que o neurônio está em **saturação**.
2. Se y está mais próximo de 1 quando a saída desejada é -1, ou vice-versa, o neurônio está **incorretamente saturado**.
   - Quando isso ocorre, o ajuste nos pesos será pequeno, embora o erro seja de magnitude grande, e a rede levará um longo tempo para corrigir essa situação (Lee,1991).
3. No estágio inicial do BP, podem existir neurônios não-saturados ou incorretamente saturados.

Inicialização

- Para os não-saturados $\rightarrow$ os pesos mudam rapidamente.
- Para os incorretamente saturados $\rightarrow$ permanecem saturados por algum tempo.
- **Fenômeno da saturação prematura** pode ocorrer, com $\mathcal{E}$ permanecendo constante.

Inicialização

- Em Lee(1991), uma fórmula para a probabilidade de **saturação prematura** foi obtida para o **modo batch**.
- A essência dessa fórmula pode ser: [Haykin, 1994]
  1. **Saturação incorreta** é evitada escolhendo valores iniciais dos pesos sinápticos e níveis *threshold*, uniformemente distribuídos em um intervalo pequeno.
  2. É menos provável quando o número de neurônios intermediários é mantido baixo.
  3. Raramente ocorre quando os neurônios da rede operam em sua regiões lineares.
- Segundo [Haykin,1994], para o **modo padrão** de atualização dos pesos, os resultados mostram uma tendência similar ao **modo batch**.