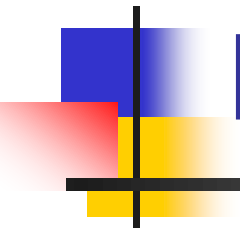
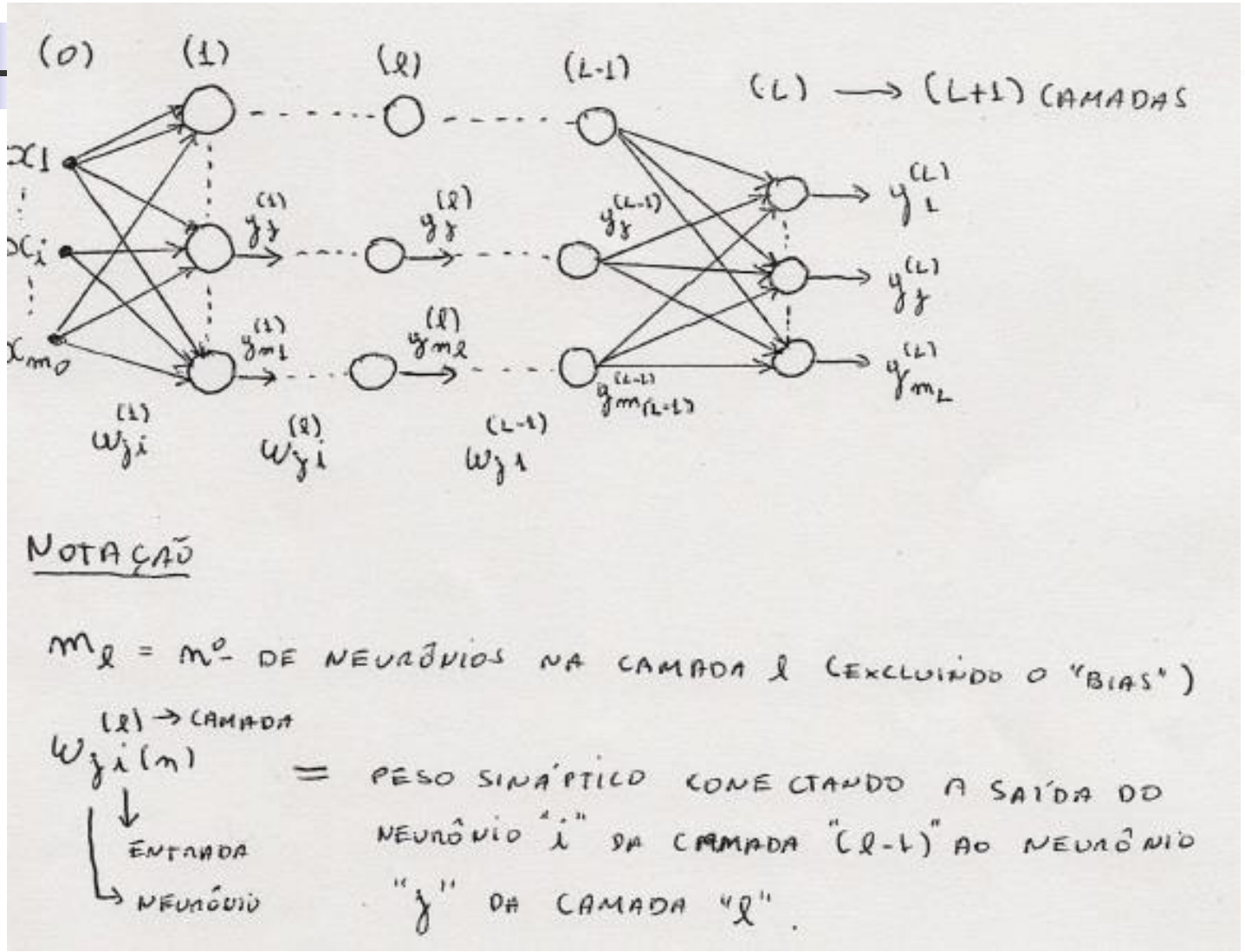


Redes Multicamadas

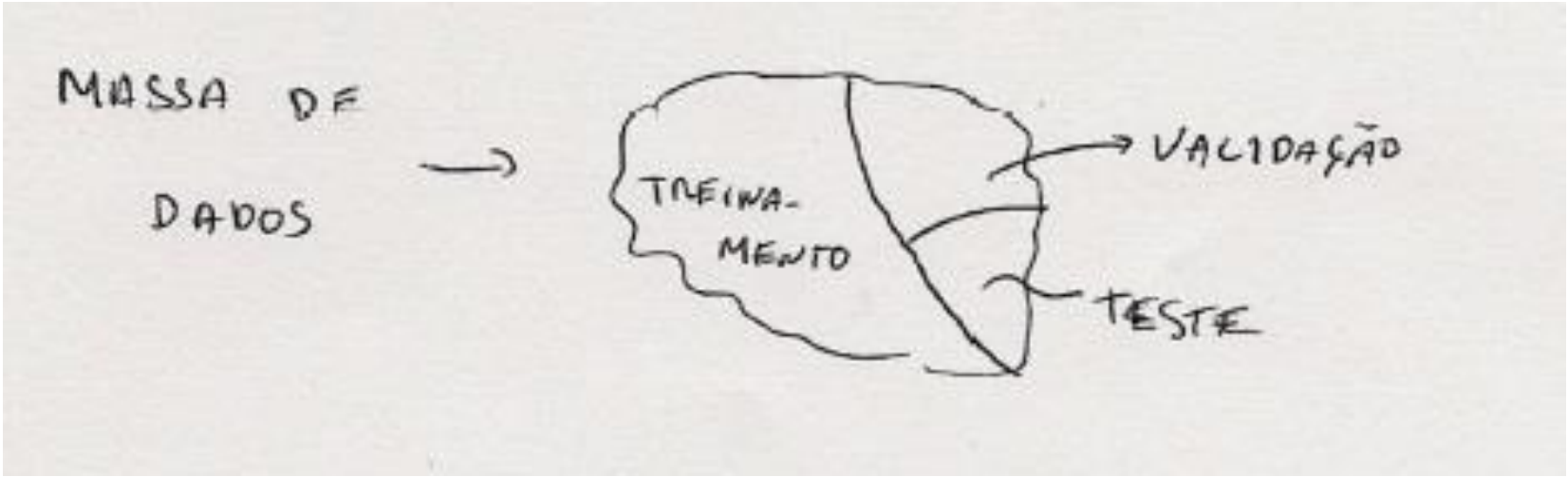
Algoritmo de Retropropagação do Erro

(*Backpropagation*)

Notação


(0)
(1)
(l)
(L-1)
(L) → (L+1) CAMADAS
NOTAÇÃO
m_l = Nº DE NEURÔNIOS NA CAMADA l (EXCLUINDO O "BIAS")
$w_{ji}^{(l)}(n)$ = PESO SINÁPTICO CONECTANDO A SAÍDA DO NEURÔNIO "i" DA CAMADA "(l-1)" AO NEURÔNIO "j" DA CAMADA "l".
(l) → CAMADA
ENTRADA
NEURÔNIO

Conjunto de dados



Algoritmo- Erros

- Erro no neurônio de saída j na iteração n (apresentação do n-ésimo exemplo de treinamento):

$$e_j^{(L)}(n) = d_j(n) - y_j^{(L)}(n) \quad , \quad j \in L$$

- Erro instantâneo total das saídas:

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j=1}^{m_L} \left[e_j^{(L)}(n) \right]^2$$

- Erro médio quadrático:

$$\mathcal{E}_{av} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}(n)$$

N = número total de padrões contidos no cons. de treinamento.

$\mathcal{E}_{av}$ é uma medida do desempenho do aprendizado
↳ função custo.

Algoritmo- objetivo

OBJETIVO: SINTONIZAR $w_{ji}^{(l)}$ PARA OBTER $\min\{E_{av}\}$

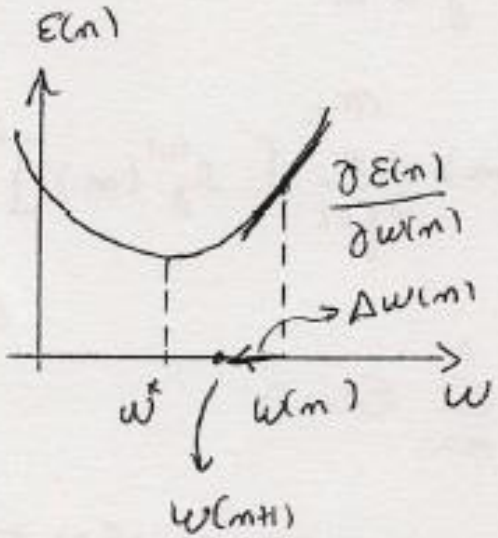
HIPÓTESE: VAMOS CONSIDERAR INICIALMENTE QUE OS PESOS $w_{ji}^{(l)}$ SERÃO ATUALIZADOS ($\Delta w_{ji}^{(l)}$) A CADA APRESENTAÇÃO DE UM PADRÃO DE TREINAMENTO (MODO "PADRÃO A PADRÃO" OU SEQÜENCIAL), ATÉ O TÉRMINO DE UMA "ÉPOCA", ISTO É, ATÉ A APRESENTAÇÃO COMPLETA DE TODOS OS PADRÕES DO CONJ. DE TREINAMENTO:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \Delta w_{ji}^{(l)}(n)$$

$\Delta w_{ji}^{(l)}(n)$ → ATUALIZADOS DE MANEIRA A MINIMIZAR $\mathcal{E}(n)$ P/ CADA PADRÃO ⇒ MINIMIZAR E_{av}

Algoritmo - método

MÉTODO USADO: GRADIENTE DESCENDENTE



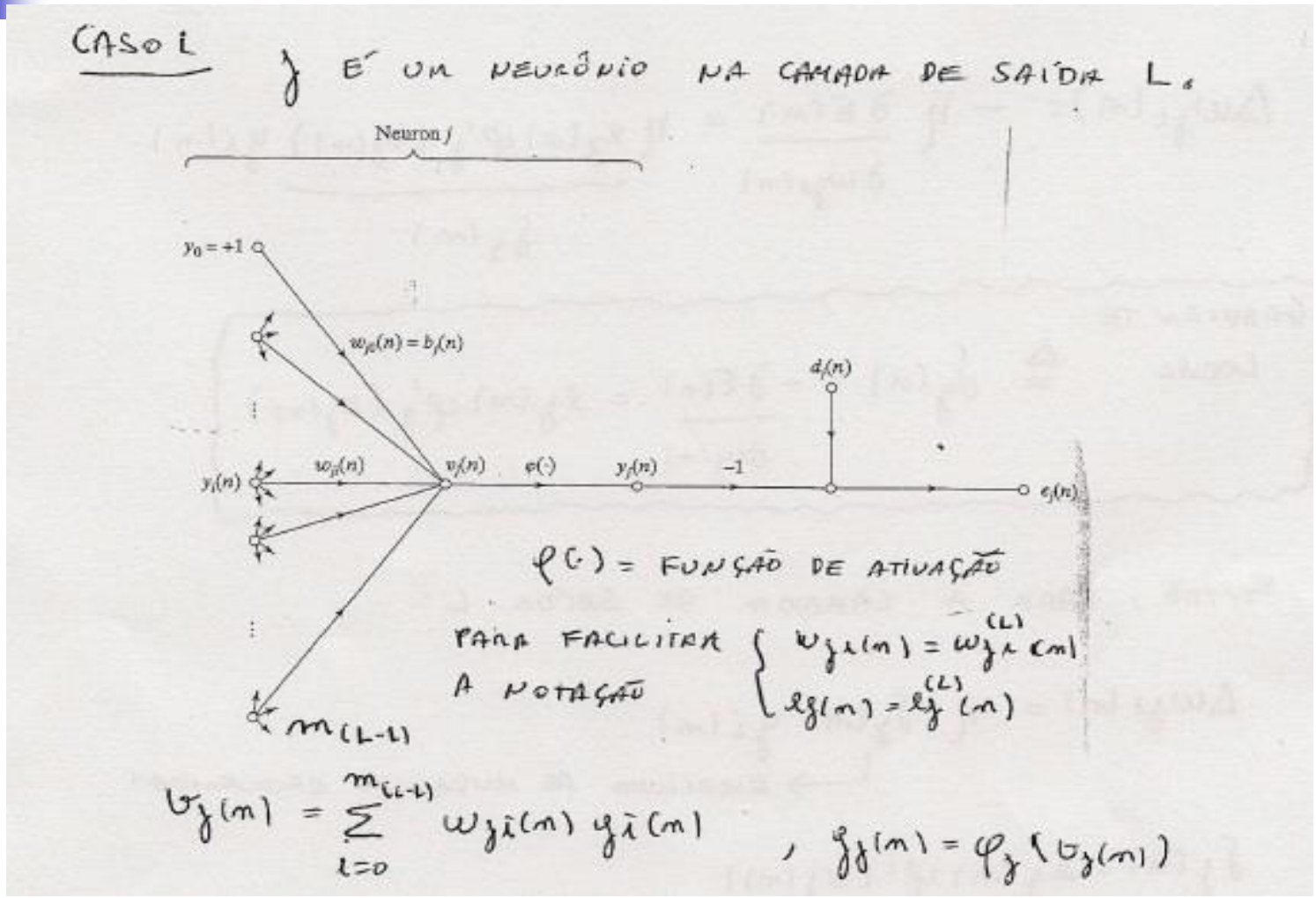
$$\Delta w(n) = -\eta \frac{\partial E(n)}{\partial w(n)}$$

$$\Delta w(n) = -\eta \nabla_w E(n)$$

ENTÃO:

$$\Delta w_{ji}^{(l)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}^{(l)}(n)}$$

Caso1: neurônio camada saída



Caso1 (cont.)

REGRA DA CADEIA:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$

$$e_j(n) \qquad -1 \qquad \varphi_j'(v_j(n)) \qquad y_i(n)$$

$$\boxed{\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot \varphi_j'(v_j(n))\, y_i(n)}$$

MAS TAL QUE:

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

Caso 1: atualização dos pesos

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \underbrace{e_j(n)\varphi'_j(v_j(n))}_{\delta_j(n)} y_i(n)$$

GRADIENTE LOCAL

$$\delta_j(n) \triangleq -\frac{\partial E(n)}{\partial v_j(n)} = e_j(n)\varphi'_j(v_j(n))$$

ENTÃO, PARA A CAMADA DE SAÍDA L:

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, y_i(n)$$

→ DIRECIONA AS MUDANÇAS REQUERIDAS

$$\delta_j(n) = e_j(n)\varphi'(v_j(n))$$

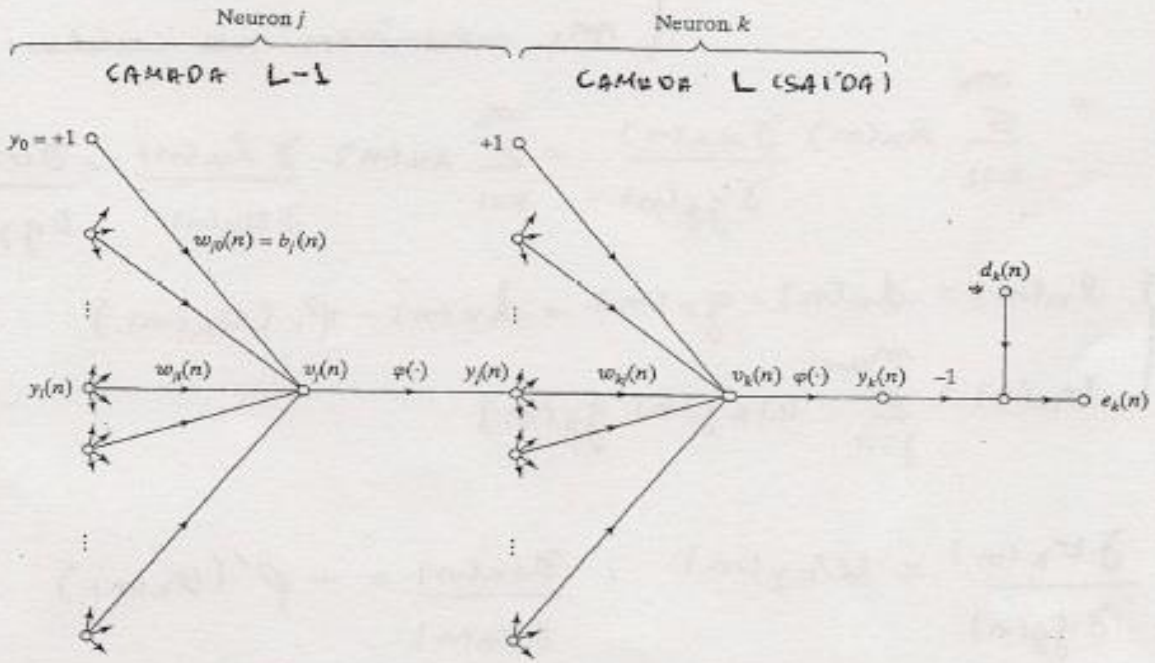
→ CONHECIDO PARA A CAMADA DE SAÍDA L:

$$e_j(n) = d_j(n) - y_j(n)$$

Caso2: neurônio camada escondida

CASO 2: NEURÔNIO j ESTÁ EM UMA UNIDADE ESCONDIDA.

- QUAL É O SINAL DE ERRO PARA UMA UNIDADE ESCONDIDA?
- R: RECURSÃO



SIMPLIFICANDO A NOTAÇÃO: $w_{kj}(n) = w_{ji}^{(l)}(n)$

$w_{ji}(n) = w_{jk}^{(l-1)}(n)$

Caso 2: gradiente camada escondida

GRADIENTE LOCAL $\delta_j(n)$ PARA O NEURÔNIO j (CAMADA ESCONDIDA)

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \cdot \varphi_j'(v_j(n)) \quad (1)$$

$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}$ → ?

Caso 2: cálculo gradiente

- CALCULANDO $\partial \mathcal{E}(n) / \partial y_j(n)$

$$\mathcal{E}(n) = \frac{1}{2} \sum_{k=1}^{m_L} e_k^2(n)$$

→ neurônio k está na camada de saída

m_L neurônios na saída

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_{k=1}^{m_L} e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_{k=1}^{m_L} e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}$$

COMO

$$\begin{cases} e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \\ v_k(n) = \sum_{j=0}^{m_{(L-1)}} w_{kj}(n)\, y_j(n) \end{cases}$$

ENTÃO:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n), \quad \frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi'(v_k(n))$$

Caso2: gradiente (cont.)

LOGO:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_{k=1}^{m_L} e_k(n)\,\varphi'_k(v_k(n))\cdot w_{kj}(n)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_{k=1}^{m_L} \delta_k(n)\cdot w_{kj}(n)$$

$\delta_k(n)$ → gradiente local da camada L (SAÍDA) NEURÔNIO k

COMO DE (1):

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)}\cdot \varphi'_j(v_j(n))$$

$$\delta_j(n) = \varphi'_j(v_j(n))\cdot \sum_{k=1}^{m_L} \delta_k(n)\,w_{kj}(n)$$

j = neurônio da camada L-L

$\varphi'_j(v_j(n))$ → DEPENDE SOMENTE DO NEURÔNIO j

$\sum_{k=1}^{m_L} \delta_k(n)\,w_{kj}(n)$ → depende dos neurônios da camada imediatamente à direita

Atualização pesos camada escondida

PODEMOS ESCREVER:

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, y_i(n) \qquad (2)$$

COM

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_{k=1}^{m_L} \delta_k(n)\, w_{kj}(n) \qquad \text{P/ } j \in \text{CAMADA ESCONDIDA } (L-1)$$

$$\delta_j(n) = e_j(n)\, \varphi'(v_j(n)) \qquad \text{P/ } j \in \text{CAMADA DE SAÍDA } L$$

Retropropagação

SE O NEURÔNIO j ESTÁ EM UMA CAMADA ESCONDIDA l, $\delta_j^{(l)}(n)$ PODE SER OBTIDO PELA SOMA DOS GRADIENTES LOCAIS DA CAMADA SEGUINTE $(l+1)$, DE MANEIRA RECURSIVA, CAMADA POR CAMADA, INICIANDO DA CAMADA DE SAÍDA L E CALCULANDO-SE ENTÃO A MUDANÇA NOS PESOS DADA POR (2) EM CADA CAMADA.

Retropopagação com momento

MOMENTO (MELHORA A ESTABILIDADE DO ALGORITMO)

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

α: FATOR DE MOMENTO

η: TAXA DE APRENDIZAGEM

$$0 < \eta < 1$$

$$0 \leq \alpha < 1$$

ou

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

Resumo Algoritmo

MODO "PADRÃO A PADRÃO", SEQÜÊNCIAL, OU ESTOCÁSTICO

Neste caso os pesos são atualizados após a apresentação de cada exemplo de treinamento

1- Inicializar $w_{ji}^{(l)}(0)$, $n=0$, η, α

2- "RANDOMIZAR" A ORDEM DO CONJUNTO DE TREINAMENTO (x, d)

Resumo Algoritmo (cont.)

(3) - PARA CADA PADRÃO DO CONJ. DE TREINAMENTO X:

$(X = [x_1\ x_2 \dots x_{m_0}]^T)$

(3.1) COMPUTAÇÃO DIRETA:

CALCULAR $v_j^{(l)}(n) = \sum_{i=0}^{m_l} w_{ji}^{(l)}(n)\, y_i^{(l-1)}(n)$

(começando da entrada e seguindo em direção à saída)

OBS: $y_j^0(n) = x_j(n)$, $y_0^{(l)}(n) = +1$

(3.2) DETERMINAR O ERRO NA SAÍDA: $e_j^{(L)}(n) = d_j(n) - y_j^{(L)}(n)$

(3.3) RETRO PROPAGAÇÃO: Calcular os ~~erros~~ gradientes locais

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\, \varphi'_j(v_j^{(L)}(n)) & , \text{se } j \in L \\ \varphi'_j(v_j^{(l)}(n)) \sum_{k=1}^{m_{(l+1)}} \delta_k^{(l+1)}(n)\, w_{kj}^{(l+1)}(n) & , \text{se } j \notin L \end{cases}$$

(3.4) AJUSTA OS PESOS A CADA PADRÃO APRESENTADO:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[\Delta w_{ji}^{(l)}(n-1)\right] + \eta\, \delta_j^{(l)}(n)\, y_i^{(l-1)}(n)$$

COM $\Delta w_{ji}^{(l)}(n-1) = w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)$

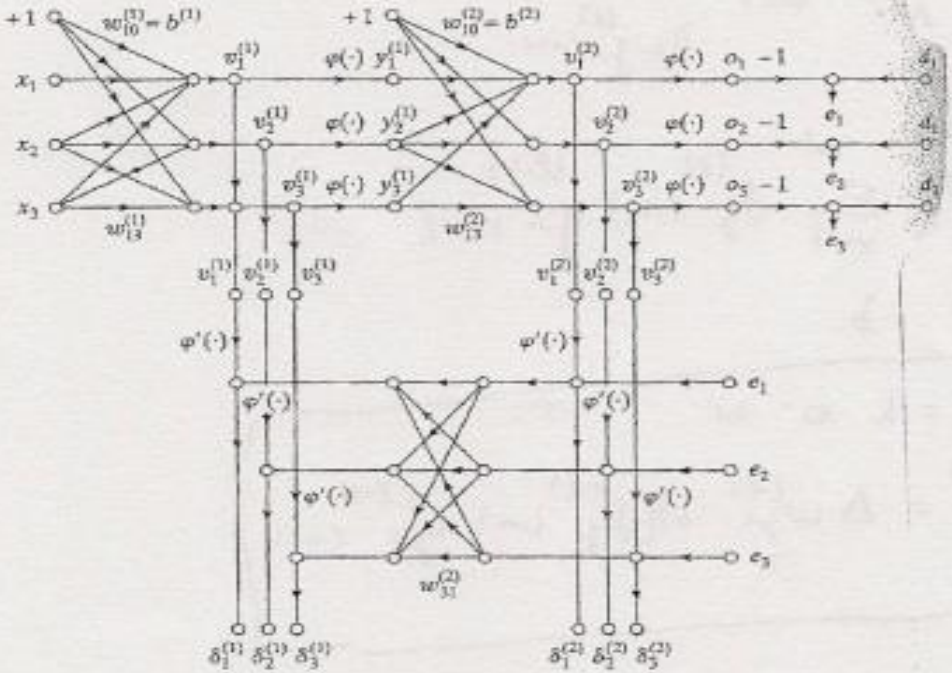
(4) SE NÃO ATINGIU CRITÉRIO DE PARADA VÁ PARA (2), CASO CONTRÁRIO FIM.

Os dois passos do algoritmo

DIAGRAMA DE FLUXO DE SINAL

DOIS PASSOS:

- COMPUTAÇÃO DIRETA
- RETROPROPAGAÇÃO DOS ERROS



Modo Batelada (*Batch*)

- A correção dos pesos é realizada após a apresentação de todos os N padrões de treinamento.

$$\Delta w_{ji}^{(l)}(n) = -\eta' \frac{\partial E_{av}}{\partial w_{ji}^{(l)}} =$$

$$\Delta w_{ji}^{(l)}(n) = \left(\frac{\eta'}{N}\right) \sum_{n=1}^{N} \frac{\partial E(n)}{\partial w_{ji}^{(l)}(n)}$$

$$\Delta w_{ji}^{(l)}(n) = \eta \sum_{n=1}^{N} \delta_j^{(l)}(n)\, y_i^{(l-1)}(n)$$

PARA $n = 1$ a N

$$\Delta w_{ji}^{(l)} = \Delta w_{ji}^{(l)} + \eta \delta_j^{(l)}(n)\, y_i^{(l-1)}(n)$$

Algoritmo modo batelada

p = número da época;

$\Delta w_{ji}(p)$ = variação dos pesos no fim da época p;

$w_{ji}^{(l)}(p)$ = peso na época p

① INICIALIZAÇÃO: $p = 0$, $\Delta w_{ji}^{(l)}(p) = 0$, η, α

$w_{ji}^{(l)}(p)$ = Valor aleatório $([-1, 1])$

Algoritmo modo batelada (cont.)

(2) Para cada apresentação p de uma nova época

(2.1) $p = p + 1$; incrementa número de épocas

(2.2) $\Delta w_{ji}^{(l)} = 0$; zera acumulador da variação do peso na época

(2.3) Para cada padrão n do conj. de treinamento

(2.3.1) Computação direta

Calcular $v_j^{(l)}(n)$ obtendo a saída da rede

(2.3.2) Determinar $\delta_j^{(l)}(n)$ e calcular

$$\Delta w_{ji}^{(l)} = \Delta w_{ji}^{(l)} + \eta\, \delta_j^{(l)}(n)\, y_i^{(l-1)}(n)$$

(3) ATUALIZA OS PESOS A CADA ÉPOCA APRESENTADA

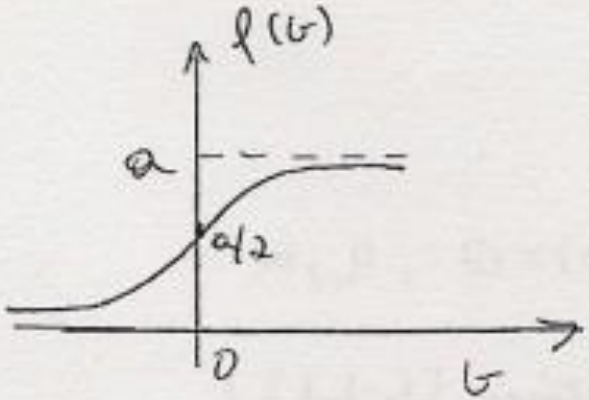
$$w_{ji}^{(l)}(p+1) = w_{ji}^{(l)}(p) + \Delta w_{ji}^{(l)} + \alpha\, \Delta w_{ant_{ji}}(p-1)$$

$$\Delta w_{ant_{ji}}^{(l)}(p) = w_{ji}^{(l)}(p+1) - w_{ji}^{(l)}(p)$$

(4) SE NÃO ATINGIR CRITÉRIO DE PARADA VÁ PARA (2), CASO CONTRÁRIO FIM.

Funções de Ativação

1) SIGMOIDAL LOGÍSTICA



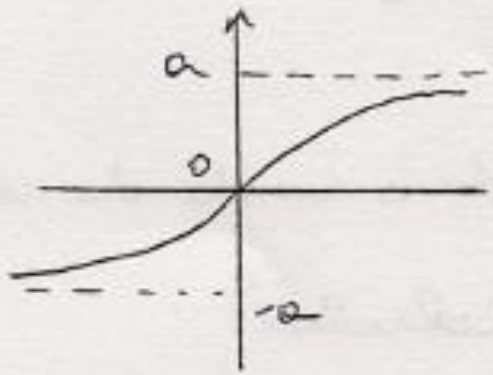
$$\varphi(v) = \frac{a}{1 + e^{-bv}}$$

$$\varphi'(v) = \frac{b}{a}\varphi(v)[a - \varphi(v)]$$

$$\varphi_j'(v_j(n)) = \frac{b}{a} y_j(n)[a - y_j(n)]$$

Funções de Ativação (cont.)

2) SIGMOIDAL TANGENTE



$$\varphi(v) = a \tanh(b \cdot v)$$

$$\varphi(v) = a \left[\frac{1 - e^{-bv}}{1 + e^{-bv}} \right]$$

$$\varphi'(v) = \frac{b}{a} \left[a^2 - \varphi^2(v) \right]$$

$$\varphi'_j(v_j(n)) = \frac{b}{a} \left[a^2 - y_j^2(n) \right]$$

Critérios de parada

• p = número de épocas

① VETOR GRADIENTE ($\|\cdot\|$ → norma euclideana)

$$\|\nabla_w \mathcal{E}(w(p))\| \leq \theta$$

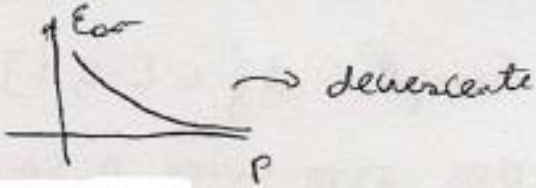
$$\|\Delta w_{ji}^{(e)}(p)\| \leq \theta$$

} PARA MODO BATELADA

② VALOR DO ERRO MÉDIO QUADRÁTICO

$$\mathcal{E}_{av}(w(p)) \leq \varepsilon$$

(PODE LEVAR A "OVERFITTING")



Critérios de parada (cont.)

(3) VARIAÇÃO DO ERRO QUADRÁTICO MÉDIO

- VARIAÇÃO ABSOLUTA

$$|\Delta E_{av}(w)| \leq \lambda$$

$$\Rightarrow E_{av}(w(p)) - E_{av}(w(p-1)) \geq -\lambda$$

- VARIAÇÃO RELATIVA

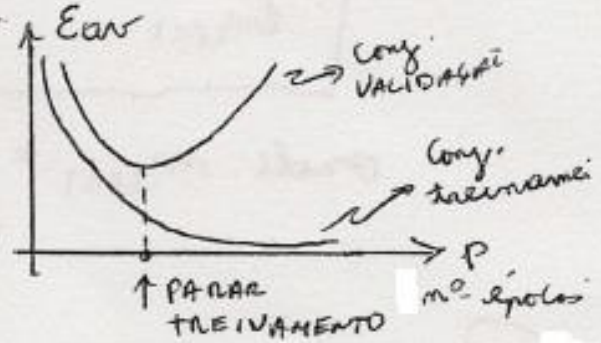
$$\frac{|\Delta E_{av}(w(p))|}{E_{av}(w(p))} \leq \lambda \Rightarrow \frac{E_{av}(w(p)) - E_{av}(w(p-1))}{E_{av}(w(p-1))} \geq -\lambda$$

(4) PARAGEM ANTECIPADA (MELHOR!)

"EARLY STOPPING"

↓

EVITA "OVERFITTING"



Técnicas para melhorar o algoritmo de retropropagação – Função de ativação

1) FUNÇÃO DE ATIVAÇÃO

Usar sigmoidal tangente (leva a um aprendizado mais rápido) com $a = 1,7159$, $b = 2/3$

Técnicas para melhorar o algoritmo de retropropagação – Valor das saídas

2) VALOR DAS SAÍDAS (QDO NÃO USANDO NEURÔNIO LINEAR NA SAÍDA).

SUPONDO FUNÇÃO NÃO LINEAR COM SATURAÇÃO EM $[-a, +a]$

$$\begin{cases} +a \Rightarrow d_j = a - \varepsilon \\ -a \Rightarrow d_j = -a + \varepsilon \end{cases}$$

Exemplo: $d_j \in [-1, 1]$ quando $a = 1,715$

OBS: ISTO EVITA A SATURAÇÃO DOS NEURÔNIOS DAS CAMADAS ESCONDIDAS, O QUE ACARRETA UMA DIMINUIÇÃO NA VELOCIDADE DO APRENDIZADO.

Técnicas para melhorar o algoritmo de retropropagação – Inicialização dos pesos

3) INICIALIZAÇÃO DOS PESOS

Para aprendizado uniforme e rápido, escolhemos aleatoriamente o valor inicial dos pesos de uma distribuição uniforme com:

$$\frac{-1}{\sqrt{m_{(l-1)}}} \leq w_{ji}^{l}(0) \leq \frac{1}{\sqrt{m_{(l-1)}}}$$

onde $m_{(l-1)}$ = nº de neurônios da camada anterior à camada l

Técnicas para melhorar o algoritmo de retropropagação – Normalizar entradas

(4) NORMALIZAR OS DADOS DE ENTRADA

Cada componente do vetor de entrada deve ser normalizado para média nula e variância unitária

$$X = [x_1 - x_1 \dots x_{m0}]^T$$

$$x_{i\,NORM} = \frac{x_i - \mu_i}{\sigma_i}$$

OBS: É interessante que os x_i sejam descorrelacionados p/ acelerar a aprendizagem.

Técnicas para melhorar o algoritmo de retropropagação – Taxa de aprendizagem

(5) TAXA DE APRENDIZAGEM

- $\eta = 0,1$ COMO PRIMEIRA TENTATIVA
- Diminui-se η se a função critério E ou $\frac{dE}{dt}$ oscila durante a aprendizagem.
- $\uparrow \eta$ se a aprendizagem parece indevidamente vagarosa
- η pode variar durante a aprendizagem

Técnicas para melhorar o algoritmo de retropropagação – momento

(6) MOMENTO

$\alpha \cong 0,9$ TÍPICO.

(7) TREINAMENTO SEQUENCIAL (BATELADA OU "ON LINE"

ON LINE ⇒ CONJ. TREINAMENTO GRANDE

BATELADA É TIPICAMENTE MAIS VAGAROSO QUE SEQUENCIAL

Conjunto de dados

- Técnicas de pré-processamento são frequentemente utilizadas para tornar o conjunto de dados mais adequados para uso de algoritmos de aprendizagem:
  - **Eliminação manual de atributos**: quando um atributo de entrada não contribui para a estimativa do atributo alvo (saída) ele é considerado irrelevante (ex: valor igual para todos os padrões/objetos);
  - **Integração de dados**: dados a serem utilizados em diferentes conjuntos;
  - **Amostragem de dados**: a amostragem de dados deve ser representativa dos conjunto de dados original;
  - **Balanceamento de dados**: evitar que que o número de objetos de uma dada classe seja muito diferente das demais
  - **Limpeza de dados**: dados inconsistentes, dados redundantes, dados com ruídos;
  - **Redução de dimensionalidade**: agregação (PCA), seleção;
  - **Transformação de dados**: ex: normalizaçãotransformação de atributo simbólicos.