DEMONSTRAÇÃO DO ALGORÍTMO DA REPROPAGAÇÃO

BACK-PROPAGATION ALGORITHM
DELTA RULE

por A.C.Mattos

O algorítmo clássico (1) para a otimização de redes neurais artificiais (ANN), e portanto para a determinação dos pesos das ligações (weights), é a seguir demonstrado. Antes, entretanto, alguns lemas e definições serão apresentados.

DEFINIÇÃO

Chama-se Função Sigmóide (2):

$$f(u) = \frac{1}{1+e^{-u}}$$

Um resultado interessante que dai decorre é que:

$$f'(u) = \frac{df(u)}{du} = f(u)\left[(1-f(u)\right] \qquad (1.1)$$

Esta função, juntamente com a tangente hiperbólica, é muito utilizada para modelar a sinapse dos neurônios, pois ambas produzem uma histerese. A propriedade da derivada da sigmóide, que pode ser expressa em termos da própria sigmóide apenas, simplifica bastante os cálculos computacionais.

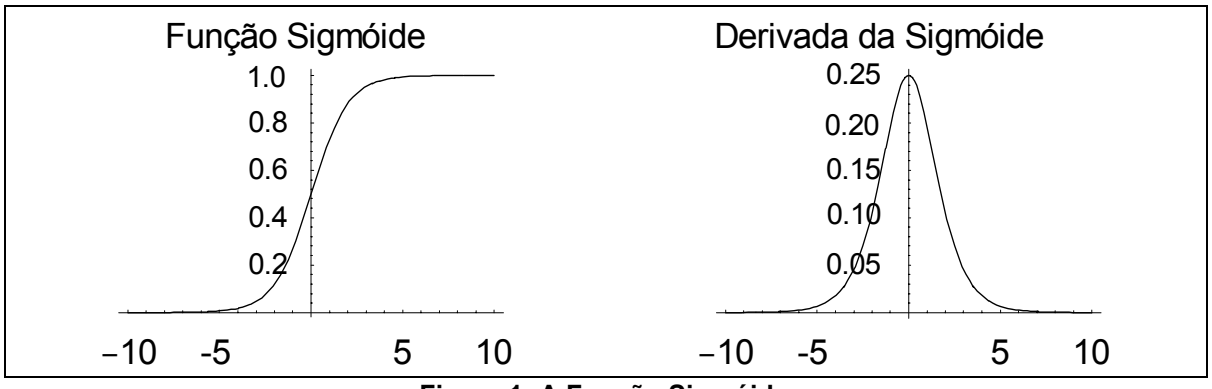

Figura 1: A Função Sigmóide

OTIMIZAÇÃO SEM RESTRIÇÕES DOS PESOS PELO ALGORÍTMO DO STEEPEST-DESCENT (GRADIENTE)

Dado as funções V_{ij} (peso associado aos neurônios da input-layer i e da hidden-layer j), W_{jk} (peso associado aos neurônios da hidden-layer j e da output-layer k) e uma função objetivo E (V,W) a ser minimizada (erro da previsão efetuada pela rede neural com os pesos V e W), o valor ótimo (local ou global, não se sabe) de V e W é obtido através dos passos a seguir, onde λ é um parâmetro que regula a velocidade da convergência (chamado de taxa de aprendizagem ou learning rate):

$$\Delta W_{jk} = -\lambda \frac{\partial E}{\partial W_{jk}} \tag{1.2}$$

$$\Delta V_{ij} = -\lambda \frac{\partial E}{\partial V_{ij}} \tag{1.3}$$

$$\left(W_{jk}\right)_{new} = \left(W_{jk}\right)_{old} + \Delta W_{jk} \tag{1.4}$$

$$\left(V_{ij}\right)_{new} = \left(V_{ij}\right)_{old} + \Delta V_{ij} \tag{1.5}$$

Note-se que ΔW e ΔV são proporcionais ao gradiente de E.

CHAIN-RULE (REGRA DO ENCADEAMENTO)

Dadas três funcões u = u [x (s, t) , w (s, t)] então:

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial w}\frac{\partial w}{\partial t}$$

$$\frac{\partial u}{\partial s} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial u}{\partial w}\frac{\partial w}{\partial s} \tag{1.6}$$

Exemplo:

$$u = x^2 + w \quad x = \frac{s}{t} \quad w = 2st$$

Então, pela chain-rule:

$$\frac{\partial u}{\partial t} = -2x\frac{s}{t^2} + 2s = -\frac{2s^2}{t^3} + 2s$$

$$\frac{\partial u}{\partial s} = 2\,x\,\frac{1}{t} + 2\,t = -\frac{2\,s}{t^2} + 2\,t$$

Agora, pela derivação direta, chegamos ao mesmo resultado:

$$u = \frac{s^2}{t^2} + 2\,s\,t$$

$$\frac{\partial u}{\partial t} = -s^2\left(\frac{2\,t}{t^4}\right) + 2\,s = -\frac{2\,s^2}{t^3} + 2\,s$$

$$\frac{\partial u}{\partial s} = \frac{2\,s}{t^2} + 2\,t$$

ERROS DE PREVISÃO DA REDE

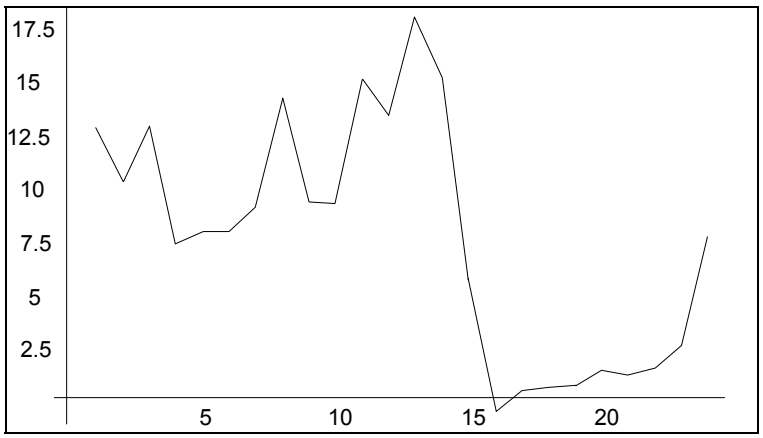

Figura 2: Taxa Mensal de Inflação medida pelo IGP-DI para 1985-86 (24 meses)

Suponhamos que uma rede deva ser treinada para apreender os dados acima e depois prever o que ocorrerá a partir de 1988. Construindo uma rede com 6 entradas e uma saída, definimos então os vetores:

X [p] = { x [p, i] } = { 12.64, 10.10, 12.70, 7.23, 7.79, 7.82 }

Y [p] = { y [p, k] } = { 8.93 }

como sendo os vetores de entrada (jan-jun/85) e saída (jul/85) para o pattern (ou época) p = 1. Com uma dada configuração de pesos { V, W } e para uma entrada X[p], a rede irá produzir uma saída:

O [p] = { o [p, k] } = { 8.75 } (8.75 é exemplificativo)

O erro ε [p, k] cometido pelos neurônios de saída no pattern p terá sido de:

ε [p, k] = y [p, k] - o [p, k] = { 0.18 }

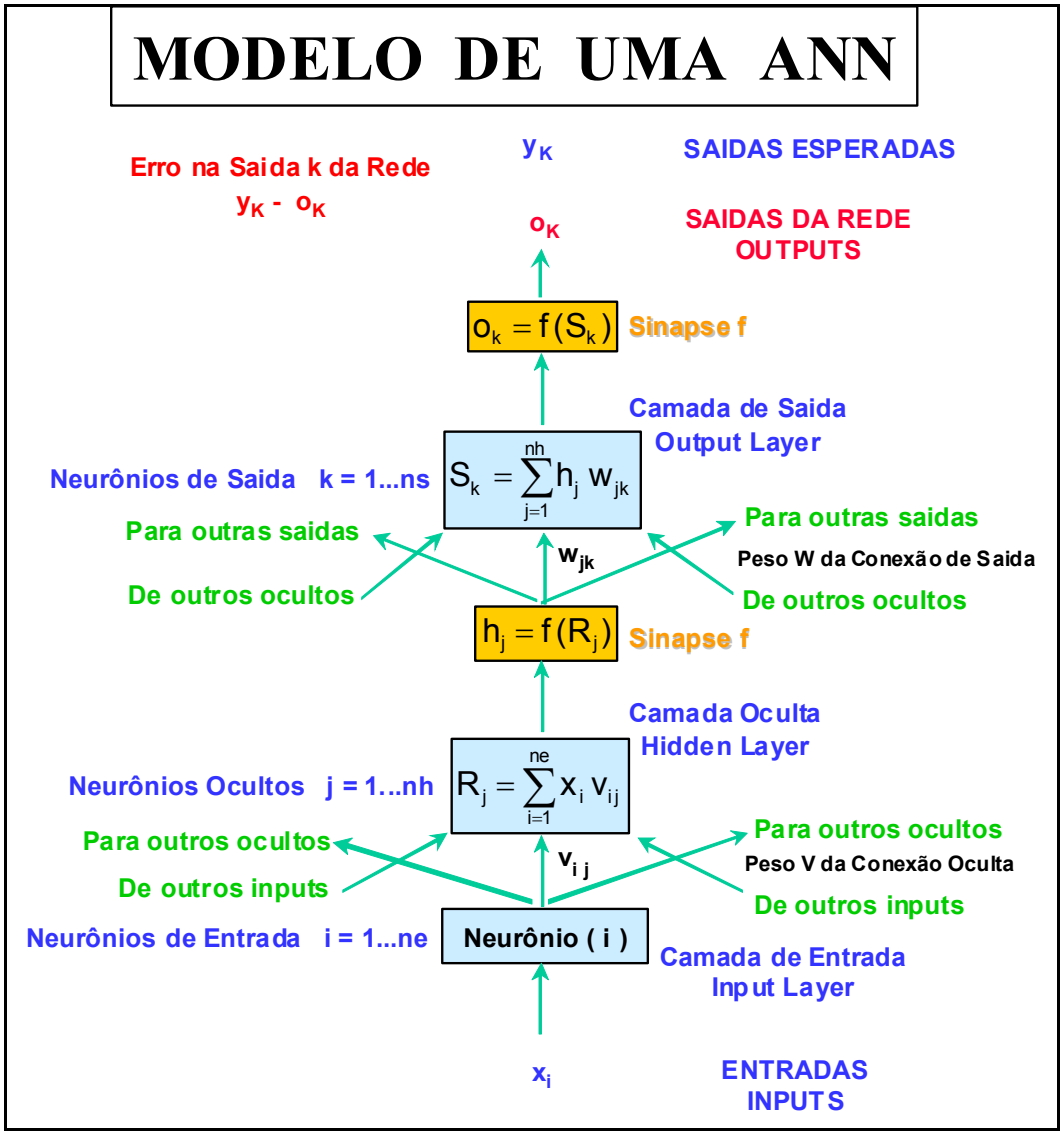

Figura 3: Trecho de uma rede neural artificial

Definem-se o erro E [p] do pattern p e o erro total da rede E por:

$$E[p] = \frac{1}{2}\sum_{k=1}^{ns}\varepsilon^2[p,k]$$

$$E = \frac{1}{2}\sum_{p=1}^{np}\sum_{k=1}^{ns}\varepsilon^2[p,k] \qquad (1.7)$$

onde

ns = nº de neurônios de saída
ne = nº de neurônios de entrada
nh = nº de neurônios na hidden-layer
np = nº de patterns

DELTA-RULE PARA O TREINAMENTO DA REDE (CÁLCULO DOS PESOS)

Para o cálculo dos pesos usaremos o conhecido algorítmo do steepest-descent (descida mais íngreme). Precisamos calcular as derivadas parciais do erro E com relação aos pesos W e V (Eq. 1.2 e 1.3). Para tanto, definamos a soma R [j] dos sinais que chegam a cada hidden-neuron, o sinal de saída h [j] de cada hidden-neuron (obtido pela ação de uma sinapse f sobre R), a soma S [k] dos sinais que chegam ao output-neuron e o sinal de saída o [k] do output-neuron (também obtido pela ação de uma sinapse f sobre S) (Fig. 3):

$$R_j = \sum_{i=1}^{ne} x_i v_{ij} \qquad h_j = f(R_j) \qquad S_k = \sum_{j=1}^{nh} h_j w_{jk} \qquad o_k = f(S_k) \qquad (1.8)$$

Essas relações definem completamente o sistema e constituem o modelo matemático da rede neural (simplificada) do cérebro humano.

Vamos agora deduzir a Delta-Rule. Para abreviar a notação, usaremos E em lugar de E [p], pois estamos supondo que o treinamento seja feito otimizando-se os pesos para a apresentação de **cada** pattern, quando então E [p] se mantém constante. Existem também outros esquemas de treinamento, que não alteram a dedução abaixo. No entanto, ainda não foi demonstrado que a soma dos E [p] mínimos é igual ao mínimo de E.

$$\delta_q^{(w)} \equiv -\frac{\partial E}{\partial S_q} \qquad (1.9) \quad \text{definição do Delta}$$

$$\frac{\partial E}{\partial S_q} = \frac{\partial E}{\partial o_q}\frac{\partial o_q}{\partial S_q} \qquad (1.10) \quad \text{pela Chain-Rule}$$

$$\frac{\partial E}{\partial o_q} = \frac{\partial}{\partial o_q}\left[\frac{1}{2}\sum_k (y_k - o_k)^2\right] = \frac{1}{2}\sum_k \left[\frac{\partial}{\partial o_q}(y_k - o_k)^2\right] = \frac{1}{2}\frac{\partial}{\partial o_q}(y_q - o_q)^2 = -(y_q - o_q)$$

$$\frac{\partial o_q}{\partial S_q} = o_q(1-o_q) \qquad \text{de (1.1) e (1.8)}$$

$$\boxed{\delta_q^{(w)} = (y_q - o_q)\, o_q\,(1-o_q)} \qquad (1.11)$$

$$\delta_q^{(v)} \equiv \frac{\partial E}{\partial R_q} \qquad \text{definição de Delta}$$

$$\frac{\partial E}{\partial R_q} = \frac{\partial E}{\partial h_q}\frac{\partial h_q}{\partial R_q} \qquad \text{pela Chain-Rule}$$

$$\frac{\partial h_q}{\partial R_q} = h_q\,(1-h_q) \qquad \text{de (1.1) e (1.8)}$$

$$\frac{\partial E}{\partial h_q} = \sum_r \left(\frac{\partial E}{\partial S_r}\right)\left(\frac{\partial S_r}{\partial h_q}\right) \qquad \text{pela Chain-Rule generalizada (Eq. 1.6)}$$

$$\frac{\partial E}{\partial h_q} = \sum_r \left(-\delta_r^{(w)}\right)\left(\frac{\partial}{\partial h_q}\left[\sum_j h_j w_{jr}\right]\right) = -\sum_r \delta_r^{(w)}\, w_{qr} \qquad \text{de (1.9) e (1.11)}$$

$$\boxed{\delta_q^{(v)} = h_q\,(1-h_q)\sum_r \delta_r^{(w)}\, w_{qr}} \qquad (1.12)$$

$$-\frac{\partial E}{\partial w_{jk}} = -\frac{\partial E}{\partial S_k}\frac{\partial S_k}{\partial w_{jk}} = \delta_r^{(w)}\, h_j \qquad \text{de (1.2) e (1.8) e (1.11)}$$

$$\frac{\partial E}{\partial v_{ij}} = -\frac{\partial E}{\partial R_j}\frac{\partial R_j}{\partial v_{ij}} = \delta_j^{(v)} x_i \qquad \text{de (1.3) e (1.8) e (1.12)}$$

$$\Delta w_{jk} = \lambda\, \delta_k^{(w)}\, h_j \qquad \text{expressão do steepest-descent}$$

$$\Delta v_{ij} = \lambda \, \delta_j^{(v)} x_i$$ expressão do steepest-descent

Assim, o treinamento da rede se dá pelos seguintes passos:

1. Randomizam-se os pesos W e V da rede;

2. Injeta-se um pattern X nos neurônios de entrada;

3. Calculam-se por (1.11) os deltas dos w's, usando-se a saída esperada Y;

4. Calculam-se por (1.12) os deltas dos v's; note-se que aqui os deltas das saídas são repropagados nos hidden-neurons, dado pela somatória em (1.12), donde o nome de back-propagation algorithm;

5. Calculam-se os novos pesos W e V por (1.4) e (1.5);

6. Volta-se a (2), desde que o erro E [p] (1.7) tenha diminuído mais que um certo limite.

A convergência desse algorítmo, num caso real, pode levar horas, donde a necessidade de sua programação ser feita em linguagem rápida (como C ou Assembly), e CPU bastante rápida (processamento paralelo, por exemplo).

Notas:

(1) Ver [Kosko, p.207], [Rumelhart, vol.1, p.322], [Masters, p.94], [Freeman, p.72], [Blum, p.55]. A notação é bastante confusa nesses autores. Aqui traduzimos para uma notação inteligível. Há algoritmos mais eficientes, porém mais complexos.

(2) "Sigmóide" vem do grego Σîγμα = letra sigma, mais o sufixo ειδής = parecido com.