UM ALGORITMO ACELERADOR PARA TREINAMENTO DE REDES NEURONAIS MLP

Magno T. Madeira da Silva[1], Max Gerken[2] *

Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica da Universidade de São Paulo
<http://www.lcs.poli.usp.br>
{magno, mgk}@lcs.poli.usp.br

RESUMO

É apresentado um novo algoritmo para treinamento de redes neuronais do tipo *Multilayer Perceptron*. Ele é obtido a partir da discretização no tempo de um algoritmo de tempo contínuo baseado na aceleração de parâmetros. Resultados de simulações evidenciam que, às custas de um moderado aumento da complexidade computacional, o algoritmo proposto apresenta uma maior velocidade de convergência que o conhecido algoritmo *Backpropagation*. Além disso, apresenta uma complexidade computacional bastante menor do que algoritmos de treinamento baseados no método dos mínimos quadrados.

Palavras-chaves— **redes neuronais, algoritmos de treinamento, equalização.**

1. INTRODUÇÃO

Um problema comum em controle adaptativo e na estimação de parâmetros é o ajuste recursivo de uma estimativa $\mathbf{w}(t)$ de um vetor de parâmetros $\mathbf{w}_{\mathrm{o}}$ a partir de um sinal medido

$$d(t) = \mathbf{x}^T(t)\mathbf{w}_{\mathrm{o}} + \xi(t), \tag{1}$$

com $\mathbf{x}(t)$ sendo o vetor do sinal de entrada (regressor) e $\xi(t)$ o ruído de medida. Em princípio o objetivo é minimizar tanto o erro de estimação

$$e(t) = \mathbf{x}^T(t)\mathbf{w}(t) - d(t), \tag{2}$$

como o erro de parâmetro

$$\boldsymbol{\Delta}\mathbf{w}(t) = \mathbf{w}(t) - \mathbf{w}_{\mathrm{o}}\,. \tag{3}$$

O método tradicionalmente utilizado em controle adaptativo ajusta a primeira derivada da estimativa dos parâmetros fazendo-a proporcional ao erro de estimação:

$$\begin{aligned} \dot{\mathbf{w}}(t) &= -\mathbf{M}\mathbf{x}(t)e(t) \\ e(t) &= \mathbf{x}^T(t)\mathbf{w}(t) - d(t), \end{aligned} \tag{4}$$

com $\mathbf{M}$ sendo uma matriz positiva definida de dimensões apropriadas.

É interessante observar que este algoritmo é o análogo em tempo contínuo do popular algoritmo LMS. Este pode ser obtido discretizando-se as equações (4) segundo o método de Euler direto. Para isso basta considerar em (4) $\mathbf{M} = \mu_{\mathrm{o}}\mathbf{I}$ e $\mu = \mu_{\mathrm{o}}T$, com T sendo o passo de integração. Assim resultam as expressões

$$\begin{aligned} \mathbf{w}[n+1] &= \mathbf{w}[n] - \mu\mathbf{x}[n]e[n] \\ e[n] &= \mathbf{x}^T[n]\mathbf{w}[n] - d[n] \end{aligned}$$

que correspondem ao popular algoritmo LMS.

Recentemente foi introduzido o chamado algoritmo acelerador (de tempo contínuo) que ajusta a segunda derivada ("aceleração") dos parâmetros [1]. Segundo resultados de [1], este algoritmo pode apresentar, em comparação com o algoritmo tradicional, um melhor comportamento transitório e em regime, às custas de um moderado aumento da complexidade computacional.

Baseado neste algoritmo, foi introduzido em [2], [3] um algoritmo acelerador de tempo discreto, obtido a partir da discretização do algoritmo de tempo contínuo. Para isso foi utilizado o método de Euler reverso. Como foi mostrado em [2], o método de Euler direto resulta neste caso em um algoritmo instável e a discretização segundo a regra do trapézio leva a um algoritmo com uma complexidade computacional superior. Assim, a versão apresentada em [2], [3] é a princípio a menos complexa para a qual se consegue garantir a estabilidade. Em [2], [3] é mostrado que, para sinais coloridos, o algoritmo acelerador lá proposto consegue atingir um melhor compromisso entre rapidez de convergência e qualidade da estimação do que os algoritmos LMS e LMS normalizado.

Neste trabalho é introduzido um algoritmo acelerador para treinamento de redes neuronais do tipo MLP. Tal algoritmo é obtido a partir de uma versão ligeiramente modificada do algoritmo acelerador de tempo contínuo.

*Este trabalho foi financiado pela FAPESP[1] (proc. 99/00188-0) e CNPq[2] (proc. 300521/92-8).

No texto que segue, o algoritmo acelerador de tempo contínuo é apresentado de forma resumida. A modificação efetuada neste algoritmo consiste na introdução de uma não-linearidade na saída. A discretização do mesmo é feita segundo o método de Euler reverso, obtendo-se dessa forma, as equações do cálculo progressivo do algoritmo de treinamento. As equações do cálculo regressivo são apresentadas logo a seguir. A retropropagação do erro segue o mesmo princípio que a do tradicional algoritmo de retropropagação (*Backpropagation*), só que considerando a propagação do erro *a priori*. Posteriormente, são apresentados alguns resultados de simulações considerando a equalização de canais de comunicação. Nestas simulações são comparadas as velocidades de convergência do algoritmo acelerador, do algoritmo *Backpropagation* (BP) [6] e do algoritmo de treinamento baseado no método dos mínimos quadrados recursivo (RLS-MLP) apresentado em [4].

2. O ALGORITMO ACELERADOR DE TEMPO CONTÍNUO MODIFICADO

Como pode ser visto em [1] e [2], o algoritmo acelerador de tempo contínuo é descrito pelas seguintes equações:

$$\dot{\mathbf{w}}(t) = \mathbf{q}(t) \quad (5)$$

$$\dot{\mathbf{q}}(t) = -\mathbf{M}_1\mathbf{x}(t)e(t) + -2\mathbf{M}_1(\mathbf{M}_2 + \mathbf{x}(t)\mathbf{x}^T(t)\mathbf{M}_1\mathbf{M}_3)\mathbf{q}(t) \quad (6)$$

$$e(t) = \mathbf{x}(t)\mathbf{w}(t) - d(t), \quad (7)$$

sendo:
- $\mathbf{x}(t)$ o vetor regressor (sinal de entrada);
- $\mathbf{w}(t)$ o vetor de parâmetros;
- $d(t)$ o sinal desejado;
- $e(t)$ o erro de predição; e
- $\mathbf{q}(t)$ a derivada do vetor de parâmetros.

Todos os vetores acima têm dimensão N_0. As matrizes $\mathbf{M}_1$, $\mathbf{M}_2$ e $\mathbf{M}_3$ são simétricas e positivas definidas tendo dimensão N_0 x N_0.

As expressões (5) a (7) podem ser implementadas com N_0 integradores, sendo que na prática as matrizes $\mathbf{M}_i$, $i = 1, 2, 3$, são tomadas proporcionais à matriz identidade, o que diminui consideravelmente a complexidade da implementação [2]. Além disso, conforme [1] e [2], este sistema é estável segundo o critério de Liapunov quando as matrizes $\mathbf{M}_1$, $\mathbf{M}_2$ e $\mathbf{M}_3$ satisfizerem as seguintes condições:

$$4\mathbf{M}_1\mathbf{M}_3\mathbf{M}_1\mathbf{M}_2 \geq \mathbf{I} \quad (8)$$

$$\mathbf{M}_2\mathbf{M}_1\mathbf{M}_3 + \mathbf{M}_1\mathbf{M}_3\mathbf{M}_2 \geq \mathbf{M}_1^{-1}/2. \quad (9)$$

Introduzindo uma função não-linear $\varphi(\cdot)$ na saída, a equação (7) resulta em

$$e(t) = \varphi(\mathbf{x}^T(t)\mathbf{w}(t)) - d(t). \quad (10)$$

As equações (5), (6) e (10) descrevem o algoritmo acelerador de tempo contínuo modificado e serão usadas na seção seguinte para obtenção das equações progressivas do algoritmo de treinamento de uma rede neuronal do tipo MLP.

3. O ALGORITMO ACELERADOR PARA TREINAMENTO DE REDES MLP

3.1. Cálculo Progressivo

Para obter as equações do cálculo progressivo do algoritmo acelerador para treinamento de uma rede MLP, basta considerar inicialmente apenas um neurônio de uma camada k, cujo esquema é mostrado na Figura 1. Além disso, considera-se inicialmente a camada k como a de saída de modo que o erro seja calculado através da comparação do sinal desejado com a saída da rede. Para simplificar a notação, os índices da camada e do neurônio serão omitidos nesta dedução, ou seja $\mathrm{w}_{\mathrm{i1}}^{(\mathrm{k})}[\mathrm{n}] = \mathrm{w}[\mathrm{n}]$.

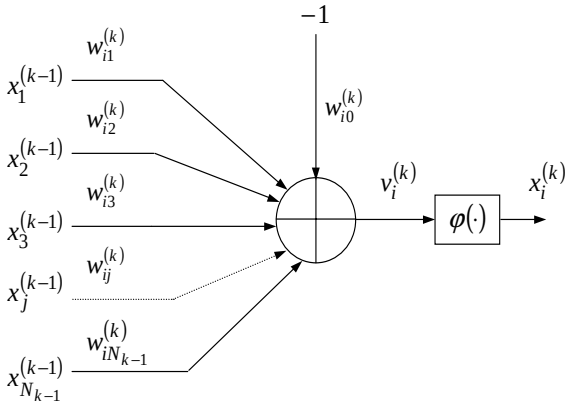

Fig. 1. Modelo do i-ésimo neurônio da camada k.

Discretizando as equações (5), (6) e (10) pelo método de Euler reverso ($f[n] = f[n-1] + Tg[n]$, sendo $\dot{f}(t) = g(t)$) e considerando $T = \alpha$, resulta:

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \alpha\mathbf{q}[n] \quad (11)$$

$$\mathbf{q}[n] = \mathbf{q}[n-1] - \alpha\mathbf{M}_1\{\mathbf{x}[n]e[n] + +2(\mathbf{M}_2 + \mathbf{x}[n]\mathbf{x}^T[n]\mathbf{M}_1\mathbf{M}_3)\mathbf{q}[n]\} \quad (12)$$

$$e[n] = \varphi(\mathbf{x}^T[n]\mathbf{w}[n]) - d[n]. \quad (13)$$

Substituindo (11) em (13), obtém-se:

$$e[n] = \varphi(\mathbf{x}^T[n]\mathbf{w}[n-1] + \alpha\mathbf{x}^T[n]\mathbf{q}[n]) - d[n]. \quad (14)$$

Expandindo $\varphi(v)$, com $v = \mathbf{x}^T[n]w[n-1] + \alpha\mathbf{x}^T[n]\mathbf{q}[n]$, em torno de $v_0 = \mathbf{x}^T[n]\mathbf{w}[n-1]$ por meio da série de Taylor e mantendo-se somente os dois termos de menor ordem da série, obtém-se

$$\varphi(v_0 + \alpha\mathbf{x}^T[n]\mathbf{q}[n]) \approx \varphi(v_0) + +\varphi'(v_0)(\alpha\mathbf{x}^T[n]\mathbf{q}[n]), \quad (15)$$

com $\varphi'(\cdot)$ indicando a derivada de $\varphi(\cdot)$. Substituindo (15) em (14) resulta então:

$$e[n] = \varphi(\mathbf{x}^T[n]\mathbf{w}[n-1]) + \\ +\varphi'(\mathbf{x}^T[n]\mathbf{w}[n-1])(\alpha\mathbf{x}^T[n]\mathbf{q}[n]) - d[n]. \quad (16)$$

As equações (12) e (16) não definem um algoritmo computável. Este problema pode ser resolvido introduzindo-se um erro *a priori* $e_a[n]$ definido como

$$e_a[n] = \varphi(\mathbf{x}^T[n]\mathbf{w}[n-1]) - d[n]. \quad (17)$$

Assim, pode-se rescrever a equação (16) como

$$e[n] = e_a[n] + \varphi'(\mathbf{x}^T[n]\mathbf{w}[n-1])\alpha\mathbf{x}^T[n]\mathbf{q}[n]. \quad (18)$$

e substituir este resultado na equação (12). Resulta

$$\mathbf{q}[n] = \mathbf{C}^{-1}(\mathbf{q}[n-1] - \alpha\mathbf{M}_1\mathbf{x}[n]e_a[n]), \quad (19)$$

com

$$\mathbf{C} = \{\mathbf{I} + \alpha^2\varphi'(\mathbf{x}^T[n]\mathbf{w}[n-1])\mathbf{M}_1\mathbf{x}[n]\mathbf{x}^T[n] + \\ +2\alpha\mathbf{M}_1(\mathbf{M}_2 + \mathbf{x}[n]\mathbf{x}^T[n]\mathbf{M}_1\mathbf{M}_3)\}. \quad (20)$$

As equações (17), (20), (19), (11) e (13) constituem um algoritmo computável, cuja complexidade computacional pode ser reduzida considerando matrizes $\mathbf{M}_i$ diagonais, isto é, $\mathbf{M}_i = m_i\mathbf{I}$, $i = 1, 2, 3$. Assim obtém-se

$$\mathbf{C} = a\mathbf{I} + b[n]\mathbf{x}[n]\mathbf{x}^T[n],$$

sendo

$$a = 1 + 2\alpha m_1 m_2$$

e

$$b[n] = \alpha^2 m_1\varphi'(\mathbf{x}^T[n]\mathbf{w}[n-1]) + 2\alpha m_1^2 m_3.$$

A inversa da matriz $\mathbf{C}$ é dada por

$$\mathbf{C}^{-1} = \frac{1}{a}\left[\mathbf{I} - \frac{b[n]}{a + b[n]\|\mathbf{x}[n]\|^2}\mathbf{x}[n]\mathbf{x}^T[n]\right], \quad (21)$$

o que pode ser verificado calculando-se o produto $\mathbf{C}\mathbf{C}^{-1}$. Utilizando-se a equação (21), pode-se mostrar que o cálculo explicito de $\mathbf{C}^{-1}$ não é necessário, se reduzindo ao cálculo de um escalar como indicado na Tabela I. Dessa forma, chega-se a um conjunto de equações que exigem um menor esforço computacional.

3.2. Cálculo Regressivo

Para obter as equações do cálculo regressivo, deve-se considerar toda a estrutura da rede. Como um exemplo, na Figura 2 é mostrada uma rede MLP de três camadas segundo a configuração (N_1, N_2, N_3), sendo que cada nó consiste de um neurônio como indicado na Figura 1. No desenvolvimento a seguir são utilizadas as seguintes variáveis:

- o número N_l de neurônios da camada l, $l = 1, 2, \ldots, L$;
- o peso $\mathrm{w}_{ij}^{(l)}$ que liga o neurônio j da camada $l-1$ ao neurônio i da camada l;
- o vetor de pesos $\mathbf{w}_i^{(l)}[n]$ de dimensão $N_{l-1}+1$ reúne os pesos $\mathrm{w}_{ij}^{(l)}$ com $j = 0,\ 1,\ \cdots, N_{l-1}$;
- a saída $x_i^{(l)} = \varphi(v_i^{(l)})$ do neurônio i da camada l, sendo $v_i^{(l)}[n] = (\mathbf{x}^{(l-1)}[n])^T\mathbf{w}_i^{(l)}[n]$;
- o vetor $\mathbf{x}^{(l-1)}[n]$ de dimensão $N_{l-1}+1$ reúne o nível de ajuste -1 e todas as saídas $x_j^{(l-1)}[n]$, $j = 1,\ 2,\ \cdots,\ N_{l-1}$, dos neurônios da camada $l-1$;
- a saída *a priori* $x_{a,i}^{(l)} = \varphi(v_{a,i}^{(l)})$ do neurônio i da camada l, sendo $v_{a,i}^{(l)}[n] = (\mathbf{x}^{(l-1)}[n])^T\mathbf{w}_i^{(l)}[n-1]$.

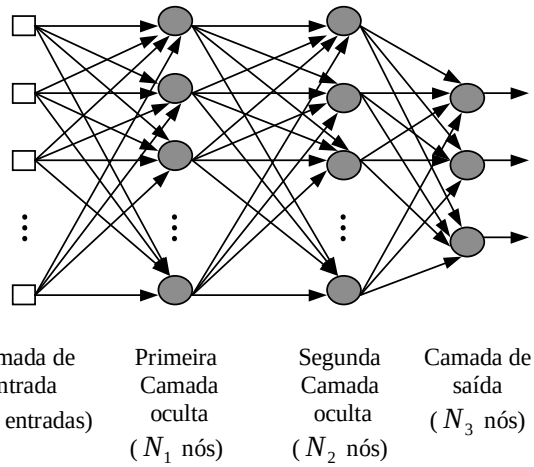

Fig. 2. Arquitetura de uma rede MLP com 3 camadas.

Para se propagar regressivamente o erro da camada de saída de uma MLP, introduz-se uma medida de desempenho

$$Q[n] = \frac{1}{2}\sum_{j=1}^{N_L}(e_{a,j}^{(L)}[n])^2, \quad (22)$$

que deve ser minimizada. Para isso toma-se sua derivada parcial em relação a $\mathbf{w}_i^{(k)}[n-1]$ e iguala-se a zero, ou seja:

$$\frac{\partial Q[n]}{\partial \mathbf{w}_i^{(k)}[n-1]} = \sum_{j=1}^{N_L}\frac{\partial e_{a,j}^{(L)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]}e_{a,j}^{(L)}[n] = \\ = \sum_{j=1}^{N_L}\frac{\partial x_{a,j}^{(L)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]}e_{a,j}^{(L)}[n] = 0\,. \quad (23)$$

Esta última expressão pode ser rescrita como

$$\sum_{j=1}^{N_L}\frac{\partial x_{a,j}^{(L)}[n]}{\partial v_{a,j}^{(L)}[n]}\sum_{p=1}^{N_{L-1}}\frac{\partial v_{a,j}^{(L)}[n]}{\partial x_p^{(L-1)}[n]}\frac{\partial x_p^{(L-1)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]}e_{a,j}^{(L)}[n] = \\ = \sum_{p=1}^{N_{L-1}}\frac{\partial x_p^{(L-1)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]}\sum_{j=1}^{N_L}\frac{\partial x_{a,j}^{(L)}[n]}{\partial v_{a,j}^{(L)}[n]}\mathrm{w}_{jp}^{(L)}[n-1]e_{a,j}^{(L)}[n] = 0.$$

A comparação desta última expressão com a equação (23) mostra que os erros *a priori* da camada $L-1$ podem ser obtidos pela expressão

$$e_{a,p}^{(L-1)}[n] = \sum_{j=1}^{N_L} \frac{\partial x_{a,j}^{(L)}[n]}{\partial v_{a,j}^{(L)}[n]} \mathrm{w}_{jp}^{(L)}[n-1] e_{a,j}^{(L)}[n]\,. \qquad (24)$$

Assim, pode-se rescrever a equação anterior à (24) da seguinte forma:

$$\sum_{p=1}^{N_{L-1}} \frac{\partial x_{p}^{(L-1)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]} e_{a,p}^{(L-1)}[n] = 0\,. \qquad (25)$$

Este procedimento (a partir de (23)) pode ser repetido para a camada anterior ($l = L-1$) e assim por diante. Desta forma, para a camada $l = k$ obtém-se:

$$\sum_{q=1}^{N_k} \frac{\partial x_{q}^{(k)}[n]}{\partial \mathbf{w}_i^{(k)}[n-1]} e_{a,q}^{(k)}[n] = 0\,, \qquad (26)$$

com

$$e_{a,q}^{(k)}[n] = \sum_{j=1}^{N_{k+1}} \underbrace{\frac{\partial x_{a,j}^{(k+1)}[n]}{\partial v_{a,j}^{(k+1)}[n]}}_{\varphi'(v_{a,j}^{(k+1)}[n])} \mathrm{w}_{jq}^{(k+1)}[n-1] e_{a,j}^{(k+1)}[n], \quad (27)$$

que é a expressão geral para o erro *a priori* propagado regressivamente.

Essencialmente a diferença entre a propagação do erro do algoritmo de treinamento acelerador em comparação com os algoritmos BP e RLS-MLP é que no primeiro se faz propagação do erro *a priori*, enquanto nos demais se faz a propagação do erro *a posteriori*.

3.3. Algoritmo acelerador para treinamento de redes neuronais MLP.

Reunindo as equações do cálculo progressivo e regressivo obtidas anteriormente e considerando $\mathbf{M}_i = m_i\mathbf{I}$, $i = 1, 2, 3$, chega-se no algoritmo acelerador para treinamento de redes do tipo MLP que está apresentado na Tabela I. A complexidade computacional do algoritmo acelerador na fase de treinamento de uma rede MLP ($N_1, N_2, \ldots, N_L$) é mostrada na Tabela II. Nesta tabela, N_0 é o número de entradas da rede e NL indica o número de vezes em que se utiliza a função não-linear $\varphi(\cdot)$, incluindo também a utilização de sua derivada $\varphi'(\cdot)$. A função de ativação $\varphi(\cdot)$ usualmente utilizada é a tangente hiperbólica [6].

4. OBSERVAÇÕES SOBRE CONVERGÊNCIA E INICIALIZAÇÃO

Para garantir a estabilidade do algoritmo acelerador (sem a não-linearidade $\varphi(\cdot)$) os valores de m_1, m_2 e m_3 devem satisfazer:

$$4m_1^2 m_2 m_3 \geq 1\,. \qquad (28)$$

$a = 1 + 2\alpha m_1 m_2$

Para $l = 1, 2, ..., L$ e $i = 1, 2, ..., N_l$:

$v_{a,i}^{(l)}[n] = \mathbf{x}^{(l-1)T}[n]\mathbf{w}_i^{(l)}[n-1]$

$v_i^{(l)}[n] = \mathbf{x}^{(l-1)T}[n]\mathbf{w}_i^{(l)}[n]$

$x_{a,i}^{(l)}[n] = \varphi(v_{a,i}^{(l)}[n])$

$x_i^{(l)}[n] = \varphi(v_i^{(l)}[n])$

$b_i^{(l)}[n] = \alpha^2\varphi'(v_{a,i}^{(l)}[n])m_1 + 2\alpha m_1^2 m_3$

$$e_{a,i}^{(l)}[n] = \begin{cases} x_{a,i}^{(l)}[n] - d_i^{(l)}[n] & l = L \\ \sum_{j=1}^{N_l+1} \varphi'(v_{a,j}^{(l+1)}[n]) \cdot & \\ \quad \cdot \mathrm{w}_{ji}^{(l+1)}[n-1] e_{a,j}^{(l+1)}[n] & l \neq L \end{cases}$$

$c_i^{(l)}[n] = \frac{1}{a}\left[\frac{b_i^{(l)}[n]\mathbf{x}^{(l-1)T}[n]\mathbf{q}_i^{(l)}[n-1] + \alpha m_1 a e_{a,i}^{(l)}[n]}{a + b_i^{(l)}(n)\|\mathbf{x}^{(l-1)}[n]\|^2}\right]$

$\mathbf{q}_i^{(l)}[n] = \frac{1}{a}\mathbf{q}_i^{(l)}[n-1] - c_i^{(l)}[n]\mathbf{x}^{(l-1)}[n]$

$\mathbf{w}_i^{(l)}[n] = \mathbf{w}_i^{(l)}[n-1] + \alpha\mathbf{q}_i^{(l)}[n]$

TabelaI

ALGORITMO ACELERADOR PARA TREINAMENTO DE REDES DO TIPO MLP.

Essa condição, como pode ser verificado em [2], resulta da análise de estabilidade, segundo o critério de Liapunov, do algoritmo de tempo contínuo (equações (8) e (9)). Para o caso do algoritmo acelerador com a não-linearidade $\varphi(\cdot)$ é necessário também que esta restrição seja satisfeita. Estudos preliminares utilizando linearização indicam que no caso não-linear outras restrições não são necessárias para garantir a convergência. Entretanto, deve-se lembrar que o algoritmo é utilizado no ajuste dos pesos de uma rede em que erros são propagados regressivamente e não somente no ajuste dos pesos de um neurônio. Assim, da mesma forma que para os algoritmos BP e RLS-MLP, uma escolha criteriosa dos parâmetros e das condições iniciais é muito importante para o bom funcionamento do algoritmo.

Em [2] foi feita uma análise da velocidade de convergência e verificou-se que ela é maior quando os parâmetros satisfazem $4m_1^2 m_2 m_3 = 1$ e, dentro de certos limites, é tanto maior quanto maiores forem os valores de m_1 e α, sendo α o passo de adaptação. Este comportamento se repete quando se trata do algoritmo para treinamento de redes neuronais do tipo MLP.

Os pesos e os níveis de ajuste (*bias*) devem ser iniciali-

zados com números aleatórios uniformemente distribuídos, assumindo pequenos valores (por exemplo da ordem de 10^{-3}). Cabe observar que na formulação utilizada os níveis de ajuste (*bias*) estão incluídos no vetor de pesos. Os vetores $\mathbf{q}$ e $\mathbf{w}$ do instante anterior ao inicial devem ser inicializados com vetores nulos: $\mathbf{q}_i^{(l)}(-1) = \mathbf{w}_i^{(l)}(-1) = \mathbf{0}$.

Operações	$N^{\underline{o}}$ de operações
$\times$	$NA + 13NC - N_1$
$\div$	NC
$+$	$NB + 4NC + 2N_L$
NL	$3NC$

TabelaII

COMPLEXIDADE COMPUTACIONAL DO ALGORITMO ACELERADOR NA FASE DE TREINAMENTO DE UMA REDE MLP ($N_1, N_2, \ldots, N_L$), SENDO $NA = 7N_0N_1 + 8(N_1N_2 + N_2N_3 + \ldots + N_{L-1}N_L)$, $NB = 6N_0N_1 + 7(N_1N_2 + N_2N_3 + \ldots + N_{L-1}N_L)$, $NC = N_1 + N_2 + \ldots + N_L$.

5. RESULTADOS EXPERIMENTAIS

Com o objetivo de se comparar o algoritmo de treinamento aqui apresentado com o conhecido *Backpropagation* (BP) [6] e o algoritmo de treinamento de redes MLP baseado no método dos mínimos quadrados recursivo (RLS-MLP) [4], utilizou-se como aplicação a equalização de canais de comunicação.

Considerou-se o modelo de canal de comunicação com resposta $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ com o objetivo de se reconstruir sinais binários (0 e 1), equiprováveis e estatisticamente independentes, representados pelos níveis -1 e $+1$ (2-PAM). O sinal de saída do canal $y(k)$ é corrompido na entrada do receptor por ruído aditivo $n(k)$, gaussiano, branco, de média zero e variância σ_n^2. A partir deste sinal, um equalizador baseado numa rede neuronal MLP procura reconstituir o sinal originalmente transmitido. Maiores detalhes sobre este esquema de equalização podem ser encontrados em [5].

Foi considerada uma rede MLP de 4 camadas com a configuração $(5, 9, 3, 1)$ e 5 entradas (4 atrasadores na entrada da rede). Além disso, utilizou-se um atraso de três amostras para obter-se o sinal desejado a partir do sinal transmitido. Para uma relação sinal-ruído (SNR) de 20 dB foram obtidas curvas de erro quadrático médio (MSE) para os algoritmos de treinamento BP (*Backpropagation*), AC (acelerador) e RLS-MLP [4] (Figura 3). Na Figura 3, pode-se observar que a velocidade de convergência do algoritmo BP é bastante lenta. Na verdade, o que acontece é que este algoritmo é sensível às condições iniciais, o que torna sua convergência dependente das mesmas. Já o algoritmo acelerador e o RLS-MLP não apresentam este tipo de problema, talvez pelo fato de seus passos de adaptação não

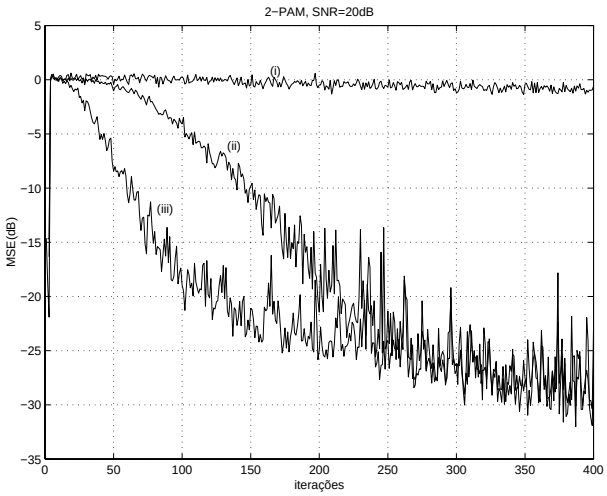

Fig. 3. Curvas de MSE (média de 100 realizações). (i)BP ($\eta = 0,1$; $\alpha = 0,3$). (ii)AC ($\alpha = 0,1$; $m_1 = 1,5$; $m_2 = 0,2100$; $m_3 = 0,5291$). (iii)RLS-MLP ($\lambda = 0,995$; $\delta = 0,5$).

serem fixos [6]. Na Figura 4, são mostrados os sinais de saída dos equalizadores durante o treinamento com os algoritmos BP, AC e RLS-MLP para SNR=20 dB. Nesta figura, considerou-se uma situação em que houve convergência rápida do algoritmo BP, devendo-se ressaltar que mesmo nestas condições ele resultou mais lento que os outros dois algoritmos. Nestes experimentos o passo de aprendizagem

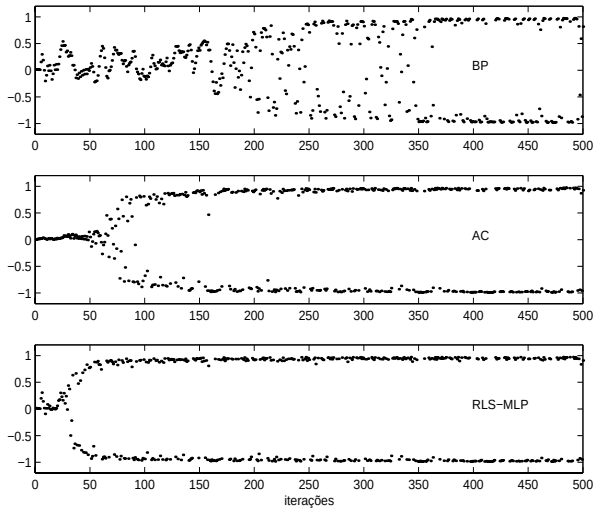

Fig. 4. Sinais de saída dos equalizadores treinados com os algoritmos BP ($\eta = 0,1$; $\alpha = 0,3$), AC ($\alpha = 0,1$; $m_1 = 1,5$; $m_2 = 0,2100$; $m_3 = 0,5291$) e RLS-MLP ($\lambda = 0,995$; $\delta = 0,5$) para SNR=20 dB.

do algoritmo *Backpropagation* foi feito igual ao do acelerador sendo que os demais parâmetros deste último foram escolhidos conforme descrito a seguir.

De [2], sabe-se que a velocidade de convergência do algoritmo acelerador é máxima quando os parâmetros m_1,

m_2 e m_3 estão próximos da condição de estabilidade ($4m_1^2 m_2 m_3 \approx 1$). Além disso, dentro de certos limites a velocidade de convergência é diretamente proporcional a m_1 e α. Mais precisamente, considerando um passo de aprendizagem (α) fixo e as condições $4m_1^2 m_2 m_3 = 1$ e $m_2 = m_3$, não se consegue aumentar a velocidade de convergência do algoritmo após um certo limite máximo para m_1, sendo que de certa forma a velocidade de convergência fica limitada pelo passo de aprendizagem. Além disso, em situações em que a relação sinal-ruído é inferior a 20 dB, valores muito grandes de m_1 provocam uma maior variância do erro médio quadrático, o que é indesejável. Dessa forma, um procedimento que se mostrou útil na prática foi o seguinte:

- usar m_2 com a mesma ordem de grandeza de m_3, podendo inclusive serem iguais a $1/2m_1$;
- fixar um valor de α;
- achar um valor de m_1 que garanta a maior velocidade de convergência sem provocar grandes variações do erro.

Para comparar a velocidade de convergência dos três algoritmos de treinamento considerados, é necessário tomar parâmetros compatíveis. Na falta de uma equivalência teórica entre os algoritmos, os parâmetros utilizados foram obtidos de forma experimental. Para o algoritmo RLS-MLP, considerou-se um fator de esquecimento $\lambda = 0.995$. Na presença de ruído, há algumas situações em que este algoritmo diverge para valores de fator de esquecimento menores. Para o algoritmo BP, considerou-se um passo de aprendizagem $\eta = 0,1$, já que valores maiores levavam o algoritmo mais próximo a situações de divergência e não se observaram modificações significativas no resultado apresentado na Figura 3. Isto se deve a sua maior sensibilidade às condições iniciais e ao fato de na Figura 3 ser apresentada a média de 100 realizações. Para o algoritmo AC, considerou-se $\alpha = 0,1$, de tal forma que o passo de adaptação desse algoritmo fosse o mesmo que o do algoritmo BP. Os parâmetros m_1, m_2 e m_3 foram escolhidos para satisfazer a condição $4m_1^2 m_2 m_3 = 1$ e evitar uma excessiva variância do erro médio quadrático.

É interessante observar que, mesmo com um fator de esquecimento (λ) maior que $1 - \eta$ (BP) ou $1 - \alpha$ (AC), a velocidade de convergência do algoritmo RLS-MLP resultou maior que a dos algoritmos BP ou AC.

A Tabela III mostra o número de operações por iteração dos três algoritmos aplicados ao treinamento da rede MLP utilizada nos experimentos (configuração $(5, 9, 3, 1)$). O número de operações por iteração dos algoritmos BP e RLS-MLP foi calculado a partir dos algoritmos apresentados em [6] e [4] respectivamente. Como se pode observar na Tabela III, o algoritmo acelerador apresenta um moderado aumento da complexidade computacional em relação ao algoritmo *Backpropagation*. Já o número de multiplicações e somas do algoritmo RLS-MLP são respectivamente 3,8 vezes e 2,5 vezes maiores do que os correspondentes valores do algoritmo AC. Diante destes valores o uso do algoritmo RLS-MLP praticamente não se justifica, embora apresente uma velocidade de convergência um pouco maior.

Algoritmo	BP	AC	RLS-MLP
$\times$	465	1004	3858
$\div$	-	18	18
$+$	395	749	1917
NL	36	54	36

TabelaIII
NÚMERO DE OPERAÇÕES POR ITERAÇÃO DE CADA ALGORITMO DE TREINAMENTO PARA UMA REDE MLP (5,9,3,1).

6. CONCLUSÃO

Neste trabalho foi introduzido um algoritmo de treinamento para redes neuronais MLP do tipo acelerador. Considerando um problema de equalização, o algoritmo em questão foi comparado por meio de simulações aos algoritmos *Backpropagation* e RLS-MLP. Ele mostrou um desempenho intermediário entre esses dois algoritmos, sendo nitidamente menos complexo que o algoritmo RLS-MLP e bem mais rápido que o algoritmo *Backpropagation*.

7. REFERÊNCIAS

[1] PAIT, F. M. A tuner that accelerates parameters. *Systems and Control Letters*, v.35, p. 65-68, 1998.

[2] JOJOA, P. E.; GERKEN, M.; PAIT, F. Um algoritmo acelerador de parâmetros. In: SIMPóSIO BRASILEIRO DE TELECOMUNICAÇÕES, 17., Vila Velha, 1999. *Anais*. Vila Velha, Sociedade Brasileira de Telecomunicações / Universidade Federal do Espírito Santo, 1999. p.197-202.

[3] GERKEN, M.; PAIT, F.; JOJOA, P. E. An Adaptive Filtering Algorithm with Parameter Acceleration. In: PROCEEDINGS OF THE IEEE ICASSP'2000, Istambul, Turquia, 5 a 9 de julho de 2000.

[4] BILSKI, J.; RUTKOWISKI, L. A fast training algorithm for neural networks. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, v.45, n.6, p.749-753, June 1998.

[5] GIBSON, G. J.; COWAN, C. F. N.; GRANT, P. M. On the decision regions of multilayer perceptrons. *Proceedings of IEEE*, v.78, p.1590-1599, 1990.

[6] HAYKIN, S. *Neural Networks*. 2.ed. New Jersey, Prentice Hall, 1999.