PERGAMON

Computers and Electrical Engineering 28 (2002) 587–598

Computers and Electrical Engineering

www.elsevier.com/locate/compeleceng

An on-line hybrid learning algorithm for multilayer perceptron in identification problems

Daohang Sha, Vladimir B. Bajić *,[1]

Centre for Engineering Research, Technikon Natal, P.O. Box 953, Durban 4000, South Africa

Received 3 December 1999; received in revised form 4 September 2000; accepted 6 September 2000

Abstract

A hybrid learning algorithm for multilayered perceptrons (MLPs) and pattern-by-pattern training, based on optimized instantaneous learning rates and the recursive least squares method, is proposed. This hybrid solution is developed for on-line identification of process models based on the use of MLPs, and can speed up the learning process of the MLPs substantially, while simultaneously preserving the stability of the learning process. For illustration and test purposes the proposed algorithm is applied to the identification of a non-linear dynamic system.

Keywords: Soft computing; Neural networks; Hybrid learning; Gradient descent method; Recursive least squares parameter estimation

1. Introduction

Multilayer perceptrons (MLPs) are the most popular class of artificial neural networks (ANNs) which have been widely applied to pattern recognition, time series prediction, non-linear control, identification problems, etc. The basic learning rule of MLPs is based on the gradient descent (GD) optimization method and the chain rule, as initially proposed by Werbos [1] in the 1970s. Since the basic learning rule is based on the GD method, which is known for its slowness and its frequent confinement to local minima [2], other faster training algorithms for MLPs have been developed in an attempt to reduce these shortcomings (for surveys on these results, see Refs. [3–6]). The faster learning algorithms were developed for both batch training and pattern-by-pattern

* Corresponding author. Tel./fax: +27-31-204-2560.
E-mail addresses: dhsha@hotmail.com (D. Sha), bajic.v@cer.co.za (V.B. Bajić).
[1] http://www.cer.co.za.

0045-7906/02/$ - see front matter
PII: S0045-7906(00)00063-X

(on-line, or sometimes called stochastic) training, as well as for global and for local learning rate (LR) change. In the global LR change only one LR is used for updates of all network parameters, whilst in the local LR change methods, the LRs can be different for each of the network parameters. Most of the adaptive LR methods are developed for batch type training (see Refs. [3–6]). Since the purpose of this paper is to develop an efficient on-line ANN based identification method, our attention is focused on pattern-by-pattern training with a global LR change, due to its direct and simple applicability to on-line identification problems. There are, in fact, very few results that provide adaptive change of the LRs for pattern-by-pattern learning [7–12]. Generally, all these algorithms have an improved convergence property, but most of these methods do not use the optimized instantaneous LRs. Of all pattern-by-pattern training methods that use the adaptive change of the LRs, only [12] uses their instantaneous optimized values.

There are other ways to accelerate the MLPs learning by the use of second-order gradient based non-linear optimizing methods, such as the conjugate gradient algorithm (see Ref. [13]) or the Levenberg–Marquardt based method (see Ref. [14]). The crucial drawbacks of these methods, however, are that in many applications computational demands are so large that their effective use in on-line identification problems is not viable.

Recently, a layer-by-layer (LBL) optimizing algorithm was proposed in which each layer of the MLPs is decomposed into both a linear part and a non-linear part [15–18]. The linear part of each layer is solved via the least squares problem formulation. These algorithms show fast convergence with reduced computational complexity, compared to algorithms based on the conjugate-gradient or Newton methods. However, if the targets for the hidden layer cannot be linearly separated, then the mean squared error cannot be sufficiently reduced at both the hidden layer and the output layer. To overcome this problem, a new error function at the hidden layers was proposed for the fast training of the MLPs [19]. With this method, the LBL algorithm only approximately converges to the error backpropagation algorithm with the optimum LRs. Another class of fast learning algorithms is based on the extended Kalman filter technique for training MLPs (see Refs. [20–23]). They have an improved convergence rate and show good performance. However, their numerical stability is not guaranteed and may degrade learning convergence, increase the training time and, generally, this can make potential on-line implementation questionable [23]. An additional problem is that most of these algorithms are developed only for off-line training of the ANNs.

In this paper we propose an on-line hybrid learning algorithm that combines the optimized instantaneous LR of GD and the recursive least squares (RLS) method. This development is motivated by the fact that many output layers of MLPs are linear. Thus, it seems reasonable to employ the RLS technique to tune the parameters of the output layer during training. This development is based on Ref. [26], in which we developed an on-line optimized instantaneous LR algorithm for training an MLP, based on (a) the concept of a hybrid learning rule proposed by Jang [24,25] and (b) our work on optimized instantaneous LRs for MLPs in identification problems [12]. This class of hybrid learning can speed up the learning process substantially and, simultaneously, enhance its stability.

The emphasis of this paper is on on-line identification. Thus, since ANNs are widely used for the identification and prediction of non-linear systems (see Refs. [27–29]), our hybrid learning algorithm is tested on the identification problem and used in the prediction of the behavior of non-linear systems.

2. Model of multilayer perceptron

In what follows, we will adopt a compact matrix–vector notation of the network description in order to be able to express later by simple formula the optimized instantaneous LR. In what follows q^{T} denotes the transpose of a matrix or a vector q, while q' stands for a partial derivative of q. Let IN and HN stand for the number of input nodes and the number of hidden layer neurons, respectively. Denote by x, v, and W the following

$$x = [\,x_0 \quad x_1 \quad \cdots \quad x_{\mathrm{IN}}\,]^{\mathrm{T}} \in R^{(\mathrm{IN}+1)\times 1},$$

$$v = [\,v_0 \quad v_1 \quad \cdots \quad v_{\mathrm{HN}}\,]^{\mathrm{T}} \in R^{(\mathrm{HN}+1)\times 1},$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{\mathrm{HN}} \end{bmatrix} = \begin{bmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,\mathrm{IN}} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,\mathrm{IN}} \\ \vdots & \vdots & \vdots & \vdots \\ w_{\mathrm{HN},0} & w_{\mathrm{HN},1} & \cdots & w_{\mathrm{HN},\mathrm{IN}} \end{bmatrix} \in R^{\mathrm{HN}\times(\mathrm{IN}+1)}.$$

The activation function for non-linear (hidden layer) nodes is a symmetric hyperbolic tangent function, i.e. $s(x) = \tanh(u_0^{-1}x)$, and its derivative is $s'(x) = u_0^{-1}[1 - s^2(x)]$, where u_0 is the shape factor of s.

Consider a three layer MLP shown in Fig. 1, and assume its model is given by

$$y_{\mathrm{NN}} = \sum_{i=1}^{\mathrm{HN}} s\left(\sum_{j=1}^{\mathrm{IN}} w_{i,j}x_j + w_{i,0}\right)v_i + v_0 = \sum_{i=0}^{\mathrm{HN}} s(\mathrm{net}_i)v_i, \tag{1}$$

where $\mathrm{net}_i = \sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j = w_i x$ is the output of the ith hidden node, with $x_0 \equiv 1$, for $i = 0, 1, \ldots,$ HN, and with $s(\mathrm{net}_0) \equiv 1$. Here x_j, $j = 1, \ldots, \mathrm{IN}$, are inputs of the neural network, while y_{NN} is the output of the network. Weights from the input layer to the hidden layer are $w_{i,j}$, $i = 1, \ldots, \mathrm{HN}$, $j = 1, \ldots, \mathrm{IN}$, and the biases of the hidden nodes are $w_{i,0}$, $i = 1, \ldots, \mathrm{HN}$. The weights from the hidden layer to the output layer are v_i, $i = 1, \ldots, \mathrm{HN}$, while v_0 is the bias of the output node. The output layer is linear and we assume that it has only one output node. Further, Eq. (1) can be rewritten as

$$y_{\mathrm{NN}}(k+1) = v^{\mathrm{T}}(k)S(W(k)x(k)) \tag{2}$$

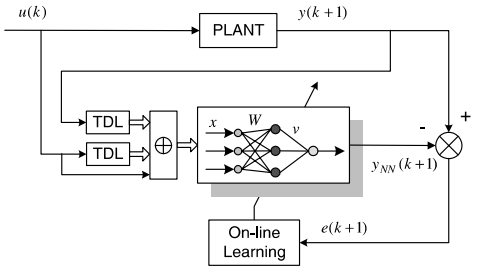

Fig. 1. Identification structure for non-linear systems based on MLP.

where

$$S(Wx) = [\, s(\mathrm{net}_0) \quad s(\mathrm{net}_1) \quad \cdots \quad s(\mathrm{net}_{\mathrm{HN}}) \,]^{\mathrm{T}} \in R^{(\mathrm{HN}+1)\times 1}.$$

3. Optimized instantaneous learning rate algorithm

The algorithm for determining the optimized instantaneous LR is based on pattern-by-pattern learning. Since our method is aimed at on-line identification, the presentation of input patterns is associated with the sampling values of signals used to train the MLP. This means that each pattern to be presented to the MLP is associated with the current value of time, i.e. with the sampling moment. Thus, to simplify notation, instead of writing t_k for the sampling moment, we will use only index k, having in mind that it relates to t_k. In what follows the notation $[a]_k$ indicates that the expression a is evaluated at the moment k. Also, let Δ be an increment operator with the meaning $\Delta a(k) = a(k+1) - a(k)$ for some variable a.

Let us define the error function for pattern-by-pattern learning as

$$J(k) = \tfrac{1}{2}e^2(k) = \tfrac{1}{2}[y(k) - y_{\mathrm{NN}}(k)]^2.$$

By applying the GD method to J and using Lemma 2 (see Appendix A), one obtains

$$\Delta v(k) = -\eta(k)\frac{\partial J(k)}{\partial v(k)} = \eta(k)\frac{\partial y_{\mathrm{NN}}(k)}{\partial v(k)}e(k) = [\eta S(Wx)e]_k. \tag{3}$$

In the same way, using Lemma 3 (see Appendix A), one gets

$$\Delta W(k) = -\eta(k)\frac{\partial J(k)}{\partial W(k)} = \eta(k)\frac{\partial y_{\mathrm{NN}}(k)}{\partial W(k)}e(k) = \left[\eta\overline{S'}(Wx)\bar{v}x^{\mathrm{T}}e\right]_k. \tag{4}$$

where

$$\bar{v} = [\, v_1 \quad \cdots \quad v_{\mathrm{HN}} \,]^{\mathrm{T}} \in R^{\mathrm{HN}\times 1},$$

$$\overline{S'}(Wx) = \mathrm{diag}\big[\, s'(\mathrm{net}_1) \quad s'(\mathrm{net}_2) \quad \cdots \quad s'(\mathrm{net}_{\mathrm{HN}}) \,\big],$$

and where $s'(\mathrm{net}_i) = \partial s(\mathrm{net}_i)/\partial \mathrm{net}_i$, for $i = 1, \ldots, \mathrm{HN}$. In Eqs. (3) and (4), η represents the instantaneous positive LR. We aim at finding its optimized value for each sampling moment.

In order to determine the optimized value for η we consider the error increment

$$e(k+1) - e(k) = y(k+1) - y_{\mathrm{NN}}(k+1) - y(k) + y_{\mathrm{NN}}(k).$$

Assume that $\Delta y(k) = y(k+1) - y(k)$ is such that $|\Delta y(k)| \ll |\Delta y_{\mathrm{NN}}(k)|$, i.e. that the absolute value of the change of the process output is much smaller than the absolute value of the change of the ANN output. This implies that the value $y(k)$ can approximate $y(k+1)$ locally during the training process. This assumption is realistic for many processes because of energy constraints in practical systems, while no constraints are imposed to the output of the ANN. Also, if this condition is not satisfied, then the ANN identification system will not be able to adapt sufficiently fast to the change in the process output, and no identification of the process model will be possible. With the above-mentioned assumption and using Lemma 4 (see Appendix A), the error increment can be approximated as follows

$$
\begin{aligned}
e(k+1)-e(k) &= \Delta y(k)-\Delta y_{\mathrm{NN}}(k) \approx -\Delta y_{\mathrm{NN}}(k) \approx -\left[\frac{\mathrm{d}y_{\mathrm{NN}}}{\mathrm{d}t}\right]_k \Delta t \\
&= -\left[S^{\mathrm{T}}\frac{\mathrm{d}v}{\mathrm{d}t}+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\frac{\mathrm{d}W}{\mathrm{d}t}x\right]_k \Delta t \approx -\left[S^{\mathrm{T}}\frac{\Delta v}{\Delta t}+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\frac{\Delta W}{\Delta t}x\right]_k \Delta t \\
&= -\left[S^{\mathrm{T}}(Wx)\Delta v+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\Delta Wx\right]_k .
\end{aligned}
$$

This, together with Eqs. (3) and (4), gives

$$
\begin{aligned}
e(k+1)-e(k) &\approx -\left[S^{\mathrm{T}}(Wx)\eta S(Wx)e+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\eta\overline{S'}(Wx)\bar{v}x^{\mathrm{T}}ex\right]_k \\
&= -\eta(k)\left[S^{\mathrm{T}}(Wx)S(Wx)+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\overline{S'}(Wx)\bar{v}x^{\mathrm{T}}x\right]_k e(k) = -\eta(k)\zeta(k)e(k),
\end{aligned}
$$

i.e. the approximate error dynamics is governed by

$$
e(k+1) \approx [1-\eta(k)\zeta(k)]e(k), \tag{5}
$$

where $\zeta(k)=[S^{\mathrm{T}}(Wx)S(Wx)+\bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\overline{S'}(Wx)\bar{v}x^{\mathrm{T}}x]_k$. In order to ensure $\lim_{k\to\infty} e(k)=0$, the condition $|1-\eta(k)\zeta(k)|<1$ has to be satisfied, giving $0<\eta(k)<2\zeta^{-1}(k)$. Apparently the upper range of the LR η is variable because the value of $\zeta(k)$ depends on the input x and the network parameters v and W. The expression for the optimized instantaneous LR η is determined from the following consideration. If the optimized instantaneous LR, $\eta_{\mathrm{opt}}(k)$, is applied in the update of network parameters at the moment t_k, then, if the current input pattern is presented to the MLP with the updated parameters, the estimated error $\hat{e}(k+1)$ at the next moment t_{k+1} to be produced by the MLP should be zero, i.e.

$$
\hat{e}(k+1)=[1-\eta_{\mathrm{opt}}(k)\zeta(k)]e(k)=0.
$$

This implies

$$
\eta_{\mathrm{opt}}(k)=\zeta^{-1}(k). \tag{6}
$$

Note that this η_{opt} satisfies the condition $0<\eta_{\mathrm{opt}}(k)<2\zeta^{-1}(k)$, which implies the convergence of the approximate error dynamics (5).

Finally, the increments of the network parameters, by using the optimized LR, are obtained by replacing the η_{opt} value given by Eq. (6) to Eqs. (3) and (4), which yield

$$
\Delta v(k)=\left[\zeta^{-1}S(Wx)e\right]_k, \tag{7}
$$

$$
\Delta W(k)=\left[\zeta^{-1}\overline{S'}(Wx)\bar{v}x^{\mathrm{T}}e\right]_k . \tag{8}
$$

4. The hybrid learning algorithm with optimized LRs for updating W and RLS for updating v

Although we can apply the GD method to identify all the weights of the considered MLP using the optimized instantaneous LR, such as in Ref. [12] and in Section 3, we can further speed up the

network parameter determination by using a combination of (a) the optimized LR method for the update of W, which appears in a non-linear way in the learning model, and (b) the least squares error estimate, to identify v of the MLP that appears linearly in the output layer part of the model. This combination is based on the hybrid learning rule proposed in Refs. [24,25].

If the value of W is known, since we also know the target network output, i.e. the output of the process, the linear parameters v can be obtained by the well-known RLS algorithm [30]. Each iteration of this hybrid learning procedure is composed of a forward pass and a backward pass. In the forward pass, we supply input data to the network, and the functional signals progress forward toward the network output, so that we can calculate the output of each of the hidden neurons until $[S(Wx)]_k$ is obtained. Since the target value at t_k of the MLPs output is equal to the current value of the process output, $y(k)$, one has a linear equation $y(k) = [S^{\mathrm{T}}(Wx)]_k\, v$ from which v can be determined using the RLS calculation [30]. After determining v at t_k, the functional signals keep progressing forward until the output error e is calculated. In the backward pass, the error propagates from the output towards the input end, and the parameters in W are updated according to Eq. (8).

By this hybrid learning algorithm the dimension of the parameter search space in the GD method is reduced, and, in general, in the case of a multi-output MLP, it will also reduce substantially the convergence time.

5. Simulation

Let us consider a nonlinear process used for the simulation of a spring–mass–damper system containing a hardening spring in Ref. [31], governed by

$$y''(t) + y'(t) + y(t) + y^3(t) + F_L = u(t), \tag{9}$$

where u is the system input, y is the system output, F_L is the external disturbance. We will assume that no disturbance is present, i.e. $F_L = 0$. In Ref. [12], the advantages of the optimized instantaneous LR over the different fixed LR values were demonstrated in the identification problems, and therefore will not be repeated here. We will only compare the hybrid and non-hybrid learning with the optimized LRs as applied to the non-linear model identification and model output prediction. We select an MLP in the identification structure of Fig. 1, as a three layer network. The number of nodes of the input layer is four; two nodes are from the input of the MLP through the tapped delay line (TDL) block; the others are from the output of the non-linear system passing through another TDL block. The hidden layer has six sigmoid neurons. The input u for both the non-linear system and the MLP is taken as a random input signal. The target for the MLP is the output y of the simulated non-linear system (9). The sampling interval is $T_s = 0.1$ s. The total simulation time is 250 s, i.e. we have 2500 sampling intervals. The MLP is trained during the first 2000 sampling intervals. After the training stage, i.e. from the time instant of 200 s, the trained MLP is used to predict the output of the non-linear system during the subsequent 500 sampling intervals. The time evolution of the variance of the training error of the MLP, during training under two different sets of initial values, are plotted in Figs. 2 and 3. These results are compared with the non-hybrid learning algorithm that uses the optimized instantaneous LR. We carried out a large number of simulation experiments using different initial values of the parameters of the MLP and we obtained very similar results in all cases. This indicates that for the system con-

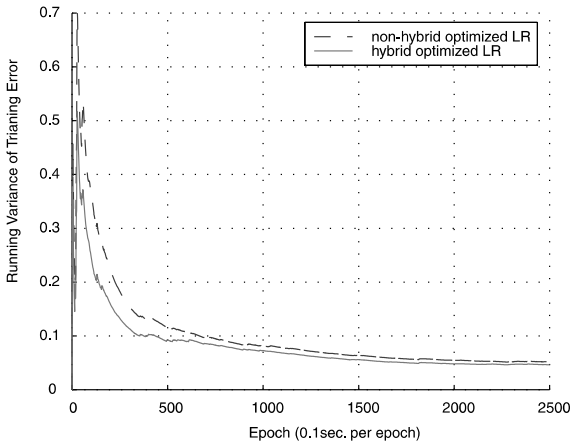

Fig. 2. Evolution of the training error variance.

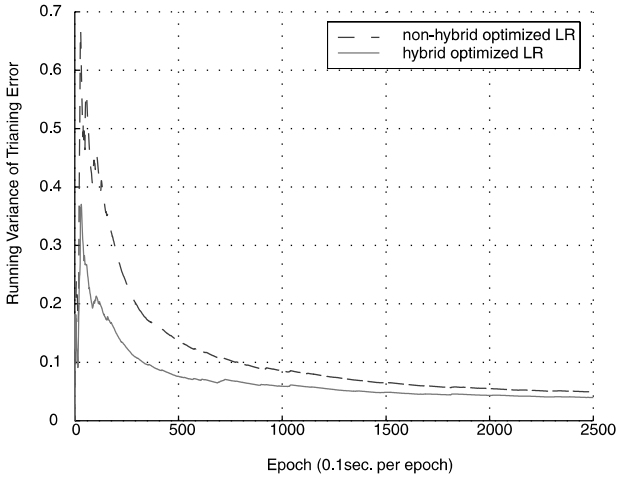

Fig. 3. Evolution of the training error variance with a different set of initial values.

sidered the hybrid learning algorithm, with the optimized LR for W and RLS estimates for v, has performed much better (as expected), in terms of the learning speed and the training error, than the optimized LR algorithm applied in the update of all network parameters.

6. Conclusion

By combining the optimized instantaneous LR algorithm for determining input-to-hidden layer parameters of the MLP with the RLS estimation of the hidden-to-output layer parameters, a new efficient on-line hybrid learning algorithm for MLPs is obtained. This learning algorithm is developed specifically for the implementation of on-line identification problems. It is demonstrated, through the identification of a non-linear dynamic system by simulation, that the proposed

method may perform much better, in terms of the learning speed and the training error, than non-hybrid learning with the optimized instantaneous LR.

Acknowledgements

The authors acknowledge the constructive comments of the reviewers.

Appendix A

Lemma 1

$$\overline{S'}(Wx) = \operatorname{diag}\begin{bmatrix} s'(\mathrm{net}_1) & \cdots & s'(\mathrm{net}_{\mathrm{HN}}) \end{bmatrix} = \frac{1}{u_0}\begin{bmatrix} 1-s^2(w_1x) & 0 & \cdots & 0 \\ 0 & 1-s^2(w_2x) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-s^2(w_{\mathrm{HN}}x) \end{bmatrix},$$

where

$$\overline{S}(Wx) = \begin{bmatrix} s(\mathrm{net}_1) \\ \vdots \\ s(\mathrm{net}_{\mathrm{HN}}) \end{bmatrix},$$

and $s'(\mathrm{net}_i) = \partial s(\mathrm{net}_i)/\partial \mathrm{net}_i$, $i = 1, \ldots, \mathrm{HN}$, $s(x) = \tanh(u_0^{-1}x)$, *with its derivative* $s'(x) = u_0^{-1}[1 - s^2(x)]$.

Proof. Direct calculation of the Jacobian of $\overline{S}(Wx)$ produces

$$\overline{S'}(Wx) = \begin{bmatrix} \frac{\partial s(\mathrm{net}_1)}{\partial \mathrm{net}_1} & \cdots & \frac{\partial s(\mathrm{net}_1)}{\partial \mathrm{net}_{\mathrm{HN}}} \\ \vdots & \vdots & \vdots \\ \frac{\partial s(\mathrm{net}_{\mathrm{HN}})}{\partial \mathrm{net}_1} & \cdots & \frac{\partial s(\mathrm{net}_{\mathrm{HN}})}{\partial \mathrm{net}_{\mathrm{HN}}} \end{bmatrix} = \begin{bmatrix} \frac{\partial s(\mathrm{net}_1)}{\partial \mathrm{net}_1} & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{\partial s(\mathrm{net}_{\mathrm{HN}})}{\partial \mathrm{net}_{\mathrm{HN}}} \end{bmatrix}$$

$$= \begin{bmatrix} s'(\mathrm{net}_1) & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & s'(\mathrm{net}_{\mathrm{HN}}) \end{bmatrix} = \operatorname{diag}\begin{bmatrix} s'(\mathrm{net}_1) & \cdots & s'(\mathrm{net}_{\mathrm{HN}}) \end{bmatrix}$$

$$= \begin{bmatrix} u_0^{-1}[1-s^2(\mathrm{net}_1)] & 0 & \cdots & 0 \\ 0 & u_0^{-1}[1-s^2(\mathrm{net}_2)] & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & u_0^{-1}[1-s^2(\mathrm{net}_{\mathrm{HN}})] \end{bmatrix}$$

$$= \frac{1}{u_0}\begin{bmatrix} 1-s^2(w_1x) & 0 & \cdots & 0 \\ 0 & 1-s^2(w_2x) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-s^2(w_{\mathrm{HN}}x) \end{bmatrix}. \qquad \square$$

Lemma 2. *Derivative of the ANN output y_{NN} with respect to v is obtained by*

$$\frac{\partial y_{NN}}{\partial v} = S(Wx).$$

Proof. Since $y_{NN} = v^T S(Wx)$ is a scalar, its derivative with respect to the vector v is

$$\frac{\partial [v^T S(Wx)]}{\partial v} = \left[\frac{\partial [v^T S(Wx)]}{\partial v_i}\right]_{(HN+1)\times 1} = [s(\text{net}_i)]_{(HN+1)\times 1} = S(Wx). \qquad \square$$

Lemma 3. *Partial derivatives of the ANN output y_{NN} with respect to the matrix W is given by*

$$\frac{\partial y_{NN}}{\partial W} = \overline{S'}(Wx)\bar{v}x^T.$$

Proof. Using $y_{NN} = v^T S(Wx)$ and making a derivative of a scalar function with regard to a matrix, one obtains

$$\frac{\partial [v^T S(Wx)]}{\partial W} = \left[\frac{\partial [v^T S(Wx)]}{\partial w_{ij}}\right]_{HN\times(IN+1)} = \left[\frac{\partial \left[\sum_{i=0}^{HN} v_i s\left(\sum_{j=0}^{IN} w_{ij}x_j\right)\right]}{\partial w_{ij}}\right]_{HN\times(IN+1)}$$

$$= \left[v_i s'(\text{net}_i)x_j\right]_{HN\times(IN+1)}$$

$$= \begin{bmatrix} v_1 s'(\text{net}_1)x_0 & v_1 s'(\text{net}_1)x_1 & \cdots & v_1 s'(\text{net}_1)x_{IN} \\ \vdots & \vdots & \vdots & \vdots \\ v_{HN} s'(\text{net}_{HN})x_0 & v_{HN} s'(\text{net}_{HN})x_1 & \cdots & v_{HN} s'(\text{net}_{HN})x_{HN} \end{bmatrix}$$

$$= \begin{bmatrix} s'(\text{net}_1) & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & s'(\text{net}_{HN}) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_{HN} \end{bmatrix} [x_0 \quad x_1 \quad \cdots \quad x_{IN}] = \overline{S'}(Wx)\bar{v}x^T. \qquad \square$$

Lemma 4. *Total time derivative of the ANN output y_{NN} is given by*

$$\frac{dy_{NN}}{dt} = S^T(Wx)\frac{dv}{dt} + \bar{v}^T\overline{S'}(Wx)\frac{dW}{dt}x = \frac{dv^T}{dt}S(Wx) + \bar{v}^T\,\overline{S'}(Wx)\frac{dW}{dt}x.$$

Proof. Derivation of $\mathrm{d}y_{\mathrm{NN}}/\mathrm{d}t$ is as follows:

$$\begin{aligned}
\frac{\mathrm{d}y_{\mathrm{NN}}}{\mathrm{d}t} &= \frac{\mathrm{d}[v^{\mathrm{T}}S(Wx)]}{\mathrm{d}t} = \frac{\mathrm{d}\left[\sum_{i=0}^{\mathrm{HN}} v_i s\left(\sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j\right)\right]}{\mathrm{d}t} \\
&= \sum_{i=0}^{\mathrm{HN}} \frac{\partial\left[\sum_{i=0}^{\mathrm{HN}} v_i s\left(\sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j\right)\right]}{\partial v_i}\frac{\mathrm{d}v_i}{\mathrm{d}t} + \sum_{i=0}^{\mathrm{HN}}\sum_{j=0}^{\mathrm{IN}} \frac{\partial\left[\sum_{i=0}^{\mathrm{HN}} v_i s\left(\sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j\right)\right]}{\partial w_{ij}}\frac{\mathrm{d}w_{i,j}}{\mathrm{d}t} \\
&= \sum_{i=0}^{\mathrm{HN}} s\left(\sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j\right)\frac{\mathrm{d}v_i}{\mathrm{d}t} + \sum_{i=0}^{\mathrm{HN}}\sum_{j=0}^{\mathrm{IN}} v_i s'\left(\sum_{j=0}^{\mathrm{IN}} w_{i,j}x_j\right)x_j\frac{\mathrm{d}w_{i,j}}{\mathrm{d}t} \\
&= s(\mathrm{net}_0)\frac{\mathrm{d}v_0}{\mathrm{d}t} + s(\mathrm{net}_1)\frac{\mathrm{d}v_1}{\mathrm{d}t} + \cdots + s(\mathrm{net}_{\mathrm{HN}})\frac{\mathrm{d}v_{\mathrm{HN}}}{\mathrm{d}t} + v_0 s'(\mathrm{net}_0)\left[x_0\frac{\mathrm{d}w_{0,0}}{\mathrm{d}t} + x_1\frac{\mathrm{d}w_{0,1}}{\mathrm{d}t}\right. \\
&\quad \left. + \cdots + x_{\mathrm{IN}}\frac{\mathrm{d}w_{0,\mathrm{IN}}}{\mathrm{d}t}\right] + v_1 s'(\mathrm{net}_1)\left[x_0\frac{\mathrm{d}w_{1,0}}{\mathrm{d}t} + x_1\frac{\mathrm{d}w_{1,1}}{\mathrm{d}t} + \cdots + x_{\mathrm{IN}}\frac{\mathrm{d}w_{1,\mathrm{IN}}}{\mathrm{d}t}\right] \\
&\quad + \cdots + v_{\mathrm{HN}} s'(\mathrm{net}_{\mathrm{HN}})\left[x_0\frac{\mathrm{d}w_{\mathrm{HN},0}}{\mathrm{d}t} + x_1\frac{\mathrm{d}w_{\mathrm{HN},1}}{\mathrm{d}t} + \cdots + x_{\mathrm{IN}}\frac{\mathrm{d}w_{\mathrm{HN},\mathrm{IN}}}{\mathrm{d}t}\right] \\
&= \left[s(\mathrm{net}_0) \quad s(\mathrm{net}_1) \quad \cdots \quad s(\mathrm{net}_{\mathrm{HN}})\right]\left[\tfrac{\mathrm{d}v_0}{\mathrm{d}t} \quad \tfrac{\mathrm{d}v_1}{\mathrm{d}t} \quad \cdots \quad \tfrac{\mathrm{d}v_{\mathrm{HN}}}{\mathrm{d}t}\right]^{\mathrm{T}} \\
&\quad + \left[v_1 \quad \cdots \quad v_{\mathrm{HN}}\right]\begin{bmatrix} s'(\mathrm{net}_1) & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & s'(\mathrm{net}_{\mathrm{HN}}) \end{bmatrix}\cdot\begin{bmatrix} \frac{\mathrm{d}w_{0,0}}{\mathrm{d}t} & \frac{\mathrm{d}w_{0,1}}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}w_{0,\mathrm{IN}}}{\mathrm{d}t} \\ \frac{\mathrm{d}w_{1,0}}{\mathrm{d}t} & \frac{\mathrm{d}w_{1,1}}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}w_{1,\mathrm{IN}}}{\mathrm{d}t} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\mathrm{d}w_{\mathrm{HN},0}}{\mathrm{d}t} & \frac{\mathrm{d}w_{\mathrm{HN},1}}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}w_{\mathrm{HN},\mathrm{IN}}}{\mathrm{d}t} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\mathrm{IN}} \end{bmatrix} \\
&= S^{\mathrm{T}}(Wx)\frac{\mathrm{d}v}{\mathrm{d}t} + \bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\frac{\mathrm{d}W}{\mathrm{d}t}x = \frac{\mathrm{d}v^{\mathrm{T}}}{\mathrm{d}t}S(Wx) + \bar{v}^{\mathrm{T}}\,\overline{S'}(Wx)\frac{\mathrm{d}W}{\mathrm{d}t}x. \qquad \square
\end{aligned}$$

References

[1] Widrow B, Lehr MA. 30 Years of adaptive neural networks: perceptron, madaline, and backpropagation. Neural Networks, I: Theory & Modeling (special issue), Proceedings of the IEEE, 1990;78(9):1415–42.

[2] Rumelhart DE, McClelland JL. Parallel distributed processing. Cambridge, MA: MIT Press; 1986.

[3] Moreira M, Fiesler E. Neural networks with adaptive learning rate and momentum terms. Technical report 95-04, Institut Dalle Molle d'Intelligence Artificielle Perceptive, Martigny, Switzerland, 1995.

[4] Schiffmann W, Joost M, Werner R. Comparison of optimized backpropagation algorithms. Proceedings of the European Symposium on Artificial Neural Networks, ESANN'93, Brussels, 1993. p. 97–104.

[5] Schiffmann W, Joost M, Werner R. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, University of Koblenz, Germany, 1994.

[6] Riedmiller M. Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms. Int J Comput Standards Interf 1994;16:265–78.

[7] Schmidhuber J. Accelerated learning in back-propagation nets, In: Pfeifer R, Schreter Z, Fogelman Z, Steels L, editors. Connectionism in perspective. Amsterdam: Elsevier; 1989. p. 429–38.
[8] Darken C, Moody J. Towards faster stochastic gradient search. In: Moody, Hanson, Lippmann, editors. Advances in neural information processing systems, vol. 4. San Mateo: Morgan Kaufmann; 1992. p. 1009–16.
[9] Darken C, Moody J. Learning rate schedules for faster stochastic gradient search. Neural networks for signal processing 2 – Proceedings of the 1992 IEEE Workshop. Piscataway NJ: IEEE Press; 1992.
[10] Leen TK, Orr GB. Using curvature information for fast stochastic search. In: Jordan MI, Mozer MC, Petsche T, editors. Neural information processing systems 9. Cambridge, MA: MIT Press; 1996.
[11] Almeida L, Langlois L, Amaral J. On-line step size adaptation. Technical report, INESC RT07/97, Lisbon, Portugal, 1997.
[12] Sha D, Bajić VC. Adaptive on-line ANN learning algorithm and application to identification of non-linear systems. Informatica: Int J Comput Informat 1999;23(4):251–9.
[13] Brent RP. Fast training algorithms for multilayer neural mets. IEEE Trans Neural Networks 1991;2(3): 346–54.
[14] Hagan MT, Menhaj M. Training feedforward networks with Marquardt algorithm. IEEE Trans Neural Networks 1994;5(6):989–93.
[15] Parisi R, Di Claudio ED, Orlandi G, Rao BD. A generalized learning paradigm exploiting the structure of feedforward neural networks. IEEE Trans Neural Networks 1996;7:1450–9.
[16] Wang G-J, Chen C-C. A fast multilayer neural networks training algorithm based on the layer-by-layer optimizing procedures. IEEE Trans Neural Networks 1996;7:768–75.
[17] Yam JYF, Chow WS. Extended least squares based algorithm for training feedforward networks. IEEE Trans Neural Networks 1997;8:806–10.
[18] Ergezinger S, Thomsen E. An accelerated learning algorithm for multilayer perceptrons: optimization layer by layer. IEEE Trans Neural Networks 1995;6:3142.
[19] Oh S-H, Lee S-Y. A new error function at hidden layers for fast training of multilayer perceptrons. IEEE Trans Neural Networks 1999;10(4):960–4.
[20] Chen G, Ogmen H. Modified extended Kalman filtering for supervised learning. Int J Syst Sci 1993;24(6): 1207–14.
[21] Iiguni Y, Sakai H, Tokumaru H. A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter. IEEE Trans Signal Process 1992;40(4):959–66.
[22] Shah S, Palmieri F, Datum M. Optimal filtering algorithms for fast learning in feedforward neural networks. Neural Networks 1992;5:779–87.
[23] Zhang Y, Li XR. A fast U-D factorization - based learning algorithm with applications to nonlinear system modeling and identification. IEEE Trans Neural Networks 1999;10(4):930–8.
[24] Roger Jang JS. Fuzzy modeling using generalized neural networks and Kalman filter algorithm. Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI-91. 1991. p. 762–67.
[25] Jang J-SR. ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Systems, Man, Cybernetics 1993;23(3):665–85.
[26] Sha D, Bajić VB. On-line hybrid learning algorithm for MLP: gradient descent and RLS. Proceedings of the Information Society, IS'99, Ljubljana, Slovenia, 1999 Oct 12–14. p. 111–14.
[27] Narendra KS, Parthasarathy K. Identification and control of dynamical systems using neural networks. IEEE Trans Neural Networks 1990;1(1):4–27.
[28] Tsypkin YZ, Mason JD, Avedyan ED, Warwick K, Levin IK. Neural networks for identification of nonlinear systems under random piecewise polynomial disturbances. IEEE Trans Neural Networks 1999;10(2): 303–11.
[29] Philip Chen CL, Wan JZ. A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. IEEE Trans Systems, Man, Cybernetics, Part B: Cybernetics 1999;29(1): 62–72.
[30] Wellstead PE, Zarrop MB. Self-tuning systems: control and signal processing. Chichester: Wiley; 1991.
[31] Norgaard M. Neural network based control system design toolkit, Technical report 96-E-830, Department of Automation, Technical University of Denmark, 1996.

Vladimir B. Bajić received his D.Eng.Sc. (E.E.) from University of Zagreb, Yugoslavia, in 1989. He is author or co-author of over 200 research publications and edited volumes. He is Director of the Centre for Engineering Research and Professor at Technikon Natal, South Africa. Dr Bajić is Co-Editor-in-Chief of the Stability and Control: Theory and Applications (SACTA) journal and the International Journal of Computers, Systems and Signals. From 1995 he is Secretary General of IAAMSAD and he holds many international offices. He chaired SSCC'98 and ICAI'99 and participated in the organization of over 20 international conferences. Dr. Bajić's current research interest is in Bioinformatics and in different applications of Artificial Intelligence.

Daohang Sha received his Dr. Eng. degree in Mechanical Engineering in 1995 from Xi'an Jiaotong University, Peoples Republic of China. He was a postdoctoral research fellow with Zhejiang University from 1996 to 1998. At present he is a postdoctoral research fellow with Technikon Natal, South Africa. His research interests include discrete sliding mode control, neural networks, and fuzzy logic.