Linear Least-Squares Based Methods for Neural Networks Learning

Oscar Fontenla-Romero[1], Deniz Erdogmus[2], J.C. Principe[2],
Amparo Alonso-Betanzos[1], and Enrique Castillo[3]

[1] Laboratory for Research and Development in Artificial Intelligence,
Department of Computer Science, University of A Coruña,
Campus de Elviña s/n, 15071 A Coruña, Spain
[2] Computational NeuroEngineering Laboratory,
Electrical and Computer Engineering Department,
University of Florida, Gainesville, FL 32611, USA
[3] Department of Applied Mathematics and Computational Sciences,
University of Cantabria and University of Castilla-La Mancha,
Avda de Los Castros s/n, 39005 Santander, Spain

Abstract. This paper presents two algorithms to aid the supervised learning of feedforward neural networks. Specifically, an initialization and a learning algorithm are presented. The proposed methods are based on the independent optimization of a subnetwork using linear least squares. An advantage of these methods is that the dimensionality of the effective search space for the non-linear algorithm is reduced, and therefore it decreases the number of training epochs which are required to find a good solution. The performance of the proposed methods is illustrated by simulated examples.

1 Introduction

During the last decade, several techniques have been presented to speed up the convergence of the artificial neural networks learning methods. Among them, some of the most successful are second order methods [1, 2]. These techniques have been demonstrated to be significantly faster than gradient based methods. In addition, other algorithms were presented, such as adaptive step size methods. In this last class of methods, Almeida et al. [3] developed a new method for step size adaptation in stochastic gradient optimization. This method uses independent step sizes for all parameters and adapts them employing the estimates of the derivatives available in the gradient optimization procedure. Moreover, a new on line algorithm for local learning rate adaptation was proposed by Schraudolph [4].

Likewise, several solutions have been proposed for the appropriate initialization of weights. Among others, Nguyen and Widrow [5] presented an algorithm that selects initial weights and biases for a layer, so that the active regions of the layer's neurons will be distributed approximately evenly over the input space.

Drago and Ridella [6] proposed a statistical analysis aimed to determine the relationship between a scale factor proportional to the maximum magnitude of the weights and the percentage of paralyzed neurons. These methods were shown to be very useful to improve the convergence speed.

In addition, some least squares applications have been proposed [7, 8]. These approaches are based on heuristic assumptions that do not consider the scaling effects of the nonlinear activation function. In this work, new theoretical results are presented that enhance the previous studies. Specifically, two algorithms, based on linear least squares, are presented to aid the current supervised learning methods for neural networks in order to accelerate their convergence speed.

2 Theoretical background

In this section, previous theoretical results are presented that will be used in the proposed algorithms. Consider the one-layer neural network in figure 1. The input of the network is represented by the vectorial variable $\mathbf{x} = (x_1, x_2, \ldots, x_I)^T$ and the output by the vector $\mathbf{y} = (y_1, y_2, \ldots, y_J)^T$. The outcome of the network is computed as $\mathbf{y} = \mathbf{f}(\mathbf{z})$, where $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$ and $\mathbf{f} \in \mathbb{R}^J$ is a nonlinear function. If the weighted mean squared error (weighted MSE) is employed as cost function

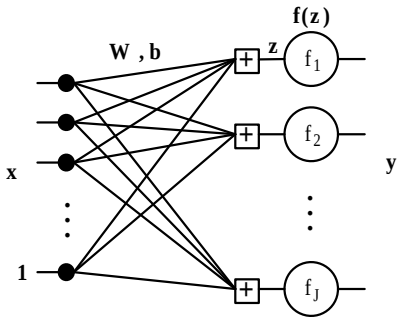

Fig. 1. One-layer neural network and nomenclature employed.

then the optimal weights and bias of the network can be obtained by solving the following minimization problem:

$$\min_{\mathbf{W},\mathbf{b}} C = E[(\mathbf{d} - \mathbf{y})^T \mathbf{H} (\mathbf{d} - \mathbf{y})] \tag{1}$$

where $\mathbf{d} = (d_1, d_2, \ldots, d_J)^T$ is the desired output of the network and $\mathbf{H} = [h_{ij}]$ is a weight matrix (symmetric by definition). However, this problem is, in general, a nonlinear optimization problem and hence it is currently solved using nonlinear methods. In this work a new result is presented that allows to solve this problem using a linear method.

Lemma 1. *Let* $\mathbf{d}$ *and* $\mathbf{y}$ *be the desired and actual outputs of a one-layer neural network,* $\mathbf{W}$ *and* $\mathbf{b}$ *be the weight matrix and the bias vector, and* $\mathbf{f}, \mathbf{f^{-1}}, \mathbf{f'}$ *be the nonlinear function, its inverse and its derivative. Then the following equivalence holds up to the first order of the Taylor series expansion:*

$$\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{d}-\mathbf{y})^T\mathbf{H}(\mathbf{d}-\mathbf{y})] \approx \min_{\mathbf{W},\mathbf{b}} E[(\mathbf{f'}(\bar{\mathbf{d}}).*\bar{\boldsymbol{\varepsilon}})^T\mathbf{H}(\mathbf{f'}(\bar{\mathbf{d}}).*\bar{\boldsymbol{\varepsilon}})] \tag{2}$$

where '.∗' denotes the element-wise product, $\bar{\mathbf{d}} = \mathbf{f^{-1}}(\mathbf{d})$ *and* $\bar{\boldsymbol{\varepsilon}} = \bar{\mathbf{d}} - \mathbf{z}$.

The proof of this and following lemmas is not included due to space restrictions. Thus, using the previous lemma the initial minimization problem in (1) can be alternatively formulated as:

$$\min_{\mathbf{W},\mathbf{b}} C^* = E[(\mathbf{f'}(\bar{\mathbf{d}}).*\bar{\boldsymbol{\varepsilon}})^T\mathbf{H}(\mathbf{f'}(\bar{\mathbf{d}}).*\bar{\boldsymbol{\varepsilon}})] \tag{3}$$

If the alternative cost function in (3) is used then the minimization problem is linear in the parameters (weights an biases). This is due to the variable $\bar{\boldsymbol{\varepsilon}}$ depending linearly on $\mathbf{W}$ and $\mathbf{b}$. Let $\{(\mathbf{x}_s, \mathbf{d}_s), s = 1, \ldots, S\}$ be a set of training pairs then the minimization problem in (3) can be rewritten as follows:

$$\min_{\mathbf{W},\mathbf{b}} C^* = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{J}\sum_{j=1}^{J} h_{ij} f_i'(\bar{d}_{is}) f_j'(\bar{d}_{js})\bar{\varepsilon}_{is}\bar{\varepsilon}_{js} \tag{4}$$

where $\bar{d}_{is}$, $\bar{\varepsilon}_{is}$ and f_i are the i^{th} component of the vectors $\bar{\mathbf{d}}_s$, $\bar{\boldsymbol{\varepsilon}}_s$ and $\mathbf{f}$, respectively. The optimal solution for the minimization problem in (4) can be obtained taking the derivatives of the cost function with respect to the weights and biases of the system and equating all the derivatives to zero. In this way, the following linear system of equations ($(I+1)\times J$ equations and unknowns) is obtained:

$$\begin{aligned}
&\sum_{i=1}^{J} b_i h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is}) x_{ls}\right] + \sum_{p=1}^{I}\sum_{i=1}^{J} w_{ip} h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is}) x_{ls} x_{ps}\right] \\
&\qquad = \sum_{i=1}^{J} h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is}) x_{ls} d_{is}\right]; k = 1, \ldots, J; l = 1, \ldots, I \\
&\sum_{i=1}^{J} b_i h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is})\right] + \sum_{p=1}^{I}\sum_{i=1}^{J} w_{ip} h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is}) x_{ps}\right] \\
&\qquad = \sum_{i=1}^{J} h_{ik}\left[\sum_{s=1}^{S} f_k'(\bar{d}_{ks}) f_i'(\bar{d}_{is}) d_{is}\right]; k = 1, \ldots, J.
\end{aligned} \tag{5}$$

where b_i is the i^{th} component of the bias vector and w_{ip} is the weight connecting the output i and the input p. Therefore, the unique solution (except for degenerated systems) of the proposed minimization problem in (4) can be achieved by solving the system of equations in (5) for the variables w_{ip} and b_i; $i = 1, \ldots, J$; $p = 1, \ldots, I$.

3 Proposed algorithms

In this work two algorithms for the supervised learning of multilayer feedforward neural networks are proposed. These methods are based on the results obtained in the previous section for one-layer neural networks. Consider the feedforward multilayer neural network in figure 2. It is composed of L layers where each layer l contains N_l neurons. The output of neuron i in layer l, y_i^l, is determined by the activation z_i^l, defined as the weighted sum of the outputs coming from the neurons in the previous layer, and an activation function f_i^l. Specifically, $\mathbf{y}^0 = (y_1^0, \ldots, y_{N_0}^0)^T$ is the input vector of the network.

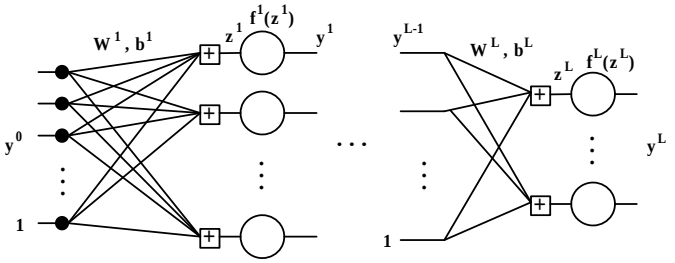

Fig. 2. Feedforward multilayer neural network and nomenclature employed.

3.1 Initialization method

The result presented in lemma 1 can be used to measure alternatively the error before the nonlinear functions of each layer in the multilayer feedforward neural network presented in figure 2. In this context, the input of the one-layer neural network ($\mathbf{x}$) will correspond to the output of the previous layer. In addition, it is necessary to obtain a result that allows to backpropagate the error from the output (before the activation function) of layer l to the output of the previous layer $(l-1)$. This result is presented in the following lemma.

Lemma 2. *Let $\mathbf{d}^{l-1}, \mathbf{y}^{l-1}, \bar{\mathbf{d}}^l, \mathbf{z}^l$ be the desired signals and corresponding output signals of layers l and $l-1$, $\mathbf{W}^l$ and $\mathbf{b}^l$ be the fixed weight matrix and the bias vector. Minimization of the weighted MSE between $\bar{\mathbf{d}}^l$ and $\mathbf{z}^l$ at the output of the linear layer is equivalent to minimizing a weighted MSE between $\mathbf{d}^{l-1}$ and $\mathbf{y}^{l-1}$, i.e., finding the constrained linear least squares solution for the optimal input vector. Mathematically, this is given by*

$$\min_{\mathbf{y}^{l-1}} E[(\bar{\mathbf{d}}^l - \mathbf{z}^l)^T \mathbf{H}^l (\bar{\mathbf{d}}^l - \mathbf{z}^l)] = \min_{\mathbf{y}^{l-1}} E[(\mathbf{d}^{l-1} - \mathbf{y}^{l-1})^T \mathbf{W}^{lT} \mathbf{H}^l \mathbf{W}^l (\mathbf{d}^{l-1} - \mathbf{y}^{l-1})] \tag{6}$$

There are two different situations in the problem presented in the previous lemma: a) if $N_l \geq N_{l-1}$, then $\mathbf{d}^{l-1} = (\mathbf{W}^{lT}\mathbf{H}^l\mathbf{W}^l)^{-1}\mathbf{W}^{lT}\mathbf{H}^l(\bar{\mathbf{d}}^l - \mathbf{b}^l)$ is the unique weighted least squares solution for the overdetermined system of linear equations ($\mathbf{W}^l\mathbf{d}^{l-1} + \mathbf{b}^l = \bar{\mathbf{d}}^l$); and b) if $N_l < N_{l-1}$, then the QR factorization may be used to determine the minimum norm least squares solution for this undetermined system of linear equations ($\mathbf{W}^l\mathbf{d}^{l-1} + \mathbf{b}^l = \bar{\mathbf{d}}^l$).

In both cases, given a desired signal $\bar{\mathbf{d}}^l$, for the linear output $\mathbf{z}^l$ of the layer l^{th}, it can translate as a desired signal $\mathbf{d}^{l-1}$ for the output (after the nonlinearity) of the previous layer. Subsequently, this value can be backpropagate through the nonlinearity as described in lemma 1. Thus, using lemma 1 and 2 the following algorithm is proposed:

Given $\{(\mathbf{y}^0, \mathbf{d}_s^L), s = 1, \ldots, S\}$, *select random initial values for weights and biases* $\mathbf{W}^l$, $\mathbf{b}^l$; $l = 1, \ldots, L$.
Evaluate $\mathbf{z}_s^l$, $\mathbf{y}_s^l$; $s = 1, \ldots, S$; $l = 1, \ldots, L$; *using* $\mathbf{y}_s^0$, $\mathbf{W}^l$ *and* $\mathbf{b}^l$; $l = 1, \ldots, L$.
Set C_{opt} *to the MSE between* $\mathbf{y}_s^L$ *and* $\mathbf{d}_s^L$. *Set* $\mathbf{W}_{opt}^l = \mathbf{W}^l$, $\mathbf{b}_{opt}^l = \mathbf{b}^l$; $l = 1, \ldots, L$.
Compute $\bar{\mathbf{d}}_s^L = (\mathbf{f}^L)^{-1}(\mathbf{d}_s^L) \forall s$, *as the desired signal for* $\mathbf{z}_s^L$.
$n = 1$.
while $n \leq MaxIterations$
    for $l = L - 1$ *downto* 1
        Compute $\bar{\mathbf{d}}_s^l = (\mathbf{f}^l)^{-1}(\mathbf{d}_s^l) \forall s$, *as the desired signal for* $\mathbf{z}_s^l$.
        Compute $\mathbf{d}_s^{l-1} = (\mathbf{W}^{lT}\mathbf{W}^l)^{-1}\mathbf{W}^{lT}(\bar{\mathbf{d}}_s^l - \mathbf{b}^l)$ *as the desired signal for* $\mathbf{y}_s^{l-1}$ *(this is the case for the overdetermined case, for the undetermined case, the minimum norm solution could be used).*
    end
    for $l = 1$ *to* L
        Optimize $\mathbf{W}^l$ *and* $\mathbf{b}^l$ *using the linear system of equations in (5), using* $\mathbf{y}_s^{l-1}$ *as input samples and* $\bar{\mathbf{d}}_s^l$ *as desired output samples.*
        Evaluate $\mathbf{z}_s^l$ *and* $\mathbf{y}_s^l$ *using the new values of the weights and bias.*
    end
    Evaluate the value of C *(the MSE between* $\mathbf{y}_s^L$ *and* $\mathbf{d}_s^L$*).*
    If $C < C_{opt}$ *then set* $C_{opt} = C$, $\mathbf{W}_{opt}^l = \mathbf{W}^l$, $\mathbf{b}_{opt}^l = \mathbf{b}^l$, $l = 1, \ldots, L$.
    $n = n + 1$.
end

Finally, an important issue to remark is that this algorithm is proposed as an initialization method and not as a learning method because it has not a smooth convergence to the solution. Instead, it jumps from one region to another of the weight space.

3.2 Learning method

In this subsection a new learning method based on linear least squares is presented. In this case only the last layer of the network (L) is optimized using

linear least squares whereas the other layers are optimized using any standard method. The proposed algorithm for a multilayer feedforward neural network is as follows:

1 *Select initial weights* $\mathbf{W}_l$, $\mathbf{b}_l$, $\forall l$, *using an initialization method or randomly.*
2 *Evaluate the value of* C_0 *(MSE between* y_s^L *and* d_s^L*) using the initial weights.*
3 $n = 1$.
4 ***while*** $n \leq MaxIterations$ *and* ($\neg$*stop_criterion_1)*
5 $\quad \mathbf{W}^l = \mathbf{W}^l + \Delta\mathbf{W}^l; l = 1, \ldots, L$.
6 $\quad \mathbf{b}^l = \mathbf{b}^l + \Delta\mathbf{b}^l; l = 1, \ldots, L$.
7 $\quad$ *Evaluate the value of* C_n *(MSE between* y_s^L *and* d_s^L*).*
8 $\quad$ ***If*** $|C_n - C_{n-1}| < \lambda$ *then*
9 $\quad\quad$ ***while*** ($\neg$*stop_criterion_2)*
10 $\quad\quad\quad \mathbf{W}^l = \mathbf{W}^l + \Delta\mathbf{W}^l; l = 1, \ldots, L-1$.
11 $\quad\quad\quad \mathbf{b}^l = \mathbf{b}^l + \Delta\mathbf{b}^l; l = 1, \ldots, L-1$.
12 $\quad\quad\quad$ *Update* $\mathbf{W}^L$ *and* $\mathbf{b}^L$ *using the linear system of equations in (5).*
13 $\quad\quad$ ***end***
14 $\quad$ ***end_if***
15 ***end***

The method works as follows: in the first phase (first epochs of training), all layers of the network are updated using any standard learning rule (steps 5 and 6). In the second stage, when the obtained decrement of the error is small (step 8), then the update procedure switches to the hybrid approach. Thus, $\mathbf{W}_L$ of the network is optimally obtained using linear least squares (step 12) while $\mathbf{W}_l$; $l = 1, \ldots, L-1$ is still updated using the standard learning method (steps 10 and 11).

4 Results

In this section, a comparative study between the proposed methods and standard algorithms is presented. In order to accomplish the study, several nonlinear system identification data sets were employed. However, due to space restrictions, only the results obtained for one of the data sets are shown. The data employed in this paper is the time series from the K.U. Leuven prediction competition[4] which was part of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling in 1998 [9]. The data were generated from a computer simulated 5-scroll attractor, resulting from a generalized Chua's circuit, and it consists of 2000 samples. In this work, the desired signal was normalized in the interval $[0.05, 0.95]$. The time series is shown in figure 3(a). The topology employed in the following experiments was 6-7-1 though other topologies were used to obtain a similar behaviour. Also, logistic and linear functions were used in the processing elements (PEs) of the hidden and output layer, respectively. For all the experiments a Monte Carlo simulation, using 100 different initial random weights sets, was carried out.

[4] ftp://ftp.esat.kuleuven.ac.be/pub/sista/suykens/workshop/ datacomp.dat

4.1 Initialization method

In this subsection, the results for the initialization method (section 3.1) are presented. In this case, the backpropagation method was used to train the network using as initial weights either random values or those obtained by the proposed initialization method (using 3 iterations of the algorithm). In the former case the network was trained during 4000 epochs while in the latter only during 2000 epochs. Figure 3 shows the results obtained in the 100 simulations. Figure 3(b) contains the histogram of the errors using only the initialization method. Moreover, figures 3(c) and 3(d) contain, respectively, the histogram of the errors, in the last epoch of training, for the backpropagation algorithm using random weights and the initial weights obtained by the initialization method (LS).

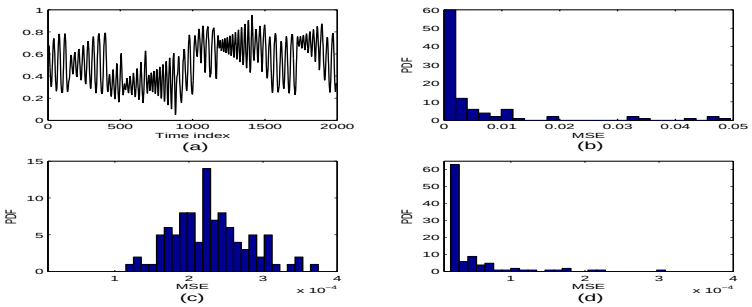

Fig. 3. (a) Time serie of the K.U. Leuven competition and histogram of the final MSE using (b) the initialization method, (c) backpropagation and (d) LS+backpropagation.

4.2 Learning method

In this subsection, a comparative study between the proposed hybrid learning method (section 3.2) and the scaled conjugate gradient method (SCG) [10] is presented. In order to accomplish a fair comparison, the hybrid method employed also the SCG for weights and biases update in steps 5, 6, 10 and 11. In all the simulations (100) the initial weights employed by both algorithms (SCG and the hybrid method) were the same, therefore they started at identical initial conditions. The results obtained for this time series are shown in figure 4. Figures 4(a) and 4(b) present the learning curves of the Monte Carlo simulation for the SCG and the proposed method, respectively. As it can be seen, the proposed learning scheme obtains a faster convergence to the solution than the standard SCG method.

5 Conclusions

In this paper new algorithms for the initialization and supervised learning of the parameters of a feedforward neural network were presented. The proposed

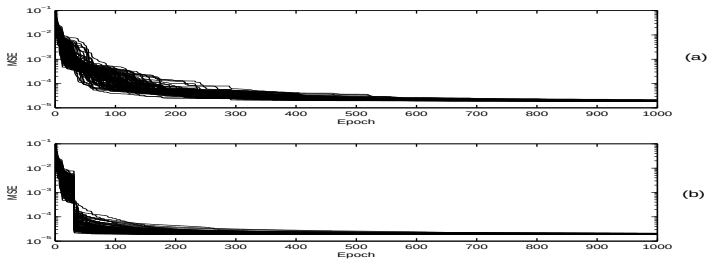

Fig. 4. Learning curves for (a) the SCG algorithm and (b) the proposed hybrid method.

algorithms are based on linear least squares for the independent optimization of at least one layer of the network. An advantage of these methods is that the number of epochs needed to achieve a good solution is decreased as a consequence of the dimensionality reduction of the effective search space for the non-linear algorithm. The application to benchmark data confirmed the good performance of the proposed methods.

6 Acknowledgements

This work is partially supported by NSF grant ECS-9900394 and the Xunta de Galicia (project PGIDT-01PXI10503PR).

References

1. Battiti, R.: First and second order methods for learning: Between steepest descent and newton's method. Neural Computation **4** (1992) 141–166
2. Buntine, W.L., Weigend, A.S.: Computing second derivatives in feed-forward networks: A review. IEEE Trans. on Neural Networks **5** (1993) 480–488
3. Almeida, L.B., Langlois, T., Amaral, J.D., Plakhov, A.: 6. In: Parameter adaptation in stochastic optimization. Cambridge University Press (1999) 111–134
4. Schraudolph, N.N.: Fast curvature matrix-vector products for second order gradient descent. Neural Computation **14** (2002) 1723–1738
5. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Proc. of the Int. Joint Conference on Neural Networks **3** (1990) 21–26
6. Drago, G., Ridella, S.: Statistically controlled activation weight initialization (SCAWI). IEEE Trans. on Neural Networks **3** (1992) 899–905
7. Biegler-Konig, F., Barnmann, F.: A learning algorithm for multilayered neural networks based on linear least squares problems. Neural Networks **6** (1993) 127–131
8. Yam, Y., Chow, T.: A new method in determining the initial weights of feedforward neural networks. Neurocomputing **16** (1997) 23–32
9. Suykens, J., Vandewalle, J., eds.: Nonlinear Modeling: advanced black-box techniques. Kluwer Academic Publishers, Boston (1998)
10. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks **6** (1993) 525–533