Accelerating the convergence speed of neural networks learning methods using least squares

Oscar Fontenla-Romero[1]*, Deniz Erdogmus[2], Jose C. Principe[2], Amparo Alonso-Betanzos[1], 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. In this work a hybrid training scheme for the supervised learning of feedforward neural networks is presented. In the proposed method, the weights of the last layer are obtained employing linear least squares while the weights of the previous layers are updated using a standard learning method. The goal of this hybrid method is to assist the existing learning algorithms in accelerating their convergence. Simulations performed on two data sets show that the proposed method outperforms, in terms of convergence speed, the Levenberg-Marquardt algorithm.

1 Introduction

The error back-propagation method has been greatly used for the supervised training of feedforward neural networks. However, as it is well-known, this method has a slow convergence. Several techniques have been developed to speed up this method, such as, among others, second order algorithms [2, 5], adaptive step size methods [1, 12, 13], appropriate weights initialization [7, 11, 14] and approximate optimization based on heuristic least squares application [3, 14]. These latter methods are based on measuring the error of the network before the output nonlinear functions and on the backbackpropagation of the

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

*Corresponding author. Email: oscarfon@mail2.udc.es

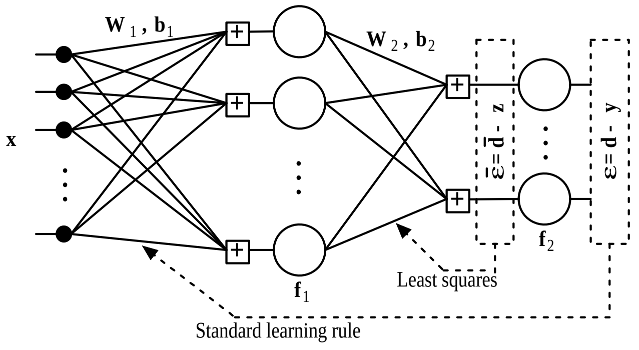

Figure 1: Two-layer neural network and scheme of the proposed hybrid method.

error through the layers. These works employ heuristic approaches that do not consider the scaling effects of the nonlinear function's slope.

In this work, a hybrid algorithm for the supervised learning of feedforward neural networks is presented. This method is different from the previous least squares approaches in two aspects: 1) new theoretical results are presented that enhance the previous studies that are based on heuristic assumptions, and 2) the linear least squares is not employed for all the layers of the network but only for the last layer. The aim of the proposed method is to assist the current algorithms in order to accelerate their convergence. This is achieved by learning optimally the last layer weights of the network using linear least squares. The proposed algorithm is described for a two-layer neural network, although, it can be easily generalized for n layers. Although the simulations presented in this work use the Levenberg-Marquardt algorithm, the proposed method is appropriate for speeding up the convergence of any of the current algorithms.

2 Proposed learning method

Consider the neural network in figure 1, that is composed of two layers of weights ($\mathbf{W}_1$ and $\mathbf{W}_2$) and biases ($\mathbf{b}_1$ and $\mathbf{b}_2$), where the variables $\mathbf{y}$ and $\mathbf{z}$ represent the outputs of the network after and before the nonlinearity of the last layer ($\mathbf{f}_2$). The variable $\mathbf{d}$ represents the desired response of the network. In this work, we consider a supervised learning (the assumed cost function is the mean square error), thus, we can compute the error ($\boldsymbol{\varepsilon}$) between the actual output ($\mathbf{y}$) and the desired response ($\mathbf{d}$), i.e., $\boldsymbol{\varepsilon} = \mathbf{d} - \mathbf{y}$. The standard methods employed to train neural networks usually use some first and second order information, in the optimization rule, to update both layers. In this way, the learning method modifies, at the same time, the basis functions ($\mathbf{W}_1$) of the network and the projections ($\mathbf{W}_2$). However, in this paper we propose an alternative training method, illustrated in figure 1, that uses two different learning methods to update the weights: (a) one of the standard learning methods, for the first layer, and (b) linear least squares, for the last layer. This approach is based on the results presented in [6] and in the following lemma (the proof is not included due to space restrictions):

Lemma 1 *Let* $\mathbf{x}$ *be the input of a one-layer neural network,* $\mathbf{d}, \mathbf{y}$ *be the desired and actual outputs,* $\mathbf{W}$*,* $\mathbf{b}$ *be the weights, 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:*

$$\min_{\mathbf{W},\mathbf{b}} E[(\mathbf{d}-\mathbf{y})^T(\mathbf{d}-\mathbf{y})] \approx \min_{\mathbf{W},\mathbf{b}} E[(\mathbf{f}'(\bar{\mathbf{d}}).*\bar{\varepsilon})^T(\mathbf{f}'(\bar{\mathbf{d}}).*\bar{\varepsilon})] \tag{1}$$

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

It is clear that if we use the alternative cost function on the right hand side of (1) the weights of the network do not appear inside the nonlinear function. Therefore, it is possible to train optimally a one-layer neural network with non-linear neural functions employing linear least squares. The optimal weights can be obtained taking the derivatives of this alternative cost function with respect to the weights of the system. This optimization problem is solved using a linear system of equations. Another advantage is that the learning is independent of the transfer functions employed in the output layer of the network. This result is used to learn the weights of the last layer in the two-layer neural network presented in figure 1. For a given value of $\mathbf{W}_1$ we can carry out a one-step exact minimization with respect to $\mathbf{W}_2$ using least squares, in which $\mathbf{W}_1$ is held fixed. Every time the value of $\mathbf{W}_1$ is changed the weights $\mathbf{W}_2$ are recomputed. An evident advantage of this method is that the dimensionality of the effective search space for the non-linear algorithm is reduced, and we might hope that this would reduce the number of training iterations which is required to find a good solution. In this case, the $\mathbf{x}$ parameters (inputs of the one-layer network) are replaced by the output of the first layer. In summary, the proposed algorithm is as follows:

1. *Select* $\mathbf{W}_1$*,* $\mathbf{b}_1$*,* $\mathbf{W}_2$ *and* $\mathbf{b}_2$ *randomly or using an initialization method.*
2. *Update the weights and the biases employing any standard optimization rule (e.g., Levenberg-Marquardt, conjugate gradient).*
3. *Evaluate the stopping rule. If it is not satisfied, then go to step 4; otherwise, the training is finished.*
4. *Compute the value of the cost function in this iteration* ($J(n)$)*. If* $|J(n) - J(n-1)| < \lambda$ *(where* λ *is a fixed threshold) then go to step 5; otherwise, go back to step 2.*
5. *Update* $\mathbf{W}_1$*,* $\mathbf{b}_1$ *using the standard optimization rule and* $\mathbf{W}_2$*,* $\mathbf{b}_2$ *using least squares:*
   - $\mathbf{W}_1(n+1) = \mathbf{W}_1(n) + \Delta\mathbf{W}_1(n+1)$; $\mathbf{b}_1(n+1) = \mathbf{b}_1(n) + \Delta\mathbf{b}_1(n+1)$.

- $\mathbf{W}_2(n+1)$ *and* $\mathbf{b}_2(n+1)$ *are obtained using linear least squares.*

6. *Evaluate the stopping rule. If not satisfied, go back to step 5; otherwise, go back to step 2.*

In the first stage of the method (first epochs of training), both layers of the network are updated using a standard learning rule. In the second stage, when the error obtained by the network remains flat over iterations, the update procedure switches to the hybrid approach (step 5). Then, $\mathbf{W}_2$ of the network is optimally obtained using linear least squares while $\mathbf{W}_1$ is still updated using the standard learning method. The proposed algorithm was presented using a switching criterion (step 4) based on the difference between the current and the previous error, however, a different criterion could also be employed; for example, switching at a predetermined epoch of training.

3 Simulations

In this section, we present a comparative study between the proposed method and the Levenberg-Marquard (LM) algorithm [8]. We used the LM algorithm as a standard for comparison, because it is considered as one of the fastest methods for training moderate-size feedforward neural networks. In all the simulations we have used $\mu = 0.01$ as the initial step size, and logistic and linear functions in the processing elements (PEs) of the hidden and output layer, respectively. The stopping rule employed in steps 3 and 6 of the proposed algorithm was that the algorithm achieves a μ value greater than a maximum value (1×10^4) or a maximum number of epochs of training (different for each data set).

To make the comparative study, we employed two different nonlinear system identification data sets: the Dow-Jones-closing-index series [9] (1000 samples) and the Mackey-Glass time series [10] (2000 samples). The desired output was normalized in all the cases in the interval $[0.1, 0.9]$ and the employed topology was 3-6-1 and 5-7-1 for Dow-Jones and Mackay-Glass data, respectively. For all the data sets a Monte Carlo simulation, using 100 different initial random weights sets, were carried out. In all the simulations the initial weights employed by both algorithms (LM and the proposed method) were the same, therefore they start at identical initial conditions. The results of these simulations are shown in figures 2 and 3. Figures 2(a) and 2(b) presents the learning curves of the Monte Carlo simulations for the LM and the proposed method, respectively, using the Dow-Jones data. These figures show that the proposed method gives a better convergence rate compared to the standard LM algorithm. Moreover, we carried out the same simulations (with the same initial conditions) but using logistic functions in the output layer of the network. This setup (sigmoid outputs and sum-squared loss) is not desirable because it is well known that produces a worse convergence for regression problems [4]. However, we did these simulations to compare the performance of the proposed algorithm. Figures 2(c) and 2(d) show the results for LM and the presented method, respectively. As it can be observed, the proposed method is able to

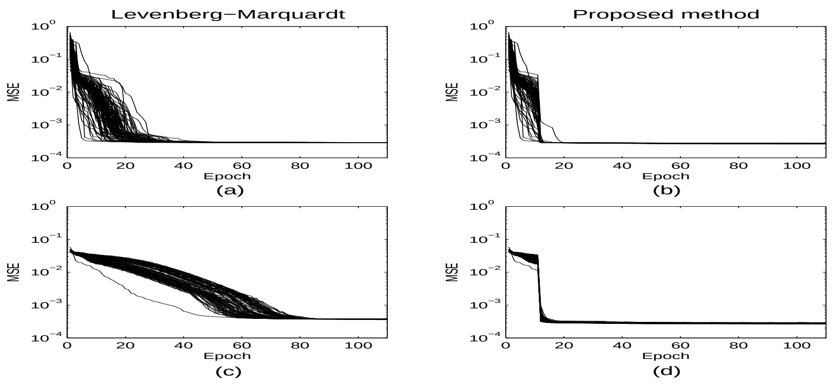

Figure 2: Dow-Jones data: learning curves (MSE for the training set) for linear outputs, (a) and (b), and for logistic outputs, (c) and (d).

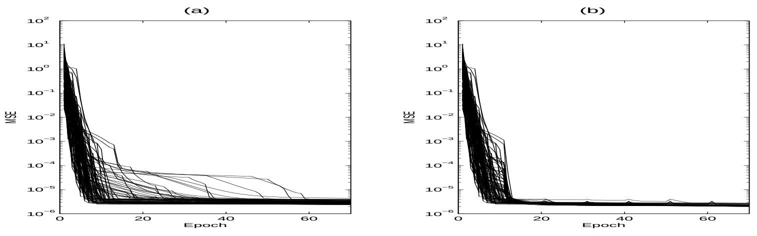

Figure 3: Results for the Mackey-Glass data: (a) LM and (b) LM + Least-squares.

overcome the problem of the poor convergence when a sigmoid outputs and sum-squared loss setup is used. Therefore, the convergence properties are independent of the activation functions used in the output layer. This is an additional advantage of the proposed algorithm. Figure 3 shows the learning curves for the Mackey-Glass data set. Again, the proposed method (figure 3(b)) achieved a faster convergence than the regular LM (figure 3(a)).

4 Conclusion

In this work, a new method to assist the supervised learning of feedforward neural networks to reduce their convergence time has been presented. The proposed approach combines the power of the standard methods (e.g., LM, conjugate gradient, etc.) to learn the weights of the first layer and the linear least squares method to obtain the weights of the second layer. The performance of the method was tested experimentally using two data sets. The simulations showed that it clearly outperforms the regular Levenberg-Marquardt algorithm, in terms of convergence time needed to achieve an optimum.

References

[1] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. *Parameter adaptation in stochastic optimization*, chapter 6, pages 111–134. Cambridge University Press, 1999.

[2] R. Battiti. First and second order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4(2):141–166, 1992.

[3] F. Biegler-Konig and F. Barnmann. A learning algorithm for multilayered neural networks based on linear least squares problems. *Neural Networks*, 6:127–131, 1993.

[4] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[5] W. L. Buntine and A. S. Weigend. Computing second derivatives in feedforward networks: A review. *IEEE Trans. on Neural Networks*, 5(3):480–488, 1993.

[6] E. Castillo, O. Fontenla-Romero, A. Alonso-Betanzos, and B. Guijarro-Berdiñas. A global optimum approach for one-layer neural networks. *Neural Computation*, 14(6):1429–1449, 2002.

[7] G.P. Drago and S. Ridella. Statistically controlled activation weight initialization (SCAWI). *IEEE Trans. on Neural Networks*, 3:899–905, 1992.

[8] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Trans. on Neural Networks*, 5(6):989–993, 1994.

[9] E. Ley. On the peculiar distribution of the US stock indices first digits. *The American Statistician*, 50(4):311–314, 1996.

[10] M.C. Mackey and L. Glass. Oscillation and chaos is physiological control systems. *Science*, 197:287–289, 1977.

[11] D. Nguyen and B. Widrow. 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*, 3:21–26, 1990.

[12] Genevieve B. Orr and Todd K. Leen. Using curvature information for fast stochastic search. In M.I. Jordan, M.C. Mozer, and T. Petsche, editors, *Neural Information Processing Systems*, volume 9, pages 606–612, Cambridge, 1996. MIT Press.

[13] Nicol N. Schraudolph. Fast curvature matrix-vector products for second order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

[14] Y.F. Yam and T.W.S Chow. A new method in determining the initial weights of feedforward neural networks. *Neurocomputing*, 16(1):23–32, 1997.