An Improved Training Approach for Feedforward Neural Networks

Yan Li, David Powers and Peng Wen
School of Informatics and Engineering
The Flinders University of South Australia,
GPO Box 2100, Adelaide, SA 5001, Australia
{yan.li, david.powers, peng.wen}@flinders.edu.au

Abstract

A new approach based on least square back-propagation algorithm (LSB) for improving training speed of feedforward neural networks (FNNs) is presented in this paper. A set of optimal initial weights for FNNs are obtained by using LSB algorithm first, the traditional back-propagation (BP) algorithm is then employed to train the network to further reduce the error. The simulation results through different networks demonstrate that the proposed method is much better than classical BP algorithm in terms of convergence speed and training error.

1. Introduction

Feedforward neural networks are perhaps the most widely used connectionist models (Ergezinger, 1995; Humpert, 1994; Battiti, 1989). Most training techniques of feedforward neural networks are based on different variations on gradient descent (Li, 1999; Humpert, 1994) with training started by assigning the initial weights of the network randomly (Rumelhart and McClelland; 1986). So far, there is no analytical approach in determining the initial values of the weights. However, the rate of convergence using gradient descent approach can be significantly affected by initial weights configuration. The network also is prone to getting stuck in the nearest local minimum. Even with the addition of an inertial momentum term, these problems can be partially alleviated. In the ongoing search to find better and faster variants of the back-propagation, researchers tend to vary the learning rate α and the momentum coefficient η in the algorithm (Jacobs, 1988). As another approach, researchers have discovered that a set of optimal initial weights of networks can lead to a dramatic increase in network training speed.

In this paper, a new approach based on least square back-propagation algorithm (Friedrich and Barmann, 1993) for evaluating optimal initial weights is presented. The convergence speed of the LSB algorithm is fast, however, only at the beginning, the first or second iteration, of the iteration process. A main drawback of the algorithm is the limited ability to further reduce the error after the first several iterations. In this paper, a set of optimal initial weights for FNN is obtained by using LSB algorithm. The traditional Backpropagation is then employed to train the FNN to improve the training accuracy.

The proposed method is described in the section 2. In section 3, we present simulations and compare the proposed method with conventional randomly initialised back-propagation version. Finally, the paper is concluded in section 4.

2. The Algorithm

Let us consider a three-layer neural network which is fully interconnected. In the classical back-propagation training method, all of the weights are set randomly before starting the training process. Knowing the network input and the desired output, the output of the hidden layer must be calculated by using the weighted sum and the activation function for selected training pairs. This output is then applied to the next layer as input. The error between the network output and the desired output is then calculated for adjusting the weights of the network such that the error is minimized. In this method, the initial error is usually very large so it usually takes a long time to get the desired accuracy. Sometimes it cannot escape from local minimal traps without outside interference. However, if the initial weight assignment causes the network to be close enough to the desired output, the learning speed will improve.

It has been proved that a three_layer network can estimate any continuous function to any precision (Rumelhart and McClelland, 1986). This implies that the output of hidden layer can be influenced dramatically on networks to achieve the exact mapping desired.

The main idea of the proposed algorithm is to separate each layer of the network into a linear part (usually a scalar product---the weighted sums) and a non-linear part (a one-dimensional activity function). In the output layer, using the inverse of the activity function to determine the weighted sums mapping with the desired output, and solving the linear part optimally; we can get a set of optimal weights. Then, according to calculated weights and the weighted sums of output layer, we would determine a "required" output of the hidden layer.

After getting the desired output of the hidden layer, The optimal weights connecting the input layer and the hidden layer can be obtained by using a similar method. The learning error is minimized on each layer separately. Therefore, the training error of the network is sufficiently small after one or two iteration(s).

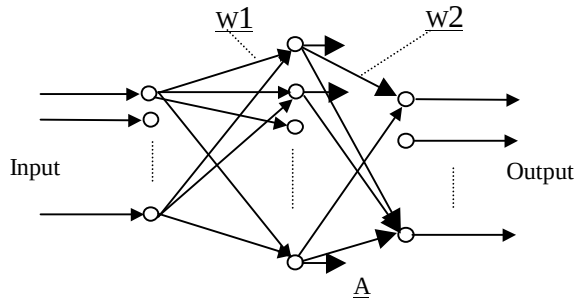

Figure 1. Architecture of a three-layer network

The architecture of such a network is shown in Figure 1, where X and T are the input and output training data sets, respectively. The network may be represented in block diagram form as a series of transformations W1 and W2, and a diagonal non-linear operator *Q* with identical sigmoidal activations. In other words, each layer of the network is regarded as the composition of a transformation with a non-linear mapping:

$$\underline{A} = Q(\underline{X}*\underline{W1}) \quad (1)$$
$$\underline{A2} = Q(\underline{A}*\underline{W2}) \quad (2)$$

Here A2 is the actual output of the network. Teaching the network means trying to adjust the weights such that A2 is equal to (or as close as possible to) T. We do this backwards by introducing

$$\underline{S}=Q^{-1}(\underline{T}) \quad (3)$$

Adjust the weights of the network such that S is as close as possible to A*W2. The problem of determining W2 optimally can be formulated as a linear least square problem:

$$\text{minimise}\|\underline{A}*\underline{W2}-\underline{S}\|_2 \quad (4)$$

Note that, since *Q* is a non-linear function, the minimum of equation (4) is not necessarily identical with the minimum of $\| \underline{A2} - \underline{T} \|_2$ (Coetzee and Stonick, 1995).

After obtaining an optimal set W2 of weights, we could determine a required output matrix R of hidden layer which is as close as possible to A using the linear square problem again. And optimal weights W1 are also obtained using the similar procedure described above.

This optimal sets of weights W1 and W2 are used as initial weights for back-propagation algorithm to train the three-layer network. It is expected that the convergence speed of the method is efficiently improved.

3. Numerical Simulation Results

In this section we present numerical experimental results about the performances of the method. The efficiency of a training procedure depends mainly on the training error and convergence speed. To evaluate the performance of the presented method, we applied it to the following non-linear function approximation problem. The results were then compared with the performance of networks started with random weights initialisation A brief description of the problems investigated is given below.

Consider the following traditional eight input/three output non-linear mapping function:

$$\begin{aligned} y_1&=(x_1x_2+x_3x_4+x_5x_6+x_7x_8)/4 \\ y_2&=(x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8)/8 \\ y_3&=(1-y_1)^2 \end{aligned} \quad (5)$$

All three functions are defined for values between *0* and *1* and produce values in this range. For the training set, 150 sets of input signals x_i were generated with a random number generator, the corresponding y_i were computed using equation (5). The error reported is the average absolute error on this learning set example and output node:

$$\Delta = \left(\sum_{i=1,\ldots N} \sum_{j=1,\ldots n_L} \left| A^L{}_{i,j} - T^L{}_{i,j} \right| \right) \Big/ (Nn_L) \quad (6)$$

Table 1 Initial errors using proposed method versus randomly initialised

Method	8 neurones in hidden layer	12 neurones in hidden layer	24 neurones in hidden layer	36 neurones in hidden layer
Proposed	0.0370	0.0355	0.0310	0.0299
Conventional	0.2517	0.4508	0.4602	0.4612
Improved Rate	85.62%	92.11%	93.26%	93.5%

Table 2 Iterations to achieve the required error criterion E=0.015

Method	8 neurones in hidden layer	12 neurones in hidden layer	24 neurones in hidden layer	36 neurones in hidden layer
Proposed	360	532	565	443
Conventional	804	910	1269	3552

Four three-layer neural networks were trained with architectures 8-8-3, 8-12-3, 8-24-3, 8-36-3 respectively.

MATLAB is used to implement the above simulations. Every case has been repeated many times and the final error is slightly different every time. The average results for the non-linear mapping problem are shown in Tables 1 and 2. Table 1 shows that the initial errors using the proposed method is only 8.87 % of the average initial error started with randomly initialized weights. For these error values, the network with randomly initialized weights needs hundreds or even thousands of iterations to reach this point. Table 2 shows the number of iteration needed for each of the networks with initialized weights and with random weights to achieve the required error criterion E=0.015. From Tables 1 and 2, one can infer that the more nodes in the hidden layer, the better the results.

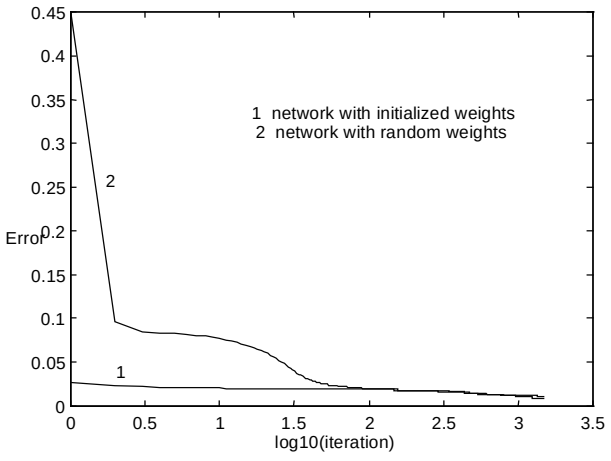

Figure 2 The performances of two networks

The performances of typical runs on 8-12-3 networks using the two approaches are shown in Fig. 2. The number of training iterations was 1500.

The network started with random weights is identical with the initialized version in all aspects except initial weights. The learning rate and momentum are set to 0.1 and 0.9, respectively. These values are selected because the rate of convergence for the networks started with random weights is fastest under these condition. The networks can therefore be trained more efficiently using the optimal initial weights evaluated by the proposed method.

4. Conclusion

A novel approach for accelerating the training speed of a neural networks is successfully established in this paper. By utilising the optimal weights, the initial network error can be greatly reduced. For some of the examples in this paper, the initial network error with optimal weights is small enough for direct applications. If a smaller network error is required, the network can be further trained using back-propagation algorithm. The simulation results through different networks demonstrate that the proposed method is much better than classical BP algorithm in terms of convergence speed and training error.

It should be noted that there are now many variants of BP which promise faster convergence, however often at the expense of considerable additional memory requirements. For example, Matlab provides the Levenberg-Marquart method which is supposed to be faster, but on even fairly small problems it can end up being orders of magnitude slower due to thrashing owing to the large memory overheads.

It remains to compare the speed of this approach with all these other variants, but in all cases we would

expect that starting from a better set of initial values will tend to produce a reduction in training time. It should be noted that the training cost of our method grows very slowly – in fact the more neurones there are in the hidden layer, the more accurate the LSB approximation will be, so that training time can actually decrease (as seen in Table 2).

Although the time taken for the LSB iteration will increase, the approximation will be so good that the number of BP iterations required will be dramatically reduced. However, the training time taken by the conventional approach grows quadratically with the size of the network, and memory requirements will increase similarly, even before taking into account any increase in the number of iterations required.

References

[1] S. Ergezinger. and E. Thomsen, "An Accelerated Learning Algorithm for Multilayer Perceptrons: Optimisation Layer by Layer", IEEE Transaction on Neural Networks, Vol. 6, No. 1, Jan., 1995, pp. 31-42.

[2] B.K Humpert, “Improving backgropagation with a new error function, Neural Networks, 1994; 7(8): pp. 1191-1192.

[3] R. Battiti, "Accelerating Back-propagation Learning, Two Optimization Methods, Complex system", Vol. 3, 1989, pp.331-342.

[4] Y. Li, A. B Rad and P. Wen, “An Enhanced Training Algorithm for Multilayer Neural Networks Bsed on Reference Output of Hidden Layer”, Neural Computing and Applications, Vol. 8, 1999, pp. 218-225.

[5] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagatoin”, Parellel Distributed Processing: Explorations of the Microstructure of Cognition. Cambridge MA: Bradford, MIT Press, Vol. 1, 1996, pp 318-362.

[6] R. A. Jacobs, "Increased Rates of Convergence through Learning Rate Adaption", Neural Networks, Vol.1, 1988, pp.295-307.

[7] B. K. Friedrich and B Frank, "A Learning Algorithm for Mutilayer Neural Networks Based on Linear Least Square Problems", Neural Networks, Vol. 6**,** 1993, pp. 127—131.

[8] F M Coetzee, V L Stonick, “Topology and geometry of single hidden layer network least squares weights solutions, Neural Computation, Vol. 7, 1995, pp. 672-705.