ELSEVIER

Neurocomputing 30 (2000) 219–232

NEUROCOMPUTING

www.elsevier.com/locate/neucom

A weight initialization method for improving training speed in feedforward neural network

Jim Y.F. Yam, Tommy W.S. Chow*

Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

Received 3 April 1998; accepted 22 March 1999

Abstract

An algorithm for determining the optimal initial weights of feedforward neural networks based on the Cauchy's inequality and a linear algebraic method is developed. The algorithm is computational efficient. The proposed method ensures that the outputs of neurons are in the active region and increases the rate of convergence. With the optimal initial weights determined, the initial error is substantially smaller and the number of iterations required to achieve the error criterion is significantly reduced. Extensive tests were performed to compare the proposed algorithm with other algorithms. In the case of the sunspots prediction, the number of iterations required for the network initialized with the proposed method was only 3.03% of those started with the next best weight initialization algorithm. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Initial weights determination; Feedforward neural networks; Backpropagation; Linear least squares; Cauchy inequality

1. Introduction

Weight initialization has been widely recognized as one of the most effective approaches in speeding up the training of neural network [2–4,8,9,11–14,16–19]. In this section, we will review the weight initialization algorithms developed by other researchers.

Shepanski regards the training of a multilayer feedforward network as an estimation problem [13]. The optimal set of weights is determined using the least-squares

---

* Corresponding author.
E-mail address: eetchow@cityu.edu.hk (T.W.S. Chow)

criterion employing a standard pseudoinverse matrix technique. Masters also uses least-squares method as a part of his weight initialization algorithm [9]. For a network with one hidden layer, he suggests usage of either simulated annealing method or genetic algorithm in initializing the weights between the input and the hidden layers. After that, the output weights are computed using singular-value decomposition. However, the overall computational complexity of this method is very high. Yam and Chow proposed two weight initialization methods based on a least-squares method [16,17]. In [16], the system is assumed to be linear. The parameters relating the input and the output are obtained by a linear least-squares method. From these parameters, the optimal initial weights between layers are successively determined by factorization. However, this algorithm is not applicable to neural networks in which the number of hidden neurons is smaller than the number of neurons in the preceding layer plus one. In [17], the outputs of hidden neurons are assigned values in the non-saturation region and the optimal initial weights between the input and hidden layers are evaluated by a linear algebraic method. The actual outputs of the hidden neurons are obtained by propagating the input patterns through the networks. The optimal weights between the hidden layer and the output layer are evaluated by a least-squares method. In the problems investigated, the initial errors obtained by this technique were only 4.52–39.1% of those obtained with random weights. However, our further investigations indicated that the networks initialized with [17] occasionally caused the training process to get in a bad local optimum. The probability of getting stuck for some applications may be up to 10%.

Buttou [2] uses an interval $[-a/\sqrt{d_{\text{in}}}, a/\sqrt{d_{\text{in}}}]$, where a is chosen in such a way that the weight variance corresponds to the points of some fraction of the maximal curvature of the activation function, and d_{in} is the fan-in of a neuron, without justifying this interval further in a theoretical manner. Bottou trains the neural network only on speech problems and does not compare this method with others. Nguyen and Widrow speed up the training process by setting the initial weights of the hidden layer so that each hidden node is assigned to approximate a portion of the range of the desired function at the start of training [11]. Through approximating the activation function with piece-wise linear segments, the weights are evaluated. Next, the thresholds of neurons are selected by assuming the input variables to be between -1 and 1. Osowski extended the idea of Nguyen and Widrow by developing another approach to select the initial values of weights [12]. Osowski approximates the activation function by piece-wise linear segments. However, he suggests an explicit method to determine the number of hidden neurons and uses information of the desired function $y = f(x)$ to determine the initial weights. Firstly, the whole region of x is split into sections containing only one negative or positive slope parts of $f(x)$. After that, a set of neurons in the hidden layer is chosen equal in number to that of sections determined. Each section of the curve is associated with one neuron in the hidden layer. The weights and the bias of each neuron are determined and its active (middle) region coincides with the proper section. Finally, the weights to the output layer are evaluated using the change of y in the corresponding section. In the example given in Osowski's paper, the optimum weights obtained after training the network using the BP algorithm are very close to the initial weights suggested by Osowski's algorithm.

Hisashi Shimodaira proposes another method called optimal initial value setting (OIVS) to determine the distribution of initial values of the weights and the length of the weight vector. This method enables the outputs of the neurons in the active region where the derivative of the sigmoid function is large [14]. Shimodaira considers the input and output values of a neuron to be located inside a unit hypercube (for the unipolar sigmoid function). It is assumed that the activation region width of a neuron is larger than the length of the diagonal line of a unit hypercube, and that the center of the activation region coincides with the center of the unit hypercube. Shimodaira is then able to determine the distribution of the weight and the magnitude of a weight vector.

Drago and Ridella propose a method called statistically controlled activation weight initialization (SCAWI) to find the optimal initial weights [4]. The aim of the SCAWI method is to prevent neurons from saturating in an early stage of the training. They determine the maximum magnitude of the weights through statistical analysis. They show that the maximum magnitude of the weights is a function of the paralyzed neuron percentage (PNP), which is in turn related to the convergence speed. By determining the optimal range of PNP through computer simulations, the maximum magnitude of the weights can be obtained. Martens also propose a weight initialization scheme for pattern classifying multi-layer perceptrons (MLP) based on a statistical method [8]. The scheme ensures that all training examples and all nodes have an equal opportunity to contribute to the improvement of the network during the training. The scheme also pursues input scale invariance and can be applied to initialize MLPs comprising both Gaussian and sigmoidal nodes. However, Martens only applied his algorithm to pattern recognition problems and did not compare his results with other well-known methods.

Denoeux and Lengellé suggest another weight initialization method for a single hidden layer feedforward networks [3]. The proposed method relies on the use of reference patterns or prototypes and on a transformation that maps each vector in the original feature space onto a unit-length vector in a space with one additional dimension. Simulation results show that the method is able to reduce the training time, improve robustness against local minima, and give better generalization. Thimm and Fiesler investigate various weight initialization methods by examining their performance on eight real-world benchmark problems in the form of an online backpropagation [15]. Their results generally concluded that the weight initialization method of Wessels performs better than the others.

In this paper, a new approach for evaluating the optimal weights of feedforward neural networks is presented. The rationale behind this approach is to reduce the initial network error while preventing the network from getting stuck with the initial weights. The proposed algorithm ensures that the outputs of the hidden units are in the active region, i.e., where the derivative of the activation function has a large value. From the outputs of the last hidden layer and the given output patterns, the optimal values of the last layer of weights are evaluated by a least-squares method.

2. Weight initialization method

A multilayer neural network with L fully interconnected layers is considered. Layer l consists of $n_l + 1$ neurons ($l = 1, \dots, L - 1$) in which the last neuron is a bias node with a constant output of 1. If there are P patterns for network training, all given inputs can be represented by a matrix A^1 with P rows and $n_1 + 1$ columns. All entries of the last column of the matrix A^1 are constant 1. Similarly, the target can be represented by a matrix T with P rows and n_L columns. The weights between neurons in the layers l and $l + 1$ form a matrix W^l with entries $w^l_{i,j}$ ($i = 1, \dots, n_1 + 1$, $j = 1, \dots, n_{l+1}$). Entry $w^l_{i,j}$ connects neuron i of layer l with neuron j of layer $l + 1$.

The output of all hidden layers and the output layer are obtained by propagating the training patterns through the network. Let us define the matrix

$$O^l = A^l W^l. \tag{1}$$

The entries of A^{l+1} for all hidden layers (i.e., $l = 1, \dots, L - 2$) are evaluated as follows:

$$a^{l+1}_{p,j} = f(o^l_{p,j}), \quad p = 1, \dots, P \quad \text{and} \quad j = 1, 2, \dots, n_{l+1}, \tag{2}$$

$$a^{l+1}_{p,j} = 1.0, \quad p = 1, \dots, P \quad \text{and} \quad j = n_{l+1} + 1, \tag{3}$$

where $f(x)$ is the activation function. The activation function used here is the sigmoid function with range between 0 and 1:

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{4}$$

The entries of output layer A^L are evaluated as follows:

$$a^L_{p,j} = f(o^{L-1}_{p,j}), \quad p = 1, \dots, P \quad \text{and} \quad j = 1, 2, \dots, n_L. \tag{5}$$

Learning is achieved by adjusting the weights such that A^L is as close as possible or equal to T.

In the classical backpropagation algorithm, the weights are changed according to gradient descent direction of an error surface E,

$$E = \sum_{p=1}^{P} E_p \quad \text{and} \quad E_p = \frac{1}{2} \sum_{j=1}^{n_L} (t_{p,j} - a^L_{p,j})^2. \tag{6}$$

The weights are changed according to

$$\Delta w^l_{i,j}(m + 1) = -\frac{\eta}{P} \sum_{p=1}^{P} \frac{\partial E_p(m)}{\partial w^l_{i,j}(m)}, \tag{7}$$

where m is the update epoch in the learning process and η is the learning rate. If the standard sigmoid function with a range between 0 and 1 is used, the rule of changing the weights can be shown to be

$$\Delta w^l_{i,j} = \frac{\eta}{P} \sum_{p=1}^{P} \delta^l_{p,j} a^l_{p,i}. \tag{8}$$

For the output layer, i.e. $l = L-1$

$$\delta_{p,j}^{L-1} = (t_{p,j} - a_{p,j}^{L}) f'(o_{p,j}^{L-1}) = (t_{p,j} - a_{p,j}^{L}) a_{p,j}^{L}(1 - a_{p,j}^{L}). \tag{9}$$

For the other layers, i.e. $l = 1, \ldots, L-2$

$$\delta_{p,j}^{l} = f'(o_{p,j}^{l}) \sum_{k=1}^{n_{l+2}} \delta_{p,k}^{l+1} w_{j,k}^{l+1} = a_{p,j}^{l+1}(1 - a_{p,j}^{l+1}) \sum_{k=1}^{n_{l+2}} \delta_{p,k}^{l+1} w_{j,k}^{l+1}. \tag{10}$$

From Eqs. (9) and (10), we note that the change of a weight depends on the outputs of neurons connected to it. When the outputs of neurons are 0 or 1, the derivative of the activation function evaluated at this value is zero. Therefore, there will be no weight change at all, even if there is a difference between the value of the target and the actual output. Instead of using a statistical method to evaluate the maximum magnitudes of the weights [4], the magnitudes required to ensure that the outputs of hidden units are in the active region and are derived by solving the following problem:

$$1 - \bar{t} \le a_{p,j}^{l+1} \le \bar{t} \quad \text{or} \quad -\bar{s} \le o_{p,j}^{l} \le \bar{s} \tag{11}$$

where $\bar{s} = f^{-1}(\bar{t})$. In this paper, an active region is assumed to be the region in which the derivative of the activation function is greater than 4% of the maximum derivative, i.e.

$$\bar{s} \approx 4.59 \quad \text{for the standard sigmodial function} \tag{12a}$$

and

$$\bar{s} \approx 2.29 \quad \text{for the hyperbolic tangent function.} \tag{12b}$$

Eq. (11) can then be simplified to

$$(o_{p,j}^{l})^2 \le \bar{s}^2 \quad \text{or} \quad \left(\sum_{i=1}^{n_l+1} a_{p,i}^{l} w_{i,j}^{l} \right)^2 \le \bar{s}^2. \tag{13}$$

By Cauchy's inequality,

$$\left(\sum_{i=1}^{n_l+1} a_{p,i}^{l} w_{i,j}^{l} \right)^2 \le \sum_{i=1}^{n_l+1} (a_{p,i}^{l})^2 \sum_{i=1}^{n_l+1} (w_{i,j}^{l})^2. \tag{14}$$

Consequently, Eq. (13) is replaced by

$$\sum_{i=1}^{n_l+1} (a_{p,i}^{l})^2 \sum_{i=1}^{n_l+1} (w_{i,j}^{l})^2 \le \bar{s}^2. \tag{15}$$

If n_l is a large number and if the weights are values between $-\theta_p^l$ to θ_p^l with zero mean independent identical distributions,

$$\sum_{i=1}^{n_l+1} (w_{i,j}^{l})^2 \approx \frac{(n_l+1)}{3} (\theta_p^l)^2. \tag{16}$$

By substituting Eq. (16) into Eq. (15), the following inequality is obtained:

$$\theta_p^l \le \bar{s} \sqrt{\frac{3}{(n_l+1) \sum_{i=1}^{n_l+1} (a_{p,i}^{l})^2}}. \tag{17}$$

If the weight has normal distribution $N(0,(\theta_p^l)^2$ and n_l is a large number,

$$\sum_{i=1}^{n_l+1}(w_{i,j}^l)^2 \approx (n_l+1)(\theta_p^l)^2. \tag{18}$$

In this case, the following inequality is obtained:

$$\theta_p^l \le \bar{s}\sqrt{\frac{1}{(n_l+1)\sum_{i=1}^{n_l+1}(a_{p,i}^l)^2}} \tag{19}$$

In the proposed algorithm, the magnitude of weights θ_p^l for pattern p is chosen to be

$$\theta_p^l \le \bar{s}\begin{cases} \bar{s}\sqrt{\dfrac{3}{(n_l+1)\sum_{i=1}^{n_l+1}(a_{p,i}^l)^2}} & \text{for weights with uniform distribution,} \\ \bar{s}\sqrt{\dfrac{1}{(n_l+1)\sum_{i=1}^{n_l+1}(a_{p,i}^l)^2}} & \text{for weights with normal distribution.} \end{cases} \tag{20}$$

For different input patterns, the values of θ_p^l are different. To make sure the outputs of hidden neurons are in the active region for all patterns, the following value is selected:

$$\theta^l = \min_{p=1,\dots,P}(\theta_p^l)\,. \tag{21}$$

The procedures of the weight initialization algorithm are as follows:

(i) Evaluate θ^1 using the input training patterns by applying Eqs. (20) and (21) with $l=1$.
(ii) The weights $w_{i,j}^1$ are initialized by a random number generator with uniform distribution between $-\theta^1$ to θ^1 or normal distribution $N(0,(\theta_p^l)^2)$.
(iii) Evaluate $a_{p,i}^2$ by feedforwarding the input patterns through the network using $w_{i,j}^1$.
(iv) For $l = 1,2,\dots,L-2$.
   (a) Evaluate θ^l using the outputs of layer l, i.e. $a_{p,i}^l$ and applying Eqs. (20) and (21).
   (b) The weights $w_{i,j}^l$ are initialized by a random number generator with uniform distribution between $-\theta^l$ to θ^l or normal distribution $N(0,(\theta_p^l)^2)$.
   (c) Evaluate $a_{p,i}^{l+1}$ by feedforwarding the outputs of $a_{p,i}^l$ through the network using $w_{i,j}^l$.
(v) After finding $a_{p,i}^{L-1}$ or A^{L-1}, we can find the last layer of weights W^{L-1} by solving the following equation using a least squares method,

$$\text{minimize}\|A^{L-1}W^{L-1}-S\|_2, \tag{22}$$

S is a matrix, which has entries,

$$s_{i,j} = f^{-1}(t_{i,j}), \tag{23}$$

where $t_{i,j}$ are the entries of target matrix T.

The weight initialization process is then completed. The linear least-squares problem shown in Eq. (22) can be solved by QR factorization using Householder reflections [6]. In the case of an overdetermined system, QR factorization produces a solution that is the best approximation in a least-squares sense. In the case of an underdetermined system, QR factorization computes the minimal-norm solution.

Simulation results are presented in the next section. This weight initialization algorithm was validated by real-world problems and the learning performance of the networks initialized with this algorithm is compared to that initialized by other weight initialization algorithms.

3. Results and discussion

In order to demonstrate that the magnitude of each layer of weights evaluated by Eqs. (20) and (21) is able to reduce the number of iterations, 10 networks with all layers of weights initialized by these two equations (i.e. without using the least-squares method to evaluate the weights connecting to the output layer) were applied to solve the nonlinear function approximation problem. In this simulation, initial weights were generated by a random number generator with uniform distribution. The networks with sigmoidal neurons were trained by the batch-mode backpropagation algorithm. Twenty-five simulations with different learning rate and momentum combinations were carried out for each network. The average number of iterations required to reach the (root mean squares error) RMSE $= 0.1$ was recorded and compared to 250 runs in which the networks were initialized with random weights of different magnitudes ranging from 0.1 to 2.0 with a step-size 0.1.

A nonlinear function approximation of eight input quantities x_i into three output quantities y_i, defined by the following equations, was used [1]:

$$\begin{aligned} y_1 &= (x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8)/4, \\ y_2 &= (x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8)/8, \\ y_3 &= (1 - y_1)^{0.5}. \end{aligned} \tag{24}$$

Fifty sets of input signals $x_i \in (0,1)$ were generated by a random number generator, the corresponding y_i were computed using Eq. (24). The network architecture used to learn this function was an 8–12–3 network, i.e., the network had 8 input neurons, one hidden layer with 12 neurons and 3 output neurons.

The average number of iterations required versus the magnitude of random weights is plotted in Fig. 1. The results showed that the number of iterations required was the smallest when the magnitude of the random weights was 1.3. The minimum average number of iterations required was 393.56. With the networks initialized with the Eqs. (20) and (21), the average number of iterations for the 250 runs was 385.24. The weight magnitude emerging from these two equations is able to reduce the number of iterations, although each layer of weights has different magnitude. The magnitude of weights of the first and second layers are approximately 1.10 and 0.95, respectively. In order to investigate the advantages of using the least-squares method to evaluate the

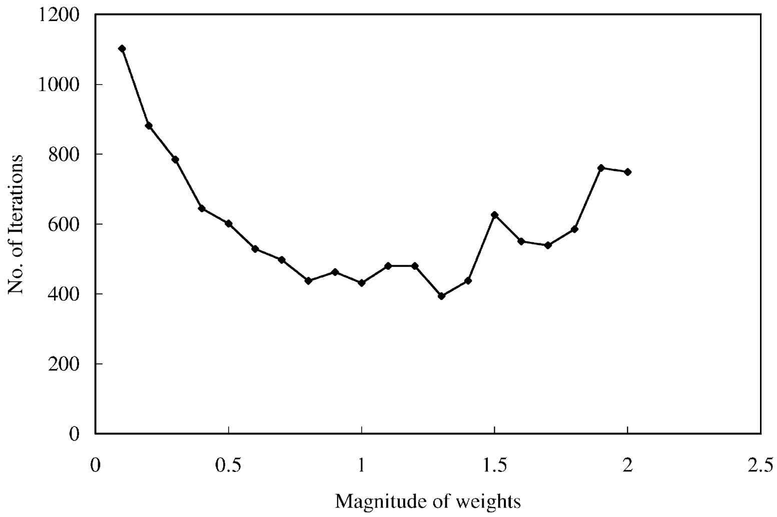

Fig. 1. The average number of iterations for the 250 runs by the networks with different magnitude of initial weights.

output weights in the proposed algorithm, we applied the proposed algorithm to the same nonlinear function approximation problem described above and two real-world problems — prediction of sunspot activities [5] and diagnosis of breast cancer [10]. To make the results more convincing, 50 networks initialized with the proposed method were applied to each problem. Nine simulations with different learning rate and momentum combinations were carried out for each network. For the nonlinear function approximation problem, the termination criterion was set to a more realistic value, RMSE $= 0.02$. The average number of iterations required to reach the termination criterion was recorded and compared with that obtained from 50 networks initialized with other well-known weight initialization algorithms.

The sunspots problem is to predict the activities of sunspots of the coming year from the past data. The sunspots data were divided into three groups. The first group contained the data from 1700 to 1919 and has 208 patterns. The second and third groups contained data from 1920–1955 and 1956–1979, respectively. The first group was used to train the neural network. The network architecture was chosen to be 12–8–1 so that the input data covered the period of one year. The rest were used as the test sets. The termination criterion is 0.1 RMSE.

Breast cancer problem is to diagnose whether the breast tumors are benign or malignant using 30 input attributes. In this investigation, the first 360 patterns were used as the training set. As the range of each attribute in the original data set varies greatly, they are scaled to the interval $[-1, 1]$ in the training set. The output patterns use 0.1 and 0.9 to represent whether the tumors are benign or malignant. The number of hidden neurons used for this problem is 5. The networks are trained to correctly classify 95% of the training patterns.

Table 1
The performance of networks on the nonlinear function approximation problem when the networks are further trained by the batch-mode backpropagation algorithm

Method	Average initial error	Mean no. of iterations to achieve RMSE = 0.02, based on 450 runs	Number of networks getting stuck when further trained
Buttou	0.299085	15784.0	0
Drago	0.291846	24412.0	0
Ngugen	0.428279	30698.6	0
Proposed with uniform weights	0.0257929	929.648	1
Proposed with normal weights	0.0254045	1175.73	4

The results for the nonlinear function approximation problem are shown in Table 1. The average number of iterations required by the networks initialized by the proposed method were only 5.90% (uniform distribution) – 7.45% (normal distribution) of that required by the networks initialized by the weight initialization method proposed by Buttou. By closely examining the simulation results, four out of 50 networks initialized with the proposed algorithm with uniform distributed weights had already reached the termination criterion; therefore, no further training was required for these four networks. However, one network initialized by the proposed algorithm with uniform distributed weights got stuck. Four networks initialized by the proposed algorithm with normal distributed weights were found stuck at local minima. The results showed that the proposed algorithm dramatically reduced the initial error.

The results for the prediction of sunspots activities problem is shown in Table 2. The results again show that the proposed algorithm dramatically reduced the initial error and number of iterations required. The number of iterations required by the networks initialized by the proposed algorithm was only 3.03% (uniform distribution) – 3.19% (normal distribution) of that required by the next best weight initialization algorithm.

The results for the diagnosis of breast cancer are shown in Table 3. The results again showed that the proposed algorithm dramatically reduced the number of iterations required. The number of iterations required by the networks initialized by the proposed algorithm were only 42.5% (uniform distribution) – 45.9% (normal distribution) of those required by the weight initialization algorithm proposed by Ngugen.

The results of all simulations showed that the proposed algorithm dramatically reduced the number of iterations required to achieve the termination criterion when compared with other well-known algorithms. In general, the proposed algorithm with uniform distributed weights outperforms the proposed algorithm with normal distributed weights when the number of networks getting stuck is compared. This phenomenon is attributed to some weights having very large values when the proposed

Table 2
The performance of networks on the prediction of sunspots problem when the networks are further trained by the batch-mode backpropagation algorithm

Method	Average initial error	Mean no. of iterations to achieve RMSE = 0.1, based on 450 runs	Number of networks getting stuck when further trained
Buttou	0.334583	2794.83	0
Drago	0.332903	2826.54	0
Ngugen	0.440809	2984.98	0
Proposed with uniform weights	0.135969	84.7844	0
Proposed with normal weights	0.136923	89.0866	5

Table 3
The performance of networks on the diagnosis of breast cancer when the networks are further trained by the batch-mode backpropagation algorithm

Method	Average initial percentage of correct classification	Mean no. of iterations based on 450 runs	Number of networks getting stuck when further trained
Buttou	14.7	3989.25	0
Drago	3.94	4082.85	0
Ngugen	33.0	3381.02	0
Proposed with uniform weights	60.8	1225.93	0
Proposed with normal weights	62.2	1133.95	5

algorithm with normal distributed weights is used. In general, it is noticed that under the same investigated problem the maximum value of weights generated by a normal distributed random generator is $\sqrt{3}$ times of that generated by a uniform distributed random generator.

The proposed algorithm, in fact, can be viewed as combining a Cauchy's inequality-based initialization algorithm with the least-squares method. As other weight initialization algorithms do not use the least squares method to evaluate the output weights, Table 4 compares the Cauchy's inequality-based initialization algorithm alone with other initialization algorithms to evaluate the performance of the Cauchy's inequality-based initialization method (without using least-squares method). From the table, it is clear that the networks solely initialized by the Cauchy's inequality algorithm still perform significantly better than those initialized by other weight initialization algorithms. As discussed in this paper, the least-squares method is

Table 4
The average number of iterations required by the networks initialized with Cauchy inequality based algorithm (i.e. without using the least-squares method to evaluate the output weights) and other weight initialization algorithms

Problem	Weight initialization method			
	Cauchy inequality based	Bottou	Drago	Ngugen
	Average number of iterations for 450 runs			
Nonlinear function mapping	11380.5	15784.0	24412.0	30698.6
Sunspot	2159.59	2794.83	2826.54	2984.98
Breast Cancer	3353.75	3989.25	4082.85	3381.02

Table 5
The performance of networks on the classification of wines when the networks are further trained by the batch-mode backpropagation algorithm

Method	Average initial percentage of correct classification	Mean no. of iterations based on 450 runs	Number of networks getting stuck when further trained
Drago	2.01124	80.9667	0
Ngugen	0.561798	78.7834	0
Proposed	57.6517	25.1089	0

assumed to work together with the Cauchy's inequality method because no noticeable side effects; such as increasing the rate of getting stuck, was ever experienced.

To demonstrate that the weight initialization algorithm can be applied to networks with other activation functions, networks with hyperbolic tangent activation function are used to classify wines from three different cultivars using 13 constituents [10]. The input data are scaled to the interval $[-1, 1]$. The output patterns use -0.9 and 0.9 to represent whether a wine is of that type. The networks are trained to correctly classify 95% of the training patterns.

In this wines classification problem, the results in Table 5 again showed that the proposed algorithm outperformed other weight initialization methods. Also, it is worth noting that the proposed algorithm can be applied to networks with different activation functions.

To evaluate the compatibility of the proposed algorithm with other fast training algorithms, networks initialized by the proposed method were trained by the Levenberg–Marquardt training algorithm [7]. The networks were again used to solve the nonlinear function approximation and prediction of sunspots activities problems.

Table 6
The number of iterations required for the networks to convergence when the networks were further trained by the Levenberg–Marquardt algorithm

Method	Average no. of iterations, based on 50 runs			
	Nonlinear function approximation		Prediction of sunspots activities	
	Number of iterations	Number of networks networks getting stuck	Number of iterations	Number of getting stuck
Bottou	55.21	5	67.53	3
Drago	51.58	7	48.77	4
Ngugen	276.79	5	161.06	8
Proposed	5.72	0	7.48	0

Fifty different initial weights were generated for each problem. The termination criterion for the nonlinear function approximation problem was 0.01 RMSE, whereas the termination criterion for the prediction of sunspots activities was 0.05. The results are shown in Table 6. Table 6 The simulation results showed that the proposed method reduced the number of iterations required to achieve the error criteria even if a fast training algorithm was used to train the networks. In addition, it is worth noting that the networks never got stuck when they were further trained by the Levenberg–Marquardt algorithm.

4. Conclusions

An algorithm for determining the optimal initial weights of feedforward neural networks based on the Cauchy's inequality and a linear algebraic method was successfully developed. This method ensures that the outputs of neurons are in the active region, and increases the rate of convergence. For all the problems investigated, the proposed algorithm outperformed other well-known weight initialization methods. In the prediction of the sunspots activities problem, the number of iterations required for the network initialized with the proposed method was only 3.03% of those started with the weight initialization algorithm proposed by Buttou. In general, the proposed algorithm with uniform distributed weights performs better than the algorithm with normal distributed weights. The proposed algorithm is also applicable to networks with different activation functions. At last, it is worth noting that the time required for the initialization process is negligible when compared with the training process.

References

[1] F. Biegler-König, F. Bärmann, A learning algorithm for multilayered neural networks based on linear least squares problems, Neural Networks 6 (1993) 127–131.

[2] L.-Y. Bottou, Reconnaissance de la parole par reseaux multi-couches, Proceedings of the International Workshop Neural Networks Application, Neuro-Nimes'88, EC2 and Chambre de Commerce et d'Industrie de Nimes, 1988, pp. 197–217.
[3] T. Denoeux, R. Lengellé, Initializing back propagation networks with prototypes, Neural Networks 6 (1993) 351–363.
[4] G.P. Drago, S. Ridella, Statistically controlled activation weight initialization (SCAWI), IEEE Trans. Neural Networks 3 (1992) 627–631.
[5] T. Fröhlinghaus, A. Weichert, P. Ruján, Hierarchical neural networks for time-series analysis and control, Network 5 (1994) 101–106.
[6] G.H. Golub, C.F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 1989.
[7] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Networks 5 (1994) 989–993.
[8] J.-P. Martens, A stochastically motivated random initialization of pattern classifying MLPs, Neural Process. Lett. 3 (1996) 23–29.
[9] T. Masters, Practical Neural Network Recipes in C + +, Academic Press, Boston, 1993.
[10] P.M. Murphy, D.W. Aha (Librarians), UCI Repository of machine learning databases [Machine-readable data repository], ftp:ics.uci.edu:/pub/machine-learning-databases, 1994.
[11] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, International Joint Conference on Neural Networks, Vol. 3, San Diego, CA (1990) 21–26.
[12] S. Osowski, New approach to selection of initial values of weights in neural function approximation, Electron. Lett. 29 (1993) 313–315.
[13] J.F. Shepanski, Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation, IEEE International Conference on Neural Networks 1, IEEE Press, New York, 1988, pp. 465–472.
[14] H. Shimodaira, A weight value initialization method for improving learning performance of the back propagation algorithm in neural networks, Proceedings of the International Conference on Tools with Artificial Intelligence, New Orleans, LA, 1994, pp. 672–675.
[15] G. Thimm, E. Fiesler, High-order and multilayer perceptron initialisation, IEEE Trans. Neural Networks 8 (1997) 349–359.
[16] Y.F. Yam, T.W.S. Chow, Determining initial weights of feedforward neural networks based on least squares method, Neural Process. Lett. 2 (1995) 13–17.
[17] Y.F. Yam, T.W.S. Chow, A new method in determining the initial weights of feedforward neural networks, Neurocomputing 16 (1997) 23–32.
[18] L.F.A. Wessels, E. Barnard, Avoiding false local minima by proper initialization of connections, IEEE Trans. Neural Networks 3 (1992) 899–905.
[19] N. Weymaere, J.P. Martens, On the initialization and optimization of multilayer perceptrons, IEEE Trans. Neural Networks 5 (1994) 738–751.

Jim Y.F. Yam received a B.Eng. (Hons.) and Ph.D. degree from the City University of Hong Kong. He is now a Research Fellow at the Department of Electronics Engineering of the City University of Hong Kong. His current research areas are artificial neural systems, fuzzy systems and revolutionary computation.

Tommy W.S. Chow received a B.Eng. (Hons.) and Ph.D. degree from the University of Sunderland, UK. He is now an Associate Professor at the Department of Electronics Engineering of the City University of Hong Kong. His current research areas are neural networks, system identification and fault diagnostic analysis.